WestVirginiaUniversity
THE RESEARCH REPOSITORY @ WVU

Graduate Theses, Dissertations, and Problem Reports

1974

A Simulator for the IBM 3705 Communications Controller

J. William Strider
West Virginia University

Follow this and additional works at: https://researchrepository.wvu.edu/etd

 Part of the Computer Sciences Commons

Recommended Citation

Strider, J. William, "A Simulator for the IBM 3705 Communications Controller" (1974). *Graduate Theses, Dissertations, and Problem Reports*. 7805.
https://researchrepository.wvu.edu/etd/7805

This Thesis is protected by copyright and/or related rights. It has been brought to you by the The Research Repository @ WVU with permission from the rights-holder(s). You are free to use this Thesis in any way that is permitted by the copyright and related rights legislation that applies to your use. For other uses you must obtain permission from the rights-holder(s) directly, unless additional rights are indicated by a Creative Commons license in the record and/ or on the work itself. This Thesis has been accepted for inclusion in WVU Graduate Theses, Dissertations, and Problem Reports collection by an authorized administrator of The Research Repository @ WVU. For more information, please contact researchrepository@mail.wvu.edu.

A Simulator for the IBM 3705 Communications Controller

by

J. William Strider

Submitted to

The Department of Statistics and Computer Science

of

West Virginia University

In Partial Fulfillment of the Requirements for the Degree of

Master of Science in Computer Science

December 12, 1974

ABSTRACT

This paper describes a computer program which was developed to simulate the IBM 3705 Communications Controller, using the IBM System/360 and System/370 computers. The architecture of the 3705 is discussed in some detail, and the structure of the simulator is described. Future enhancements to the present program are suggested, along with possible applications.

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my appreciation to everyone who helped to make this work possible. I am particularly indebted to my friend and advisor, Dr. Malcolm G. Lane, for both the initial idea for the project, and for his continuing support and encouragement. I am also grateful to Ms. Rita Saltz for her efforts in proofreading the manuscript, and to the management of the West Virginia University Computer Center for the marvelous facilities which were made available to me almost without limit. And finally, I want to thank my wife, Jean, for her infinite patience during the course of the project.

Table of Contents

INTRODUCTION

For a number of years, the concept of asynchronous processing has been an important one in the design of digital computer systems. One of the most significant trends in this area has been the development of intelligent control units in an attempt to relieve main processors from some of the drudgery involved in supervising input/output operations. One area which lends itself readily to such treatment is data communication, and a considerable amount of effort has been expended in developing intelligent communications controllers. This is particularly desirable because of the relatively large size of communication networks, the wide variety of terminal codes and line disciplines in general use, and the difficulties involved in transmitting data over long distances.

These devices are actually small computers whose sole function is to supervise the transmission of data using

various types of teleprocessing equipment. They are particularly suited to such menial tasks as code translation and elementary editing, but they are also being used to accomplish automatic terminal and speed recognition, and even simple forms of message routing.

The advent of the communications controller has brought with it a great potential for the computer user to tailor a network to his specific needs. Many of the options which formerly had to be chosen when the control unit was wired can now be decided dynamically by software. Unfortunately, the very nature of the communications controller makes it somewhat inconvenient as a tool for software development. Its capacity for multiprogramming is somewhat limited, and the necessity for a stable and continually available communications network seriously curtails the amount of time which can be dedicated to software debugging. Also, the normal lack of peripheral devices, particularly printers, makes debugging a cumbersome and time-consuming chore. The only alternative so far has been to have a second communications controller available for developmental work, which may not be attractive financially. Thus users have been effectively discouraged from learning more about communications controllers, and developing their own

software.

For these reasons, it was felt that a simulator for a communications controller could be a valuable tool for both the data communications user and the educator. The controller chosen for simulation was the IBM 3705, both because of its widespread use, and because one was available for experimentation while the simulator was being developed. The simulator was written in 360 assembler language, and was implemented on the IBM 360/75 at the West Virginia University Computer Center. It simulates the 3705 at the machine instruction level, and produces optional trace output for debugging purposes.

Programs written in 3705 assembler language can be directly loaded and interpretively executed by the simulator. At this time, it does not fully simulate all aspects of the communications network and host computer interface; hence it could not be used to extensively test a 3705 control program. It is, however, quite adequate for initial program development and testing, and for training students in the basic concepts of communications controllers.

The 3705 communications controller consists of four major components. The central control unit (CCU) contains most of the arithmetic and logic circuitry necessary for the operation of the 3705; the core memory serves its usual function of providing a storage area for both machine instructions and data; the channel adapter controls the interface between the 3705 and the host computer; and the communication scanner serves as the interface between the 3705 and the communications network.

Programs in the 3705 can execute at any of five priority levels which are controlled by hardware. Levels one through four are interrupt driven; that is, they are entered only on the occurrence of specific hardware interrupt conditions. Level one, the highest priority level, is used mainly for handling error conditions. It is entered when either a hardware failure or programming error occurs. Level two deals with the communication network, and is entered whenever a communication line must be serviced by the software. Level three is used to handle processing of a less critical nature, including communication with the host

computer, timer maintenance, and operator intevention. Level four is the lowest level of the supervisor, and is entered only upon request of one of the other program levels.

Level five, the lowest priority level, is unique in several respects. It is not interrupt driven, and is executed only when there are no outstanding requests for any of the other program levels. It is intended for non-critical background processing, and hence is not allowed to execute the privileged instructions available to the other four levels.

When an interrupt occurs for a particular program level, the action taken depends upon the relative priority of the currently active level. If the active level is of the same or higher priority, the interrupt request is stacked by the hardware until processing returns to a lower level. At that point, a latch is set to indicate the cause of the interrupt, and the appropriate program level is entered. Each program level begins execution at a predefined storage address: level one starts at location 10 (hexadecimal), level two at location 80, level three at location 100, and level four at location 180. From this initial starting point, each interrupt handling routine may branch to any

other portion of the machine storage without restriction. Processing continues at a particular level, uninterrupted, until either a higher priority interrupt occurs, or the program exits from that level.

The 3705 core storage is organized in bytes of eight bits each; these may be grouped into halfwords (two bytes) and fullwords (four bytes). Storage addressing is by byte; the first byte of memory is designated byte 0, and successive bytes are numbered sequentially. Thus the last byte in a 16K machine, for example, would have the address 16,383.

Basic 3705 addressing allows for an address of 16 bits (one halfword). This will accommodate a storage size of up to 64K bytes. For larger machines, an additional two-bit byte, known as byte X, is appended to the address, thus allowing addressability up to the maximum storage size of 240K bytes. Machines equipped with this extra address byte are said to have the extended addressing feature.

