AN EMULATION OF THE IBM SYSTEM/3 ON THE MICRODATA 1600 MINICOMPUTER

Donald M. Wyckoff
S.P. Teale Data Center, Sacramento, Calif.

I. Introduction.

This paper is the outgrowth of a project conducted by several students at the California State University, Sacramento, commencing during the Fall semester of 1972 and lasting into the summer of 1974. The end product of the student project was an emulation of the IBM System/3 Model 10 computer on the MICRODATA 1600 minicomputer. Phase I of the project culminated in November of 1973 with the completion of a microprogrammed Read Only Memory (ROM). Subsequently, in Phase II, a second, more compact ROM, was microprogrammed and tested. Phase III will consist of an analysis of the performance of the minicomputer in emulating the System/3.

The interest in emulation stems from the increasing costs of software, coupled with the decreasing costs of hardware (1). Emulation appears to offer some possibility of providing transferability of software from one system to another, thereby reducing the costs of conversion from one hardware system to another.

Many authors have defined and discussed simulation, emulation and microprogramming (2) (3) (4) (5) (6) (7) (8) (9) (10) (11) (12) (13). The microprogramming performed in creating this emulation was performed in accordance with the MICRODATA "Microprogramming Handbook" (14).

The rationale for the choice of the IBM System/3 for emulation was based on the large number of System/3 units in existence, the largest single model population in the world (15), and on the consequent large quantity of software that would ultimately need conversion when System/3 units are to be replaced. After consideration of the many methods of computer performance evaluation presented in the literature, (16) (17) (18) (19), a simple software monitor was selected, as described below. Only the initial steps of the performance evaluation were completed to date; an attempt will be made to complete the analysis in the near future.

II. Analysis And Comparison Of The System/3 And The MICRODATA 1600.

A. The IBM System/3.

The organization of the IBM System/3 (20) is shown in Figure 1. The Storage Data Register (SDR) serves as a buffer between main storage and the Arithmetic and Logical Unit (ALU) (via the B register). Data from the ALU can be sent to any of the following:

1. Opcode register
2. Q register
3. Condition register
4. One of the local storage registers (LSRs).
5. Out to an I/O unit.
6. Return to storage through SDR.

Each byte includes 8 bits of information plus a ninth, or parity, bit. The Storage Data Register is used for temporary storage of data as it is passed between the processing portions of the processing unit and main storage. Data enters the SDR from ALU or from main storage. The one byte A and B registers are used to supply temporary storage locations for data entering the ALU. The ALU accomplishes all processing of data, accepts two bytes of input and produces one byte of output. The Storage Address Register (SAR) holds the 2 byte address that is to be accessed in main storage. The Condition register indicates the results of the operations completed in the ALU. The Op Code Register holds the op code byte of each instruction while the Q Register holds the second, or Q byte of the instruction. The Local Storage Registers (LSR) are two-byte registers that hold further data or addresses needed for the execution of instructions. These include:

IAR	Instruction Address Register
PSR	Program Status Register
XR1,2	Index Registers
DAR	Data Address Register
ARR	Address Recall Register
AAR	Operand 2 Address Register
BAR	Operand 1 Address Register

Plus several peripheral device registers.

B. The MICRODATA 1600 Minicomputer.

Figure 2 depicts the block diagram of the MICRODATA 1600, which is also an eight-bit computer, but it differs from the System/3 in that it employs microprogram control. The computer uses eight-bit registers and data paths, executing with every clock pulse a 16 bit microcommand stored in a high speed semiconductor control memory.

The T register acts as a buffer for data being transferred between memory, ALU, and output register, similar in function to the A and B registers of the System/3. The T register is manipulated by the programmer, however, whereas the A and B registers are not.

The M and N registers serve as 16 bit memory address registers similar to the SAR of the System/3.

The file registers provide two banks of 16 8-bit registers to be used as required by the microprogrammer. Only registers 1-15 are available to the user, as register 16 is common to both sets and contains condition code flags. These registers serve the same function as the Local Storage Registers of the System/3 but since they are only one byte registers, two of them must be assigned the function of each LSR to be emulated.

The LINK register acts as two one-bit registers, but actually consists of one two-bit register. One bit is used to indicate the carry-out of the high order bit position of the adder or the shifted off end bit for the shift commands when the information is used as an address in the M/N registers, and the other bit is used for the same purpose for all similar operations on information that is used for any other purpose (normal arithmetic and logical operations). The functions of the LINK register in the 1600 are handled by the Condition Register in the System/3.

The I/O control register specifies the I/O control signal to be enabled, similar to the channel control of the System/3.

The MD register is an additional buffer for data being written into memory, and thus performs part of the functions of the SDR of the System/3. Information may be entered into the MD register from the ALU, as it is in the System/3, but additionally it may be entered into the T register by the programmer.

The output register is a buffer for data being sent to output devices, and thus is similar to the output portion of the I/O channel of the System/3. However, in the 1600 data may be sent from memory via the T register to the Output register without passing through the ALU as in the System/3.

The Arithmetic Logic Unit handles all transfers and manipulation of data as it goes in the System/3.

The Control Memory is a 16 bit high-speed memory implemented with semiconductor Read Only Memory (ROM) or read-write memory providing an alterable control memory (ACM - also referred to as AROM, Alterable Read Only Memory). The control memory can be randomly accessed with an access time, including logic delays, of less than 200 nanoseconds.

The 12 bit L register holds the address of the next control word in sequence. The L-Save register saves the incremented value of the L register when a Jump-Extended command is executed, thus providing a return jump capability when the Return command is used. The R register holds the microcommand currently being executed.

The U register is used to modify the 8 high-order bits of the control memory output. The contents of the U register are ORed with the control memory output on the R bus as it is gated into the R register for specific bit combinations.

III. Design Considerations.

A. General.

The emulation is a complete, 100% instruction emulation, for the peripherals included, with no dedicated main core memory except a hexadecimal to ASCII code conversion table and two bytes to store the Q byte and control code byte for Start I/O (SIO) procedure for the line printer, a total of 66 bytes. Q byte options in the System/3 designed for diagnostic programming have not been implemented. The System/3 front panel is not emulated, which will necessitate slight changes in maintenance procedures.

B. Peripheral Devices Included In The Emulation.

At the inception of this project it was decided to include only the bare minimum of peripheral devices, in order to keep the code as simple as possible, but to maintain the capability for adding other peripheral devices later, after the completion of the basic programming and testing. The initial implementation was limited to the use of the Microdata Model 2720, 80 column card reader and the Microdata 2732, 132 column line printer without the VFU option. Consequently, the only System/3 I/O devices being emulated are the IBM 1442 card reader/punch and the IBM 1403 line printer. No 1442 punching operations are being emulated. However, the need for incorporating additional I/O devices was kept in mind while designing and coding the emulator. The resultant modular organization should facilitate future expansion. For example; addition of an 80 column card punch should be relatively easy to accomplish by adding a few instructions to the Start I/O (SIO) routine for the card reader and to the Concurrent I/O (CIO) routine which handles the actual data transfer. Also five file registers in the secondary file are still available for possible use in an expanded system. Use of these registers along with a DMA Channel would provide for relatively fast, efficient data transfer to or from a disc assembly.

