A program to operate a remote terminal for a small research computer

Kinread, W. R.

Report (National Research Council of Canada. Radio and Electrical Engineering Division : ERB), 1968-08

ERB-789

UNCLASSIFIED

NATIONAL RESEARCH COUNCIL OF CANADA

RADIO AND ELECTRICAL ENGINEERING DIVISION

A PROGRAM TO OPERATE A REMOTE TERMINAL FOR A SMALL RESEARCH COMPUTER

- W. R. KINREAD -

OTTAWA

AUGUST 1968

ERB-789

UNCLASSIFIED

NATIONAL RESEARCH COUNCIL OF CANADA

RADIO AND ELECTRICAL ENGINEERING DIVISION

A PROGRAM TO OPERATE A REMOTE TERMINAL FOR A SMALL RESEARCH COMPUTER

– W.R. Kinread –

OTTAWA

AUGUST 1968

ABSTRACT

A program written for the first stage of development of a system of remote typewriters for a small research computer with 8K words is described. The computer used must have an on-line typewriter and two interrupts which are devoted to typewriter input/output. An operator at the typewriter is allowed to list and modify words in a preassigned area of memory and to debug a program in this area while the rest of memory is guarded against interference from the operator and his program.

CONTENTS

FIGURES

TABLE

A PROGRAM TO OPERATE A REMOTE TERMINAL FOR A SMALL RESEARCH COMPUTER

– W.R. Kinread –

INTRODUCTION

The remote terminal dealt with in this report is an on-line typewriter connected to a Systems Engineering Laboratories (SEL) 840A computer. The typewriter is an interfaced ASR-33 teletype with a paper tape reader and punch as well as a keyboard and printer.

The purpose of a remote terminal is to permit better use of computer time by allowing more than one person to use the computer concurrently. The remote terminal program described here is the first stage in the development of a multiterminal system, each terminal of which would allow an operator to load a program into memory as well as being able to examine, alter, debug, and execute it while another program is running on the computer. It was thought that the best way to achieve this aim would be to give the typewriter most of the facilities of the console and to add any features that could be easily implemented.

The remote terminal program will operate on the basic SEL 840A computer with 4K word memory and the two standard interrupts. The console typewriter is used as the terminal and the program occupies 2072_8 words of memory (excluding an area set aside for the terminal operator's program).

SYSTEM DESIGN

One problem associated with running two programs concurrently is the danger of one program unintentionally altering the other. The program that is sharing computer time with the remote terminal program[1] is assumed to be debugged and will not affect the terminal program. The terminal operator must therefore be prevented from altering the background program. This is achieved by setting aside an area of memory for his use and by preventing him and his program from changing words outside of this area.

Certain instructions in the operator's program such as PIE (interrupt enable) which affect the entire system must not be executed. These instructions must be detected and declared illegal. The execution of others must be simulated because some console functions have been simulated at the typewriter (e.g., sense switches). Since the computer used does not have an instruction trap[2], each instruction must be checked for legality and its execution must be simulated to prevent restricted areas of memory from being changed. The operator's program is, therefore, data for the terminal program which simulates execution. (This will be referred to as pseudo-execution.)

---

[1] *This program will be referred to as 'background' or the 'background program'.*

[2] *An instruction trap consists of combinational logic which issues an interrupt signal when certain instructions are contained in the execution register.*

If the operator's program and the background program attempt to use the same input/output (I/O) device, their data may become mixed. This difficulty was solved by restricting the terminal operator to only one device – the terminal typewriter.

The I/O interrupts[3] are used for communication between the operator and the computer. The input interrupt signal occurs when the typewriter's input buffer receives a new character as a result of a key being depressed or of the tape reader reading one character. The input interrupt is therefore used to *initiate* the terminal program. The output interrupt occurs when the typewriter's output buffer is emptied. Thus, the output interrupt is used when data are to be typed out and control is *returned* to the terminal program.

Figure 1*a* is a memory map and illustrates that the terminal program is independent of all other programs. It should be noted that no monitor is required since the terminal program interprets its own two interrupts.

Figure 1*b* is a block diagram of the system. When a character is keyed in from the typewriter, the input interrupt routine is entered and control is transferred to either the console-simulating block, the pseudo-execution block, or to the background program. If data are to be printed, the routine FIX is entered. The contents of all registers are saved and control is returned to the background program. When the output interrupt occurs, the output interrupt routine is entered, the registers saved by FIX are restored, and the character stored in the A accumulator is printed. Control is then returned to the appropriate memory location in the terminal program which was saved by FIX.

THE COMMAND LANGUAGE

The command language is the means by which the operator communicates with the computer and is described in Appendix A.

Most of the commands are used to simulate the console functions. The A, B, program counter, and index registers are simulated by assigning one word in memory to each and the terminal operator is able to load or display these registers from the typewriter. To load a register, the appropriate command is keyed in, followed by the data in octal format. When a display command is keyed in, the contents of the register are displayed by printing them as octal numbers. These are much easier to read than the lights on the hardware console.

The control switches on the main console, including the three-way sense/halt switch, are also simulated. The operator can set, reset, or display these control switches as well as operate the simulated sense/halt switch and display the sense switches. He can also load or display consecutive memory locations in the specified area of core. The single-cycle execution, start, and halt switches are also simulated.

---

[3] *An interrupt is a signal which stops the normal execution of a program so that control can be transferred to another program. Provision is made so that control may be returned to the original program at the point where the interrupt occurred. Each interrupt has a priority assigned to it so that if more than one interrupt occurs, the one with the highest priority is attended to first.*

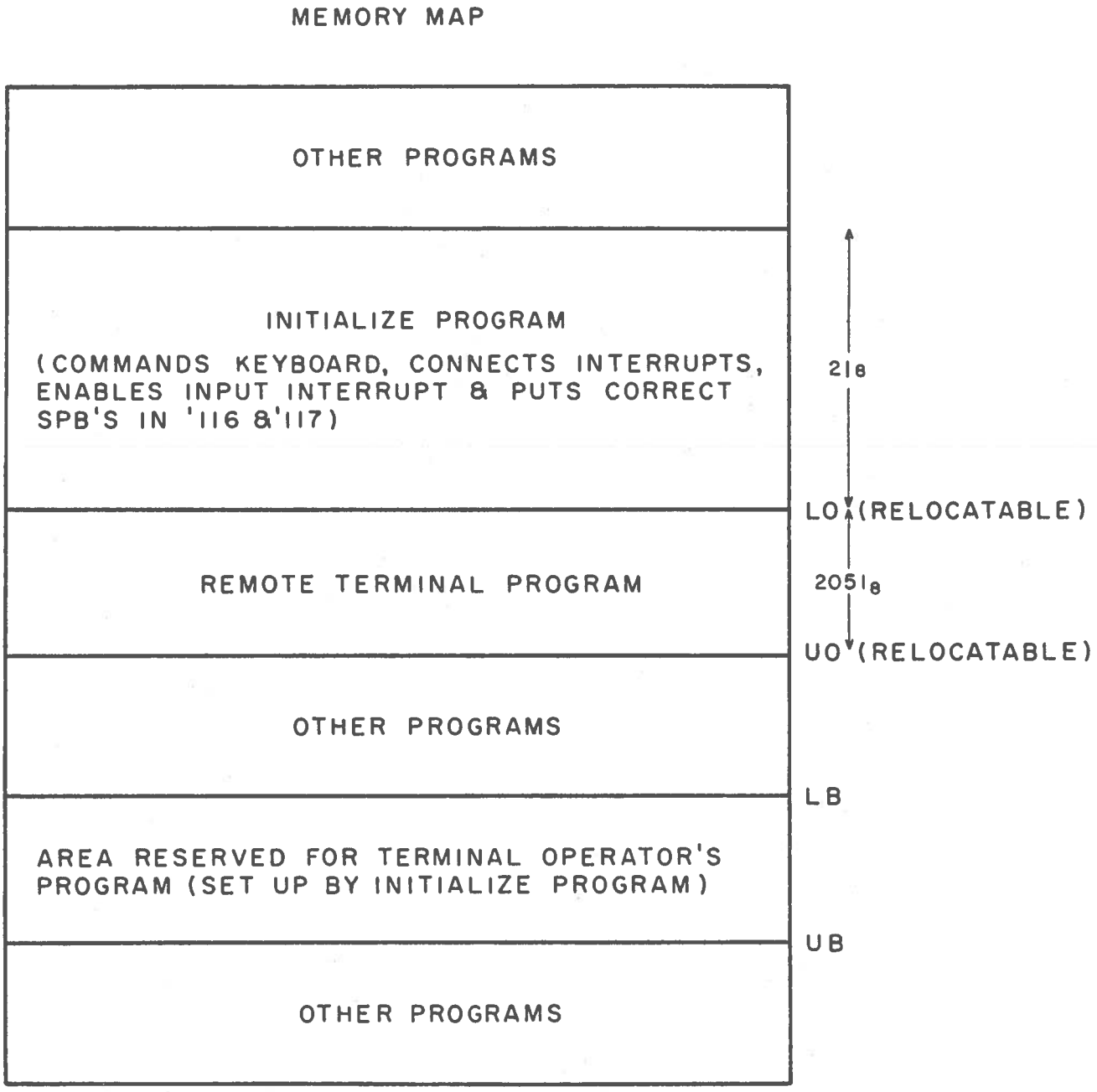

MEMORY MAP
OTHER PROGRAMS
INITIALIZE PROGRAM
(COMMANDS KEYBOARD, CONNECTS INTERRUPTS, ENABLES INPUT INTERRUPT & PUTS CORRECT SPB'S IN '116 &'117)
218
LO (RELOCATABLE)
REMOTE TERMINAL PROGRAM
2051₈
UO (RELOCATABLE)
OTHER PROGRAMS
LB
AREA RESERVED FOR TERMINAL OPERATOR'S PROGRAM (SET UP BY INITIALIZE PROGRAM)
UB
OTHER PROGRAMS

Figure 1(a)

BLOCK DIAGRAM OF THE SYSTEM

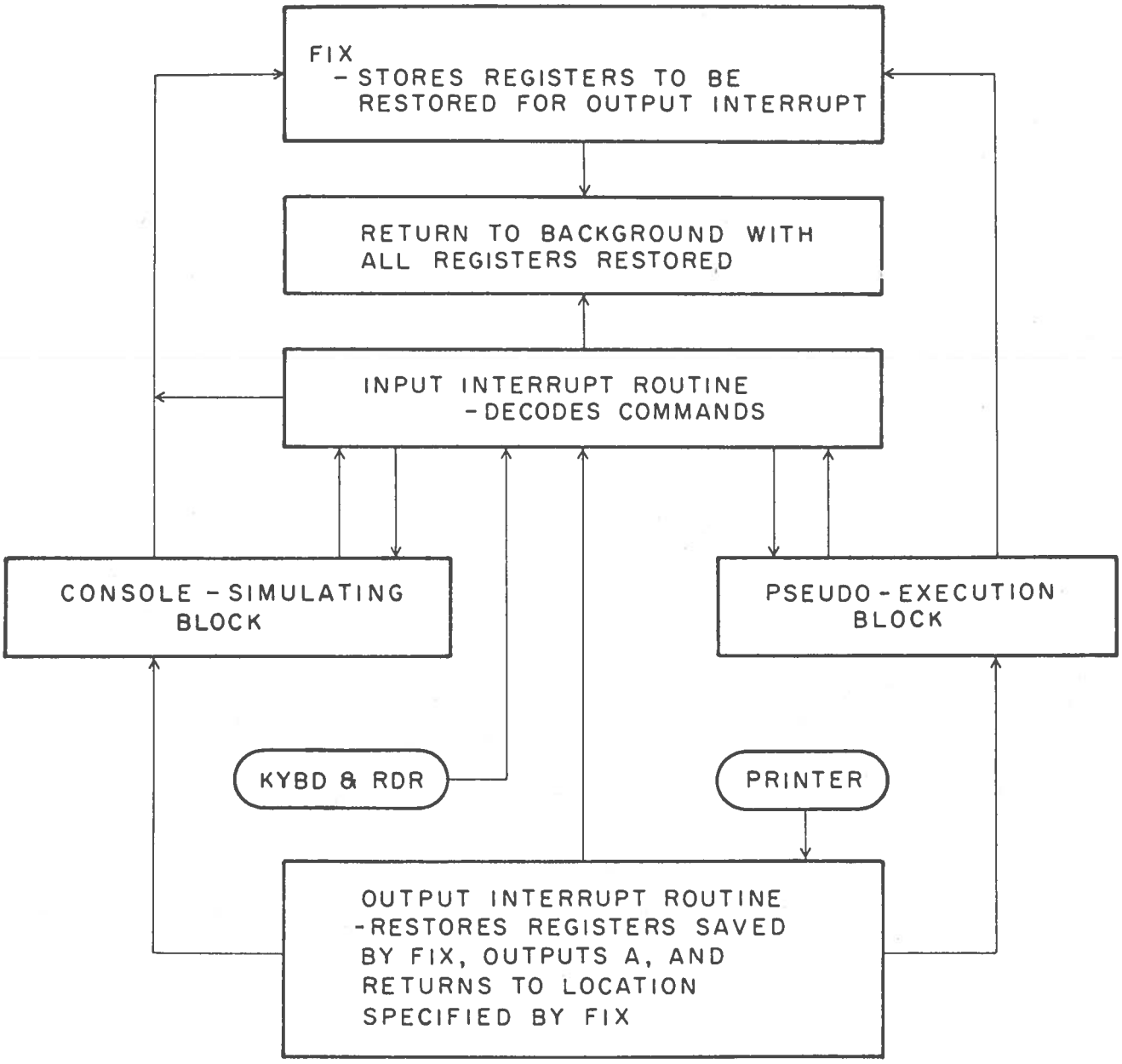


Figure 1(b)

In addition to the hardware console operations, the terminal typewriter can display all registers simultaneously and the operator can examine or change the contents of memory and the various registers during single-cycle execution. This feature is convenient for program debugging.

Certain functions of the console cannot be simulated but, fortunately, these are unnecessary for the remote typewriter 'console'. These functions, which affect the entire system, are program protect, memory parity, I/O hold release, interrupt release, and a master clear.

The command language[4] was designed to simplify debugging by using a set of short, easily-remembered command codes.

All commands are entered from the keyboard which operates on the input interrupt. The slash character (/) was chosen to indicate the beginning of all commands (except start). A mnemonic command is formed by the two characters that follow the slash. Each character keyed in (except data for the program being pseudo-executed) will be printed out before the next character is accepted. If an error is made it can be corrected by retyping the entire command.

The command is interpreted by making a linear search of a table of commands. A linear search is used because few instructions are required to program the search, the table is short, and because commands can be easily added or deleted from the set of legal commands.

The commands are divided into two classes – those that will be followed by a number and those that will not. A command is followed by a number if it is a command to load data into a register or if it refers to a memory location. All other commands do not require a number. If no number is to follow, control is transferred to the appropriate command routine. Otherwise, the execution of the command is delayed until the octal number has been keyed in. After the third command character has been read in, a blank is printed for ease in reading and to signal that the octal digits can be keyed in.

After the three-character command has been keyed in, all octal digits are right justified and the entire octal number is stored in memory. All other characters (except /, RUBOUT, and .) are ignored so that only the digits 0 to 7 will be interpreted as numbers. The period character (.) serves as the number terminator so that the number is right justified. Thus, errors may be corrected by keying in digits until the last five or eight are correct, depending on the size of the register being loaded.

The blank character ( ) is used in the single-cycle mode as the command to pseudo-execute the next instruction. The contents of the program counter and the instruction to be pseudo-executed are typed out and then the instruction is pseudo-executed. The contents of the A, B and index registers are then typed out.

---

[4] *See Appendix A*

The RUBOUT key is used as the start key and pseudo-execution will stop if a halt or illegal instruction is encountered, if the halt switches equal the program counter or if the / character is keyed in.

When each command has been executed, the computer advances the paper for the next command and to indicate that the previous command has been completed.

PROGRAM DETAILS – INTERRUPT ROUTINES

The remote terminal operates on the two standard I/O interrupts – with input having a higher priority than output. When a character is keyed in, operations which *must* be done (i.e., accepting the character and finding its meaning) are done at the high level of priority. Control is then dropped down to background level[5] so that interrupts of lower priority can be serviced. Everything that *can* be done is then done at the background level before control is returned to the background. (For instance, the pseudo-execution program is done at this level.)

When the input interrupt occurs, a check is made (Fig. 2) to see if an instruction was being pseudo-executed at the time that the interrupt occurred. If this was the case, an interrupt request flag is set and control is returned to the pseudo-execution program until pseudo-execution of that instruction is completed. The pseudo-execution flag is then reset and the input interrupt routine is entered.

A check (Fig. 3) is then made to see if the interrupt occurred in the terminal program. If a program other than the terminal program was interrupted, the address at which the interrupt occurred and all registers are saved. Otherwise, they are lost. The reasoning behind the check of that address is that if the operator interrupts his own program, he wishes to issue a new command and does not want his last command to be completed. (For instance, the operator may make an error and wish to continue before the error message has been completely typed out.) The loss of the address at which the interrupt occurred, when the interrupt occurred in the terminal program, ensures that control can always be returned to the background program.

The output interrupt routine (Fig. 39) is used to print messages, register contents, and any data output by the operator's program. The output interrupt remains disabled until there are data available to be typed. When data are available, the routine FIX is entered which stores the return address and registers before enabling the output interrupt. Control is then returned to the background program. When the output interrupt is raised, the address at which the interrupt occurred is checked, just as for the input interrupt, and the registers are either saved or lost. The registers that were saved for the terminal program are then restored, the contents of the A accumulator are printed, and the output interrupt is disabled before control is returned to the appropriate place in the terminal program.

---

[5] *It is assumed that the background program operates at the lowest priority.*

PROGRAM DETAILS – PSEUDO-EXECUTION (INTERPRETER) PROGRAM[6]

The pseudo-execution program must trap all illegal instructions and separate those instructions which can be executed by hardware from those which must be simulated. If an illegal instruction is encountered, an error message and the contents of the program counter are typed out. The PON, POF, PIE, PID, and PIR instructions, which deal with the entire system, and the TEU instruction are declared illegal as well as all EAU instructions which are not available on the computer being used.

Before pseudo-execution begins, the interrupt request flag is checked and, if set, control is returned to the input interrupt routine. Otherwise, an indicator is set to show that pseudo-execution is occurring. The instruction to be executed is stored in a software 'execution register' and the program counter is checked to see that it refers to a legal memory location. If the address is legal, pseudo-execution continues. Otherwise, an error message is typed.

The legal instructions have been classified into three groups:

(i) the type that does not refer to memory and can be executed by using an EXU instruction,

(ii) the type which refers to memory and must be checked to see if it refers to a legal memory address and can then be executed by using an EXU, and

(iii) the type whose execution must be simulated.

All instructions are classified and grouped in numerical order in Table I.

The instruction is decoded by taking the second and third octal digits of the instruction and comparing them (Fig. 8) with certain limits which are derived from Table I. By this method, the instruction is placed into one of the groups given in Table I or is declared illegal. Control is then transferred to the correct routine.

When an instruction refers to memory, the effective address is checked to see if it is legal (Fig. 11). The effective address is the address after indexing and indirect chaining has been taken into account. If the effective address is illegal or if an indirect chain exceeds five levels, an error message is printed.

Branch instructions (Fig. 14 and 15) must be simulated so that the terminal program does not lose control. The address specified by the branch instruction is checked (Fig. 11) to see that it is between the allowable limits in core. In the case of the conditional branches (except BOF), the actual instruction is executed by hardware to make the decision of whether or not to branch. For the BOF instruction (Fig. 15) a software overflow indicator is checked and reset to make the decision.

---

[6] *The instruction set for this program is described in Appendix E.*

TABLE I

Classification of instructions in numerical order

Augmented			Regular					
00-00	HLT		01	LAA		22	BAZ	
-01	TAI		02	LBA		23	BAN	
-02	TBI		03	STA		24	BAP	
-03	CLA	*	04	STB		25	BOF	
-04	TBA		05	AMA	**	26	MEA	
-05	TAB		06	SMA		27	MAA	
-06	IAB		07	MPY				**
-07	CSB					30	MOA	
			10	DIV		31	AAM	
00-10	RSA		11	BRU		32	LIX	
-11	LSA		12	SPB		33	STI	
-12	FRA		13 (0)	CEU		34	IIB	
-13	FLA		13 (2)	TEU	***	35	SMP	**
-14	FRL		13 (4)	SNS		36	PIR	***
-15	RSL	*	14	IMS				
-16	LSL		15	CMA	*	(0) 43	PID	
-17	FLL		16	EXU		(1) 43	PIE	***
			17 (0)	AOP		44	IAM	**
00-20	ASC		17 (2)	AIP		56	NEG	*
-21	SAS		17 (4)	MOP		57	LCS	
-22	NOP		17 (6)	MIP		60	RNA	*
						61	AMX	
00-32	NOR	*	20	CNS	*	62	PON	***
			21	EAU's	***	63	POF	***

NOTATION:

*	pseudo-executed by an EXU
**	pseudo-executed by checking memory and using EXU
***	illegal instructions
	all unmarked instructions must be simulated
}	instructions which require the same routine

The input/output instructions are the most difficult to simulate because the command device and the I/O device are the same. Each I/O instruction is first checked to see that it refers to the teletype.

The CEU instruction (Fig. 21) is not actually executed until input is required from paper tape because the keyboard and reader cannot both operate at the same time. No CEU command is executed for output because the keyboard is always enabled when output can occur. The above details ensure that the operator can always stop the pseudo-execution.

The CEU data word is checked to see that the interrupt will not be disconnected or that the mode will not be cleared. If the data are illegal, 'ILLEGAL CEU DATA' is printed. Otherwise, the data are saved so that the mode (keyboard or reader) can be checked when input is requested (AIP or MIP instructions).

The simulation of the AIP and MIP instructions (Fig. 27) for input from paper tape uses a twenty-word buffer in memory rather than attempting to switch the input mode between reader and keyboard so that the operator would always be able to issue a command. Switching modes was attempted but the program could not distinguish whether the character came from the reader or keyboard. This was blamed (correctly or incorrectly) on the fact that the two modes of input share a common buffer. If input is requested from paper tape and the buffer is empty, SPCE is typed and, when the operator depresses the space bar, the reader will be enabled to fill the buffer. When the buffer is filled, the keyboard is enabled to accept more commands.

If the AIP or MIP instruction refers to the keyboard, the instruction is simulated by an in-line AIP. Each character read from the keyboard is checked to see if it is a / . If a / is found, a message is typed out stating that the operator must type OK if the / was meant to be data. Otherwise, a command is expected.

For the AOP and MOP instructions (Fig. 28), the A accumulator is loaded with the contents of the appropriate memory address, the pseudo-execution flag is reset, the output interrupt is enabled, and control is returned to the background program to wait for the interrupt. When that interrupt occurs, the pseudo-execution flag is set and the contents of the A accumulator are typed out.

The SNS instruction (Fig. 24) is simulated for only the first four switches. If the instruction does not refer to a switch in group zero, an error message is printed. Otherwise, the program counter is advanced by one if the specified software sense switch is set and by two if it is not.

The LCS instruction (Fig. 25) is simulated by loading the software A accumulator using the software control switches. These switches are loaded by means of the /LC command (see Appendix A).

The index instructions, namely, LIX, AMX, STI (Fig. 17), and IIB (Fig. 29) must also be simulated. The effective memory address is checked for all of the instructions and it should be noted that index modification of an indirect chain is different from the method used with other instructions. Once memory has been checked, the AMX, STI, and LIX instructions are executed by an EXU. The IIB instruction is simulated by having the program increment the index register. If the index is not equal to zero, and if the memory address is legal, the branch will occur.

The halt instruction, HLT (Fig. 17), is not executed so that the computer will not stop. When the HLT instruction is encountered, the program counter is advanced, the pseudo-execution flag is reset, and HALT AT is typed out, followed by the contents of the program counter.

In the CSB routine (Fig. 10), the B accumulator is loaded, the CSB indicator is set, and a CSB instruction is executed. The old B accumulator is saved so that it may be used again. If the next instruction is one which is normally executed by using an EXU, it is put in-line following a CSB instruction and executed (Fig. 33). The B accumulator that was saved is used in this case. This procedure is used so that the carry flip-flop will be set when the next instruction is executed and so that the sign of the B accumulator can be seen to change during single-cycle execution. If the instruction following the CSB is not normally executed by using an EXU, the CSB indicator is ignored because the carry flip-flop has no effect on these instructions.

The EXU instruction (Fig. 25) is simulated by resetting the CSB indicator, checking memory, and by loading the contents of the effective address into the software instruction register. This instruction is then decoded as if there had been no EXU.

All other instructions can be executed by hardware and they fall into two classes – those that require a memory check and those that do not (Fig. 16). After the memory check is made, both types of instructions are treated in the same way.

An indicator is then reset to show that the instruction is not one whose execution is simulated. The hardware registers are loaded from the software registers (Fig. 30), the hardware overflow indicator is reset, and the CSB indicator is checked. If the preceding instruction was not CSB, the instruction is executed by an EXU. Many of the simulation routines return control to this point where the program counter is advanced by 1, 2, or 3 as required. If the instruction was executed by an EXU, the registers are stored and the hardware overflow indicator is checked. If that indicator was set, the software overflow indicator is set (Fig. 33). All of the remaining simulated instructions (except EXU and CSB) return control to this point where the CSB indicator is reset. Control would be returned from the CSB routine to the next instruction where the pseudo-execution indicator is reset.

If single-cycle execution was being done, control would be returned to the single-cycle routine. Otherwise, the software halt switches are tested and, if they are equal to the program counter, CONTROL HALT AT and the contents of the program counter are printed out and control is returned to the background program. Otherwise, the next instruction is pseudo-executed.

CONCLUSION AND REMARKS

The program described above completes the first stage of the development of the system of remote teletypes. It lacks several features of the final system but these should be added in stage two.

It may be desirable to have certain functions such as relocatable loading available at the remote terminal. These can easily be added by expanding the command language by inserting the new command and the address of the routine in the correct tables.

The final system will use typewriters which are not identical to the console typewriter and also interrupts other than the standard I/O interrupts. Since the program described

uses the console typewriter and the standard interrupts, the interrupt routines and, possibly, the routines for the pseudo-execution of I/O instructions will have to be rewritten.

The command and pseudo-execution routines have been written for only one terminal but they should be easily adaptable to a multi-terminal system. It was thought that an index register would be helpful to specify a certain terminal, and index register three was never used in order to make it available for this purpose. The extra core required for adding more terminals should be considerably less than that used by the first stage program.

Most of the CPU time used by the remote terminal program is for testing memory addresses and instructions and for pseudo-execution. If the instruction trap, memory protect, and stall alarm options were added to the computer, it would be expected that the program could be written to use less memory and CPU time by executing the operator's program directly rather than by pseudo-execution.

The SEL memory protect option uses one bit stored with each word to indicate whether or not the word is protected. If the computer is operating in the protected mode, an interrupt will occur if an unprotected instruction attempts to branch to or alter a protected word. This means that the background program could be protected from the terminal operator's program; but, if another terminal was added, the two terminals could not be protected from each other. A more suitable memory protection design for a multi-terminal system would be one which has boundaries in memory and which raises an interrupt if a program attempts to access a word outside of its boundary.

Let us assume that memory is suitably protected. The SEL instruction trap raises an interrupt for the instructions PON, POF, PIE, PID, CEU, AOP, AIP, MIP, HLT, PIR, and all illegal instructions. This must be modified to include LCS and SNS because the sense and control switches are simulated. When the interrupt is raised, these instructions will have to be pseudo-executed.

The stall alarm will generate an override interrupt if the program counter has not advanced for 32 machine cycles. This override interrupt can interrupt an I/O instruction, an indirect chain or a halt condition. Thus, the stall alarm will protect against endless indirect chains.

If the program was written to use these options, a memory saving of 200_8 (about 10%) would probably result. The largest gain, however, would be in the use of less CPU time for execution of the terminal operator's program, since, on the average, it requires thirty or more instructions to pseudo-execute each instruction.

If a one-terminal time-shared system is required, this first stage program is adequate. The only feature that it lacks that may be defined as 'necessary' is a relocatable load option.

ACKNOWLEDGMENT

I would like to thank Mr. J. Wolfe, Dr. M. Wein, Mr. J.K. Pulfer, and Mr. F.V. Cairns of the Radio and Electrical Engineering Division for their valuable suggestions.

REFERENCE

1. SEL 840A general purpose computer reference manual. Systems Engineering Laboratories, Inc., Fort Lauderdale, Florida. 1967

APPENDIX A

COMMANDS – EXPLANATION

LOAD COMMANDS

When loading an octal number, the digits are right justified so that leading zeros do not have to be typed. If an error is made, it can be corrected by typing until the last five or eight digits (depending on the size of the register being loaded) are correct. A . denotes the termination of the number. All digits represented here are octal.

The load commands are:

/LA	–	load A accumulator (Fig. 50)
e.g.		/LA xxxxxxxx.*
/LB	–	load B accumulator (Fig. 50)
e.g.		/LB xxxxxxxx.
/LP	–	load program counter (Fig. 50)
e.g.		/LP xxxxx.
/LC	–	load control switches (Fig. 50)
e.g.		/LC xxxxxxxx.
/LM	–	load memory (Fig. 52)
e.g.		/LM aaaaa.xxxxxxxx. aaaaa+1.ƀƀƀ (ƀ = blank)
	–	xxxxxxxx is loaded into memory location aaaaa.
	–	the computer types aaaaa+1 to allow loading in this location by keying in the next octal number.
/Ll	–	load Index 1 (Fig. 51)
e.g.		/Ll xxxxx.
		similarly, /L2, L3

DISPLAY COMMANDS

The number terminator . is required only after the instruction to display memory. For all other commands in this group, the contents of the specified register are output – following the third character of the command.

---

*A blank is printed after the third character of the code has been read in. The command has been executed when *carriage return*, and *line feed* advance the paper.

These commands are:

/DA – display A accumulator (Fig. 44)

e.g. /DA xxxxxxxx

/DB – display B accumulator (Fig. 44)

e.g. /DB xxxxxxxx

/DC – display control switches (Fig. 44)

e.g. /DC xxxxxxxx

/DP – display program counter (Fig. 45)

e.g. /DP xxxxx

/DM – display specified memory location (Fig. 46)

e.g. /DM aaaaa. xxxxxxxx

– where aaaaa is the address to be displayed

– type another . and aaaaa+1 will be printed, followed by its contents

/DW – display program counter, A, B, X1, X2, X3 (Fig. 48)

e.g. /DW ppppp AAAAAAAA BBBBBBBB 11111 22222 33333

where: ppppp – contents of program counter
AAAAAAAA – contents of A accumulator
BBBBBBBB – contents of B accumulator
11111 – contents of Index 1
22222 – contents of Index 2
33333 – contents of Index 3

/D1 – display Index 1 (Fig. 45)

e.g. /D1 xxxxx

Similarly, /D2, /D3.

SENSE/HALT SWITCH COMMANDS

All of these software switches act exactly as the hardware switches as far as the instructions are concerned.

/SD – sense switch display (Fig. 49)

e.g. /SD ƀƀƀƀn_0 ƀƀƀn_1 ƀƀƀn_2 ƀƀƀn_3

where $n_i = \begin{cases} 0 \text{ if } ss_i \text{ is off} \\ 1 \text{ if } ss_i \text{ is on} \end{cases}$ $(0 \leqslant i \leqslant 3)$

/SS – set sense mode (Fig. 47)

– this command puts the sense/halt switch in the sense position and control switches 0 – 3 become sense switches as well.

/SR – sense/halt reset (Fig. 47)

– this command puts the sense/halt switch in the centre position.

/HS – halt set (Fig. 47)

e.g. /HS xxxxx.

– the address portion of the control switches is replaced by xxxxx and the other switches are unchanged.

– the sense/halt switch is put into the halt position.

– the program being pseudo-executed will halt when xxxxx appears in the program counter and CONTROL HALT AT xxxxx will be typed.

SINGLE-CYCLE EXECUTION COMMAND

/SW – single-cycle execute and display all registers (Fig. 43)

– for each execution, type in ƀ (blank)

e.g. ƀƀƀƀ ppppp xxxxxxxx AAAAAAAA BBBBBBBB 11111 22222 33333 command to single-cycle execute

where: xxxxxxxx represents the instruction to be executed (contained in ppppp)

See /DW for definition of other symbols

MISCELLANEOUS COMMANDS

/CL – master clear (Fig. 42)

– this command clears the A, B, X1, X2, X3 registers and control switches.

– it also resets the sense/halt switch and resets the software overflow indicator

– clears the memory buffer used for simulating AIP or MIP from tape.

/ – halt

– this command will stop the execution of the last command

– the / re-initializes the program to accept the next two command characters.

RUBOUT – is the start command and START AT ppppp is printed out

– execution starts and will continue until a halt or illegal instruction is found, the program counter = the halt switches, or a / is typed in.

/MP – memory map (Fig. 53)

e.g. /MP LB = LLLLL UB = UUUUU

where: LLLLL – lowest allowable memory location
UUUUU – highest allowable memory location

/OK – this command is to be used only to indicate that when a / has been read in while pseudo-executing an AIP or MIP it was meant to be data.

– this command causes execution to continue with the / being interpreted as data.

/BC – buffer clear

– clears the memory used for simulating AIP or MIP from tape.

