OREGON STATE UNIVERSITY COMPUTER SCIENCE DEPARTMENT

Performance Evaluation of Three Microcomputer Based Systems in a Small Business Dataprocessing Environment

T. G. Lewis
Computer Science Department
Oregon State University

78-1-6

Performance Evaluation of Three Microcomputer Based Systems in a Small Business Dataprocessing Environment

T. G. Lewis
Computer Science Department
Oregon State University

78-1-6

Performance Evaluation Of Three Microcomputer Based Systems In A Small Business Dataprocessing Environment.

T. G. Lewis, Ph.D.
Associate Professor
Computer Science Department
Oregon State University
Corvallis, Oregon 97331
(503) 754-3273

A. INTRODUCTION

In March, 1977, a team of students under the guidance of the author began studying three personal computer systems to determine their strengths and weaknesses when placed in a data processing environment. The three systems were chosen to represent a broad spectrum of contemporary microcomputer based data processing equipment. System W is a firmware BASIC computer originally designed for scientific applications but found being used in a variety of business processing applications. System T is a storage display graphics computer based on the Motorola 6800 microprocessor ROM BASIC, and ROM operating system. System D is an Intel 8080 based computer designed specifically for business data-processing.

The evaluation team set about to answer five (5) broad questions using intuition, reason, and hard facts collected from actual use of the systems. The computers were subjected to a variety of tests, used to develop several "typical" data processing application programs, and evaluated from the standpoint of a first-time user with very little computer background or training.

The team wanted to answer broad "selection" questions, and specific performance questions. The broad questions were:

I. How do the three systems compare in cost with other systems?

The comparative analysis ruled out differences due to cost by using roughly equivalently priced systems.

II. How well suited are the systems to business data processing applications?

We developed a set of criterion thought to be important in the business environment. Then we used the microcomputers to implement a variety of applications in business. These application programs, and the problems encountered in attempting to implement them, gave the team a basis for evaluating the systems as data processing systems.

III. How effective are the documents and manuals?

The objective of this part of the study was to place the team in the role of a first-time user and determine if the manuals and documents were adequate.

IV. How well does the hardware and system software perform in "typical" data processing applications?

The team's goal in this part of the evaluation was to perform a variety of benchmark tests and use the results to compare performance. The benchmarks were selected to be simple, repeatable, and oriented toward a specific subcomponent of each system. For example, the team was able to determine the effect of a random access diskette subsystem versus a sequential access tape subsystem on the performance of the total system. This result is useful in predicting performance gained by replacing tape with diskette.

V. What should be done to improve systems W, D, and T when used in the dataprocessing environment?

A great deal was learned about performance of all the systems studied. The lessons are summarized in this final portion of the report. The recommendations include new ideas, combinations of old ideas, duplication of features found in one or the other systems, and small or minor modifications that contribute to increased desirability of microcomputer systems in small data processing applications by novice users.

B. HOW DO THE THREE SYSTEMS COMPARE IN COST WITH OTHER SYSTEMS?

This part of the study briefly details three systems used in the comparative analysis: machines W, D, and T.

Microcomputer W

The W and T systems share a common conception. Both machines were originally designed for sophisticated, engineering oriented users. Both systems were built around an interpretive BASIC language and command interpreter stored in a firmware ROM. Both systems were originally designed as an "intelligent" terminal containing only a cassette tape drive for storing programs.

The W processor is claimed to be an extremely fast processor. It features a large, bright, character CRT, aesthetic terminal, and well engineered keyboard. The basic price of the W is $9,200 for a 32K memory version.

The W may be connected to other devices and/or telephone lines via an intelligent interface. This interface contains an Intel 8080 processor and ROM that simulates a variety of common protocols. This gives the system a limited amount of communications capability at very low cost.

W's BASIC interpreter, operating system commands, and system control reside in a 42.5K byte ROM which may be expanded to include additional scientific and/or business related functions. For example, in addition to an editor ROM, an optional sort/merge ROM, General I/O ROM, and diskette file maintenance ROM (KFAM, keyed file access method) may be purchased at additional cost.

Additional devices for printing and storage may be included in a total W system. Dual diskettes and a printer increase the purchase price to over $20,000.

W's BASIC system is completely described in a single, easy to use, easy to reference, and aesthetic manual. While a beginner may have difficulty understanding many of the terms found in this manual, the team's evaluation placed the W manual far above the others evaluated with respect to "usefulness."

The W is an attractive machine, utilizing colorful hardware and aesthetic enclosures likely to be admired by a businessman. The only feature possibly resulting in unfavorable comment by a businessman or novice user is the keyboard. The keyboard is too cluttered, containing additional control keys (beyond a typewriter) and dual designation to assist a BASIC language programmer. The inclusion of BASIC and operating system keywords is viewed as a cryptic remnant by naive business users.

Additional features that may be important to a user in selecting a small business system also found in W include: atomization of keywords (short tokens replace the keywords to save memory space), and multiple statements per line (saves up to 4 times the space).

Microcomputer D

The D system was originally conceived as an intelligent terminal designed to replace IBM terminals. As a result of this orientation, the D offers the most sophisticated communications options of the three systems studied here. IBM emulators, autodial peripherals, and network software is provided as part of D's operating system. Therefore, a user is able to communicate with other computers without programming or purchasing an interface (the purchase price includes a communications adapter, but may be dropped to lower the price). The basic processor price of $8,500 places this microcomputer slightly lower than the other two systems analyzed here.

The D was selected because of its business orientation and because it is designed around an Intel 8080 architecture. BASIC and other languages (COBOL) are software translated rather than pre-stored in ROM. The resident operating system is also a software system taking nearly 12K of the 16K main memory. In spite of this load, the D machine as a stand-alone minicomputer system for business data processing, is in many ways more suited to business applications than the two other systems evaluated. The reasons are:

1) supports virtual memory,
2) supports sequential, random, and inverted file structures,
3) supports uniform record and file formats for a variety of languages,
4) can operate unattended under control of a time-of-day clock
5) supports protected files, error detection, and error recovery procedures.

