7-1972

Advanced Computer Program Models : A Talking Textbook Based on Three Languages

John Thomas Gardner
Walden University

ADVANCED COMPUTER PROGRAM MODELS
A TALKING TEXTBOOK
BASED ON THREE LANGUAGES

by

John Thomas Gardner

B.M.E. Kansas State College, 1951

M.S. Kansas State University, 1957

Robert W. Swanson, Ph.D., Advisor
Director of Computer Center
California State University
San Diego

A Project Submitted in Partial Fulfillment of
The Requirements for the Degree of
Doctor of Philosophy

Walden University
July, 1972

AN ABSTRACT OF A DISSERTATION

By

John Thomas Gardner

B.M.E., Kansas State College, 1951

M.S., Kansas State University, 1957

A Thesis Submitted in Partial Fulfillment of

The Requirements for the Degree of

Doctor of Philosophy

Walden University

July, 1972

The purpose of this dissertation was to develop a learning instrument, to be used by programmers preparing for the Data Processing Management Association Test as a self study book, or by college business programming and computer science students who have completed a course in data processing and a course in programming a higher level language.

The mathematical ability requirement was minimized by developing the algorithms in parallel with the programs.

The learner should experience emphasis in the following areas:

1. The type of activities required to pass the DPMA test (the programming part)
2. Data Structures
3. Fortran (at the level of the DPMA test)
4. RPG (at the level of the DPMA test)
5. Flow chart reading and writing

Fortran and RPG (Report Program Generator) languages were used, since their proficiency is required for the DPMA test; however a subset of IBM Basic Assembler language was used, because the author believed that a person who is more than superficially interested in computers should demonstrate a proficiency with a machine language.

An important part of this method of presentation are the cassette recordings which allow the learner to progress outside the classroom. The recordings plus the hard copy of the actual programs, diminished in size, give the learner material which he can move to any location and study without the presence of the instructor.

Organization

The book is divided into four main parts:

1. Introduction (containing the Swanson Systems Study)
2. Data Structures
3. Fortran
4. RPG

The Introduction contains the Swanson Study which provides a model to answer the question "WHY" for much of the rest of the book.

Section II, Data Structures, includes many of the topics included in the recommendations of the Curriculum Committee on Computer Science of the Association of Computing Machinery.

Section III, Fortran, is designed to prepare the advanced programmer for the DPMA test, but can be used to develop programs in other languages by using the fortran programs presented here as models. This section does contain some BAL programs.

Section IV, RPG, provides models which will aid in the preparation for the DPMA test. The BAL models used here were selected to match the same topic, Invoicing, as the RPG program.

Experimental Results

The experimental objective was to determine to what extent, if any, Advanced Computer Program Models affected the attitudes towards and proficiencies in programming of college level students in San Diego County.

The original design of the experiment was a completely crossed three by four factorial experiment: three level (novice, beginner and expert) versus four types of computer instructions: (1) standard classroom, (2) no classroom, only the talking textbook material, (3) classroom and material, (4) open study with no guidance from instructor or experimental material.

The observations are individual students' different scores with respect to a standard examination given in January at the start of the school semester, and the same examination given in May at the end of the semester. Classes were held for one night a week for a total of eighteen meetings. One section consisted of students learning computer programming. The other sections consisted, with one exception, of programmers who had some experience with programming. In this section the objective was to develop techniques for data structures.

There were several factors which prevented us from fulfilling the requirements of the originally completed cross factorial experiment:

1. We were dependent on the availability of computer equipment.
2. We could not control the composition of the programming classes so that one class was a mixed grade level and the other class was not mixed.
3. We were dependent upon volunteers and had no control over the number of programming sections assigned to these volunteers.

To measure the affect on proficiency, we used the CTSS test data bank maintained in the California State University, San Diego. To measure the affect on attitude, we used the Aiken Devised Programming Attitude Scale, which is a Likert-type scale, developed by Louis Aiken, professor of psychology, at Guilford College. They were administered once within a week period between January 15th and January 21st, and the second time between May 15th to May 22, 1972. The tests were administered by the instructors to their own classes. The grading and scoring was performed by the staff at San Diego University.

A statistical analysis was performed with the assistance of the computer facilities of San Diego State University. Biomed Statistical Program, BMD-V, was used in one analysis. The remaining analyses and data summaries were made using the statistical program library (S.T.L.) available at the University.

Conclusions

The conclusions derived from the testing and experiments were: Novices scored significantly higher with classroom instruction with little difference in text used. The absence of significance in the effects of the classroom experience and text on student proficiency and attitude in the other groups was contrary to expectations. The curriculum in question had been developed for a different approach for the subject. At the end of the experiment, the instructors were of the opinion that for a more effective utilization of the material, it would be necessary to adapt the text to the curriculum.

Whether or not a curriculum especially adapted to this experiment would show any changes in student proficiency or attitude is another question for objective evaluation.

In any case, it is clear that the function of this text, as it was coordinated for this experiment, did not provide any reasonable improvement in the overall student proficiency or attitude beyond the variations associated with students, teachers and experience level.

PREFACE

This book is a learning instrument, to be used by programmers preparing for the Data Processing Management Association Test as a self study book, or by college business programming and computer science students who have completed a course in data processing and a course in programming a higher level language.

The mathematical ability requirement of the student has been minimized, since the algorithms are developed in parallel with the programs.

The learner will experience emphasis in the following areas:

1. The type of activities required to pass the DPMA test, (the programming part)
2. Data Structures
3. RPG (at the level of the DPMA test)
4. Fortran (at the level of the DPMA test)
5. Flow chart reading and writing

Fortran and RPG (Report Program Generator) languages were used since their proficiency is required for the DPMA test; however a subset of IBM BAL language was used because I believe that the person who is more than superficially interested in computers should be able to demonstrate a proficiency with a machine language.

The cassette recordings are an integral part of the presentation, since they allow the reader to progress outside the classroom. The recordings, plus the hard copy of actual programs, diminished in size, give the learner material which he can move to any location and study without the presence of the instructor.

ORGANIZATION

The book is divided into four main parts:

1. Introduction (containing the Swanson System Study)
2. Data Structures
3. Fortran
4. RPG

Section I, the Introduction, contains the Swanson System Study, which provides a model to answer the question "WHY" for much of the rest of the book.

Section II, Data Structures, includes many of the topics included in the recommendations of the Curriculum Committee on Computer Science of the Association of Computing Machinery.

Section III, Fortran, is designed to prepare the advanced programmer for the DPMA test, but can be used to develop programs in other languages by using the Fortran programs presented here as models. This section does contain some Basic Assembler programs.

Section IV, RPG, provides models and information which will aid in the preparation for the DPMA test. The Basic Assembler program presented here is used as a comparison for the invoicing program, written in RPG.

HOW TO USE THIS BOOK

The programmer reviewing for the DPMA test can study the Swanson Study, use the cassettes, review the Fortran Section (paying particular attention to the first half), and review the RPG Section. This can be done in six weeks without an instructor.

The college student can make a semester, or a year study, with or without an instructor. The type of computer is incidental, since the tape cassettes analyze each BAL program. In fact a computer isn't needed (however it does work as a motivating force). This allows several different languages to be programmed at the same time during a semester.

An instructor who uses this book need not know the Basic Assembler Language, because the tapes will assist him.

The long programs were diminished in size intentionally so that a learner could manipulate them with ease. It is sometimes better for analysis, if the learner copies certain sections manually, if he wants to carry on a concentrated study.

The book starts with a real systems study, which uses a fictitious name. The study was made by the firm of Swanson Associates, (real name), and the corporation in the study is the Marine Division of the Oceano-Graphic Corporation (fictitious name).

Why use a systems study to start a programming text? The answer to this question is that the study gives meaning to the whole text, and serves as a vehicle for the text. The programmer reviewing for the DPMA test might have lost view of the whole corporation, he might have forgotten some of his systems work he had a few years ago, and might not have a model of a systems study to review before his test.

The college student might have a good background in accounting, but it will add to his ability if he can work with a model that develops a management information system, which includes the whole or at least a part of the corporation.

ACKNOWLEDGEMENTS

Information and illustrative materials have been used from publications copyrighted by the International Business Machines Corporation.

Direct quotations appear as itemized below:

1. C-26-3600 RPG Programs Invoicing and Pre-Billing Calculation with Inventory Control
2. C-33-6000 RPG Disk Examples

The Systems Study was adapted from a real systems study by Dr. R. L. Swanson and Associates.

Help from many sources has contributed to this book. To my students and teachers, named and unnamed, I extend my deepest gratitude. I also express my thanks to Mr. Wayne Langley, Mr. Richard Schenck, Mr. Lawrence Huster, Mr. Herman Dailey, Mr. Roy Towne and Mrs. Ethel Castle, who taught, shared knowledge and provided aid in working various programs.

Special thanks go to Dr. Robert Swanson, who patiently guided me through the dissertation and provided the systems study upon which the book is based.

Most of all, I wish to express my deepest appreciation to my wife, Patty, and seven children, Clorinda, Patrice, Deborah, Curtis, Melisa, Michele and Melanie, who were, I am sure, very pleased when "the computer book" was finished.

TABLE OF CONTENTS

Table of Contents (con't.)

Page

SECTION I

INTRODUCTION

THE SWANSON SYSTEMS STUDY

The objective of this section is to present a complete model of a systems study, and the presentation of the files serve as models to be programmed at the end of the book.

Since so many elementary programming books start with explanations of computers and use only short examples, it was felt that an advanced book should contain a system study and complete models of programs when possible.

It is suggested that the reader return again and again to the charts of this study so he can develop a feeling for the functional groupings of this real corporation.

PRODUCTION SYSTEM PROPOSAL
for the
MARINE DIVISION
of
OCEANO-GRAPHIC CORPORATION

by

Dr. Daryl G. Mitton

Dr. Robert W. Swanson

1.0 ABSTRACT

This is a proposal of a system design for the Marine division of the Oceano-Graphic Corporation. We believe its unique features will provide a multitude of advantages for the firm including:

a. A planning and control capability which cannot even be approached by your current system and which, we feel, has not been attained for any similar sized company in the past.

b. A capability of response through information accessibility for all tactical and strategic situations.

c. A reduction in both direct and indirect production costs--at present or increased production levels.

d. A competitive advantage in the ability to obtain proper proposal information from a technical, cost, and delivery standpoint.

e. A reduction in inventory and inventory costs (we feel that under existing output levels, the present inventory level of $85,000 can be cut to $20,000).

f. The availability of analyses of consequences of alternative decisions.

g. The many sundry advantages brought about by good systems design, such as more complete and improved cost information, the signaling of impending operational difficulties, the reduction of operation cycle time, the improvement of customer service and feedback, the determination of future equipment and manpower needs, and the preparation of most essential paperwork.

The strength of the system is the ability to obtain needed information easily. This strength provides a strong motivation to keep the system operative. No operational system, regardless of its detailed design, will function properly unless it is used as its design intended it to be used. Most systems are designed in such a way that their intended function is obstructed in some way to satisfy organizationally local needs. Therefore, most systems fall short of their intended goals.

At present, the Marine Division of the Oceano-Graphic Corporation has a very detailed and elaborate system for operational planning and control. It does not function. Its literal atrophy is caused by an informal realization at all levels within the division that the struggle to supply input and the difficulty of receiving output are not worth the rewards obtainable by the functioning system.

This is no direct criticism of the Marine Division system, for what is symptomatic here is found in most organizations: the superimposition of an informal "make-do" system which operates on "get by" and exists to accommodate certain paper work demands that may or may not have any real relevance to the effective running of an organization. In fact, it is not uncommon for the resulting working system to be actively dysfunctional, particularly in the long run.

Our study first confirmed the existence of a non-working system. We next submitted preliminary designs of two possible alternative systems which would provide a greater degree of automaticity in speeding input and spilling output. The first was a somewhat conventional approach, an integrated data processing system, which would mechanize the various subsystems (a subsystem at a time) that impacted on the production function--production, purchasing, accounting, engineering, etc. Each system, being separate to orient its particular output demands, would, nonetheless, be updated by information fed into any of the subsystems. Output would have a specialist orientation--not suited to generalship. The data files would be redundant and the system would never be totally current. We admit to editorializing against this system in favor of a second approach, which the Marine Division decided to pursue.

The system devised is built around the concept of a single flow of information. Essential to this system is a common data base. Information input is, therefore, immediately triggered to update purchasing, production, engineering, etc. Information is always current. No output is made unless called for. Information release is on a "need to know" basis rather than a "nice to know" basis, and it is in an interpretable form to serve as a decision base. The ability to obtain needed information very easily serves as the strongest motivation to supply essential input. In spite of the universality of the system, we have devised a way to introduce it in modules, which will enable faster realization from the system as it is programmed and introduced, and which provides for current debugging of the programming.

We recommend an IBM 000/000 computer as the equipment best suited for immediate and future ne-ds of the Marine Division. We estimate its first-year costs at $00,000 ($0,000 per month for ten months) to program and debug the system recommended. Current Marine Division personnel would be able to man the program, so no additional cost would be incurred here. It is unfortunate, in a report such as this, that we cannot include as a cost the price of not making the change, for we feel, if the Marine Division is to be a part of the growth of oceanographic explosion that is sure to come, these changes must be made. What is the cost of losing a position of leadership in oceanographic instrumentation? What is the cost of losing a contract because of poor performance? What is the cost of slipping ever so slightly behind the competitor? There are costs that do not show in the profit and loss statement directly, but their costs are just as real as those that are displayed.

1.1 PLAN OF REPORT

The report is presented in five parts, starting with a discussion of the objectives and limitations. This is followed by an explanation of the methods used and alternatives available to us in conducting the study and design. The next section of the report is devoted to a summary analysis of the existing system and what we considered to be the major limitations for continued successful use of the system as it currently exists. Beginning in Section 5.0, we present the proposed single information flow system, complete with the details of the proposed filing system. In Section 5.5, we further specify the benefits that can accrue from the employment of such an approach to the production information system. Section 6.0 of this report deals with our estimated cost and schedule of implementation.

2.0 OBJECTIVES

The objective of this study is to analyze existing procedures and conditions of operations prevalent in the Marine Division of the Oceano-Graphic Corporation to determine the feasibility of use and choice between available methods of operations of automatic data processing systems.

A part of this objective has been to develop an operating system that would provide the management of this division with insight and control of the production function, insight being defined as access to the facts in an orderly manner with an insured degree of accuracy.

A second function of the objective was to develop the state of automation to provide individual response to the system.

The functional criteria for measuring how well these objectives can be obtained are:

a. Reduction of the number of manual checking functions by one-half (i.e., routine decisions and paperwork).

b. The increase of the division's profit to thirty percent prior to taxes.

c. Reduction of the total direct and indirect cost of production by twenty percent.

d. Reduction of the response time to a customer's order to thirty days.

2.1 SCOPE

There are many limitations to a report of this nature. Time only permits the minimum of attention to the existing disciplines and methods of management information systems. The chief center of interest here is to determine the prospective utilization or applicability of present systems. Further, since many management decision factors are prefaced upon future events not totally definable now, no effort has been made to define the future needs of Oceano-Graphic other than what is needed now. These decisions must be reserved for the future as the technology evolves; however, the provisions for growth were considered and incorporated as a function of the study. When we consider the peculiarly sensitive nature of a company's investment in computers--with its potential for major impact on its position in competition--judgments on how, where and when--the investment should be made to assume special importance. As a result of advances in technology, the computer can now play a more central role in corporate planning and operations. Since present criteria for allocating resources to ADP are based on outdated data processing operations or on rules of thumb, they do not take into account the ability of computers to contribute to profits and/or cut operating costs outside of ADP. Clearly, it would be more appropriate to evaluate the computer investment in terms of its contribution to the entire management process.

We recognize how difficult a task this will be. There can be no simple rules of thumb for analyzing Oceano-Graphic's ADP expenditures. These expenditures must be geared not to the company's size or competitor's spending, but to the benefits to be derived in each specific case. Since these benefits often accrue from the value of the information to be supplied by the system, as well as from the added efficiencies and direct savings in data processing operations, they are elusive and extremely difficult to measure.

It should further be expressed as a function of the scope of this study that we, as consultants, are not in the business of selling computer hardware. Our task is to evaluate the present system and recommend changes as they relate to the objectives of the study. If the results of this study are implemented, the choice of equipment would be the decision of the Oceano-Graphic Corporation.

3.0 BACKGROUND

Our approach to this particular study was to review the present system as it is currently operating. This review included those documents and procedures that are defined in the systems and procedure manuals and a complete survey of the actual operations and procedures as interpreted by those performing the functions. There always exists a certain dichotomy in all enterprises. There are the procedures set by the workers and lower levels of management and the procedures formally defined by top management. Rarely are the interpretations identical. Suborganizations with organizations adopt symptoms to satisfy the needs of their organizational area. They add to and ignore the system as they see it and make it tolerable and workable from their point of view. Therefore, there exists a system working the way the lower echelons believe it has to work. It is necessary to review both systems.

Each of the activities was reviewed from the perspective of how well it contributed to the production of the end product and how well it fit into the function of management control. Instead of starting with the costs of processing data, we started with the positive value of the information produced by the processing. The benefits do not stem directly from the fact that given data are processed; they stem from the results of data processing. The value of information from data processing is not what it costs to obtain but, rather, what it can do for management. Costs have their place, but in a different part of the equation.

This, the first effort becomes one of analysis of the intended system. It is an effort to establish a system based upon the total objectives of the company. Recognizing the profitability of rapid growth in the oceanographic field and the limitations of time and cost in designing and implementing a total system, we feel there are two feasible alternatives to choose from--integrated data processing or single information flow.

The integrated data processing approach is an attractive alternative from several aspects. First of all, it is the conventional approach. Most commitments in the field of data processing take this approach. It is a step-by-step approach. You can merchandise a subsystem at a time, thereby getting a system on the computer, gaining some utilization and economic value out of the system without the total system being designed. In this case, production could be put into operation first, followed by a purchasing system, then accounting, etc. All of the subsystems would be designed so that information from one system would be used to update other subsystems, thus integrating all the systems into a single integrated whole. Each of the subsystems would produce the desired reports. In the technical jargon, we refer to it as an "output oriented system."

The single information flow concept developed from some of the apparent limitations of the integrated data processing (IDP) approach. First, consider the IDP method each subsystem being linked to the other subsystems. This means each subsystem must be updated without respect to the volume of transactions affecting it. Secondly, each subsystem must, by its nature of independence, support separate filing systems, much of which is redundant data. This requirement for processing and redundant files necessitates considerably more machine capacity than would be required if they could be eliminated. A second deficiency arises in that the system is never totally current by the nature of the interlacing of subsystems. Thus, management in each of the separate functions always sees different data creating a continuous point of contention.

This system is designed whereby the source information in any department updates a single information file. For example, an engineering change would update the purchasing files and production files, as well as the engineering files at the time of entry. Conceptually, they would be a single file. In this mode, we could eliminate a considerable amount of processing. In the technical sense, the system would become input oriented. The results would then be a system containing the same information bank, but requiring less equipment and more current data.

When presented with these alternatives, the management of the Marine Division chose to pursue development in the direction of the single information flow concept. The design of the system was then undertaken with the intent of developing this concept on a limited scope, primarily directed to production, inventory control, material control, and purchasing functions.

3.1 PROCEDURAL ANALYSIS

The benefits accruing from any management information system, whether computerized or not, may be seen to fall into three categories:

a. Cost savings--i.e., savings in data processing cost because of reductions in clerical work force and other changes.

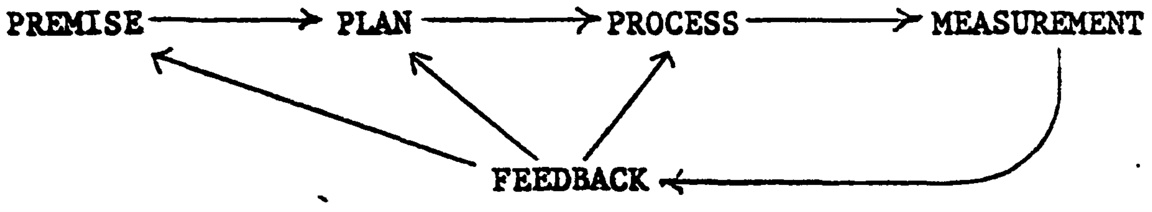
b. Operational gains--i.e., efficiencies in corporate operations resulting from the application by managers of information received through the system (for instance, data on inventory reductions and faster production).

c. Intangible benefits--i.e., improvement in customer service, corporate planning and forecasting, the ability to sustain growth, and other advantages which may not be present without the system but which depend upon management's astuteness in using it.

Thus, the functions for measuring the value of the present system were categorized into these three areas in an attempt to establish a baseline.

It is also necessary at this point to differentiate between the business system and the information system of the enterprise. For the most part, the business system has its main function of converting the raw materials, through a process of manufacturing, testing, and assembling, into oceano-graphic measuring instruments. The management information system, on the other hand, translates from the environment and from within its own components as its inputs. It stores this information and associates it with previously stored information in order to provide a frame of reference for the next vital step, that of making a decision.

Defined in this manner, the analysis of the present system and design of the proposed system were evaluated against the following premise. Management control is a function of measuring how well a task is being accomplished as compared to what was planned. This is pictorially described as:



4.0 THE EXISTING SYSTEM

As previously stated in the procedure of analysis section of this report, the existing system was evaluated primarily from the following points of view:

a. What are the objectives or premise for operations?

b. How was the premise put into operation, i.e., how did the planning take place?

c. In what manner was the plan implemented?

d. What was the measurement for determining how well the plan was being accomplished?

e. What was the medium for feedback and how did it affect the process?

The first step was to define the processing system as it currently exists. EXHIBIT "E1" describes the system by means of a PERT type network as it was described by the employees within the system.

As we appraise the system from an overall standpoint, we find

Detail of Production

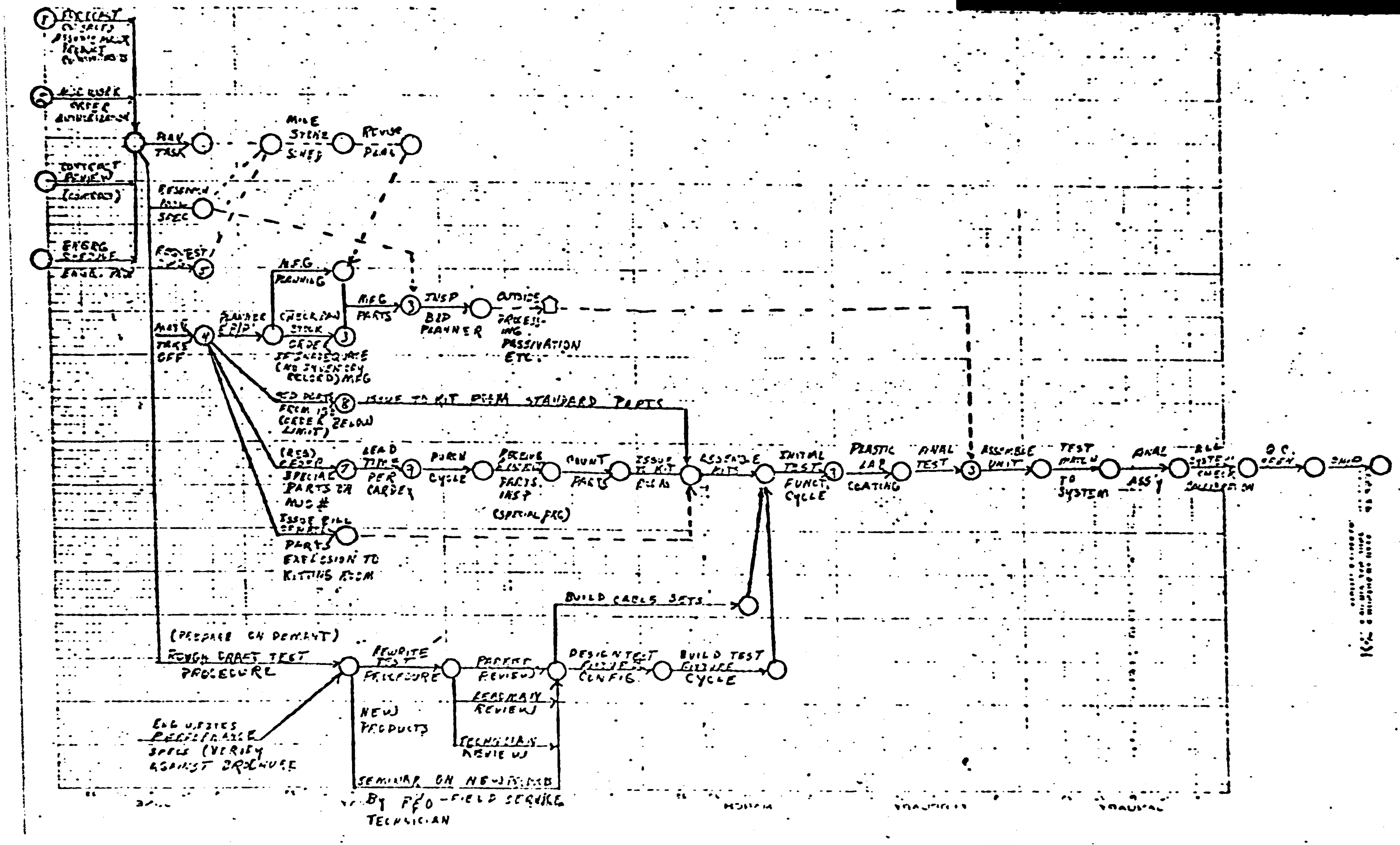


Figure SWAN 1

the information is usually defined in two documents, the MJO and contract. This information is then combined with the forecast of sales to plan the task. The planning task for inventory MJO's is completed and issued in the form of a milestone schedule, which is filed and never distributed. As a matter of interest, during the period of this investigation, much dissatisfaction was expressed by those making this schedule and they had discontinued its process and were in the process of designing a new one. No where in the organization was any other reference made of the milestone schedule nor did anyone seem to miss it. This was somewhat overcome by the authorization for work document. This document does give the information as to what is to be done and some limiting criteria, such as budget and when due.

In pursuing the authorization for work as the planning and control mechanism, we found its primary use was to provide the major task definition and a charge number by which to accumulate costs. It contains no provision for stating how the job is to be done and if there is a change in the plan. For instance, there is no revising of this document to reflect the change. Hence, the document doesn't really provide the planning function nor the feedback medium for control.

To further pursue the process of planning, we found the foreman in each of the departments was familiar with what had to be accomplished, based upon past experience, and assigned the work accordingly. This work assignment was related to their area only and reflected none of the interface requirements of the other subsystems. Thus, the product test plans were in no way coordinated with the machine shop. Again, there was a complete lack of any formal feedback mechanism. It would appear to an outsider that the system primarily reacted to crises and was not

Procedure Flow Chart

R.W.SWANSON, associates

PROCEDURE FLOW CHART

Company OCEANO- GRAPHIC MARINE DIVISION

Application

Prepared by

Reviewed

Opr.

Description

The manager of manufacturing Receives The MUO from The financial administrator. He also secures a copy of the contract to determine what must be produced; He also refers to the forecast of sales.

with this information he personally develops a milestone chart. AT the same time there is some research of Mil specs by Q.C. and a request for the drawings. If there elements have any affect the plan is then revised and then filed.

The MUO also triggers a materials take off in the materials Control Station. After this operation a planner and blueprint are issued to machine shop. Machine shops determine material needs, orders same; plans job and does the processing. No Reference to schedule prepared earlier. Planner did no planning in sense of normal function of that piece of paper.

STANDARD PARTS were issued to kit Room along with a Bill of material maintained on file. The Precise ORIGIN of This Bill of material could not be made traceable to the Engineering change but was assumed to be correct.

Non standard parts were then requisitioned to purchasing and went through the purchasing cycle.

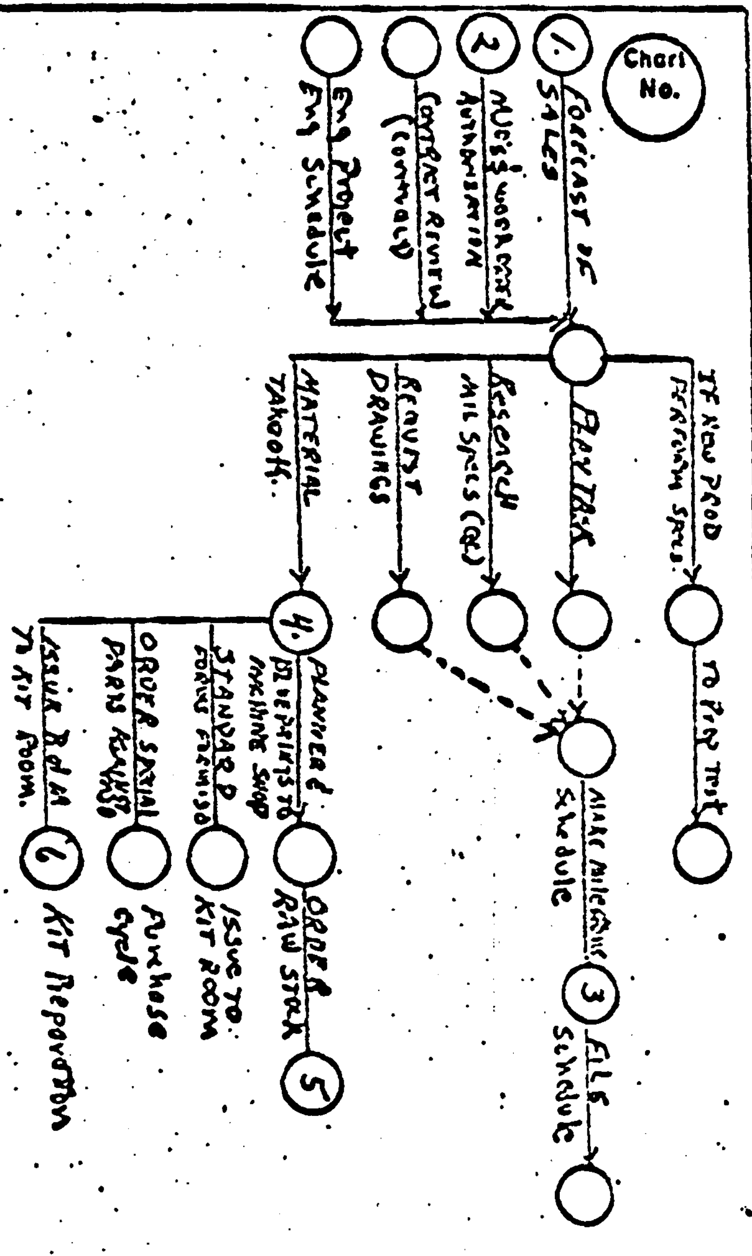


Figure SWAN 2

R.W. SWANSON, associates

SOURCE DOCUMENTS

Customer OCEAND - GEAPHIC MARINE DIVISION Date 00/00/00

(Where in the prospects office are the documents you need origin d / where are they used?)

				Department or Name of Originator												Distribution - Name and Title of Receiving Person or Department											
Process Frequency	Description	Legibility	Volume	1 SALES	2 CONTACTS	3 ACCTING	4 ENG.	5 R&D	6 MFG	7 PROD ENGR	8	9	10	11	12	13 BUDGETS	14 PROJ. E	15 VICE PRES	16 PRES	17 CONTACTS	18 EST. CYCLE	19 ACCTNG	20	21	22	23	24
				Number of copies												Which copy/How many											
D	REQUEST FOR	/																									
	MASTER JOB ORDER	A	9/w	1	1	1	1	1	1	1	1					0	0	0	0	0	0	0					

Notes

1 COPY, THE ORIGINAL, LATER REPRODUCED TO 16 COPIES AFTER APPROVAL; VOLUME = 450/yr. 75/OPEN/ONE TIME.

Processing Frequency

D – Daily	M – Monthly
W – Weekly	Q – Quarterly
B - Bi-Weekly	SA - Semi-Annually
S – Semi-Monthly	A – Annual

Figure SWAN 3

R. W. SWANSON, associates

REPORT DISTRIBUTION and USE

Customer OCEANO-GRAPHIC MARINE DIVISION .Date 00/00/00

Use Value Comment Sheet

Report Distribution (Dept. Function, Person, Etc.)

Process Frequency	Report Title	Ltr				1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	
						BUDGETS	PURCHASING	FILE	SHIP	MKT.	DAD	ACCT	ENG	Q.C.	MFG CONTR	DOC CONTR	R&D	MFG	INV. CONTROL	PROD TEST	ELECT ASSY	MACH SHOP	OTHER	CONTRACTS	Optional Reroute	Method of Delivery	Burst	Bind	Decollate	Total No. of Copies
D	WORK ORDER	a																								H	L		✓	
	AUTHORIZATION	b				✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓						16 MIN
		c																												20 MIN
		d																												
D	MJO	e				✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			H	L		✓	16-20
		f																												
		g																												
		h																												

Notes

a. WORK ORDERS CUT ON AN AS NEED BASIS. APPROXIMATELY 450 PER YEAR WITH AN AVERAGE OF 75 BEING OPEN AT ONE TIME. COST/VALUE STUDY NOT MADE AT PRESENT TIME. NOT CONSIDERED AS AN ACCOUNTABLE COST OR A DIRECT COST.

b. MJO TRIGGERS WORK ORDER AUTHORIZATION - ONE IS REDUNDANT.

KEY Burst-Bind
Reg Left - L
Top - T
Right - R
Decollate - ✓

Processing Frequency

D - Daily	M - Monthly
W - Weekly	Q - Quarterly
B - Bi-Weekly	SA - Semi-Annually
S - Semi-Monthly	A - Annual

Figure SWAN 4

R. W. SWANSON, associates

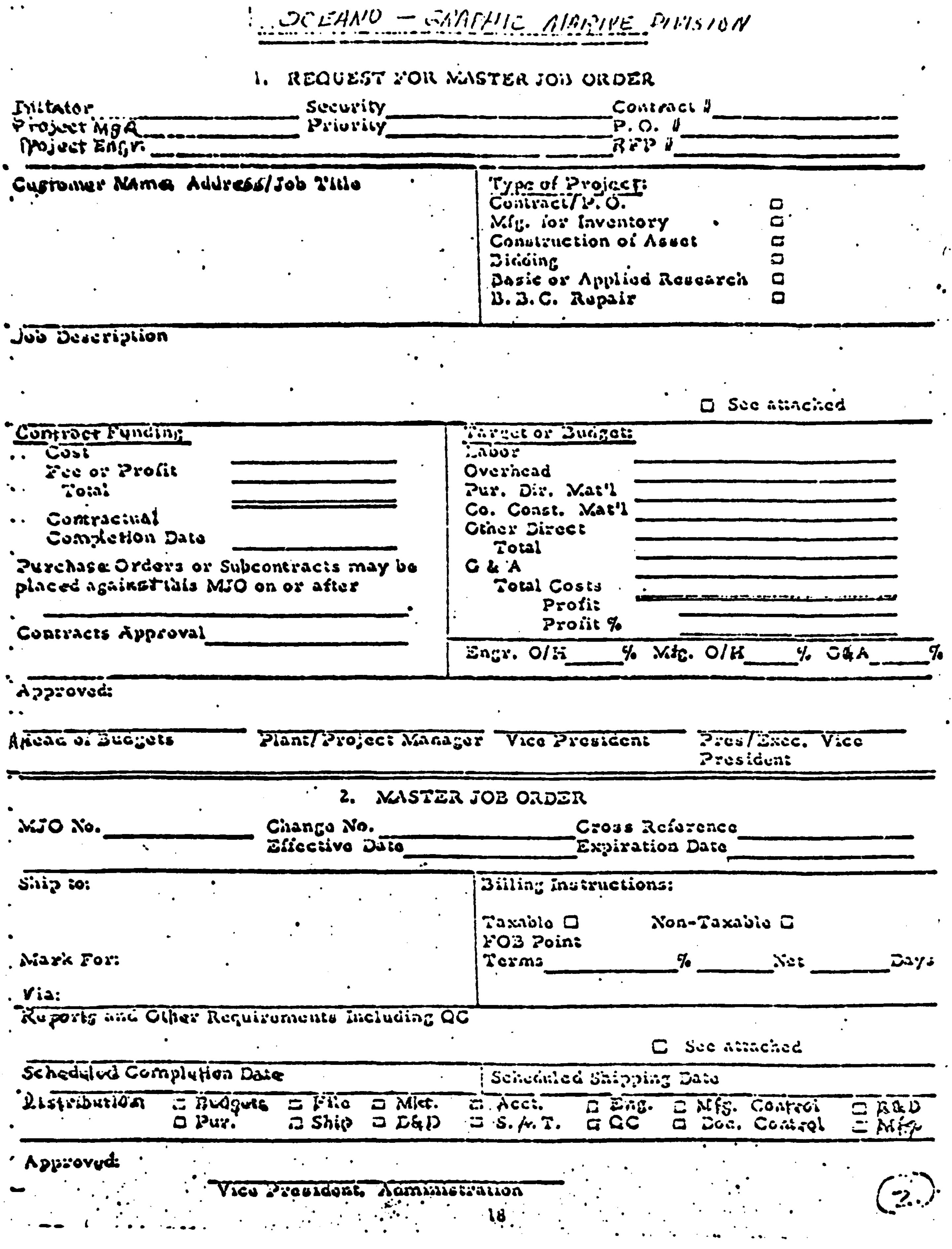
VALUE COMMENT*

Ref Visl No	Rpt Usd Ltr	Report Title	Value Narrative
F-2	1	FORECAST OF SALES	THE forecast is of value TO some echelons of management, but is of little value to Production-planning. No commitment to production can be made on the basis of a forecast. NOR is there any correlation factor between the issue of the WORK AUTHORIZATION AND THE SALES FORECAST. THERE ARE NO STATISTICAL CORRELATIONS computed to establish any measure of uniformity. In my discussions with respect to production planning under present system, Report has only AN immediate Value. If Any.
F2	2	MJO & WORK ORDER AUTHORIZATION	BASICALLY three documents. THE MJO IS THE ONLY documented evidence of planning that appears to tie all operations together. Document contains Customer identification, funding, Budget data and approvals. The authorizations PART DOES identify the major functions to be accomplished. However it is department oriented and not process ORIENTED. Budget is not related to time nor work station. May or may not reflect estimate.

*Consider at least.(1) Earning on freed capital.(2)Yield on inventory reductions. (3) Benefits.(4)Equipment. (5) Displaceable costs. (6)Profit on increased sales.

Figure SWAN 5

Request For Master Job Order



OCEANO – GRAPHIC MARINE DIVISION

1. REQUEST FOR MASTER JOB ORDER

Initator ____________ Security ____________ Contract # ____________
Project MgR ____________ Priority ____________ P. O. # ____________
Project Engr. ____________ RFP # ____________

Customer Name Address/Job Title

Type of Project:
Contract/P. O. □
Mfg. for Inventory □
Construction of Asset □
Bidding □
Basic or Applied Research □
B. B. C. Repair □

Job Description

□ See attached

Contract Funding
Cost
Fee or Profit ____________
Total ____________
Contractual Completion Date ____________

Purchase Orders or Subcontracts may be placed against this MJO on or after ____________.

Contracts Approval ____________

Target or Budget:
Labor ____________
Overhead ____________
Pur. Dir. Mat'l ____________
Co. Const. Mat'l ____________
Other Direct ____________
Total ____________
G & A ____________
Total Costs ____________
Profit ____________
Profit % ____________

Engr. O/H ____ % Mfg. O/H ____ % G&A ____ %

Approved:

____________ Head of Budgets
____________ Plant/Project Manager
____________ Vice President
____________ Pres/Exec. Vice President

2. MASTER JOB ORDER

MJO No. ____________ Change No. ____________ Cross Reference ____________
Effective Date ____________ Expiration Date ____________

Ship to:

Mark For:

Via:

Billing Instructions:
Taxable □ Non-Taxable □
FOB Point
Terms ____________ % ________ Net ________ Days

Reports and Other Requirements Including QC

□ See attached

Scheduled Completion Date

Scheduled Shipping Date

Distribution □ Budgets □ File □ Mkt. □ Acct. □ Eng. □ Mfg. Control □ R&D
□ Pur. □ Ship □ D&D □ S. A. T. □ QC □ Doc. Control □ Mfg.

Approved:

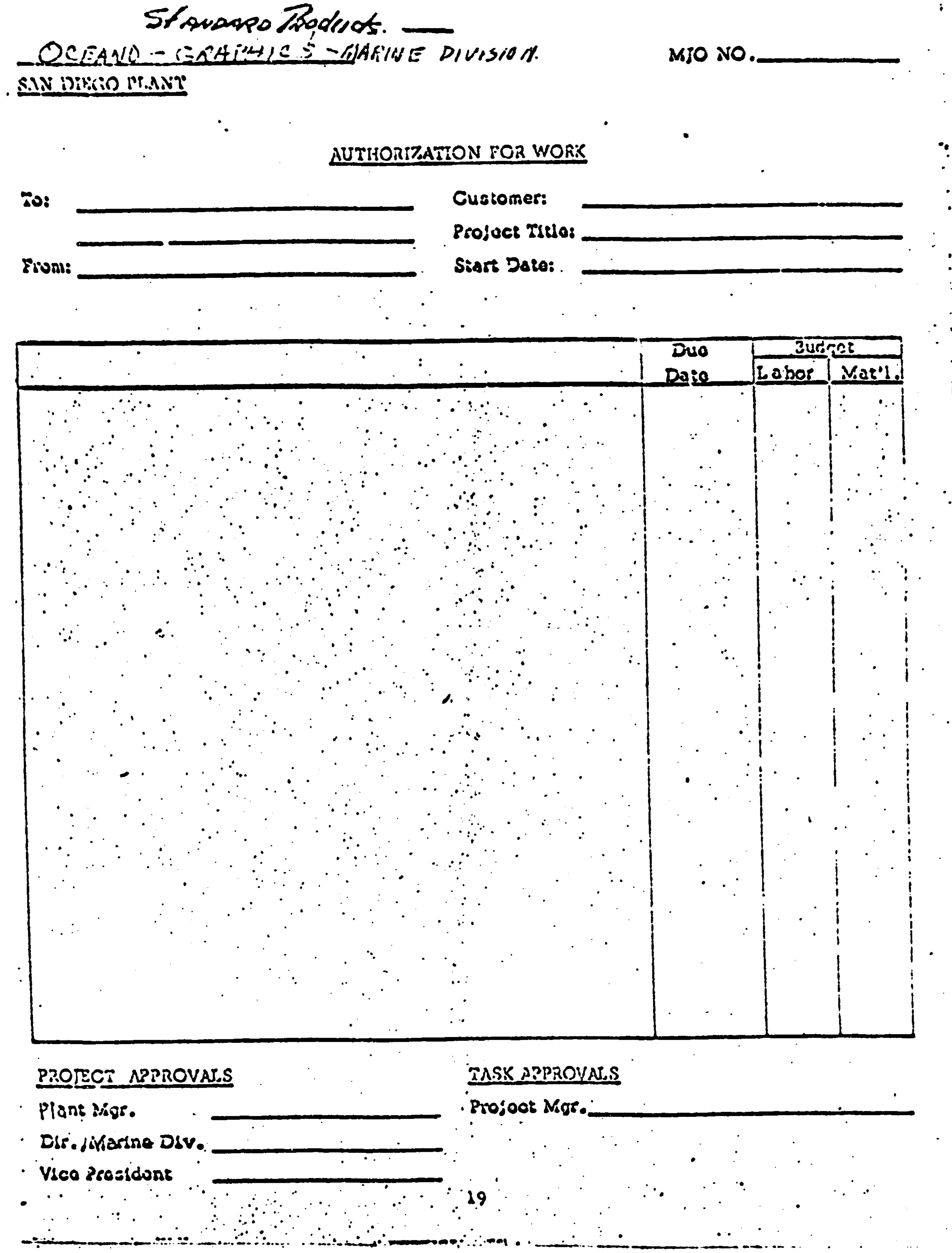
____________ Vice President, Administration

18

(2)

Figure SWAN 6

Authorization For Work



Standard Products.

OCEANO - GRAPHICS - MARINE DIVISION.

SAN DIEGO PLANT

MJO NO.________

AUTHORIZATION FOR WORK

To: ________ Customer: ________

________ Project Title: ________

From: ________ Start Date: ________

	Due Date	Budget	
		Labor	Mat'l.

PROJECT APPROVALS

Plant Mgr. ________

Dir./Marine Div. ________

Vice President ________

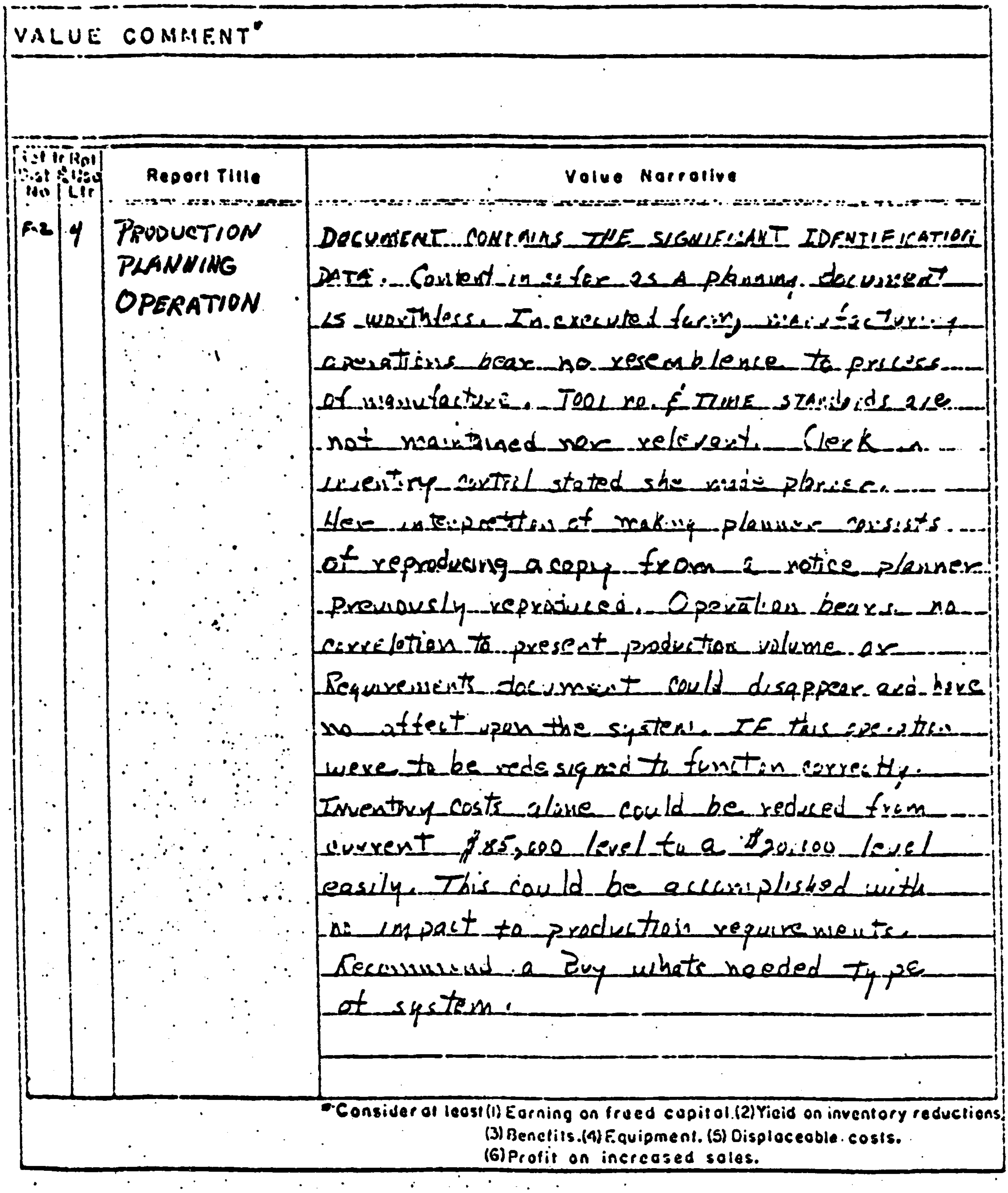
TASK APPROVALS

Project Mgr. ________

19

Figure SWAN 7

R. W. SWANSON, associates

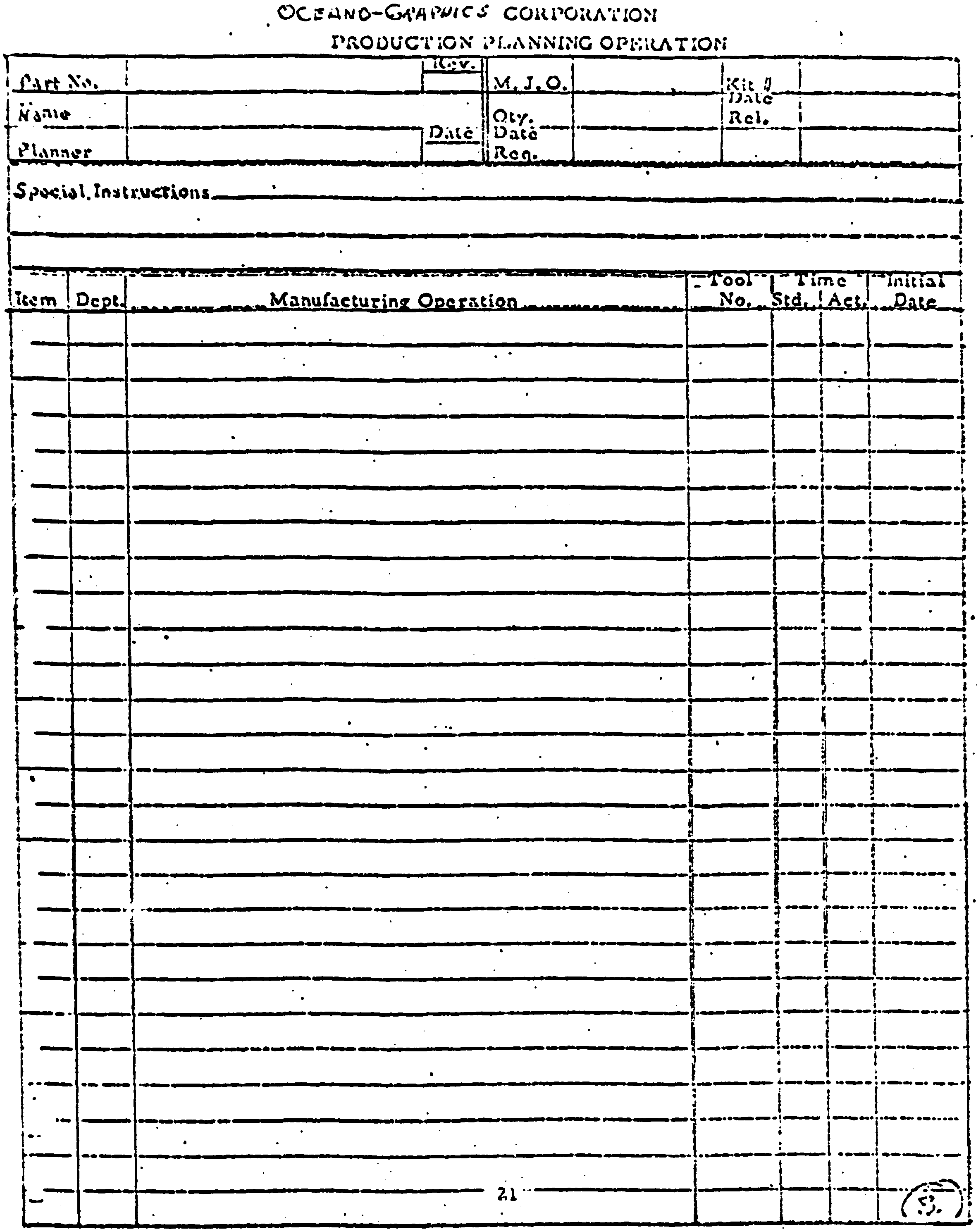


VALUE COMMENT*

[illegible] Dat No	Ir Rpt [illegible] Ltr	Report Title	Value Narrative
F-2	4	PRODUCTION PLANNING OPERATION	DOCUMENT CONTAINS THE SIGNIFICANT IDENTIFICATION DATA. Content insofar as a planning document is worthless. In executed form manufacturing operations bear no resemblence to process of manufacture. TOOL no. & TIME standards are not maintained nor relevant. Clerk in inventory control stated she made planner. Her interpretation of making planner consists of reproducing a copy from a notice planner previously reproduced. Operation bears no correlation to present production volume or Requirements document could disappear and have no affect upon the system. If this operation were to be redesigned to function correctly. Inventory costs alone could be reduced from current $85,000 level to a $20,100 level easily. This could be accomplished with no impact to production requirements. Recommend a Buy whate needed type of system.

*Consider at least (1) Earning on freed capital. (2) Yield on inventory reductions. (3) Benefits. (4) Equipment. (5) Displaceable costs. (6) Profit on increased sales.

Figure SWAN 8



OCEANO-GRAPHICS CORPORATION

PRODUCTION PLANNING OPERATION

Part No.		Rev.	M.J.O.		Kit #	
Name			Qty.		Date Rel.	
Planner		Date	Date Req.			

Special Instructions ____________________

____

____

Item	Dept.	Manufacturing Operation	Tool No.	Time Std.	Time Act.	Initial Date
		21				(3.)

Figure SWAN 9

a function of discerning which of many alternatives should be taken. Unfortunately, it is the empty wagon that makes the most noise.

Because of the lack of a formal flow of the planning function, it became impossible to measure economically the cost of that system and we, therefore, abandoned attempting to establish that of baseline. We also recognize that the Marine Division is basically an intermittent type of operation or job shop--organized about multipurpose machines to perform specific functions. This type of environment normally relies on heavy paperwork in order to control closely its cost and production capacities.

We believe the present management has designed systems that would accomplish the intent of control. However with the dynamics, particularly present in the oceano-graphic field, any effort to accomplish this manually with a sizable production volume would be futile. First, the cost of manpower, primarily clerical and management, would be economically infeasible. Second, the response time would be of such length as to negate most of the competitive advantages of the product and seriously affect the function of sales.

For the purposes of this report, we have included some of the analytical data used to arrive at the above conclusion. This, of course, is only a sample of the effort and volume of data collected. It is not our intent to burden this report with data that is meaningful only to fellow colleagues in this field. However, some insight should be given the reader as to the logic of the conclusions.

5.0 PROPOSED SYSTEM

Experience has shown that the gathering and dissemination of information is the manufacturing company's most difficult problem. Information is voluminous, scattered, and often difficult to obtain. In one of the early stages of this study, Mr. Fictitious emphatically stated he wanted a system that would give him "access to the facts." In addition, if we are to maintain the function of management control, we must provide for a means of planning and measurement that is currently lacking authenticity in the current system. Thus, our system must have a central information system and a framework that will facilitate mechanization.

We are proposing a single flow information system where the information will be accumulated in a production control center and where, literally, one set of books is maintained. This takes the form of records stored on computer disk files, readily accessible to all interested parties at a moment's inquiry. These records are designed to contain as much data as is deemed important to management. The accuracy of the records, also an essential element, will be easier to achieve and maintain, for only one set of archives will now be put to use.

The essential element of this system is the development of a data base that generally covers all the operational information needed to handle this company's business. It will be stored on disk files and, therefore, directly on-line with a computer. Because of this, summary and detail information can be accessed, updated, and retrieved from multiple entry points. Data is stored one way through reference to symbolic record field labels and can be printed in various output formats.

Each system of records is linked in a particular manner. For example, a part number, accessed through the basic record, may lead to what we call the product structure—a where-used going-to file, or standard routing record or the manufacturing sequence, or an open order status, or an open job summary, or a detail record. In the latter instance, the specific work center in which the job is being performed may even be pinpointed.

5.1 THE SYSTEM OVERVIEW

Drawing No. 1, System Overview, shows the interaction on data flow within the Marine Division organization. The interaction of these events is grouped within seven major areas:

a. Sales analysis

b. Engineering

c. Inventory control and production scheduling

d. Manufacturing facilities

e. Finance

f. Purchasing

g. Sales and distribution

While the data base is designed to process segments of all these areas, it does not contain the additional detail records needed to handle sales analysis, finance, and sales/distribution.

5.2 PRIMARY FLOW OF INFORMATION

The input of information leads from an initial input of customer orders and statistical sales background data to the final shipment of an order. A generalized statement of the system flow is divided into a planning phase and an execution phase.

Planning begins with the preparation and projection of order forecasts. Stock availability and on-order status are screened across

System Overview

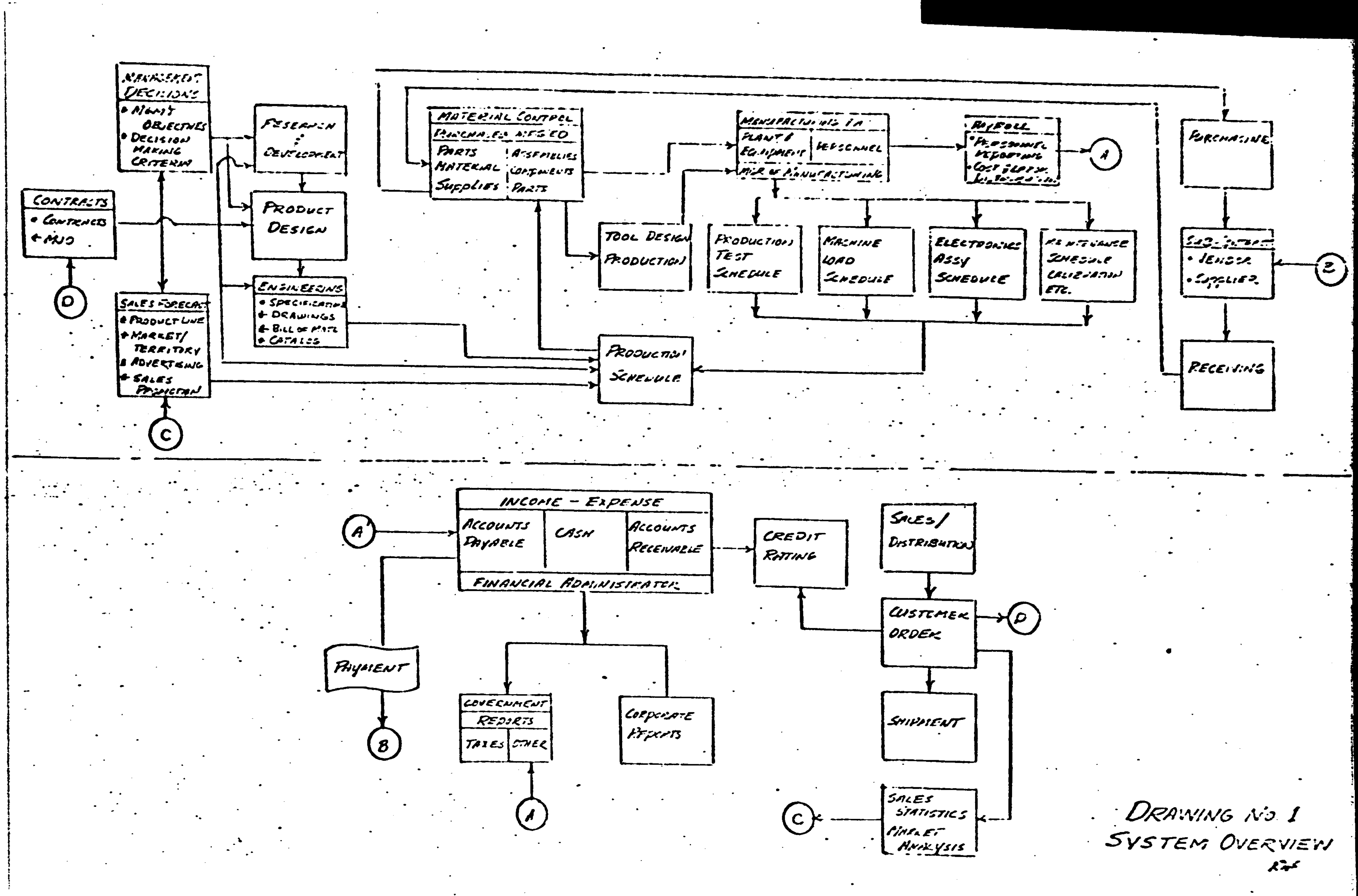


Figure SWAN 10

product inventory records, but family component characteristics of the product line must also be recognized. Product structure on bills of material enter into these decisions.

After a determination of net requirements, an order quantity analysis takes place to ascertain lot sizes and lead times for both purchased and manufactured items.

To-buy items are routed to where items are placed on a purchase requisition. At this point a selection of vendor is made, price and delivery are negotiated, and purchase order is released. Receipt cards and/or a scheduled receipt document may be prepared simultaneously with the purchase order and forwarded to the inspection-receiving area of the plant. An open purchase order record is now initiated for follow-up.

To-make items are routed to production planning for assembly and fabrication. Some similarity exists within these two units. An assembly order is generated for the assembly area, a shop order for the fabrication area. Material requisitions and job tickets accompany both documents. Three basic types of records (standard routing, work center load, and open job order) permit assembly and fabrication to schedule, to load, and to level the line or shop and to release the order paperwork.

Execution begins at the purchasing level with the need for order follow-up and vendor expediting. The vendor ships material, accompanying his shipment with packing lists and an invoice.

Varied execution functions are performed at the assembly and fabrication levels. Orders are dispatched, rescheduled, and expedited between work centers. In the meantime, current production reporting updates work center and open job order records.

5.3 SYSTEM FLOW

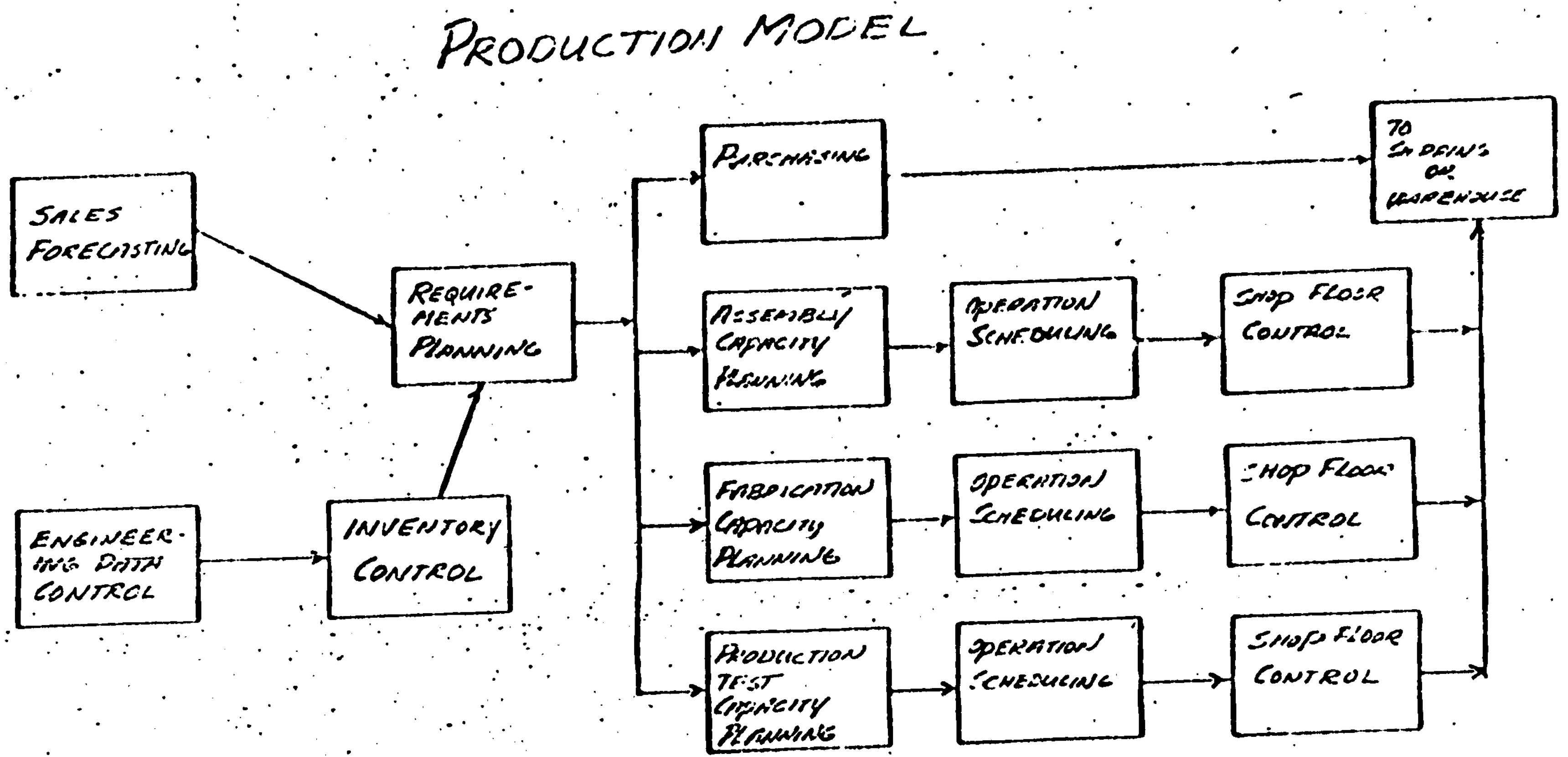
Our proposed system has been designed to fit this basic production model. For ease of implementation, eight subsystems have been developed. This allows for modular programming and should substantially reduce the cost of implementation and speed of accomplishment. The information flow begins from two directions. (See Drawing No. 2, Production Model.) The first path moves from engineering data control,. to inventory control, to requirements planning. The mission of the engineering data control subsystem is to organize and maintain basic records. These basic records are what we call the item master file, product structure file, standard routing file, and work center master. The subsystem has the added capacity or capability of retrieving information from the data base. Six retrieval functions are presently developed, three in assembly sequence and three in parts usage sequence.

An inventory control subsystem follows organization of basic records. On-hand inventory, usage history, and on-order fields are used in the item master file so that stock status reports can be generated. Thus, a major objective of this application area is record maintenance and updating inventory. With accessibility to such data, "when to order" and "how much to order" decisions are made.

A second flow line moves from sales forecasting to requirements planning. The sales forecasting subsystem analyzes historical demand data, which may be stored on the item master, to provide requirements planning with a gross finished product forecast plan.

The merger of requirements planning with inventory control now

Production Model



DATA BASE RECORDS

ENG DATA CONTROL

ITEM MASTER
PRODUCT STRUCTURE
STANDARD ROUTING
WORK CONTROL MASTER
OPEN JOB ORDER
ORDER SUMMARY

SALES FORECASTING

ITEM MASTER

INV. CONTROL

ITEM MASTER

REQUIREMENTS PLANNING

ITEM MASTER
PRODUCT STRUCTURE

PURCHASING

ITEM MASTER
PURCHASE MASTER
VENDOR MASTER
OPEN PURCHASE

CAPACITY PLANNING

ITEM MASTER
STANDARD ROUTING
WORK CENTER MASTER

OPERATION SCHEDULING

STANDARD ROUTING
WORK CENTER MASTER
OPEN JOB ORDER
TOOL MASTER

SHOP FLOOR CONTROL

ITEM MASTER
STANDARD ROUTING
PRODUCT STRUCTURE
OPEN JOB ORDER

DRAWING NO 2
[illegible]

Figure SWAN 11

makes it possible to determine net requirements, projected into time periods and schedule due dates. Product structure records are used at this point to allow breakdown of finished product items into individual components. These are similarly netted and projected into time periods. All of this results in planned orders, destined to each of the four links: purchasing, electronic assembly, fabrication, and product test.

Planned orders to purchasing result in material requisitions being prepared. Through the use of purchase master and vendor master records, a vendor may be selected and a purchase order with receiving documents created so that purchase follow-up can be initiated in the next sequence of events.

Planned orders to the three production areas go to the capacity planning sub-systems, or long-range scheduler. Its purpose is to identify overloads far enough in the future for both facility and manpower planning. After order start date calculations are performed (utilizing standard routing records), consideration is given to plant capacity. The work center master is used for this purpose. Available techniques are then used to level the loads. A work center load report, projected by time period, is one of the key output documents. The operation scheduling accepts orders which have gone through a releasing cycle from capacity planning and schedules the work center within its short-range time span. Dispatching sequences are prepared and analyses made of the loads. Priority rules are set and order completion dates determined. To the short-range scheduling phase of this subsystem, we have added the control of tools. A tool master record is designed for this function.

Shop floor control is the final subsystem in the flow line. It prepares the shop packets and other factory documentation. It also constructs the open job summary and operation detail records so that the progress of the work can be reported. Feedback is one of its more important functions so that the system can respond to change.

5.4 STANDARDIZED RECORDS

A set of standardized record layouts is included in this report. These records are designed as the data base to mechanize the application areas we have discussed and lead toward developing a single information flow concept.

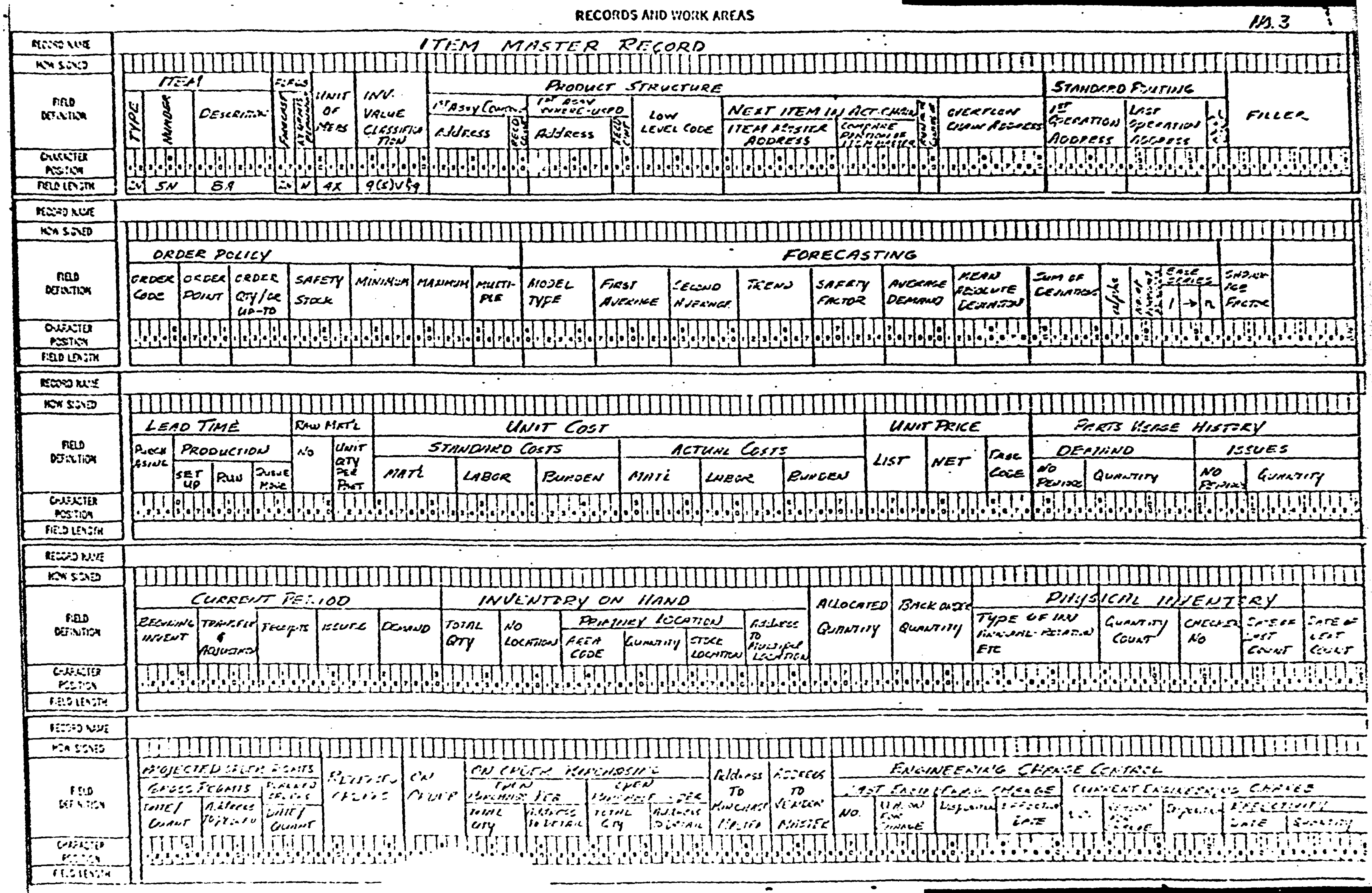
These records contain the fields we consider necessary to enable Oceano-Graphic Division to utilize and control their control production output requirements. Each record is described in detail in Drawings numbered three through eight. Their respective lengths determine the size of the storage requirements of any data processing equipment employed. Chart No. 9 is included to show how the chaining sequence of the records is interfaced. This interaction of the records achieve for us two major functions, first to provide access to the facts and secondly, through this chaining sequence a single data base is developed to allow for implementation of the single information flow showing how a change, such as an engineering change, updates the whole data base, which is common to all. Thus, we accomplish one of our main objectives; one set of books is maintained.

5.5 BENEFITS OF THE PROPOSED SYSTEM

a. A Plan for Growth

A plan can be developed to begin implementation for each of the application subsystems leading to the single information flow system. The system can grow as Oceano-

File Layout



RECORDS AND WORK AREAS

No. 3

RECORD NAME	ITEM MASTER RECORD																			
HOW SIGNED																				
FIELD DEFINITION	ITEM			FLAGS		UNIT OF MEAS	INV. VALUE CLASSIFICATION	PRODUCT STRUCTURE									STANDARD ROUTING			FILLER
								1st Assy Comp		1st ASSY WHERE-USED			NEXT ITEM IN ACT. CHAIN							
	TYPE	NUMBER	DESCRIPTION	FORECAST	AT HAND PLANNING			Address	RECD CODE	Address	RECD CONT	LOW LEVEL CODE	ITEM MASTER ADDRESS	COMPARE PORTION OF ITEM MASTER	[illegible]	OVERFLOW CHAIN ADDRESS	1st OPERATION ADDRESS	LAST OPERATION ADDRESS	[illegible]	
CHARACTER POSITION																				
FIELD LENGTH	2N	5N	6A	2X	N	4X	9(5)V99													

RECORD NAME																					
HOW SIGNED																					
FIELD DEFINITION	ORDER POLICY							FORECASTING													
																		BASE SERIES			
	ORDER CODE	ORDER POINT	ORDER QTY/OR UP-TO	SAFETY STOCK	MINIMUM	MAXIMUM	MULTIPLE	MODEL TYPE	FIRST AVERAGE	SECOND AVERAGE	TREND	SAFETY FACTOR	AVERAGE DEMAND	MEAN ABSOLUTE DEVIATION	SUM OF DEVIATIONS	alpha	NO. OF PERIODS	1	→	n	SHRINKAGE FACTOR
CHARACTER POSITION																					
FIELD LENGTH																					

RECORD NAME																				
HOW SIGNED																				
FIELD DEFINITION	LEAD TIME				RAW MAT'L		UNIT COST						UNIT PRICE			PARTS USAGE HISTORY				
	PURCHASING	PRODUCTION			NO	UNIT QTY PER PART	STANDARD COSTS			ACTUAL COSTS			LIST	NET	TABLE CODE	DEMAND		ISSUES		
		SET UP	RUN	QUEUE MOVE			MAT'L	LABOR	BURDEN	MAT'L	LABOR	BURDEN				NO PERIODS	QUANTITY	NO PERIODS	QUANTITY	
CHARACTER POSITION																				
FIELD LENGTH																				

RECORD NAME																			
HOW SIGNED																			
FIELD DEFINITION	CURRENT PERIOD					INVENTORY ON HAND						ALLOCATED QUANTITY	BACK ORDER QUANTITY	PHYSICAL INVENTORY					
								PRIMARY LOCATION											
	BEGINNING INVENT	TRANSFER & ADJUSTMENTS	RECEIPTS	ISSUES	DEMAND	TOTAL QTY	NO LOCATIONS	AREA CODE	QUANTITY	STOCK LOCATION	ADDRESS TO MULTIPLE LOCATION			TYPE OF INV ANNUAL-ROTATING ETC	QUANTITY COUNT	CHECKER NO	DATE OF LAST COUNT	DATE OF LAST COUNT	
CHARACTER POSITION																			
FIELD LENGTH																			

RECORD NAME																				
HOW SIGNED																				
FIELD DEFINITION	PROJECTED GROSS REQUIREMENTS						ON ORDER PURCHASING						ENGINEERING CHANGE CONTROL							
	GROSS REQMTS		PLANNED ORDERS	RELEASED ORDERS	ON ORDER	OPEN PURCHASE ORDER		OPEN [illegible] ORDER		ADDRESS TO PURCHASE MASTER	ADDRESS TO VENDOR MASTER	LAST ENGINEERING CHANGE				CURRENT ENGINEERING CHANGES				
	DATE/QUANT	ADDRESS TO DETAIL	DATE/QUANT			TOTAL QTY	ADDRESS TO DETAIL	TOTAL QTY	ADDRESS TO DETAIL			NO.	REASON FOR CHANGE	DISPOSITION	EFFECTIVE DATE	[illegible]	REASON FOR CHANGE	DISPOSITION	EFFECTIVITY DATE	EFFECTIVITY [illegible]
CHARACTER POSITION																				
FIELD LENGTH																				

Figure SWAN 12

Chaining Concept

NO. 4

OTHER FILES TO BE DESIGNED

PRODUCT STRUCTURE FILE
PEGGED REQUIREMENTS
OPEN PURCHASE REQUISITIONS
OPEN PURCHASE ORDERS
VENDOR MASTER
PURCHASE MASTER
OPEN JOB ORDER SUMMARY
STANDARD ROUTING
WORK CENTER MASTER
OPEN JOB OPERATION DETAIL
TOOL MASTER

CHAINING CONCEPT

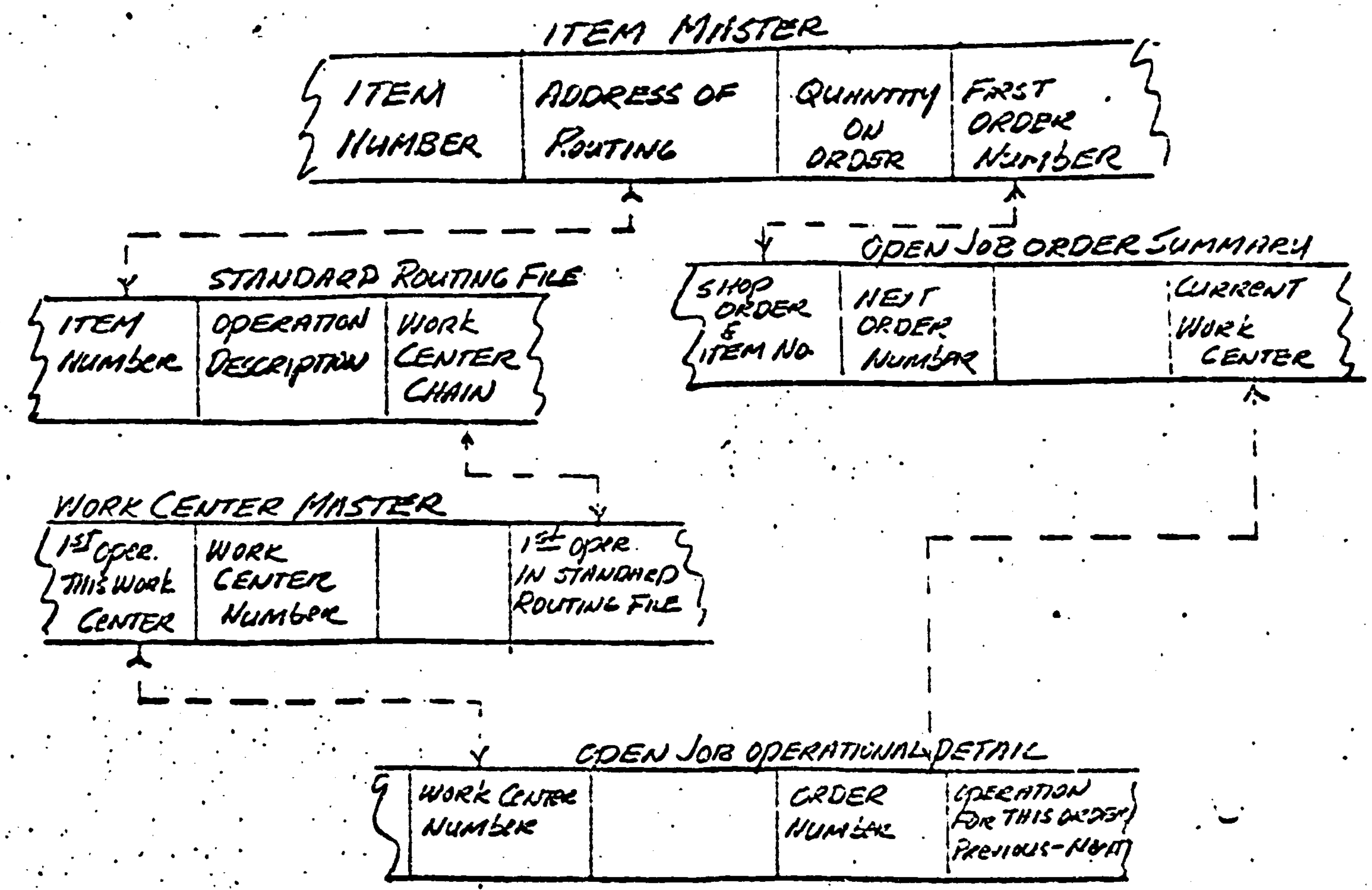
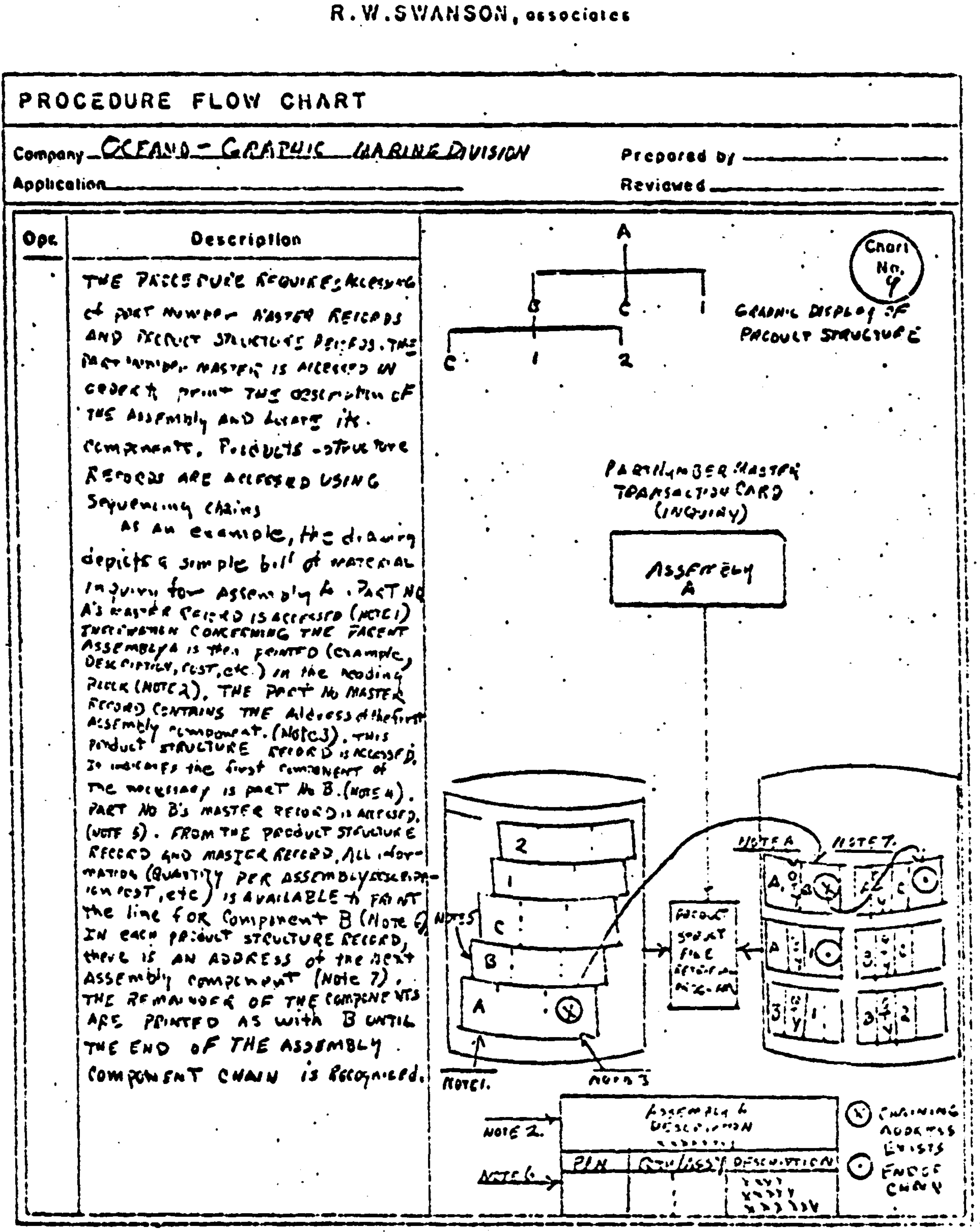


Figure SWAN 13

Procedure Flow Chart



R.W.SWANSON, associates

PROCEDURE FLOW CHART

Company OCEANO - GRAPHIC MARINE DIVISION Prepared by ________

Application ________ Reviewed ________

Opr.	Description
	THE PROCEDURE REQUIRES ACCESSING of PART NUMBER MASTER RECORDS AND PRODUCT STRUCTURE RECORDS. THE PART NUMBER MASTER IS ACCESSED IN ORDER to print THE DESCRIPTION OF THE ASSEMBLY AND LOCATE its COMPONENTS. PRODUCTS STRUCTURE RECORDS ARE ACCESSED USING Sequencing chains As an example, the drawing depicts a simple bill of MATERIAL inquiry for ASSEMBLY A. PART NO A's MASTER RECORD IS ACCESSED (NOTE 1) INFORMATION CONCERNING THE PARENT ASSEMBLY A IS THEN PRINTED (example, DESCRIPTION, COST, etc.) in the heading BLOCK (NOTE 2). THE PART No MASTER RECORD CONTAINS THE Address of the first ASSEMBLY COMPONENT. (NOTE 3). THIS PRODUCT STRUCTURE RECORD is ACCESSED. It indicates the first component of the necessary is PART No B. (NOTE 4). PART NO B's MASTER RECORD is ACCESSED. (NOTE 5). FROM THE PRODUCT STRUCTURE RECORD AND MASTER RECORD, ALL information (QUANTITY PER ASSEMBLY DESCRIPTION COST, etc) IS AVAILABLE to PRINT the line for Component B (Note 6) IN EACH PRODUCT STRUCTURE RECORD, there IS AN ADDRESS of the next ASSEMBLY COMPONENT (Note 7). THE REMAINDER OF THE COMPONENTS ARE PRINTED AS with B UNTIL THE END OF THE ASSEMBLY COMPONENT CHAIN is Recognized.

Figure SWAN 14

Graphic grows. And Oceano-Graphic will obtain tangible results long before the total system is installed.

b. Standardization

Information is common to all--it is known and called by one description. One set of records makes this possible. Thus, it makes planning easier with standardized consistency in estimating and pricing.

c. Extensive Data Base

The record base has two important features: accessibility and accuracy. Information is accessible through inquiry to multiple points; detail is available through chaining to all related records. No longer will it be necessary to spend hours or days searching file drawers or ledger cards. Also, information is more accurate; it is updated in only one place. Standard transactions processed within each subsystem assure complete record maintenance.

d. Modular Program Design

Chart No. 10 shows the modular programming concepts. We can obtain tangible results before the total system is installed. This has been the major drawback of the single information flow concept. This will also allow a systematic plan of implementation with easily defined milestones for time and budget control.

e. All production information is now directed into a single channel. Levels of operating and management personnel are made more aware. With this assurance, the following should happen:

Modular Programming Concept

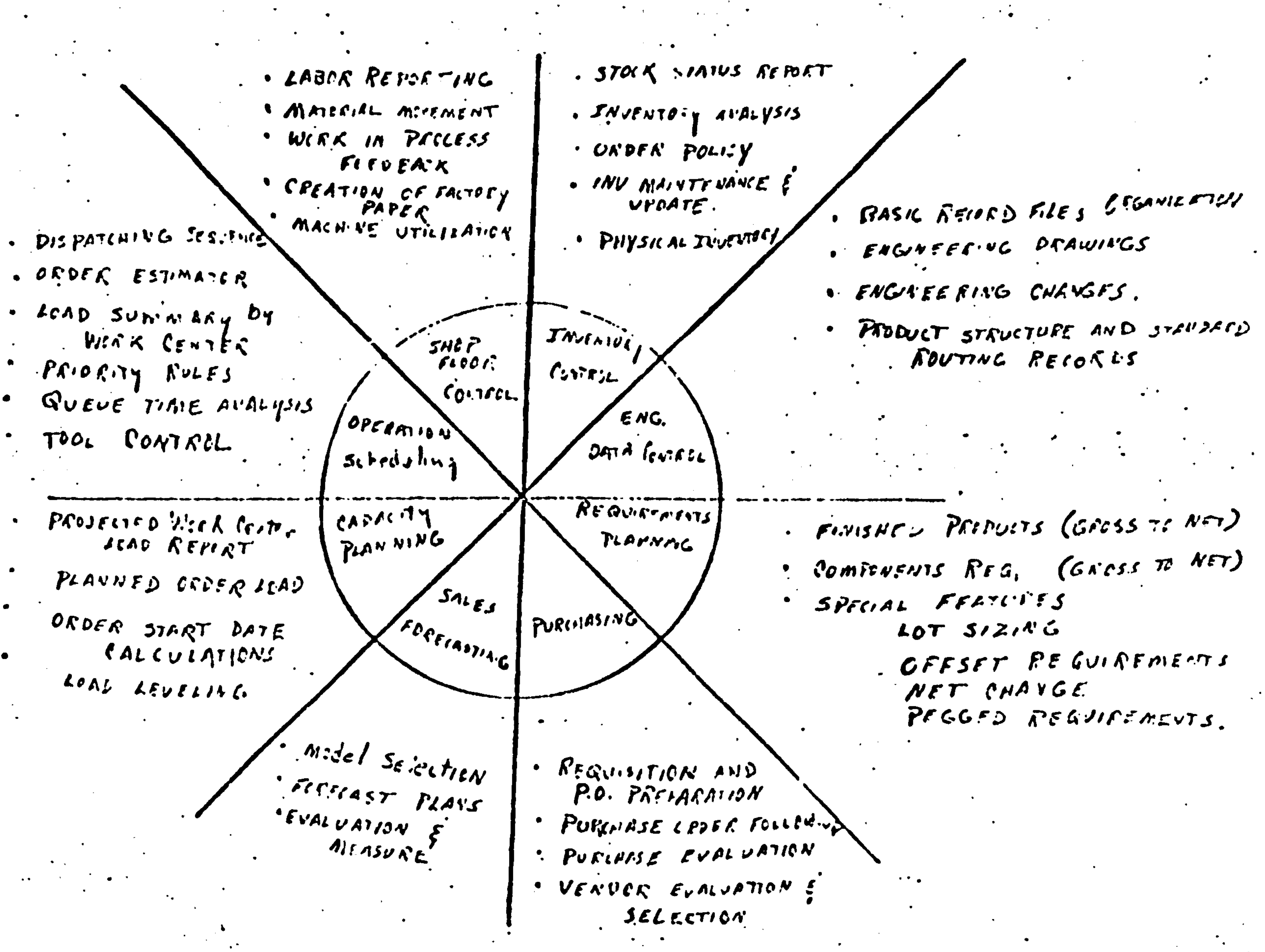


Figure SWAN 15

(1) Cost can be closely controlled, better surveillance over overtime hours, inventory, and machines--the key to cost reduction.

(2) More efficient planning.

(3) More time available to react to changes.

(4) Less waste, reduced information costs, more profits.

6.0 COST OF IMPLEMENTATION

The cost of implementation is a function of two variables. Machine costs of equipment capable of the level of mechanization required for this system represent one variable. Software cost, the cost of programming, represents a one-time cost necessary to the operation of the system.

6.1 HARDWARE COSTS

The hardware costs have not, for the purposes of this report, been precisely defined by the soliciting of quotes or bid proposals. However, with our experience directly in this field, a realistic estimate can be submitted. For instance, an IBM System 000/000 with a configuration to handle this application would cost $0,000 per month through the first years of operation. The addition of terminals as the need arises will increase the hardware costs to approximately $0,000 per month at later stages of development.

Operating and programming costs after the system is fully in operation will be approximately $000 per month. It is imagined this capability would be developed from within the current organization and no additional people added to the payroll.

6.2 SOFTWARE COSTS

The normal rule of thumb for predicting software cost for a new system such as this is to take the first year's cost for the hardware; in this case, $00,000. However, the sophistication of this system, which is considerably more elaborate than a payroll system, etc. (which is usually the makeup of software costs), can be accomplished for considerably less investment, since developed, canned programs, programs already written, can be used to minimize this expenditure. However, the canned programs still must be modified and patched to fit the Oceano-Graphic operation. We, therefore, estimate the cost of programming, debugging, and implementation of the system to be $00,000, approximately $0,000 per month for a period of ten months (Reference Drawing No. 11, Implementation Plan).

ACTIVITIES

Question 1. The existing system was evaluated from what points of view?

Answer:

A. What are the objectives or premise for operation?

B. How was the premise put into operation, i.e., how did the planning take place?

C. In what manner was the plan implemented?

D. What was the measurement for determining how well the plan was being accomplished?

E. What was the medium for feedback and how did it affect the process?

Question 2. What was the object of the engineering data control?

Answer: To organize and maintain basic records.

Question 3. What are the basic records?

Answer: Item Masters, Product Structure, Standard Routing and Work Center Master.

Question 4. What other record is included under Engineering Data Control?

Answer: Open Job Order and Order Summary.

Question 5. What is the major objective of the Inventory Control System?

Answer: Record maintenance and updating of the inventory (it indicates when to order and how to order).

Question 6. What else is of importance as far as inventory control is concerned?

Answer: On-order fields are used in the Item Master so that Stock Status reports can be generated.

Question 7. What functions do the Sales Forecasting and Requirements Planning Connection perform?

Answer: The analization of historical demand data, which may be stored on the item master to provide requirements planning with a gross finished product plan.

Question 8. What is the result of the merger of Requirements Planning with Inventory Control?

Answer: This makes it possible to determine the net time requirements, projected into time periods and schedule due dates.

Question 9. What other records are used?

Answer: Product Structure records are used to allow the breakdown of finished product items into individual components.

Question 10. What are the names of the four links that the Planned Orders are distributed to?

Answer: Purchasing, Electronic Assembly, Fabrication and Production Test.

Question 11. What are the four files involved when the Planned Orders reach the purchasing link?

Answer: Item Master, Purchase Master, Vendor Master and Open Purchase Order.

Question 12. How may a vendor be selected?

Answer: Through the use of a Purchase Master and a Vendor Master record.

Question 13. What is the purpose of the Capacity Planning Subsystem?

Answer: It identifies overloads far enough in the future to allow manpower planning and facility planning.

Question 14. What records are used to calculate the Order Start Date?

Answer: Standard Routing Records.

Question 15. Give the name of one of the key documents developed in this area according to the Production Model.

Answer: Work Center Load Report, projected by time period.

Question 16. What is the purpose of the Operation Scheduling Section?

Answer: It accepts orders which have gone through a releasing cycle from Capacity Planning, and schedules the work center within its short range time span.

Question 17. Shop Floor Control is the final subsystem in the flow. Give two of its functions.

Answer: A. It prepares the shop packets and other factory documentation.

B. It constructs the Open Job Summary and Operating

Detain Records so that the progress of the work can be reported.

Question 18. If we examine the set of Standardized Layout Records, we note that certain fields were used. Why?

Answer: The fields were used to allow Oceano-Graphic to utilize and control their production output requirements.

Question 19. What can the lengths of the records help determine?

Answer: The lengths can determine the size of storage requirements of any data processing equipment used.

Question 20. Chart 9 shows how chaining sequence of the record is interfaced. What two functions does this achieve?

Answer: A. This allows the development of the necessary retrieval functions to provide access to the facts.

B. The chaining (or linking) allows the development of a single data base which means a single information concept.

(Make a check mark for your answer to Data Management Topics--included, not included, or irrelevant in the Swanson Study.)

FEASIBILITY STUDY

1. Basic Steps in Development of Business Systems

    1. Problem definition
    2. Application Research
    3. Scope of study
        a. Organizational boundaries
        b. Objective of study
        c. Resources available for study
    4. Objectives
    5. Target dates
    6. Study phase responsibility
    7. Education of several departments
    8. Management role

2. Factors Included in Systems Report to Management

    1. Introduction
    2. Description of study
    3. Costs
    4. Organizational changes

3. Systems and Procedures Covered

4. Functions of Systems and Procedures Department

    1. Systems analysis and design
    2. Forms design and control
    3. Records management and retention
    4. Report analysis
    5. Preparation of written procedures
    6. Work distribution
    7. Process flow chart
    8. Procedure flow chart
    9. Work measurement
    10. Time standards
    11. Time and motion study
    12. Forms control
        a. Retain only necessary forms
        b. Renew and reuse
        c. Proper design and manufacture
        d. One person assigned to forms control

5. Forms Design Basic Consideration

    1. Purpose
    2. Decision for specific information

   3. Use and purpose
   4. Logical placement of items
   5. Physical placement of items
   6. Analyzation of number of copies

6. Records Management

   1. Creation and use of records
   2. Distribution
   3. Simplification of paperwork
   4. Data retained
   5. Filing procedures
   6. Security

7. Company Manuals

   1. Organizational
   2. Policy
   3. Operation
   4. General information

8. Installation of Electronic Data Processing

   1. Systems design
   2. Personnel selection and training
   3. File conversion
   4. Forms design
   5. Programming
   6. Testing of programs
   7. Facilities design and preparation
   8. Scheduling and testing the system
   9. Changeover
   10. Feedback evaluation

9. Elements of Good Reports

   1. Accurate
   2. Clear
   3. Relevant
   4. Current

10. File Design

    1. Volume consideration
    2. Timing
    3. Sorting

11. Fact Gathering Techniques of a System Study

    1. Interview
    2. Questionnaire
    3. Review Records

12. Type of Informaticn Gathered

    1. Historical
    2. Cost
    3. Current procedure
    4. Effectiveness
    5. Relationship between departments

CONSTRUCTION

The Swanson Study uses a PERT type plan with Chart 11. The construction program can be used with this type of plan, and the cost program, which follows can be used to account for costs that occur daily.

Use the tape cassette to analyze this program. It will develop an introduction to the Basic Assembly Language used in this book.

The object of this program is to find the shortest time in which a construction project can be completed and to find the activities that limit the time for completion.

First a network for the project, such as Figure C-1 is drawn. Each arrow represents an activity. The numbered circles represent events. The numbers alongside the arrows are the work time, in days, for the activity. The diagram is laid out such that all activities leading to an event must be completed before any activities leading away from that event can begin. The ending number must be higher than the beginning event number.

The data is read into the computer, one activity per IBM card. The first sixteen spaces are reserved for the activity description, the next three spaces for the starting event number, then three spaces for the end event number, finally three spaces for the work time.

The data is stored in core storage, first the sixteen bytes of description, then two bytes for start event (packed format is used for all number data in the storage tables), two bytes for end event number, two bytes for normal work time. In addition, space is reserved in memory for start time, end time, slack time and an index. Two bytes of core storage are reserved for each activity in the table.

The program proceeds:

1. Initialize
2. Print header
3. Read all data cards and store information in table
4. Print out table
5. Sort the table on basis of lowest starting event number
6. Locate activity with the lowest starting event number
7. Locate and place in table TTWO all activities with end event numbers that match the start event numbers of (6)
8. Sort table TTWO on the basis of ending times
9. Subtract ending time of each activity from the ending time of the largest activity in table TTWO
10. Enter difference in slack time position
11. Enter the longest event time from table TTWO into start time for activity (6)
12. Compute end time for activity (6)
13. Index activity (6)
14. Transfer table TTWO and activity (6) back into main table
15. Test for last event
16. NO-back to (6)
17. Print out main table for second time
18. Sort main table on basis of lowest slack time first
19. Cut table off after a last event with no slack time
20. Sort new short table on basis of start event number, highest first
21. Print headers
22. Print out first activity in table (this activity is actually the last activity to be accomplished on the project)

23. Index first activity with a %%

24. Locate and print all activities with end event numbers the same start event as 21

25. Index each activity printed with a cret

26. Return to beginning of table, find first activity not indexed with %%, but indexed with a cret

27. Locate and print all activities with end event numbers the same as start event number 25

28. At last event end program

Construction--Flow Chart

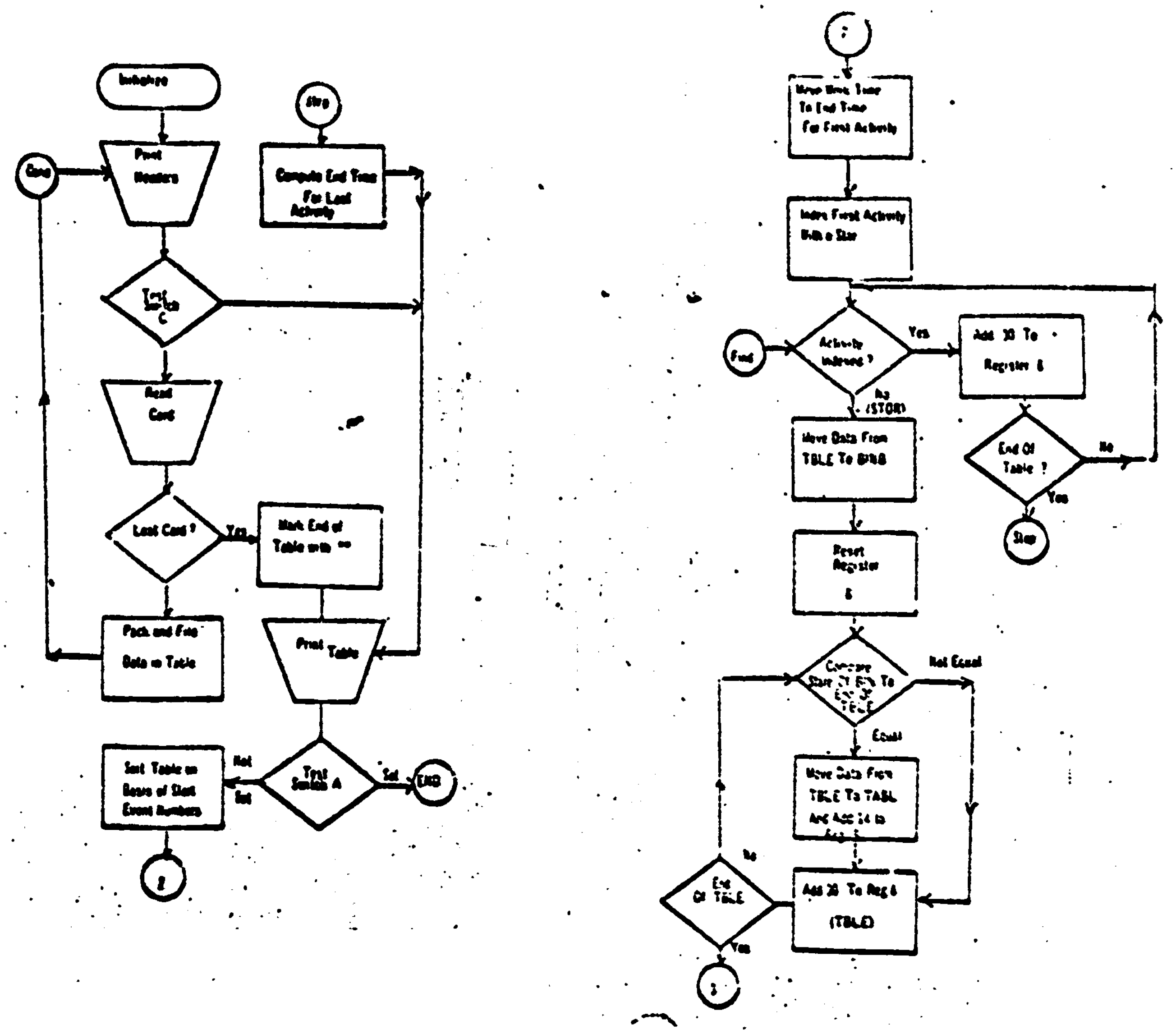

Initialize
Cont
Print Headers
Step
Compute End Time For Last Activity
Test Switch C
Read Card
Last Card ?
Yes
Mark End of Table with **
Pack and File Data in Table
Print Table
Sort Table on Basis of Start Event Numbers
Not Set
Test Switch A
Set
END
2
Move Work Time To End Time For First Activity
Index First Activity With a Star
Find
Activity Indexed ?
Yes
Add 30 To Register 6
End Of Table ?
No
Yes
Stop
No (STOR)
Move Data From TBLE To BINB
Reset Register 6
Compare Start Of BINB To End Of TBLE
Not Equal
Equal
Move Data From TBLE To TABL And Add 14 to Reg 5
Add 30 To Reg 6 (TBLE)
End Of TBLE
No
Yes
3

Figure FC-02

Construction--Flow Chart

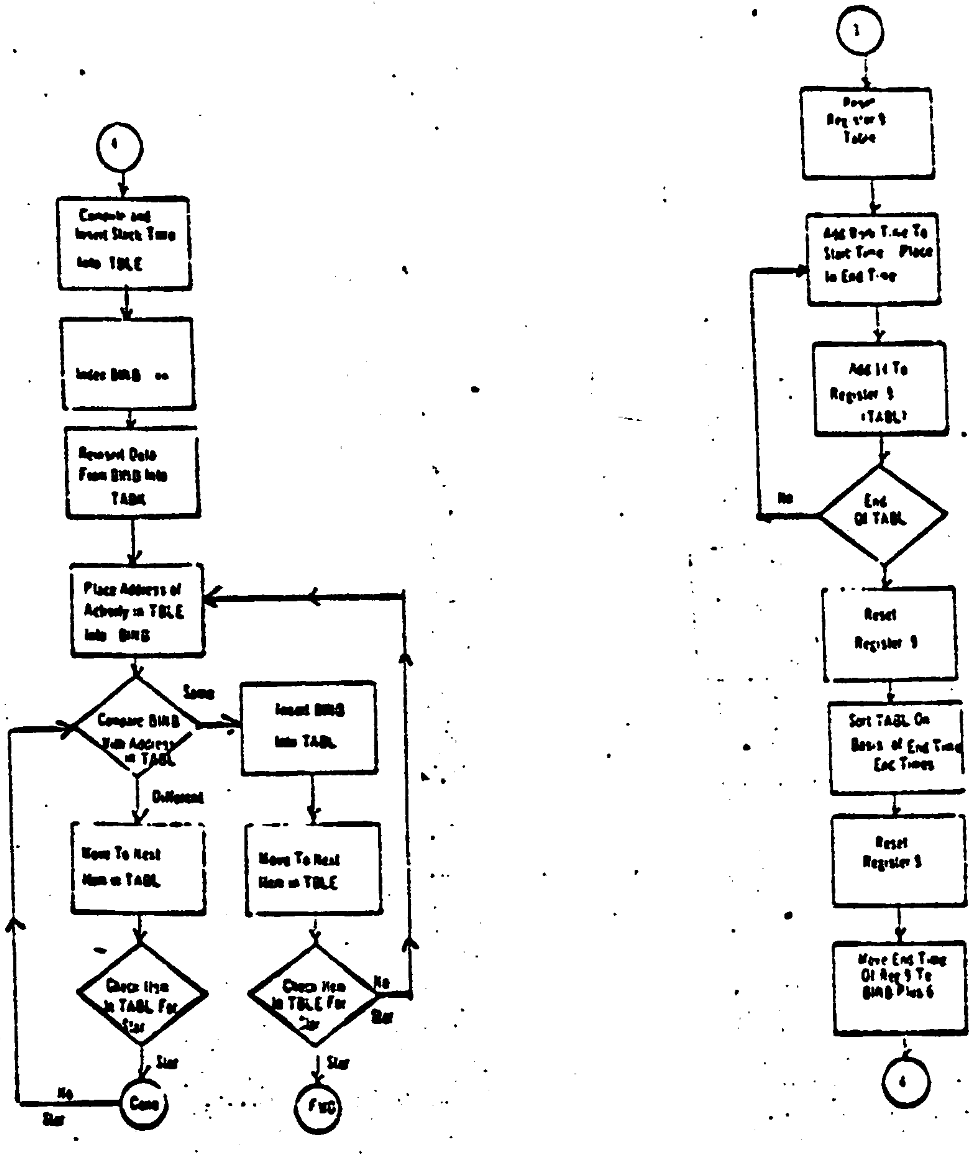

4
Compute and Insert Slack Time Into TBLE
Index BIRB
Reverse Data From BIRB Into TABL
Place Address of Activity in TBLE Into BIRB
Compare BIRB With Address in TABL
Same
Insert BIRB Into TABL
Different
Move To Next Item in TABL
Move To Next Item in TBLE
Check Item in TABL For Star
Check Item in TBLE For Star
No Star
Star
Star
No Star
Cons
FWC
3
Reset Register 9 Table
Add Dur. Time To Start Time Place In End Time
Add 14 To Register 9 (TABL)
End Of TABL
No
Reset Register 9
Sort TABL On Basis of End Time End Times
Reset Register 9
Move End Time Of Reg 9 To BIRB Plus 6
4

Figure FC-03

Construction--Flow Chart

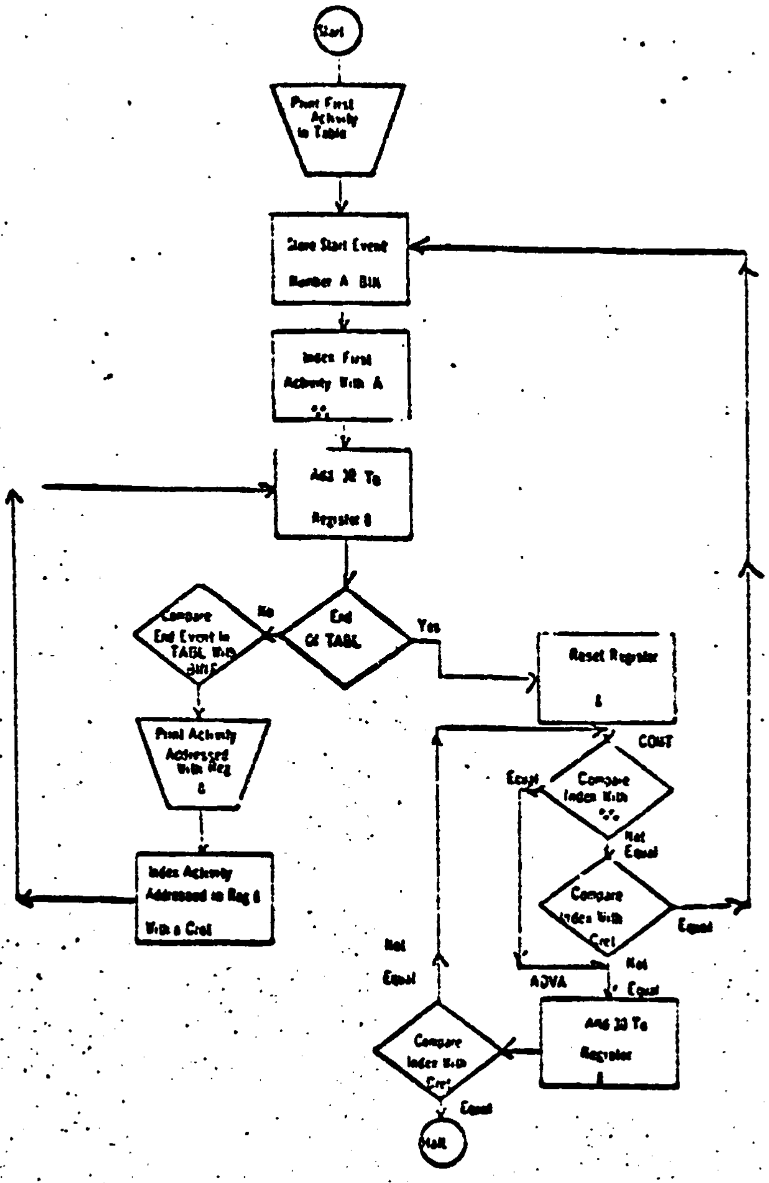

Start
Print First Activity in Table
Store Start Event Number A BIN
Index First Activity With A
Add 30 To Register 8
End Of TABL
No
Yes
Compare End Event In TABL With BIN
Print Activity Addressed With Reg 8
Index Activity Addressed in Reg 8 With a Crel
Reset Register 8
CONT
Compare Index With
Equal
Not Equal
Compare Index With Crel
Equal
Not Equal
ADVA
Add 30 To Register 8
Compare Index With Crel
Not Equal
Equal
Halt

Figure FC-01

Construction Program--Network Layout

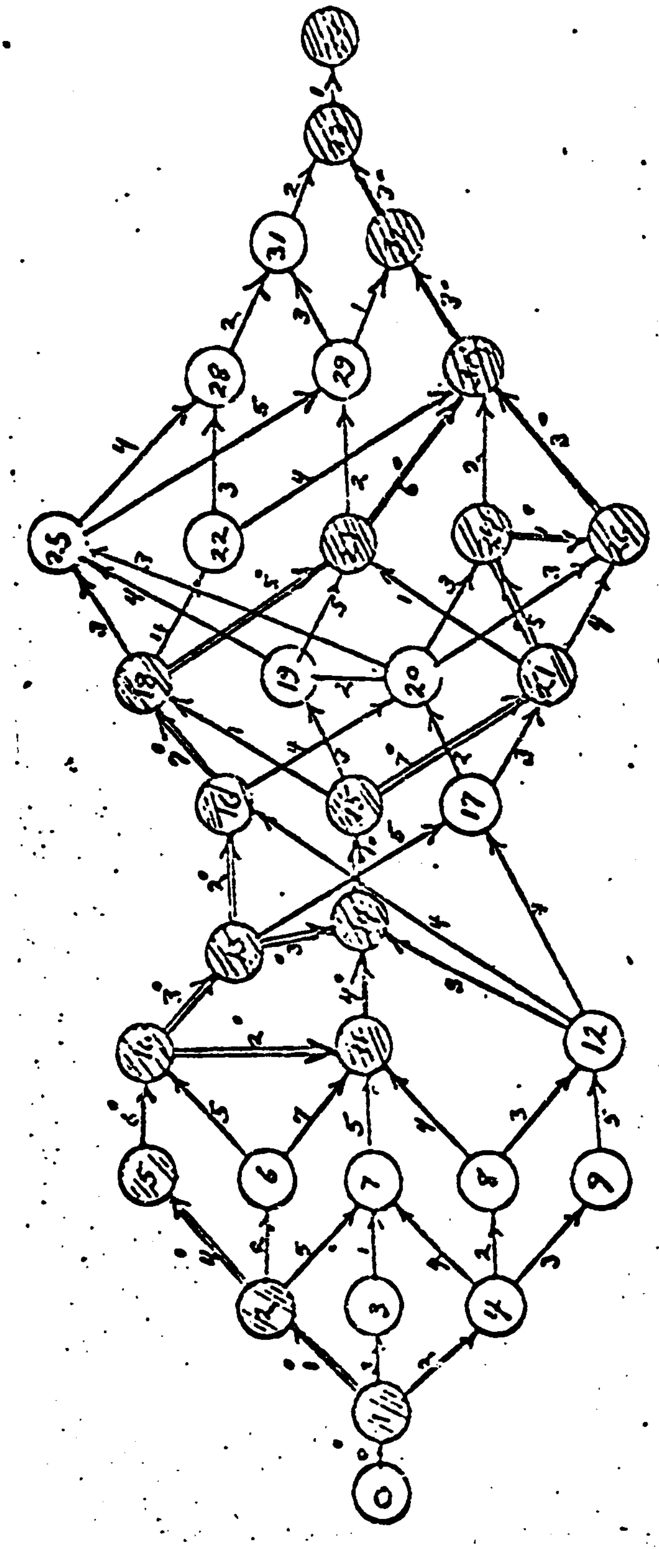


Figure NET-01

Construction Program

Sort Routine

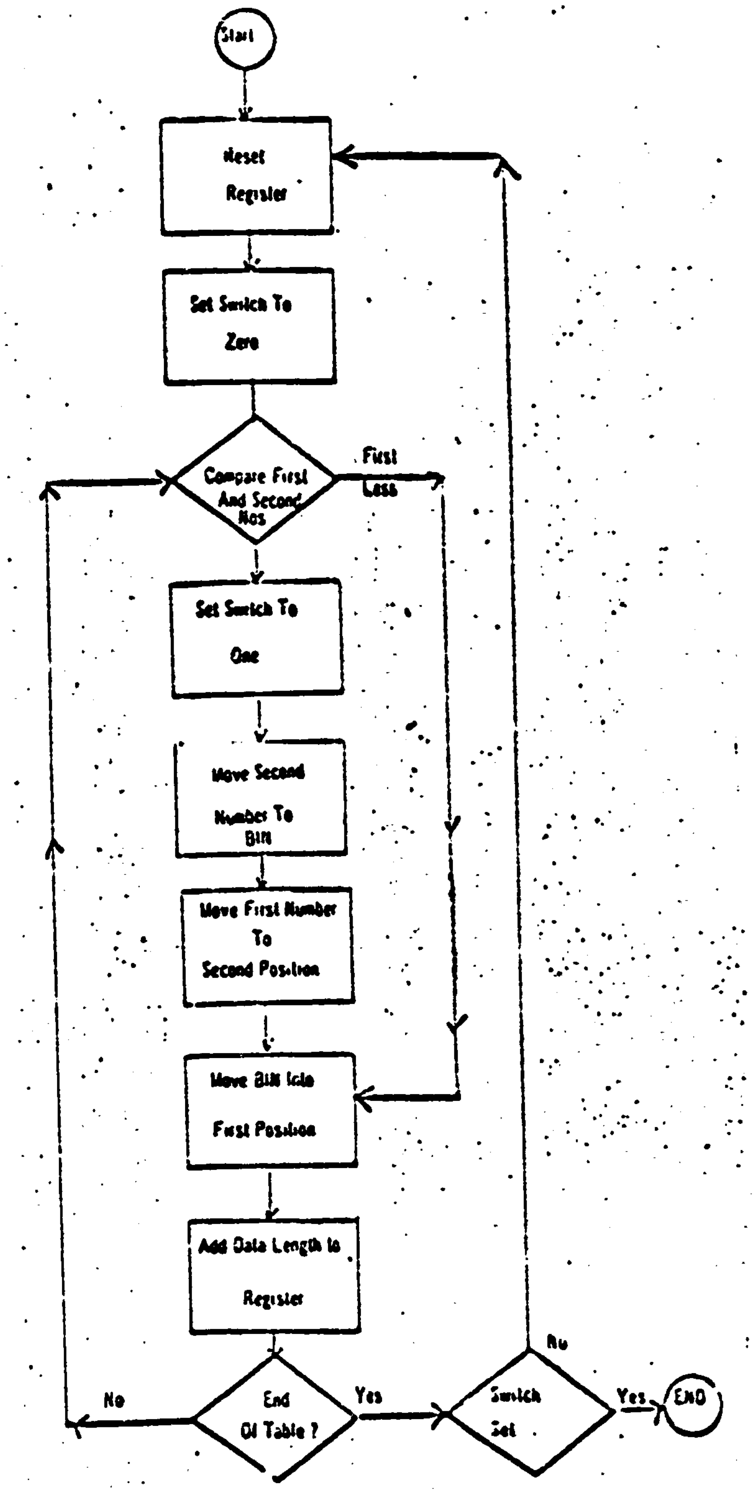

Start
Reset Register
Set Switch To Zero
Compare First And Second Nos
First Less
Set Switch To One
Move Second Number To Bin
Move First Number To Second Position
Move Bin Into First Position
Add Data Length to Register
End Of Table ?
No
Yes
Switch Set
No
Yes
END

Figure SRT-01

Construction Program Print Out

```
ALCA    10  090A  01      ADDA    10  0214  05      ADVA    10  0504  01      BASE    10  090F  01
BIN     10  0999  10      BINA    10  09A7  00      BING    10  09C5  00      BINE    10  090C  01
CDIN    10  0AA7  4F      COMP    10  0256  01      COMU    10  0A6A  01      COMB    10  0500  05
COMS    10  0512  01      CONF    10  05C2  05      CONE    10  015A  05      CONS    10  0154  00
CRLT    10  0995  01      DATA    10  07E1  2F      DATB    10  0A10  0A      DESC    10  0667  0F
ENDE    10  0A7A  02      ENDI    10  0A84  01      FILE    10  01DA  05      FINA    10  0200  05
FINB    10  05CA  01      FIND    10  02A8  05      FINE    10  0460  01      FIEN    10  0904  01
GIN1    10  01A1  05      GIGO    10  01A8  01      GIVN    10  01AA  01      HALT    10  0424  03
HOR1    10  0730  1A      HOR2    10  0A20  1A      HOR3    10  0BA5  2C      HOR4    10  07A5  77
HOR5    10  0910  1C      HERE    10  0156  03      INPL    10  04BC  05      INST    10  04A2  01
ITEM    10  0101  01      LOOP    10  0234  05      LOPA    10  0318  05      LOPR    10  0494  05
LOPC    10  0912  05      LOPB    10  0542  05      LOPF    10  05A4  03      LSCO    10  021C  05
MASA    10  0910  07      NINO    10  0600  03      NULL    10  0A1F  43      OUT     10  0444  01
OUTO    10  094C  05      PASY    10  0A54  03      PLAG    10  0997  01      PREP    10  0A5F  03
PRNN    10  0518  05      PRNT    10  0A1F  05      PRR     10  053C  05      PRT     10  06C4  77
ABSY    10  01C0  01      ADER    10  01D0  03      READ    10  01A4  05      SKIP    10  0432  03
SALT    10  0520  01      SORT    10  0226  03      SURY    10  033A  03      SUM2    10  0306  01
SUM3    10  04AA  01      SUM4    10  04FA  01      SPAC    10  042C  03      STAA    10  0627  07
STAR    10  0AA7  01      STAB    10  0AC0  01      STEP    10  0412  03      STOP    10  04E2  05
STOR    10  02A6  05      SUM     10  034E  01      SW      10  06C2  00      SWA     10  073C  00
SWB     10  090A  00      SWC     10  0909  00      TABL    10  0AEE  EF      TBEG    10  0AEC  01
TBLE    10  0A26  C7      THTY    10  CABE  01      TOTI    10  0360  05      TTIM    10  090A  01
TTWO    10  0A22  01      WORK    10  0670  02      WORY    10  0688  01      ZERO    10  0A1C  01
```

```
0154                              CONS    START 340                          001
0154   00L0                               MASA  14,0                         002
0156                                      USING HERE,14                      002
0156   48D0 E8BA                  HERE    LH    13,BASE                      002
1156                                      USING HERE+4096,13                 002
0154   D277 E5AE E560             CONE    MVC   PRT(120),PRT-1               002
0160   0277 E56E E5E7                     MVC   PRT(120),HOR1                002
0166   4DA0 E4DA                          BAS   10,SPAC                      002
016A   4DA0 E4EA                          BAS   10,PRNT                      002
016E   4DA0 E4D6                          BAS   10,SPAC                      002
0172   D277 E5AE E560                     MVC   PRT(120),PRT-1               002
0178   D277 E5AE E6D7                     MVC   PRT(120),HOR2                002
017E   4DA0 E4EA                          BAS   10,PRNT                      002
0182   D277 E56E E560                     MVC   PRT(120),PRT-1               002
0188   D277 E56E E76F                     MVC   PRT(120),HOR3                002
018E   4DA0 E4EA                          BAS   10,PRNT                      003
0192   4DA0 E4D6                          BAS   10,SPAC                      303
0196   9101 EAA3                          TM    SWC,X'01'                    G03
019A   4710 E2F0                          BC    1,OUT                        003
019E   1B88                               SR    8,8                          003
01A0   48B0 E996                          LH    8,TBEG                       003
01A4   D24F E511 E510             READ    MVC   CDIN(80),CDIN-1              003
01AA   D022 E511 0050                     XIO   CDIN(X'22'),80               003
01B0   4780 E0AA                          BC    8,ABSY                       003
01B4   4740 E04E                          BC    4,READ                       003
01B8   9900 0234                          HPR   X'234',0                     003
01BC   47F0 E04E                          BC    15,READ                      003
01C0   9A20 E06A                  ABSY    TIOB  *,X'20'                      003
01C4   9A21 E07A                          TIOB  ADER,X'21'                   004
01C8   9A24 E0C6                          TIOB  LSCO,X'24'                   004
01CC   47F0 E082                          BC    15,FILE                      C04
0100   9900 0999                  ADER    HPR   X'999',0                     004
0104   47F0 E04E                          BC    15,READ                      004
01D8   D20F A000 E511             FILE    MVC   0(16,8),DESC                 C04
01DE   F212 E561 E521                     PACK  STAB(2),STAA(3)              004
01E4   F811 A010 E561                     ZAP   16(2,8),STAB(2)              004
01EA   F212 E563 E574                     PACK  ENDI(2),ENDE(3)              004
01F0   F811 A012 E563                     ZAP   18(2,8),ENDI(2)              004
01F6   F212 E565 E527                     PACK  WORY(2),WORK(3)              004
01FC   F811 A014 E565                     ZAP   20(2,8),WORY(2)              005
0202   F811 A016 E8C6                     ZAP   22(2,8),ZERO                 005
0208   F811 A018 E8C6                     ZAP   24(2,8),ZERO                 005
020E   F811 A01A E8C6                     ZAP   26(2,8),ZERO                 005
0214   4A80 E568                          AH    8,THTY                       005
0218   47F0 E04E                          BC    15,READ                      005
021C   D201 A000 E56A             LSCO    MVC   0(2,8),STAR                  005
0222   4DA0 E2F0                          BAS   10,OUT                       005
0226   9601 E5E6                  SOAT    OI    SWA,X'01'                    005
022A   1B88                               SR    8,8                          005
022C   48B0 E996                          LH    8,TBEG                       005
0230   9400 E56C                          NI    SW,X'00'                     005
0234   F911 A010 802E             LOOP    CP    16(2,8),46(2,8)              006
023A   4700 E0FE                          BC    13,COMP                      006
023E   9601 E56C                          OI    SW,X'01'                     006
0242   D21D E843 8000                     MVC   BIN(30),0(8)                 006
```