APPENDIX B – FLOW CHARTS

NOTES

– REPRESENTS SUBROUTING OR EXTERNAL PROCESS.

– INDICATES INTERRUPT ROUTINE.

– INDICATES INPUT OR OUTPUT.

– OTHER BOXES ARE STANDARD (DECISION AND ASSIGNMENT).

A ⟵ B MEANS: THE CONTENTS OF 'A' ARE REPLACED BY THE CONTENTS OF 'B'.

A ⟵ (MEM LOC SPECIFIED BY EADD)

MEANS: THE CONTENTS OF 'A' ARE REPLACED BY THE CONTENTS OF THE MEMORY LOCATION SPECIFIED BY EADD.

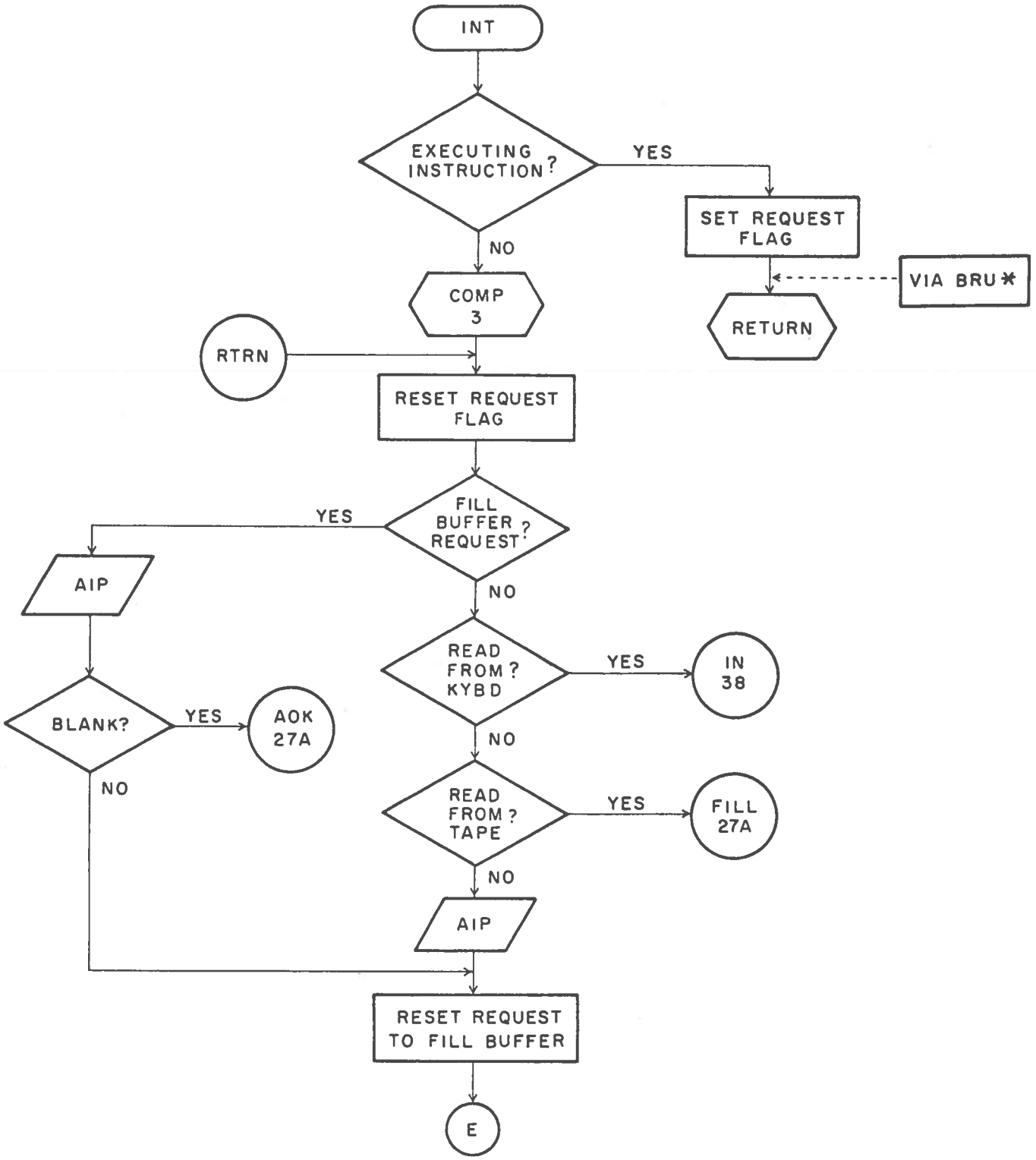

INT
EXECUTING INSTRUCTION?
YES
SET REQUEST FLAG
VIA BRU*
RETURN
NO
COMP 3
RTRN
RESET REQUEST FLAG
FILL BUFFER REQUEST?
YES
AIP
NO
READ FROM? KYBD
YES
IN 38
BLANK?
YES
AOK 27A
NO
NO
READ FROM? TAPE
YES
FILL 27A
NO
AIP
RESET REQUEST TO FILL BUFFER
E

Figure 2(a)

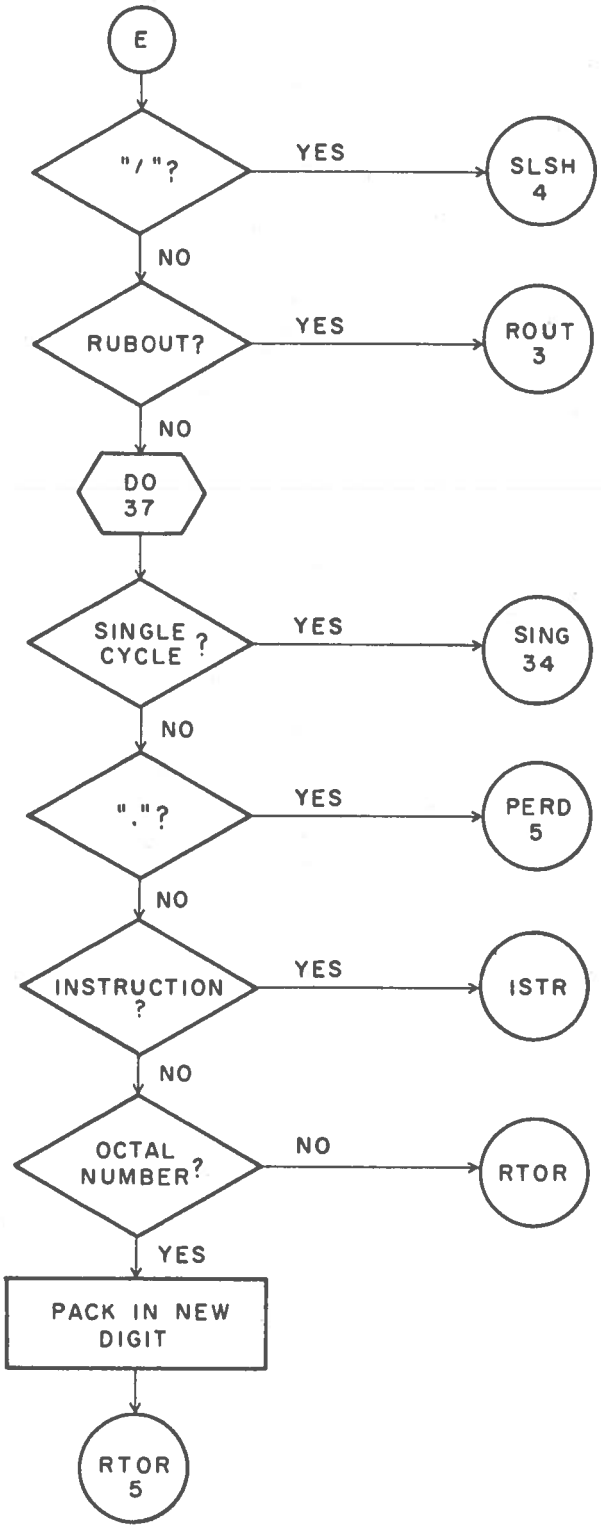

E
"/"?
YES
SLSH 4
NO
RUBOUT?
YES
ROUT 3
NO
DO 37
SINGLE CYCLE ?
YES
SING 34
NO
"."?
YES
PERD 5
NO
INSTRUCTION ?
YES
ISTR
NO
OCTAL NUMBER?
NO
RTOR
YES
PACK IN NEW DIGIT
RTOR 5

Figure 2(b)

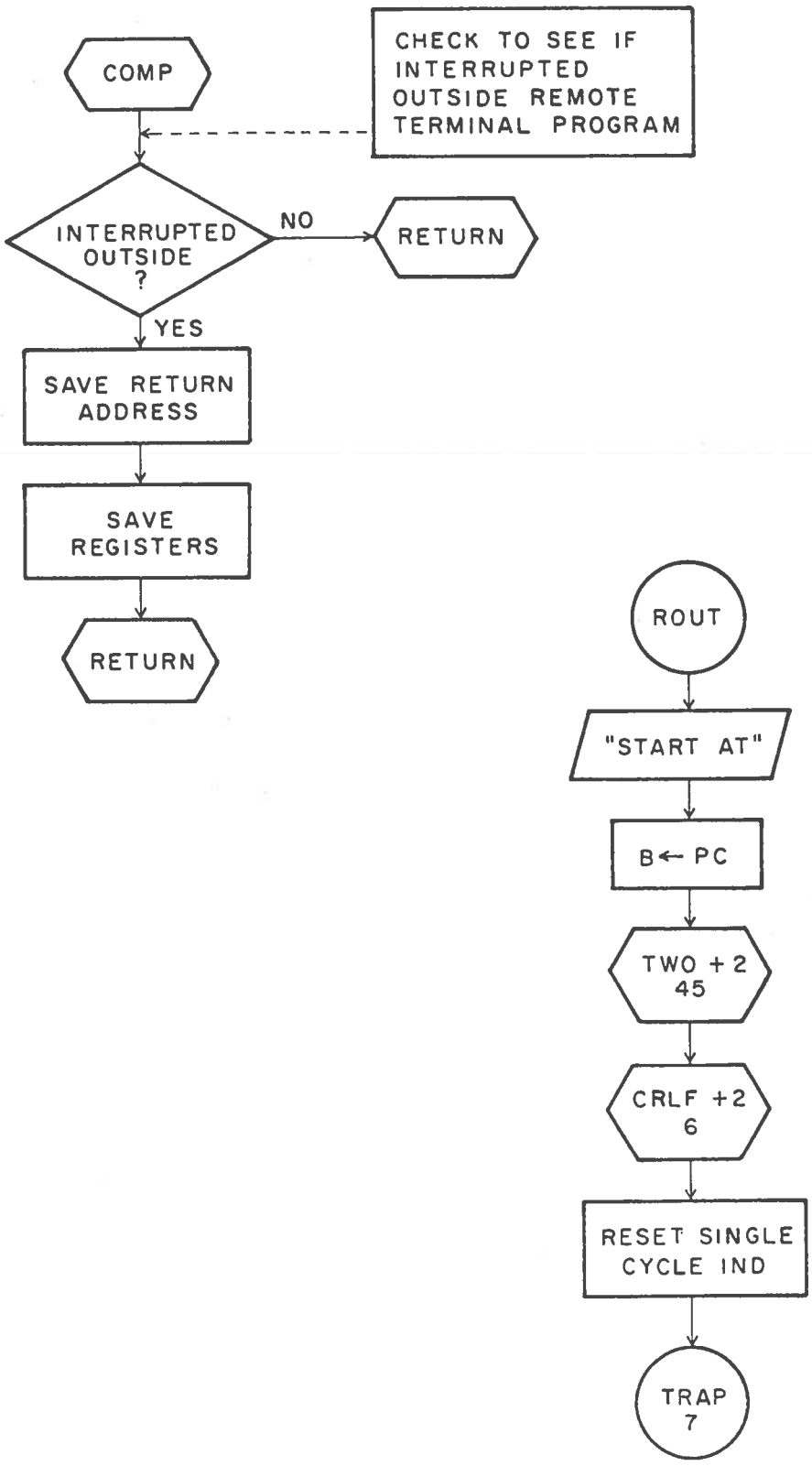

COMP
CHECK TO SEE IF INTERRUPTED OUTSIDE REMOTE TERMINAL PROGRAM
INTERRUPTED OUTSIDE ?
NO
RETURN
YES
SAVE RETURN ADDRESS
SAVE REGISTERS
RETURN
ROUT
"START AT"
B← PC
TWO +2 45
CRLF +2 6
RESET SINGLE CYCLE IND
TRAP 7

Figure 3

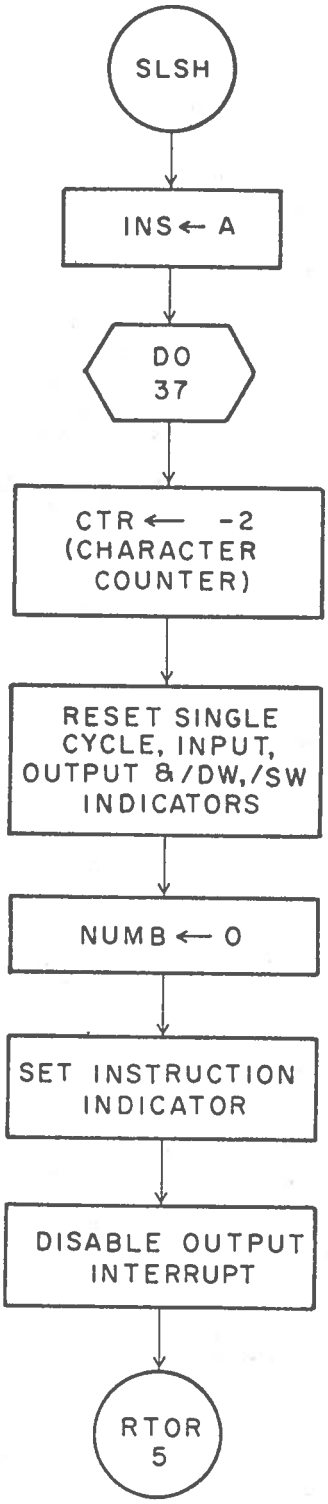

SLSH
INS ← A
DO
37
CTR ← -2
(CHARACTER
COUNTER)
RESET SINGLE
CYCLE, INPUT,
OUTPUT &/DW,/SW
INDICATORS
NUMB ← 0
SET INSTRUCTION
INDICATOR
DISABLE OUTPUT
INTERRUPT
RTOR
5

Figure 4

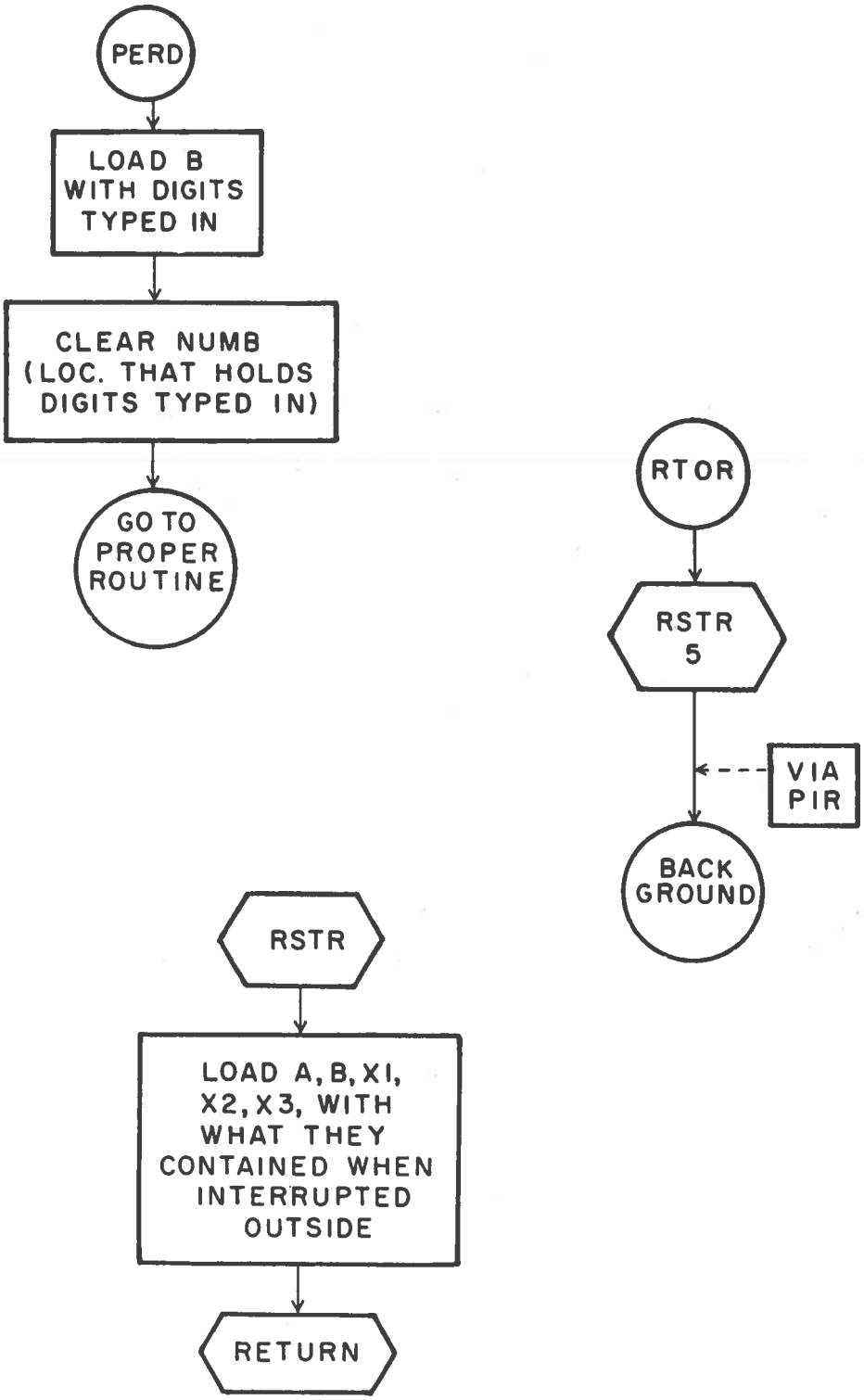

PERD
LOAD B WITH DIGITS TYPED IN
CLEAR NUMB (LOC. THAT HOLDS DIGITS TYPED IN)
GO TO PROPER ROUTINE
RTOR
RSTR 5
VIA PIR
BACK GROUND
RSTR
LOAD A, B, X1, X2, X3, WITH WHAT THEY CONTAINED WHEN INTERRUPTED OUTSIDE
RETURN

Figure 5

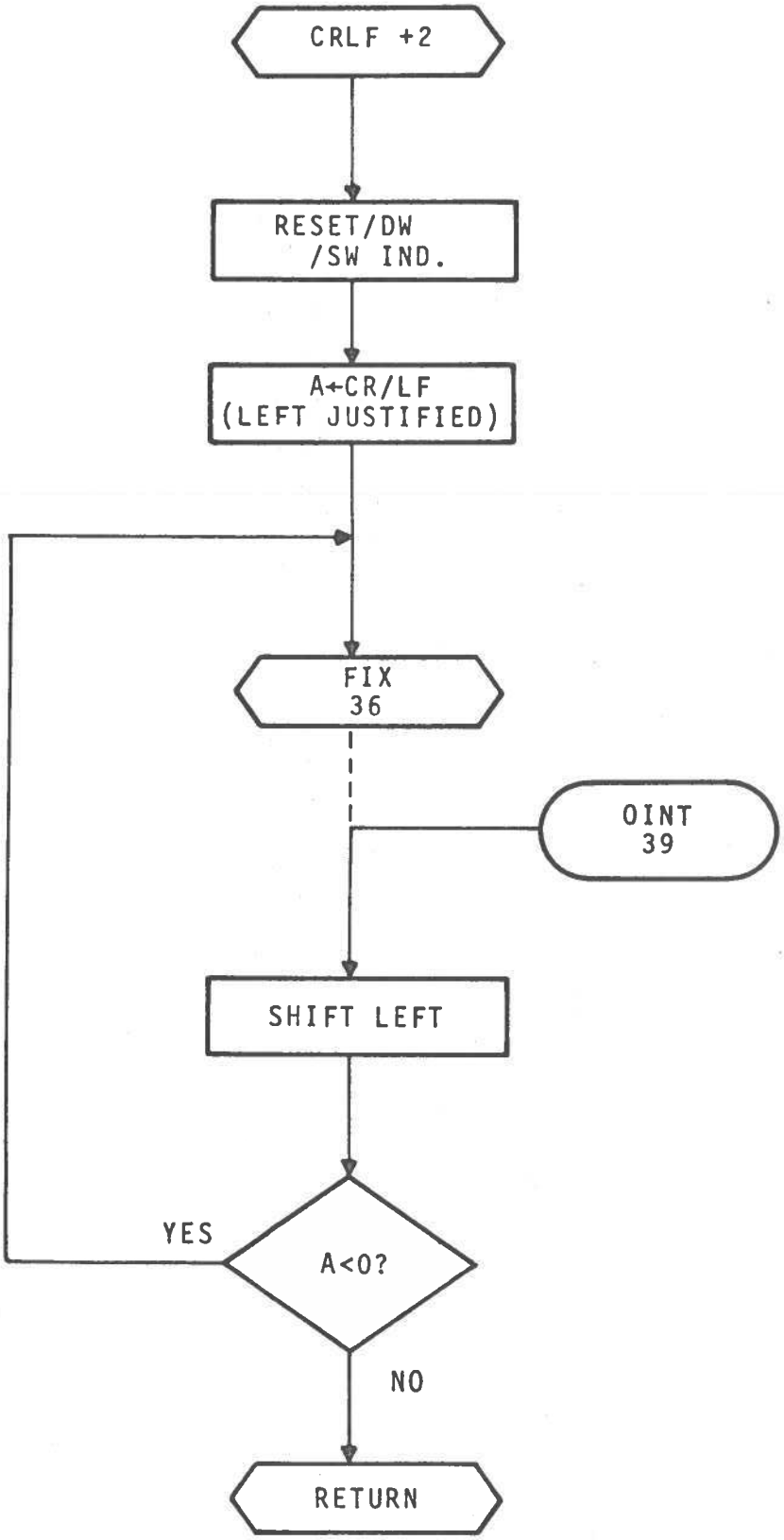

CRLF +2
RESET/DW
/SW IND.
A←CR/LF
(LEFT JUSTIFIED)
FIX
36
OINT
39
SHIFT LEFT
YES
A<0?
NO
RETURN

Figure 6

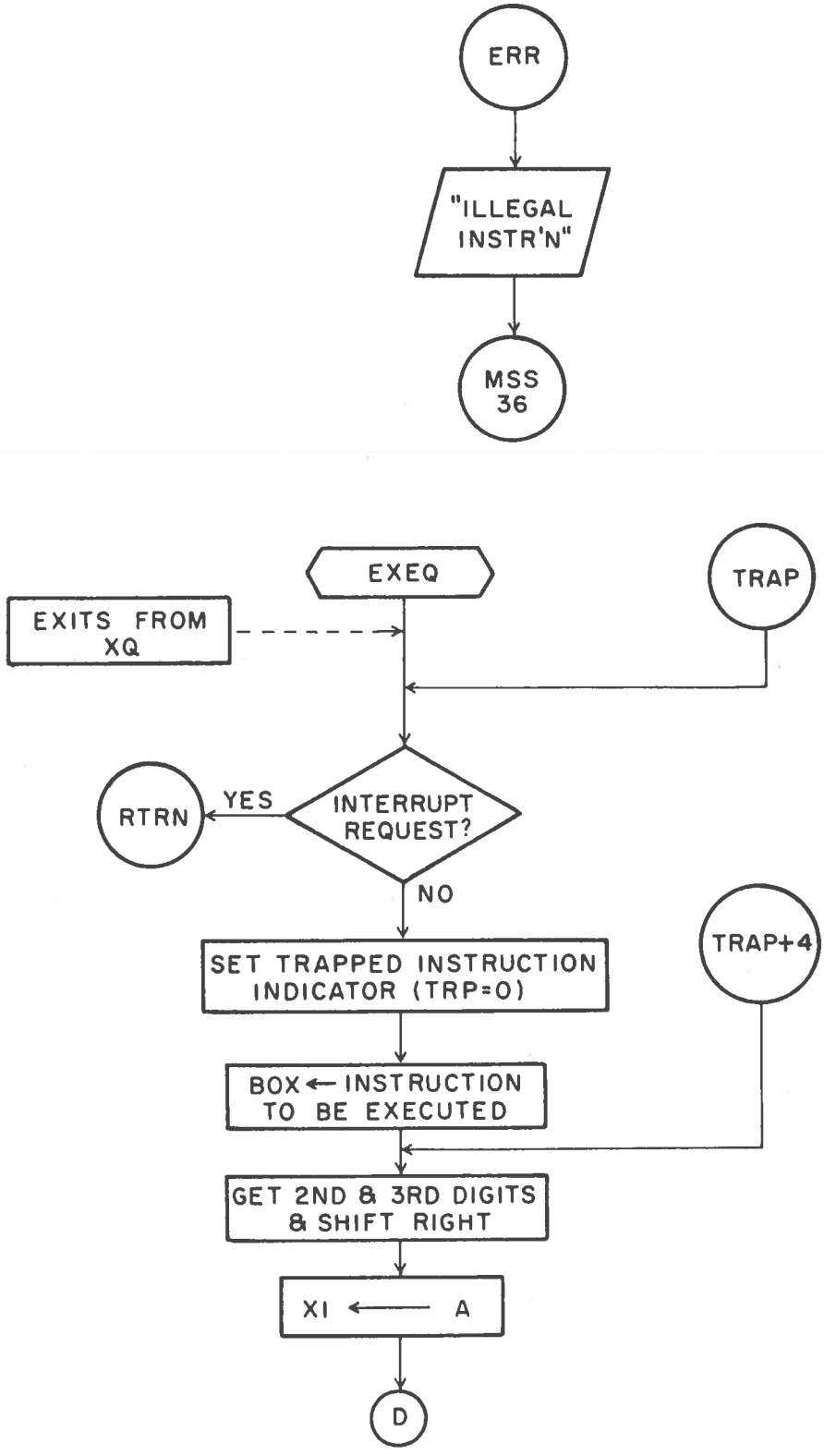

ERR
"ILLEGAL INSTR'N"
MSS 36
EXEQ
TRAP
EXITS FROM XQ
RTRN
YES
INTERRUPT REQUEST?
NO
TRAP+4
SET TRAPPED INSTRUCTION INDICATOR (TRP=0)
BOX ← INSTRUCTION TO BE EXECUTED
GET 2ND & 3RD DIGITS & SHIFT RIGHT
XI ← A
D

Figure 7

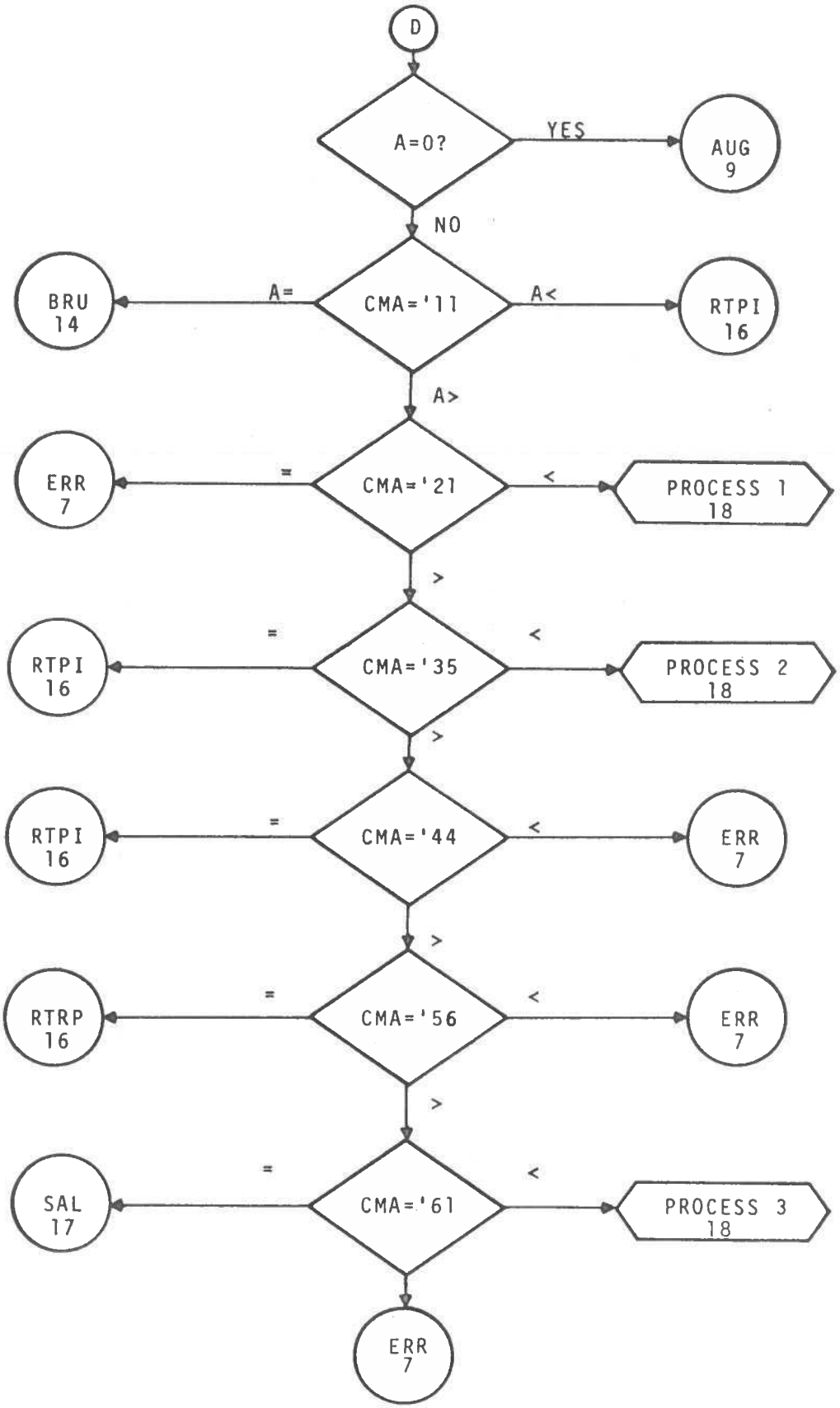

D
A=0?
YES
AUG
9
NO
BRU
14
A=
CMA='11
A<
RTPI
16
A>
ERR
7
=
CMA='21
<
PROCESS 1
18
>
RTPI
16
=
CMA='35
<
PROCESS 2
18
>
RTPI
16
=
CMA='44
<
ERR
7
>
RTRP
16
=
CMA='56
<
ERR
7
>
SAL
17
=
CMA='61
<
PROCESS 3
18
ERR
7

Figure 8

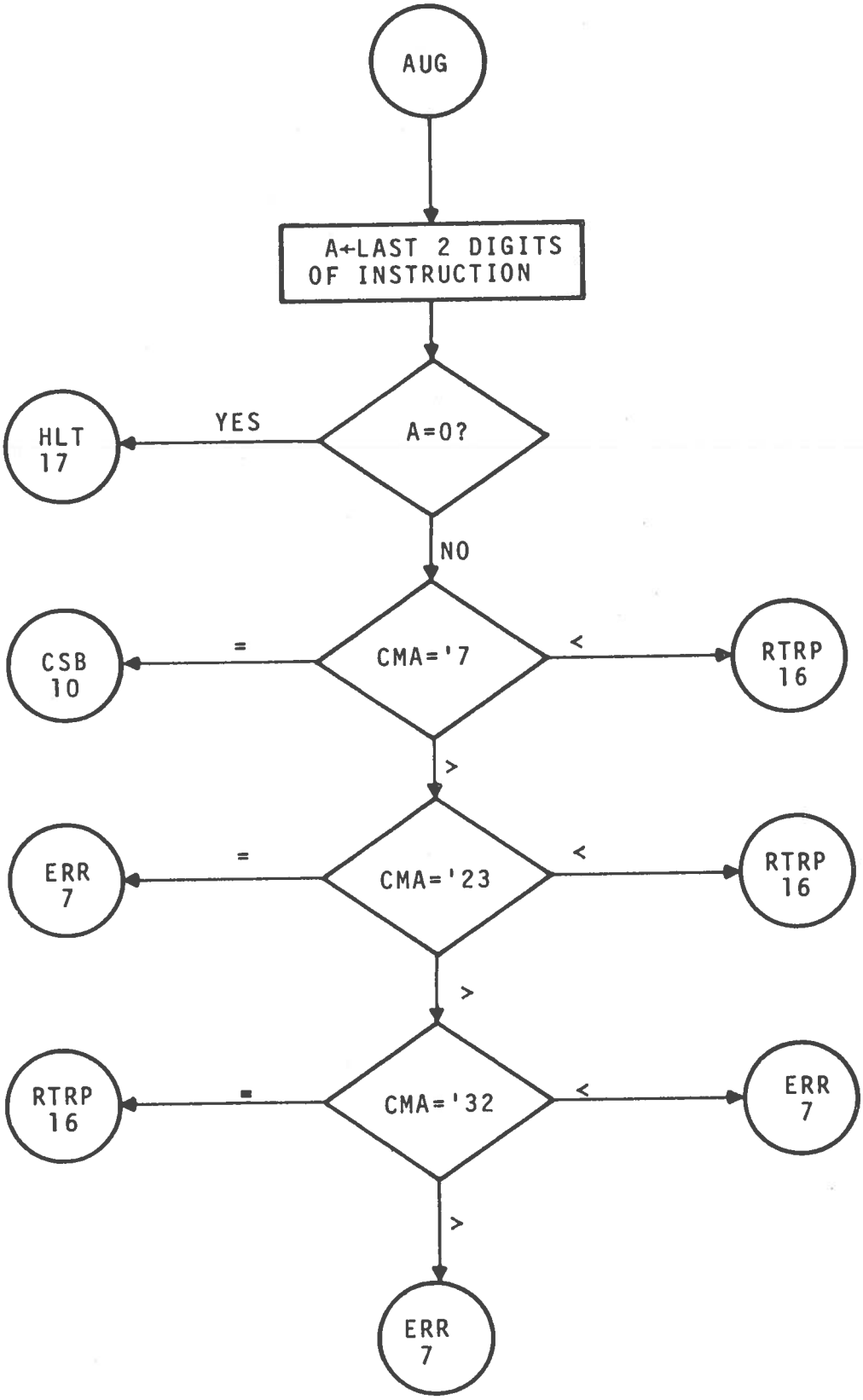

AUG
A←LAST 2 DIGITS OF INSTRUCTION
A=0?
YES
HLT 17
NO
CMA='7
=
CSB 10
<
RTRP 16
>
CMA='23
=
ERR 7
<
RTRP 16
>
CMA='32
=
RTRP 16
<
ERR 7
>
ERR 7