D supports a variety of languages, but for purposes of comparison we evaluate only the BASIC interpreter here. This interpreter is not typically used to program solutions to business problems, however, because other languages are available that are more suited to data processing than BASIC (virtual memory, forms control, user prompting, error recovery, unattended operation, communications interfacing, and numerous other features that partially meet the goals of business computing). The D machine, with two diskettes, 16K main memory, a CRT, and line printer was used in this evaluation.

D's BASIC is a minimal version of Dartmouth BASIC with few useful features for business applications. It is a software interpreter developed over 5 years ago, and whose further development has not been pursued by the company.

D runs under control of an operating system that includes autokeyed operation for unattended operation, communications utilities for (DAA) autocalling other systems, merge/sort utility, file maintenance utilities, and file security (Delete, write protect) utilities.

The BASIC interpreter does not run under the virtual memory mapping interpreter, hence, we were unable to measure its performance. BASIC does support sequential and random access to disk files, however.

The D system was judged by the evaluation team as an aesthetically pleasing system. It was considered by some of the team members to be the least obtrusive system for a novice user of the three systems we tested. There are only 5 keys on the keyboard that are not found on a typewriter, and the console, diskette, and CRT are housed in one attractive desk. The console CRT screen, however, is much too small for easy viewing.

The D system is described in several manuals, each one easy to index into, but in many cases not clearly written.

Microcomputer T

The T computer system was originally conceived as an intelligent graphics terminal. It consists of a keyboard, cassette tape, and storage scope CRT.

The system is driven by a Motorola 6800 processor under control of 32K of ROM. The ROM system contains a BASIC interpreter with extensions for graphics picture processing, text editing, control of I/O devices, and a moderate level of autokey operation.

The storage screen has a hardware viewport of 100 x 135 grid points. It has high resolution capability, but the screen is very dim and difficult to view in normal light. Plotting speed is moderate, and since it is a storage tube, interactive graphics is very restricted.

Additional devices were not available to the study, but a Diablo printer was connected to the RS232-C port for the duration of the tests. The system without printer was priced at $11,500 (32K memory). This placed machine D at the top of the price scale when compared with the W and D systems.

The T system is described in several extensive manuals, mostly overlapping in content. Indeed, the manuals contributed to a level of confusion because they were too prolific.

Additional communication is provided by an IEEE Standard GPIO bus and firmware to allow access to other devices from BASIC. The system may be further enhanced in this direction by gaining access to the 6800 processor through the BASIC interpreter's CALL primitive.

The processor is enclosed in an attractively styled box, but the rather austere color and confusing keyboard may alienate many novice businessmen. The keyboard is designed for a software developer rather than a naive user.

Comparison Summary

The W, D, and T microcomputer systems were chosen as benchmarks for the study because of the following:

1) They are among the dominant microprocessor-based systems currently in the small data processing market.
2) They were originally designed for application extremes of scientific, graphical, and business environments.
3) They were readily available.

The results of performance evaluation reveal strengths and weaknesses of all three systems. An examination of each system is essential in order to conclude which is "best" for a specific application.

C. HOW WELL SUITED ARE D, T, AND W TO BUSINESS DATA PROCESSING?

APPLICATIONS

The suitability of any computer system for use in a data processing environment depends on the ability of the system to perform the data manipulation tasks of the given environment. While data processing covers a broad range of computer applications, there are certain basic capabilities that are required by most data processing environments. Data processing generally involves the functions of data entry, data storage and access, calculations with data, and the preparation and printing of reports. These basic processing functions may need to be performed separately, together, or not at all, depending on the application.

Some common data processing applications which use these basic processing functions are listed below.

A. Accounting Functions:

1. Payroll file processing.
2. Sales record processing.
3. Expenditure record processing.
4. Mortgage or loan payment schedules.
5. Depreciation schedules.
6. Tax Calculations.
7. Simulation and trend analysis.
8. Updating of current budget and general ledger.
9. Calculations of customer billing and services.
10. Calculations of computer usage and cost.

B. Report Writing Functions:

1. Printing of payroll checks.
2. Printing audit trails on all processing and file updating.
3. Printing sales reports.
4. Printing expenditure reports.
5. Printing budget reports (current and annual).
6. Printing net and gross profit reports.
7. Printing bills for customers.
8. Printing inventory reports (price and level control).
9. Printing mailing lists.
10. Printing advertisements (media Preparation).
11. Printing file inquiry reports.
12. Printing of government and other agency report forms.

C. Database Management Functions (Updating, Data Entry, and Creation):

1. Maintain employee records such as salary, benefits, tax, and other personal statistics.
2. Maintain inventory file of parts, stock on hand, and equipment used by the business.
3. Maintain a budget file for current and annual expenditures and income (including liabilities, assets, and accounts receivable).

4. Maintain current customer billing, service records, and mailing lists.

The applications of a small computer system when placed in a data processing environment, reflect a set of well defined objectives and rules. These rules, once discovered, can be used by a computer manufacturer to improve the product.

The evaluation team was able to condense many of the thoughts and lessons learned into the following rules. Awareness of these rules can be a great aid in understanding the goals of a well designed business system.

RULES OF SMALL COMPUTERS IN THE BUSINESS DATA PROCESSING WORLD

There are a few known "laws" of small business computing that guide the current market. The following informal laws are called personal computing maxims because the main feature of the small business system is its impact on personal computing.

1). Personal Computing Equals Interactive Computing: The "personal-ness" of a computer increases directly proportional to its interactiveness.

The trend of the past decade has been toward more personalized (humanized) systems. Personal computers require little training to use, little maintenance, and no computer specialists to manage.

From batch to remote-job-entry, and eventually to timesharing; computers have tended toward more dispersed useage. In the last 8 years the trend has been even more highly oriented toward stand-alone, dispersed, systems. In fact, hardware/software integration has lead to turnkey systems requiring very little involvement on the part of a user.