C. ROM Size Versus Speed.

When the project was initiated it was believed that the most important consideration was the speed with which instructions were executed. Hence, it was assumed that we would have access to unlimited Read Only Memory. The resultant ROM, completed in phase I of the project, included slightly less than 1500 microprogram steps, thus requiring 6 - 256 word pages of ROM. It is estimated that the addition of coding to handle a card punch and the System/3 cartridge type disc will increase the ROM size requirement to 8 pages, i.e. 2K words.

In phase II of the project an attempt was made to reduce the size of the ROM as much as possible, with the goal of incorporating the coding in a 1K ROM. This was accomplished by combining several of the instruction routines, and condensing the Fetch Address routine drastically by utilizing portions of the code in common in all of the addressing modes. This compaction of the micro-code resulted in a substantial increase in execution time of several of the individual System/3 instructions. It should be noted, however, that in any commercial program environment, in which the program is I/O bound, the difference between the two ROM's will represent a relatively small fraction of the total execution time.

IV. Design Approach.

A. Top Level Flow Charts.

Figures 3 and 4 show the top level flow charts of the 1.5 K and 1 K ROM respectively. The basic differences are as follows:

1. The Fetch Address routines in the 1.5 K ROM are separate and independent for each addressing mode, whereas they are combined into a single routine in the 1 K ROM. This is described in more detail in paragraph B. below.
2. The Advance Program Level (APL) and Test I/O (TIO) instructions are combined.
3. The Jump On Condition (JC) and Branch on Condition (BC) instructions are combined.
4. The Add Logical Characters (ALC), Subtract Logical Characters (SLC), Compare Logical Characters (CLC), and Compare Logical Immediate (CLI) routines are combined.
5. The Load Register (L) and Add to Register (A) routines are combined.
6. The Test Bits On (TBN) and Test Bits Off (TBF) routines are combined.
7. The Zoned Decimal instructions, ZAZ, AZ, SZ are considerably shortened and combined.

B. System/3 Instruction Handling In The 1.5 K ROM.

The System/3 instruction formats are described in Figure 5; it is to be noted that they may vary from 3 to 6 bytes in length. The first four bits of each instruction is a hexadecimal digit indicating the addressing mode, while the last four bits are another hexadecimal digit indicating the specific op code of the instruction. The second byte, or Q byte, gives additional information about the options of the instruction, and the remaining bytes furnish immediate operands or address information.

The 1.5 K ROM includes a separate routine for each addressing mode, 0 through F, which are labeled FA0 through FAF. The individual routines are entered from the Read Next Instruction (RNI) routine. The RNI routine, after initializing some registers and checking for interrupts, reads the op code byte and saves it in the OPR register, reads the Q byte and saves it in the Q register, then loads the upper four bits of the op code byte (i.e. the hex digit designating the addressing mode) into the L register. The RNI instruction is located at such an address in the ROM that the page number in the L register is set to the page number of the Fetch Address routines. The hex digit, X, that has been placed in the L register, then directs the execution to position 02X0 in the page (256 words) of Fetch Address routines, which is precisely where the X Fetch Address routine has been placed. Thus Fetch Address FA0 is located at 0200, FA1 at 0210, FA2 at 0220, etc. Fortuitously, all of the Fetch Address routines were coded within 16 instructions or less.

The RNI routine and the first two Fetch Address routines are shown in Figure 6. Within the FA routine the appropriate addresses are obtained by reading the next byte or bytes from core, adding the index register values when appropriate, and putting the address in the address registers. Each of the FA routines ends with an instruction that places the second hex digit (the op code) in the K register (the L register with the ninth bit set) for an automatic jump to 030X or 031X, which are jump tables to direct execution of the specific instruction designated by the op code. In Figure 6 the RNI routine has been moved out of its normal sequence in the ROM and placed ahead of the Fetch Address routines for ease in reading.

C. Variations Introduced In The 1 K ROM.

The original 1.5 K ROM utilized a full page (256 words) for the Fetch Address routines, since each of the hex digits 0 through F are located in it at 16 step intervals. Since seven of the routines are shorter than 16 steps, this design left a few gaps of 4 or 6 steps, which were not used, but the whole page was dedicated to the FA routines. It was found to be possible to reduce this (at a cost in execution time) to considerably less than half the size by simply introducing tests and jumps to use identical portions of the code only once rather than repeating them for each Fetch Address mode. However, a much more economical method (in terms of space) was derived by careful review of the bits in the first hex digit of the instruction (that is, the address mode digit). We note that the first two bits (leftmost) determine the type of addressing for the first operand in the following manner:

00	First Address direct
01	First Address indexed by XR1.
10	First Address indexed by XR2.
11	No first address, either no address (F mode) or 2nd address only (C, D or E modes).

It was found that by shifting these bits three positions to the right and then loading them into the L register in a manner similar to that described in the 1.5 K ROM we could create a multiple entry point Fetch Address routine with 4 entry points at intervals of 8 steps, namely 200, 208, 210, 218 (hex) covering the four cases described above. This permitted entry to the coding at the correct point for each of the four first address cases. To take care of the variations of the address register used for the first and second addresses, the Execute Command (EOT) was used to provide a variable command, dependent upon which address was being saved. When this command is executed, the U register is ORed with the oo code to create the desired command, the U register having been loaded previously with the desired bits. The RNI and Fetch Address (FAD) routines are shown in Figure 7.

The 1 K ROM uses the EOT instruction in several places to allow one instruction to do multiple duty.

V. Implementation Methods.

At the time of the initiation of the project the only I/O device available on the Minicomputer at CSUS, Sacramento was an ASR 33 Teletype. In addition, only 1 K of AROM was available on the machine. So a complete ROM could not be loaded into the machine, since the early versions were of the 1.5 K type, and any work done on the machine itself was quite time consuming, hence another means of assembling and testing the firmware was sought.

A. Alternatives Available.

1. Assembly Of Firmware.

Manual assembly was considered but dropped as too tedious and also error prone.

Using the assembler on the 1600 was felt to be too time

consuming with the peripherals that were available. The use of a cross-assembler (written in Fortran,and furnished by MICRODATA) run on the school computer, a CDC 3150, was considered to be the best alternative and was used throughout the project. Portions of typical assemblies were shown in Figures 6 and 7.

2. Testing Of System/3 Instructions.

Console testing directly on the 1600 was considered to be too time consuming and too tedious for the person doing the testing.

The use of the simulator on the MICRODATA 1600 was also considered too time-consuming. It would not, however, been quite as tedious as as console testing, since the Simulator would have printed out the results of each microinstruction without intervention.

The use of the Simulator available on the CDC 3150 (also written in FORTRAN and supplied by MICRODATA) was chosen as the best method and used for all of the Phase I and II testing.

B. Description Of Selected Testing Method.

The Simulator used on the CDC 3150 required as input the version of the ROM undergoing testing; the System/3 instructions to be tested as they would appear in the core of the System/3; a series of event cards establishing the events of the simulation including core dumps, listing or non-listing of each microinstruction, and halting of the simulation at a given number of cycles; and various other parameters of the simulation.