Figure 9

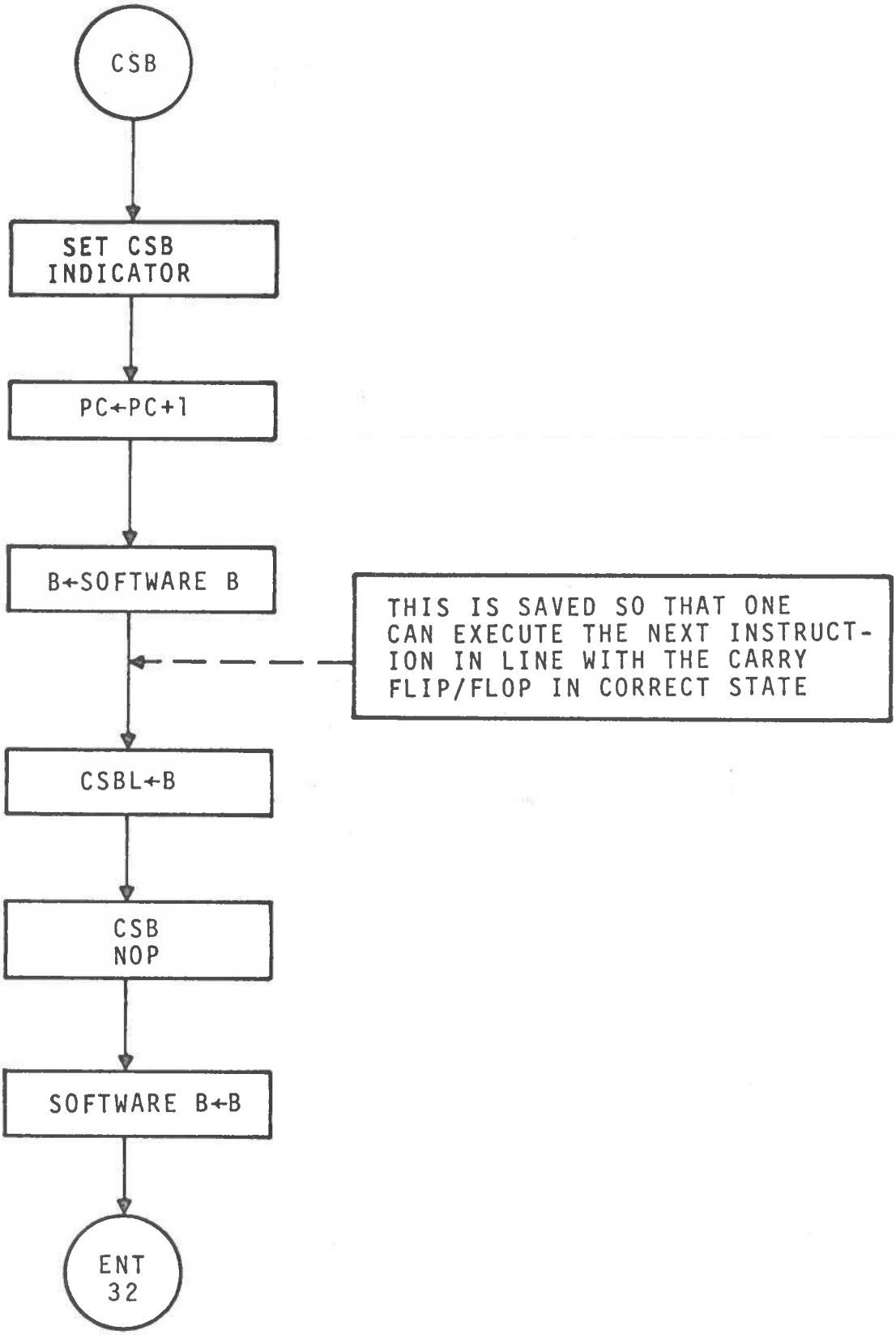

CSB
SET CSB INDICATOR
PC←PC+1
B←SOFTWARE B
THIS IS SAVED SO THAT ONE CAN EXECUTE THE NEXT INSTRUCT-ION IN LINE WITH THE CARRY FLIP/FLOP IN CORRECT STATE
CSBL←B
CSB NOP
SOFTWARE B←B
ENT 32

Figure 10

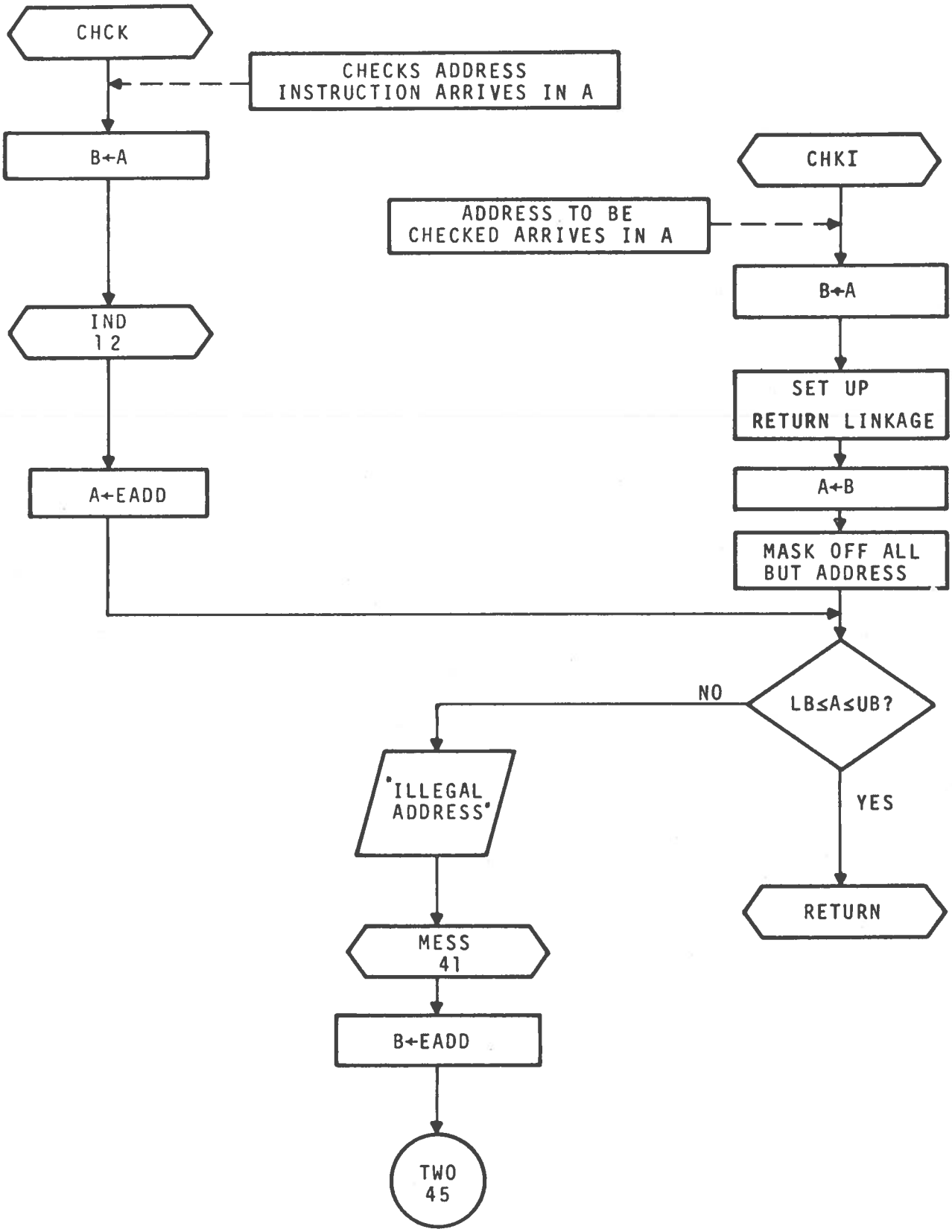

CHCK
CHECKS ADDRESS INSTRUCTION ARRIVES IN A
B←A
IND 1 2
A←EADD
CHKI
ADDRESS TO BE CHECKED ARRIVES IN A
B←A
SET UP RETURN LINKAGE
A←B
MASK OFF ALL BUT ADDRESS
LB≤A≤UB?
NO
YES
'ILLEGAL ADDRESS'
MESS 41
B←EADD
TWO 45
RETURN

Figure 11

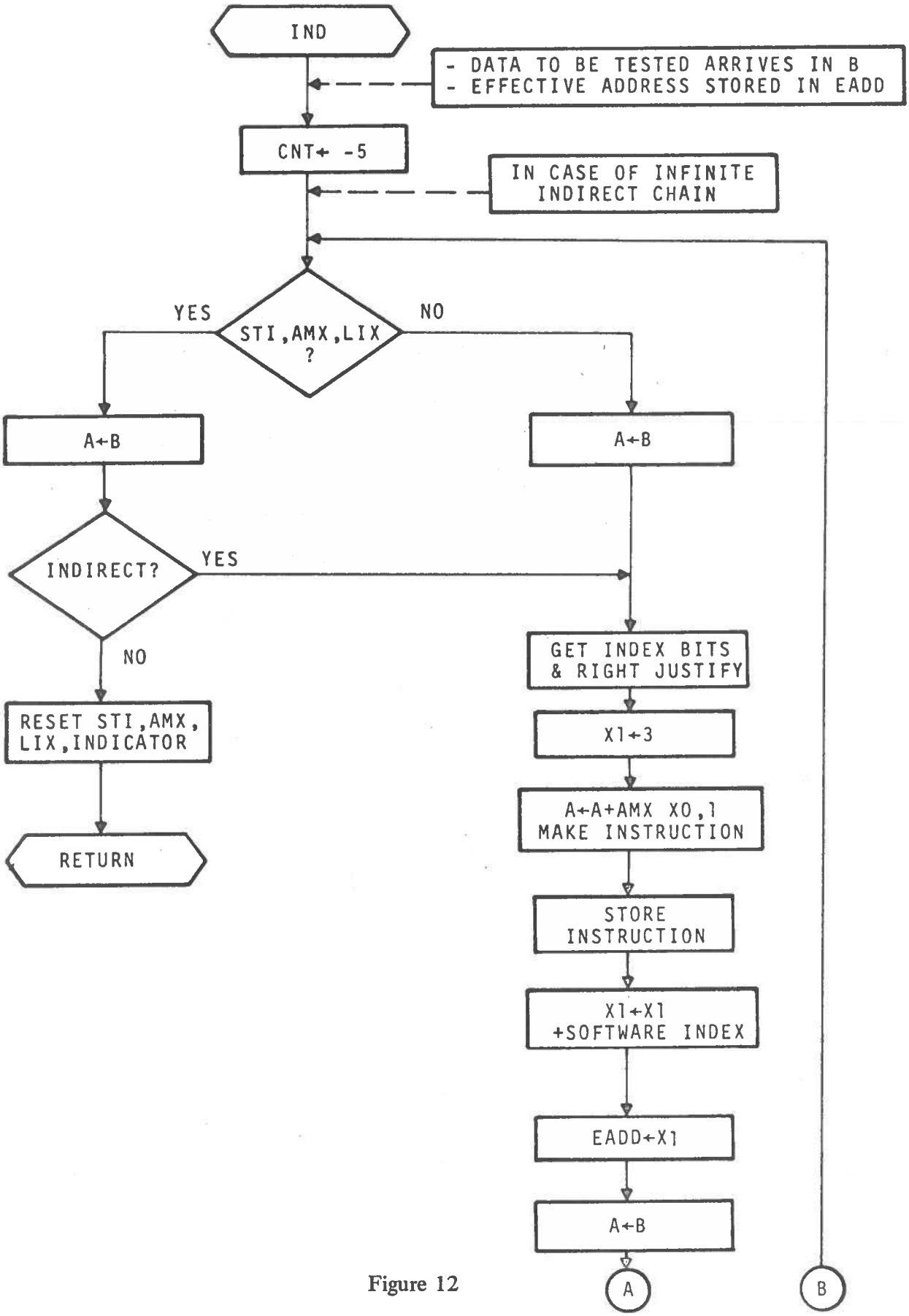

IND
- DATA TO BE TESTED ARRIVES IN B
- EFFECTIVE ADDRESS STORED IN EADD
CNT← -5
IN CASE OF INFINITE INDIRECT CHAIN
YES
STI,AMX,LIX ?
NO
A←B
A←B
INDIRECT?
YES
NO
RESET STI,AMX, LIX,INDICATOR
RETURN
GET INDEX BITS & RIGHT JUSTIFY
X1←3
A←A+AMX X0,1 MAKE INSTRUCTION
STORE INSTRUCTION
X1←X1 +SOFTWARE INDEX
EADD←X1
A←B
A
B

Figure 12

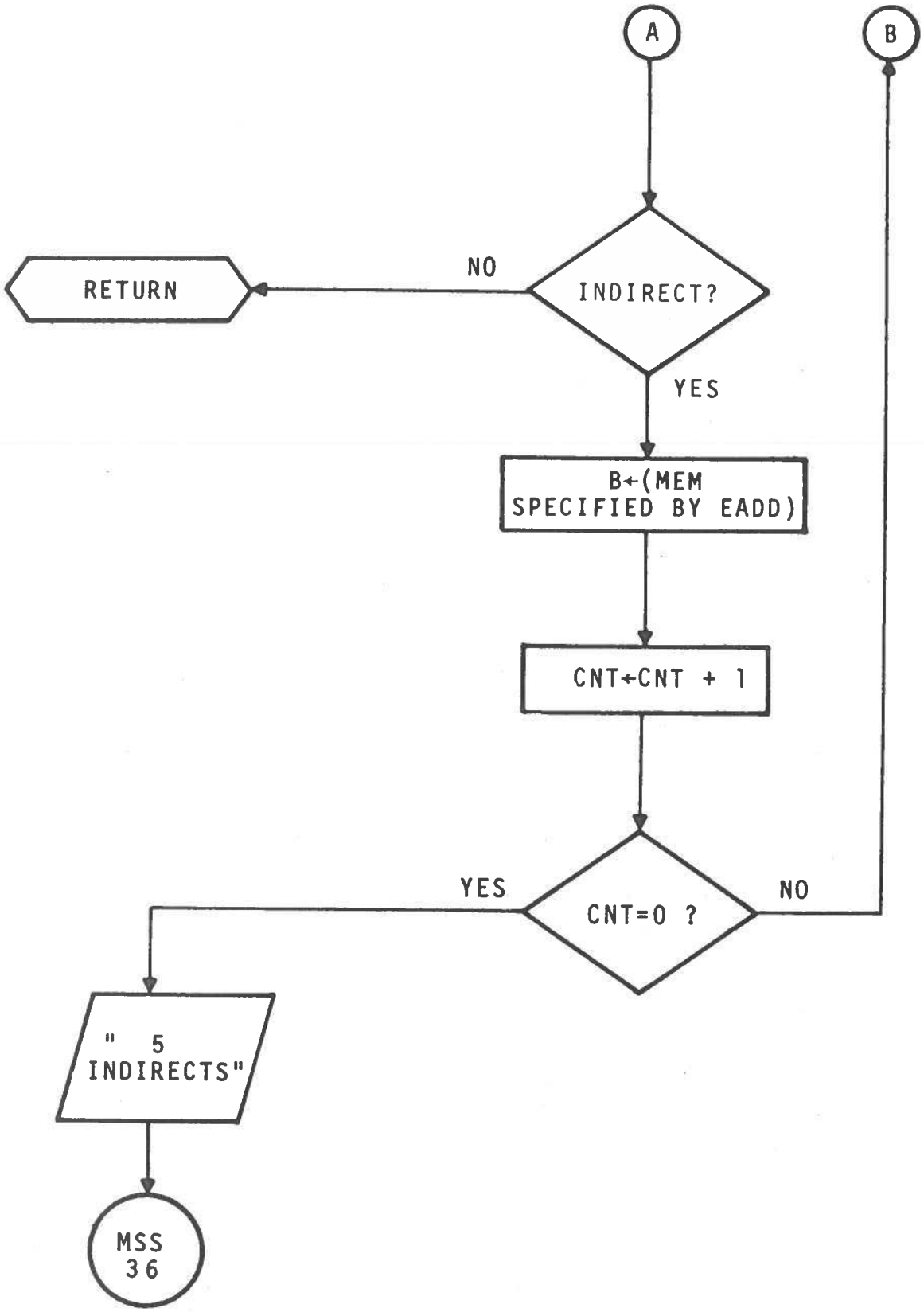

A
B
INDIRECT?
NO
RETURN
YES
B←(MEM
SPECIFIED BY EADD)
CNT←CNT + 1
CNT=0 ?
YES
NO
" 5
INDIRECTS"
MSS
36

Figure 13

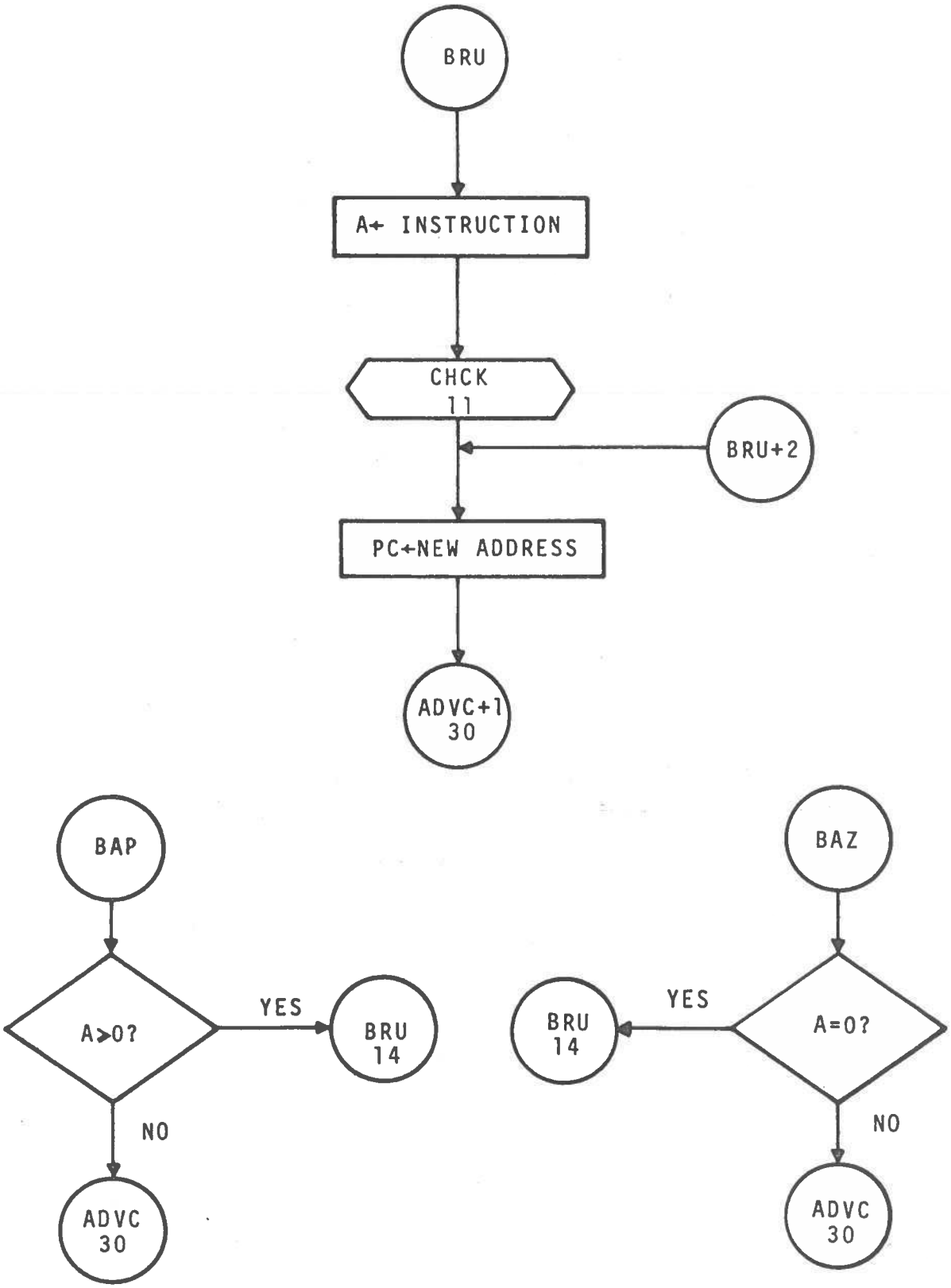

BRU
A← INSTRUCTION
CHCK
11
BRU+2
PC←NEW ADDRESS
ADVC+1
30
BAP
A>0?
YES
BRU
14
NO
ADVC
30
BAZ
A=0?
YES
BRU
14
NO
ADVC
30

Figure 14

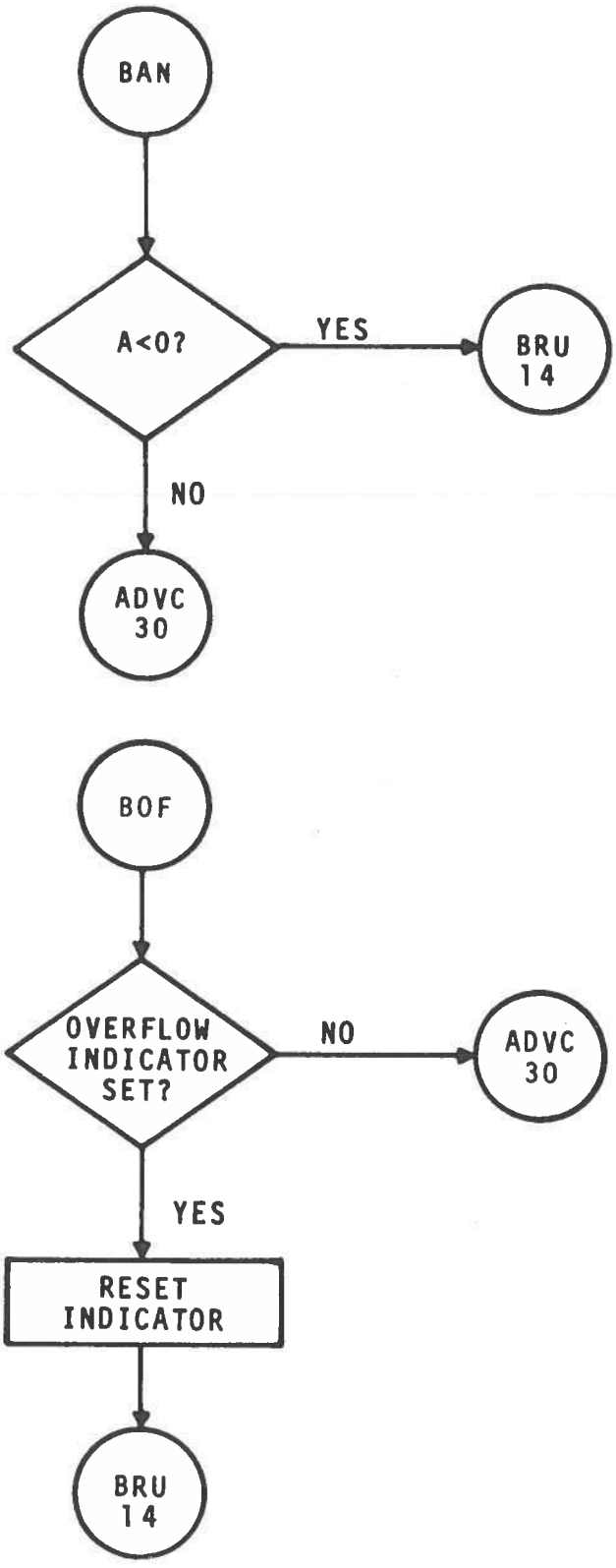

BAN
A<0?
YES
BRU
14
NO
ADVC
30
BOF
OVERFLOW
INDICATOR
SET?
NO
ADVC
30
YES
RESET
INDICATOR
BRU
14

Figure 15

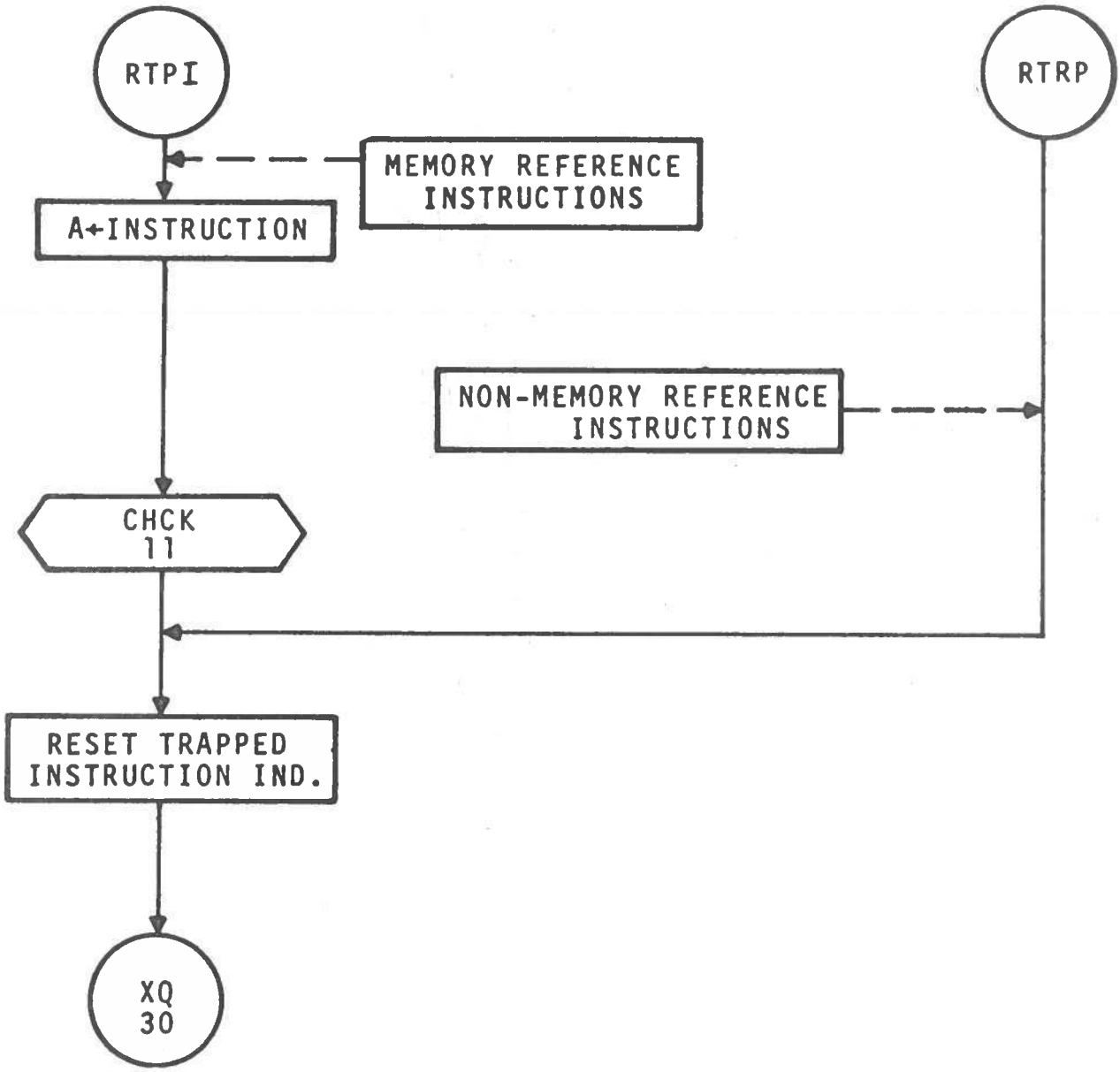

RTPI
RTRP
MEMORY REFERENCE INSTRUCTIONS
A←INSTRUCTION
NON-MEMORY REFERENCE INSTRUCTIONS
CHCK 11
RESET TRAPPED INSTRUCTION IND.
XQ 30

Figure 16

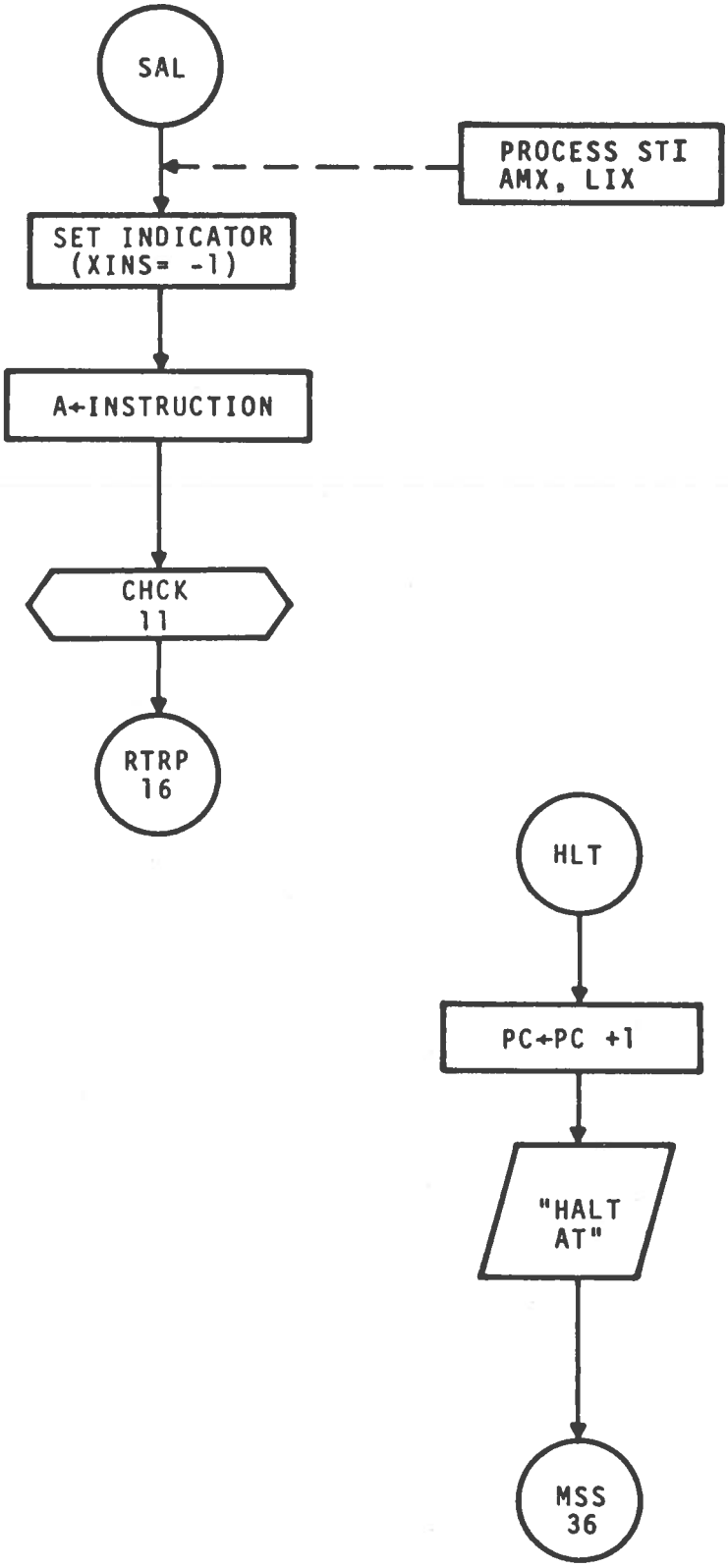

SAL
PROCESS STI
AMX, LIX
SET INDICATOR
(XINS= -1)
A←INSTRUCTION
CHCK
11
RTRP
16
HLT
PC←PC +1
"HALT
AT"
MSS
36

Figure 17

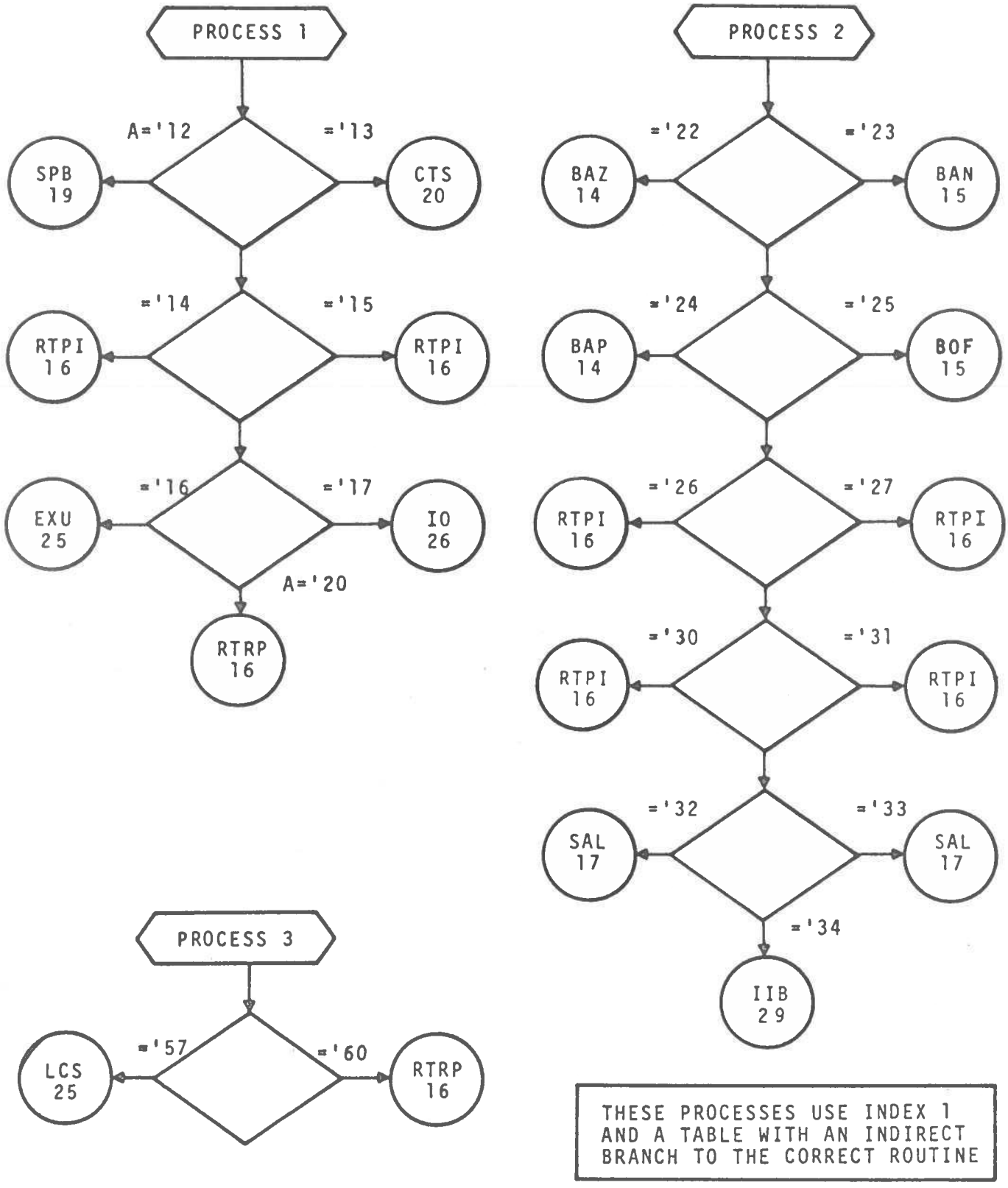

PROCESS 1
A='12
='13
SPB
19
CTS
20
='14
='15
RTPI
16
RTPI
16
='16
='17
EXU
25
IO
26
A='20
RTRP
16
PROCESS 2
='22
='23
BAZ
14
BAN
15
='24
='25
BAP
14
BOF
15
='26
='27
RTPI
16
RTPI
16
='30
='31
RTPI
16
RTPI
16
='32
='33
SAL
17
SAL
17
='34
IIB
29
PROCESS 3
='57
='60
LCS
25
RTRP
16
THESE PROCESSES USE INDEX 1
AND A TABLE WITH AN INDIRECT
BRANCH TO THE CORRECT ROUTINE