2). As The Power Of A Personal Computer Increases, Its Price Decreases.

This law states a counter intuitive observation; "as the power quadruples, the price doubles." A very rough measure of power follows:

$$\text{Power} = \frac{\text{MIPS} * \text{STORAGE CAPACITY}}{\text{COST}}$$

MIPS = Million instructions per second

STORAGE CAPACITY = Characters of Mass (on-line) Storage in bytes

COST = Purchase price

As a system's capability increases, the number of problems it can solve increases. The increased base of applications leads to greater sales volume which in turn leads to mass production, quantity discounts, and a general decline in unit cost.

As a good example of this phenomena, observe the effect of increased cpu speeds in the mid-1960's. Greater increases in MIPS and STORAGE CAPACITY supported timesharing. As a result of timesharing, the cost per unit of computation declined.

Currently, increased storage density is propelling development of large databases. The database craze is creating more applications, leading to decreases in storage device costs.

The logical conclusion of the second law is that hardware will become so inexpensive as to remove itself from consideration in a small business system. This effect has already removed the cpu from dominance in computer selection. This leads to the next maxim.

3). Software Is Hard, Hardware Is Soft: It is economically more feasible to build a computer than to program it.

Clearly, the increasing cost of software, and the increasing complexity it represents is the major cost factor in contemporary computer based systems. There are two consequences of this rule:

a) Programs and data should be shared, but hardware should be replicated.

It is far better to replicate a hardware unit than to risk increased software complexity by placing burdens on the software. For example, a multiterminal system may be desired if common access to a database is needed (shared data). The most economical solution to this problem dictates that each terminal contain its own cpu rather than burden a central operating system with multiplexing algorithms. This philosophy is seen in at least one commercially available system in which dispersed processors are connected to a central disk processor.

The second consequence is this:

b) Timesharing failed: people couldn't understand it.

Complex software has a tendency to be visible to a user, whereas hardware has a tendency to be transparent (vanish). Exposing a businessman to complex operating systems (OS/360), languages (COBOL, PL/I), and operating procedures (switches and knobs) alien to the operation of his business, is counter to personal computing.

A proper man-machine interface is most important in a data processing environment. The "best" system is one that requires very little effort to use. This is succinctly summarized in the next law.

4). Knowledge Costs More Than Software and Hardware:
The usefulness of a personal computer increases inversely proportional to how much people need to know in order to use it.

Past systems have failed due to the amount of training required (JCL, telephone modems, jargon), the myths ("it's complicated"), and poor service that results when a user is unwilling to pay for advice. In short, the system selected by a businessman must appear to be simple, elegant, and aesthetic.

5). The Color, Shape, And Size Of A Personal Computer Is More Important To A User Than What Is Inside Of It.

The "package" is the message in data processing environments. The system should require as few buttons, switches, lights, and knobs as possible. The least amount of jargon (especially in manuals), and the least number of steps to follow for a given procedure is the most desirable.

Service fills the gap between a user's knowledge about computers, and the computer's lack of capability (in human terms). Since service is a very expensive commodity, it is to the manufacturer's advantage to minimize the need for service. This can be done by conforming to the rules of simple use, elegant appearance, and uncomplicated interactiveness built into the system.

Finally, there are several aspects of interactive computing that impact software. In particular, effective high level languages and hidden operating systems are critical to a simple, aesthetic, and articulate interface with business application users.

6). BASIC Is To Personal Computers As Sign Language Is To English.

English has it's "pig Latin", and programming has BASIC. Like speaking in pig Latin, BASIC programs are easy to write. Unfortunately they are hard to understand. Few systems permit indentation, structuring, comments (without memory penalty), and full control over the system. Some common shortcomings of BASIC are:

a) poor error recovery facilities,
b) dynamic memory mapping of programs too large for main memory,
c) restricted data structures,
d) limited user prompting,
e) inadequate software security and protection - copyrighting, file security, interlock mechanisms for shared files.
f) slow executions - interpretive,
g) inadequate primitives for data processing, e.g. no sorts, file access constructs, or forms handling constructs.

One compensating advantage of BASIC is that it is easily implemented, easily learned, and rapidly put to use in developing programs. It is an ideal language for beginners and for implementing small programs that will eventually be thrown away rather than modified.

7). An Operating System Is A Feeble Attempt To Include What Was Overlooked In The Design Of A Programming Language.

Clearly, the operating system of most systems represents nothing more than an extension of the programming language used to develop an application. A job-control-language is usually invented as a remedy to the inadequacy of the language. In interpretive systems, the operating system disappears into the language interpreter.

The systems analyzed in this evaluation do not realize all of the desired features stated here. How well each system meets these criterion must be judged after examining data presented in subsequent sections of this report. The following "ultimate" law should be kept in mind while examining the data.

8). The Ultimate Personal Computer Is A Robot; The goal of personal computing is to reduce the differences between people and computers.

The personal computing maxims suggest the following general evaluation criterion for systems D, T, and W. We measure the effect of;

A. Interactiveness,
B. Programming,

C. Training A User, and
D. Making The System Secure and Fault-tolerant.

GENERAL CAPABILITIES

All three machines were considered interactive, small computers by their manufacturers. One of the main reasons given was the use of "interactive" BASIC as a programming language. Therefore, it was suprising to discover that the actual dataprocessing capabilities of these machines was considered modestly interactive, only.

Interactive BASIC means the developer of software can interact with the processor during debugging of software. The user, on the contrary, is unable to interact in many essential ways with these systems. Here are some of the reasons why interaction was restricted.

All systems tested were too limited in file storage capacity to support on-line, interactive databases. Processing steps had to be divided into separate batch operations before data entry, retrieval, and report writing was possible.

System T (cassette tape) was totally unsuited to active retrieval of records from its (tape) files. System W (diskette) made small databases possible, but file structuring was very limited in its version of BASIC. System D offered the greatest capability with indexed access, sort, and reformat utility routines built into the operating system. Only System D provided delete, and write protection on files. None of the systems provided file semaphores for multiaccess from two or more concurrent processes.