Figure CON-01

Construction Program

```
0248     D21D 8000 801E                MVC    0(30,8),30(8)                   006
024E     D21D 801E EA63                MVC    30(30,8),BIN                    006
0254     4A80 E5A8              COMP   AH     8,THIY                          006
0258     D501 801E E56A                CLC    30(2,8),STAR                    006
025E     4770 E00E                     BC     7,LOOP                          006
0262     9101 E5AC                     TM     SW,X'01'                        006
0266     4710 E0D0                     BC     1,SOR1                          006
026A     1B88                          SR     8,8                             006
026C     4880 E49A                     LH     8,TREG                          007
0270     1B99                          SR     9,9                             007
0272     4890 EACC                     LH     9,TTWO                          007
0276     F811 8018 8014                ZAP    24(2,8),20(2,8)                 007
027C     F811 801A EACA                ZAP    26(2,8),ZERO                    007
0282     D201 801C E56A                MVC    28(2,8),STAR                    007
0288     D501 801C E56A         FIND   CLC    28(2,8),STAR                    007
028E     4770 E14E                     BC     7,STOR                          007
0292     4A80 E5A8                     AH     8,THIY                          007
0296     D501 8000 E56A                CLC    0(2,8),STAR                     007
029C     4780 E28C                     BC     8,STEP                          007
02A0     47F0 E132                     BC     15,FIND                         007
02A4     D20D EA81 8010         STOR   MVC    BINB(14),16(8)                  008
02AA     1B88                          SR     8,8                             008
02AC     4880 E49A                     LH     8,TREG                          008
02B0     F911 8012 EA81         FINA   CP     18(2,8),BINB(2)                 008
02B6     4770 E174                     BC     7,FINB                          008
02BA     D20D EA8F 8010                MVC    BINC(14),16(8)                  008
02C0     D20D 9000 EA8F                MVC    0(14,9),BINC                    008
02C6     4A90 EA7E                     AH     9,FTEN                          008
02CA     4A80 E5A8              FINB   AH     8,THIY                          008
02CE     D501 8000 E56A                CLC    0(2,8),STAR                     008
02D4     4770 E15A                     BC     7,FINA                          008
02D8     D201 9000 E56A                MVC    0(2,9),STAR                     008
02DE     1B99                          SR     9,9                             009
02E0     4890 EACC                     LH     9,TTWO                          009
02E4     F811 EA80 9004         ADDA   ZAP    ACCA,4(2,9)                     009
02EA     FA11 EA80 9006                AP     ACCA,6(2,9)                     009
02F0     F811 9008 EA80                ZAP    8(2,9),ACCA                     009
02F6     4A90 EA7E                     AH     9,FTEN                          009
02FA     D501 9000 E56A                CLC    0(2,9),STAR                     009
0300     4770 E18E                     BC     7,ADDA                          009
0304     1B99                   SOR2   SR     9,9                             009
0306     4890 EACC                     LH     9,TTWO                          009
030A     D501 900E E56A                CLC    14(2,9),STAR                    009
0310     4780 E1FA                     BC     8,SUM                           009
0314     9400 EAA2                     NI     SWA,X'00'                       010
0318     F911 9006 9016         LOPA   CP     6(2,9),22(2,9)                  010
031E     47B0 E1E2                     BC     11,SORY                         010
0322     9601 EAA2                     OI     SWA,X'01'                       010
0326     D20D EA8F 9000                MVC    BINC(14),0(9)                   010
032C     D20D 9000 900E                MVC    0(14,9),14(9)                   010
0332     D20D 900E EA8F                MVC    14(14,9),BINC                   010
0338     4A90 EA7E              SORY   AH     9,FTEN                          010
033C     D501 900E E56A                CLC    14(2,9),STAR                    010
0342     4770 E1C2                     BC     7,LOPA                          010
0346     9101 EAA2                     TM     SWA,X'01'                       010
```

Figure CON-02

Construction Program

```
034A   4710 E14E                  BC    1,SUM2                    010
034E   1899              SUM      SR    9,9                       011
0350   4890 EACC                  LH    9,TTWO                    011
0354   F811 E8A6 900A             ZAP   TTIM(2),A(2,9)            011
035A   F811 E8A7 E8A6             ZAP   BINH+6(2),TTIM            011
0360   F811 E8B6 E8A6    TOTI     ZAP   BINE,TTIM                 011
0366   FB11 E8B6 900A             SP    BINE,A(2,9)               011
036C   F811 900A E8B6             ZAP   10(2,9),BINE              011
0372   4A90 E87E                  AH    9,FTEN                    011
0376   D501 9000 E56A             CLC   0(2,9),STAR               011
037C   4770 E20A                  BC    7,TOTI                    011
0380   D200 E8AD E56A             MVC   BINH+12,STAR              011
0386   1AAA                       SR    A,A                       012
0388   48A0 E996                  LH    A,THEG                    012
038C   F911 A010 E861    HOPE     CP    16(2,A),BINB(2)           012
0392   4770 E254                  BC    7,GOON                    012
0396   F911 A012 E863             CP    18(2,A),BINB+2(2)         012
039C   4770 E254                  BC    7,GOON                    012
03A0   D20D A010 E861             MVC   16(14,A),BINB             012
03A6   47F0 E25C                  BC    15,INST                   012
03AA   4AA0 E56A         GOON     AH    A,THTY                    012
03AE   47F0 E236                  BC    15,HOPE                   012
03B2   1899              INST     SR    9,9                       012
03B4   4890 EACC                  LH    9,TTWO                    012
03B8   1AAA              GOGO     SR    A,A                       012
03BA   48A0 E996                  LH    A,THEG                    012
03BE   D20D E861 9000    GOGI     MVC   BINB(14),0(9)             013
03C4   F911 A010 E861             CP    16(2,A),BINB(2)           013
03CA   4770 E2AA                  BC    7,NOGO                    013
03CE   F911 A012 E863             CP    18(2,A),BINB+2(2)         013
03D4   4770 E2AA                  BC    7,NOGO                    013
03D8   D20D A010 E861             MVC   16(14,A),BINB             013
03DE   4A90 E87E                  AH    9,FTEN                    013
03E2   D501 9000 E56A             CLC   0(2,9),STAR               013
03E8   4780 E29A                  BC    8,FINO                    013
03EC   47F0 E262                  BC    15,GOGO                   013
03F0   1AAA              FINO     SR    A,A                       013
03F2   48A0 E996                  LH    A,THEG                    013
03F6   1899                       SR    9,9                       014
03F8   4890 EACC                  LH    9,TTWO                    014
03FC   47F0 E132                  BC    15,FIND                   014
0400   4AA0 E56A         NOGO     AH    A,THTY                    014
0404   D501 A000 E56A             CLC   0(2,A),STAR               014
040A   4780 E28C                  BC    8,STEP                    014
040E   47F0 E268                  BC    15,GOGI                   014
0412   9601 EAA3         STEP     OI    SWC,X'01'                 014
0416   4BA0 E568                  SH    A,THTY                    014
041A   F811 E8A0 A014             ZAP   ACCA,20(2,A)              014
0420   FA11 E8A0 A016             AP    ACCA,22(2,A)              014
0426   F811 A018 E8A0             ZAP   24(2,A),ACCA              014
042C   F811 A01A E8C0             ZAP   26(2,A),ZERO              015
0432   9B45 0001         SKIP     CIO   1(0),X'45'                015
0436   4780 E004                  BC    8,CONE                    015
043A   4740 E2DC                  BC    4,SKIP                    015
043E   9900 0078                  HPR   X'078',0                  015
```

Figure CON-03

Construction Program

```
0442     47F0 E20C                  BC    15,SKIP                    015
0446     1888              OUT      SR    8,8                        015
0448     4880 E996                  LH    8,TREG                     015
044C     D20F E87A 8000    OUTO     MVC   ITEM(16),0(8)              015
0452     D22F E888 E88A             MVC   DATA,MASK                  015
0458     D21B E88A 8010             MVC   DATA(12),16(8)             015
045E     DE2F E888 E88A             ED    DATA,DATR                  015
0464     D277 E5AE E5AD             MVC   PRT(120),PRT-1             016
046A     D277 E5AE E63F             MVC   PRT(120),HOR4              016
0470     4DA0 E4EA                  BAS   10,PRNT                    016
0474     4A80 E5A8                  AH    8,THTY                     016
0478     D501 8000 E56A             CLC   0(2,8),STAR                016
047E     4770 E2FA                  BC    7,OUTO                     016
0482     9101 E51A                  TM    SWA,X'01'                  016
0486     4780 E000                  BC    8,SORT                     016
048A     1888              SOR3     SR    8,8                        016
048C     4880 E996                  LH    8,TREG                     016
0490     9400 E56C                  NI    SW,X'00'                   016
0494     F911 801A 803A    LOPB     CP    26(2,8),58(2,8)            016
049A     47D0 E35E                  BC    13,COMO                    016
049E     9601 E56C                  OI    SW,X'01'                   017
04A2     D21D E843 8000             MVC   BIN(30),0(8)               017
04A8     D21D 8000 801E             MVC   0(30,8),30(8)              017
04AE     D21D 801E E843             MVC   30(30,8),BIN               017
04B4     4A80 E5A8         COMO     AH    8,THTY                     017
04B8     D501 801E E56A             CLC   30(2,8),STAR               017
04BE     4770 E33E                  BC    7,LOPB                     017
04C2     9101 E56C                  TM    SW,X'01'                   017
04C6     4710 E334                  BC    1,SOR3                     017
04CA     1888                       SR    8,8                        017
04CC     4880 E996                  LH    8,TREG                     017
04D0     F911 801A E8C6    COMR     CP    26(2,8),ZERO               017
04D6     4770 E3FC                  BC    7,STOP                     018
04DA     4A80 E5A8                  AH    8,THTY                     018
04DE     47F0 E37A                  BC    15,COMR                    018
04E2     D201 8000 E56A    STOP     MVC   0(2,8),STAR                018
04E8     1888              SOR4     SR    8,8                        018
04EA     4880 E996                  LH    8,TREG                     018
04EE     9400 E56C                  NI    SW,X'00'                   018
04F2     F911 8010 802E    LOPC     CP    16(2,8),46(2,8)            018
04F8     47B0 E3AC                  BC    11,COMS                    018
04FC     9601 E56C                  OI    SW,X'01'                   018
0500     D21D E843 8000             MVC   BIN(30),0(8)               018
0506     D21D 8000 801E             MVC   0(30,8),30(8)              018
050C     D21D 801E E843             MVC   30(30,8),BIN               019
0512     4A80 E5A8         COMS     AH    8,THTY                     019
0516     D501 801E E56A             CLC   30(2,8),STAR               019
051C     4770 E39C                  BC    7,LOPC                     019
0520     9101 E56C                  TM    SW,X'01'                   019
0524     4710 E392                  BC    1,SOR4                     019
0528     9A45 0001         SKIT     CIO   1(0),X'45'                 019
052C     4780 E3E6                  BC    8,PRR                      019
0530     4740 E302                  BC    4,SKIT                     019
0534     9900 0323                  HPR   X'323',0                   019
0538     47F0 E302                  BC     5,SKIT                    019
```

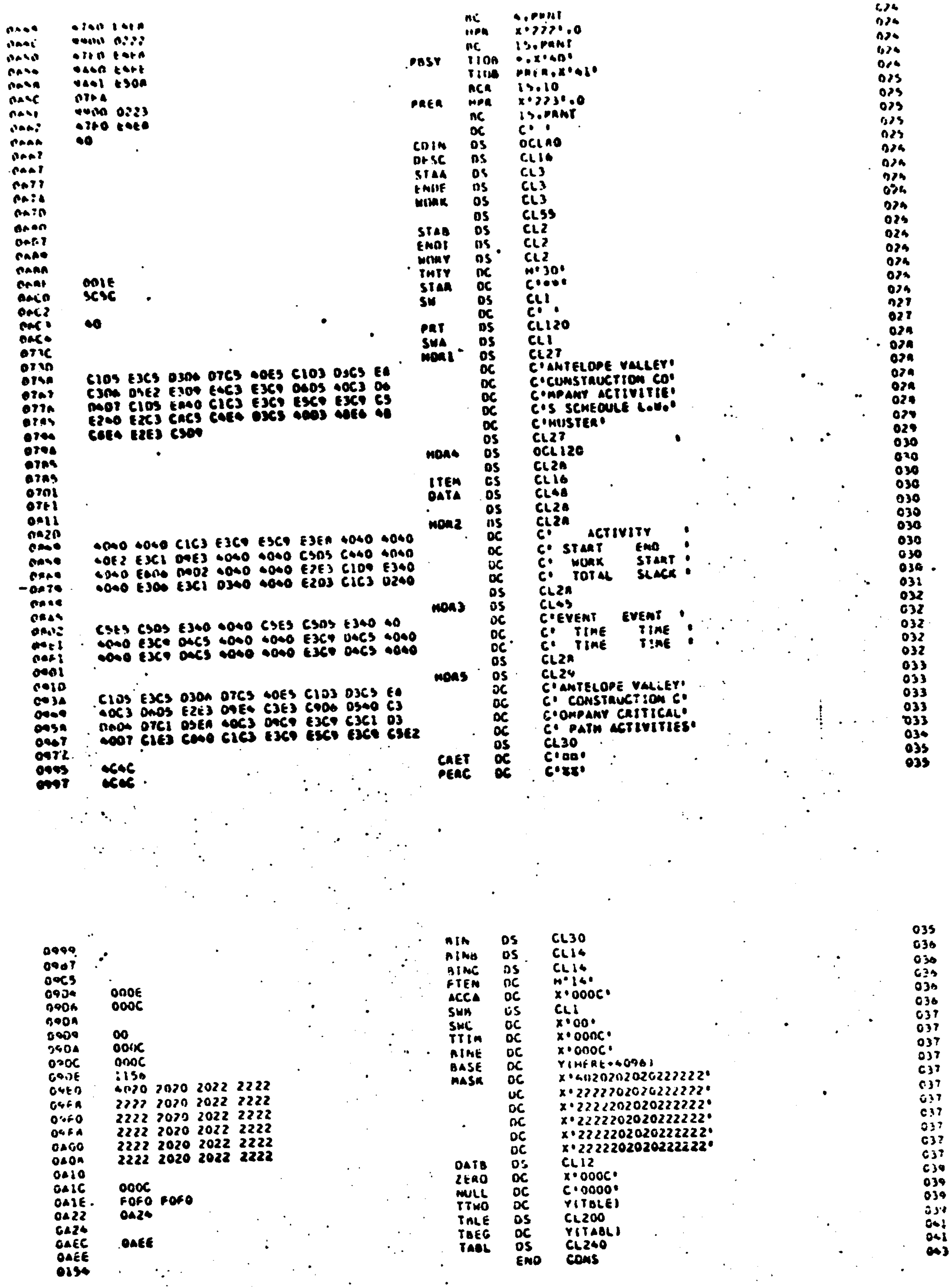
Figure CON-04

Construction Program

```
053C    D277 E56E E56D          PRA    MVC    PRT(120),PRT-1                019
0542    D277 E56E E7C7                 MVC    PRT(120),MOR5                 019
0548    4DA0 E4E8                      BAS    10,PRNT                       020
054C    4DA0 E4D6                      BAS    10,SPAC                       020
0550    D277 E56E E56D                 MVC    PRT(120),PRT-1                020
0556    D277 E56E E6D7                 MVC    PRT(120),MOR2                 020
055C    4DA0 E4E8                      BAS    10,PRNT                       020
0560    D277 E56E E56D                 MVC    PRT(120),PRT-1                020
0566    D277 E56E E74F                 MVC    PRT(120),MOR3                 020
056C    4DA0 E4E8                      BAS    10,PRNT                       020
0570    4DA0 E4D6                      BAS    10,SPAC                       020
0574    1B88                           SR     8,8                           020
0576    4880 E996                      LH     8,TREG                        020
057A    D20F E87A 8000                 MVC    ITEM(16),0(8)                 020
0580    D22F E6AA E88A                 MVC    DATA,MASK                     021
0586    D20B E8AA 8010                 MVC    DATB(12),16(8)                021
058C    DE2F E6AA E8BA                 ED     DATA,DATB                     021
0592    D277 E56E E56D                 MVC    PRT(120),PRT-1                021
0598    D277 E56E E65F                 MVC    PRT(120),MOR4                 021
059E    4DA0 E4E8                      BAS    10,PRNT                       021
05A2    F811 E886 8010          LOPD   ZAP    BINE,16(2,8)                  021
05A8    D201 801C E841                 MVC    28(2,8),PERC                  021
05AE    4A80 E5A8               LOPE   AH     8,THTY                        021
05B2    D501 8000 E56A                 CLC    0(2,8),STAR                   021
05B8    4770 E492                      BC     7,PRNN                        022
05BC    1B88                           SR     8,8                           022
05BE    4880 E996                      LH     8,TREG                        022
05C2    D501 801C E841          COMT   CLC    28(2,8),PERC                  022
05C8    4780 E480                      BC     8,ADVA                        022
05CC    D501 801C E83F                 CLC    28(2,8),CRET                  022
05D2    4780 E44C                      BC     8,LOPD                        022
05D6    4A80 E5A8               ADVA   AH     8,THTY                        022
05DA    D501 8000 E56A                 CLC    0(2,8),STAR                   022
05E0    4780 E4CE                      BC     8,HALT                        022
05E4    47F0 E46C                      BC     15,COMT                       022
05E8    F911 8012 E886          PRNN   CP     18(2,8),BINE                  022
05EE    4770 E458                      BC     7,LOPE                        023
05F2    D20F E87A 8000                 MVC    ITEM(16),0(8)                 023
05F8    D22F E6AA E88A                 MVC    DATA,MASK                     023
05FE    D20B E89A 8010                 MVC    DATB(12),16(8)                023
0604    DE2F E6AA E8BA                 ED     DATA,DATB                     023
060A    D277 E56E E56D                 MVC    PRT(120),PRT-1                023
0610    D277 E56E E65F                 MVC    PRT(120),MOR4                 023
0616    4DA0 E4E8                      BAS    10,PRNT                       023
061A    D201 801C E83F                 MVC    28(2,8),CRET                  023
0620    47F0 E458                      BC     15,LOPE                       023
0624    9900 0444               HALT   HPR    X'444',0                      024
0628    47F0 E4CE                      BC     15,HALT                       024
062C    9B44 0001               SPAC   CIO    1(0),X'44'                    024
0630    07CA                           BCR    8,10                          024
0632    4740 E4D6                      BC     4,SPAC                        024
0636    9909 0111                      HPR    X'111',0                      024
063A    47F0 E4D6                      BC     15,SPAC                       024
063E    D040 E56E 0078          PRNT   XIO    PRT(X'40'),120                024
0644    4780 E4FE                      BC     8,PBSV
```

Figure CON-05

Construction Program

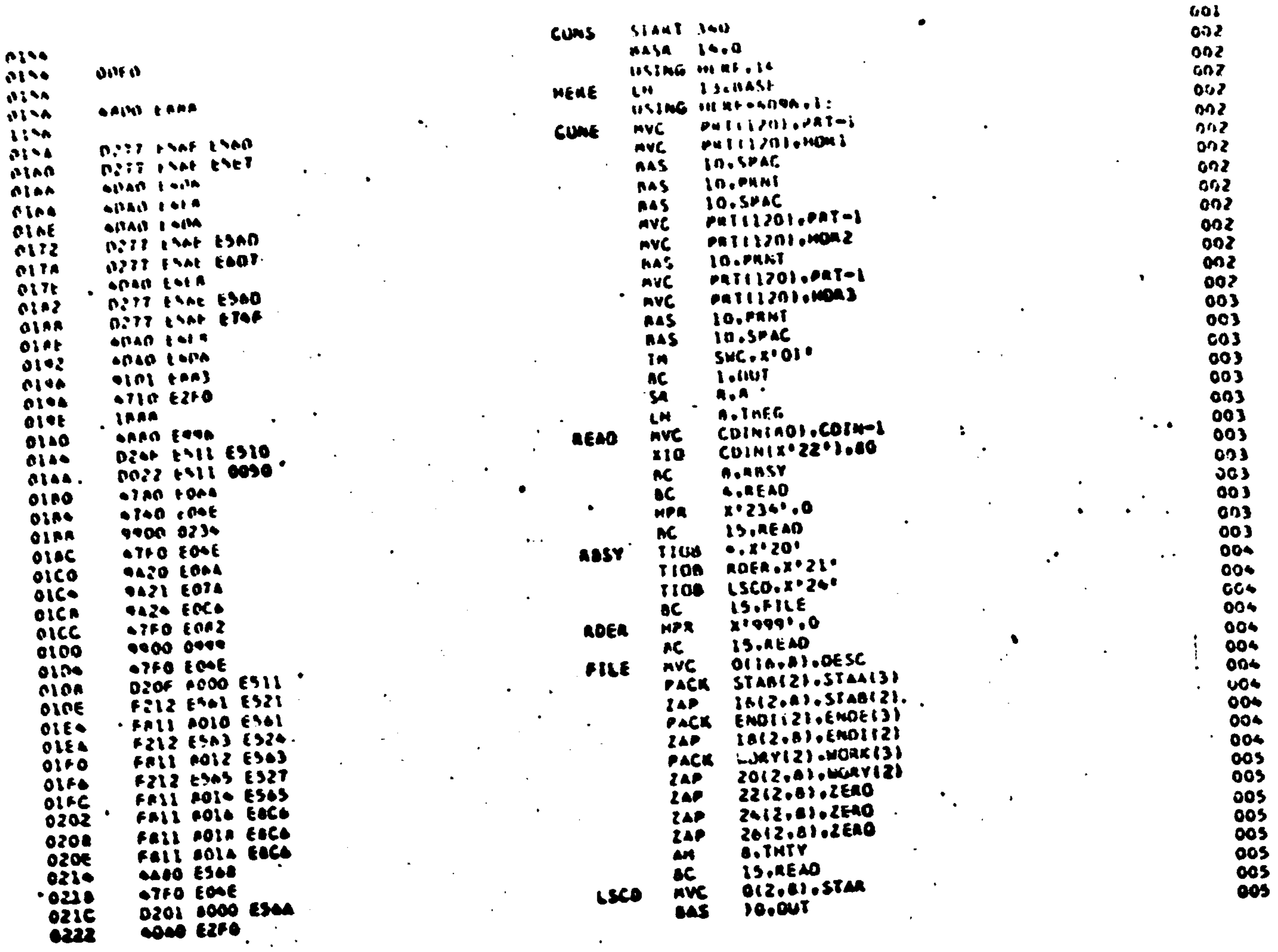


```
                                                             024
0648   4740 E4E8                              MC     4,PRNT                  024
064C   9E00 0222                              HPR    X'222',0                024
0650   47F0 E4EA                              MC     15,PRNT                 024
0654   9D40 E4FE                      PBSY    TIOA   *,X'40'                 024
0658   9D41 E50A                              TIOB   PREA,X'41'              025
065C   07FA                                   RCR    15,10                   025
065E   9E00 0223                      PREA    HPR    X'223',0                025
0662   47F0 E4E8                              MC     15,PRNT                 025
0666   40                                     DC     C' '                    025
0667                                  CDIN    DS     OCL80                   026
0667                                  DESC    DS     CL16                    026
0677                                  STAA    DS     CL3                     026
067A                                  ENDE    DS     CL3                     026
067D                                  WORK    DS     CL3                     026
0680                                          DS     CL55                    026
06B7                                  STAB    DS     CL2                     026
06B9                                  ENDT    DS     CL2                     026
06BB                                  WORY    DS     CL2                     026
06BE   001E                           THTY    DC     H'30'                   026
06C0   5C5C                           STAR    DC     C'**'                   026
06C2                                  SW      DS     CL1                     027
06C3   40                                     DC     C' '                    027
06C4                                  PRT     DS     CL120                   028
073C                                  SWA     DS     CL1                     028
073D                                  HDR1    DS     CL27                    028
0758   C1D5 E3C5 D3D6 D7C5 40E5 C1D3 D3C5 E8          DC     C'ANTELOPE VALLEY'      028
0767   C3D6 D5E2 E3D9 E4C3 E3C9 D6D5 40C3 D6          DC     C'CONSTRUCTION CO'      028
0776   D4D7 C1D5 E840 C1C3 E3C9 E5C9 E3C9 C5          DC     C'MPANY ACTIVITIE'      028
0785   E240 E2C3 C8C5 C4E4 D3C5 40D3 4BE6 4B          DC     C'S SCHEDULE L.W.'      029
0794   C8E4 E2E3 C5D9                                 DC     C'HUSTER'               029
079A                                          DS     CL27                    030
07B5                                  HDR4    DS     OCL120                  030
07B5                                          DS     CL28                    030
07D1                                  ITEM    DS     CL16                    030
07E1                                  DATA    DS     CL48                    030
0811                                          DS     CL28                    030
082D                                  HDR2    DS     CL28                    030
0849   4040 4040 C1C3 E3C9 E5C9 E3E8 4040 4040        DC     C'    ACTIVITY    '     030
0859   40E2 E3C1 D9E3 4040 4040 C5D5 C440 4040        DC     C' START    END   '     030
0869   4040 E6D6 D9D2 4040 4040 E2E3 C1D9 E340        DC     C'  WORK    START '     030
0879   4040 E3D6 E3C1 D340 4040 E2D3 C1C3 D240        DC     C'  TOTAL   SLACK '     031
0889                                          DS     CL28                    032
08A5                                  HDR3    DS     CL45                    032
08D2   C5E5 C5D5 E340 4040 C5E5 C5D5 E340 40          DC     C'EVENT   EVENT '       032
08E1   4040 E3C9 D4C5 4040 4040 E3C9 D4C5 4040        DC     C'  TIME    TIME  '     032
08F1   4040 E3C9 D4C5 4040 4040 E3C9 D4C5 4040        DC     C'  TIME    TIME  '     032
0901                                          DS     CL28                    033
091D                                  HDR5    DS     CL29                    033
093A   C1D5 E3C5 D3D6 D7C5 40E5 C1D3 D3C5 E8          DC     C'ANTELOPE VALLEY'      033
0949   40C3 D6D5 E2E3 D9E4 C3E3 C9D6 D540 C3          DC     C' CONSTRUCTION C'      033
0958   D6D4 D7C1 D5E8 40C3 D9C9 E3C9 C3C1 D3          DC     C'OMPANY CRITICAL'      033
0967   40D7 C1E3 C840 C1C3 E3C9 E5C9 E3C9 C5E2        DC     C' PATH ACTIVITIES'     034
0977                                          DS     CL30                    035
0995   4C4C                           CAET    DC     C'<<'                   035
0997   6C6C                           PERC    DC     C'%%'
```

```
0999                                  RIN     DS     CL30                    035
09B7                                  RINB    DS     CL14                    036
09C5                                  RINC    DS     CL14                    036
09D4   000E                           FTEN    DC     H'14'                   036
09D6   000C                           ACCA    DC     X'000C'                 036
09D8                                  SWH     DS     CL1                     036
09D9   00                             SWC     DC     X'00'                   037
09DA   000C                           TTIM    DC     X'000C'                 037
09DC   000C                           RINE    DC     X'000C'                 037
09DE   1156                           BASE    DC     Y(HERE+4096)            037
09E0   4020 2020 2022 2222            MASK    DC     X'4020202020222222'     037
09E8   2222 2020 2022 2222                    DC     X'2222202020222222'     037
09F0   2222 2020 2022 2222                    DC     X'2222202020222222'     037
09F8   2222 2020 2022 2222                    DC     X'2222202020222222'     037
0A00   2222 2020 2022 2222                    DC     X'2222202020222222'     037
0A08   2222 2020 2022 2222                    DC     X'2222202020222222'     037
0A10                                  DATB    DS     CL12                    037
0A1C   000C                           ZERO    DC     X'000C'                 039
0A1E   F0F0 F0F0                      NULL    DC     C'0000'                 039
0A22   0A24                           TTWO    DC     Y(TBLE)                 039
0A24                                  TALE    DS     CL200                   039
0AEC   0AEE                           TBEG    DC     Y(TABL)                 041
0AEE                                  TABL    DS     CL240                   041
0154                                          END    CONS                    043
```

Figure CON-06

Construction Program--Analysis

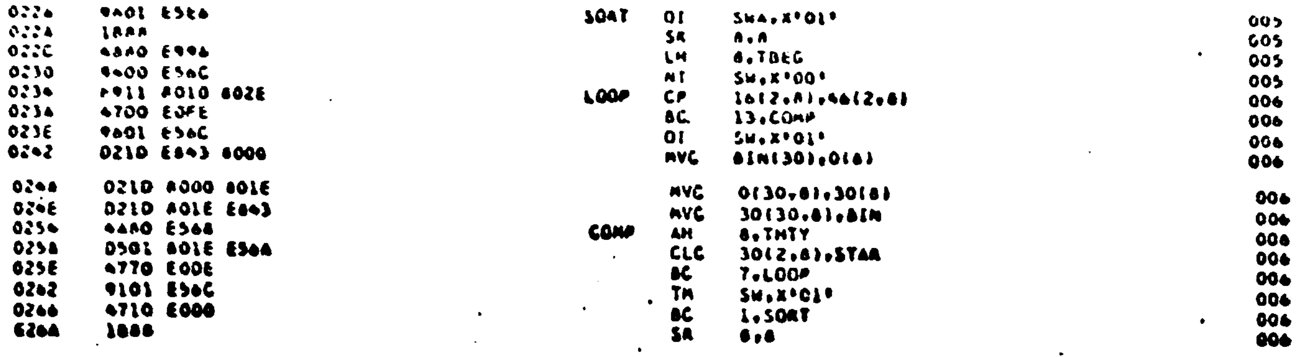


```
                                                                  001
0154                     CONS     START 340                       002
0154    05E0                      BALR  14,0                      002
0156                              USING HERE,14                   002
0156    48D0 E888        HERE     LH    13,BASE                   002
015A                              USING HERE+4096,13              002
015A    D277 E5AF E560   CONE     MVC   PRT(120),PRT-1            002
0160    D277 E5AF E5E7            MVC   PRT(120),HDR1             002
0166    4DA0 E4DA                 BAS   10,SPAC                   002
016A    4DA0 E4E8                 BAS   10,PRNT                   002
016E    4DA0 E4DA                 BAS   10,SPAC                   002
0172    D277 E5AF E5A0            MVC   PRT(120),PRT-1            002
0178    D277 E5AF E607            MVC   PRT(120),HDR2             002
017E    4DA0 E4E8                 BAS   10,PRNT                   002
0182    D277 E5AF E560            MVC   PRT(120),PRT-1            002
0188    D277 E5AF E74F            MVC   PRT(120),HDR3             003
018E    4DA0 E4E8                 BAS   10,PRNT                   003
0192    4DA0 E4DA                 BAS   10,SPAC                   003
0196    9101 E8A3                 TM    SWC,X'01'                 003
019A    4710 E2F0                 BC    1,OUT                     003
019E    1B88                      SR    8,8                       003
01A0    4880 E49A                 LH    8,INEG                    003
01A4    D24F E511 E510   READ     MVC   CDIN(80),CDIN-1           003
01AA    D022 E511 0050            XIO   CDIN(X'22'),80            003
01B0    4780 E06A                 BC    8,ABSY                    003
01B4    4740 E04E                 BC    4,READ                    003
01B8    9900 0234                 MPR   X'234',0                  003
01BC    47F0 E04E                 BC    15,READ                   003
01C0    9A20 E06A        ABSY     TIOB  *,X'20'                   004
01C4    9A21 E07A                 TIOB  RDER,X'21'                004
01C8    9A24 E0C6                 TIOB  LSCO,X'24'                004
01CC    47F0 E082                 BC    15,FILE                   004
01D0    9900 0999        RDER     MPR   X'999',0                  004
01D4    47F0 E04E                 BC    15,READ                   004
01D8    D20F 8000 E511   FILE     MVC   0(16,8),DESC              004
01DE    F212 E561 E521            PACK  STAB(2),STAA(3)           004
01E4    F811 8010 E561            ZAP   16(2,8),STAB(2)           004
01EA    F212 E563 E524            PACK  ENDI(2),ENDE(3)           004
01F0    F811 8012 E563            ZAP   18(2,8),ENDI(2)           004
01F6    F212 E565 E527            PACK  WORY(2),WORK(3)           005
01FC    F811 8014 E565            ZAP   20(2,8),WORY(2)           005
0202    F811 8016 E8C6            ZAP   22(2,8),ZERO              005
0208    F811 8018 E8C6            ZAP   24(2,8),ZERO              005
020E    F811 801A E8C6            ZAP   26(2,8),ZERO              005
0214    4A80 E568                 AH    8,THTY                    005
0218    47F0 E04E                 BC    15,READ                   005
021C    D201 8000 E56A   LSCO     MVC   0(2,8),STAR               005
0222    4DA0 E2F0                 BAS   10,OUT
```

Figure CONA-01

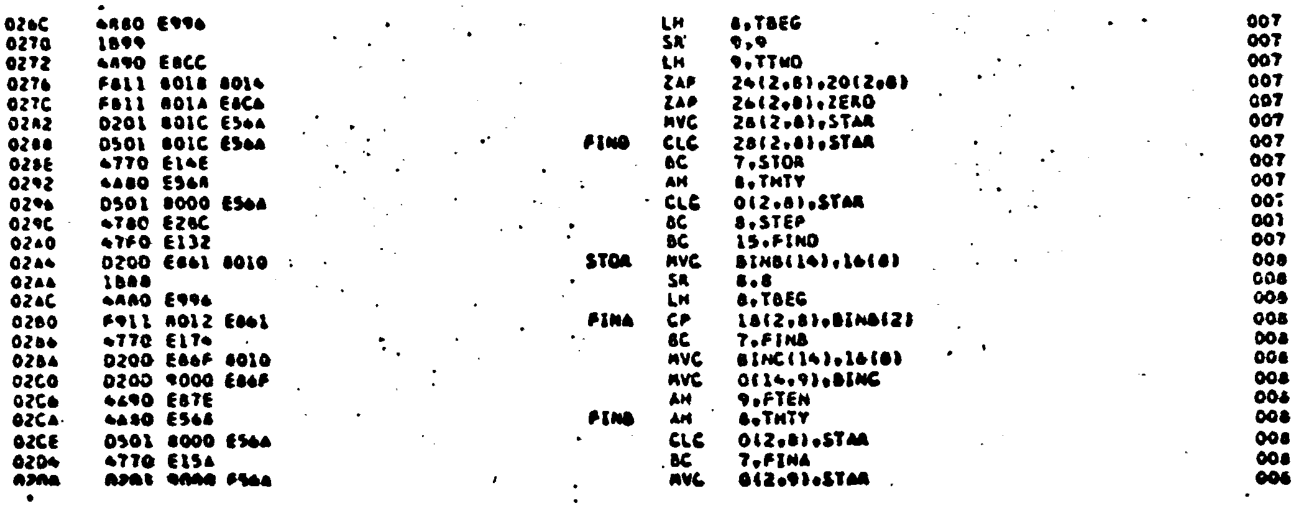
This part of the program does the initialization and reads all the cards. Data is placed into a table in core storage by means of indexing a table. The index register used here is register 8. When all cards are read, a star (*) is placed at the end of the table. The table is then printed out by means of the BAS to OUT after the last card is read.

Construction Program--Analysis



```
0226     9601 E56C                      SORT  OI    SWA,X'01'                     005
022A     1888                                 SR    8,8                           005
022C     4880 E996                            LH    8,TBEG                        005
0230     9400 E56C                            NI    SW,X'00'                      005
0234     F911 8010 802E                 LOOP  CP    16(2,8),46(2,8)               006
023A     47D0 E0FE                            BC    13,COMP                       006
023E     9601 E56C                            OI    SW,X'01'                      006
0242     D21D E843 8000                       MVC   BIN(30),0(8)                  006

0248     D21D 8000 801E                       MVC   0(30,8),30(8)                 006
024E     D21D 801E E843                       MVC   30(30,8),BIN                  006
0254     4A80 E568                      COMP  AH    8,THTY                        006
0258     D501 801E E56A                       CLC   30(2,8),STAR                  006
025E     4770 E0DE                            BC    7,LOOP                        006
0262     9101 E56C                            TM    SW,X'01'                      006
0266     4710 E000                            BC    1,SORT                        006
026A     1888                                 SR    8,8                           006
```

The table is sorted on the basis of the start event number.



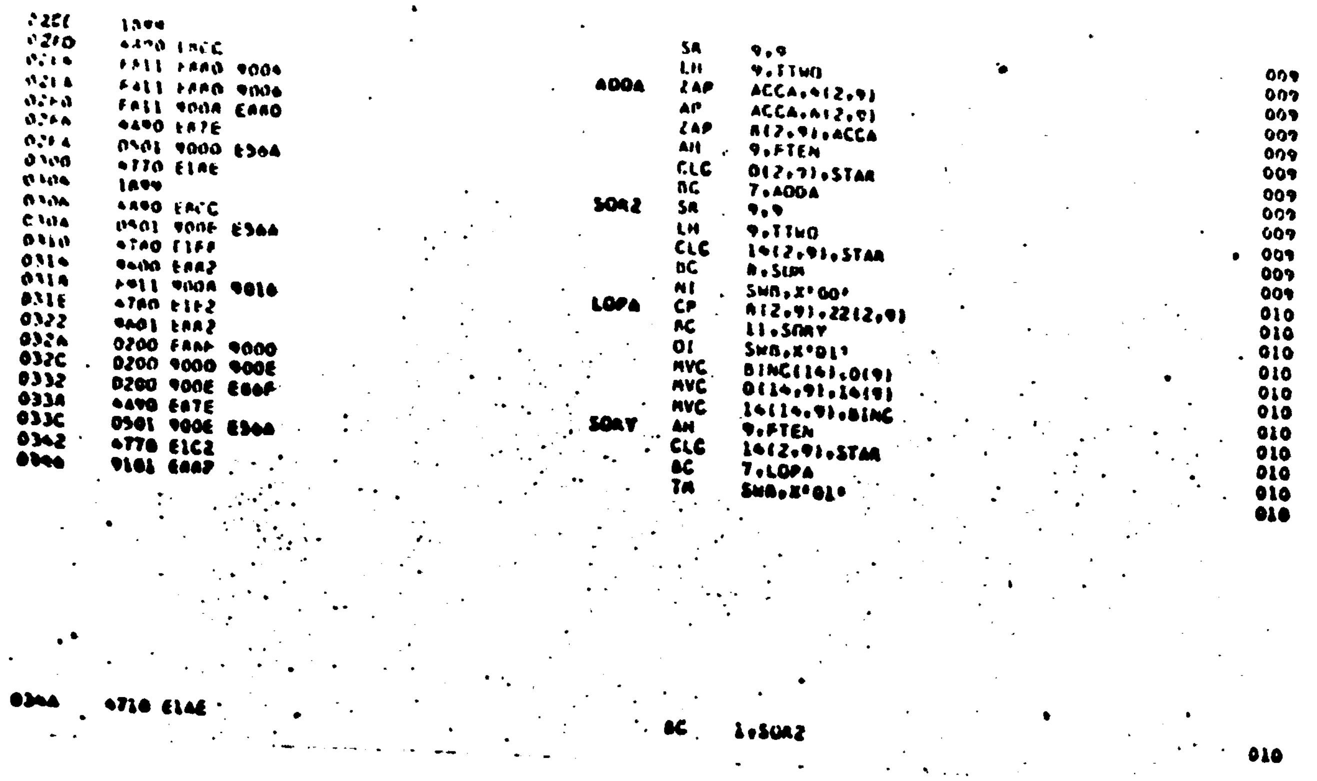
```
026C     4880 E996                            LH    8,TBEG                        007
0270     1899                                 SR    9,9                           007
0272     4890 E8CC                            LH    9,TTWO                        007
0276     F811 8018 8014                       ZAP   24(2,8),20(2,8)               007
027C     F811 801A E8CA                       ZAP   26(2,8),ZERO                  007
0282     D201 801C E56A                       MVC   28(2,8),STAR                  007
0288     D501 801C E56A                 FIND  CLC   28(2,8),STAR                  007
028E     4770 E14E                            BC    7,STOR                        007
0292     4A80 E568                            AH    8,THTY                        007
0296     D501 8000 E56A                       CLC   0(2,8),STAR                   007
029C     4780 E28C                            BC    8,STEP                        007
02A0     47F0 E132                            BC    15,FIND                       007
02A4     D20D E861 8010                 STOR  MVC   BINB(14),16(8)                008
02AA     1888                                 SR    8,8                           008
02AC     4880 E996                            LH    8,TBEG                        008
02B0     F911 8012 E861                 FINA  CP    18(2,8),BINB(2)               008
02B6     4770 E174                            BC    7,FINB                        008
02BA     D20D E86F 8010                       MVC   BINC(14),16(8)                008
02C0     D20D 9000 E86F                       MVC   0(14,9),BINC                  008
02C6     4A90 E87E                            AH    9,FTEN                        008
02CA     4A80 E568                      FINB  AH    8,THTY                        008
02CE     D501 8000 E56A                       CLC   0(2,8),STAR                   008
02D4     4770 E15A                            BC    7,FINA                        008
02D8     D201 9000 E56A                       MVC   0(2,9),STAR                   008
```

Figure CONA-02

FIND--A compare is made to determine if it is the end of the table. If not, the starting time is moved to BINB. The end event of the table is compared to the start event in BINB. If they are equal, the start event is moved to BINC and then BINC is moved to TTWO.

The registers are incremented and a compare is made for the star. If no star has been reached, a branch is made back to FINA. When a star is found for the end of the table, indexed by register 8, a star is placed in TTWO which is indexed by register 9.

Construction Program--Analysis

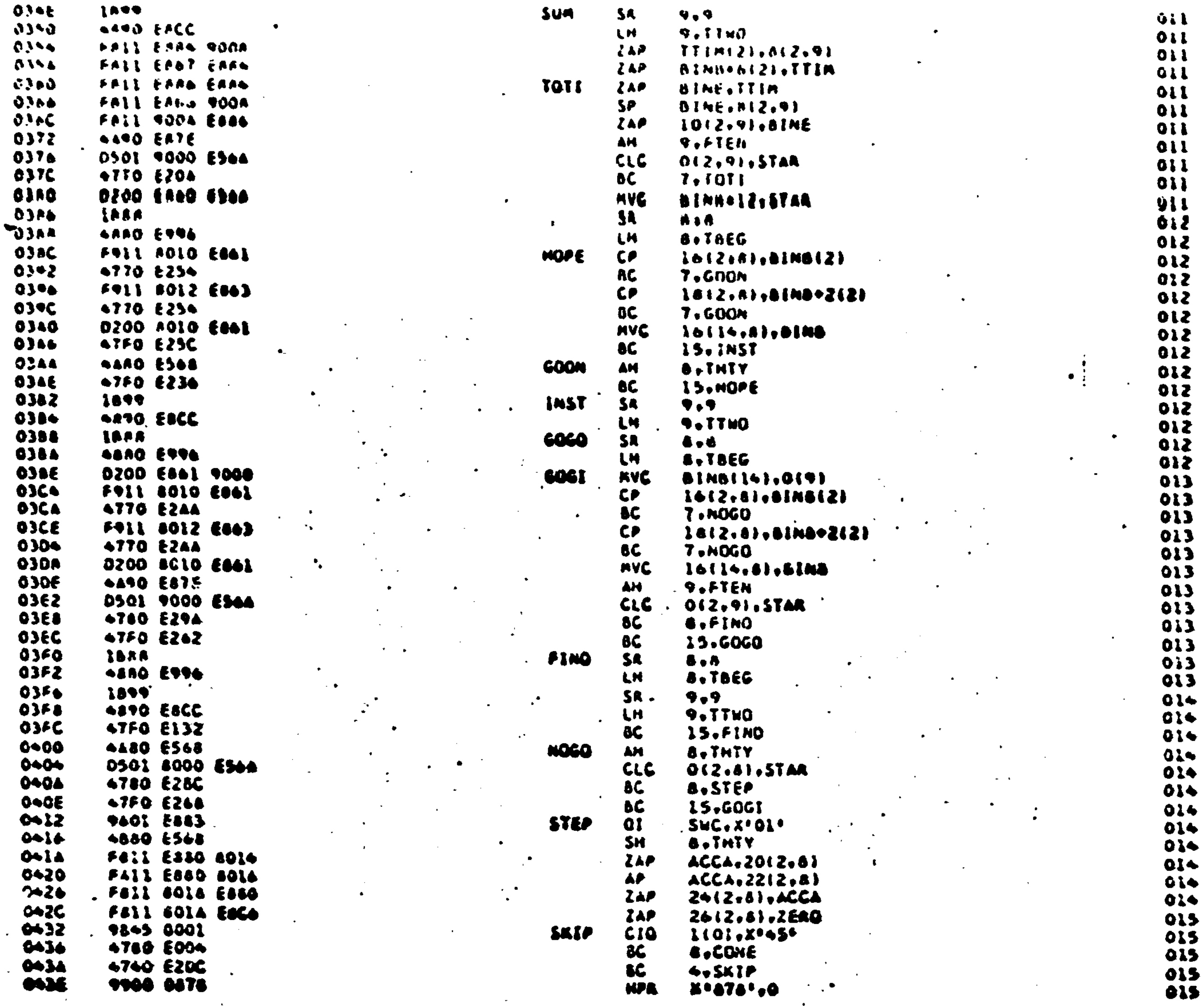


```
02EE     1399                           SA    9,9
02F0     4890 E8CC                      LH    9,TTWO                                  009
02F4     F811 E8A0 9004        ADDA     ZAP   ACCA,4(2,9)                             009
02FA     FA11 E8A0 9006                 AP    ACCA,6(2,9)                             009
0300     F811 9004 E8A0                 ZAP   4(2,9),ACCA                             009
0306     4A90 E87E                      AH    9,FTEN                                  009
030A     D501 9000 E56A                 CLC   0(2,9),STAR                             009
0300     4770 E1AE                      BC    7,ADDA                                  009
0306     1A99                  SOR2     SR    9,9                                     009
030A     4890 E8CC                      LH    9,TTWO                                  009
C30A     D501 900E E56A                 CLC   14(2,9),STAR                            009
0310     47A0 E1FF                      BC    8,SUM                                   009
0314     9400 E8A2                      NI    SWB,X'00'                               009
0318     F911 900A 9016        LOPA     CP    8(2,9),22(2,9)                          010
031E     47B0 E1F2                      BC    11,SORY                                 010
0322     9601 E8A2                      OI    SWB,X'01'                               010
032A     D200 E8AF 9000                 MVC   BINC(14),0(9)                           010
032C     D200 9000 900E                 MVC   0(14,9),14(9)                           010
0332     D200 900E E8AF                 MVC   14(14,9),BINC                           010
033A     4A90 E87E             SORY     AH    9,FTEN                                  010
033C     D501 900E E56A                 CLC   14(2,9),STAR                            010
0342     4770 E1C2                      BC    7,LOPA                                  010
0346     9101 E8A2                      TM    SWB,X'01'                               010
                                                                                      010

034A     4710 E1AE                      BC    1,SOR2                                  010
```

Figure CONA-03

ADDA--The work time is added to each start time and then placed back into TTWO as the new end time.

SOR2--Sorts the secondary table TTWO on the basis of the end events.



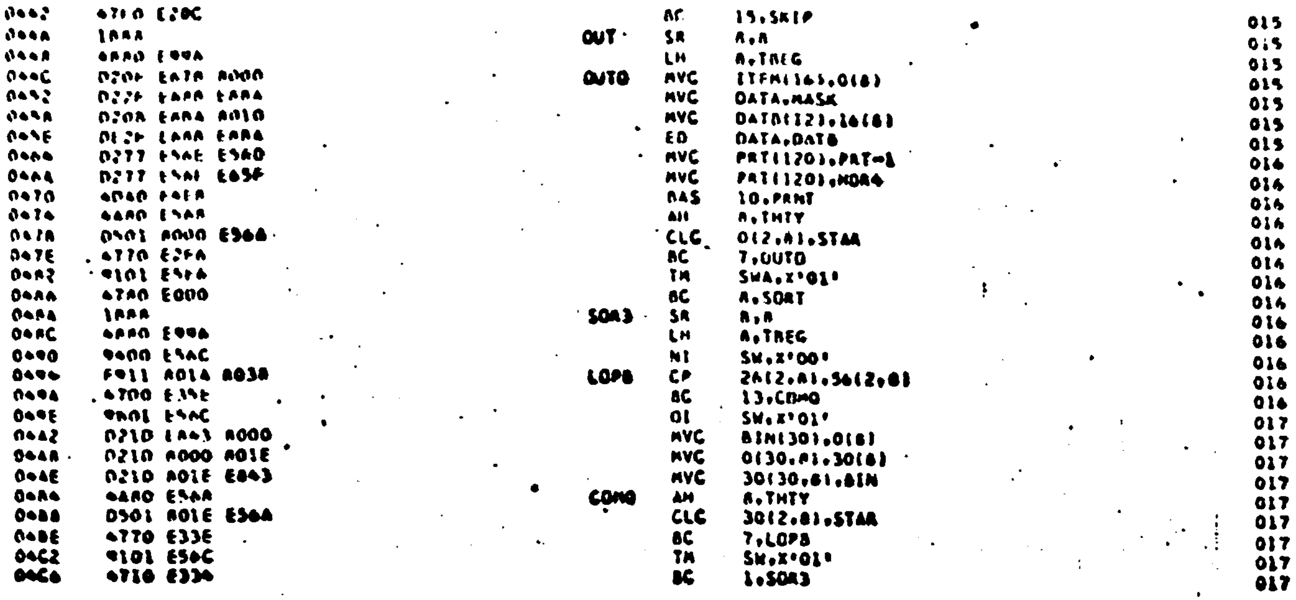
```
034E   1B99                     SUM    SR     9,9                    011
0350   4890 E8CC                       LH     9,TTWO                 011
0354   F811 E8A4 900A                  ZAP    TTIM(2),A(2,9)         011
035A   F811 E867 E8A4                  ZAP    BINB+6(2),TTIM         011
0360   F811 E8A6 E8A4           TOTI   ZAP    BINE,TTIM              011
0366   FB11 E8A6 900A                  SP     BINE,A(2,9)            011
036C   F811 900A E8A6                  ZAP    10(2,9),BINE           011
0372   4A90 E87E                       AH     9,FTEN                 011
0376   D501 9000 E56A                  CLC    0(2,9),STAR            011
037C   4770 E20A                       BC     7,TOTI                 011
0380   D201 E86D E56A                  MVC    BINB+12,STAR           011
0386   1B88                            SR     8,8                    012
0388   4880 E996                       LH     8,TBEG                 012
038C   F911 8010 E861           HOPE   CP     16(2,8),BINB(2)        012
0392   4770 E254                       BC     7,GOON                 012
0396   F911 8012 E863                  CP     18(2,8),BINB+2(2)      012
039C   4770 E254                       BC     7,GOON                 012
03A0   D20D 8010 E861                  MVC    16(14,8),BINB          012
03A6   47F0 E25C                       BC     15,INST                012
03AA   4A80 E568                GOON   AH     8,THTY                 012
03AE   47F0 E236                       BC     15,HOPE                012
03B2   1B99                     INST   SR     9,9                    012
03B4   4890 E8CC                       LH     9,TTWO                 012
03B8   1B88                     GOGO   SR     8,8                    012
03BA   4880 E996                       LH     8,TBEG                 012
03BE   D20D E861 9000           GOGI   MVC    BINB(14),0(9)          013
03C4   F911 8010 E861                  CP     16(2,8),BINB(2)        013
03CA   4770 E2AA                       BC     7,NOGO                 013
03CE   F911 8012 E863                  CP     18(2,8),BINB+2(2)      013
03D4   4770 E2AA                       BC     7,NOGO                 013
03D8   D20D 8010 E861                  MVC    16(14,8),BINB          013
03DE   4A90 E87E                       AH     9,FTEN                 013
03E2   D501 9000 E56A                  CLC    0(2,9),STAR            013
03E8   4780 E29A                       BC     8,FINO                 013
03EC   47F0 E262                       BC     15,GOGO                013
03F0   1B88                     FINO   SR     8,8                    013
03F2   4880 E996                       LH     8,TBEG                 013
03F6   1B99                            SR     9,9                    014
03F8   4890 E8CC                       LH     9,TTWO                 014
03FC   47F0 E132                       BC     15,FIND                014
0400   4A80 E568                NOGO   AH     8,THTY                 014
0404   D501 8000 E56A                  CLC    0(2,8),STAR            014
040A   4780 E2BC                       BC     8,STEP                 014
040E   47F0 E268                       BC     15,GOGI                014
0412   9601 E883                STEP   OI     SWC,X'01'              014
0416   4B80 E568                       SH     8,THTY                 014
041A   F811 E880 8014                  ZAP    ACCA,20(2,8)           014
0420   FA11 E880 8016                  AP     ACCA,22(2,8)           014
0426   F811 8018 E880                  ZAP    24(2,8),ACCA           014
042C   F811 801A E8C6                  ZAP    26(2,8),ZERO           015
0432   9545 0001                SKIP   CLI    1(0),X'45'             015
0436   4780 E004                       BC     8,CONE                 015
043A   4740 E2DC                       BC     4,SKIP                 015
043E   9900 0878                       HPR    X'878',0               015
```

Figure CONA-04

SUM--Subtract ending time of each activity from the ending time of the longest activity in TTWO. Enter the difference, which is the slack time, in the slack time position. The index register is then reset and the beginning of the table is again loaded into register 8. HOPE--A compare is made with BINB (the beginning time and the end time). If they match, we return it to the main table.

INST--Transfer table TTWO back to the main table. If it is the end of the main table a zero is zapped into slack time here because there is no slack time in the last event. Skip to the next page and print out the headings and the table. This time the switch at the end of section CONE send us to OUT for the next sort and print.

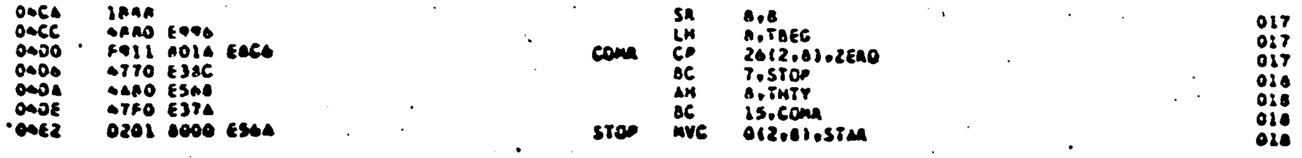
Construction Program--Analysis



```
0442   47F0 E20C                    BC     15,SKIP              015
0446   1B88               OUT       SR     8,8                  015
0448   4880 E99A                    LH     8,TREG               015
044C   D20F EA7A 8000     OUTO      MVC    ITEM(16),0(8)        015
0452   D22F EA8A EA8A               MVC    DATA,MASK            015
0458   D20B EABA 8010               MVC    DATB(12),16(8)       015
045E   DE2F EA8A EABA               ED     DATA,DATB            015
0464   D277 E5AE E5AD               MVC    PRT(120),PRT-1       016
046A   D277 E5AE E65F               MVC    PRT(120),HDR4        016
0470   4DA0 F4EA                    BAS    10,PRNT              016
0474   4A80 E5A8                    AH     8,THTY               016
0478   D501 8000 E56A               CLC    0(2,8),STAR          016
047E   4770 E2FA                    BC     7,OUTO               016
0482   9101 E5FA                    TM     SWA,X'01'            016
0486   4780 E000                    BC     8,SORT               016
048A   1B88               SOR3      SR     8,8                  016
048C   4880 E99A                    LH     8,TREG               016
0490   9400 E5AC                    NI     SW,X'00'             016
0494   F911 801A 8038     LOPB      CP     26(2,8),56(2,8)      016
049A   47D0 E35E                    BC     13,COMO              016
049E   9601 E5AC                    OI     SW,X'01'             017
04A2   D21D EA43 8000               MVC    BIN(30),0(8)         017
04A8   D21D 8000 801E               MVC    0(30,8),30(8)        017
04AE   D21D 801E EA43               MVC    30(30,8),BIN         017
04B4   4A80 E5A8          COMO      AH     8,THTY               017
04B8   D501 801E E56A               CLC    30(2,8),STAR         017
04BE   4770 E33E                    BC     7,LOPB               017
04C2   9101 E5AC                    TM     SW,X'01'             017
04C6   4710 E336                    BC     1,SOR3               017
```

Figure CONA-05

The table is sorted on the basis of the lowest slack time first.

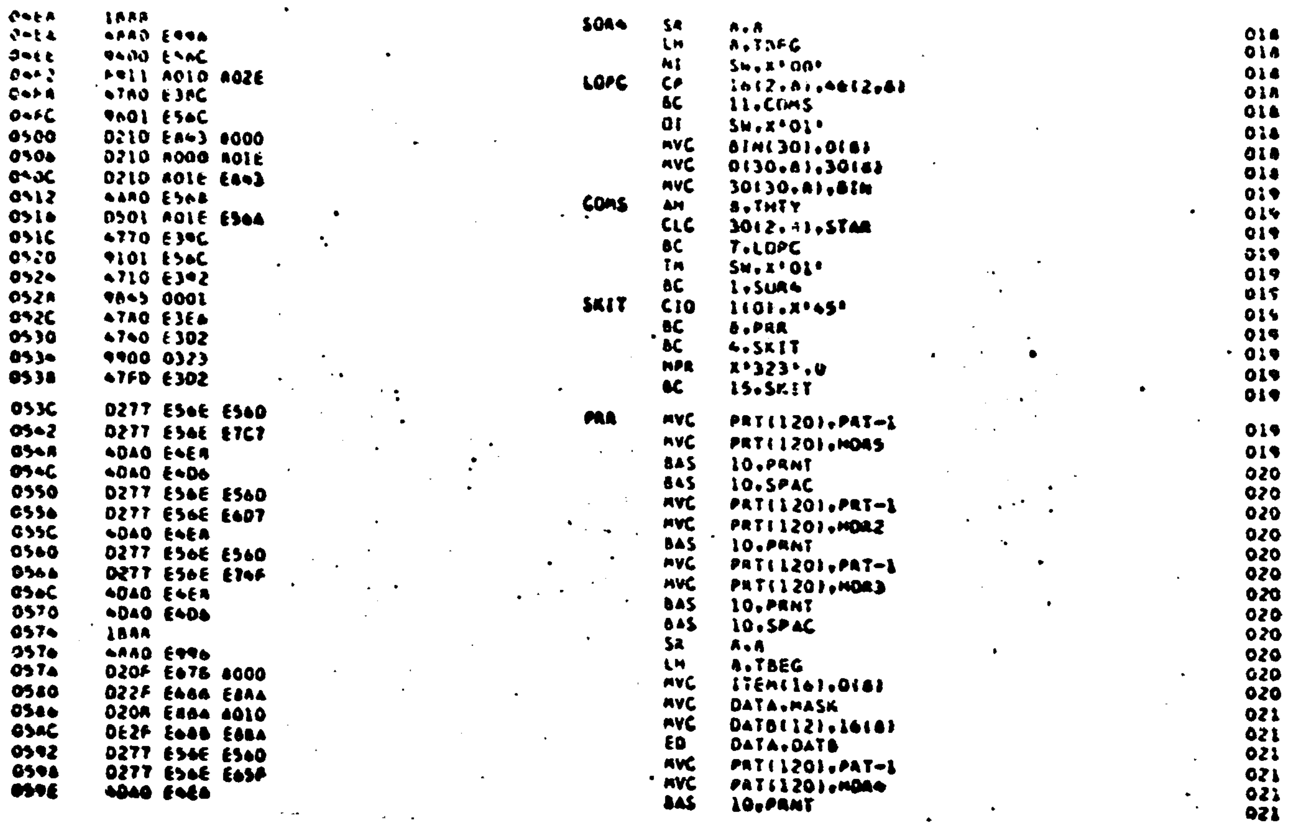


```
04CA   1B88                         SR     8,8                  017
04CC   4880 E996                    LH     8,TBEG               017
04D0   F911 801A E8C6     CONA      CP     26(2,8),ZERO         017
04D6   4770 E38C                    BC     7,STOP               018
04DA   4A80 E5A8                    AH     8,THTY               018
04DE   47F0 E37A                    BC     15,CONA              018
04E2   D201 8000 E56A     STOP      MVC    0(2,8),STAR          018
```

Figure CONA-05 (Part-2)

COMR--A star is placed at the point where the zero slack time ends.

Construction Program--Analysis



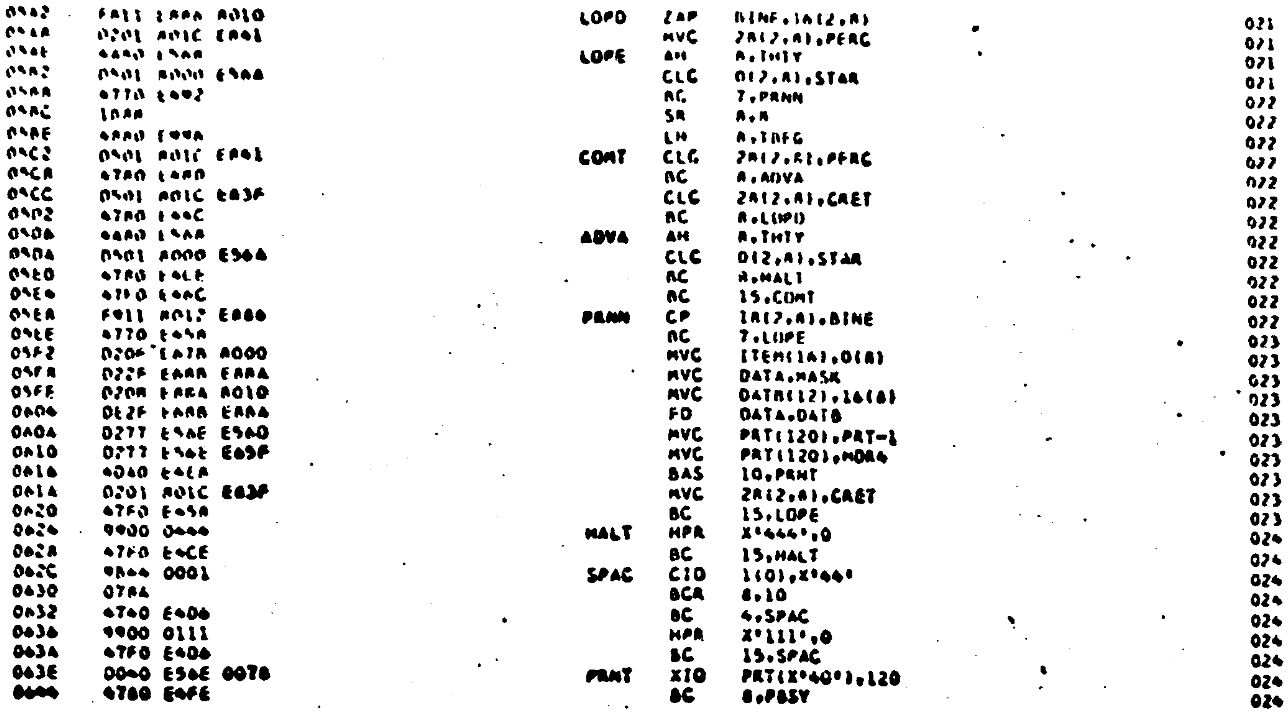
```
04E8   18AA                 SOR4   SR    A,A                      018
04EA   48A0 E99A                   LH    A,TBEG                   018
04EE   9400 E56C                   NI    SW,X'00'                 018
04F2   F911 A010 A02E       LOPC   CP    16(2,A),46(2,A)          018
04F8   47A0 E3AC                   BC    11,COMS                  018
04FC   9601 E56C                   OI    SW,X'01'                 018
0500   D210 E843 8000              MVC   BIN(30),0(A)             018
0506   D210 A000 A01E              MVC   0(30,A),30(A)            018
050C   D210 A01E E843              MVC   30(30,A),BIN             019
0512   4AA0 E568            COMS   AH    A,TMTY                   019
0516   D501 A01E E564              CLC   30(2,A),STAR             019
051C   4770 E39C                   BC    7,LOPC                   019
0520   9101 E56C                   TM    SW,X'01'                 019
0524   4710 E392                   BC    1,SOR4                   019
0528   9A45 0001            SKIT   CIO   1(0),X'45'               019
052C   47A0 E3E6                   BC    8,PRR                    019
0530   4740 E302                   BC    4,SKIT                   019
0534   9900 0323                   NPR   X'323',0                 019
0538   47F0 E302                   BC    15,SKIT                  019

053C   D277 E56E E560       PRR    MVC   PRT(120),PRT-1           019
0542   D277 E56E E7C7              MVC   PRT(120),MORS            019
0548   40A0 E4EA                   BAS   10,PRNT                  020
054C   40A0 E4D6                   BAS   10,SPAC                  020
0550   D277 E56E E560              MVC   PRT(120),PRT-1           020
0556   D277 E56E E6D7              MVC   PRT(120),MOR2            020
055C   40A0 E4EA                   BAS   10,PRNT                  020
0560   D277 E56E E560              MVC   PRT(120),PRT-1           020
0566   D277 E56E E74F              MVC   PRT(120),MOR3            020
056C   40A0 E4EA                   BAS   10,PRNT                  020
0570   40A0 E4D6                   BAS   10,SPAC                  020
0574   18AA                        SR    A,A                      020
0576   48A0 E996                   LH    A,TBEG                   020
057A   D20F E676 8000              MVC   ITEM(16),0(A)            020
0580   D22F E6AA E8AA              MVC   DATA,MASK                021
0586   D20B E8B4 A010              MVC   DATB(12),16(A)           021
058C   DE2F E6AB E8BA              ED    DATA,DATB                021
0592   D277 E56E E560              MVC   PRT(120),PRT-1           021
0598   D277 E56E E65F              MVC   PRT(120),MOR4            021
059E   40A0 E4EA                   BAS   10,PRNT                  021
```

Figure CONA-06

SOR4--The new short table is sorted on the basis of the highest start event number first. This table will contain all the events with no slack time. This means we start at the back of the critical path drawing.

SKIT--Skip to a new page, print out headers, and space. The address of the beginning of the main table is loaded into register 8. The first activity printed out here is actually the last activity of the drawing.

Construction Program--Analysis



```
05A2    F811 E8A8 A010              LOPD  ZAP   BINE,1A(2,A)                    021
05A8    D201 A01C E8A1                    MVC   2A(2,A),PERC                    021
05AE    4AA0 E5A8                   LOPE  AH    A,THTY                          021
05B2    D501 A000 E5AA                    CLC   0(2,A),STAR                     021
05B8    4770 E492                         BC    7,PRNN                          022
05BC    1BAA                              SR    A,A                             022
05BE    48A0 E99A                         LH    A,TDFG                          022
05C2    D501 A01C E8A1              COMT  CLC   2A(2,A),PERC                    022
05C8    4780 E4A0                         BC    8,ADVA                          022
05CC    D501 A01C E83F                    CLC   2A(2,A),CRET                    022
05D2    4780 E44C                         BC    8,LOPD                          022
05D6    4AA0 E5AA                   ADVA  AH    A,THTY                          022
05DA    D501 A000 E56A                    CLC   0(2,A),STAR                     022
05E0    4780 E4CE                         BC    8,HALT                          022
05E4    47F0 E4AC                         BC    15,COMT                         022
05EA    F911 A012 E880              PRNN  CP    1A(2,A),BINE                    022
05EE    4770 E45A                         BC    7,LOPE                          023
05F2    D206 E47A A000                    MVC   ITEM(1A),0(A)                   023
05F8    D22F E8A8 E8AA                    MVC   DATA,MASK                       023
05FE    D20B E8EA A010                    MVC   DATB(12),16(A)                  023
0604    DE2F E8A8 E8AA                    FD    DATA,DATB                       023
060A    D277 E5AE E5A0                    MVC   PRT(120),PRT-1                  023
0610    D277 E5AE E65F                    MVC   PRT(120),MOR4                   023
0616    4DA0 E4EA                         BAS   10,PRNT                         023
061A    D201 A01C E83F                    MVC   2A(2,A),CRET                    023
0620    47F0 E45A                         BC    15,LOPE                         023
0624    9900 0444                   HALT  HPR   X'444',0                        024
0628    47F0 E4CE                         BC    15,HALT                         024
062C    9A44 0001                   SPAC  CIO   1(0),X'44'                      024
0630    078A                              BCR   8,10                            024
0632    4740 E4D6                         BC    4,SPAC                          024
0636    9900 0111                         HPR   X'111',0                        024
063A    47F0 E4D6                         BC    15,SPAC                         024
063E    0040 E5AE 0078              PRNT  XIO   PRT(X'40'),120                  024
0644    4780 E4FE                         BC    8,PBSY                          024
```

Figure CONA-07

LOPD--Index the first activity with a percent sign and add 30 to the main table. Reset the index register and check for a percent sign. If so, jump to ADVA; if not, check for a cret. If it is a cret, jump back to LOPD. This process now replaces each cret with a percent sign. The percent tells us how far we have gone, and the cret tells us the next adjacent event along the path that has no slack time.

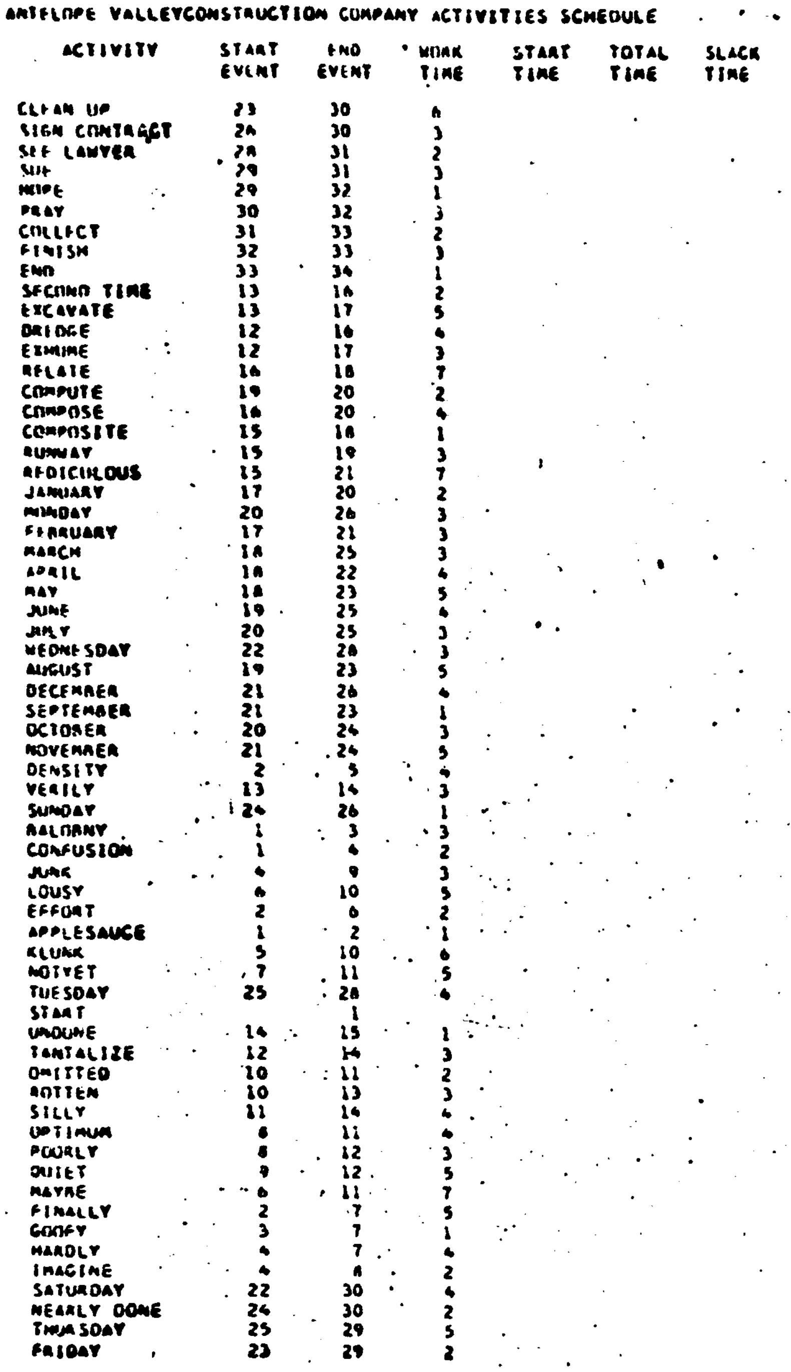
Construction Program--Analysis

```
0A48    4740 E4EA                                    BC    4,PRNT                      024
0A4C    9900 0222                                    HPR   X'222',0                    024
0A50    47F0 E4EA                                    BC    15,PRNT                     024
0A54    9A40 E4FE                             PBSY   TIOB  *,X'40'                     024
0A58    9A41 E306                                    TIOB  PRER,X'41'                  024
0A5C    07FA                                         BCR   15,10                       025
0A5E    9900 0223                             PRER   HPR   X'223',0                    025
0A62    47F0 E4E0                                    BC    15,PRNT                     025
0A66    40                                           DC    C' '                        025
0A67                                          CDIN   DS    0CL80                       025
0A67                                          DESC   DS    CL16                        026
0A77                                          STAA   DS    CL3                         026
0A7A                                          ENDE   DS    CL3                         026
0A7D                                          WORK   DS    CL3                         026
0A80                                                 DS    CL55                        026
0AB7                                          STAB   DS    CL2                         026
0AB9                                          ENDI   DS    CL2                         026
0ABB                                          WORY   DS    CL2                         026
0ABE    001E                                  THTY   DC    H'30'                       026
0AC0    5C5C                                  STAR   DC    C'**'                       026
0AC2                                          SW     DS    CL1                         026
0AC3    40                                           DC    C' '                        027
0AC4                                          PRT    DS    CL120                       027
0B3C                                          SWA    DS    CL1                         028
0B3D                                          WOR1   DS    CL27                        028
0B58    C1D5 E3C5 D3D6 D7C5 40E5 C1D3 D3C5 E8        DC    C'ANTELOPE VALLEY'          028
0B67    C3D6 D5E2 E3D9 E4C3 E3C9 D6D5 40C3 D6        DC    C'CONSTRUCTION CO'          028
0B76    D4D7 C1D5 E840 C1C3 E3C9 E5C9 E3C9 C5        DC    C'MPANY ACTIVITIE'          028
0B85    E240 E2C3 C8C5 C4E4 D3C5 40D3 4BE6 4B        DC    C'S SCHEDULE L.W.'          028
0B94    C8E4 E2E3 C5D9                               DC    C'HUSTER'                   029
0B9A                                                 DS    CL27                        029
0BB5                                          WOR4   DS    0CL120                      030
0BB5                                                 DS    CL28                        030
0BD1                                          ITEM   DS    CL16                        030
0BE1                                          DATA   DS    CL48                        030
0C11                                                 DS    CL28                        030
0C2D                                          WOR2   DS    CL28                        030
0C49    4040 4040 C1C3 E3C9 E5C9 E3E8 4040 4040      DC    C'    ACTIVITY    '         030
0C59    40E2 E3C1 D9E3 4040 4040 C5D5 C440 4040      DC    C' START    END   '         030
0C69    4040 E6D6 D9D2 4040 4040 E2E3 C1D9 E340      DC    C'  WORK    START '         030
0C79    4040 E3D6 E3C1 D340 4040 E2D3 C1C3 D240      DC    C'  TOTAL   SLACK '         030
0C89                                                 DS    CL28                        031
0CA5                                          WOR3   DS    CL45                        032
0CD2    C5E5 C5D5 E340 4040 C5E5 C5D5 E340 40        DC    C'EVENT   EVENT '           032
0CE1    4040 E3C9 D4C5 4040 4040 E3C9 D4C5 4040      DC    C'  TIME    TIME  '         032
0CF1    4040 E3C9 D4C5 4040 4040 E3C9 D4C5 4040      DC    C'  TIME    TIME  '         032
0D01                                                 DS    CL28                        032
0D1D                                          WOR5   DS    CL29                        033
0D3A    C1D5 E3C5 D3D6 D7C5 40E5 C1D3 D3C5 E8        DC    C'ANTELOPE VALLEY'          033
0D49    40C3 D6D5 E2E3 D9E4 C3E3 C9D6 D540 C3        DC    C' CONSTRUCTION C'          033
0D58    D6D4 D7C1 D5E8 40C3 D9C9 E3C9 C3C1 D3        DC    C'OMPANY CRITICAL'          033
0D67    40D7 C1E3 C840 C1C3 E3C9 E5C9 E3C9 C5E2      DC    C' PATH ACTIVITIES'         033
0D77                                                 DS    CL30                        034
0D95    6C6C                                  CAET   DC    C'%%'                       035
0D97    6C6C                                  PERC   DC    C'%%'                       035

0D99                                          RIN    DS    CL30                        035
0DB7                                          RINH   DS    CL14                        036
0DC5                                          HINC   DS    CL14                        036
0DD4    000E                                  FTEN   DC    H'14'                       036
0DD6    000C                                  ACCA   DC    X'000C'                     036
0DD8                                          SWH    DS    CL1                         036
0DD9    00                                    SWC    DC    X'00'                       037
0DDA    000C                                  TTIM   DC    X'000C'                     037
0DDC    000C                                  RINE   DC    X'000C'                     037
0DDE    119A                                  BASE   DC    Y(HERE+4096)                037
0DE0    4020 2020 2022 2222                   MASK   DC    X'4020202020222222'         037
0DE8    2222 2020 2022 2222                          DC    X'2222202020222222'         037
0DF0    2222 2020 2022 2222                          DC    X'2222202020222222'         037
0DF8    2222 2020 2022 2222                          DC    X'2222202020222222'         037
0E00    2222 2020 2022 2222                          DC    X'2222202020222222'         037
0E08    2222 2020 2022 2222                          DC    X'2222202020222222'         037
0E10                                          DATB   DS    CL12                        037
0E1C    000C                                  ZERO   DC    X'000C'                     039
0E1E    F0F0 F0F0                             NULL   DC    C'0000'                     039
0E22    0E24                                  TTWO   DC    Y(TBLE)                     039
0E24                                          TBLE   DS    CL200                       039
0EEC    0EEE                                  TBEG   DC    Y(TABL)                     041
0EEE                                          TABL   DS    CL240                       041
0134                                                 END   CONS                        043
```

Figure CONA-08

Construction Program--Output

Output Number 01

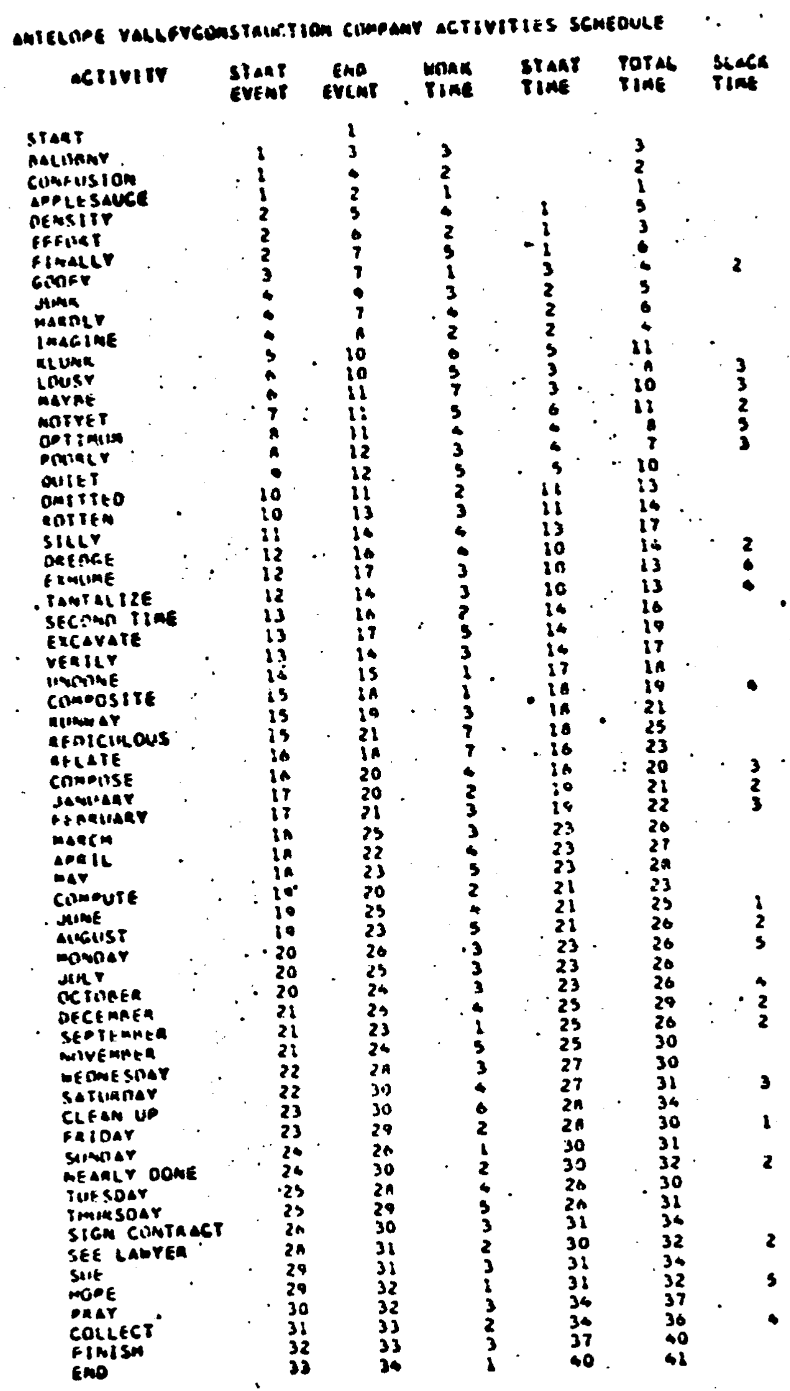


```
ANTELOPE VALLEYCONSTRUCTION COMPANY ACTIVITIES SCHEDULE

      ACTIVITY        START    END    WORK    START   TOTAL   SLACK
                      EVENT   EVENT   TIME    TIME    TIME    TIME

CLEAN UP               23      30      6
SIGN CONTRACT          26      30      3
SEE LAWYER             28      31      2
SUE                    29      31      3
HOPE                   29      32      1
PRAY                   30      32      3
COLLECT                31      33      2
FINISH                 32      33      3
END                    33      34      1
SECOND TIME            13      16      2
EXCAVATE               13      17      5
BRIDGE                 12      16      4
EXMINE                 12      17      3
RELATE                 16      18      7
COMPUTE                19      20      2
COMPOSE                16      20      4
COMPOSITE              15      18      1
RUNWAY                 15      19      3
REDICULOUS             15      21      7
JANUARY                17      20      2
MONDAY                 20      26      3
FEBRUARY               17      21      3
MARCH                  18      25      3
APRIL                  18      22      4
MAY                    18      23      5
JUNE                   19      25      4
JULY                   20      25      3
WEDNESDAY              22      28      3
AUGUST                 19      23      5
DECEMBER               21      26      4
SEPTEMBER              21      23      1
OCTOBER                20      24      3
NOVEMBER               21      24      5
DENSITY                 2       5      4
VERILY                 13      14      3
SUNDAY                 24      26      1
BALONEY                 1       3      3
CONFUSION               1       4      2
JUNK                    4       9      3
LOUSY                   6      10      5
EFFORT                  2       6      2
APPLESAUCE              1       2      1
KLUNK                   5      10      6
NOTYET                  7      11      5
TUESDAY                25      28      4
START                           1
UNDONE                 14      15      1
TANTALIZE              12      14      3
OMITTED                10      11      2
ROTTEN                 10      13      3
SILLY                  11      14      4
OPTIMUM                 8      11      4
POORLY                  8      12      3
QUIET                   9      12      5
MAYBE                   6      11      7
FINALLY                 2       7      5
GOOFY                   3       7      1
HARDLY                  4       7      4
IMAGINE                 4       8      2
SATURDAY               22      30      4
NEARLY DONE            24      30      2
THURSDAY               25      29      5
FRIDAY                 23      29      2
```

Figure CON-07

Construction Program--Output

Output Number 02

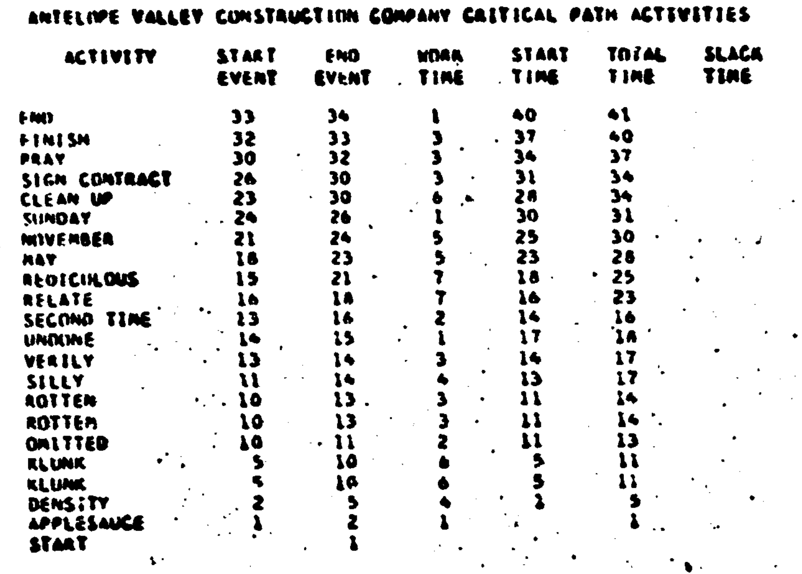


```
ANTELOPE VALLEYCONSTRUCTION COMPANY ACTIVITIES SCHEDULE

     ACTIVITY         START   END     WORK    START   TOTAL   SLACK
                      EVENT   EVENT   TIME    TIME    TIME    TIME

START                           1
BALCONY                 1       3       3               3
CONFUSION               1       4       2               2
APPLESAUCE              1       2       1               1
DENSITY                 2       5       4       1       5
EFFORT                  2       6       2       1       3
FINALLY                 2       7       5       1       6
GOOFY                   3       7       1       3       4       2
JUNK                    4       9       3       2       5
HARDLY                  4       7       4       2       6
IMAGINE                 4       8       2       2       4
KLUNK                   5      10       6       5      11
LOUSY                   6      10       5       3       8       3
MAYBE                   6      11       7       3      10       3
NOTYET                  7      11       5       6      11       2
OPTIMUM                 8      11       4       4       8       5
POORLY                  8      12       3       4       7       3
QUIET                   9      12       5       5      10
OMITTED                10      11       2      11      13
ROTTEN                 10      13       3      11      14
SILLY                  11      14       4      13      17
DREDGE                 12      16       4      10      14       2
EXHUME                 12      17       3      10      13       6
TANTALIZE              12      14       3      10      13       4
SECOND TIME            13      16       2      14      16
EXCAVATE               13      17       5      14      19
VERILY                 13      14       3      14      17
UNDONE                 14      15       1      17      18
COMPOSITE              15      18       1      18      19       4
RUNWAY                 15      19       3      18      21
REDICULOUS             15      21       7      18      25
RELATE                 16      18       7      16      23
COMPOSE                16      20       4      16      20       3
JANUARY                17      20       2      19      21       2
FEBRUARY               17      21       3      19      22       3
MARCH                  18      25       3      23      26
APRIL                  18      22       4      23      27
MAY                    18      23       5      23      28
COMPUTE                19      20       2      21      23
JUNE                   19      25       4      21      25       1
AUGUST                 19      23       5      21      26       2
MONDAY                 20      26       3      23      26       5
JULY                   20      25       3      23      26
OCTOBER                20      24       3      23      26       4
DECEMBER               21      25       4      25      29       2
SEPTEMBER              21      23       1      25      26       2
NOVEMBER               21      24       5      25      30
WEDNESDAY              22      28       3      27      30
SATURDAY               22      30       4      27      31       3
CLEAN UP               23      30       6      28      34
FRIDAY                 23      29       2      28      30       1
SUNDAY                 24      26       1      30      31
NEARLY DONE            24      30       2      30      32       2
TUESDAY                25      28       4      26      30
THURSDAY               25      29       5      26      31
SIGN CONTRACT          26      30       3      31      34
SEE LAWYER             28      31       2      30      32       2
SUE                    29      31       3      31      34
HOPE                   29      32       1      31      32       5
PRAY                   30      32       3      34      37
COLLECT                31      33       2      34      36       4
FINISH                 32      33       3      37      40
END                    33      34       1      40      41
```

Figure CON-08

Construction Program--Output

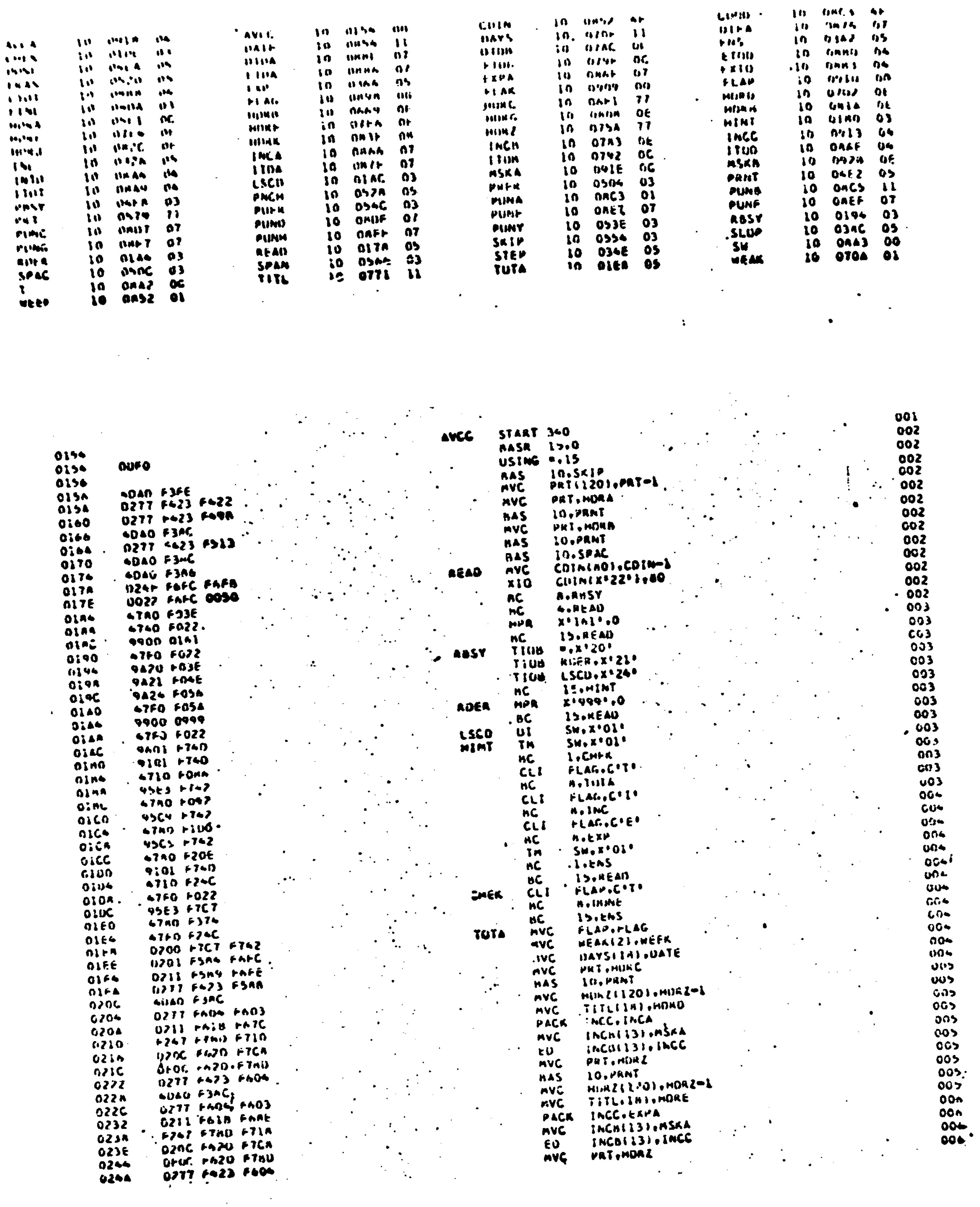
Output Number 03



```
ANTELOPE VALLEY CONSTRUCTION COMPANY CRITICAL PATH ACTIVITIES
    ACTIVITY      START    END     WORK    START   TOTAL   SLACK
                  EVENT    EVENT   TIME    TIME    TIME    TIME

END                33       34      1       40      41
FINISH             32       33      3       37      40
PRAY               30       32      3       34      37
SIGN CONTRACT      26       30      3       31      34
CLEAN UP           23       30      6       28      34
SUNDAY             24       26      1       30      31
NOVEMBER           21       24      5       25      30
MAY                18       23      5       23      28
REDICULOUS         15       21      7       18      25
RELATE             16       18      7       16      23
SECOND TIME        13       16      2       14      16
UNDONE             14       15      1       17      18
VERILY             13       14      3       14      17
SILLY              11       14      4       13      17
ROTTEN             10       13      3       11      14
ROTTEN             10       13      3       11      14
OMITTED            10       11      2       11      13
KLUNK               5       10      6        5      11
KLUNK               5       10      6        5      11
DENSITY             2        5      4        1       5
APPLESAUCE          1        2      1                1
START                        1
```

Figure CON-09

Antelope Valley Construction Co. Program



```
[illegible] 10 [illegible] 0A      AVCC  10 0154 00     CDIN  10 [illegible] 4F   [illegible] 10 [illegible] 4F
[illegible] 10 [illegible] [illegible]  DATE 10 [illegible] 11  DAYS 10 [illegible] 11  DTFA 10 [illegible] 07
[illegible] 10 [illegible] 05      DTDA  10 [illegible] 07  DTDB 10 07AC 01     ENS   10 03A2 05
[illegible] 10 [illegible] 05      ETDA  10 [illegible] 07  ETDB 10 079F 0C     ETDD  10 [illegible] 04
[illegible] 10 [illegible] 06      EXP   10 [illegible] 05  EXPA 10 0AAF 07     EXTO .10 [illegible] 04
[illegible] 10 [illegible] 03      FLAG  10 [illegible] 00  FLAK 10 0909 00     FLAP  10 [illegible] 00
[illegible] 10 [illegible] 0C      HORB  10 0AA9 0F     HORC  10 0AF1 77     HORD  10 [illegible] 01
[illegible] 10 [illegible] [illegible]  HORE 10 [illegible] 0F  HORG 10 [illegible] 0E  HORM 10 [illegible] 0E
[illegible] 10 [illegible] 0F      HORK  10 0A3F 0A     HORZ  10 075A 77     HINT  10 01B0 03
[illegible] 10 [illegible] 05      INCA  10 0AAA 07     INCH  10 07A3 0E     INCC  10 0913 04
[illegible] 10 [illegible] 0A      ITDA  10 0A7F 07     ITOM  10 0792 0C     ITDD  10 0AAF 04
[illegible] 10 [illegible] 0A      LSCO  10 01AC 03     MSKA  10 041E 0C     MSKB  10 0924 0E
PRST  10 [illegible] 03            PNCH  10 052A 05     PNFR  10 0504 03     PRNT  10 04E2 05
PRT   10 0579 77                   PUFR  10 054C 03     PUNA  10 0AC3 01     PUNB  10 0AC5 11
PUNC  10 0AD7 07                   PUNO  10 0ADF 07     PUNF  10 0AEE 07     PUNF  10 0AEF 07
PUNG  10 0AF7 07                   PUNM  10 0AFF 07     PUNY  10 053E 03     ABSY  10 0194 03
[illegible] 10 0146 03             READ  10 017A 05     SKIP  10 0554 03     SLOP  10 03AC 05
SPAC  10 [illegible] 03            SPAN  10 056A 03     STEP  10 034E 05     SW    10 0AA3 00
T     10 0AA2 0C                   TITL  10 0771 11     TOTA  10 01EA 05     WEAK  10 070A 01
WEEK  10 0A52 01
```

```
                                                                001
0154                      AVCC   START 340                      002
0154   0DF0                      BASR  15,0                     002
0156                             USING *,15                     002
015A   4DA0 F3FE                 BAS   10,SKIP                  002
015A   D277 F623 F622            MVC   PRT(120),PRT-1           002
0160   D277 F623 F49A            MVC   PRT,HORA                 002
0166   4DA0 F3AC                 BAS   10,PRNT                  002
016A   D277 F623 F513            MVC   PRT,HORB                 002
0170   4DA0 F3AC                 BAS   10,PRNT                  002
0174   4DA0 F3A6                 BAS   10,SPAC                  002
017A   D24F F6FC F6FB     READ   MVC   CDIN(80),CDIN-1          002
017E   0027 F6FC 0050            XIO   CDIN(X'22'),80           002
0186   4780 F03E                 BC    8,ABSY                   002
0188   4740 F022                 BC    4,READ                   003
018C   9900 0141                 MPR   X'1A1',0                 003
0190   47F0 F022                 BC    15,READ                  003
0194   9A20 F03E          ABSY   TIOB  *,X'20'                  003
0198   9A21 F04E                 TIOB  ADER,X'21'               003
019C   9A24 F05A                 TIOB  LSCO,X'24'               003
01A0   47F0 F054                 BC    15,HINT                  003
01A6   9900 0999          ADER   MPR   X'999',0                 003
01AA   47F0 F022                 BC    15,READ                  003
01AC   9A01 F740          LSCO   UI    SW,X'01'                 003
01B0   9101 F740          HINT   TM    SW,X'01'                 003
01B6   4710 F0AA                 BC    1,CHEK                   003
01BA   95E3 F742                 CLI   FLAG,C'T'                003
01BC   4780 F097                 BC    8,TOTA                   003
01C0   95C9 F742                 CLI   FLAG,C'I'                004
01C4   4780 F1D6                 BC    8,INC                    004
01CA   95C5 F742                 CLI   FLAG,C'E'                004
01CC   4780 F20E                 BC    8,EXP                    004
01D0   9101 F740                 TM    SW,X'01'                 004
01D4   4710 F24C                 BC    1,ENS                    004
01DA   47F0 F022                 BC    15,READ                  004
01DC   95E3 F7C7          CHEK   CLI   FLAP,C'T'                004
01E0   4780 F374                 BC    8,INNE                   004
01E6   47F0 F24C                 BC    15,ENS                   004
01EA   D200 F7C7 F742     TOTA   MVC   FLAP,FLAG                004
01EE   D201 F5A6 F6FC            MVC   WEAK(2),WEEK             004
01F4   D211 F5A9 F6FE            MVC   DAYS(18),DATE            004
01FA   D277 F623 F5AA            MVC   PRT,HORC                 005
020C   4DA0 F3AC                 BAS   10,PRNT                  005
0204   D277 F6A4 F6A3            MVC   HORZ(120),HORZ-1         005
020A   D211 F61B F67C            MVC   TITL(18),HORD            005
0210   F247 F7B0 F710            PACK  INCC,INCA                005
0216   D20C F620 F7CA            MVC   INCH(13),MSKA            005
021C   DE0C F620 F7B0            ED    INCB(13),INCC            005
0222   D277 F623 F6A4            MVC   PRT,HORZ                 005
022A   4DA0 F3AC                 BAS   10,PRNT                  005
022C   D277 F6A4 F6A3            MVC   HORZ(120),HORZ-1         005
0232   D211 F61B F6AE            MVC   TITL(18),HORE            006
023A   F247 F7B0 F71A            PACK  INCC,EXPA                006
023E   D20C F620 F7CA            MVC   INCH(13),MSKA            006
0244   DE0C F620 F7B0            ED    INCB(13),INCC            006
024A   D277 F623 F604            MVC   PRT,HORZ
```

Figure AVC-01

AVCC Program

```
0250   4DA0 F3AC                  BAS   10,PRNT                    006
0254   D277 F604 F603             MVC   HORZ(120),HORZ-1           006
025A   D211 F61B F6A0             MVC   TITL(18),HORF              006
0260   F247 F7BD F718             PACK  INCC,INCA                  006
0266   F247 F7C2 F718             PACK  ACCA,EXPA                  006
026C   FA44 F7BD F7C2             SP    INCC,ACCA                  007
0272   D20E F620 F7D5             MVC   INCB,MSKB                  007
0278   DE0E F620 F7BD             ED    INCB,INCC                  007
027E   D277 F423 F604             MVC   PRT,HORZ                   007
0284   4DA0 F3AC                  BAS   10,PRNT                    007
0288   D277 F604 F603             MVC   HORZ(120),HORZ-1           007
028E   D211 F61B F6B2             MVC   TITL(18),HORG              007
0294   F247 F758 F728             PACK  ITOD,ITDA                  007
029A   F247 F7BD F72A             PACK  INCC,ITDA                  007
02A0   D20C F63C F7CA             MVC   ITDB,MSKA                  007
02A6   DE0C F63C F7BD             ED    ITDB,INCC                  008
02AC   D277 F423 F604             MVC   PRT,HORZ                   008
02B2   4DA0 F3AC                  BAS   10,PRNT                    008
02B6   D277 F604 F603             MVC   HORZ(120),HORZ-1           008
02BC   D211 F61B F6C4             MVC   TITL(18),HORH              008
02C2   F247 F767 F730             PACK  ETOD,ETDA                  008
02C8   F247 F7BD F730             PACK  INCC,ETDA                  008
02CE   D20C F649 F7CA             MVC   ETDB,MSKA                  008
02D4   DE0C F649 F7BD             ED    ETDB,INCC                  008
02DA   D277 F423 F604             MVC   PRT,HORZ                   008
02E0   4DA0 F3AC                  BAS   10,PRNT                    009
02E4   D277 F604 F603             MVC   HORZ(120),HORZ-1           009
02EA   D211 F61B F6D6             MVC   TITL(18),HORJ              009
02F0   F247 F7BD F728             PACK  INCC,ITDA                  009
02F6   F247 F7C2 F730             PACK  ACCA,ETDA                  009
02FC   FA44 F7BD F7C2             SP    INCC,ACCA                  009
0302   D20E F656 F7D5             MVC   DTOB,MSKB                  009
0308   DE0E F656 F7BD             ED    DTOB,INCC                  009
030E   D277 F423 F604             MVC   PRT,HORZ                   009
0314   4DA0 F3AC                  BAS   10,PRNT                    010
0318   D277 F604 F603             MVC   HORZ(120),HORZ-1           010
031E   4DA0 F410                  BAS   10,SPAN                    010
0322   47F0 F022                  BC    15,READ                    010
0326   D200 F7C7 F742       INC   MVC   FLAP,FLAG                  010
032C   D501 F5B4 F6FC             CLC   WEAK(2),WEEK               010
0332   4780 F1FA                  BC    8,STEP                     010
0336   D277 F5AB F5AA             MVC   HORC(120),HORC-1           010
033C   D201 F5B4 F6FC             MVC   WEAK(2),WEEK               010
0342   D211 F5B9 F6FE             MVC   DAYS(18),DATE              010
0348   D277 F423 F5BB             MVC   PRT,HORC                   010
034E   F247 F74E F710       STEP  PACK  INTO,INCA                  011
0354   FA44 F753 F74E             AP    ITOT,INTO                  011
035A   FA44 F758 F74E             AP    ITOD,INTO                  011
0360   47F0 F022                  BC    15,READ                    011
0364   D200 F7C7 F742       EXP   MVC   FLAP,FLAG                  011
036A   D501 F5B4 F6FC             CLC   WEAK(2),WEEK               011
0370   4780 F236                  BC    8,SLOP                     011
0374   D277 F5BA F5BA             MVC   HORC(120),HORC-1           011
037A   D201 F5B4 F6FC             MVC   WEAK(2),WEEK               011
0380   D211 F5B9 F6FE             MVC   DAYS(18),DATE              011
```

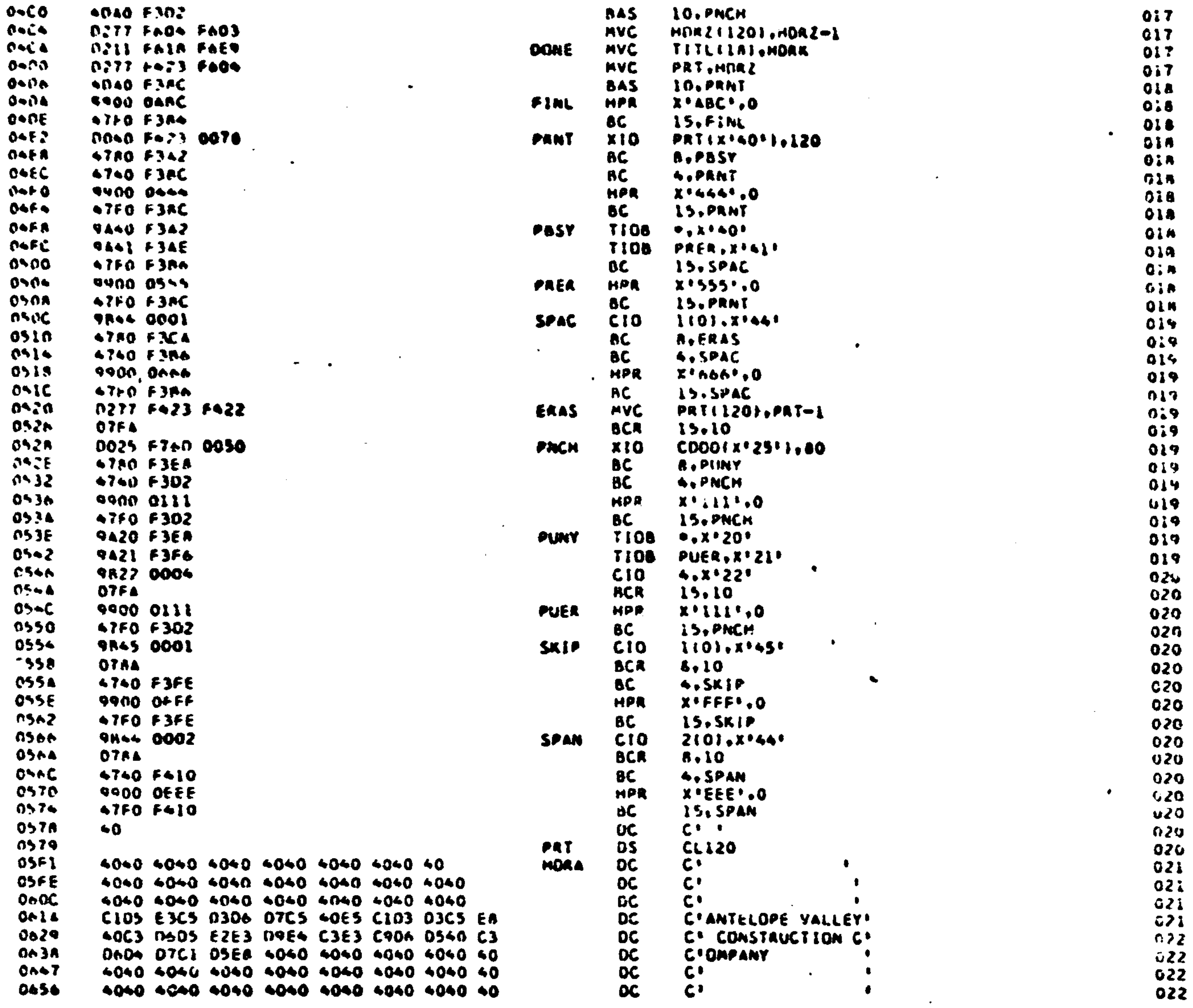
Figure AVC-02

AVC Program

```
0386     D277 F423 F58B               MVC   PRT,HORC                                012
038C     F247 F75D F718         SLOP  PACK  EXTO,EXPA                               012
0392     FA44 F762 F75D               AP    ETOT,EXTO                               012
0398     FA44 F767 F75D               AP    ETOD,EXTO                               012
039E     47F0 F022                    BC    15,READ                                 012
03A2     D277 F423 F58B         ENS   MVC   PRT,HORC                                012
03A8     D24F F76D F76C               MVC   COOD(80),COOD-1                         012
03AE     D201 F76D F584               MVC   PUNA(2),WEAK                            012
03B4     D211 F76F F589               MVC   PUNB(18),DAYS                           012
03BA     4DA0 F38C                    BAS   10,PRNT                                 012
03BE     D277 F604 F603               MVC   HORZ(120),HORZ-1                        013
03C4     F374 F7A1 F753               UNPK  PUNC,ITOT                               013
03CA     D20C F62D F7C8               MVC   INCB(13),MSKA                           013
03D0     DE0C F62D F753               ED    INCB(13),ITOT                           013
03D6     D211 F618 F67C               MVC   TITL(18),HORD                           013
03DC     D277 F423 F604               MVC   PRT,HORZ                                013
03E2     4DA0 F38C                    BAS   10,PRNT                                 013
03E6     D277 F604 F603               MVC   HORZ(120),HORZ-1                        013
03EC     F374 F789 F762               UNPK  PUND,ETOT                               013
03F2     D20C F62D F7C8               MVC   INCB(13),MSKA                           013
03F8     DE0C F62D F762               ED    INCB(13),ETOT                           014
03FE     D211 F618 F6AE               MVC   TITL(18),HORE                           014
0404     D277 F423 F604               MVC   PRT,HORZ                                014
040A     4DA0 F38C                    BAS   10,PRNT                                 014
040E     D277 F604 F603               MVC   HORZ(120),HORZ-1                        014
0414     F844 F7C2 F753               ZAP   ACCA,ITOT                               014
041A     FB44 F7C2 F762               SP    ACCA,ETOT                               014
0420     D20E F62D F705               MVC   INCB,MSKB                               014
0426     DE0E F62D F7C2               ED    INCB,ACCA                               014
042C     D211 F618 F6A0               MVC   TITL(18),HORF                           015
0432     D277 F423 F604               MVC   PRT,HORZ                                015
0438     4DA0 F38C                    BAS   10,PRNT                                 015
043C     D277 F604 F603               MVC   HORZ(120),HORZ-1                        015
0442     F374 F799 F75A               UNPK  PUNF,ITOD                               015
0448     D20C F63C F7C8               MVC   ITOB,MSKA                               015
044E     DE0C F63C F758               ED    ITOB,ITOD                               015
0454     D211 F618 F682               MVC   TITL(18),HORG                           015
045A     D277 F423 F604               MVC   PRT,HORZ                                015
0460     4DA0 F38C                    BAS   10,PRNT                                 015
0464     D277 F604 F603               MVC   HORZ(120),HORZ-1                        015
046A     F374 F7A1 F767               UNPK  PUNG,ETOD                               016
0470     D20C F649 F7C8               MVC   ETOB,MSKA                               016
0476     DE0C F649 F767               ED    ETOB,ETOD                               016
047C     D211 F618 F6C4               MVC   TITL(18),HORH                           016
0482     D277 F423 F604               MVC   PRT,HORZ                                016
0488     4DA0 F38C                    BAS   10,PRNT                                 016
048C     D277 F604 F603               MVC   HORZ(120),HORZ-1                        016
0492     F844 F7C2 F758               ZAP   ACCA,ITOD                               016
0498     FB44 F7C2 F767               SP    ACCA,ETOD                               016
049E     D20E F656 F705               MVC   DTOB,MSKB                               017
04A4     DE0E F656 F7C2               ED    DTOB,ACCA                               017
04AA     D200 F783 F74C               MVC   FLAK,T                                  017
04B0     D211 F618 F6D6               MVC   TITL(18),HORJ                           017
04B6     D277 F423 F604               MVC   PRT,HORZ                                017
04BC     4DA0 F38C                    BAS   10,PRNT                                 017
```

Figure AVC-03

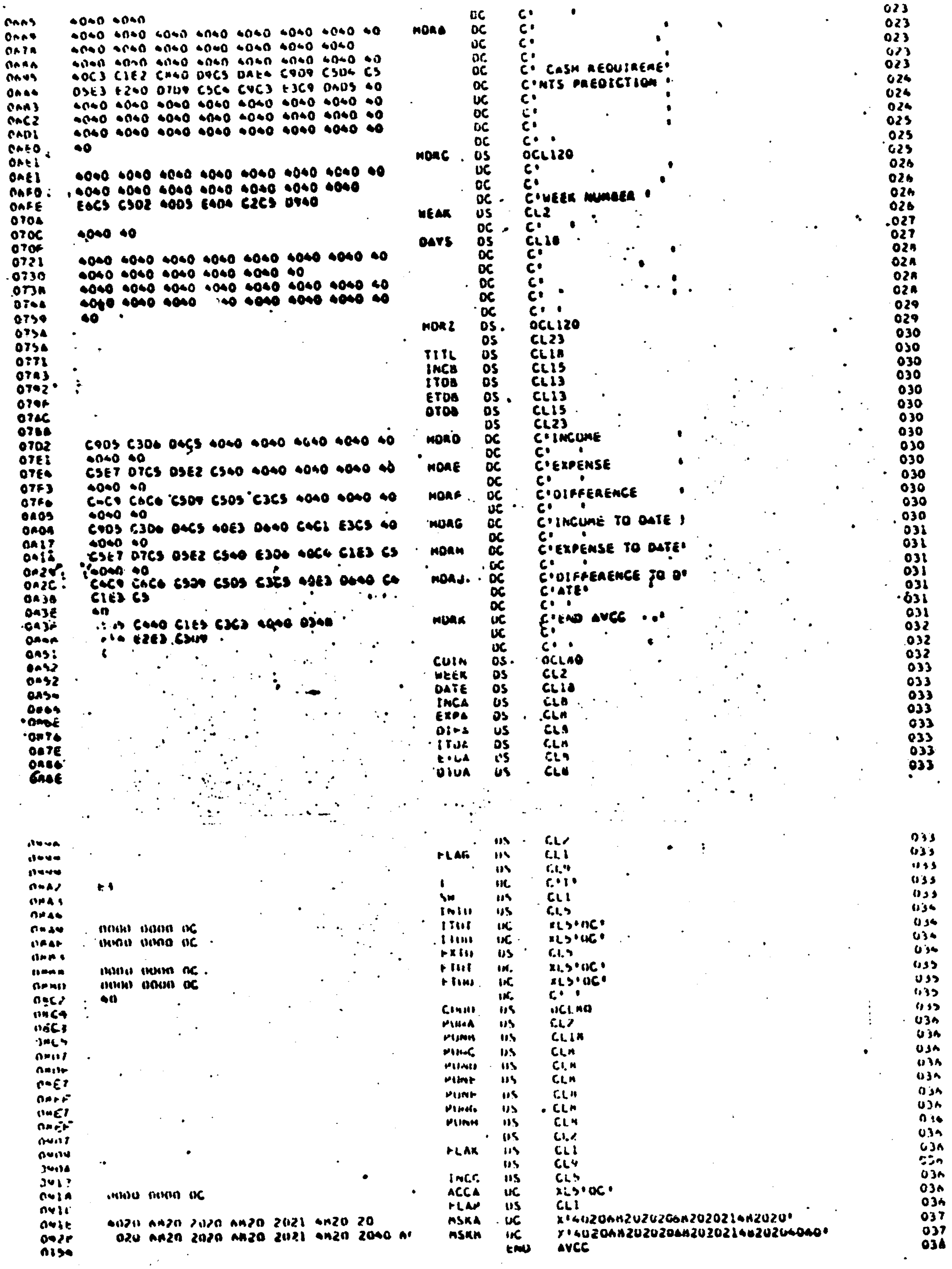
AVCC Program



```
04C0    4DA0 F3D2                                  BAS   10,PNCH                                017
04C4    D277 F604 F603                             MVC   HORZ(120),HORZ-1                       017
04CA    D211 F61A F6E9                     DONE    MVC   TITL(18),HORK                          017
04D0    D277 F423 F604                             MVC   PRT,HORZ                               017
04D6    4DA0 F3AC                                  BAS   10,PRNT                                018
04DA    9900 0ABC                          FINL    HPR   X'ABC',0                               018
04DE    47F0 F3A4                                  BC    15,FINL                                018
04E2    D040 F423 0078                     PRNT    XIO   PRT(X'40'),120                         018
04E8    4780 F342                                  BC    8,PBSY                                 018
04EC    4740 F3AC                                  BC    4,PRNT                                 018
04F0    9900 0444                                  HPR   X'444',0                               018
04F4    47F0 F3AC                                  BC    15,PRNT                                018
04F8    9A40 F342                          PBSY    TIOB  *,X'40'                                018
04FC    9A41 F3AE                                  TIOB  PRER,X'41'                             018
0500    47F0 F3D6                                  BC    15,SPAC                                018
0504    9900 0555                          PRER    HPR   X'555',0                               018
0508    47F0 F3AC                                  BC    15,PRNT                                018
050C    9B44 0001                          SPAC    CIO   1(0),X'44'                             019
0510    4780 F3CA                                  BC    8,ERAS                                 019
0514    4740 F3D6                                  BC    4,SPAC                                 019
0518    9900 0666                                  HPR   X'666',0                               019
051C    47F0 F3D6                                  BC    15,SPAC                                019
0520    D277 F423 F422                     ERAS    MVC   PRT(120),PRT-1                         019
0526    07FA                                       BCR   15,10                                  019
0528    D025 F740 0050                     PNCH    XIO   CD00(X'25'),80                         019
052E    4780 F3EA                                  BC    8,PUNY                                 019
0532    4740 F3D2                                  BC    4,PNCH                                 019
0536    9900 0111                                  HPR   X'111',0                               019
053A    47F0 F3D2                                  BC    15,PNCH                                019
053E    9A20 F3EA                          PUNY    TIOB  *,X'20'                                019
0542    9A21 F3F6                                  TIOB  PUER,X'21'                             019
0546    9B27 0004                                  CIO   4,X'22'                                020
054A    07FA                                       BCR   15,10                                  020
054C    9900 0111                          PUER    HPR   X'111',0                               020
0550    47F0 F3D2                                  BC    15,PNCH                                020
0554    9B45 0001                          SKIP    CIO   1(0),X'45'                             020
0558    078A                                       BCR   8,10                                   020
055A    4740 F3FE                                  BC    4,SKIP                                 020
055E    9900 0FFF                                  HPR   X'FFF',0                               020
0562    47F0 F3FE                                  BC    15,SKIP                                020
0566    9B44 0002                          SPAN    CIO   2(0),X'44'                             020
056A    078A                                       BCR   8,10                                   020
056C    4740 F410                                  BC    4,SPAN                                 020
0570    9900 0EEE                                  HPR   X'EEE',0                               020
0574    47F0 F410                                  BC    15,SPAN                                020
0578    40                                         DC    C' '                                   020
0579                                       PRT     DS    CL120                                  020
05F1    4040 4040 4040 4040 4040 4040 40   HORA    DC    C'             '                      021
05FE    4040 4040 4040 4040 4040 4040 4040         DC    C'              '                      021
060C    4040 4040 4040 4040 4040 4040 4040         DC    C'              '                      021
0614    C1D5 E3C5 D3D6 D7C5 40E5 C1D3 D3C5 E8      DC    C'ANTELOPE VALLEY'                     021
0629    40C3 D6D5 E2E3 D9E4 C3E3 C9D6 D540 C3      DC    C' CONSTRUCTION C'                     022
063A    D6D4 D7C1 D5E8 4040 4040 4040 4040 40      DC    C'OMPANY         '                     022
0647    4040 4040 4040 4040 4040 4040 4040 40      DC    C'               '                      022
0456    4040 4040 4040 4040 4040 4040 4040 40      DC    C'               '                      022
```

Figure AVC-04

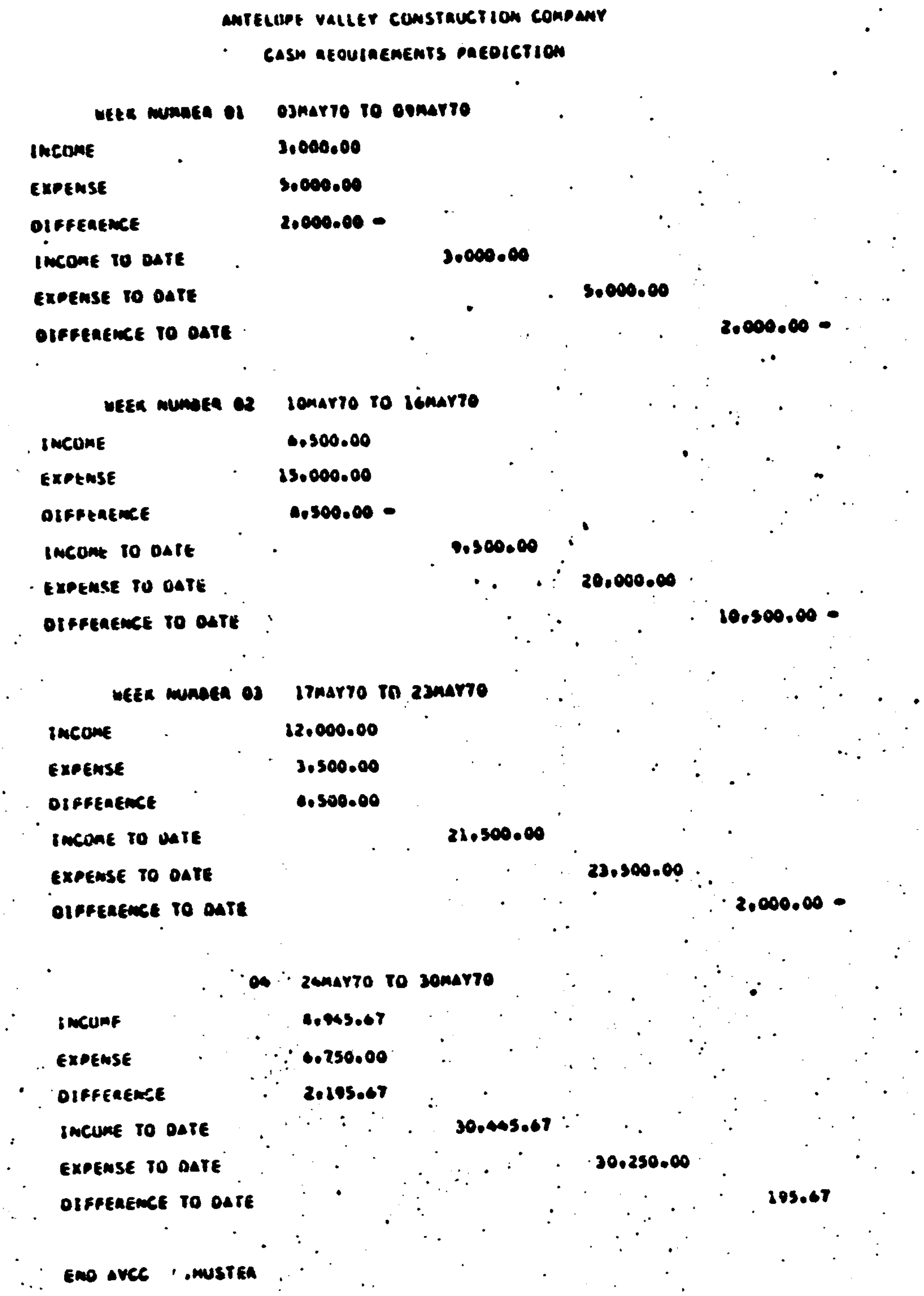
AVCC Program



```
                                                              DC    C'                       023
0AA5    4040 4040                                       HORA  DC    C'                       023
0AA9    4040 4040 4040 4040 4040 4040 4040 40                 DC    C'                       023
0A78    4040 4040 4040 4040 4040 4040 4040                    DC    C'                       023
0AAA    4040 4040 4040 4040 4040 4040 4040 40                 DC    C' CASH REQUIREME'       023
0A95    40C3 C1E2 C840 D9C5 DAE4 C909 C5D4 C5                 DC    C'NTS PREDICTION '       024
0AA4    D5E3 E240 D7D9 C5C4 C9C3 E3C9 0AD5 40                 DC    C'                       024
0AA3    4040 4040 4040 4040 4040 4040 4040 40                 UC    C'                       024
0AC2    4040 4040 4040 4040 4040 4040 4040 40                 DC    C'                       025
0AD1    4040 4040 4040 4040 4040 4040 4040 40                 DC    C'                       025
0AE0    40                                                    DC    C' '                     025
0AE1                                                    HORC  DS    0CL120                   026
0AE1    4040 4040 4040 4040 4040 4040 4040 40                 UC    C'                       026
0AF0    4040 4040 4040 4040 4040 4040 4040                    DC    C'                       026
0AFE    E6C5 C5D2 40D5 E4D4 C2C5 D940                         DC    C'WEEK NUMBER '          026
0704                                                    WEAK  US    CL2                      027
070C    4040 40                                               DC    C'                       027
070F                                                    DAYS  DS    CL18                     028
0721    4040 4040 4040 4040 4040 4040 4040 40                 DC    C'                       028
0730    4040 4040 4040 4040 4040 40                           DC    C'                       028
073A    4040 4040 4040 4040 4040 4040 4040 40                 DC    C'                       028
0744    4040 4040 4040  40 4040 4040 4040 40                  DC    C'                       029
0759    40                                                    DC    C' '                     029
075A                                                    HOR2  DS.   0CL120                   030
075A                                                          DS    CL23                     030
0771                                                    TITL  DS    CL18                     030
0783                                                    INCB  DS    CL15                     030
0792                                                    ITOB  DS    CL13                     030
079F                                                    ETDB  DS    CL13                     030
07AC                                                    DTDB  DS    CL15                     030
07BB                                                          DS    CL23                     030
07D2    C9D5 C3D6 D4C5 4040 4040 4040 4040 40           HORD  DC    C'INCOME                 030
07E1    4040 40                                               DC    C'                       030
07E4    C5E7 D7C5 D5E2 C540 4040 4040 4040 40           HORE  DC    C'EXPENSE                030
07F3    4040 40                                               DC    C'                       030
07F6    C4C9 C6C6 C5D9 C5D5 C3C5 4040 4040 40           HORF  DC    C'DIFFERENCE             030
0805    4040 40                                               UC    C'                       030
0808    C9D5 C3D6 D4C5 40E3 D640 C4C1 E3C5 40           HORG  DC    C'INCOME TO DATE )       031
0817    4040 40                                               DC    C'                       031
081A    C5E7 D7C5 D5E2 C540 E3D6 40C4 C1E3 C5           HORH  DC    C'EXPENSE TO DATE'       031
0829    4040 40                                               DC    C'                       031
082C    C4C9 C6C6 C5D9 C5D5 C3C5 40E3 D640 C4           HORJ. DC    C'DIFFERENCE TO D'       031
083B    C1E3 C5                                               DC    C'ATE'                   031
083E    40                                                    DC    C' '                     031
083F       C440 C1E5 C3C3 4040 D340                     HORK  UC    C'END AVCC   '           032
0844       E2E3 C5D9                                          UC    C'                       032
0851                                                          UC    C' '                     032
0852                                                    CUIN  DS.   0CL80                    033
0852                                                    WEEK  DS    CL2                      033
0854                                                    DATE  DS    CL18                     033
0866                                                    INCA  US    CLB                      033
086E                                                    EXPA  DS    CL8                      033
0876                                                    DIFA  US    CL8                      033
087E                                                    ITDA  DS    CL8                      033
0886                                                    ETDA  US    CL8                      033
088E                                                    DTDA  US    CL8                      033


0896                                                          DS    CL7                      033
0896                                                    FLAG  DS    CL1                      033
0896                                                          DS    CL9                      033
08A7    E3                                              I     DC    C'T'                     033
08A8                                                    SW    DS    CL1                      033
08A9                                                    INTD  DS    CL5                      034
08AA    0000 0000 0C                                    ITDT  DC    XL5'0C'                  034
08AF    0000 0000 0C                                    ITDD  DC    XL5'0G'                  034
08B4                                                    EXTD  DS    CL5                      034
08B9    0000 0000 0C                                    ETDT  DC    XL5'0C'                  035
08BD    0000 0000 0C                                    ETDD  DC    XL5'0C'                  035
08C2    40                                                    DC    C' '                     035
08C4                                                    CARD  DS    0CL80                    035
08C3                                                    PUNA  DS    CL7                      036
08C9                                                    PUNB  DS    CL18                     036
08D7                                                    PUNC  DS    CL8                      036
08DF                                                    PUND  DS    CL8                      036
08E7                                                    PUNE  DS    CL8                      036
08EF                                                    PUNF  DS    CL8                      036
08E7                                                    PUNG  DS    CL8                      036
08EF                                                    PUNH  DS    CL8                      036
0907                                                          DS    CL2                      036
0909                                                    FLAK  DS    CL1                      036
090A                                                          DS    CL9                      036
0913                                                    INCC  DS    CL5                      036
091A    0000 0000 0C                                    ACCA  DC    XL5'0C'                  036
091C                                                    FLAP  DS    CL1                      036
091E    4020 6820 2020 6820 2021 4820 20                MSKA  DC    X'40206820202068202021482020'  037
092F     020 6820 2020 6820 2021 4820 2040 4(           MSKB  DC    X'402068202020682020214820204040'  037
0154                                                          END   AVCC                     038
```

Figure AVC-05

AVCC--Output



```
                     ANTELOPE VALLEY CONSTRUCTION COMPANY
                         CASH REQUIREMENTS PREDICTION

      WEEK NUMBER 01   03MAY70 TO 09MAY70
INCOME                 3,000.00
EXPENSE                5,000.00
DIFFERENCE             2,000.00 -
INCOME TO DATE                       3,000.00
EXPENSE TO DATE                                  5,000.00
DIFFERENCE TO DATE                                              2,000.00 -

      WEEK NUMBER 02   10MAY70 TO 16MAY70
INCOME                 6,500.00
EXPENSE               15,000.00
DIFFERENCE             8,500.00 -
INCOME TO DATE                       9,500.00
EXPENSE TO DATE                                 20,000.00
DIFFERENCE TO DATE                                             10,500.00 -

      WEEK NUMBER 03   17MAY70 TO 23MAY70
INCOME                12,000.00
EXPENSE                3,500.00
DIFFERENCE             8,500.00
INCOME TO DATE                      21,500.00
EXPENSE TO DATE                                 23,500.00
DIFFERENCE TO DATE                                              2,000.00 -

                  04   24MAY70 TO 30MAY70
INCUMF                 8,945.67
EXPENSE                6,750.00
DIFFERENCE             2,195.67
INCUME TO DATE                      30,445.67
EXPENSE TO DATE                                 30,250.00
DIFFERENCE TO DATE                                                195.67

END AVCC  ',MUSTER
```

Figure AVC-06

SECTION II

DATA STRUCTURES

II DATA STRUCTURES

The Swanson Study was presented to help the reader become aware of a need for other programming tools other than the higher level languages. If he is going to be able to function at the level of this study, it will be well to master certain aspects of Data Structures.