The central control unit contains 32 general registers, each of which is large enough to accommodate a storage address; thus a basic 3705 has 16-bit registers, while a 3705 with extended addressing has 18-bit registers. The registers

are divided into four groups of eight, and only one group is directly accessible at any given time. The first group (group 0) is available to program levels one and two; group 1 is used by level three; group 2 is accessible to level four; and group 3 is reserved for level five. Thus it is possible to change from one program level to another without saving and restoring the contents of the general registers.

At each program level, the first general register (register 0) of the associated group serves as the instruction address register (IAR). It always contains the address of the next machine instruction to be executed. It is incremented sequentially as processing proceeds, unless it is modified by the executing program. When an interrupt occurs, the IAR of the appropriate level is loaded with the starting address for that level.

The 3705 contains a number of external registers, of which the general registers are a subset. The external registers, rather than being directly available to the processing program, are accessed via the privileged machine instructions, input and output. External registers are found in the communication scanner and channel adapter, as well as in the CCU. They are the primary means of communication

between hardware and software.

Each program level has a pair of condition latches known as the C and Z latches. These are used to record the results of certain arithmetic, comparative, and logical operations. The latches may then be tested by the program to decide among various courses of action.

The 3705 recognizes 51 machine instructions. Many of these are similar to the instructions found on other small computers, although they are noticeably character oriented. There are versions of many of the operations which access only one byte of a register. Multiply and divide operations are totally absent, while shifting operations are severely limited--the only shift available is one bit to the right. All arithmetic and logical operations operate on registers only; the only instructions which directly reference storage are of the load and store variety. There are a number of branching instructions available, although branching may be accomplished by any instruction which modifies register 0 (the IAR). Special purpose instructions include input and output, which are used to access the external registers, and exit, which effects a transfer from one program level to another.

It is possible, via the use of input and output instructions, to mask the various program levels. When any of the four higher priority levels is masked, interrupts for that level will not occur, but will be stacked until that level is unmasked. Masking has no effect upon a previous entry to a particular level; execution will continue until the program exits from that level, but subsequent entries will be inhibited. In this manner, a program operating at a lower level can ensure its uninterrupted execution during particularly critical processing. Masking operates differently with respect to level five, since level five is not interrupt driven. If level five is masked, execution at that level is suspended immediately; if no interrupt requests are outstanding, the 3705 enters the wait state.

The 3705 storage protection feature allows the software to limit access to main storage by the channel adapter and by program level five. By means of output instructions, the program can set a protection key to be associated with each channel adapter and with level five, as well as a storage key to be associated with each block of storage. Program levels one through four are permanently assigned the protection key of zero. When an access to storage is attempted, the

applicable protection key and storage key are checked. Access is permitted if the keys match, if the protection key is zero, or if the storage key is seven, which signifies unprotected storage. In the case of instruction execution, however, the test is more stringent: access is allowed only if the keys match. Any violation of storage protection causes a level one interrupt, with the appropriate latch set to indicate a protection check.

The CCU contains an interval timer which runs continuously, and causes a level three interrupt every 100 milliseconds. It also contains a set of hardware registers which perform the cyclic redundancy check (CRC) accumulation function required for binary synchronous communication. Special circuits within the CCU, including a read-only storage array, are provided to facilitate the initial loading of a program from the host computer. Two display registers are provided to allow the program to communicate with the control console, and a lagging address register (LAR), which contains the address of the last instruction executed, is available to help locate programming errors.

Two types of communication scanner are available for the 3705. The type 1 scanner is bit-oriented; it causes an

interrupt, and requires program intervention, as each bit is transmitted and received. The type 2 scanner is character-oriented; it assembles bits into characters when data is received from a terminal, the number of bits per character having been specified by the software. Hence, with the type 2 scanner, interrupts occur only as each character is sent or received. The scanner is connected to the 3705 via an attachment base of the appropriate type; communication lines are attached by means of line interface bases (LIBs) and line sets. We shall restrict the following discussion to the type 2 scanner, since the simulator assumes a 3705 equipped with such a scanner.

The type 2 communication scanner contains a local storage array of 96 interface control words (ICWs); each word consists of 46 information bits and 2 parity bits. Each ICW contains control information for a single communication line interface; all communication with a line takes place via the interface control word.

The type 2 scanner contains several external registers which are used in communicating with the network. The attachment base address register (ABAR) is used to hold the interface address for the line currently being examined. The

display register is periodically updated with control information concerning a requested interface. Several other registers are used to hold parameters for the scanner, and to facilitate examination and update of the various ICWs.

The scanner examines interface addresses sequentially, and notifies the software whenever character service is required for a particular line. There is a scan counter in the type 2 attachment base which generates the addresses for the scanner; this is done rapidly enough to ensure operation at speeds up to 4800 bps. For higher speed communication, the scanner can be directed, by the software, to establish an upper scan limit. This will cause certain interfaces to be ignored, and hence those interfaces which are used can be scanned more often, resulting in higher line speeds. The software can also cause certain addresses to be substituted for other addresses, further increasing the maximum line speed.

The type 2 scanner recognizes a number of input and output instructions. These are used to access and modify the interface control words and external registers, and to control various operations of the scanner, including the upper scan limit and address substitution features. Each

scanner can have from one to four business machine clocks, which are used to provide character timing pulses for low speed lines. Special circuitry is also present to perform diagnostic functions, including modem testing and simulated line activity.

As with the communication scanner, two types of channel adapter are available for the 3705; the type 1 adapter attaches to a byte multiplexer channel, and transfers data in bursts of up to four bytes. Software action is required before and after each burst. The type 2 adapter attaches to a byte multiplexer, a block multiplexer, or a selector channel, and transfers data in large bursts (up to 1023 bytes). Since the simulator assumes a 3705 equipped with a type 1 channel adapter, we shall restrict the following discussion to this adapter.

The type 1 adapter can operate in either of two modes: native subchannel (NSC) mode allows up to 352 communication lines to be supported using a single subchannel address in the host computer. Emulation subchannel (ESC) mode allows the software to emulate the IBM 2701, 2702, and 2703 transmission control units, using a separate host subchannel address for each line.

The type 1 adapter can operate in any of three states: initial selection state is entered when a particular line is first addressed by the host computer; data transfer state is entered when data is to be transferred to or from the host; and final status state is entered when status information is to be transferred to the host.