Figure 18

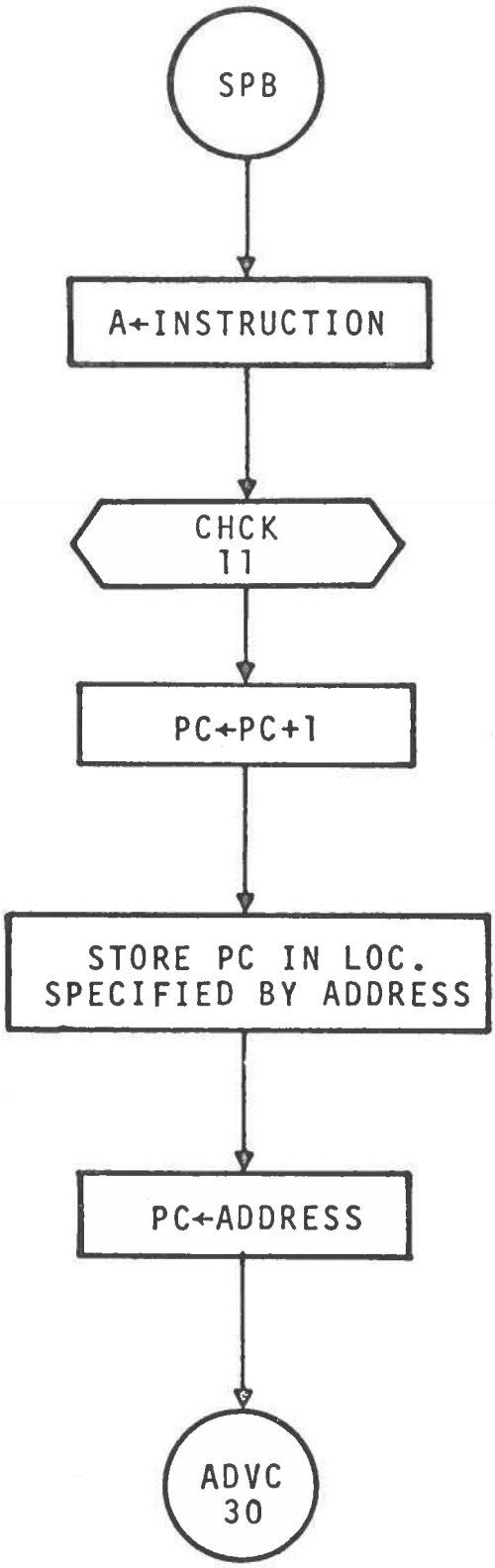

SPB
A←INSTRUCTION
CHCK
11
PC←PC+1
STORE PC IN LOC.
SPECIFIED BY ADDRESS
PC←ADDRESS
ADVC
30

Figure 19

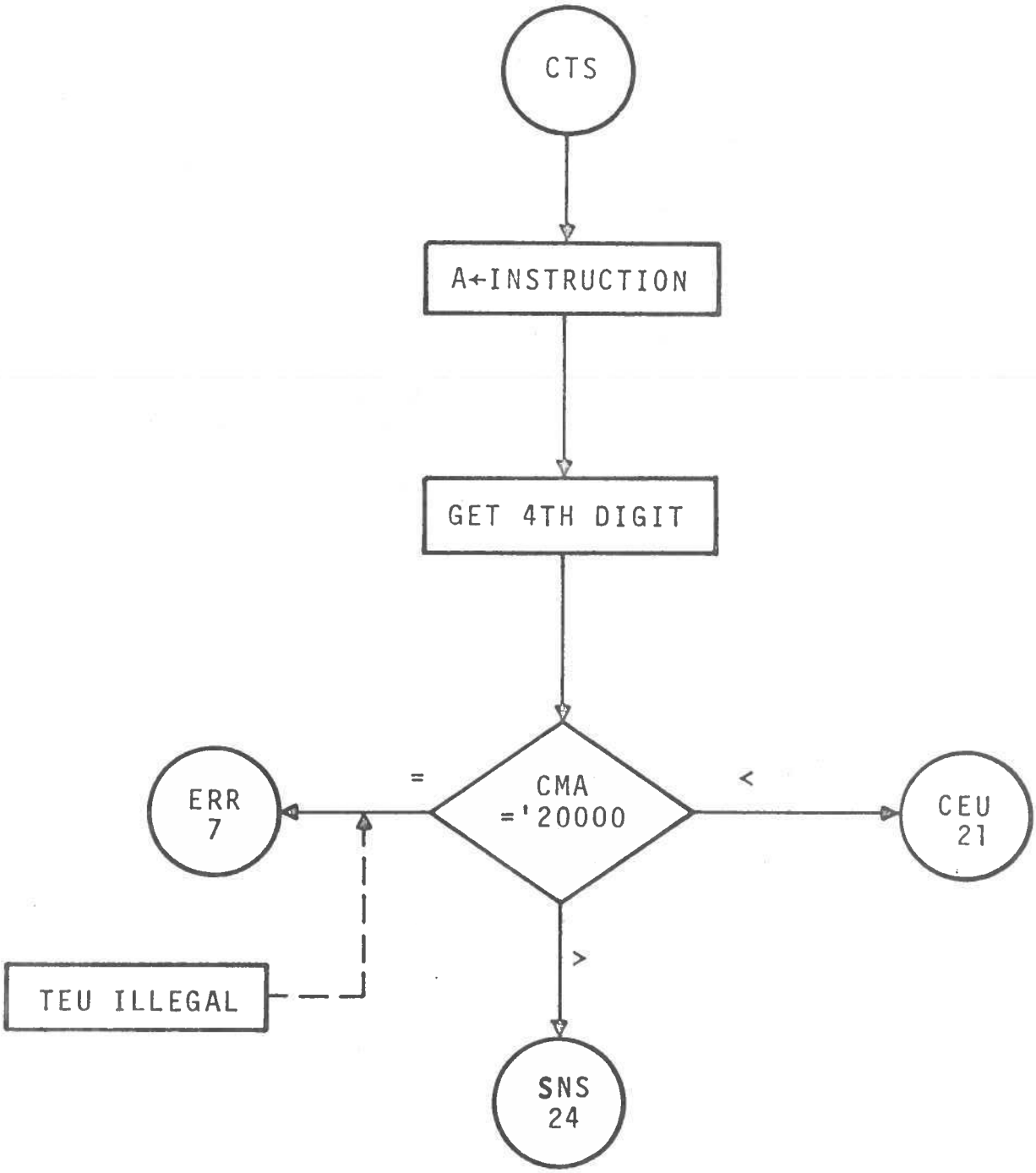

CTS
A←INSTRUCTION
GET 4TH DIGIT
CMA
='20000
=
<
>
ERR
7
CEU
21
SNS
24
TEU ILLEGAL

Figure 20

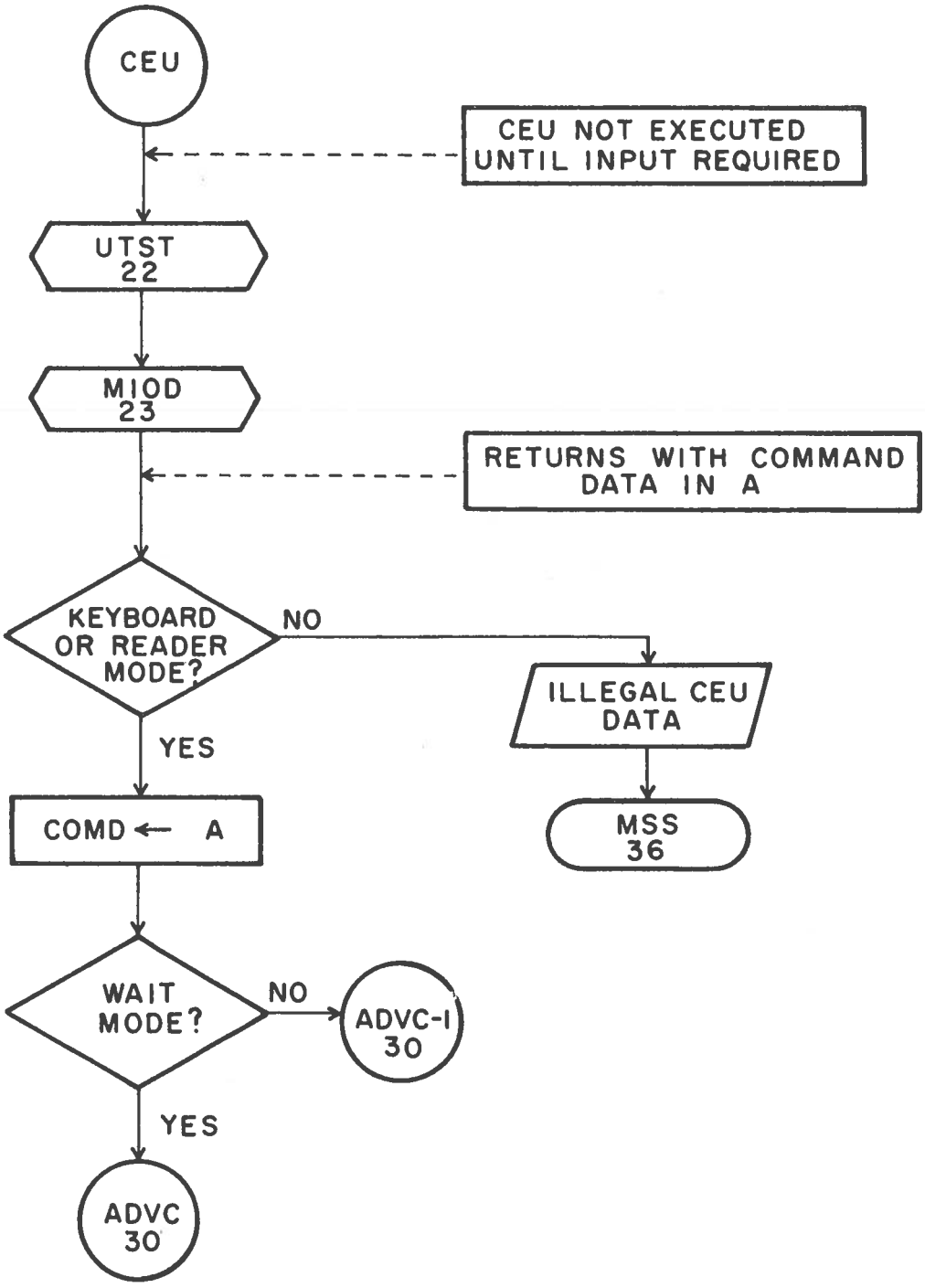

CEU
CEU NOT EXECUTED UNTIL INPUT REQUIRED
UTST 22
MIOD 23
RETURNS WITH COMMAND DATA IN A
KEYBOARD OR READER MODE?
NO
ILLEGAL CEU DATA
YES
COMD ← A
MSS 36
WAIT MODE?
NO
ADVC-I 30
YES
ADVC 30

Figure 21

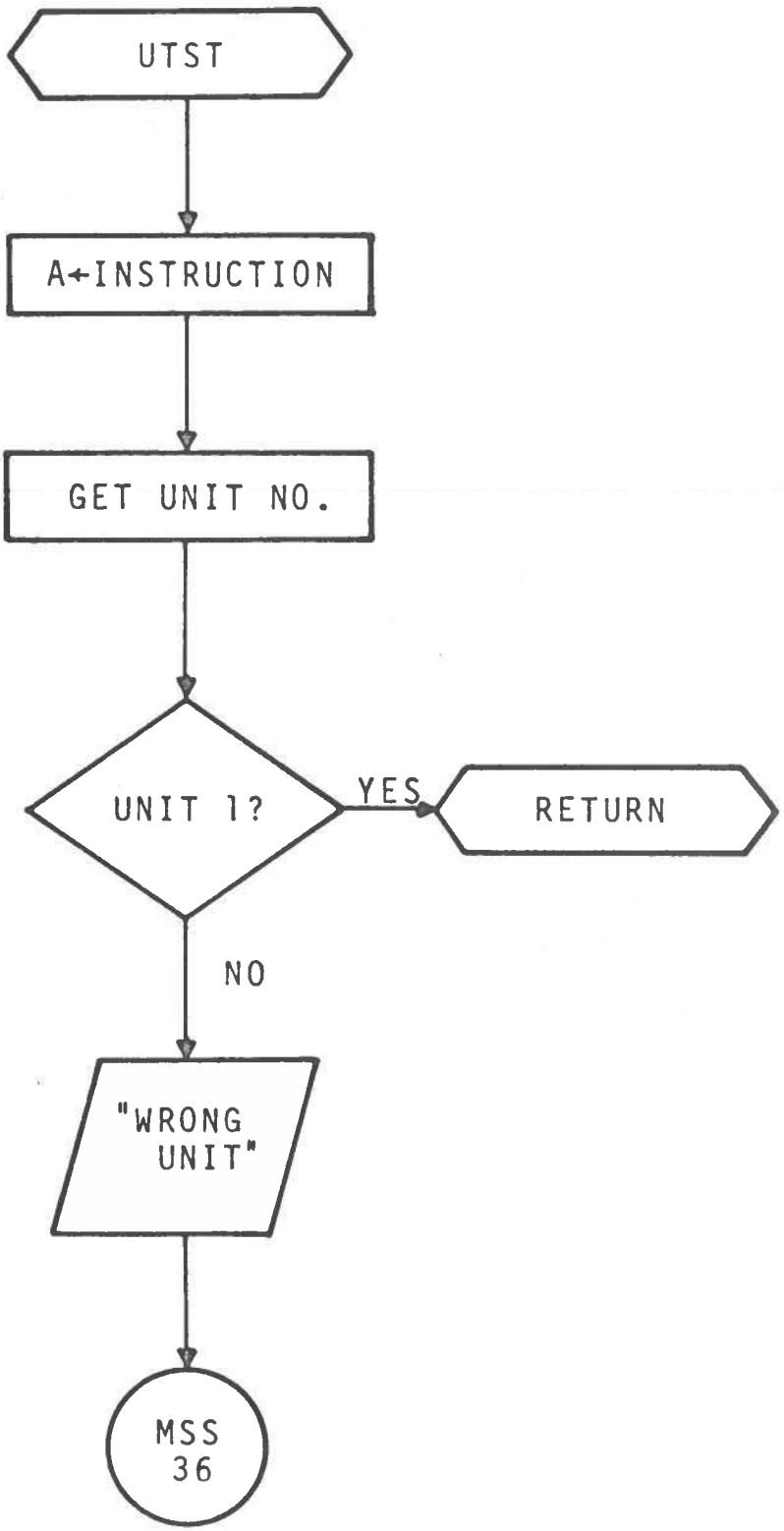

UTST
A←INSTRUCTION
GET UNIT NO.
UNIT 1?
YES
RETURN
NO
"WRONG UNIT"
MSS 36

Figure 22

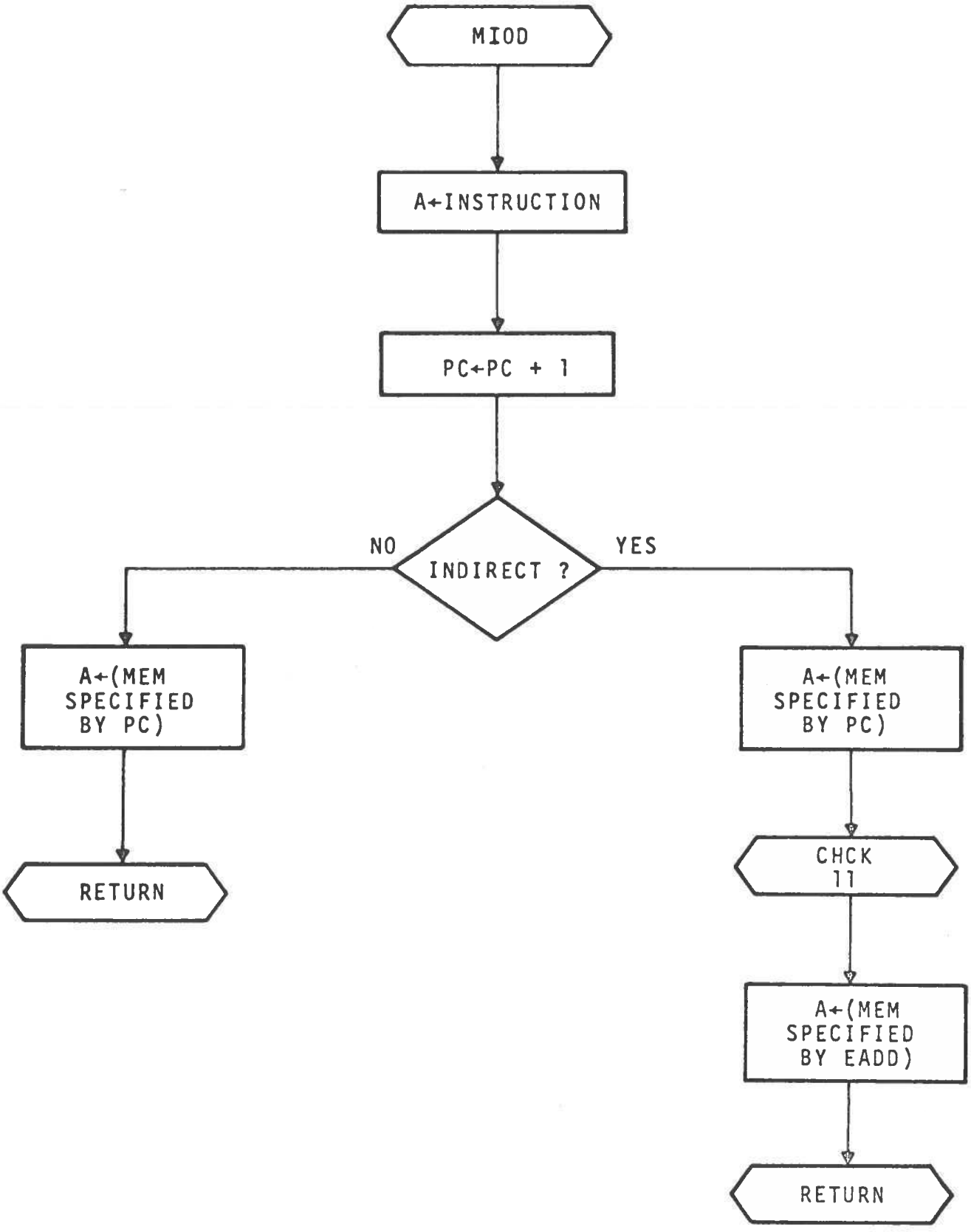

MIOD
A←INSTRUCTION
PC←PC + 1
INDIRECT ?
NO
YES
A←(MEM SPECIFIED BY PC)
RETURN
A←(MEM SPECIFIED BY PC)
CHCK 11
A←(MEM SPECIFIED BY EADD)
RETURN

Figure 23

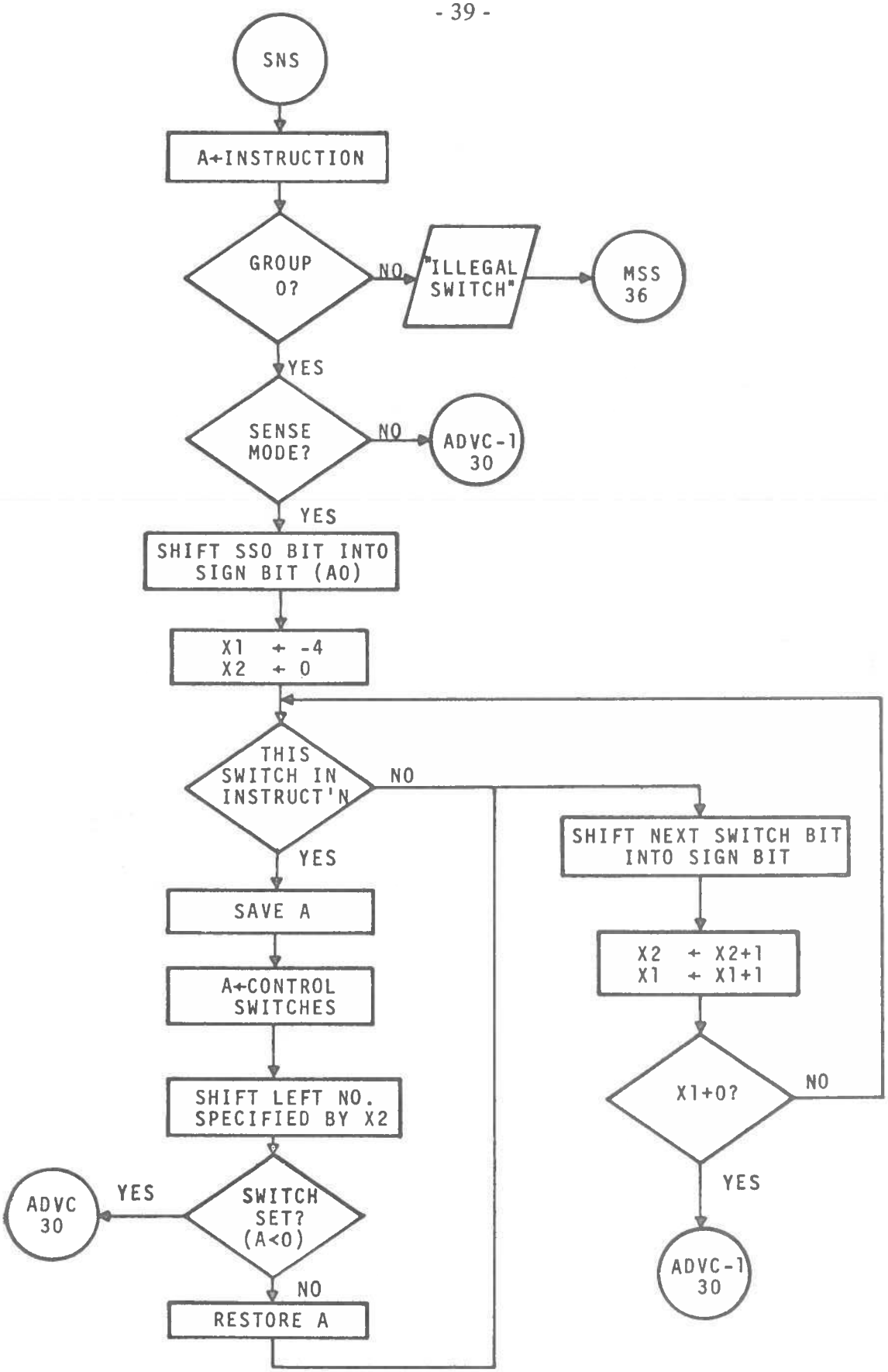

SNS
A←INSTRUCTION
GROUP 0?
NO
"ILLEGAL SWITCH"
MSS 36
YES
SENSE MODE?
NO
ADVC-1 30
YES
SHIFT SSO BIT INTO SIGN BIT (A0)
X1 ← -4
X2 ← 0
THIS SWITCH IN INSTRUCT'N
NO
YES
SHIFT NEXT SWITCH BIT INTO SIGN BIT
X2 ← X2+1
X1 ← X1+1
X1+0?
NO
YES
ADVC-1 30
SAVE A
A←CONTROL SWITCHES
SHIFT LEFT NO. SPECIFIED BY X2
SWITCH SET? (A<0)
YES
ADVC 30
NO
RESTORE A

Figure 24

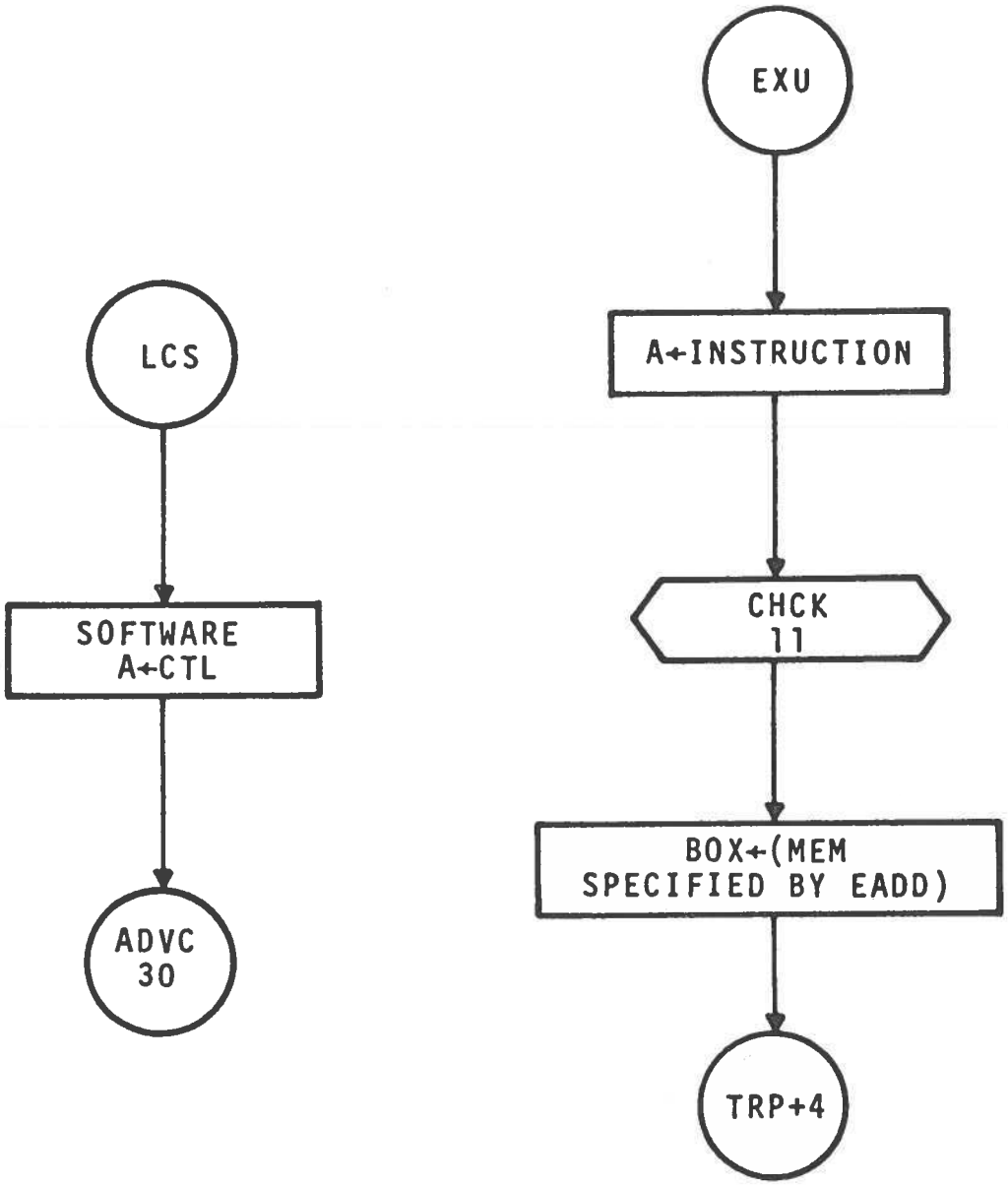

LCS
SOFTWARE
A←CTL
ADVC
30
EXU
A←INSTRUCTION
CHCK
11
BOX←(MEM
SPECIFIED BY EADD)
TRP+4

Figure 25

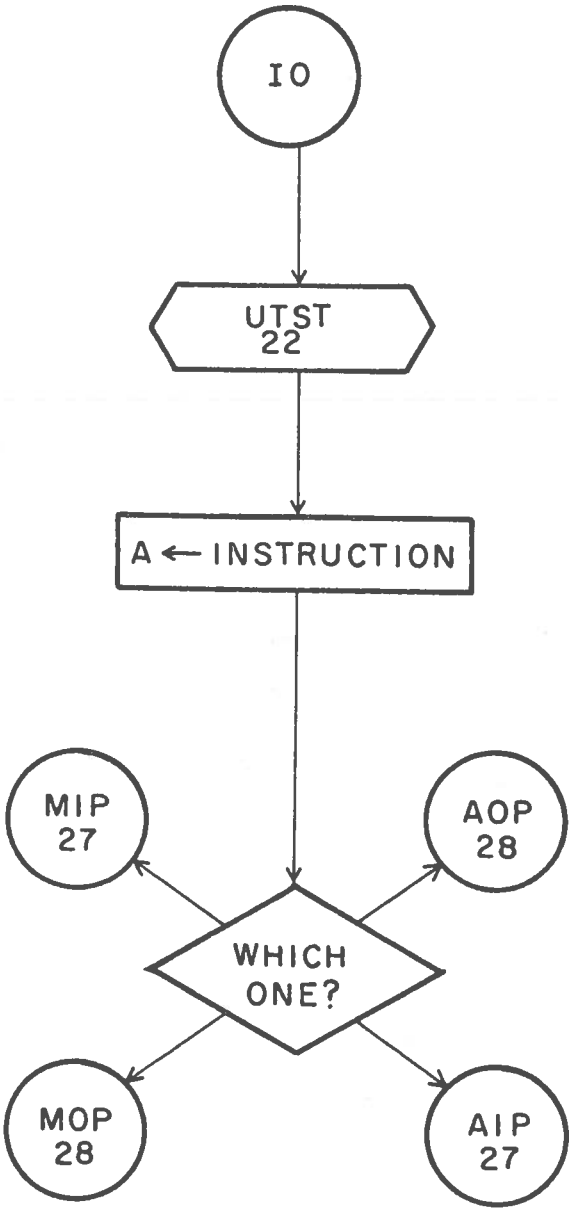


Figure 26

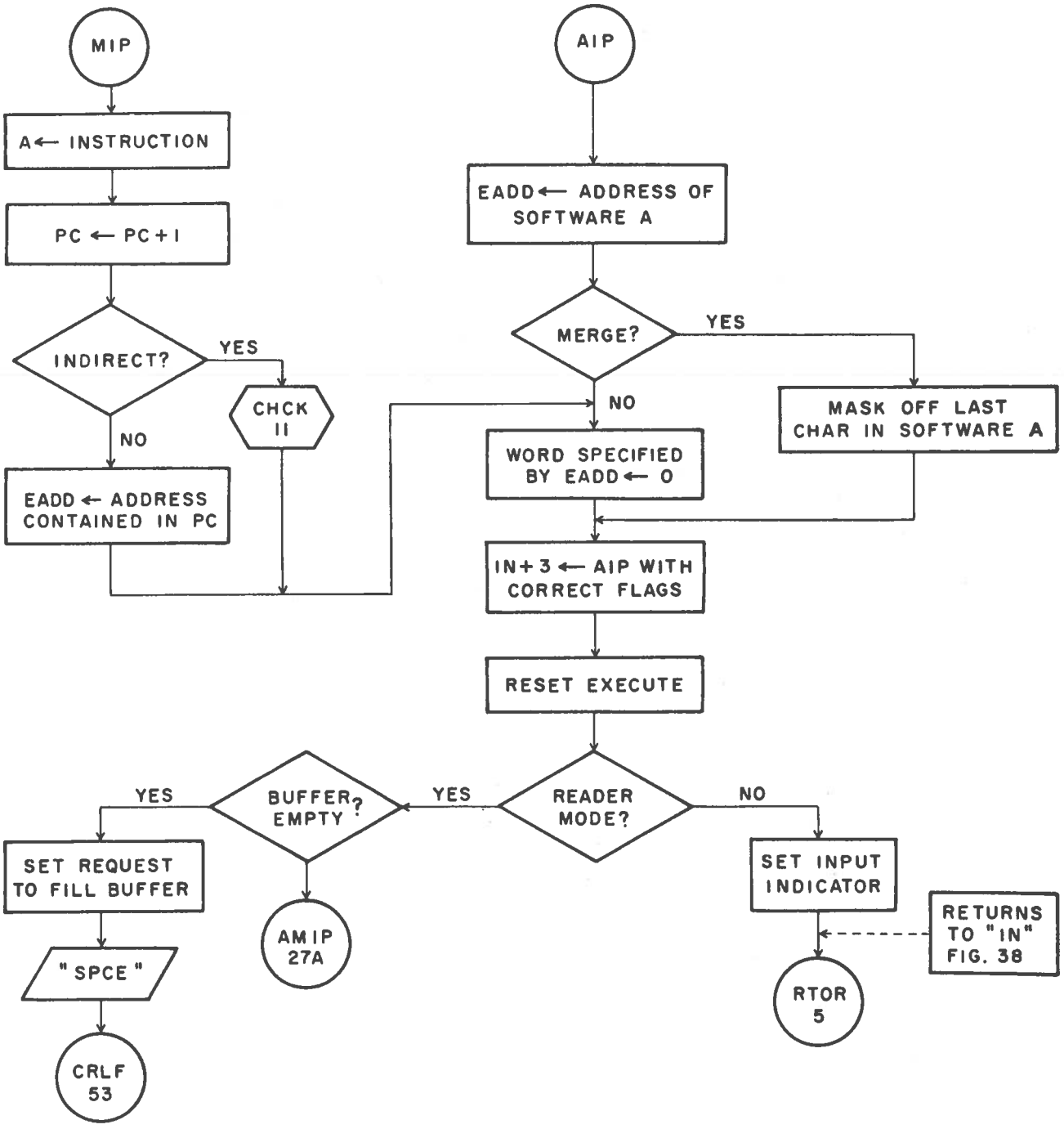

MIP
A← INSTRUCTION
PC ← PC+1
INDIRECT?
YES
NO
CHCK
11
EADD ← ADDRESS
CONTAINED IN PC
AIP
EADD← ADDRESS OF
SOFTWARE A
MERGE?
YES
NO
MASK OFF LAST
CHAR IN SOFTWARE A
WORD SPECIFIED
BY EADD← 0
IN+3 ← AIP WITH
CORRECT FLAGS
RESET EXECUTE
READER
MODE?
YES
NO
BUFFER
EMPTY?
YES
NO
SET REQUEST
TO FILL BUFFER
" SPCE "
CRLF
53
AMIP
27A
SET INPUT
INDICATOR
RETURNS
TO "IN"
FIG. 38
RTOR
5