The output of the Simulator included a listing of each microinstruction (optionally omitted under control of a NOLIST event card) complete with the changes of values of all registers, a partial or complete core dump of System/3 simulated core when called for by a DUMP event card, and messages from the Simulator indicating any values of simulated input or output. The time in (200 nanosecond) cycles is also indicated on each listed microinstruction, thus giving us the information necessary for comparing timing of the emulation with actual System/3 timing, and for performance analysis. Samples of the output of the simulator are shown in Figures 8 and 9 for the RNI and FA routines shown in Figures 6 and 7. A feature of the 1.5 K ROM is illustrated in Figures 6 and 8. In this longer version of the ROM, a counting routine was included for the purpose of obtaining a dynamic instruction count during the execution of a program run on the emulation. The counting routine and all calls to it are not included in the 1.5 K instructions, and the time to perform the counting has been deducted from all timing reported in this report.

Calls to the counting routine appear at locations 03F3, 020D, and 021D in Figure 6. The execution of the counting routine appears in Figure 8 at time cycles 16 through 27 and 70 through 81. The counting routine was omitted entirely from the 1 K ROM since the goal was compactness.

V Useful MICRODATA Micro-Programming Features.

A. Features That Were Useful In The Emulation.

1. Byte Orientation.

The byte orientation of the 1600 machine simplified the handling of all of the System/3 instruction bytes, all of the I/O byte handling, and minimized the use of 1600 core for storage of System/3 data. The lack of a parity bit, on the other hand, caused us to decide to ignore parity in the emulation.

2. Jump Extended.

The Jump Extended (JE) instruction, or return jump, was used extensively to permit the use of sub-routines within several of the instruction routines, and is considered to be a necessary instruction to have available in microprogramming.

3. Interleaving Of Register And Memory Instructions.

Since most of the microinstructions are completed in 200 nanoseconds, and memory instructions consume 1 microsecond, the memory instructions would appear to be a considerable slowing factor. It is possible, however, to accomplish most register instructions simultaneously with memory instructions by placing the register instructions adjacent to the memory instructions, thus reducing, or in some cases eliminating the disparity in timing.

4. Register Setting With Memory Instructions.

The read and write instructions in the MICRODATA 1600 micro-code have the optional capability of performing a register transfer operation as well as the memory read or write. This ability was used widely in the emulation to minimize the number of instructions as well as to reduce execution time.

5. Concurrent I/O (CIO).

The Concurrent I/O provision of the 1600 provided a ready means of setting up the cycle stealing used by the System/3 to perform I/O concurrently with other instructions.

6. Separate Links.

The provision of separate overflow links for address and arithmetic instructions greatly simplified the incrementation of addresses without affecting the arithmetic link in multi-byte arithmetic operations.

7. Ability To Change L Register.

The ability of the programmer to set the L register with a single micro-instruction was used in many instances to set up jump tables or jumps directly to instruction coding without the use of the Jump instruction (JP) (which uses 400 nanoseconds); this is described earlier in paragraph IV. B.

8. The Execute Commands.

The special Execute commands, in which the 8 high-order bits of the U register are ORed with the 8 high-order bits of the control memory output, were used in several instances to create a variable command, whose operation was determined by the value of the U register.

B. Desirable Additional Features.

1. Easier Access To Secondary Files.

The ability to directly address the secondary file registers without having to set the secondary files (SSF) would have been very useful in reducing both execution time and number of instructions.

2. More File Registers.

A larger number of file registers would be very useful, particularly if more I/O devices were to be added to the emulation. The alternative that will be utilized is the storing of some of the I/O registers in core, which will, of course, increase the execution time.

3. Inter-Register Transfers.

The direct transfer of the contents of file registers to other file registers would be a distinct boon in reducing both execution time and number of instructions.

4. Ability To Increment A Working Register (Other than file registers).

In many instances this ability would have been very useful.

5. Shift-Left-Four.

The machine has a Shift-Right-Four instruction which proved quite useful, but when a left shift of four bits was required, it was necessary to use the Shift Left (one bit) instruction four times.

6. Direct Access To Arithmetic Link.

A method of accessing the value of the arithmetic link would be desirable in several of the arithmetic operations in the emulation. The method utilized, in the absence of this capability, involved performing an add to register operation (with zero operand) to get the link value into the register and then using the register in the arithmetic operation.

7. A Ninth (Parity) Bit.

The provision of a ninth bit in each byte for use as a parity bit would enhance the emulation of a system such as the System/3, which uses a parity bit.

VII. Conclusions And Recommendations.

Emulation of the IBM System/3 has proven to be relatively easily accomplished be micro-programming on the MICRODATA 1600. The results of the preliminary tests of individual instruction execution times on the System/3, the 1.5 K ROM emulation, and the 1 K ROM emulation are summarized in Table I. The average instruction execution time for the individual instructions was approximately twice that of the System/3 for the 1.5K ROM, and three times the System/3 time for the 1K ROM. A better figure of merit should be developed by obtaining some instruction mix information and weighting each instruction by its relative usage rate. Even more important than the relative frequency of instructions is the degree of I/O boundedness, and a weighting factor that reflects the use of time by the I/O peripherals will provide a much more realistic comparison, and, it is felt, will show the emulation to be much closer in execution time to the System/3.

It would be very desirable to run some System/3 programs on the 1600 with the appropriate I/O peripherals installed. To do this it will probably be necessary to add to the emulation the necessary coding for the cartridge type disk. It would also be desirable to add coding for the Multi-Function Card Unit (MFCU), both for card reading and punching. It is hoped that these tasks can be undertaken in the near future.

BIBLIOGRAPHY

1. Sharpe, William F. The Economics of Computers, First Ed. Columbia University Press, N.Y., 1969

2. Schmidt & Taylor, Simulation and Analysis of Industrial Systems, 1st Ed., Richard D. Irwin Inc., Homewood, Illinois, 1970, Pg:1-9, 479 & 508

3. Jordain, Phillip B. Condensed Computer Encyclopedia, 1st Ed., McGraw-Hill Book Co., 1969.

4. Husson, Samir S, Microprogramming: Principles and Practices, 1st Ed., Prentice-Hall Inc., Englewood Cliffs, N.J., Pg: 1-185

5. Gear, William C, Computer Organization and Programming, 1st Ed., McGraw-Hill Book CO., New York, 1969, Pg: 171-203

6. Lichstein, Henry A, "When Should You Emulate" DATAMATION, Vol 15, No. 11, Pg: 205-21o, Nov., 1969

7. Bell, Gordan and Newell, Allen, Computer Structures: Readings and Examples, 1st Ed., McGraw-Hill Book Co., New York, 1971, Pg: 335-337,339,562-563.

8. Wilkes, M.V, "The Best Way To Design an Automatic Calculating Machine", Manchester University Computer Conference Proceeding, Pg: 16, July, 1951.

9. Wegner, Peter, Programming Languages, Information Structures and Machine Organization, 1st Ed., McGraw-Hill Book Co.. New York, 1968, Pg: 8,28,81,124.

10. Foster, Caxton C. Computer Architecture, 1st Ed., Van Nostrand-Reinhold Co., New York, 1970, Pg: 163.

11. Rosin, Robert F., Frieder, Gideon, and Eckhouse, Robert Jr, "An environment for Research in Microprogramming and Emulation", Communications of The A.C.M. Vol. 15 No. 8 , Pg: 748-760, Aug., 1972.