The object of this section of the book is to provide the material so that the reader can develop his own information system, or at least work at the level of proficiency the Swanson Study demands.

This introductory section presents a definition of data structures by depicting the recommendations of the Committee of the Association of Computing Machinery for a college level course in data structures. This allows the reader to choose other subjects he might desire and pursue them by using the bibliography in the appendix.

HISTORY

Prior to the 1950's programmers were concerned with both memory allocation and the algorithm. Each programmer accepted the fact that he must exercise adequate control over fast-access storage, if the program were too large for the storage and he had to use auxiliary.

During the 1950's higher level languages were introduced, and the programmer became more involved with problem solving than with the machine and its various aspects. The programs grew in size and so did the problem of storage control and storage allocation.

Due to the shielding effect of the higher level languages, the programmer moved further and further from the reality of the efficiency and inefficiency of the machine, and the requirement was for larger and more expensive memories. When the cost became so great, the realization came that it was time for the programmer to face reality and go back and woik with such elements as information structures and storage allocation.

It would appear that we have come back to a point that has characteristics similar to the pre-1950's.

It will now be necessary for the programmer to put forth the extra effort to master the areas he avoided even though they may appear to be more difficult in some instances.

The term "INFORMATION STRUCTURE" as used in this book relates to the contents of the course of study presented by the Association of Computing Machinery in 1968. This term is used to infer the structural relationships between various data elements.

The Curriculum Committee on Computer Science of the Association for Computing Machinery[1] has offered recommendations for a college level course, entitled "Data Structures".

1. Basic Concepts of Data
    a. Representation of information as data inside and outside the computer, bits, nodes and data elements
    b. Data files and tables
    c. Names, values, environments
    d. Use of pointer or linkage variables to represent data structures

[1] "Curriculum 68, Recommendations for Academic Programs in Computer Science," Communications of the ACM, Vol. II, No. 3, March 1968, pp. 151-197.

- e. Identifying entities about which data is to be maintained
- f. Selecting data nodes and structures which are to be used in problem solution
- g. Storage media, storage structures, encoding data and transformations from one medium and/or code to another
- h. Alternative representations of information and data
- i. Packing, unpacking and compressing data
- j. Data formats, data description languages, specifications of data transformations

2. Linear Lists and Strings
   - a. Stacks, last-in first-out, first-in first-out, double ended, and other linear lists
   - b. Sequential versus linked storage
   - c. Single versus double link circular lists
   - d. Character strings of variable length
   - e. Word packing, part word addressing
   - f. Pointer manipulation
   - g. Insertion, deletion and accessing of list elements
3. Arrays and Orthagonal Lists
   - a. Storage of rectangular arrays in a one dimensioned media
   - b. Storage mapping functions
   - c. Direct and indirect address computation
   - d. Space requirements
   - e. Set up time
   - f. Accessing time and dynamic relocation times
   - g. Storage and accessing triangular arrays
   - h. Tetrahedral arrays

i. Space matrices

4. Tree Structures

a. Trees, subtrees, ordered trees, free trees, oriented trees, binary trees

b. Representation of trees using binary trees

c. Sequential techniques

d. Threaded lists

e. Insertion, deletion and accessing of elements of trees

f. Relative referencing, finding successors and predecessors

g. Walking through trees

h. Examples of tree structures such as algebraic formulas, arrays and other heirarchic data structures (PLI/and COBOL)

5. Storage Systems and Structures

a. Behavioral properties of Unit record (card)

b. Random access (core)

c. Linear (tape)

d Intermediate (disk and drum)

e. Storage media including cost, size, speed, reusability

f. Inherent record and file structure

g. Deficiencies and interrelation of these properties

h. Influence and machine structure--addressing on data structuring

i. Hierarchies of storage, virtual memory, segmentation, paging and bucketing

j. Influence of data structures and data manipulation on storage systems

k. Associative structures, both hardware and software

6. Storage Allocation and Collection

7. Multi-linked Structures

8. Sorting (ordering techniques)

   a. Radix sorting

   b. Radix exchange sorting

   c. Merge sorting

   d. Bubble sorting

   e. Address table sorting

   f. Topological sorting

   g. Comparative efficiency of sorting techniques

   h. Effect of data structures and storage structures on sorting techniques

9. Symbol Table and Searching

10. Data Structures in Programming Languages

    a. Compile time and run time data structures needed to implement source language data structures of programming languages

    b. Linkage between partially executed procedures

    c. Data structures of co-routines, scheduled procedures, and control structures

    d. Storage management of data structures in procedure oriented languages

    e. Examples of higher level languages which include list processing and other data structure features

11. Formal Specification of Data Structures

a. Specification of syntax of classes of data structures

b. Predicate selectors and constructors for data manipulation

c. Data definition facilities

d. Programs as data structures

e. Computers as data structures and transformations

f. Formal specification of semantics

g. Formal systems viewed as data structures

12. Generalized Data Management Systems

a. Structures of generalized data management systems

b. Directory maintenance

c. User languages (query)

d. Data description maintenance

e. Job management

f. Embedding data structures in generalized data management systems

g. Examples of generalized data management systems and comparison of system features

II DATA STRUCTURES

If the reader will notice the numbering sequence used for this section, it does contain a certain structuring. The section begins with Linear Lists and ends with Virtual Memory.

There was a definite reason in presenting the subjects in this particular order, and that was one of logical development of the tools needed to create information systems.

The basic information starts with lists (and links), which will partially answer the problem of linking, created by the Swanson Study. If the reader will examine the definition of data structures, he will see that Files are listed at lb. Files are listed in this section at 2.4 because they fit logically into the development of the material for an information system.

The final culmination of this section ends with Dynamic Storage Allocation and Virtual Memory. They are presented as a little icing for the cake, since certain programming positions do not require an awareness of them, however they are presented for the reader who wants to do just a little bit more.

II DATA STRUCTURES

- 2.1 Introduction
- 2.2 Linear Lists
    - 2.2.1 Stacks, Deques and Queues
    - 2.2.2 Linked Lists
- 2.3 Trees
    - 2.3.1 Hierarchical Ring Structure
    - 2.3.2 Traversing Trees - Preorder and Post Order
    - 2.3.3 Directories
        - 2.3.3.1 Structured Tree Directory
        - 2.3.3.2 Random Ordered Directories
- 2.4 Files
    - 2.4.1 Sequential Files
    - 2.4.2 Inverted Files
        - 2.4.2.1 Creation, Maintenance and Deletion of Keys

- 2.4.2.2 Logical Operators
    - 2.4.2.2.1 Logical Operators
    - 2.4.2.2.2 Logical Operator "OR"
    - 2.4.2.2.3 Logical Operator "AND"
- 2.4.3 Multilist File
    - 2.4.3.1 Deletion of Records and Addition of Keys
- 2.4.4 Cellular Multilist File

- 2.5 Sorting
    - 2.5.1 Sorting Methods
        - 2.5.1.1 Key Sort
        - 2.5.1.2 Detached Key Sort
        - 2.5.1.3 Non Detached Key Sort
    - 2.5.2 Radix Sort
    - 2.5.3 Quicksort
    - 2.5.4 Topological Sort
    - 2.5.5 Quadratic Sort
    - 2.5.6 Calculated Address Sort
    - 2.5.7 Sorting by Insertion
    - 2.5.8 Interchange Sort
- 2.6 Merging
    - 2.6.1 Two Way Merge
        - 2.6.1.1 Merging Ordered Files with Subfiles
            - 2.6.1.1.1 Merging Step Downs
- 2.7 Search
    - 2.7.1 Linear Search
    - 2.7.2 Binary Search
- 2.8 Dynamic Storage Allocation

2.9 Virtual Memory

List Formats

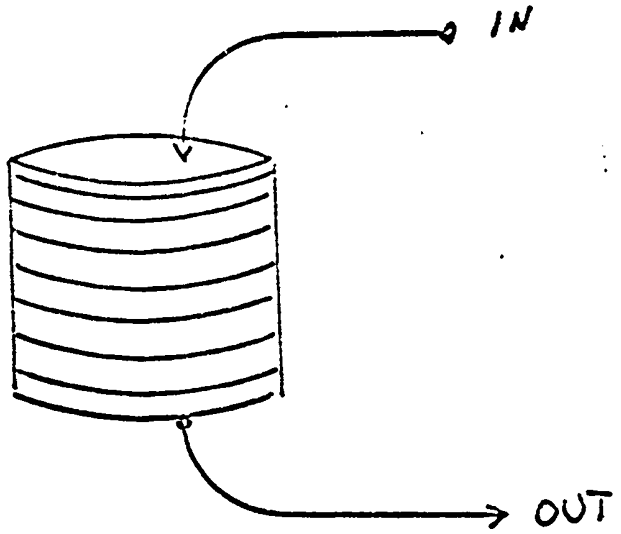
An important aspect of the list languages is that the original list of available space must be either set up automatically by the system or manually by the programmer. This original space amounts to a pool or block of storage which the program can draw from as needed.

Three kinds of fixed format lists have been adopted by various writers. The format presented here was originally named by Knuth:

1. QUEUES enable opposite end accessibility; input one end and output the other end.
2. DEQUES enable double end accessibility; a word or record can be added or deleted at either end.
3. STACKS enable double end accessibility; a word or record can be added to one end.

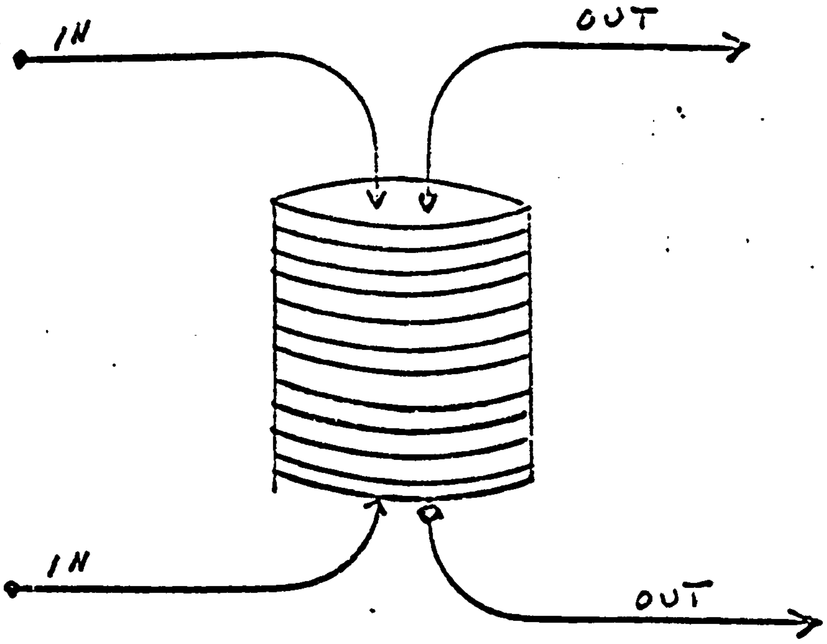
It is important to note that with the formats presented here, it is not possible to access records that are located in the center of either method.

QUEUE



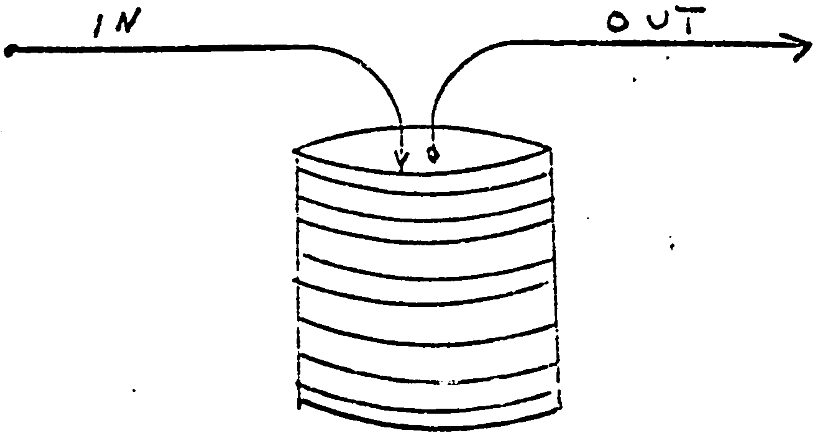
The Que has two ends, and the records are entered at the top and withdrawn from the bottom. This method is generally used when records are to be serviced in the order of receipt, (FIFO, which is first-in first-out). This method also is used in accounting. A Queue is used for a list of areas to receive records from sequential files, and it is also used as a list of areas to be written on a sequential file or files.

DEQUE



It is possible to insert or delete a record from either the top or the bottom of the deque. This could be called last-in first-out or first-in first-out.

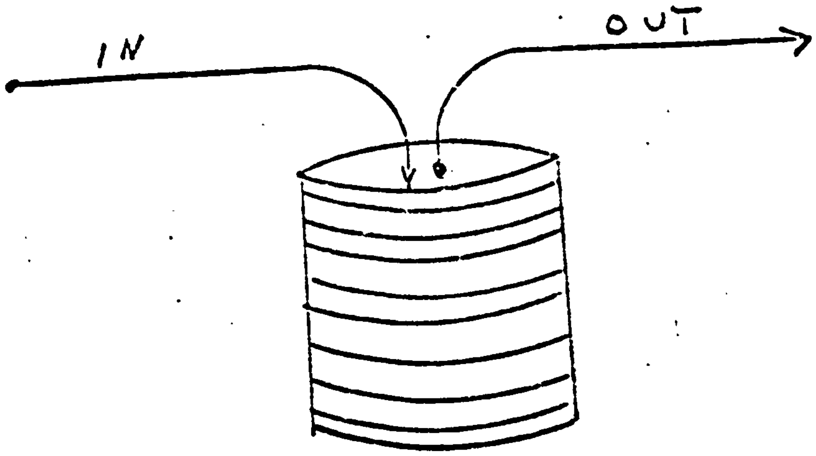
STACK



A stack takes out the most recent item that was put in. It uses the last-in first-out method. This method is one that is used more for list than the other methods. The list of available space is one application for this method.

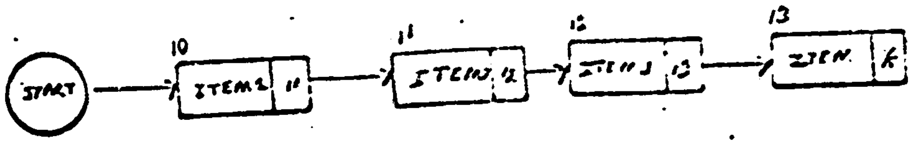
List of Available Space

A linked list indicates some type of instrument to control the space not in use. As evidenced by the language SLIP, written by Wizenbaum, a List of Available Space is quite necessary. It is essentially a stack.



The last (top) record is the first one out. All nodes that are not being used are linked by this list.

Linked Lists



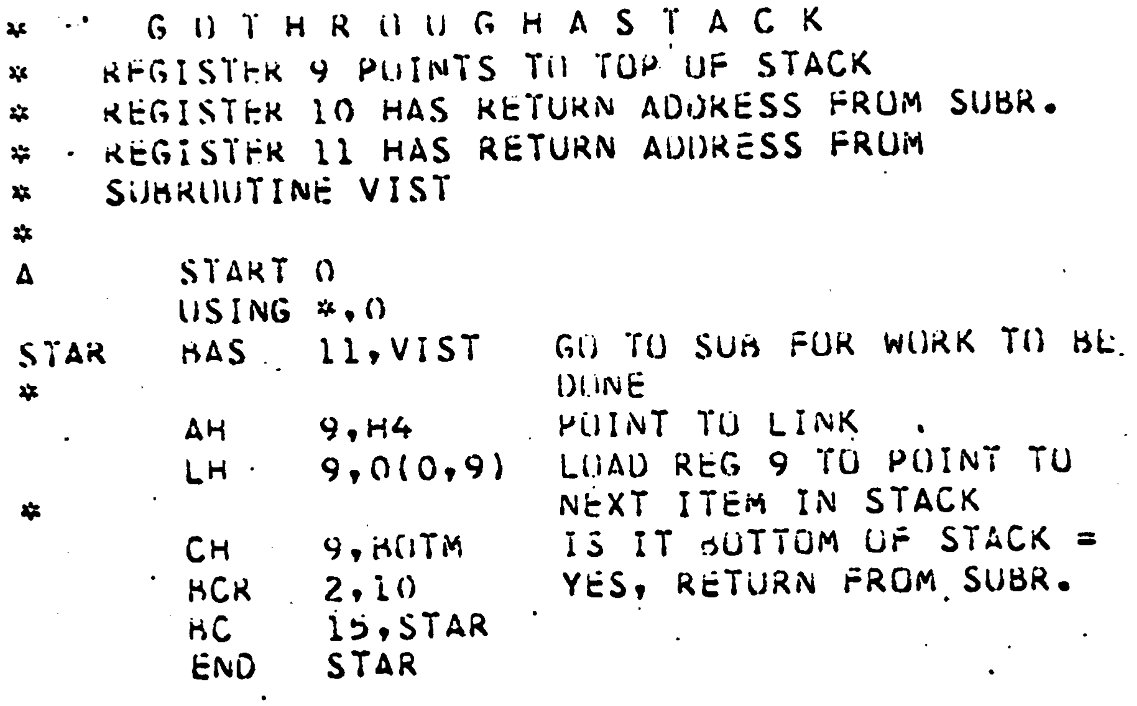
LINK VARIABLE

ADDRESS	ITEM	LINK
10	ITEM 1	11
11	ITEM 2	12
12	ITEM 3	13
13	ITEM 4	*

Go Through a Stack

The linked list is much more flexible than the sequential list. The addresses 10, 11, 12 and 13 are addresses in memory, and the figure (*) is a null link, which stops the action.

The program requires a link variable that points to the address number 10, and from address 10, all other items on the list are easily found.



```
*     G O T H R O U G H A S T A C K
*    REGISTER 9 POINTS TO TOP OF STACK
*    REGISTER 10 HAS RETURN ADDRESS FROM SUBR.
*    REGISTER 11 HAS RETURN ADDRESS FROM
*    SUBROUTINE VIST
*
A         START 0
          USING *,0
STAR      BAS    11,VIST     GO TO SUB FOR WORK TO BE
*                            DONE
          AH     9,H4        POINT TO LINK
          LH     9,0(0,9)    LOAD REG 9 TO POINT TO
*                            NEXT ITEM IN STACK
          CH     9,BOTM      IS IT BOTTOM OF STACK =
          BCR    2,10        YES, RETURN FROM SUBR.
          BC     15,STAR
          END    STAR
```

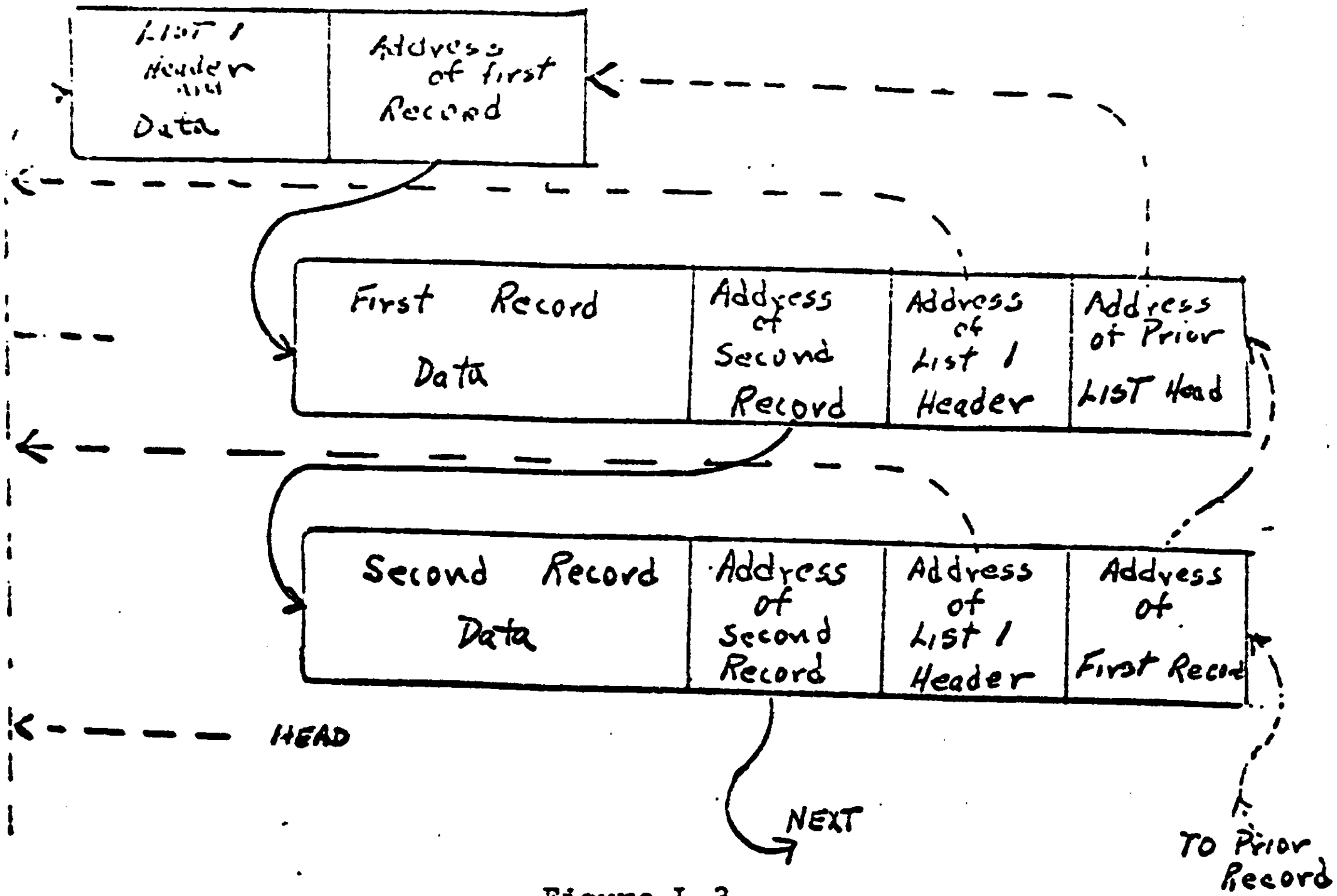
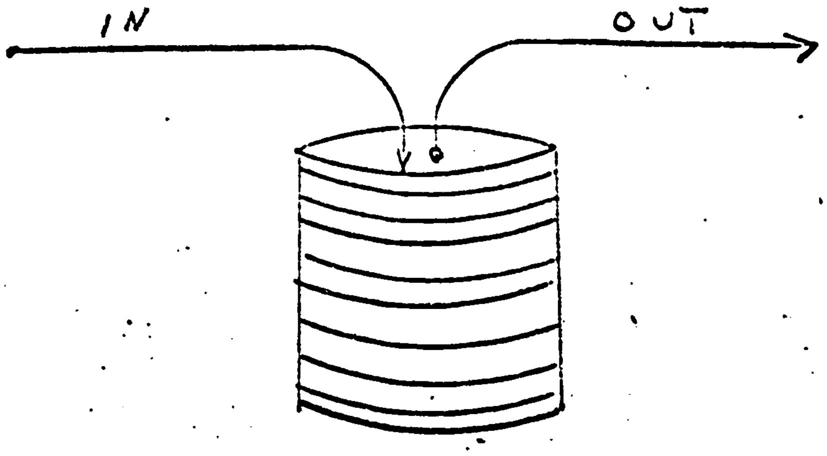


Figure L-3

Refer to the Swanson Study to Chart #9, which depicts the Bill of Material retrieval. Also refer to File Page #4, which links the various files together. The above drawing is intended to carry the ideas the two Swanson charts contain, and prepare the reader for the linking process.

Activities

1. Duplicate the drawing of the linkage of File Page #4.
2. Duplicate the drawing of Chart #9.
3. Listen to the tape cassettes that pertain to this section, and review the Basic Assembler set of instructions.



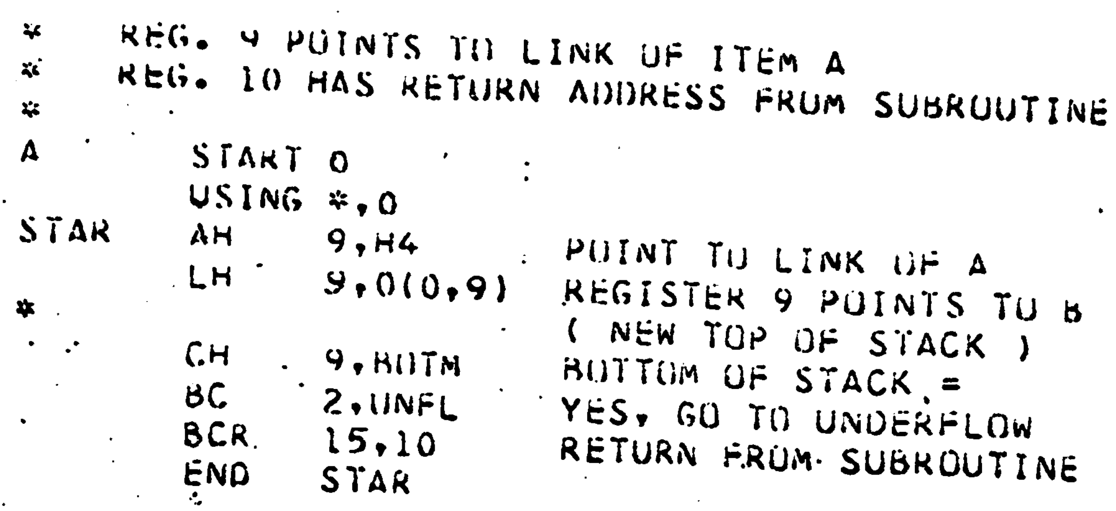
```
*  ADD TO TOP OF A STACK
*
*   REGISTER 8 POINTS TO AVAILABLE STORAGE
*   REGISTER 9 POINTS TO TOP OF STACK
*   REGISTER 10 HAS RETURN ADDRESS FROM SUBR.
*
A        START 0
         USING *,0
STAR     CH     8,LIMT      ANY MORE AVAILABLE SPACE
         BC     2,OVFL      NO, BRANCH TO OVERFLOW
         MVC    0(4,8),DATA  STORE DATA IN AVAIL.
         STH    8,TEMP      SAVE POINTER FOR NEW TOP
*                           OF STACK
         AH     8,H4        REG. 8 POINTS TO LINK OF
*                            AVAIL.
         STH    9,0(0,8)    AVAIL LINKED TO TOP OF
*                           STACK
         LH     9,TEMP      REG. 9 POINTS TO LINK OF
*                           AVAIL
         AH     8,H2        REG. 8 POINTS TO NEW
*                            AVAILABLE STORAGE
         BCR    15,10       RETURN FROM SUBROUTINE
         END    STAR
```

Figure L-4

The disadvantage of the extra space used by the links may be circumvented by including several items to each link, and a link need not take a large amount of space. It would take only one byte on a byte oriented computer. It is also possible to create an overlap of tables and share common parts. (This will be discussed later.)

This example of a one way link assumes that in any type of search, the direction can proceed forward only.

Delete From Top of Stack



```
*    REG. 9 POINTS TO LINK OF ITEM A
*    REG. 10 HAS RETURN ADDRESS FROM SUBROUTINE
*
A          START 0
           USING *,0
STAR       AH    9,H4          POINT TO LINK OF A
           LH    9,0(0,9)      REGISTER 9 POINTS TO B
*                              ( NEW TOP OF STACK )
           CH    9,BOTM        BOTTOM OF STACK =
           BC    2,UNFL        YES, GO TO UNDERFLOW
           BCR   15,10         RETURN FROM SUBROUTINE
           END   STAR
```

Figure L-5

Delete From Inside a Stack

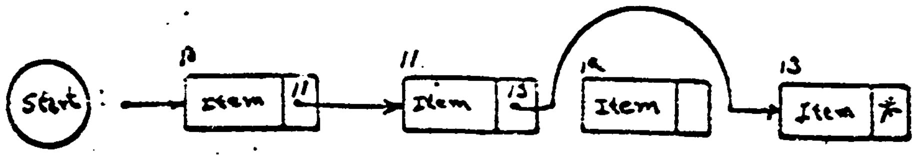
The deletion of an item (or items) is easy, since the deletion requires a change in the link to address 13. The same change with a sequential file would entail a deletion of the item in address 12, and a move up of the following items.

```
*    REG. 9 POINTS TO LINK OF ITEM A
*    REG. 10 HAS RETURN ADDRESS FROM SUBROUTINE
*    REG. 11 IS USED AS A TEMPORARY REGISTER
*
A        START 0
         USING *,0
STAR     AH    9,H4        POINT TO LINK OF A
         LH    11,0(0,9)   REG. 11 POINTS TO B
         AH    11,H4       POINT TO LINK OF B
         MVC   0(2,9),0(11) LINK OF A POINTS TO C
         SH    9,H4        POINT TO A
         CH    11,BOTM     BOTTOM OF STACK =
         BC    2,UNFL      YES, GO TO UNDERFLOW
         BCR   15,10       RETURN FROM SUBROUTINE
         END   STAR
```

Figure L-6

Insert Into Stack (or string) Between B and C

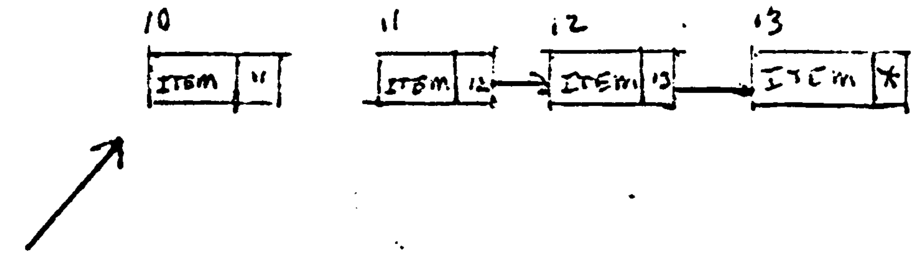
The insertion of an item into a list is easy because the inserted node can reside at any available location.



Address	Item	Link
10	Item	11
11	Item	13
12	Item	*
13	Item	*

```
*
*       REG. 8 POINTS TO AVAILABLE STORAGE
*       REG. 9 POINTS TO LEFT LINK OF ITEM C
*       REG. 10 HAS RETURN ADDRESS FROM MAIN SUBR.
*
A          START 0
           USING *,0
STAR       CH      8,LIMT   ANY MORE AVAILABLE STORAGE
           BC      2,OVFL    NO, GO TO OVERFLOW
           MVC     0(2,8),0(9)   STORE RIGHT LINK TO B
*                               INTO LEFT LINK OF AVAIL
           AH      8,H2      POINT TO DATA OF AVAIL
           MVC     0(2,8),DATA   STORE DATA IN AVAIL
           AH      8,H2   POINT TO RIGHT LINK OF AVAIL
           STH     9,0(0,8)   STORE LINK TO C INTO
*                              RIGHT LINK OF AVAIL
           AH      8,H2      POINT TO NEW AVAILABLE
*                            STORAGE
           MVC     0(2,9),0(8)   STORE LINK TO AVAIL
*                                  INTO LEFT LINK OF C
           SH      8,H6      POINT TO LEFT LINK OF AVAL
           LH      9,0(0,8)   REG. 9 WILL POINT TO
*                              RIGHT LINK OF B
           STH     8,0(0,9) RIGHT LINK OF B WILL
*                            POINT TO LEFT LINK OF AVAL
           AH      8,H4      POINT TO RIGHT LINK OF AV.
           LH      9,0(0,8) REG. 9 WILL POINT TO
*                            LEFT LINK OF C
           AH      8,H2      POINT TO NEW AVAILABLE ST.
           BCR     15,10     RETURN FROM SUBROUTINE
           END     STAR
```

Delete From Left End of List



```
*    DELETE FROM THE LEFT END OF A LIST
*
*    REG. 9 POINTS TO THE LEFT END OF THE LIST
*    REG. 10 HAS RETURN ADDRESS FROM MAIN SUBR.
*
A       START 0
        USING *,0
STAR    AH     9,H4      POINT TO RIGHT LINK OF A
        LH     9,0(0,9)  POINT TO NEW LEFT END OF
*                        THE LIST
        CH     9,REND    IS IT THE RIGHT END OF
*                        THE LIST
        BC     2,UNFL    YES, GO TO UNDERFLOW
        BCR    15,10     RETURN FROM SUBROUTINE
        END    STAR
```

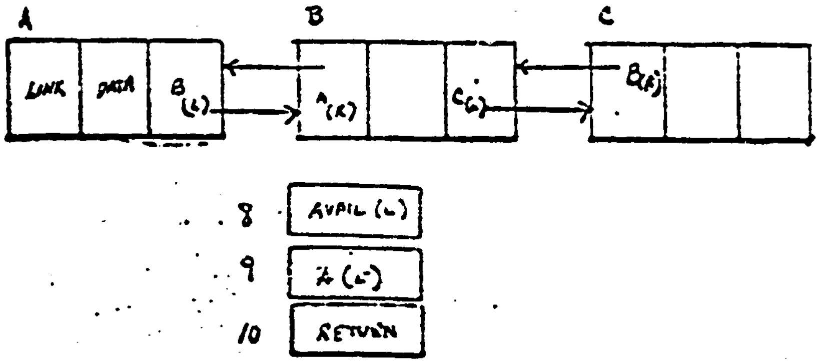
Go Through a Doubly Linked List

Reg 9---points to left end of the list

Reg 10---contains the return address from the main subroutine

Reg 11---contains the return address from the subroutine Visit.

(This subroutine performs some function and then returns us to the main program or subroutine we were in before the jump to Visit.)



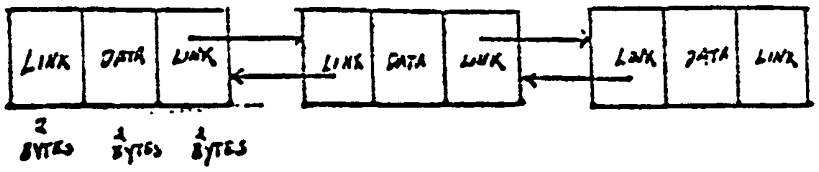
```
*
*     REG. 9 POINTS TO THE LEFT END OF THE LIST
*     REG. 10 HAS RETURN ADDRESS FROM MAIN SUBR.
*     REG. 11 HAS RETURN ADDRESS FROM SUB. VIST
*
A        START 0
         USING *,0
STAR     AH     9,H2      POINT TO DATA
         BAS    11,VIST   GO TO SUBR. FOR WORK TO BE
*                         DONE
         AH     9,H2      POINT TO RIGHT LINK
         LH     9,0(0,9)  LOAD REG. 9 TO POINT TO
*                         NEXT ITEM IN THE LIST
         CH     9,REND    IS IT THE RIGHT END OF
*                         THE LIST
         BCR    2,10      YES, RETURN FROM SUBR.
         BC     15,STAR   NO, CONTINUE THROUGH LIST
         END    STAR
```

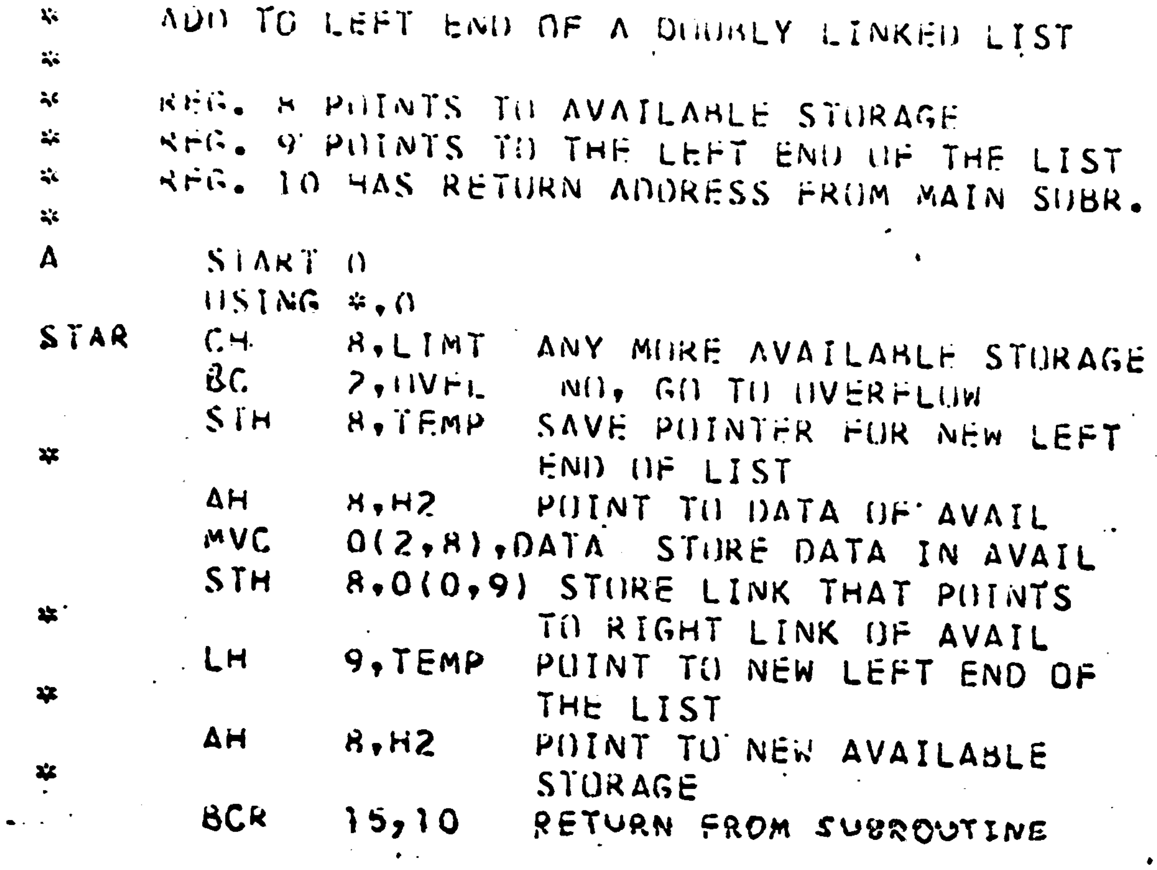
Add to Left End of Doubly Linked List

Reg 8---points to available storage

Reg 9---points to left end of list

Reg 10---contains the return address from the subroutine





```
*     ADD TO LEFT END OF A DOUBLY LINKED LIST
*
*     REG. 8 POINTS TO AVAILABLE STORAGE
*     REG. 9 POINTS TO THE LEFT END OF THE LIST
*     REG. 10 HAS RETURN ADDRESS FROM MAIN SUBR.
*
A        START 0
         USING *,0
STAR     CH     8,LIMT   ANY MORE AVAILABLE STORAGE
         BC     2,OVFL    NO, GO TO OVERFLOW
         STH    8,TEMP   SAVE POINTER FOR NEW LEFT
*                        END OF LIST
         AH     8,H2     POINT TO DATA OF AVAIL
         MVC    0(2,8),DATA  STORE DATA IN AVAIL
         STH    8,0(0,9) STORE LINK THAT POINTS
*                        TO RIGHT LINK OF AVAIL
         LH     9,TEMP   POINT TO NEW LEFT END OF
*                        THE LIST
         AH     8,H2     POINT TO NEW AVAILABLE
*                        STORAGE
         BCR    15,10    RETURN FROM SUBROUTINE
```

Figure L-8

Doubly Linked Lists

Reg 8---points to left link of available storage

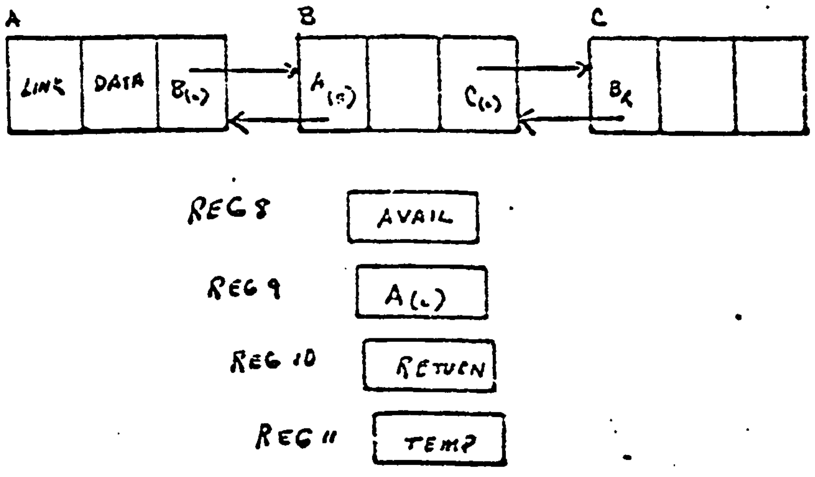
Reg 9---points to left link of an item in list

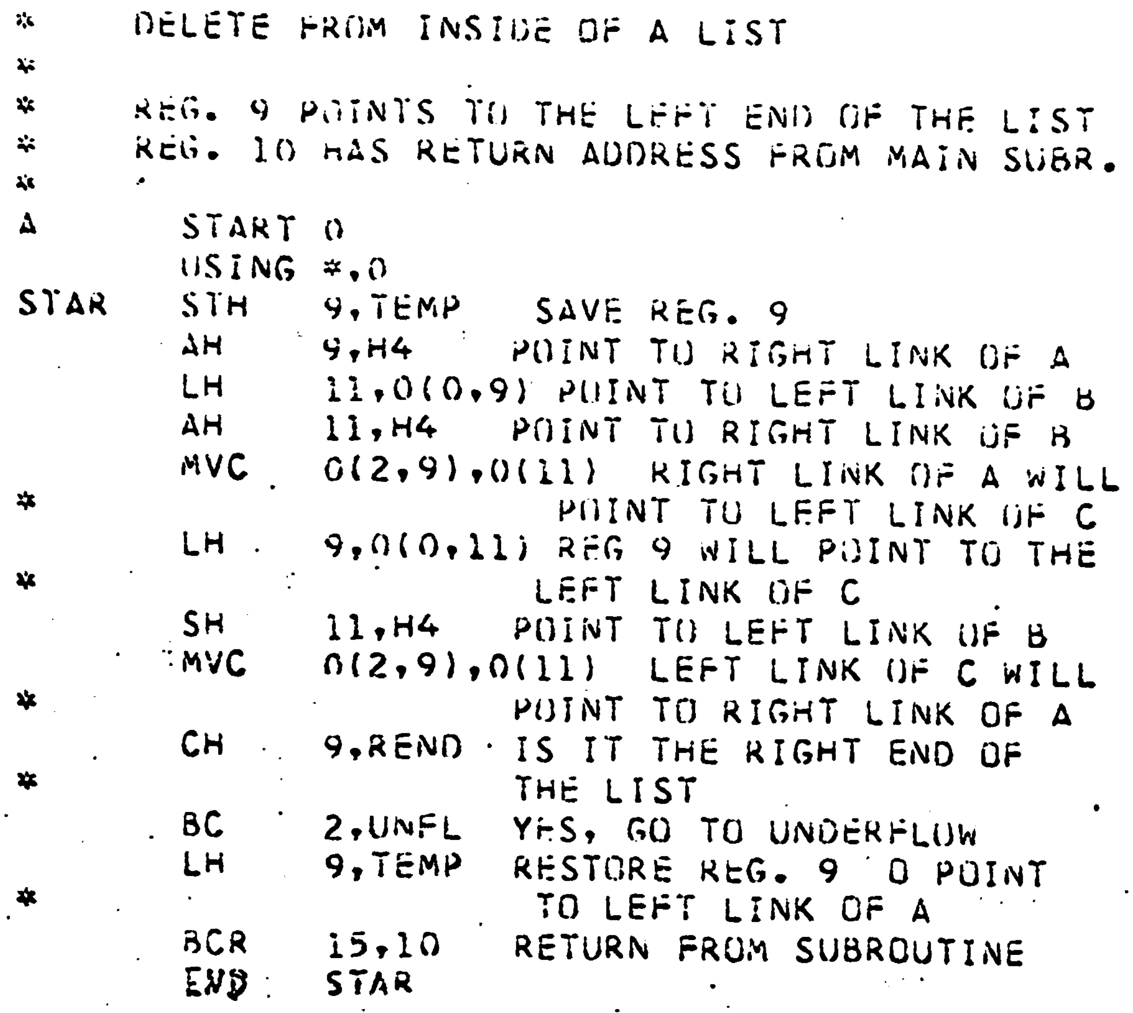
Reg 10---contains the return address from the subroutine

Insert Into List Between B and C (when Reg 9 points to left link of G)

```
*     INSERT INTO A LIST BETWEEN ITEMS B AND C .
*
*     REG. 8 POINTS TO AVAILABLE STORAGE
*     REG. 9 POINTS TO LEFT LINK OF ITEM C
*     REG. 10 HAS RETURN ADDRESS FROM MAIN SUBR.
*
A        START 0
         USING *,0
STAR     CH     8,LIMT   ANY MORE AVAILABLE STORAGE
         BC     2,OVFL    NO, GO TO OVERFLOW
         MVC    0(2,8),0(9)  STORE RIGHT LINK TO B
*                           INTO LEFT LINK OF AVAIL
         AH     8,H2     POINT TO DATA OF AVAIL
         MVC    0(2,8),DATA  STORE DATA IN AVAIL
         AH     8,H2  POINT TO RIGHT LINK OF AVAIL
         STH    9,0(0,8)  STORE LINK TO C INTO
*                           RIGHT LINK OF AVAIL
         AH     8,H2     POINT TO NEW AVAILABLE
*                         STORAGE
         MVC    0(2,9),0(8)  STORE LINK TO AVAIL
*                               INTO LEFT LINK OF C
         SH     8,H6     POINT TO LEFT LINK OF AVAL
         LH     9,0(0,8)  REG. 9 WILL POINT TO
*                           RIGHT LINK OF B
         STH    8,0(0,9) RIGHT LINK OF B WILL
*                         POINT TO LEFT LINK OF AVAL
         AH     8,H4     POINT TO RIGHT LINK OF AV.
         LH     9,0(0,8) REG. 9 WILL POINT TO
*                         LEFT LINK OF C
         AH     8,H2     POINT TO NEW AVAILABLE ST.
         BCR    15,10    RETURN FROM SUBROUTINE
         END    STAR
```

Delete From Inside of List





```
*    DELETE FROM INSIDE OF A LIST
*
*    REG. 9 POINTS TO THE LEFT END OF THE LIST
*    REG. 10 HAS RETURN ADDRESS FROM MAIN SUBR.
*
A      START 0
       USING *,0
STAR   STH   9,TEMP   SAVE REG. 9
       AH    9,H4    POINT TO RIGHT LINK OF A
       LH    11,0(0,9) POINT TO LEFT LINK OF B
       AH    11,H4   POINT TO RIGHT LINK OF B
       MVC   0(2,9),0(11)  RIGHT LINK OF A WILL
*                      POINT TO LEFT LINK OF C
       LH    9,0(0,11) REG 9 WILL POINT TO THE
*                     LEFT LINK OF C
       SH    11,H4   POINT TO LEFT LINK OF B
       MVC   0(2,9),0(11)  LEFT LINK OF C WILL
*                    POINT TO RIGHT LINK OF A
       CH    9,REND  IS IT THE RIGHT END OF
*                    THE LIST
       BC    2,UNFL  YES, GO TO UNDERFLOW
       LH    9,TEMP  RESTORE REG. 9 TO POINT
*                     TO LEFT LINK OF A
       BCR   15,10   RETURN FROM SUBROUTINE
       END   STAR
```

Figure L-9

Activities

1. Flowchart example #1 "Go Through Stack"
2. Flowchart "Add To Top of Stack"
3. Flowchart "Delete From Top of Stack"
4. Flowchart "Insert Into Stack Between B and C"
5. Flowchart "Add to Left of Doubly Linked List"
6. Flowchart "Delete From Inside of List"
7. Flowchart "Doubly Linked Lists"

Trees

Lists and strings are understood easier, developing the hierarchal structure idea with trees.

Lists**** ((((d)) (Q, R,), L, () (PTA))

The comma and parentheses become quite important in depicting atoms from the list.

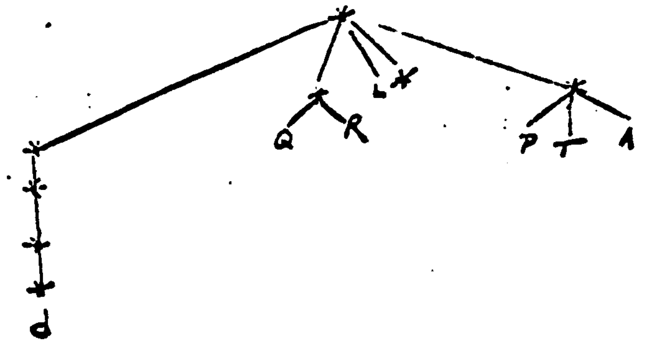


Figure T-1

The asterisks help to define the recursion. Here, the list contains the list (((d))) which consists of the list ((d)), which consists of list (d), which consists of the atom d. The asterisks indicate a blank or null.

This idea exists in the Dewey Decimal System for the libraries:

000-099	General Works
100-199	Philosophy, Psychology, Ethics
200-299	Religion and Mythology
300-399	Sociology (Economics, Civics, Education, Vocations)
400-499	Philology (Language, Dictionaries, Grammar)
500-599	Science
600-699	Useful Arts (Medicine, Engineering, Agriculture, Radio, Aviation, etc.)
900-999	History, Geography, Biography, Travel

Each of these ten main classes is broken up into more specialized fields. For example, class 600-699, Useful Arts, is broken into ten classifications including Medicine, Engineering, and Manufacturing. Each of these divisions is further subdivided.

This hierarchical tree idea exists in the paging scheme of this text. For example:

1.1

1.2

1.1.2

The idea also exists in set theory. For example it exists in nested sets:

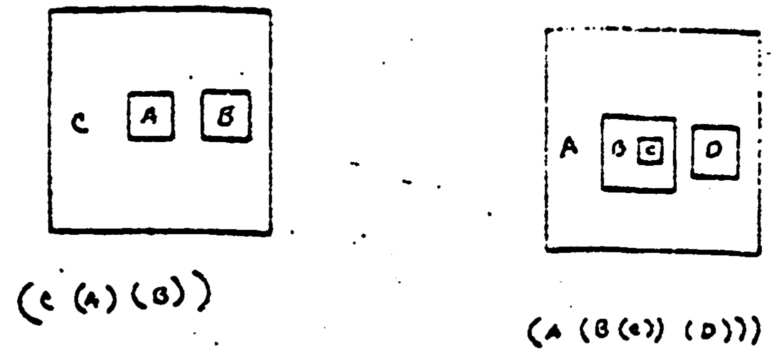


Figure T-2

A Combination of Sets and Trees

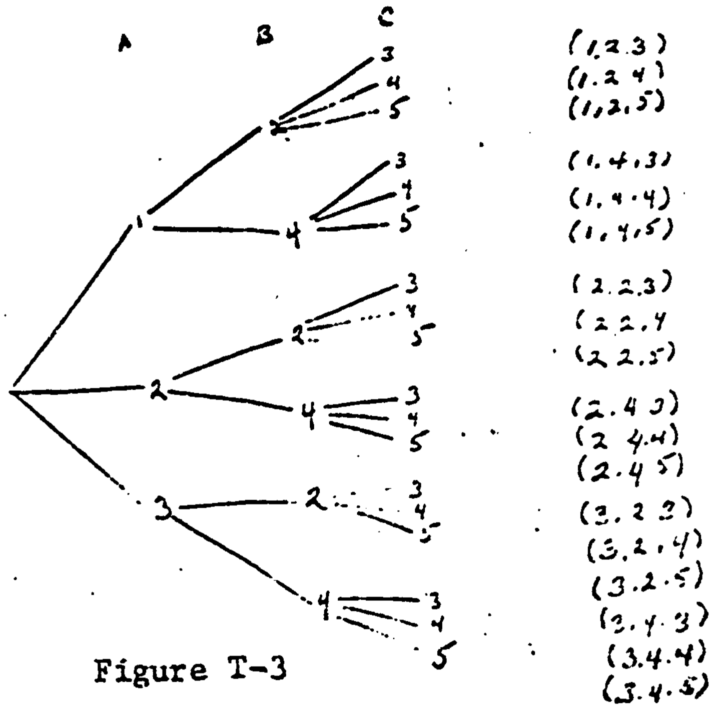


Figure T-3

The hierarchical idea allows us to combine sets and trees. For example:

Let A -- the set 123

Let B -- the set 242

Let C -- the set 345

With the help of a hierarchical tree, we can find A X B X C. We see that A X B X C consists of the ordered triplets to the right of the tree.

Tree Hierarchical Structures

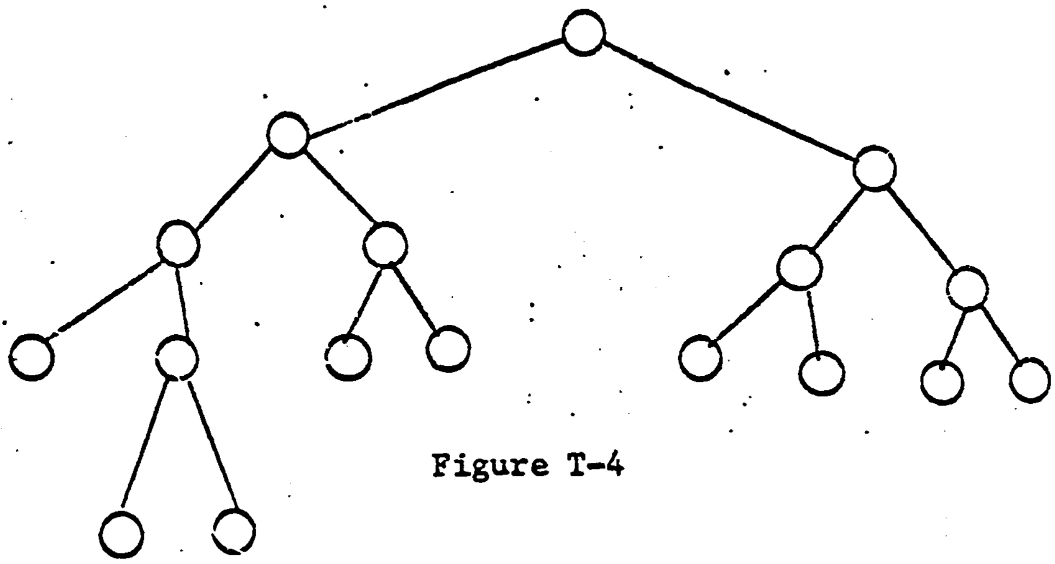


Figure T-4

The idea of a tree is presented here to form a concept of structure that might exist within core memory.

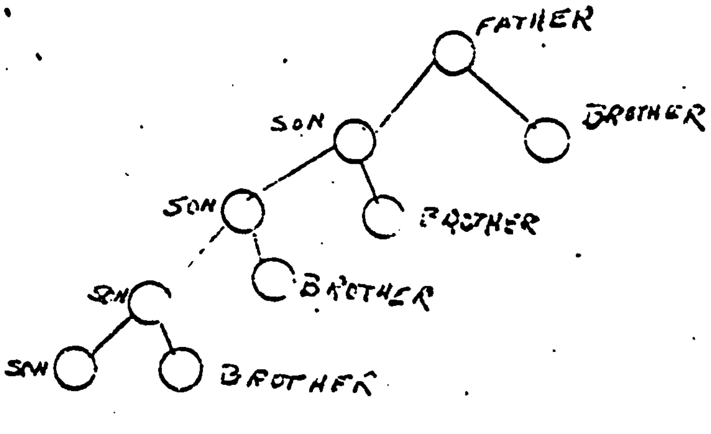


Figure T-5

2.3.1 Hierarchical Ring Structure

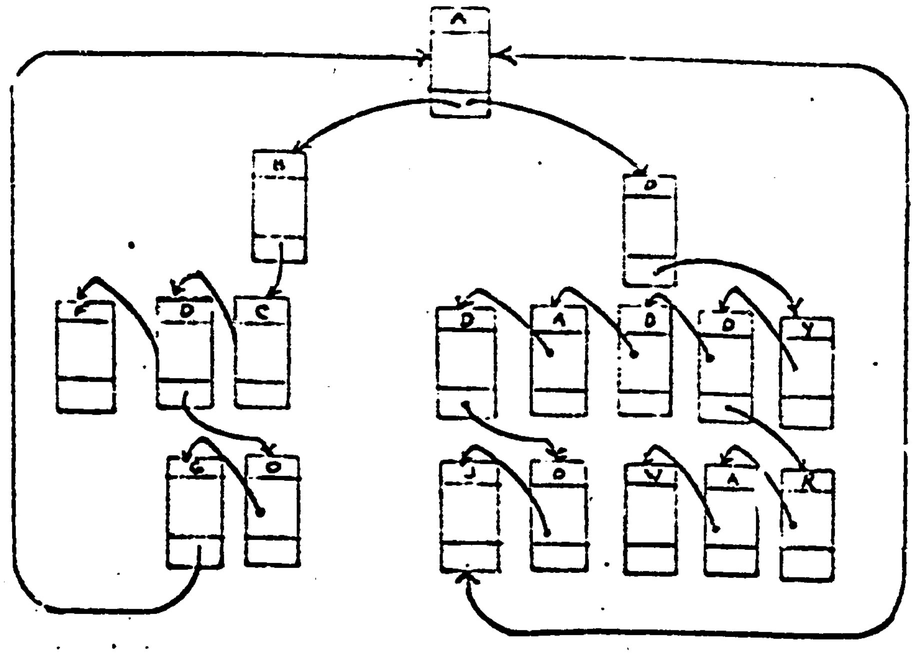


Figure H-1

The tree nodes are divided into three parts, which can be adjusted to fit most computers. This search procedure scans the descendants of the head (root) by following the chain of part two of each element (the pointer in part two). If a match is found on one level a branch can be made to the next level by means of the address of the pointer of part three of the element.

It is easy to insert a new node here because the chain can be broken and then the pointers can be changed to effect a new link.

This arrangement of the node, Figure H-1, amounts to the insertion of an atom in the first element of the node. The second part links to the brother on the same level, and the third part of the node links to the next structural level of descendancy.

The complete ring is made, which allows the return to the head of the list, and at the same time facilitates entering the ring at any point.

Algebriac Formulas

Algebriac formulas are better understood with the help of a tree hierarchy structure.

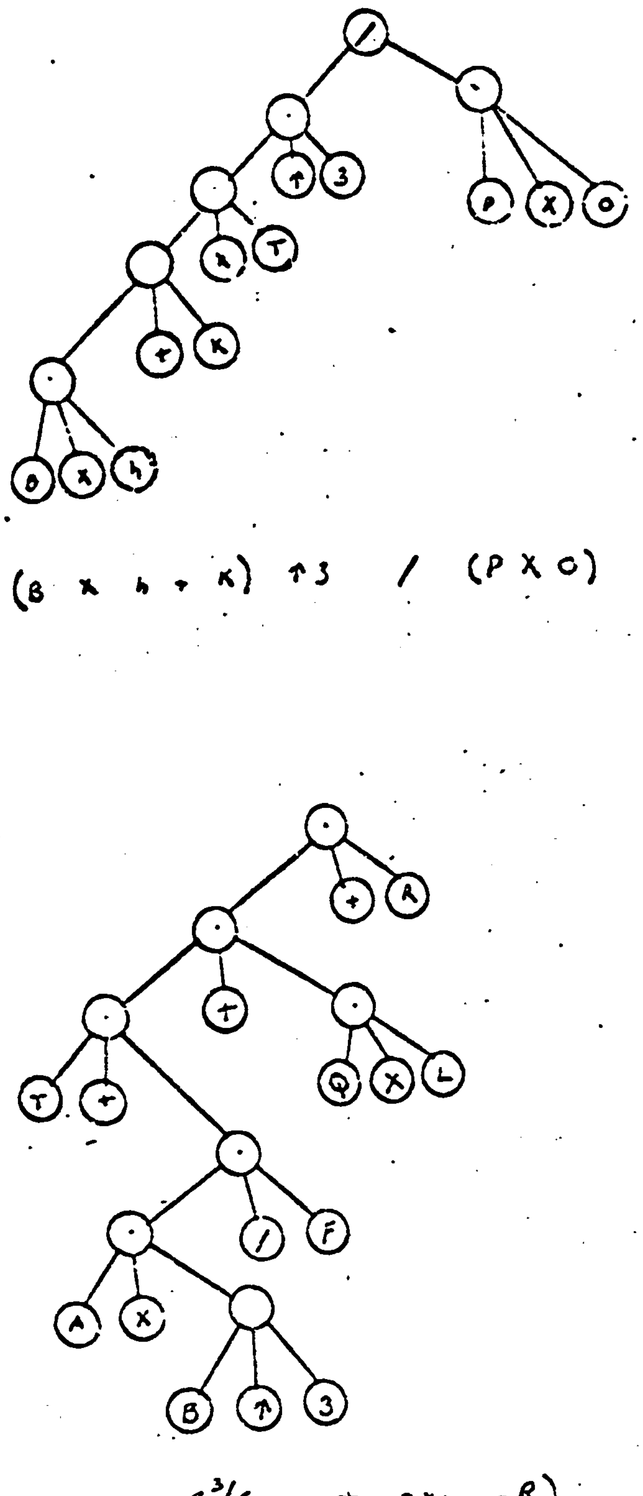


Figure H-2

Double Ring Link to Tree

This type of ring structure represents a double link type structure, which allows a much freer movement, but does take more space due to the pointers.

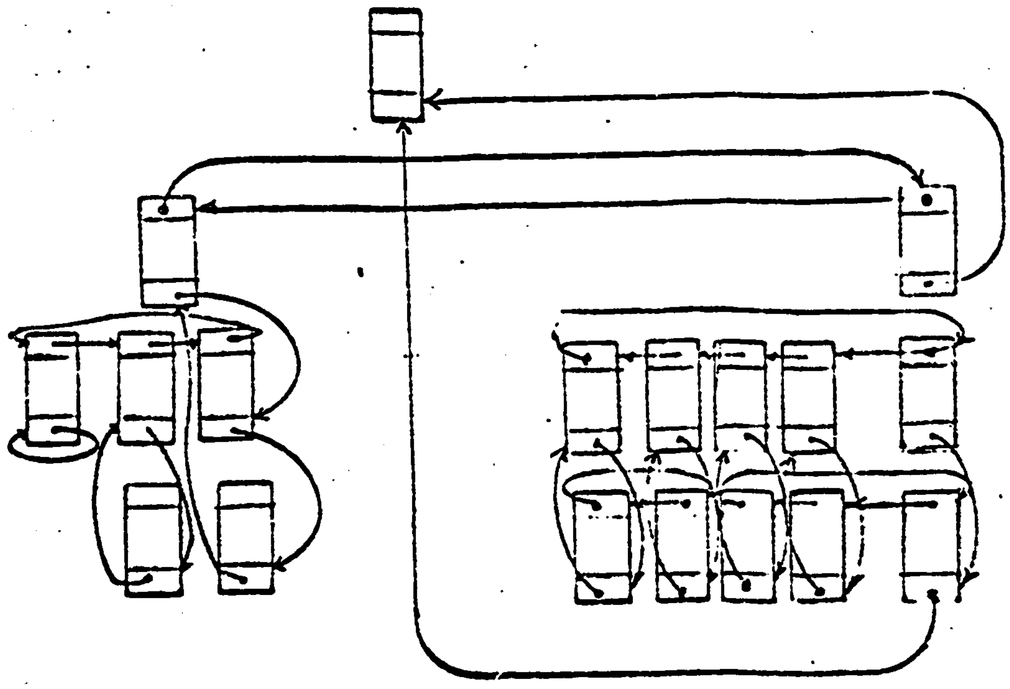


Figure H-3

Traversing Trees

Traverse a Tree in Preorder Sequence

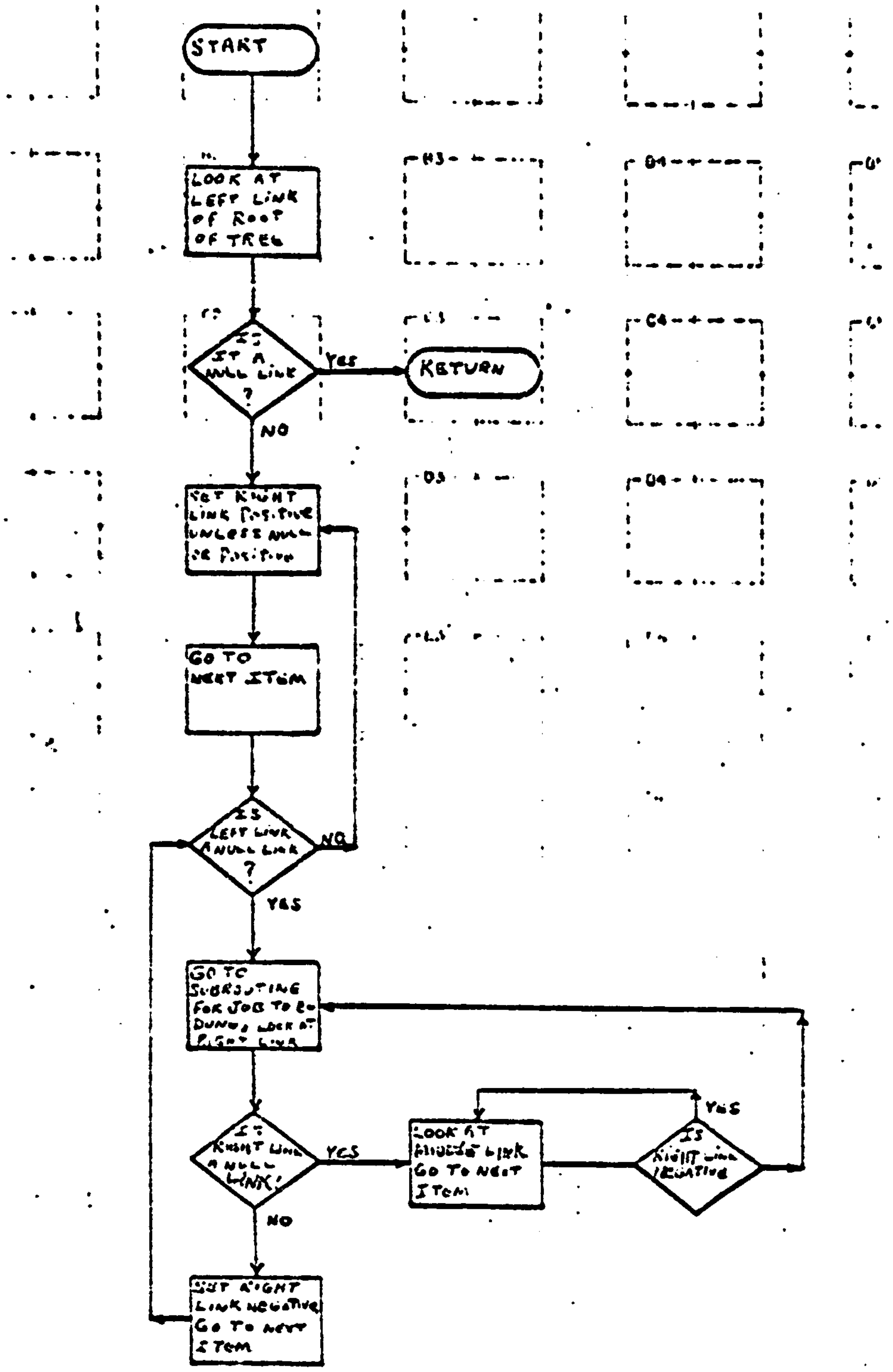

START
LOOK AT LEFT LINK OF ROOT OF TREE
IS IT A NULL LINK ?
YES
RETURN
NO
SET RIGHT LINK POSITIVE UNLESS NULL OR POSITIVE
GO TO NEXT ITEM
IS LEFT LINK A NULL LINK ?
NO
YES
GO TO SUBROUTINE FOR JOB TO BE DONE, LOOK AT RIGHT LINK
IS RIGHT LINK A NULL LINK?
YES
LOOK AT MIDDLE LINK GO TO NEXT ITEM
IS RIGHT LINK NEGATIVE
YES
NO
SET RIGHT LINK NEGATIVE GO TO NEXT ITEM

Figure TT-1

Preorder

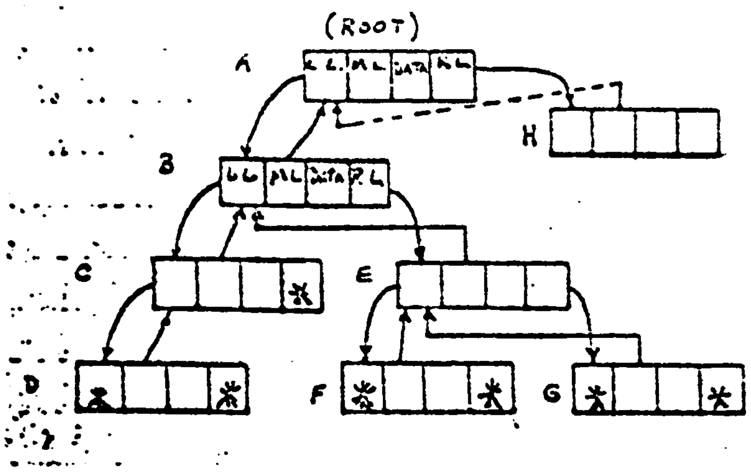


Figure TT-2

Traverse the Tree in Preorder

A, B, C, D, E, F, G, H

Left Link = 2 bytes

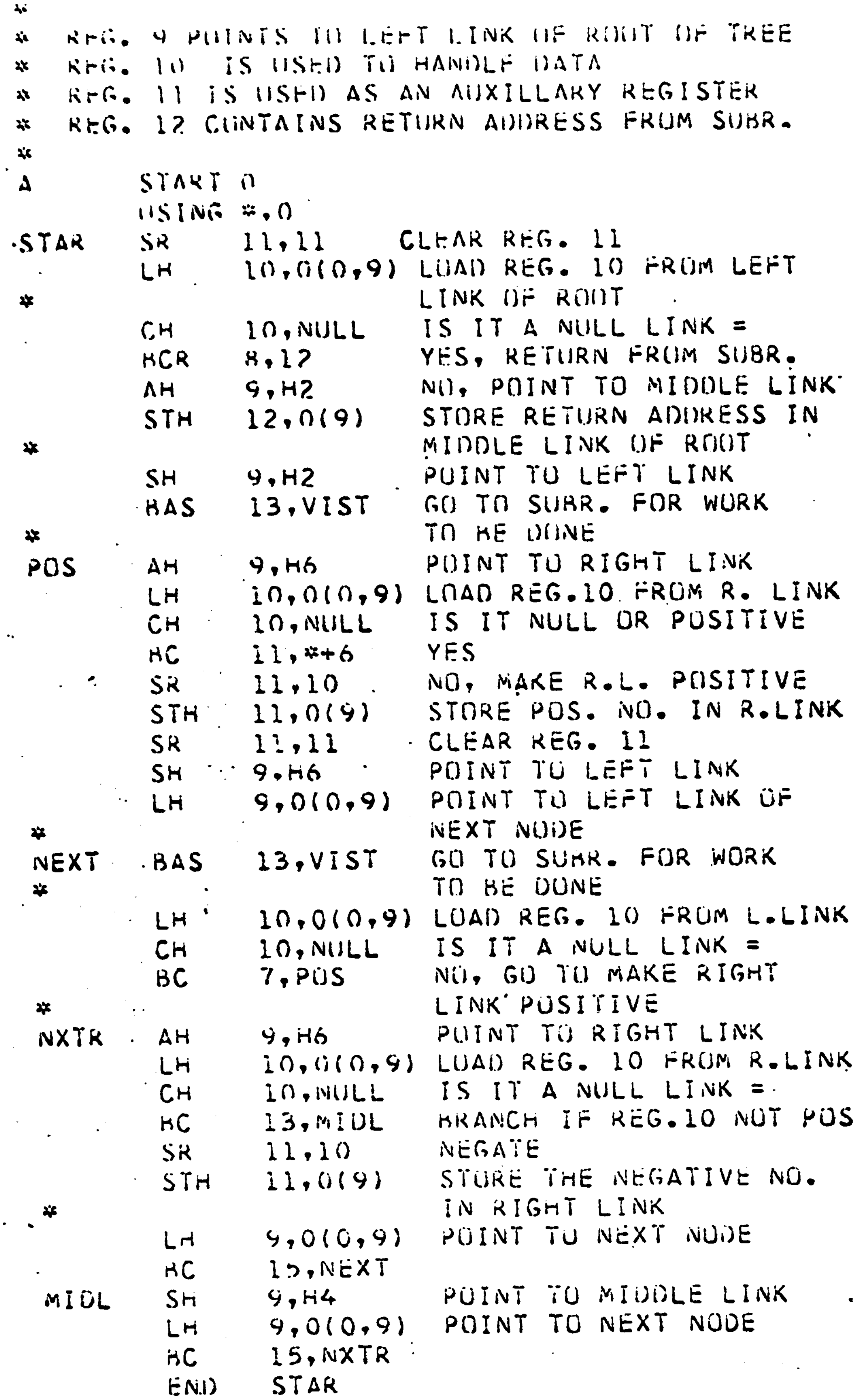
Right Link = 2 bytes

Middle Link = 2 bytes

Data = 2 bytes

Null Link = *

Traverse a Tree in Preorder



```
*
*    REG. 9 POINTS TO LEFT LINK OF ROOT OF TREE
*    REG. 10  IS USED TO HANDLE DATA
*    REG. 11 IS USED AS AN AUXILLARY REGISTER
*    REG. 12 CONTAINS RETURN ADDRESS FROM SUBR.
*
A        START 0
         USING *,0
STAR     SR     11,11       CLEAR REG. 11
         LH     10,0(0,9)  LOAD REG. 10 FROM LEFT
*                           LINK OF ROOT
         CH     10,NULL     IS IT A NULL LINK =
         BCR    8,12        YES, RETURN FROM SUBR.
         AH     9,H2        NO, POINT TO MIDDLE LINK
         STH    12,0(9)     STORE RETURN ADDRESS IN
*                           MIDDLE LINK OF ROOT
         SH     9,H2        POINT TO LEFT LINK
         BAS    13,VIST     GO TO SUBR. FOR WORK
*                           TO BE DONE
POS      AH     9,H6        POINT TO RIGHT LINK
         LH     10,0(0,9)  LOAD REG.10 FROM R. LINK
         CH     10,NULL     IS IT NULL OR POSITIVE
         BC     11,*+6      YES
         SR     11,10       NO, MAKE R.L. POSITIVE
         STH    11,0(9)     STORE POS. NO. IN R.LINK
         SR     11,11       CLEAR REG. 11
         SH     9,H6        POINT TO LEFT LINK
         LH     9,0(0,9)    POINT TO LEFT LINK OF
*                           NEXT NODE
NEXT     BAS    13,VIST     GO TO SUBR. FOR WORK
*                           TO BE DONE
         LH     10,0(0,9)  LOAD REG. 10 FROM L.LINK
         CH     10,NULL     IS IT A NULL LINK =
         BC     7,POS       NO, GO TO MAKE RIGHT
*                           LINK POSITIVE
NXTR     AH     9,H6        POINT TO RIGHT LINK
         LH     10,0(0,9)  LOAD REG. 10 FROM R.LINK
         CH     10,NULL     IS IT A NULL LINK =
         BC     13,MIDL     BRANCH IF REG.10 NOT POS
         SR     11,10       NEGATE
         STH    11,0(9)     STORE THE NEGATIVE NO.
*                           IN RIGHT LINK
         LH     9,0(0,9)    POINT TO NEXT NODE
         BC     15,NEXT
MIDL     SH     9,H4        POINT TO MIDDLE LINK
         LH     9,0(0,9)    POINT TO NEXT NODE
         BC     15,NXTR
         END    STAR
```

Figure TT-3

Traverse a Tree in Post Order

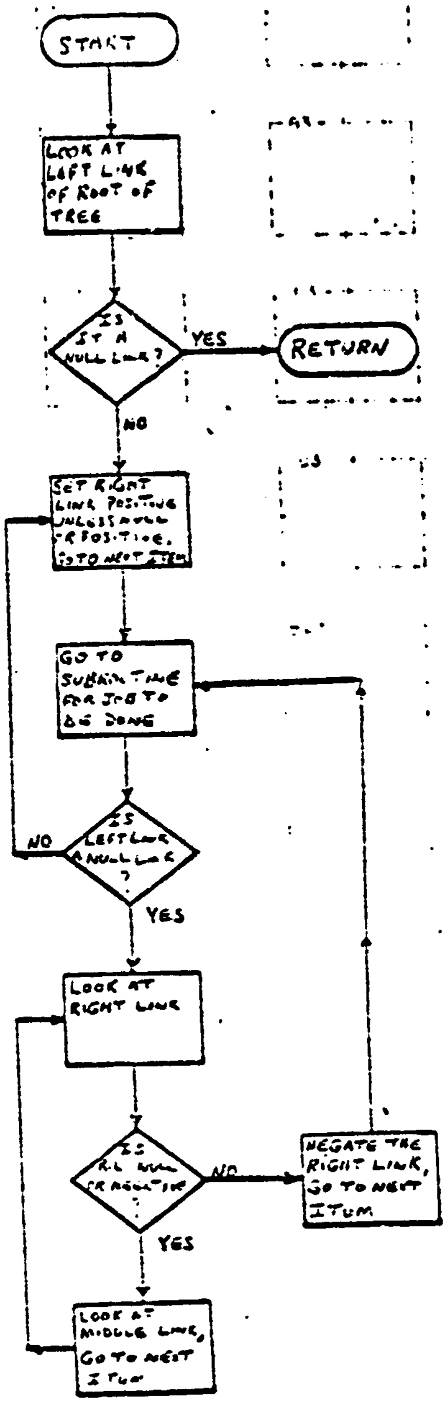

START
LOOK AT LEFT LINK OF ROOT OF TREE
IS IT A NULL LINK?
YES
RETURN
NO
SET RIGHT LINK POSITIVE UNLESS NULL OR POSITIVE. GO TO NEXT ITEM
GO TO SUBROUTINE FOR JOB TO BE DONE
IS LEFT LINK A NULL LINK ?
NO
YES
LOOK AT RIGHT LINK
IS R.L. NULL OR NEGATIVE ?
NO
NEGATE THE RIGHT LINK, GO TO NEXT ITUM
YES
LOOK AT MIDDLE LINK, GO TO NEXT ITUM

Figure TT-5

Post Order

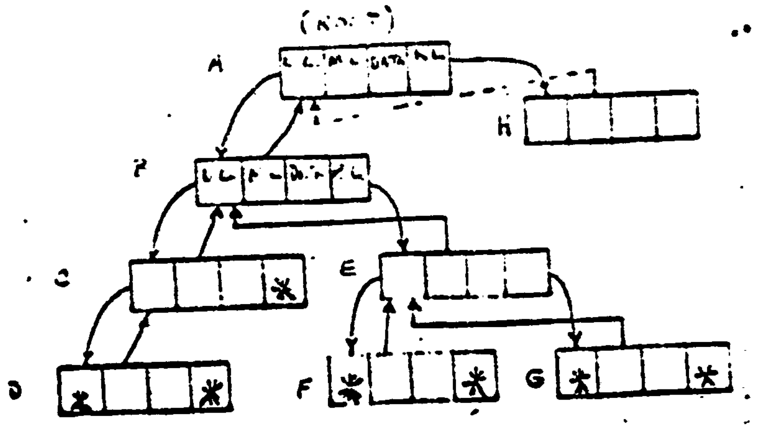
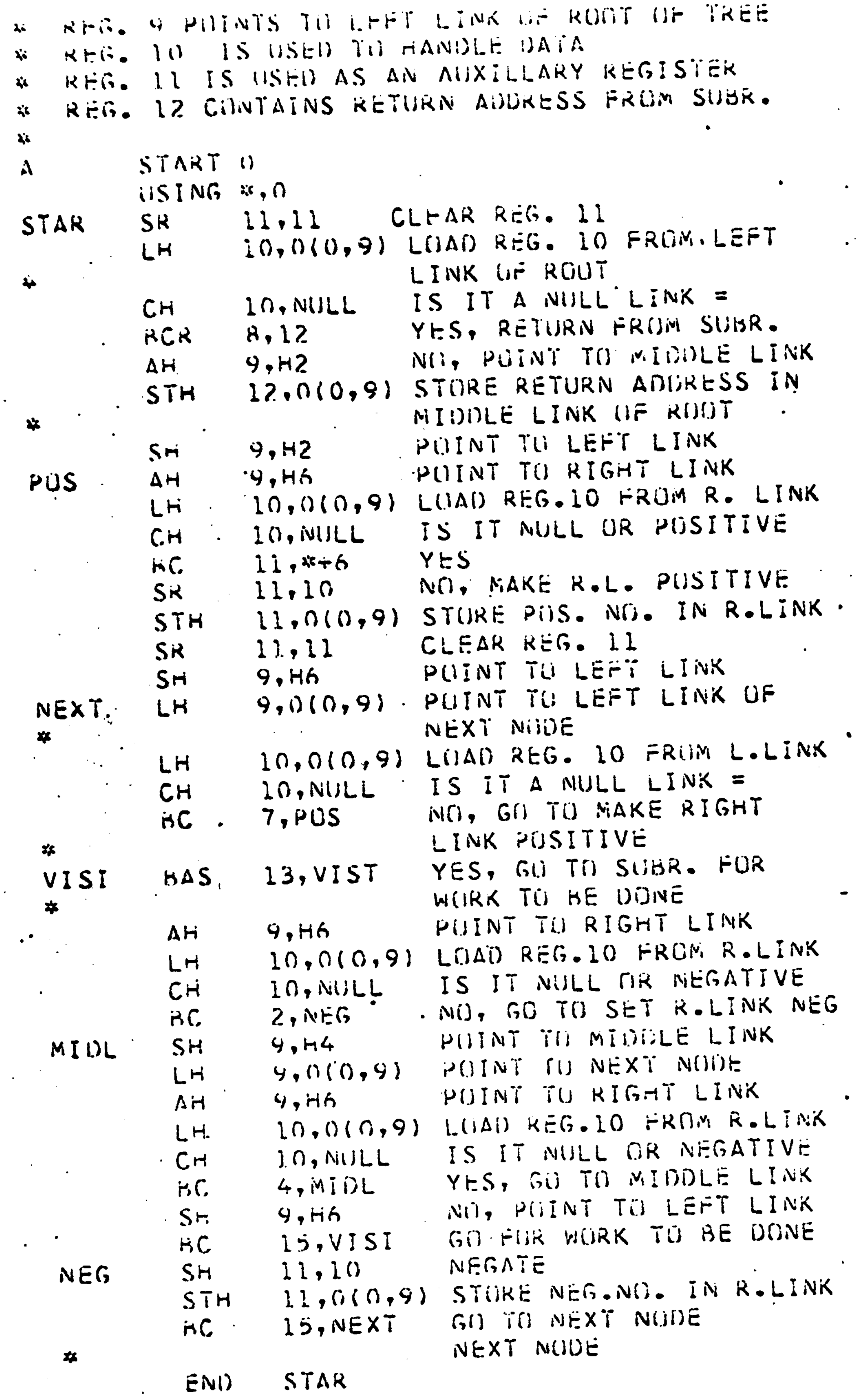


Figure TT-6

Traverse the Tree in Post Order

D, C, B, F, E, G, A, H

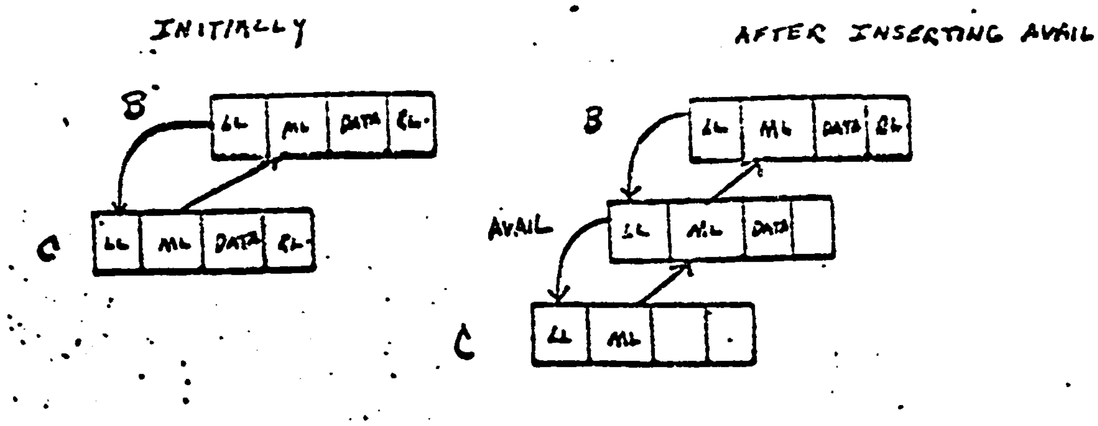
Traverse a Tree in Post Order

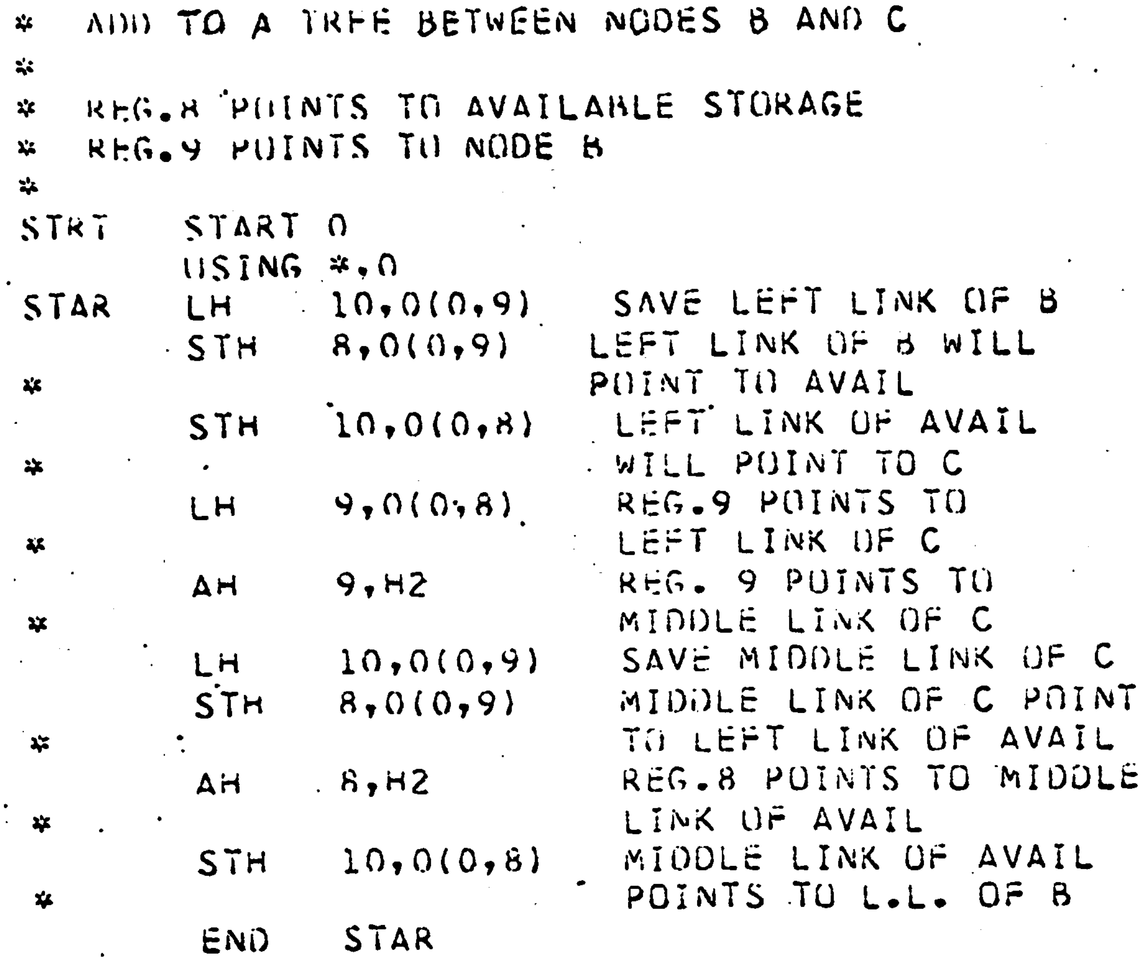


```
*    REG. 9 POINTS TO LEFT LINK OF ROOT OF TREE
*    REG. 10  IS USED TO HANDLE DATA
*    REG. 11 IS USED AS AN AUXILLARY REGISTER
*    REG. 12 CONTAINS RETURN ADDRESS FROM SUBR.
*
A        START 0
         USING *,0
STAR     SR    11,11      CLEAR REG. 11
         LH    10,0(0,9)  LOAD REG. 10 FROM LEFT
*                         LINK OF ROOT
         CH    10,NULL    IS IT A NULL LINK =
         BCR   8,12       YES, RETURN FROM SUBR.
         AH    9,H2       NO, POINT TO MIDDLE LINK
         STH   12,0(0,9)  STORE RETURN ADDRESS IN
*                         MIDDLE LINK OF ROOT
         SH    9,H2       POINT TO LEFT LINK
POS      AH    9,H6       POINT TO RIGHT LINK
         LH    10,0(0,9)  LOAD REG.10 FROM R. LINK
         CH    10,NULL    IS IT NULL OR POSITIVE
         BC    11,*+6     YES
         SR    11,10      NO, MAKE R.L. POSITIVE
         STH   11,0(0,9)  STORE POS. NO. IN R.LINK
         SR    11,11      CLEAR REG. 11
         SH    9,H6       POINT TO LEFT LINK
NEXT     LH    9,0(0,9)   POINT TO LEFT LINK OF
*                         NEXT NODE
         LH    10,0(0,9)  LOAD REG. 10 FROM L.LINK
         CH    10,NULL    IS IT A NULL LINK =
         BC    7,POS      NO, GO TO MAKE RIGHT
*                         LINK POSITIVE
VISI     BAS   13,VIST    YES, GO TO SUBR. FOR
*                         WORK TO BE DONE
         AH    9,H6       POINT TO RIGHT LINK
         LH    10,0(0,9)  LOAD REG.10 FROM R.LINK
         CH    10,NULL    IS IT NULL OR NEGATIVE
         BC    2,NEG      NO, GO TO SET R.LINK NEG
MIDL     SH    9,H4       POINT TO MIDDLE LINK
         LH    9,0(0,9)   POINT TO NEXT NODE
         AH    9,H6       POINT TO RIGHT LINK
         LH    10,0(0,9)  LOAD REG.10 FROM R.LINK
         CH    10,NULL    IS IT NULL OR NEGATIVE
         BC    4,MIDL     YES, GO TO MIDDLE LINK
         SH    9,H6       NO, POINT TO LEFT LINK
         BC    15,VISI    GO FOR WORK TO BE DONE
NEG      SH    11,10      NEGATE
         STH   11,0(0,9)  STORE NEG.NO. IN R.LINK
         BC    15,NEXT    GO TO NEXT NODE
                          NEXT NODE
*
         END   STAR
```

Figure TT-8

Add to a Tree Between B and C

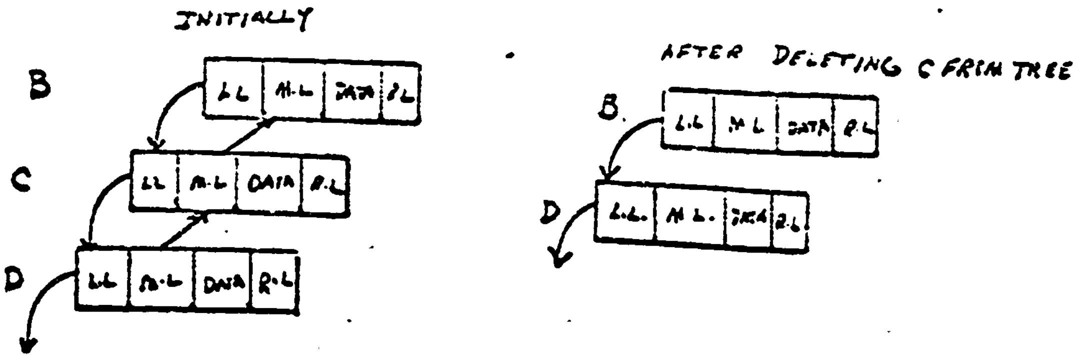




```
*   ADD TO A TREE BETWEEN NODES B AND C
*
*   REG.8 POINTS TO AVAILABLE STORAGE
*   REG.9 POINTS TO NODE B
*
STRT     START 0
         USING *,0
STAR     LH     10,0(0,9)    SAVE LEFT LINK OF B
         STH    8,0(0,9)    LEFT LINK OF B WILL
*                           POINT TO AVAIL
         STH    10,0(0,8)    LEFT LINK OF AVAIL
*                            WILL POINT TO C
         LH     9,0(0,8)     REG.9 POINTS TO
*                            LEFT LINK OF C
         AH     9,H2         REG. 9 POINTS TO
*                            MIDDLE LINK OF C
         LH     10,0(0,9)    SAVE MIDDLE LINK OF C
         STH    8,0(0,9)     MIDDLE LINK OF C POINT
*                            TO LEFT LINK OF AVAIL
         AH     8,H2         REG.8 POINTS TO MIDDLE
*                            LINK OF AVAIL
         STH    10,0(0,8)    MIDDLE LINK OF AVAIL
*                            POINTS TO L.L. OF B
         END    STAR
```

Figure TT-9

Delete Node C From A Tree



```
*     DELETE NODE C FROM A TREE
*
*     REG.9 POINTS TO LEFT LINK OF B
*
STRT    START 0
        USING *,0
STAR    LH      8,0(0,9)      REG.8 POINTS TO L.L. C
        MVC     0(2,9),0(8)    L.L. OF B WILL POINT
*                             TO LEFT LINK OF D
        LH      9,0(0,8)      REG.9 POINTS TO L.L. D
        AH      8,H2          REG.8 POINTS TO M.L. C
        AH      9,H2          REG.9 POINTS TO M.L. D
        MVC     0(2,9),0(8)    M.L. OF D POINTS TO
*                             LEFT LINK OF B
        END     STAR
```

Figure TT-10

2.3.3 STRUCTURED TREE DIRECTORY

This section, which includes Random Organization, and the complete File Section, which follows, are an integral part of an information management system.

The directory, files and type of structure, which is chosen for each, will have a great bearing on the speed and efficiency of the system. In the Swanson study, one approach was used. These two parts of the book will allow us to widen our area of selection of approaches we might choose.

We have developed the hierarchic structure concept thus far in the book. Now we will make use of it and others in developing index systems to files for search and retrieval purposes. As with the Swanson study, these methods are slanted toward disk systems.

2.3.3.1 STRUCTURED TREE DICTIONARY

It is important to note that each node (leaf) represents a complete record in the system. The use of three letters for a key is used here as an example. Actually a three-level tree can accommodate several million keys, if there are enough characters in the addressing method. Addressing method here means the KEY/ADDRESS/LIST LENGTH, which amounts to a multiword method.

The first level of the tree is maintained in core storage, and the other two levels are maintained in auxiliary storage. A typical disk notation is used in which T12 means track 2 of disk 1.

The key fragments across the bottom of the tree stand for complete names, but were truncated in order to form a fixed length key format. This fixed length format could be imposed by the hardware or could be the choice of the programmer. Any number of letters could be used, but as mentioned earlier, the number of letters in the key affects the number of records possible. Three were used here for the sake of brevity.

The search format for the tree is the one mentioned above:

NAME/ADDRESS/LIST LENGTH

The name, address and list length exist in auxiliary storage, and the addressing scheme proceeds from low to high as do the IBM 360 computers.

Level 1 resides in core storage, and it directs us to level 2, which resides in auxiliary storage. This address at level 2 is a track address. The 0 denotes that it is track 0 and 1 denotes that cylinder 1 is used. The highest alphabetic letter here is denoted by COW/T12/*, which signifies that COW marks an upper limit for this section of the tree. The asterisk denotes that there is no list length here, since the branch is not made to the actual list as yet. Level 2 then takes us to T12, which resides on level 3. This level contains the actual lists. T12 takes us to track 2 of cylinder 1. This level contains COW, which is the name part of the address we want. The actual address and length of the record is given here.

The process of decoding this type of dictionary proceeds by the following manner. For example, assume that we have the word COWBOY. The first three letters will be truncated (COW). This key is then compared to the dictionary file in core storage. It is smaller than HOT so COW is selected. This key takes us to track T10, track 0, disk 1, and then proceeds to compare at this level. COW is larger than CAM, but it does match COW.

This causes a branch to track 2 of disk 1 (T12), where a match is made. The link address takes us to the actual address of COWBOY, and we are given the length of the record that resides at that location.

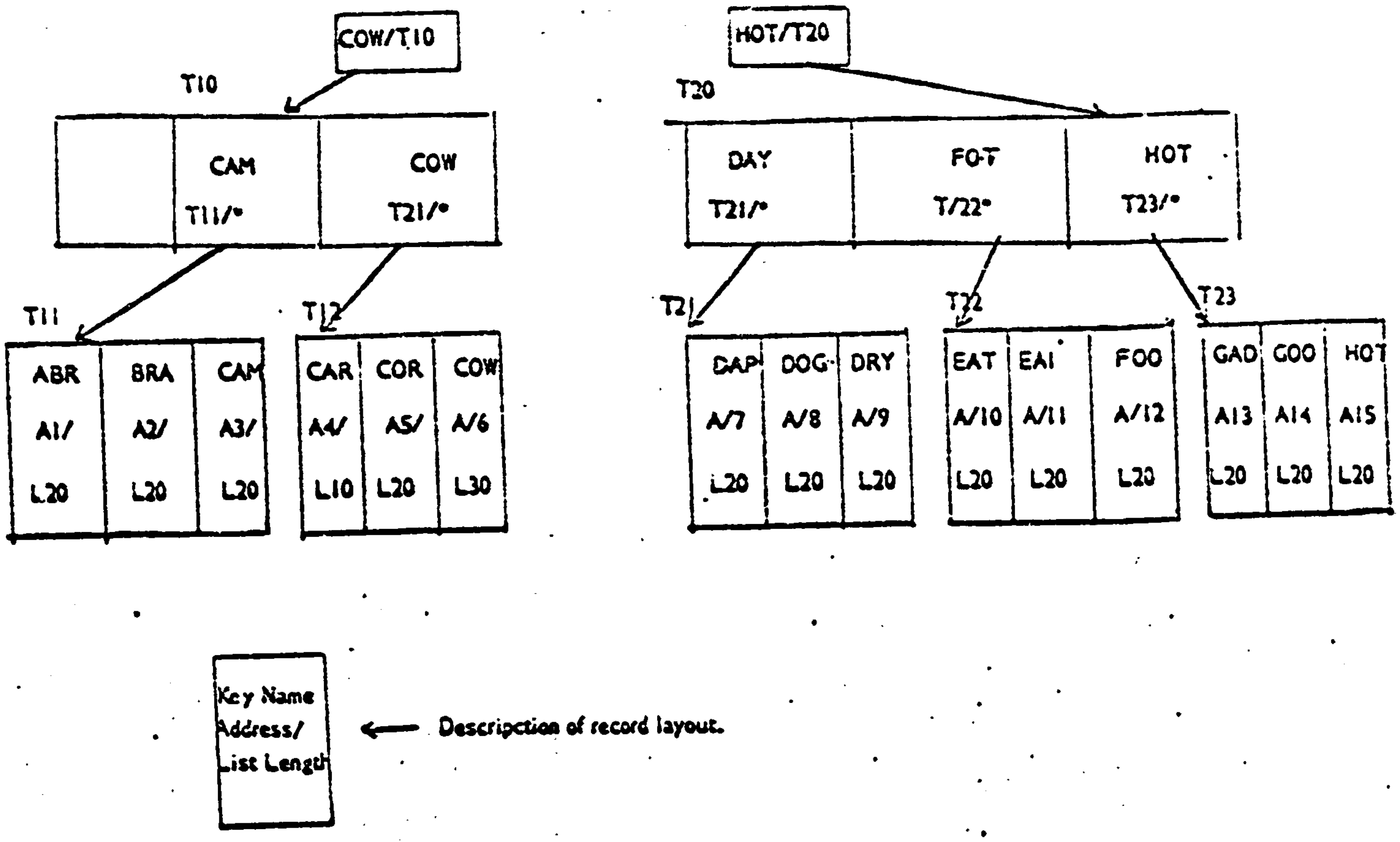


Figure STD-1

2.4 FILES

The file organization can be divided into three major groups:

1. Sequential File Organization
2. Inverted Files (or inverted lists)
3. String structures

The other types of organizational methods listed in this section can be considered to be combinations of the basic three methods.

A slight emphasis is placed on the Inverted File for reasons of comparison to the RPG methods which occur later in the book.

Figure F.1. depicts the structural organization from the most common at the base, Sequential Organization, to the more sophisticated and complex at the top.

The idea of file partitioning is introduced in this section. The partitioning concept is intended to make access to the disk records faster. The two basic methods of partitioning we will use are Inverted Lists and Multiple Threaded Lists. Finally, we will introduce combinations of these methods, and develop a Page Partition Multilist.

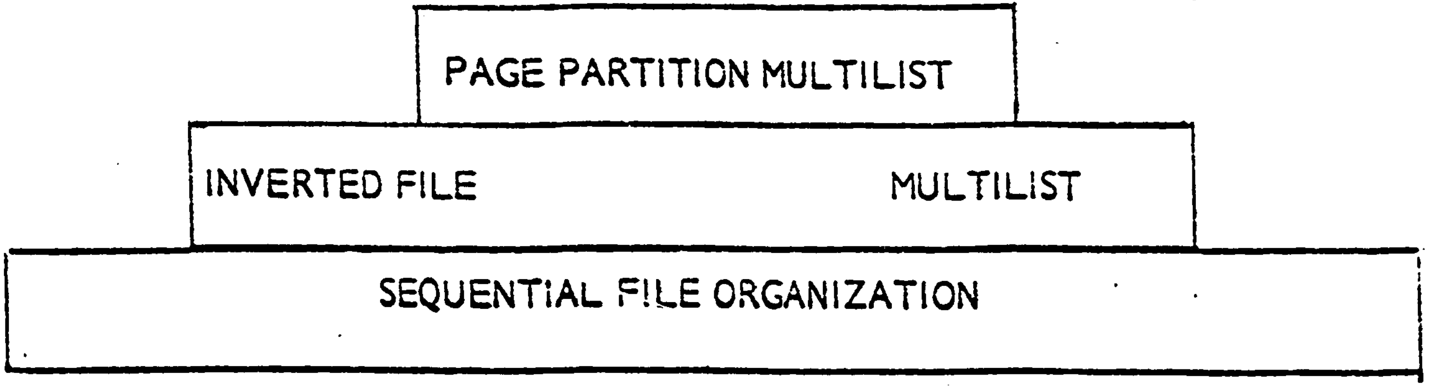


Figure F-1

The Inverted File and the Multilist exist at the extremes of a continuum as depicted by Figure F.1. They represent opposing or contrasting methods of file organization for search and retrieval purposes. The other methods represented here were developed as faster methods were needed. The Multilist and Page Partitioned Multilist evolved as a result of the use of file structures in the auxiliary and core storage. As will be seen in a later part of the book, the Virtual Memory techniques evolved at a later date.

2.4.1 SEQUENTIAL FILES

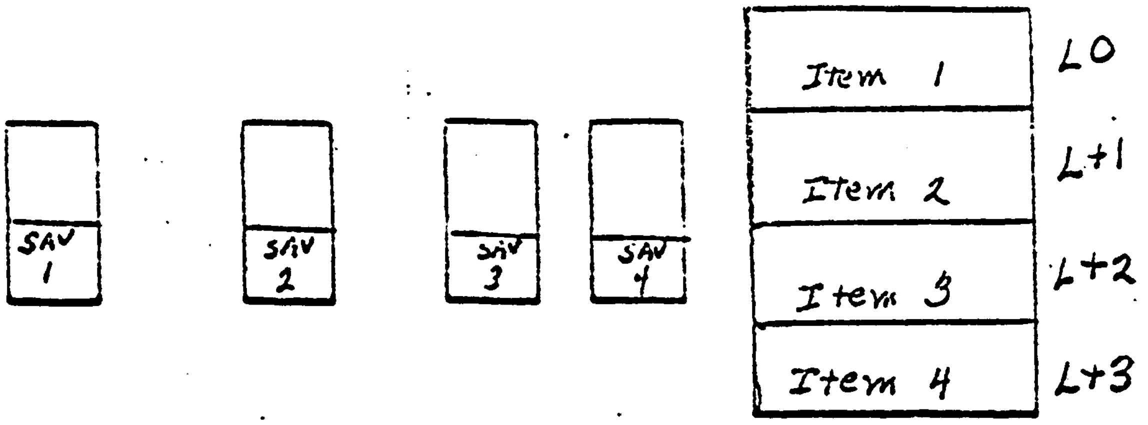


Figure F-2

Sequential date organization is the most common type used. For example, an inventory file is made up of many records, and each record in turn contains many fields:

Stock number

Unit of measure

On-hand quantity

Material description

On-order-quantity

The relationship of these fields of the record is that they do relate to a particular stock item. An inventory record for a hand saw would have a stock number for hand saws, a unit of measure, a material description of hand saws, and a measure for which the information about on-hand, on-order and reorder quantities for hand saws if kept.

The sequential file has certain advantages in that there is a fast access for each relationship. There is a disadvantage when a file is to be searched until a record having a particular key is found. This requires an examination of the first record, if the key is wrong, the next record, etc. This process goes on until the correct record is found. The updating process is also rather difficult with a sequential file. If the new record is shorter or longer than the original record in the file, the adjacent records may be affected or destroyed. The update process then becomes very costly when there is only one record to update. The update process is generally used when there are a number of records to update.

If we have a file of records, it becomes necessary to discover certain features that exist in the file. For example, if a saw is sold, the inventory record for saws needs to be isolated so that the value of the on-hand quantity field can be reduced.

The common attribute here is called a key. By selecting a different key for a file and sorting on the basis of that particular key, the sequence of records in the file may change, but we can obtain the desired information.

It is not necessary to store records with keys. There are times when the sequential arrangement is based solely on the arrival within the system.

Activities

Linkages

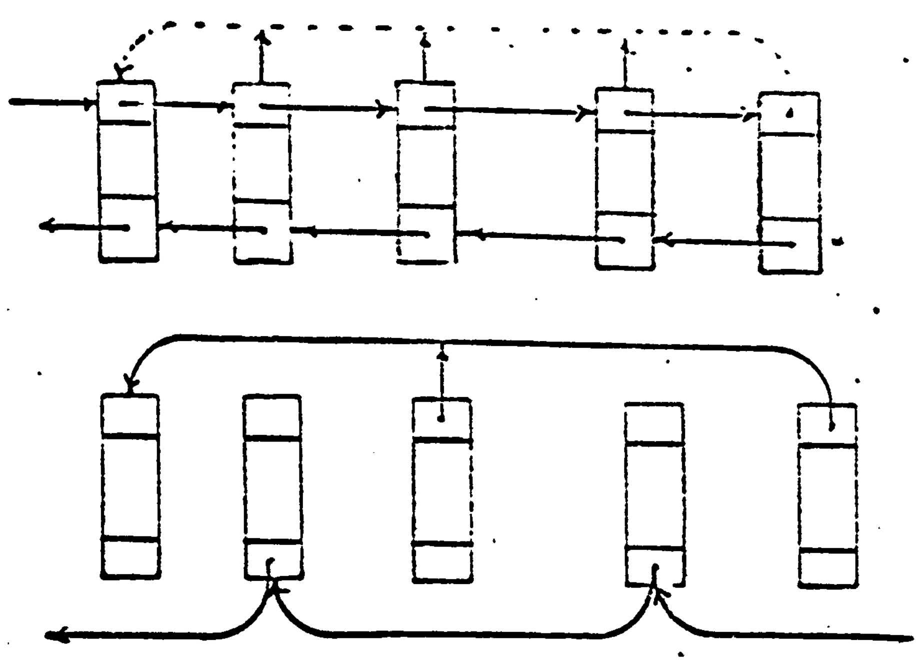
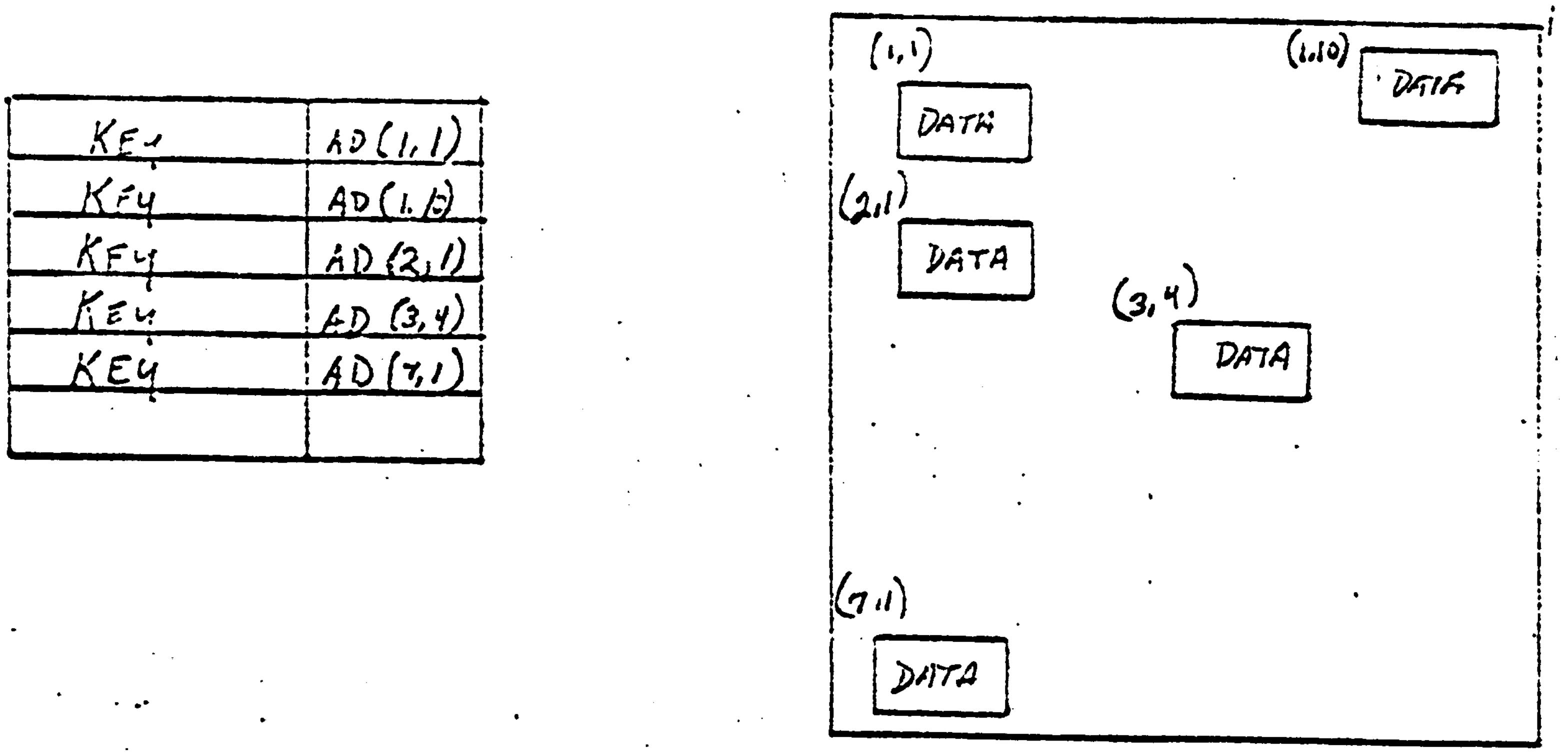


Figure A-4

Both these methods can be used for the string structure of a file.

1. Compare them to the Swanson Study.
2. Compare them to the linkages in Section II of Data Structures.
3. Where do the actual linkage processes take place, in core or on disk?

2.4.2 INVERTED FILES



KEY	AD(1,1)
KEY	AD(1,10)
KEY	AD(2,1)
KEY	AD(3,4)
KEY	AD(7,1)

Figure IF-1

The above example IF-1 is a drawing of the Inverted File structure taken to its ultimate, because the list has a length of 1. This type of structure requires the index to take up as much or more room as the list. The pointer AD(1,1) is a disk address to take us to track 1 of cylinder 1 to a particular location, which has a list length of 1.

A comparison could be made between State and City to determine which ones belong together. This could be done without linking to the list, which means a comparison of the indexes.

The main disadvantage to this method is the large directories needed. It works best as a partially inverted file combined with a random method, such as a hierarchical tree method or sequential method. This makes it possible to invert on just a few keys.

2.4.2.1 CREATION, MAINTENANCE and DELETION OF KEYS

If there is a master file, an inverted file may be created from it. This entails searching through the master file, making a directory from the selected keys, and applies to both the inverted file and the multilist file.

Every time a record is added to the master file, it becomes necessary to update the directories it pertains to.

The Inverted File Algorithm:

1. After the record is entered in the master file, prepare it for entry into the inverted file.
2. Assign the auxiliary address AAD, which may entail some form of the list of available space.
3. Key n in the directory is decoded to the variable length inverted file index.
4. Place Key n in its proper sequence into the directory; if the space is exhausted, use a link to the next page or block.
5. Add one to the list length in the index.
6. Continue steps 3 to 5 for all keys of the new record.
7. Store the new auxiliary address at AAD.

Deletion of Keys

The deletion of keys causes no problem, as far as overflow is concerned, since there is a reduction of space being used.

The Deletion algorithm:

1. Retrieve record from auxiliary storage and delete Key n.

2. Decode Key n in the proper directory of the inverted file list.
3. Delete the address from the list (this address is the address AAD residing in the inverted file list).
4. Repeat steps 2 and 3 until all keys of the record have been deleted that are required to be deleted.

Deletion of Records

The Record Deletion algorithm:

1. Transfer record from auxiliary storage location to core.
2. Set the record delete bit.
3. Decode every key of the record in the index and remove the record address (AAD) from all inverted file lists.
4. Decrement each file list, which has been affected by 1.

Addition of Keys

A problem arises here pertaining to record length. After the addition of a key, it is possible that the track could overflow. If so, the record is deleted and a link is inserted to another track where this whole record is inserted.

The Addition of Keys algorithm:

1. Decode the Key n in the proper directory to the variable length inverted file.
2. Determine the proper sequence, insert AAD of the record of Key A, and add 1 to the list length.
3. Continue steps 1 and 2 for all keys to be added.
4. Transfer the record from AAD (auxiliary address), and

add new keys to record.

5. If the record doesn't cause overflow of the track after packing it, return it to AAD.
6. If the record causes overflow, insert a link to another track, and place the whole record there.

2.4.2.2 LOGICAL OPERATOR "OR"

The logical operator "OR" effects a union of Directory A and Directory B. A key that resides in either Directory A or Directory B will be included in Directory C, and if the key appears in both directories, it appears only once in Directory C.

$$C = A \text{ OR } B$$

The Logical Operator "OR" algorithm:

1. Set 3 index registers (one register each for A, B, and C) to 1.
2. Compare unit 1 of A to unit 1 of B (the keys). If the key of A is smaller than the key of B, move the key of A to Directory C and increment index A and index C. If the key of A is equal to the key of B, move the key of A to Directory C and increment index A and index C. If the key of A is larger than the key of B, move the key of A to Directory C and increment index A and index C.
3. Repeat step 2 until Directories A and B have both been exhausted.
4. Stop.