The adapter contains a number of external registers which are used to record status and control information, and to communicate with the host computer. The adapter can be wired to recognize and respond to a number of subchannel addresses; this would apply only to operation in ESC mode. The type 1 adapter recognizes only three channel commands directly: test I/O, write IPL, and no-operation. All other commands are passed to the 3705 software, which must decode the command and take appropriate action. Input and output instructions are available to allow the control program to communicate with the channel adapter.

The simulator is divided into four load modules which are executed sequentially. S3705BGN is the first module, and serves to invoke the other modules in the proper order. S3705INT is the initialization module, which parses the user-specified parameters, and sets up data areas. S3705LDR is the loader, which brings a 3705 program into main storage for execution. S3705GO, the main body of the simulator, interpretively executes the 3705 program, optionally producing trace output, along with error and information messages.

The contents of the 3705 core storage are kept in a contiguous area in subpool 1 of the 360 storage. The size of this area can be specified by the user, and matches the simulated 3705 storage byte for byte. Most of the global information for the simulator is kept in an area of S3705BGN (symbolic name DATA1). This area contains a number of essential flags and addresses, as well as all of the 3705 registers, latches, and control information.

A number of options can be specified by the user prior to execution. These include the size of the 3705 storage,

the name of the load module to be executed, and the types of trace to be performed. The types of trace currently supported are interrupt trace, which causes a message to appear each time an interrupt occurs; level trace, which causes a message to appear each time a new program level is entered; execution trace, which prints a brief message as each instruction is executed; and extended trace, which implies execution trace, and includes current level and latch settings, and the results of each operation.

The trace output contains the following information for each instruction (see Appendix B): the elapsed time in machine cycles, the value of the IAR, the hexadecimal machine instruction, the instruction mnemonic code, the instruction operands, the currently active level, the two condition latches at that level, and the results of the operation. In certain cases, the result field contains descriptive comments enclosed in asterisks.

S3705BGN receives control from the operating system, and sets up save areas for the entire simulator. It then opens the data control block for printed output, initializes the time and date in the page headings, and passes control to S3705INT. Control is subsequently returned from S3705LDR,

and the return code is tested. If it is satisfactory, control passes to S3705GO; if not, the simulation phase is bypassed. When the simulation is complete, S3705BGN frees the storage area in subpool 1, and returns to the operating system.

S3705BGN contains the subroutine PSUB, which is used by all simulator modules. PSUB receives as input a message of variable length, which it moves to a buffer area and prints. It keeps track of the number of lines printed, and inserts a heading at the top of every page. The length of each message is saved and used to blank a variable portion of the output buffer on the subsequent call.

S3705INT, after receiving control from S3705BGN, processes the parameter field which the user specified on the EXEC statement which invoked the simulator. (Refer to Appendix A for a discussion of the procedures necessary to invoke the simulator). Valid parameter keywords are identified by lookup in a table which is sorted in descending order of keyword length. The current position in the parameter string is compared with each successive keyword until either a match is found, or the end of the table is reached. If the keyword matches an entry in the table, a

corresponding table entry is used to branch to the appropriate processing routine.

After the parameter list has been fully parsed, assuming no errors have been detected, S3705INT examines the core storage size specified by the user, and acquires the necessary area in subpool 1. The addresses of the beginning and end of the storage area are saved for use in later phases of the simulation. Finally, S3705INT passes control to S3705LDR via an XCTL macro instruction. This removes S3705INT from the chain of control, so that S3705LDR can return control directly to S3705BGN.

S3705LDR receives control from S3705INT, and opens the data control block for the input data set. Since the input data has the partitioned format, a BLDL macro instruction is issued to obtained the directory entry for the requested load module. The directory entry contains the entry point address for the load module, which is placed in general register 0 of group 0. When the program subsequently is given control, execution will thus start at the proper location.

After issuing a FIND macro instruction to position the data set to the requested module, S3705LDR starts reading the

member. Since the 3705 is a dedicated machine, no relocation is necessary, and all records in the load module are skipped except control and text records. When a control record is encountered, the length from the channel command word is placed in the data control block, and the data address is added to the address of the storage area in subpool 1 to produce the address of the start of the following text. The text record which follows the control record is then read using the previously obtained address and length, and the process is repeated until end of file is reached.

Several error conditions can be detected during the loading process. The requested load module may not exist, it may contain excessively long records, or it may be unreadable because of I/O errors. In any of these cases, the simulation is aborted at the point where the error is detected. In the event that the requested load module is larger than the 3705 storage, a warning message is issued, and execution continues, although the module is truncated at the end of storage.

If all is well at the end of the loading phase, S3705GO receives control from S3705BGN. After a few preliminaries, which include disabling program level five and adjusting mask

fields if extended addressing is present, the actual simulation begins. Register 0 of the active level (initially level one) is examined, and the instruction at that address is decoded. Since the operation code in the 3705 is not contiguous, a rather involved procedure is necessary to identify the particular operation, given the machine language representation. Several pages of code are necessary to produce a numeric operation code which can be used to index a table for subsequent processing. Once the operation is identified, a routine is entered to separate the various operands and place them in working storage locations. Finally, the numeric operation code is used as an index to a table of execution routines, and the operation is executed interpretively.

Because of the great diversity in function of the input and output instructions, a second table-lookup procedure is necessary. The hexadecimal input or output code, which can range from zero to 7F, is used to index a table of 1-byte codes. The code thus obtained is then used to index a table of routine addresses, and the appropriate routine is entered to perform the required function. The extra table of 1-byte codes is necessary to save space, since many of the input and output codes are unused, and those which are used are often

similar or identical in function. Thus only a few routines are needed to interpret the 256 possible input and output instructions.

If execution trace or extended trace has been specified, the trace buffer is built during the interpretation process. The trace message is printed just prior to fetching the next instruction; hence any information or error message printed as a result of instruction execution will appear in the listing before the trace line for the instruction.

Program levels are controlled by means of two ordered lists and four counters, along with a set of flags to indicate which levels are masked. One list contains the currently active levels, while the other contains the pending levels. Each list is padded on the right with the number five to a total of five elements. The four counters contain the number of times each of the four interrupt levels are pending. If no interrupts are pending, all counters are zero, and the lists contain all fives. When an interrupt is to be scheduled, the counter corresponding to the level of the interrupt is incremented by one, and the level number is inserted in the pending list in such a way that the list is

always kept in ascending order. As each instruction is fetched, the active and pending lists are tested to see if there is a pending interrupt with a higher priority than the currently active level. If so, and if the pending level is not masked, then the pending level becomes the currently active level (the first entry in the active list). When an exit instruction is encountered, the active level is removed from the active list, and the corresponding counter is decremented. If the counter is zero, then the level number is removed from the pending list. Otherwise, another interrupt for this level is still pending, and the list is not altered.