In general, user interaction on a small computer depends on the availability of large mass storage and a multiaccess operating system with provisions for file security. These features were lacking in the three microcomputer-based systems tested.

Programming of these systems was also hampered by restrictive BASIC. In all cases, BASIC fails to give the systems developer access to machine-level control. This means the following activities are severely restricted.

1. Keyboard/CRT formatting and prompting.
2. Dynamic (automatic) program overlay.
3. Application level detection and recovery from processing exceptions.

System T allowed the greatest amount of keyboard/CRT formatting because it included a graphics display screen. It also allowed modest detection capabilities for error recovery, e.g. end-of-tape, overflow, record type error.

System D provided a second language (besides BASIC) that handled forms, cursors, bells (audible prompt), and data entry checks suitable to data processing. Furthermore, it included dynamic program overlay in this "other" language which made the second criteria above a solved problem. Programs of any length could be compiled and run on machine D within the limits of diskette space. Furthermore, system D detected and recovered from all errors trapped by the operating system. System D provided the best programming system of the three microcomputers, but only through a non-BASIC language. We did not test this language against the other systems because the T and W machines had no equivalent language to compare with.

User training was judged to be the greatest for machine T, due to its scientific orientation, large number of knobs and switches, and graphics capabilities. Machine W was next in line as the most complicated to use. W contained a well designed keyboard, but the user had to be trained to use it. System D, again due to its business orientation, had the fewest number of switches and knobs (5 more than on a typewriter).

The only secure and fault-tolerant microcomputer tested was machine D. While system T had an "encode" feature for scrambling the BASIC source code, it failed to provide file protection for data. System W failed to provide a way to enter a password without it being displayed on the screen.

The security and fault-tolerance of system D stemmed from:

1) All errors are trapped to the application program.
2) All files may be backed-up and restarted without explicit commands from the user.
3) All user commands must be filtered through an application program. Thus they can be checked before being passed on to the operating system.
4) The user does not see the language, error messages, or operating system commands.

In short, the user of system D is protected from "computerese" by the application program that acts as an interface to him or her. Therefore, all interaction is done in the language and terminology of the application (accounting, real-estate, etc.).

SUMMARY

The three systems were found to have strengths and weaknesses when placed in the business dataprocessing environment. Their major strengths were low cost and generality. Their major weaknesses were lack of user interaction, good programming systems, ease of use, and suitable exception checking and error recovery. These are serious shortcomings that must be overcome in future microcomputers in business.

Next we examine the benchmark programs used to test the three systems. These were selected for their diversity and representativeness of business systems.

D. BUSINESS APPLICATION BENCHMARKS

The application programs include: (1) An accounts receivable program (data entry, file update, and posting); (2) A general ledger program (data entry, file update, and simple accounting); (3) a mailing list program (sorting); (4) and a database retrieval program (file creation, update, and inquiry). Also, to evaluate the graphics capability of system T, a program was developed to draw pie charts for the general ledger program data.

General Ledger Program

The general ledger program uses a chart of accounts file which contains the budget for the year, the year-to-date income and expenses, and the current

month income and expenses for each item in the chart. Each month a sequential transaction file is created to update the year-to-date totals and calculate the balance (income-expenses) for the month that has just ended.

Accounts Receivable Program

The accounts receivable program is a typical data processing application program used by a business. The idea is to maintain a customer data base with a master file and a transaction file. A program must create transaction files containing customer charges. Another program is used to update the master file and print bills for customers owing a payment each month. Three programs were written for this benchmark. The first is used to create the master file, the second is used to create the transaction file, and the third program is used to merge the master file with the transaction file, compute balances, print a statement for each active account, and update the master records.

Mailing List Program

The mailing list program comes from a problem which required a large list of names and addresses to be sorted by zip code. The program has several versions. One version uses a selection sort with arrays in memory. Another version uses a merge/sort on arrays in memory. The last version uses a merge/sort on mass storage files.

Database Retrieval Program

The database retrieval program is another typical data processing application which involves the creation, update, and inquiry of a database. The program consists of two segments. The first segment is used to create the database. The second segment is used to allow additions, updates, or inquiries-by-attribute to be made of the information in the database. Index files are used to access the records in the database. The indexes store information about the existence and location of records stored in the database. A system of chains is used with a hashing function based on multiple valued attributes. The program performs retrieval on a real-estate database (multiple listing file) where each attribute describes a feature of a house for sale. Four attributes were used: price, location, number of bedrooms, and number of bathrooms.

RESULTS OF THE BENCHMARKS

In the process of converting and running the test programs listed above, several strong and weak points were noted. All of the application programs were able to be converted and run but in machine T some features had to be eliminated where the application depended upon a disk based computer system.

In systems T and D an autostart feature allows an inexperienced user to place a tape or diskette in the slot and press one button to execute a canned program.

Ability to control the CRT display from the program allows menus to be displayed for a user to make a selection and then have the screen automatically erased to avoid cluttering of information and avoid having the user worry about page control.

Entering programs or data is made easier by a backspace key, clear key, and rubout key. However, once a program is created, editing is very limited in the three systems. A programmer is thus unable to search and replace characters. The insert and delete operations require using BASIC line numbers or the DELETE command of the operating system.

In system T and W, programs can be modified to a limited degree, but data and text files cannot be easily changed, therefore hindering program debugging. In system D, all data is stored in ASCII and can be read, modified, and deleted by the editor without concern for conversion.

Error messages generated at run-time are often difficult to understand to a programmer and even more confusing to an inexperienced or first time user. In system T the ON THEN statement allows some error recovery to be made in application programs. But error trap and restart routines are difficult to write due to a lack of convenient instructions. System W failed to provide any means of exception handling. System D's "other" language traps all errors in a single common area accessed by the application program.

The most severe limitation found on these systems was in the reliance on magnetic tape for mass storage of data and programs. The tape commands and medium are easy to use for some data processing applications, but not all applications in business data processing are well suited to sequential processing.