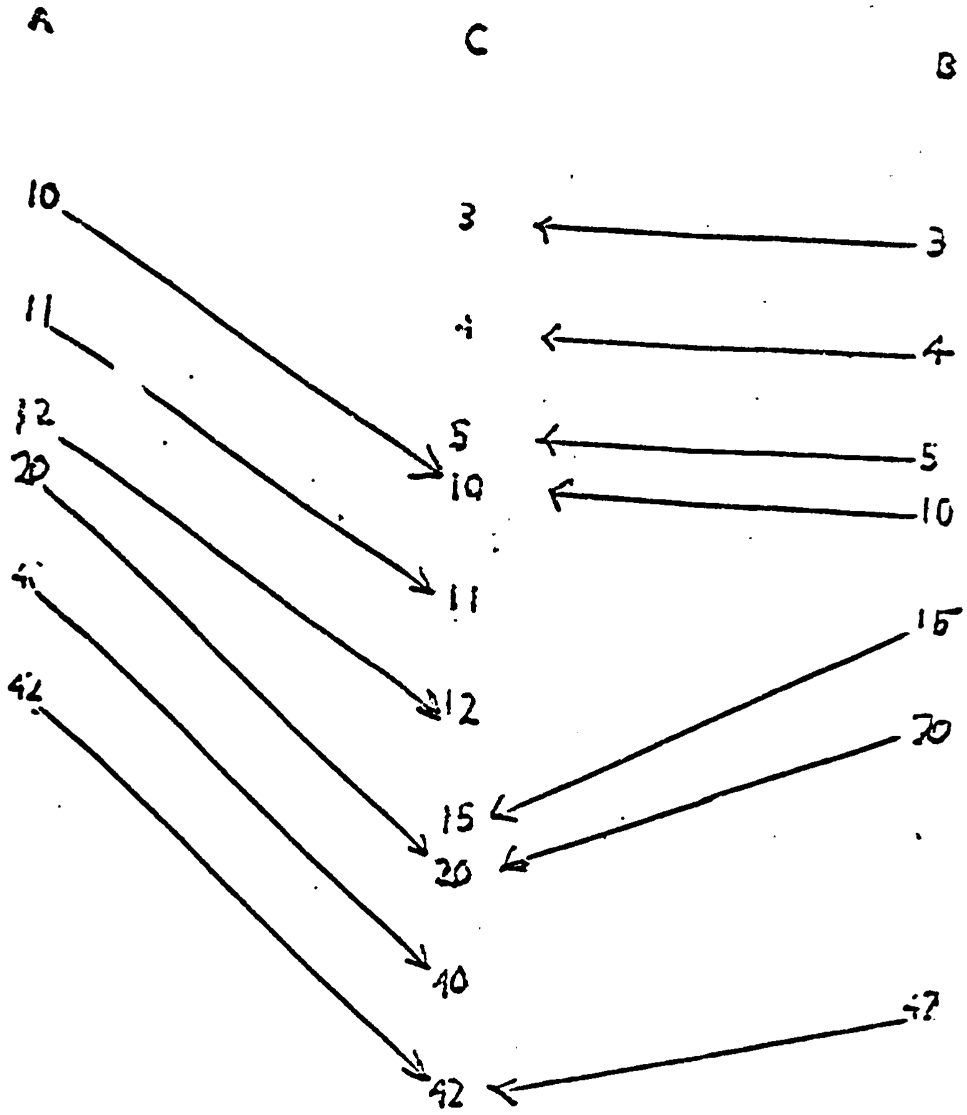
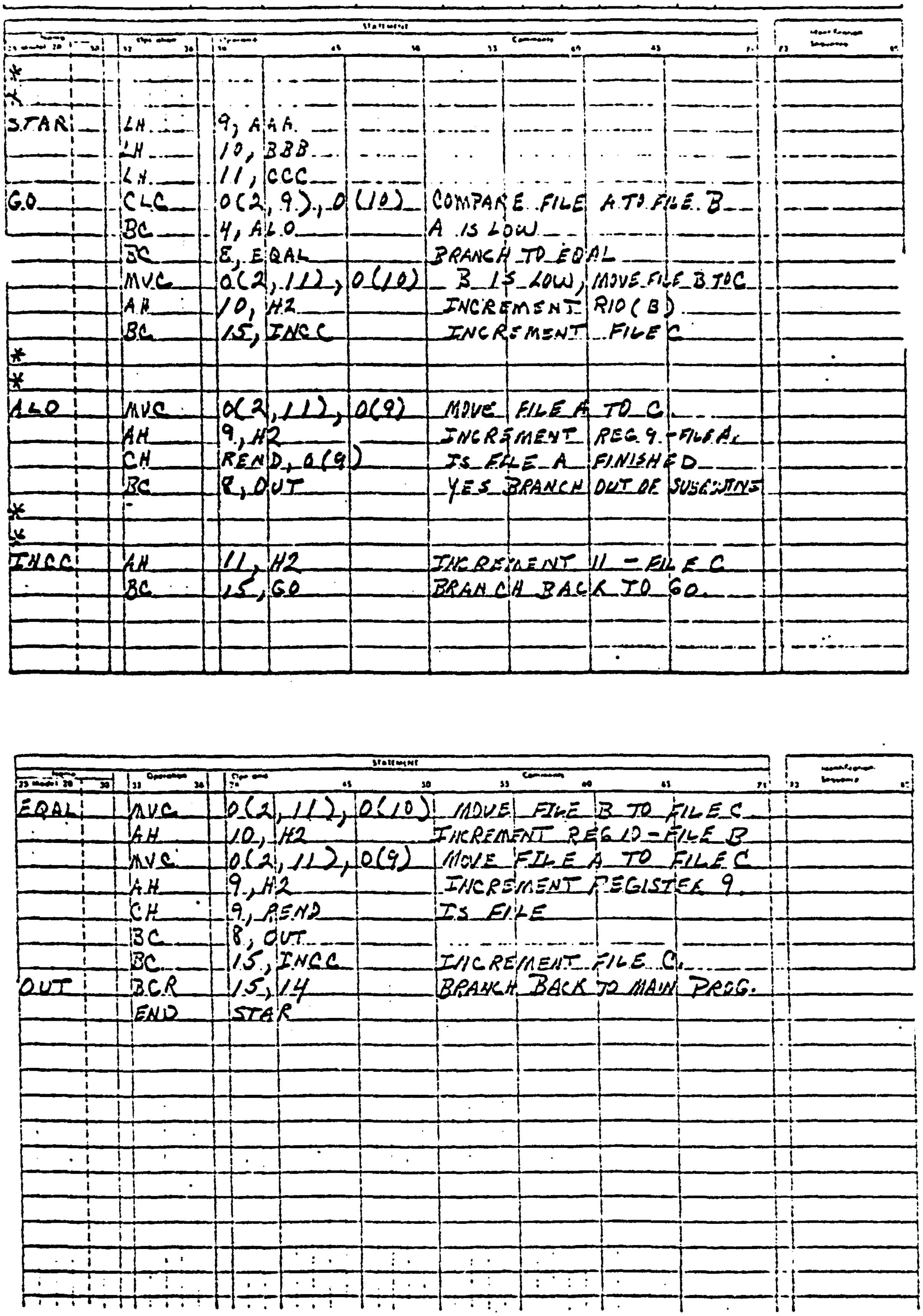


Figure IF-2

C = A "OR" B



```
*
*
STAR     LH     9,AAA
         LH     10,BBB
         LH     11,CCC
GO       CLC    0(2,9),0(10)    COMPARE FILE A TO FILE B
         BC     4,ALO           A IS LOW
         BC     E,EQAL          BRANCH TO EQAL
         MVC    0(2,11),0(10)   B IS LOW, MOVE FILE B TO C
         AH     10,H2           INCREMENT RIO(B)
         BC     15,INCC         INCREMENT FILE C
*
*
ALO      MVC    0(2,11),0(9)    MOVE FILE A TO C
         AH     9,H2            INCREMENT REG 9 - FILE A
         CH     REND,0(9)       IS FILE A FINISHED
         BC     8,OUT           YES BRANCH OUT OF SUBRTNS
*
*
INCC     AH     11,H2           INCREMENT 11 - FILE C
         BC     15,GO           BRANCH BACK TO GO.
```

```
EQAL     MVC    0(2,11),0(10)   MOVE FILE B TO FILE C
         AH     10,H2           INCREMENT REG 10 - FILE B
         MVC    0(2,11),0(9)    MOVE FILE A TO FILE C
         AH     9,H2            INCREMENT REGISTER 9.
         CH     9,REND          IS FILE
         BC     8,OUT
         BC     15,INCC         INCREMENT FILE C.
OUT      BCR    15,14           BRANCH BACK TO MAIN PROG.
         END    STAR
```

Figure IF-3

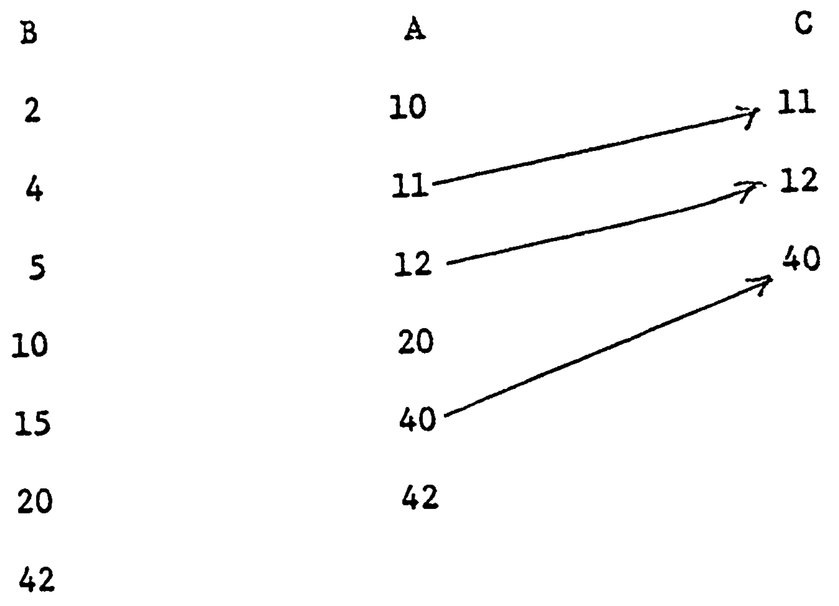
2.4.2.2.2 A AND NOT B

Directory C will contain only the keys of Directory A that are not contained in Directory B.

The algorithm for C - A "AND NOT" B

1. Set 3 index registers (one each) for A, B and C to 1.
2. If the key of A is smaller than the key of B, move the key of A to Directory C, increment indexes A and C.
   If the key of A is greater than the key of B, increment the index of B.
3. Repeat step 2 until Directories A and B have been exhausted.
4. Stop.

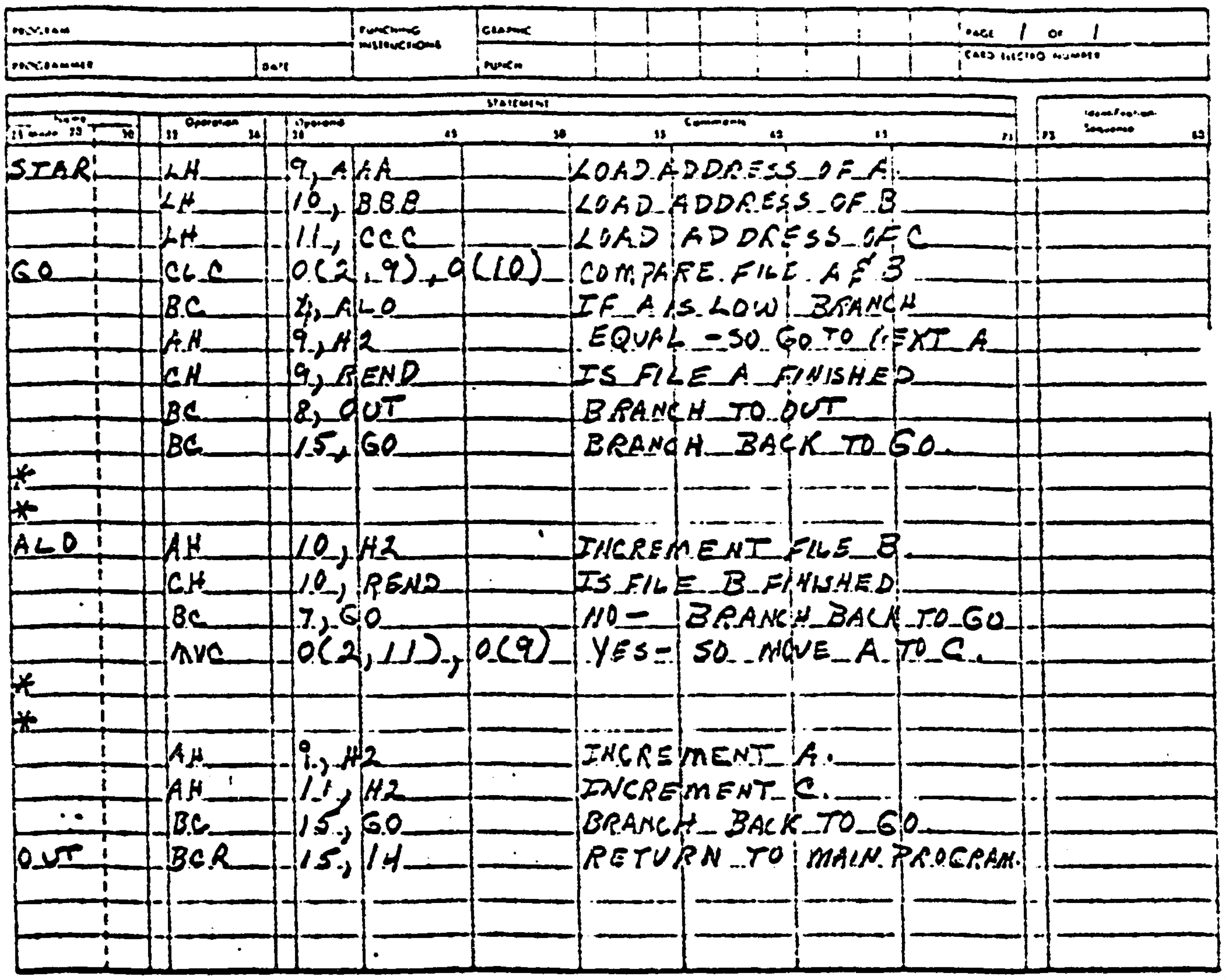
C = A "AND NOT" B



B	A	C
2	10	11
4	11	12
5	12	40
10	20	
15	40	
20	42	
42		

Figure IF-4

A "AND NOT" B



PAGE 1 OF 1

```
STAR   LH    9,AAA              LOAD ADDRESS OF A
       LH    10,BBB             LOAD ADDRESS OF B
       LH    11,CCC             LOAD ADDRESS OF C
GO     CLC   0(2,9),0(10)       COMPARE FILE A & B
       BC    4,ALO              IF A IS LOW BRANCH
       AH    9,H2               EQUAL - SO GO TO NEXT A
       CH    9,REND             IS FILE A FINISHED
       BC    8,OUT              BRANCH TO OUT
       BC    15,GO              BRANCH BACK TO GO.
*
*
ALO    AH    10,H2              INCREMENT FILE B
       CH    10,REND            IS FILE B FINISHED
       BC    7,GO               NO - BRANCH BACK TO GO
       MVC   0(2,11),0(9)       YES - SO MOVE A TO C.
*
*
       AH    9,H2               INCREMENT A.
       AH    11,H2              INCREMENT C.
       BC    15,GO              BRANCH BACK TO GO
OUT    BCR   15,14              RETURN TO MAIN PROGRAM.
```

Figure IF-5

2.4.2.2.3 LOGICAL OPERATOR "AND"

Let us examine the operations it takes the system to create a resultant file, if we take two indexed files and request a retrieval of keys for which both files apply. Let's scan file A and file B.

It is written this way in set theory:

INDEX C = A U B

This allows us to retrieve the keys from Directory A that also reside in Directory B.

The algorithm for Logical Operator "AND"

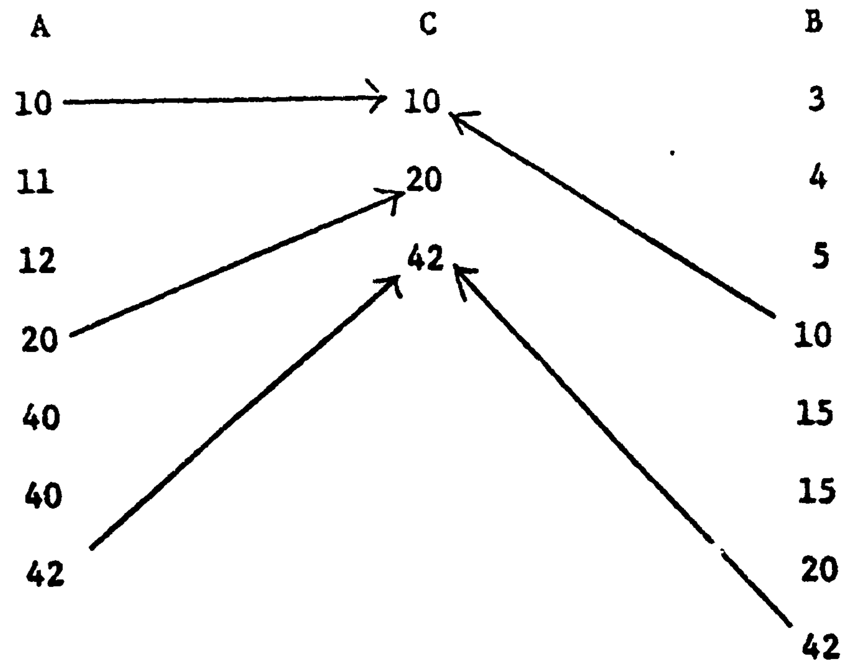
1. Set 3 index registers (one each) for A, B, and C to 1.
2. Compare unit 1 of A to unit 1 of B.

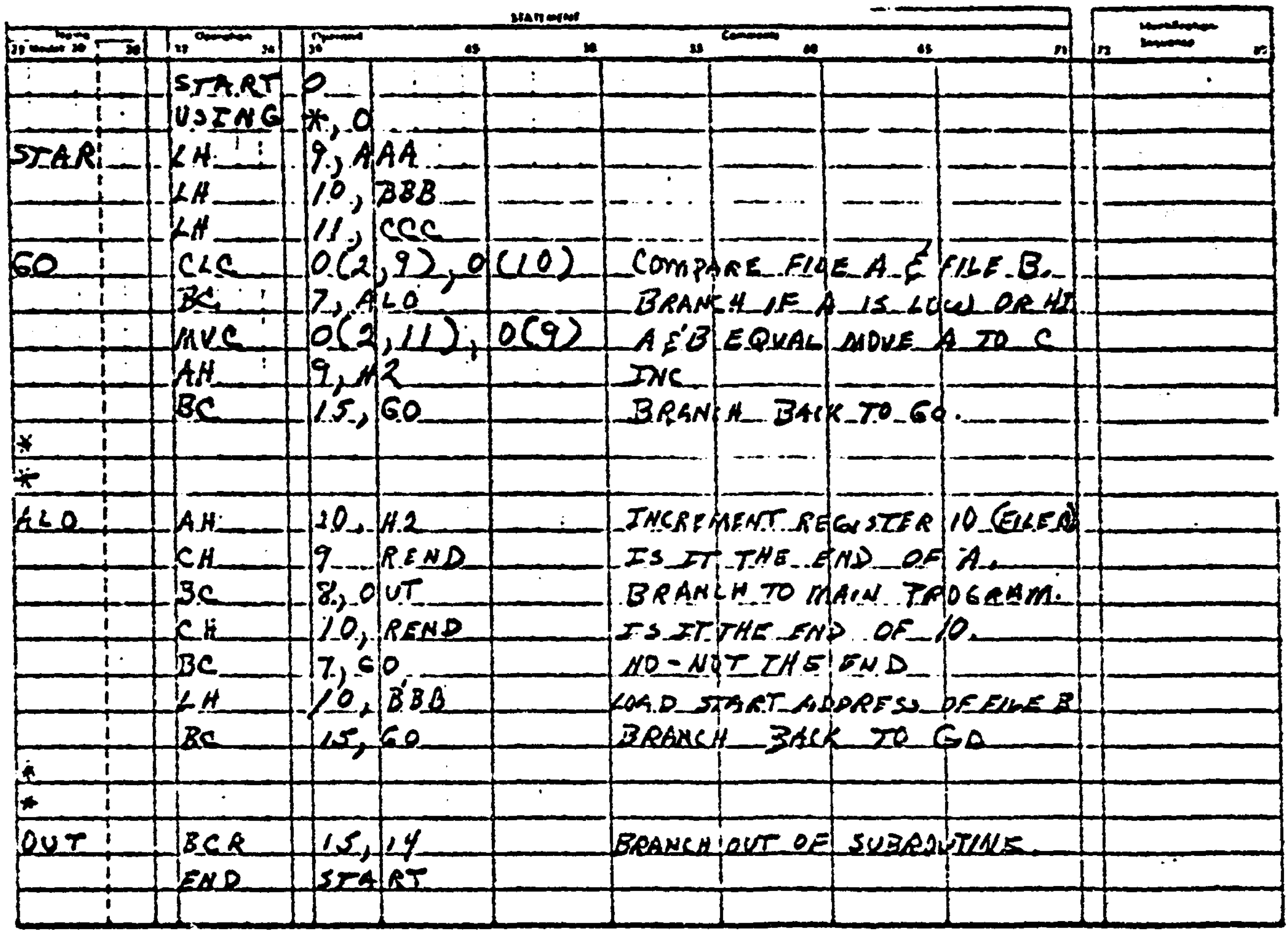
   If the key of A is equal to the key of B, increment A to the next key of A, and increment B to the next key of B, move A to C and increment C.

   If the key of A is smaller than the key of B, increment A to the next key of A.

   If the key of A is larger than the key of B, increment B to the next key of B.
3. Repeat step 2 until all the keys in B have been investigated.

A "AND" B





```
         START 0
         USING *,0
STAR     LH    9,AAA
         LH    10,BBB
         LH    11,CCC
GO       CLC   0(2,9),0(10)     COMPARE FILE A & FILE B.
         BC    7,ALO            BRANCH IF A IS LOW OR HI
         MVC   0(2,11),0(9)     A & B EQUAL MOVE A TO C
         AH    9,H2             INC
         BC    15,GO            BRANCH BACK TO GO.
*
*
ALO      AH    10,H2            INCREMENT REGISTER 10 (FILEB)
         CH    9 REND           IS IT THE END OF A.
         BC    8,OUT            BRANCH TO MAIN PROGRAM.
         CH    10,REND          IS IT THE END OF 10.
         BC    7,GO             NO-NOT THE END
         LH    10,BBB           LOAD START ADDRESS OF FILE B
         BC    15,GO            BRANCH BACK TO GO
*
*
OUT      BCR   15,14            BRANCH OUT OF SUBROUTINE.
         END   START
```

Figure 1F-6

2.4.3 MULTILIST FILE

The use of "cells" or "pages" allows us to use fixed-size blocks to implement a method called the Multilist File. A sequential index contains the key values by which records are indexed. The pointer, which is associated with each key, points to the list of records that has the particular key value in question. The addressing method uses a page number and record number within the page.

A group of records can be retrieved at one time. This implies that access time can be reduced, however the saving is brought about only when a number of records are desired.

See figure M.F.-1, the Multilist File. The index, or directory contains keys L, M, N, and O. If we had a key that started with the letter L, the directory would decode it and take us to (3,3)/11, which translates to page 3, record 3 and tells us that there are 11 records in the list.

The other two methods discussed here use the same records and the same scheme for presenting them. There is a key, an address, and a list length sequence for each system.

The other two methods discussed here are presented with the same type drawing for ease of understanding.

The same logical operations performed on the inverted files can be performed on the multilist files.

Multilist

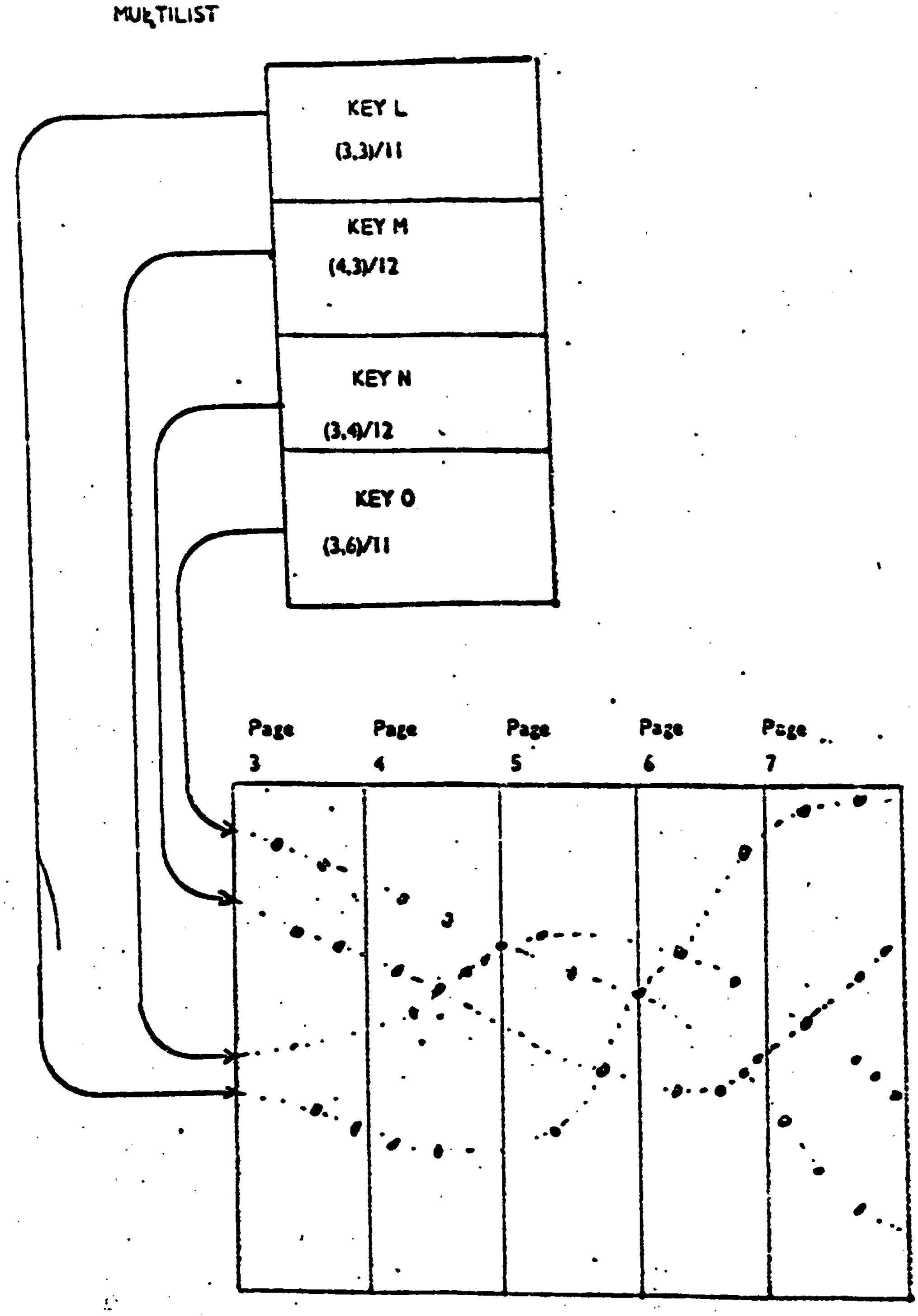
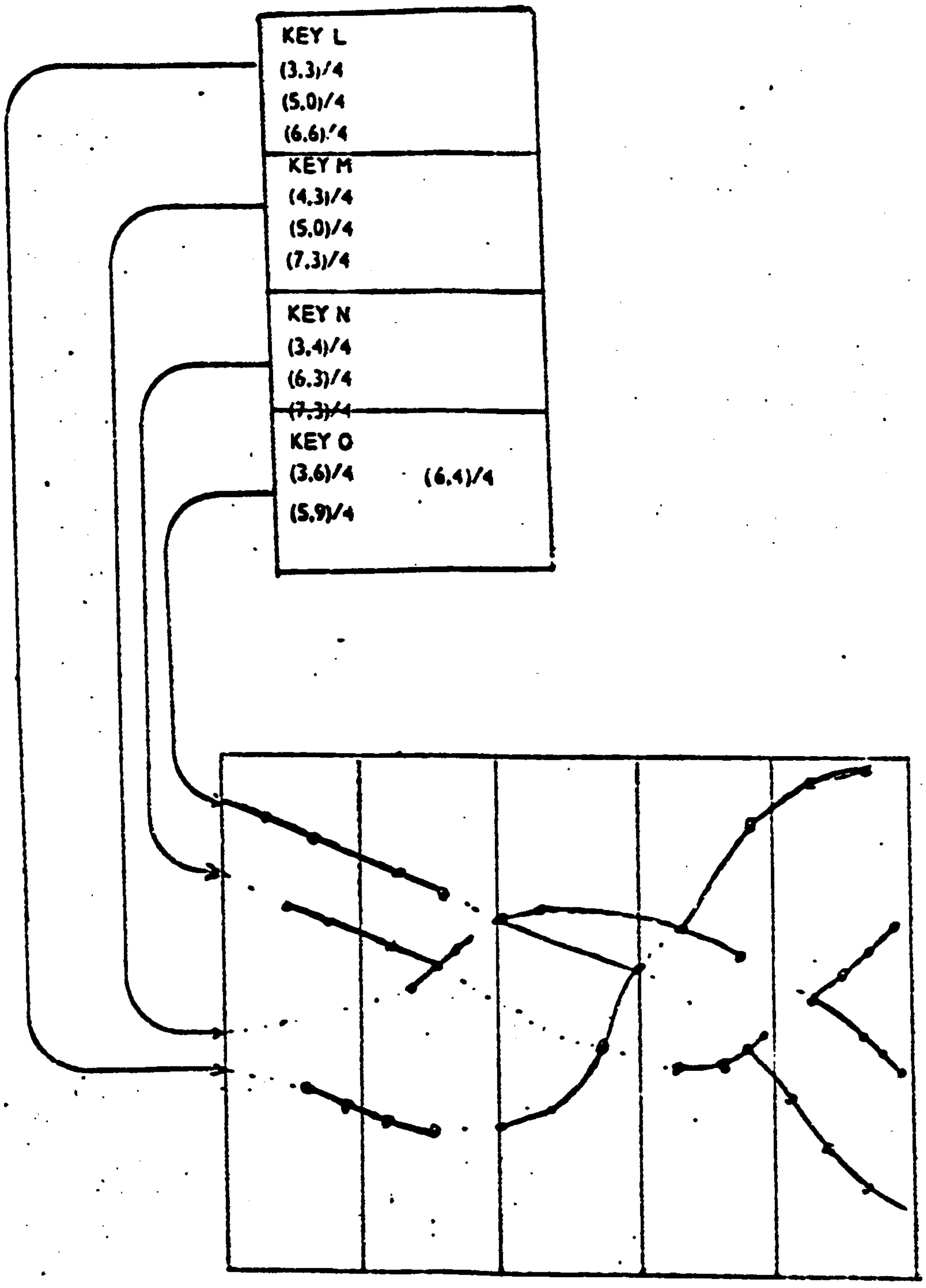


Figure M-1

Controlled Multilist Lengths

The same pattern is followed here with the exception that the list lengths are not more than four records long. The number in the lists is arbitrary. This avoids the long lists that the Multilist method had.

See Figure M.F.2, the Controlled Multilist File. The index contains the same keys that the Multilist File did, but we notice that there are more divisions within the key group. This time L has three divisions and the same number of records.



Multilist Organization for Pages

The Page Multilist files are linked so that they do not cross page boundaries. The advantage to this is that it does not require several pages of information when a key is used to retrieve certain information. A Multilist File might require several pages to be accessed.

The cells or pages become the partition instrument. This allows the list structure and the partition (page partition) to function.

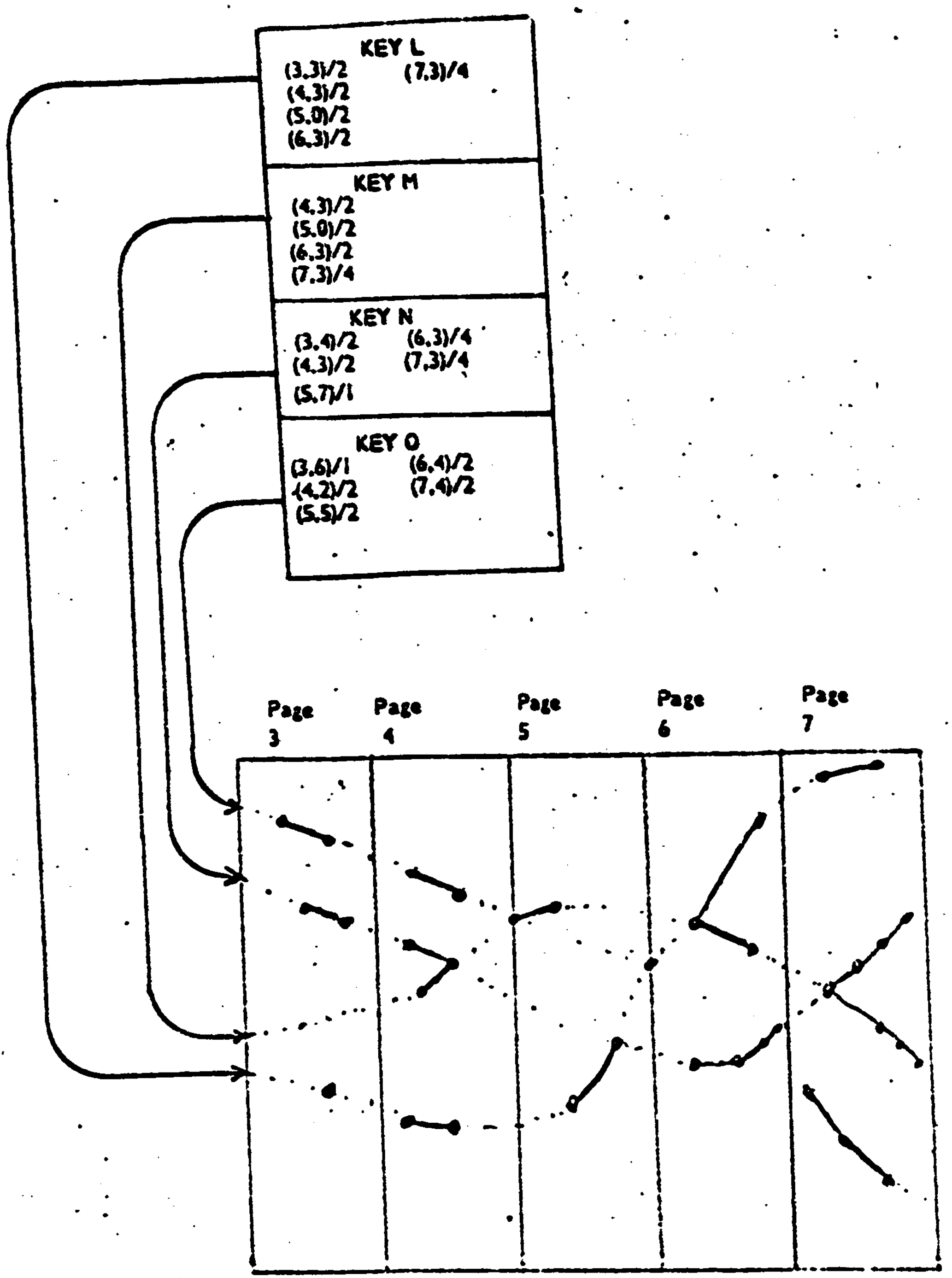


Figure M-3

2.5 SORTING

The object of the sort section is to present various methods of sorting to allow the reader a wider choice of methods from which to select.

The activity section at the end of this section suggests examples that are contained in the various Basic Assembler Language programs in the book.

SORTING METHODS

RECORD SORT

The sort algorithm is applied directly to the complete record by using the key in the record. This means the complete record is moved when a change is indicated. The end result is that each record is in its proper place when the file is sorted.

2.5.1.1 KEY SORT

The key sort works with the keys only, as opposed to the whole record. This includes an associative table of address pointers. Associative in this instance refers to a memory device, in which each cell contains information pertaining to the key and entry. The entry part tells us whether a cell has an entry pertaining to the key that accessed the particular cell. If there is no entry, this signal is returned as "not found" or something similar.

There are two types of key sorts, the detached and the non-detached key sort.

2.5.1.2 DETACHED KEY SORT

The algorithm for the Detached Key Sort:

1. The record keys are placed in an associative table with address pointers.
2. Apply the internal sorting algorithm to the associative table only.
3. If a transfer is to be made, make it include only the key and associated record address.
4. When the associative table is sorted, the complete records in the file are moved to output according to the sorted order of the address pointers.

This algorithm is quite forward, if we separate the key and address from the rest of the record to form a new table.

2.5.1.3 THE NONDETACHED KEY SORT

The nondetached key sort forms a table which contains pointers to the keys of the associated records. This method applies the sort algorithm indirectly through the associated address pointers in the table. The address pointers (not the keys) are moved during the sort. Both the table of addresses and the stored file are referenced during the sorting procedure.

When the sorting algorithm is finished, only the record address pointers have been sorted. The retrieval process requires that the sequence of the record address associative table be followed.

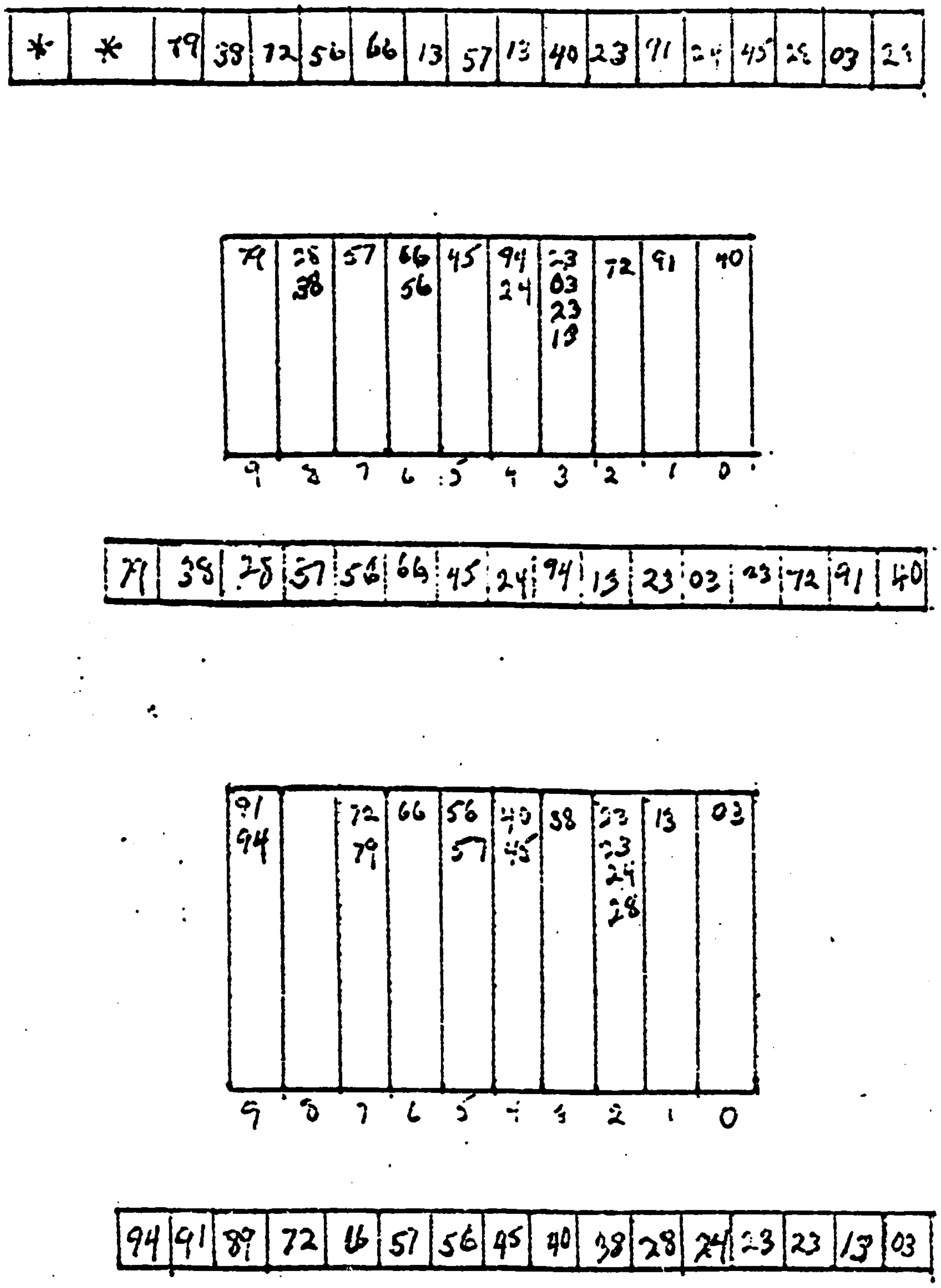
2.5.2 RADIX SORT

This method is used for sorting punched cards mechanically, but can be used for the computer as well.

The least significant digit of each key is examined. Then the record is assigned to a pocket which depends on the value of the digit. After all the records have been examined, the records are distributed again, and so on, until all the digits of the key have been used for distribution. After the collection of the pass following the most significant digit, the records are in order. For decimal keys, ten pockets are needed, or else each pass could be replaced by two passes, in which case fewer pockets would be required.

The total number of passes is equal to the number of digits in the key, and the capacity of each pocket must be sufficient for all the records that might end up there.

Radix Sort



*	*	79	38	72	56	66	13	57	13	40	23	91	24	45	28	03	23

79	28 38	57	66 56	45	94 24	23 03 23 13	72	91	40
9	8	7	6	5	4	3	2	1	0

79	38	28	57	56	66	45	24	94	13	23	03	23	72	91	40

91 94		72 79	66	56 57	40 45	38	23 23 24 28	13	03
9	8	7	6	5	4	3	2	1	0

94	91	89	72	66	57	56	45	40	38	28	24	23	23	13	03

Figure RS-1

2.5.3 QUICKSORT

The quicksort algorithm can be classified as a partitioning type algorithm, and it was developed by Hoare. The main aspects of the algorithm are:

1. The first pass located the item that occupies the middle spot of the list and classifies it as a bound.
2. This item is copied into a temporary location and replaced with the bottom of the list.
3. The list is scanned from the top, and each item is compared in sequence to the bound until an item is found that is larger than, or equal to, the bound.
4. If the new item is larger than the bound item, the new item is moved to the bottom of the list, creating a vacant spot in the top part of the list.
5. The top-down scan is then continued, alternating the scan each time a transfer is made, until all the items have been scanned. When the two top pointers coincide, the list will be partitioned into two parts.
6. The count is then placed in the vacant spot between the two partitions. This is the spot where the two pointers coincide. It now occupies its sorted position, because all items above it are smaller, and all items below it are larger or equal to it.

The partition that has the most elements is then stacked for later processing, while the algorithm proceeds to partition the smaller list in order. This continues until each item has been placed into its proper sorted position either by being selected as a bound or by remaining a single item in a partition.

2.5.4 TOPOLOGICAL SORT

The topological sort is used in situations that require partial ordering. Pert and Critical Path methods are good examples of this process. Partial ordering occurs in mathmatics in situations such as A = B (between real numbers), and also between A and B as A   B in set theory.

The critical path program presented in a previous chapter has many good examples of the topological sort. Presented here is the algorithm (in part).

5. Sort the table on the basis of lowest starting event number.
6. Locate activity of the lowest starting event number.
7. Locate and place in table TTO all activities with end event numbers that match the start events of (6).
8. Sort TTWO on the basis of ending times.
9. Subtract ending time of each activity from the ending time of the longest activity in table TTWO.
10. Enter difference in slack time position.
11. Enter longest event time from table TTWO into start time for activity (6).
12. Compute end time for activity (6).
13. Index activity (6).
14. Transfer table TTWO and activity (6) back to the main table.

15. Test for last event.

16. NO---back to (6).

2.5.5 QUADRATIC SORT

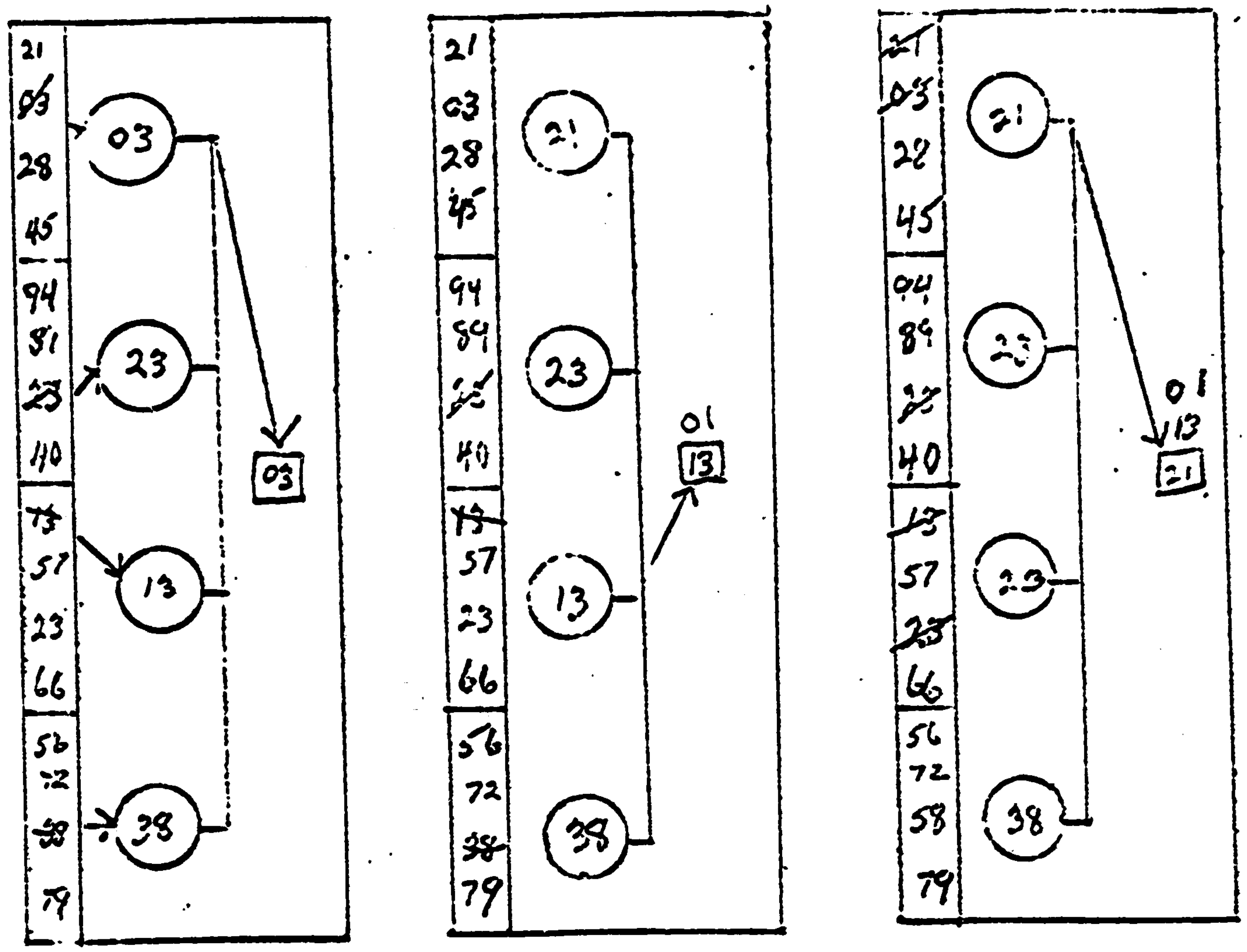
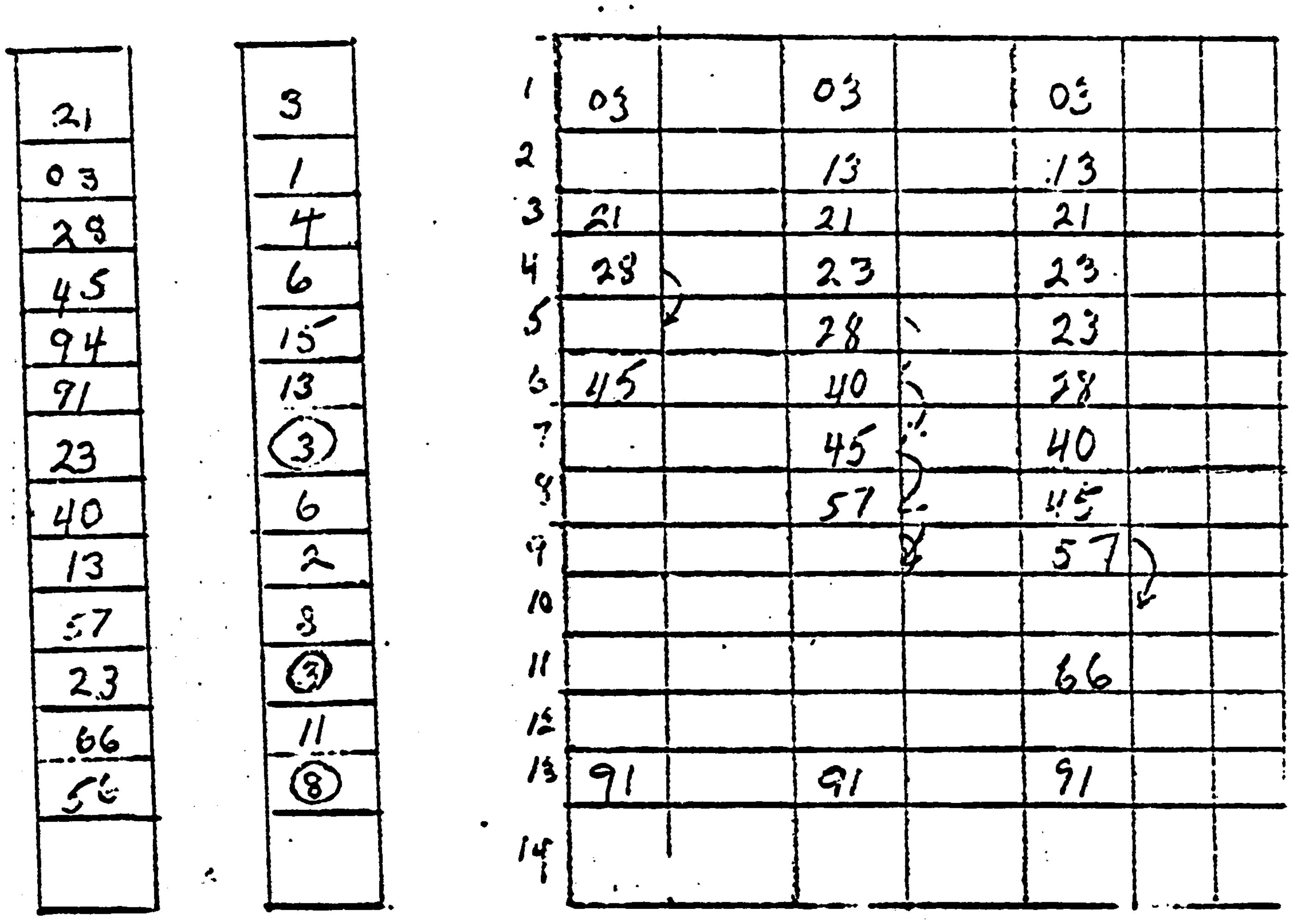


Figure QS-1

The quadratic sort divides the array into four equal parts and then searches each of the four groups for the smallest data. This number is erased and moved to a temporary area, where each of four are compared. The smallest of these is transferred to the output array. The process, with the exception of the division into four units, is combined until all the original array is blank, and all the intermediate temporary areas are blank. The final result is a sequential ordering in the output array.

2.5.6 CALCULATED ADDRESS SORT



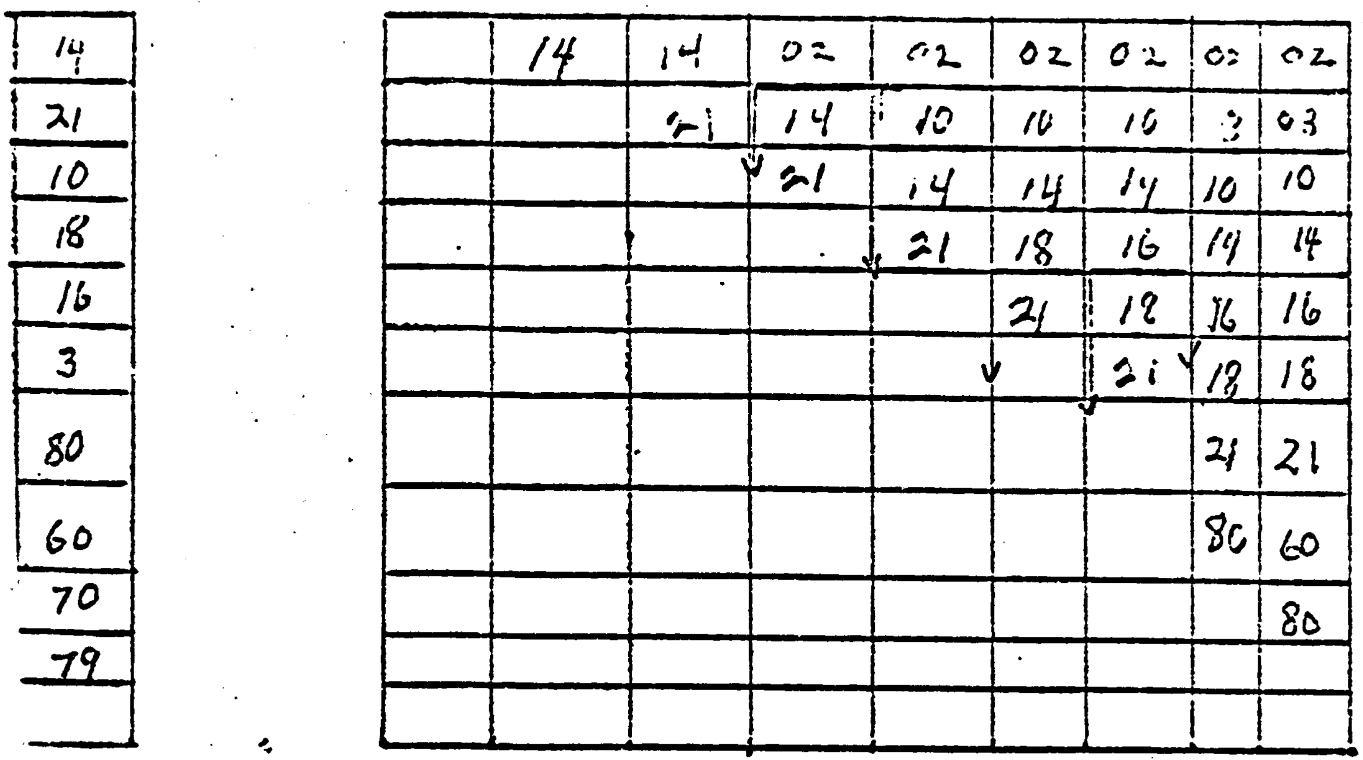
21	3
03	1
28	4
45	6
94	15
91	13
23	3
40	6
13	2
57	8
23	3
66	11
56	8

1	03		03		03	
2			13		13	
3	21		21		21	
4	28		23		23	
5			28		23	
6	45		40		28	
7			45		40	
8			57		45	
9					57	
10						
11					66	
12						
13	91		91		91	
14						

Figure CAS-1

Addresses are calculated by one of the random methods presented earlier. If the cell is vacant, the data is placed there, if the cell is occupied, a compare is made, and the smaller of the two is inserted into the cell, and the larger goes to the next cell, after all the following data has been moved down one cell sequentially.

2.5.7 SORTING BY INSERTION

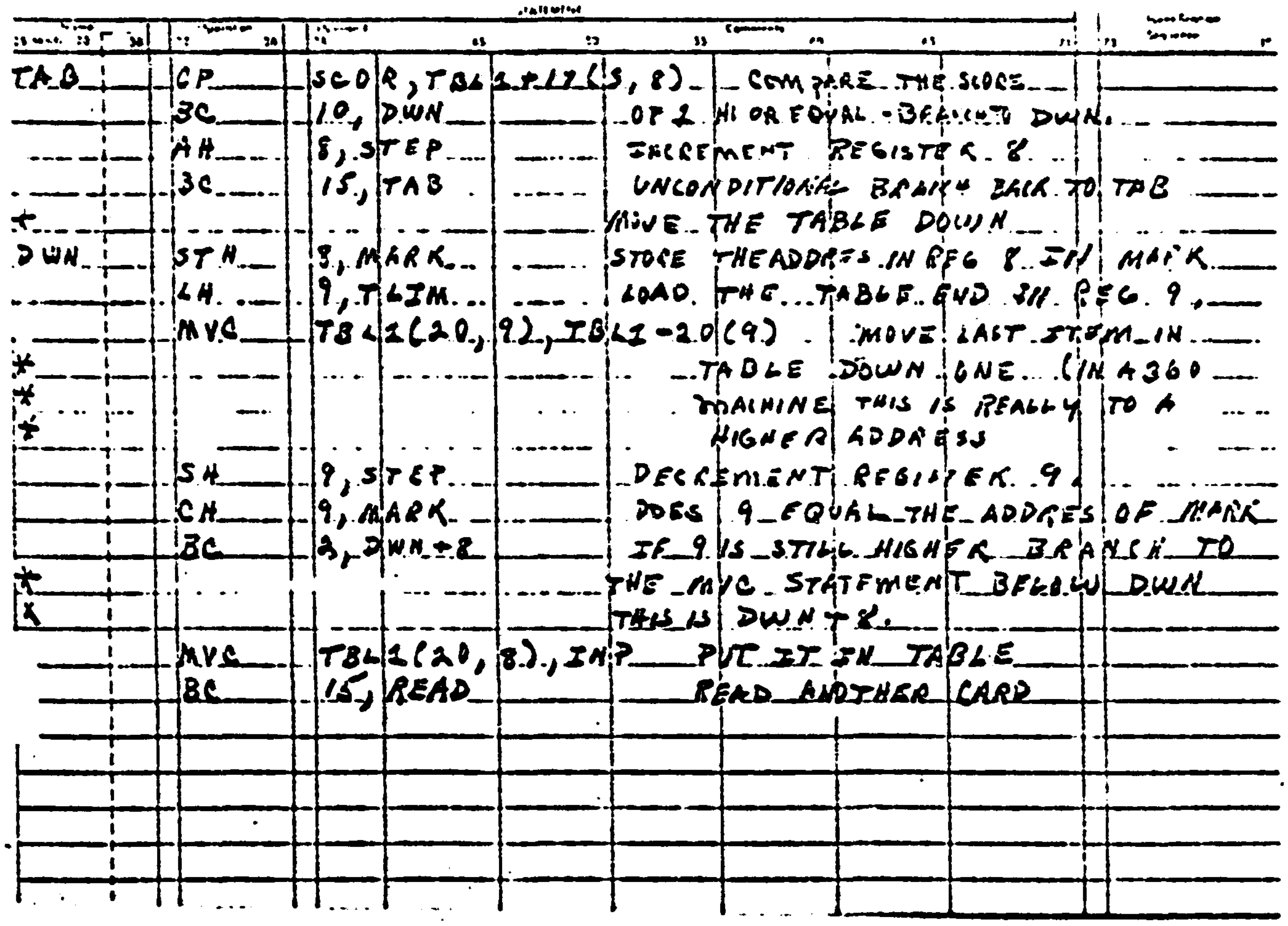


14
21
10
18
16
3
80
60
70
79

	14	14	02	02	02	02	02	02
		21	14	10	10	10	3	03
			21	14	14	14	10	10
				21	18	16	14	14
					21	18	16	16
						21	18	18
							21	21
							80	60
								80

Figure SI-1

Each key is examined in turn and inserted into the correct place. The earlier members are pushed down when the need arises. The mechanics of moving the earlier records down is also used as a method of insertion with lists. The actual number is n/4. This method of sorting is useful when there isn't much core storage.



```
TAB      CP    SCOR,TBL1+17(3,8)      COMPARE THE SCORE
         BC    10,DWN                 OF 2 HI OR EQUAL - BRANCH TO DWN.
         AH    8,STEP                 INCREMENT REGISTER 8
         BC    15,TAB                 UNCONDITIONAL BRANCH BACK TO TAB
*                                     MOVE THE TABLE DOWN
DWN      STH   8,MARK                 STORE THEADDRES IN REG 8 IN MARK
         LH    9,TLIM                 LOAD THE TABLE END IN REG 9,
         MVC   TBL1(20,9),TBL1-20(9)   MOVE LAST ITEM IN
*                                      TABLE DOWN ONE (IN A 360
*                                      MACHINE THIS IS REALLY TO A
*                                      HIGHER ADDRESS
         SH    9,STEP                 DECREMENT REGISTER 9
         CH    9,MARK                 DOES 9 EQUAL THE ADDRES OF MARK
         BC    2,DWN+8                IF 9 IS STILL HIGHER BRANCH TO
*                                     THE MVC STATEMENT BELOW DWN
*                                     THIS IS DWN+8.
         MVC   TBL1(20,8),INP         PUT IT IN TABLE
         BC    15,READ                READ ANOTHER CARD
```

Figure S1-2

2.5.8 INTERCHANGE SORT

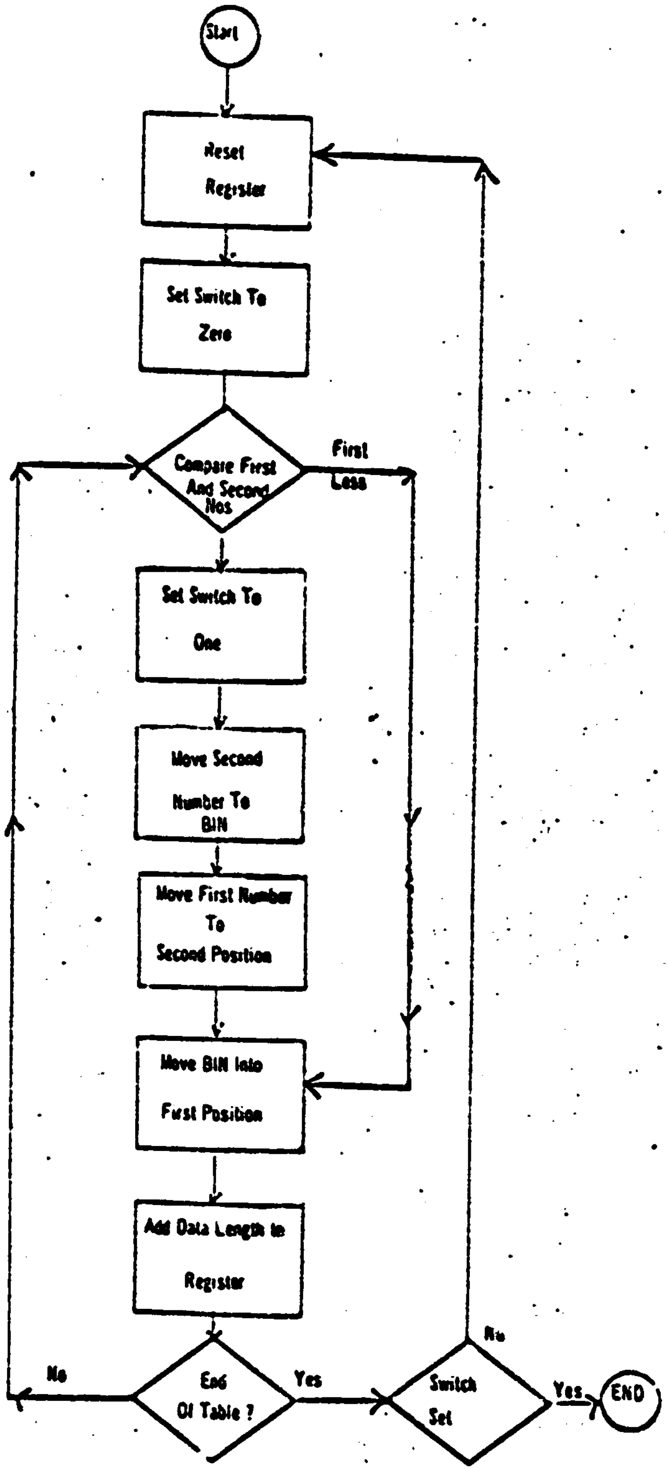
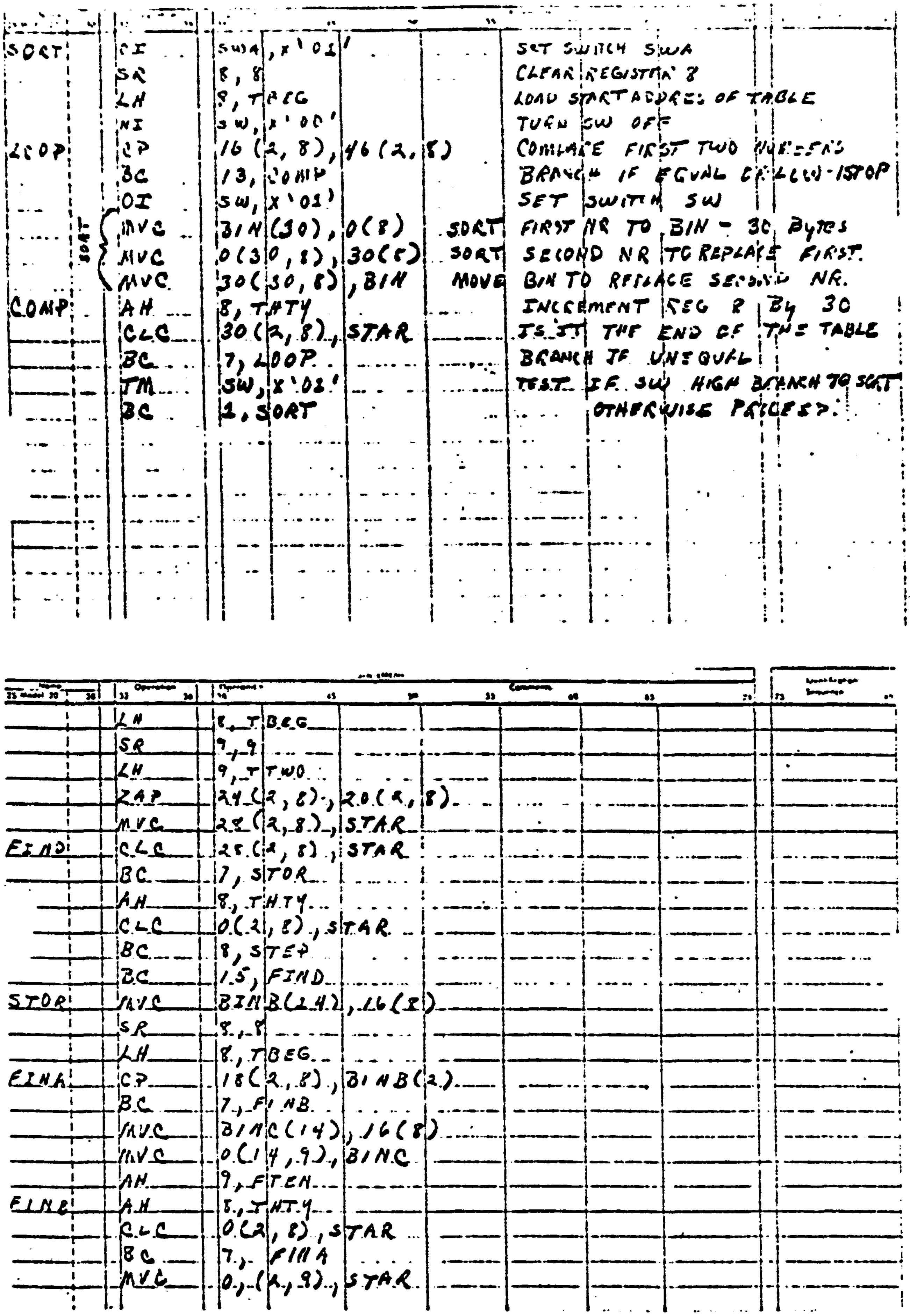

Start
Reset Register
Set Switch To Zero
Compare First And Second Nos
First Less
Set Switch To One
Move Second Number To BIN
Move First Number To Second Position
Move BIN Into First Position
Add Data Length to Register
No
End Of Table ?
Yes
Switch Set
No
Yes
END

Figure IS-1

Interchange Sort



```
SORT   OI    SWA,X'01'                      SET SWITCH SWA
       SR    8,8                            CLEAR REGISTER 8
       LH    8,TBEG                         LOAD START ADDRES OF TABLE
       NI    SW,X'00'                       TURN SW OFF
LOOP   CP    16(2,8),46(2,8)                COMPARE FIRST TWO NUMBERS
       BC    13,COMP                        BRANCH IF EQUAL OR LOW-1STOP
       OI    SW,X'01'                       SET SWITCH SW
       MVC   BIN(30),0(8)           SORT    FIRST NR TO BIN - 30 BYTES
       MVC   0(30,8),30(8)          SORT    SECOND NR TO REPLACE FIRST.
       MVC   30(30,8),BIN           MOVE    BIN TO REPLACE SECOND NR.
COMP   AH    8,THTY                         INCREMENT REG 8 BY 30
       CLC   30(2,8),STAR                   IS IT THE END OF THE TABLE
       BC    7,LOOP                         BRANCH IF UNEQUAL
       TM    SW,X'01'                       TEST IF SW HIGH BRANCH TO SORT
       BC    1,SORT                             OTHERWISE PROCEED.
```

```
       LH    8,TBEG
       SR    9,9
       LH    9,TTWO
       ZAP   24(2,8),20(2,8)
       MVC   28(2,8),STAR
FIND   CLC   28(2,8),STAR
       BC    7,STOR
       AH    8,THTY
       CLC   0(2,8),STAR
       BC    8,STEP
       BC    15,FIND
STOR   MVC   BINB(24),16(8)
       SR    8,8
       LH    8,TBEG
FINA   CP    18(2,8),BINB(2)
       BC    7,FINB
       MVC   BINC(14),16(8)
       MVC   0(14,9),BINC
       AH    9,FTEN
FINB   AH    8,THTY
       CLC   0(2,8),STAR
       BC    7,FINA
       MVC   0,(2,9),STAR
```

Figure M-2

2.6 MERGING

2.6.1 TWO WAY MERGE

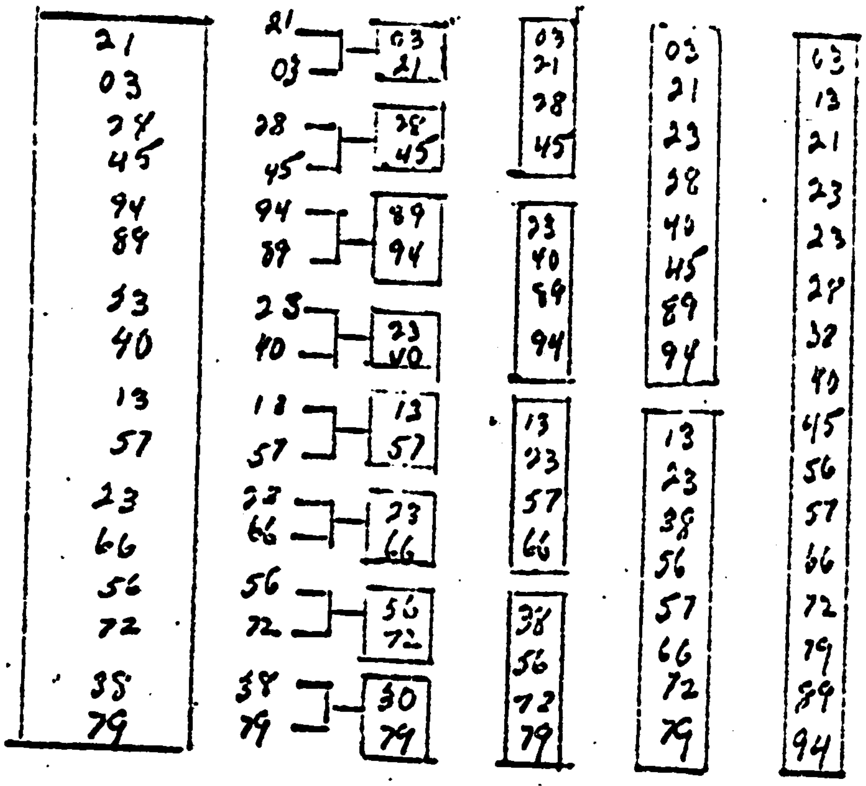


Figure M-1

Many different types of sorting are descendents of this basic approach to the merge. The two way merge compares pairs of keys, and each pair, of which the smaller key is placed first. After one pass, the initial group N consists of N/2 strings of length two. These pairs of strings are combined to form N/4 strings of length four. These pairs are combined to form N/8 strings, until one string results. If N is of the form 2p, it will require p passes to complete the sort.

Merging Two Sorted Files

The comparison is made between the keys in the records. A merge can be made with the order ascending or descending. For our example, we will use the ascending merge, since our two input files are sorted in ascending order.

This merge effects the union of File A and File B into File C. If a number appears in both files, the number in File A will receive first precedence.

The Merge Algorithm

1. Set two indexes (1 each) for A and B, also set an index for C. (Set these indexes to 1.)
2. Compare the key of File A to the key of File B, if the key of A is smaller than the key of B, move the key of A to File C and increment index A and index C.
    a. If the key of A is equal to the key of B, move the key of A to File C.
    b. If the key of File A is larger than the key of File B, move the key of B into File C, and increment index B and index C.
3. Repeat step 2 until all of Files A and B have both been exhausted.

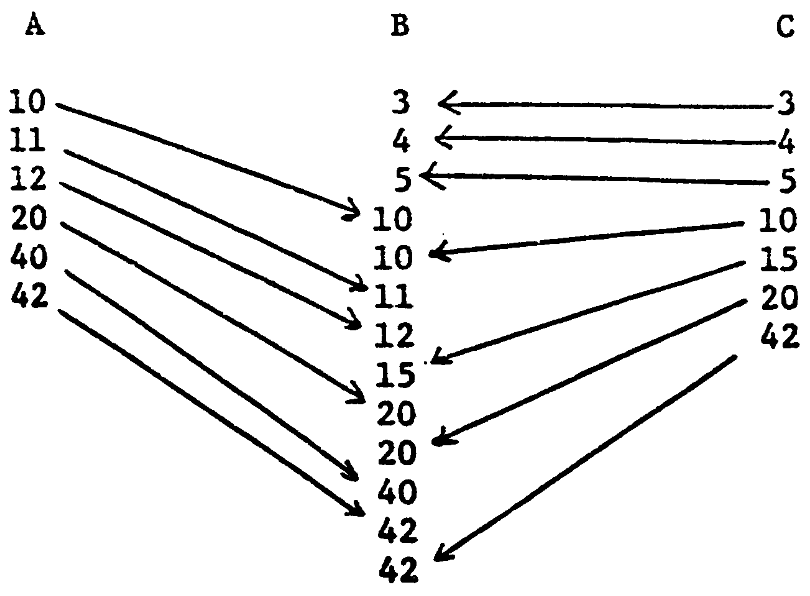


Figure M-3

2.6.1.2 MERGING ORDERED FILES WITH SUBFILES

There are four possibilities that can exist here:

1. There can be no problem, such as the preceding example, where no changes are necessary.
2. A step down condition can exist (a single step down).
3. A double step down condition can exist.
4. It may be necessary to perform a roll out.

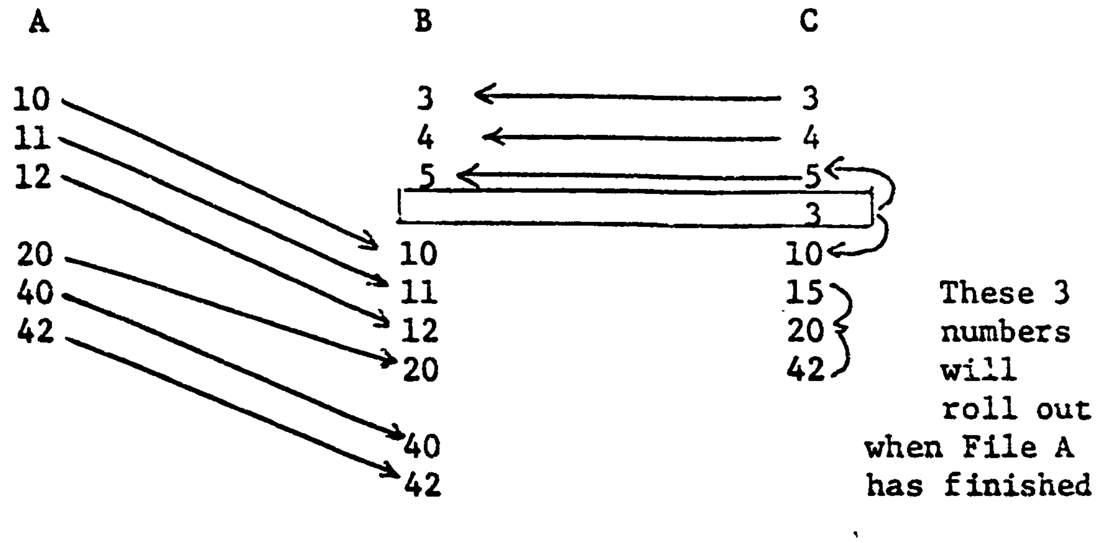
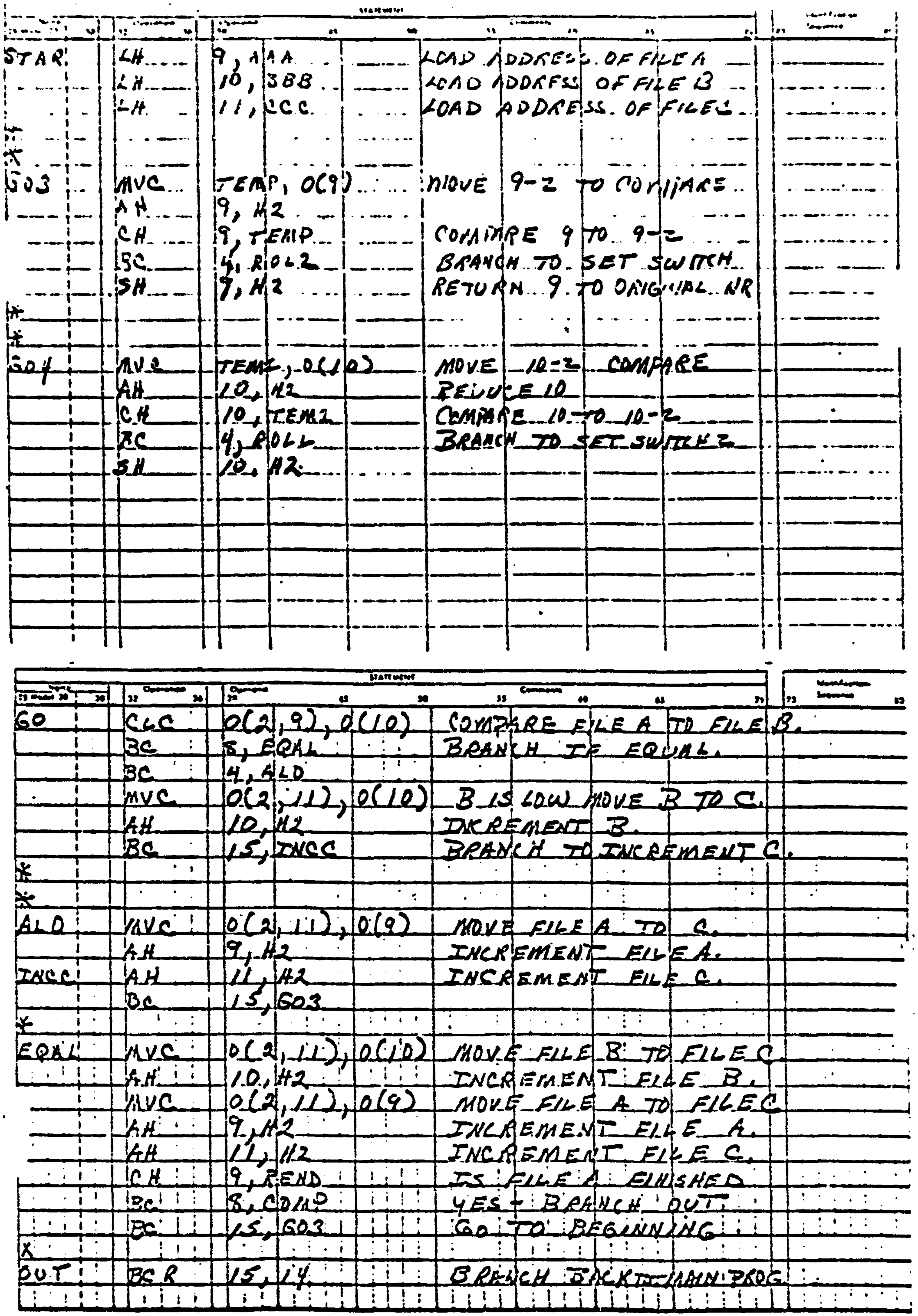


Figure M-5

The single step down is given in the example above, M 1. The number 3 in File B is a step down, because it is smaller than the number preceding it in the file, and it is also smaller than the number that follows it in File B. This means that File B will not be used again until File A has a step down also. The sequential process continues with File A until A results in the step down mentioned above. If there are no further step downs in File A, then File B will continue sequentially. This is called a roll out.

Merging Ordered Files with Subfiles

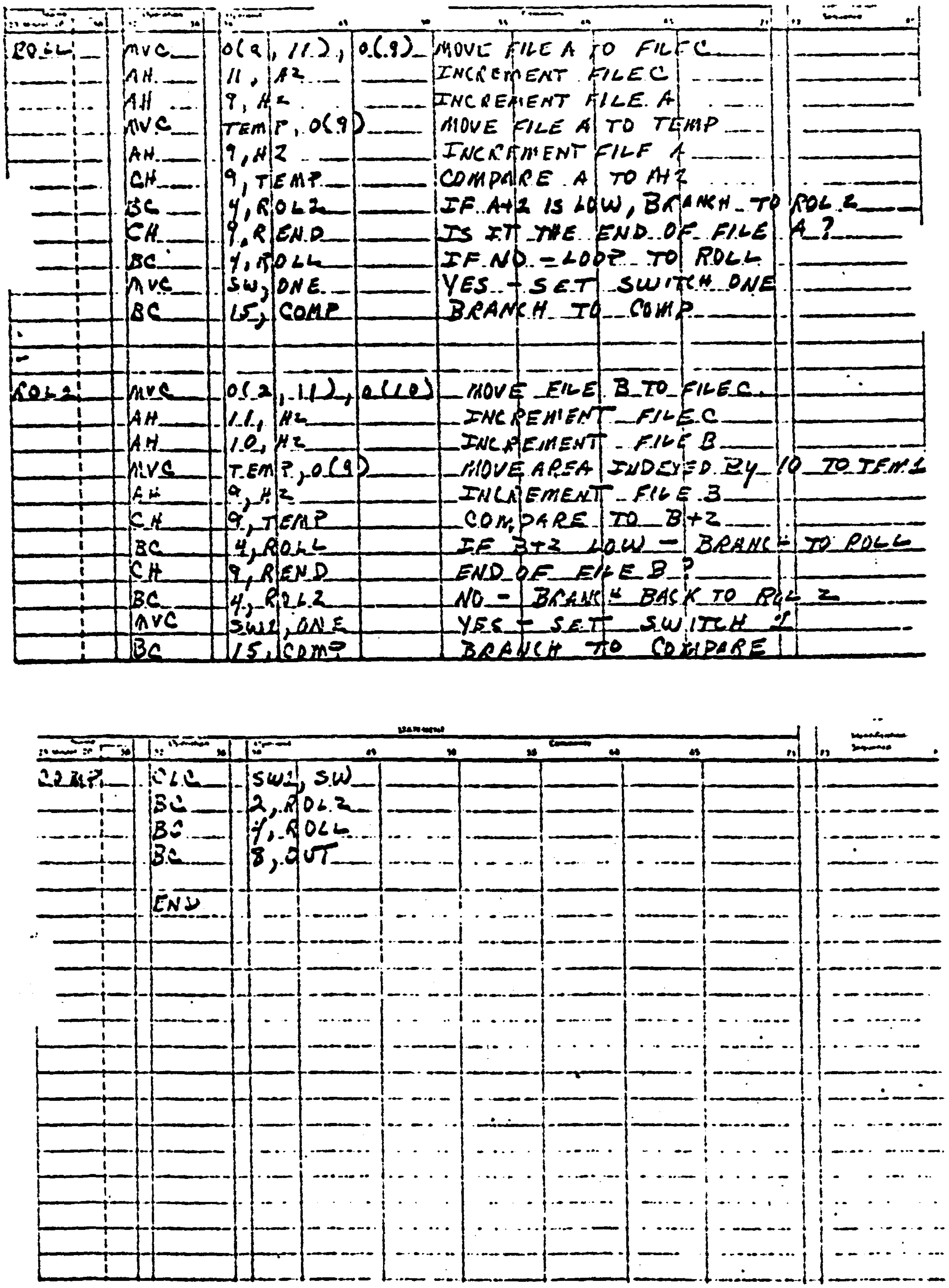


```
STAR     LH    9,AAA            LOAD ADDRESS OF FILE A
         LH    10,BBB           LOAD ADDRESS OF FILE B
         LH    11,CCC           LOAD ADDRESS OF FILE C
*
G03      MVC   TEMP,0(9)        MOVE 9-2 TO COMPARE
         AH    9,H2
         CH    9,TEMP           COMPARE 9 TO 9-2
         BC    4,ROL2           BRANCH TO SET SWITCH
         SH    9,H2             RETURN 9 TO ORIGINAL NR
*
*
G04      MVC   TEM2,0(10)       MOVE 10-2 COMPARE
         AH    10,H2            REDUCE 10
         CH    10,TEM2          COMPARE 10 TO 10-2
         BC    4,ROLL           BRANCH TO SET SWITCH 2
         SH    10,H2
```

```
GO       CLC   0(2,9),0(10)     COMPARE FILE A TO FILE B.
         BC    8,EQAL           BRANCH IF EQUAL.
         BC    4,ALD
         MVC   0(2,11),0(10)    B IS LOW MOVE B TO C.
         AH    10,H2            INCREMENT B.
         BC    15,INCC          BRANCH TO INCREMENT C.
*
*
ALD      MVC   0(2,11),0(9)     MOVE FILE A TO C.
         AH    9,H2             INCREMENT FILE A.
INCC     AH    11,H2            INCREMENT FILE C.
         BC    15,G03
*
EQAL     MVC   0(2,11),0(10)    MOVE FILE B TO FILE C.
         AH    10,H2            INCREMENT FILE B.
         MVC   0(2,11),0(9)     MOVE FILE A TO FILE C
         AH    9,H2             INCREMENT FILE A.
         AH    11,H2            INCREMENT FILE C.
         CH    9,REND           IS FILE A FINISHED
         BC    8,COMP           YES - BRANCH OUT.
         BC    15,G03           GO TO BEGINNING
X
OUT      BCR   15,14            BRANCH BACK TO MAIN PROG.
```

Figure M-6

Merging Ordered Files with Subfiles



```
ROLL   MVC   0(9,11),0(9)   MOVE FILE A TO FILEC
       AH    11,H2          INCREMENT FILEC
       AH    9,H2           INCREMENT FILE A
       MVC   TEMP,0(9)      MOVE FILE A TO TEMP
       AH    9,H2           INCREMENT FILF A
       CH    9,TEMP         COMPARE A TO A+2
       BC    4,ROL2         IF A+2 IS LOW, BRANCH TO ROL2
       CH    9,REND         IS IT THE END OF FILE A?
       BC    7,ROLL         IF NO - LOOP TO ROLL
       MVC   SW,ONE         YES - SET SWITCH ONE
       BC    15,COMP        BRANCH TO COMP

ROL2   MVC   0(2,11),0(10)  MOVE FILE B TO FILEC.
       AH    11,H2          INCREMENT FILEC
       AH    10,H2          INCREMENT FILE B
       MVC   TEMP,0(9)      MOVE AREA INDEXED BY 10 TO TEM1
       AH    9,H2           INCREMENT FILE B
       CH    9,TEMP         COMPARE TO B+2
       BC    4,ROLL         IF B+2 LOW - BRANCH TO ROLL
       CH    9,REND         END OF FILE B?
       BC    4,ROL2         NO - BRANCH BACK TO ROL 2
       MVC   SW1,ONE        YES - SET SWITCH 1
       BC    15,COMP        BRANCH TO COMPARE
```

```
COMP   CLC   SW1,SW
       BC    2,ROL2
       BC    4,ROLL
       BC    8,OUT

       END
```

Figure M-6 (Part-2)

Double Step-Down

The double step-down is treated as if there were no step down at all. In fact it is handled like the straight merge.

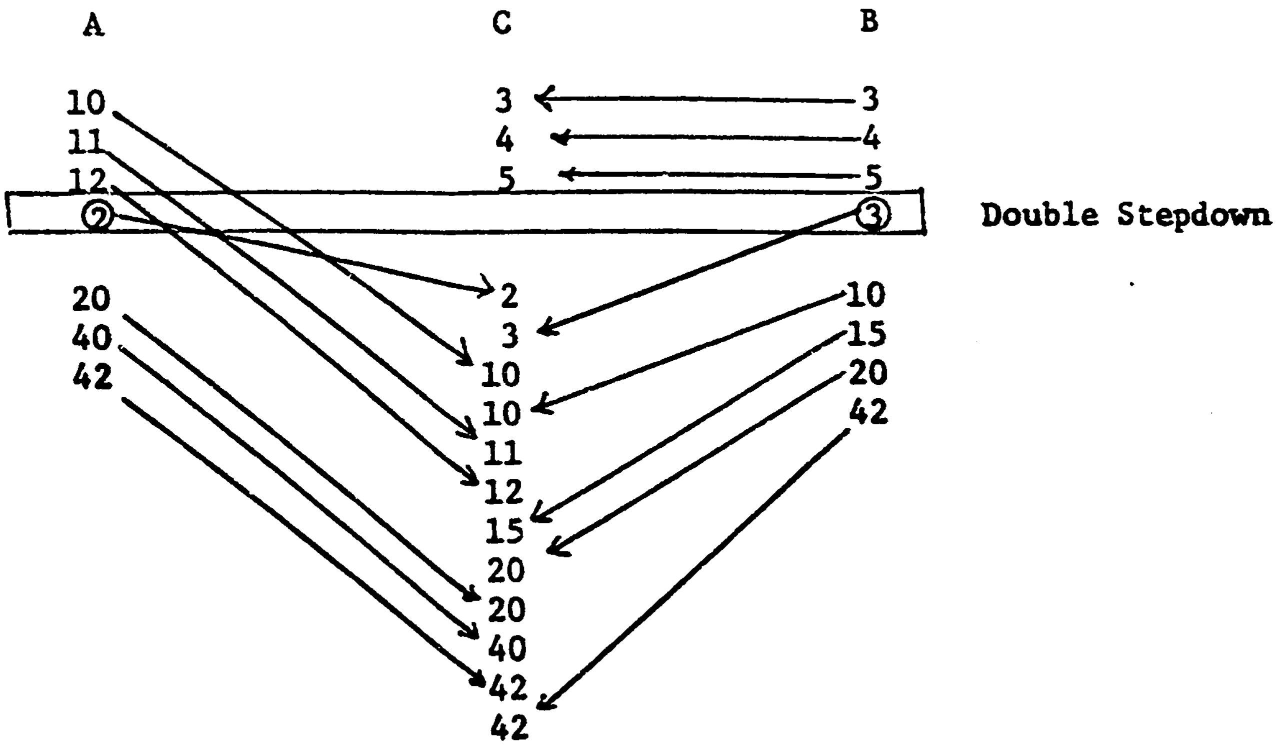


Figure M-7

The double step-down is a step down which occurs in both files, so it operates in the same manner as a merge.

Activities

1. What type of sort does the construction program use? If the reader will notice, the sort flow chart was taken out of context for emphasis.

2. What type of sort does the STAT program use? Compare it to the example presented in this chapter.

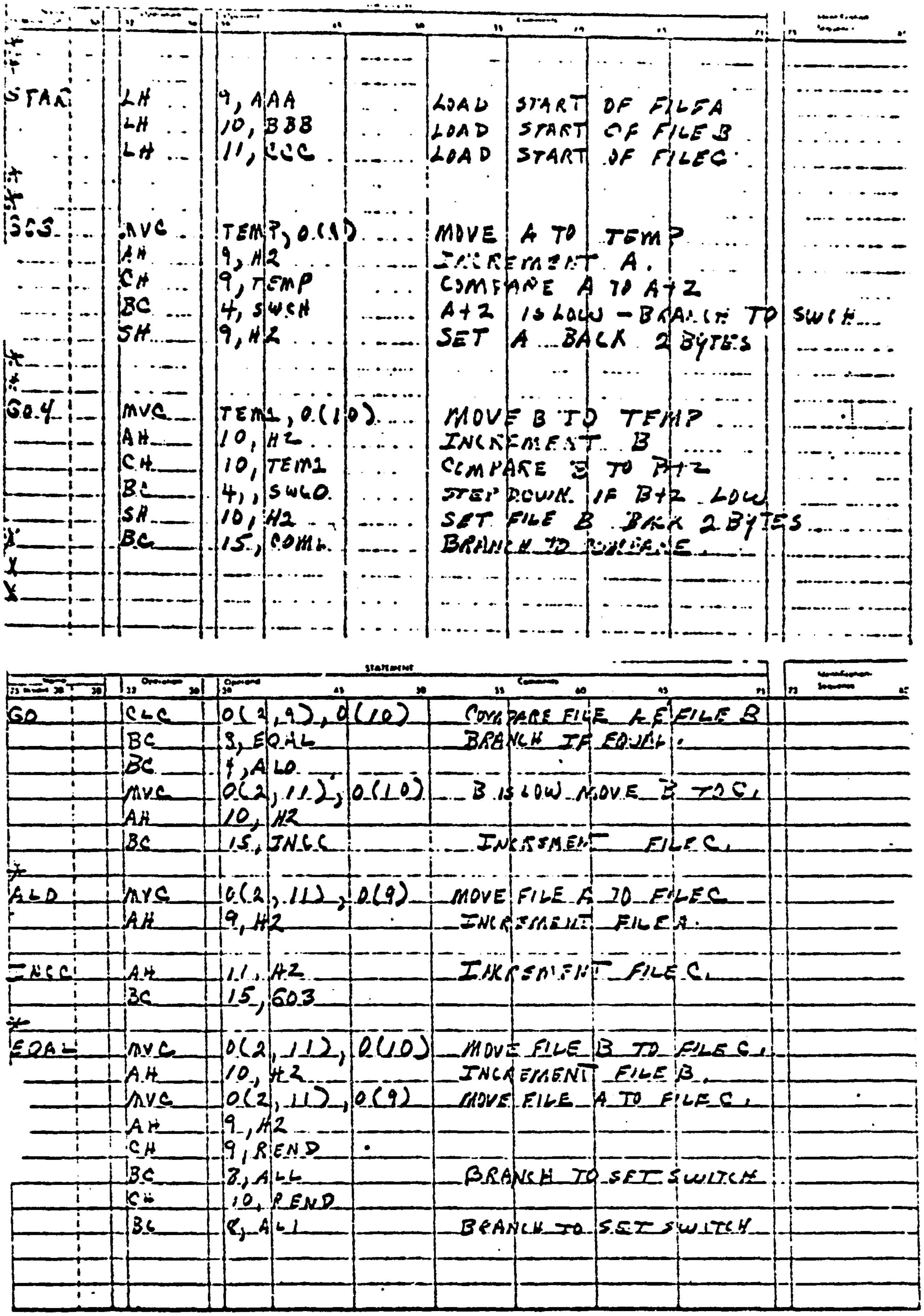
   (Answer) A displacement insertion type.

3. What type of sort does cause problems when there are many

digits to sort?

(Answer) Interchange Sort.

Merging--Double Step-Down

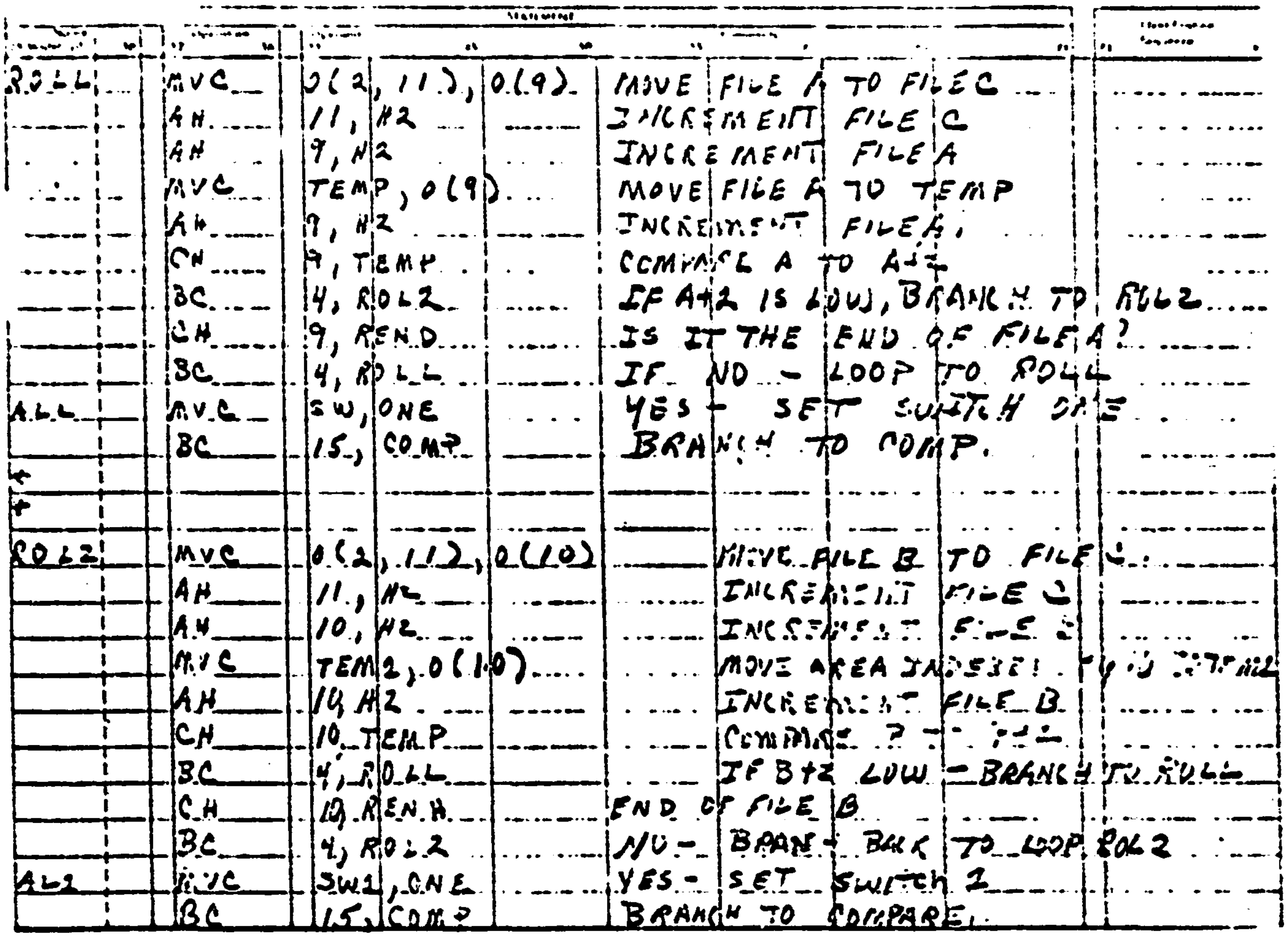


```
*
*
STAR     LH    9,AAA          LOAD START OF FILEA
         LH    10,BBB         LOAD START OF FILEB
         LH    11,CCC         LOAD START OF FILEC
*
*
GO3      MVC   TEMP,0(9)      MOVE A TO TEMP
         AH    9,H2           INCREMENT A.
         CH    9,TEMP         COMPARE A TO A+2
         BC    4,SWCH         A+2 IS LOW - BRANCH TO SWCH
         SH    9,H2           SET A BACK 2 BYTES
*
*
GO4      MVC   TEM1,0(10)     MOVE B TO TEMP
         AH    10,H2          INCREMENT B
         CH    10,TEM1        COMPARE B TO B+2
         BC    4,SWLO         STEP DOWN IF B+2 LOW
         SH    10,H2          SET FILE B BACK 2 BYTES
*        BC    15,COMP        BRANCH TO COMPARE
*
*
```

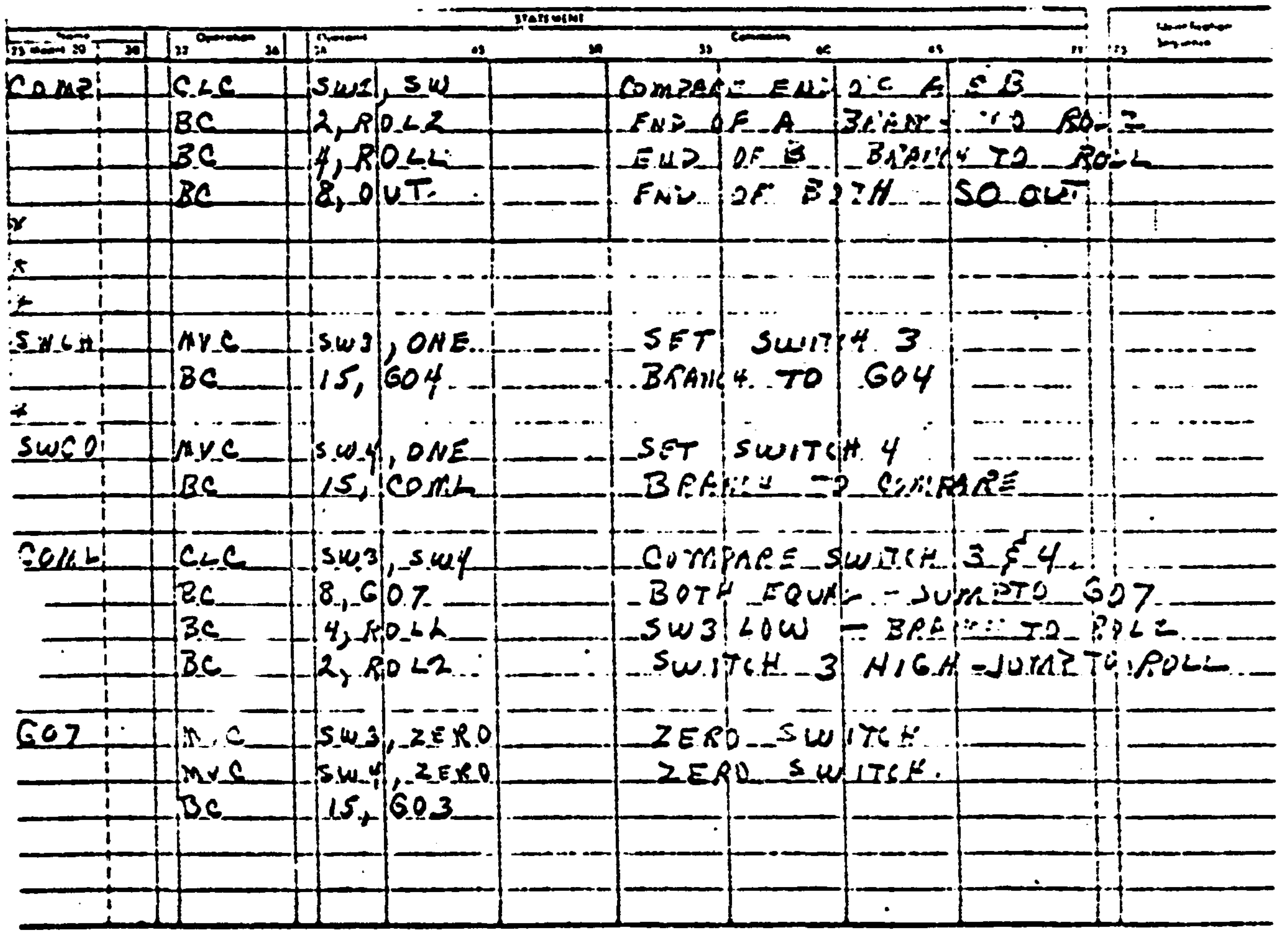
```
GO       CLC   0(2,9),0(10)   COMPARE FILE A & FILE B
         BC    8,EQAL         BRANCH IF EQUAL.
         BC    4,ALO
         MVC   0(2,11),0(10)  B IS LOW MOVE B TO C.
         AH    10,H2
         BC    15,INCC        INCREMENT FILE C.
*
ALO      MVC   0(2,11),0(9)   MOVE FILE A TO FILEC
         AH    9,H2           INCREMENT FILE A.

INCC     AH    11,H2          INCREMENT FILE C.
         BC    15,GO3
*
EQAL     MVC   0(2,11),0(10)  MOVE FILE B TO FILE C.
         AH    10,H2          INCREMENT FILE B.
         MVC   0(2,11),0(9)   MOVE FILE A TO FILE C.
         AH    9,H2
         CH    9,REND
         BC    8,ALL          BRANCH TO SET SWITCH
         CH    10,REND
         BC    8,ALL          BRANCH TO SET SWITCH
```

Figure M-8

Merging--Double Step-Down



```
ROLL    MVC   0(2,11),0(9)     MOVE FILE A TO FILE C
        AH    11,H2            INCREMENT FILE C
        AH    9,H2             INCREMENT FILE A
        MVC   TEMP,0(9)        MOVE FILE A TO TEMP
        AH    9,H2             INCREMENT FILE A.
        CH    9,TEMP           COMPARE A TO A+2
        BC    4,ROL2           IF A+2 IS LOW, BRANCH TO ROL2
        CH    9,REND           IS IT THE END OF FILE A?
        BC    4,ROLL           IF NO - LOOP TO ROLL
ALL     MVC   SW,ONE           YES - SET SWITCH ONE
        BC    15,COMP          BRANCH TO COMP.
*
*
ROL2    MVC   0(2,11),0(10)    MOVE FILE B TO FILE C.
        AH    11,H2            INCREMENT FILE C
        AH    10,H2            INCREMENT FILE B
        MVC   TEM2,0(10)       MOVE AREA INDEXED [illegible]
        AH    10,H2            INCREMENT FILE B
        CH    10,TEMP          COMPARE B [illegible]
        BC    4,ROLL           IF B+2 LOW - BRANCH TO ROLL
        CH    10,RENH          END OF FILE B
        BC    4,ROL2           NO - BRANCH BACK TO LOOP ROL2
AL2     MVC   SW1,ONE          YES - SET SWITCH 2
        BC    15,COMP          BRANCH TO COMPARE.
```



```
COMP    CLC   SW1,SW           COMPARE END OF A & B
        BC    2,ROL2           END OF A BRANCH TO ROL2
        BC    4,ROLL           END OF B BRANCH TO ROLL
        BC    8,OUT            END OF BOTH SO OUT
*
*
*
SWCH    MVC   SW3,ONE          SET SWITCH 3
        BC    15,GO4           BRANCH TO GO4
*
SWC0    MVC   SW4,ONE          SET SWITCH 4
        BC    15,COML          BRANCH TO COMPARE

COML    CLC   SW3,SW4          COMPARE SWITCH 3 & 4.
        BC    8,GO7            BOTH EQUAL - JUMP TO GO7
        BC    4,ROLL           SW3 LOW - BRANCH TO ROLL
        BC    2,ROL2           SWITCH 3 HIGH - JUMP TO ROLL

GO7     MVC   SW3,ZERO         ZERO SWITCH
        MVC   SW4,ZERO         ZERO SWITCH.
        BC    15,GO3
```

Figure M-8 (Part-2)

2.7 SEARCH

Search structures are concerned with the retrieval of data from data structures. The key plays an important part, because the key is the part of the record which is searched. Frequently, large data structures must be processed and tabulated.

A search structure consists of a search process together with a data structure method. Generally, it takes work on both of these, if either one is affected or improved; therefore it is quite important for the programmer to be cognizant of the system as a whole, not just the search, but the search and the structure also.

The selection of a search structure does in fact determine the search and insert and delete methods for a particular data base.

The speeds of these methods are also an important determining factor for selecting a search structure.

The linear search, as presented with the sequential file previously, examines each key in sequence and is terminated with a matching key, or when the last item in a file matches, a termination is effected.

Generally the number of items tested depends on the number of items in the complete data base. The number of items actually tested during a linear search is estimate at N/2.

2.7.1 LINEAR SEARCH

A search is made of the array TH of N integers for one that is equal to Q.

2.7.2 BINARY SEARCH

The binary search provides a fast search process through the ordering of the data base, however the insert and delete process is relatively slow. The continual ordering and reordering of the data base each time a new item is added or taken out, is relatively slow.

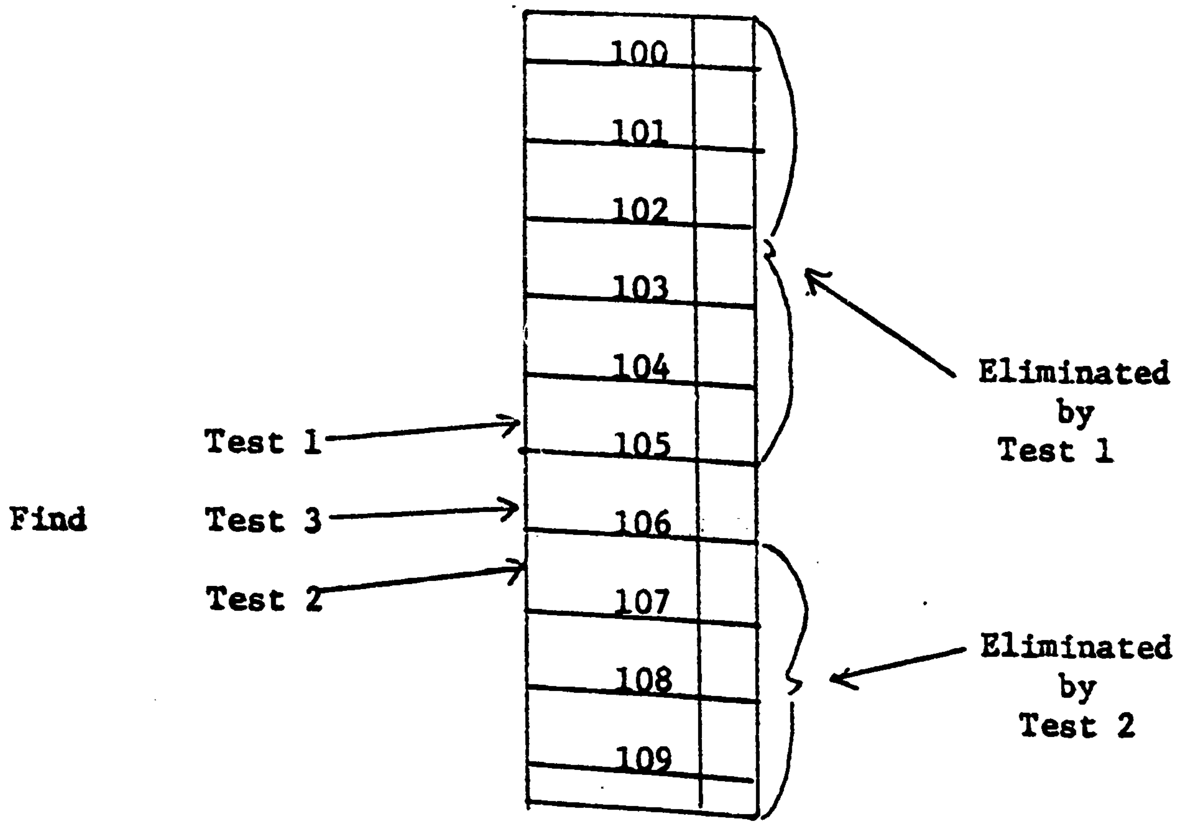


Figure BS-1

The search starts by entering the file or list near the middle. The word that divides the list in two is sometimes called a fence (a form of partitioning). By comparing a fence key with the key of the desired record, a decision is made whether to look further in the sublist above or below. If the sublist above is taken, the list is divided by 2, (X/2), and another fence is created at this location. This process continues until either a list top, a list bottom, or matching record is determined.

Binary Search

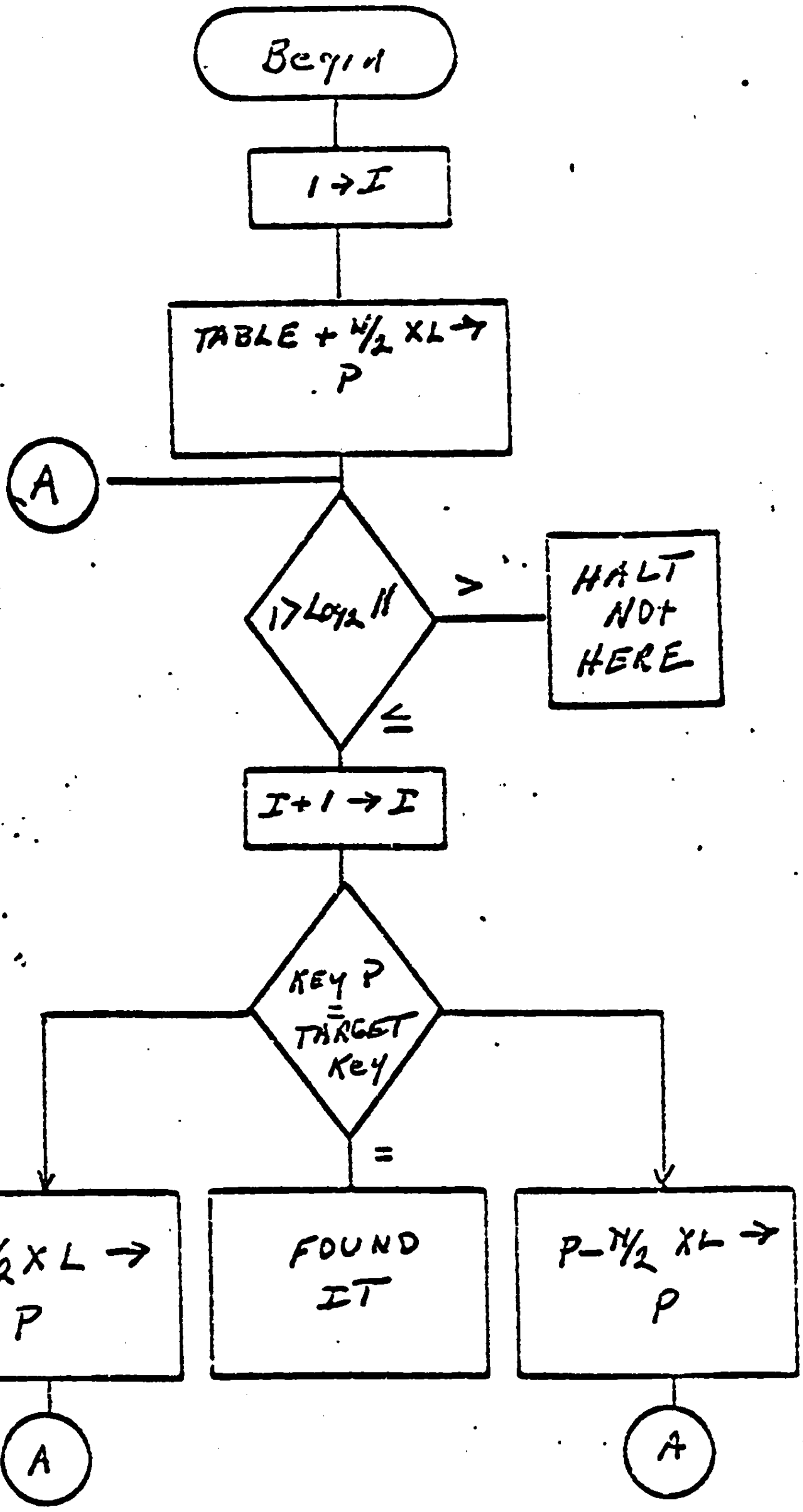


Table = First address of table

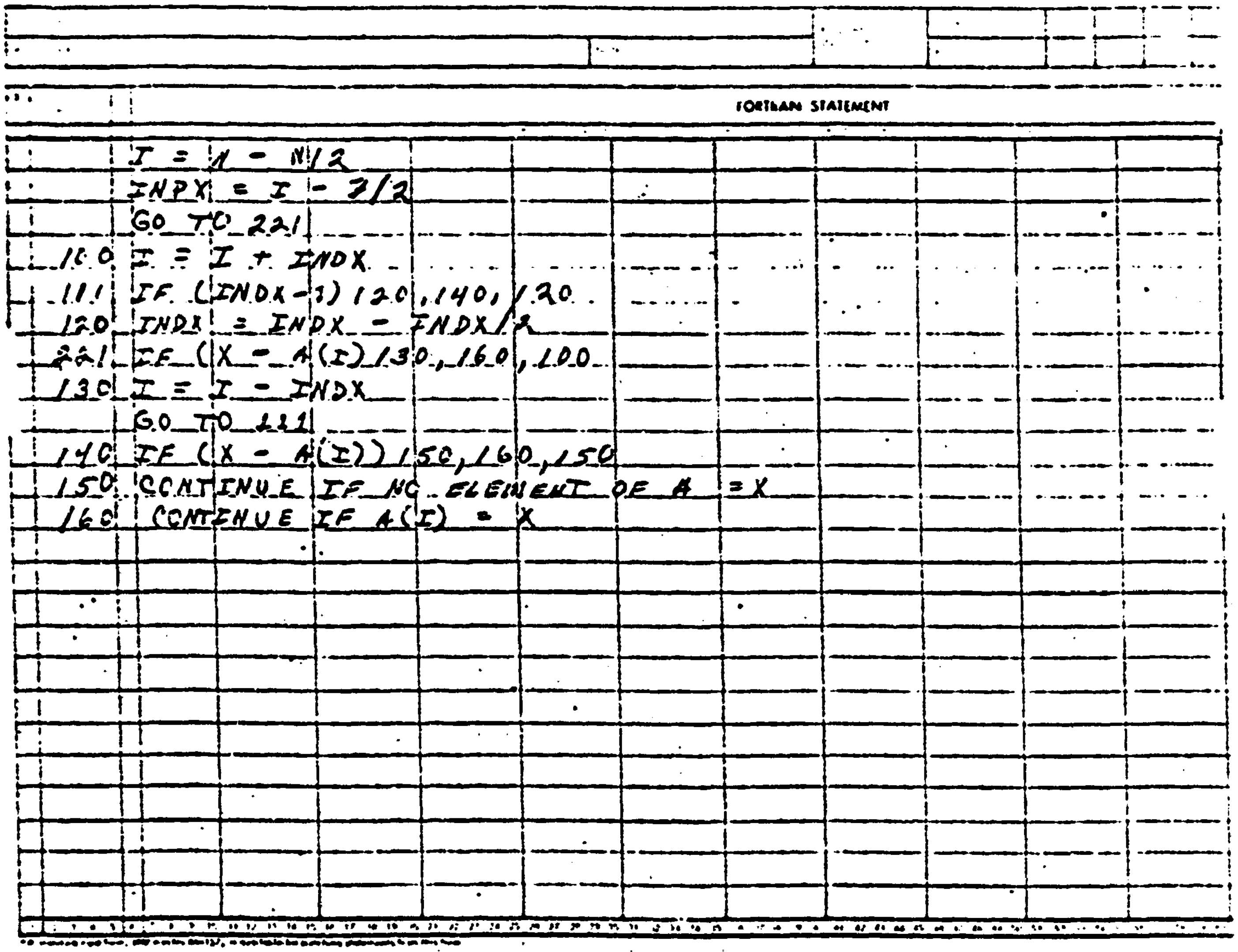
L = Length of entry in table

P = Pointer

I = Iteration

Figure BS-2

Binary Search

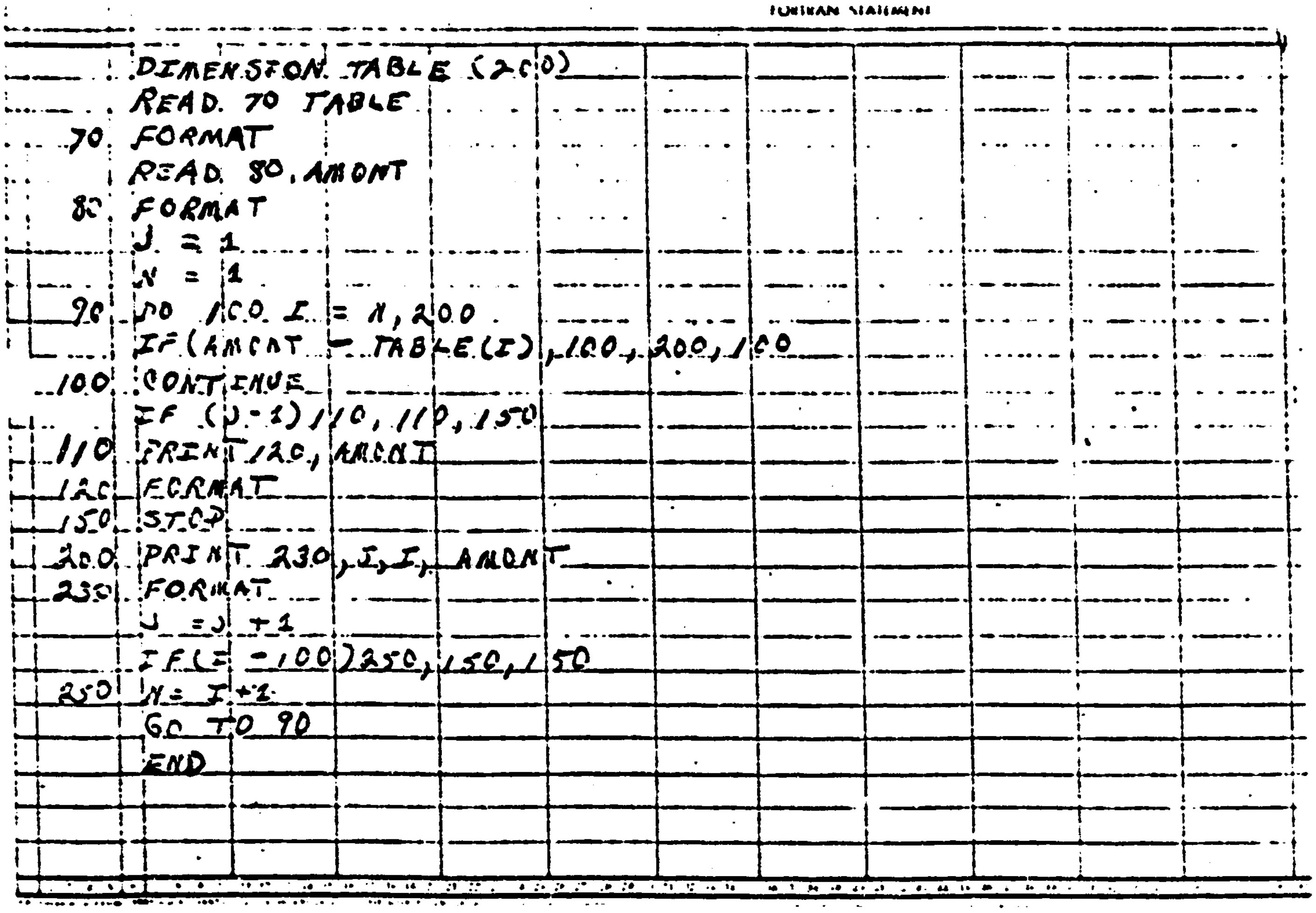


FORTRAN STATEMENT

```
      I = 1 - N/2
      INDX = I - 1/2
      GO TO 221
100   I = I + INDX
111   IF (INDX-1) 120,140,120
120   INDX = INDX - INDX/2
221   IF (X - A(I) 130, 160, 100
130   I = I - INDX
      GO TO 221
140   IF (X - A(I)) 150,160,150
150   CONTINUE IF NO ELEMENT OF A = X
160    CONTINUE IF A(I) = X
```

Figure BS-3

Table Search Program



FORTRAN STATEMENT

```
      DIMENSION TABLE (200)
      READ 70 TABLE
   70 FORMAT
      READ 80, AMONT
   80 FORMAT
      J = 1
      N = 1
   90 DO 100 I = N, 200
      IF (AMONT - TABLE(I)), 100, 200, 100
  100 CONTINUE
      IF (J-1) 110, 110, 150
  110 PRINT 120, AMONT
  120 FORMAT
  150 STOP
  200 PRINT 230, I, I, AMONT
  230 FORMAT
      J = J + 1
      IF (J -100) 250, 150, 150
  250 N = I + 1
      GO TO 90
      END
```

Figure TS-1

2.8 DYNAMIC STORAGE ALLOCATION

The list processing languages have a common feature, which is, memory space for data structures does not have to be preassigned. The storage for each structure is allocated as it is needed and usually not sequentially. This is accomplished by the linking process.

In order to be able to reassign the use of memory cells during execution of a list processing program, a list processing language must provide for:

1. A storage of cells available for use.
2. Systems for obtaining "new" cells from, and returning unneeded cells to the store.

We will see that this also applies to virtual memory, which follows this section. Knuth mentions two methods for storage allocation but considers the First Fit Method the best.