Figure 27(a)

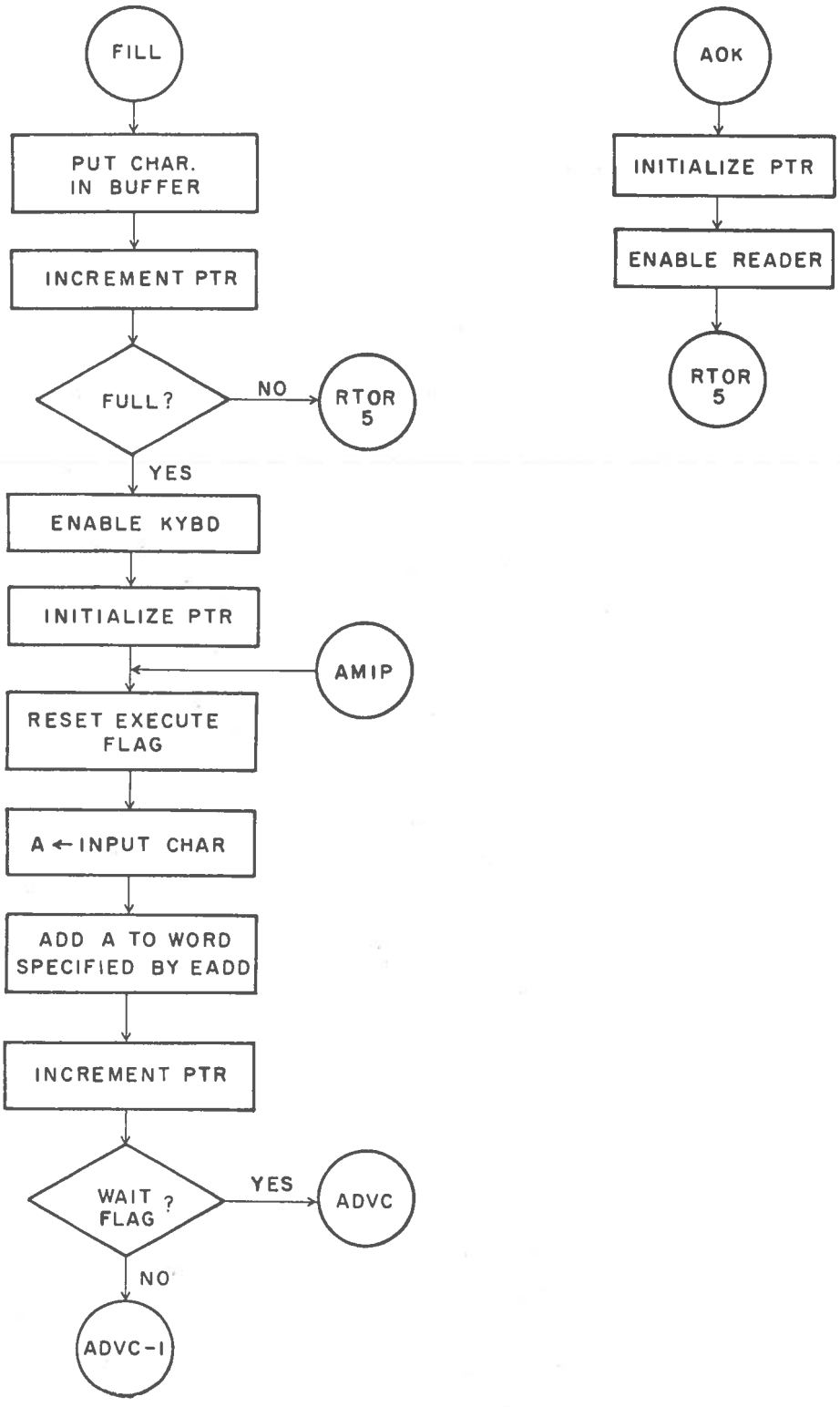

FILL
PUT CHAR. IN BUFFER
INCREMENT PTR
FULL?
NO
RTOR 5
YES
ENABLE KYBD
INITIALIZE PTR
AMIP
RESET EXECUTE FLAG
A ← INPUT CHAR
ADD A TO WORD SPECIFIED BY EADD
INCREMENT PTR
WAIT ? FLAG
YES
ADVC
NO
ADVC-1
AOK
INITIALIZE PTR
ENABLE READER
RTOR 5

Figure 27(b)

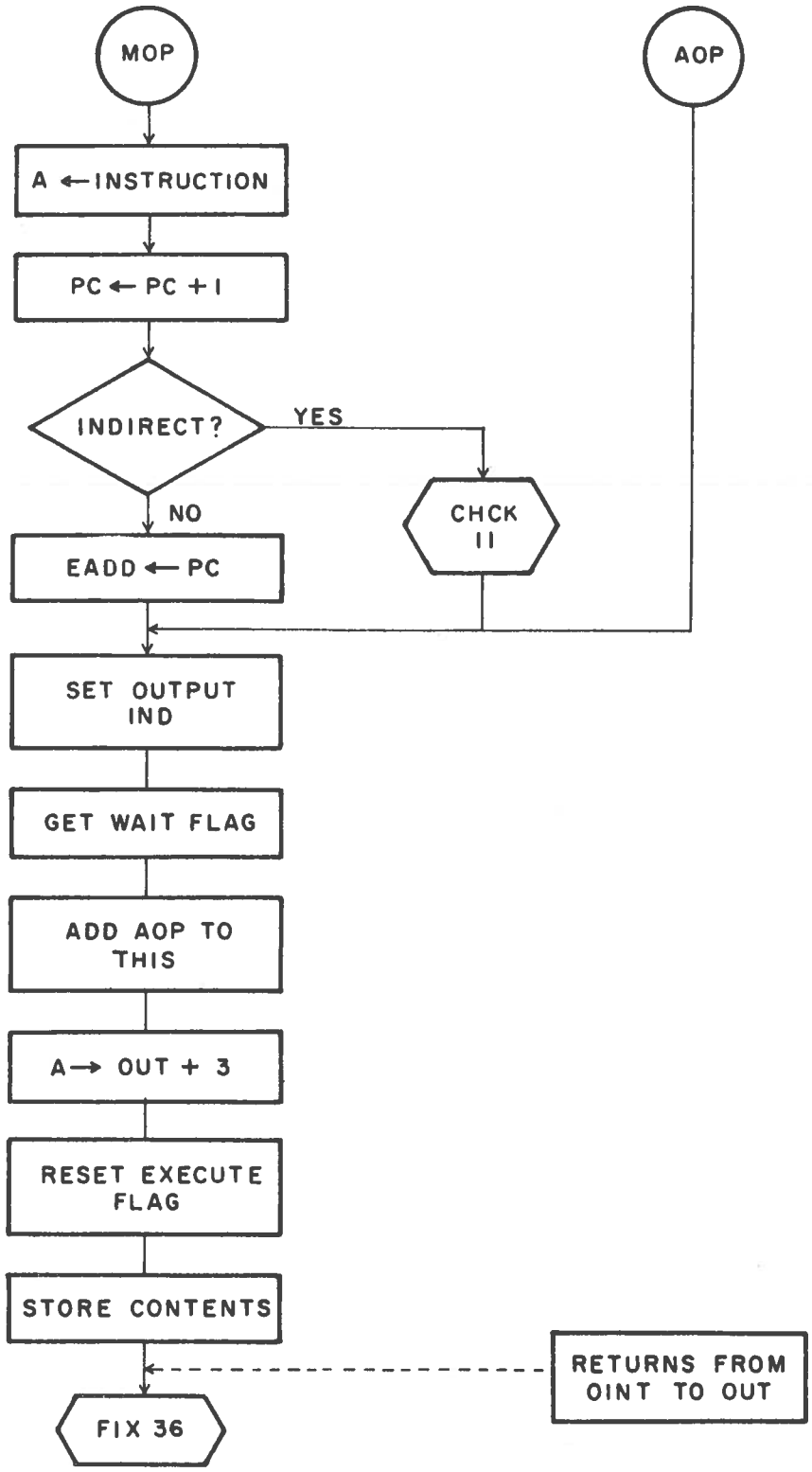

MOP
AOP
A ← INSTRUCTION
PC ← PC + 1
INDIRECT?
YES
NO
CHCK
11
EADD ← PC
SET OUTPUT
IND
GET WAIT FLAG
ADD AOP TO
THIS
A → OUT + 3
RESET EXECUTE
FLAG
STORE CONTENTS
RETURNS FROM
OINT TO OUT
FIX 36

Figure 28

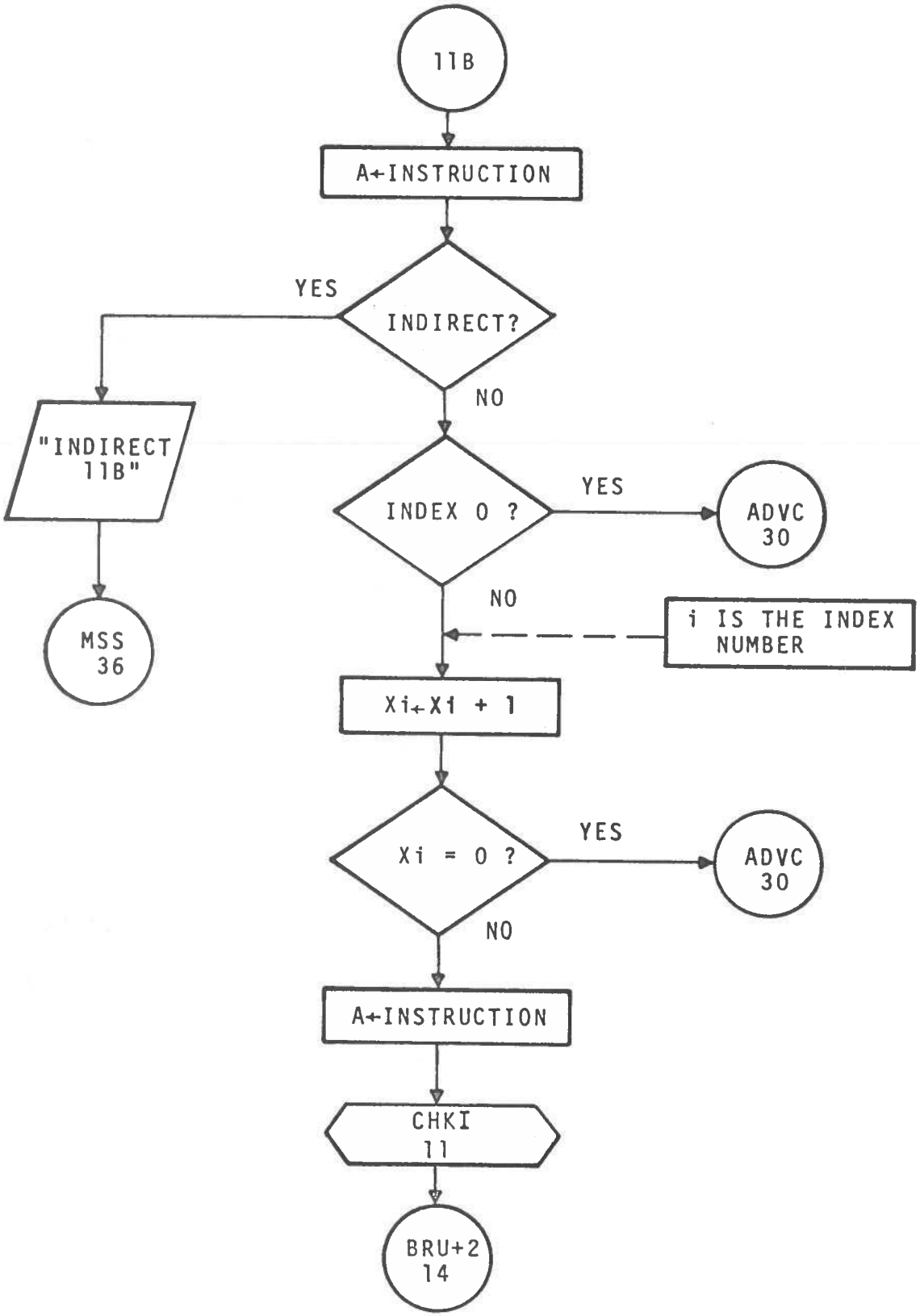

11B
A←INSTRUCTION
YES
INDIRECT?
NO
"INDIRECT 11B"
INDEX 0 ?
YES
ADVC 30
NO
i IS THE INDEX NUMBER
MSS 36
Xi←Xi + 1
Xi = 0 ?
YES
ADVC 30
NO
A←INSTRUCTION
CHKI 11
BRU+2 14

Figure 29

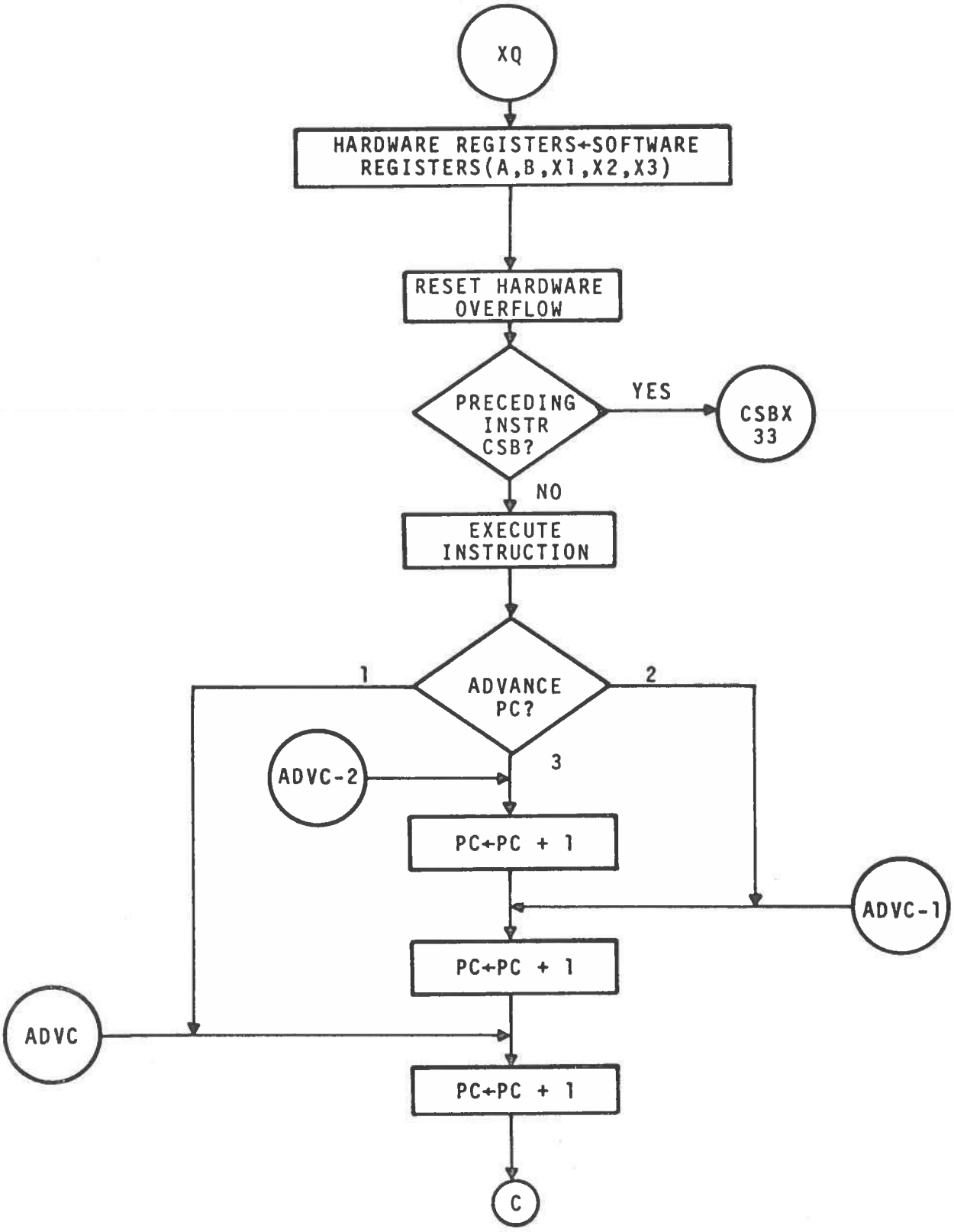

XQ
HARDWARE REGISTERS←SOFTWARE REGISTERS(A,B,X1,X2,X3)
RESET HARDWARE OVERFLOW
PRECEDING INSTR CSB?
YES
CSBX 33
NO
EXECUTE INSTRUCTION
ADVANCE PC?
1
2
3
ADVC-2
PC←PC + 1
ADVC-1
PC←PC + 1
ADVC
PC←PC + 1
C

Figure 30

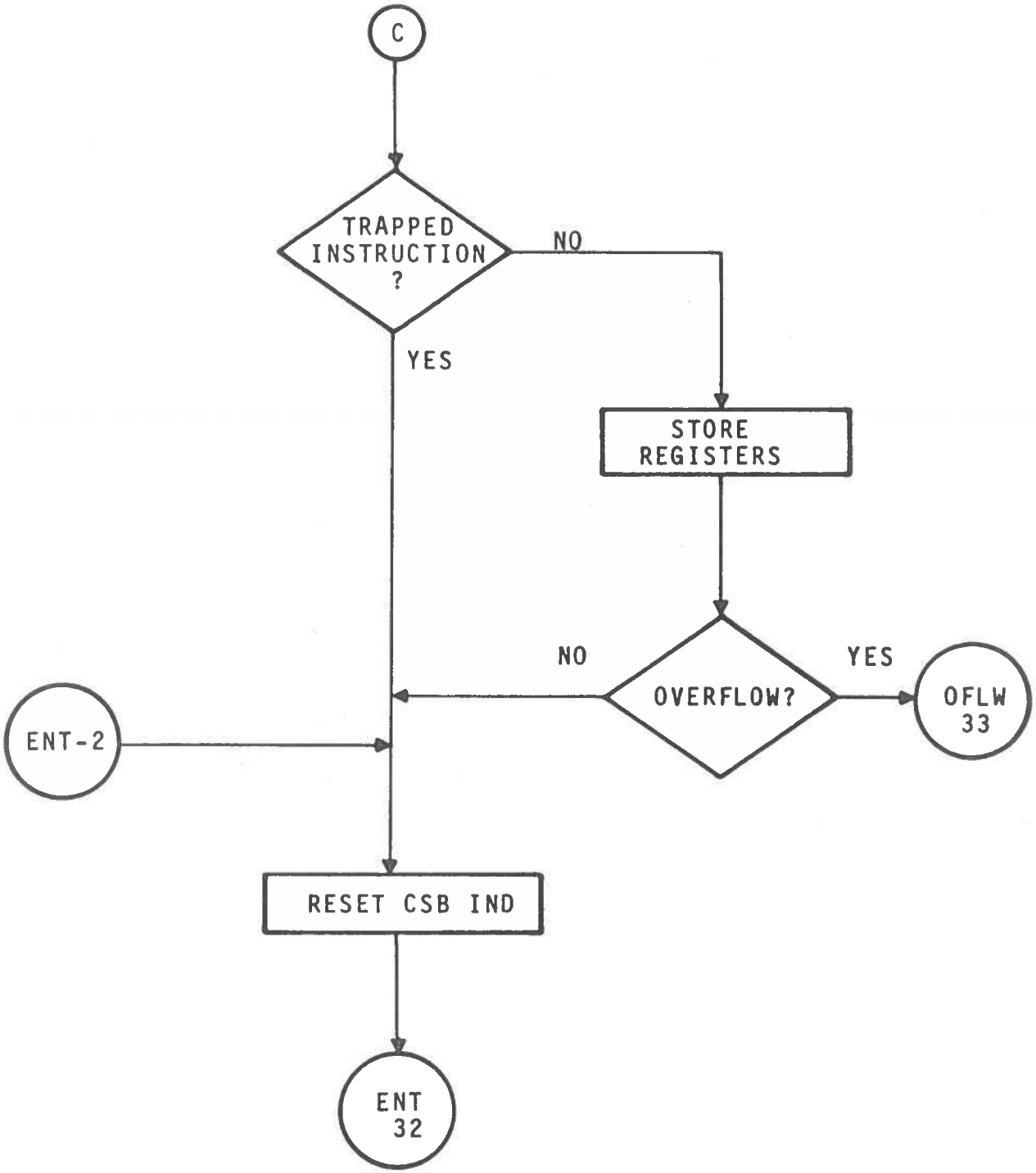

C
TRAPPED INSTRUCTION ?
NO
YES
STORE REGISTERS
NO
OVERFLOW?
YES
OFLW 33
ENT-2
RESET CSB IND
ENT 32

Figure 31

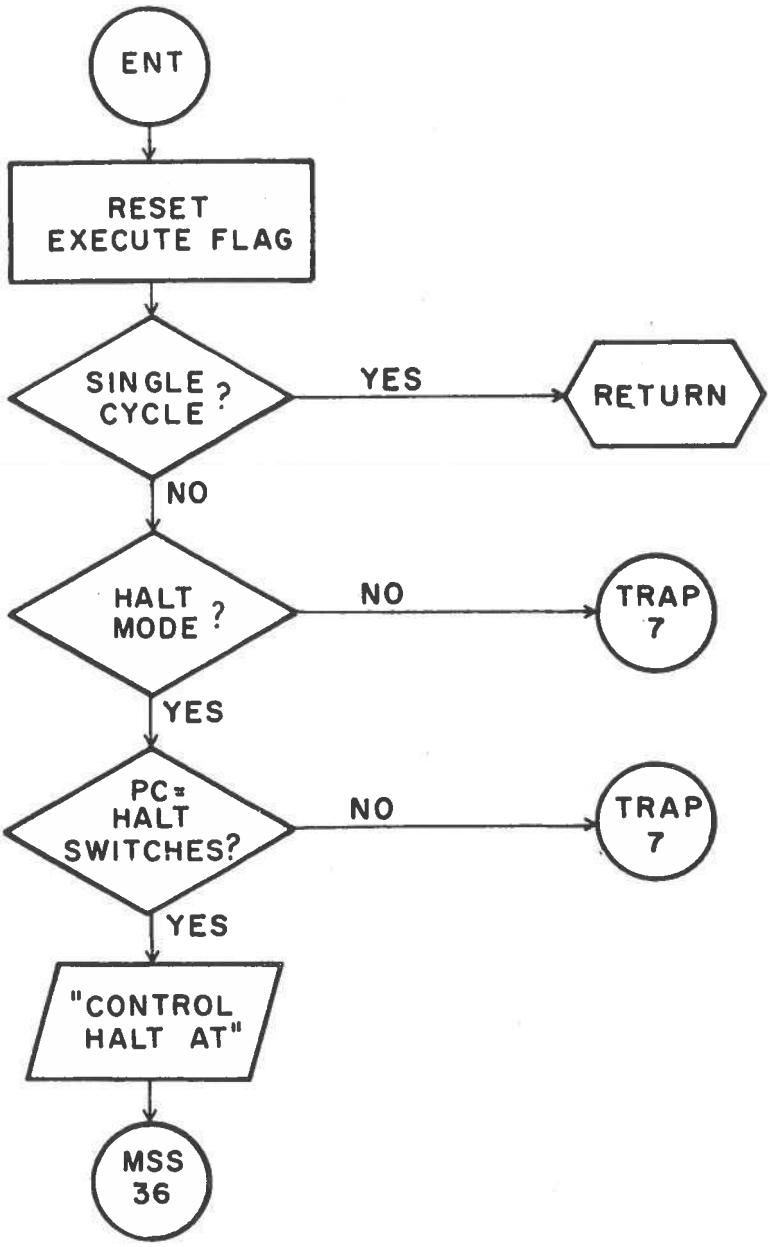

ENT
RESET EXECUTE FLAG
SINGLE ? CYCLE
YES
RETURN
NO
HALT MODE ?
NO
TRAP 7
YES
PC= HALT SWITCHES?
NO
TRAP 7
YES
"CONTROL HALT AT"
MSS 36

Figure 32

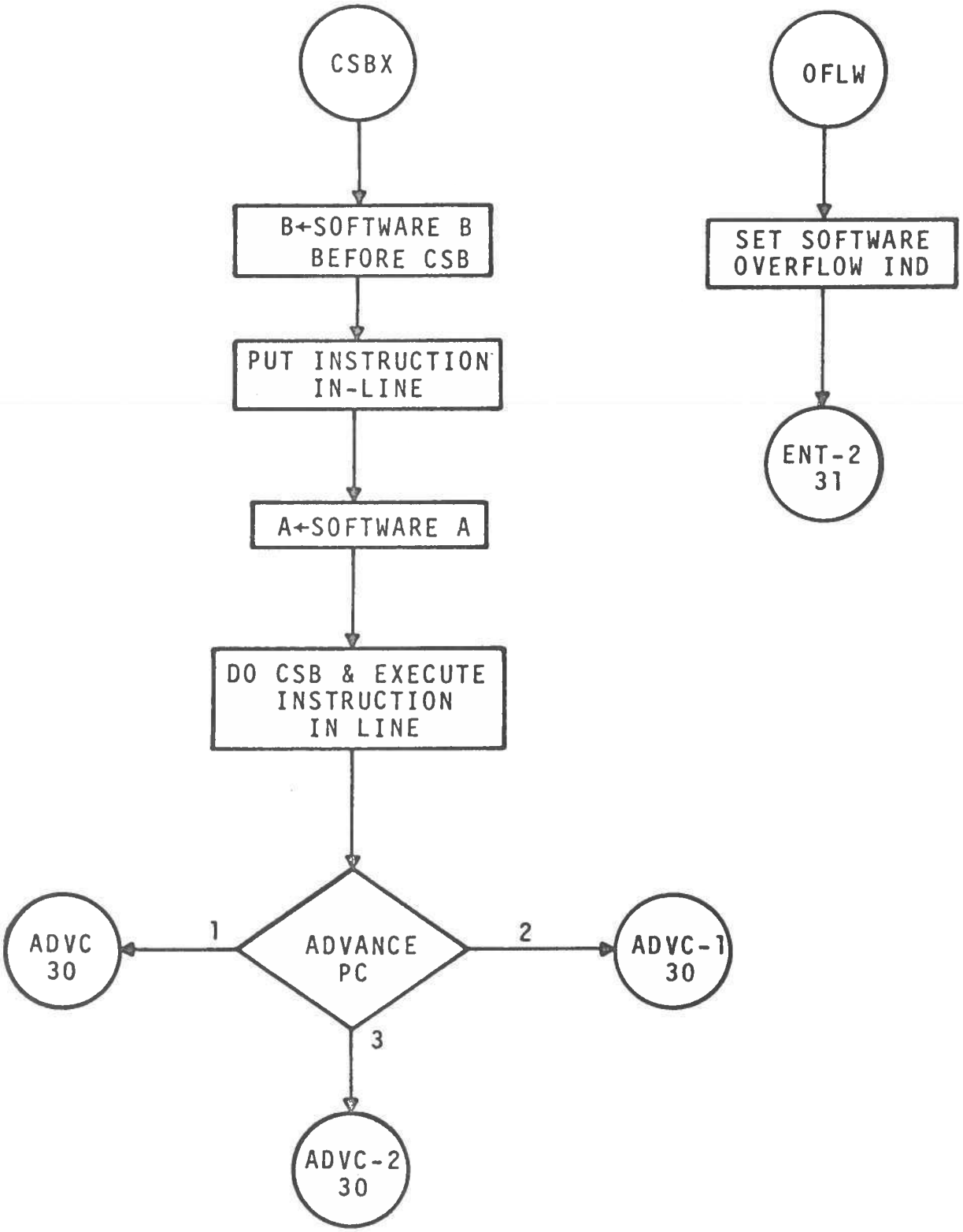

CSBX
B←SOFTWARE B BEFORE CSB
PUT INSTRUCTION IN-LINE
A←SOFTWARE A
DO CSB & EXECUTE INSTRUCTION IN LINE
ADVANCE PC
1
2
3
ADVC 30
ADVC-1 30
ADVC-2 30
OFLW
SET SOFTWARE OVERFLOW IND
ENT-2 31