Storage protection is implemented by means of two arrays of bytes. One, of length eight, represents the protection keys, while the second, of length 128, represents the storage keys. As each instruction is executed, any storage references are checked to see that protection is not violated. The effective storage address is used to compute an index into the array of storage keys; the number of the active level is used to index the array of protection keys. The keys are then compared, and if a protection violation is indicated, a level one interrupt is scheduled, with the appropriate latch set to indicate a protection check.

Error conditions are handled in accordance with the actual operation of the 3705, as much as possible. Program checks cause the appropriate level one interrupt, unless level one is already active. In that case, a CCU check is indicated; the standard 3705 action is to initiate a new IPL, but since this is not feasible, the simulation is halted at the point where the CCU check occurs.

FUTURE IMPROVEMENTS

There are a number of deficiencies in the present version of the simulator. A few hardware features are currently unsupported, including cyclic redundancy check accumulation, the diagnostic test mode, and several of the control functions of both the channel adapter and communication scanner. This does not seriously impair the usefulness of the simulator, since programs which use any of the unsupported features receive an informational message, and are allowed to continue execution. The modular structure of S3705GO makes it an easy task to add these features when the need arises.

A more serious limitation at this point is the lack of a facility for generating a large number of interrupts, such as might be found in a real communications network. In order to give any software a real workout, the simulator will need to provide a high volume of level two and three interrupts, together with the associated characters which might be received from both the host computer and the various terminals of the network.

The problem of generating interrupts from the communications network is relatively straightforward. Each type of terminal can have associated with it a table of valid characters with a distribution calculated to insert an ending sequence every n characters on the average, where n is the mean length of a block of data. The interval between character interrupts can be represented relative to the elapsed time in machine cycles. For synchronous terminals, the interval will be a constant depending on the line speed; for start-stop terminals, the intervals will be exponentially distributed. A suitable pair of pseudo-random number generators, then, can be used to effectively simulate the input from a communications network.

The channel adapter, unfortunately, presents a different and more complex problem. It is not enough to merely present the 3705 with a random series of characters as if they had come from the adapter, since the 3705 software must recognize and interpret the great variety of channel commands which can be generated by the host processor. The data from the simulated channel adapter must make sense; it must conform to the rigid structure set forth in the 360 channel architecture.

The manual generation of a large amount of such data would be a mammoth task, to say the least. A more feasible approach might be to design a software monitor which could intercept the output of a number of actual teleprccessing systems, recording the data on magnetic tape. This tape could then be used as input to the simulation, with the added benefit that it could easily be tailored to closely reflect the actual communication environment.

This concept can be extended to include the network as well; data input to the real teleprocessing system could be captured similarly, so that the simulation of both the channel adapter and communication scanner would closely approximate the system for which the 3705 would be used.

Some judicious planning might then produce a benchmark set of input data for the simulator, thus providing an excellent means for comparing various 3705 control programs.

A number of conveniences will probably be added to the simulator in the near future. These will include the facility for the user to dump selected areas of the 3705 storage and registers, and a means for dynamically enabling and disabling the trace facility. Eventually, the user will be given the ability to specify switch settings on the 3705 control panel, in order to utilize and test the operator communication facilities.

POSSIBLE APPLICATIONS

As mentioned previously, the 3705 simulator should be of considerable value as an aid to software development. An immediate application would be the testing of new releases of IBM-supplied control programs. Although complete testing of the interrupt handlers would not be possible with the current version, the 3705 program could complete its initialization phase, and any gross errors in control program generation would probably be detected.

As users gain familiarity with the 3705, they might well consider writing their own software, in the form of either modifications to IBM-supplied code, or the design of entirely original control programs. Because of its excellent trace facility, the simulator would prove an invaluable aid to efforts of this type.

The simulator can be equally useful as an educational tool. An introductory course in data communications might well spend some time discussing the 3705, and the simulator could provide students with the opportunity to learn first-hand the problems involved with such processes as editing and code translation. In more advanced courses, the simulator could serve as the base for any number of major projects in communications. It is currently being used by a graduate student who is designing modifications to the IBM-supplied emulator program; without the simulator, such a project would almost certainly be impractical.

Appendix A

User Instructions

The 3705 simulator, as implemented at the West Virginia University Computer Center, can be invoked by means of the cataloged procedure S3705CLG. The following JCL is typical:

```
//S1 EXEC S3705CLG
//SYSIN DD *
   3705 Assembler Source Deck
        .
        .
        .
//
```

S3705CLG is a three-step procedure which executes the 3705 assembler, the F-88 linkage editor, and the 3705 simulator. The following parameters are passed to the executed programs:

```
Assembler -- NODECK,LOAD
Linkage editor -- XREF,LET,LIST,DC
Simulator -- E,X,I,L,SIZE=48K,EP=PGM3705
```

(Note: If these parameters are overridden, the user should specify DC to the linkage editor, and EP=PGM3705 to the simulator.)

The following is a complete description of the parameters which may be specified to the simulator (via PARM.SIM on the EXEC statement):

E or ETRACE -- specifies the execution trace facility.

X or XTRACE -- specifies the extended trace facility.

I or ITRACE -- specifies the interrupt trace facility.

L or LTRACE -- specifies the level trace facility.

SIZE=nnnK -- specifies the size of the 3705 storage in multiples of 1024 bytes. It should normally have a value of 16+32*N, where N is a non-negative integer less than or equal to seven.

EP=name -- specifies the name of the load module to be executed.