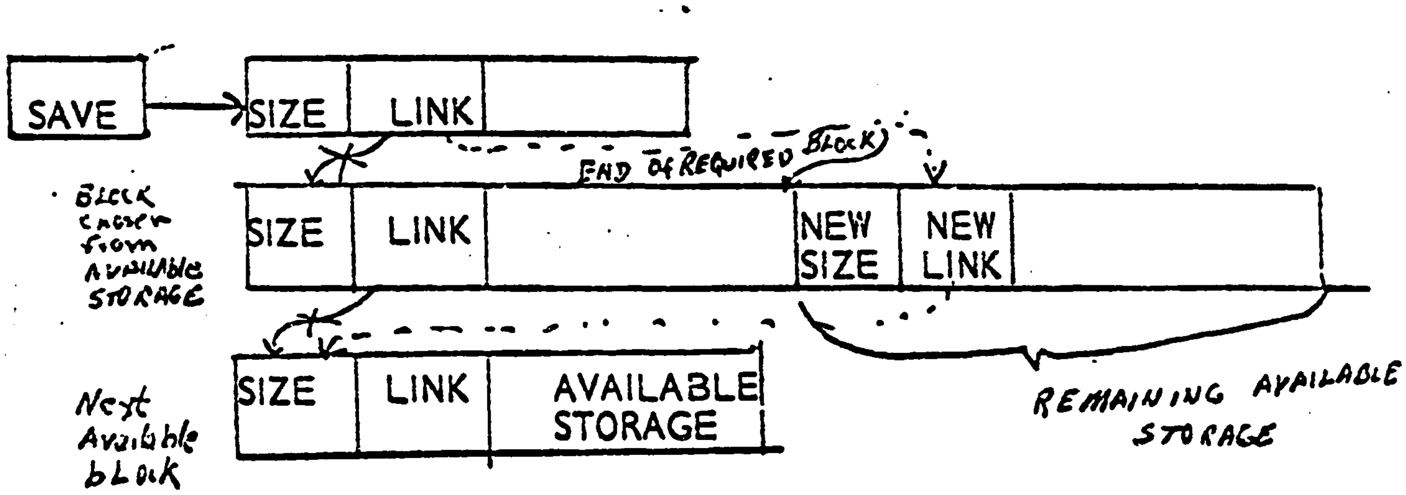


Figure DS-1

Dynamic Storage Allocation

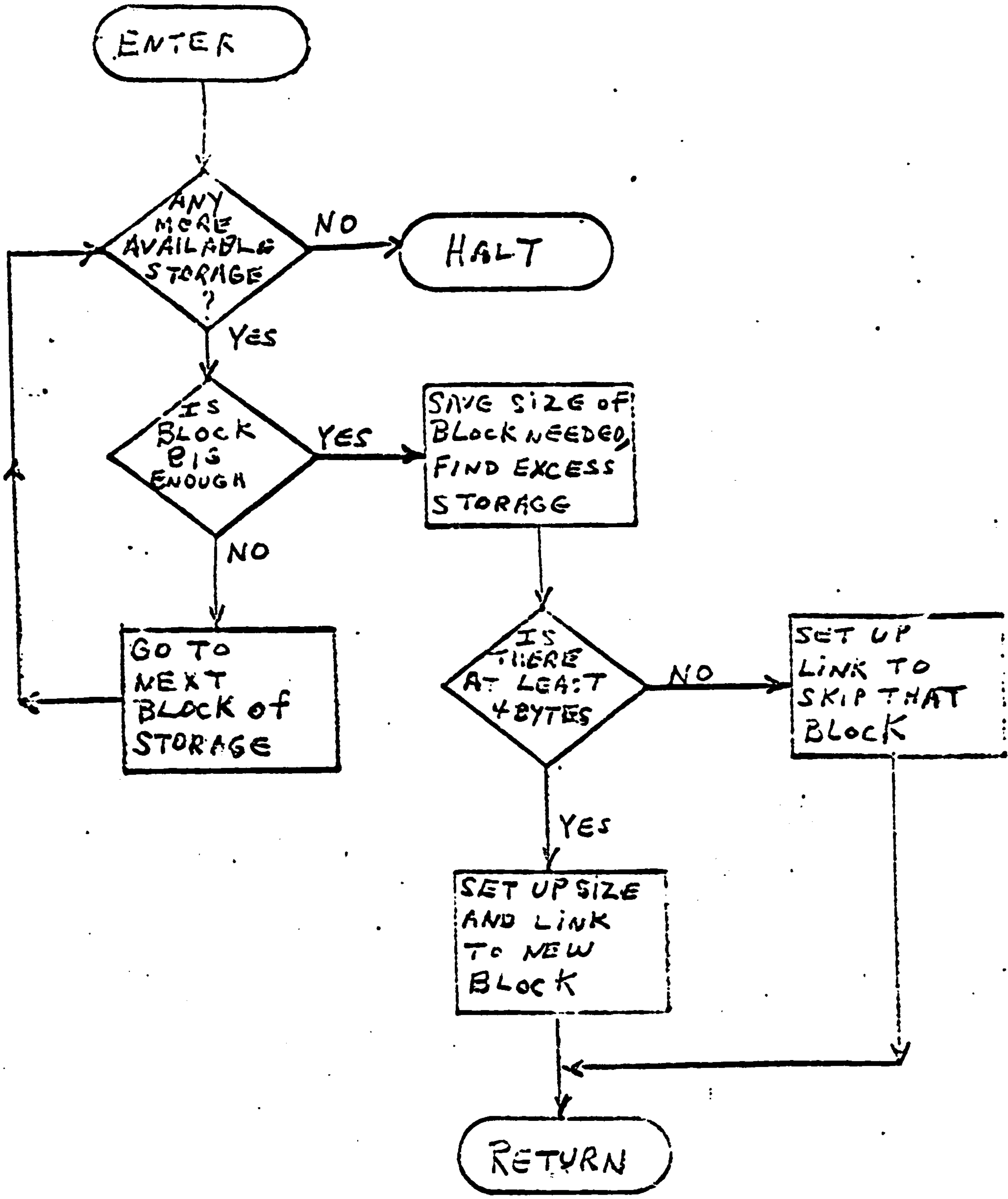
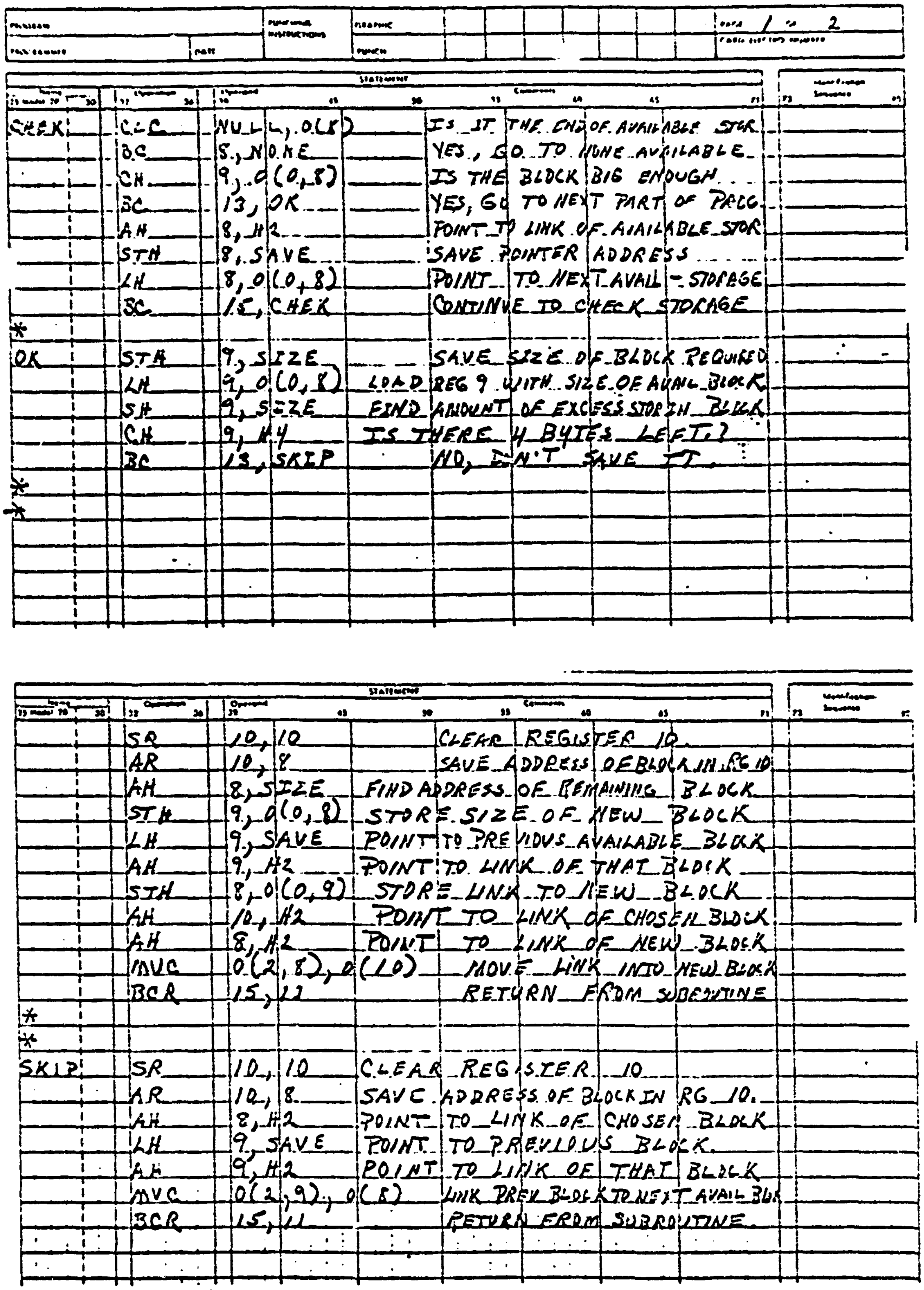

ENTER
ANY MORE AVAILABLE STORAGE ?
NO
HALT
YES
IS BLOCK BIG ENOUGH
YES
SAVE SIZE of BLOCK NEEDED, FIND EXCESS STORAGE
NO
GO TO NEXT BLOCK of STORAGE
IS THERE AT LEAST 4 BYTES
NO
SET UP LINK TO SKIP THAT BLOCK
YES
SET UP SIZE AND LINK TO NEW BLOCK
RETURN

Figure DS-2

Dynamic Storage Allocation



PAGE 1 OF 2

```
CHEK    CLC   NULL,0(8)       IS IT THE END OF AVAILABLE STOR
        BC    8,NONE          YES, GO TO NONE AVAILABLE
        CH    9,0(0,8)        IS THE BLOCK BIG ENOUGH
        BC    13,OK           YES, GO TO NEXT PART OF PROG.
        AH    8,H2            POINT TO LINK OF AVAILABLE STOR
        STH   8,SAVE          SAVE POINTER ADDRESS
        LH    8,0(0,8)        POINT TO NEXT AVAIL - STORAGE
        BC    15,CHEK         CONTINUE TO CHECK STORAGE
*
OK      STH   9,SIZE          SAVE SIZE OF BLOCK REQUIRED
        LH    9,0(0,8)   LOAD REG 9 WITH SIZE OF AVAIL BLOCK
        SH    9,SIZE     FIND AMOUNT OF EXCESS STOR IN BLOCK
        CH    9,H4       IS THERE 4 BYTES LEFT?
        BC    13,SKIP         NO, DON'T SAVE IT.
*
*

        SR    10,10           CLEAR REGISTER 10.
        AR    10,8            SAVE ADDRESS OF BLOCK IN RG 10
        AH    8,SIZE     FIND ADDRESS OF REMAINING BLOCK
        STH   9,0(0,8)   STORE SIZE OF NEW BLOCK
        LH    9,SAVE     POINT TO PREVIOUS AVAILABLE BLOCK
        AH    9,H2       POINT TO LINK OF THAT BLOCK
        STH   8,0(0,9)   STORE LINK TO NEW BLOCK
        AH    10,H2      POINT TO LINK OF CHOSEN BLOCK
        AH    8,H2       POINT TO LINK OF NEW BLOCK
        MVC   0(2,8),0(10)    MOVE LINK INTO NEW BLOCK
        BCR   15,11           RETURN FROM SUBROUTINE
*
*
SKIP    SR    10,10      CLEAR REGISTER 10
        AR    10,8       SAVE ADDRESS OF BLOCK IN RG 10.
        AH    8,H2       POINT TO LINK OF CHOSEN BLOCK
        LH    9,SAVE     POINT TO PREVIOUS BLOCK.
        AH    9,H2       POINT TO LINK OF THAT BLOCK
        MVC   0(2,9),0(8)     LINK PREV BLOCK TO NEXT AVAIL BLK
        BCR   15,11           RETURN FROM SUBROUTINE.
```

Figure DS-2

2.9 VIRTUAL MEMORY

This discussion is based on a paper by Denning,[1] who has made a survey of virtual memory.

Earlier developments brought the state of the art to a point where computer costs would be too great to provide the large memories that were being suggested.

The proposal for the Atlas computer in 1961 set forth a one level store memory. It was known as Virtual Memory. The central idea is that "address" is entirely different from "physical location." The computer hardware and software automatically transfer information into memory at the precise time it is needed for processing. Also the hardware and software arrange for the program-generated addresses to be directed to memory locations that contain the information addressed. Virtual memory presented a great potential for overcoming part of the problems of storage allocation, because the memory use was based on system-observed actual use of space.

The mechanisms for affecting virtual memory now become quite important. The mechanisms referred to here are Segmentation and Paging.

Segmentation organizes address space (not memory) into variable size segments, while Paging organizes address space into fixed size pages of contiguous addresses.

The programmer is allowed to work with addresses which are different from memory, then the system provides the mechanism for translation of these program-generated addresses into the proper memory

[1] Peter J. Denning, "Virtual Memory," *Computing Surveys*, II (September, 1970), 153-189.

location addresses. The programmer uses addresses called "names" or "virtual addresses".

This set of names is called a name space or address space, while the address used by memory is "location" or "memory address".

Since these two types of addresses are different, it requires a mapping system to compact the larger system address space into the smaller memory space.

Address Translation Mechanism

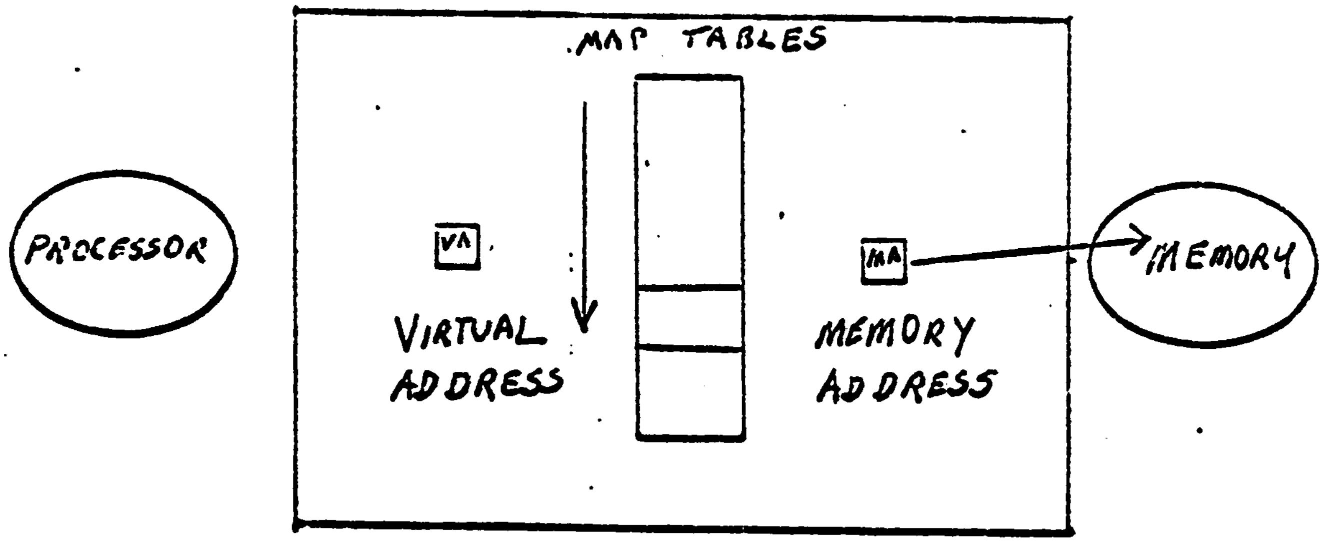


Figure VM-1

The addressing scheme allows the programmer to use a two component address technique which is (s,w). The "s" is the segment and the "w" is the word name which resides within the segment.

The segment is loaded into a contiguous area of memory at base address a. The letter b designates the number of locations s occupies.

Each entry segment is called a descriptor; the sth descriptor contains the base limit information (a,b) for segment s if s is present in memory, and is blank if it is not present.

Segment Table

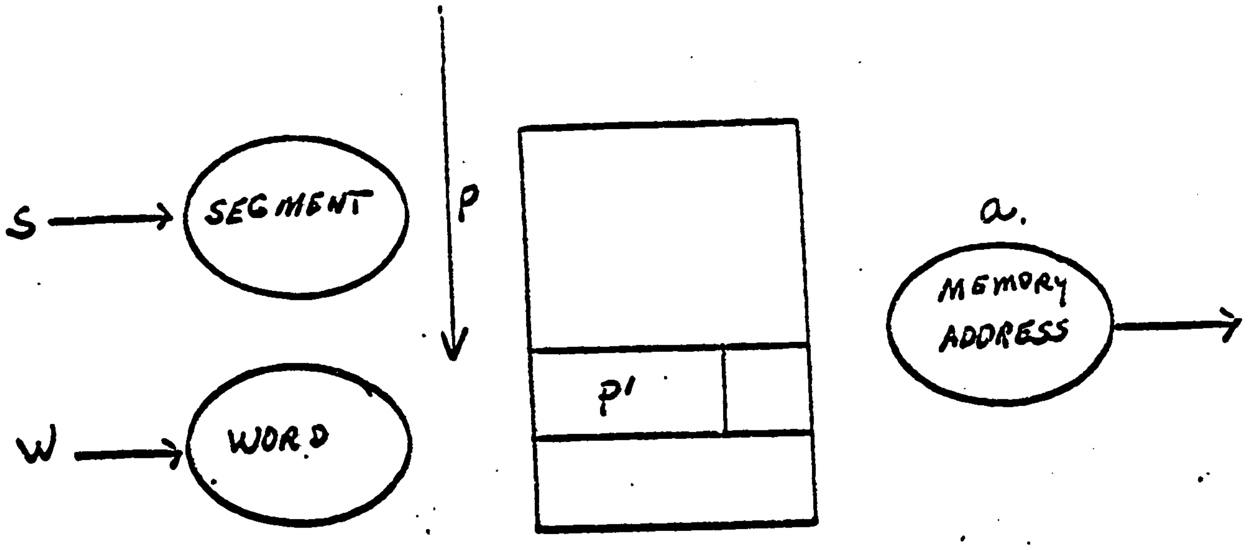


Figure VM-2

Paging divides memory into equal size blocks of locations. "Page Frames" function as sides of residence for matching size blocks of virtual addresses. A Page serves dual functions:

1. Unit of information storage.
2. Transfer between main and auxiliary storage.

Page Frames are identified by a "frame address" (the location of first word of the page frame). The addresses for pages are written (p, w) where p is a page number and w is word number contained in page p.

Page Table

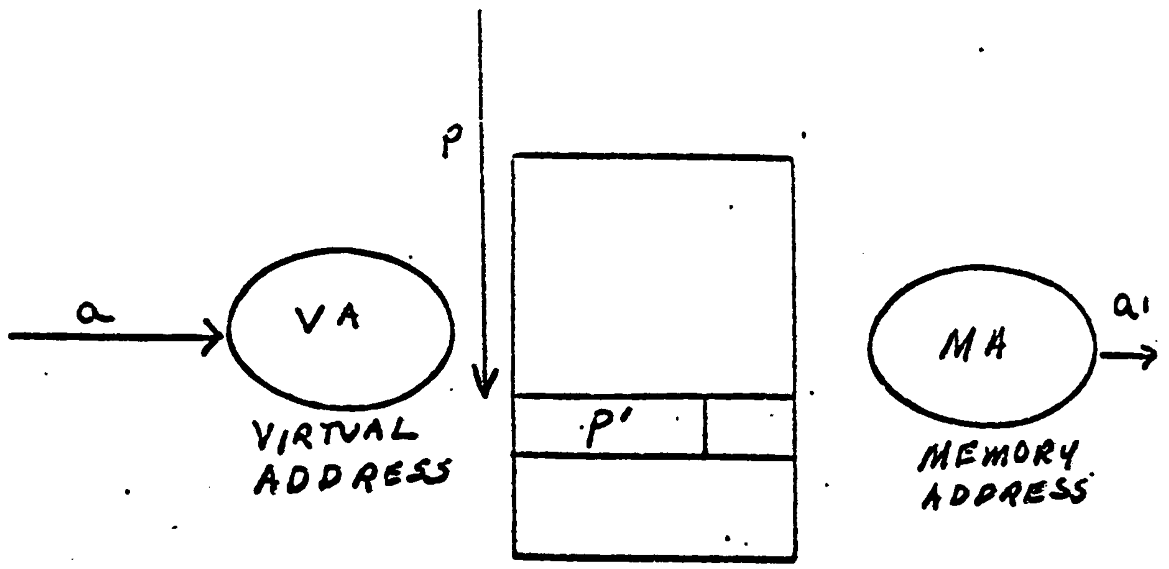


Figure VM-3

Page size was found to work best at forty-five words per page. Segmentation and Paging can be combined, if computer systems have a good selection of register to register operations.

2.3.3.2 RANDOM ORGANIZATION

The Random Orgánization data structure is based on the principle of retrieving and storing records on the basis of a predictable relationship between a key of the record and the address of the location the record is stored in storage media.

This random organization is used in information retrieval and symbol tables, which is to say, a directory or dictionary type process. One method of developing a directory or dictionary type process, is to use the alphabet as an array in which pointers link the directory to the various addresses which partition storage.

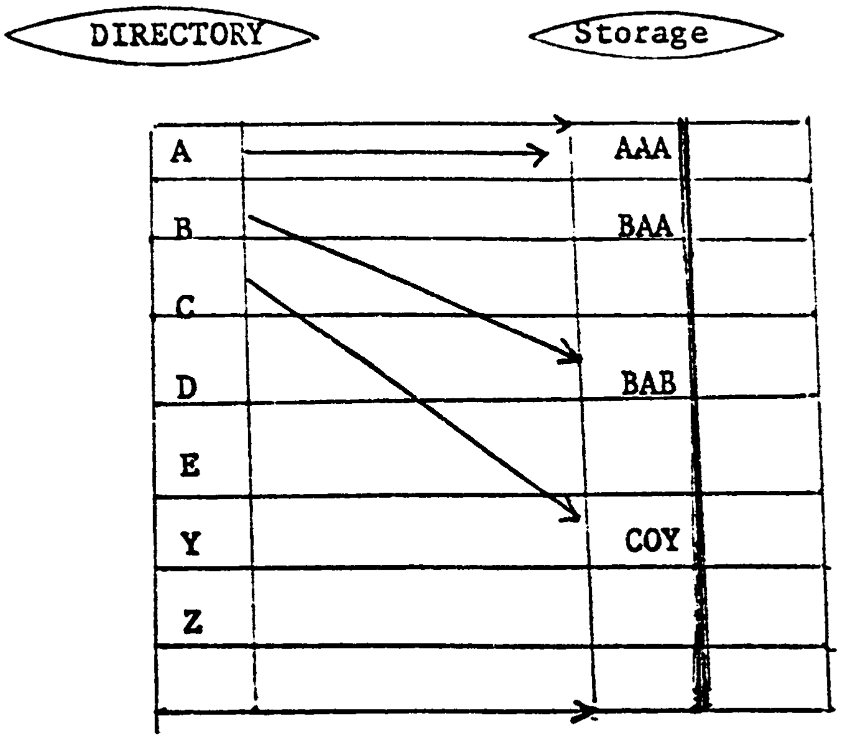


Figure RO-1

The relationship represented here is the binary code of the alphabet letters themselves. The subtables can be chained together. This random organization method has had various names, such as computed table, hash table, key transformation table, but they all achieve the predictable relationship between the key of the record and the location it is to be stored in or retrieved from. The alphabetic method above in Figure RO-1 is an example of the directory look up method. The key of the record was compared to the directory and compare was equal; the direct core address, and the key were brought together.

Another method of dividing the directory is to divide it into ten tables. This method is used frequently with the linear search. This can be done without any ordering of data because the size of the tables will be relatively small. The key is divided by ten, and the remainder (0-9) is used to compute the location to store the data in one of the ten tables. This is sometimes called a hash table.

Another Random Organization method is to choose some of the bits from the middle of the square of the key. This means choose enough bits to be used as an index to determine the address of any item in the table. Since the square depends on all of the bits of the key, different keys will give rise to different hash addresses in the hash table.

When keys are multiword items such as NAME/ADDRESS/LIST LENGTH, it is possible to take the sum of the bits and transform them into an address or location. Also multiplication and division can be performed on parts of the key, but it is important that the calculation does not come out to zero a good percentage of the time.

Another way of presenting the random address method is to cut the key up into n-bit sections, where n is the number of bits needed for the hash address, and then to form the sum of all the sections. The low order n bits of the sum is used as the calculated address. This method can be used for both single-word keys and multi-word keys.

SECTION III

FORTRAN

III FORTRAN

The Fortran section of this book is slanted more toward the programming proficiency required for the DPMA test; however it also includes some Basic Assembler Language examples for comparison.

The first part of the section presents a little chart for the statements that exist in Fortran I, II, and IV. The following material is pertinent to the type required for the test. The remaining material is presented in the form of questions, with answers given.

It is assumed that the reader has a working knowledge of Fortran (at least to the degree that the basic statements are understood). If the reader does not have this facility as yet, he can still develop the proficiency to pass the DPMA test by analyzation of the small programs given herein.

III FORTRAN

1. Introduction
2. Summary of Statements that Exist in Fortran I, II, and IV
3. Expression Formation
4. Hierarchy of Operation
5. Nested Iterative Input and Output
6. Fortran Shortcuts
7. Buffering with Fortran
8. Syntax
9. Array Search
10. Sort Ascending and Descending
11. Installment Note Program

12. Statistics

    12.1 BAL Statistics Program--STAT

13. Selected Fortran Programs in the Form of Answers to Questions

    13.1 Compound Interest Rate

    13.2 Plotting

    13.3 Integration by Simpsons Rule

    13.4 Quadratic Equation

    13.5 Linear Equation

    13.6 Computed Go To

    13.7 The Derivative

    13.8 Evaluate a Polynomial

    13.9 Functions

14. Arithmetic Questions and Answers

2. SUMMARY OF STATEMENTS THAT EXIST IN FORTRAN I, II, AND IV

STATEMENT	FORTRAN I	II	IV
ACCEPT n, list	X	X	
Arithmetic Statement v = a	X	X	X
Arithmetic Statement function name (x, y, z, ..) = h		X	X
ASSIGN i TO n			X
BACKSPACE i		X	X
BLOCK DATA			X
CALL name (h, i, j, ..)		X	X
CALL SWITCH (n, i,)			X
COMMON (h, i, j, ...)		X	X
COMPLEX			X
CONTINUE	X	X	X
DATA ijk.../ e, f, g,../o, p, q, .../			X
DEFINE DISK (n,m)		X	
DIMENSION u(i), v(i),.	X	X	X
DO n i = j, k, m	X	X	X
DOUBLE PRECISION d, e, f,...			X
END	X	X	X
END FILE i		X	X
EQUIVALENCE (i, j, k, ...), (x, y, z, ...)		X	X
EXTERNAL e, f, g,...			
FETCH (i) d, e, f,...		X	
FIND (i)		X	
FORMAT (s, s, s, ...)	X	X	X

STATEMENT	FORTRAN I	II	IV
FUNCTION name (i, j, k, ...)		X	X
GO TO n, (i, j, k, ...)			X
GO TO (J, K, L, ...), i	X	X	X
GO TO n	X	X	X
IF (x) i, j, k (arithmetic)	X	X	X
IF (sense switch n) i, j	X	X	
IF (x) s (Logical)			X
INTEGER a, b, c,			X
LOGICAL a, b, c, ...			X
Logical statement v = x			X
PAUSE	X	X	X
PRINT n, list	X	X	X
PRINT n, iterative list		X	X
PUNCH n, list	X	X	X
PUNCH n, iterative list		X	X
READ n, list	X	X	X
READ n, iterative list		X	X
READ INPUT TAPE i, n, iterative list		X	
READ TAPE i, LIST		X	
READ TAPE i, iterative list		X	
READ (i, n) list			X
READ (i, n) iterative list			X
READ (i) list			X
READ (i) iterative list			X
READ a, b, c,			X

STATEMENT	FORTRAN	I	II	IV
RECORD (i) a, b, c,				X
RETURN			X	X
REWIND i			X	X
STOP		X	X	X
SUBROUTINE name (1k m, n, ...)		X	X	
TYPE n iterative list			X	
WRITE OUTPUT TAPE i, n, list			X	
WRITE OUTPUT TAPE i, n, iterative list			X	
WRITE TAPE i, list			X	
WRITE TAPE i, iterative list		X		
WRITE (i, n) list			X	
WRITE (i, n) iterative list			X	
WRITE (i) list			X	
WRITE (i) iterative list			X	

3. EXPRESSION FORMATION

There are certain rules that will aid the programmer who does not work with Fortran constantly.

1. Variable operands must be previously defined.

   Explanation: This is accomplished by reading them or computing them prior to their use in an arithmetic statement.

2. Operators may not occupy adjacent positions.

   Explanation: They are separated by parentheses.

   ```
   X = A * (-20)
   ```

3. The expression must not be made up of mixed mode.

   Explanation: This can be either fixed or floating point but not both.

   20 * Y creates a mixture of fixed and floating point.

   20. * Y is correct since both are floating point.

4. A value may be assigned an exponent of different mode.

   Explanation: This is one exception to the mixed mode rule.

5. Spacing can be varied.

   Explanation: Items can be written together or spaced apart.

6. Operators may not be "assumed".

   Explanation: This is particularly true in multiplication. They must be written out.

   ```
   10 * Y
   ```

4. HIERARCHY OF OPERATION

As with manual mathematics, the computer uses its own manner and sequence.

1. Parentheses are considered first.
2. Exponentiation is carried out second.

   Explanation: If there are more than one operand to be raised to a power, parentheses must be used.

   Example:

   $(gh)^{10}$ must be written (g. * h.)**10
3. Multiplication and division are carried out in their order of appearance from left to right.

   Example:

   A/3X is written A/ (3. * X)
4. Addition and subtraction are evaluated last, and occupy the same hierarchical level.

Parentheses

Parentheses are used to set off part of the arithmetic expression. The enclosed part is evaluated first.

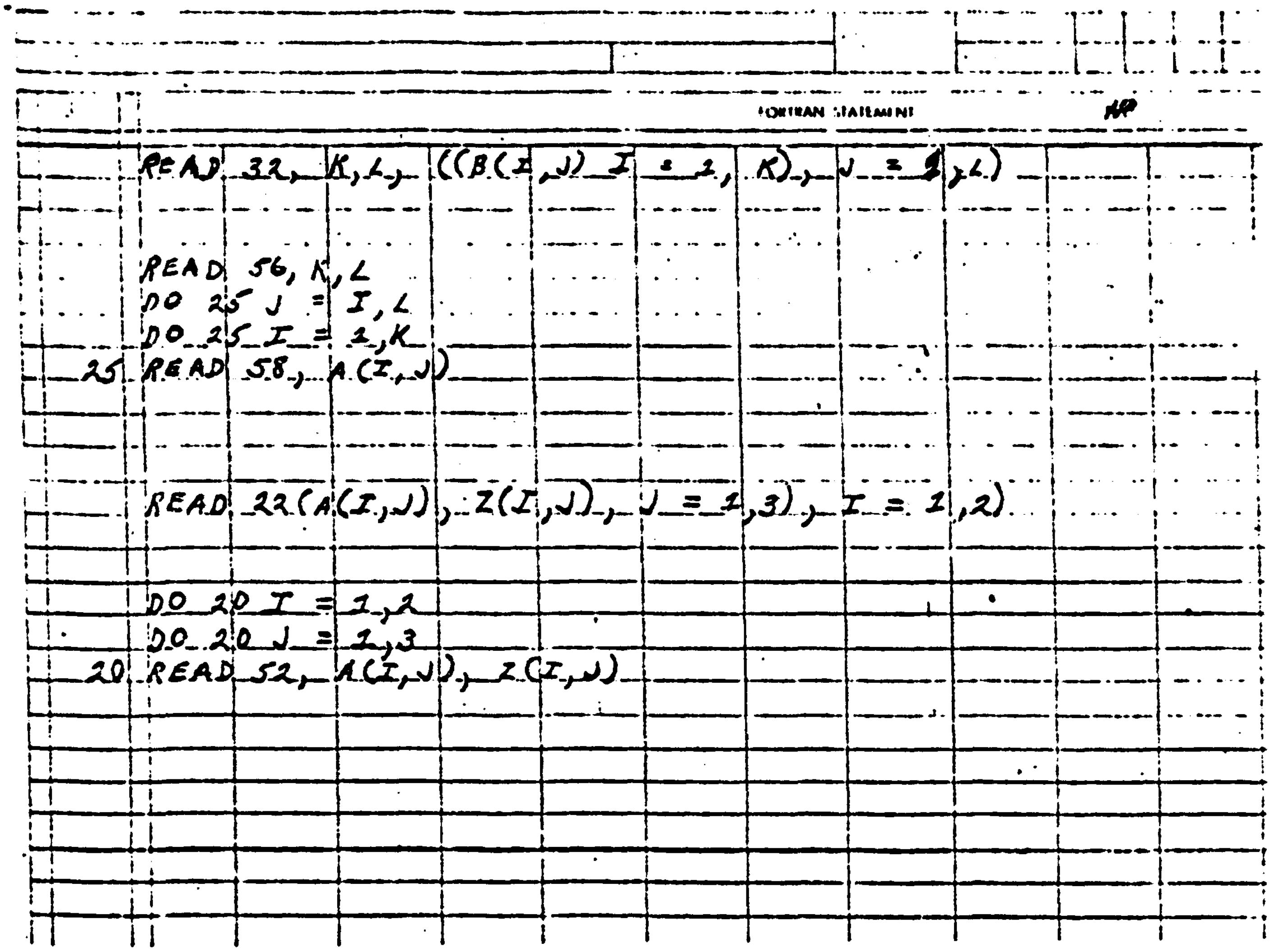
X = D (((C + B) /E** 1/2))

The sum of C and B are calculated first. Their sum is divided by E and raised to the one-half power. D would then be added to the sum.

5. NESTED INPUT/OUTPUT STATEMENTS VERSUS THE DO LOOP EQUIVALENTS

Fortran I uses the DO loop, while Fortran II uses the nested iterative statements. The pattern set by Fortran I is that of making the innermost loop do the number of iterations. This continues with Fortran II by using the innermost nesting for this purpose. This compares to what we learned in data structures.

The do loops are easier to debug however, since the logic is easier to follow. A comparison can be made with the following read, print and punch routines:

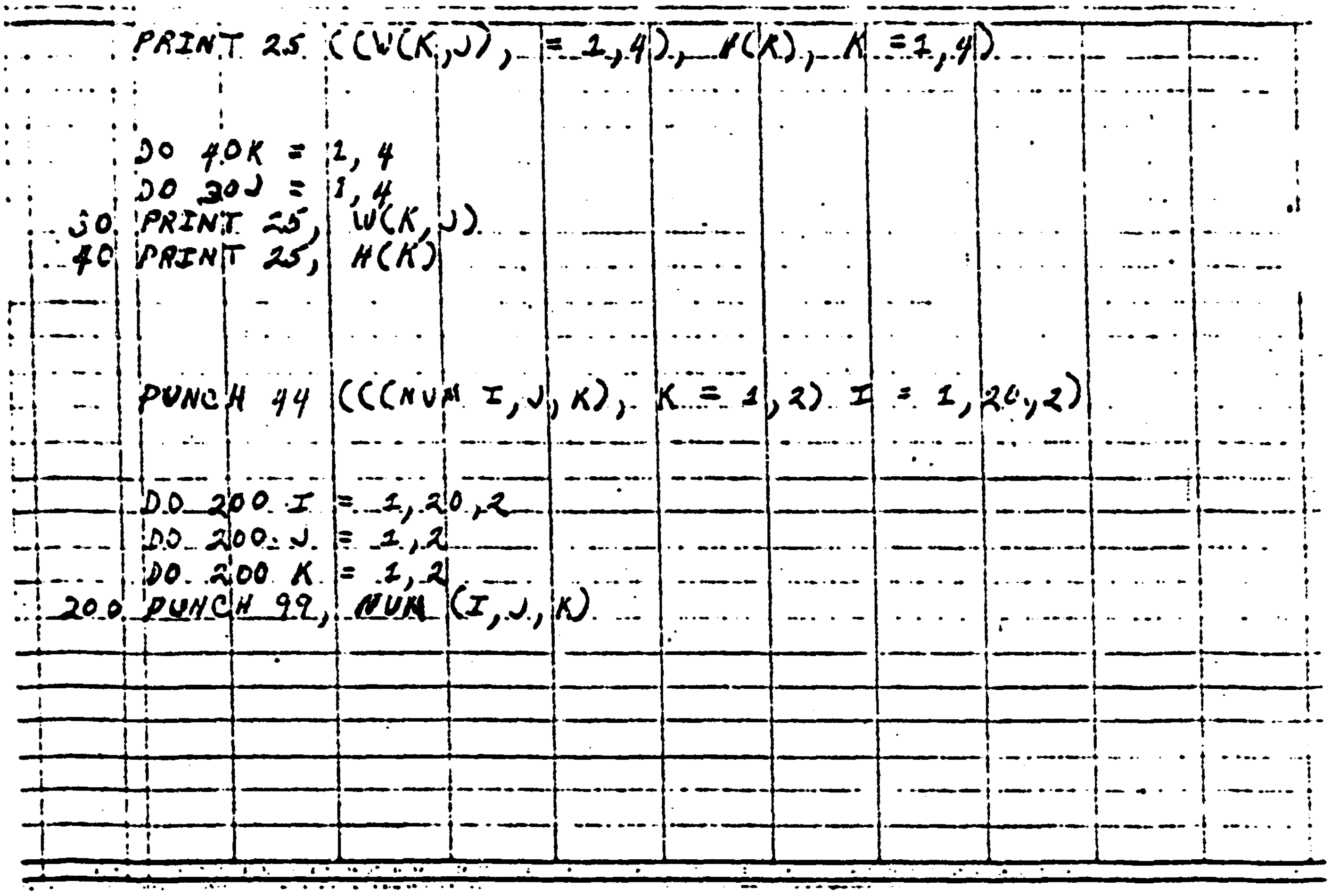


FORTRAN STATEMENT

```
      READ 32, K,L, ((B(I,J) I = 1, K), J = 1,L)

      READ 56, K,L
      DO 25 J = I,L
      DO 25 I = 1,K
   25 READ 58, A(I,J)

      READ 22(A(I,J), Z(I,J), J = 1,3), I = 1,2)

      DO 20 I = 1,2
      DO 20 J = 1,3
   20 READ 52, A(I,J), Z(I,J)
```

Figure FO-5

Nested Input Output Statements (con't.)



```
      PRINT 25 ((W(K,J), = 1,4), H(K), K =1,4)

      DO 40K = 1, 4
      DO 30J = 1, 4
   30 PRINT 25, W(K,J)
   40 PRINT 25, H(K)


      PUNCH 44 (((NUM I,J,K), K = 1,2) I = 1,20,2)

      DO 200 I = 1,20,2
      DO 200 J = 1,2
      DO 200 K = 1,2
  200 PUNCH 99, NUM (I,J,K)
```

Figure FO-5 (Part-2)

6. FORTRAN SHORTCUTS

Space may be saved when a loop has within it an expression, whose variables do not change during the sequence through a loop. The space saving comes from evaluating the expression outside the loop, and retaining the result until it is needed.

```
   DO 60 I = I,N
60 X (I) = Y*Z*W(I)/C
```

An easier method:

```
   HOLD I Y * Z/C
   DO 60 I I,N
60 X(I) = HOLD I * W (I)
```

Repeated calculations can be improved by removing the redundant parts of the expressions.

```
X (C * D/A) * CO S(C * D/A).
```

These redundant expressions can be improved by using an area such as HOLD:

```
HOLDI - C * D/A
X = HOLDI * COS(HOLDI)
```

This makes it possible for the computer to make the computation of the expression one time only.

Polynomial Computation

The Fortran writing of:

```
X = C + D*Y + A* Y**2 + E*Y**3
```

If we analyze the requirements for calculation, we find that there are two exponentiations, three multiplications and three additions. The nested form of writing can be used to write the equation:

X = C + Y*(D+Y*(A+Y*E))

This method of presentation eliminates the multiplication and addition.

7. BUFFERING WITH FORTRAN

Input-output devices such as the teletype, card reader and punch work at speeds much slower than the computer. In the typical installation where computing time is quite costly, it is imperative that the computer be used for computing, with a minimal time for transmission to and from input-output devices.

Most computers now allow the computer to perform computations while data transmission is in process. The output data can be transmitted from memory to an intermediate buffer storage at high speed. This allows the computer to return to its computational tasks while data are being transmitted from the buffer to the output device, at the prescribed timing the output device requires. The process is similar for input data. This can be expanded for auxiliary storage such as disk and tape.

8. SYNTAX

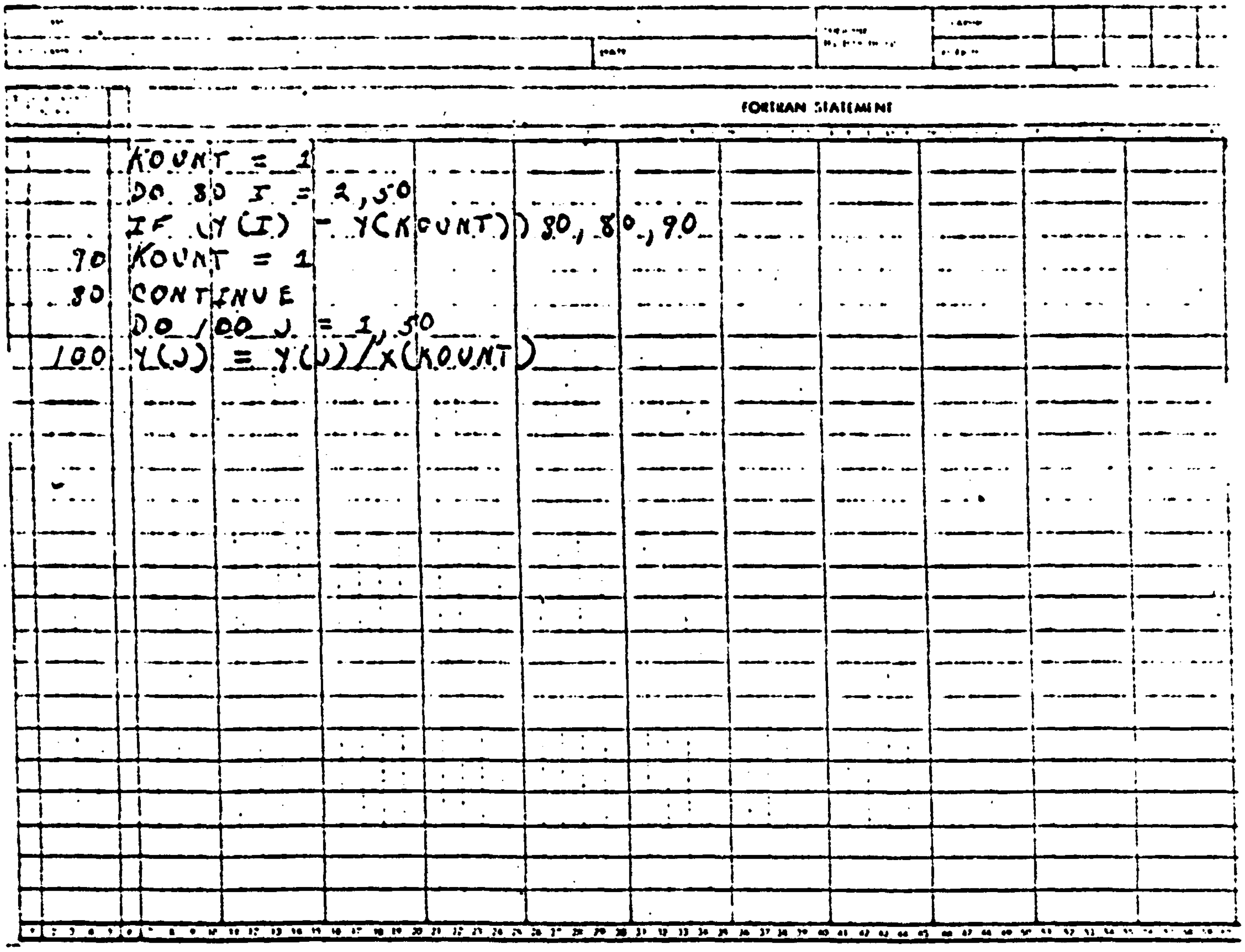
Syntax is a study of language structure, not by a study of words themselves. A language syntax is a set of rules that dictate how the words, or basic elements, of the language are ordered to form meaningful phrases and statements.

Syntax for computer languages is not usually so simple as to have each character for one well-defined use letters representing themselves, symbols representing themselves, and symbols being used their usual way. It is more usual that a single character is used in more than one way. For example, the asterisk is used for powers and also used to imply multiplication.

It is necessary that the algorithms can make decisions. To do so, we must make certain that each expression can be constructed one way only, according to syntactic rules.

9. ARRAY SEARCH

Search array Y, which has fifty elements, for its largest element. When the largest element is found, divide it into all the remaining elements of Y so that finally Y contains only elements with values less than or equal to one.



FORTRAN STATEMENT

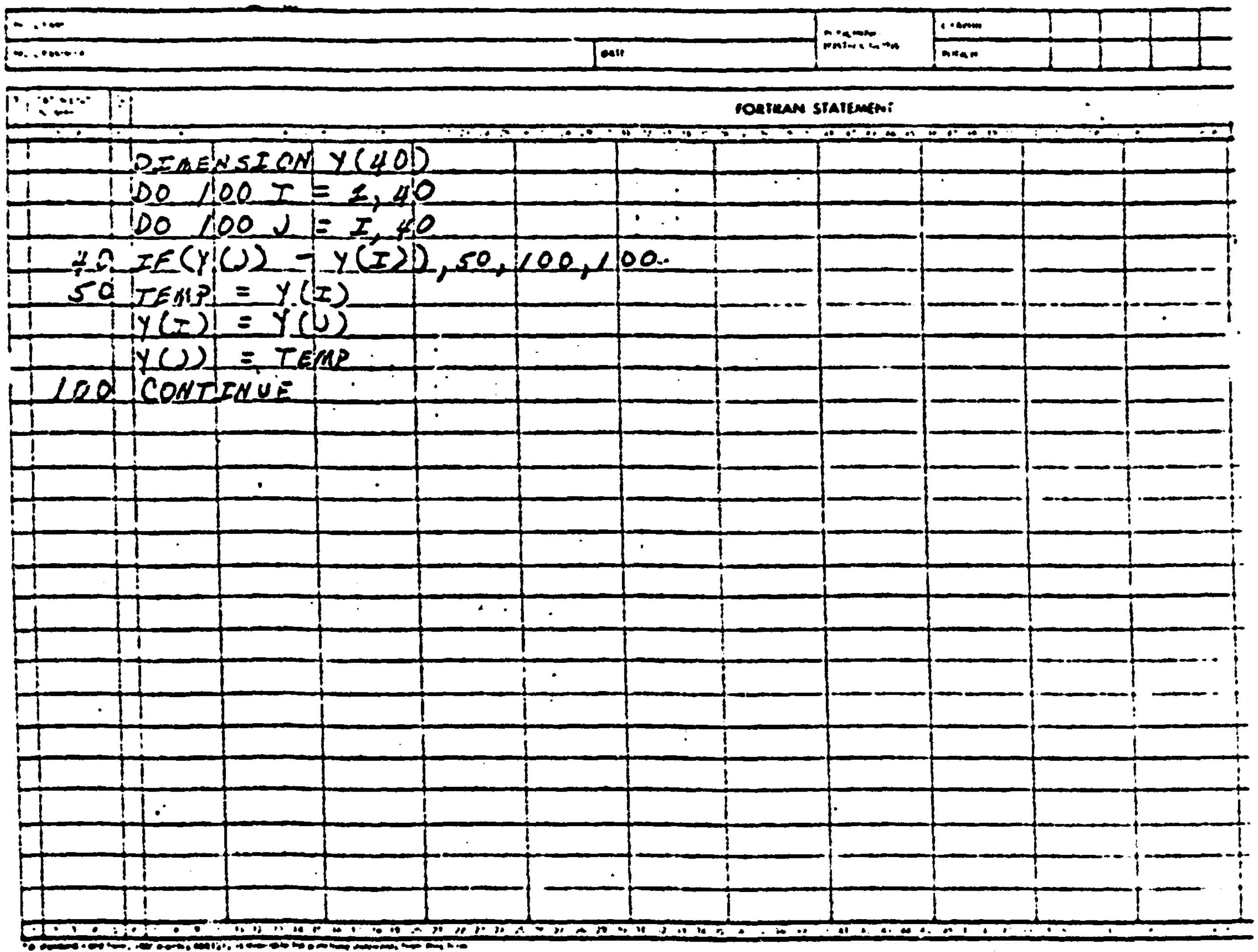
```
      KOUNT = 1
      DO 80 I = 2,50
      IF (Y(I) - Y(KOUNT)) 80, 80, 90
   90 KOUNT = 1
   80 CONTINUE
      DO 100 J = 1,50
  100 Y(J) = Y(J)/X(KOUNT)
```

Figure FO-9

10. SORT ASCENDING AND DESCENDING

A one dimensional array has forty elements. Prepare a program which will rank the array from smallest to largest, also write a program which will rank the array from largest to smallest.

1. Sort from largest to smallest:

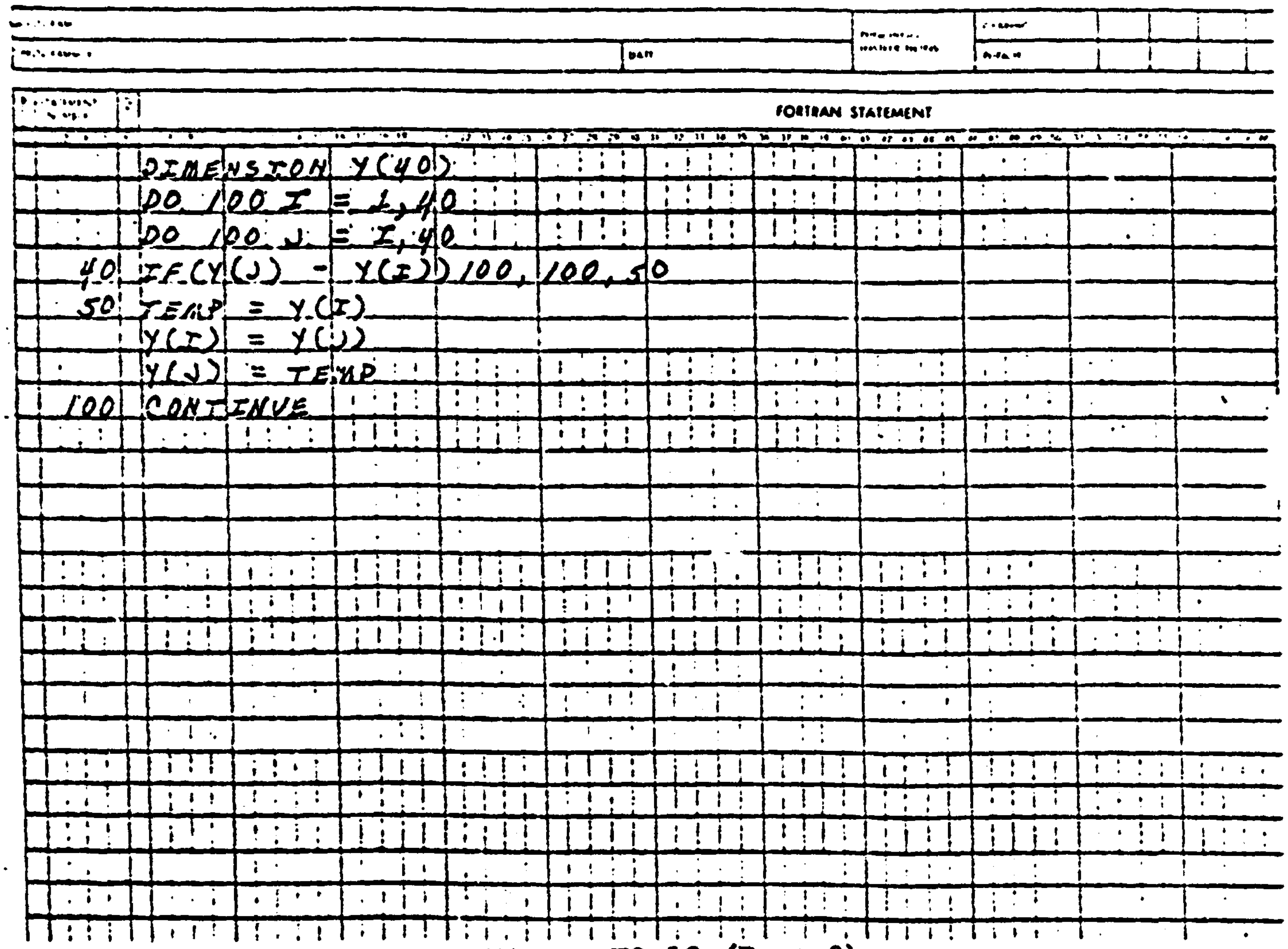


FORTRAN STATEMENT

```
      DIMENSION Y(40)
      DO 100 I = 1,40
      DO 100 J = I,40
   40 IF(Y(J) - Y(I)),50,100,100
   50 TEMP = Y(I)
      Y(I) = Y(J)
      Y(J) = TEMP
  100 CONTINUE
```

Figure FO-10

2. Sort from largest to smallest.



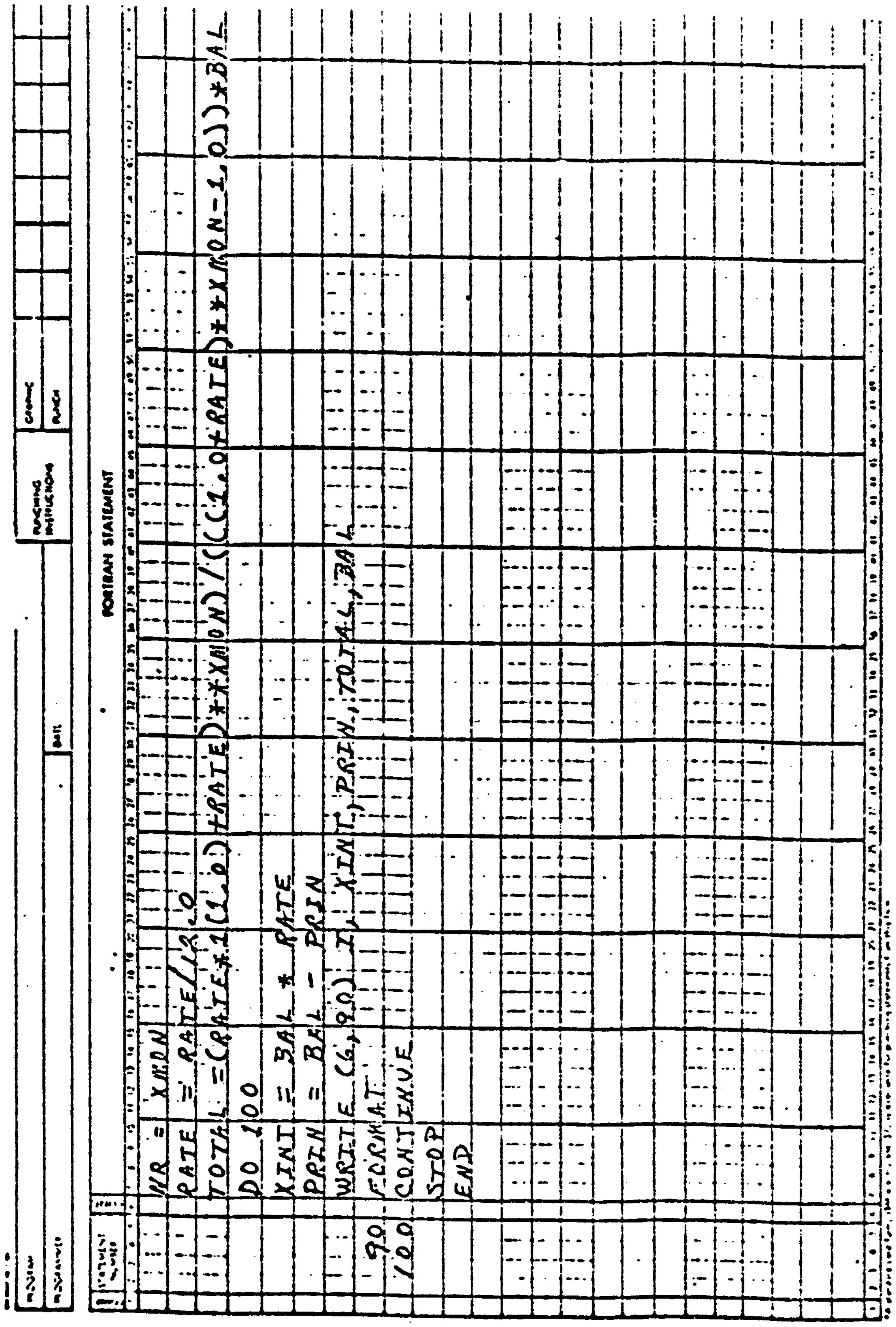
FORTRAN STATEMENT

```
      DIMENSION Y(40)
      DO 100 I = 1,40
      DO 100 J = I,40
   40 IF(Y(J) - Y(I))100,100,50
   50 TEMP = Y(I)
      Y(I) = Y(J)
      Y(J) = TEMP
  100 CONTINUE
```

Figure FO-10 (Part-2)

The outer DO loop, with one as index, ranks the elements one at a time (either largest to smallest or smallest to largest). Once an element is found, it is stored in Y (1). Then to get the next rank, the inner DO loop must examine Y (1+1) through Y (40). The inner DO loop searches the remaining elements each time to find the smallest (or largest).

11. INSTALLMENT NOTE PROGRAM

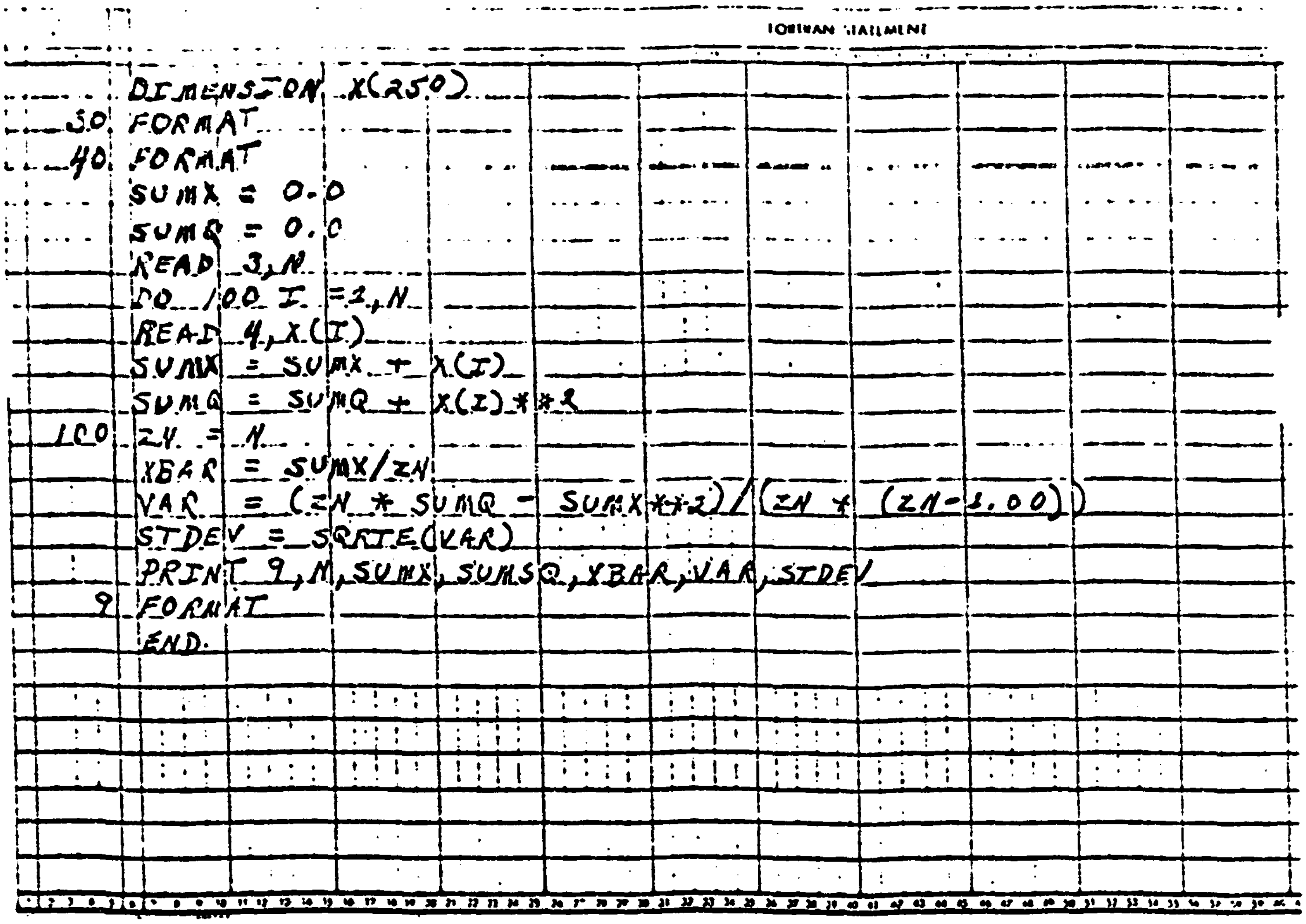

FORTRAN STATEMENT

```
      NR = XMON
      RATE = RATE/12.0
      TOTAL =(RATE*1(1.0)+RATE)**XMON)/(((1.0+RATE)**XMON-1.0))*BAL
      DO 100
      XINT = BAL * RATE
      PRIN = BAL - PRIN
      WRITE (6,90) I, XINT,PRIN,TOTAL,BAL
   90 FORMAT
  100 CONTINUE
      STOP
      END
```

Figure FO-11

12. STATISTICS

Mean, Variance, Standard Deviation

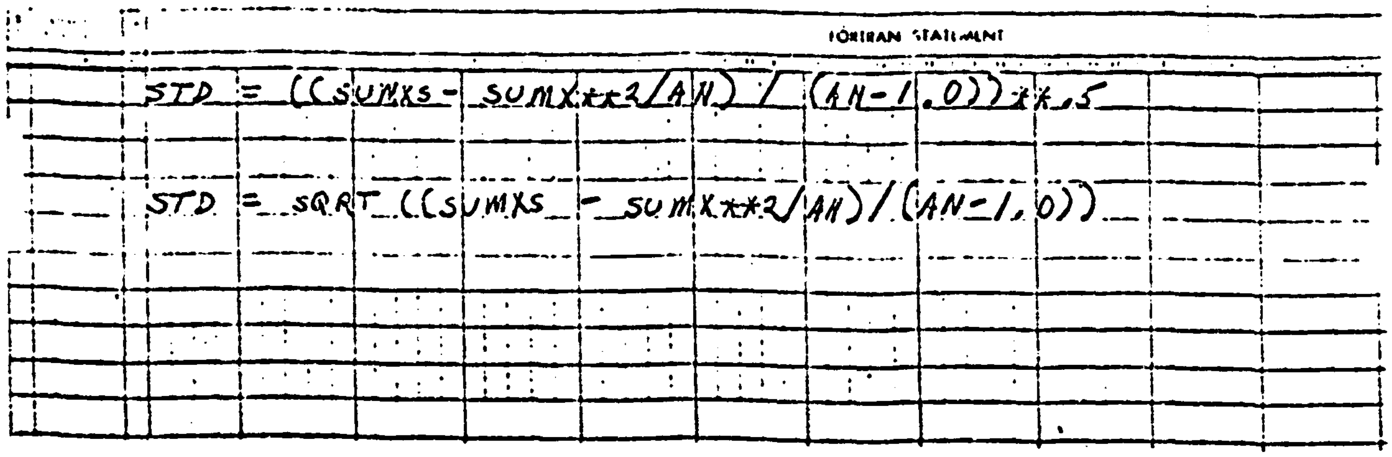


FORTRAN STATEMENT

```
      DIMENSION X(250)
   50 FORMAT
   40 FORMAT
      SUMX = 0.0
      SUMQ = 0.0
      READ 3,N
      DO 100 I =1,N
      READ 4,X(I)
      SUMX = SUMX + X(I)
      SUMQ = SUMQ + X(I)**2
  100 ZN = N
      XBAR = SUMX/ZN
      VAR = (ZN * SUMQ - SUMX**2)/(ZN * (ZN-1.00))
      STDEV = SQRTF(VAR)
      PRINT 9,N,SUMX,SUMSQ,XBAR,VAR,STDEV
    9 FORMAT
      END
```

Figure FO-12

Two Approaches to the Standard Deviation

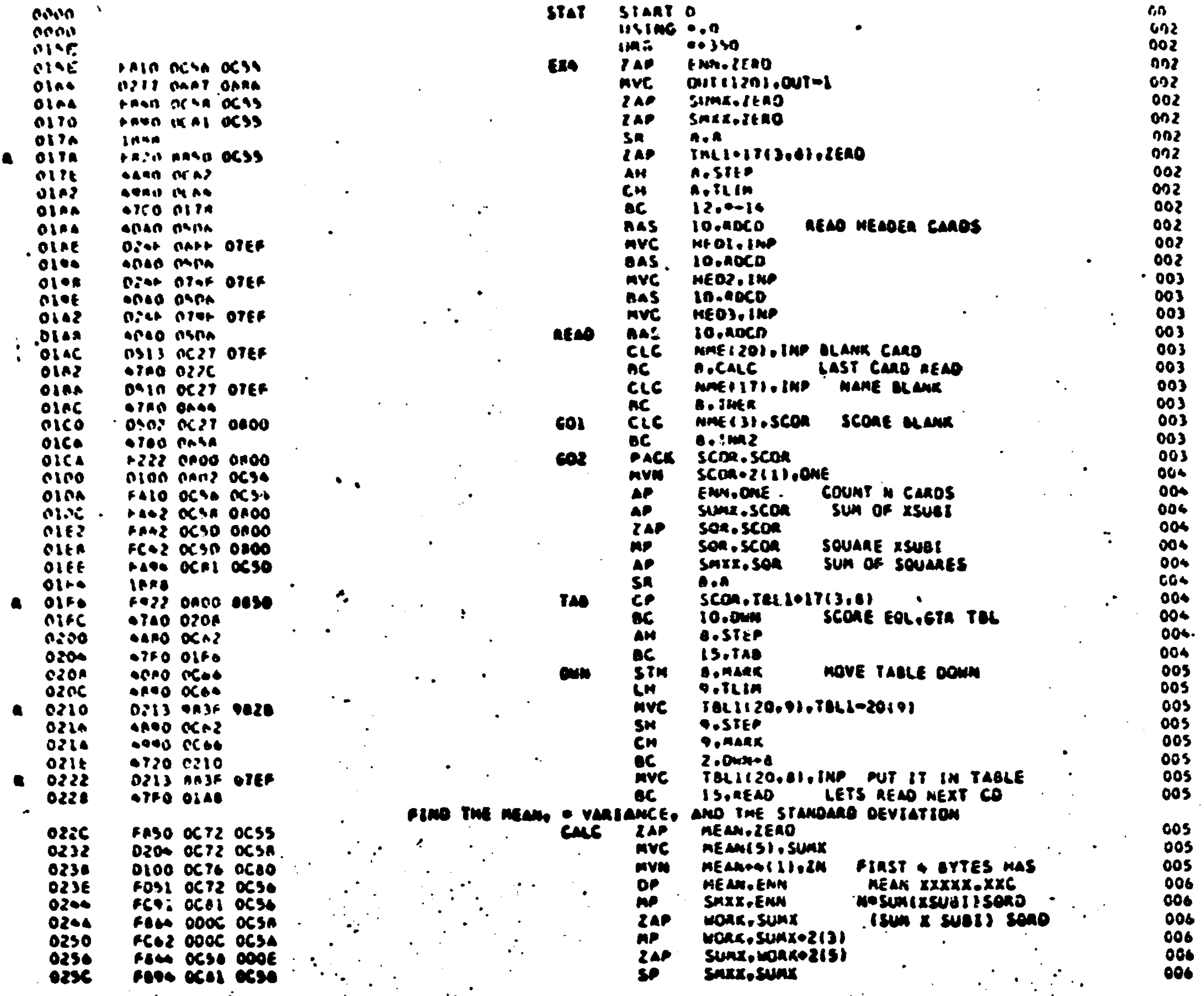


FORTRAN STATEMENT

```
      STD = ((SUMXS - SUMX**2/AN) / (AN-1.0))**.5

      STD = SQRT ((SUMXS - SUMX**2/AN)/(AN-1.0))
```

Figure FO-12 (Part-2)

Statistics Program



```
  0000                                 STAT   START 0                                       00
  0000                                        USING *,0                                     002
  015E                                        ORG   *+350                                   002
  015E    F810 0C56 0C55               EX4    ZAP   ENN,ZERO                                002
  0164    D213 0A87 0A86                      MVC   OUT+1(20),OUT-1                         002
  016A    F840 0C58 0C55                      ZAP   SUMX,ZERO                               002
  0170    F890 0C81 0C55                      ZAP   SMXX,ZERO                               002
  0176    1B88                                SR    8,8                                     002
A 0178    F820 8A50 0C55                      ZAP   TBL1+17(3,8),ZERO                       002
  017E    4A80 0CA2                           AH    8,STEP                                  002
  0182    4980 0CA4                           CH    8,TLIM                                  002
  0186    47C0 0178                           BC    12,*-14                                 002
  018A    4DA0 05DA                           BAS   10,RDCD     READ HEADER CARDS           002
  018E    D24F 0AFF 07EF                      MVC   HED1,INP                                002
  0194    4DA0 05DA                           BAS   10,RDCD                                 002
  0198    D24F 0B4F 07EF                      MVC   HED2,INP                                003
  019E    4DA0 05DA                           BAS   10,RDCD                                 003
  01A2    D24F 0B9F 07EF                      MVC   HED3,INP                                003
  01A8    4DA0 05DA                    READ   BAS   10,RDCD                                 003
  01AC    D513 0C27 07EF                      CLC   NME(20),INP BLANK CARD                  003
  01B2    4780 022C                           BC    8,CALC      LAST CARD READ              003
  01B6    D510 0C27 07EF                      CLC   NME(17),INP   NAME BLANK                003
  01BC    4780 0A44                           BC    8,INER                                  003
  01C0    D502 0C27 0800              GO1     CLC   NME(3),SCOR   SCORE BLANK               003
  01C6    4780 0A5A                           BC    8,SNR2                                  003
  01CA    F222 0800 0800              GO2     PACK  SCOR,SCOR                               003
  01D0    D100 0802 0C54                      MVN   SCOR+2(1),ONE                           004
  01D6    FA10 0C56 0C54                      AP    ENN,ONE       COUNT N CARDS             004
  01DC    FA42 0C58 0800                      AP    SUMX,SCOR      SUM OF XSUBI             004
  01E2    F842 0C5D 0800                      ZAP   SQR,SCOR                                004
  01E8    FC42 0C5D 0800                      MP    SQR,SCOR      SQUARE XSUBI              004
  01EE    FA94 0C81 0C5D                      AP    SMXX,SQR      SUM OF SQUARES            004
  01F4    1B88                                SR    8,8                                     004
A 01F6    F922 0800 8A50              TAB     CP    SCOR,TBL1+17(3,8)                       004
  01FC    47A0 0208                           BC    10,DWN        SCORE EQL,GTR TBL         004
  0200    4A80 0CA2                           AH    8,STEP                                  004
  0204    47F0 01F6                           BC    15,TAB                                  004
  0208    4080 0CA6                   DWN     STH   8,MARK        MOVE TABLE DOWN           005
  020C    4890 0CA4                           LH    9,TLIM                                  005
A 0210    D213 9A3F 9A2B                      MVC   TBL1(20,9),TBL1-20(9)                   005
  0216    4B90 0CA2                           SH    9,STEP                                  005
  021A    4990 0CA6                           CH    9,MARK                                  005
  021E    4720 0210                           BC    2,DWN+8                                 005
A 0222    D213 8A3F 07EF                      MVC   TBL1(20,8),INP  PUT IT IN TABLE         005
  0228    47F0 01A8                           BC    15,READ       LETS READ NEXT CD         005
                           FIND THE MEAN, * VARIANCE, AND THE STANDARD DEVIATION
  022C    F850 0C72 0C55              CALC    ZAP   MEAN,ZERO                               005
  0232    D204 0C72 0C58                      MVC   MEAN(5),SUMX                            005
  0238    D100 0C76 0C80                      MVN   MEAN+4(1),ZN   FIRST 4 BYTES HAS        005
  023E    FD51 0C72 0C56                      DP    MEAN,ENN        MEAN XXXXX.XXC          006
  0244    FC91 0C81 0C56                      MP    SMXX,ENN       N*SUM(XSUBI)SQRD         006
  024A    F844 000C 0C58                      ZAP   WORK,SUMX       (SUM X SUBI) SQRD       006
  0250    FC42 000C 0C5A                      MP    WORK,SUMX+2(3)                          006
  0256    F844 0C58 000E                      ZAP   SUMX,WORK+2(5)                          006
  025C    FB94 0C81 0C58                      SP    SMXX,SUMX                               006
```

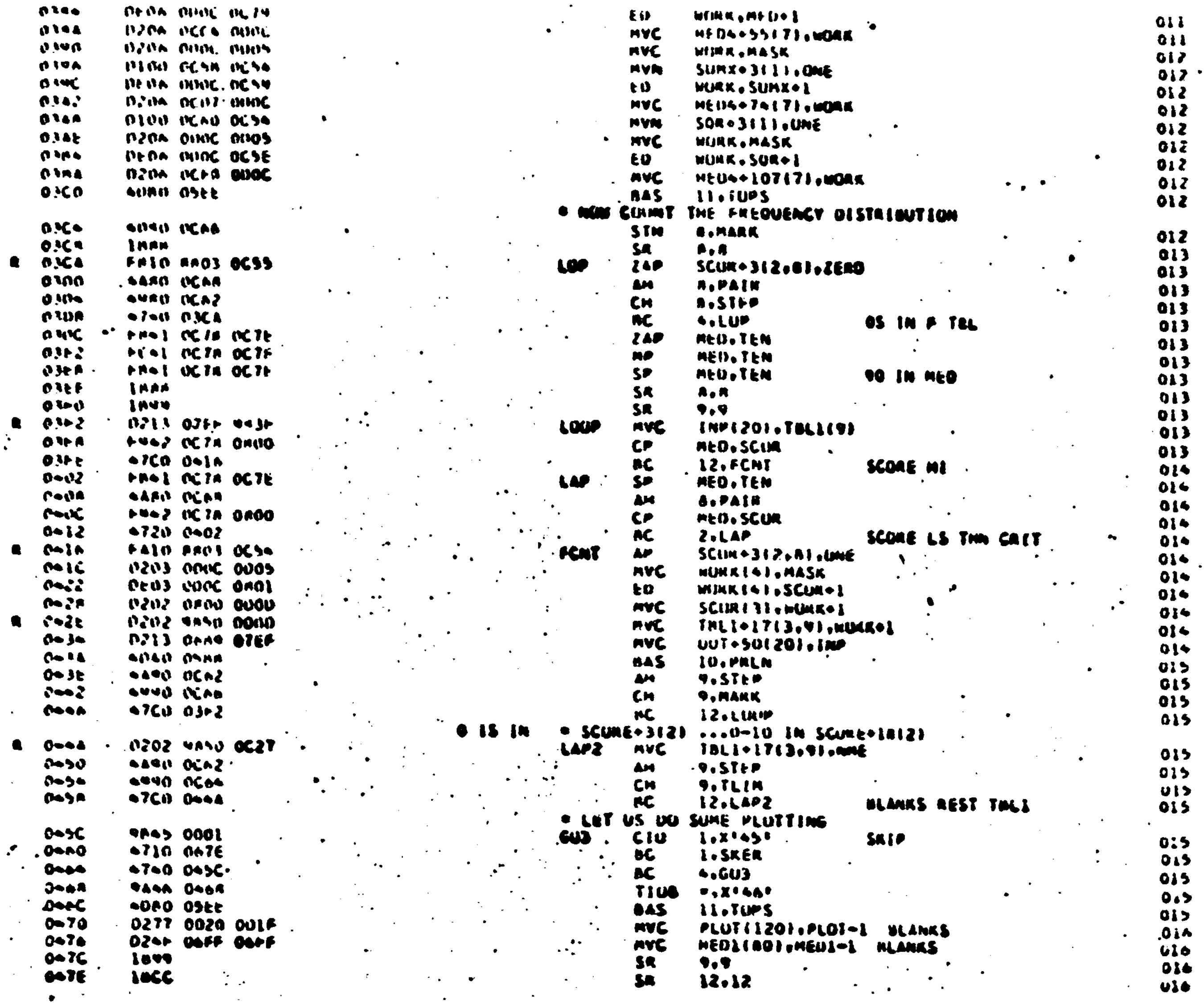
Figure SP-01

Statistics Program

```
0262    4740 066C                         BC    4,NEG          SQRT OF NEG NO      006
0266    4780 02FE                         BC    8,NULL         SQRT OF ZERO        006
026A    F841 0C5D 0C56                    ZAP   SQR,ENN                            006
0270    FC41 0C5D 0C56                    MP    SQR,ENN                            006
0276    FB41 0C5D 0C56                    SP    SQR,ENN        N(N-1)              007
027C    F810 0C8B 0C55                    ZAP   VARI+10(2),ZERO VARI NOW HAS 4     007
0282    D100 0C8A 0C80                    MVN   SMXX+9(1),ZN   ZEROS ON RIGHT      007
0288    FD84 0C81 0C5D                    DP    VARI,SQR                           007
028E    F844 0C5A 0C83                    ZAP   SUMX,VARI+2(5) VARIANCE IN SUMX    007
0294    F896 0C81 0C81                    ZAP   SMXX,VARI(7)   AND SMXX            007
029A    FA90 0C81 0C54                    AP    SMXX,ONE                           007
02A0    FD90 0C81 0C7D                    DP    SMXX,TWO                           007
                                 * NOW LETS TRY TO TAKE A SQUARE ROOT
02A6    1B88                              SR    8,8                                007
02A8    F898 0C81 0C81                    ZAP   SMXX,SMXX(9)                       007
02AE    F844 0C5D 0C86             SQRT   ZAP   SQR,SMXX+5(5)  APPROX IN SQR       008
02B4    F894 0C81 0C58                    ZAP   SMXX,SUMX      VARIANCE IN SMXX    008
02BA    FD94 0C81 0C5D                    DP    SMXX,SQR                           008
02C0    F894 0C81 0C81                    ZAP   SMXX,SMXX(5)                       008
02C6    F844 0C78 0C86                    ZAP   MED,SMXX+5(5)  TEST OUR ANSWER     008
02CC    FB44 0C78 0C5D                    SP    MED,SQR                            008
02D2    F941 0C78 0C7E                    CP    MED,TEN        ANSWER GOOD TO ONE  008
02D8    47A0 030A                         BC    10,NULL+12                         008
02DC    FA94 0C81 0C5D                    AP    SMXX,SQR                           008
02E2    FD90 0C81 0C7D                    DP    SMXX,TWO                           008
02E8    FA98 0C81 0C81                    ZAP   SMXX,SMXX(9)   STANDARD DEVIATION  009
02EE    4A80 0C6A                         AH    8,BIT                              009
02F2    4980 0C62                         CH    8,STEP                             009
02F6    4780 030A                         BC    8,NULL+12                          009
02FA    47F0 02AE                         BC    15,SQRT        IN SQR XXXXX.XXXXC  009
02FE    F840 0C5D 0C55             NULL   ZAP   SQR,ZERO                           009
0304    F840 0C58 0C55                    ZAP   SUMX,ZERO      VARIANCE WAS ZERO   009
030A    1B88                              SR    8,8                                009
030C    F861 000C 0C56                    ZAP   WORK,ENN                           009
0312    FD40 000C 0C7D                    DP    WORK,TWO                           009
0318    4A80 0C62                  LAP1   AH    8,STEP                             009
031C    FB50 000C 0C54                    SP    WORK(6),ONE                        010
0322    4720 0318                         BC    2,LAP1                             010
0326    F832 0C78 8050                    ZAP   MED(4),TBL1+17(3,8)                010
032C    4080 0C66                         STH   8,MARK                             010
0330    F900 0012 0C55                    CP    WORK+6(1),ZERO                     010
0336    4720 033E                         BC    2,LAP6                             010
033A    4B80 0C62                         SH    8,STEP                             010
033E    FA32 0C78 8050             LAP6   AP    MED(4),TBL1+17(3,8)                010
0344    4A80 0C66                         AH    8,MARK                             010
034A    D100 0C7B 0C80                    MVN   MED+3(1),ZN    MEDIAN IS IN FIRST  010
034E    F800 0C7C 0C55                    ZAP   MED+4(1),ZERO  4 BYTES OF MED      010
0354    FD40 0C78 0C7D                    DP    MED,TWO        XXXXX.XXC           010
035A    F341 000C 0C56                    UNPK  WORK,ENN                           011
0360    D300 0012 0C80                    MVZ   WORK+6(1),ZN                       011
0366    D201 0C98 0011                    MVC   MED4+14(2),WORK+5                  011
036C    D206 000C 0005                    MVC   WORK,MASK                          011
0372    DE06 000C 0C73                    ED    WORK,MEAN+1                        011
0378    D206 0CAC 000C                    MVC   MED4+31(7),WORK                    011
037E    D206 000C 0005                    MVC   WORK,MASK
```

Figure SP-02

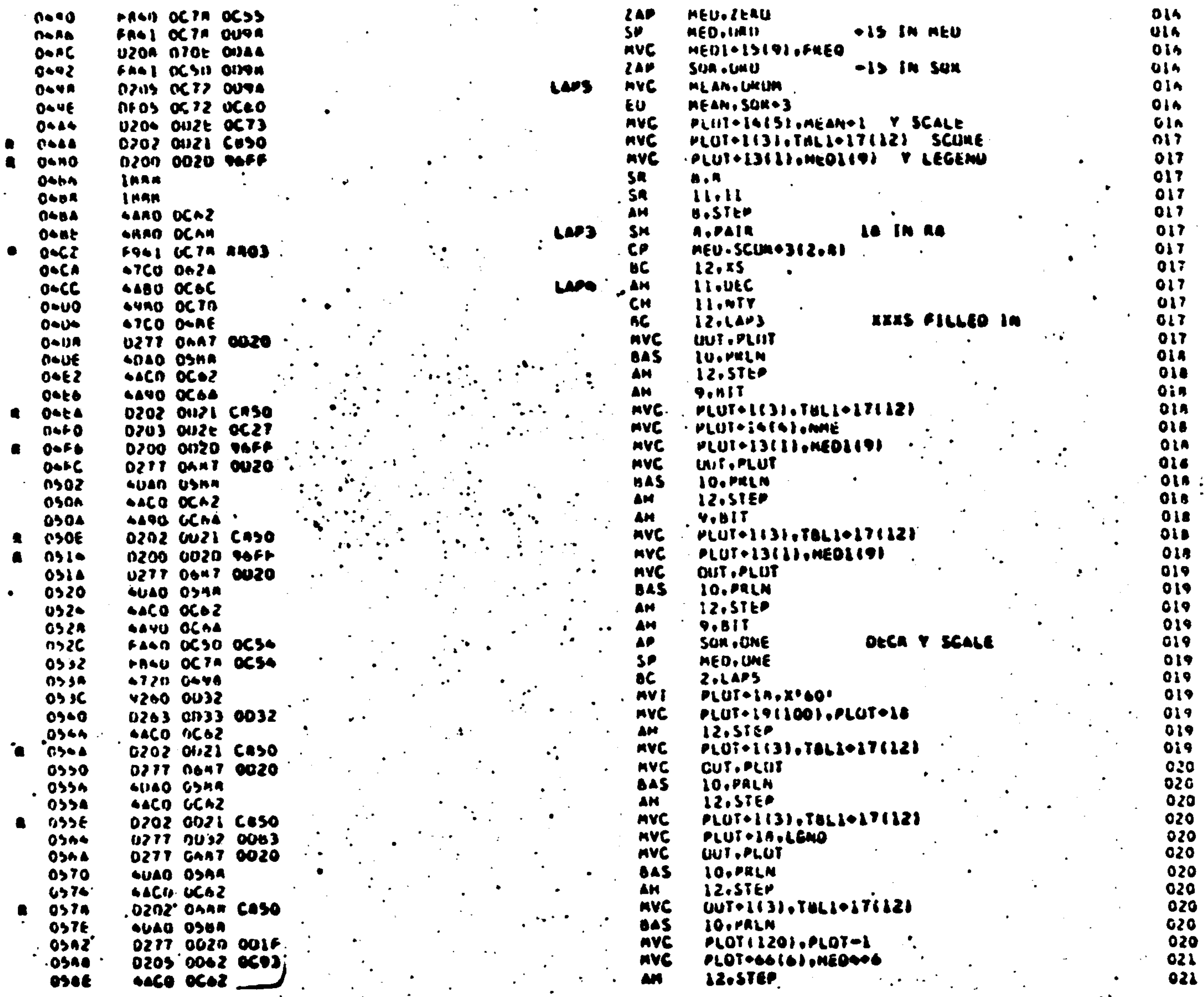
Statistics Program



```
  0384    DE0A 000C 0C74                ED    WORK,MED+1                                    011
  038A    D206 0CE6 000C                MVC   MED4+55(7),WORK                               011
  0390    D20A 000C 0005                MVC   WORK,MASK                                     012
  0396    D100 0C5A 0C56                MVN   SUMX+3(1),ONE                                 012
  039C    DE0A 000C 0C59                ED    WORK,SUMX+1                                   012
  03A2    D206 0CD7 000C                MVC   MED4+74(7),WORK                               012
  03A8    D100 0CA0 0C56                MVN   SQR+3(1),ONE                                  012
  03AE    D20A 000C 0005                MVC   WORK,MASK                                     012
  03B4    DE0A 000C 0C5E                ED    WORK,SQR+1                                    012
  03BA    D206 0CF8 000C                MVC   MED4+107(7),WORK                              012
  03C0    4DB0 05EE                     BAS   11,TOPS                                       012
                                * NOW COUNT THE FREQUENCY DISTRIBUTION
  03C4    4080 0CA6                     STH   8,MARK                                        012
  03C8    1B88                          SR    8,8                                           013
R 03CA    F810 8A03 0C55         LOP    ZAP   SCUR+3(2,8),ZERO                              013
  03D0    4A80 0CA8                     AH    8,PAIR                                        013
  03D4    4980 0CA2                     CH    8,STEP                                        013
  03D8    4740 03CA                     BC    4,LUP              0S IN F TBL                013
  03DC    F841 0C7A 0C7E                ZAP   MED,TEN                                       013
  03E2    FC41 0C7A 0C7F                MP    MED,TEN                                       013
  03E8    FB41 0C7A 0C7E                SP    MED,TEN            90 IN MED                  013
  03EE    1B88                          SR    8,8                                           013
  03F0    1B99                          SR    9,9                                           013
R 03F2    D213 07FF 943E         LOUP   MVC   INP(20),TBL1(9)                               013
  03F8    F942 0C7A 8A00                CP    MED,SCUR                                      013
  03FE    47C0 041A                     BC    12,FCNT            SCORE HI                   014
  0402    FB41 0C7A 0C7E         LAP    SP    MED,TEN                                       014
  0408    4A80 0CA8                     AH    8,PAIR                                        014
  040C    F942 0C7A 8A00                CP    MED,SCUR                                      014
  0412    4720 0402                     BC    2,LAP              SCORE LS THN CRIT          014
R 0416    FA10 8A03 0C56         FCNT   AP    SCUR+3(2,8),ONE                               014
  041C    D203 000C 0005                MVC   WORK(4),MASK                                  014
  0422    DE03 000C 8A01                ED    WORK(4),SCUR+1                                014
  0428    D202 8A00 000D                MVC   SCUR(3),WORK+1                                014
R 042E    D202 9A50 000D                MVC   TBL1+17(3,9),WORK+1                           014
  0434    D213 0A4A 07EF                MVC   OUT+50(20),INP                                014
  043A    4DA0 09AA                     BAS   10,PRLN                                       015
  043E    4A90 0CA2                     AH    9,STEP                                        015
  0442    4990 0CA6                     CH    9,MARK                                        015
  0446    47C0 03F2                     BC    12,LOUP                                       015
                     0 IS IN    * SCORE+3(2) ...0-10 IN SCORE+18(2)
R 044A    D202 9A50 0C27         LAP2   MVC   TBL1+17(3,9),ONE                              015
  0450    4A90 0CA2                     AH    9,STEP                                        015
  0454    4990 0CA4                     CH    9,TLIM                                        015
  0458    47C0 044A                     BC    12,LAP2            BLANKS REST TBL1           015
                                * LET US DO SOME PLOTTING
  045C    9A45 0001              GO3    CIO   1,X'45'            SKIP                       015
  0460    4710 047E                     BC    1,SKER                                        015
  0464    4740 045C                     BC    4,GO3                                         015
  0468    9A4A 0468                     TIOB  *,X'4A'                                       015
  046C    4DB0 05EE                     BAS   11,TOPS                                       015
  0470    0277 0020 001F                MVC   PLOT(120),PLOT-1  BLANKS                      016
  0476    D24F 00FF 00FE                MVC   MED1(80),MED1-1  BLANKS                       016
  047C    1B99                          SR    9,9                                           016
  047E    1BCC                          SR    12,12                                         016
```

Figure SP-03

Statistics Program



```
     0480    F840 0C78 0C55                  ZAP   MED,ZERO                                 016
     0486    FB41 0C78 0D98                  SP    MED,UNO            +15 IN MED            016
     048C    D208 070E 0DA4                  MVC   MED1+15(9),FREQ                          016
     0492    F841 0C50 0D98                  ZAP   SQR,UNO            -15 IN SQR            016
     0498    D205 0C72 0D9A            LAPS  MVC   MEAN,UNUM                                016
     049E    DE05 0C72 0C60                  ED    MEAN,SQR+3                               016
     04A4    D204 0D2E 0C73                  MVC   PLOT+14(5),MEAN+1   Y SCALE              016
8    04AA    D202 0D21 C850                  MVC   PLOT+1(3),TBL1+17(12)   SCORE            017
8    04B0    D200 0D2D 96FF                  MVC   PLOT+13(1),MED1(9)   Y LEGEND            017
     04B6    1B88                            SR    8,8                                      017
     04B8    1BBB                            SR    11,11                                    017
     04BA    4A80 0C62                       AH    8,STEP                                   017
     04BE    4B80 0C6A                 LAP3  SH    8,PAIR             18 IN R8              017
8    04C2    F961 0C78 8803                  CP    MED,SCUM+3(2,8)                          017
     04C8    47C0 062A                       BC    12,XS                                    017
     04CC    4AB0 0C6C                 LAP4  AH    11,DEC                                   017
     04D0    49B0 0C70                       CH    11,NTY                                   017
     04D4    47C0 04BE                       BC    12,LAP3            XXXS FILLED IN        017
     04D8    D277 0A87 0D20                  MVC   OUT,PLOT                                 017
     04DE    4DA0 05B8                       BAS   10,PRLN                                  018
     04E2    4AC0 0C62                       AH    12,STEP                                  018
     04E6    4A90 0C64                       AH    9,BIT                                    018
8    04EA    D202 0D21 C850                  MVC   PLOT+1(3),TBL1+17(12)                    018
     04F0    D203 0D2E 0C27                  MVC   PLOT+14(4),NME                           018
8    04F6    D200 0D2D 96FF                  MVC   PLOT+13(1),MED1(9)                       018
     04FC    D277 0A87 0D20                  MVC   OUT,PLOT                                 018
     0502    4DA0 05B8                       BAS   10,PRLN                                  018
     0506    4AC0 0C62                       AH    12,STEP                                  018
     050A    4A90 0C64                       AH    9,BIT                                    018
8    050E    D202 0D21 C850                  MVC   PLOT+1(3),TBL1+17(12)                    018
8    0514    D200 0D2D 96FF                  MVC   PLOT+13(1),MED1(9)                       018
     051A    D277 0A87 0D20                  MVC   OUT,PLOT                                 019
     0520    4DA0 05B8                       BAS   10,PRLN                                  019
     0524    4AC0 0C62                       AH    12,STEP                                  019
     0528    4A90 0C64                       AH    9,BIT                                    019
     052C    FA40 0C50 0C54                  AP    SQR,ONE            DECR Y SCALE          019
     0532    FB40 0C78 0C54                  SP    MED,ONE                                  019
     0538    4720 0498                       BC    2,LAPS                                   019
     053C    9260 0D32                       MVI   PLOT+18,X'60'                            019
     0540    D263 0D33 0D32                  MVC   PLOT+19(100),PLOT+18                     019
     0546    4AC0 0C62                       AH    12,STEP                                  019
8    054A    D202 0D21 C850                  MVC   PLOT+1(3),TBL1+17(12)                    019
     0550    D277 0A87 0D20                  MVC   OUT,PLOT                                 020
     0556    4DA0 05B8                       BAS   10,PRLN                                  020
     055A    4AC0 0C62                       AH    12,STEP                                  020
8    055E    D202 0D21 C850                  MVC   PLOT+1(3),TBL1+17(12)                    020
     0564    D277 0D32 0083                  MVC   PLOT+18,LGND                             020
     056A    D277 0A87 0D20                  MVC   OUT,PLOT                                 020
     0570    4DA0 05B8                       BAS   10,PRLN                                  020
     0574    4AC0 0C62                       AH    12,STEP                                  020
8    0578    D202 0A88 C850                  MVC   OUT+1(3),TBL1+17(12)                     020
     057E    4DA0 05B8                       BAS   10,PRLN                                  020
     0582    D277 0D20 001F                  MVC   PLOT(120),PLOT-1                         020
     0588    D205 0D62 0C93                  MVC   PLOT+66(6),MEDO+6                        021
     058E    4AC0 0C62                       AH    12,STEP                                  021
```

Figure SP-04

Statistics Program

```
A  0592     D202 0021 CA50                MVC   PLOT+1(3),TBL1+17(12)                              021
   0598     D277 0687 0020                MVC   OUT,PLOT                                           021
   059E     4DA0 05B8                     BAS   10,PRLN                                            071
   05A2     4AC0 0CA2                     AH    12,STEP                                            071
A  05A6     D202 0688 C850                MVC   OUT+1(3),TBL1+17(12)                               071
   05AC     4DA0 05B8                     BAS   10,PRLN                                            071
   05B0     9900 01FF                     MPR   X'FFF',0                                           021
   05B4     47F0 015E                     BC    15,EX4                                             021
                                   * PRINT AND READ SUBROUTINES FOLLOW
   05B8     D040 0687 0078         PRLN   XIO   OUT(X'40'),120                                     021
   05BE     4710 0634                     BC    1,ERRP                                             022
   05C2     4740 05B8                     BC    4,PRLN                                             022
   05C6     9A40 05C6                     TIOB  *,X'40'                                            027
   05CA     9A41 0634                     TIOB  ERRP,X'41'                                         022
   05CE     D277 0687 0686                MVC   OUT(120),OUT-1                                     022
   05D4     07FA                          BCR   15,10                                              022
   05D6     D022 07EF 0050         RDCD   XIO   INP(X'22'),80                                      022
   05DC     4710 063C                     BC    1,ERR2                                             022
   05E0     4740 05D6                     BC    4,RDCD                                             022
   05E4     9A20 05E4                     TIOB  *,X'20'                                            022
   05E8     9A21 063C                     TIOB  ERR2,X'21'                                         022
   05EC     07FA                          BCR   15,10                                              022
   05EE     D24F 069B 06FF         TOPS   MVC   OUT+20(80),HED1                                    022
   05F4     4DA0 05B8                     BAS   10,PRLN                                            022
   05F8     D24F 069B 074F                MVC   OUT+20(80),HED2                                    023
   05FE     4DA0 05B8                     BAS   10,PRLN                                            023
   0602     D24F 069B 079F                MVC   OUT+20(80),HED3                                    023
   0608     4DA0 05B8                     BAS   10,PRLN                                            023
   060C     4DA0 05B8                     BAS   10,PRLN                                            023
   0610     D277 0687 0C8D                MVC   OUT,HED4                                           023
   0616     4DA0 05B8                     BAS   10,PRLN          HEADINGS PRINTED                  023
   061A     4DA0 05B8                     BAS   10,PRLN                                            023
   061E     D205 0687 0C93                MVC   OUT(6),HED4+6                                      023
   0624     4DA0 05B8                     BAS   10,PRLN                                            023
   0628     07FB                          BCR   15,11                                              023
A  062A     D209 8033 00A0         XS     MVC   PLOT+19(10,11),FILL                                023
   0630     47F0 04CC                     BC    15,LAP4                                            024
                                   * ERROR SUBROUTINES FOLLOW
   0634     9900 0100              ERRP   MPR   X'100',0                                           024
   0638     47F0 05B8                     BC    15,PRLN                                            024
   063C     9900 0002              ERR2   MPR   X'002',0                                           024
   0640     47F0 05D6                     BC    15,RDCD                                            024
   0644     D20B 0805 0C38         INEA   MVC   INP+22(12),MSG1                                    024
   064A     D24F 0687 07EF                MVC   OUT(80),INP                                        024
   0650     4DA0 05B8                     BAS   10,PRLN                                            024
   0654     47F0 01C0                     BC    15,GO1                                             024
   0658     D20C 0813 0C47         INA2   MVC   INP+36(13),MSG2                                    024
   065E     D24F 0687 07EF                MVC   OUT(80),INP                                        024
   0664     4DA0 05B8                     BAS   10,PRLN                                            024
   0668     47F0 01CA                     BC    15,GO2                                             025
   066C     D20B 0687 0D13         MEG    MVC   OUT(12),MSG3                                       025
   0672     4DA0 05B8                     BAS   10,PRLN                                            025
   0676     9900 0003                     MPR   X'003',0                                           025
   067A     47F0 0304                     BC    15,NULL+12                                         025
   067E     9900 0004              SKEA   MPR   X'004',0                                           025
```

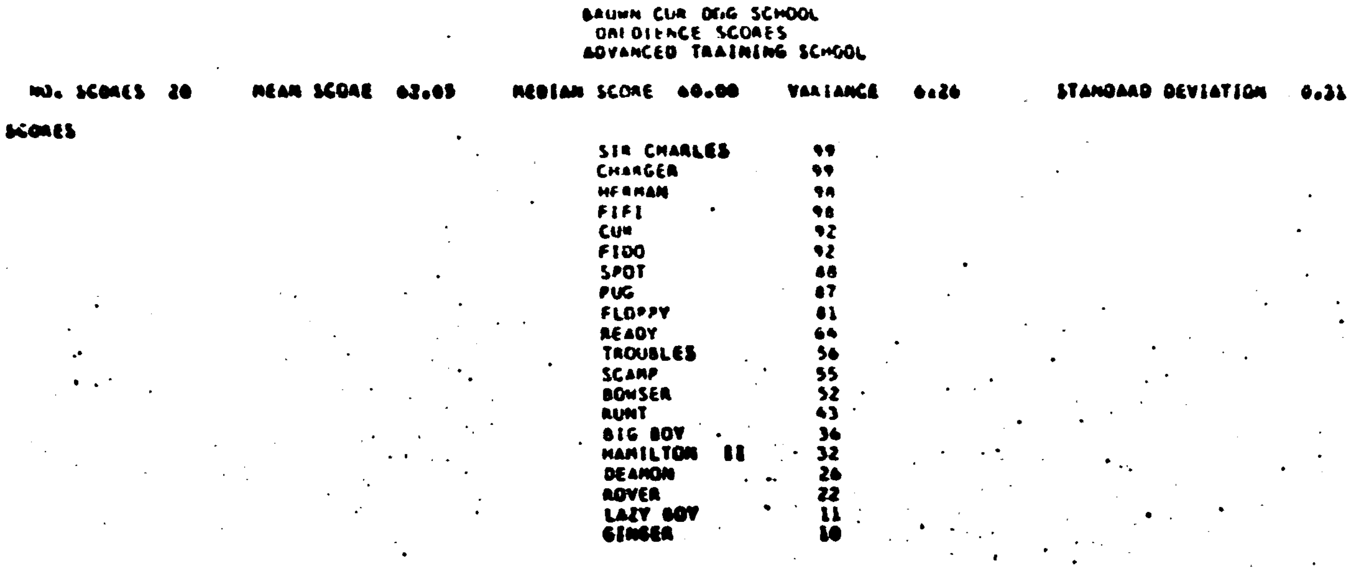
Figure SP-05

Statistics Program

```
0A82     47F0 045C
                                                      BC    15,G03
0A86     40                                   * CONSTANT AND STORAGE FOLLOW                025
0A87                                          BLNK   DC    C' '
0AFF                                          OUT    DS    CL120                           025
0B4F                                          MED1   DS    CL80                            025
0B9F                                          MED2   DS    CL80                            026
0BEF                                          MED3   DS    CL80                            026
0BEF                                          INP    DS    0CL80                           026
0C00                                          NAME   DS    CL17                            026
0C03                                          SCOR   DS    CL3                             026
0C3F                                                 DS    CL60                            026
0C27     4040 4040 4040 4040 4040 4040 4040 4040  TBL1   DS    50CL20                       026
0C37     4040 4040                            NME    DC    C'                              026
0C3D     D5C1 D4C5 40D4 C9E2 E2C9 D5C7               DC    C'                              026
0C47     E2C3 D6D9 C540 D4C9 E2E2 C9D5 C7     MSG1   DC    C'NAME MISSING'                 026
0C54     1C                                   MSG2   DC    C'SCORE MISSING'                026
0C55     0C                                   ONE    DC    X'1C'                           026
0C56                                          ZERO   DC    X'0C'                           026
0C58                                          ENM    DS    CL2                             026
0C5D                                          SUMX   DS    CL5                             026
0C62     0014                                 SQR    DS    CL5                             027
0C64     03D4                                 STEP   DC    H'20'                           027
0C66                                          TLIM   DC    H'980'                          027
0C68     0002                                 MARK   DS    H                               027
0C6A     0001                                 PAIR   DC    H'2'                            027
0C6C     000A                                 BIT    DC    H'1'                            028
0C6E     0064                                 DEC    DC    H'10'                           028
0C70     005A                                 CEN    DC    H'100'                          028
0C72                                          NTY    DC    H'90'                           028
0C78                                          MEAN   DS    CL6                             028
0C7D     2C                                   MED    DS    CL5                             028
0C7E     010C                                 TWO    DC    X'2C'                           029
0C80     F0                                   TEN    DC    X'010C'                         029
0C81                                          ZN     DC    X'F0'                           029
0C81                                          VARI   DS    0CL12                           029
0C8B                                          SMXX   DS    CL10                            029
0C8D                                                 DS    CL2                             030
0C8D     4040 D5D6 4B40 E2C3 D6D9 C5E2 40     MEDA   DS    0CL120                          030
0C9A                                                 DC    C'  NO. SCORES '                030
0C9C     4040 4040 4040 D4C5 C1D5 40E2 C3D6 D9C5     DS    CL2                             030
0CAC                                                 DC    C'       MEAN SCORE'            030
0CB3     4040 4040 40D4 C5C4 C9C1 D540 E2C3 D6       DS    CL7                             031
0CC2     D9C5                                        DC    C'     MEDIAN SCO'             031
0CC4                                                 DC    C'RE'                           032
0CCB     4040 4040 E5C1 D9C9 C1D5 C3C5               DS    CL7                             032
0CD7                                                 DC    C'    VARIANCE'                 032
0CDE     4040 4040 4040 4040 E2E3 C1D5 C4C1 D9C4     DS    CL7                             033
0CEE     40C4 C5E5 C9C1 E3C9 D6D5                    DC    C'        STANDARD'             033
0CF8                                                 DC    C' DEVIATION'                   034
0CFF     4040 4040 4040                              DS    CL7                             034
0005     4020 2120 4B20 20                           DC    C'                              034
000C                                          MASK   DC    X'402021204B2020'               035
0013     E2D6 D9E3 40D5 C5C7 40D5 D64B        WORK   DS    CL7                             035
001F     40                                   MSG3   DC    C'SORT NEG NO.'                 035
                                                     DC    C' '                            036
                                                                                           036




0D20
0D98     015D                                 PLOT   DS    CL120
0D9A     4020 2020 60C9                       ORG    DC    X'015D'                         036
0DA0     E7E7 E7E7 E7E7 E7E7 E7E7             DRDM   DC    X'4020202060C9'                 037
0DAA     C6D9 C5D8 E4C5 D5C3 E8               FILL   DC    C'XXXXXXXXXX'                   037
0DB3                                          FREQ   DC    C'FREQUENCY'                    037
0DB3     F040 4040 4040 4040 4040 F1F0 4040 4040  LGND   DS    0CL120                       037
0DC3     4040 4040 F2F0 4040 4040 4040 4040 F3F0         DC    C'0         10    '         037
0DD3     4040 4040 4040 4040 F4F0 4040 4040 4040         DC    C'    20         30'         038
0DE3     4040 F5F0 4040 4040 4040 4040 F6F0 4040         DC    C'        40       '         038
0DF3     4040 4040 4040 F7F0 4040 4040 4040 4040         DC    C'  50         60  '         038
0E03     F8F0 4040 4040 4040 4040 F9F0 4040 4040         DC    C'      70         '         038
0E13     4040 40F1 F0F0                                  DC    C'80         90    '         039
0E1E                                                     DC    C'   100'                    039
                                                         END   EX4                         039
                                                                                           040
```

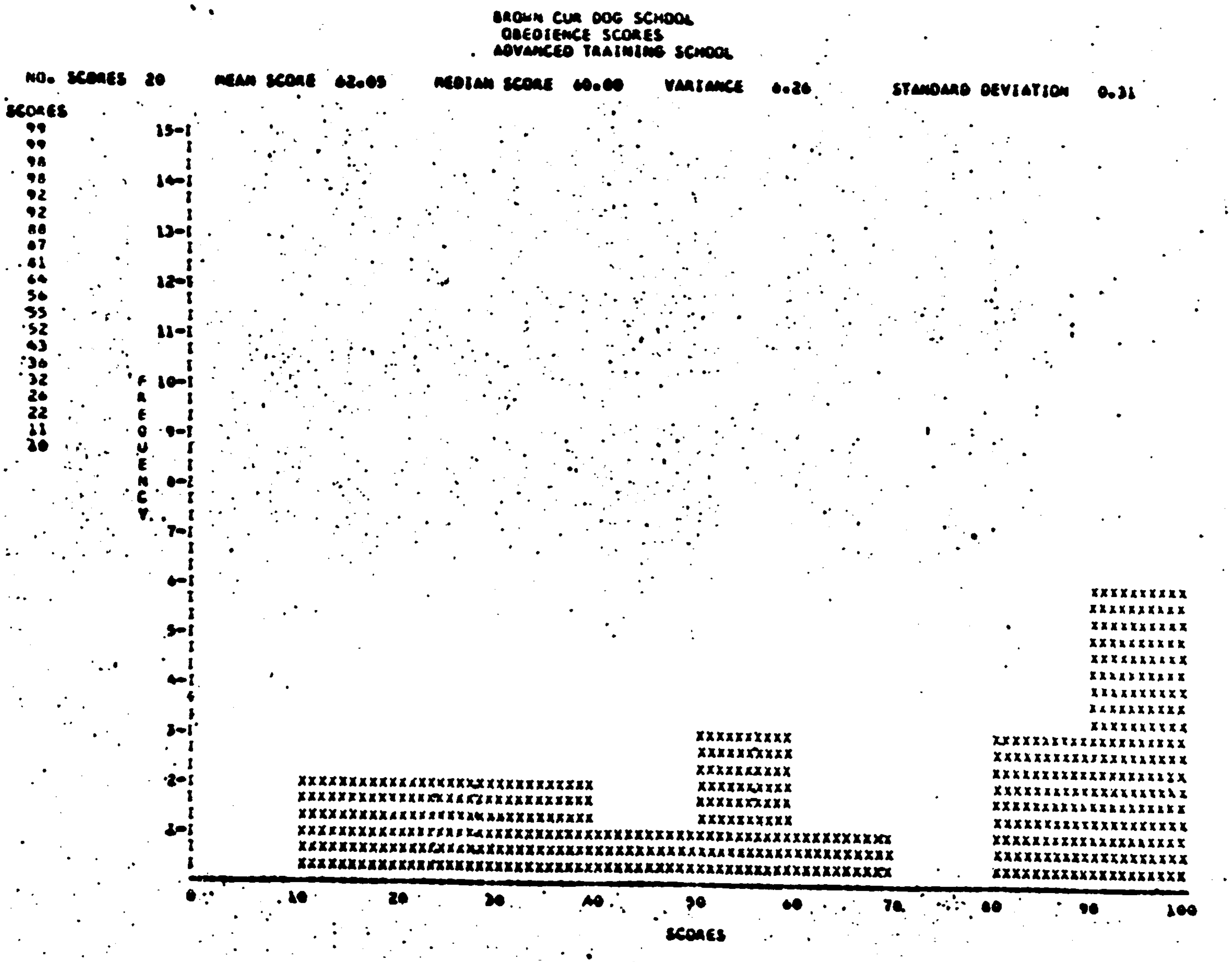
Figure SP-06

Statistics Program Print Out



```
                                        BROWN CUR DOG SCHOOL
                                         OBEDIENCE SCORES
                                        ADVANCED TRAINING SCHOOL

  NO. SCORES  20      MEAN SCORE  62.05      MEDIAN SCORE  60.00      VARIANCE    6.26         STANDARD DEVIATION   0.31

SCORES
                                         SIR CHARLES       99
                                         CHARGER           99
                                         HERMAN            98
                                         FIFI              98
                                         CUR               92
                                         FIDO              92
                                         SPOT              88
                                         PUG               87
                                         FLOPPY            81
                                         READY             64
                                         TROUBLES          56
                                         SCAMP             55
                                         BOWSER            52
                                         RUNT              43
                                         BIG BOY           36
                                         HAMILTON  II      32
                                         DEAMON            26
                                         ROVER             22
                                         LAZY BOY          11
                                         GINGER            10
```

Statistics Program Print Out Number 2



```
                                      BROWN CUR DOG SCHOOL
                                       OBEDIENCE SCORES
                                      ADVANCED TRAINING SCHOOL

NO. SCORES  20      MEAN SCORE  62.05      MEDIAN SCORE  60.00      VARIANCE    6.26         STANDARD DEVIATION   0.31
SCORES
  99
  99
  98
  98
  92
  92
  88
  87
  81
  64
  56
  55
  52
  43
  36
  32
  26
  22
  11
  10
```

Figure SP-07

13. SELECTED FORTRAN PROGRAMS IN THE FORM OF ANSWERS TO QUESTIONS

Write the following Fortran programs:

13.1 Compound Interest Rate

13.2 Plotting

13.3 Integration by Simpson's Rule

13.4 Quadratic Equation

13.5 Linear Equation

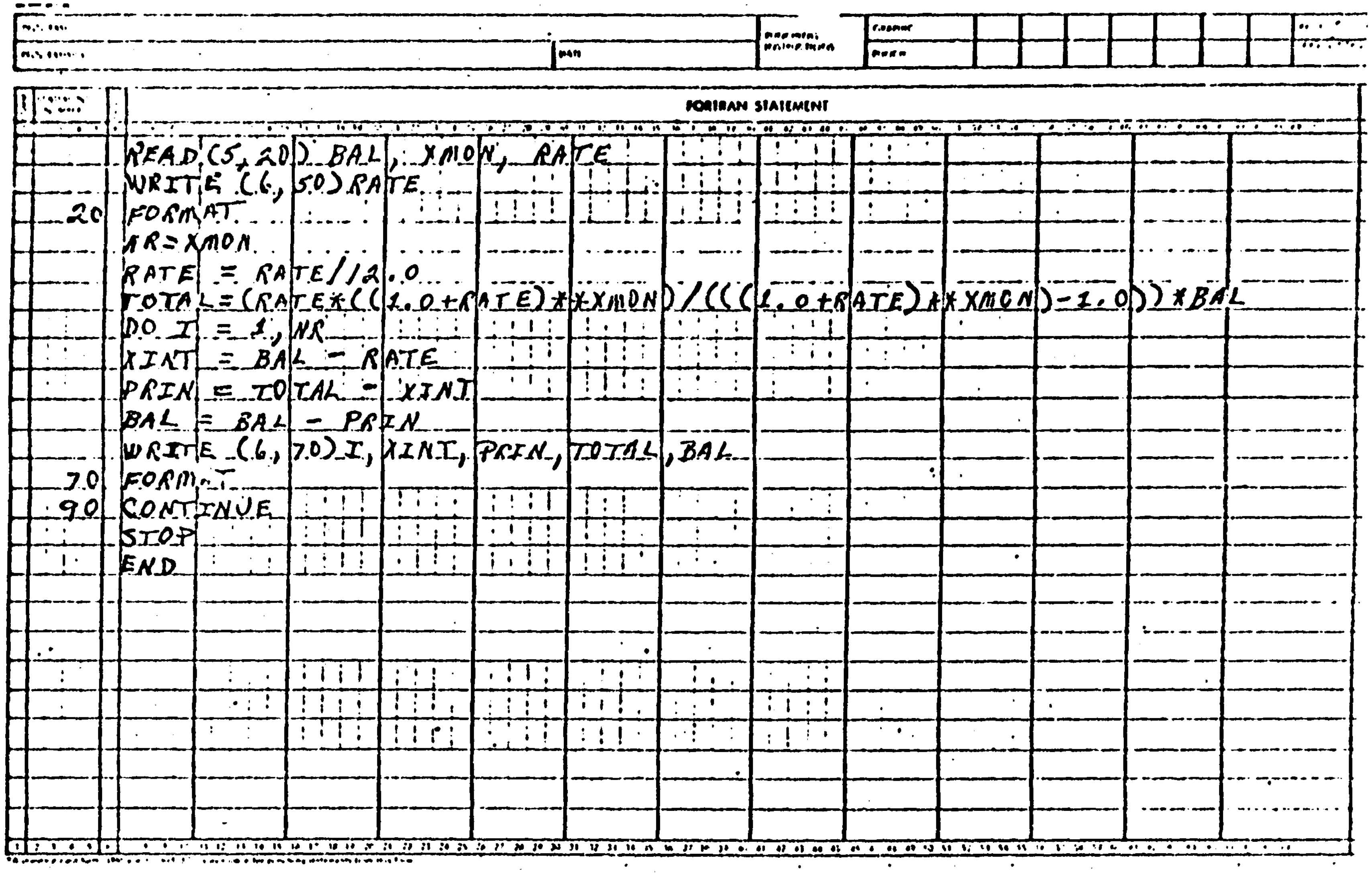
13.6 Computed Go To

13.7 The Derivative

13.8 Evaluate a Polynomial

13.9 Functions

13.1 COMPOUND INTEREST RATE

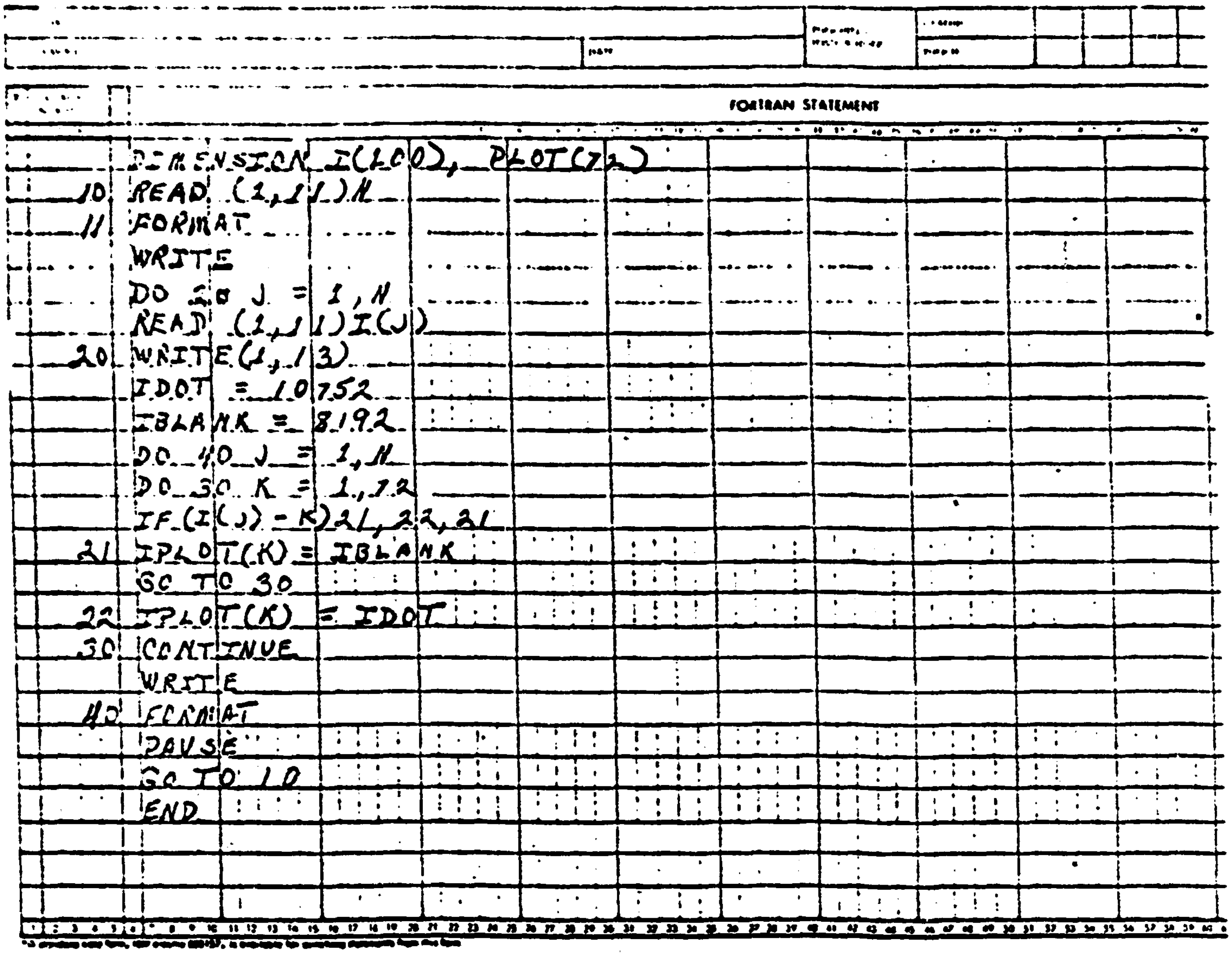


FORTRAN STATEMENT

```
      READ(5,20) BAL, XMON, RATE
      WRITE (6,50)RATE
   20 FORMAT
      NR=XMON
      RATE = RATE/12.0
      TOTAL=(RATE*((1.0+RATE)**XMON)/(((1.0+RATE)**XMON)-1.0))*BAL
      DO I = 1, NR
      XINT = BAL - RATE
      PRIN = TOTAL - XINT
      BAL = BAL - PRIN
      WRITE (6,70)I, XINT, PRIN, TOTAL, BAL
   70 FORMAT
   90 CONTINUE
      STOP
      END
```

Figure FO-13.1

13.2 PLOTTING

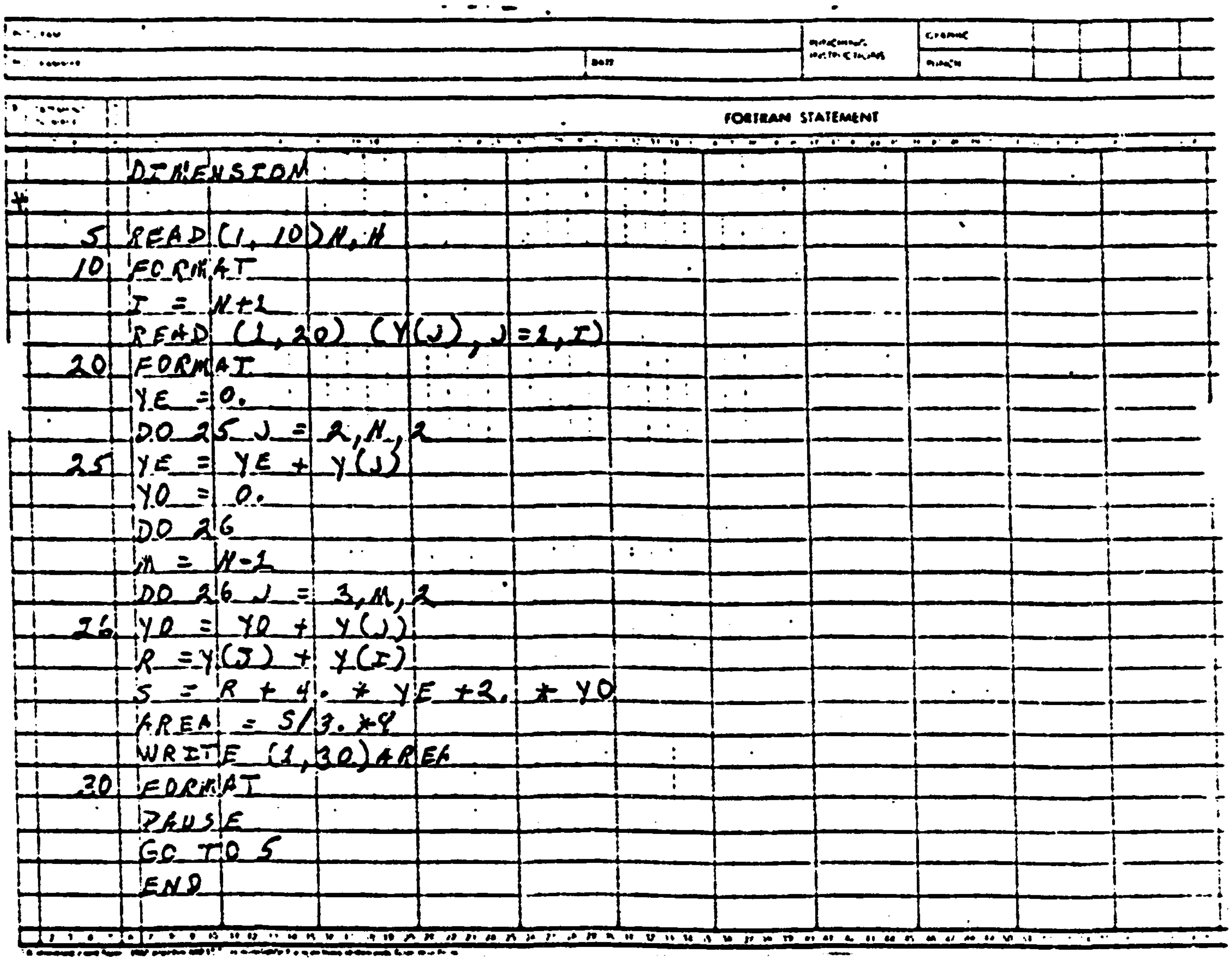


FORTRAN STATEMENT

```
      DIMENSION I(100), PLOT(72)
   10 READ (1,11)N
   11 FORMAT
      WRITE
      DO 20 J = 1,N
      READ (1,11)I(J)
   20 WRITE(1,13)
      IDOT = 10752
      IBLANK = 8192
      DO 40 J = 1,N
      DO 30 K = 1,72
      IF (I(J) - K)21,22,21
   21 IPLOT(K) = IBLANK
      GO TO 30
   22 IPLOT(K) = IDOT
   30 CONTINUE
      WRITE
   40 FORMAT
      PAUSE
      GO TO 10
      END
```

Figure FO-13.2

13.3 INTEGRATION BY SIMPSON'S RULE

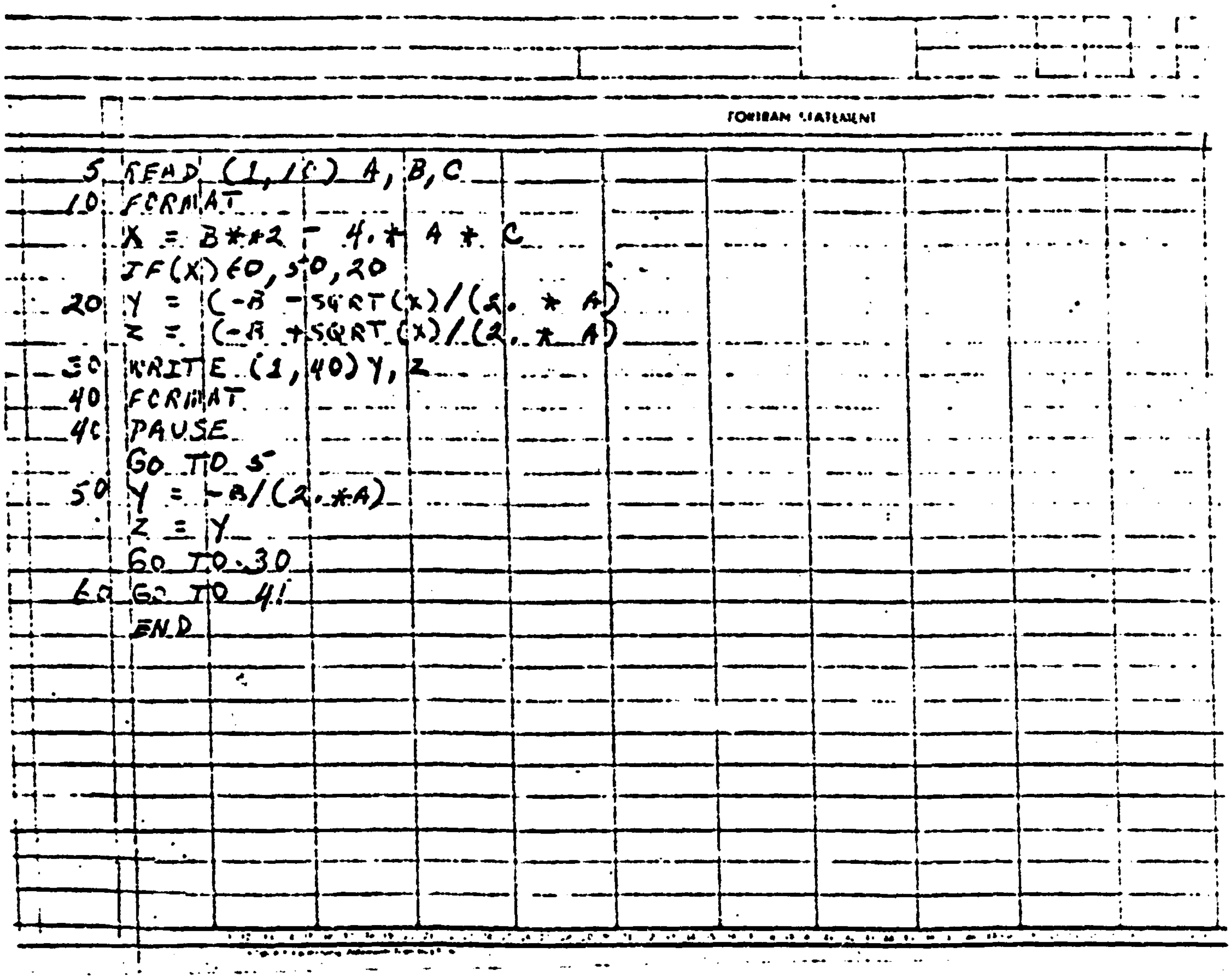


FORTRAN STATEMENT

```
      DIMENSION

    5 READ (1, 10)N,H
   10 FORMAT
      I = N+1
      READ (1,20) (Y(J), J=1,I)
   20 FORMAT
      YE = 0.
      DO 25 J = 2,N,2
   25 YE = YE + Y(J)
      YO = 0.
      DO 26
      M = N-1
      DO 26 J = 3,M,2
   26 YO = YO + Y(J)
      R = Y(1) + Y(I)
      S = R + 4. * YE +2. * YO
      AREA = S/3.*H
      WRITE (1,30)AREA
   30 FORMAT
      PAUSE
      GO TO 5
      END
```