Figure 33

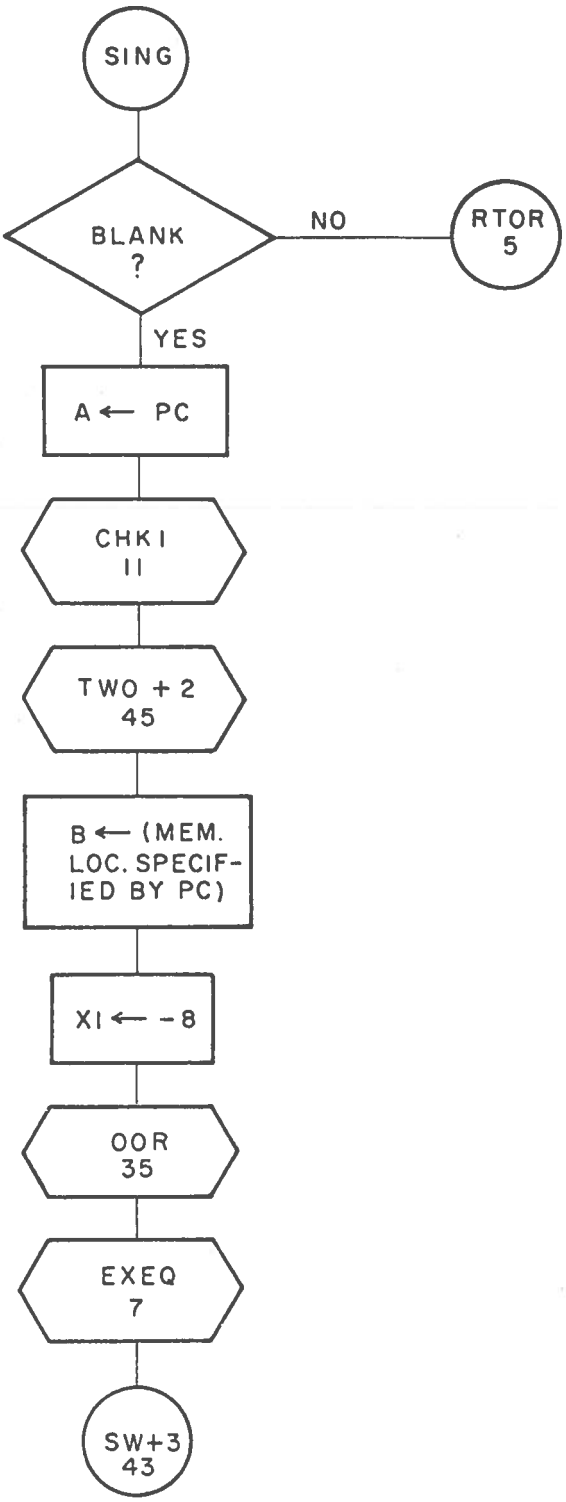

SING
BLANK ?
NO
RTOR 5
YES
A ← PC
CHKI 11
TWO + 2 45
B ← (MEM. LOC. SPECIF-IED BY PC)
XI ← -8
OOR 35
EXEQ 7
SW+3 43

Figure 34

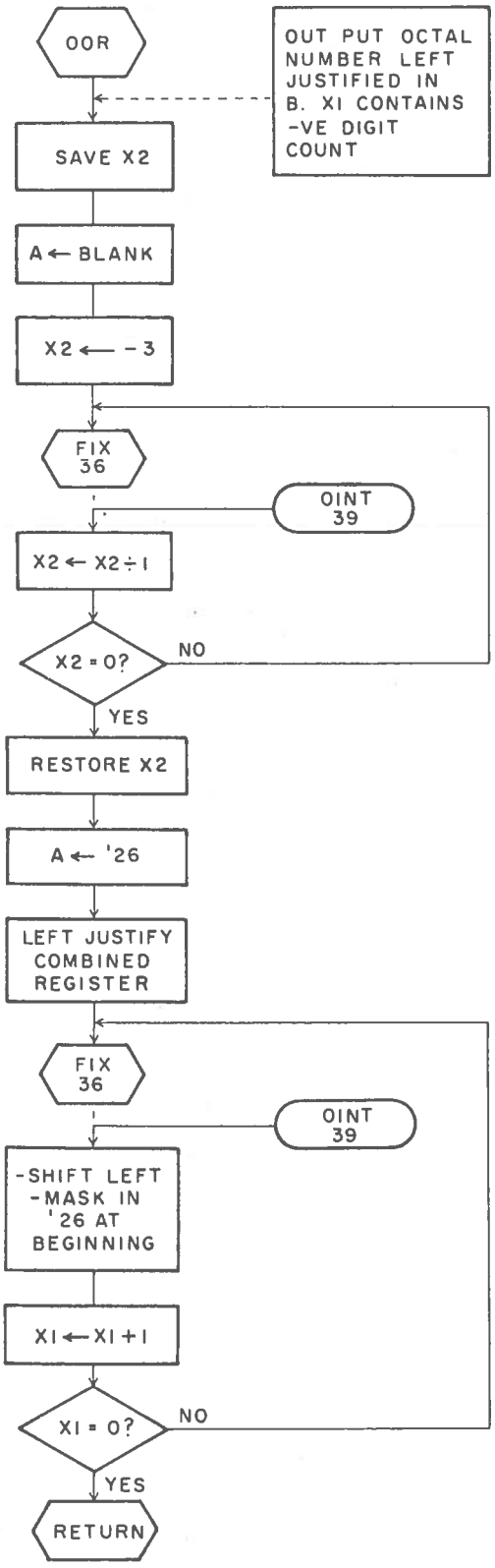

OOR
OUT PUT OCTAL NUMBER LEFT JUSTIFIED IN B. XI CONTAINS -VE DIGIT COUNT
SAVE X2
A ← BLANK
X2 ← -3
FIX 36
OINT 39
X2 ← X2 ÷ 1
X2 = 0?
NO
YES
RESTORE X2
A ← '26
LEFT JUSTIFY COMBINED REGISTER
FIX 36
OINT 39
-SHIFT LEFT -MASK IN '26 AT BEGINNING
XI ← XI + 1
XI = 0?
NO
YES
RETURN

Figure 35

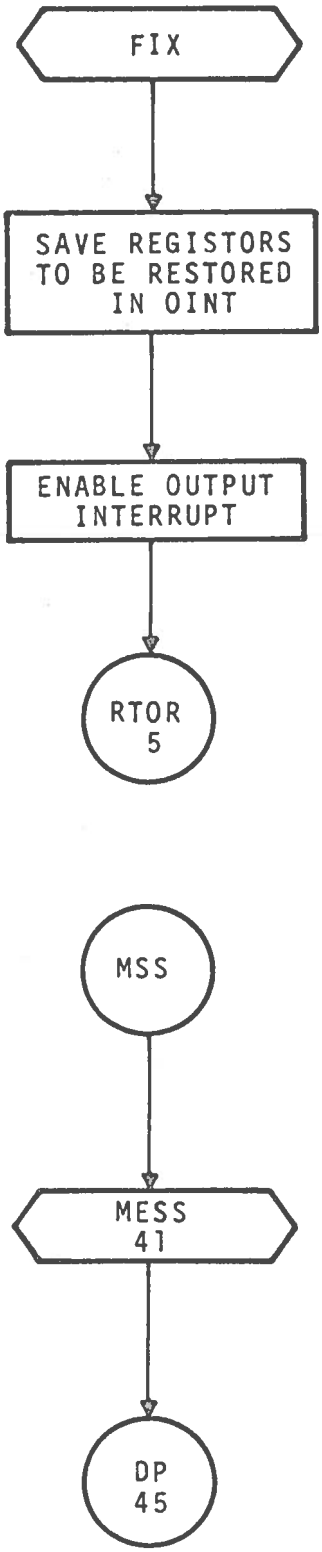

FIX
SAVE REGISTORS TO BE RESTORED IN OINT
ENABLE OUTPUT INTERRUPT
RTOR 5
MSS
MESS 41
DP 45

Figure 36

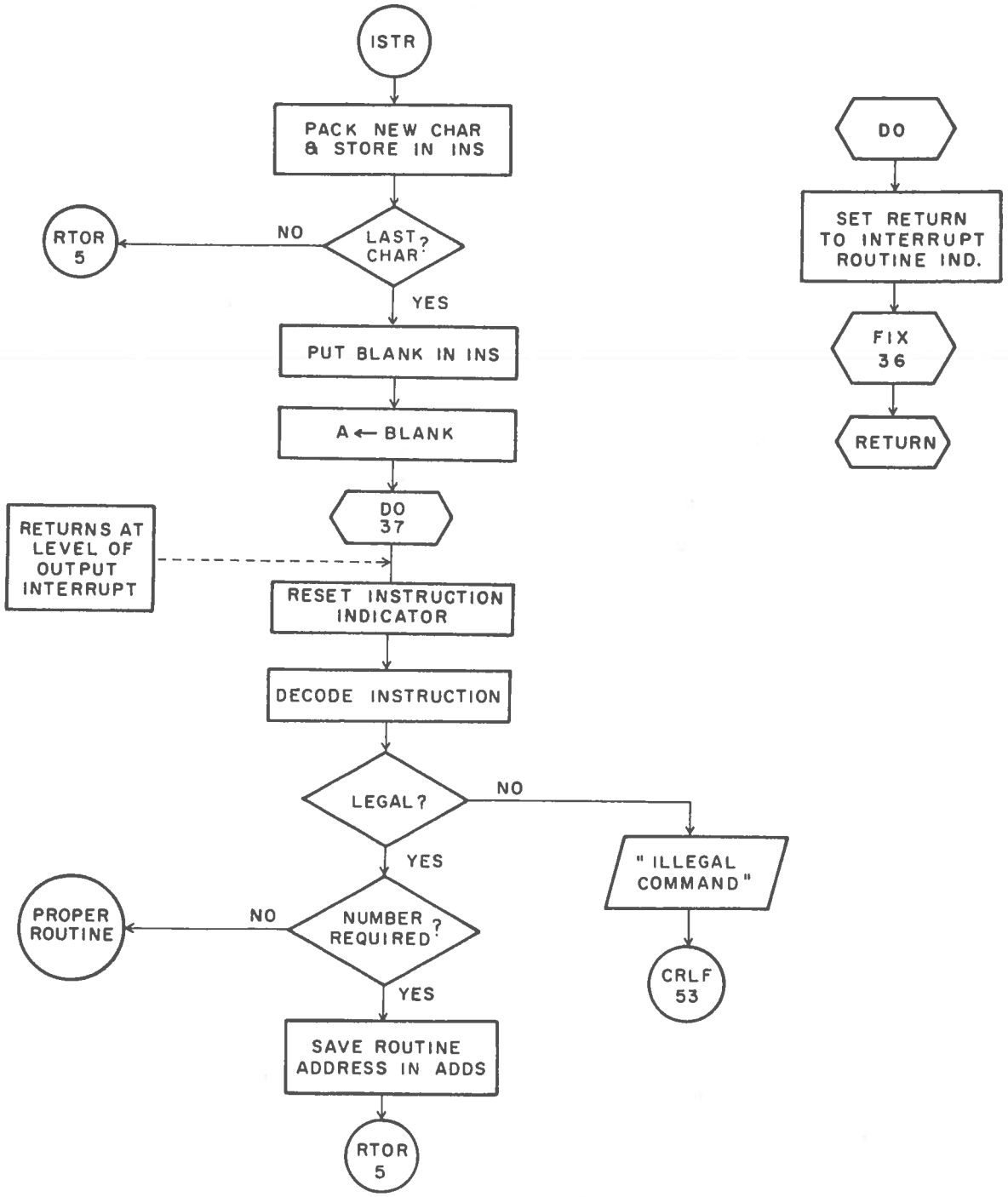

ISTR
PACK NEW CHAR & STORE IN INS
LAST CHAR?
NO
RTOR 5
YES
PUT BLANK IN INS
A ← BLANK
DO 37
RETURNS AT LEVEL OF OUTPUT INTERRUPT
RESET INSTRUCTION INDICATOR
DECODE INSTRUCTION
LEGAL?
NO
" ILLEGAL COMMAND"
CRLF 53
YES
NUMBER REQUIRED?
NO
PROPER ROUTINE
YES
SAVE ROUTINE ADDRESS IN ADDS
RTOR 5
DO
SET RETURN TO INTERRUPT ROUTINE IND.
FIX 36
RETURN

Figure 37

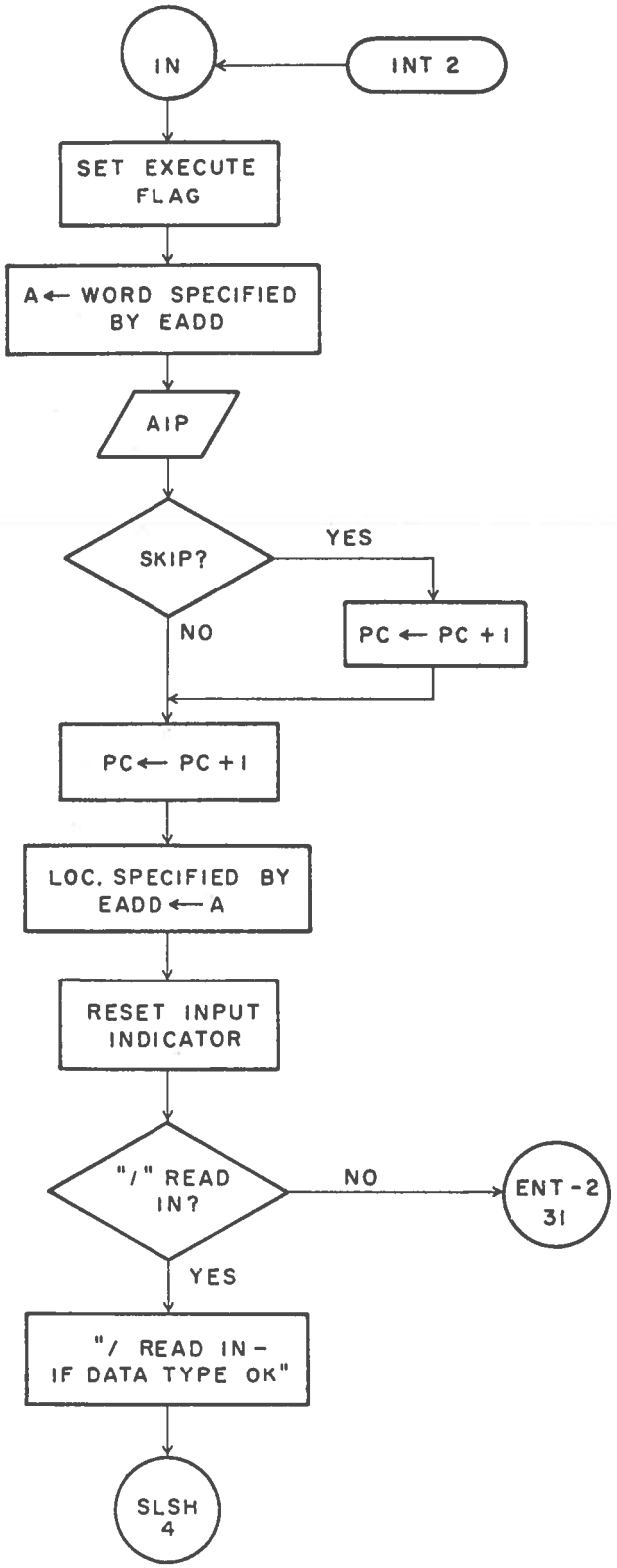

IN
INT 2
SET EXECUTE FLAG
A← WORD SPECIFIED BY EADD
AIP
SKIP?
YES
NO
PC ← PC + 1
PC← PC + 1
LOC. SPECIFIED BY EADD ← A
RESET INPUT INDICATOR
"/" READ IN?
NO
ENT - 2 31
YES
"/ READ IN - IF DATA TYPE OK"
SLSH 4

Figure 38

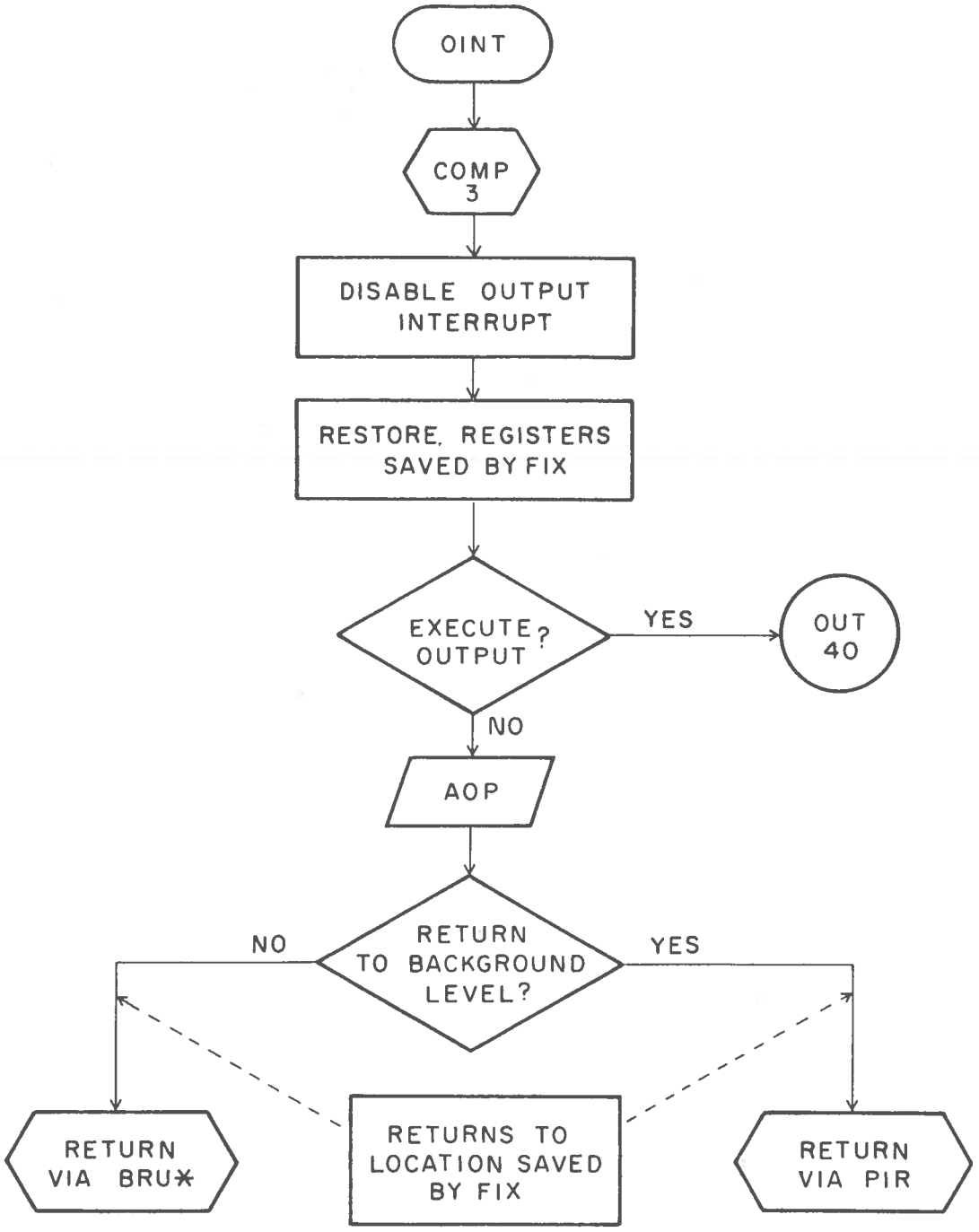

OINT
COMP 3
DISABLE OUTPUT INTERRUPT
RESTORE REGISTERS SAVED BY FIX
EXECUTE? OUTPUT
YES
OUT 40
NO
AOP
RETURN TO BACKGROUND LEVEL?
NO
YES
RETURN VIA BRU*
RETURNS TO LOCATION SAVED BY FIX
RETURN VIA PIR

Figure 39

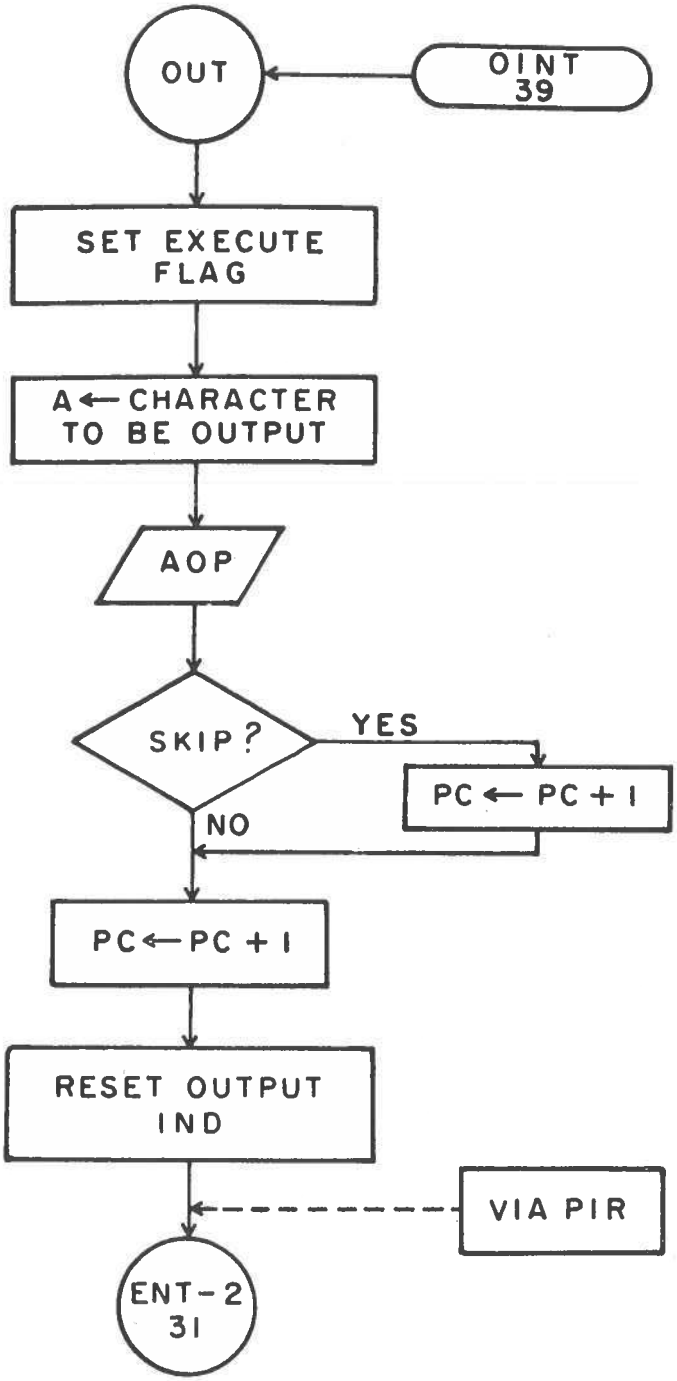

OUT
OINT
39
SET EXECUTE
FLAG
A ← CHARACTER
TO BE OUTPUT
AOP
SKIP ?
YES
PC ← PC + 1
NO
PC ← PC + 1
RESET OUTPUT
IND
VIA PIR
ENT - 2
31

Figure 40

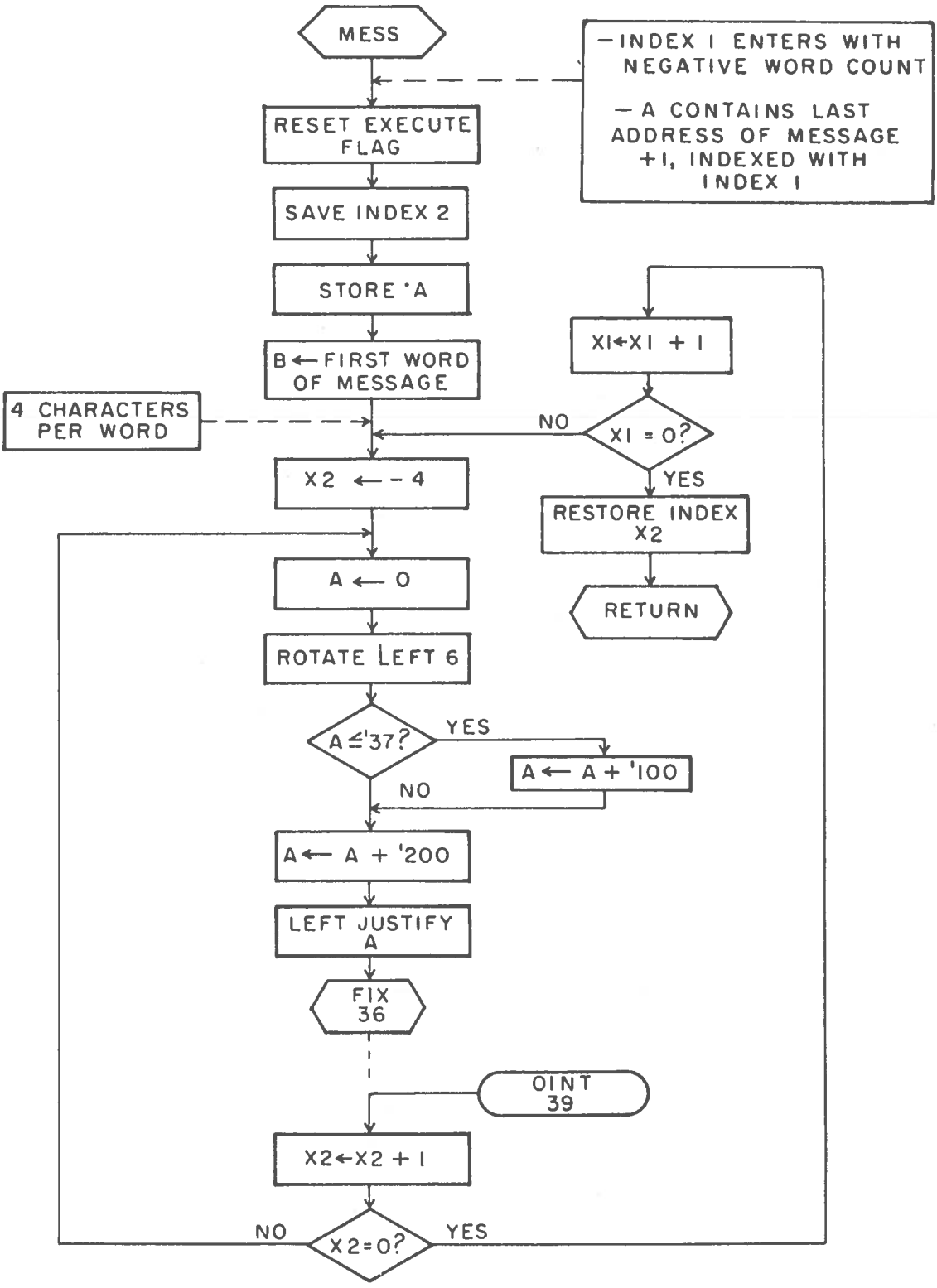

MESS
- INDEX 1 ENTERS WITH NEGATIVE WORD COUNT
- A CONTAINS LAST ADDRESS OF MESSAGE +1, INDEXED WITH INDEX 1
RESET EXECUTE FLAG
SAVE INDEX 2
STORE *A
B ← FIRST WORD OF MESSAGE
4 CHARACTERS PER WORD
X2 ← - 4
A ← 0
ROTATE LEFT 6
A ≤ '37?
YES
A ← A + '100
NO
A ← A + '200
LEFT JUSTIFY A
FIX 36
OINT 39
X2 ← X2 + 1
X2=0?
NO
YES
X1 ← X1 + 1
X1 = 0?
NO
YES
RESTORE INDEX X2
RETURN

Figure 41

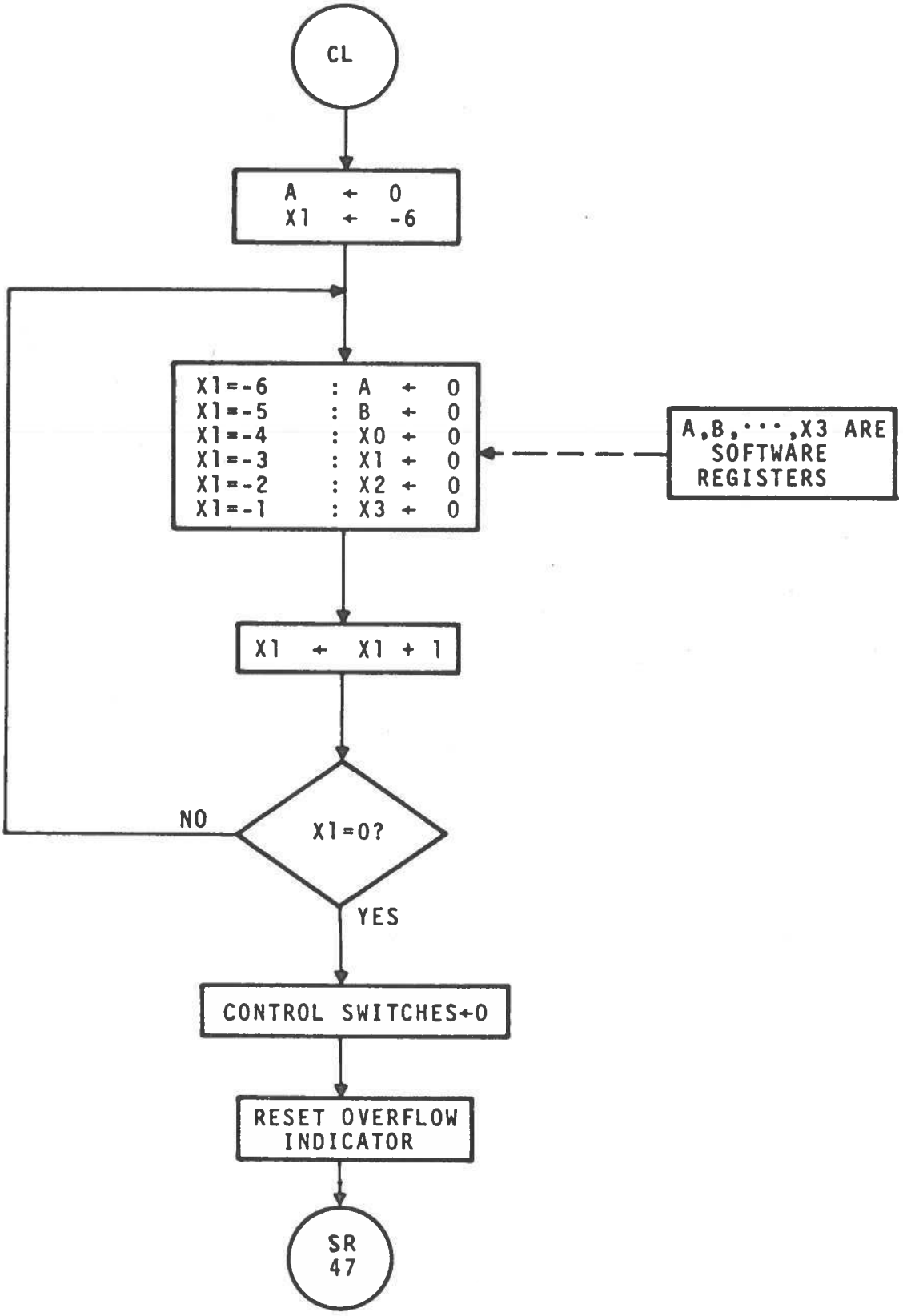

CL
A ← 0
X1 ← -6
X1=-6 : A ← 0
X1=-5 : B ← 0
X1=-4 : X0 ← 0
X1=-3 : X1 ← 0
X1=-2 : X2 ← 0
X1=-1 : X3 ← 0
A,B,···,X3 ARE SOFTWARE REGISTERS
X1 ← X1 + 1
X1=0?
NO
YES
CONTROL SWITCHES←0
RESET OVERFLOW INDICATOR
SR 47