12. Martin, James, Design of Real-Time Computer Systems, 1st Ed., Prentice Hall, Englewood Cliffs, N.J., 1967, Pg: 268, 346-370

13. Cashman, Michael W. 'Microprogramming For The Many', DATAMATION, Vol. 17, No. 21, Pg: 32, Nov. 1, 1971

14. Microdata Microprogramming Handbook, second edition, Microdata Corp., Santa Ana, CA, 1971.

15. Ferguson, David, "System/3 Doesn't Belong to I.B.M.', DATAMATION, Vol. 19, No. 6, Pg: 62-64, June, 1973

16. Johnson, R. R, "Needed: A Measure For Measure", DATAMATION, Vol. 16, No. 17, Pg: 22-30, Dec. 15, 1970

17. Kolence, K. W, 'Software Physics and Computer Performance Measurements', A.C.M. Proceedings, Pg: 1018-1023, 1972

18. Drummond, Mansfield E., Jr, Evaluation and Measurement Techniques for Digital Computer Systems, 1st Ed., Prentice Hall Inc., Englewood Cliffs, N.J., 1973.

19. Lucas, Henry C., Jr," "Performance Evaluation and Monitoring', Computing Surveys - A.C.M., Vol. 3, No. 3, Pg: 79-92, Sept., 1971

20. I.B.M., I.B.M. System/3 Model 10 Components Reference Manual,, I.B.M. GA21-9103.

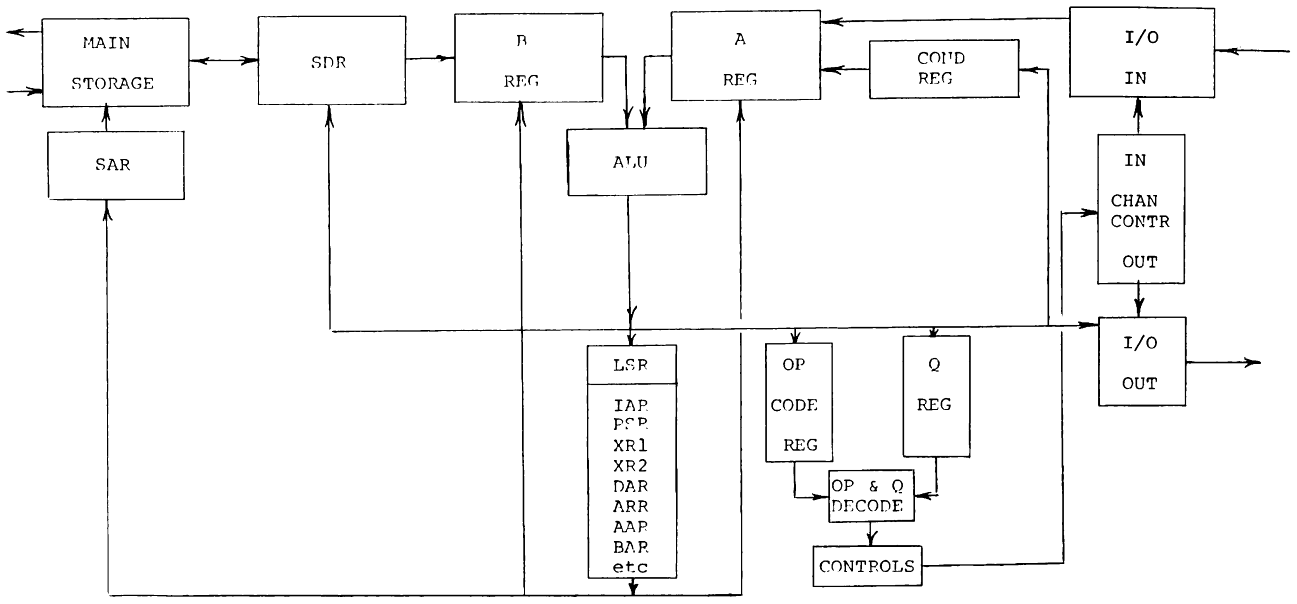

Figure 1. IBM SYSTEM/3 BLOCK DIAGRAM.

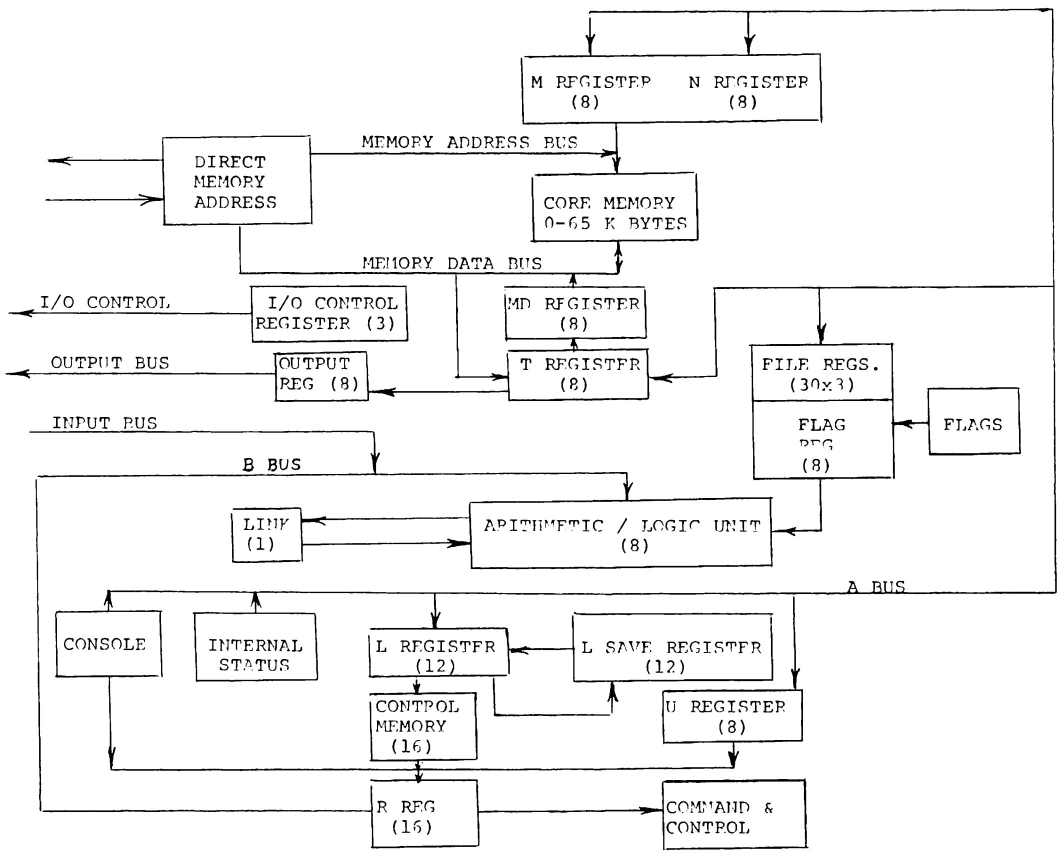

Figure 2. MICRODATA 1600 BLOCK DIAGRAM.