Figure FO-13.3

13.4 QUADRATIC EQUATION

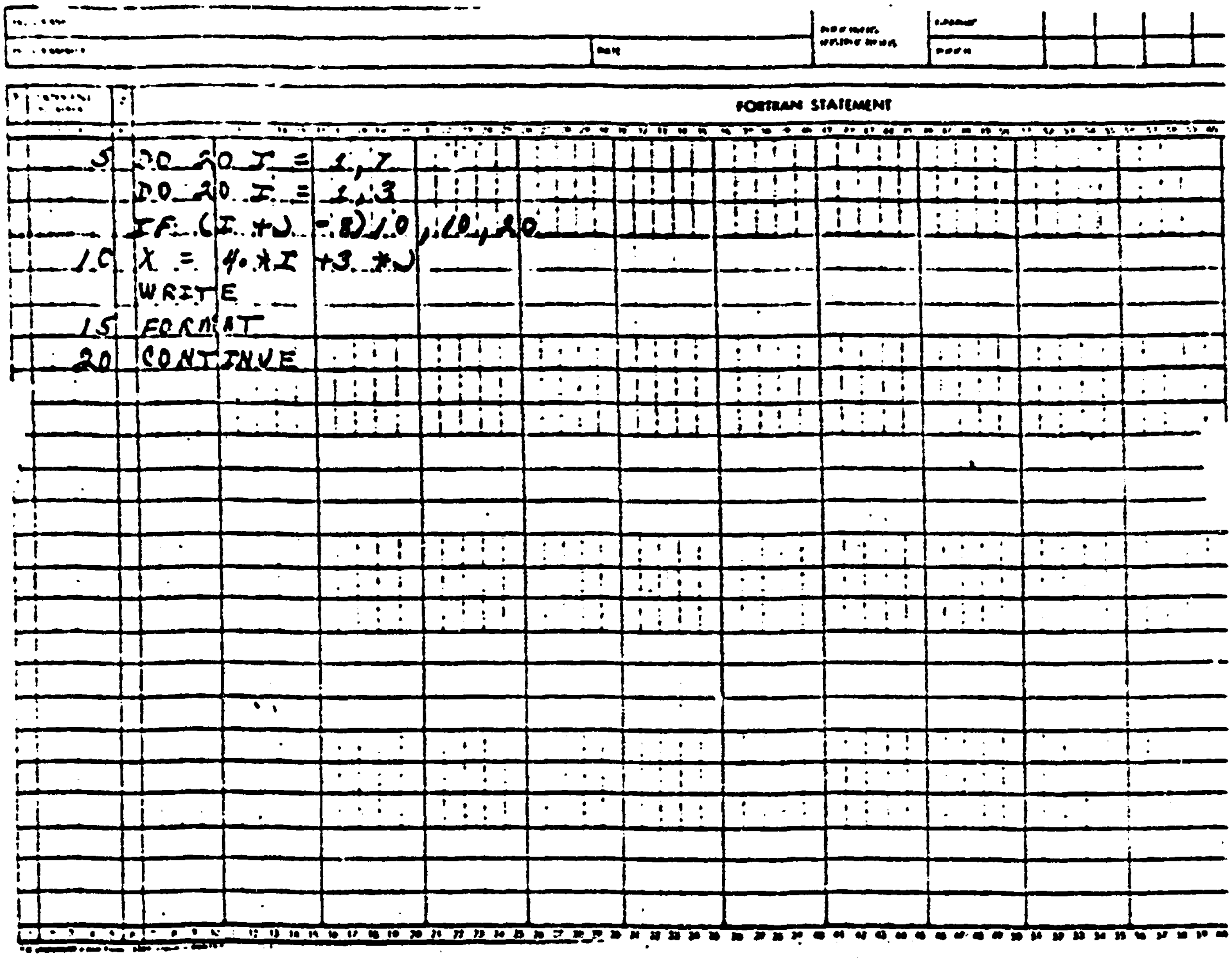


FORTRAN STATEMENT

```
    5 READ (1,10) A, B, C
   10 FORMAT
      X = B**2 - 4.* A * C
      IF(X) 60, 50, 20
   20 Y = (-B - SQRT(x)/(2. * A)
      Z = (-B + SQRT (x)/(2. * A)
   30 WRITE (1, 40) Y, Z
   40 FORMAT
   41 PAUSE
      GO TO 5
   50 Y = -B/(2.*A)
      Z = Y
      GO TO 30
   60 GO TO 41
      END
```

Figure FO-13.4

13.5 LINEAR EQUATION

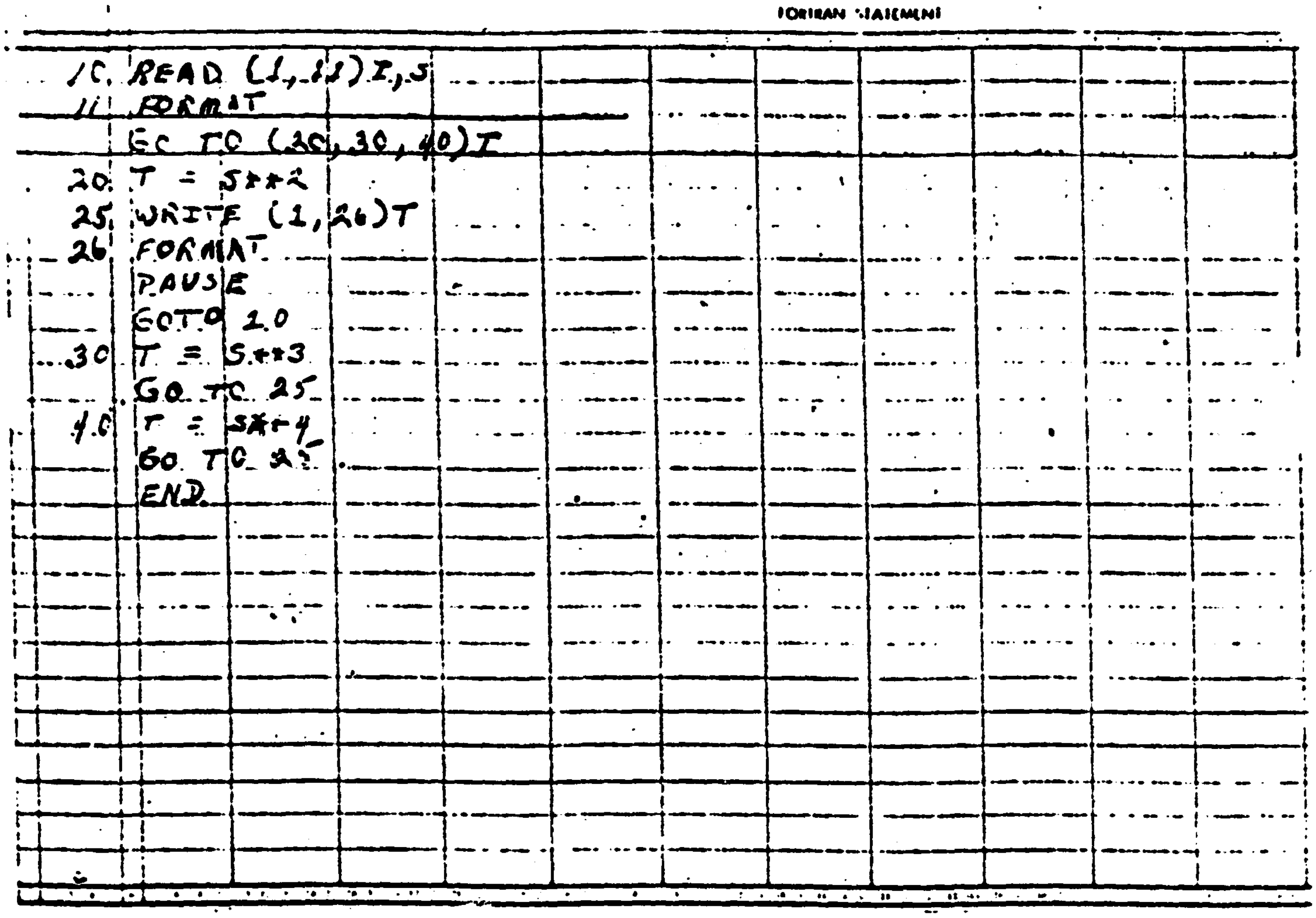


FORTRAN STATEMENT

```
    5 DO 20 J = 1,7
      DO 20 I = 1,3
      IF (I+J-8)10,10,20
   10 X = 4.*I+3.*J
      WRITE
   15 FORMAT
   20 CONTINUE
```

Figure FO-13.5

13.6 COMPUTED GO TO

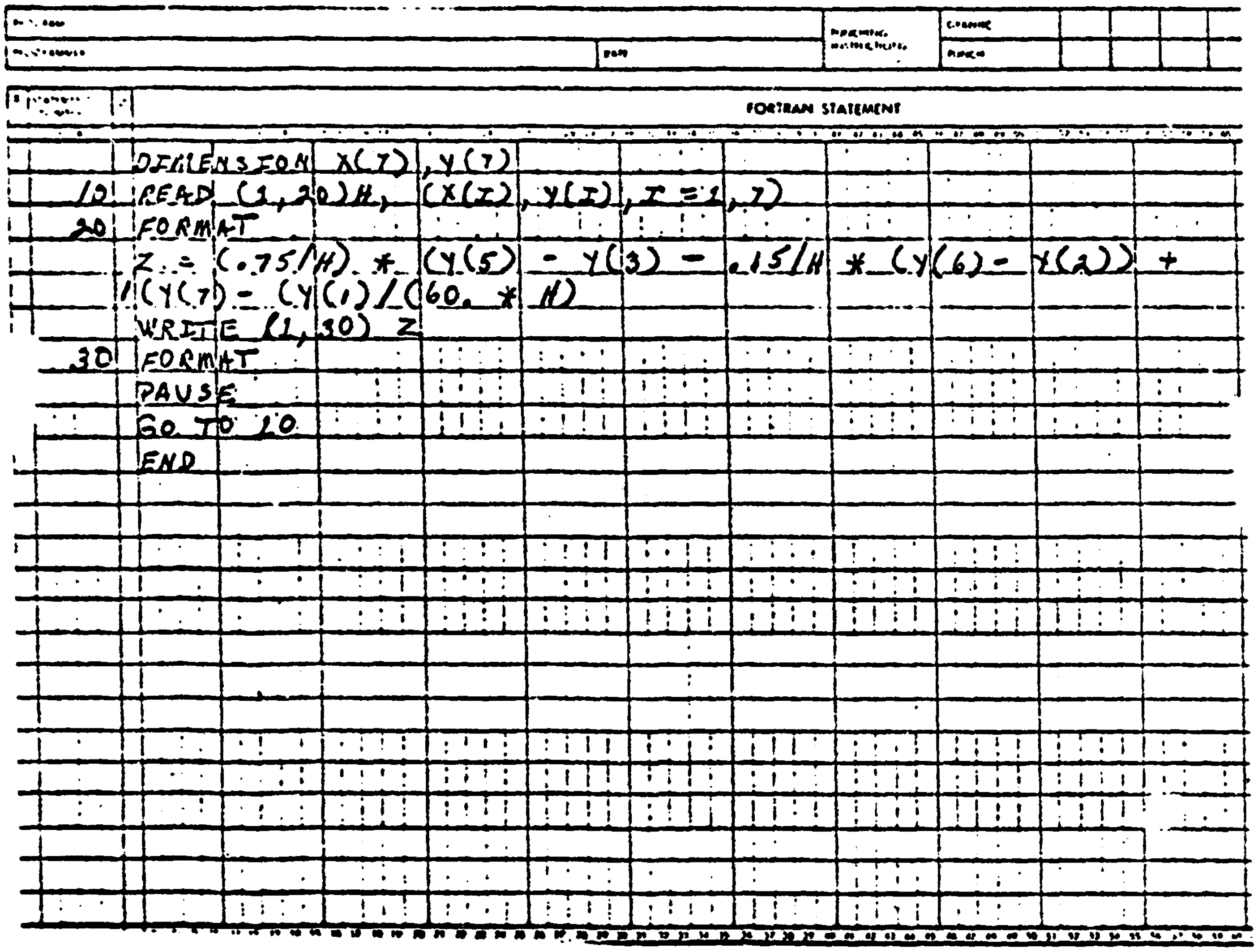


FORTRAN STATEMENT

```
   10 READ (1,11)I,S
   11 FORMAT
      GO TO (20,30,40)I
   20 T = S**2
   25 WRITE (1,26)T
   26 FORMAT
      PAUSE
      GOTO 10
   30 T = S**3
      GO TO 25
   40 T = S**4
      GO TO 25
      END
```

Figure FO-13.6

13.7 THE DERIVATIVE

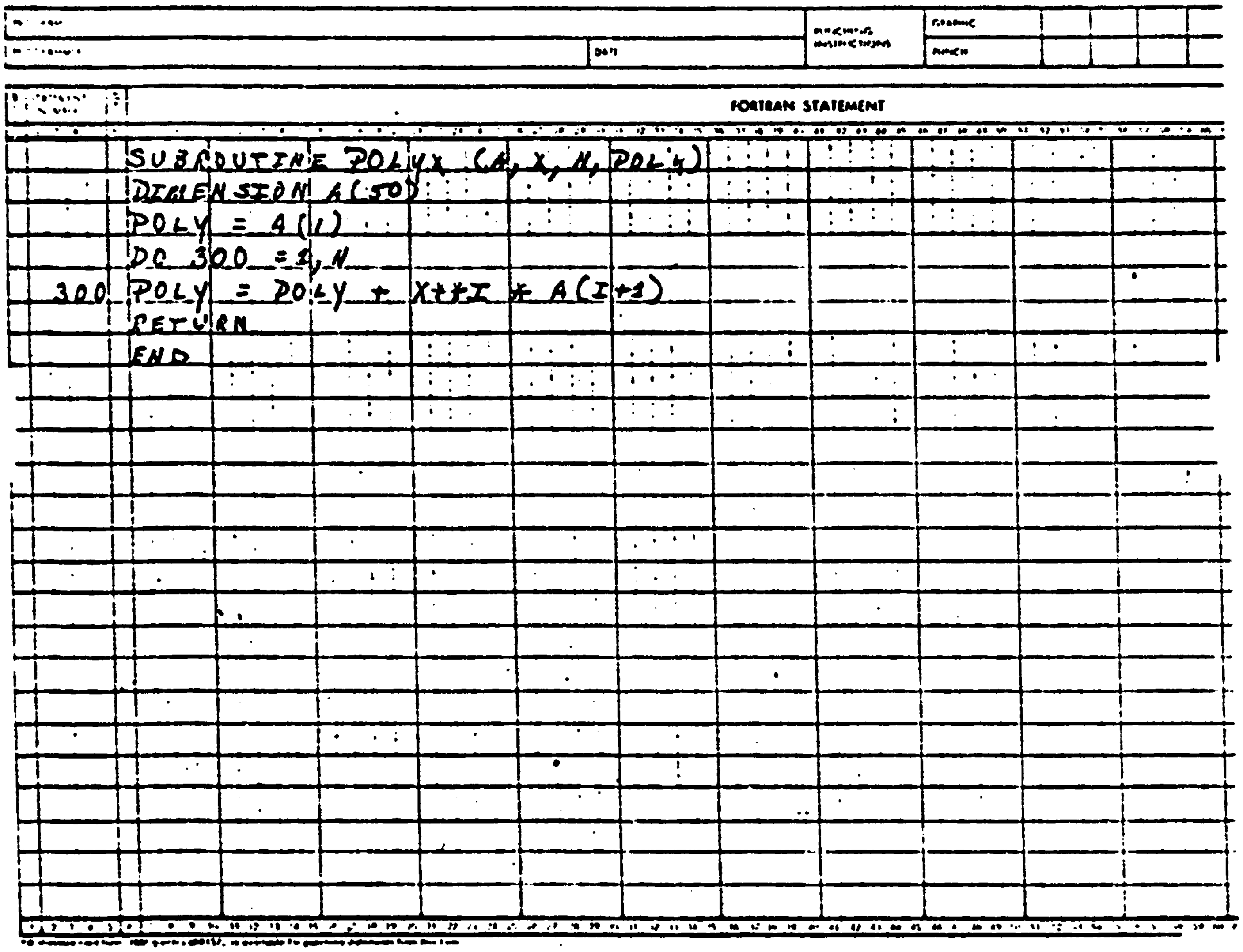


FORTRAN STATEMENT

```
      DIMENSION X(7),Y(7)
   10 READ (1,20)H, (X(I),Y(I),I=1,7)
   20 FORMAT
      Z = (.75/H) * (Y(5) - Y(3) - .15/H * (Y(6) - Y(2)) +
     1(Y(7) - (Y(1)/(60. * H)
      WRITE (1,30) Z
   30 FORMAT
      PAUSE
      GO TO 10
      END
```

Figure FO-13.7

13.8 EVALUATE A POLYNOMIAL

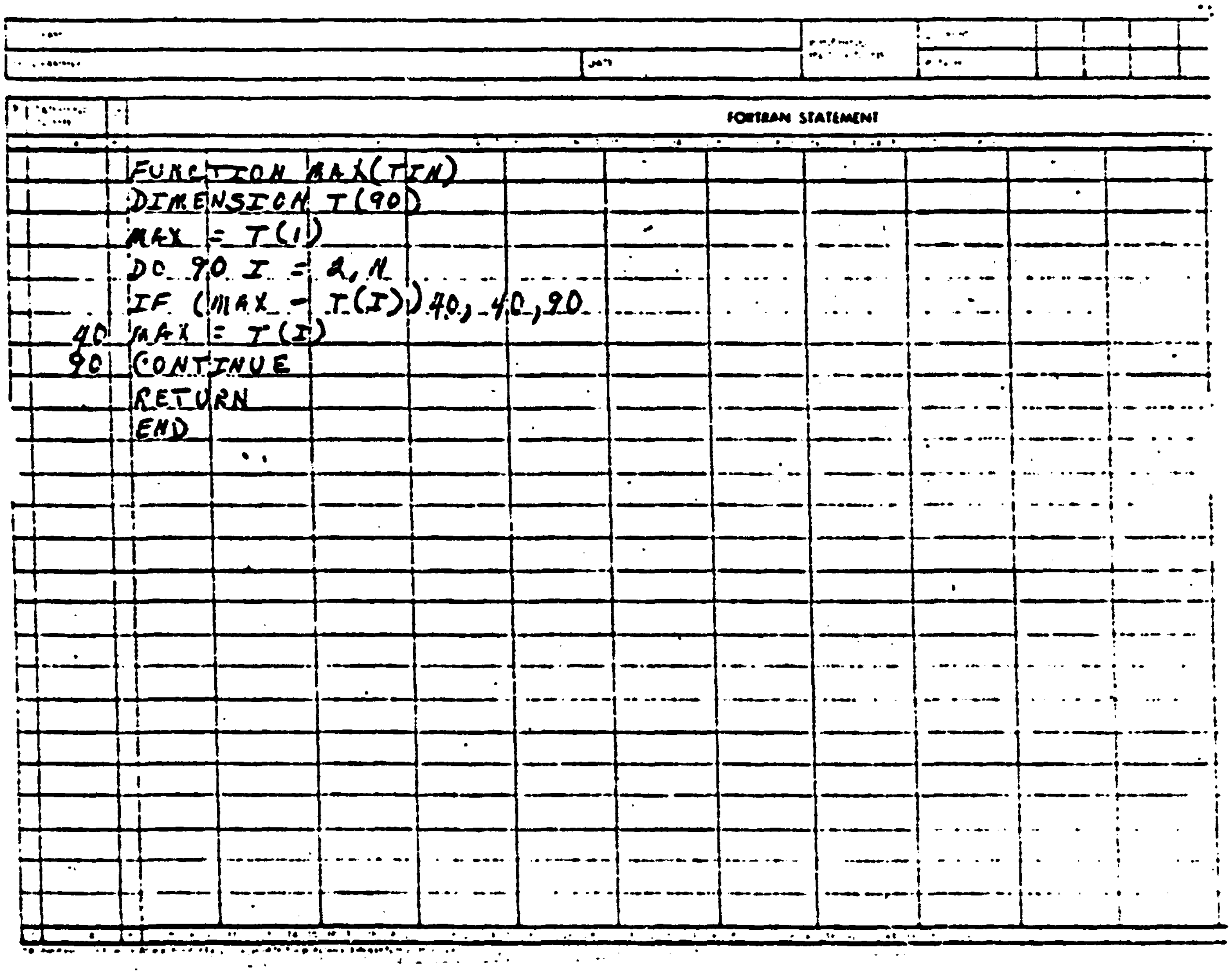


DATE
PUNCHING INSTRUCTIONS
GRAPHIC
PUNCH
FORTRAN STATEMENT

```
      SUBROUTINE POLYX (A, X, N, POLY)
      DIMENSION A(50)
      POLY = A(1)
      DO 300 = 1,N
  300 POLY = POLY + X**I * A(I+1)
      RETURN
      END
```

Figure FO-13.8

13.9 FUNCTIONS



FORTRAN STATEMENT

```
      FUNCTION MAX(TIN)
      DIMENSION T(90)
      MAX = T(1)
      DO 90 I = 2,N
      IF (MAX - T(I))40,40,90
   40 MAX = T(I)
   90 CONTINUE
      RETURN
      END
```

Figure FC-13.9

14. ARITHMETIC QUESTIONS AND ANSWERS

The answers immediately follow the questions for the first six numbers.

Examples

1. V = C + E**(1./N)*(R/2)-4

   $V = c + e \;\; 1/n. \; r/2-4$

2. $Y = (a + b)^{1/6}$

   Y = (A + B)**(1/6)

3. $r = \frac{a(a + b)}{b^2 - c}$

   $R = A*(A + B)/(B{**}^{2}-C)$

4. $V = 4/3 \; \pi r^3$

   V = (4./3)*3.14*(R**3)

5. $y = \frac{A + B}{C}$

   Y = (A + B)/C

6. x/y^{P-1}

   X/Y**P-1

Questions

1. $(3 \; . \; B)^{-1/3}$

2. (2.* D * C)**(1/2)

3. (3.*A*B)**(X-3)

4. $1/8 \; \frac{(4 + A)}{3-B}$

5. 3 (A + B)

6. 5A/2B

7. 4A + 2

8. 6(2/A)B

9. (6 + 2/A) B

10. 6 - 2/A - B

11. (4B/2C)

12. ((5.-A)/(3.-B))*C

13. B/2+C

14. 6 + A/2 - B

15. 6 + A/3B

16. 6 + 2/A

17. 6. + 2*A

18. A - C + 4

19. $y = \frac{a^2 + b}{b^2 - 2ab}$

20. B*2**L-4. *P/C

21. $3 \cdot \frac{sales \cdot ocost}{Xinv - ucost} \quad 1/2$

22. $\frac{(y + a)/r}{f + 1} \; . \; P^{1/3} + \underline{v \cdot 2}$

23. $h + (d/e)^2 \quad . \quad \frac{f \cdot g \cdot r}{s + t} \quad 1/3$

24. $y - 3mn^2 - r/s^3$

Answers

1. (3*A*B)**(-1/3)
2. $(2DC)^{1/2}$
3. $(3AB)^{x-3}$
4. (1./8.)*((4. + A)/(2.-B))
5. 3.*(A + B)
6. .5*(2. + B)
7. 4. * A + 2.
8. 6.-* 2./A*B
9. (6 + 2./A)*B
10. 6. - 2./A - B
11. (4.*B/((2.*C))**P
12. $\frac{5 + A}{3 - B} \quad C$
13. B/(2.*C)
14. (6. + A)/(2. - B)
15. (6. + A)/(2.*B)
16. 6. + 2./A
17. 6 + 2B
18. A - C + 4
19. y - (A**2 + B)/(B**2 - 2*A*B)

20. $b2^{1} - 4p/c$

21. (3*SALES * OCOST)/(XINV-UCOST))**0.5

22. ((Y + A)/4)/(F + 1) * (P**(1./3.) + (V*Z)/(B*W))

23. (h + (D/E)**2*((F*G*R)/(S + T))**(1./3.)

24. Y = 3*M*N**2 - R/S**3

Activities

1. Write a Fortran program pertaining to the compound interest rate.
2. Write an installment note program.
3. Prepare an integration model using Simpson's Rule.
4. Write a program and draw a flowchart for the quadratic equation.
5. Write a program to solve linear equations.
6. Prepare a program utilizing the computed GO TO statement.
7. Prepare a program to compute the derivative.
8. Write a program to evaluate a polynomial.
9. Write a subroutine subprogram to invert an array.

   Use three arguments:

   1. The given array
   2. The array into which the inverted array will be placed.
   3. An integer quantity that tells how long the arrays are. The array will be called TURBACK.
10. Write a function subprogram, which includes a dimension statement, to find the algebraically smallest quantity in

an array with a maximum of five hundred values. The function is to be called LEAST, and it will have two arguments:

1. The dummy array name.
2. An integer variable telling how many numbers are in the array.

Fortran Language Statements

Equivalence Statement

Form:

```
EQUIVALENCE (D, A, C(3))
```

1. The variables D, A, and C(3) will be stored in the same location in memory, however they can't be stored together at the same location at the same time.
2. This allows a saving of space, since after one variable is used, the next variable then occupies the space etc.

Subroutine Statement

Form:

SUBROUTINE NAME (A_1, A_2 A_3...)

1. The Dimension statement is non-executable.
2. It tells the compiler the amount of storage to set aside for each variable.
3. Any number of variables, separated by commas, may be written in the statement.

Dimension Statement

Form:

```
DIMENSION B (40, 40)
```

1. The Dimension statement is non-executable.

Form:

```
       READ 13 IMNAM, RATE
13     FORMAT F6.1, F6.3)
```

1. The format statement is non-executable.
2. It specifies input.
3. It specifies output.
4. The F specification transmits only floating point numbers to and from internal storage.

Go To Statement

Form:

```
GO TO (100, 101, 105, 120),L
```

1. Control is transferred to the statement numbers when the value of the indexing register is equal to each one.

Go To Statement (unconditional branch)

Form:

```
GO TO 15
```

An unconditional branch is made to statement 15.

Continue Statement

Form:

```
10 CONTINUE
```

X Specification Statement

Form:

```
                READ 105, SRATE
        105     FORMAT (33X F5.0)
```

1. It describes the number of positions to be ignored when a card is read.
2. Output causes blank characters to be inserted into the positions indicated. (There will be thirty-three blanks in this example.)
3. The numerical value of SRATE is punched into the thirty-fourth columns.

Read Statement

Form:

```
                READ 110, RATE
        110     FORMAT (E12.1)
```

1. It is nonexecutable.
2. It depicts the form in which the data will appear on input.
3. It shows how the data will be converted for storage in memory.
4. It also specifies, for output, the manner in which the data will be converted from memory and how it will appear in the output device for printing or punching.
5. The number is converted to internal floating point, (ENAM).

F Specification Statement

1. It tells the compiler the amount of storage to set aside for each variable.
2. Any number of variables, separated by commas, may be written in the statement.
3. The example presented here has two dimensions and will set aside sixteen hundred locations (forty times forty).

H Specification Statement

Form

```
          READ 150
    150   FORMAT (15H H GO TO TOWN)
```

1. The alphameric information punched in the first fifteen columns of a card replace the fifteen characters that are in temporary storage. These fifteen characters in temporary storage were previously written in statement one hundred fifty.

I Specification Statement

Form:

```
          READ 105, SRATE
    105   FORMAT 33X F5.0
```

1. It describes the number of positions to be ignored when a card is read.
2. Output causes blank characters to be inserted into the positions.

3. It is the last statement of a DO loop, when the DO loop would cause a transfer type statement of the wrong kind.

DO Statement

Form:

```
          DO 8 = 1, 10, 3
          A(1) - 8
          A(1,1) = 1,1
          A(1,2) = 1,2
8         PRINT 9, A(1), A(1,1), A(1,2)
9         FORMAT  (3 F8.0)
```

1. The DO statement tells the computer to execute repeatedly the statements which follow up to, and including the statement, with the statement number given (it is eight in this case).
2. The third number to the right of the equal sign is the amount of the increment for the index register.
3. After the statements have been executed ten times (n), control passes to the statement number given here as eight.

If Statement

Form:

```
IF(INT-100) 20, 2, 4
```

The expression within the parentheses is evolved:

1. If the value is negative, control is transferred to the second control statement.
2. If the value is zero, control is transferred to the second

control statement.

3. If the value is positive, control is transferred to the third statement.

Arithmetic Statements

Form:

b = d

1. The b is a variable, which can be subscripted.
2. The d is arithmetic in type, which is used to form a sequence of constants, operation symbols and variables.
3. The sequence connotates a series of calculations. The expression on the right hand side of the equal sign is completed, then the resulting numerical value is stored in the location assigned to the variable on the left side of the equal sign.

Call Statement

Form:

CALL TAX (BIC)

1. Tax is the name of a subroutine subprogram, and BIC is an expression of fixed or floating-point constants or variables.
2. The CALL statement calls the subroutines and does one of the following:
   a. It receives results back from the subprogram, or
   b. Sends data to the subprogram through an argument list.

Print Statement

Form:

```
                PRINT 10 PROD, YTOT
          10    FORMAT (F12.2, F10.2)
```

1. The two totals in the Format statement are printed.

Pause Statement

Form:

```
                PAUSE 32
```

1. A halt is taken when the program reaches the pause statement.
2. Continue by pressing the start.

Slash and Repetition Statements

Form:

```
                FORMAT (4/ E 10.2//(3F10.7))
```

1. The first slash causes the typewriter to start a new line.
2. Then skip a line.
3. Print R numbers per line until all numbers in the list are printed.

End Statement

Form:

```
                END
```

1. It informs the compiler the translation is completed.

Type Statement

Form:

```
          TYPE 100 TAX, ITEM
100       FORMAT (F10.0, F9.0)
```

1. The two variables are retrieved, converted and printed in the first nineteen positions.

Return Statement

Form:

1. Return indicates a subprogram has been completed and causes an unconditional branch back to the main program.

Stop Statement

Form:

```
          STOP 999
```

1. It causes a halt.

Read Statement

Form:

```
          READ 10, EARN
10        FORMAT (F10.2)
```

1. When READ is reached in the program a sufficient number of cards are read to satisfy the list of variables.
2. The format statement describes the data in detail.
3. The data is stored in memory as the floating point EARN.

Punch Statement

Form:

```
          PUNCH 100, STAX, EARN
100       FORMAT (F10.0, 16)
```

1. This statement is similar to the read statement.

Function Statement

Form:

FUNCTION LOW (P, Q)

1. One argument must be included in the function statement.
2. The function name must consist of six alphameric characters, and the first must be alphabetical.
3. The function statement operates when the name of the function is encountered in the arithmetic statement.
4. Listed arguments move data to the function routine, and a single quantity is returned to the calling program.

SECTION IV

RPG

IV RPG

The purpose of this section is to provide models and information which will aid in the preparation for the DPMA test. The Basic Assembler program presented here is used as a comparison for the invoicing program written in Report Program Generator (RPG).

The reader is advised to review all the pertinent forms for RPG since they are considered important for the test. The latter part of the section provides the necessary information to perform the Swanson Type programs with RPG, and presents the assignment that could make this text last a year or longer.

Object Program Logic

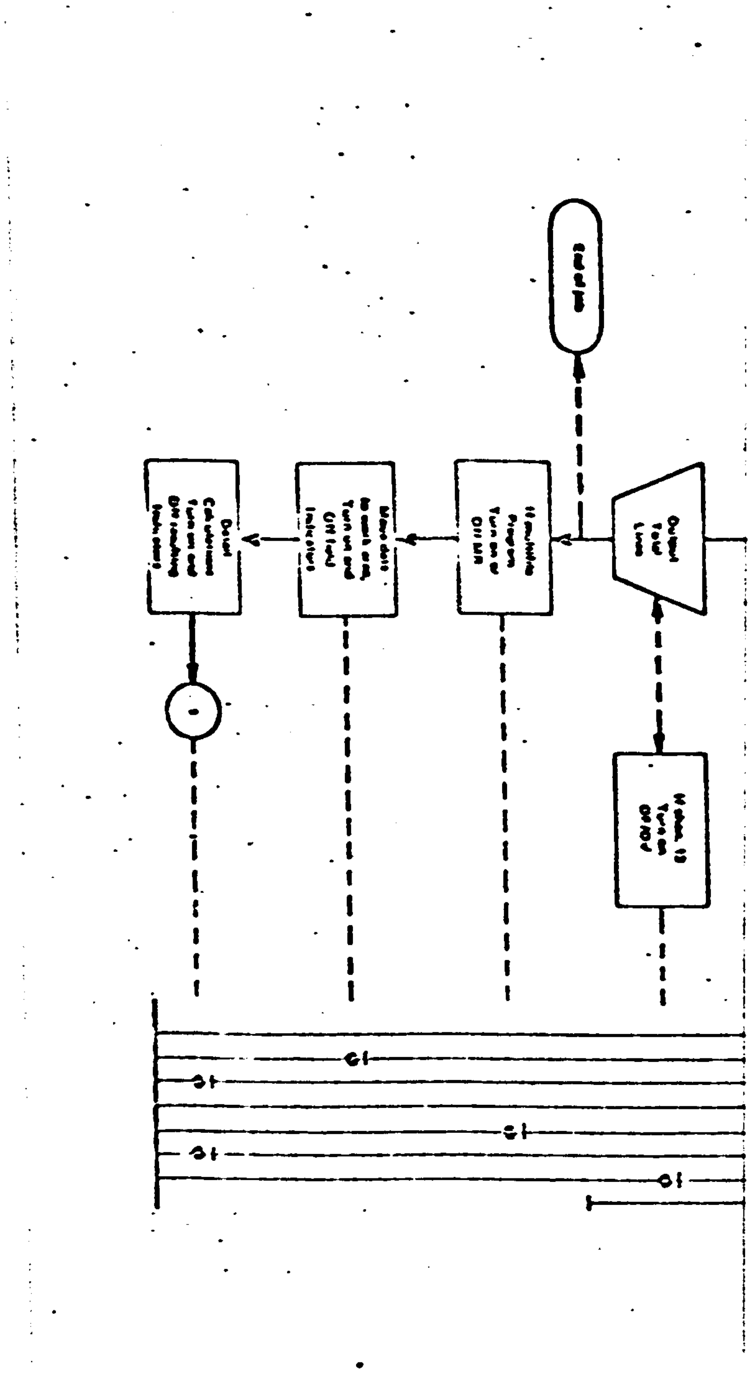

Output Total Lines
If chan. 12 Turn on OF/OV
End of job
If multifile Program Turn on or Off MR
Move data to work area, Turn on and Off field Indicators
Detail Calculations Turn on and Off resulting Indicators

Figure RPG-0

PRE-BILLING CALCULATION WITH INVENTORY CONTROL

This example illustrates one of numerous approaches to an order-processing/inventory control job. The application has been arbitrarily slanted to a distribution business (perhaps a mail-order house) with customer orders to be filled from warehouse stock. An attempt has been made to be reasonably realistic in the application, including the complexities of such a multipurpose operation.

BASIC ASSUMPTIONS

1. A card has been keypunched:

   a. For each item line on a customer order--Card 9, no X in col. 11

   b. For each item line on a customer return--Card, X in col. 11

   c. For each item line on a stock receipt (or purchase-order cards are used as stock receipt cards)--Card 5

   d. For each stock adjustment--Card 6: No X in col. 11 to reduce on hand, X in col. 11

   e. For each item on a stock purchase order--Card 7, no X in col 11 when ordered, X in col. 11 if order is cancelled or reduced

   f. For a new stock item or a change in price, description, warehouse location, etc. (Obsolete master cards are removed manually or, at least, separately from this operation.)

2. An Inventory Master Card file exists, with one card per item carried in stock. Changes to the file are made manually, or in some other data processing operation (i.e., addition and deletion of items, changes in price, warehouse location, etc).

3. It is desired to process customer order-item cards against inventory records before attempting to fill the orders in the warehouse. At the same time, the inventory records will be updated and an up-to-date inventory report prepared. The customer-order cards are thereafter ready for invoicing. (The cards could be sorted by warehouse location prior to invoicing.) A copy of the invoice, or the cards themselves, serve as order-picking medium, i.e., either sequential or bulk picking is employed. If orders are processed once daily on this basis, the inventory records are always up to date.

4. If the quantity on-hand is insufficient to satisfy the quantity in the customer-order card, no partial quantity will be applied for that item. The item order:

   a. Will be marked "cancelled" if no stock is on order; or

   b. Will be marked for back order if not previously back-ordered, and provided stock is on order; or

   c. Will be marked "cancelled" if previously back-ordered.

5. Where previously back-ordered item cards are re-entered, they are to receive priority for available inventory.

6. Some items have a lower unit selling price when at least the specified criterion quantity is ordered by the customer.

7. Stock adjustments are made without attempt at modifying the unit cost of the item.

8. a. Besides price extension, gross profit is to be included in the item detail cards for a subsequent report by merchandise class and division, and by Stock No. (The first digit of Stock No. represents merchandising division, the second the

classification within division.)

b. Value of inventory on hand (average cost basis) is to be continually available.

c. Available quantity (on-hand plus on-order) less than an established minimum to be signalled.

PROCEDURAL DETAILS

1. Safeguards

a. Certain control totals will be carried, partially as audit trails. Control totals are presumed to have been established for the various kinds of transaction cards, so that new on-order totals can be proved out.

b. Customer-order detail cards that are being cancelled will be identified. If such a card is re-entered, it is selected out, and calculation for it is bypassed.

c. Matched old master cards (for which new ones are created) will be identified, and selected to a separate stacker. If such a card is accidentally re-entered, the entire stock-number group is selected to separate stacker, and calculation is bypassed.

d. The entire stock-number group (except the first card) is selected to a separate stacker, processing is bypassed, and the system halts after the second card, whenever there is more than one master card for a group.

e. The entire stock-number group is selected, and calculations are bypassed, when a master card with a negative on-hand quantity has been read. When a negative on-hand quantity is created as a result of calculation, the cards from the point of error are selected, and calculations ar bypassed.

f. Whenever the blank trailer card is missing or mispositioned within the group, all cards in the group from the point of the error detection are selected, the system halts, and further calculation is bypassed for the group.

g. Unmatched transaction cards, including the trailing blank card, are selected, and calculations are bypassed.

h. If the on-order quantity turns negative, the system halts. The inventory report also indicates this condition.

i. For known error conditions that affect the new inventory values, the data is omitted.

2. Any merchandise receipts, stock adjustments, and customer returns precede order-item details, so that the customer orders are correctly applied to the latest on-hand status. Stock purchase-order cards are also placed ahead of customer order details, because it was decided not to back-order items for which no stock is on order. Former back-order cards precede other order-item cards to get first chance at on-hand goods.

3. The cards are assumed to be in ascending sequence by Stock No. Inventory master cards are to be in the primary feed of the MFCM--preceded by a single card to read in today's date. All other cards will be placed in the secondary feed.

A previous operation has placed a blank card at the end of each Stock No. group of secondary file cards. These blank cards will become the new (updated) inventory master cards for stock numbers for which there are transactions. (These blank cards were merged in on the MFCM of the Model 20, or they could have been merged on a collator.)

4. Stacker Selection

a. The date header card is directed to stacker 1; any other stacker would do equally well.

b. All old inventory master cards with stock numbers, for which there are transaction cards in the secondary file, are directed to stacker 1 (the normal stacker-chosen to contain obsoleted cards), because a new inventory master card will be punched and placed in stacker 2.

Each unmatched old inventory master is selected to stacker 2, because no new master is punched in such case.

Stacker 2 ultimately contains the complete up-to-date inventory master file (except for known error-condition cards) consisting of new cards where transactions occurred and old masters where no transactions applied.

c. Stacker 3, receives the customer order-item cards, ready for warehouse picking (if cancelled and BO (back-orders) are sorted out), or to be sorted on order and account numbers for invoicing.

d. Stacker 4 has been assigned to unmatched transaction cards (secondary file), and to all other detected error-condition cards.

e. Stacker 5 has been assigned to stock orders, receipts, adjustments, and merchandise returns. These may also be left together with the other transaction cards by directing them to stacker 3 instead; they could easily be segregated by sorting cols. 1 and 11.

Card Layouts

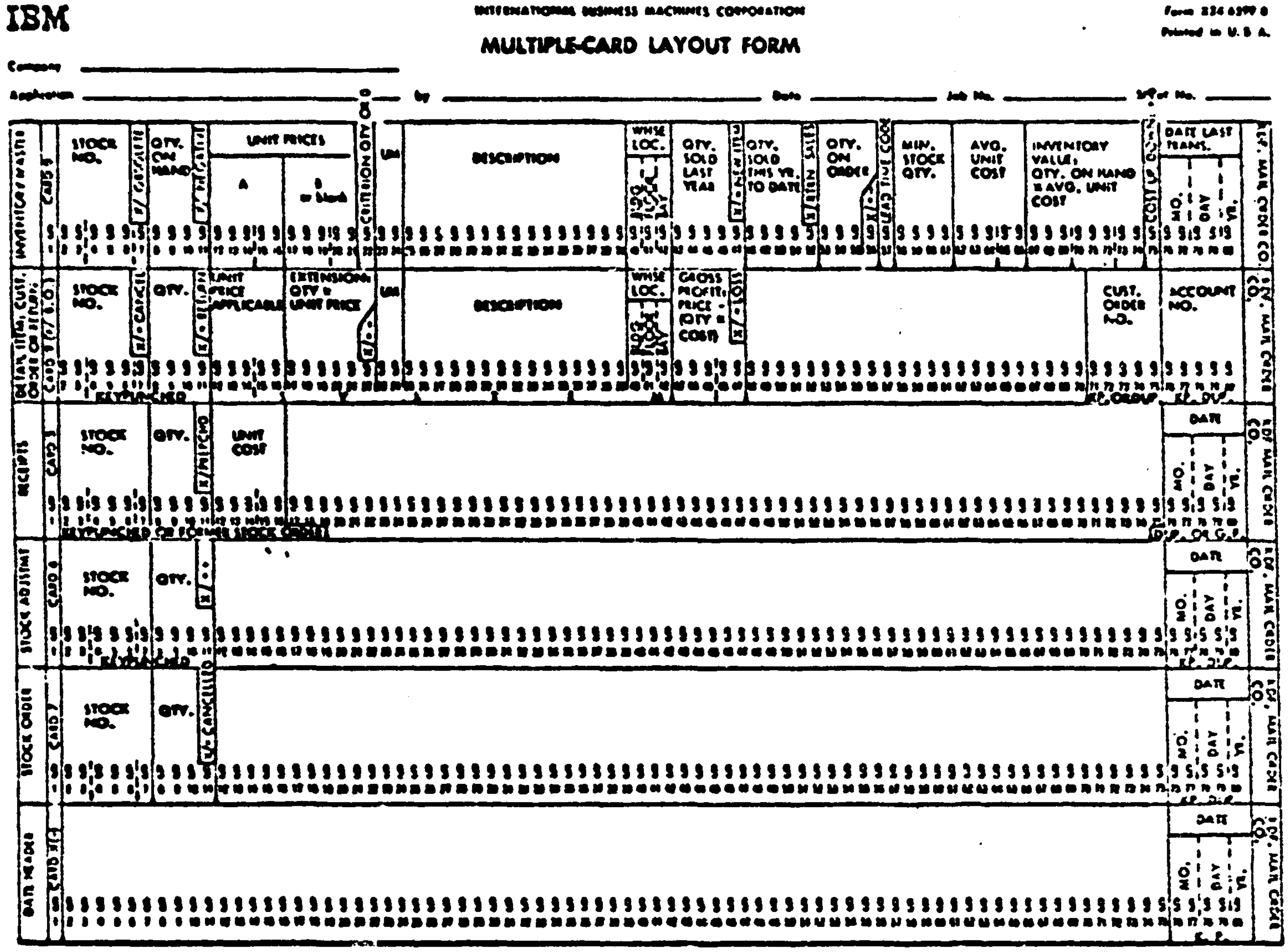


Figure PO

Diagram of Card Flow

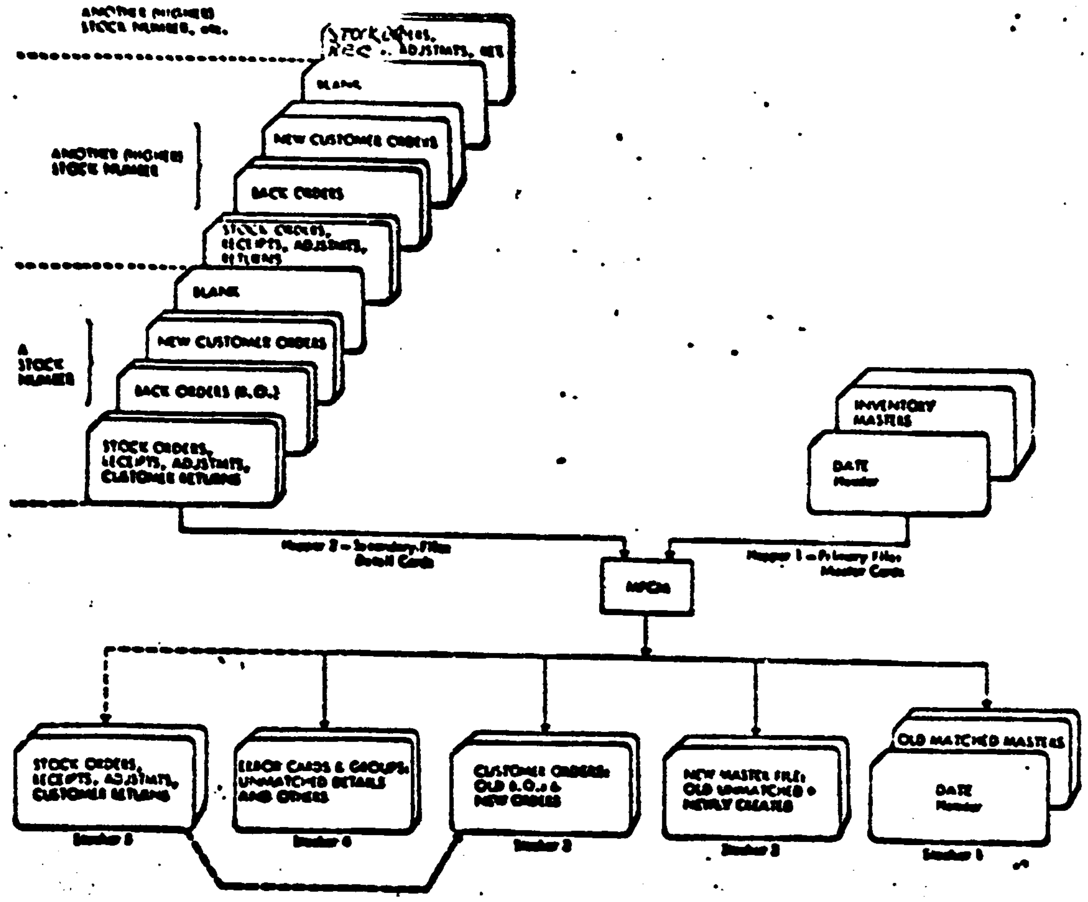

ANOTHER (HIGHER) STOCK NUMBER, etc.
STOCK ORDERS, REC., ADJSTMTS, ETC
BLANK
ANOTHER (HIGHER) STOCK NUMBER
NEW CUSTOMER ORDERS
BACK ORDERS
STOCK ORDERS, RECEIPTS, ADJSTMTS, RETURNS
BLANK
A STOCK NUMBER
NEW CUSTOMER ORDERS
BACK ORDERS (B.O.)
STOCK ORDERS, RECEIPTS, ADJSTMTS, CUSTOMER RETURNS
INVENTORY MASTERS
DATE Header
Hopper 2 – Secondary Files Detail Cards
Hopper 1 – Primary Files Master Cards
MFCM
STOCK ORDERS, RECEIPTS, ADJSTMTS, CUSTOMER RETURNS
Stacker 5
ERROR CARDS & GROUPS: UNMATCHED DETAILS AND ORDERS
Stacker 4
CUSTOMER ORDERS: OLD B.O.s & NEW ORDERS
Stacker 3
NEW MASTER FILE: OLD UNMATCHED + NEWLY CREATED
Stacker 2
OLD MATCHED MASTERS
DATE Header
Stacker 1

Figure POO

Study of card layouts and the systems flow as seen in Figure P-1 will clarify the details of the operation. The report has been laid out to fit within the 120-position print span of all IBM 2203 and 1403 Printers attachable to model 20. Explanation of specifications sheets follows.

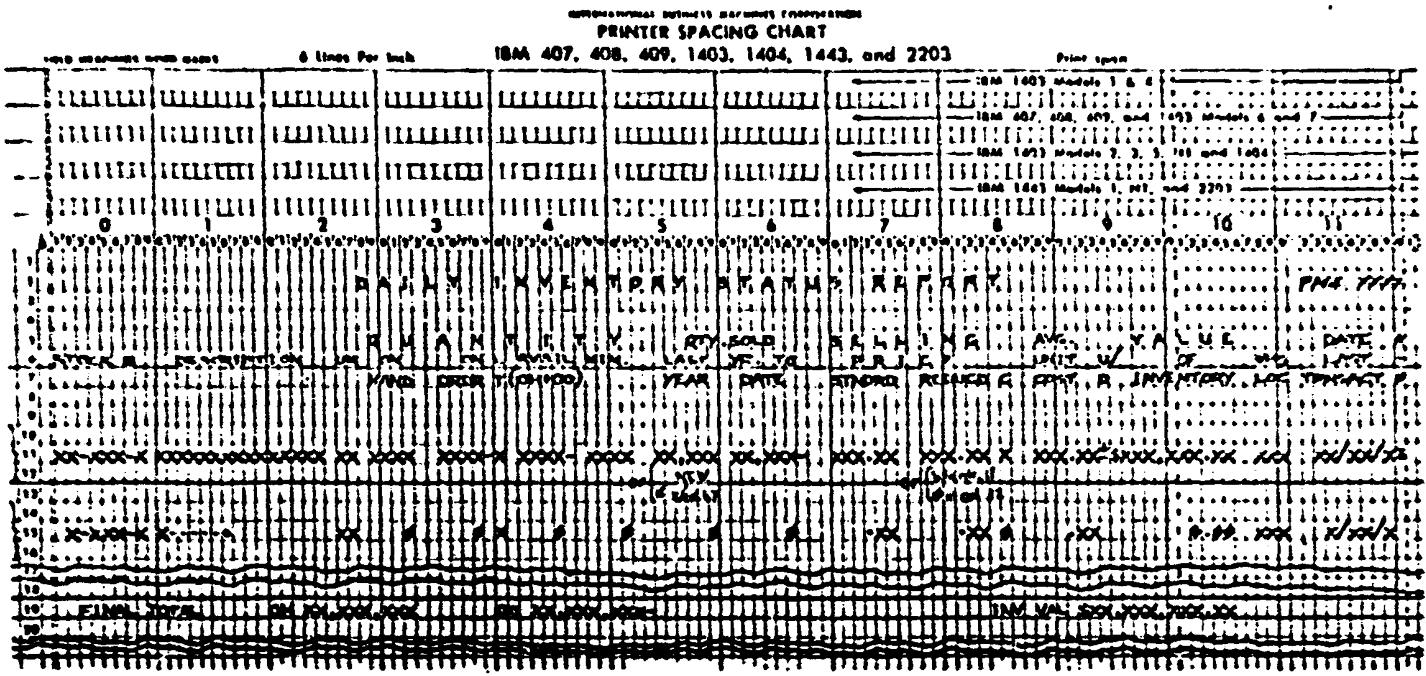


Figure POOO

FILE DESCRIPTION SPECIFICATIONS (Figure P-1)

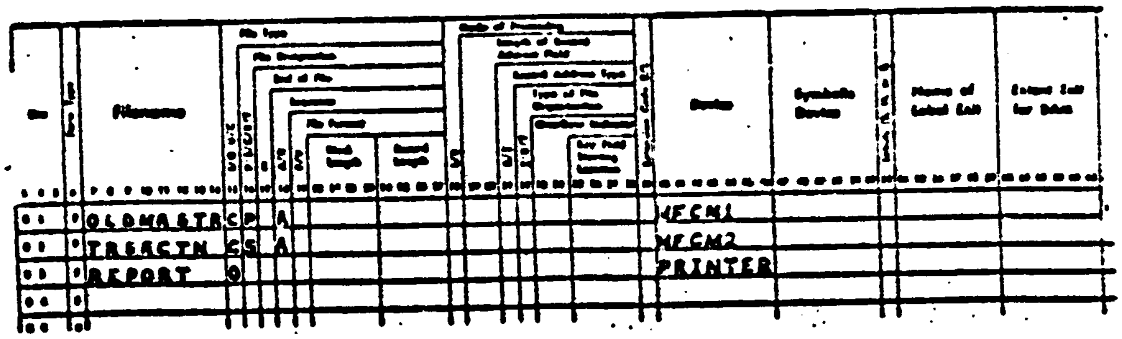
The file inventory master cards is named OLDMASTER, and associated with the primary hopper of the MFCM. It is defined as a combined file (C in col. 15) so that stacker selection may be performed via output specifications, and to allow punching of a code for "obsolete" at output time into those old masters that are replaced as a result of new transactions.

The detail transaction cards are assigned to the file named TRSACTN, and associated with the secondary hopper of the MFCM. Stacker selection is dependent on calculation operations; therefore--and because output is required to some customer-order item cards--TRSACTN is a combined file.

The input files are in ascending sequence (A in col. 18). A sequence is required, and must be uniform for the input files, when matching of records is called for in two or more files. If col. 17 is blank, or contains E, for all input files, the LR indicator does not turn on until all input files are exhausted.

The printer is associated with an output file named REPORT.

File Description Specifications



OLDMASTR	C	P	A	MFCM1
TRSACTN	C	S	A	MFCM2
REPORT	O			PRINTER

Figure P-1

INPUT SPECIFICATIONS (Figure P-2)

Because the file OLDMASTR is specified ahead of the TRSACTN file, it is therefore the primary file; i.e., matching cards from the OLDMASTR file are processed ahead of their matching TRSACTN file cards.

					Record Identification Codes				Field Location						Field Indicators		
		Filename			Position				From	To		Field Name					
		OLDMASTR	IM	Ø1	1	C	Ø										
									2	7	Ø	STKNO	L1	M1		97	
									8	11	Ø	ONHAND				99	
									12	16	2	PRICEA					
									17	21	2	PRICEB					
									22	22	Ø	CRITQT					2Ø
									23	24		UNMEAS					
									25	39		DESCR					
									4Ø	42		WHSLOC					
									43	47	Ø	LASTYR					
									47	47		NEVITM					
									48	52	Ø	YRTDAT					
									53	56	Ø	ONORDR					
									57	57		LEADTI					
									58	61	Ø	MINQTY					
16									62	66	2	AVGCST					
17									67	74	2	INVVAL					
18									76	8Ø	Ø	TRNSDA					

Figure P-2

Input Specifications

					Record Identification Codes				Field Location						Field Indicators		
		Filename			Position				From	To		Field Name					
			D4	Ø9	1	C	-	1									
									76	8Ø	Ø	DATE					
		TRSACTN	TR	11	1	C	5										
			OR	12	1	C	6										
			OR	13	1	C	7										
			OR	15	1	D	9										
									1	1	Ø	DOCARD				21	
									2	7	Ø	STKNO	L1	M1		98	
									8	11	Ø	QTY				22	
									12	16	2	UNCOST					
									76	8Ø	Ø	ACCNT					
			BL	19	1	C	1										

Figure P-3

INVENTORY MASTER CARDS--OLDMASTR File (Page 02)

The old Inventor/ Master cards are identified by 0 in col. 1, and assigned indicator 01. Since they are the only card type in the file, apart from the initial single Date card, an alphabetic code is specified in Sequence (cols. 15-16). If any other undefined card types (besides the Date or Master card) appear in the file, the system halts.

Lines 02-18 contain the normal specifications for reading those fields from the old Inventory Master cards that may be needed for processing of the application. Fields defined as numeric are used in calculations, edit operations, or numeric compare. Points of special interest are:

1. Stock No. is defined as numeric to allow formatting in the output by edit word, and to simplify detection of an obsolete master card (see 4 below).
2. The files are matched and sequence checked on Stock No. (M1 in cols. 61-62 for Stock No.).
3. The L1 indicator is turned on for the first card of each stock-number group (L1 in Control Level for Stock No.). L1 is not used in this program for total printing or punching, it is used solely to inherent connection with matching of files, and L1 is not needed merely because M1 is assigned.
4. Whenever an old Master Card is replaced by a new one, to reflect transactions, the old card is overpunched with an 11 -punch in col. 7 at output time (Fig. P7, line 06), to mark it as obsolete. If such a card is accidentally reentered next time, indicator 97 on-the 11-punch causes the

Stock No. to read in as negative (a matching-field sequence error does not, however, arise because all zone punches are eliminated from the matching-fields operations of a numeric field).

5. Indicator 99 turns on if the Quantity On-Hand is negative in the old Inventory Master card. Such a card should never appear, because subsequent specifications (Fig. P4) cause output to be bypassed if On-Hand turns negative.

6. Indicator 20 turns on if the Criterion Quantity field is zero. The zero code indicates that only Unit Price A applies, and that the Price-B field is to remain blank both in the report and in a new Inventory Master card.

7. Col. 47 appears twice among the input fields--the first time as part of a normal numeric field: the second time with another name and as a single-column alphameric field. If col. 47 is X-punched (11 -punch), Quantity Sold Last Year does not apply because the item is new this year. The word NEW is then to appear in the report, and the field is only to contain an X-punch in a new Inventory Master card. But a numeric field that is blank or zero with an X-overpunch in the units position will set on an indicator for Zero or Blank--not for Minus. Therefore, the column that contains the X-punch for "new" is separately defined as alphameric. It can then be tested for a minus done by a TESTZ calculation specification.

8. Stacker assignment is not known until calculations are per-

formed. It must therefore be specified at outpui time.

DATE CARD OLDMASTER FILE (Fig. P3)

The single Date card at the front of the file is identified by an X-punch in col. 1 and assigned indicator 09. The date is stored in a field given the name DATE. It is defined as numeric to allow editing.

No matching is specified for this card. It is therefore processed first. The date card is to enter the normal stacker for the MFCM primary hopper and therefore need not have stacker selection specified. However, when no output operation is to be performed on a combined-file card type, and the desired stacker number is known at input time, a stacker-selection specification, even for the normal stacker, should be given in the input specifications; this maximized I/C overlap. (For a single card in an entire file, this is of course insignificant). The File Name need not be repeated where no others intervened.

TRANSACTION CARDS (Except Blank Trailer) - TRSACTN FILE (Fig. p4)

The four types are identified and assigned separate indicators. The customer-order or merchandise-return item card is checked for digit rather than character 9 because back orders have an X-overpunch in col. 1.

Stacker selection is dependent on calculations, and is therefore assigned in the output specifications. In the case of card type 9 (indicator 15), output to the card is also required; this precludes stacker selection in the input specifications.

Points to note:

1. Indicator 21 is turned on for order-item cards that were previously back-ordered: 11/9 in the low-order, or sole, position of a numeric field indicates a negative value. (Back-order cards are so designated at output time--Fig. P7, line 17.) The field BOCARD is not used in the program; it is assigned only so that a Field Indicator may be set. Alternatively, a separate card-type Resulting Indicator could have been assigned via an OR line.

2. The same name is assigned to Stock No. here as for the OLDMASTR file, to conserve core storage space. No harm is done because there is no situation in this program where the distinction needs to be preserved.

3. When an order-item cannot be filled, and is not to be back-ordered, col. 7 of the card is overpunched with an 11 -punch (Fig. P7, line 18) to designate "cancelled." If such a card is inadvertently reentered, indicator 98 turns on because the 11 -overpunch causes Stock No. to be read as negative.

4. Indicator 22 distinguishes between order-item and merchandise-return cards--both card-type Resulting Indicator 15.

5. The field UNCOST applies only to Receipt cards. No harm is done reading it also from card types with Resulting Indicators 12, 13, and 15, because utilization in the calculation specifications is confined to card type 11

(Fig. P5, line 06). If it were necessary to restrict the input of this field to Receipt cards, the indicator number (11) would be entered in Field-Record Relation (cols. 63-64).

BLANK TRAILER CARD-TRSACTN FILE (Fig. P3)

The trailer cards-destined to become new Inventory Master cards--are identified by absence of a punch in col. 1, and are assigned indicator 19.

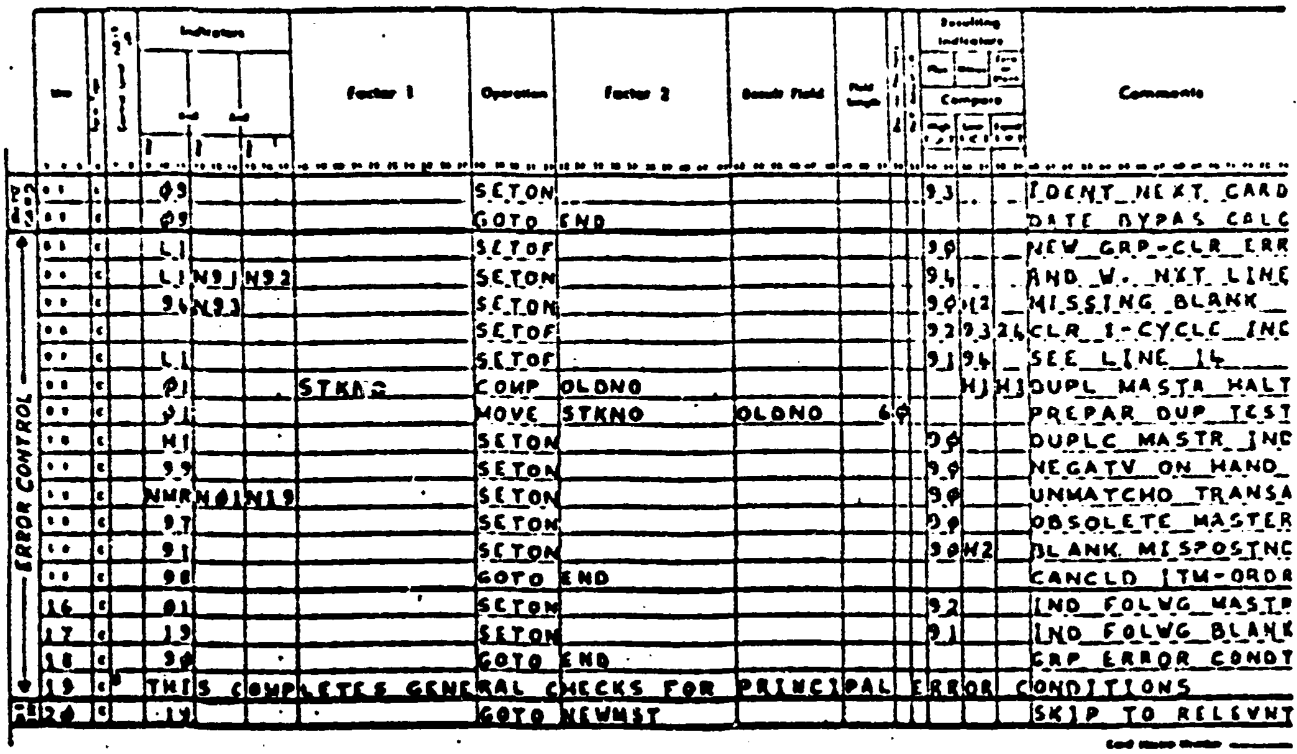
The blank trailer card at the end of each stock-number group in the TRSACTN file is not matched (no entry in Matching Fields) against the OLDMASTR file; therefore, it is processed immediately after the card it follows in the same file, before the Inventory Master card for the next Stock No.

CALCULATION SPECIFICATIONS (Fig. R4)

In order to minimize the need for conditioning indicators (Indicators, cols. 9-17), branching (GOTO) over entire sections has been employed to bypass a series of inapplicable calculation specifications.

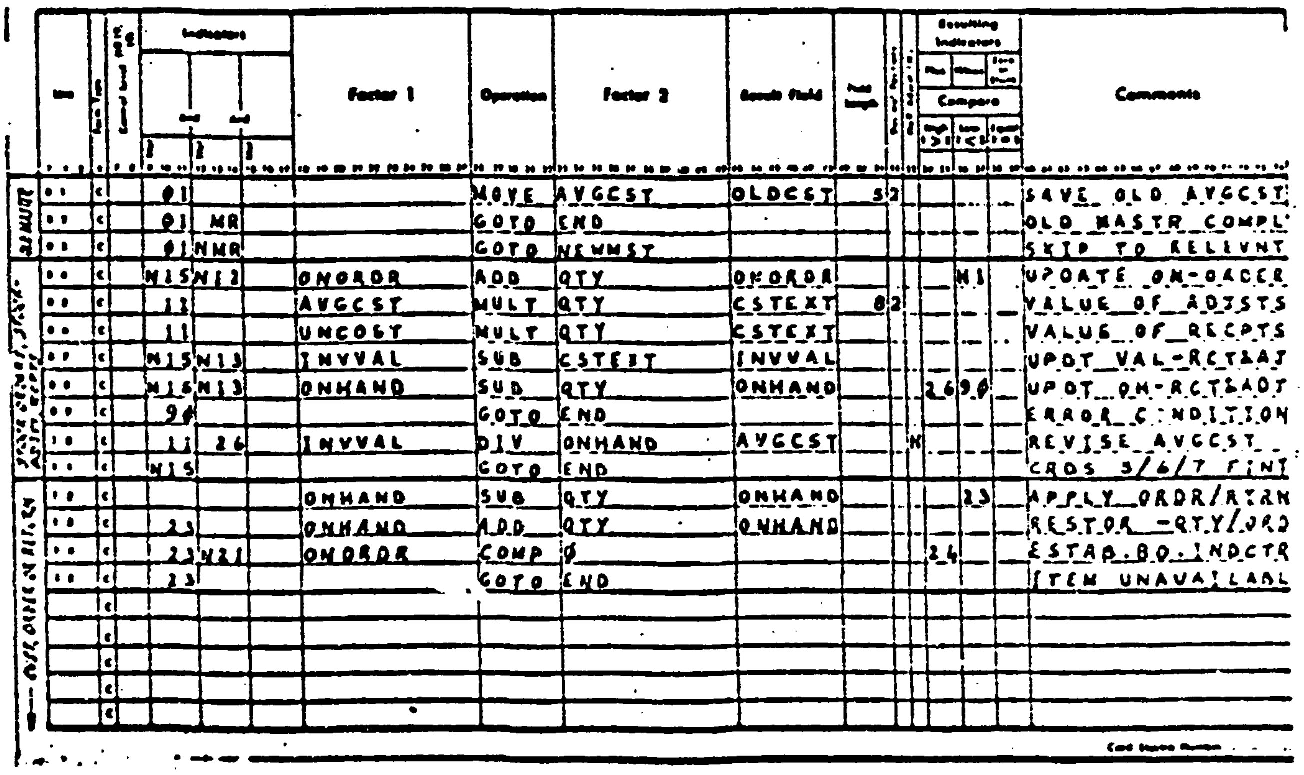
Where practical, specifications lines are discussed sequentially. In some areas, however, it is preferable, for clarity, to relate non-consecutive lines. Note: In several instances, result fields are defined as smaller than the theoretically possible maximum. We assumed that knowledge of the particular business indicated that these field sizes are adequate for the actual figures that could occur.

Calculations Specifications



	Line		Indicators	Indicators	Indicators	Factor 1	Operation	Factor 2	Result Field	Field Length		Resulting Indicators			Comments
[illegible]		C	Ø9				SETON					93			IDENT NEXT CARD
		C	Ø9				GOTO	END							DATE BYPAS CALC
ERROR CONTROL		C	L1				SETOF					9Ø			NEW GRP-CLR ERR
		C	L1	N91	N92		SETON					94			AND W. NXT LINE
		C	94	N93			SETON					9Ø	12		MISSING BLANK
		C					SETOF					92	93	24	CLR 1-CYCLE INC
		C	L1				SETOF					91	94		SEE LINE 14
		C	Ø1			STKNO	COMP	OLDNO					H1	H1	DUPL MASTR HALT
		C	Ø1				MOVE	STKNO	OLDNO	6	Ø				PREPAR DUP TEST
		C	H1				SETON					9Ø			DUPLC MASTR IND
		C	99				SETON					9Ø			NEGATV ON HAND
		C	NMR	NØ1	N19		SETON					9Ø			UNMATCHD TRANSA
		C	97				SETON					9Ø			OBSOLETE MASTER
		C	91				SETON					9Ø	H2		BLANK MISPOSTNG
		C	98				GOTO	END							CANCLD ITM-ORDR
	16	C	Ø1				SETON					92			IND FOLWG MASTR
	17	C	19				SETON					91			IND FOLWG BLANK
	18	C	9Ø				GOTO	END							GRP ERROR CONDT
	19	C	THIS COMPLETES GENERAL CHECKS FOR PRINCIPAL ERROR CONDITIONS												
[illegible]	2Ø	C	19				GOTO	NEWMST							SKIP TO RELEVNT

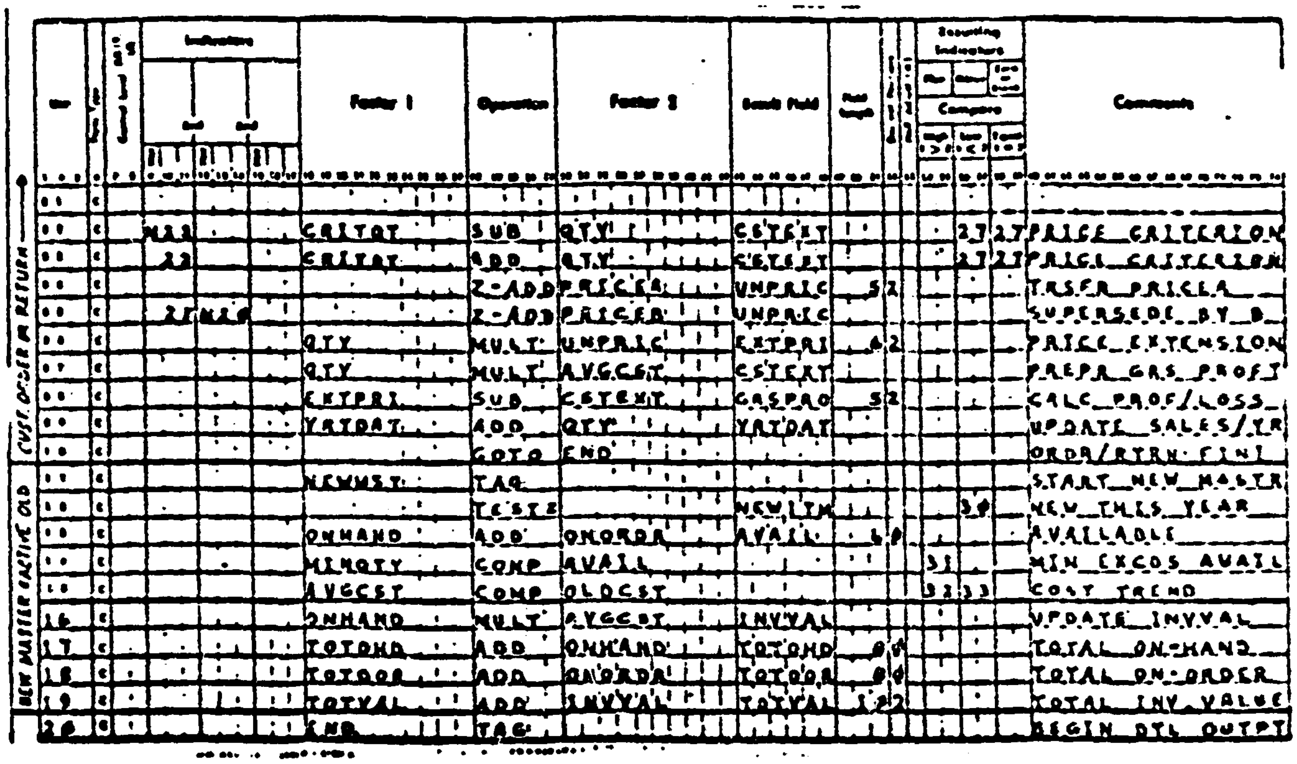
Figure P4



	Line		Indicators	Indicators	Indicators	Factor 1	Operation	Factor 2	Result Field	Field Length		Half Adjust	Resulting Indicators			Comments
[illegible]		C	Ø1				MOVE	AVGCST	OLDCST	5	2					SAVE OLD AVGCST
		C	Ø1	MR			GOTO	END								OLD MASTR COMPL
		C	Ø1	NMR			GOTO	NEWMST								SKIP TO RELEVNT
[illegible]		C	N15	N12		ONORDR	ADD	QTY	ONORDR					H1		UPDATE ON-ORDER
		C	11			AVGCST	MULT	QTY	CSTEXT	8	2					VALUE OF ADJSTS
		C	11			UNCOST	MULT	QTY	CSTEXT							VALUE OF RECPTS
		C	N15	N13		INVVAL	SUB	CSTEXT	INVVAL							UPDT VAL-RCT&AJ
		C	N16	N13		ONHAND	SUB	QTY	ONHAND				26	9Ø		UPDT ON-RCT&ADJ
		C	9Ø				GOTO	END								ERROR CONDITION
		C	11	26		INVVAL	DIV	ONHAND	AVGCST			H				REVISE AVGCST
		C	N15				GOTO	END								CRDS 5/6/7 FINI
[illegible]		C				ONHAND	SUB	QTY	ONHAND					23		APPLY ORDR/RTRN
		C	23			ONHAND	ADD	QTY	ONHAND							RESTOR -QTY/ORD
		C	23	N21		ONORDR	COMP	Ø					24			ESTAB.BO.INDCTR
		C	23				GOTO	END								ITEM UNAVAILABL
		C														
		C														
		C														
		C														
		C														

Figure P5

Calculations Specifications (con't.)



CUST. ORDER OR RETURN

NEW MASTER INACTIVE OLD

			Indicators										Resulting Indicators			
						Factor 1	Operation	Factor 2	Result Field				Compare			Comments
[illegible]	C															
[illegible]	C		N22			CRITQT	SUB	QTY	CSTEXT					27	27	PRICE CRITERION
[illegible]	C		22			CRITQT	ADD	QTY	CSTEXT					27	27	PRICE CRITERION
[illegible]	C						Z-ADD	PRICEA	UNPRIC	5	2					TRSFR PRICEA
[illegible]	C		22	N26			Z-ADD	PRICEB	UNPRIC							SUPERSEDE BY B
[illegible]	C					QTY	MULT	UNPRIC	EXTPRI	6	2					PRICE EXTENSION
[illegible]	C					QTY	MULT	AVGCST	CSTEXT							PREPR GRS PROFT
[illegible]	C					EXTPRI	SUB	CSTEXT	GASPRO	5	2					CALC PROF/LOSS
[illegible]	C					YRTDAT	ADD	QTY	YRTDAT							UPDATE SALES/YR
[illegible]	C						GOTO	END								ORDR/RTRN FINI
[illegible]	C					NEWMST	TAG									START NEW MASTR
[illegible]	C						TESTZ		NEWITM					3Ø		NEW THIS YEAR
[illegible]	C					ONHAND	ADD	ONORDR	AVAIL	6	Ø					AVAILABLE
[illegible]	C					MINQTY	COMP	AVAIL					31			MIN EXCOS AVAIL
[illegible]	C					AVGCST	COMP	OLDCST					32	33		COST TREND
16	C					ONHAND	MULT	AVGCST	INVVAL							UPDATE INVVAL
17	C					TOTOHD	ADD	ONHAND	TOTOHD	8	Ø					TOTAL ON-HAND
18	C					TOTOOR	ADD	ONORDR	TOTOOR	8	Ø					TOTAL ON-ORDER
19	C					TOTVAL	ADD	INVVAL	TOTVAL	12	2					TOTAL INV VALUE
2Ø	C					END	TAG									BEGIN DTL OUTPT

Figure P6

Date Card (Card-Type Resulting Indicator 09)
(Fig. P4, lines 01 and 02)

No calculation operations are performed on this card. Indicator 93 is turned on (line 01) solely for use in a subsequent check on proper card-type sequence (line 05). The entries in line 02 cause branching to the end of the calculation specifications (page 06, line 20), so that No. 9 need not be specified in Indicators in subsequent lines.

Error Control - (Fig. P4, lines 03-18)

Calculation specifications are employed to test for certain error conditions. Where an error is recognized that affects only the individual card, calculations are bypassed for that card, and the card will be selected by output specifications to stacker 4. Where the effect pervades the entire stock-number group, all calculations for the group are bypassed from the point of error recognition, and those cards will be selected to stacker 4. For certain error situations, the system is also halted.

Indicator 90 is set on for all of the major error conditions tested for, and is used to specify the bypassing of calculations and the selection (see output specifications) of the group to stacker 4. Specifications line 03 clears indicator 90 at the beginning of each control group (i.e., stock-number group), so that the error actions do not carry through to the next group.

Indicator 91 is turned on in line 17 if indicator 19 (blank card) is on. Next program cycle, indicators 90 and H2 are turned on if indicator 91 is still on when the instructions in line 14 are reached by the program. However, if indicator L1 (first card of control group) is on when the instructions in line 07 are reached, indicator 91 is turned off.

Thus, an error is signalled (90 and H2 are turned on) if there is no control break (L1) following a blank card (91 turned on by 19). Trailer card present but not at end of group.

Duplicate Master or Sequence Step-Down

In line 08, the stock number in the old Inventory Master card is compared algebraically with that of the previous old Master card. If the number is the same (duplicate master) or lower, H1 is turned on to halt the system after the card has been processed. In line 09, the next master card is processed.

In line 10, indicator 90 is turned on if H1 was turned on in line 08, so that all processing for the remainder of the group will be bypassed, and the cards selected to stacker 4.

Note: Because the matching fields assigned in the input specifications were defined as numeric (line 03 of page 02, and line 08 of Fig. P3), the sequence check performed as a result of the M1 specification ignores sign. For that reason, the H1 indicator is also turned on for a negative comparison result--otherwise a duplicate is not detected if one card is positive and one negative in the stock-number field. However, indicators 97 and 98 also signal a negative stock number, but without a halt.

Obsolete Old Inventory Master Card

Indicator 97 turns on if the Stock No. in the old Master card is negative, signalling reentry into the operation of a previously obsoleted card. In line 13, indicator 90 is turned on if that situation exists.

Negative on-Hand in Old Inventory Master Card

Indicator 99 turns on if the On-Hand field is negative at input time of the old Master card. In line 11, indicator 90 is set on for that condition.

Cancelled Order-Item Card

Indicator 98 turns on when a transaction card with a negative stock-number field is read. This signals reentry of a previously cancelled order-item card.

Indicator 98 is used to specify bypassing of calculations for that card only (see line 15), and its selection to stacker 4, but the remainder of the group is processed normally because it is not otherwise affected.

Unmatched Transaction Cards

The specifications in line 12 cause indicator 90 to be turned on for unmatched cards (NMR), other than Inventory Master cards (N01), and other than blank trailer cards (N19) which are always unmatched.

Bypassing Calculations for the Error Group

In line 18 the program branches to END (line 20 on Fig. P6) when indicator 90 is on. This makes detail output the next operation, omitting all calculations below line 17 on Fig. P4.

Line 19 illustrates use of a Comments card (* in col. 7). It will be printed during generation as punched, but otherwise it does not enter the generation process. (It is checked for proper position, based on cols. 1-6.)

Bypassing Detail-Card Operations on Master Cards (Fig. P4, line 20 and Fig. P5, lines 01-03)

Line 20 of page 04 provides program skipping past all the specifications lines that do not apply to the new Inventory Master card (i.e., the blank trailer card). This minimizes the need for N19 specifications in Indicators in subsequent lines.

In line 01 of page 05 the average Unit Cost from the old Inventory Master card is saved for later determination of cost trend when compared with new merchandise costs.

In line 02, all calculations are terminated for old Inventory Master cards that will be replaced by new ones (i.e., there are matching transaction cards).

In line 03, the program skips--for old Master cards that are to be retained (i.e., there are no transactions)--to the same point at which calculations are resumed for new Inventory Master cards. This permits uniform preparation of report data for both situations.

Merchandise-Receipt, Stock-Adjustment, and Stock-Order Cards (lines 04-11 of Fig. P5)

In line 04, the On-Order quantity is revised to reflect Merchandise Receipts, new Purchase Stock Orders, and cancellation of Stock Orders. Cards 5 and 7 are so coded in col. 11 that addition provides the proper algebraic operation (see Figure POO). The system is halted if the operation results in a negative On-Order quantity. (Indicator 90 is not turned on, because such an error was not deemed of sufficient significance to require bypassing of the remainder of the group.)

Line 05 provides for extending the cost of a stock adjustment, based on last-known unit cost, so that the value of the inventory may be adjusted (in line 07). A new work field (CSTEST) is set up for the product.

Line 06 provides for the same operation as line 05, but using the specific unit of cost at which new merchandise was received.

In line 07, the extended cost of an Adjustment or Merchandise Receipt is algebraically subtracted from the total Inventory Value of the stock item. The signs in cards 5 and 6 are appropriately coded (see Card Layout).

In line 08, the On-Hand quantity is updated to reflect Receipts and Adjustments. Indicator 90 is turned on if On-Hand has become negative; further calculations are then by-passed for that stock-number group (by entry in line 09), and the cards from this point on are selected to stacker 4 (output specifications).

In line 10, a new Average Unit Cost is established during processing of Receipt cards, because each of these cards contain unit cost. (In lines 06 and 07 we adjusted the Inventory Value to reflect the cost of the new Receipt proportionately.) The quotient is half-adjusted.

Division by zero is not permitted, nor meaningful. Indicator 26 (turned on in line 08 if On-Hand was greater than zero) is therefore a conditioning indicator.

Line 11 causes termination of calculations for cards 5, 6 and 7.

Order-Item and Merchandise-Return cards cannot cause On-Hand to turn negative. If On-Hand was already negative, entries in lines 08 and 09 caused branching to END. Therefore, indicator 23 turns only for a customer order-item card containing a quantity larger than the positive or zero On-Hand quantity.

Lines 13-15 are executed only to handle the insufficient-stock situation (i.e., indicator 23 is on). In accordance with our Basic Assumptions:

1. No order-item will be partially filled.
2. No item card will be back-ordered if it was previously back-ordered.

3. No item will be back-ordered unless merchandise is on order.

In line 13, the quantity is added back to On-Hand to restore the prior status.

In line 14, indicator 24 is turned on if Quantity On-Order is greater than zero (COMP operation), provided the card was not previously back-ordered (N21--see Fig. P3, line 07). Indicators 23 and 24 determine, in the output specifications, whether the card is to be identified as Back-Ordered or Cancelled (Fig. P7, lines 17 and 18).

Indicator 24 is turned off each cycle (see Fig. P4, line 06) before this point is reached, because line 14 is not executed each time. Incorrect card identification in col. 1 would otherwise be punched when non-back-order cards follow a back-order card.

Line 15 causes branching to the end of the calculation specifications for order-item cards that could not be filled. The specifications in lines 02-10 of Fig. P6 will not be executed for these cases.

On Fig. P6, lines 02 and 03, respectively, set on indicator 27 if the customer-order or merchandise-return quantity is equal to or greater than the Criterion Quantity that qualifies for Price B.

We are only interested in the Resulting Indicator, not the actual result quantity. However, an arithmetic operation requires a result field. In order not to waste core storage space, a field only temporarily needed elsewhere (Fig. P5), but now available, has been utilized. A numeric Compare operation is always algebriac, therefore a more complex routine would have had to be substituted for the ADD operation in line 03 (where QTY is negative) if COMP were to be used instead.

Line 04 places Price into a new field, UNPRIC, which will be used for the unit-price factor in the selling-price extension.

In line 05, Price B is substituted for Price A in the UNPRIC field--but only provided the quantity in the Order-Item or Merchandise-Return card satisfied the criterion (lines 02 and 03) and provided Criterion Quantity was not) (N20--see Fig. P2, line 06): zero in col. 22 indicates that Price A applies in all cases.

In line 06, the quantity in the item card is multiplied by the unit price previously selected (lines 02-05). The new field, EXTPRI, will be negative for a Merchandise-Return card, because quantity is negative.

In line 07, cost of the item sale or return is calculated, using the Average Unit Cost as updated during processing of any stock Receipt cards (Fig. P5, line 10). Again, the same work field CSTEXT, is utilized, because the product is not needed beyond line 08.

In line 08, gross profit is calculated for each item card. For merchandise returns, the sign is automatically reversed: -EXTPRI-(-) CSTEXT = -GRSPRO (unless selling price is less than Average Unit Cost).

In line 09, Quantity Sold This Year is updated for this item card. Returns reduce the value, because quantity in these cards is negative. Because it is possible for returns early in a year to exceed sales, provision is made for a negative total (Fig. P11, line 17, edit word).

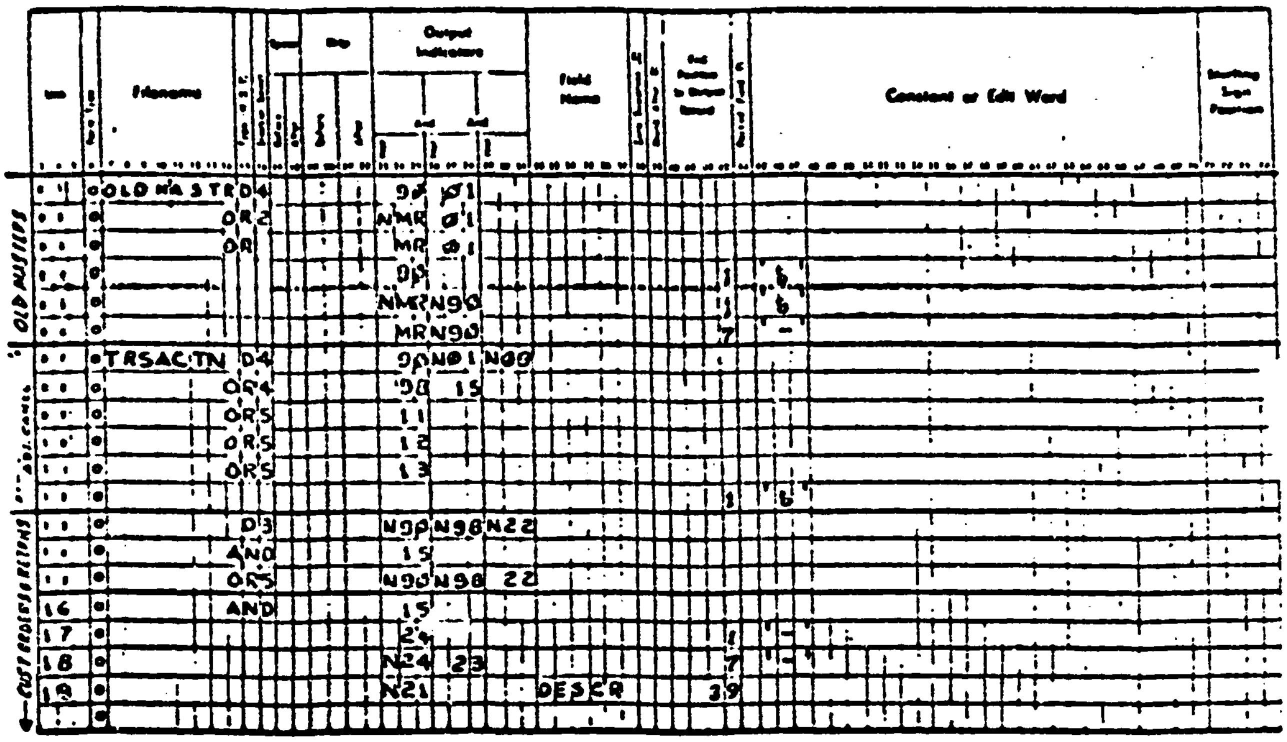
Line 10 terminates calculations for card 9.

New Inventory Totals (lines 11-19, Fig. P6)

This section contains the specifications for completing the data needed (1) to punch the new inventory Master cards for stock numbers with transactions, and (2) to print the Inventory Status Report for all stock numbers.

No conditioning indicators are required because the program has been instructed, in earlier lines, to branch past this section--to END (line 20)--for all card types except blank trailer cards or unmatched (i.e., no transaction) old Master cards.

Output Specifications

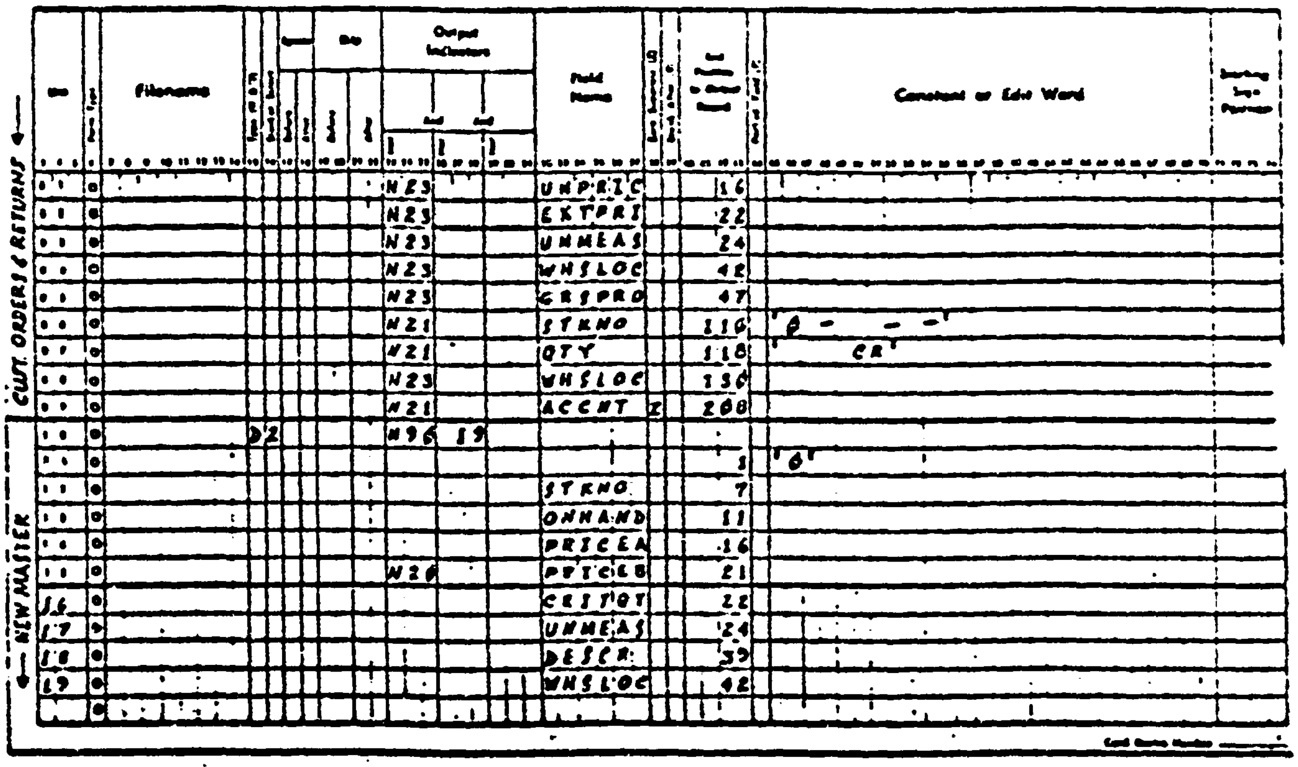


Line	Form Type	Filename	Type	Output Indicators	And	And	Field Name	End Position in Output Record	Constant or Edit Word
	O	OLD MASTR	D4	9Ø	Ø1				
	O		OR 2	NMR	Ø1				
	O		OR	MR	Ø1				
	O			9Ø				1	b
	O			NMR	N9Ø			1	b
	O			MR	N9Ø			7	-
	O	TRSACTN	D4	9Ø	NØ1	N9Ø			
	O		OR 4	9Ø	15				
	O		OR 5	11					
	O		OR 5	12					
	O		OR 5	13					
	O							1	b
	O		D 3	N9Ø	N98	N22			
	O		AND	15					
	O		OR 5	N9Ø	N98	22			
16	O		AND	15					
17	O			24				1	-
18	O			N24	23			7	-
19	O			N21			DESCR	39	

OLD MASTERS
ADJ. CARDS
CUST ORDERS & RETURNS

Figure P7

Pre-Billing - With Inventory Control



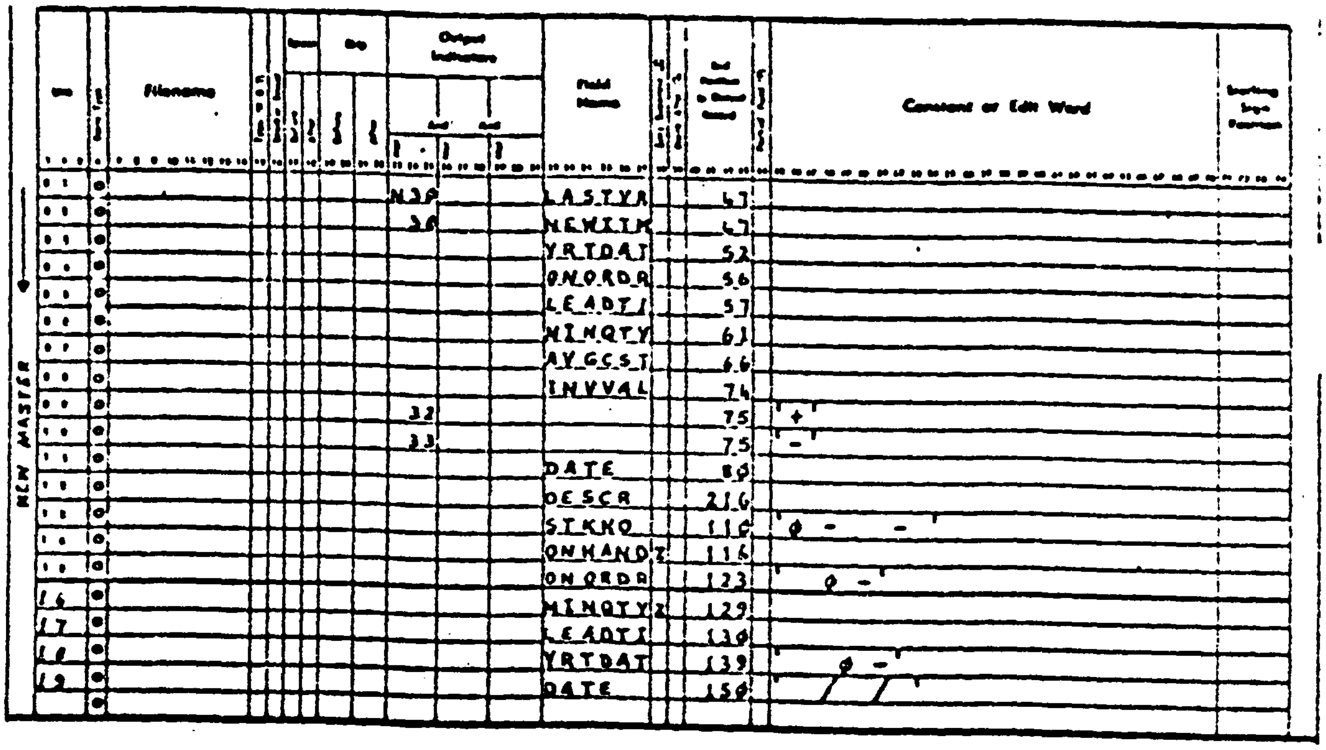
CUST. ORDERS & RETURNS

NEW MASTER

		Filename		Output Indicators			Field Name		End Position		Constant or Edit Word
	O			N23			UNPRIC		16		
	O			N23			EXTPRI		22		
	O			N23			UNMEAS		24		
	O			N23			WHSLOC		42		
	O			N23			GRSPRO		47		
	O			N21			STKNO		116		'0 - - -'
	O			N21			QTY		118		' CR'
	O			N23			WHSLOC		136		
	O			N21			ACCNT	Z	208		
	O		D2	N96	19						
	O								1		'6'
	O						STKNO		7		
	O						ONHAND		11		
	O						PRICEA		16		
	O			N26			PRICEB		21		
16	O						CRITQT		22		
17	O						UNMEAS		24		
18	O						DESCR		39		
19	O						WHSLOC		42		
	O										

Figure P8

Output Specifications

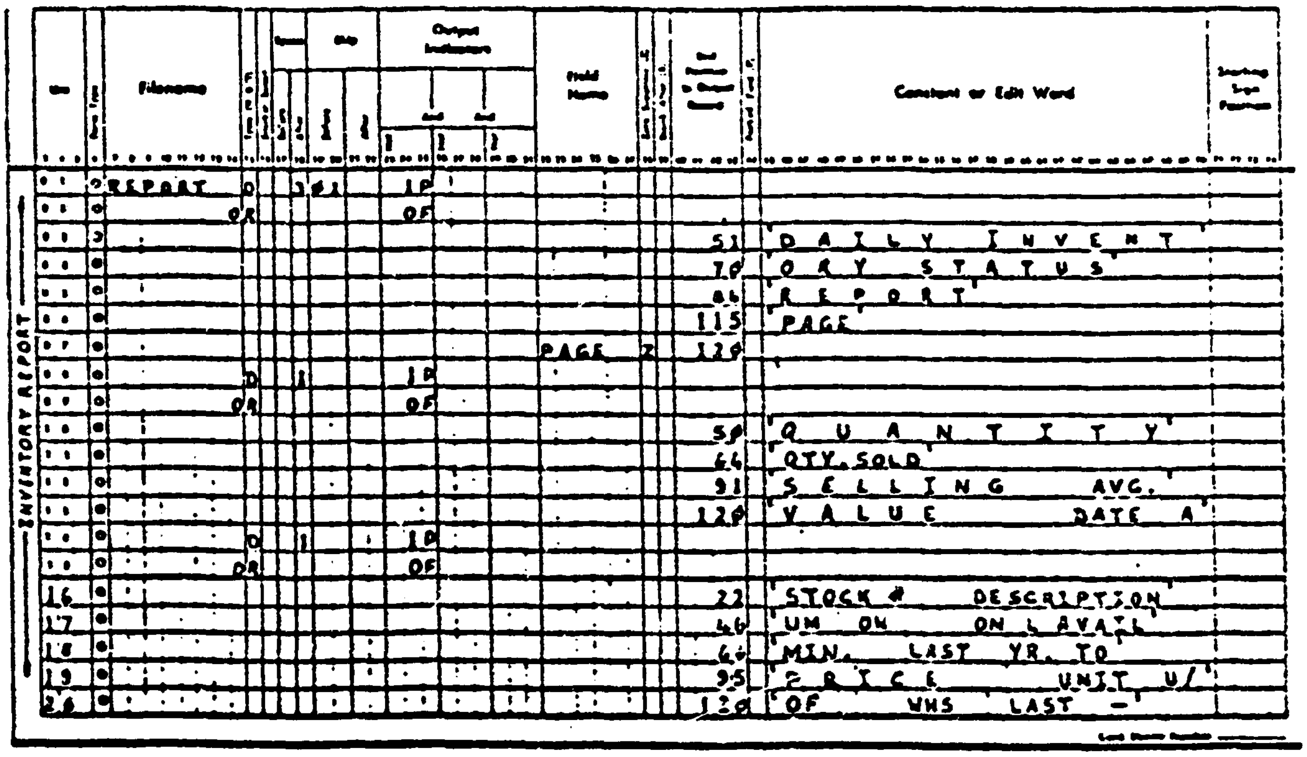


NEW MASTER

		Filename		Output Indicators			Field Name		End Position		Constant or Edit Word
	O			N30			LASTYR		47		
	O			30			NEWITM		47		
	O						YRTDAT		52		
	O						ONORDR		56		
	O						LEADTI		57		
	O						MINQTY		61		
	O						AVGCST		66		
	O						INVVAL		74		
	O			32					75		'+'
	O			33					75		'-'
	O						DATE		89		
	O						DESCR		216		
	O						STKNO		116		'0 - -'
	O						ONHAND	Z	116		
	O						ONORDR		123		' 0 -'
16	O						MINQTY	Z	129		
17	O						LEADTI		130		
18	O						YRTDAT		139		' 0 -'
19	O						DATE		156		' / / '
	O										

Figure P9

Pre-Billing with Inventory Control

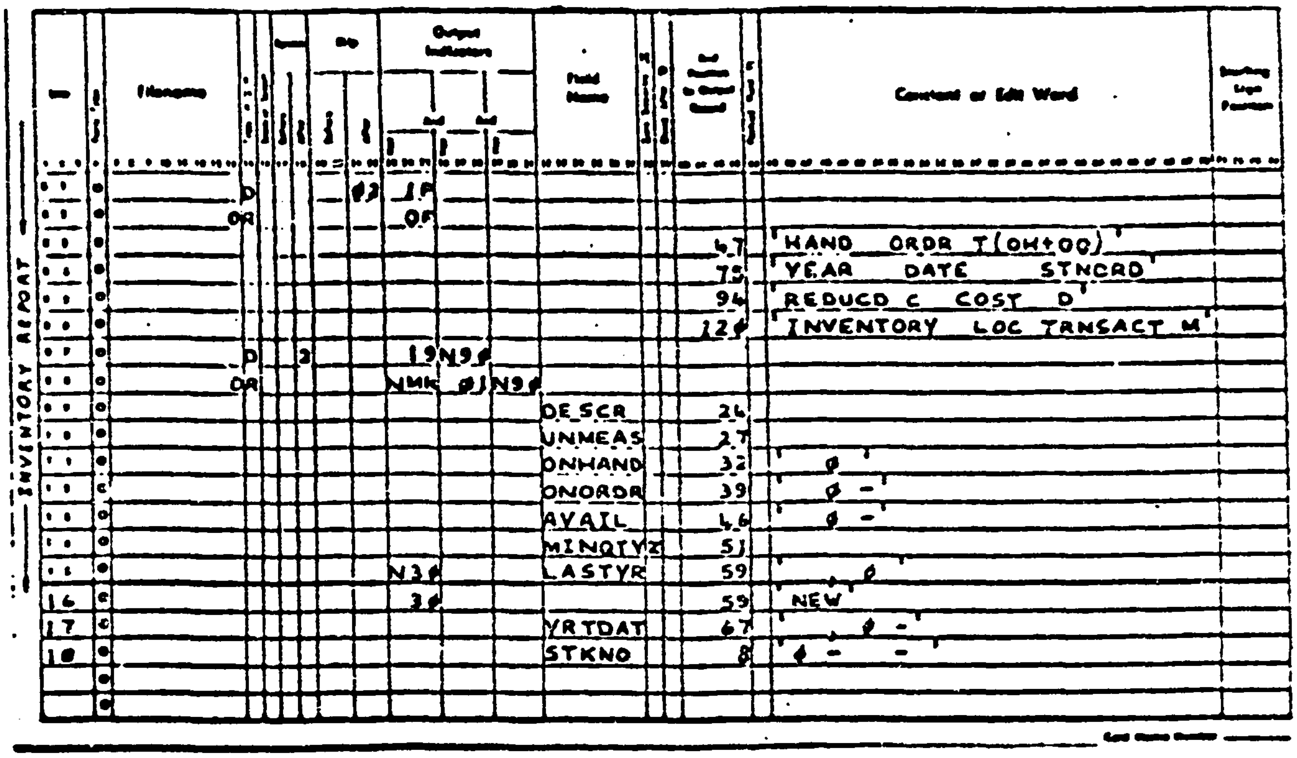


INVENTORY REPORT

Line		Filename		Space	Skip	Output Indicators	Field Name		End Position in Output Record	Constant or Edit Word
	O	REPORT	D	3	Ø1	1P				
	O		OR			OF				
	O								51	'DAILY INVENT'
	O								7Ø	'ORY STATUS'
	O								86	'REPORT'
	O								115	'PAGE'
	O						PAGE	Z	12Ø	
	O		D	1		1P				
	O		OR			OF				
	O								5Ø	'QUANTITY'
	O								64	'QTY.SOLD'
	O								91	'SELLING AVG.'
	O								12Ø	'VALUE DATE A'
	O		D	1		1P				
	O		OR			OF				
16	O								22	'STOCK # DESCRIPTION'
17	O								46	'UM ON ON L AVAIL'
18	O								64	'MIN. LAST YR. TO'
19	O								95	'PRICE UNIT U/'
2Ø	O								12Ø	'OF WHS LAST -'

Figure P10

Output Specifications

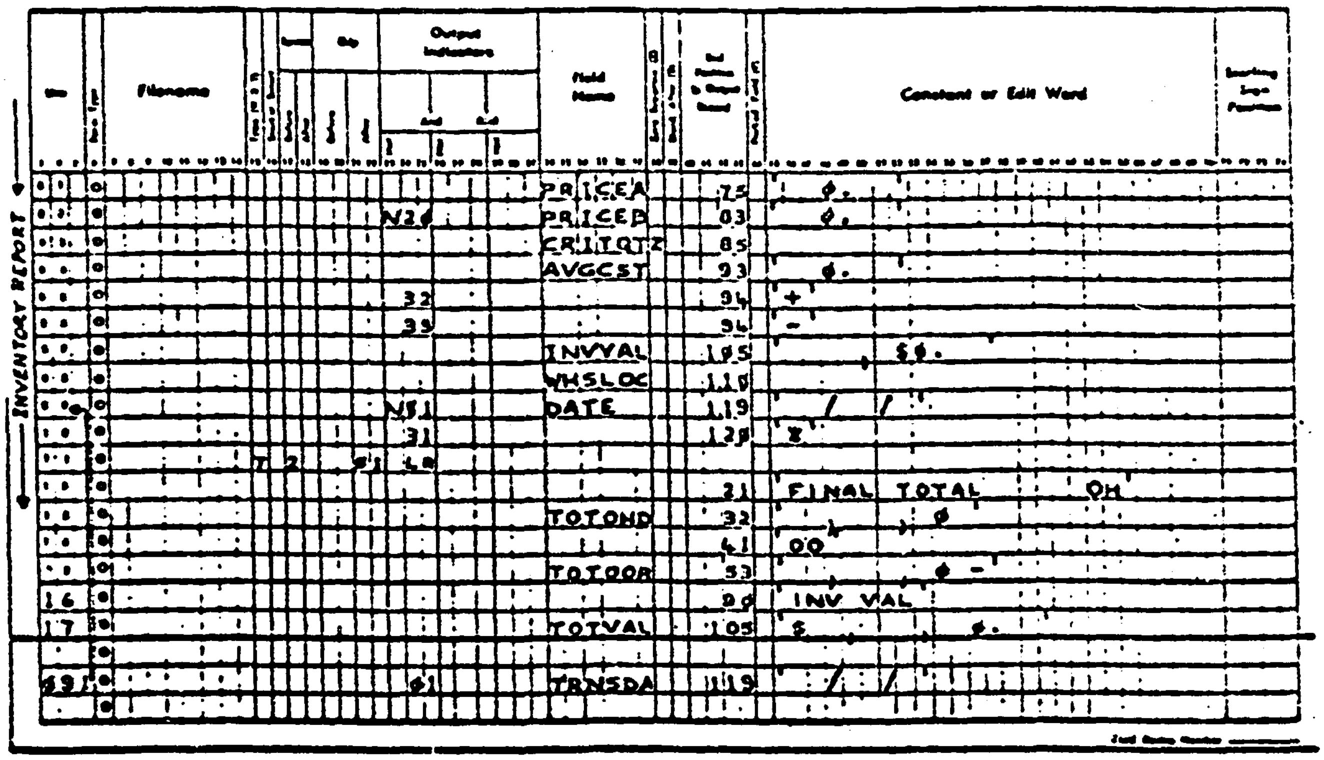


INVENTORY REPORT

Line		Filename		Space	Skip	Output Indicators	Field Name		End Position in Output Record	Constant or Edit Word
	O		D		Ø2	1P				
	O		OR			OF				
	O								47	'HAND ORDR T(OH+OO)'
	O								75	'YEAR DATE STNCRD'
	O								94	'REDUCD C COST D'
	O								12Ø	'INVENTORY LOC TRNSACT M'
	O		D	2		19N9Ø				
	O		OR			NMK Ø1N9Ø				
	O						DESCR		24	
	O						UNMEAS		27	
	O						ONHAND		32	' Ø'
	O						ONORDR		39	' Ø -'
	O						AVAIL		46	' Ø -'
	O						MINQTY	Z	51	
	O					N3Ø	LASTYR		59	' , Ø'
16	O					3Ø			59	'NEW'
17	O						YRTDAT		67	' , Ø -'
18	O						STKNO		8	'Ø - -'
	O									
	O									

Figure P11

Pre-Billing with Inventory Control



INVENTORY REPORT

		Filename				Output Indicators	Field Name		Constant or Edit Word
	O						PRICEA	75	ɸ.
	O					N2ɸ	PRICEP	03	ɸ.
	O						CRITQTZ	85	
	O						AVGCST	93	ɸ.
	O					32		94	+
	O					33		96	-
	O						INVVAL	105	, $ɸ.
	O						WHSLOC	110	
	O					N81	DATE	119	/ /
	O					31		128	X
	O		T	2		LR			
	O							21	FINAL TOTAL OH
	O						TOTOHD	32	, , ɸ
	O							41	OO
	O						TOTOOR	53	, , ɸ -
16	O							99	INV VAL
17	O						TOTVAL	105	$, , ɸ.
	O								
ɸ3	O					ɸ1	TRNSDA	119	/ /
	O								

Line 16 is not needed when there are no transactions; but there is no harm in executing it. Although there is no change in Average Unit Cost when there are no Merchandise-Receipt cards in the group, line 15 (in conjunction with line 01, Fig. P5) provides a uniform method of determining cost trend that sets the indicators appropriately, regardless of whether there has been a Receipt.

Line 11 is the destination point to which the program branched from page 04, line 20 (blank trailer card) or page 05, line 03 (unmatched old Inventory Master card).

Line 12 provides for determining whether the item is new this year (X-punch in col. 47 of old Master card--see line 11, Fig. P2). Indicator 30 will be used in output specifications to control punching into cols. 43-47, and printing in print positions 51-59.

In line 13, the available quantity (On-Hand + On-Order) is calculated for the report.

In line 14, indicator 31 is turned on if the available quantity is less than the minimum specified in the old Master card, so that this condition can be signalled by a symbol in the report (print position 120).

In line 16, the updated Inventory Value is calculated after all transactions have been processed.

Lines 17-19 contain the specifications for summing quantity On-Hand, Quantity On-Order, and Inventory Value for report grand totals.

The first two serve only as audit trails and control totals, to balance out former totals with control totals for Receipts, Adjustments, Stock Orders, Merchandise Returns, Back-Orders, and Order-Item cards. The Inventory Value total is also important as far as a management figure.

Line 20 represents merely the destination point to which the program branched from a number of previous lines when calculations were complete. It is followed by detail-time output.

Output Format Specifications (Fig. P7)

All output is at detail time (D or H in col. 15) except for grand totals, based on LR indicator which terminates the job after total-output time.

Old-Inventory Master Cards—Oldmastr File (Fig. P7, lines 01-06)

Lines 01-03 specify different stacker selection for card types in an OR relationship. Cards with major errors (indicator 90 - see Fig. P4) are selected to stacker 4; the remainder (i.e., the bulk) are selected to stacker 2 if unmatched (NMR), or stacker 1 if matched (MR). Stacker 1--the normal stacker--need not be specified.

Thus, when a new master card will be created because there were transactions, the matched old Master is directed to pocket 1; if no new Master is created, it is directed to stacker 2, which will also receive the newly punched Masters for groups with transactions.

Indicator 01 is needed:

1. To prevent old Master cards of the following stock-number groups being passed through output operations, without being read, in the program cycles during which matched secondary cards are being processed (MR on).

2. To prevent an old Master card being passed through output operations, without being read, during the detail-time output preceding the reading of the first card at 1p time (MR is then off; thus, NMR would apply).

3. To prevent performance of this output for the Date card (during whose processing NMR applies). The Date card was specified as requiring input only, by the stacker selection designated for it in the input specifications.

Line 04 specifies that obsolete old Inventory Master cards (which are replaced by the trailer card of the matched transaction-card group) are to receive an 11-punch in col. 7. This is the safeguard against accidental reentry of these cards next time (see indicator 97: Fig. P2, line 02 and Fig. P4, line 13).

Note: Indicators in File-Identification lines of card types in an OR relationship are tested in sequence. If indicator 90 is on, line 01 is applied and therefore, N90 is not needed in the next two lines. However, in line 04, N90 is necessary, because each Field-Description is considered separately for all card types in an OR relationship.

TRANSACTION CARDS: RECEIPTS, ADJUSTMENTS, STOCK ORDERS, AND ERRORS - TRSACTN FILE (Fig. P7, Lines 07-12)

Cards of groups with major recognized errors are selected to stacker 4. A previously cancelled order-item card that was inadvertently reentered (indicator 98--see page 03, line 08) is also selected to stacker 4. Fifteen is specified in line 08 (with indicator 98) so that additional cards following a cancelled order-item card are not also selected. Indicator 98, once on, is not reset until the next transaction card other than a blank is read.

Receipt, Stock-Adjustment, and Stock-Order cards are selected to stacker 5, they could instead be directed to stacker 3 with the order-item cards and subsequently sorted apart on card No. (col. 1).

ORDER-ITEM AND MERCHANDISE-RETURN CARDS-TRSACTN FILE (Fig. P7, Lines 13-19 and Fig. P8, Lines 01-09)

By the entries in lines 13-16, Merchandise-Return cards (indicator 22 on-see Fig. P3, line 09) are directed to stacker 5, whereas order-item cards are selected to stacker 3. The Returns cards could also, of course, be selected to stacker 3, and subsequently sorted apart by the X-overpunch in col. 11. The file name need not be repeated in line 13.

Line 17 provides for an 11-overpunch in col. 1 or order-item cards being back-ordered. (See Fig. P5, lines 12 and 14, for indicators.)

Line 18 specifies an 11-overpunch in col. 7 for order-items to be cancelled. (See page 05, lines 12 and 14, for indicators.)

Line 19 on page 07 and lines 01-05 on page 08 provide for punching of the pertinent data. Descriptions (line 19) are not punched the first time these cards are processed. The other fields (lines 01-05) are not punched into cards now being back-ordered or cancelled (indicator 23 on); these fields will be punched into the back-ordered cards when they are reprocessed, if the order-item is then filled.

Lines 06-09 provide for document printing (interpreting) on the Order-Item and Merchandise-Return cards, on an MFCM Model A1.

Warehouse location (line 08) is printed only if the item was filled, because the goods could be at a different location when new merchandise is received and the back-orders are filled.

The other three items are interpreted the first time the card is processed (to facilitate card handling), and are therefore not printed again on previously back-ordered cards.

Stock No., Quantity, and Warehouse Location are printed by print head 1; Account No. is printed by print head 2.

Stock No. (line 06) is edited with hyphens between digit positions two and three, and between the fifth and sixth (the presumed self-check digit). The third hyphen in the edit word is the status portion and identifies a cancelled card. All leading zeros, except the first, are preserved.

Zero Suppress is used to eliminate leading zeros in Account No. (line 09).

Note: These cards hereafter contain all the information needed to:

1. Run invoices.

2. Serve as warehouse picking tickets.

3. Run sales, cost-of-sales, and gross profit reports by stock number and merchandise class.

PUNCHING NEW MASTER CARDS (BLANK TRAILER CARDS)--
TRSACTN FILE (Fig. P8, Lines 10-19 and Fig. P9, Lines 01-11)

The pertinent fixed data from the old Master card and the updated variable information are specified for punching as per the card layout (Fig. PO).

The pertinent fixed data from the old Master card and the updated variable information are specified for punching as per the card layout (Fig. PO).

If Criterion Quantity was 0 (indicator 20 on--see Fig. P2, line 06), the field for Price (line 15) is left blank. If the item is new this year (indicator 30 is on--see Fig. P6, line 12), the single-position alphameric NEWITM field (consisting of an 11) is punched into col. 47, line 02. If the item existed last year, the five-digit numeric field LASTYR is punched into cols. 43-47 (line 01). If cost trend is up (indicator 32 is on), a plus (12/6/8) is punched into col. 75 (line 09); if it is down (indicator 33 is on), a minus (11) is punched (line 10, fig. P9); if there was no change in merchandise cost (indicators 32 and 33 are off), col. 75 is left however, 10 is only on at detail time; therefore, detail output time is the simplest way to handle the operation.)

The heading consists of constants, with one exception, the output field PAGE is specified. This is the only field name (as contrasted with constants) that can provide other than blank or zeros before the first card has been read (i.e., when 1P is on). Page No. 1 will be printed in the first heading line of the first page (it is not possible to start with any other value before a card has been read); it will be incremented by 1 before printing on the first line of each succeeding page. Zero Suppress is specified to eliminate leading zeros.

Lines 08-13 contain the specifications for the first print line of column headings, 3 lines beneath the report heading. The form is single-spaced after printing.

The column headings, too, are to appear on each page (first and overflow pages). The file name need not be repeated.

Lines 14-20 contain the specifications for the second print-line of column headings. A single space follows printing.

Lines 01-06 on Fig. P11 take care of the third print line of column headings. After printing, the form is advanced to the next channel -2 punch.

PRINTING THE ITEM LINES-REPORT (Fig. P11, Lines 07-18 and Fig. P12, Lines 01-10)

Lines 07 and 08 specify that the data is to be printed at detail-time (D in col. 15) while ,rocessing either:

1. A former blank trailer card (indicator 19 on) that does

not belong to a recognized error group (N90--see Fig. P4; and Fig. 05, line 08); or

2. An unmatched (NMR) old Inventory Master card (indicator 01 on) that does not belong to a recognized error group (N90).

Thus one line will be printed per stock number, showing the original old Inventory Master card data for items without new transactions (NMR), and the updated information where transaction cards exist.

Points to note:

1. In lines 11, 12, 13, 15, and 17 on page 11, the edit word is designed so that one 0 is printed when the quantity is completely zero, and a minus sign is printed for negative values in fields that can be negative. In lines 01, 02, 04 and 07 on Fig. P12, the edit word provides for printing of .00 when the amount field is all-zero. The edit word in line 07 of Fig. P12 also provides for a floating dollar sign.

2. The maximum number of leading zeros (i.e., all but one) is preserved for the Stock No., in line 18 on page 11, and hyphens separate merchandise class from the remainder of the number, and the principal number from the self-check digit.

3. The dates-lines 09 and 091 or Fig. P12 are edited to be printed with slashes between Month, Day, and Year. There is no point in placing a 0 in the edit word; the date can at most have one leading zero (months 01-09), and its suppression cannot be prevented by an edit-word entry.

4. Line 091 on Fig. P12 illustrates insertion of a specifications

line that had been forgotten initially, by assigning it a line number sequentially between two pre-printed numbers.

5. Lines 15 and 16 on page 11 cause the Quantity Sold Last Year to be printed (in print positions 54-59) if indicator 30 (see page 6, line 12) is off, but the word NEW to be printed instead (in print positions 57-59) if indicator 30 is on (i.e., new item this year).

6. Lines 05 and 06 on Fig. P12 provide for printing a + symbol if the cost trend is up, a - symbol if it is down, and leaving the print position blank if there has been no change in cost since the previous report. (See page 06, line 15, for setting of indicators 32 and 33.)

7. Lines 09 and 091 on Fig. P12 determines whether today's date (DATE) from the Date card or the date (TRNSDA) from the old Inventory Master card is to be printed. If there were transactions i.e., the report data is not printed while a Master card is being processed--N01), DATE is selected; if there were no transactions (i.e., the report is based on data in the old Inventory Master card-indicator 01), TRNSDA is selected.

8. Line 10 on Fig. P12 provides for printing an asterisk in print position 120 when Quantity Available (i.e., On-Hand + On-Order) is less than the Minimum Stock Quantity (see Fig. P6, lines 13 and 14, for setting of indicator 31).

PRINTING THE GRAND TOTALS - REPORT FILE
(Fig. P12, Lines 21-17)

The line is printed at total-output time, (T in col. 15), after the last data card has been processed (LR indicator on). It must be at total-output time, because the job is thereafter terminated if the LR indicator is on. The form is upspaced 2 lines before printing, providing 3 blank lines between the last detail line and the grand totals.

The form is advanced to channel 1 afterwards. Constants describing the fields are printed preceding the values. A fixed dollar sign is used in the edit word for Inventory Values.

INVOICING

This report utilizes the order-item cards processed in the previous program example (Pre-Billing with Inventory Control), in conjunction with sold-to and ship-to name-and-address cards. The same mail-order company is assumed, with modifications to illustrate more features.

The example is deliberately kept fairly simple, its main purpose being to provide an illustration of:

1. Printing sold-to name and address side by side, each of three lines, and each from a single card
2. Predetermined total line
3. Summary punching. The summary cards can be used for:
   a. Accounts receivable
   b. Sales report by customer
   c. Sales report by salesman
4. Card-type sequence check by sequence entry (cols. 15-16, input specifications)
5. Table lookup

ASSUMPTIONS

1. The item cards from the preceding example serve as detail cards (customer order-item cards), excluding Merchandise-return cards with 11 -overpunch in col. 11). They are assumed to have been sorted by Warehouse Location and Account No. after the Pre-Billing operation.

2. The heading and detail cards have been previously match-merged so that there are no missing masters or legitimate missing details. (This match-merging could have been done on a collator, or on an MFCM.)

The card with today's date and the starting invoice number (less 1) is placed ahead of this group of cards. The deck is placed in the primary hopper of the MFCM.

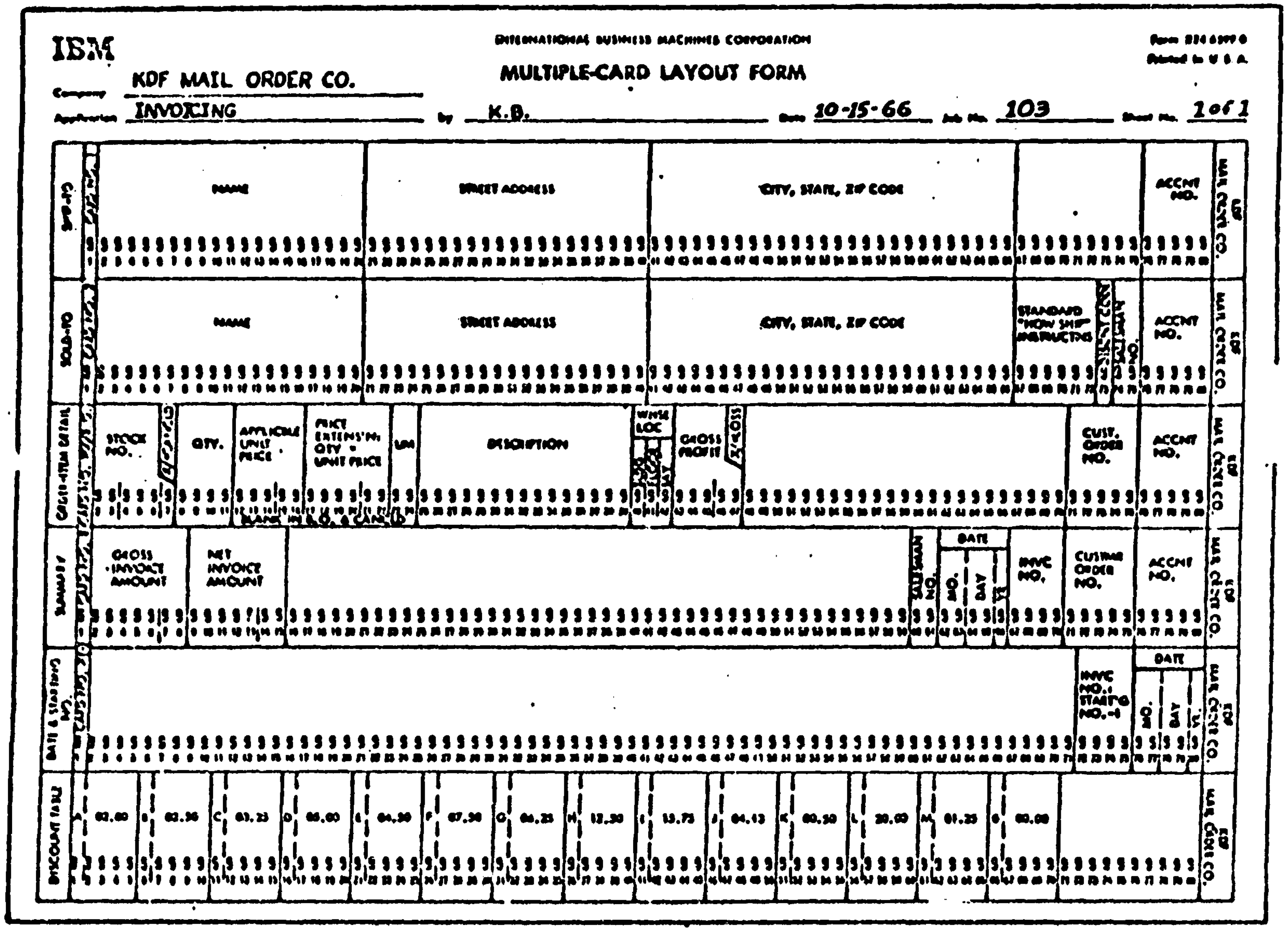
3. Name and address are confined to three lines from a single card. The presence of a ship-to card is optional. When it is present, it precedes the sold-to card; when there is no ship-to card, the sold-to name and address are to be printed in both positions.