Figure 42

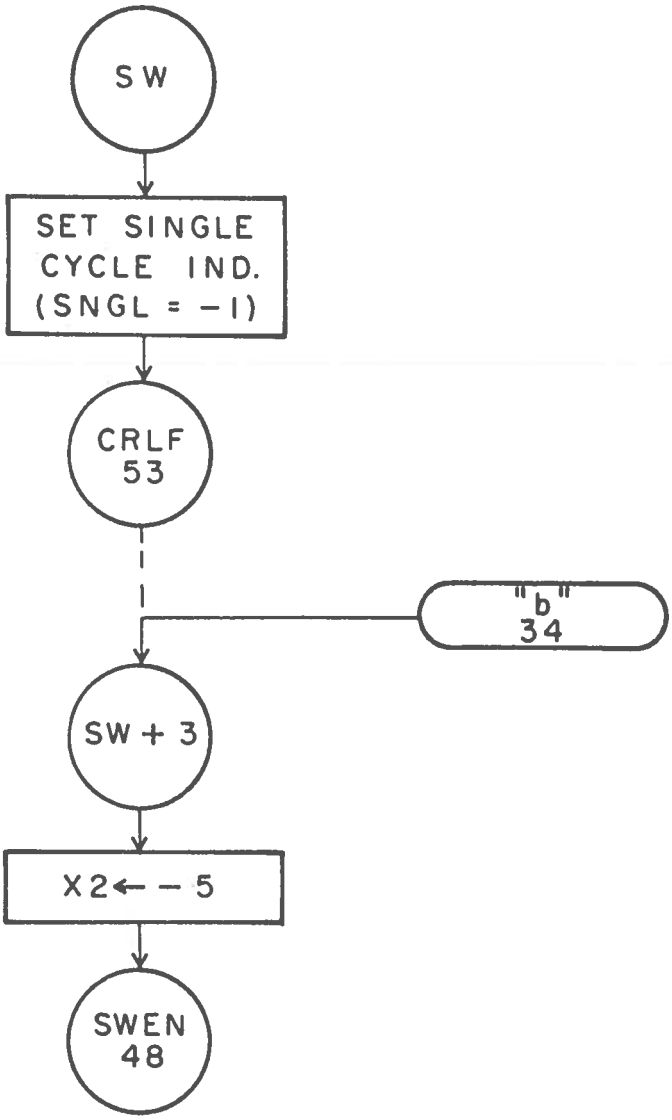


Figure 43

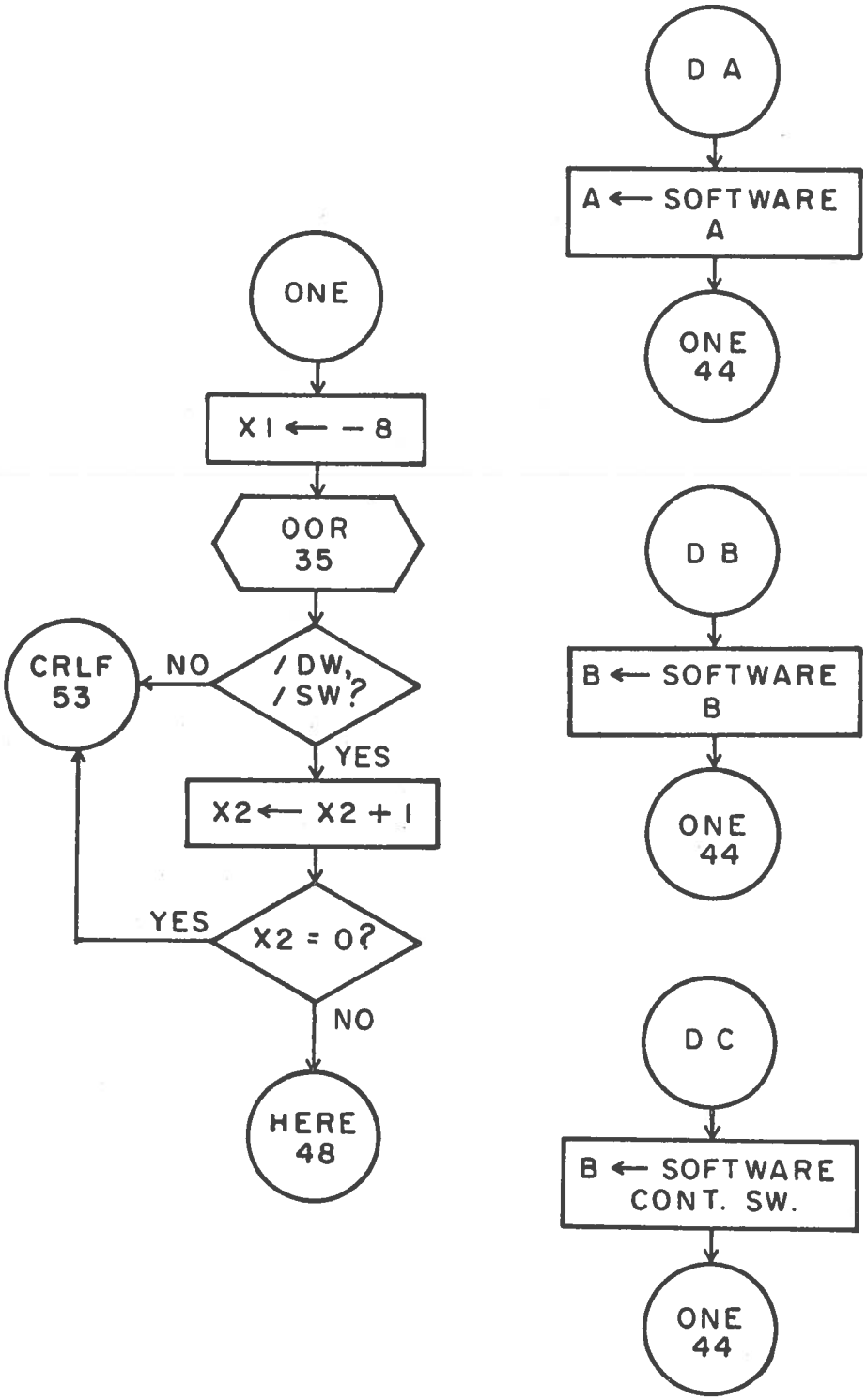

ONE
X1 ← −8
OOR 35
/DW, /SW?
NO
CRLF 53
YES
X2 ← X2 + 1
X2 = 0?
YES
NO
HERE 48
D A
A ← SOFTWARE A
ONE 44
D B
B ← SOFTWARE B
ONE 44
D C
B ← SOFTWARE CONT. SW.
ONE 44

Figure 44

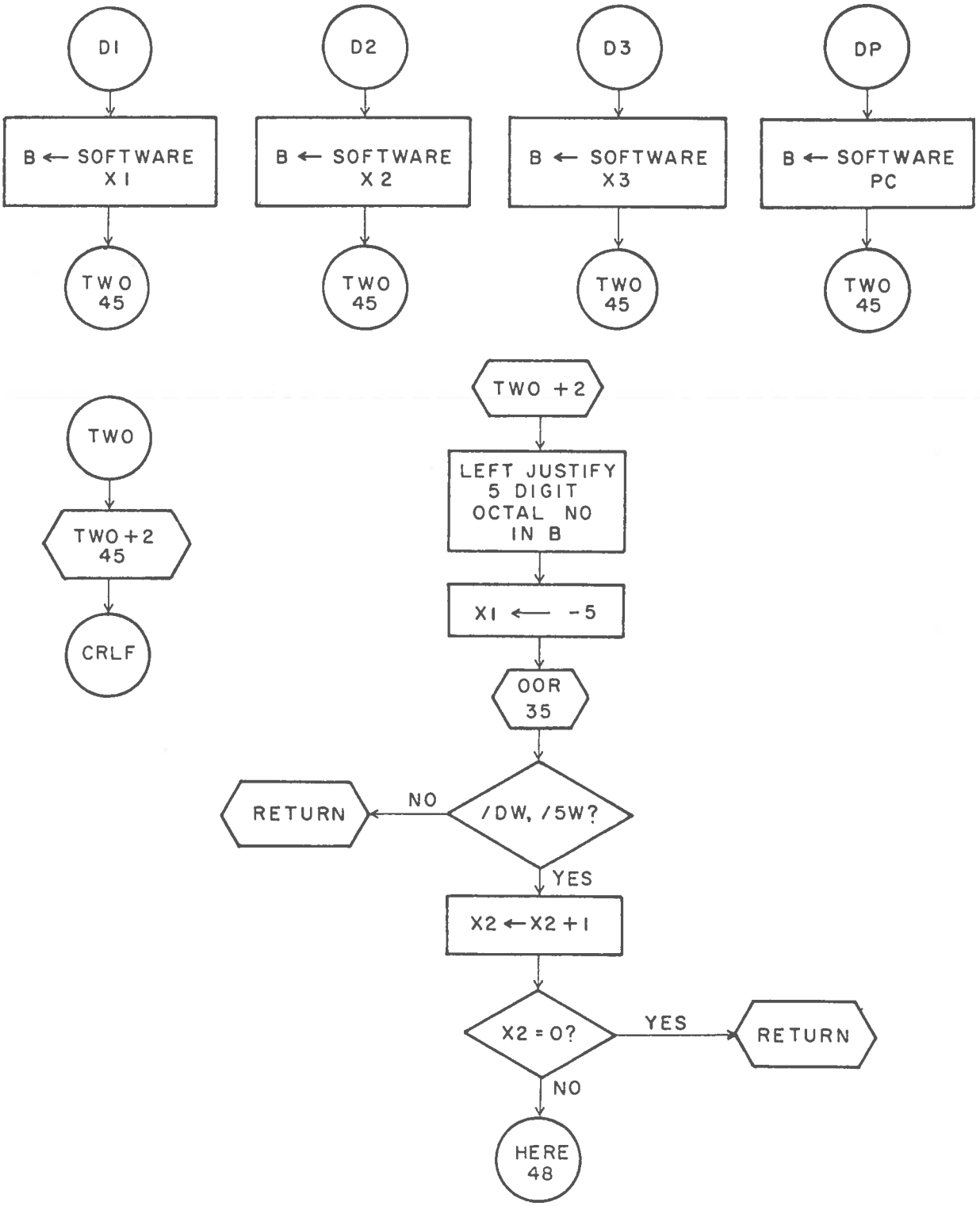

D1
B ← SOFTWARE X1
TWO 45
D2
B ← SOFTWARE X2
TWO 45
D3
B ← SOFTWARE X3
TWO 45
DP
B ← SOFTWARE PC
TWO 45
TWO
TWO+2 45
CRLF
TWO +2
LEFT JUSTIFY 5 DIGIT OCTAL NO IN B
X1 ← -5
OOR 35
/DW, /5W?
NO
RETURN
YES
X2 ← X2 + 1
X2 = 0?
YES
RETURN
NO
HERE 48

Figure 45

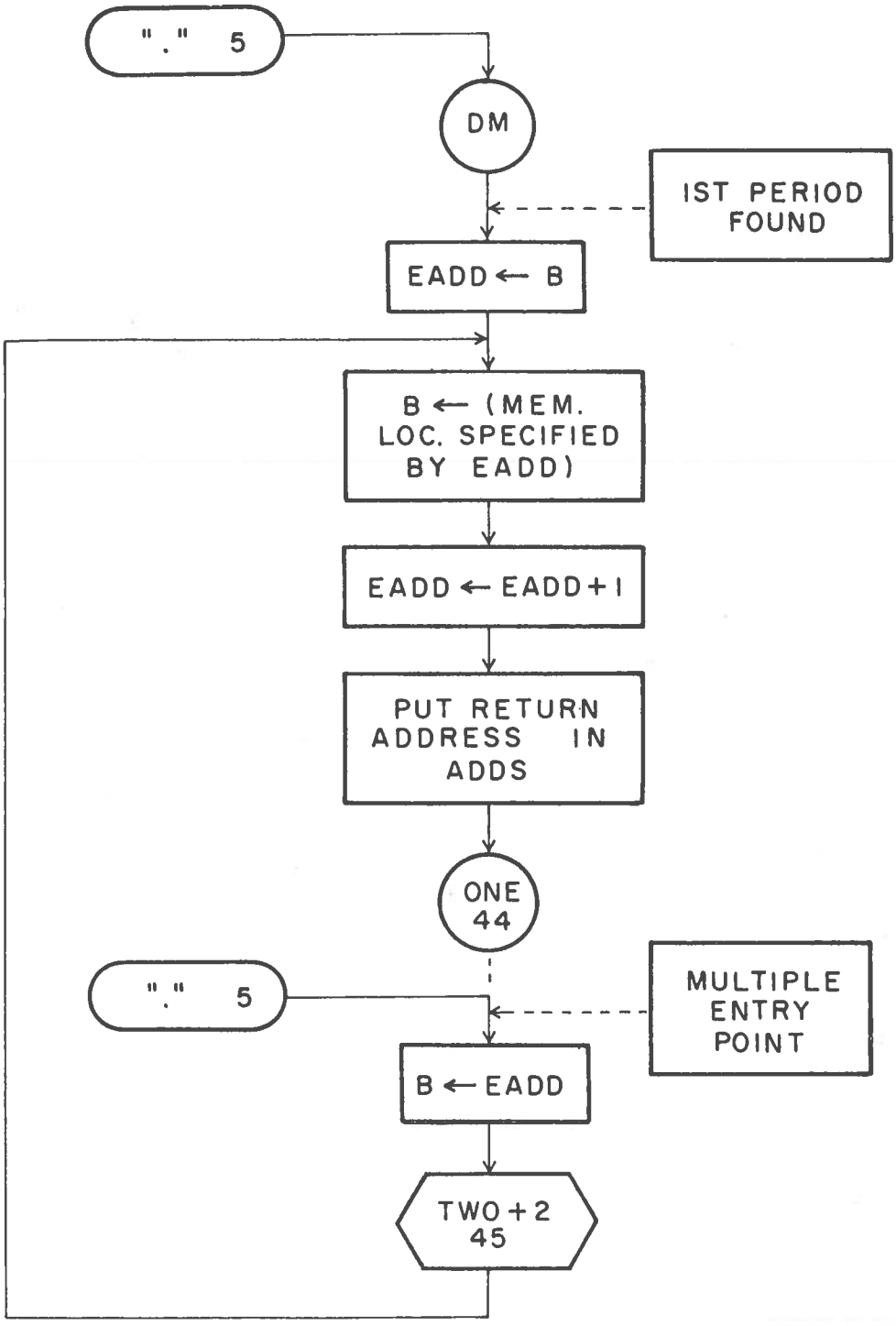

"." 5
DM
IST PERIOD FOUND
EADD ← B
B ← (MEM. LOC. SPECIFIED BY EADD)
EADD ← EADD+1
PUT RETURN ADDRESS IN ADDS
ONE 44
"." 5
MULTIPLE ENTRY POINT
B ← EADD
TWO+2 45

Figure 46

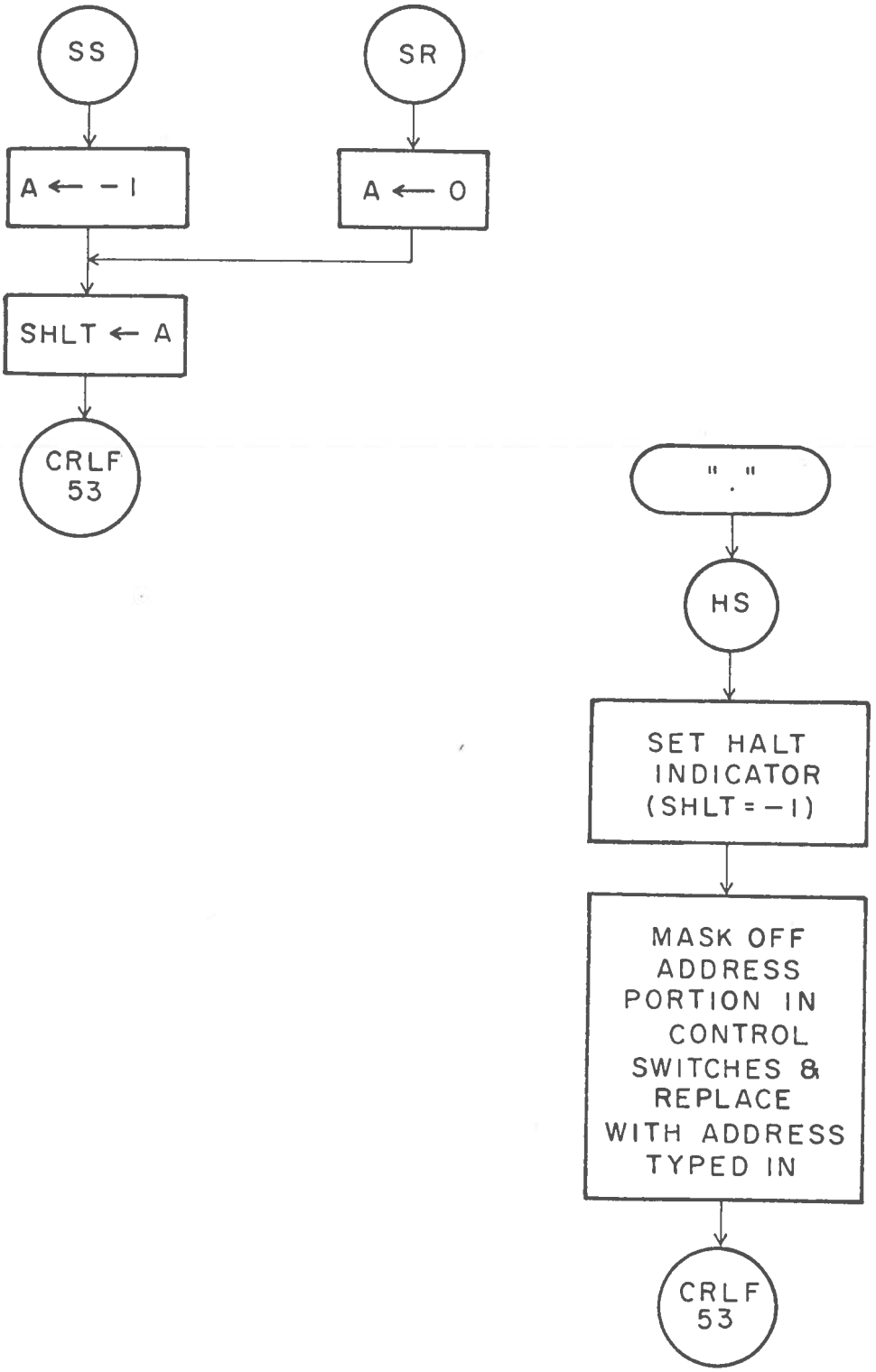

SS
SR
A ← -1
A ← 0
SHLT ← A
CRLF
53
" . "
HS
SET HALT
INDICATOR
(SHLT = -1)
MASK OFF
ADDRESS
PORTION IN
CONTROL
SWITCHES &
REPLACE
WITH ADDRESS
TYPED IN
CRLF
53

Figure 47

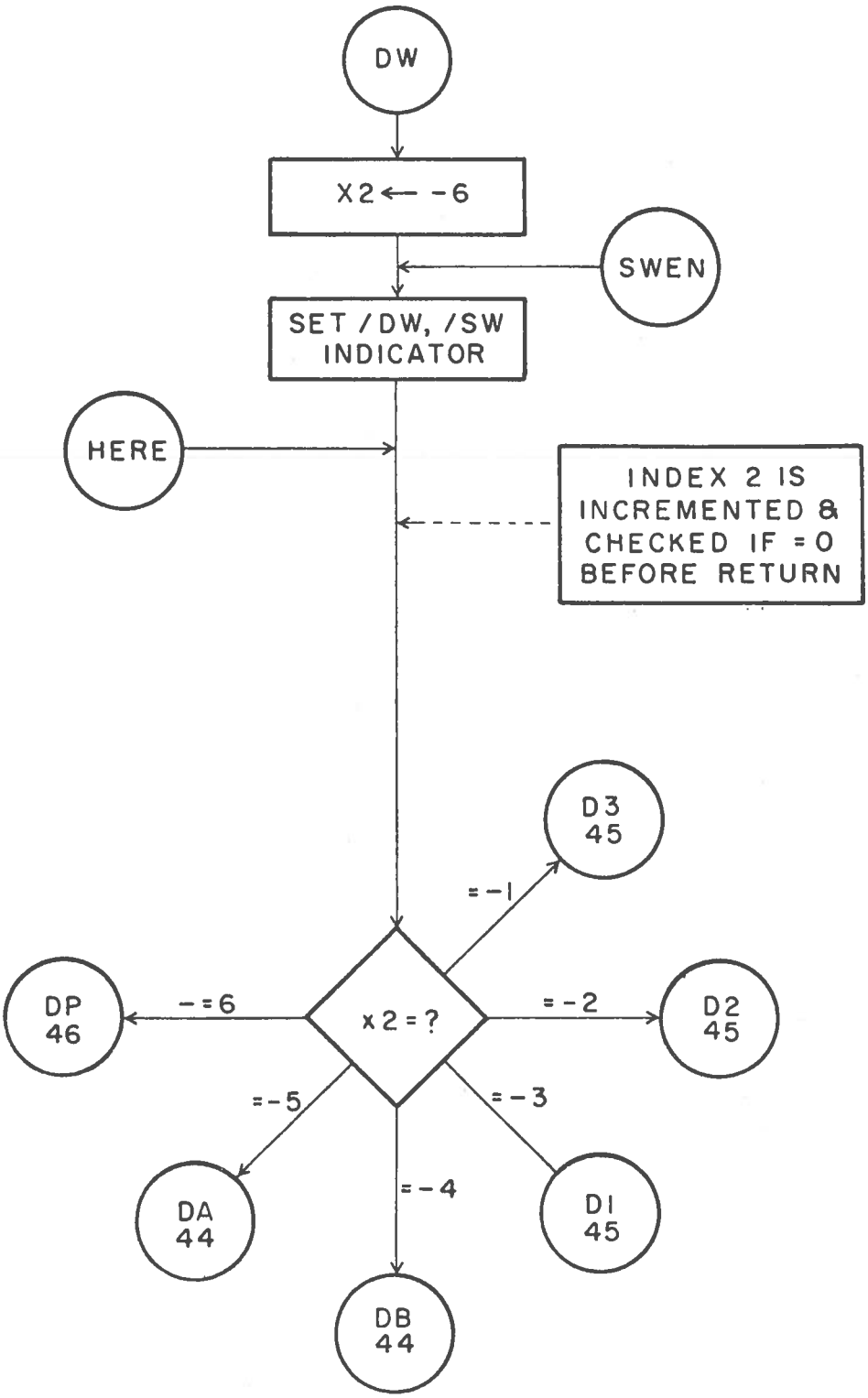

DW
X2 ← -6
SWEN
SET /DW, /SW INDICATOR
HERE
INDEX 2 IS INCREMENTED & CHECKED IF = 0 BEFORE RETURN
D3 45
= -1
DP 46
- = 6
x 2 = ?
= -2
D2 45
= -5
= -3
DA 44
= -4
DI 45
DB 44

Figure 48

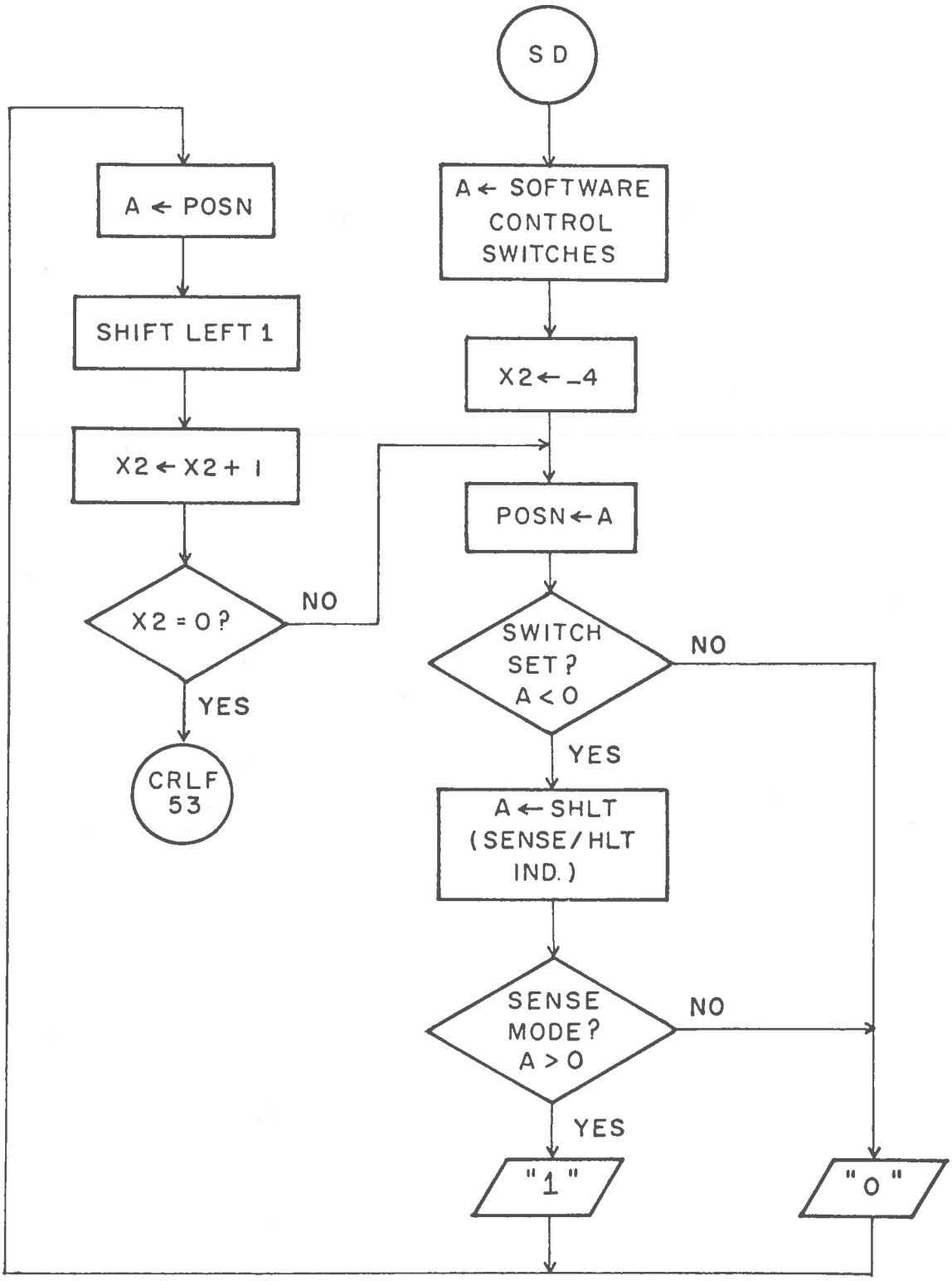
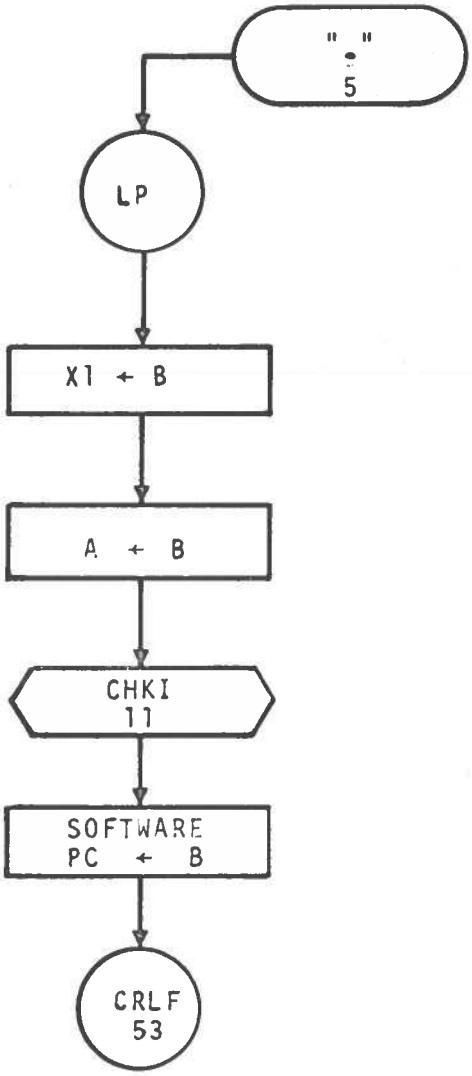
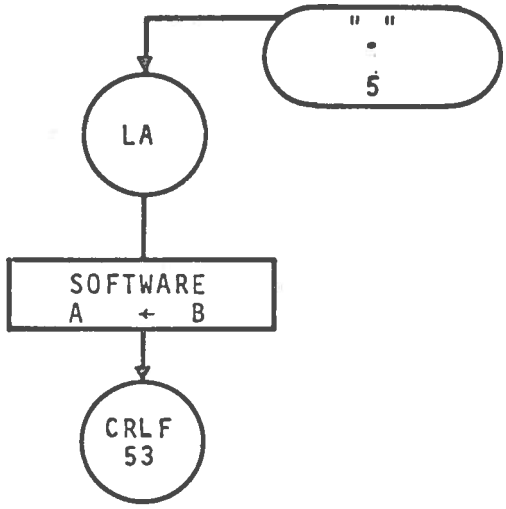
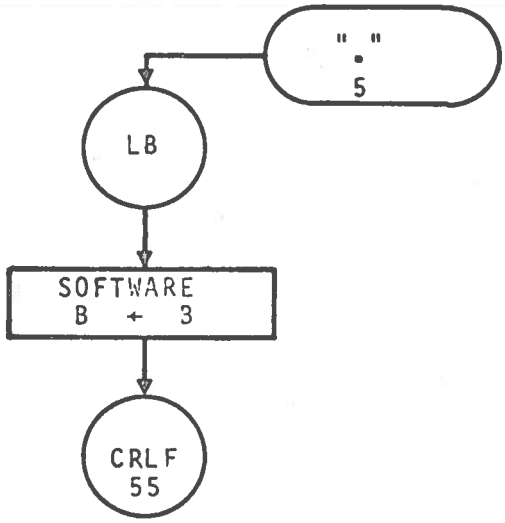

S D
A ← SOFTWARE CONTROL SWITCHES
X2 ← -4
POSN ← A
SWITCH SET? A < 0
NO
YES
A ← SHLT (SENSE/HLT IND.)
SENSE MODE? A > 0
NO
YES
"1"
"0"
A ← POSN
SHIFT LEFT 1
X2 ← X2 + 1
X2 = 0?
NO
YES
CRLF 53

Figure 49


" "
:
5
LP
X1 ← B
A ← B
CHKI
11
SOFTWARE
PC ← B
CRLF
53


" "
:
5
LA
SOFTWARE
A ← B
CRLF
53


" "
:
5
LB
SOFTWARE
B ← 3
CRLF
55

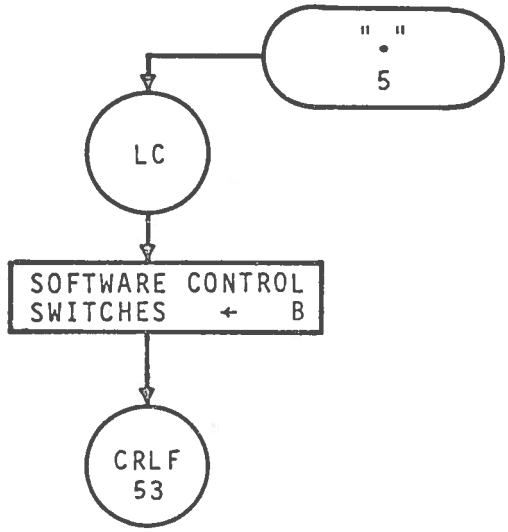

" "
:
5
LC
SOFTWARE CONTROL
SWITCHES ← B
CRLF
53

Figure 50

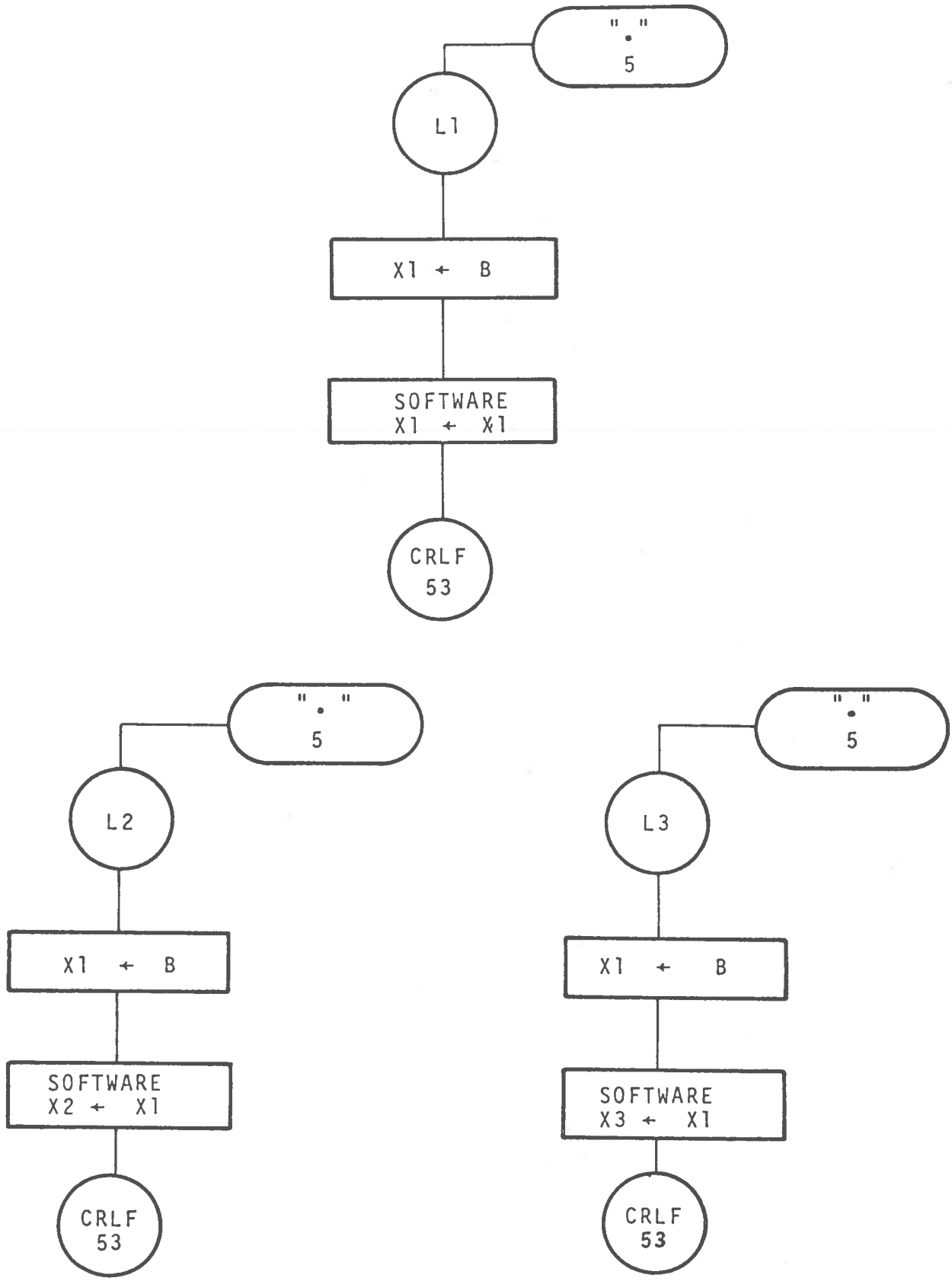

" . "
5
L1
X1 ← B
SOFTWARE
X1 ← X1
CRLF
53
" . "
5
L2
X1 ← B
SOFTWARE
X2 ← X1
CRLF
53
" . "
5
L3
X1 ← B
SOFTWARE
X3 ← X1
CRLF
53