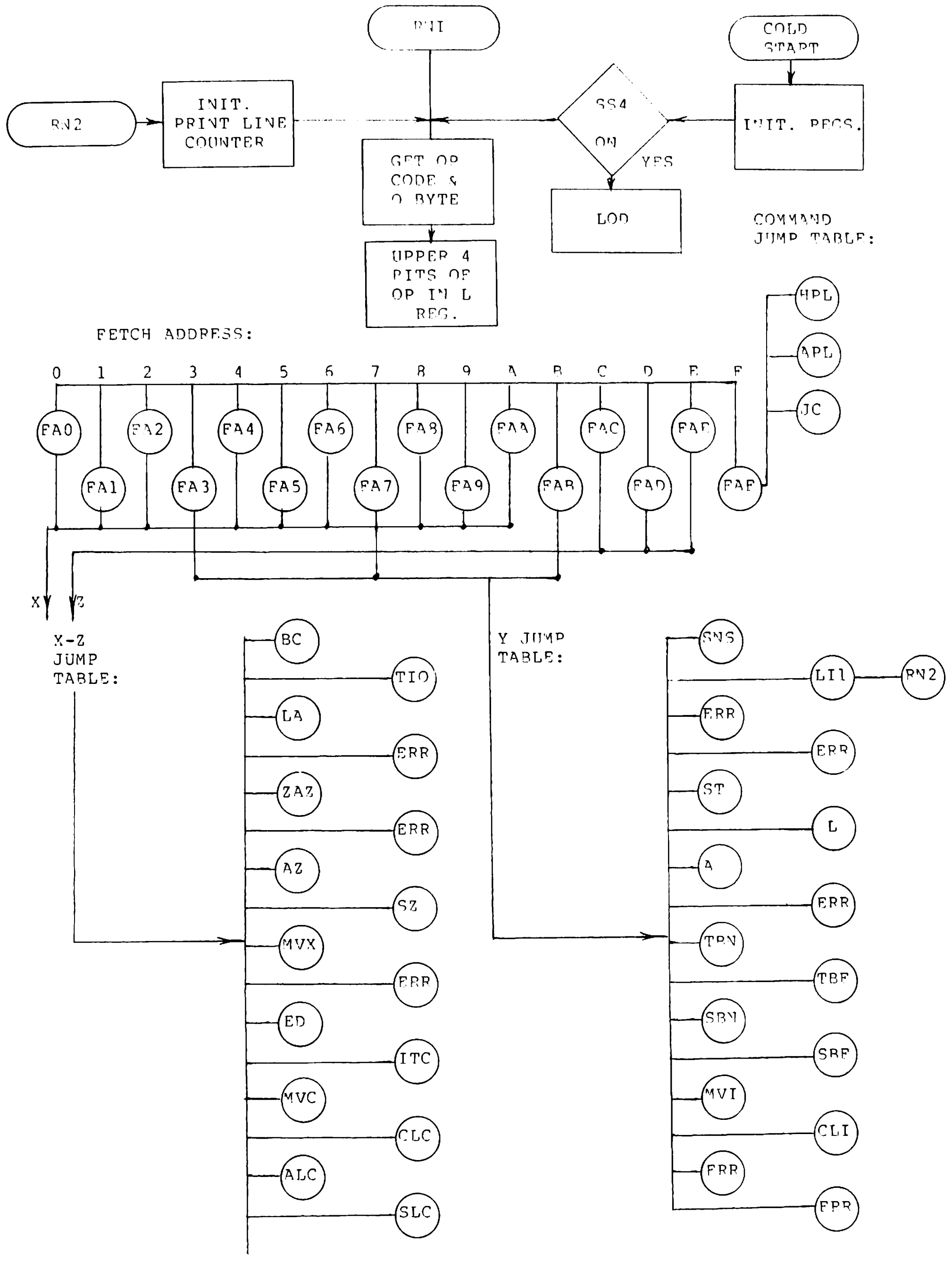
RNI
COLD START
RN2
INIT. PRINT LINE COUNTER
SS4 ON
YES
INIT. REGS.
GET OP CODE & Q BYTE
LOD
COMMAND JUMP TABLE:
UPPER 4 BITS OF OP IN L REG.
HPL
APL
JC
FETCH ADDRESS:
0 1 2 3 4 5 6 7 8 9 A B C D E F
FA0 FA1 FA2 FA3 FA4 FA5 FA6 FA7 FA8 FA9 FAA FAB FAC FAD FAE FAF
X
Z
X-Z JUMP TABLE:
BC TIO LA ERR ZAZ ERR AZ SZ MVX ERR ED ITC MVC CLC ALC SLC
Y JUMP TABLE:
SNS LI1 RN2 ERR ERR ST L A ERR TBN TBF SBN SBF MVI CLI ERR ERR

Figure 3. TOP LEVEL FLOW CHART - 1.5 K ROM.