Placing the optional ship-to card ahead improves throughput and printing of name and address can proceed during processing of the sold-to card. If the sold-to card were placed first, printing of name and address could not be commenced, when there is no ship-to card, until the first detail card is being processed; only then can the program know that no further Name-Address card (namely, ship-to card) must be awaited.

4. The blank cards, which are to become summary cards, are a separate file in the secondary hopper of the MFCM.

5. Arbitrarily, the MFCM is used for the two files; other

Card Layout



IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

MULTIPLE-CARD LAYOUT FORM

Company KDF MAIL ORDER CO.

Application INVOICING by K.B. Date 10-15-66 Job No. 103 Sheet No. 1 of 1

Figure 10

Card Flow

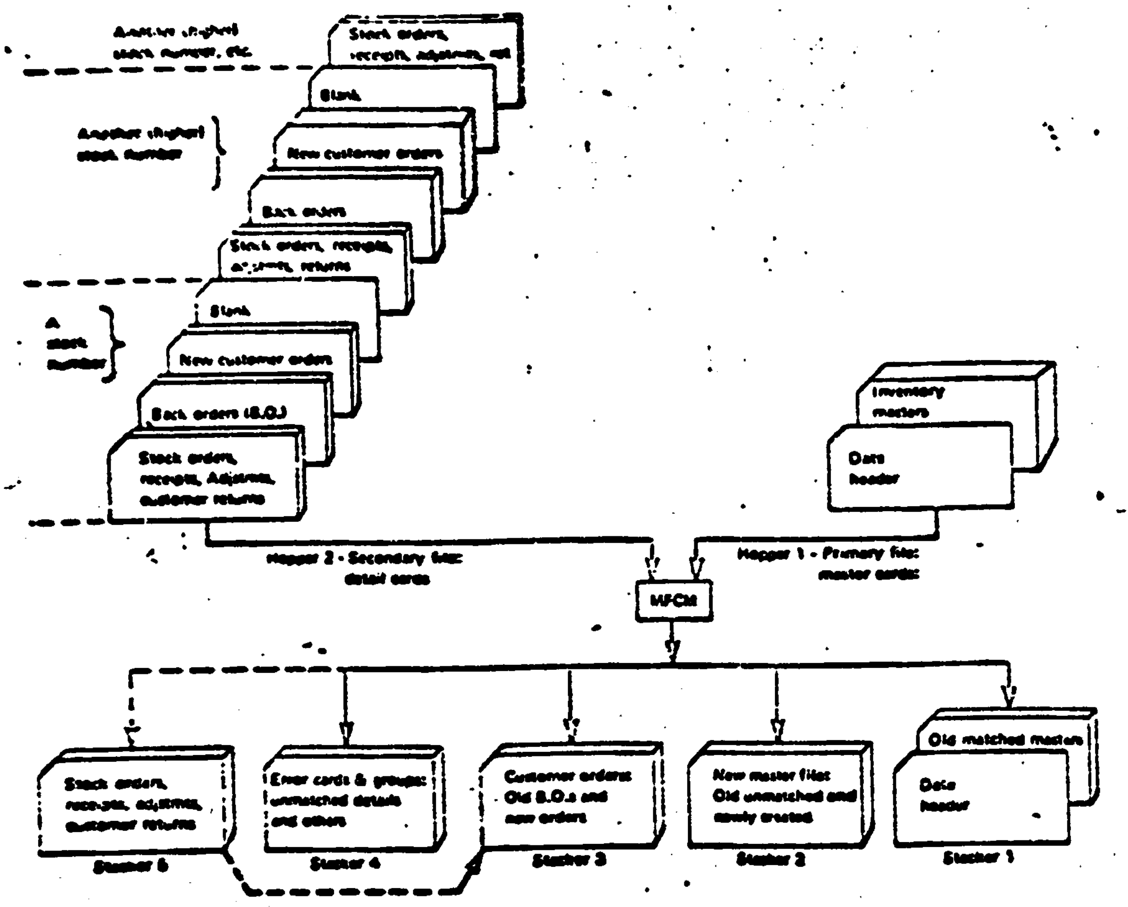
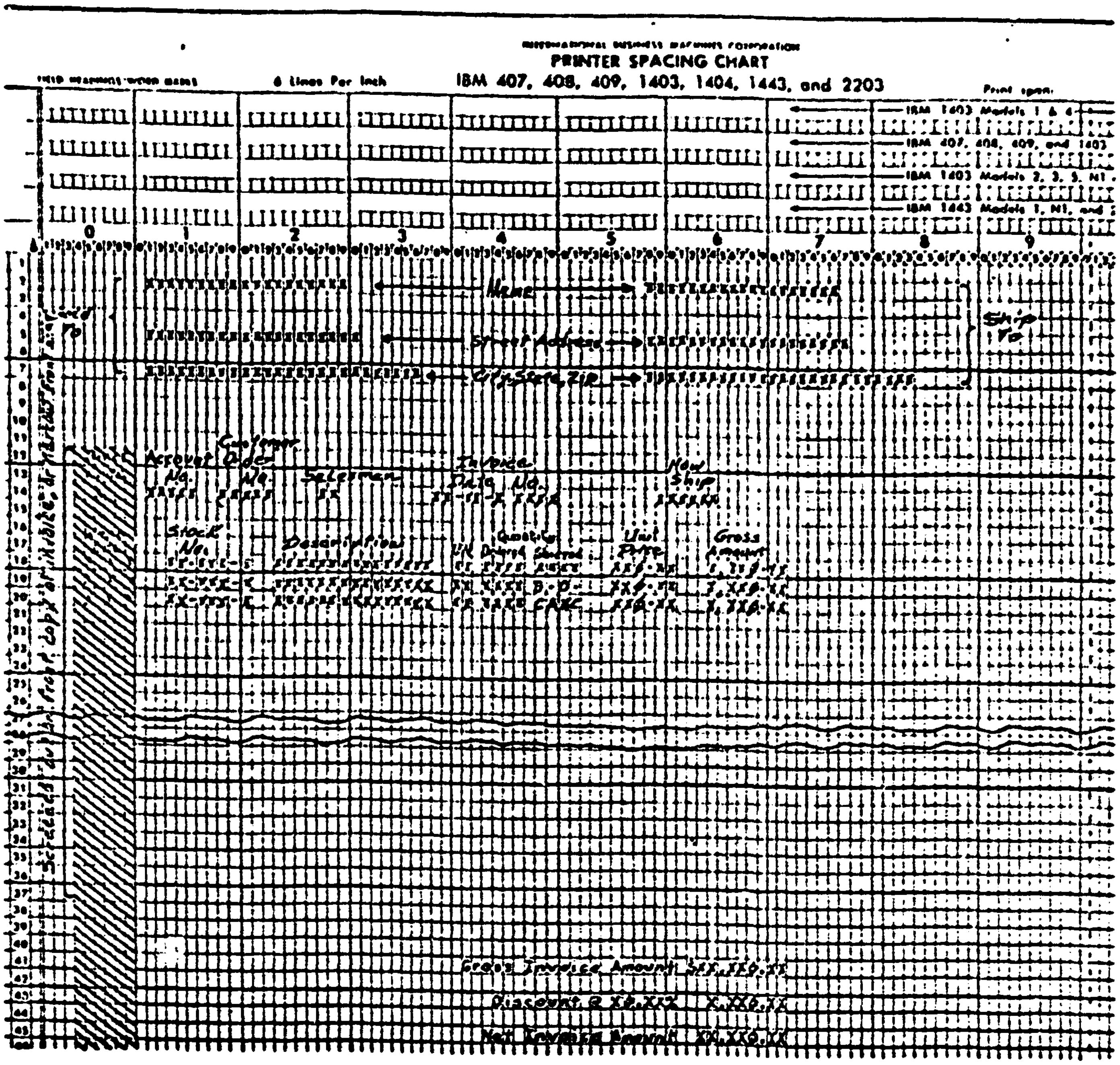

Another (higher) stock number, etc.
Stock orders, receipts, adjstmts, etc.
Blank
Another (higher) stock number
New customer orders
Back orders
Stock orders, receipts, adjstmts, returns
Blank
A stock number
New customer orders
Back orders (B.O.)
Stock orders, receipts, Adjstmts, customer returns
Inventory masters
Date header
Hopper 2 - Secondary file: detail cards
Hopper 1 - Primary file: master cards
MFCM
Stock orders, receipts, adjstmts, customer returns
Stacker 5
Error cards & groups: unmatched details and others
Stacker 4
Customer orders: Old B.O.s and new orders
Stacker 3
New master file: Old unmatched and newly created
Stacker 2
Old matched masters
Date header
Stacker 1

Figure 100

Invoice Layout



INTERNATIONAL BUSINESS MACHINES CORPORATION

PRINTER SPACING CHART

FIELD HEADINGS-WORD MARKS | 6 Lines Per Inch | IBM 407, 408, 409, 1403, 1404, 1443, and 2203 | Print span:

IBM 1403 Models 1 & 4

IBM 407, 408, 409, and 1403

IBM 1403 Models 2, 3, 5, N1

IBM 1443 Models 1, N1, and

Sold To

Name

Street Address

City, State, Zip

Ship To

Account No. XXXXX

Customer Order No. XXXXX

Salesman XX

Invoice Date XX-XX-XX

Invoice No. XXXX

How Ship XXXXXX

Stock No. | Description | UN | Quantity Ordered | Quantity Shipped | Unit Price | Gross Amount

XX-XXX-X | XXXXXXXXXXXXXXXXXXX | XX | XXXX | B.O. | XXX.XX | X,XXX.XX

XX-XXX-X | XXXXXXXXXXXXXXXXXXX | XX | XXXX | CANC | XXX.XX | X,XXX.XX

Gross Invoice Amount XXX,XXX.XX

Discount @ XX.XXX X,XXX.XX

Net Invoice Amount XX,XXX.XX

Figure 1000

Model 20 1/0 devices can be used if the File Description Specifications are changed.

6. Stacker Selection has been arbitrarily determined thus:

Date and Invoice-No. heading card-stacker-stacker 1;

Name and address cards-stacker 2;

Detail cards-stacker 2;

Summary cards-stacker 3.

7. A discount percentage is applied to the invoice total based on a customer-type code in the sold-to card. For this, table lookup is employed.

8. Certain identifying data is repeated on overflow pages. Invoice totals are to be printed at a predetermined point on the page.

Fig. 1000 shows input and output formats. Constant headings are not printed by the program, because use of a pre-printed invoice form is usual.

In the explanations that follow for the application example, most of the obvious points will be omitted, as the reader is by this time familiar with them.

FILE DESCRIPTION SPECIFICATIONS (Fig. 11)

The input file, named INPCARDS, is associated with the primary hopper of the MFCM. It consists of one card containing the day's date and the starting invoice number (less 1) and, for each customer account number represented, contains--in this order:

One Ship-to Name-and-Address card (optional)

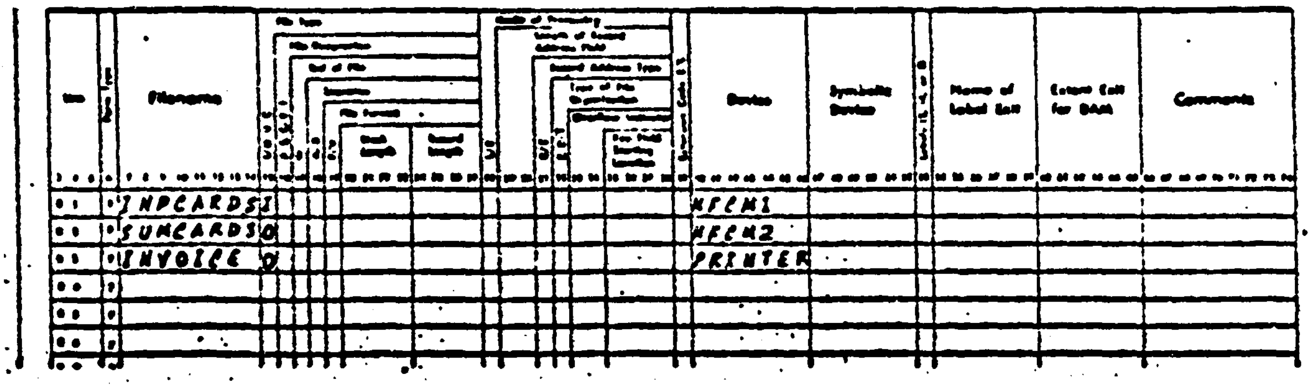
One Sold-to Name-and-Address card;

At least one Order-Item detail card.

A file of blank cards (named SUMCARDS), which will become the Invoice Summary cards, is to be placed in the secondary hopper of the MFCM.

The printer has been assigned the file named INVOICE.

File Description Specification



Filename	[illegible]	Device	Symbolic Device	Name of Label Exit	Extent Exit for DAM	Comments
INPCARDS	I	MFCM1				
SUMCARDS	O	MFCM2				
INVOICE	O	PRINTER				

Figure 11

INPUT SPECIFICATIONS (Figure 12)

There is only one input file, named INPCARDS, constituted of four card types.

Date/Invoice-No. Card (Fig. 12, Lines 01-03)

Sequence (cols. 15-16) is alphabetic, because the card appears only once, and does not fall into a sequence within each account-number group.

Stacker selection need not be specified, because 1 is the normal stacker for the primary hopper of the MFCM.

No card-type Resulting Indicator is needed; the card is never referenced, and all calculations are conditioned by indicators of the appropriate cards.

Ship-To Card

If present, this card is to precede all others of the group; therefore, it is sequence number 01 (cols. 15-16). If present, only one is permitted; therefore, 1 is specified in col. 17. Its presence is optional; therefore, and 0 in col. 18.

Control Level 1 is assigned to customer account number-both (1) to perform end-of-invoice routines, and (2) to guard against cards out of sequence, or missing Sold-To card (see calculation specifications).

Stacker 1 is the normal stacker, and need not be designated.

Sold-To Card (Fig. 12, Lines 09-16)

Exactly one card (1 in col. 17) of this type must be present (no 0 in col. 18), and it follows the Ship-To card, but lower than for the detail cards (Fig. 13, line 01).

Different field names are used for name and address in this card: the name-and-address data from the Ship-To card (if any) is to be printed alongside that from the Sold-To card.

The same field name is used for AC NTNO in all cards because the data should be the same from all cards within the group and therefore need not be saved from card to card. If it is not the same, a control break will occur (L1 is assigned to Account No.).

Indicator 20 is utilized to recognize the first detail card of each invoice (see page 06, lines 12 and 13).

Stacker 1 need not be specified.

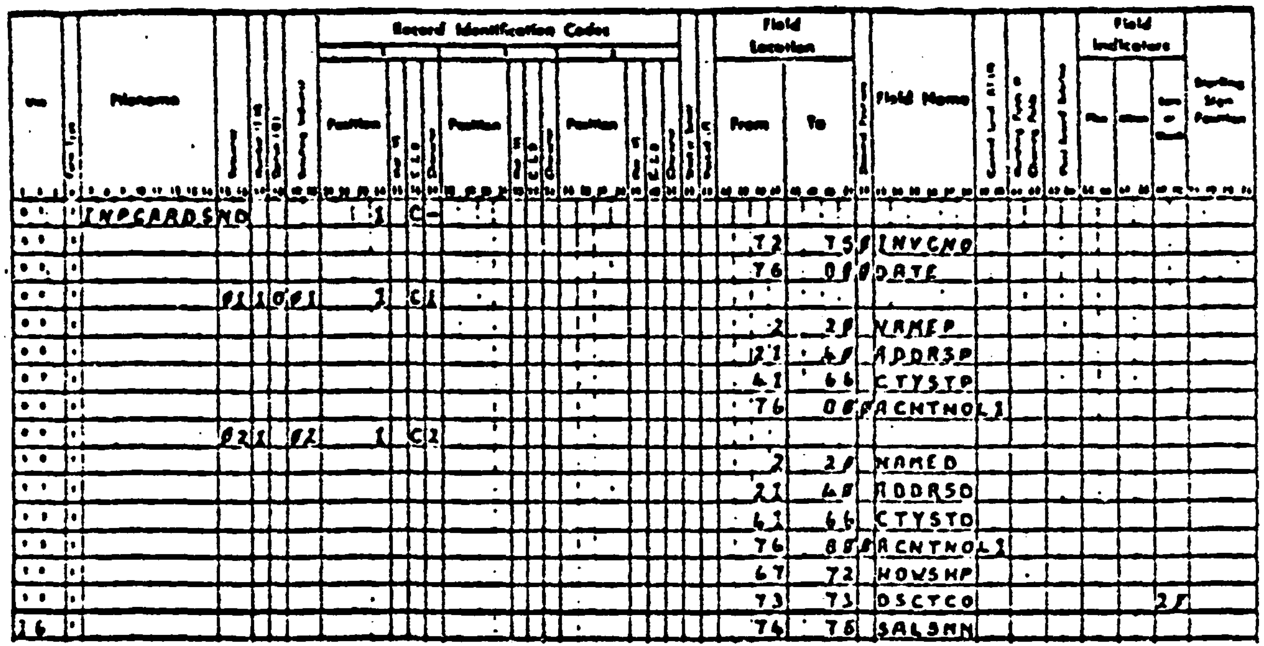
Order-Item Detail Cards (Fig. 13)

Our assumptions called for selecting these cards to stacker 2; therefore, a 2 is entered in col. 42 of line 01.

The field BOCARD in line 02 is specified only to provide an indicator (21) for recognition of back-order cards (11-overpunch in col. 1, making the field negative).

If the item card was to be cancelled, because of unavailability, an 11-overpunch was punched in col. 7 in the previous application example. This makes the Stock No. (line 03) negative. Indicator 22 is utilized subsequently to control operations for cancelled items.

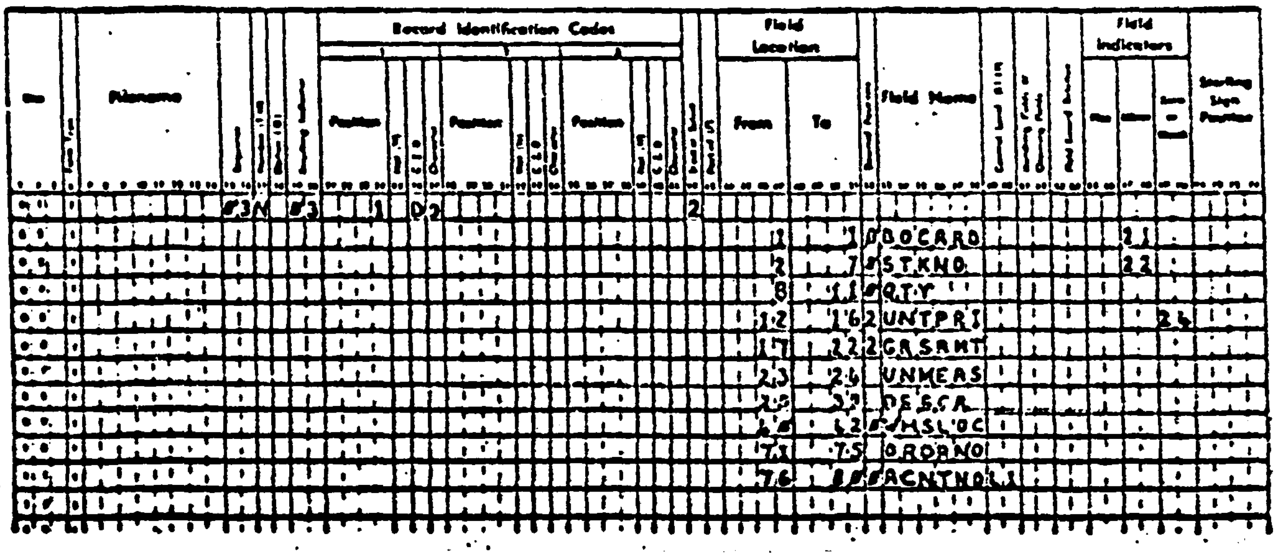
Unit Price, among other fields, was left blank in the previous operation whenever the item could not be filled. Indicator 24 is subsequently utilized to control operations for unfilled items.



						Record Identification Codes			Field Location					Field Indicators	
	Filename					Position			From	To		Field Name			Sterling Sign Position
	INPCARDS	NO				1	C	-							
									72	75	0	INVCNO			
									76	80	0	DATE			
		01	1	O	01	1	C	1							
									2	20		NAMEP			
									21	40		ADDRSP			
									41	66		CTYSTP			
									76	80	0	ACNTNO	L1		
		02	1		02	1	C	2							
									2	20		NAMED			
									21	40		ADDRSD			
									41	66		CTYSTD			
									76	80	0	ACNTNO	L1		
									67	72		HOWSHP			
									73	73		DSCTCO		25	
16									74	76		SALSMN			

Figure 12

Input Specifications



Line	Filename				Record Identification Codes Position				Field Location From	To		Field Name		Field Indicators			Sterling Sign Position
		ø3	N	ø3	1	D	2	2									
									1	1	ø	DOCARD		21			
									2	7	ø	STKNO		22			
									8	11	ø	QTY					
									12	16	2	UNTPRI				24	
									17	22	2	GRSAMT					
									23	24		UNMEAS					
									25	37		DESCR					
									38	42	ø	WHSLOC					
									71	75		ORDRNO					
									76	80	ø	ACNTNO	L1				

Figure 13

CALCULATION SPECIFICATIONS (Figure 14)

Lines 01-03 cause the Sold-To name-and-address data to be moved to the corresponding Ship-To fields whenever there was no Ship-To card (i.e. the first card of the control group is a Sold-To card). At output time, this will cause the same information to be printed in the Sold-To and Ship-To areas on the invoice.

Line 031 causes the Invoice No. to be incremented during processing of the first card of each Account No. control group. (It was loaded with a value one less than the desired starting number.)

Line 04 specifies cumulation of the gross amount from each item card for an invoice total. If the item was not filled, the GRSAMT field is blank.

Line 05 causes a search through the argument table TABCOD for a code that exactly matches the Discount Code in the permanent customer Sold-To Date-and-Address card. When a match is found, indicator 23 turns on, and the discount percentage in the equivalent position of the function table TABPRC is stored and becomes available as a calculation factor and as output-field data.

The tables are defined in File Extension Specifications--see Fig. 15.

In line 06, indicator H1 is turned on to stop the system after this card--if no Discount-Code match was achieved.

Lines 06-12 provide for the following calculations during total time following the last detail card of each invoice:

1. The invoice gross total is multiplied by the table-supplied percent of discount to establish the discount amount (line 07). (Note that half-adjustment is used, and 4 decimal positions are dropped. There are 2 decimals in INVGRS and 4 in TABPRC, since percentages less than 100 expressed as ratios fall to the right of the decimal point).

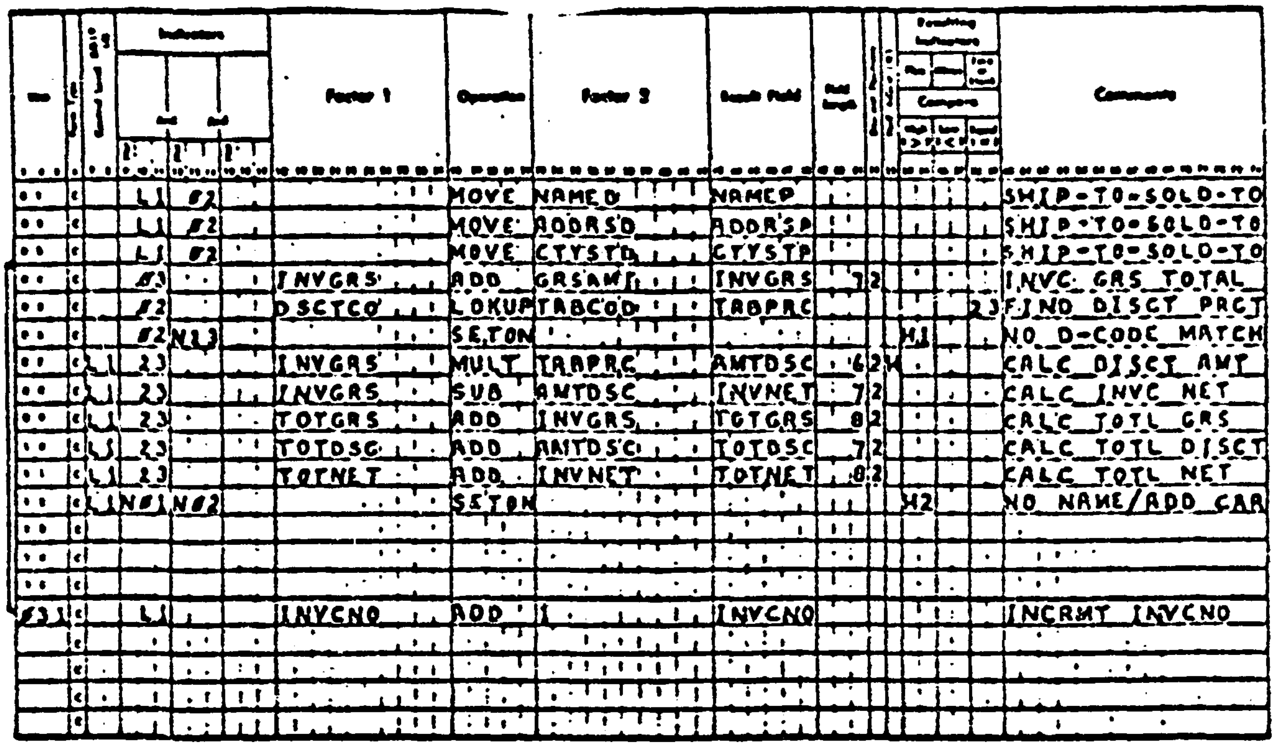
2. The discount amount is subtracted from the gross invoice amount to produce the net invoice amount (line 08).

3. The three invoice amount totals (gross, discount, net) are accumulated in three other fields, to provide grand totals (lines 09-11).

The operations in lines 07-11 are executed only when the Discount Code matched an entry in the argument table (indicator H2, and halt the system after this card, if the first card of a control group is not a Name-and-Address card (i.e., neither a Ship-To nor a Sold-To card).

Note: Since the test is made at total time (L1 in cols. 7-8), the first group will not be checked: total time is bypassed on the first card with Control Level specifications. (The test could have been programmed for detail time instead; but our approach offers the opportunity to remind the reader of the initial total-time bypass.)

Calculation Specifications



			Indicators			Factor 1	Operation	Factor 2	Result Field	Field Length			Resulting Indicators			Comments
	C		L1	Ø2			MOVE	NAMEØ	NAMEP							SHIP-TO-SOLD-TO
	C		L1	Ø2			MOVE	ADDRSD	ADDRSP							SHIP-TO-SOLD-TO
	C		L1	Ø2			MOVE	CTYSTD	CTYSTP							SHIP-TO-SOLD-TO
	C		Ø3			INVGRS	ADD	GRSAMT	INVGRS	7	2					INVC GRS TOTAL
	C		Ø2			DSCTCO	LOKUP	TABCOD	TABPRC						23	FIND DISCT PRCT
	C		Ø2	N23			SETON						H1			NO D-CODE MATCH
	C	L1	23			INVGRS	MULT	TABPRC	AMTDSC	6	2	H				CALC DISCT AMT
	C	L1	23			INVGRS	SUB	AMTDSC	INVNET	7	2					CALC INVC NET
	C	L1	23			TOTGRS	ADD	INVGRS	TOTGRS	8	2					CALC TOTL GRS
	C	L1	23			TOTDSC	ADD	AMTDSC	TOTDSC	7	2					CALC TOTL DISCT
	C	L1	23			TOTNET	ADD	INVNET	TOTNET	8	2					CALC TOTL NET
	C	L1	NØ1	NØ2			SETON						H2			NO NAME/ADD CAR
	C															
	C															
	C															
Ø31	C		L1			INVCNO	ADD	1	INVCNO							INCRMT INVCNO
	C															
	C															
	C															
	C															

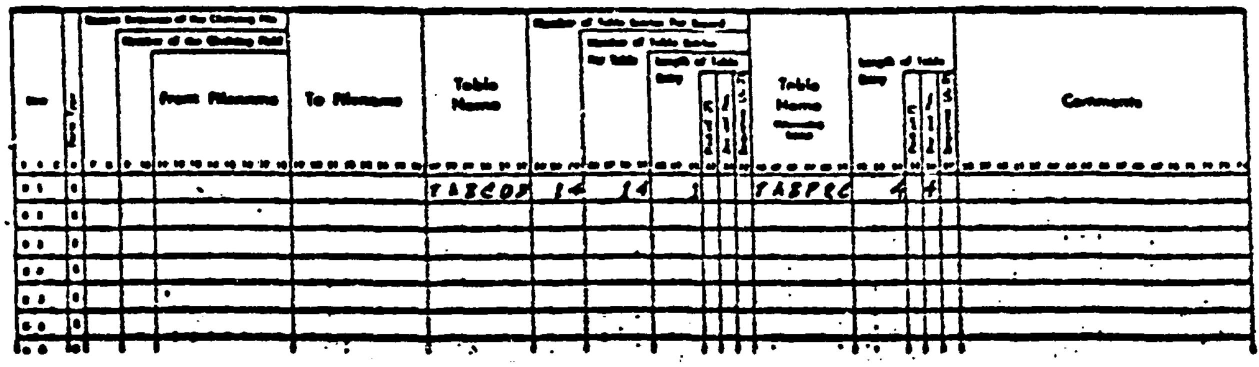
Figure 14

FILE EXTENSION SPECIFICATIONS (Figure 15)

Two tables are used in this application--one as an argument table (TABCOD) and the other as a function table (TABPRC). For convenience, the two tables are punched alternately in the same card, but this has nothing to do with the manner in which they are employed (argument or function). The table cards (in this instance, a single card) must be loaded at program-generation time.

There are only 14 codes, and all fit in one card; therefore, both the number of table entries per card and per table are the same. The code is a single character (thus, 1 in col. 42), and the percentage is 4 digits long (format XX.XXX). Since the term "percent" means "per hundred," the decimal point must be moved two positions further to the left when multiplying by a percentage; thus, the field contains 4 decimal positions (not 2).

File Extension Specifications



Line	Form Type	Record Sequence of the Chaining File	Number of the Chaining Field	From Filename	To Filename	Table Name	Number of Table Entries Per Record	Number of Table Entries Per Table	Length of Table Entry	P/B	Decimal Positions	Sequence	Table Name (Alternating Table)	Length of Table Entry	P/B	Decimal Positions	Sequence	Comments
01	E					TABCOD	14	14	1				TABPRC	4		4		
02	E																	
03	E																	
04	E																	
05	E																	
06	E																	

Figure 15

OUTPUT FORMAT SPECIFICATIONS (Figure 16)

(Refer to Card and Report Layouts.)

Note that all detail output is specified ahead of all total output.

Detail Printing on the Invoice-File (Fig. 16 and
Fig. 17, Line 01-11)

This output is performed at detail time (D or H in col. 15). INVOICE was associated with the printer (see Fig. 12, line 03).

Lines 01-04 on Fig. 16 control the printing of the first print line of the page. The Sold-To name (NAMED=Name sold), read card 2 assigned card-type Resulting indicator 02, is printed in positions 11-29; the Ship-To name (NAMEP-Name ship), read from card 01 (indicator 01) is printed in positions 58-76. Both are printed in the same line on the invoice form.

The printing at the beginning of each Account-No. group takes place as the Sold-To card is being processed (indicator 02 on); at that time, both the ship-to and sold-to information is available, and can be printed concurrently (if L1, instead of 02 were specified, only data from the first card of the group would be available).

The names are also repeated at the top of overflow pages, at overflow-output time (indicator OF). NLI is specified, so that the old names are not printed at overflow time at the top of one new page--followed by the new names on the next page from card type 02--when the overflow point and the end of a group coincide.

In the calculation specifications (Fig. 14, line 01), the Sold-To name was moved into the Ship-To name field if there was no Ship-To card; therefore, both names are the same in that case.

Note: If the Ship-To area on the invoice is to be left blank when there is no Ship-To card (rather than repeating the Sold-To card information), lines 01-03 on page 04 (calculation specifications) would be omitted. However, a B must then be placed in col. 39 (Blank After) of lines 04, 07, and 10 of page 06; otherwise, whenever there is no Ship-To card in a group, the data from the last preceding Ship-To card remains in storage, and will be printed.

Lines 05-07 and 08-10 provide the equivalent functions for the second and third lines of the addresses. However, the street addresses and city/state are not repeated on overflow pages.

No entry is required in cols. 17-22 of line 08, because spacing to the miscellaneous-data print line is specified in line 11. The 0 in col. 18 is entered only for compatibility with other RPG's (any entry would satisfy that requirement).

Lines 11-12 (and, as explained below, line 13) control the conditions under which the miscellaneous data is printed above the first detail line on the invoice.

The form is skipped to the next channel-2 punch before the miscellaneous-data line is printed, and to the next channel-3 punch (first detail line) thereafter.

Note: Instead of utilizing Skip/Before in specification line 11 to reach the miscellaneous-data print line (the simplest way to program this), Skip/After-which is usually more efficient in terms of throughput-can be used in the name-and-address specifications lines. It requires several entries, however, because all three name-and-address lines are printed at the start of a new customer group, but only the name line is printed on overflow pages. The entries in cols. 17-22 (forms control) of specifications (lines 01, 02, 08, and 11) should then read:

Line 01-ƀƀ 01 02
Line 02-ƀ3 01 ƀƀ
Line 08-ƀƀ ƀƀ 02
Line 11-bb bb 03

The miscellaneous-data line is printed after the name and address for a new group, and ahead of the first detail line. It is also printed in the same position on overflow pages (when overflow does not coincide with the end of a group); but some of the fields are not printed (NOF) on overflow pages.

Because Customer Order No. (ORDRNO) is not available until the first detail item card has been read, the miscellaneous-data line must be printed after the first detail item card has been read, yet above the regular detail data. Therefore, it is printed during processing of a detail card (indicator 03 in specifications line 12), yet before the print line for the regular detail data (see Fig. 17, lines 01-11). It is to be printed only before the first detail line (apart from overflow identification specified in line 11); therefore, the first detail card of a group must be identified. We chose to accomplish this as follows: The data for the field DSCTCO is supplied by the Sold-To card, where it is never blank. When the first detail card is processed, the DSCTCO field, therefore, contains data. (One of the possible Discount Codes in this example is 0-see Discount Table in Figure 71-but 0 is treated as non-blank in an alphameric field. DSCTCO was defined as alphameric--see Fig. 12, line 15). Indicator 20 is on only when DSCTCO is blank (see page 02, line 15); it is therefore off when the first detail card is processed.

Specifying N20 with 03 in line 12 permits the output to be performed for the first detail card, because indicator 20 is off. As the data from the DSCTCO field is transferred to the output-storage area, the Blank-After (B in col. 39) instruction causes the field to be cleared, and indicator 20 to turn on. The output controlled by the specifications in line 12 will thus never be performed again until another Sold-To card has preceded a detail card because indicator 20 remains on until data is read into the DSCTCO field again. (The entries in line 11 provide for the output at overflow time.) The field DSCTCO was chosen because its data is not needed again in the remainder of the operations for a group.

Note: An alternate approach would be:

Change all L1 specifications to L2.

Then, specify Control Level L1 for ORDRNO (Fig. 13, line 10).

In place of N20 on Fig. 16, line 12, specify L1.

The B in line 13, Fig. 16 is then not needed; nor is indicator 20 in line 15, Fig. 12 required.

This technique might be employed if the contents of all pertinent fields had to be preserved for summary punching.

Specifications lines 13-19 specify the data to be printed in the miscellaneous-data print line.

Although the field DSCTCO is not suppressed for overflow lines (no NOF entry), nothing will be printed from it, because it is blank at that time (see above).

Lines 01-11, fig. 17 contain the specifications for printing of the item detail lines.

The ampersand symbols in the edit word for WHSLOC provide blank spaces on the invoice between the three digits.

If the order item was not filled (i.e., it was back-ordered or cancelled), the Unit Price (UNTPRI) field was left blank (in the previous operation), and indicator 24 is on (see Fig. 13, line 05). Outputs of Unit Price (UNTPRI) and Gross Amount (GRSAMT) are suppressed (N24) when these fields do not apply (i.e., they are blank, with UNTPRI used as the criterion to set indicator 24). Although the fields are blank at input, blank numeric fields are converted to zeros, and .00 would be printed if the output is not suppressed.

The QTY in line 06 pertains to Quantity Ordered; in line 07, it represents Quantity Shipped, although the data is taken from the same field. The quantity in line 07 is therefore allowed to print only if the order item was filled (N24-UNTPRI field not blank)--it was part of the assumptions in the preceding application example that no partial fills would be made: either stock was sufficient to satisfy the quantity ordered, or the order item was not filled at all (it was then back-ordered or cancelled).

B.O. is printed in the Quantity-Shipped area on the invoice (see specifications line 08) if the order item was back-ordered and not cancelled: indicator 21 is on if the card is identified in col. 1 as a back-order card (see Fig. 13, line 02); indicator 22 is on if the order item was cancelled (see page 03, line 03). All three indicators (24, 21, N22) are needed to establish an active back order, because the item might have been previously back-ordered, and filled or cancelled in the most recent pre-billing pass (see preceding application example).

CANC is printed in the Quantity-Shipped area on the invoice (see specifications line 09) if the item was cancelled (indicator 22--see Fig. 13, line 03).

SUMMARY PUNCHING-SIMCARDS FILE (Fig. 17, Lines 12-20 and Fig. 18, Lines 01-05)

This output is performed at total time (T in col. 15), at the end of an Account-No. control group (L1 in output indicators, line 12), when all totals accumulated from the cards of the group are available.

The file name SUMCARDS was associated with an output file in the secondary hopper of the MFCM (see Fig. 11, line 02). The cards are directed to stacker 3.

Lines 13-[illegible] page 07 contain punch, rather than interpret, instructions, because col. 41 is blank or 0.

Lines 01-05 on page 08 contain interpreting instructions for selected fields--they are interpreting, rather than punching, specifications because col. 41 contains a print-head number (i.e., is not blank or 0).

Note 1: The interpreting feature is available only on the MFCM Model A1.

Note 2: Punching of the summary card was specified between detail and total printing to optimize throughput; generally, alternating forms printing and card punching tends to increase throughput.

TOTAL PRINTING ON THE INVOICE-INVOICE FILE
(Fig. 18, Lines 06-16)

The form is first advanced to a predetermined total line (04 in cols. 19-20, specifications line 06). Three lines of totals are then printed at total time (T in col. 15) when the L1 indicator is on (i.e., after each Account-No. group). The form is double spaced between the total lines. In specifications line 11, no entry is needed in col. 18, because forms advance before the grand-total line is specified in line 13. Zero is entered only for compatibility with other RPG's (for that purpose, any difit 0-3 is satisfactory).

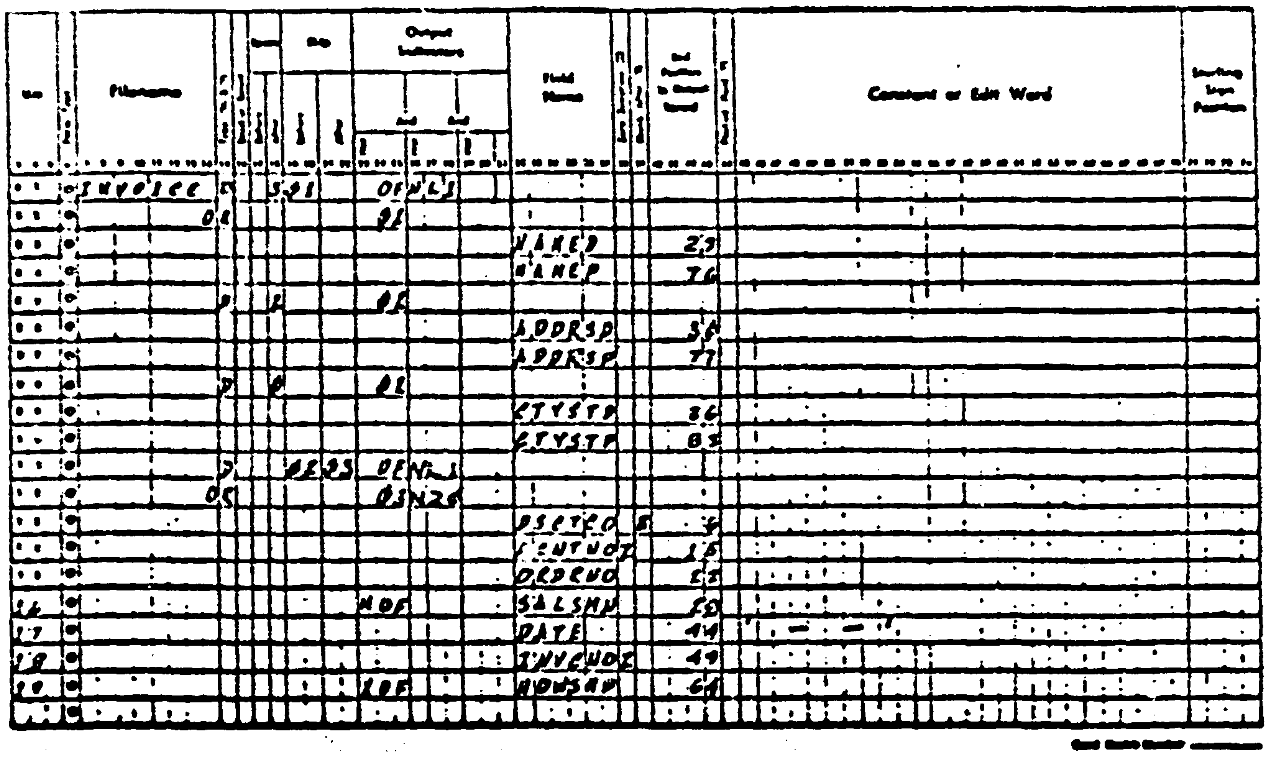
Output for the second and third total lines (see specifications lines 08 and 11) is also subject to indicator 23 being on. This suppresses the discount and net amount lines when no match on Discount Code is achieved between the code in the Sold-To card and those in the argument table. While calculation of these amounts was suppressed in such case--see page 04, lines 07 and 08--.00 (not blank) would be printed for the two amount fields (because of the format of the edit words) if output were not suppressed, and a percentage figure from an earlier LOKUP operation would be printed from TABPRC.

Whenever the total in specification line 07 is transferred to the output-storage area, the field is cleared to zero (B in col. 39) to be ready for accumulation of the total for the next group. Note that the Blank-After instruction could not be entered on page 07 (SUMCARDS) otherwise, the field would be zero before output for printing.

In line 09, note the location of the decimal point in the edit word: in the file-extension specifications TABPRC is defined as consisting of four decimal places so that decimal alignment is correct when calculating the percentage amount. When printing the figure, however, it is to appear as a percentage again--the printing of a decimal point (like any other constant) has no connection with the location of the decimal point for arithmetic operations, as specified in the field definition.

Lines 13-16 control the printing of the grand totals at the end of the report (LR indicator on). The form is advanced to a new invoice page, and all three final totals are printed on the first line.

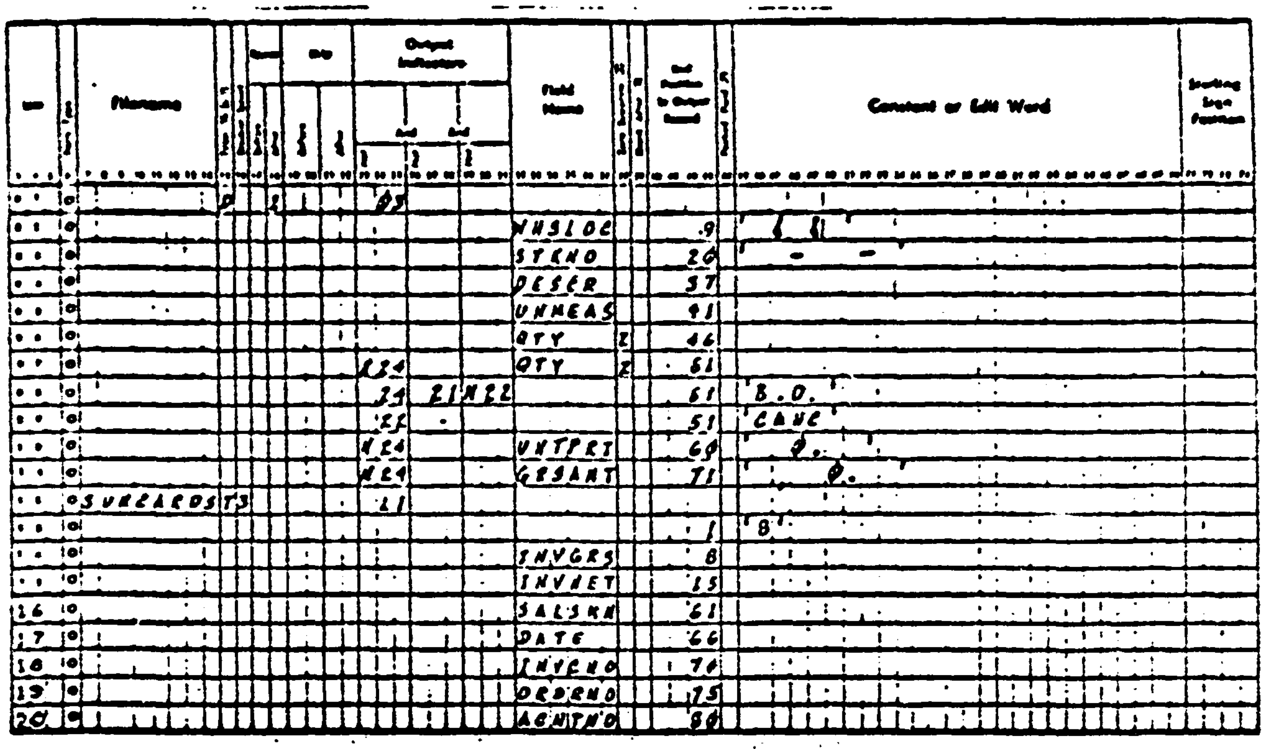
Output Specifications



Line	Filename	Type	Space	Skip	Output Indicators	Field Name	Edit Codes	Blank After	End Position in Output Record	Constant or Edit Word
	INVOICE	E	3	01	OF NL1					
	OR				01					
						NAMED			29	
						NAMEP			76	
		D	1		01					
						ADDRSD			56	
						ADDRSP			77	
		D	0		01					
						CTYSTD			36	
						CTYSTP			85	
		D	2	23	OF NL1					
	OR				03 N26					
						DSCTCD		B	4	
						CENTNO	Z		15	
						ORDRNO			55	
16					N0F	SALSMN			50	
17						DATE			44	' - - '
18						INVCNO	Z		49	
19					10F	HOWSHP			64	

Figure P6

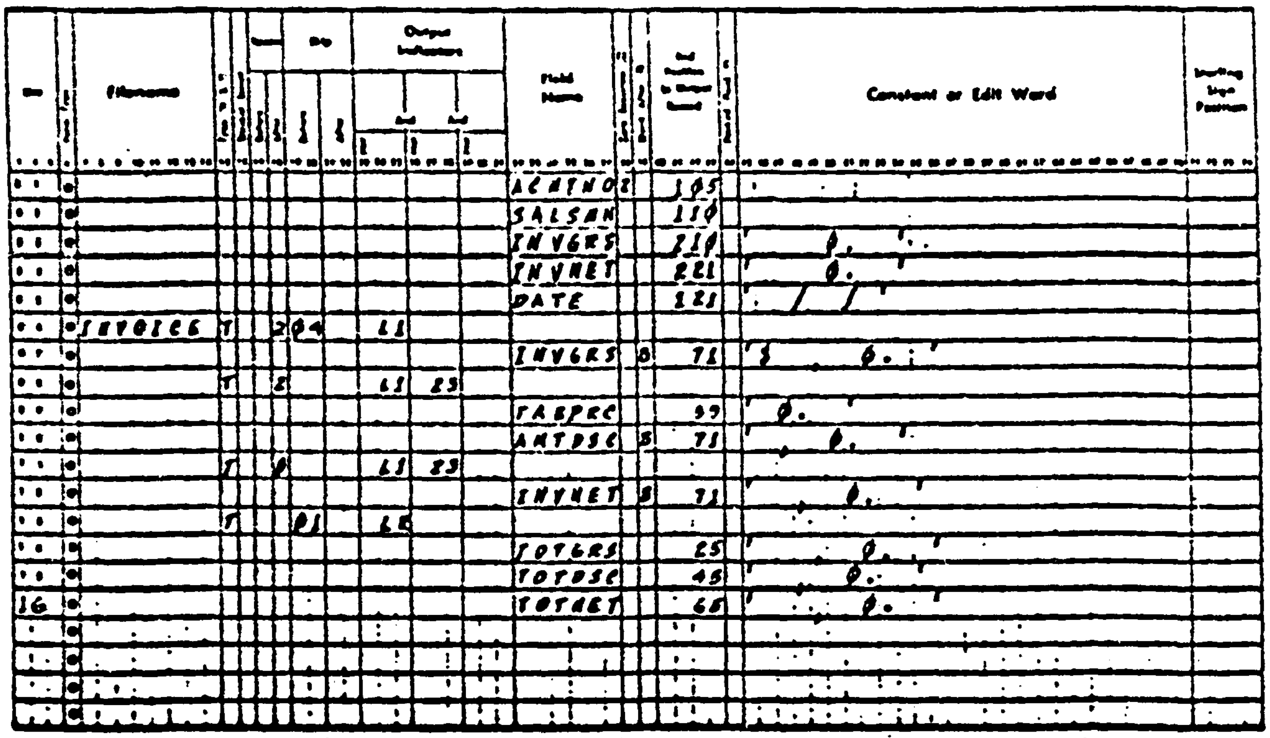
Invoicing



Line	Filename	Type	Space	Skip	Output Indicators	Field Name	Edit Codes	Blank After	End Position in Output Record	Constant or Edit Word
		D	2	1	05					
						WHSLOC			9	' & &'
						STKNO			26	' - - '
						DESCR			37	
						UNMEAS			41	
						QTY	Z		46	
					N24	QTY	Z		61	
					24 21N22				61	'B.O.'
					25				51	'CANC'
					N24	UNTPRI			69	' 0. '
					N24	GRSAMT			71	' 0. '
	SURCARDS	T	3		L1					
									1	'B'
						INVGRS			8	
						INVNET			15	
16						SALSMN			61	
17						DATE			66	
18						INVCNO			70	
19						ORDRNO			75	
20						AGNTNO			80	

Figure P7

Output Specifications



	Filename			Output Indicators		Field Name				Constant or Edit Word
						ACCTNO	Z		1Ø5	
						SALSMN			11Ø	
						INVGRS			21Ø	' Ø. '
						INVNET			221	' Ø. '
						DATE			121	' / / '
	INVOICE	T	2Ø4	L1						
						INVGRS		B	71	'$. Ø. '
		T	2	L1	23					
						TABPEC			39	' Ø. '
						AMTDSC		B	71	' Ø. '
		T	Ø	L1	23					
						INVNET		B	71	' Ø. '
		T	Ø1	LR						
						TOTGRS			25	' Ø. '
						TOTDSC			45	' Ø. '
16						TOTNET			65	' Ø. '

Figure P9

CUSTOMER INVOICING

The purpose of this program is to compare the RPG invoicing program to a BAL invoicing program by means of models. The reader, who is familiar with the invoice can compare the methods, while the reader, who is not familiar with the invoice, can benefit two ways, one from the programming and the other from the invoice.

Cards

The invoice summary card contains the first billing number to be used, and the date of the billing. It has a card code of zero.

The customer address card is comparable to an accounts receivable subsidiary ledger in the file, but being processed only if a purchase was made by the customer who has purchased an item between billing dates. One card has the customer number, order number, and order date. This is used for each customer who has purchased an item between billing dates. One card is made of each item purchased. This card has a card code of zero.

The detail item card contains a customer number, the item number, name of item, quantity of the item, the unit price of each item and the discount allowance to the customer. Later in the program, the net amount and gross amount will be punched and printed on the card.

The invoice summary card is produced at the end of the customer billing. One is produced for each customer that has ordered an item during the billing period. It will contain the customer number, the invoice number, the invoice date, and the total net amount of all items purchased. It has a card code of four.

At the end of all the billing, a new invoice card is produced, which contains only the starting invoice number. The date is punched in by the operator at the date the new invoicing is to be started. This card has a card code of zero. The last card on the deck has a cc of 5, and it is a dummy to facilitate ending the program.

Sequencing

Cards must first be sorted to be put into sequence. The customer card, the order card and the detail cards must be sorted in sequence by customer number.

Sorting

Sort on col. 1 first. This will leave three stacks of cards with a card code of 1, 2 and 3. Then remove cards with card code 2 and sort on cols. 17-16. This is the date. In case one customer ordered more than one item during the billing period, then sort on col. 15-14. This sorts the month. The year in cols. 89-18 may also be sorted if necessary.

Now place all three stacks together. If the cards are apart to begin with only the order cards will need to be sorted by the date of the month and then the month.

All cards are then sorted on cols. 7-2, the customer number. This should place the stack ready for use. Add the invoice summary and date card to the top of the deck in front of it. Then place the two dummy cards on the back of the deck. These two cards are used to put all the cards through the machine and print and punch the final cards. The final print out will be "end of invoicing."

Every order card should have an address card behind it, followed by at least one detail card and perhaps a second order card from the same person. This person will not have an address card, but the card must be followed by at least one detail card. There may be several address cards in a row, but these are from people who have not purchased anything. They will be skipped.

Invoicing

Place all the data cards in the primary hopper.

Register 13 is being used as a base register, and the first card read is the inventory date card. It gives the starting invoice number and the date of billing. This is dumped into stacker #5. This information is moved from area 5 to area 1 (74 bytes). Then the card code only is blanked out. The inventory number, located in number 2 is packed into number 12.

The second card is read. It could be an address card or it could be an order card. If it is an order card, the information from AR5 is moved into AR3. We then compare the card code, which is Fl. It is an Fl so we branch and store to LB1.

At LB1, we will store the information such as the customer number, the order number, the month, the day and the year. The rest of the area is now cleared. This card drops into stacker number 3. BAS 9 takes us back to where we came from, and then we locate a customer address card.

At LB3, the next card is read, and this information is stored into AR2. The customer number is compared. If it does compare with the order card number, the mode of payment, located on the customer address card, is moved into number 9.

The print area is cleared (work area AR4). The name is loaded into AR4 and printed. After the next two steps are skipped, a space is taken and street address, customer, inventory number and the date are loaded into AR4. These are moved to print where they are printed and followed by a space.

We now blank out AR4-47(23) to clear the work area. By doing this we can use the same area for many things. City and State are moved to PRIN and printed, then a space is taken. The order card contents are now printed out (they are stored in LB1). Number 14, which is the customer number, is moved to AR4 and printed.

This will print out as "YOUR ORDER CARD", with customer number and dated. This is followed by a space.

At LB8, the next card is read and stored into AR3. The following step is a no-op statement the first time through.

LB6--Compare the card code to see if it is a detail card. If it is a detail card, we go to LB4.

LB4--Process the detail item card. First check to see that there are no punches in columns 8 and 27. If no punches are located, load the quantity into number 7 and load price into 6, then multiply these two together. Now we add number 7 to number 5. They will continue to work as an accumulator to produce the total Gross Income. The Mask is moved to AR4 and the gross amount is edited. This information is moved to BRIN and printed.

The new amount is determined by packing discount into number 8 and multiplying the gross amount by the discount. The 01 instruction rounds off this amount. We may lose a penny or gain a penny by using this method, but it averages out over a period of time.

The discount is subtracted from the gross amount and the mask is moved into AR4 followed by the edit. The rest of AR4 (43 bytes), is now blanked out. The net plus gross amount are moved into AR6. This allows us to punch the information into the card.

CONT--Uses print head 2 and at AGAI (XIO AR6-7), we print on the card head the net amount and gross amount. The area is now blanked out. We check for channel 12, if yes, we go back to LB8. This means that we have now read the first invoice card, the first address card and the first detail card. Let's assume that there are no more detail cards for this person, and proceed to go back through the file again. This puts us back to where we would normally read a detail card, LB8.

LB8--A card is read and put into AR3. Now we go past the no-op step and check to see if it is a detail card. If it is not, it must be an address card. If so, go back to LB8.

Read another card and store the information into AR3. We go past the no-op step and compare to see if it is a detail card (F3). If it is not, we pass through the next step. Is it an address card? If not we assume it is an order card. Therefore we take the BAS9 to LB2. At LB2, we store the order number, the month, the day, year and clear part of this area again. The card falls into stacker 3 and a BCR is taken back to where we came from. A compare is made to determine whether this was a detail card. If not, we proceed to LB5, and print the invoice summary information which will be the ending statement for this customer.

A mask is moved into AR4, and the gross amount is edited and printed. After printing, a space is taken, and the percent of discount is entered.

The discount (no. 10) is packed into number 8. The percent is multiplied times the amount to produce the net amount. A round is taken and the mask is moved into PRIN and edited, after which, we print and space and then blank the rest of AR4.

We take the gross amount and subtract the net amount from it, and move the mask to AR4. Then we edit the net amount into AR4. The net area is cleared by subtracting it from itself. This information is printed and spaced.

Read a card and store this information into AE2. Compare the customer number (let's say they do compare). If they did not compare, we would go back to LB3 and cycle until we did find a number that does compare, or we would go back and cycle until we came to the customer number that has a higher number than the one located in LB2. If this happens, we branch to SEQR, which is an out of sequence routine, which halts the program.

LB3--Read a card from the order file. At LB3, the information from this card is put into AR3 and we go past the no-op. We now compare the card to see if it is a detail card. If it is, we branch to LB4.

LB4--A compare is made of columns 8-27 to be sure there are no punches. However, this time there is a punch in the card. Branch off to PCHF, which is a read format error routine. This card is checked again to make sure it is a detail card. (We assume it is.) Switch with the MVI instruction EOFQ-1 and move 00 into this condition code. The next instruction moves a blank into AR4 followed by a clear with a MVC instruction.

The print error test (number 13) is moved into AR4+50 for 24 bytes. This error message will print out on the paper as a "punch error" (in the item card). Then blank out the text-error message and place this card into stacker number 2.

Read the next card and place the information into AR3, check the customer numbe. to see if it is a new customer. If it is not a new customer, we go back to CB9 and read another card. We continue this until all of a customer's cards are used up. This is indicated by the appearance of a different customer number.

If a new number appears, we move down to the next MVI instruction and change the card code that changed before. This time it is changed to 15. Then we unconditionally branch back to LB3-4; the instruction is a BAS9 to LB1. Store the customer number, order number, day and year, then clear the area. This card will go to stacker number 3. The BCR is taken back to LB3.

At LB3, locate a customer's address card to match this order number. If the numbers do match, print the invoice header information, order card contents.

Read a card from the order file, and go to LB4 for the processing of the detail item card. Check for punches, if no, go on down and print the information, and punch and write on the card. The TIOB instruction checks for the overflow which is "43". Check for channel 12. If we were there, go to OVF1, which is the channel 12 condition routine. The MV1 15 into LB8-11 moves the unconditional branch condition into LB8+11. We then branch unconditionally back to LB11 - 4. This instruction is a BC 15 to LB8.

At LB8, we read a card from the order file. The information is moved into AR3. Then BC15 to LB11. The zero condition code has just been changed to a 15 by the MV1 instruction in the overflow area (OVFL).

At LB11, we immediately make LB8+11 a zero condition code again. Compare customer numbers, if not equal, branch to LB12 and at LB12 (BASF) to LB2.

LB2--Store the order number, month, day and year, and then clear the area. Put this card into stacker 3, and branch back to the statement following the BAS 9 where we just came from (LB12).

The next instruction is a CL1 statement, whereby, we compare to see if this is a detain card. If it is not a detain card, we branch to LB5.

LB5 is an area where the ending routine is effected for this customer. We print the invoice summary information, produce the summary card, increase the invoice number, and now b· ınch back to LB3.

The dummy card is read at LB3, which is a 5 in column 1, and in columns 7-27 is punched "end of invoicing". This information is stored into AR2. We then compare the customer numbers. The customer numbers do not compare because they are all blanks. We drop through the next two routines. Blank out the mode of payment because this area is blank.

The net amount is to be saved so it can be punched into the invoice summary card. The area is cleared and mode is moved. Amount of days, percent of discount and the days net are moved into mode.

The last line is printed out as Mode of Payment 20 days 2% discount OK 30 days Net. Naturally the figures will change with each customer. Check for channel 12, if not, go down and print and skip to column 1 and blank out more of AR4 to clear the area.

Move the card number, which is NO. into AR2 number (24). The customer number, inventory number, the date and the net amount (2), and put AR2 into AR7.

We take the BAS 8 and branch and store to punch.

This action produces a new invoice summary card, which has a card code of 4. A return is now made from the punch routine.

Move AR12, which is the invoice number, into CPL1. Move area 13, which is an inventory date into CPL 2, clear part of AR2, and move customer number into CPL4 and BAS 8 to HEAD, where we write on the card.

The card will contain customer number, invoice number, invoice date and net amount. This ca: ı will then be placed into stacker 5.

To increase the invoice number, we take number 19, which is the number 1, and add it to 12, which is the invoice number. Move in the mask and edit the number, and go back to LB3.

At LB3, we are going to try to locate the customer address card number to match the order number, that we have stored in LB2.

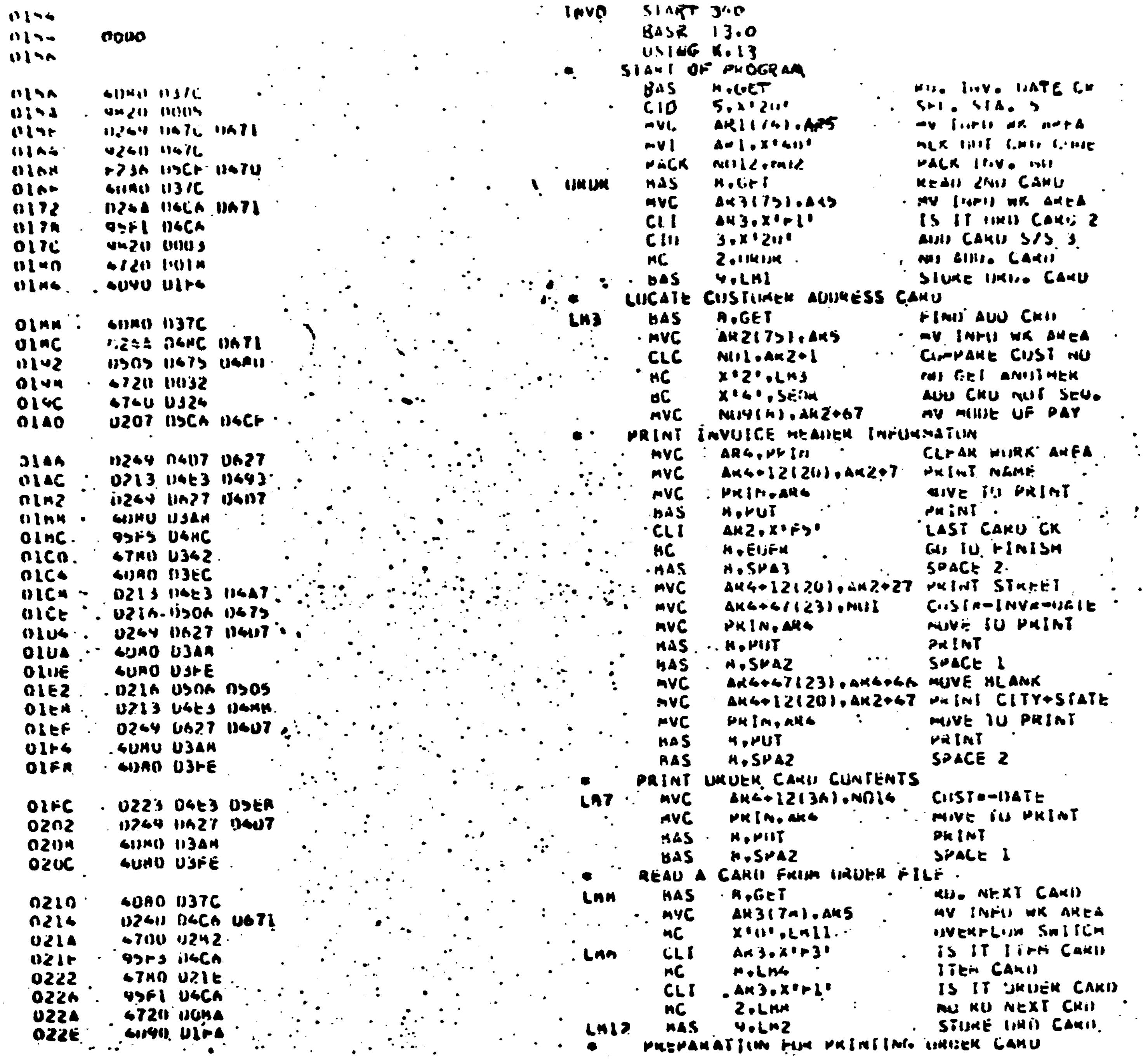
The header information is printed by first printing the name, but this time it is not the same, it is the "END OF INVOICING". This is printed out after which we come to the CL1 we have been passing up all the time during the program. We check to see if it is a card code of 5. We assume it is so we go to EOFR, which is the end of the file routine.

EOFR--Move the new invoice number, which is at number 2 to AR4+35. Reset the switch at PNCH-5. This cuts down the amount of information that is going to be printed on the card from 16 down to 8. Now we take the next MV1 instruction which moves a 00 into PUER3. This changes the hopper code from 4 to 5. This card will now drop into stacker number 5.

We then move a card indicator code into AR4+34. Move AR9 into AR7, which is the invoice number. We then branch to PNCH and pick out this last card. Only the invoice summary number will be punched. The date will have to be punched in by the operator at the next billing period.

After number 1 is selected to print this same information, the final action is the halt.

Invoicing--Basic Assembler



```
0154                          INVO    START 340
0154   00D0                           BASR  13,0
0156                                  USING *,13
                              *    START OF PROGRAM
0156   4D80 D37C                      BAS   8,GET              RD. INV. DATE CD
015A   9420 0005                      CIO   5,X'20'            SET. STA. 5
015E   D249 D47C D471                 MVC   AR1(74),AR5        MV INFO WK AREA
0164   9240 D47C                      MVI   AR1,X'40'          BLK OUT CRD CODE
0168   F23A D5CF D47D                 PACK  NO12,NI12          PACK INV. NO
016E   4D80 D37C              ORDR    BAS   8,GET              READ 2ND CARD
0172   D24A D4CA D471                 MVC   AR3(75),AR5        MV INFO WK AREA
0178   95F1 D4CA                      CLI   AR3,X'F1'          IS IT ORD CARD 2
017C   9420 0003                      CIO   3,X'20'            ADD CARD S/S 3
0180   4720 D018                      BC    2,ORDR             NO ADD. CARD
0184   4D90 D1F4                      BAS   9,LA1              STORE ORD. CARD
                              *    LOCATE CUSTOMER ADDRESS CARD
0188   4D80 D37C              LA3     BAS   8,GET              FIND ADD CRD
018C   D24A D48C D471                 MVC   AR2(75),AR5        MV INFO WK AREA
0192   D505 D475 D48D                 CLC   NO1,AR2+1          COMPARE CUST NO
0198   4720 D032                      BC    X'2',LA3           NO GET ANOTHER
019C   4740 D324                      BC    X'4',SEQN          ADD CRD NOT SEQ.
01A0   D207 D5CA D4CF                 MVC   NO9(8),AR2+67      MV MODE OF PAY
                              *    PRINT INVOICE HEADER INFORMATON
01A6   D249 D407 D627                 MVC   AR4,PRIN           CLEAR WORK AREA
01AC   D213 D4E3 D493                 MVC   AR4+12(20),AR2+7   PRINT NAME
01B2   D249 D627 D407                 MVC   PRIN,AR4           MOVE TO PRINT
01B8   4D80 D3A8                      BAS   8,PUT              PRINT
01BC   95F5 D48C                      CLI   AR2,X'F5'          LAST CARD CK
01C0   4780 D342                      BC    8,EOFR             GO TO FINISH
01C4   4D80 D3EC                      BAS   8,SPA3             SPACE 2
01C8   D213 D4E3 D4A7                 MVC   AR4+12(20),AR2+27  PRINT STREET
01CE   D216 D506 D475                 MVC   AR4+47(23),NO1     CUST#-INV#-DATE
01D4   D249 D627 D407                 MVC   PRIN,AR4           MOVE TO PRINT
01DA   4D80 D3A8                      BAS   8,PUT              PRINT
01DE   4D80 D3FE                      BAS   8,SPA2             SPACE 1
01E2   D216 D506 D505                 MVC   AR4+47(23),AR4+46  MOVE BLANK
01E8   D213 D4E3 D4BB                 MVC   AR4+12(20),AR2+47  PRINT CITY+STATE
01EE   D249 D627 D407                 MVC   PRIN,AR4           MOVE TO PRINT
01F4   4D80 D3A8                      BAS   8,PUT              PRINT
01F8   4D80 D3FE                      BAS   8,SPA2             SPACE 2
                              *    PRINT ORDER CARD CONTENTS
01FC   D223 D4E3 D5E8         LA7     MVC   AR4+12(36),NO14    CUST#-DATE
0202   D249 D627 D407                 MVC   PRIN,AR4           MOVE TO PRINT
0208   4D80 D3A8                      BAS   8,PUT              PRINT
020C   4D80 D3FE                      BAS   8,SPA2             SPACE 1
                              *    READ A CARD FROM ORDER FILE
0210   4D80 D37C              LA8     BAS   8,GET              RD. NEXT CARD
0214   D249 D4CA D471                 MVC   AR3(74),AR5        MV INFO WK AREA
021A   4700 D242                      BC    X'0',LA11          OVERFLOW SWITCH
021E   95F3 D4CA              LA9     CLI   AR3,X'F3'          IS IT ITEM CARD
0222   4780 D21E                      BC    8,LA4              ITEM CARD
0226   95F1 D4CA                      CLI   AR3,X'F1'          IS IT ORDER CARD
022A   4720 D0BA                      BC    2,LA8              NO RD NEXT CRD
022E   4D90 D1FA              LA12    BAS   9,LA2              STORE ORD CARD
                              *    PREPARATION FOR PRINTING ORDER CARD
```

Figure INVO-1

Invoicing--Basic Assembler

```
0232    D505 0475 04C7            CLC   NO1,AR3+1          ? NO ORDER
0238    4770 00FE                 BC    7,LBS              YES
023C    4080 03FE                 BAS   8,SPA2             SPACE 1
0240    47F0 00A6                 BC    15,LB7             GO TO ORDER RET
                            *     PRINT INVOICE SUMMARY INFORMATION
0244    D20B 0515 0610      LBS   MVC   AR4+62(12),MSK2    MOVE IN MASK
024A    DE0B 0515 0585            ED    AR4+62(12),NO5     EDIT GROSS AMT
0250    4080 03FE                 BAS   8,SPA2             SPACE 1
0254    D249 0627 04D7            MVC   PRIN,AR4           MOVE TO PRINT
025A    4080 03A8                 BAS   8,PUT              PRINT
025E    4080 03FE                 BAS   8,SPA2             SPACE 2
0262    D201 050C 05CC            MVC   AR4+53(2),HOLD     DISCOUNT %
0268    F271 0591 05CC            PACK  NUM,HOLD           PACK DISCOUNT
026E    FC75 0591 0585            MP    NUM,NO5(6)         MULT % X AMOUNT
0274    960F 0597                 OI    NUM+6,X'0F'        ROUND OFF AMT
0278    D20B 0515 0610            MVC   AR4+62(12),MSK2    MOVE IN MASK
027E    DE0B 0515 0592            ED    AR4+62(12),NUM+1   EDIT MASK
0284    D249 0627 04D7            MVC   PRIN,AR4           MOVE IN PRINT
028A    4080 03A8                 BAS   8,PUT              PRINT
028E    4080 03FE                 BAS   8,SPA2             SPACE 1
0292    D201 050C 050B            MVC   AR4+53(2),AR4+52   BLANK AREA
0298    FB55 0585 0592            SP    NO5(6),NUM+1(6)    GROSS -NET
029E    D20B 0515 0610            MVC   AR4+62(12),MSK2    MOVE IN MASK
02A4    DE0B 0515 0585            ED    AR4+62(12),NO5     EDIT NET AMT
02AA    FB55 0585 0585            SP    NO5(6),NO5(6)      CLEAR AREA
02B0    D249 0627 04D7            MVC   PRIN,AR4           MOVE TO PRINT
02B6    4080 03A8                 BAS   8,PUT              PRINT
02BA    4080 03FE                 BAS   8,SPA2             SPACE 1
02BE    D20A 04A4 0516            MVC   AR2+24(11),AR4+63  SAVE NET AMT
02C4    D20A 0516 0515            MVC   AR4+63(11),AR4+62  CLEAR AREA
02CA    D235 04D7 0700            MVC   AR4(54),HDNE       MOVE IN HEADING
02D0    D201 04ED 05CA            MVC   AR4+22(2),NO4      DAYS
02D6    D201 04F9 05CC            MVC   AR4+34(2),NO4+2    % OF DISCOUNT
02DC    D201 0507 05CE            MVC   AR4+48(2),NO4+4    DAYS NET
02E2    D249 0627 04D7            MVC   PRIN,AR4           MOVE TO PRINT
02E8    9143 019A                 TM    JUMP,X'43'         TEST CH. 12
02EC    47F0 019E                 BC    15,MOVE            NOT CH. 11
02F0    4080 030A           JUMP  BAS   8,SKIP             CH12 SKP TO CH1
02F4    4080 03A8           MOVE  BAS   8,PUT              PRINT
02F8    4080 030A                 BAS   8,SKIP             SKIP TO CH 1
02FC    D20F 04F9 04F8            MVC   AR4+34(16),AR4+33  BLANK AREA
                            *     PRODUCE INVOICE SUMMARY CARD
0302    D217 048C 0474            MVC   AR2(24),NO         CARD NO.
0308    D222 0604 048C            MVC   AR7(35),AR2        CUST-INV-DAT-NET
030E    4080 0410                 BAS   8,PUCH             PUNCH CARD
0312    D204 0521 0495            MVC   CPL1,AR12          MOVE INVOICE NO
0318    D205 0541 049E            MVC   CPL2,AR13          MOVE INV. DATE
031E    D20F 0495 0494            MVC   AR2+9(15),AR2+8    CLAR AREA
0324    D221 0541 04B0            MVC   CPL4,AR11          MOVE CUST NO
032A    4080 043C                 BAS   8,HEAD             PRINT CARD
032E    D205 0475 04C7            MVC   NO1,AR3+1          STORE CUST NO
                            *     INCREASE INVOICE NUMBER
0334    FA30 05CF 060F            AP    NO12(4),NO14(1)    ADD 1 TO INV NO.
033A    D207 047C 0594            MVC   AR1(8),MSK1        MOVE IN MASK
0340    DE07 047C 05CF            ED    AR1(8),NO12        EDIT CUST NO
```

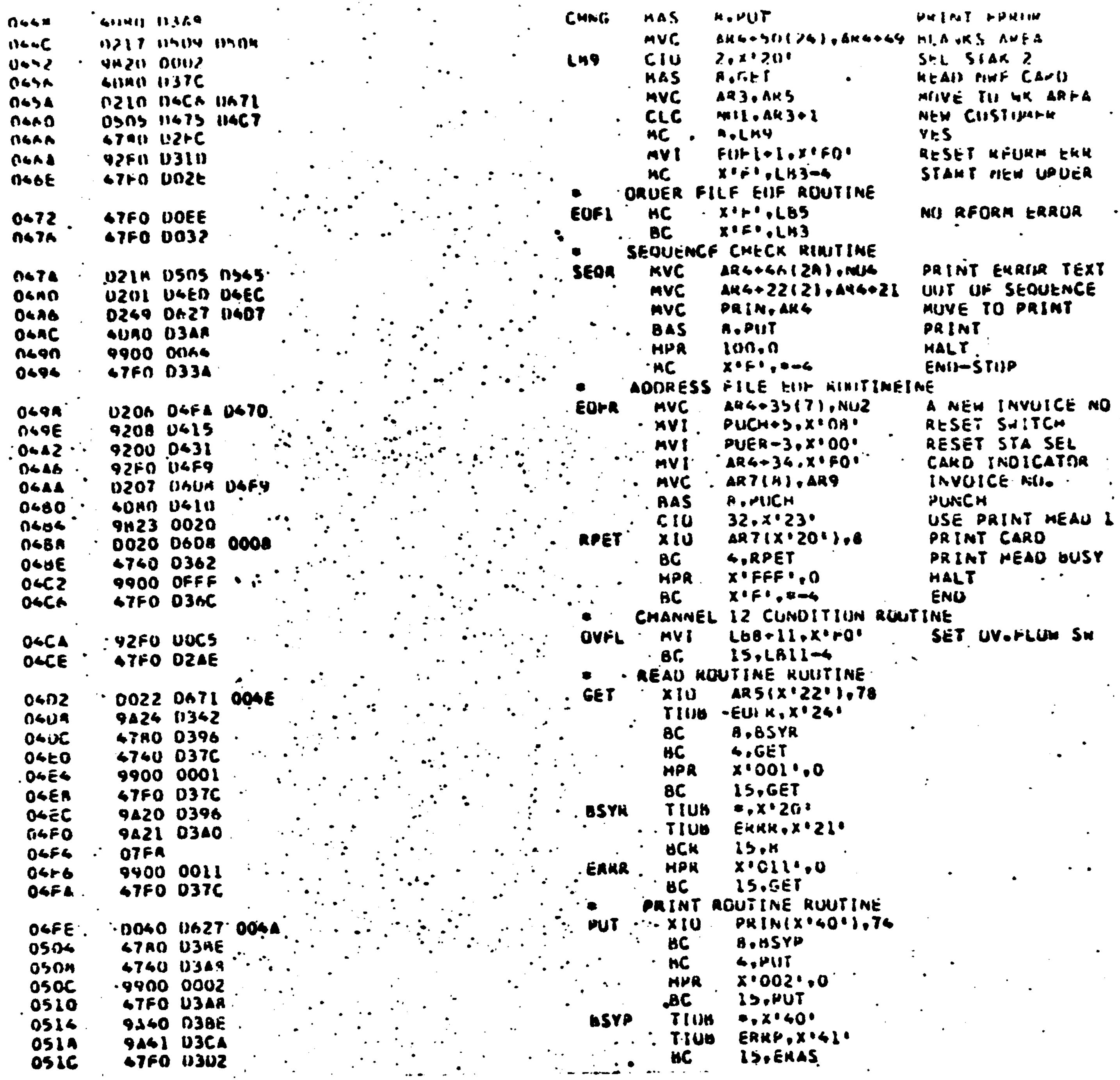
Figure INVO-2

Invoicing--Basic Assembler

```
0346    47F0 0032                        BC    X'F',LB3             GET ADD. CRD
                                 *     STORE CONTENTS OF ORDER CARD
034A    D205 D475 D4C7           LB1     MVC   NO1,AR3+1            CUSTOMER NO
0350    D205 D5F9 D4CD           LB2     MVC   NO15,AR3+7           ORDER NO
0356    D201 D607 D4D3                   MVC   NO16,AR3+13          MONTH
035C    D201 D60A D4D5                   MVC   NO17,AR3+15          DAY
0362    D201 D60D D4D7                   MVC   NO18,AR3+17          YEAR
0368    D201 D4D7 D4D9                   MVC   AR3+17(2),AR3+19     CLEAR AREA
036E    9820 0003                        CIO   3,X'20'              S/S 3
0372    07F9                             BCR   X'F',9               PRINT ORDER CARD
                                 *     PROCESSING OF DETAIL ITEM CARD
0374    D511 D4CD D6FB           LB4     CLC   AR3+7(18),AR8        CK NO PUNCHES
037A    4770 D2C4                        BC    7,PCHF               YES-PUNCHES
037E    F253 D5AA D506                   PACK  NO7,AR4+47(4)        QUANITY
0384    F224 D5C3 D50F                   PACK  NO6,AR4+56(5)        UNIT PRICE
038A    FC52 D5AA D5C3                   MP    NO7,NO6              GROSS AMT
0390    FA55 D585 D58B                   AP    NO5(6),NO7           ADD TOTAL GROSS
0396    D209 D517 D61C                   MVC   AR4+64(10),MSK3      PRINT ITEM CARD
039C    DE09 D517 D58C                   ED    AR4+64(10),NO7+1     EDIT GROSS
03A2    D249 D627 D4D7                   MVC   PRIN,AR4             MOVE TO PRINT
03A8    4D80 D3A8                        BAS   8,PUT                PRINT
03AC    F271 D591 D5CC                   PACK  NO8,NO10             NET AMOUNT
03B2    FC75 D591 D58B                   MP    NO8,NO7              MULT % X GROSS
03B8    960F D597                        OI    NO8+6,X'0F'          ROUND OFF NET
03BC    FB55 D58B D592                   SP    NO7,NO8+1(6)         SUB. DIS-GROSS
03C2    D209 D50E D61C                   MVC   AR4+55(10),MSK3      PUNCH ITEM CARD
03C8    DE09 D50E D58C                   ED    AR4+55(10),NO7+1     EDIT NET
03CE    D22A D4DF D4DE                   MVC   AR4+8(43),AR4+7      BLANK AREA
03D4    D218 D68F D508                   MVC   AR6,AR10             MOVE NET+GROSS
03DA    D024 D68F 0019           REPT    XIO   AR6(X'24'),25        PUNCH ITEM CARD
03E0    4740 D2B4                        BC    4,REPT               PUNCH BUSY
03E4    9822 0002                        CIO   2,X'22'              SEL. STAK 2
03E8    9823 0010                CONT    CIO   16,X'23'             SEL PRT HD 2
03EC    4740 D292                        BC    4,CONT               PRT HD BUSY
03F0    D020 D6C6 0012           AGAI    XIO   AR6+7(X'20'),18      PRINT ON CARD
03F6    4740 D29A                        BC    4,AGAI               PRINT HEAD BUSY
03FA    D211 D50F D50E                   MVC   AR4+56(18),AR4+55    BLANK AREA
0400    9A43 D374                        TIOB  OVFL,X'43'           TEST CH 12
0404    47F0 D0BA                        BC    X'F',LB8             READ NEXT CARD
                                 *     OVERFLOW CONDITION
0408    9200 D0C5                LB11    MVI   LB8+11,X'00'         RESET SWITCH
040C    D505 D475 D4C7                   CLC   NO1,AR3+1            NEW CUSTOMER
0412    4770 D0DA                        BC    X'7',LB12            YES
0416    47F0 D0A6                        BC    X'F',LB7             ORDER CARD
                                 *     READ FORMAT ERROR ROUTINE
041A    95F3 D4C6                PCHF    CLI   AR3,X'F3'            IS IT ITEM CARD
041E    4770 D000                        BC    7,LB6+8              NO IT IS NOT
0422    9200 D31D                        MVI   EOF1+1,X'00'         SET RFORM INDIC
0426    9240 D4D7                        MVI   AR4,X'40'            BLANK CARD CODE
042A    D230 D4D8 D4D7                   MVC   AR4+1(49),AR4        BLANKS AREA
0430    D217 D509 D5D3                   MVC   AR4+50(24),NO13      PRINT ERROR TEXT
0436    D249 D627 D4D7                   MVC   PRIN,AR4             MOVE TO PRINT
043C    9A43 D2EE                        TIOB  HOP,X'43'            TEST CH 12
0440    47F0 D2F2                        BC    15,CHNG              NOT CH 12
0444    4D80 D30A                HOP     BAS   8,SKIP               YES GO TO CH 1
```

Figure INVO-3

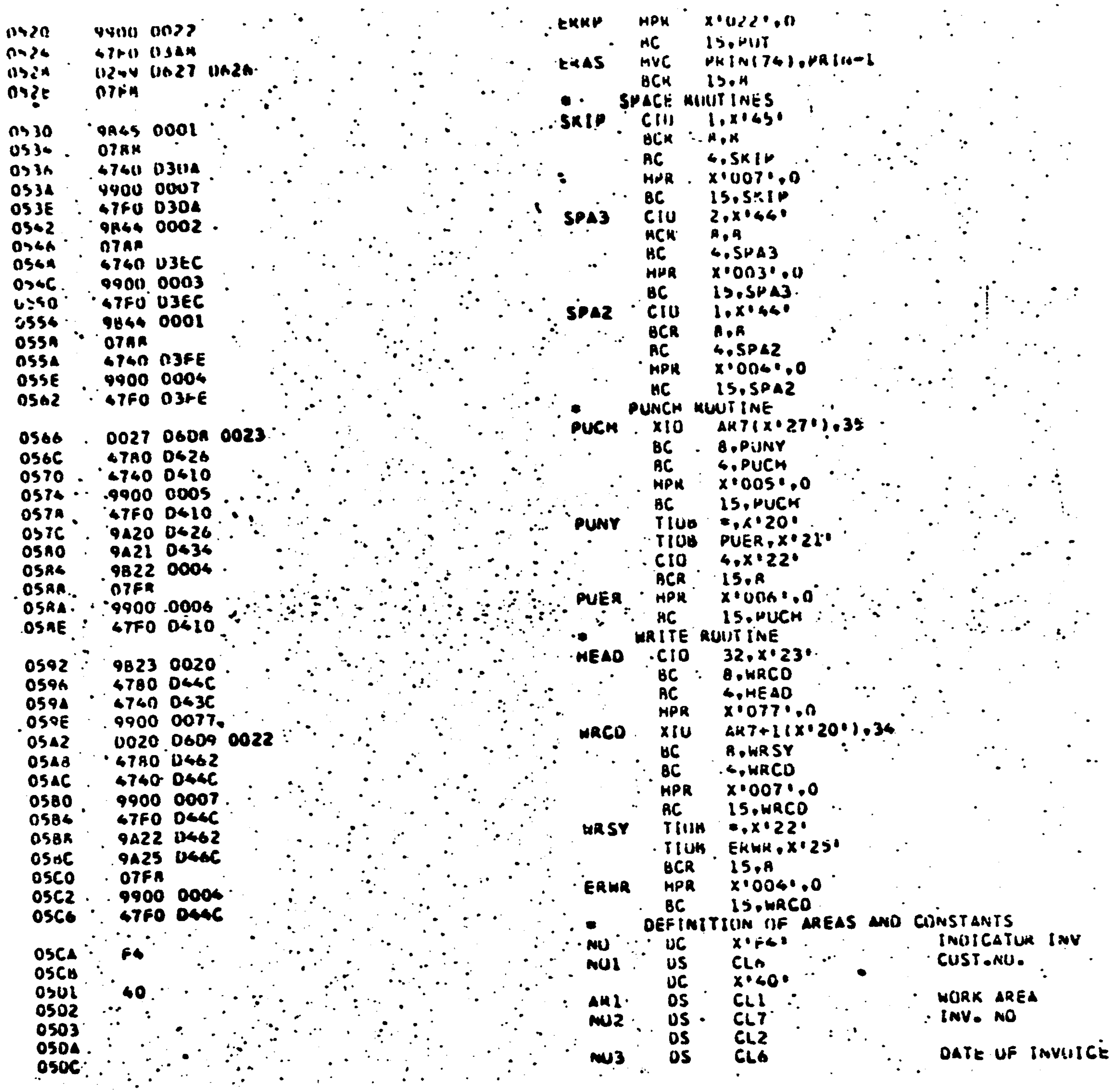
Invoicing--Basic Assembler



```
0448    4080 D3A9                 CHNG   BAS   R,PUT                 PRINT ERROR
044C    D217 D509 D508                   MVC   AR4+50(24),AR4+49     BLANKS AREA
0452    9820 0002                 LH9    CIO   2,X'20'               SEL STAK 2
0456    4080 D37C                        BAS   R,GET                 READ NWF CARD
045A    D210 D4CA D471                   MVC   AR3,AR5               MOVE TO WK AREA
0460    D505 D475 D4C7                   CLC   NU1,AR3+1             NEW CUSTOMER
0466    4790 D2FC                        BC    R,LH9                 YES
046A    92F0 D310                        MVI   EOF1+1,X'F0'          RESET RFORM ERR
046E    47F0 D02E                        BC    X'F',LB3-4            START NEW ORDER
                                  *      ORDER FILF EOF ROUTINE
0472    47F0 D0EE                 EOF1   BC    X'F',LB5              NO RFORM ERROR
0476    47F0 D032                        BC    X'F',LB3
                                  *      SEQUENCF CHECK RIUTINE
047A    D21B D505 D545            SEQR   MVC   AR4+46(28),NU4        PRINT ERROR TEXT
0480    D201 D4ED D4EC                   MVC   AR4+22(2),AR4+21      OUT OF SEQUENCE
0486    D249 D627 D407                   MVC   PRIN,AR4              MOVE TO PRINT
048C    40B0 D3AR                        BAS   R,PUT                 PRINT
0490    9900 0064                        HPR   100,0                 HALT
0494    47F0 D33A                        BC    X'F',*-4              END-STOP
                                  *      ADDRESS FILE EOF RINITINEINE
0498    D206 D4FA D470            EOFR   MVC   AR4+35(7),NU2         A NEW INVOICE NO
049E    9208 D415                        MVI   PUCH+5,X'08'          RESET SWITCH
04A2    9200 D431                        MVI   PUER-3,X'00'          RESET STA SEL
04A6    92F0 D4F9                        MVI   AR4+34,X'F0'          CARD INDICATOR
04AA    D207 D6D8 D4F9                   MVC   AR7(8),AR9            INVOICE NO.
04B0    40B0 D410                        BAS   R,PUCH                PUNCH
04B4    9823 0020                        CIO   32,X'23'              USE PRINT HEAD 1
04B8    D020 D6D8 0008            RPET   XIO   AR7(X'20'),8          PRINT CARD
04BE    4740 D362                        BC    4,RPET                PRINT HEAD BUSY
04C2    9900 0FFF                        HPR   X'FFF',0              HALT
04C6    47F0 D36C                        BC    X'F',*-4              END
                                  *      CHANNEL 12 CONDITION ROUTINE
04CA    92F0 D0C5                 OVFL   MVI   LB8+11,X'F0'          SET OV.FLOW SW
04CE    47F0 D2AE                        BC    15,LB11-4
                                  *      READ ROUTINE RUUTINE
04D2    D022 D471 004E            GET    XIO   AR5(X'22'),78
04D8    9A24 D342                        TIOB  -EOI K,X'24'
04DC    4780 D396                        BC    8,BSYR
04E0    4740 D37C                        BC    4,GET
04E4    9900 0001                        HPR   X'001',0
04E8    47F0 D37C                        BC    15,GET
04EC    9A20 D396                 BSYR   TIOB  *,X'20'
04F0    9A21 D3A0                        TIOB  ERRR,X'21'
04F4    07FB                             BCR   15,R
04F6    9900 0011                 ERRR   HPR   X'011',0
04FA    47F0 D37C                        BC    15,GET
                                  *      PRINT ROUTINE RUUTINE
04FE    D040 D627 004A            PUT    XIO   PRIN(X'40'),74
0504    4780 D3BE                        BC    8,BSYP
0508    4740 D3A8                        BC    4,PUT
050C    9900 0002                        HPR   X'002',0
0510    47F0 D3A8                        BC    15,PUT
0514    9A40 D3BE                 BSYP   TIOB  *,X'40'
051A    9A41 D3CA                        TIOB  ERRP,X'41'
051C    47F0 D3D2                        BC    15,ERAS
```

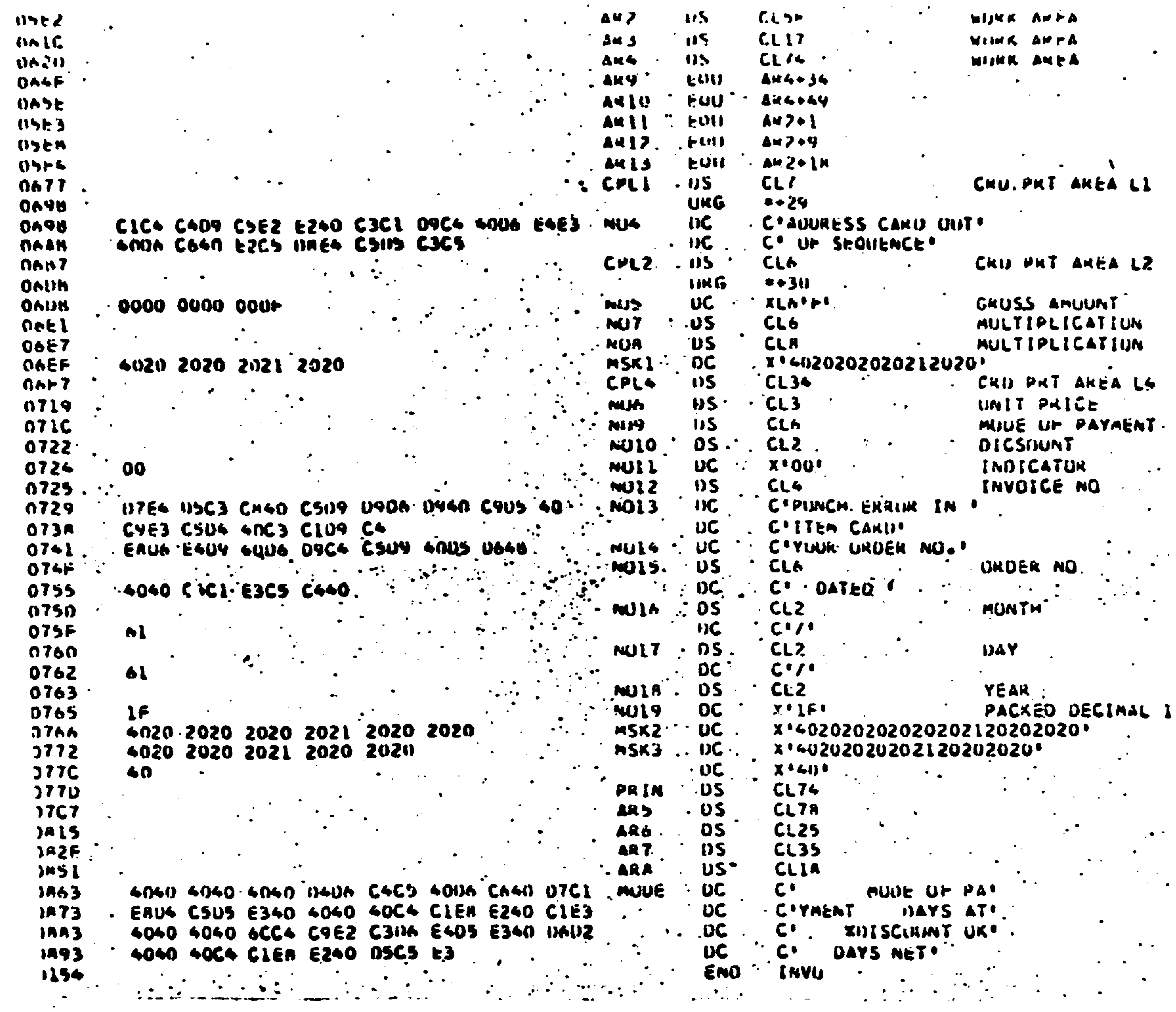
Figure INVO-04

Invoicing--Basic Assembler



```
0520    9900 0022               ERRP    HPR     X'022',0
0524    47F0 D3A8                       BC      15,PUT
0528    D249 D627 D62A          ERAS    MVC     PRIN(74),PRIN-1
052C    07F8                            BCR     15,8
                                *       SPACE ROUTINES
0530    9A45 0001               SKIP    CIO     1,X'45'
0534    0788                            BCR     8,8
0536    4740 D3DA                       BC      4,SKIP
053A    9900 0007                       HPR     X'007',0
053E    47F0 D3DA                       BC      15,SKIP
0542    9A44 0002               SPA3    CIO     2,X'44'
0546    0788                            BCR     8,8
0548    4740 D3EC                       BC      4,SPA3
054C    9900 0003                       HPR     X'003',0
0550    47F0 D3EC                       BC      15,SPA3
0554    9A44 0001               SPA2    CIO     1,X'44'
0558    0788                            BCR     8,8
055A    4740 D3FE                       BC      4,SPA2
055E    9900 0004                       HPR     X'004',0
0562    47F0 D3FE                       BC      15,SPA2
                                *       PUNCH ROUTINE
0566    0027 D6DA 0023          PUCH    XIO     AR7(X'27'),35
056C    4780 D426                       BC      8,PUNY
0570    4740 D410                       BC      4,PUCH
0574    9900 0005                       HPR     X'005',0
0578    47F0 D410                       BC      15,PUCH
057C    9A20 D426               PUNY    TIOB    *,X'20'
0580    9A21 D434                       TIOB    PUER,X'21'
0584    9B22 0004                       CIO     4,X'22'
0588    07F8                            BCR     15,8
058A    9900 0006               PUER    HPR     X'006',0
058E    47F0 D410                       BC      15,PUCH
                                *       WRITE ROUTINE
0592    9B23 0020               HEAD    CIO     32,X'23'
0596    4780 D44C                       BC      8,WRCD
059A    4740 D43C                       BC      4,HEAD
059E    9900 0077                       HPR     X'077',0
05A2    D020 D609 0022          WRCD    XIO     AR7+1(X'20'),34
05A8    4780 D462                       BC      8,WRSY
05AC    4740 D44C                       BC      4,WRCD
05B0    9900 0007                       HPR     X'007',0
05B4    47F0 D44C                       BC      15,WRCD
05B8    9A22 D462               WRSY    TIOB    *,X'22'
05BC    9A25 D46C                       TIOB    ERWR,X'25'
05C0    07F8                            BCR     15,8
05C2    9900 0004               ERWR    HPR     X'004',0
05C6    47F0 D44C                       BC      15,WRCD
                                *       DEFINITION OF AREAS AND CONSTANTS
05CA    F4                      NO      DC      X'F4'                   INDICATOR INV
05CB                            NO1     DS      CL6                     CUST.NO.
05D1    40                              DC      X'40'
05D2                            AR1     DS      CL1                     WORK AREA
05D3                            NO2     DS      CL7                     INV. NO
05DA                                    DS      CL2
05DC                            NO3     DS      CL6                     DATE OF INVOICE
```

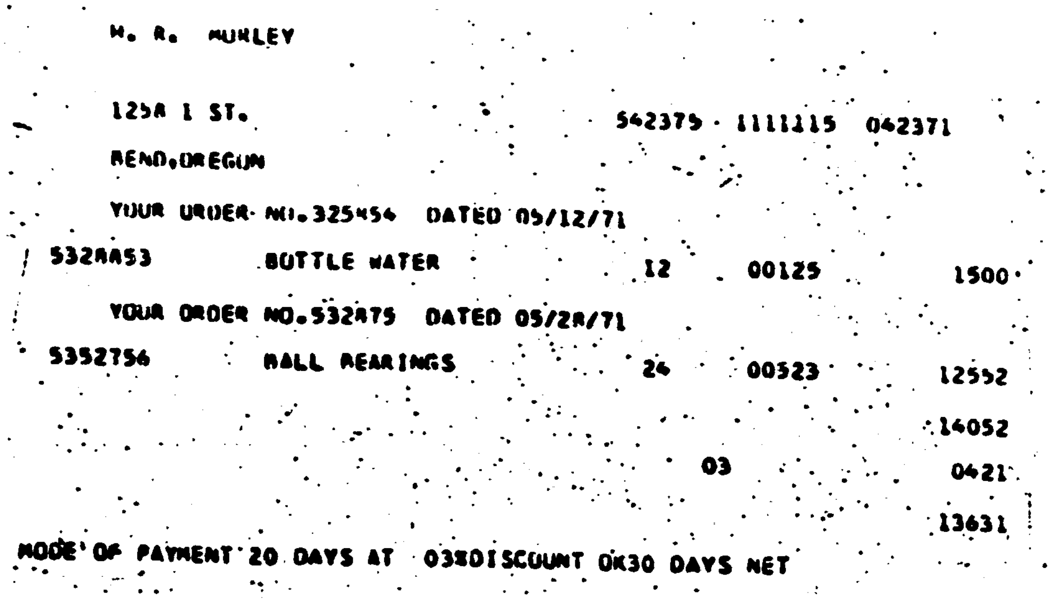
Figure INVO-05



```
05E2                                               AR2     DS    CL58               WORK AREA
061C                                               AR3     DS    CL17               WORK AREA
062D                                               AR4     DS    CL74               WORK AREA
064F                                               AR9     EQU   AR4+34
065E                                               AR10    EQU   AR4+49
05E3                                               AR11    EQU   AR2+1
05EB                                               AR12    EQU   AR2+9
05F4                                               AR13    EQU   AR2+18
0677                                               CPL1    DS    CL4                CRD.PKT AREA L1
0698                                                       ORG   *+29
0698   C1C4 C4D9 C5E2 E240 C3C1 D9C4 40D6 E4E3     NO4     DC    C'ADDRESS CARD OUT'
06A8   40D6 C640 E2C5 D8E4 C5D5 C3C5                       DC    C' OF SEQUENCE'
06B4                                               CPL2    DS    CL6                CRD PKT AREA L2
06D8                                                       ORG   *+30
06D8   0000 0000 000F                              NO5     DC    XL9'F'             GROSS AMOUNT
06E1                                               NO7     DS    CL6                MULTIPLICATION
06E7                                               NO8     DS    CL8                MULTIPLICATION
06EF   4020 2020 2021 2020                         MSK1    DC    X'4020202020212020'
06F7                                               CPL4    DS    CL34               CRD PKT AREA L4
0719                                               NO6     DS    CL3                UNIT PRICE
071C                                               NO9     DS    CL6                MODE OF PAYMENT
0722                                               NO10    DS    CL2                DICSOUNT
0724   00                                          NO11    DC    X'00'              INDICATOR
0725                                               NO12    DS    CL4                INVOICE NO
0729   D7E4 D5C3 C840 C5D9 D9D6 D940 C9D5 40       NO13    DC    C'PUNCH ERROR IN '
0738   C9E3 C5D4 40C3 C1D9 C4                              DC    C'ITEM CARD'
0741   E8D6 E4D9 40D6 D9C4 C5D9 40D5 D64B          NO14    DC    C'YOUR ORDER NO.'
074F                                               NO15    DS    CL6                ORDER NO
0755   4040 C4C1 E3C5 C440                                 DC    C'  DATED '
075D                                               NO16    DS    CL2                MONTH
075F   61                                                  DC    C'/'
0760                                               NO17    DS    CL2                DAY
0762   61                                                  DC    C'/'
0763                                               NO18    DS    CL2                YEAR
0765   1F                                          NO19    DC    X'1F'              PACKED DECIMAL 1
0766   4020 2020 2020 2021 2020 2020               MSK2    DC    X'402020202020202120202020'
0772   4020 2020 2021 2020 2020                    MSK3    DC    X'40202020202120202020'
077C   40                                                  DC    X'40'
077D                                               PRIN    DS    CL74
07C7                                               AR5     DS    CL78
0815                                               AR6     DS    CL25
082E                                               AR7     DS    CL35
0851                                               AR8     DS    CL18
0863   4040 4040 4040 D4D6 C4C5 40D6 C640 D7C1     MODE    DC    C'      MODE OF PA'
0873   E8D4 C5D5 E340 4040 40C4 C1E8 E240 C1E3             DC    C'YMENT    DAYS AT'
0883   4040 4040 6CC4 C9E2 C3D6 E4D5 E340 D6D2             DC    C'    %DISCOUNT OK'
0893   4040 40C4 C1E8 E240 D5C5 E3                         DC    C'   DAYS NET'
1154                                                       END   INVO
```

Figure INVO-06

Invoicing Output



```
     M. R.  MURLEY

     1258 I ST.                              542375 1111115  042371
     BEND,OREGON
     YOUR ORDER NO.325454  DATED 05/12/71
 5328853       BOTTLE WATER                 12     00125         1500
     YOUR ORDER NO.532875  DATED 05/28/71
 5352756       BALL BEARINGS                24     00523        12552
                                                                14052
                                                 03              0421
                                                                13631
MODE OF PAYMENT 20 DAYS AT  03%DISCOUNT OR30 DAYS NET
```

Figure INVO-7

RPG DISK OPERATION

This section will aid in the preparation of the work required for the Swanson study, since a disk operation or sorts is assumed. The main topics are presented on the next page, and are presented as File Description Specifications. The two main divisions of the section are:

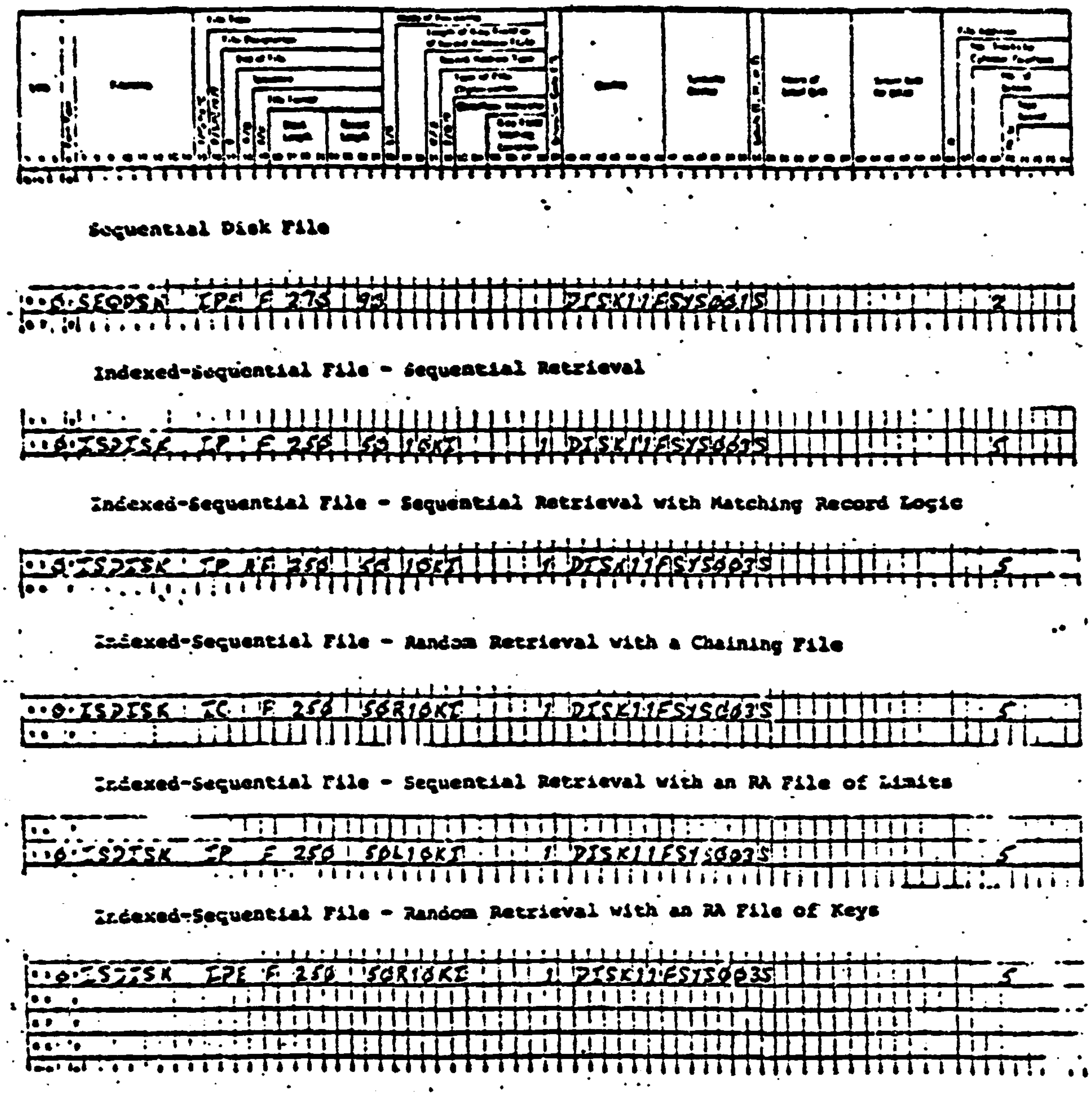
a. Sequential Disk Files

b. Indexed Sequential Files

SEQUENTIAL DISK FILES

Pre-sorted records are read from another external file (or disk) and written in sequence within the extents on the disk as fixed length records, either blocked or unblocked. The Job Control program is presented the actual location of the file on the disk at the time the file is loaded. The file label is written in the VOTC (volume table of contents). The label locates, identifies and protects the files.

File Description Specifications



Sequential Disk File

```
Ø SEQDSK  IPE F 275  93              DISK11FSYS001S          2
```

Indexed-Sequential File - Sequential Retrieval

```
Ø ISDISK  IP  F 256  50 10KI      1 DISK11FSYS003S          5
```

Indexed-Sequential File - Sequential Retrieval with Matching Record Logic

```
Ø ISDISK  IP KF 250  50 10KI      1 DISK11FSYS003S          5
```

Indexed-Sequential File - Random Retrieval with a Chaining File

```
Ø ISDISK  IC  F 250  50R10KI      1 DISK11FSYS003S          5
```

Indexed-Sequential File - Sequential Retrieval with an RA File of Limits

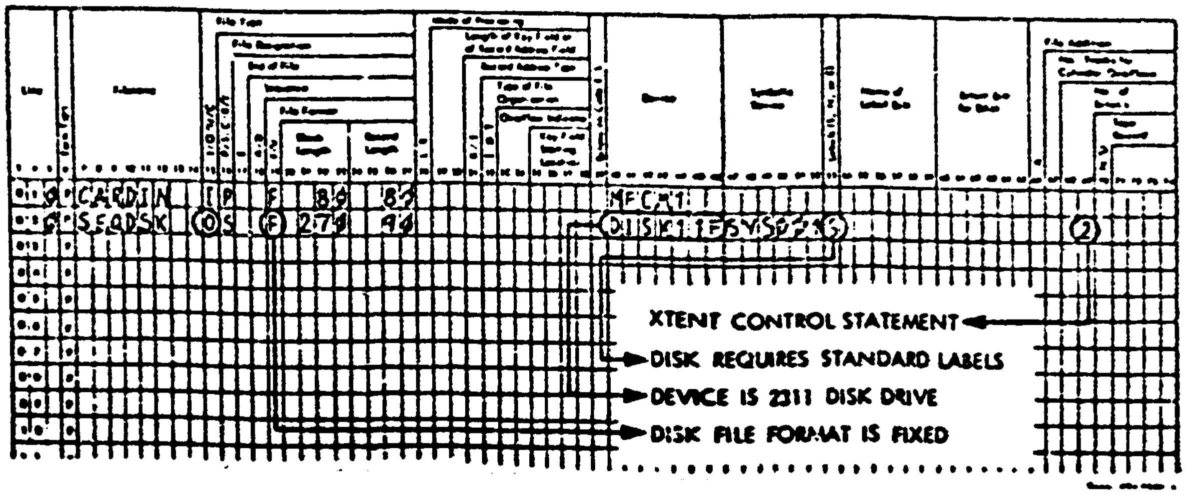
```
Ø ISDISK  IP  F 250  50L10KI      1 DISK11FSYS003S          5
```

Indexed-Sequential File - Random Retrieval with an RA File of Keys

```
Ø ISDISK  IPE F 250  50R10KI      1 DISK11FSYS003S          5
```

Figure S-0 RPG Coding for Disk Operations

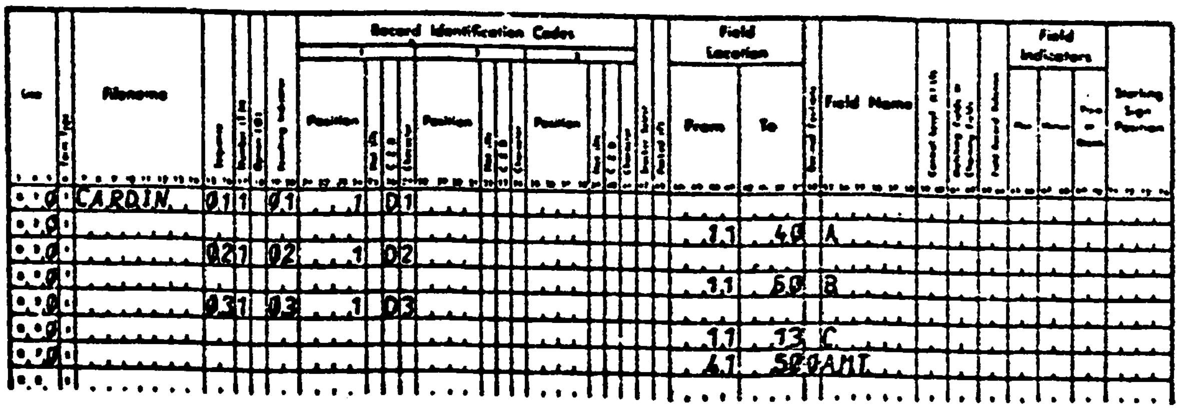
File Description Specifications



```
0 1 0 F CARDIN  IP  F  80  80            MFCM1
0 2 0 F SEQDSK  OS  F 270  90            DISK11 1FSYS002 S            2
```

XTENT CONTROL STATEMENT
DISK REQUIRES STANDARD LABELS
DEVICE IS 2311 DISK DRIVE
DISK FILE FORMAT IS FIXED

Input Specifications

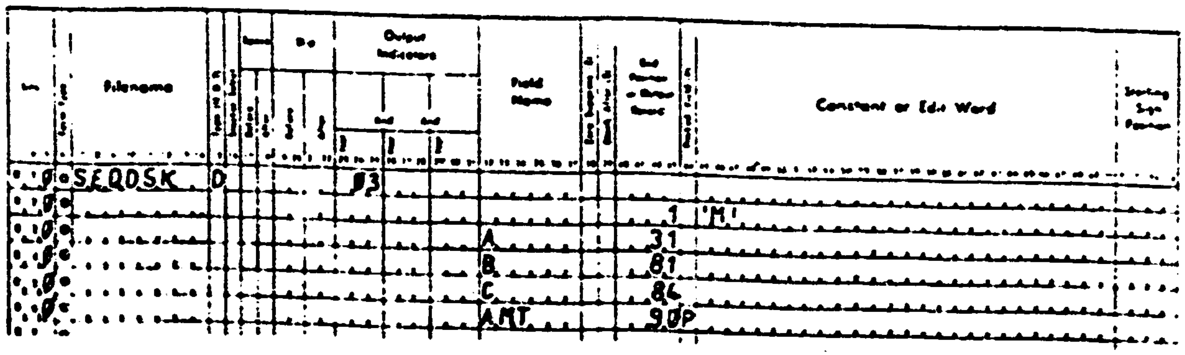


```
0 1 0 I CARDIN  01 1  01     1 D1
0 2 0 I                                      11  40 A
0 3 0 I         02 1  02     1 D2
0 4 0 I                                      11  60 B
0 5 0 I         03 1  03     1 D3
0 6 0 I                                      11  13 C
0 7 0 I                                      41  500AMT
```

Figure S-1. RPG Coding
Sequential File - Loading the File, Part 1 of 2

This RPG program extracts fields from types of cards and writes a sequential field of ninety character records per block. A numeric field is packed on a disk that was unpacked on cards. A program constant "M" is placed in the first byte of a record. The "M" is a constant which is not contained in data cards.

Output-Format Specifications

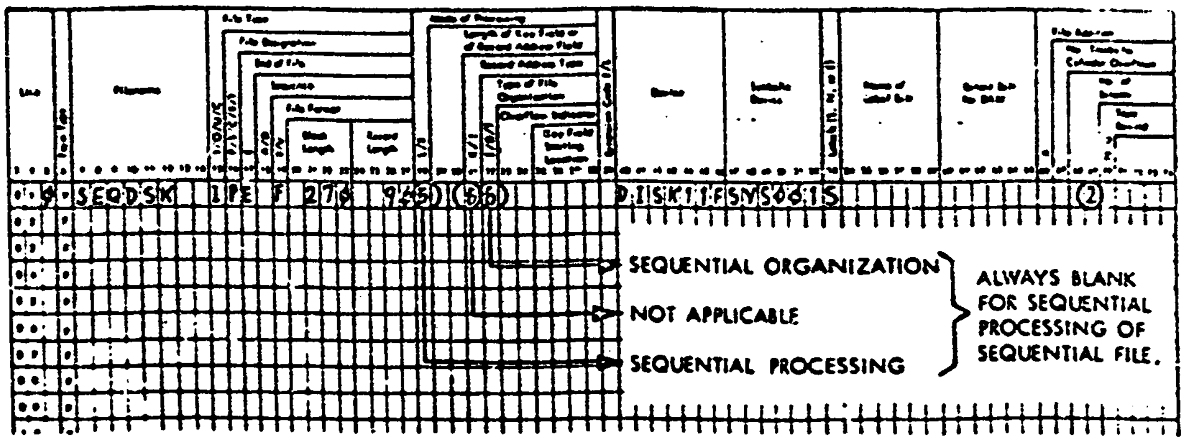


Line	Filename	Type	Output Indicators	Field Name	End Position in Output Record	Constant or Edit Word
Ø	SEQDSK	D	Ø3			
Ø					1	'M'
Ø				A	31	
Ø				B	81	
Ø				C	86	
Ø				AMT	9ØP	

Figure S-1 (continued). RPG Coding
Sequential File - Loading the File, Part 2 of 1

SEQUENTIAL FILE - SEQUENTIAL RETRIEVAL

File Description Specifications



```
9 SEQDSK  IPE F 276  945)(56)      DISK11FSYS061S                  (2)
```

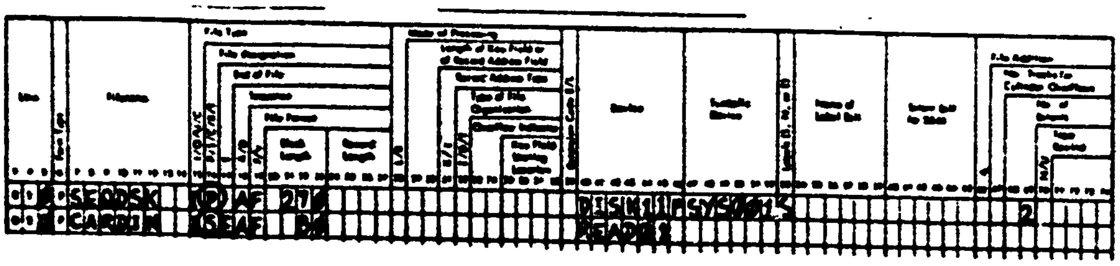
SEQUENTIAL ORGANIZATION
NOT APPLICABLE
SEQUENTIAL PROCESSING
ALWAYS BLANK FOR SEQUENTIAL PROCESSING OF SEQUENTIAL FILE.

Figure S-2 Sequential File - Sequential Retrieval

Sequential Retrieval is accomplished by letting column 28, column 31 and column 32 remain blank, but the number of extents as indicated by column 68-69 must agree with the number assigned when the file was written.

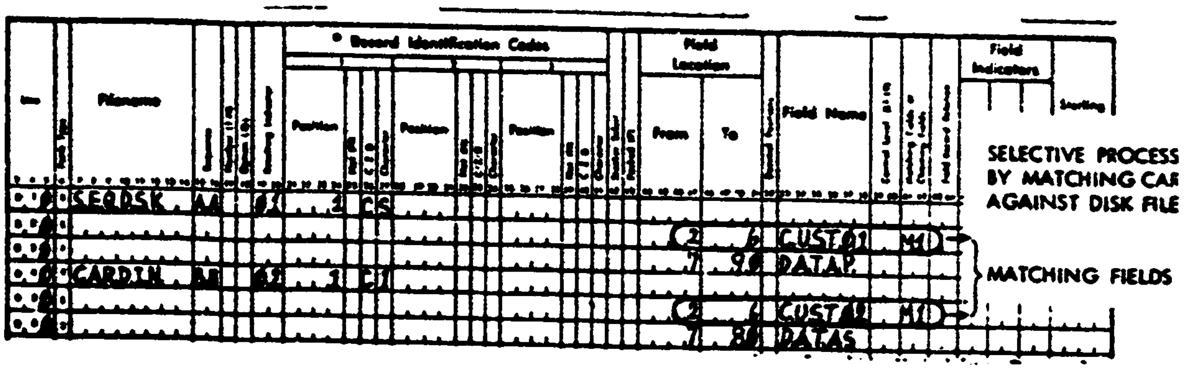
SEQUENTIAL FILE - SEQUENTIAL RETRIEVAL WITH MATCHING RECORD LOGIC

File Description Specifications



```
SEQDSK  IP AF  270      DISK11 SYS001 S        2
CARDIN  ISEAF   80      READ01
```

Input Specifications



```
SEQDSK  AA  01  1 CS
                        2   6 CUST01  M1
                        7  90 DATAP
CARDIN  BB  02  1 C1
                        2   6 CUST02  M1
                        7  80 DATAS
```

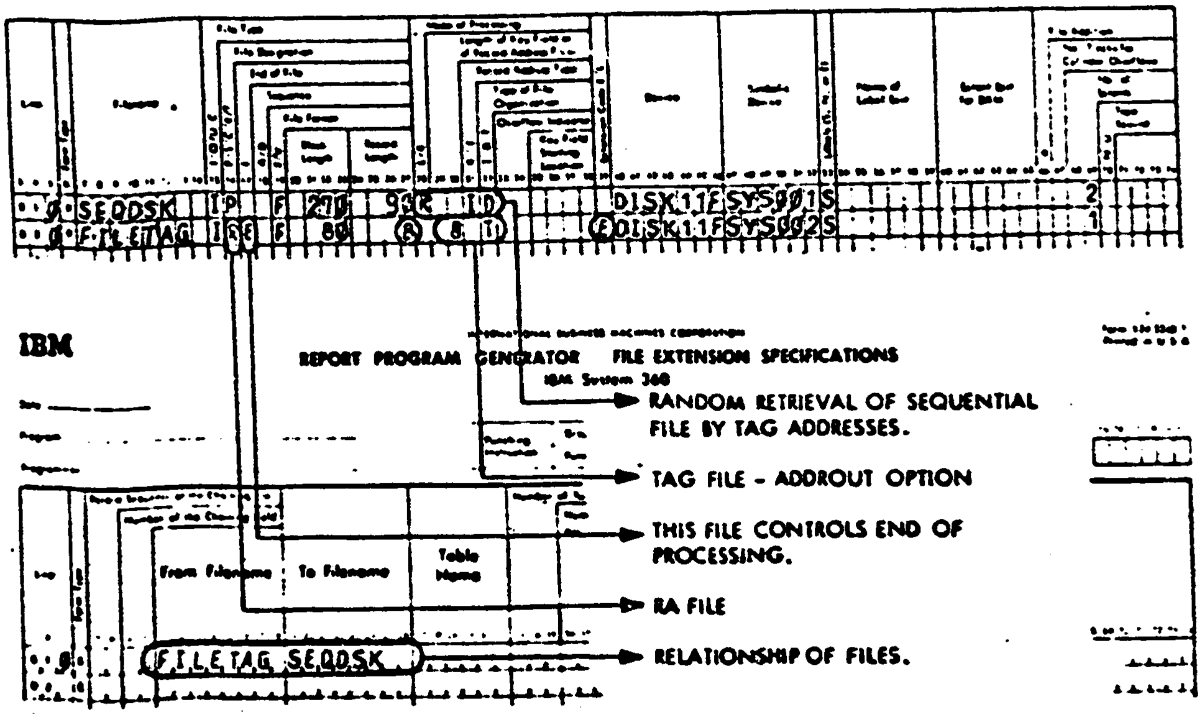
SELECTIVE PROCESS BY MATCHING CAR AGAINST DISK FILE
MATCHING FIELDS

Figure S-3. RPG Coding Sequential File Sequential Retrieval with Matching Record Logic

This permits selective processing of from two through nine input files. The files must be in the same collating sequence, but may be on any ˙nput device. The matching fields are assigned on the Input Specifications in columns 61 and 62 as matching fields M1 through M9. The files are processed first by selecting them in the order they are entered on the Input Specifications. The internal matching record indicator controls the processing of matching fields. The MR is tested on the calculation specification and processing is modified by its setting.