Appendix B

Sample Output

This appendix contains a sample program written in 3705 assembler language, together with the associated simulator output. The program was designed to test the more intricate parts of the simulator, including interrupt masking, storage protection, and Interface Control Word (ICW) access. It switches levels several times, and accesses a number of external registers, including the ABAR, the display registers, and general registers belonging to other than the active level. The interrupts which occur include program controlled interrupts (PCIs), supervisor calls (SVCs), and a level one protection violation.

```
  LOC  OBJ CODE  R1N1M R2N2 ADDR   STMT   SOURCE STATEMENT
000000                              1 SIMTEST  CSECT
000000 00000000                     2          DC    A(0)              BLOW UP IF WE COME HERE



000010                              4          ORG SIMTEST+X'10'       TO LEVEL 1 START ADDRESS
000010                              5 LEVEL1   DS    0H                LEVEL 1 INTERRUPT ROUTINE
000010 0082       0     0           6          ST    R0,0(R0)          SAVE REGISTERS ZERO
000012 0186       1     0           7          ST    R1,4(R0)            AND ONE
000014 71EC       1    7E           8          IN    R1,X'7E'          CAUSE OF INTERRUPT
000016 7114       1    71           9          OUT   R1,X'71'          DISPLAY
000018 B920 0001  1         00001   10          LA    R1,1              INTERRUPT LEVEL
00001C 7124       1    72          11          OUT   R1,X'72'          DISPLAY
00001E 7004       0    70          12          OUT   R0,X'70'          HARDSTOP



000080                             14          ORG   SIMTEST+X'80'     TO LEVEL 2 START ADDR
000080                             15 LEVEL2   DS    0H                LEVEL 2 INTERRUPT ROUTINE
000080 717C       1    77          16          IN    R1,X'77'          INTERRUPT CAUSE
000082 7114       1    71          17          OUT   R1,X'71'          DISPLAY IT
000084 B920 0002  1         00002   18          LA    R1,2              INTERRUPT LEVEL
000088 7124       1    72          19          OUT   R1,X'72'          DISPLAY
00008A 7004       0    70          20          OUT   R0,X'70'          HARDSTOP



000100                             22          ORG   SIMTEST+X'100'    LEVEL 3 START ADDRESS
000100                             23 LEVEL3   DS    0H                LEVEL 3 INTERRUPT ROUTINE
000100 71FC       1    7F          24          IN    R1,X'7F'          INTERRUPT CAUSE
000102 F90A       1(1,6)    0010E   25          BB    R1(1,6),PCIL3     BRANCH IF PCI
000104 7114       1    71          26          OUT   R1,X'71'          DISPLAY
000106 B920 0003  1         00003   27          LA    R1,3              INTERRUPT LEVEL
00010A 7124       1    72          28          OUT   R1,X'72'          DISPLAY
00010C 7004       0    70          29          OUT   R0,X'70'          HARDSTOP

00010E B920 0020  1         00020   31 PCIL3    LA    R1,X'20'          BIT TO RESET
000112 7174       1    77          32          OUT   R1,X'77'          RESET THE INTERRUPT
000114 71FC       1    7F          33          IN    R1,X'7F'          GET IT BACK
000116 7114       1    71          34          OUT   R1,X'71'          DISPLAY FOR VERIFICATION
000118 B920 0003  1         00003   35          LA    R1,3              INTERRUPT LEVEL
00011C 7124       1    72          36          OUT   R1,X'72'          DISPLAY
00011E 110C       1    10          37          IN    R1,X'10'          GET OLD L4 ADDRESS
000120 9102       1(1)              38          ARI   R1(1),2           AND ADD TWO
```

```
 LOC  OBJ CODE  R1N1M R2N2 ADDR   STMT   SOURCE STATEMENT                    18MAR73  12/09/74
000122 1104       1    10          39          OUT   R1,X'10'          AND PUT IT BACK
000124 B840                         40          EXIT  ,                 AND RETURN TO NEXT LEVEL


000180                              42          ORG   SIMTEST+X'180'    TO LEVEL 4 START ADDR
000180                              43 LEVEL4   DS    0H                LEVEL 4 INTERRUPT ROUTINE
000180 71FC       1    7F           44          IN    R1,X'7F'          GET INTERRUPT CAUSE
000182 F9B4       1(1,7)    001B8   45          BBE   R1(15),TERM       TERMINATE IF SVC L4
000184 710C       1    70           46          IN    R1,X'70'          GET STORAGE SIZE
000186 33A8       3     3           47          SR    R3,R3             ZERO R3
000188 0308       3(1)  1(0)        48          LCR   R3(1),R1(0)       TO LOW ORDER BYTE
00018A 7314       3    71           49          OUT   R3,X'71'          STORAGE SIZE IN K
                                    50 *
                                    51 *        TEST INTERFACE CONTROL ACCESS
                                    52 *
00018C BA20 0840  2               00840 53          LA    R2,X'840'         LOAD FIRST INTERFACE ADDRESS
000190 BC20 0002  4               00002 54          LA    R4,2              INCREMENT FOR INTERFACE ADDR
000194 8204       3(0)                  55          LRI   R3(0),4           COUNT FOR BCT LOOP
000196 4204       2    40           56 ICWLOOP  OUT   R2,X'40'          LOAD ABAR
000198 B920 0028  1               00028 57          LA    R1,X'28'          SET DATA TERMINAL READY, EXT CLOCK
00019C 4164       1    46           58          OUT   R1,X'46'          SET SDF FOR SUBSEQUENT MODE SET
00019E B920 00C1  1               000C1 59          LA    R1,X'C1'          BISYNCH EBCDIC, SET MODE
0001A2 4154       1    45           60          OUT   R1,X'45'          SET LCD AND PCF
0001A4 4290       2     4           61          AHR   R2,R4             GO TO NEXT INTERFACE
0001A6 BA93       3(0)        00196 62          BCT   R3(0),ICWLOOP     AND LOOP BACK

0001A8 B920 1000  1               01000 64          LA    R1,LEVEL5         -> LEVEL 5 CODE
0001AC 1184       1    18           65          OUT   R1,X'18'          SET UP THE IAR FOR L5
0001AE BA20 0010  2               00010 66          LA    R2,X'10'          MASK FOR L3
0001B2 72E4       2    7E           67          OUT   R2,X'7E'          SET IT
0001B4 70C4       0    7C           68          OUT   R0,X'7C'          SET PCI L3
0001B6 B840                         69          EXIT  ,                 AND ON TO LEVEL 5...

0001B8 72F4       2    7F           71 TERM     OUT   R2,X'7F'          ENABLE LEVEL 3
0001BA 2004       0    20           72          OUT   R0,X'20'          ABEND SIMULATOR
0001BC B920 01BC  1               001BC 73          LA    R1,*              POINT TO OURSELF
0001C0 1184       1    18           74          OUT   R1,X'18'          SEND LEVEL 5 BACK HERE
0001C2 B840                         75          EXIT  ,                 AND GO TO LEVEL 5
```