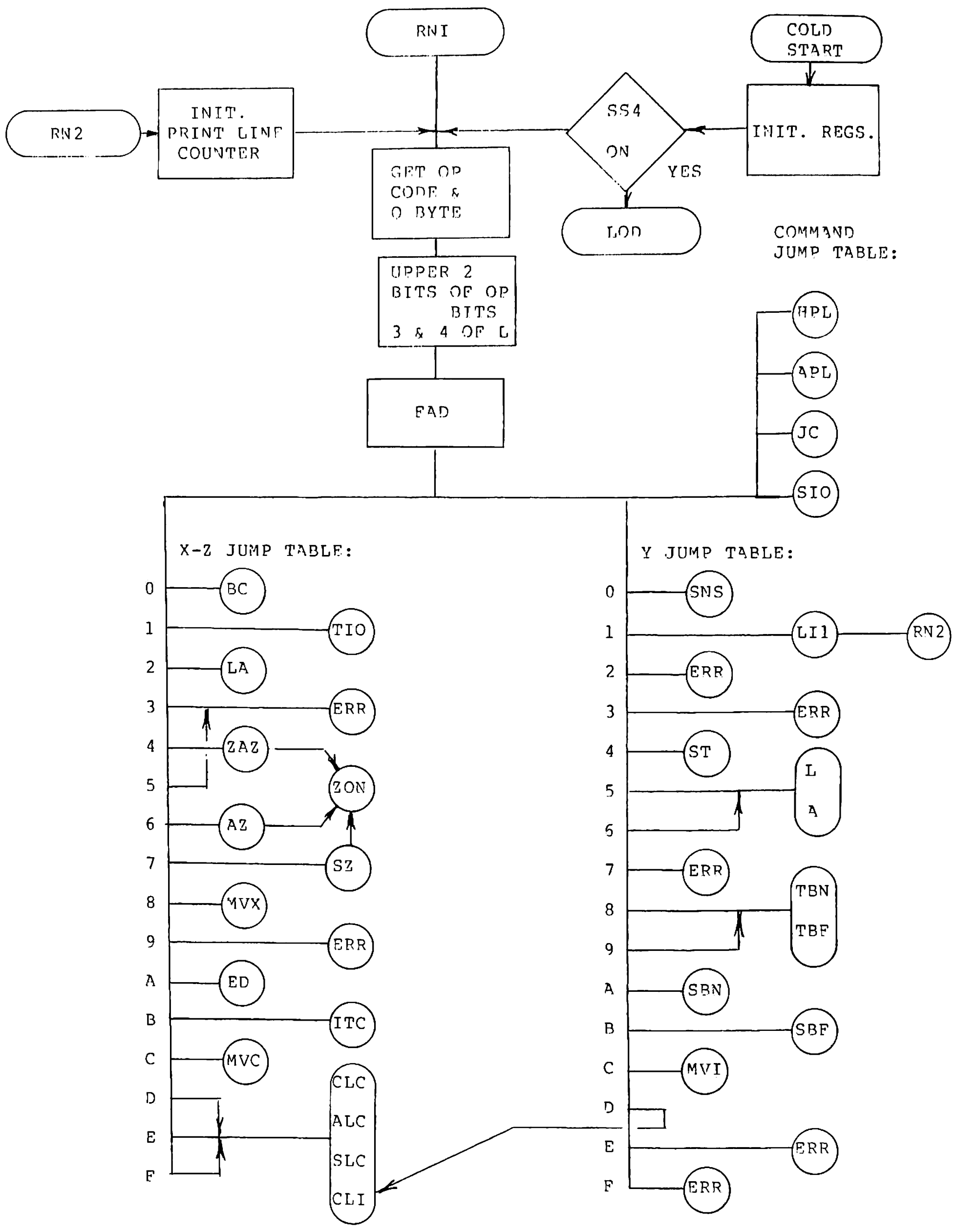

Figure 4. TOP LEVEL FLOW CHART - 1 K ROM.

Command Instruction

				Index Regs.
	Op Code	Q Byte	Command	
F	1111xxxx			None

0 3
Bits

One Address Instruction - Indexed

E	1110		Displac.	-,2
D	1101			-,1
B	1011		Operand	2,-
7	0111			1,-

One Address Instruction - Direct

3	0011		Op X Hi ord. Addr. Byte	Op X Lo ord. Addr. Byte	D,-
C	1100				-,D

Two Address Instruction - Both Indexed

5	0101		Operand 1 Displac.	Operand 2 Displac.	1,1
6	0110				1,2
9	1001				2,1
A	1010				2,2

Two Address Instruction - Op 1 Direct

1	0001		Op 1 Hi ord Addr. Byte	Op 1 Low Ord Addr. Byte	Op 2 Displ.	D,1
2	0010					D,2

Two Address Instruction - Op 2 Direct

4	0100		Op 1 Displ.	Op2 Hi ord Addr. Byte	Op 2 Low ord Addr. Byte	1,D
8	1000					2,D

Two Address Instruction - Both Addresses Direct

0	0000	Byte	Op 1 Hi ord. Address Byte	Op 1 Low ord. Address Byte	Op 2 Hi ord. Address Byte	Op 2 Low ord. Address	D,D

Figure 5. SYSTEM/3 INSTRUCTION FORMATS.

```
LOCN CODE FLAGS LABELS OP *   OPERANDS      COMMENTS

03F1 1600        RNI    LU     X'00'          CLEAR U FOR EXT JUMP
03F2 2DFE               LF     W1,X'FE'       COUNTER ID
03F3 000A               JE     CNT            GO TO COUNTER ROUTINE
                 *
                 *     INTERRUPT TEST
                 *
03F4 4018               TZ     F0,X'18'       IS THERE AN INTERRUPT?
03F5 03D7               JE     INT            YES, TEND TO IT
                 *
03F6 8343        RN5    INC    IAL,(N)        SET UP TO
03F7 A282               RMF    IAU,L,(M)      GET OP CODE
03F8 BF21        RN6    CPY    OPR,T,(T)      SAVE OP CODE
03F9 2DF0               LF     W1,X'F0'       MASK FOR 1ST HEX CHARACTER
03FA 8343               INC    IAL,(N)        BUMP UP POINTER TO CORE
03FB A282               RMF    IAU,L,(M)      GET BYTE
03FC 8C20               CPY    Q,T            SAVE Q BYTE
03FD CF01               MOV    OPR,(T)        GET OP CODE
03FE ED24               AND    W1,T,(L)       JUMP INTO TABLE
                 *
                 *     FETCH ADDRESS
                 *
0200                    ORG    X'200'
0200 8343        FA0    INC    IAL,(N)        SET UP N
0201 A282               RMF    IAU,L,(M)      FETCH 1ST BYTE OF ADDRESS
0202 B620               CPY    BAU,T          PUT IT IN BAR
0203 8343               INC    IAL,(N)        SET UP N
0204 A282               RMF    IAU,L,(M)      READ 2ND BYTE OF ADDRESS
0205 B720               CPY    BAL,T          PUT IN BAR, LOWER
0206 8343               INC    IAL,(N)        SET N
0207 A282               RMF    IAU,L,(M)      GET 1ST BYTE OF 2ND ADDRESS
0208 B420               CPY    AAU,T          PUT IT IN AAR, UPPER
0209 8343               INC    IAL,(N)        SET UP N AGAIN
020A A282               RMF    IAU,L,(M)      READ 2ND BYTE OF 2ND ADDRESS
020B B520               CPY    AAL,T          PUT IT IN AAR LOWER
020C 2DA0               LF     W1,X'A0'       COUNTER ID
020D 000A               JE     CNT            GO TO COUNTER ROUTINE
020E 110F               LT     X'0F'          MASK FOR OP CODE
020F EF2D               AND*   OPR,T,(K)      JUMP TO OP CODE IN X-Z JUMP TABLE
                 *
0210                    ORG X'210'
0210 8343        FA1    INC    IAL,(N)        SET UP N REG
0211 A282               RMF    IAU,L,(M)      READ 1ST BYTE OF 1ST ADDRESS
0212 B620               CPY    BAU,T          PUT IT IN BAR UPPER
0213 8343               INC    IAL,(N)        SET UP N REG
0214 A282               RMF    IAU,L,(M)      READ 2ND BYTE OF 1ST ADDRESS
0215 B720               CPY    BAL,T          PUT IT IN BAR LOWER
0216 8343               INC    IAL,(N)        SET UP N REG
0217 A282               RMF    IAU,L,(M)      READ 1ST BYTE OF 2ND ADDRESS
0218 8929               ADD*   X1L,T,(T)      ADD INDEX REG 1
0219 B520               CPY    AAL,T          PUT IT IN AAR UPPER
021A 8889               ADD*   X1U,L,(T)      ADD LINK TO UPPER INDEX REG
021B B420               CPY    AAU,T          PUT IT IN AAR UPPER
021C 2DA2               LF     W1,X'A2'       COUNTER ID
021D 000A               JE     CNT            GO TO COUNT ROUTINE
021E 110F               LT     X'0F'          MASK FOR OP CODE
021F EF2D               AND*   OPR,T,(K)      JUMP TO OP CODE IN X-Z TABLE
```