In the test programs which were used for this report, the accounts receivable program and the database retrieval program required the capability of rewriting records in order to do updating of the files. To simulate these updating functions on a sequential tape system requires either rewriting the updated files or the use of index files which can be changed more easily than the master file, and then rewritten.

In doing file-to-file interactions the tape system requires each file access to start at the tape's beginning each time an access occurs. The program must find the first record in a file and then do dummy reads until the desired record is found. On large data files, it is very time consuming to do random inquiry.

Also, it is almost impossible and impractical to do a sort/merge operation on data stored on a sequential tape. Other limitations noted with the tape usage were the inability to erase a file in the middle of a tape cartridge, and the lack of good security commands for the tape files. The MARK statement in system T was considered harmful because it destroys the contents of a file, sometimes inadvertently. Furthermore, if a user inadvertently uses MARK in the middle of a tape, all files following the file which is marked are effectively lost. This feature of the tape system greatly inhibits rewriting of data files for application programs.

A beneficial feature found on W was its extension through added ROMs. An optional data processing ROM contains operations for file manipulation, sorting and merging, and other data file utility programs. Such an addition helps decrease overhead in programs which perform common data processing tasks with files.

E. HOW EFFECTIVE ARE THE DOCUMENTS AND MANUALS?

The manuals as such were very well written. Example programs and a generous number of illustrations in the T and W manuals shortened the learning time considerably. The number of manuals for system T made it difficult to figure out which manual to look into for a particular piece of information.

We discovered a few guidelines for manuals as follows:

a) a quick reference manual with commands and procedures commonly used. This manual would have sufficient information in a concise manner for a user to sit at the machine, switch it on (where is the on/off switch!), and enter, run, edit and create a file of a program.
b) a BASIC language and system reference manual for users who need more information on the system, language and the commands.
c) an application manual with example programs to acquaint new users with the system and suggest applications that a new user might not think about.

F. HOW WELL DOES THE HARDWARE AND SYSTEM SOFTWARE PERFORM IN A "TYPICAL" DATA PROCESSING APPLICATION?

To answer this question we will look at two areas of performance:

1) Language/Processor speed,
2) Processor/peripheral speed.

All three systems were programmed in BASIC and programs were written as closely identical as possible in order to give an accurate comparison between systems.

Our aim was to get comparisons rather than to develop precise timing data.

To evaluate language/processor speed a series of benchmark programs were run on all three systems. The benchmarks performed the following operations.

a) Empty Loop Performance
   (1) Single Loop
   (2) Nested Loop

b) Calculation Loop Performance
   (1) Assignment Statement
   (2) Add and Subtract
   (3) Multiply
   (4) Divide
   (5) False "If"
   (6) True "If"

The programs were executed for 100, 1,000, 10,000 iterations and the times compared for each machine.

In all cases the T microcomputer performance figures fell between the W and D system's. The W was always fastest of the three systems but the time margin does not appear to be significant. The T machine was always faster than the D machine by a significant margin.

A series of processor/peripheral speed benchmark programs were written for all three systems. In this area we measured the speed of the following:

(1) Tape speed of T compared to the disk speed of W and D systems.

  (a) Sequential Read/Write
  (b) Merge/sort on Auxiliary Storage
  (c) Sequential Update of a file on Auxiliary Storage

The tape of system T compared very unfavorably with the disk based systems when two or more files were alternately accessed, or random accesses required. In both the merge/sort test and sequential update test the performance of the T system's tape significantly increased the required processing time.

Also programming became significantly more complex and inefficient on the T system when two or more files were involved. Efforts to simulate random accesses to tape records and update entries were hopeless.

Comparison of display speed revealed T to be slower than W. But this time margin is not felt to be significant since no detectable wait is experienced by the user. In other words, information is displayed at least as fast as the user can absorb it.

CONCLUSIONS

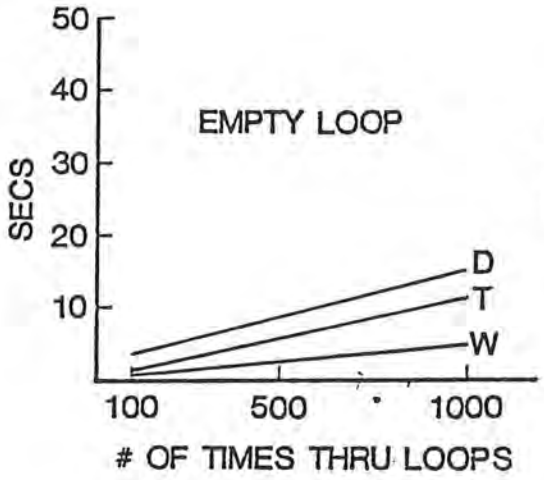
A number of conclusions can be obtained by examining the difference in measured performance, see the graphical data. First, the arithmetic capability of the 6800 microprocessor used to support the T machine is roughly equivalent to the arithmetic capability of the W processor.

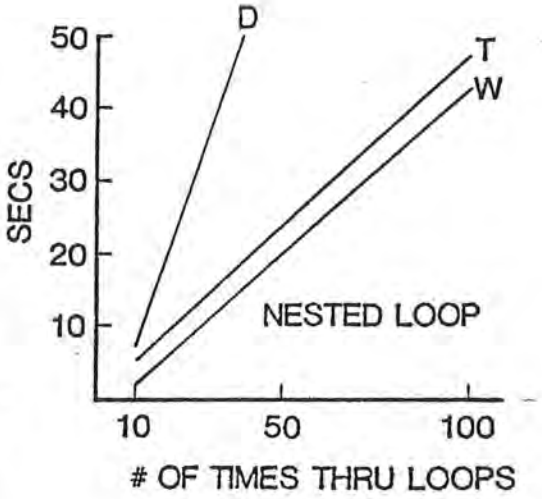
W has an unknown processor, but it is tuned to BASIC and for processing speed. The D processor sacrificed speed for control of arithmetic round-off and file record uniformity. All D microprocessor (8080) calculations are performed as string (character) manipulations using ACSII encoded numerals. D machine requires no internal conversion or additional record formats for binary versus character storage (all files can be interrogated as an aid to debugging).