12/09/74

SYMBOL	LEN	VALUE	DEFN	REFERENCES														
ENDLOOP	00002	001024	00113	0109														
ICWLOOP	00002	000196	00056	0062														
LEVEL1	00002	000010	00005															
LEVEL2	00002	000080	00015															
LEVEL3	00002	000100	00023															
LEVEL4	00002	000180	00043															
LEVEL5	00002	001000	00098	0064														
LOOP	00002	001012	00104	0114														
PCIL3	00004	00010E	00031	0025														
R0	00001	000000	00124	0006	0006	0007	0012	0020	0029	0068	0072	0079						
R1	00001	000001	00125	0007	0008	0009	0010	0011	0016	0017	0018	0019	0024	0025	0026	0027	0028	0031
				0032	0033	0034	0035	0036	0037	0038	0039	0044	0045	0046	0048	0057	0058	0059
				0060	0064	0065	0073	0074	0080	0081	0085	0086	0087	0088	0089	0090	0091	0092
				0093	0094													
R2	00001	000002	00126	0053	0056	0061	0066	0067	0071	0099	0104	0105						
R3	00001	000003	00127	0047	0047	0048	0049	0055	0062	0102	0114							
R4	00001	000004	00128	0054	0061	0100	0110											
R5	00001	000005	00129	0103	0103	0104	0108	0110	0111	0112								
R6	00001	000006	00130	0101	0112													
R7	00001	000007	00131	0105	0106	0107	0107	0108	0111									
SIMTEST	00001	000000	00001	0004	0014	0022	0042	0077	0097	0117								
SOURCE	00004	001800	00118	0099														
STARTXEQ	00002	000800	00079	0078	0132													
TARGET1	00004	001804	00119	0100														
TARGET2	00004	001808	00120	0101														
TERM	00002	0001B8	00071	0045														