FIGURE 6. ASSEMBLED RNI, FA0, AND FA1 ROUTINES - 1.5K ROM.

```
LOCN CODE FLAGS LABELS OP *   OPERANDS     COMMENTS

03D7 1600        RNI    LU     X'00'         CLEAR  U FOR JUMP EXT
03D8 4018               TZ     F0,X'18'      IS THERE AN INTERRUPT?
03D9 000A               JE     INT           YES, TEND TO IT
03DA 8343        RN5    INC    IAL,(N)       NO, SET UP N REG
03DB A282               RMF    IAU,L,(M)     READ OP CODE BYTE
03DC 8F21        RN6    CPY    OPR,T,(T)     SAVE IN OPR REG
03DD 8E21               CPY    W2,T,(T)      ALSO IN W2 REG
03DE 2018               LF     W1,X'18'      MASK FOR SIGNIF BITS
03DF 8343               INC    IAL,(N)       SET UP N REG
03E0 A282               RMF    IAU,L,(M)     READ Q BYTE
03E1 8C20               CPY    Q,T           SAVE IN Q REG
03E2 16B2               LU     X'B2'         SET UP OP CODE FOR COPY AND REG 2
03E3 8343               INC    IAL,(N)       SET UP N REG
03E4 FE21               SFR    W2,(T)        SHIFT RIGHT ONE BIT
03E5 FE21               SFR    W2,(T)        SHIFT RIGHT ONE BIT
03E6 FE21               SFR    W2,(T)        SHIFT RIGHT ONE BIT
03E7 ED2C               AND*   W1,T,(L)      MASK WITH SHIFTED OP CODE
                 *                           AND PUT IN L REG TO SET
                 *                           UP JUMP TO PROPER FA ROUTINE
0200                    ORG    X'200'        SET UP INITIAL ENTRY POINT AT 200 FOR DIR ADD
0200 A282        FAD    RMF    IAU,L,(M)     READ BYTE OF ADDRESS
0201 0420               EOT    AAU,T         THIS INSTRUCTION IS ORED WITH U REG, PICKING
                 *                           UP THE COPY OP CODE AND BITS FOR THE PROPER
                 *                           REG FOR STORING THE UPPER BYTE OF THE ADDRESS
0202 8343               INC    IAL,(N)       SET UP TO
0203 A282               RMF    IAU,L,(M)     READ NEXT BYTE
0204 0520               EOT    AAL,T         OR WITH U FOR LOWER BYTE IN SAME FASHION
0205 6EFA               CP     W2,X'FA'      IS THIS THE LAST BYTE?
0206 1C1F               JP     FA1           NO, GO ON
0207 1C27               JP     FAF           YES, GO TO INSTRUCTION ROUTINE
0208 A282               RMF    IAU,L,(M)     ENTRY FOR IR1 ADDRESS - READ BYTE
0209 8929               ADD*   X1L,T,(T)     ADD INDEX REG 1
020A 0520               EOT    AAL,T         OR U - COPY TO ADDRESS REG
020B 8889               ADD*   X1U,L,(T)     ADD LINK TO INDEX REG 1
020C 0420               EOT    AAU,T         OR U - COPY TO ADDRESS REG
020D 6EF2               CP     W2,X'F2'      IS THIS THE LAST BYTE?
020E 1C1F               JP     FA1           NO, GO ON
020F 1C27               JP     FAF           YES, GO TO INSTRUCTION
0210 A282               RMF    IAU,L,(M)     ENTRY FOR INDEX REG 2 ADDRESS - READ BYTE
0211 8829               ADD*   X2L,T,(T)     ADD INDEX REG 2
0212 0520               EOT    AAL,T         OR U - COPY TO ADDRESS REG
0213 8A89               ADD*   X2U,L,(T)     ADD LINK TO INDEX REG 2
0214 0420               EOT    AAU,T         OR U - COPY TO ADDRESS REG
0215 6EEA               CP     W2,X'EA'      IS THIS LAST BYTE?
0216 1C1F               JP     FA1           NO, GO ON
0217 1C27               JP     FAF           YES GO TO INSTRUCTION
0218 6F10               CP     OPR,X'10'     ENTRY FOR NO 1ST ADDRESS - IS IT F TYPE?
0219 1C23               JP     FA2           NO, GO TO ROUTINE FOR 2ND ADDRESS
021A 1103               LT     X'03'         MASK FOR TWO LOW BITS
021B 1600               LU     X'00'         PREP U REG FOR INSTRUCTION
021C EF29               AND*   OPR,T,(T)     SET UP FOR
021D 2E32               LF     W2,COM        COMMAND JUMP TABLE ENTRY
021E 8E24               ADD    W2,T,(L)      JUMP TO COMMAND JUMP TABLE
021F 4D80        FA1    TZ     W1,X'80'      WAS THIS THE FIRST BYTE?
0220 1C27               JP     FAF           NO, GO TO INSTRUCTION ROUTINE
0221 2D98               LF     W1,X'98'      YES, SET FLAGS FOR 2ND BYTE
0222 8343               INC    IAL,(N)       PREPARE N FOR NEXT BYTE
0223 1680        FA2    LU     X'B0'         LOAD FOR COPY AND AAU REG
0224 FF29               SFR*   OPR,(T)       SHIFT OP CODE ONE BIT RIGHT
0225 BE21               CPY    W2,T,(T)      COPY IT TO W2 AND T
0226 ED2C               AND*   W1,T,(L)      JUMP TO ROUTINE FOR 2ND BYTE
0228 110F        FAF    LT     X'0F'         MASK FOR LOWER 4 BITS
0229 1600               LU     X'00'         CLEAR U FOR INSTRUCTION
022A 4D80               TZ     W1,X'80'      WAS THIS 2ND BYTE?
022B EF2D        FAE    AND*   OPR,T,(K)     YES, GO TO INSTRUCTION - X-Z TYPE
022C 4E60               TZ     W2,X'60'      NO, IS IT A Y TYPE ADDRESS?
022D 1C2B               JP     FAE           NO, GO BACK TO X-Z TYPE
022E EF29               AND*   OPR,T,(T)     YES, STRIP OUT UPPER 4 BITS
022F BE20               CPY    W2,T          AND PUT IT IN W2
0230 1110               LT     X'10'         SET UP TO
0231 8E25               ADD    W2,T,(K)      ADD 16 AND JUMP TO Y JUMP TABLE
0232 158E        COM    JP     HPL           COMMAND
0233 1C84               JP     APL           INSTRUCTION
0234 14AA               JP     JC            JUMP
0235 1CDE               JP     SIO           TABLE
```