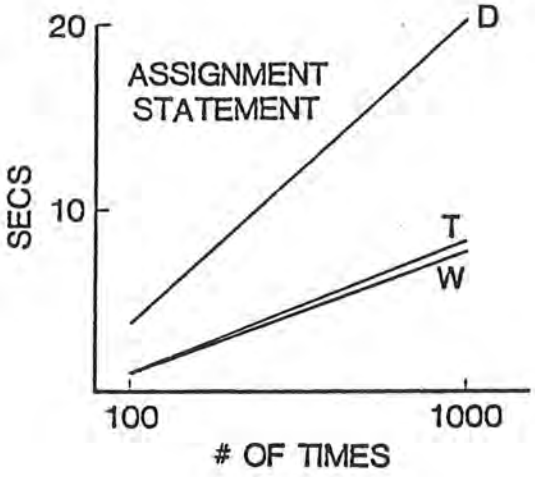
The T machine divide algorithm is slower than expected. This may be due to a hardware divide in W as compared to the software divide implemented in the 6800 ROM of machine T. The team was not able to verify which divide algorithm is used by the W system.

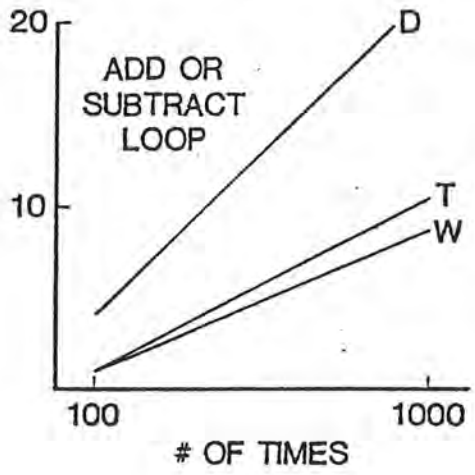
The implementation of BASIC appears to be much more efficient in the W system than in the other two systems. Loop, PRINT, IF, and other BASIC statements are performed much more quickly in W than in the other two systems.

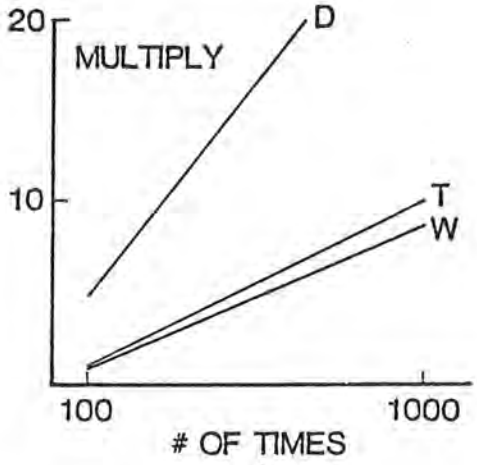
The performance ratio of I/O to mass storage reveals several important subsystem features. First, tape systems can compete only with program files that are read sequentially. Nearly all applications requiring data files destroy the usefulness of tape systems.