NO STATEMENTS FLAGGED IN THIS ASSEMBLY
STATISTICS SOURCE RECORDS (SYSIN) = 132
OPTIONS IN EFFECT LIST, NODECK, LOAD, NORENT, XREF, LINECNT = 50
179 PRINTED LINES

```
 LOC   OBJ CODE  R1N1M R2N2 ADDR   STMT   SOURCE STATEMENT
001000                              97          ORG   SIMTEST+4096      TO LEVEL 5'S STORAGE BLOCK
001000                              98 LEVEL5   DS    0H
001000 BA20 1800  2          01800  99          LA    R2,SOURCE         -> DATA
001004 BC20 1804  4          01804 100          LA    R4,TARGET1        -> FIRST RECEIVING FIELD
001008 BE20 1808  6          01808 101          LA    R6,TARGET2        -> SECOND TARGET
00100C BB20 0004  3          00004 102          LA    R3,4              COUNT FOR BCT LOOP
001010 55A8       5     5          103          SR    R5,R5             ZERO
001012 2510       5(1)  2          104 LOOP     ICT   R5(1),R2          PICK UP CHARACTER
001014 2F00       7(1)  2          105          IC    R7(1),0(R2)       GET NEXT CHARACTER
001016 8600       7(0)             106          LRI   R7(0),0           ZERO HIGH ORDER BYTE
001018 7778       7(1)  7(1)       107          LCOR  R7(1),R7(1)       SHIFT RIGHT
00101A 75C8       5     7          108          XR    R5,R7             EXCLUSIVE OR
00101C 8806                 01024  109          BZL   ENDLOOP          IF ZERO, SKIP IT
00101E 4530       5(1)  4          110          STCT  R5(1),R4          STORE FIRST RESULT
001020 57A0       7     5          111          SHR   R7,R5             SUBTRACT
001022 6530       5(1)  6          112          STCT  R5(1),R6          STORE SECOND RESULT
001024                             113 ENDLOOP  DS    0H
001024 BB95       3(1)        01012 114          BCT   R3(1),LOOP        AND LOOP BACK FOR NEXT CHARACTER
001026 B840                        115          EXIT  ,                 NOW INVOKE LEVEL 4

001800                             117          ORG   SIMTEST+6144      TO DATA BLOCK
001800 015BC6F7                    118 SOURCE   DC    X'015BC6F7'
001804 00000000                    119 TARGET1  DC    X'00000000'
001808 FFFFFFFF                    120 TARGET2  DC    X'FFFFFFFF'
                                   121 *
                                   122 * REGISTER EQUATES
                                   123 *
000000                             124 R0       EQU   0
000001                             125 R1       EQU   1
000002                             126 R2       EQU   2
000003                             127 R3       EQU   3
000004                             128 R4       EQU   4
000005                             129 R5       EQU   5
000006                             130 R6       EQU   6
000007                             131 R7       EQU   7
000800                             132          END   STARTXEQ          SPECIFY ENTRY POINT
```

```
  LOC   OBJ CODE  R1N1M R2N2 ADDR   STMT   SOURCE STATEMENT                                  18MAR73  12/09/74

000800                              77          ORG   SIMTEST+2048      TO SECOND 2K BLOCK
                                    78          ENTRY STARTXEQ
000800 70D4       0     7D          79 STARTXEQ OUT   R0,X'7D'          SET PCI L4
000802 813C       1(1)              80          LRI   R1(1),X'3C'       MASK BITS FOR LEVELS 2 THROUGH 5
000804 71F4       1     7F          81          OUT   R1,X'7F'          RESET MASK
                                    82 *
                                    83 * STORAGE PROTECTION
                                    84 *
000806 B920 0218  1           00218 85          LA    R1,X'0218'        2048 <- KEY 0
00080A 7134       1     73          86          OUT   R1,X'73'          SET KEY
00080C 8000       1(0)              87          LRI   R1(0),0           0 <- KEY 0
00080E 7134       1     73          88          OUT   R1,X'73'          SET KEY
000810 B920 0419  1           00419 89          LA    R1,X'0419'        4096 <- KEY 1
000814 7134       1     73          90          OUT   R1,X'73'          SET KEY
000816 B920 061F  1           0061F 91          LA    R1,X'061F'        6144 <- KEY 7 (UNPROTECTED)
00081A 7134       1     73          92          OUT   R1,X'73'          SET KEY
00081C B920 0009  1           00009 93          LA    R1,X'0009'        LEVEL 5 <- KEY 1
000820 7134       1     73          94          OUT   R1,X'73'          SET KEY
000822 B840                         95          EXIT  ,                 NOW GO TO LEVEL 4 CODE (PCI)
```

```
   TIME   LOCN  OBJ CODE  OP   OPERAND        LCZ       RESULT

 PARAMETERS SPECIFIED -- I,L,X,E,SIZE=48K,EP=PGM3705

 3705 LOAD COMPLETE


*INTERRUPT -- PCI L4
       0  000800 70D4       OUT  0,X'7D'        1
       1  000802 813C       LRI  1(1),X'3C'     1C    REG 1 = 0000003C
       2  000804 71F4       OUT  1,X'7F'        1C                    ** INTERRUPT MASK **
       3  000806 B920 0218  LA   1,000218       1C    REG 1 = 00000218
       5  00080A 7134       OUT  1,X'73'        1C                    ** STORAGE PROTECTION **
       6  00080C 8000       LRI  1(0),X'00'     1 Z   REG 1 = 00000018
       7  00080E 7134       OUT  1,X'73'        1 Z                   ** STORAGE PROTECTION **
       8  000810 B920 0419  LA   1,000419       1 Z   REG 1 = 00000419
      10  000814 7134       OUT  1,X'73'        1 Z                   ** STORAGE PROTECTION **
      11  000816 B920 061F  LA   1,00061F       1 Z   REG 1 = 0000061F
      13  00081A 7134       OUT  1,X'73'        1 Z                   ** STORAGE PROTECTION **
      14  00081C B920 0009  LA   1,000009       1 Z   REG 1 = 00000009
      16  000820 7134       OUT  1,X'73'        1 Z                   ** STORAGE PROTECTION **
      17  000822 B840       EXIT                5 Z
*NOW ENTERING LEVEL 4
      18  000180 71FC       IN   1,X'7F'        4     REG 1 = 00000100
      19  000182 F9B4       BB   1(1,7),0001B8  4
      20  000184 710C       IN   1,X'70'        4     REG 1 = 00003000
      21  000186 33A8       SR   3,3            4 Z   REG 3 = 00000000
      22  000188 0308       LCR  3(1),1(0)      4C    REG 3 = 00000030
      23  00018A 7314       OUT  3,X'71'        4C    XR 71 = 00000030 ** DISPLAY **
      24  00018C BA20 0840  LA   2,000840       4C    REG 2 = 00000840
      26  000190 BC20 0002  LA   4,000002       4C    REG 4 = 00000002
      28  000194 8204       LRI  3(0),X'04'     4C    REG 3 = 00000430
      29  000196 4204       OUT  2,X'40'        4C    XR 40 = 00000840
      30  000198 B920 0028  LA   1,000028       4C    REG 1 = 00000028
      32  00019C 4164       OUT  1,X'46'        4C    ICW(020) = 0000000A0000
      33  00019E B920 00C1  LA   1,0000C1       4C    REG 1 = 000000C1
      35  0001A2 4154       OUT  1,X'45'        4C    ICW(020) = 0000C10A0000
      36  0001A4 4290       AHR  2,4            4     REG 2 = 00000842
      37  0001A6 BA93       BCT  3(0),000196    4     REG 3 = 00000330 **BRANCH TAKEN**
      38  000196 4204       OUT  2,X'40'        4     XR 40 = 00000842
      39  000198 B920 0028  LA   1,000028       4     REG 1 = 00000028
      41  00019C 4164       OUT  1,X'46'        4     ICW(021) = 0000000A0000
      42  00019E B920 00C1  LA   1,0000C1       4     REG 1 = 000000C1
      44  0001A2 4154       OUT  1,X'45'        4     ICW(021) = 0000C10A0000
      45  0001A4 4290       AHR  2,4            4     REG 2 = 00000844
      46  0001A6 BA93       BCT  3(0),000196    4     REG 3 = 00000230 **BRANCH TAKEN**
      47  000196 4204       OUT  2,X'40'        4     XR 40 = 00000844
```

```
   TIME   LOCN  OBJ CODE  OP   OPERAND          LCZ      RESULT

     48  000198 B920 0028 LA   1,000028         4     REG 1 = 00000028
     50  00019C 4164      OUT  1,X'46'          4     ICW(022) = 0000000A0000
     51  00019E B920 00C1 LA   1,0000C1         4     REG 1 = 000000C1