FIGURE 7. ASSEMBLED RNI, AND FAD ROUTINES - 1K ROM.

```
MICRODATA 1600 SIMULATOR   ** IBM SYSTEM/3 MOD. 10 EMULATION ON MICRODATA 1600 - ROMB2 SIMULATION **

TIM LOCN  ROM    OP     OPND     SAVE U   M   N   I/O   T LINK  FILE REGISTERS                               A   B   C   D   E   F
CYC  (L)  (R)    CODE                                                0   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15
 12  3F1  1600   LU     00
 13  3F2  2DFE   LF     D,FE          00                        P                                                        FE
 15  3F3  000A+U JE     00A      3F4
 16  00A  8D43   INC    D(N)                  FF           0    P                                                        FF
 17  00B  1203   LM     03                03
 18  00C  BE01   ZOF    E(T)                           00       P                                                           00
 19  00D  A020   RMH    0                              00
 22  00E  8E61   ADD    E,I,T(T)                       01  0    P                                                           01
 23  00F  A030   WMH    0                              01
 24  010  4EFF   TZ     E,FF
 27  011  1020   RTN             012
 29  3F4  4018   TZ     0,18
 30  3F6  8343   INC    3(N)                  00           1    P               00
 31  3F7  A282   RMF    2,L(M)            00           00  1    P           00
 34  3F8  BF21   CPY    F,T(T)                         04       P                                                               04
 35  3F9  2DF0   LF     D,F0                                    P                                                        F0
 36  3FA  8343   INC    3(N)                  01           0    P               01
 37  3FB  A282   RMF    2,L(M)                         00  0
 40  3FC  BC20   CPY    C,T                            22       P                                                    22
 41  3FD  CF01   MOV    F(T)                           04
 43  3FE  ED24   AND    D,T(L)                         04       P                                                        00
 44  200  8343   INC    3(N)                  02           0    P               02
 45  201  A282   RMF    2,L(M)                         00  0
 48  202  B620   CPY    6,T                            02       P                           02
 50  203  8343   INC    3(N)                  03           0    P               03
 51  204  A282   RMF    2,L(M)                         00  0
 54  205  B720   CPY    7,T                            45       P                               45
 56  206  8343   INC    3(N)                  04           0    P               04
 57  207  A282   RMF    2,L(M)                         00  0
 60  208  B420   CPY    4,T                            02       P                   02
 62  209  8343   INC    3(N)                  05           0    P               05
 63  20A  A282   RMF    2,L(M)                         00  0
 66  20B  B520   CPY    5,T                            48       P                       48
 67  20C  2DA0   LF     D,A0                                    P                                                        A0
 69  20D  000A+U JE     00A      20E
 70  00A  8D43   INC    D(N)                  A1           0    P                                                        A1
 71  00B  1203   LM     03                03
 72  00C  BE01   ZOF    E(T)                           00       P                                                           00
 73  00D  A020   RMH    0                              00
 76  00E  8E61   ADD    E,I,T(T)                       01  0    P                                                           01
 77  00F  A030   WMH    0                              01
 78  010  4EFF   TZ     E,FF
 81  011  1020   RTN             012
 82  20E  110F   LT     0F                             0F
 84  20F  EF2D   AND*   F,T(K)                         0F
 86  304  0140+U JE     140      305
```

FIGURE 8. PORTION OF 1.5K ROM SIMULATION.

```
  MICRODATA 1600 SIMULATOR    **  IBM SYSTEM/3 MOD. 10 EMULATION ON MICRODATA 1600 - ROMA4 SIMULATION **

TIM LOCN  ROM    OP   OPND  SAVE  U  M  N   I/O  T LINK FILE REGISTERS                    A  B  C  D  E  F
CYC  (L)  (R)    CODE                                       0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
 12  3D7  1600   LU    00
 14  3D8  4018   TZ   0,18       00
 15  3DA  8343   INC  3,(N)            00            1   P          00
 16  3DB  A282   RMF  2,L(M)      00          00   1   P       00
 19  3DC  BF21   CPY  F,T(T)                 04       P                                               04
 20  3DD  BE21   CPY  E,T(T)                 04       P                                            04
 21  3DE  2D18   LF   D,18                            P                                         18
 22  3DF  8343   INC  3(N)             01            0   P          01
 23  3E0  A282   RMF  2,L(M)                 00   0
 26  3E1  BC20   CPY  C,T                    22       P                                      22
 27  3E2  16B2   LU    B2
 28  3E3  8343   INC  3(N)       B2    02            0   P          02
 29  3E4  FE21   SFR  E(T)                   02  0    P                                            02
 30  3E5  FE21   SFR  E(T)                   01  0    P                                            01
 31  3E6  FE21   SFR  E(T)                   00  1    P                                            00
 33  3E7  ED2C   AND* D,T(L)                 00
 34  200  A282   RMF  2,L(M)                 00   0
 37  201  0420,U CPY  6,T                    02       P                   02
 39  202  8343   INC  3(N)             03            0   P          03
 40  203  A282   RMF  2,L(M)                 00   0
 43  204  0520,U CPY  7,T                    45       P                      45
 44  205  6EFA   CP   E,FA                       0
 46  206  1C1F   JP   21F
 48  21F  4D80   TZ   D,80
 49  221  2098   LF   D,98                            P                                         98
 50  222  8343   INC  3(N)             04            0   P          04
 51  223  16B0   LU    B0
 52  224  FF29   SFR* F(T)       B0             02  0
 53  225  BE21   CPY  E,T(T)                 02       P                                            02
 55  226  ED2C   AND* D,T(L)                 02
 56  200  A282   RMF  2,L(M)                 00   0
 59  201  0420,U CPY  4,T                    02       P             02
 61  202  8343   INC  3(N)             05            0   P          05
 62  203  A282   RMF  2,L(M)                 00   0
 65  204  0520,U CPY  5,T                    48       P                48
 66  205  6EFA   CP   E,FA                       0
 68  206  1C1F   JP   21F
 69  21F  4D80   TZ   D,80
 71  220  1C27   JP   227
 72  227  8F49   INC* F(T)                   05  0
 73  228  110F   LT    0F                    0F
 74  229  1600   LU    00        B0
 75  22A  4D80   TZ   D,80       00
 77  22B  EF2D   AND* F,T(K)                 0F
 79  304  1D23   JP   323
```

FIGURE 9. PORTION OF 1K ROM SIMULATION.

TABLE I.

SUMMARY OF EMULATION & SYSTEM/3
INSTRUCTION EXECUTION TIMES

		NO. OF	TIME IN MICROSECONDS		
MNEM.	INSTRUCTION	BYTES	SYS/3	1.5 K ROM	1 K ROM
LA	Load Address	-	5.07	7.90	13.04
L	Load Register	-	8.11	13.00	15.94
ST	Store Register	-	8.11	12.47	15.53
A	Add to Register	-	8.11	15.67	19.27
ZAZ	Zero & Add Zoned	5	13.17	22.27	85.13
AZ	Add Zoned	5	13.17	33.47	90.13
SZ	Subtract Zoned	5	13.17	33.87	89.13
CLC	Compare Logical Characters	3	13.17	18.67	28.66
CLI	Compare Logical Immediate	-	6.59	9.93	15.93
MVX	Move Hex. Characters	-	10.13	13.22	17.78
MVC	Move Characters	3	10.64	16.07	20.73
MVI	Move Immediate	-	6.59	7.40	10.50
ALC	Add Logical Characters	3	10.13	20.80	29.66
SLC	Subtract Logical Character	3	10.13	21.00	29.46
ED	Edit	3	10.64	21.27	25.84
ITC	Insert & Test Characters	3	11.65	20.13	25.33
SBN	Set Bits On	-	6.59	8.50	11.70
SBF	Set Bits Off	-	6.59	8.50	11.70
TBN	Test Bits On	-	6.59	8.50	11.70
TBF	Test Bits Off	-	6.59	8.50	12.10
BC	Branch On Condition	-	5.07	11.00	16.40
JC	Jump On Condition	-	5.07	11.00	16.40
HPL	Halt Program Level	-	4.56	6.40	7.80
SIO	Start I/O	-	4.56	13.10	14.10
SNS	Sense I/O	-	8.11	15.40	19.60
LIO	Load I/O	-	8.11	12.60	16.80
TIO	Test I/O	-	5.07	13.60	19.80
APL	Advance Program Level	-	4.56	10.20	13.20

Note: All times are mean times for the group of options tested on each instruction.