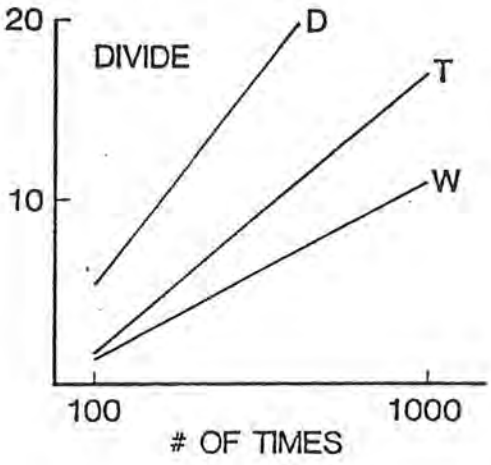

50
40
30
20
10
SECS
EMPTY LOOP
D
T
W
100
500
1000
OF TIMES THRU LOOPS

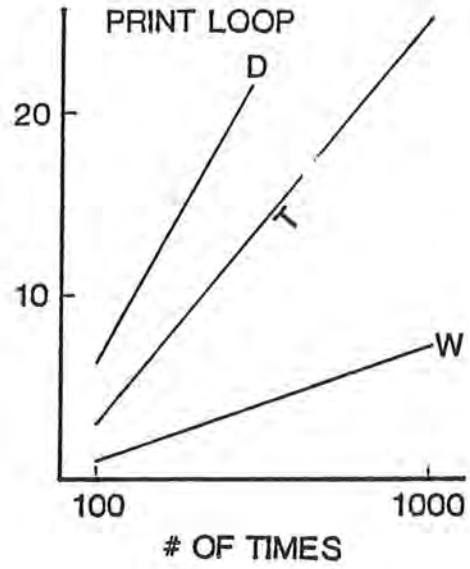

50
40
30
20
10
SECS
D
T
W
NESTED LOOP
10
50
100
OF TIMES THRU LOOPS

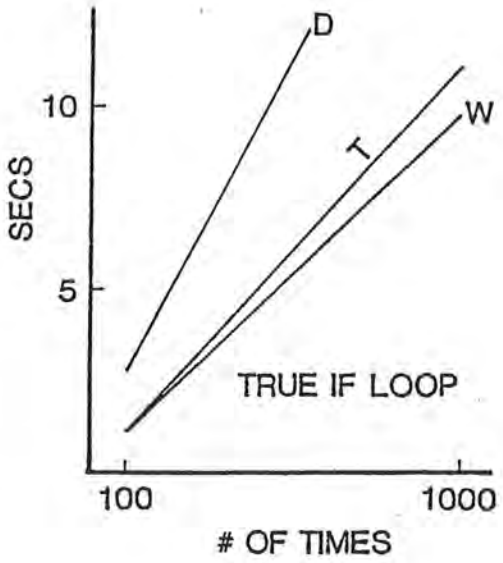

20
10
SECS
ASSIGNMENT
STATEMENT
D
T
W
100
1000
OF TIMES

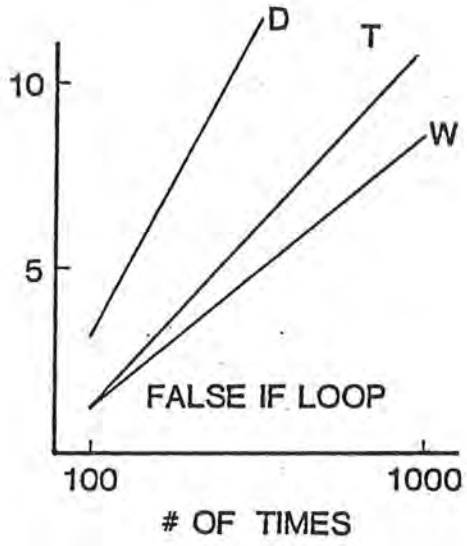

20
10
ADD OR
SUBTRACT
LOOP
D
T
W
100
1000
OF TIMES

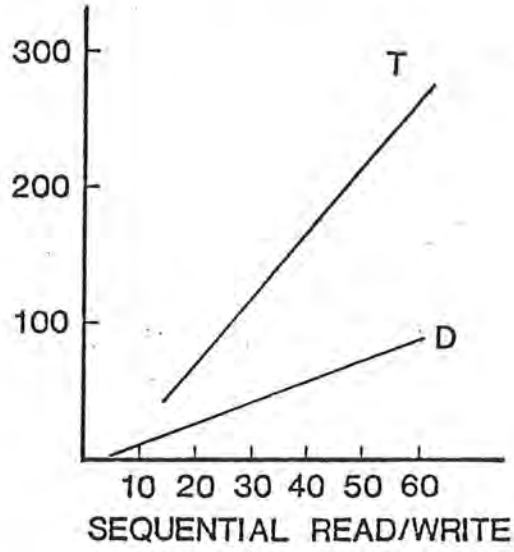

20
10
MULTIPLY
D
T
W
100
1000
OF TIMES


20
10
DIVIDE
D
T
W
100
1000
OF TIMES

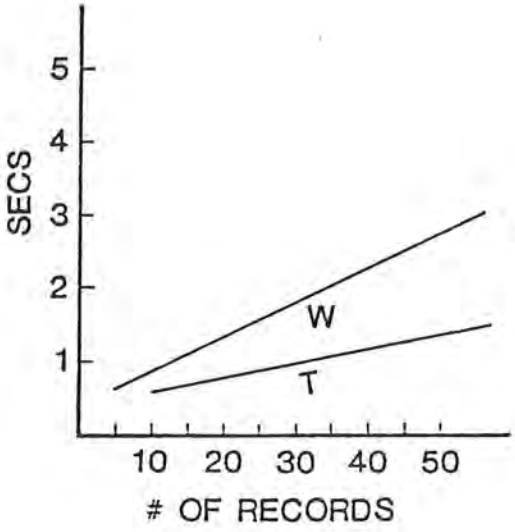


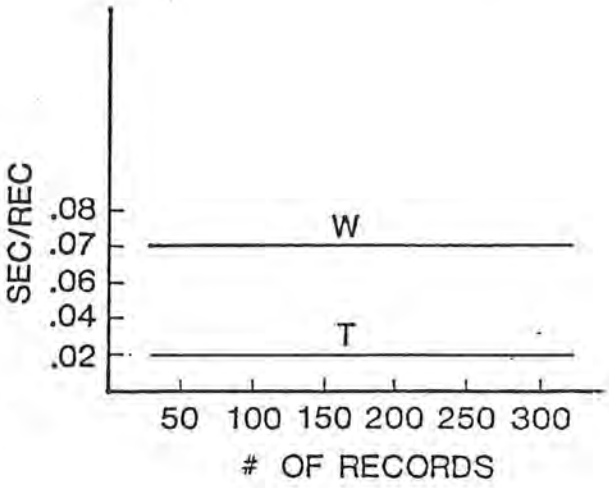




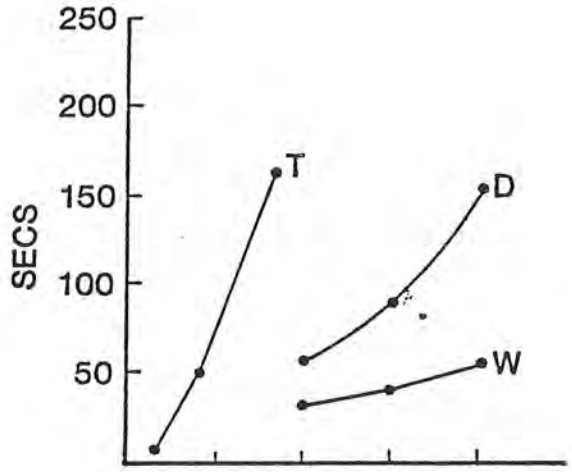


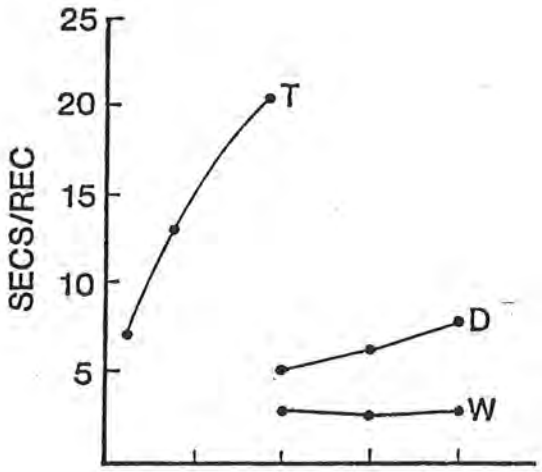
READ ONE FILE-UPDATE RECORD-
WRITE ON ANOTHER FILE



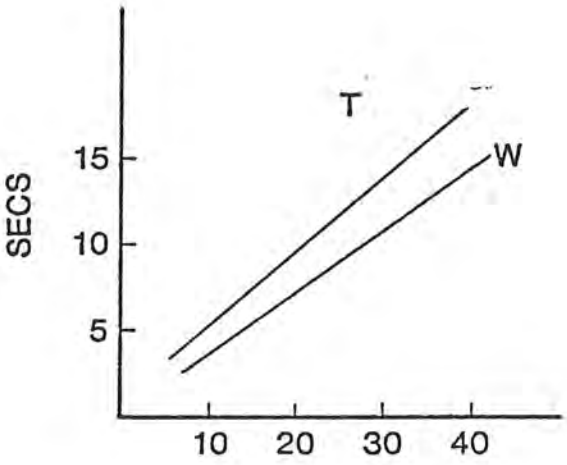


SEQUENTIAL READ/WRITE DISK VS TAPE





MERGE/SORT ON AUXILLARY STORAGE



MERGE/SORT OF ARRAY

The D diskette system performs slower than the W diskette system, even though they both use the same diskette drives. Why is there a difference? The D system includes dynamic space allocation that automatically allocates more space to a growing file. The W system forces a programmer to provide space allocation. Diskette access times for the D microcomputer were much faster when repeating the benchmarks on an allocated file. The lesson learned is that automatic allocation costs system performance.

G. WHAT SHOULD BE DONE TO IMPROVE SYSTEMS W, D, AND T WHEN USED IN THE DATAPROCESSING ENVIRONMENT?

When evaluating the performance of these systems there are areas of consideration important to the programmer, and others of significant value to the end user. First, we shall analyze features for software development which appeal to a programmer.

TO THE DEVELOPER

The use of a BASIC interpreter rather than a BASIC compiler is a definite advantage to the programmer. An interpreter significantly reduces actual program development time by eliminating waiting for typically slow compilations. As the cost of personnel increases and the price of the computer decreases, this cost in lost time is magnified.

Program development time is also decreased when BASIC allows multiple-statement lines. System T recognized only one BASIC statement or command per line. This increases coding, typing, and listing time, all of which are expensive.

A feature which enhances the readability of listings is automatic indentation of FOR - NEXT loops. In addition to improving cosmetics of the code, it enables a developer and any future programmers to better understand the logic and flow of the program.

A feature which aids both the programmer and the end user is the ability of a system to "trap" any fatal and non-fatal errors as they happen. In turnkey business applications, it is essential that this feature be included. When an error of any kind occurs, the diagnostic message should not be displayed on the screen for the naive user to worry about. A turnkey software package should have all provisions to internally cope with these errors and explain, in English, how the user should proceed to correct the error, be it system or user caused.

Block and context editors are also valuable to a programmer. They enable a programmer to cut actual code entering time by allowing him to use sections of code previously entered in other programs, and by reducing the need to line edit. It should be possible to scan an entire file for the occurence of a character string, and it should be possible to replace one character string with another.