     53  0001A2 4154      OUT  1,X'45'          4     ICW(022) = 0000C10A0000
     54  0001A4 4290      AHR  2,4              4     REG 2 = 00000846
     55  0001A6 BA93      BCT  3(0),000196      4     REG 3 = 00000130 **BRANCH TAKEN**
     56  000196 4204      OUT  2,X'40'          4     XR 40 = 00000846
     57  000198 B920 0028 LA   1,000028         4     REG 1 = 00000028
     59  00019C 4164      OUT  1,X'46'          4     ICW(023) = 0000000A0000
     60  00019E B920 00C1 LA   1,0000C1         4     REG 1 = 000000C1
     62  0001A2 4154      OUT  1,X'45'          4     ICW(023) = 0000C10A0000
     63  0001A4 4290      AHR  2,4              4     REG 2 = 00000848
     64  0001A6 BA93      BCT  3(0),000196      4     REG 3 = 00000030
     65  0001A8 B920 1000 LA   1,001000         4     REG 1 = 00001000
     67  0001AC 1184      OUT  1,X'18'          4     XR 18 = 00001000
     68  0001AE BA20 0010 LA   2,000010         4     REG 2 = 00000010
     70  0001B2 72E4      OUT  2,X'7E'          4                 ** INTERRUPT MASK **
*INTERRUPT -- PCI L3
     71  0001B4 70C4      OUT  0,X'7C'          4
     72  0001B6 B840      EXIT                  5
     73  001000 BA20 1800 LA   2,001800         5     REG 2 = 00001800
     75  001004 BC20 1804 LA   4,001804         5     REG 4 = 00001804
     77  001008 BE20 1808 LA   6,001808         5     REG 6 = 00001808
     79  00100C BB20 0004 LA   3,000004         5     REG 3 = 00000004
     81  001010 55A8      SR   5,5              5 Z   REG 5 = 00000000
     82  001012 2510      ICT  5(1),2           5 Z   REG 5 = 00000001 REG 2 = 00001801
     84  001014 2F00      IC   7(1),X'00'(2)    5     REG 7 = 0000005B ADDR = 001801 = 5BC6F700
     86  001016 8600      LRI  7(0),X'00'       5 Z   REG 7 = 0000005B
     87  001018 7778      LCOR 7(1),7(1)        5C    REG 7 = 0000002D
     88  00101A 75C8      XR   5,7              5C    REG 5 = 0000002C
     89  00101C 8806      BZL  001024           5C
     90  00101E 4530      STCT 5(1),4           5C    REG 5 = 0000002C REG 4 = 00001805
     92  001020 57A0      SHR  7,5              5     REG 7 = 00000001
     93  001022 6530      STCT 5(1),6           5     REG 5 = 0000002C REG 6 = 00001809
     95  001024 BB95      BCT  3(1),001012      5     REG 3 = 00000003 **BRANCH TAKEN**
     96  001012 2510      ICT  5(1),2           5     REG 5 = 0000005B REG 2 = 00001802
     98  001014 2F00      IC   7(1),X'00'(2)    5C    REG 7 = 000000C6 ADDR = 001802 = C6F72C00
    100  001016 8600      LRI  7(0),X'00'       5 Z   REG 7 = 000000C6
    101  001018 7778      LCOR 7(1),7(1)        5     REG 7 = 00000063
    102  00101A 75C8      XR   5,7              5C    REG 5 = 00000038
    103  00101C 8806      BZL  001024           5C
    104  00101E 4530      STCT 5(1),4           5C    REG 5 = 00000038 REG 4 = 00001806
    106  001020 57A0      SHR  7,5              5     REG 7 = 0000002B
    107  001022 6530      STCT 5(1),6           5     REG 5 = 00000038 REG 6 = 0000180A
    109  001024 BB95      BCT  3(1),001012      5     REG 3 = 00000002 **BRANCH TAKEN**
```

```
   TIME   LOCN  OBJ CODE  OP    OPERAND          LCZ       RESULT

     110  001012 2510      ICT   5(1),2           5     REG 5 = 000000C6 REG 2 = 00001803
     112  001014 2F00      IC    7(1),X'00' (2)   5     REG 7 = 000000F7 ADDR = 001803 = F72C3800
     114  001016 8600      LRI   7(0),X'00'       5 Z   REG 7 = 000000F7
     115  001018 7778      LCOR  7(1),7(1)        5C    REG 7 = 0000007B
     116  00101A 75C8      XR    5,7              5C    REG 5 = 000000BD
     117  00101C 8806      BZL   001024           5C
     118  00101E 4530      STCT  5(1),4           5C    REG 5 = 000000BD REG 4 = 00001807
     120  001020 57A0      SHR   7,5              5C    REG 7 = 0000FFBE
     121  001022 6530      STCT  5(1),6           5C    REG 5 = 000000BD REG 6 = 0000180B
     123  001024 BB95      BCT   3(1),001012      5C    REG 3 = 00000001 **BRANCH TAKEN**
     124  001012 2510      ICT   5(1),2           5C    REG 5 = 000000F7 REG 2 = 00001804
     126  001014 2F00      IC    7(1),X'00' (2)   5     REG 7 = 0000FF2C ADDR = 001804 = 2C38BD00
     128  001016 8600      LRI   7(0),X'00'       5 Z   REG 7 = 0000002C
     129  001018 7778      LCOR  7(1),7(1)        5     REG 7 = 00000016
     130  00101A 75C8      XR    5,7              5C    REG 5 = 000000E1
     131  00101C 8806      BZL   001024           5C
     132  00101E 4530      STCT  5(1),4           5C    REG 5 = 000000E1 REG 4 = 00001808
     134  001020 57A0      SHR   7,5              5C    REG 7 = 0000FF35
     135  001022 6530      STCT  5(1),6           5C    REG 5 = 000000E1 REG 6 = 0000180C
     137  001024 BB95      BCT   3(1),001012      5C    REG 3 = 00000000
*INTERRUPT -- SVC L4
     138  001026 B840      EXIT                   5C
*NOW ENTERING LEVEL 4
     139  000180 71FC      IN    1,X'7F'          4     REG 1 = 00000103
     140  000182 F9B4      BB    1(1,7),0001B8    4                     **BRANCH TAKEN**
     141  0001B8 72F4      OUT   2,X'7F'          4                     ** INTERRUPT MASK **
*NOW ENTERING LEVEL 3
     142  000100 71FC      IN    1,X'7F'          3     REG 1 = 00000103
     143  000102 F90A      BB    1(1,6),00010E    3                     **BRANCH TAKEN**
     144  00010E B920 0020 LA    1,000020         3     REG 1 = 00000020
     146  000112 7174      OUT   1,X'77'          3
     147  000114 71FC      IN    1,X'7F'          3     REG 1 = 00000101
     148  000116 7114      OUT   1,X'71'          3     XR 71 = 00000101 ** DISPLAY **
     149  000118 B920 0003 LA    1,000003         3     REG 1 = 00000003
     151  00011C 7124      OUT   1,X'72'          3     XR 72 = 00000003 ** DISPLAY **
     152  00011E 110C      IN    1,X'10'          3     REG 1 = 000001BA
     153  000120 9102      ARI   1(1),X'02'       3     REG 1 = 000001BC
     154  000122 1104      OUT   1,X'10'          3     XR 10 = 000001BC
     155  000124 B840      EXIT                   4
     156  0001BC B920 01BC LA    1,0001BC         4     REG 1 = 000001BC
     158  0001C0 1184      OUT   1,X'18'          4     XR 18 = 000001BC
     159  0001C2 B840      EXIT                   5
*ERROR -- PROTECTION CHECK
     160  0001BC                                  5C
*NOW ENTERING LEVEL 1
```

```
   TIME   LOCN  OBJ CODE  OP   OPERAND        LCZ       RESULT

    161  000010 0082      ST   0,X'00'(0)     1 Z   REG 0 = 00000012 ADDR = 000780 = 30000824
    163  000012 0186      ST   1,X'04'(0)     1 Z   REG 1 = 00000009 ADDR = 000784 = 10380009
    165  000014 71EC      IN   1,X'7E'        1 Z   REG 1 = 00000020
    166  000016 7114      OUT  1,X'71'        1 Z   XR 71 = 00000020 ** DISPLAY **
    167  000018 B920 0001 LA   1,000001       1 Z   REG 1 = 00000001
    169  00001C 7124      OUT  1,X'72'        1 Z   XR 72 = 00000001 ** DISPLAY **
*HARDSTOP
    170  00001E 7004      OUT  0,X'70'        1 Z

END OF SIMULATION
```

Appendix C

Suggested References

IBM Corporation, IBM OS Linkage Editor (F) Program Logic, Form GY28-6667, San Jose, CA, 1972.

-----, IBM 3704 and 3705 Communications Controller Principles of Operation, Form GC30-3004, Research Triangle Park, NC, 1974.

-----, IBM 3705 Communications Controller Assembler Language, Form GC30-3003, Research Triangle Park, NC, 1972.

-----, IBM 3705 Communications Controller Emulation Program Generation and Utilities, Guide and Reference Manual, Form GC30-3002, Research Triangle Park, NC, 1972.

-----, IBM 3705 Communications Controller Emulation Program Program Logic Manual, Form SY30-3001, Research Triangle Park, NC, 1972.

-----, Introduction to the IBM 3705 Communications Controller, Form GA27-3051, Research Triangle Park, NC, 1972.

Martin, James, Systems Analysis for Data Transmission, Englewood Cliffs, NJ: Prentice-Hall, 1972.