SEQUENTIAL FILE - RANDOM RETRIEVAL WITH ADDROUT OPTION

File Description Specifications


IBM
INTERNATIONAL BUSINESS MACHINES CORPORATION
REPORT PROGRAM GENERATOR FILE EXTENSION SPECIFICATIONS
IBM System 360
From Filename
To Filename
Table Name

```
F SEQDSK  IP  F 270  93R  11D    DISK11F SYS001S  2
F FILETAG IRE F  80    R   8 T  EDISK11F SYS002S  1
```

```
E FILETAG SEQDSK
```

RANDOM RETRIEVAL OF SEQUENTIAL FILE BY TAG ADDRESSES.

TAG FILE - ADDROUT OPTION

THIS FILE CONTROLS END OF PROCESSING.

RA FILE

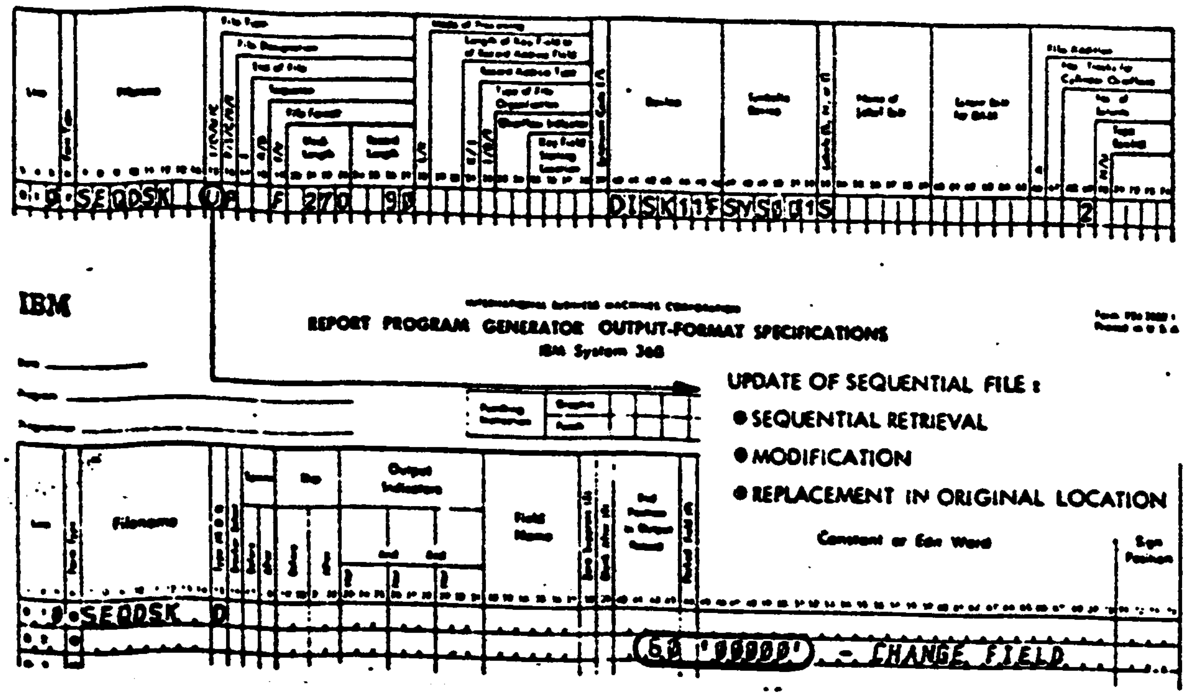
RELATIONSHIP OF FILES.

Figure S-4. Sequential File

Random Retrieval with ADDROUT Option

SEQUENTIAL FILE - UPDATING THE FILE

File Description Specifications



SEQDSK UP F 270 90 DISK11FSYS011S 2

SEQDSK D

58 '00000' - CHANGE FIELD

UPDATE OF SEQUENTIAL FILE :

- SEQUENTIAL RETRIEVAL
- MODIFICATION
- REPLACEMENT IN ORIGINAL LOCATION

IBM
REPORT PROGRAM GENERATOR OUTPUT-FORMAT SPECIFICATIONS
IBM System 360
Filename
Output Indicators
Field Name
Constant or Edit Word

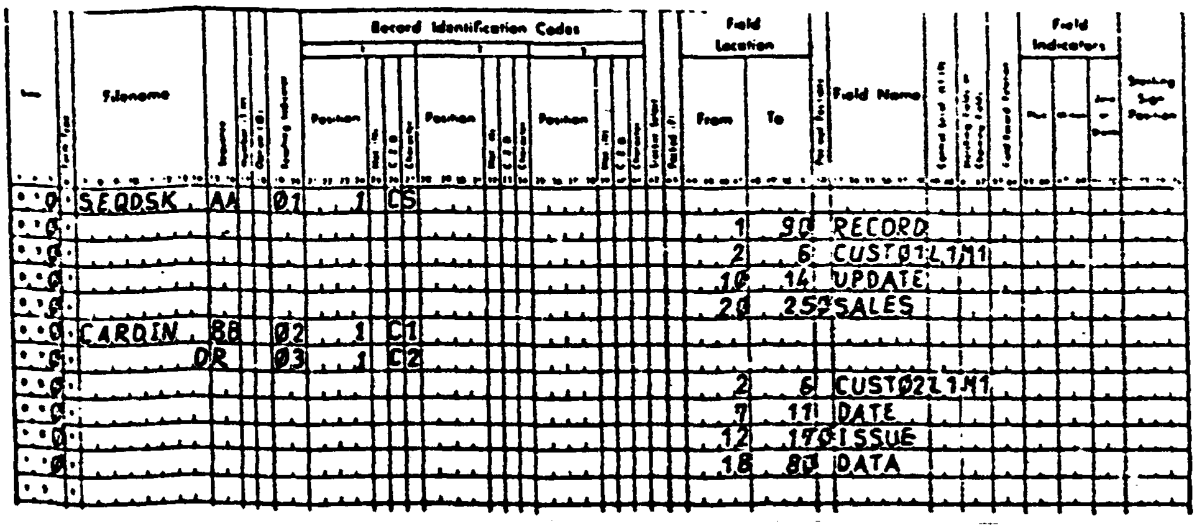
Figure S-5

Sequential File - Updating the File

It is not necessary to require the whole file to be updated if the record lengths are not modified, however if the length is to be modified, the entire file must be copied to an output-file.

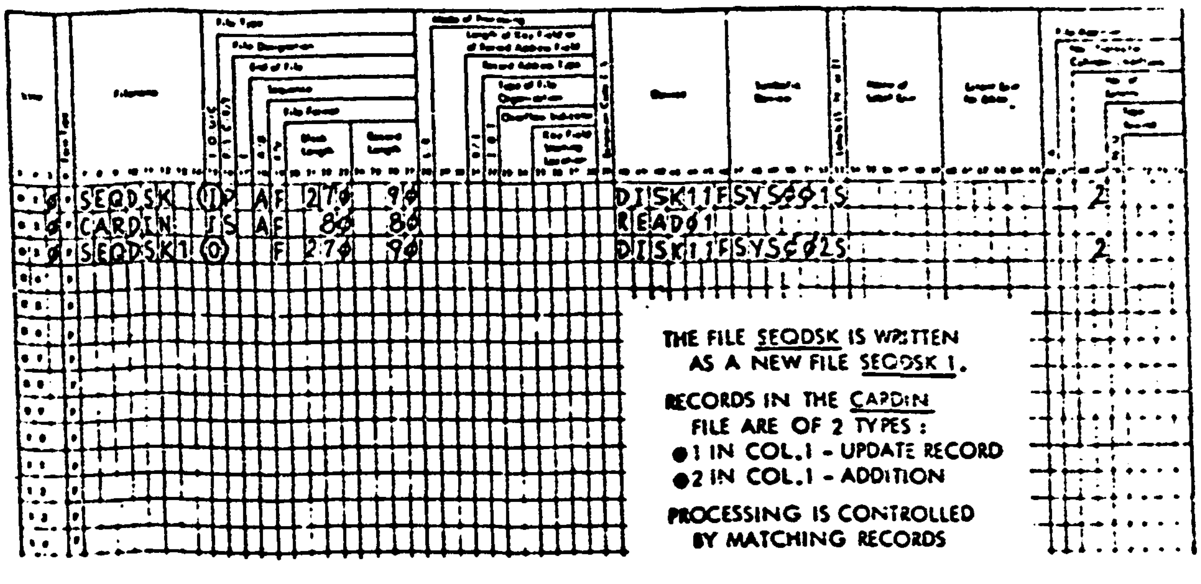
SEQUENTIAL FILE - UPDATING AND ADDING RECORDS TO THE FILE (2 pages)

File Description Specifications



		Record Identification Codes			Field Location		
Line	Filename	Sequence	Resulting Indicator	Position	C/Z/D	Character	From	To	Field Name
Ø	SEQDSK	AA	Ø1	1	C	S			
Ø							1	9Ø	RECORD
Ø							2	6	CUSTØ1L1M1
Ø							1Ø	14	UPDATE
Ø							2Ø	25Ø	SALES
Ø	CARDIN	BB	Ø2	1	C	1			
Ø	OR		Ø3	1	C	2			
Ø							2	6	CUSTØ2L1M1
Ø							7	11	DATE
Ø							12	17Ø	ISSUE
Ø							18	8Ø	DATA

Input Specifications



Line	Filename	File Type	File Designation	End of File	Sequence	File Format	Block Length	Record Length	Device	Symbolic Device	
ø	SEQDSK	I	P		A	F	27ø	9ø	DISK11F	SYSøø1S	2
ø	CARDIN	I	S		A	F	8ø	8ø	READø1		
ø	SEQDSK1	O				F	27ø	9ø	DISK11F	SYSøø2S	2

THE FILE SEQDSK IS WRITTEN
AS A NEW FILE SEQDSK 1.

RECORDS IN THE CARDIN
FILE ARE OF 2 TYPES :

- ●1 IN COL.1 - UPDATE RECORD
- ●2 IN COL.1 - ADDITION

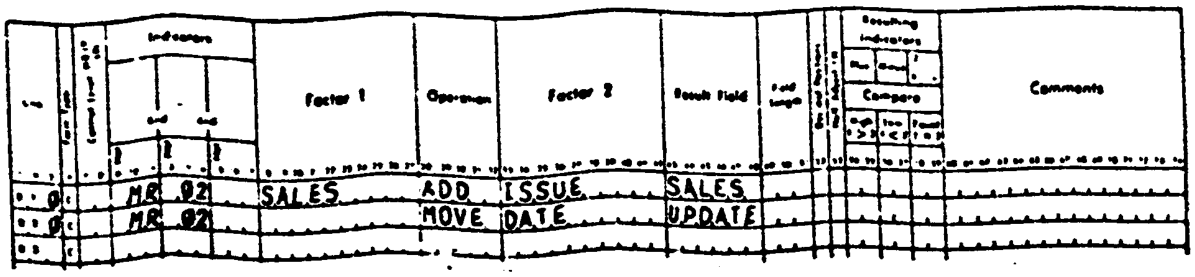
PROCESSING IS CONTROLLED
BY MATCHING RECORDS

Figure S-6, Sequential File

Updating and Adding Records to the File, Part 1 of 2

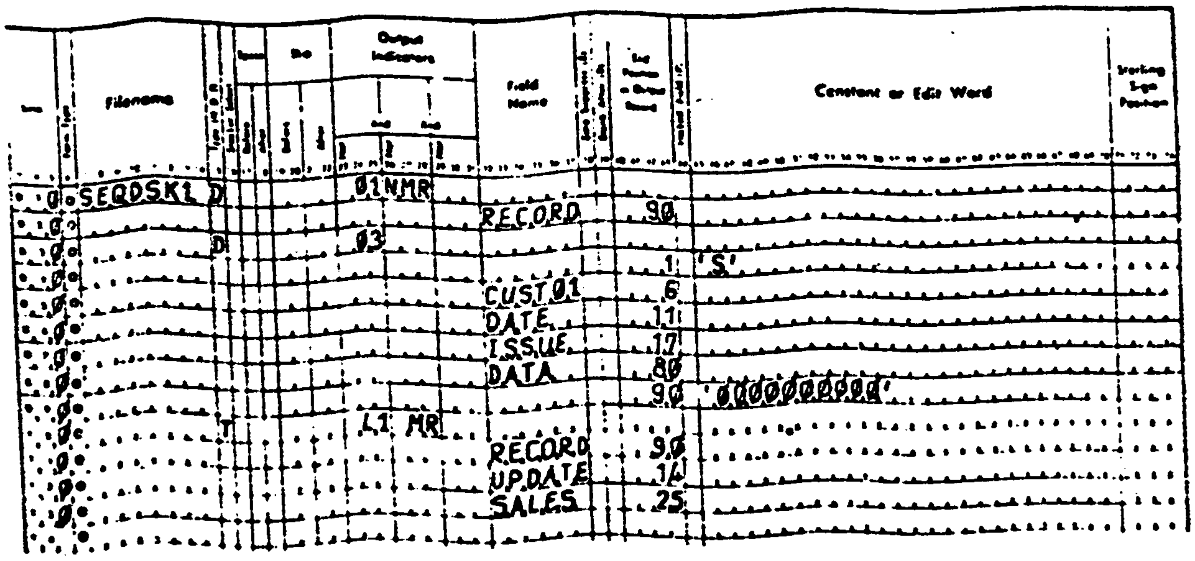
Updating and Adding Records to the File (con't.)

Calculation Specifications



		Indicators	Factor 1	Operation	Factor 2	Result Field	Field Length	Comments
Ø	C	MR 02	SALES	ADD	ISSUE	SALES		
Ø	C	MR 02		MOVE	DATE	UPDATE		
	C							

Output-Format Specifications



		Filename	Type	Output Indicators	Field Name	End Position in Output Record	Constant or Edit Word	Sterling Sign Position
Ø	O	SEQDSK1	D	Ø1NMR				
Ø	O				RECORD	9Ø		
Ø	O		D	Ø3				
Ø	O					1	'S'	
Ø	O				CUSTØ1	6		
Ø	O				DATE	11		
Ø	O				ISSUE	17		
Ø	O				DATA	8Ø		
Ø	O					9Ø	'ØØØØØØØØØØ'	
Ø	O		T	L1 MR				
Ø	O				RECORD	9Ø		
Ø	O				UPDATE	14		
Ø	O				SALES	25		

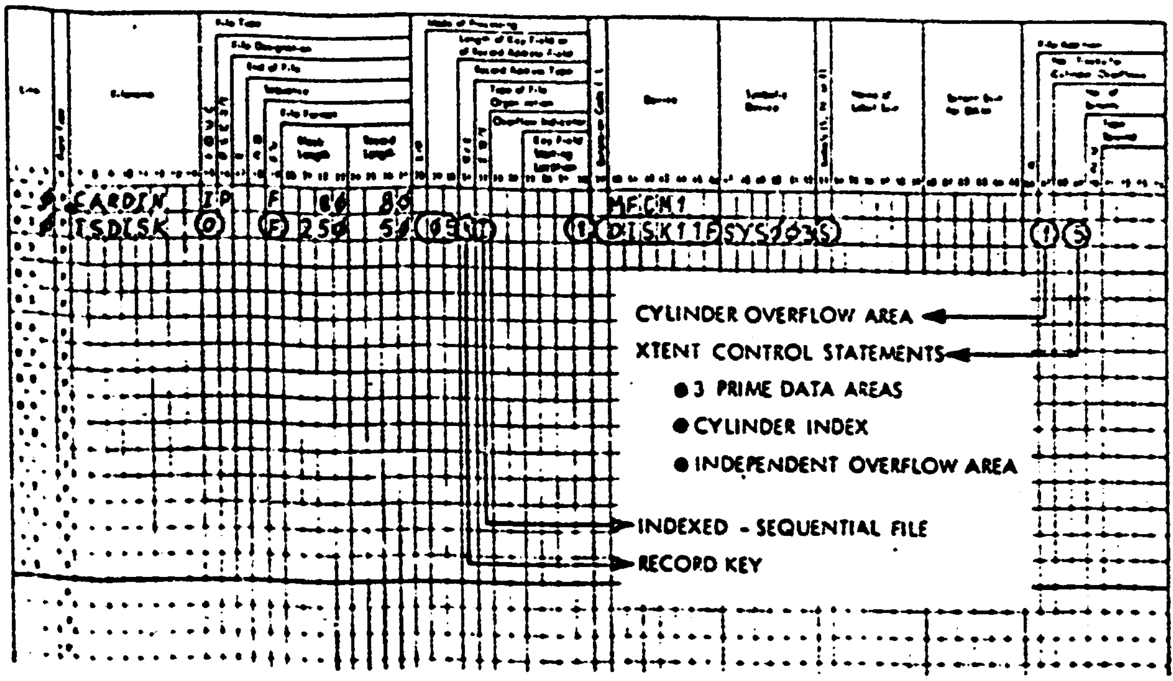
Figure S-7. RPG Coding. Sequential File

Updating and Adding Records to the File. Part 2 of 2

The File must be rewritten, but records can be updated during the same run. The files are specified as input and output, not as update, since update files must not include additions and deletions.

II INDEXED SEQUENTIAL FILE

File Description Specifications



```
CARDIN  IP  F  80  80                        MFCM1
ISDISK  O   F  250 50 10 5 KI   11  DISK11 F SYS063 S      1 S
```

CYLINDER OVERFLOW AREA
XTENT CONTROL STATEMENTS
● 3 PRIME DATA AREAS
● CYLINDER INDEX
● INDEPENDENT OVERFLOW AREA
INDEXED - SEQUENTIAL FILE
RECORD KEY

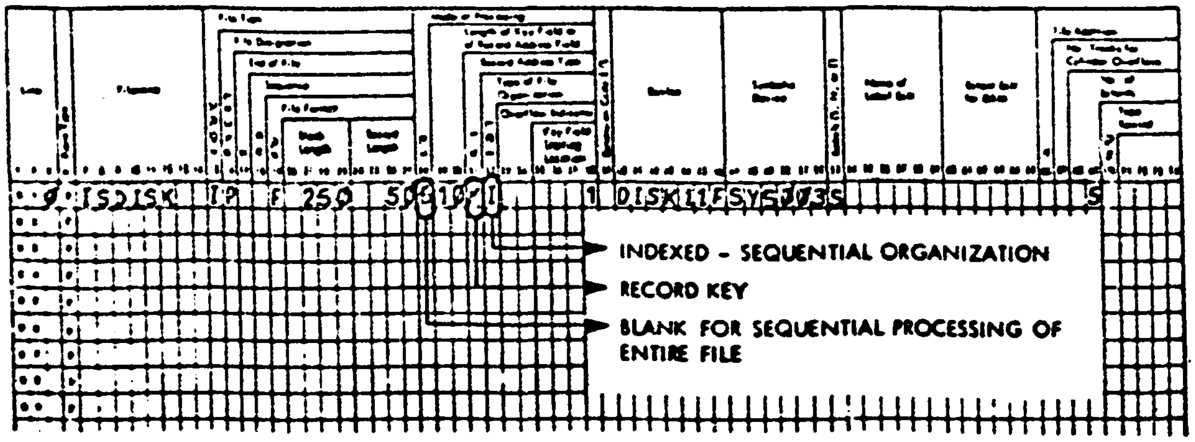
Figure S-8. RPG Coding

Indexed-Sequential File - Loading the File

Records are fixed length blocked or unblocked. As each file is written, indexes are created to provide a reference to records on each track or cylinder. The file is an output file designated by a 0 in column 15. There is a K in column 31 and an 1 in column 32. Columns 29-30 and 35-38 specify the length and location of the key field. There are two additional extents for the prime data area and one extent for the independent overflow area entered in columns 68-69. The user provides information about these with an extent control statement to the Job Control program when the load operation is enacted.

INDEXED SEQUENTIAL FILE - SEQUENTIAL RETRIEVAL

File Description Specifications



```
0 F ISDISK    IP  F 250  50 10KI     1 DISK11FSYS003S              S
```

INDEXED - SEQUENTIAL ORGANIZATION
RECORD KEY
BLANK FOR SEQUENTIAL PROCESSING OF ENTIRE FILE

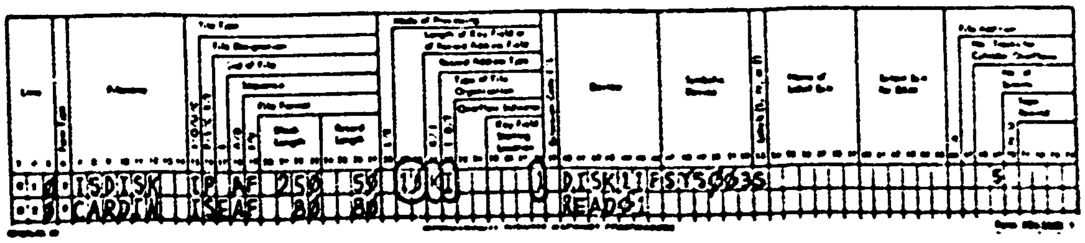
Figure S-9. RPG

Indexed-Sequential File - Sequential Retrieval

Retrieval is similar to retrieval of sequential files, and additions written in overflow areas are retrieved in sequential order. When multiple input files are specified, the order of processing is determined by the sequence they are entered on the Input Specifications. Coding follows the basic pattern for coding a disk file, plus a K in column 31 and an I in column 32. This indicates indexed-sequential organization. Column 28 is blank, and causes the entire file to be processed.

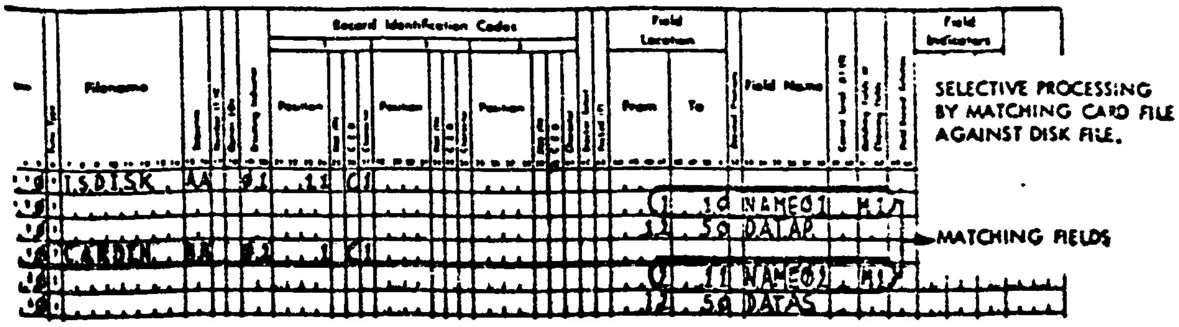
INDEXED SEQUENTIAL FILE - SEQUENTIAL RETRIEVAL
WITH MATCHING RECORD LOGIC

File Description Specifications



```
0 F ISDISK    IP AF 250  50 10KI     1 DISK11FSYS003S              S
1 F CARDIN    ISEAF  80  80            READ01
```

Input Specifications


Filename
Record Identification Codes
Position
Position
Position
Field Location
From
To
Field Name
Field Indicators

```
ISDISK  AA  01    11 C1
                              1   10 NAME01  M1
                             12   50 DATAP
CARDIN  BB  02     1 C1
                              1   11 NAME02  M1
                             12   50 DATAS
```

SELECTIVE PROCESSING BY MATCHING CARD FILE AGAINST DISK FILE.

MATCHING FIELDS

Figure S-10. RPG Coding. Indexed-Sequential File Sequential Retrieval with Matching Record Logic

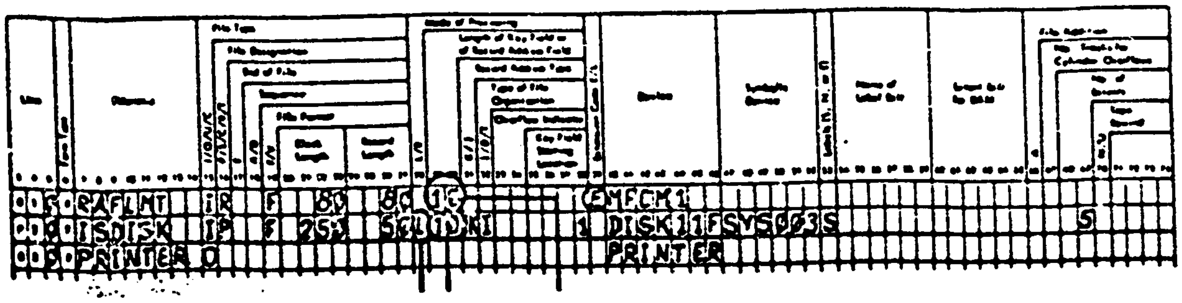
INDEXED SEQUENTIAL FILE - SEQUENTIAL RETRIEVAL WITH AN RA FILE OF LIMITS

The RA (Record Address) is used to retrieve a segment of an indexed sequential file. The RA File defines the low and high limits of data to be processed. Two keys are contained in each record of the RA File (the lower limit and the upper limit). The record with a key equal to, or higher than, the lower limit is retrieved; all others above it follow in sequence up to, or equal to, the upper limit.

Only "keys" are contained in the RA file, and their purpose is to retrieve records. No reference is made to it on the Input Specifications so fields for update purposes can't be defined.

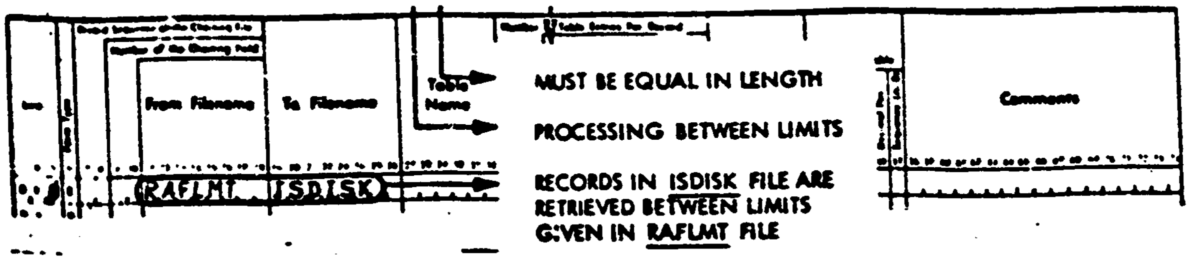
All information for the RA comes from the File Description and File Extension Specifications. An addition of a 1 in the "MODE OF PROCESSING" (column 28) of the File Extension Specifications tells that the RA File is related to the Indexed-Sequential File.

File Description Specifications



```
F RAFLMT  IR  F   8Ø  8Ø 1Ø            EMFCM1
F ISDISK  IP  F  25Ø  5ØL1ØAI         1 DISK11F SYS003S           5
F PRINTER O                              PRINTER
```

File Extension Specifications

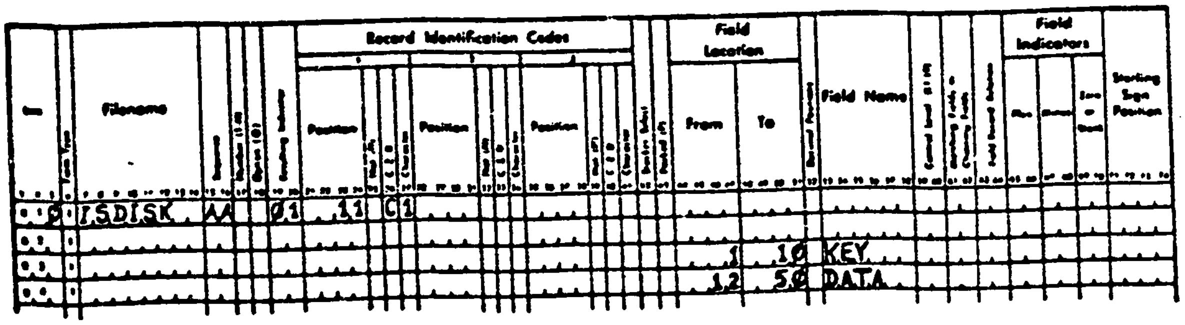


```
E      RAFLMT    ISDISK
```

Table Name
MUST BE EQUAL IN LENGTH
PROCESSING BETWEEN LIMITS
RECORDS IN ISDISK FILE ARE RETRIEVED BETWEEN LIMITS GIVEN IN RAFLMT FILE
Comments

Figure S-11. Indexed-Sequential File
Sequential Retrieval with RA File of Limits. Part 1

Input Specifications



```
I ISDISK  AA  Ø1   11 C1
I
I                                  1  1Ø KEY
I                                 12  5Ø DATA
```

Figure S-11. Indexed-Sequential File
Sequential with RA File of Limits. Part 2 of 2

INDEXED-SEQUENTIAL FILE - RANDOM
RETRIEVAL WITH CHAINING FILE

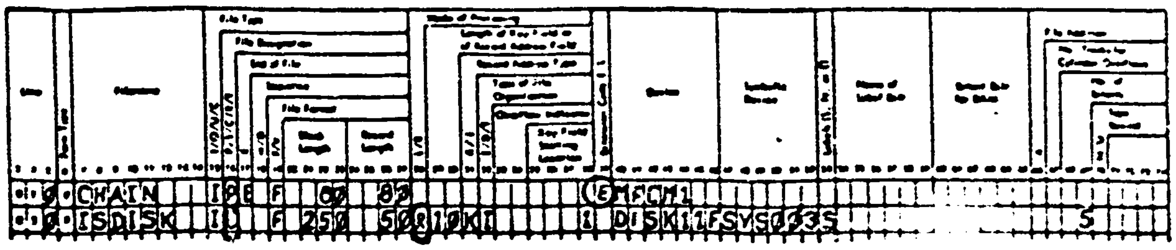
The "Chaining" technique is also used with the indexed sequential file. This method of random retrieval uses keys or chaining fields. They also serve as the link between two files. Each record in a chaining file contains the key of a record to be randomly retrieved from the chained file, which must be organized as an indexed-sequential file.

See Figure S-12. Chaining is encountered, in addition to the entries for the indexed sequential file:

1. AC is included in the File Description (column 16) of the File Description.
2. An R is placed in Mode of Processing (column 28).
3. An I is placed in Extension Code (column 39), to reference the File Extension Specifications. It relates the file as the chaining file (columns 11 to 18) to the disk file, which is the chained file (columns 19-26) by the chaining field of C1 (columns 9-10).

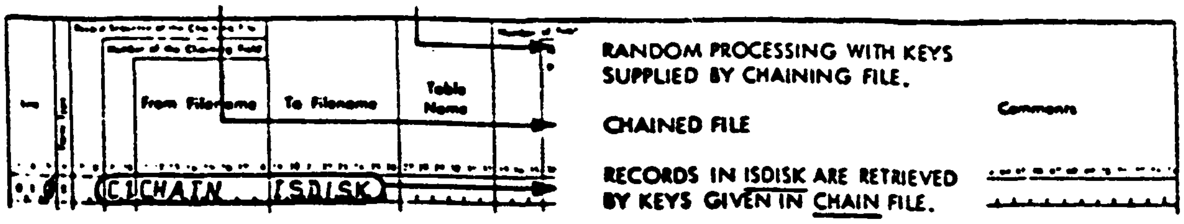
Indexed-Sequential File
Random Retrieval with Chaining File (con't.)

File Description Specifications



```
F CHAIN    IPE F   8Ø   8Ø                E MFCM1
F ISDISK   IC  F  25Ø   5ØR1ØKI           I DISK11FSYSØØ3S                 S
```

File Extension Specifications



		From Filename	To Filename	Table Name	Comments
ø	E	C1CHAIN	ISDISK		

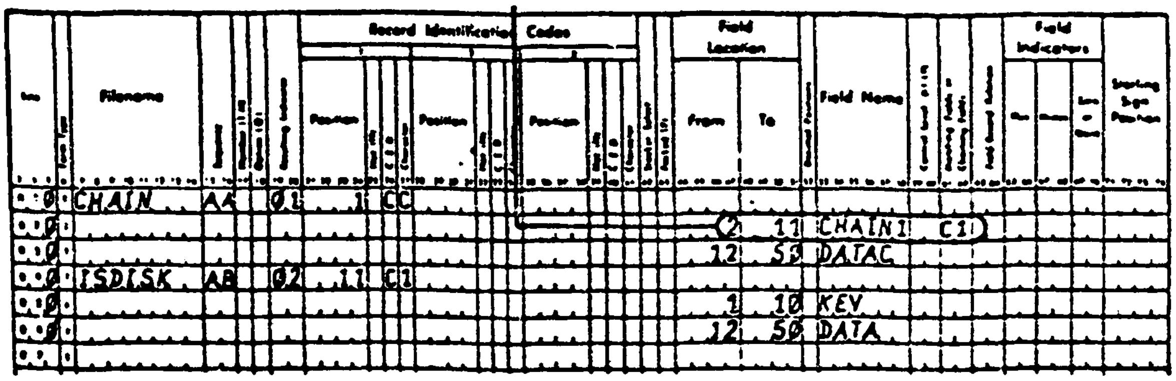
RANDOM PROCESSING WITH KEYS SUPPLIED BY CHAINING FILE.

CHAINED FILE

RECORDS IN ISDISK ARE RETRIEVED BY KEYS GIVEN IN CHAIN FILE.

Key or Chaining Field

Input Specifications

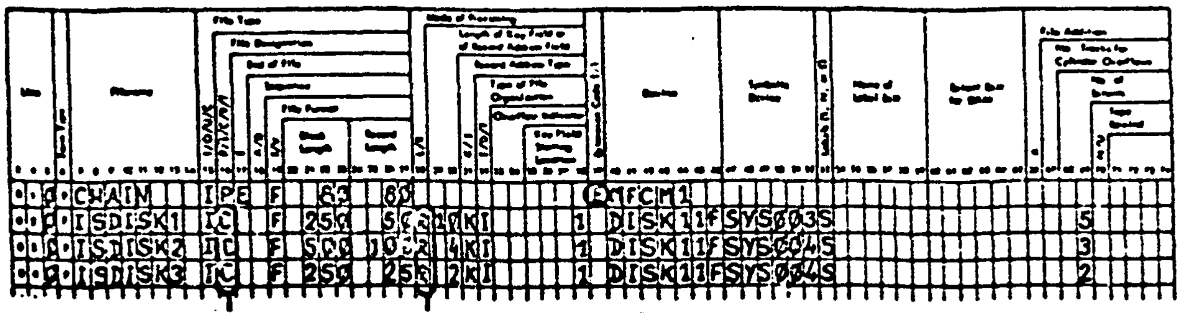


		Filename	Sequence	Resulting Indicator	Position	C/Z/D	Character	From	To	Field Name	Chaining Fields
ø	I	CHAIN	AA	ø1	1	C	C				
ø	I							2	11	CHAIN1	C1
ø	I							12	53	DATAC	
ø	I	ISDISK	AB	ø2	11	C	1				
ø	I							1	1ø	KEY	
ø	I							12	5ø	DATA	

Figure S-12. Indexed-Sequential File Random Retrieval with Chaining File

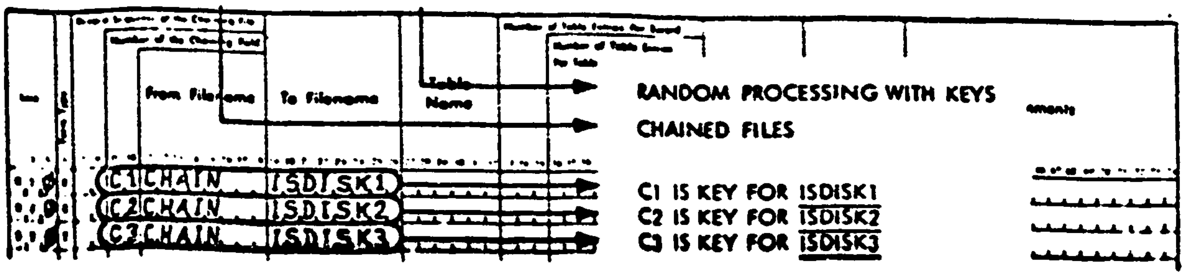
INDEXED-SEQUENTIAL FILE - RANDOM RETRIEVAL WITH ONE CHAINING FILE FROM THREE CHAINED FILES

File Description Specifications



		Filename	File Type	File Format	Block Length	Record Length	Mode of Processing	Length of Key Field	Record Address Type	Type of File Organization	Key Field Starting Location	Extension Code	Device	Symbolic Device	Labels	Number of Tracks for Cylinder Overflow
ø	F	CHAIN	IPE	F	8ø	8ø					1	E	MFCM1			
ø	F	ISDISK1	IC	F	25ø	5ø	R	1ø	K	I	1		DISK11F	SYSø03	S	5
ø	F	ISDISK2	IC	F	5øø	1øø	R	4	K	I	1		DISK11F	SYSø04	S	3
ø	F	ISDISK3	IC	F	25ø	25	R	2	K	I	1		DISK11F	SYSø04	S	2

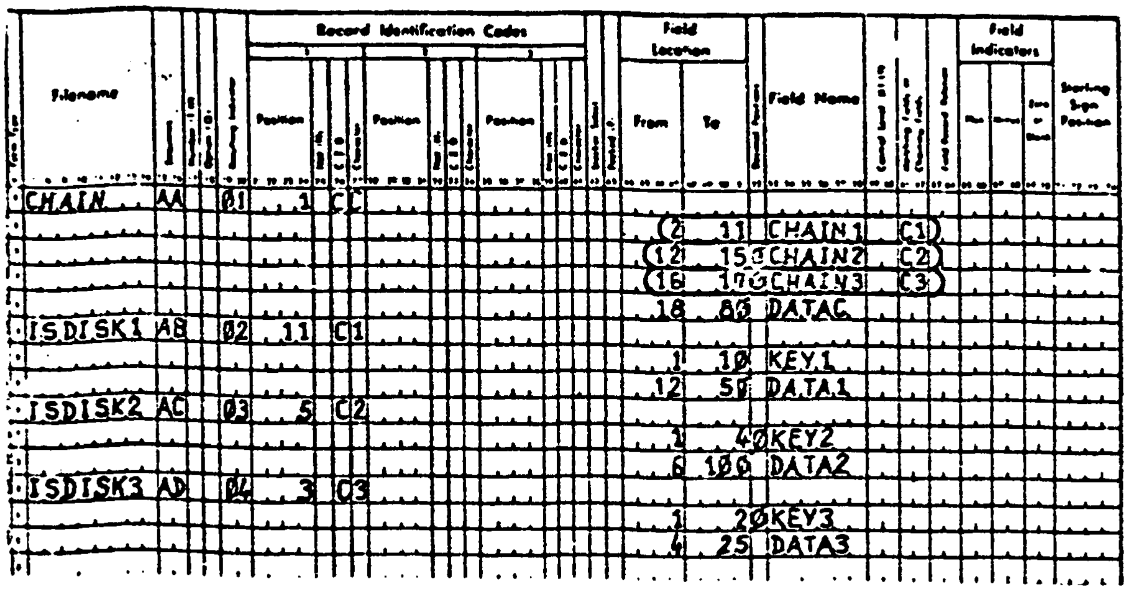
File Extension Specifications



From Filename	To Filename	
C1CHAIN	ISDISK1	C1 IS KEY FOR ISDISK1
C2CHAIN	ISDISK2	C2 IS KEY FOR ISDISK2
C3CHAIN	ISDISK3	C3 IS KEY FOR ISDISK3

RANDOM PROCESSING WITH KEYS
CHAINED FILES

Input Specifications



Filename		Resulting Indicator	Position		From	To	Field Name	Chaining Fields
CHAIN	AA	Ø1	1	C				
					2	11	CHAIN1	C1
					12	15	CHAIN2	C2
					16	17	CHAIN3	C3
					18	85	DATAC	
ISDISK1	AB	Ø2	11	C1				
					1	1Ø	KEY1	
					12	5Ø	DATA1	
ISDISK2	AC	Ø3	5	C2				
					1	4Ø	KEY2	
					6	1ØØ	DATA2	
ISDISK3	AD	Ø4	3	C3				
					1	2Ø	KEY3	
					4	25	DATA3	

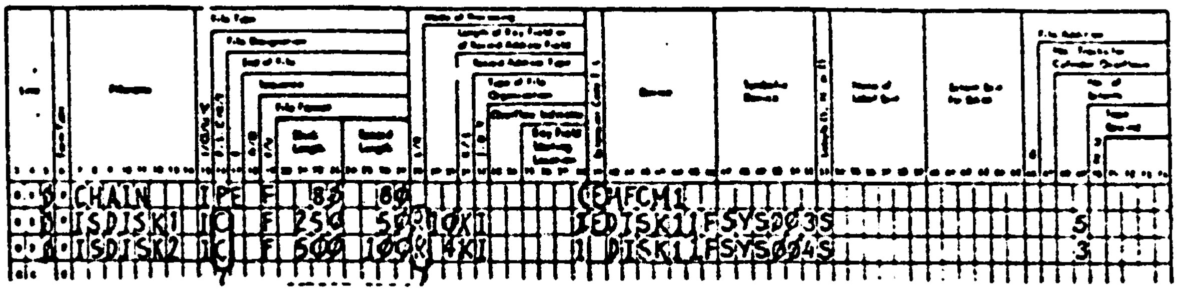
Figure S-13. RPG Coding
Indexed-Sequential File - Random Retrieval with One
Chaining File from Three Chained Files

As depicted by Figure S-13, another method of using the chaining technique is to retrieve records from a chained file. Then use a field in those records to link to another file. The chained file then becomes a chaining file to the second chained file.

The chained file is unique type of RPG file. Records in this file are not retrieved until total time in the RPG processing cycle, while other files are read before total time. As a result, a chained file cannot contain control fields or matching fields. It is the only file which can be updated at total time.

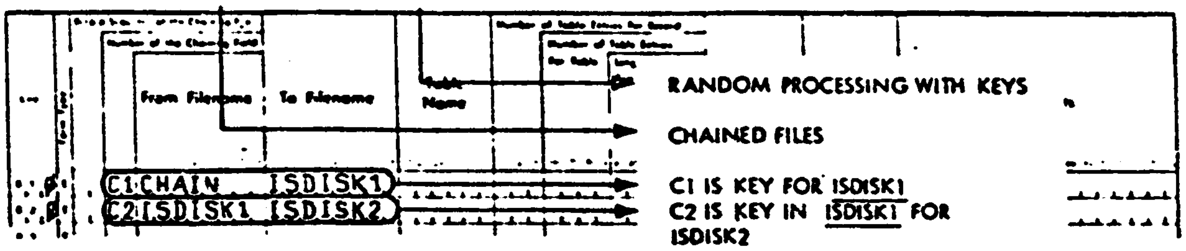
INDEXED-SEQUENTIAL FILE - RANDOM RETRIEVAL WITH CHAINED FILE AS A CHAINING FILE

File Description Specifications



```
CHAIN    IPE F  85  8Ø          C MFCM1
ISDISK1  IC  F 25Ø  5ØR1ØKI     IEDISK11FSYSØØ3S     5
ISDISK2  IC  F 5ØØ 1ØØR 4KI     I DISK11FSYSØØ4S     3
```

File Extension Specifications



From Filename
To Filename
Table Name

RANDOM PROCESSING WITH KEYS

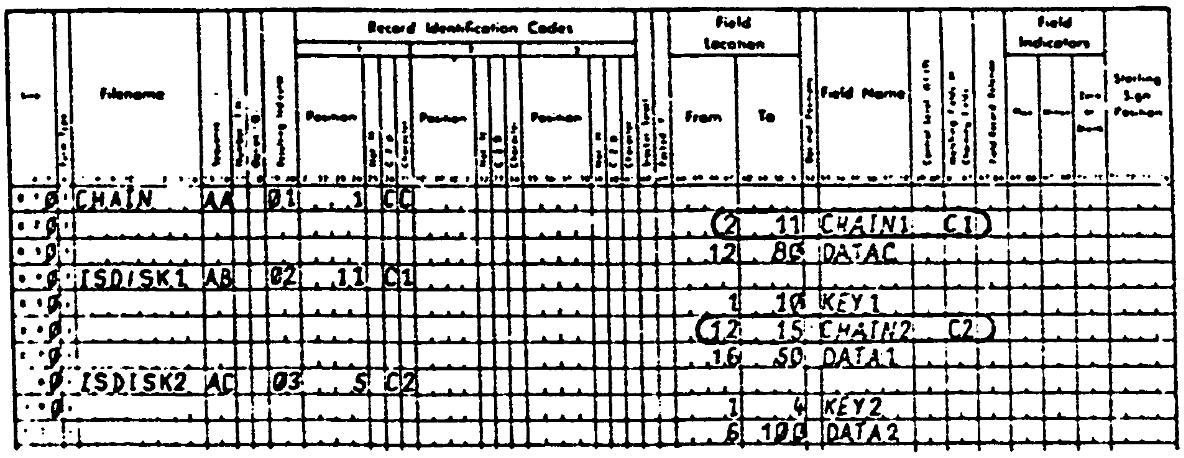
CHAINED FILES

```
C1CHAIN    ISDISK1
C2ISDISK1  ISDISK2
```

C1 IS KEY FOR ISDISK1

C2 IS KEY IN ISDISK1 FOR ISDISK2

Input Specifications



Record Identification Codes
Filename
Field Location
From
To
Field Name
Field Indicators

```
Ø CHAIN    AA Ø1   1 CC
Ø                            2   11 CHAIN1   C1
Ø                           12   8Ø DATAC
Ø ISDISK1  AB Ø2  11 C1
Ø                            1   1Ø KEY1
Ø                           12   15 CHAIN2   C2
Ø                           16   5Ø DATA1
Ø ISDISK2  AC Ø3   5 C2
Ø                            1    4 KEY2
                             6  1ØØ DATA2
```

Figure S-14. RPG Coding. Indexed-Sequential File Random Retrieval with Chained File as a Chaining File

ACTIVITIES

The final activities section presents a culmination of the book. The material was presented by models and methods for accomplishing these models.

If the reader will review Section I and the Preface, he will notice that this book could be used for a complete year.

This (year) can be easily accomplished by going back to the Swanson Study, and writing programs for the functional areas of the corporation.

It is suggested that the reader program for Financial Control first. This includes:

1. General Ledger and Budget Accounting

   This activity reports all income, expenses and budget information to Management Planning.

2. Accounts Receivable

   This activity accounts for all customer receivable information resulting from sales. Here customer records are maintained, reflecting credit status, statement of customer accounts, and customer receipts. The income is reported to General Ledger Accounting.

3. Cost Accounting

   This activity accounts for and report; costs of production which includes material costs and labor costs.

4. Labor and Payroll

   This activity accounts for payroll for all employees. A report is also made to General Ledger for payroll and labor expenses. Cost Accounting also receives a report based on labor tickets from Production Planning and Control.

5. Accounts Payable

This activity accounts for, and reports payments for, goods and services received. Expenses are reported to General Ledger Accounting.

ADD REGISTER

AR R_L'R_2 [RR]

1A	R_1	R_2
0 7	8 11	12 15

The second operand is added to the first operand, and the sum is placed in the first operand.

Both operands and the sum are 16-bit integers.

Condition Code:

0 Sum is zero

1 Sum is less than zero

2 Sum is greater than zero

AR10,11

ADD HALFWORD

AH R_1,D_2(X_2,B_2) [RX]

4A	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

The halfword second operand is added to the first operand, and the sum is placed in the first operand (register).

1. The second operand is a 16-bit signed integer.
2. The operand must be on a halfword integral boundry.
3. The first operand is a 16-bit signed integer.
4. The sum is a 16-bit signed integer.

Condition Code:

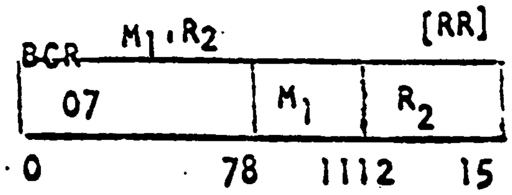
0 Sum is zero

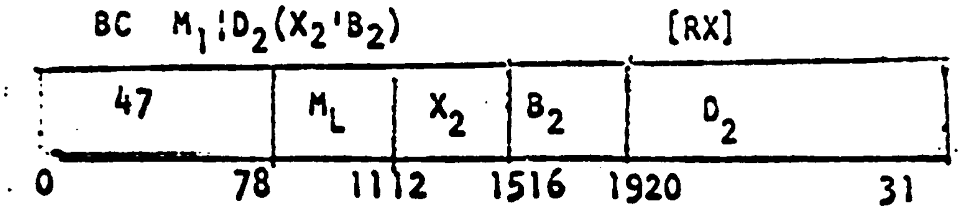
1 Sum is less than zero

2 Sum is greater than zero

```
AH 8,HALF
```

BRANCH ON CONDITION





A branch to the address specified in the second operand is taken whenever the condition code matches a condition specified in the first operand (M).

To code this instruction:

1. Place the mask value corresponding to the desired condition 1. Place the mask value corresponding to the desired condition code in the first operand.

Condition Code	0	1	2	3
Mask Value	8	4	2	1

Example:

A. Desired Condition Code is 1

B. Coding is BC4, TOP1

2. To test for more than one condition code, place the sum of the mask values corresponding to the desired condition codes in the first operand.

Example:

A. Desired condition codes are 0 and 2.

B. Coding is BC10, BRANCH

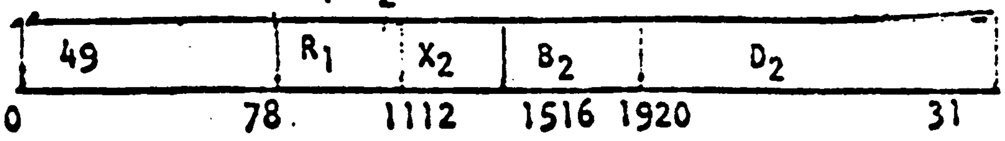
Note: Either condition code 0 or condition code 2 will cause a branch.

3. When the first operand is 15, the branch is always taken (unconditional branch).

4. When the first operand is 0, no branch is taken (a no-operation [no-op] equivalent).

COMPARE HALFWORD

CH $R_1,D_2(X_2,B_2)$ [RX]



The first operand is compared algebraically with the halfword second operand, and the result determines the setting of the condition code.

1. Both operands are 16-bit signed integers.

2. The second operand must be on a halfword integral boundry.

Condition Code:

0 operands are equal

1 First operand is low

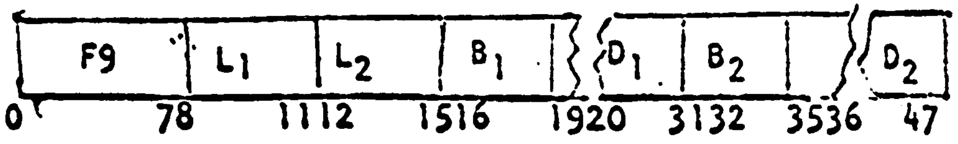
2 First operand is high

3 --

```
CH 8,LIMT
```

COMPARE DECIMAL

CE $D_1(L_1B_1)$ '$D_2(L_2 \cdot B_2)$ [SS]



The first operand is compared algebraically with the second operand, and the result determines the setting of the condition code.

1. Both operands are in packed decimal format.

2. Comparison proceeds from right to left taking into account the sign as well as all the digits of each field.

3. The operand 2 field must not be longer than the operans 1 high-order zeros.

4. Plus zero and minus zero compare equal.

Condition Code:

0 Operands are equal

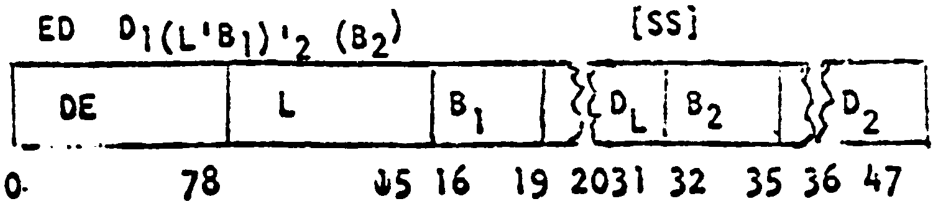
1 First operand is low

2 First operand is high

3 —

```
CP YTDF,AMT
BC 0,NOTX
```

EDIT



The format of the second (source) is changed from packed to zoned and is edited into the first operand.

1. The second operand (source) must be in packed format.

2. Editing proceeds left to right, one character at a time.

3. The edited result replaces the pattern.

Condition Code:

0 Result is zero.

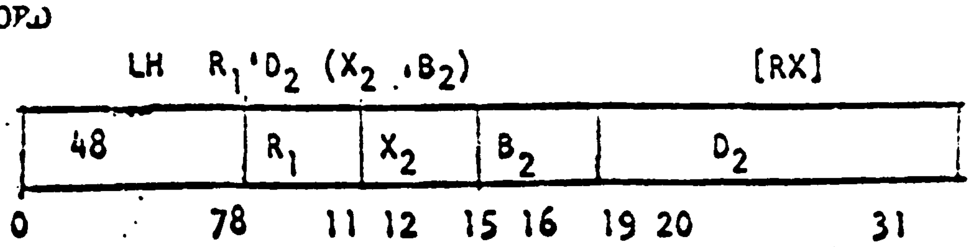
1 Result is less than zero.

2 Result is greater than zero.

3 —

```
ED OLPR-1(19), PRIN
```

LOAD HALFWORD



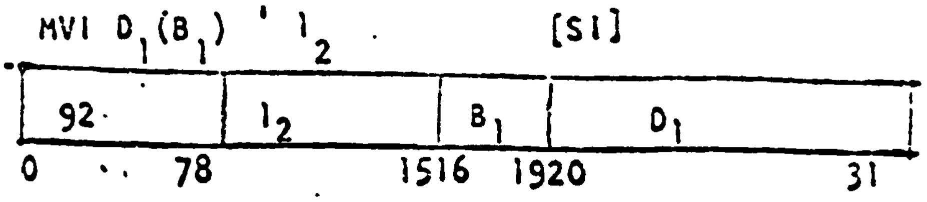
The halfword second operand must be on a halfword integral boundry.

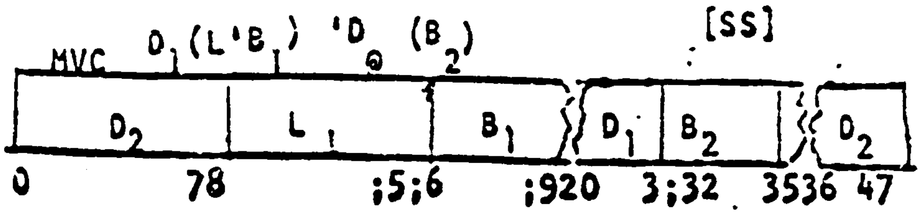
Condition Code:

The code remains unchanged.

```
LH 11,HALF
```

MOVE





The second operand is placed in the first operand location.

MVC (Move Characters)

1. The bytes are moved one at a time in each field.

2. Movement is left to right.

3. The number of bytes is determined by the implicit or explicit length of the first operand.

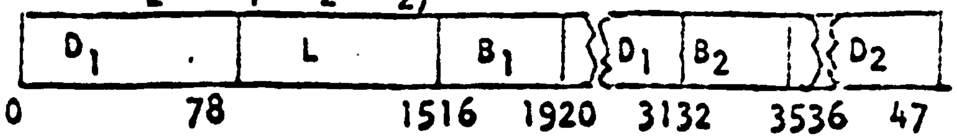
MVC (Move Immediate)

1. One byte of immediate data is stored in the first operand location. Immediate data supplied by the instruction itself, and in this case, is the second operand.

```
MV1 SW,X"01"

MV1 OLD,NEW
```

MOVE NUMBERICS

MVN $D_1(L'B_1)$ $D_2(B_2)$ [SS]



The numberic portion (low-order four bits) of each byte in the second operand are placed in the numberic portion of the corresponding bytes in the first operand.

1. The number of bytes in the operation is determined by the implicit or explicit length of the first operand.

2. Movement is from left to right through each field.

3. Movement is one byte at a time.

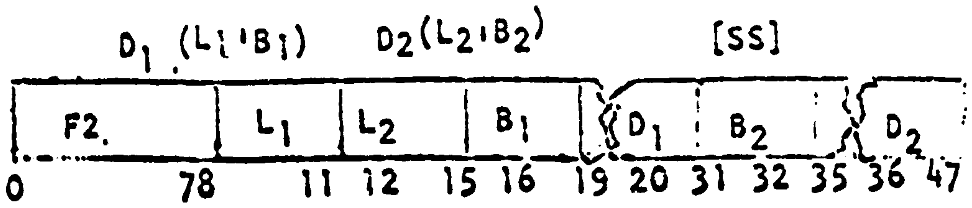
4. Zones remain unchanged.

Condition Code:

The code remains unchanged.

```
MVN PINT+2(1),PINT+4
```

PACK



The signed or unsigned number in the zoned format, in the second location is changed to packed format and stored in the first operand location.

1. The fields are processed from right to left.

2. If the first operand field is too long, it will be filled with high-order zeros.

3. If the first operand field is too short, any remaining high-order digits in the second operand will be ignored.

4. The maximum size of the second operand (zoned field) is 16 bytes.

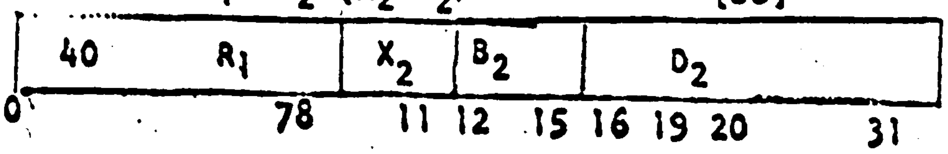
Condition Code:

The code remains unchanged.

```
PACK SGND,ZONE
```

STORE HALFWORD

STH R1, D2 (X2, B2) [SS]



The general register specified in the first operand is stored at the second operand halfword location.

The second operand must be on a halfword integral boundry.

Condition Code:

The code remains unchanged.

STH 8,BUTE

SUBTRACT HALFWORD

SH R_1 'D_2 (X_2'B_2) [RX]

48	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

The halfword second operand is subtracted from the register specified in the first operand, and the difference is placed in the register.

1. The second operand is a 16-bit signed integer.
2. The second operand must be on a halfword integral bound.
3. The first operand is a 16-bit signed integer.
4. The difference is a 16-bit signed integer.

Condition Code:

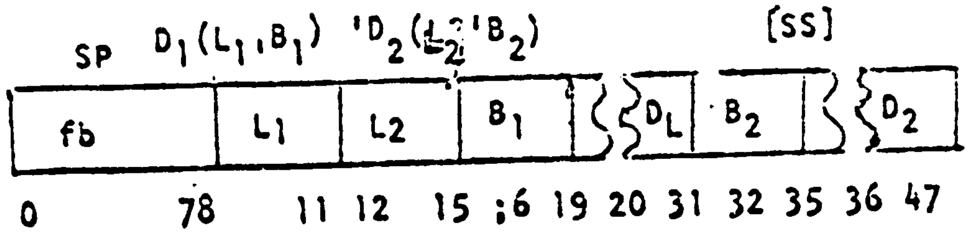
0 Difference is zero.

1 Difference is less than zero.

2 Difference is greater than zero.

SH 10,HALF

SUBTRACT DECIMAL



The second operand is subtracted from the first operand, and the difference is placed in the first operand.

1. Both operands must be in packed format.

2. The difference is in packed format.

3. If the first operand is too short to contain the difference, overflow occurs, and the carry is lost.

4. If the second operand is shorter than the first, subtraction will take place normally.

5. A field may be subtracted from itself.

6. The second operand must not be longer than the first or a program error stop occurs.

Condition Code:

0 Difference is zero

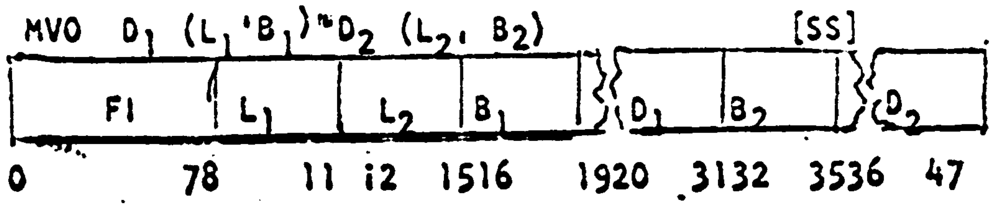
1 Difference is less than zero.

2 Difference is greater than zero.

3 Overflow.

```
SP PAWT, PINT(3)
```

MOVE WITH OFFSET



The second operand is placed to the left of, and adjacent to, the low-order four bits of the first operand.

1. The fields are processed right to left.

2. If the second operand is shorter than the first operand, it is extended with high-order zeros.

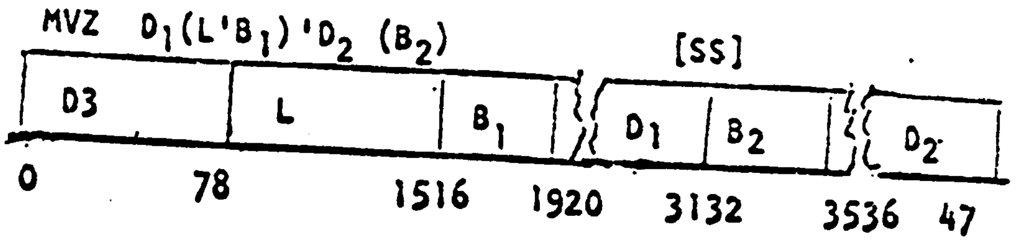
3. If the first operand field is shorter than the second operand, the remaining information is ignored.

Condition Code:

The code remains unchanged.

```
MVO BDSN,GOSN(6)
```

MOVE ZONES



The high-order 4 bits (zone portion) of each byte of the second operand are placed in the first operand field.

1. Movement is from left to right.

2. Movement is one byte at a time.

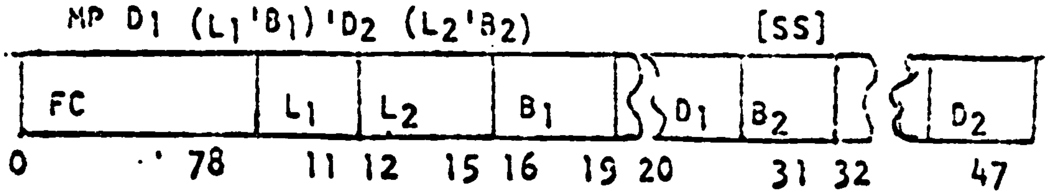
3. The number of zone portions moved is determined by the implicit or explicit length of the first operand.

Condition Code:

The code remains unchanged.

```
MVZ ZNUM+4(L),ZNUM+3
```

MULTIPLY DECIMAL



The first operand (multiplicand) is multiplied by the second operand (multiplier), and the product is placed in the first operand location.

1. Both the multiplicand and the multiplier must be packed.

2. The product is in packed format.

3. The length of the first operand in bytes must be equal to or greater than the number of bytes required to contain all of the multiplicand plus the total number of bytes in the multiplier (second operand) field.

Example:

Multiplier - XXXX 0X XX Xs (3 bytes)

Largest Multiplicand - XXXXXXX XX XX XX XS (4 bytes)

The product field length must then be 7 bytes (3-4) or larger in length

XX XX XX XX XX XX XS

4. The multiplier may not exceed 15 digits and sign (8 bytes).

5. The maximum product size is 41 digits and sign (16 bytes).

Condition Code:

The code remains unchanged.

MP PINT,PRAT

OR IMMEDIATE

OI D1 (B_1) I_2 [SS]

96	I_2	B_1	D_1
0 7	8 15	16 19	20 31

The first and second operands are examined on a corresponding bit by bit basis.

1. If either or both of the corresponding bits are ones, the result is a one and replaces the bit in the first operand.

2. If both bits are zeros, the result is zero and replaces the operand.

3. The second operand is one byte (8 bits) of immediate data which operates with one byte of data at the first operand storage location.

Condition Code:

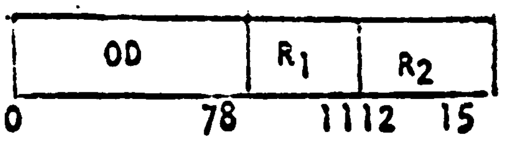
0 Result is zero.

1 Result is not zero.

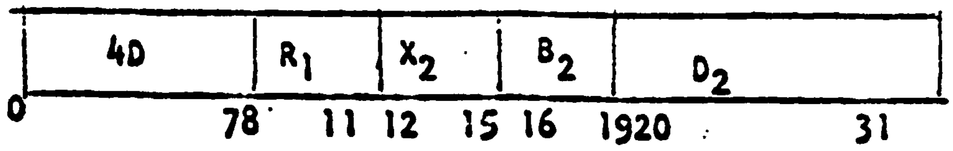
01 OPRD,X"08"

BRANCH AND STORE

BASR R_1'B_2 [RR]



BAS R_1'D_2(X_2'B_2) [RX)



The address of the next sequential instruction (nsi) is stored in the first operand, and a branch is taken to the address-stored in the second operand.

BASR (Branch and Store Registers)

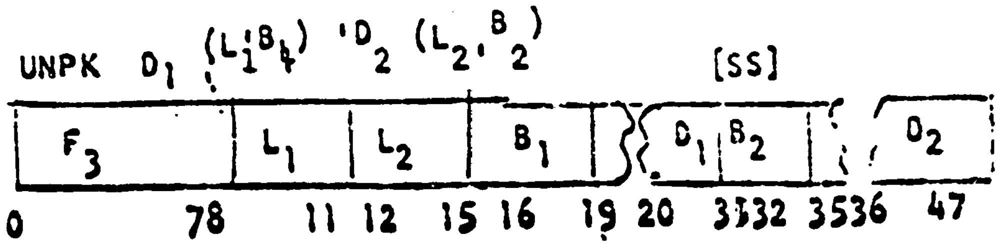
1. If the second operand is 0 (zero), no branch is taken.

Condition Code:

The code remains unchanged.

BEGIN BASR 11.0

UNPACK



The packed format second operand location is changed to signed zoned format and is placed in the first operand location.

1. The fields are processed from right to left.

2. If the first operand field is too long, it will be filled with high-order zeros.

3. If the first operand field is too short, any remaining high-order digits will be ignored.

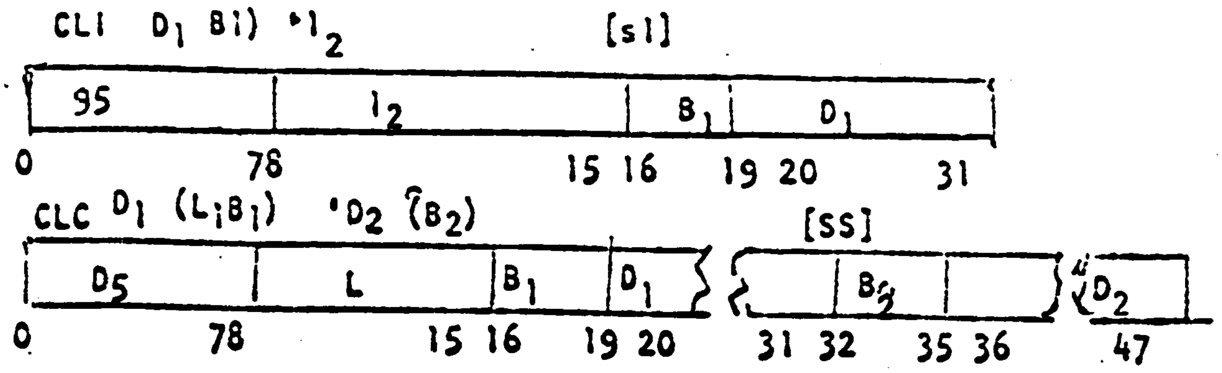
4. The maximum size of the first operand (zoned field) is 16 bytes.

Condition Code:

The code remains unchanged.

UNPK ZONE, PACK

COMPARE LOGICAL



The first operand is compared with the second operand, and the result is indicated in the condition code.

1. Comparison is binary (that is, bit by bit).
2. Comparison proceeds from left to right.
3. Comparison ends as soon as an inequality is found.

CLI only:

One byte in the storage location specified by the first operand is compared with one byte of immediate data.

CLC only:

The number of bytes to be compared is specified by the implicit or explicit length of the first operand.

Condition Code:

0 Operands are equal.

1 First operand is low.

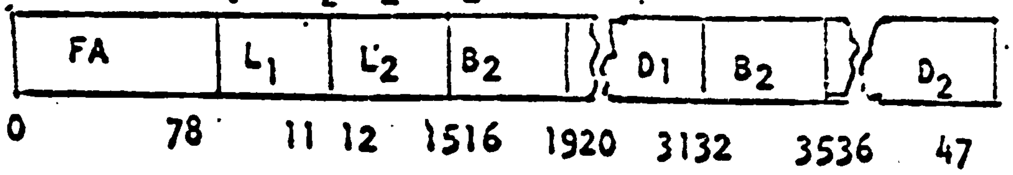
2 First operand is high.

3 ---.

```
CLI CODE, C"E"

CLC NAME, C"SMITH"
```

ADD DECIMAL

AP $D_1(L'B_1)$ $'D_2(L_2$ $'B_2)$



The second operand is added to the first operand, and the sum is placed in the first operand storage location.

1. Both operands must be in packed format.

2. The sum is in packed format.

3. If the first operand is too short to contain the sum, overflow occurs and the carry is lost.

4. If the second operand is shorter than the first operand, addition will take place normally.

5. A field may be added to itself.

6. The second operand may not be longer than the first or an error stop occurs.

Condition Code:

0 Sum is zero.

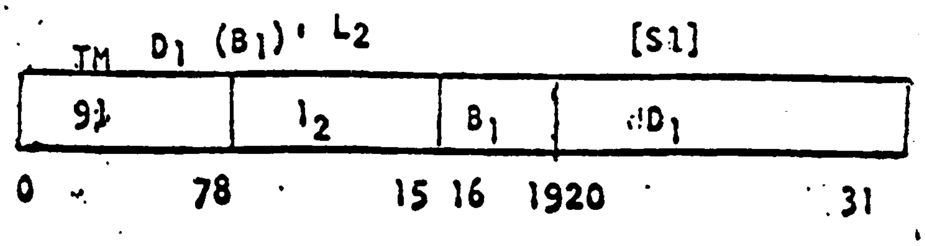
1 Sum is less than zero.

2 Sum is greater than zero.

3 Overflow

AP PINT,INT

TEST UNDER MASK



The state of the first operand bits selected by a mask (second operand) is used to set the condition code.

1. 1-8 bits may be tested.

2. The mask is one byte (8 bits) of immediate data (second operand).

3. A mask bit of one indicates that the corresponding storage bit is to be tested.

4. A mask bit of one indicates that the corresponding storage bit is to be ignored.

Condition Code:

0 Selected bits or mask, all-zero.

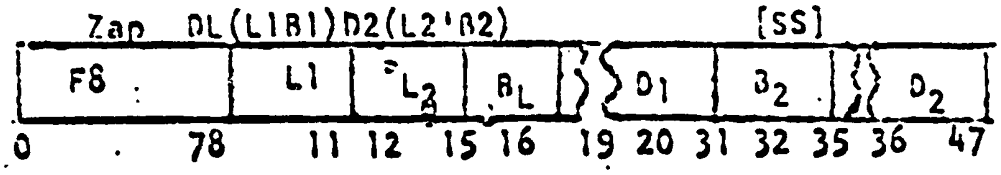
1 Selected bits mixed zero and one.

2 --

3 Selected bits all-one.

```
TM BUTE,X"FF"
BC 4,NWRT
```

ZERO AND ADD



The storage location specified by the first operand is cleared to zcro, and then the second operand data (packed format) is added to the first operand.

1. The length of the second operand must not be greater than the first operand.

2. A negative zero balance will be made positive.

Condition Code:

0 Result is zero.

1 Result is less than zero.

2 Result is greater than zero.

GLOSSARY

Absolute Address: A pattern of characters that identifies a unique storage location or device without further modification. Synonymous with machine address.

Access Method: Any of the data management techniques available to the used for transferring data between main storage and an input/output device.

Accumulator: A register in which the result of an arithmetic or logic operator is formed.

Address: An identification, as represented by a name, or number for a register location in storage, or other data source or destination. Loosely, any part of an instruction which specifies the location of an operand for the instruction.

Address Calculation: A calculation performed on a key so that the result is an address.

Address Modification: The process of changing the address part of ε machine instruction by means of coded instructions.

Address Register: A register that stores an address.

Alias: Another name by which a file is known.

Alphameric: A generic term for alphabetic letters, numerical digits and special characters.

Ascending Order: The word with the lowest key appears in the cell of the lowest number.

Assembler: A program that assembles. See assemble---A program which prepares an object language program by producing absolute or relocatable machine code from a source program of statements containing symbolic operation codes and symbolic operands.

Assemble: To prepare an object-language program from a symbolic language program by substituting machine operation codes for symbolic codes and absolute or relocatable addresses for symbolic addresses.

Auxiliary Function: A function that is either control or data oriented but does not cause data transmission.

Base Register: A register us i for addressing purposes.

Basic Assembler Language: A symbolic language for the writing of source programs.

Basic Assembler Program: A program used to translate source programs written in basic assembler language into machine language.

Basic Monitor: The main control program of the DPS. Available in a card of disk version. Resident in core storage when control is required. Loads programs into core storage and causes their execution.

Bifurcating: No vortex is higher than three, for an abresence only root-like, simple linking nodes, and one root are permitted.

Bifurcating Cell: Contains two pointers.

Binary: 1. Pertaining to a characteristic or property involving a selection, choice, or condition in which there are two possibilities.
2. Pertaining to the number representation system with a base of two.

Binary Code: A code that asks use of exactly two distinct characters, usually 0 and 1.

Binary-Coded Character: One element of a notation system for representing alphameric characters such as decimal digits, alphabetic letters, punctuation marks, etc., by a fixed number of consecutive binary digits.

Binary-Coded Decimal: Pertaining to a decimal notation in which the individual decimal digits are each represented by a binary code group: e.g., in the 8-4-2-1 coded decimal notation, the number twenty-three is represented by 0011, in binary notation, twenty-three is represented as 10111.

Binary Digit: A character used to represent one of the integers smaller than the Radix 2.

Bit: A binary digit.

Bit: The smallest unit of information in System/360. It can have either of the two binary values: zero or one.

Bit Extraction: Mapping by extracting specified bits from a key and then compressing.

Binary-to-Decimal Conversion: Conversion of a binary number to the equivalent decimal number; i.e., a base-two number to a base-ten number.

Blank: One of the characters in a character.

Blank Character: Any character or characters used to produce a character space on an output medium.

Blocks: 1. A physical set of records grouped for the purpose of conserving tape; or disk storage space; or increasing the efficiency of access or processing.

2. A group of bytes, transferred to or from a physical unit device in one operation, may contain one or more logical records.

Branch: To depart from the normal sequence of executing instructions in a computer. A machine instruction that can cause a departure as in (1). Synonymous with "transfer".

Buffer: Area in memory which holds data brought in from an input device.

Byte: A sequence of adjacent binary digits operated upon as a unit.

Byte: The basic unit of information in System/360. Every byte consists of eight bits each having a value of zero or one, (see bit).

Card Code: The combination of punched holes which represent characters (letters, digits, etc.) in a punched card.

Card Column: One of the vertical lines of punched positions on a punched card.

Card Field: A fixed number of consecutive card columns assigned to data of a specific nature.

Card Image: One-to-one representation of the contents on a punched card.

Card Punch: A device to record information in cards by punching holes in cards to represent letters, digits, and special characters.

Card Reader: A device which reads and translates into internal form the holes in punched cards.

Card-Resident-System: Consists of the card control programs, Basic Monitor, Job Control, and initial Program Loader. Used for the execution of object programs contained in punched cards.

Card Stacker: A mechanism which stacks cards in a pocket after they pass through a machine.

Cell: Continuous locations where a record is stored.

Character: One of a set of elementary symbols which may include decimal digits 0 through 9, the letters A through Z, punctuation marks, and any other symbols acceptable to a computer for reading, writing or storing.

Character Set: A list of characters acceptable for coding to a specific computer or input/output device.

Clear: To put a storage or memory device into a prescribed state, usually that denoting zero or blank.

Coded Decimal: A type of notation in which each decimal digit is identified by a group of binary ones and zeros.

Column Binary: Pertaining to the binary representation of data on punched cards in which adjacent positions in a column correspond to adjacent bits of data, e.g., each column in a 12 row card may be used to represent 12 consecutive bits of a 36 bit word.

Command: An instruction in machine language.

Communication: The process of transferring information from one point, person, or equipment to another.

Communication Region: An area of the Basic Monitor. Contains date, storage capacity specification, UPS1 byte, user areas 1 and 2, and program-name area. Provides for inter-program and intra-program communication.

Compiler: A compiler is a program which prepares an object language program.

Component: A basic part, an element.

Compound Key: Combining several fields of the record for either ordering or searching.

Computer: 1. A device capable of solving problems by accepting data, performing prescribed operations on the data, and supplying the results of these operations on the data. Various types of computers are calculators, digital computers and analog computers.

2. In information processing, usually an automatic stored-program computer.

Computer, Hexadecimal Number System: A number system using the equivalent of the decimal number sixteen as a base.

Computer Instructions: Same as machine instruction.

Constant: A fixed or invariable value or data item.

Control Field: A group of contiguous bytes that are within a data record. The sort, or merge of the records, is based on the collating sequence as applied to these bytes.

Control Statement: A punched card that contains information to tailor a program to the requirements of the user.

Convert: To change the representation of data from one form to another; e.g., to change numerical data from binary to decimal or from cards to tape.

Core-Image Directory: A table on the system disk pack containing the addresses and sizes of the program and/or program phases in the core-image library.

Core Library: A disk area containing the job control program, other IBM-supplied programs (except the Basic Monitor), and user's problem programs. Permits retrieval of programs and/or phases by the Basic Monitor.

Core-Image Maintenance Program: A service program. Updates the core-image library and directory. Is used to add and/or delete phases.

Core Storage: A form of high speed storage using magnetic cores.

Counter: A device such as a register or storage location used to represent the number of occurrences of an event.

Cycle: 1. An interval of space or time in which one set of events or phenomena is completed.

2. Any set of operations that is repeated regularly in the same sequence. The operations may be subject to variations of each repetition.

Cylinder: All the marks which can be accessed without moving the READ/WRITE Heads.

Data: Any representation, such as character quantities, to which meaning might be assigned.

Data Conversion: The process of changing data from one form of representation to another.

Data File: A collection of related records organized in a specific manner, for example, a payroll file (one record for each employee, showing his rate of pay, deductions, etc.) or an inventory file (one record for each inventory item, showing the cost, selling price, number in stock, etc.).

Data Management System: A data management system deals with records in a file. The records contain information called elementary data items, which are the object of processing.

Data Processing: A systematic sequence of operations performed on data.

Data Processing System: A network of matching components capable of accepting information, processing it according to a plan, and producing the desired results.

Data Structure: The arrangement and interrelation of records in a file form; a data structure.

Decimal: 1. Pertaining to a characteristic or property involving selection, choice or condition in which there are ten possibilities.

2. Pertaining to the number representation system with a radix of ten.

Deck: A collection of punched cards.

Decimal-to-Binary Conversion: The conversion of a decimal number to the equivalent binary number, i.e., a base-ten number to a base-two number.

Decision: A determination of future action.

Decision Instruction: An instruction that effects the selection of a branch of a program, e.g., a conditional branch instruction.

Decrement: The quantity by which a variable is decreased.

Decision Box: A flow-chart symbol whose interior contains the criterion for decision or branching.

Diagram: A schematic representation of a sequence of operations or routines.

Diagnostic: Pertaining to the detection and isolation of a malfunction or a mistake.

Digit: One of the symbols 0, 1.....9..... used to designate a quantity smaller than n for a base-n number system.

Directory: An auxiliary list for which each entry corresponds to one sublist in the object list; also an auxiliary list for searching a lower level directory and for finding a subdirectory there.

Direct Access: Retrieval or storage of data by providing a reference to its physical location on a volume.

Displacement: The difference (in bytes) between the contents of a base register (or the address represented by a symbol) and a referenced storage location.

Dummy: Pertaining to the characteristic of having the appearance of a specified thing but not having the capacity to function as such.

Edit: To modify the form or format of data; e.g., to insert or delete characters such as page numbers or decimal points.

Effective Address: The absolute address of the current operand. This may differ from that of the instruction in storage.

Error: 1. A general term to indicate that a data value is not correct or that a machine component is malfunctioning.

2. A specific term for the amount of loss in precision.

Execute: To carry out an instruction or perform a routine.

Explicit Addressing: Specification of an address by a base register and a displacement in the form D(B).

Expression: A symbol or self-defining term used in the operand of a statement.

Extent: Area of a disk file specified by an upper limit and a lower limit.

Fetch (program): 1. To obtain requested load modules and load them into main storage, relocating them as necessary.

2. A control routine that accomplishes: 1 holds to represent data and a card as in 1 before being punched.

File: A collection of related records treated as a unit, e.g., in inventory control, one line of an invoice forms an item, a complete invoice forms a record, and the complete set of such records forms a file.

File Label: Label containing information applicable to a given data file or portion of a data file stored on a particular volume.

File Reorganization: A term used to describe the process of writing a new file from an indexed sequential file, purging records that are tagged for deletion, and their sequential positions in the prime data area.

Fixed-Length Record: A record having the same length as all other records with which it is logically or physically associated.

Flow Chart: A graphical representation for the definition, analysis, or solution of a problem in which symbols are used to represent operations, data, flow and equipment.

Garbage Collection: Collection of cells deleted by linked lists operation to make available again for use.

Hardware: The mechanical magnetic, electrical and electronic devices or components of a resident system from card input.

Hierarchy: An information source which can be represented by an aboresence.

Hopper: A device that holds cards and makes them available to a card feed mechanism. Synonymous with input magazine. Contrast with card stacker.

Identification: A code number or code name which uniquely identifies a record, block, file or other unit of information.

Image: An exact logical duplicate stored in a different medium.

Immediate Address: The designation of an instruction address which is used as data by the instruction of which it is a part.

Implied Address: The address assigned to a symbol by the basic assembler program.

Index Register: A register whose content is added to or subtracted from the operand address prior to or during the execution of an instruction.

Indexing: A technique of address modification often implemented by means of index registers.

Indirect Pointer: A pointer which depicts the place a file pointer can be obtained.

Information Structure: Inherent property of the information in a given set of data. This property may be by design or by the natural way the data appear in the given set of data.

Initialize: To set certain counters, switches and addresses at specified times in a computer routine.

Initial Program Loader: A Control program, available in a card and a disk version; loads basic monitor into core storage; used to assign actual input/output addresses to symbolic addresses SYSRDR and/or SYSRES; places name of Job Control program into communication region of basic Monitor; required for the initialization of the card-resident or disk-resident system.

Input:

1. The data to be processed.

2. The state of sequence of states occurring on a specified input channel.

3. The device or collective set of devices used for bringing data into another device.

4. A channel for impressing a state on a device or logic element.

5. The process of transferring data from an external storage to an internal storage.

6. Pertaining to any entities such as are quoted above.

Input/Output:

1. Common abbreviation I/O. A general term fo. the equipment used to communicate with a computer.

2. The data involved in such communication.

3. The media carrying the data for Input/Output.

Input Area: The area of internal storage into which data is transferred from external storage.

Installation: A general term for a particular computing system in the context of the overall function it serves and the individuals who manage it, operate it, apply it to problems, service it, and use the results it produces.

Instruction: A statement that specifies an operation and the values of locations of all operands. In this context, the term instruction is preferable to the terms command or order which are sometimes used as synonyms. Command should be reserved for electronic signals. Order should be reserved for sequence, interpolation and related usage.

Instruction Format: The allocation of bits or characters of a machine instruction to specific functions.

Interpreter: Translator.

Interrupt: A break in the normal flow of a system or routine such that the flow can be reused from that point at a later time.

Inverte and File: A file that consists of one sub-file for each value a key may have and one inverted file corresponding to each key field in the record for which a search may be necessary.

Job Control Program: A control program, resides in core storage between jobs and provides for automatic job-to-job transition. Performs I/O device assignment. It causes the Basic Monitor to load the next program.

Job Control Statement: Any one of the control statements in the input stream that identifies a job or defines its requirements and options.

Label:

1. A physical identification record on magnetic tape located either preceeding or following a data file, or both. If a data file extends beyond a single reel of tape, a label can be placed preceeding and following the data on each reel.

2. A physical identification record disk which identifies the volume or file.

Language: A defined set of characters which are used to form symbols, words, etc., and the rules for combining these into meaningful communications i.e., English, French, Algol, Fortran, COBOL, etc.

Language Translator: A general term for any assembler, compiler, or other routine statements in one language which produces equivalent statements in another language.

Library: A group of files.

Library Management Programs: Collective term for four service programs; core-image maintenance, macro maintenance, directory service, and allocation organization programs.

Link: The pointer filled for a linked list.

Linkage: The interconnections between a main routine and a closed routine from the main routine.

Linkage Editor: A service program. It relocates programs or phrases and links separately assembled programs or phrases.

Load:

1. To fetch, i.e., to read a load module into main storage preparatory to executing it.

2. To place data into internal storage.

Load Program: A service program that, unlike the other service programs, is contained in punched cards. It creates a disk.

Load Module: The output of the linkage editor; a program in a format suitable for loading into main storage for execution.

Location: A position in storage that is usually identified by an address.

Logical Record: A record identified from the standpoint of its content, function, and use rather than its physical attributes. It is meaningful with respect to the information it contains. (Contrasted with physical record.)

Logical Unit Table: A feature of the Basic Monitor. It has twenty-six logical unit blocks, each of which refers to one specific symbolic I/O address. These symbolic addresses are related to physical I/O device addresses by means of ASSIGN control cards.

Loop: A sequence of instructions that is repeated until a terminal condition occurs.

Machine Address: Same as absolute addres

Machine Code: Same as operation code.

Machine Instructions: An instruction that the particular machine can recognize and execute.

Machine Language: A language that is used directly by a given machine.

Macro Instruction: An instruction that is replaced in a routine by a predetermined sequence of machine instructions.

Macro Library: A disk area containing the macro definitions required by the macro instructions issued in user-written programs. Contains source statements to generate commonly used routines.

Macro Maintenance Program: A service program. It updates the macro library and directory. It is used to add and/or delete macro definitions.

Magnetic Tape: Ink containing particles of magnetic substance which can be detected or read by automatic devices e.g., the ink used for printing on some bank checks for magnetic ink character recognition.

A tape with a magnetic surface on which data can be stored. A tape of magnetic surface used as the constituent in forms of magnetic codes.

Mask: An alphameric character string consisting of one or more digits, used to test or alter the contents of storage positions.

Mnemonic Code: A technique to assist the human memory. A mnemonic code enables the original word and is usually easy to remember, e.g., MPY for multiply and ACC for accumulator.

Module: (programming): The input to, or output from, a single execution of an assembler, compiler, or linkage editor; a source, object, or load module; hence, a program unit that is discrete and identifiable with respect to compiling, combining with other units, and loading.

Multi-Extent Disk File: File stored on a disk pack in several areas or defined by more than one extent.

Multilist: A linked list which may contain shared sublists.

Multi-Pack Disk File: File stored on more than one disk pack.

Multi-Volume Disk File: The same as Multi-Pack Disk File.

Name: An alphameric character string, normally used to identify a program.

Object Module: The output of a single execution of an assembler or compiler, which constitutes input to linkage editor. An object module consists of one or more control sections in relocatable, though not executable, form and an associated control dictionary.

Operation:

1. The act specified by a single computer instruction.

2. A program step undertaken or executed by a computer, e.g., addition, multiplication, extraction, comparison, shift or transfer. The operation is usually specified by the operation part of an instruction.

Operation Code: The code that represents the specific operations of a computer.

Output:

1. Data that has been processed.

2. The state or sequence of states occurring on a specified output of a device.

3. The device or collective set of devices used for taking data out of a device.

4. A channel for expressing a state on a device or logic element.

5. The process of transferring data from an internal storage to an external storage.

6. Pertaining to any entities such as are quoted above.

Output Area: The area of internal storage from which data is transferred to external storage.

Overflow: That portion of data that exceeds the capacity or the allocated unit of storage, pertaining to the generation of overflow as in Overflow Area.

Overflow Area: The area a load module or a segment of a load module used for addition of records to a list or sublist which has no room left in it.

Overlay: To place a load module or segment of a load module into main storage locations occupied by another load module or segment which has already been processed.

Pack: To combine two or more units of information into a single physical unit to conserve storage.

Packed Decimal: Storage technique whereby two digits or one digit and sign are stored per byte.

Padding: A technique used to fill out a block of information with dummy records, words or characters.

Phases: The smallest addressable unit in core-image library of a disk-resident system.

Physical and Logical Unit Tables Service Program: A service program. This program (PSERV) is used to display, and/or change the permanent device assignments, and/or to change the configuration byte of the Basic Monitor on the system disk pack.

Physical Record: A record identified from the standpoint of the manner or form in which it is stored and retrieved, that is, one that is meaningful with respect to access. (Contrasted with Logical Record.)

Physical Unit Table: A feature of the Basic Monitor. It has eight physical blocks, each of which contains a physical device address. Pointers to these entries are inserted into the logical unit table by means of ASSGN control cards.

Printer: A device which expresses coded characters as hard copy.

Problem Program: Any of the class of routines that perform processing of the type for which a computing system is intended, and including routines that solve problems, monitor and control industrial processes, sort and merge records, perform computations, process transactions against stored records, etc.

Processing Program: A general term for any program that is not a control program.

Program:

1. The plan for the solution of a problem including data gathering, processing and reporting.

2. A group of related routines which solve a given problem.

Programming Language: A language used to prepare computer programs.

Pseudo-Register: A register with fixed contents used in conjunction with an IBM System/360.

Punched Card: A card punched with a pattern.

Read: To transfer information from an input device to internal or auxiliary storage.

Reader: A device which converts information in one form of storage to information in another form of storage.

Record: A general term for any unit of data that is distance from all others when considered in a particular context.

Register: A device capable of storing a specified amount of data such as one half word.

Relative Address: An address expressed by a previously defined symbol and a displacement. (e.g., FID-10)

Relocatable Area: An area on the system disk pack to temporarily hold an object module, thus permitting the assembly or compilation and the execution of a program or program phase in one job.

Relocate: In programming, to move a routine from one portion of internal storage to another and to automatically adjust the necessary address references so that the routine, in its new location, can be executed.

Report Generator and Report Program Generator (RPG): A program which constructs reports or report-writing programs in accordance with input specifications of the data file and of the desired report.

Reset:

1. To restore a storage device to prescribed initial state, not necessarily that denoting zeros.

2. To place a binary cell into the zero state.

Restart: To return to a previous point in a program and resume operation from that point.

Search Key: A key used to find a record which has a presearched identity within a file.

Seek: To position the access mechanism of a direct-access device at a specified location.

Self-Defining Term: A term with an implied value (e.g., 300, X"2A", C"F").

Semantics: The meanings which govern subsequent interpretation.

Service Programs: A collective term used to refer to the Library Management Program.

Single Extent Disk File: A file stored on disk where the file has exactly one extent.

Sort: The process by which a data set of logical records is sequenced according to the collating-sequence value of the control field of the records. Also a program that performs the process.

Source Language: A language that is an input to a given translation process.

Source Program: A program written in a source language.

Special Characters: In a character set, a character that is neither a numeral nor a letter, e.q., *, -, $ and blank.

Statement: In computer programming, a meaningful expression or generalized instruction in a source language.

Step:

1. One instruction in a computer routine.

2. To cause a computer to execute one instruction.

Store:

1. To enter data into a storage device.

2. To retain data in a storage device.

Storage:

1. Pertaining to a device into which data can be entered and from which it can be retrieved at a later time.

2. Loosely, any device that can store data.

Storage Allocation: The assignment of blocks of data to specified blocks of storage.

Storage Capacity: The amount of data (in bytes) that can be contained in a storage device.

Subroutine: A routine that can be part of another routine.

Switch:

1. A symbol used to indicate a branching point, or a set of instructions to condition a branch.

2. A physical device which can alter flow.

Symbol Table: A mapping for a set of symbols to another set of symbols or numbers.

Symbolic Address: An address expressed in symbols convenient to the programmer.

Symbolic I/O Device (e.g., SYRES, SYSIPP, SYS005): This address is related to an actual address by means of the logical unit table.

Symbolic Language: The discipline that treats formal logic by means of a formalized language or symbolic calculus whose purpose is to avoid the ambiguities and logical inadequacies of natural language. Advantages of the symbolic method are greater exactness of formulation and greater exactness of formulation and power to deal with complex material.

Syntax: The rules for constructing admissable combinations of the characters in the basic alphabet.

System:

1. A collection of consecutive operations and procedures required to accomplish a specific objective.

2. An assembly of objects united to form a functional unit.

System Directory: A table on the system disk pack listing the addresses and sizes of the core-image library directory, the macro library and the directory, and the locatable area.

System Disk Pack: The disk pack on which the user's disk-resident system is located.

Table: A collection of data, each item being uniquely identified either by some label or by its relative position.

Table Look-Up: A procedure for obtaining the function value corresponding to an argument from a table of function values.

Throughput: A measure of system efficiency, the rate at which work can be handled by the computing system.

Truncate: To cut off a specified spot (as contrasted with round or pad).

Unpack: To recover the original data from packed data.

User: Anyone who requires the services of a computing system.

Volume: That portion of a single unit of storage media that is accessible to a single read/write mechanism. For example, a reel of magnetic tape for a 2415 magnetic tape drive, or one 1316 Disk Pack for an IBM 2311 Disk Storage Drive.

Volume Label: Label which uniquely identifies the volume.

Volume Table of Contents (VOTC): A table associated with a disk volume, which describes each data set on the volume.

Zero: The elimination of non-significant zeros in a number.

Zone: The 12, 11 or 0 punches in IBM card code.

BIBLIOGRAPHY

I. Books, Pamphlets, and Government Publications

Abraham, P. W. "List-Processing Languages", *Digital Computer User's Handbook* (M. Klerer and G. A. Korn, eds.). New York: McGraw-Hill, 1967.

Adams, C. W., and Laning, J. H. Jr., "The M. 8. T. Systems of Automatic Coding: Comprehensive, Summer Session, and Algebraic", *Symposium on Automatic Programming for Digital Computers*, Office of Naval Research, Dept. of Navy, D. C., 1954, pp. 40-68.

Albers, H. H., and Schoer, Lowell, *Programmed Organization and Management Principles*. John Wiley and Sons, Inc., 1966.

Allendoerfer, C. and Oakely, Cletus., *Manual to Fundamentals of Freshman Mathematics*, New York: McGraw-Hill Book Co., 1959.

American Standard Vocabulary for Information Processing. X3.12, American Standards Association (now United States of America Standards Institute), New York, 1966.

Anderson, D. M. *Basic Computer Programming, the IBM 1620, Fortran*, New York: Merideth Publishing Co., 1968.

Ayres, Frank Jr., *College Mathematics*. New York: Schaum Publishing Co., 1958.

________. *Calculus*. New York: McGraw-Hill Book Co., 1964.

________. *Mathematics of Finance*. New York: Schaum Publishing Co., 1963.

Baker, C. C. T. *Dictionary of Mathematics*. New York: Hart Publishing Company, Inc., 1961.

Bardell, Ross, and Spitzbart, Abraham. *College Algebra*, 2nd ed., Reading, Mass.: Addison-Wesley Publishing Co., 1966.

Barnew, R. M. *Motion and Time Study*. New York: John Wiley and Sons, Inc., 1958.

Barr, D. R., and Willmore, F. E. *College and University Mathematics; A Functional Approach*. Boston: Allyn and Bacon, Inc., 1968.

Barron, D. W. *Recursive Techniques in Programming*. New York: Elsevier Publishing Co., Inc., 1969.

Becker, J., and Hayes, R. M. *Information Storage and Retrieval*. New York: John Wiley and Sons, 1963.

Beighey, Clyde, and Bordhardt, G. C. *Modern Business Mathematics*. New York: McGraw-Hill Co., Inc., 1960.

Berger, C. *The Theory of Graphs and Its Applications*, New York: John Wiley, 1962.

Berkely, C. C., ed. "The Programming Language Lisp: Its Operation and Applications." *Information International*, Maynard, Mass., 1964.

Berkowits, N., and Robertson, Munro, Jr. *Automatic Data Processing and Management*. Belmont, Calif.: Dickenson Publishing Co., 1969.

Bobrow, D. G., ed. "Symbol Manipulation Languages and Techniques," *Proceedings of IFIP Working Conferences on Symbol Manipulation Languages*. North Holland Publishing Co., Amsterdam, 1968.

Bohl, Marilyn. *SRA No. 1, 2*. Chicago: Science Research Associates, Inc., 1969.

Bourne, C. P. "Review of the Methodology of Information Systems Design" *Information Retrieval Workshop*. New York: Spartan Books, 1962.

Boutell, W. S. *Computer-Oriented Business Systems*. Englewood Cliffs, N.J.: Prentice-Hall Co., 1968.

Brighten, R. W. *Practical Data Processing*. New York: Macmillan Co., 1969.

________., Luskin, B. J., and Tileon, T. *Data Processing for Decision Making*. New York: Macmillan Co., 1968.

Brooks, F. P., Jr. and Iverson, K. E. *Automatic Data Processing*. New York: Wiley, 1963.

Brownson, H. L. "Evaluation of the document searching system and procedures" *I. Doc.* 21, 4, Dec., 1965.

Budd, A. E. A Method of the Evaluation of Software: Procedural Language Compilers Particularly COBOL and FORTRAN, Mitre Corp. (DDC) AD651142, Commerce Department Clearinghouse, Springfield, Va., April 1966.

Burnaugh, H. P. "The Bold (Bibliographic On-Line Display) System". Schecter, G., ed. *Information Retrieval--A Critical View*. Washington D.C.: Thompson Book Co., 1967. pp. 53-66.

Bushnell, C. D., and Allen, D. W. *The Computer in American Education*. New York: Wiley and Sons, Inc., 1967.

Carter, Norman. *Introduction to Business Data Processing*. Belmont, Calif.: Dickenson Publishing Co., 1968.

Cary, F. T. *Data Processor Special Report Programming*. New York: T. B. Merrill, 1969.

Chapin, Ned. "What Choice of Programming Languages?" *Computers and Automation*. Vol. 14, No. 2, February, 1965. pp. 12-14.

Chernoff, H., and Moses, D. E. "Elementary Decision Theory." *Wiley Publications in Statistics*. New York: Wiley and Sons, 1969.

Chestnut, Harold. *Systems Engineering Tools*. New York: John Wiley and Sons, 1965.

Coan, J. S. *Basic Basic*. New York: Hayden Book Co., 1970.

Cohen, N. R., and Nagel, Ernest. *An Introduction to Logic*. New York: Harcourt Brace and World, Inc., 1962.

Colman, H., Smallwood, B. S., and Brown, B. *Computer Language*. New York: Harcourt, Brace and World, Inc., 1962.

Collins, F. T. *Manual Critical Path Techniques for Construction*. California: Know-How Publications, 1965.

COMIT Programming Reference Manual. Cambridge, Mass.: MIT Press, 1961.

Crawford, F. R. *Introduction to Data Processing*. New Jersey: Prentice-Hall, 1968.

Davidson, C. H., and Konig, E. E. *Computers*. New York: Wiley and Sons, 1967.

Davis, G. B. *Computer Data Processing*. New York: McGraw Hill Book Co., 1968.

________. *An Introduction to Electronic Computers*. New York: McGraw-Hill, 1969.

________. *An Introduction to System/360 Computer*. New York: McGraw-Hill, 1968.

Desmonde, W. H. *Computers and Their Uses*. Englewood Cliffs, N.J.: Prentice-Hall, 1964.

Dodd, G. G. "APL--A Language for Associative Data Handling in PL/I". *Proc. FJCC.*, Vol. 29, 1966. pp. 677-684.

Dorn, W., and Greenberg, H. J. *Mathematics and Computing*. New York: Wiley and Sons, 1967.

Downie, N.M., and Heath, R. W. *Basic Statistical Methods*. New York: Harper and Rowe, 1965.

Draper, J. E., and Klingman, J. E. *Mathematical Analysis Business and Economic Applications*. New York: Harper and Rowe, 1967.

Drew, F. L., Summit, R. K., and Whitely, R. E. "An On-Line Technical Reference Retrieval System". *Am. Doc.* 17, 1, January, 1966. pp. 3-7.

Drooyan, I., and Wooton, W. *Programmed Beginning Algebra, Radicals and Equations.* New York: Wiley and Sons, 1963.

Dupchak, R. LIPL: Linear Information Processing Language, RAND Corp., *RM 3879-PR*, Santa Monica, Calif.: October, 1963.

________. "TIPL: Linear Information Processing Language". Rand Corp. *RM 4320-PR*, Santa Monica, California: October, 1963.

Fairthorme, R. A. "Some Basic Comments on Retrieval Testing". *J. Doc.* 21:4, 1965, pp. 267-270.

Feingold, Carl. *Fundamentals of COBOL Programming.* Iowa: W. C. Brown, 1969.

Feldman, J. "TALL-List Processor for the Philco 2000 Computer". *Comm. ACM.* Vol. 5, No. 9, Sept. 1962, pp. 484-485.

Fels, E. M. "Evaluation of the Performance of an Information by the Modified Mooers Plan". *Am. Doc.* 14:1. Jan., 1963. pp. 28-34.

Fenves, S. J.; Logcher, R. D.; and Mauch, S. P. *STRESS, A Reference Manual.* Cambridge, Mass.: MIT Press, 1965.

Ferguson, H. E., and Berner, E. "Debugging System at the Source Language Level". Comm. *ACM.* Vol. 5, No. 8, Aug. 1963. pp. 430-432.

Fesher, F. P., and Swindle, G. F. *Computer Programming Systems.* New York: Holt, Reinhart, Winston, 1964.

Fletcher, D. A., and Cashman, T. C. *IBM System/360 RPG.* Vol. 1. Anaheim, Calif.: Anaheim Publishing Co., 1967.

Flores, Ivan. *Computer Sorting.* Englewood Cliffs, New Jersey: Prentice-Hall, 1965.

________. *Computer Software.* Englewood Cliffs, New Jersey: Prentice-Hall, 1965.

Gardiner, J. T. *Special Reports on Major Business Problems.* New York: McGraw-Hill, 1970.

Ghosh, S. P., Abraham, C. T., and Ray-Chaudhure, D. K. "File Organization Schemes Based on Finite Geometries". IBM Res. Center, R. C-1459 and RC-1561, Aug., 1965 and Mar. 1966. pp. 1-12.

Gibson, D. E. *An Introduction to Automated Data Processing.* Elmhurst, Ill.: Business Press, 1966.

Goldberg, S. *Probability*. Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1960.

Goodman, A. W. *Analytic Geometry and the Calculus*. London: Macmillan, 1969.

Grad, B. "Tabular Form in Decision Logic". *Datamation*. Vol. 5, No. 7, July, 1969. pp. 22-28.

Gray, L. C. "Compound Data Structure for Computer Aided Design, A Survey," *Pro. ACM 22nd Natl. Conf.*, 1967. pp. 355-365.

Grey, M., and London, K. R. *Documentation Standards*. New York: Brandon Systems Press, 1969.

Gruenberger, F. *Computing*. New York: Harcourt, Brace and World, 1970.

Gruenberger, R. *Computing, An Introduction*. New York: Harcourt, Brace and World, 1967.

Gullahorn, J. T., and Gullahorn, J. E. "A Computer Model of Elementary, Social Behavior." *Computers and Thought*. Edited by A. Felgenbaum and J. Feldman. New York: McGraw-Hill, 1963. pp. 375-385.

Halcomb, J. *Project Manager's Pert/CPM Handbook*. Calif.: Halcomb Assoc., 1969.

Halpern, M. I. "XPOP: A Meta Language Without Metaphysics." *Proc. FICC*. Vol. 26, Pt. 1, 1964. pp. 57-68.

Hart, W. L. *Trigonometry*. Boston: V. C. Heath and Co., 1954.

Haverty, L. P. "Programming Language Selection for Command and Control Applications." Rand Corp., P-2967, Santa Monica, Calif., 1964.

Hellerman, H. "Addressing Multidimensional Arrays." *Comm. ACM5*, 4, April, 1962. pp. 205-207.

________. "Parallel Processing of Algebraic Expressions." *Trans. EC-15*. (Feb., 1966), pp. 82-91.

Higman, Bryan. *A Comparative Study of Programming Languages*. New York: American Elsevier Pub. Co., 1967.

Hiller, F., and Lieberman, G. J. *Introduction to Operations Research*. San Francisco, Calif.: Holden-Day, 1968.

Holbertson, G. E., "Application of Automatic Coding to Logical Processes." *Symposium on Automatic Programming for Digital Computers*. Offices of Naval Research, Dept. of the Navy, Washington, D.C., 1954. pp. 34-39.

Holt, W. W. "Some Theorizing on Memory Structure and Information Retrieval." *Proc. ACM*, 18th Natl. Conf. 1963. pp. 27.

Homer, E. D. "An Algorithm for Selecting and Sequencing Statements as a Basis for Problem-Oriented Programming Systems." *Proc. ACM*, 21st Natl. Conf. 1966. pp. 305-312.

Hopper, G. M. "Compiling Routines." *Computers and Automation*. Vol. 2, No. 4, May, 1953. pp. 1-5.

________., and Naychly, J. W. "Influence of Programming Techniques on Design of Computers." *Proc. IRE*, Vol. 41, No. 10, October, 1953, pp. 1250-54.

________. "Automatic Coding for Digital Computers." (Talk presented at the High Speed Computer Conf., Louisiana State Univ., February 16, 1955). Remington Rand Corp. *EDOL*, 1955.

Horngren, C. T. *Cost Accounting*. Englewood Cliffs, N.J.: Prentice-Hall, 1962.

Horty, J. F. "Experience With Application of Electronic Data Processing Systems in General Law." *Mull*. 60D, December, 1960.

Howes, V. E. *Self Teaching Intermediate Algebra*. New York: Wiley and Sons, 1967.

Hsiao, D., and Prywes, N. S. "A System to Manage an Information System." *F. I. D. Dissemination*, Rome, Italy, June, 1967.

Hughes, J. T. *Digital Computer Workbook*. Maynard, Mass.: Digital Equipment Corp., 1966.

Hunt, E. B., and Hovland, C. E. "Programming a Model of Human Concept Formulation." *Computers and Thought*. Edited by E. A. Felgenbaum and J. Feldman. New York: McGraw-Hill, 1963. pp. 310-325.

Iannone, A. L. *Management Program Planning and Control*. Englewood Cliffs, N. J.: Prentice-Hall, 1967.

Jacobwitz, Gentry. *Electronic Computers*. New York: Doubleday and Company, 1963.

Johnson, Kast and Rosenweig, J. *The Theory and Management of Systems*. New York: 2nd edition, McGraw-Hill Book Co., 1963.

Johnson, L. R. "An Indirect Chaining Method for Addressing on Secondary Keys." Comm. *ACM*. Vol. IV. No. 5 (May, 1961).

Jonker Information Systems (Manual). Galthersburg, Maryland: Jonker Corporation, 1967.

Kavanaugh, T. F. "TABSON--A Fundamental Concept for System-Oriented Languages." *Proc. FICC*. Vol. 18, 1960. pp. 117-136.

Keller, L. M., Strum, C. C., and Yang, G. H. "Remote Computing: An Experiment System, Part 2: Internal Design." *Proc. AJCC*. Vol. 25, 1964. pp. 425-433.

Kelley, John L. *Student Manual to Modern Algebra*. Princeton: N.J.: D. Van Nostrand and Co., 1969.

Kells, L. M.; Kern, W. F.; and Bland, J. R. *Plane and Spherical Trigonometry*. 3rd ed. New York: McGraw Hill, 1935.

Kleppner, D., and Ramsey, Norman. *Quick Calculus*. New York: John Wiley and Sons, 1965.

Klerer, M. and Korn, G. A. *Digital Computer User's Handbook*. New York: McGraw-Hill, 1967.

Knuth, K. E. *The Art of Computer Programming, Fundamental Algorithms*. Vol. I. Reading, Mass.: Addison-Wesley, 1968.

Koontz, H. and O'Donnell, C. *Principles of Management*, 3rd ed. New York: McGraw-Hill, 1964.

Krauss, I. *Administering and Controlling the Company's Data Processing Function*. Englewood Cliffs, N.J.: Prentice-Hall, 1969.

Lande, Henry F. *How to Use the Computer in Business Planning*. Englewood Cliffs, New Jersey, 1969.

Landauer, W. I. "The Balanced Tree and Its Utilization in Information Retrieval." *Transactions on Electronic Computer of the IEE*, Vol. ECXII, No. 5. (December, 1963).

Laning, H. H. and Ziegler, W. "A Program for Translation of Mathematical Equations for Whirlwind T, MIT." *Engineering Memorandum E-364*. Instrumentation Lab., Cambridge, Mass., (Jan., 1954).

Lefkowitz, K., and Powerd, R. R. "A List-Structured Chemical Information Retrieval System." Schecter, H. (ed.). *Information Retrieval--A Critical View*. Washington D.C.: Thompson Book Co., 1957. pp. 209-230.

Lindsay, F. A. *New Techniques for Management Decision Making*. New York: McGraw-Hill Book Co., 1958.

Lipschutz, S. *Probability (Theory and Problems)*. New York: McGraw-Hill, 1968.

________. *Set Theory and Related Topics*. New York: McGraw-Hill, 1964.

Madnick, S. E. "String Processing Techniques." *Comm. ACM.* Vol. 10, No. 7, (1967).

Mager, Robert F. *A Programmer's Introduction to IBM System/360 Architectur:.* Palo Alto, Calif.: Fearon Pub., 1969.

Magnino, J. Jr., "IBM Technical Information Retrieval Centers--Progress and Plans." *Proc. 1966 Am. Doc. Inst.,* Santa Monica, Calif., 1964.

Maise, E. E. *Calculus.* Reading, Mass.: Addison, Wesley Pub. Co., 1967.

Marcus, M. and Minc. H. *Introduction to Linear Algebra.* New York: Macmillan Co., 1965.

Martin J. *Design of Real Time Computer Systems.* New Jersey: Prentice-Hall, 1966.

Martine, R. L. *Finding the Critical Path, Management and Control.* New York: The Comet Press Co., 1964.

Maurer, W. D. "An Improved Hash Code for Scatter Storage." *CACM II.* (Jan., 1968). pp. 35-38.

Mc Carthy, J. et. al. *LISP L.T. Programmer's Manual.* Cambridge, Mass.: MIT Press, 1962.

McCollough J., and Loche, V. A. *Statistical Concepts.* New York: McGraw Hill Book Co., 1968.

McCracken, D. D. *A Guide to IBM 1401 Programming.* New York: John Wiley and Sons, IBM. Inc., Copy, 1961.

________. *A Guide to COBOL Programming.* New York: John Wiley and Sons, 1967.

McDonough, A. M., and Garrett, L. J. *Management Systems.* Working Concepts and Practices. Homewood, Ill.: Richard D. Irwin, 1965.

McKeenan, W. M.; Horning, J. J.; and Wortman, D. B. *A Compiler Generator.* New Jersey: Prentice-Hall, 1970.

Meadow, C. T. *The Analysis of Information Systems.* New York: John Wiley and Sons, 1967.

Meyer, J. S. *Fun With Mathematics.* Conn.: Fawcett Publications, 1967.

Miller, G. A. and Penink. *A Computer Program and Documentation Written for the IBM 7040.* Univ. of Pennsylvania, 1964.

Minker, J. and Sable, J. "File Organization and Data Management." In Caudre, C.A. (ed.), *Annual Review of Information Science and Technology.* Vol. 2. Interscience, N.Y., 1967.

Mire, F. *Geometry Through Practical Applications*. New York: Barnes and Noble, 1942.

Neilson, K. L. *Modern Algebra*. New York: Barnes and Noble, 1969.

Nelson, E. A., et. al. "Research Into the Management of Computer Programming." *Some Characteristics of Programming Cost Data From Government and Industry*. System Development Corp. TM-2704/000/00. Santa Monica, Calif., November 1965.

Newell A. Ked. *Information Processing Language-V Manual*. Englewood Cliffs, New Jersey: Prentice-Hall, 1961.

________. "Documentation of IPL-V." *Comm. ACM* 6:3. (March, 1963). pp. 86-89.

________., and Simon, H. A. "GPS, A Program That Simulates Human Thought," *Computers and Thought*. (E. Felgenbaum and J. Feldman, editors). New York: McGraw-Hill Book Co., pp. 279-293.

________., and Shaw, J. C. "Programming the Logic Theory Machine." *PEOXI, WJCC*. (February, 1957). pp. 230-240.

________., and Simon, H. A. "The Logic Theory Machine--A Complex Information Processing System." *IRE Trans. Information Theory*. Vol. LT-2, No. 3. (Sept., 1956). pp. 61-69.

________., Shaw, J. C., and Simon, H. A. "Empirical Explorations of the Logic Theory Machine: A Case Study in Pliuristic." *PEOX, WJCC*. (Feb., 1957). pp. 218-230.

Newman, J. *The Harper Encyclopedia of Science*. New York: Harper and Rowe, 1967.

Novick, D. (ed.). *Program Budgeting*. Cambridge, Mass.: Harvard Univ. Press, 1967.

Oakley, C. *The Calculus*. New York: Barnes and Noble, 1944.

Optner, S. L. *Systems Analysis for Business and Industrial Problem Solving*. New Jersey: Prentice-Hall, 1965.

Parker, R. F., et. al. "A Lattice Model of Syntactic Description." *Repts. ML 147 and Others*. Cambridge Lang. Test Unit, July 1961.

Perlis, A. J., and Samelson, K. "Preliminary Report International Algebriac Language." *Comm. ACM*. Vol. 1, No. 12. December, 1958. pp. 8-22.

Peterson, J. and Hasiasaki. *Theory of Arithmetic*. New York: John Wiley and Sons, 1963.

Petseron, T. S. *Intermediate Algebra for College Students* (3rd ed.). New York: Harper and Rowe, 1940.

Petseson, W. W. "Addressing for Random-Access Storage." *IBM JRES Develop.* 1:2. (April, 1957).

Plumb, S. C., and Napper, D. E. *Fortran/360 for IBM System.* Chicago: Science Research Assoc., 1956.

Prywer, N. S., and Gray, H. J., et. al. "The Multi-List Type Associative Memory." Proc. of Symposium of Gigacycle Computer Systems. *AIEE PUBL.* No. S-136. January, 1962.

Randall, C. B. *The Folklore of Management.* Dubuque, Iowa: Little, Brown Co., 1963.

Raphael, B. "Aspects and Applications of Symbol Manipulation." *Proc. ACM.* 21st Natl. Conf., 1966. pp. 66-74.

Raphael, B., et. al. "A Brief Survey of Computer Languages for Symbolic and Algebraic Manipulation," Symbol Manipulation-Languages and Techniques. *Proc. of the IFIL Working Conference on Symbol Manipulation Languages.* D. G. Bobrow (ed.), North-Holland Pub. Co., Amsterdam, 1968. pp. 1-54.

Raphael, B. *Symbol Manipulation by Digital Computer.* Menlo Park, Calif.: Stanford Research Institute, 1966.

Raun, D. L. *An Introduction to COBOL Computer Programming for Accounting and Business Analysis.* Belmont, Calif.: Dickenson Publishing Company, 1966.

Rees, A. "Semantic Factors, Role Indicators," *ASLIB Proc.* 15:12 (Dec., 1963), pp. 350-363.

Retail IMPACT--Inventory Management Program and Control Techniques, *IBM Corp., E20-0188*, Data Processing Division. White Plains, New York.

Rice, J. T., and Rosen, S. "NAPSS--A Numerical Analysis Problem Solving System." *PROC. ACM* 21st Natl. Conf., 1966. pp. 412-414.

Riggs, J. A., Heath, C. O. *Guide to Cost Reduction Through Critical Path Scheduling.* Englewood Cliffs, New Jersey: Prentice-Hall, 1969.

Robinson, G. B. *An Introduction to Mathematical Logic.* Englewood Cliffs, New Jersey: Prentice-Hall, 1969.

Rosen, S. (ed.). *Programming Systems and Language.* New York: McGraw-Hill Book Co., 1967.

Rosenthal, E. B. *Understanding the New Mathematics.* Grenwich, Conn.: Fawcett Publishers, Inc., 1965.

Rubin, M. L. *System Life Cycle Standards Handbook of Data Production Management.* New York: Brandon Systems Press, 1970.

Russell, R. *Algebra Problems.* New York: Barnes and Noble, 1960.

Sackman, H. *Computer System Science and Evolving Society.* New York: John Wiley and Sons, 1967.

Salton, G. "Data Manipulation and Programming Problems in Automatic Information Retrieval." *Comm. ACM,* 8:6, 1965. pp. 391-398.

Samelson, K. and Bauer, F. L. "Sequential Formula Translation." *CACM* (February, 1960), pp. 76-83.

Satterthwalt, A. C. "Programming Languages for Computational Linguistics." *Advances in Computers.* No. 7. (F. L. Alt,and M. Rubinoff, eds.). New York: Academic Press, 1966. pp. 209-238.

Sawyer, W. W. *What is Calculus About?* New York: Singer Co., 1961.

Saxon, S. A., and Piet, E. W. *Programming the IBM 1401.* Englewood Cliffs, New Jersey: Prentice-Hall, 1962.

Scheld, F. *Introduction to Computer Science.* New York: McGraw-Hill Book Co., 1970.

Schlesinger, S., and Saskin, L. "Pose, A Language for Posing Problems to a Computer." Vol. No. 5, (May, 1967), pp. 279-851.

Schwartz, J. I. "Comparing Programming Languages." *Computers and Automation.* Vol. 4, No. 2, (February, 1965). pp. 15-26.

Seeds, H. *Programming RPG II.* New York: John Wiley and Sons, 1971.

Sharpe, W. G. *Basic Introduction to Computer Programming.* New York: Macmillan Press, 1967.

Shaw, D. D. "Assemble or Compile." *Datamation.* Vol. 10, No. 9, (September, 1966). pp. 59-62.

Shubil, J. A. *Business Management.* New York: Barnes and Noble, 1960.

Snoffner, R. M. "A Technique for Organizing Large Files." *Am. Doc.* 13:1. January, 1962. pp. 90-103.

Sippl, C. S. *Computer Dictionary.* Indianapolis, Indiana: Bobbs-Merrill Co., 1966.

Spence, D. D. *Game Playing with Computers.* New York: Spartan Books, 1968.

Spiegel, Murray. *Statistics.* New York: Schaum Publishing Co., 1961.

Spiegel, M. R. *College Algebra.* New York: McGraw-Hill Book Co., 1956.

Sprowels, R. C. *Computers, A Programming Problem Approach.* New York: Harper and Rowe, 1968.

Standish, T. A. "A Data Definition Facility for Programming Languages." Ph.D. Dissertation, Carnegie Inst. of Tech., Pittsburg, Pa., 1967. 292 pp.

Steffered, E. "The Logic Theory Machines: A Model Heuristic Program." The Rand Corporation. *RM-3731-CC.* June, 1963. Santa Monica, Calif.

Swanson, R. W. *Introduction to Business Data Processing and Computer Programming.* Belmont, Calif.: Dickenson Publishing Co., 1967.

________. *Computer Applications to Management Control.* Belmont, Calif.: Dickenson Publishing Co., 1970.

Swets, J. A. "Effectiveness of Information Retrieval Methods." Bolt, Bernard and Newman *Text 1499.* Cambridge, Mass., April, 1967. 47 pp.

Symposium of Automatic Programming for Digital Computers. Office of Naval Research. Dept. of Navy. Washington, D. C., 1954.

Symposium on Advanced Programming Methods for Digital Computers. Office of Naval Research, Dept. of Navy, Washington, D. C., June 28, 1956.

System/360 Bill of Material Processing (360-ME-16X). Programmer's Manual, IBM Corp. 1-20-1246-0. Data Processing Division, White Plains, New York., 1966.

Tague, J. "Evaluation of Statistical Association Measures." *Proc. 1966 Annual Meeting AM Doc. Inst.* Santa Monica, Calif., October, 1966.

The Bank Central Information System-Locate Files, IBM Corp. E20-0138. Data Proc. Division, White Plains, New York, 1967.

Thomas, *Calculus and Analytic Geometry.* Reading, Mass., (4th ed.), Addison-Wesley Publishing Co., 1969.

Thompson, C. E. "Development of Common Language Automatic Programming Systems". *Symposium on Advanced Programming Methods for Digital Computers.* Washington, D. C., June 28th, ONR Symposium Report ACR 15, Office of Naval Research, Dept. of the Navy, Washington, D. C., October 1956, pp. 7-14.

Tritschler, R. J. "A Computer Integrated System for Centralized Information Dissemination, Storage and Retrieval." *ASLIB Proc.* 14:12, December, 1962. pp. 473-503.

Tongue, F. M. "Summary of a Heuristic Line Balancing Procedure." *Computers and Thought.* (E. Felgenbaum and J. Feldman, eds.). New York: McGraw-Hill Book Co., pp. 168-190.

Vazony, A. *Problem Solving by Digital Computers with PL/I Programming.* Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1970.

Walnut, F. K. *Introduction to Computer Programming and Coding.* Englewood Cliffs, New Jersey: Prentice-Hall, 1968.

Wegner, P. *Programming Languages, Information Structures, and Machine Organizations.* New York: McGraw-Hill, 1968.

Weizenbaum, L. "Symetric List Processing." *Comm. ACMI* 9, September, 1963. pp. 524-544.

Welsch, G. A. *Budgeting-Profit, Planning and Control.* Englewood Cliffs: Prentice-Hall, 1971.

Whiteman, I., Jr. "New Computer Languages." *International Science and Technology.* April, 1966. pp. 62-68.

Wilkes, M. V. "Constraint Type Statements in Programming Languages." *Comm. ACM.* 7:10, 1964. pp. 587-588.

———., Wheeler, D. J., and Gill, S. *The Preparation of a Program for an Electronic Digital Computer.* Reading, Mass.: Addison-Wesley Press, 1951.

———. "Lists and Why They Are Useful." *Proc. ACM,* 19th Natl. Conf., 1964. pp. 5-6.

Wolf, F. *Elements of Probability and Statistics.* Englewood Cliffs, N. J.: Prentice-Hall, 1962.

Wooley, G. *Contemporary COBOL,* San Francisco, Calif.: Renegart Press, 1971.

Yngve, V. H. "COMIT." *Comm. ACM.* 6:3. (March, 1963). pp. 83-84.

Young, J. W., Jr. "Non-Procedural Languages." Presented at ACM Southern California Chapter, *7th Annual Technical Symposium,* March 23, 1965.

II. Journal Articles

Automatic Coding (Proceedings of the Symposium of Automatic Coding held

January 24-25, 1957 at the Franklin Institute in Philadelphia). *Journal of Franklin Institute*, Monograph No. 3. Philadelphia, Pa., April, 1957.

Baker, C. L. "The Pact I Coding System for the IBM Type 701." *JACM* 3:4, October, 1956. pp. 272-78.

Barnett, M. P., and Ruchman, W. M. "A Natural Language Programming System for Test Processing." *IEEE Trans.* EWS August, 1968. pp. 45-52.

Bobrow, D. G., and Weizenbaum, J. "List Processing and Extension of Language Facility by Embedding." *IEEE Trans., Eled.*, Comp., Vol. EC-13, No. 4 (August, 1963), pp. 395-400.

Cleverton, C. W., and Mills, L. "The Testing of Index Language Devices." *ASLIB* Proc. 15:4, 1963. pp. 173-179.

Climenson, W. D. "File Organization and Search Techniques in C.A." Caudra ed. *Annual Review of Information Science and Technology*. Vol. I. Interscience, New York, 1966. pp. 136-197.

Doyle, L. B. "Semantic Road Maps for Literature Searches." *J. ACM* 8:4. October, 1961. pp. 553-578.

Green, B. F. "Computer Language for Manipulation." *IRE Trans. Eled. Comp.* Vol. EC-10, No. 4, December, 1961. pp. 729-735.

Hext, J. and Roberts, P. "Syntax Analysis by Domolki's Algorithm. *Comp. J.* 13. (August, 1970). pp. 263-271.

Klerer, M. and May, J. "Automatic Dimensioning." *Comm., ACM.* No. 3 (March, 1967). pp. 165-166.

O'connor, J. "Automatic Subject Recognition Scientific Papers," An Empirical Study. *ACM* 12:3. (October, 1965).

Wirth, N. "L360, A Programming Language for the 360 Computers." *I, ACM.* 15:1 (January, 1968). pp. 37-74.

Steele, T. B., Jr. "PATRIAI" *I, ACM.* Vol. R., No. 1, (January, 1957). pp. 8-11.

III. Unpublished Materials

Angell, Thomas. "Automatic Classification as a Storage Strategy for an Information Storage and Retrieval System." Unpublished Masters Thesis, University of Pennsylvania, 1966.

INDEX

INDEX