Provision should be made for cursor and data entry control. Having exact cursor control enables a programmer to design forms on the screen which closely resemble paper forms the user is keying from, and create a cosmetically attractive display.

During the design process of a turnkey business system, program protection is essential. The ability to give file names is also important when dealing with a complete system of perhaps twenty or more discrete programs. The lack of these in system T and partially in system W was a source of constant annoyance to the evaluation team.

Furthermore, without "read protect", increasingly important since the advent of the Privacy Act, anyone with access to the machines and the magnetic tape cartridges has access to all data contained therein.

The implementation of user defined passwords greatly reduces the privacy problem, but programs must also be protected for their owners. A "secret" program can still be executed even though it may not be alterable. This remains a problem for all systems studied.

Virtual memory allocation is very important when designing turnkey business packages because most small systems must frequently overlay various subroutines. Currently, provisions for overlaying subroutines is very awkward to use in BASIC, and non-existent on other microcomputers.

If possible, improved diagnostics should be provided to the programmer. Time is lost when it becomes necessary to look up a diagnostic code in the user's manual. This addition, however, would only be necessary when the machine is used as a program development tool and if the ability to "trap" individual errors were provided.

TO THE END USER

The end user will benefit from a few features also. The appearance of the computer is very important in business applications and greatly affects the time required to learn its operation. A simple, non-cluttered keyboard area is much easier for the naive user to learn and operate. There are too many unused buttons and switches on the front of many computers which needlessly complicate actual use of the machine.

An applications program auto-load feature is valuable in a turnkey system. This enables the user to simply "load-and-go" with his application package and greatly reduces the visibility of the operating system.

File protection, as discussed earlier, is of prime importance to the businessman in protecting personnel and customer files. Accidental destruction of a vital file can set a business back weeks while the data is re-entered. Privileged information must be protected from unauthorized access.

A very nice feature of systems T and W is the complete transparency of operating system. This feature would be complete with the ability to "trap" fatal and non-fatal errors during execution of business packages. The naive user does not wish to be bothered by diagnostics, and their display is usually worthless in helping the user to correct mistakes.

For the end-user, use of diskettes offers rapid random access and a low-cost storage medium. Indexed and random access decreases the time required to retrieve a particular record. For fully interactive dataprocessing, larger disks are essential to store on-line data.

Manuals should be modularized and written in a "top-down" format. The system reference manual should be condensed, making it a true reference. Allowance should be made for rapid access of vital data such as diagnostic codes. A dictionary of differences between Dartmouth BASIC and the "business" BASIC of a specific system is needed.

ACKNOWLEDGEMENTS

This study was made possible by the energetic team of evaluators: W. Leggett; J. Joiner; K. Malik; S. Simon; J. Wilson; T. Elton; and D. Kim. Their hard work and active participation is a source of inspiration.