Figure 51

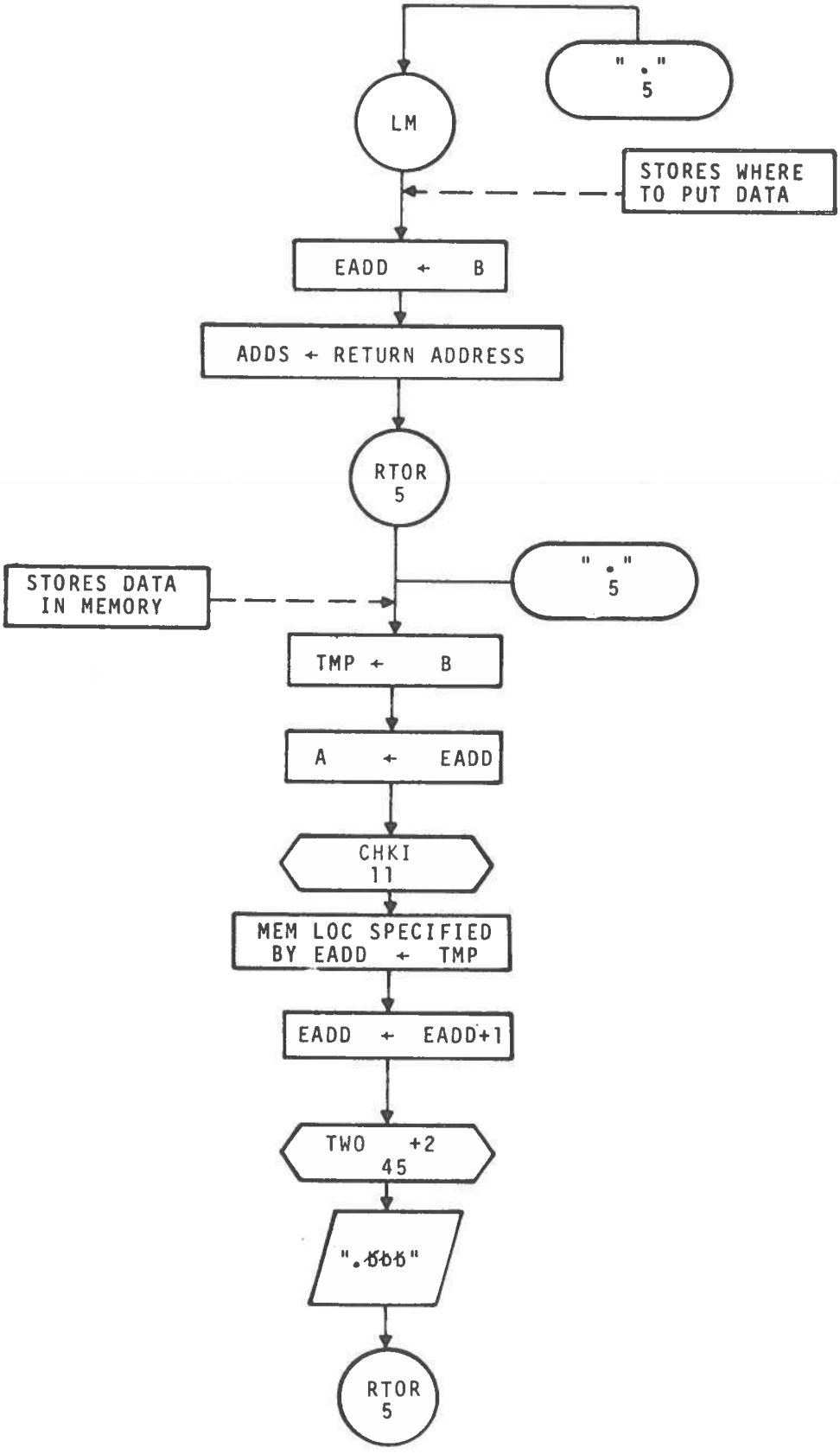

" . "
5
LM
STORES WHERE
TO PUT DATA
EADD ← B
ADDS ← RETURN ADDRESS
RTOR
5
STORES DATA
IN MEMORY
" . "
5
TMP ← B
A ← EADD
CHKI
11
MEM LOC SPECIFIED
BY EADD ← TMP
EADD ← EADD+1
TWO +2
45
".666"
RTOR
5

Figure 52

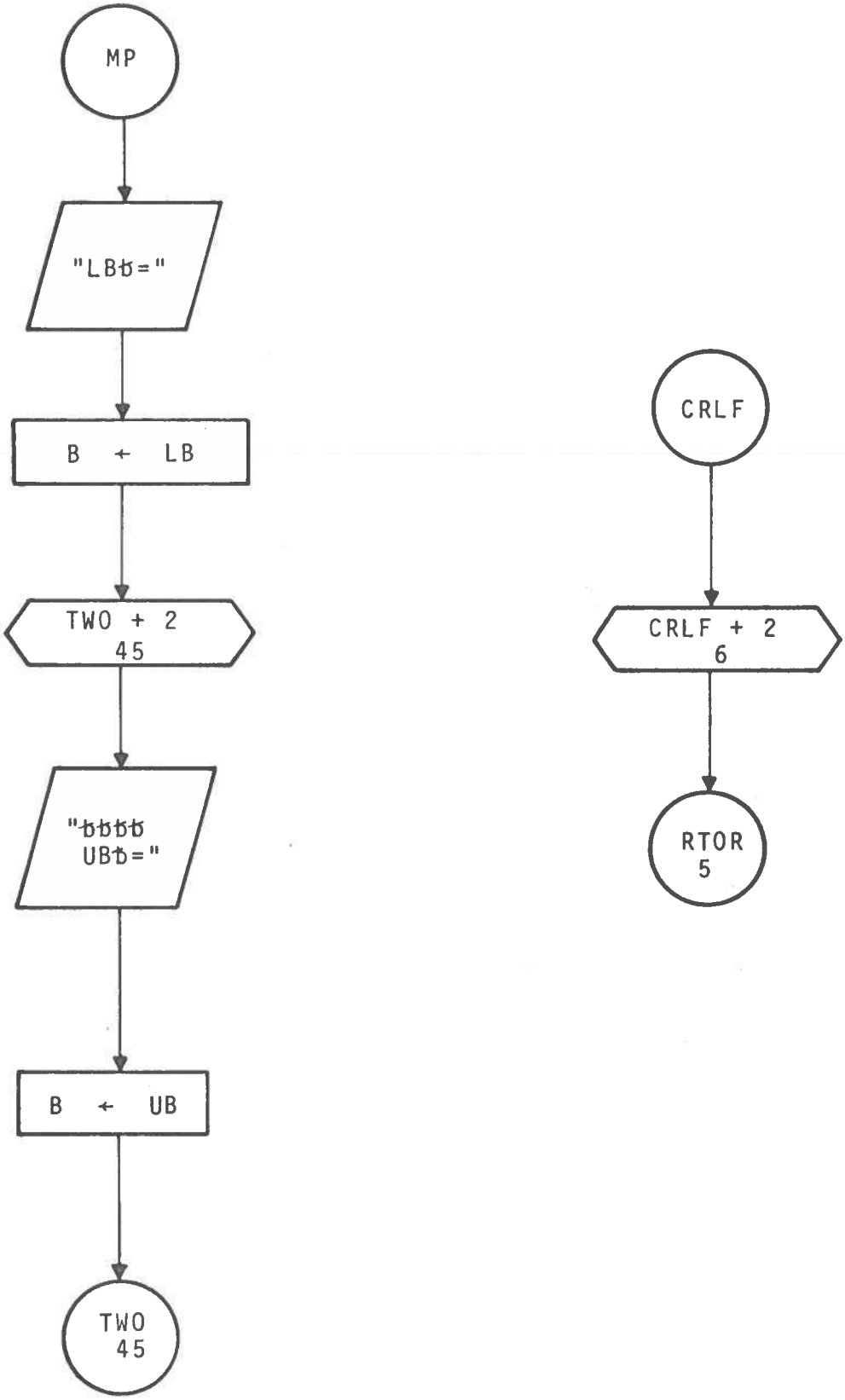


Figure 53

APPENDIX C – SOURCE LISTINGS

```
0001                          *          JULY 31,1968. R. KINREAD
0002                          *---------- INITIALIZATION ROUTINE
0003                          *---------- A AND B ENTER WITH LOWER
0004                          *---------- AND UPPER BOUNDS OF RESERVED AREA
0005                          *---------- CONTROL SWITCHES CONTAIN BEGINNING
0006                          *---------- ADDRESS OF BACKGROUND PROGRAM
0007  00000  00300000              STA  $LB
0008  00001  00100017*             LAA  C1
0009  00002  00300116              STA  '116
0010  00003  00100020*             LAA  C2
0011  00004  00300117              STA  '117
0012  00005  00400000              STB  $UB
0013  00006  05700000              LCS
0014  00007  10000001              TAI  ,1
0015  00010  13300016*             STI  GO,1
0016  00011  01310001              CEU  1,W
0017  00012  35000000              DATA '35000000
0018  00013  01710001              AOP  1,W
0019  00014  14320000              PIE  '20000,0
0020  00015  41100016*             BRU* GO
0021  00016  00000000        GO    HLT
0022                          C1    CALL INT
0023                          C2    CALL OINT
0024                                END
```

```
0002                        *          JUNE 22,1968.  PART I
0003          00000664            NAME OINT,CINT
0004          00000272            NAME CRLF,CRLF
0005          CC000022            NAME MSS,MSS
0006          00000024            NAME MESS,MESS
0007          00000136            NAME DP,DP
0008          00000121            NAME TWO,TWC
0009          00000323            NAME INT,INT
0010          00000150            NAME SHLT,SHLT
0011          00000217            NAME CTL,CTL
0012          00000210            NAME SNGL,SNGL
0013          00000647            NAME FIX,FIX
0014          00000376            NAME RTOR,RTCR
0015          00000442            NAME SLSH,SLSH
0016          00000000            NAME LO,LO
0017          00000417            NAME RFLG,RFLG
0018          00000420            NAME XFLG,XFLG
0019          00000336            NAME RTRN,RTRN
0020          00000657            NAME AA,AA
0021  00000   00000000*   LO      DAC  *            LOWER BD OF THIS PROG
0022  00001   03300000    BC      STI  $MT,0
0023  C0002   01100272*            BRU  CRLF
0024  00003   13200714*   MP      LIX  =-1,1
0025  00004   00100016*            LAA  LOBD
0026  00005   01200024*            SPB  MESS
0027  00006   00200000            LBA  $LB
0028  0C007   01200123*            SPB  TWO&2
0029  C0010   13200717*            LIX  =-2,1
0030  00011   C0100021*            LAA  UPBD
0031  0C012   01200024*            SPB  MESS
0032  0C013   00200000            LBA  $UB
0033  00014   01100121*            BRU  TWO
0034  00015   14024075            DATA ''LB =''
0035  00016   10000016*   LOBD    DAC  *,1
0036  C0017   4C404040            DATA ''    UB =''
      00020   25024075
0037  0C021   10000021*   UPBD    DAC  *,1
0038  0C022   01200024*   MSS     SPB  MESS
0039  00023   01100136*            BRU  DP
0040                        *--------- MESSAGE ROUTINE
0041                        *  -INDEX 1 ENTERS WITH -VE WORD COUNT
0042                        *  - LAST ADDRESS & 1 ENTERS IN A ACC.
0043                        *  - INDEXED WITH INDEX 1
0044                        *  - CHARACTERS PACKED 4/WORD
0045  00024   00CC0000      MESS ZZZ  **
0046  00025   033C0420*            STI  XFLG,0  RESET EXECUTE
0047  00026   23300047*            STI  XX,2
0048  00027   0C300046*            STA  ADD
0049  CC030   40200046*            LBA* ADD
0050  C0031   23200710*            LIX  =-4,2 CHARACTER COUNT
0051  00032   00000003            CLA
0052  CC033   C0006C14            FRL  6
0053  00034   01500740*            CMA  ='37
0054  CC035   00000022            NOP
```

```
0055  00036  00500737*          AMA   ='100  8 DIGIT CODE HAS  3
0056  00037  00500736*          AMA   ='200  8 DIGIT CODE HAS  2
0057  00040  00020016           LSL   16
0058  00041  01200647*          SPB   FIX
0059  00042  23400032*          IIB   MESS&6,2
0060  00043  13400030*          IIB   MESS&4,1
0061  00044  23200047*          LIX   XX,2
0062  00045  41100024*          BRU*  MESS
0063  00046  00000000     ADD   HLT
0064  00047  00000000     XX    HLT
0065                       *
0066                       *--------- DISPLAY BLOCK
0067                       *
0068                       *--------- SUBROUTINE TO OUTPUT AN OCTAL NO.
0069                       *--------- X1 CONTAINS -VE DIGIT COUNT
0070                       *--------- NUMBER LEFT-JUSTIFIED IN B
0071  00050  00000000     OOR   ZZZ   **
0072  00051  23300047*          STI   XX,2
0073  00052  00100712*          LAA   ='50000000  BLANK
0074  00053  23200735*          LIX   =-3,2
0075  00054  01200647*          SPB   FIX
0076  00055  23400054*          IIB   *-1,2   OUTPUT 3 BLANKS
0077  00056  23200047*          LIX   XX,2
0078                       *--------- OUTPUT OCTAL NO. IN B
0079  00057  00100734*          LAA   ='26
0080  00060  00023014           FRL   19
0081  00061  01200647*          SPB   FIX
0082  00062  00003014           FRL   3
0083  00063  02700733*          MAA   ='1777777
0084  00064  03000732*          MOA   ='54000000  PUT '26 AT START
0085  00065  13400061*          IIB   *-4,1
0086  00066  41100050*          BRU*  OOR
0087                       *--------- DISPLAY A
0088  00067  00200000     DA    LBA   $A
0089  00070  13200715*     ONE   LIX   =-8,1
0090  00071  01200050*          SPB   OOR
0091  00072  03500105*          SMP   WKS  /DW,/SW0
0092  00073  23400104*          IIB   HERE,2
0093  00074  01100272*          BRU   CRLF
0094                       *--------- DISPLAY B
0095  00075  00200000     DB    LBA   $B
0096  00076  01100070*          BRU   ONE
0097                       *--------- DISPLAY CONTROL SWITCHES
0098  00077  00200217*     DC    LBA   CTL
0099  00100  01100070*          BRU   ONE
0100                       *--------- DISPLAY PC,A,B,X1,X2,X3
0101  00101  23200727*     DW    LIX   =-6,2
0102  00102  00100714*     SWEN  LAA   =-1    ENTRY FOR /SW
0103  00103  00300105*          STA   WKS
0104  00104  61100637*     HERE  BRU*  REFD,2
0105  00105  00000000     WKS   HLT
0106                       *--------- DISPLAY MEMORY
0107  00106  00400000     DM    STB   $EADD      FOUND
0108  00107  40200000           LBA*  $EADD
```

```
0109  00110  01400000          IMS   $EADD
0110  00111  00100114*         LAA   *&3
0111  00112  00300423*         STA   ADDS
0112  00113  01100070*         BRU   ONE
0113  00114  00000115*         DAC   *&1
0114  00115  00200000          LBA   $EADD     .  FOUND
0115  00116  01200123*         SPB   TWO&2
0116  00117  01100107*         BRU   DM&1
0117                        *--------- DISPLAY INDEX 1
0118  00120  00200000    D1    LBA   $X1
0119  00121  01200123*   TWO   SPB   *&2
0120  00122  01100272*         BRU   CRLF
0121  00123  00000000          HLT        SUBROUTINE ENTRY
0122  00124  00011017          FLL   9  LEFT JUSTIFY 5-DIGIT NO. IN B
0123  00125  13200730*         LIX   =-5,1  5 DIGITS
0124  00126  01200050*         SPB   OOR
0125  00127  03500105*         SMP   WKS   /DW,/SWO
0126  00130  23400104*         IIB   HERE,2  YES
0127  00131  41100123*         BRU*  TWO&2
0128                        *--------- DISPLAY INDEX 2
0129  00132  00200000    D2    LBA   $X2
0130  00133  01100121*         BRU   TWO
0131                        *--------- DISPLAY INDEX 3
0132  00134  00200000    D3    LBA   $X3
0133  00135  01100121*         BRU   TWO
0134                        *--------- DISPLAY PROGRAM COUNTER
0135  00136  00200000    DP    LBA   $PC
0136  00137  01100121*         BRU   TWO
0137                        *
0138                        *--------- SENSE/HALT BLOCK
0139                        *
0140                        *--------- LOAD HALT SWITCHES
0141  00140  00100714*   HS    LAA   =-1    .  FOUND
0142  00141  00300150*         STA   SHLT  SENSE/HALT INDICATOR
0143  00142  00100217*         LAA   CTL
0144  00143  02700707*         MAA   ='77700000  MASK OFF ADDRESS
0145  00144  00300217*         STA   CTL
0146  00145  00000004          TBA
0147  00146  03100217*         AAM   CTL
0148  00147  01100272*         BRU   CRLF
0149  00150  00000000    SHLT  HLT        &VE SENSE,-VE HALT,0 NEITHER
0150                        *--------- SENSE SWITCH DISPLAY
0151  00151  00100217*   SD    LAA   CTL
0152  00152  23200710*         LIX   =-4,2
0153  00153  00300202*         STA   POSN
0154  00154  02400162*         BAP   *&6
0155  00155  00100150*         LAA   SHLT
0156  00156  02300162*         BAN   *&4
0157  00157  02200162*         BAZ   *&3
0158  00160  00100177*         LAA   ON&1
0159  00161  01100163*         BRU   *&2
0160  00162  00100201*         LAA   OFF&1
0161  00163  13200714*         LIX   =-1,1
0162  00164  01200024*         SPB   MESS
```

```
0163  00165  00100202*          LAA   POSN
0164  00166  00001016           LSL   1
0165  00167  23400153*          IIB   SD&2,2
0166  00170  01100272*          BRU   CRLF
0167                           *--------- SET SENSE MODE
0168  00171  00100731*  SS      LAA   =1
0169  00172  00300150*          STA   SHLT
0170  00173  01100272*          BRU   CRLF
0171                           *--------- RESET SENSE/HALT MODE
0172  00174  00000003     SR    CLA         RESET IND.
0173  00175  01100172*          BRU   SS&1
0174  00176  40404061     ON    DATA  ''    1''
0175  00177  10000177*          DAC   *,1
0176  00200  40400060     OFF   DATA  ''    0''
0177  00201  10000201*          DAC   *,1
0178  00202  00000000     POSN  HLT
0179                           *
0180                           *--------- SINGLE CYCLE BLOCK
0181                           *
0182                           *--------- OUTPUT A,B,X1,X2,X3
0183  00203  00100714*  SW      LAA   =-1
0184  00204  00300210*          STA   SNGL  SET SINGLE CYCLE IND.
0185  00205  01100272*          BRU   CRLF
0186  00206  23200730*          LIX   =-5,2       FOUND
0187  00207  01100102*          BRU   SWEN
0188  00210  00000000     SNGL  HLT
0189                           *--------- LOAD BLOCK
0190                           *
0191                           *--------- LOAD A
0192  00211  00400000     LA    STB   $A
0193  00212  01100272*          BRU   CRLF
0194                           *--------- LOAD B
0195  00213  00400000     LB    STB   $B
0196  00214  01100272*          BRU   CRLF
0197                           *--------- LOAD CONTROL SWITCHES
0198  00215  00400217*  LC      STB   CTL
0199  00216  01100272*          BRU   CRLF
0200  00217  00000000     CTL   HLT
0201                           *--------- LOAD PROGRAMME COUNTER
0202  00220  10000002     LP    TBI   ,1
0203  00221  00000004           TBA
0204  00222  01200000           SPB   $CHK1
0205  00223  13300000           STI   $PC,1
0206  00224  01100272*          BRU   CRLF
0207                           *--------- LOAD MEMORY
0208  00225  00400000     LM    STB   $EADD   1ST  .  FOUND
0209  00226  00100231*          LAA   *&3
0210  00227  00300423*          STA   ADDS
0211  00230  01100376*          BRU   RTOR   WAIT FOR 2ND  .
0212  00231  00000232*          DATA  *&1
0213  00232  00400303*          STB   TMP  2ND  .  FOUND
0214  00233  00100000           LAA   $EADD
0215  00234  01200000           SPB   $CHK1
0216  00235  00100303*          LAA   TMP
```

```
0217  00236  403C0000          STA*  $EADD
0218  00237  014C0000          IMS   $EADD
0219  00240  01200274*         SPB   CRLF&2
0220  00241  002C0000          LBA   $EADD
0221  00242  01200123*         SPB   TWO&2
0222  00243  13200714*         LIX   =-1,1
0223  00244  001C0250*         LAA   EASY
0224  00245  01200024*         SPB   MESS
0225  00246  011C0376*         BRU   RTOR
0226  00247  564C4040          DATA  '',''
0227  00250  100C0250*  EASY   DAC   *,1
0228                           *--------- LOAD INDEX 1
0229  00251  10000002  L1     TBI   ,1
0230  00252  133C0000          STI   $X1,1
0231  00253  01100272*         BRU   CRLF
0232                           *--------- LOAD INDEX 2
0233  00254  100C0002  L2     TBI   ,1
0234  00255  133C0000          STI   $X2,1
0235  00256  01100272*         BRU   CRLF
0236                           *--------- LOAD INDEX 3
0237  00257  10000002  L3     TBI   ,1
0238  00260  13300000          STI   $X3,1
0239  00261  01100272*         BRU   CRLF
0240                           *--------- CLEAR A,B,X1,X2,X3   RESET
0241                           *--------- SENSE HALT   CONTROL SWITCHES
0242                           *--------- AND OVERFLOW IND.
0243  00262  00000003  CL     CLA
0244  00263  132C0727*         LIX   =-6,1
0245  00264  103C0000          STA   $PC,1
0246  00265  134C0264*         IIB   *-1,1
0247  00266  00300217*         STA   CTL
0248  00267  003C0000          STA   $FLOW
0249  00270  033C0000          STI   $MT,0       CLEAR BUFFER
0250  00271  01100174*         BRU   SR
0251                           *--------- OUTPUT CARRIAGE RETURN LINE FEED
0252  00272  01200274*  CRLF   SPB   *&2
0253  00273  011C0376*         BRU   RTOR
0254  00274  000C0000          HLT         ENTRY TO SUBROUTINE
0255  00275  03300105*         STI   WKS,0  RESET /DW,/SW IND.
0256  00276  00100726*         LAA   ='43305CC0   CR/LF
0257  00277  01200647*         SPB   FIX
0258  003C0  00010016          LSL   8
0259  003C1  023C0277*         BAN   *-2
0260  00302  41100274*         BRU*  CRLF&2
0261  00303  00000000  TMP    HLT
0262                           *--------- SUBROUTINE TO SAVE REGISTERS ON
0263                           *--------- EXTERNAL INTERRUPT
0264  003C4  000C0000  COMP   ZZZ   **
0265  003C5  015C0000*         CMA   LO    LOW ADD OF THIS PROG.
0266  003C6  01100313*         BRU   *&5
0267  00307  00C00022          NOP
0268  00310  01500000          CMA   $U0   HI ADD OF THIS PROG
0269  00311  000C0C22          NOP
0270  00312  411C0304*         BRU*  COMP
```

```
0271  00313  00300421*          STA  SAVE
0272  00314  00100303*          LAA  TMP
0273  00315  00300425*          STA  AACC
0274  00316  00400426*          STB  BACC
0275  00317  13300427*          STI  IND1,1
0276  00320  23300430*          STI  IND2,2
0277  00321  33300431*          STI  IND3,3
0278  00322  41100304*          BRU* COMP
0279                       *
0280                       *---------  INPUT INTERRUPT
0281                       *
0282  00323  00000000     INT   ZZZ  **
0283  00324  00300303*          STA  TMP
0284  00325  03500420*          SMP  XFLG  EXECUTING INSTRO
0285  00326  01100330*          BRU  *&2   YES
0286  00327  01100334*          BRU  *&5   NO
0287  00330  00100714*          LAA  =-1
0288  00331  00300417*          STA  RFLG  SET REQUEST
0289  00332  00100303*          LAA  TMP
0290  00333  41100323*          BRU* INT
0291  00334  00100323*          LAA  INT
0292  00335  01200304*          SPB  COMP
0293  00336  03300417*    RTRN  STI  RFLG,0  RESET REQUEST
0294  00337  03500000          SMP  $GOON    READ 20 CHARO
0295  00340  01100411*          BRU  FIND    YES
0296  00341  03500000          SMP  $INP         READ FROM KYBDO
0297  00342  01100000          BRU  $IN    YES
0298  00343  03500000          SMP  $RCNT        READ FROM TAPEO
0299  00344  01100000          BRU  $FILL        YES
0300  00345  01720001          AIP  1
0301  00346  01100345*          BRU  *-1
0302  00347  03300000    FRET  STI  $GOON,0  RESET BUFFER TO FILL
0303  00350  01500725*          CMA  ='257     /0
0304  00351  01100353*          BRU  *&2
0305  00352  01100442*          BRU  SLSH  SLASH ROUTINE
0306  00353  01500724*          CMA  ='377  RUBOUTO
0307  00354  01100356*          BRU  *&2
0308  00355  01100461*          BRU  ROUT  RUBOUT ROUTINE
0309  00356  00020016          LSL  16
0310  00357  01200511*          SPB  DO
0311  00360  00002016          LSL  2   TRUNCATE
0312  00361  03500210*          SMP  SNGL  SINGLE CYCLE IND.
0313  00362  01100475*          BRU  SING  SINGLE CYCLE
0314  00363  01500723*          CMA  ='56000000        0
0315  00364  01100366*          BRU  *&2
0316  00365  01100405*          BRU  PERD         FOUND
0317  00366  03500422*          SMP  INST  INSTRUCTION INDICATOR
0318  00367  01100516*          BRU  ISTR
0319  00370  01500722*          CMA  ='60000000   TEST IF NUMBER
0320  00371  01100376*          BRU  RTOR
0321  00372  01100400*          BRU  *&6   =00
0322  00373  01500721*          CMA  ='67000000
0323  00374  01100400*          BRU  *&4   +7
0324  00375  01100400*          BRU  *&3   =7
```

```
0325  00376  01200433*  RTOR  SPB   RSTR  RESTORE REGISTERS
0326  00377  03600421*        PIR   SAVE
0327  00400  00003016        LSL   3    TRUNCATE
0328  00401  00200424*        LBA   NUMB  CONTAINS PREV. DIGITS
0329  00402  00003014        FRL   3    PACK
0330  00403  00400424*        STB   NUMB
0331  00404  01100376*        BRU   RTOR
0332  00405  00200424*  PERD  LBA   NUMB
0333  00406  00000003        CLA
0334  00407  00300424*        STA   NUMB
0335  00410  41100423*        BRU*  ADDS
0336                   *--------- IF BLANK,CONTINUE - OTHERWISE,STOP
0337  00411  01720001  FIND  AIP   1
0338  00412  01100411*        BRU   *-1
0339  00413  01500720*        CMA   ='240
0340  00414  01100347*        BRU   FRET
0341  00415  01100000        BRU   $AOK    BLANK - FILL BUFFER
0342  00416  01100347*        BRU   FRET
0343  00417  00000000  RFLG  DATA  0    NO REQUEST
0344  00420  00000000  XFLG  DATA  0    NOT EXECUTING
0345  00421  00000000  SAVE  HLT         WHERE INTERRUPTED OUTSIDE
0346  00422  00000000  INST  HLT       INSTRUCTION IND.
0347  00423  00000115*  ADDS  DATA  DM&7  GO WHEN  .  FOUND
0348  00424  00000000  NUMB  HLT       DIGITS TYPED IN
0349  00425  00000000  AACC  HLT
0350  00426  00000000  BACC  HLT
0351  00427  00000000  IND1  HLT
0352  00430  00000000  IND2  HLT
0353  00431  00000000  IND3  HLT
0354  00432  00000000  INS   HLT
0355  00433  00000000  RSTR  ZZZ   ** ROUTINE TO RESTORE REGS.
0356  00434  00100425*        LAA   AACC
0357  00435  00200426*        LBA   BACC
0358  00436  13200427*        LIX   IND1,1
0359  00437  23200430*        LIX   IND2,2
0360  00440  33200431*        LIX   IND3,3
0361  00441  41100433*        BRU*  RSTR
0362  00442  00300432*  SLSH  STA   INS    /  ROUTINE
0363  00443  00020016        LSL   16
0364  00444  01200511*        SPB   DO
0365  00445  00100717*        LAA   =-2
0366  00446  00300460*        STA   CTR    CHARACTER COUNTER
0367  00447  03300210*        STI   SNGL,0  RESET SINGLE CYCLE
0368  00450  03300000        STI   $INP,0  RESET INPUT
0369  00451  03300105*        STI   WKS,0  RESET /DW,/SW IND.
0370  00452  03300000        STI   $OPUT,0 RESET OUTPUT
0371  00453  03300424*        STI   NUMB,0  CLEAR NUMBER
0372  00454  00100714*        LAA   =-1
0373  00455  00300422*        STA   INST  SET INSTRUCTION IND.
0374  00456  04340000        PID   '40000,0
0375  00457  01100376*        BRU   RTOR
0376  00460  00000000  CTR   HLT
0377  00461  13200717*  ROUT  LIX   =-2,1  RUBOUT ROUTINE (START)
0378  00462  00100474*        LAA   SING-1
```

```
0379  00463  01200024*          SPB   MESS
0380  00464  00200000           LBA   $PC
0381  00465  01200123*          SPB   TWO&2
0382  00466  01200274*          SPB   CRLF&2
0383  00467  00000003           CLA
0384  00470  00300210*          STA   SNGL
0385  00471  01100000           BRU   $TRAP
0386  00472  23240122           DATA  ''START AT''
      00473  24400124
0387  00474  10000474*          DAC   *,1
0388  00475  01500716*    SING  CMA   ='40000000  SINGLE CYCLE
0389  00476  01100376*          BRU   RTOR
0390  00477  01100501*          BRU   *&2
0391  00500  01100376*          BRU   RTOR
0392  00501  00100000           LAA   $PC
0393  00502  01200000           SPB   $CHK1  PC RETURNS IN B
0394  00503  01200123*          SPB   TWO&2
0395  00504  40200000           LBA*  $PC
0396  00505  13200715*          LIX   =-8,1
0397  00506  01200050*          SPB   OOR
0398  00507  01200000           SPB   $EXEQ  EXECUTE ROUTINE
0399  00510  01100206*          BRU   SW&3
0400  00511  00000000     DO    ZZZ   **
0401  00512  00200714*          LBA   =-1
0402  00513  00400705*          STB   HIGH
0403  00514  01200647*          SPB   FIX
0404  00515  41100511*          BRU*  DO
0405  00516  00200432*    ISTR  LBA   INS  INSTRUCTION BLOCK
0406  00517  00006014           FRL   6  PACK
0407  00520  00400432*          STB   INS
0408  00521  01400460*          IMS   CTR
0409  00522  01100376*          BRU   RTOR  BRANCH IF NOT LAST CHAR.
0410  00523  00000004           TBA
0411  00524  00006016           LSL   6
0412  00525  00500713*          AMA   ='40  PUT BLANK IN
0413  00526  00300432*          STA   INS
0414  00527  00100712*          LAA   ='50000000   BLANK
0415  00530  01200511*          SPB   DO
0416  00531  03300422*          STI   INST,0  RESET INSTRUCTION IND.
0417  00532  00100432*          LAA   INS
0418                       *--------- DECODE THE COMMAND
0419  00533  13200711*    DCOD  LIX   =-26,1     COUNT PLACE IN TABLE
0420  00534  11500615*          CMA   TABL,1
0421  00535  01100537*          BRU   *&2
0422  00536  01100544*          BRU   TEST
0423  00537  13400534*          IIB   *-3,1
0424  00540  00100562*          LAA   PLCE
0425  00541  13200710*          LIX   =-4,1  4 WORD MESSAGE
0426  00542  01200024*          SPB   MESS
0427  00543  01100272*          BRU   CRLF
0428  00544  13300047*    TEST  STI   XX,1
0429  00545  00100707*          LAA   ='77700000
0430  00546  03100047*          AAM   XX
0431  00547  00100706*          LAA   =-17
```

```
0432  00550  01500047*        CMA   XX
0433  00551  51100647*        BRU*  THER,1      NO NUMBER FOLLOWS
0434  00552  00000022         NOP
0435  00553  10100647*        LAA   THER,1      NUMBER FOLLOWS
0436  00554  00300423*        STA   ADDS
0437  00555  01100376*        BRU   RTOR
0438  00556  11141405    CERR DATA  ''ILLEGAL COMMAND''
      00557  07011440
      00560  03171515
      00561  01160440
0439  00562  10000562*  PLCE DAC   *,1
0440                     *--------- LIST OF COMMANDS
0441  00563  57142040         DATA  ''/LP''
0442  00564  57140140         DATA  ''/LA''
0443  00565  57140240         DATA  ''/LB''
0444  00566  57146140         DATA  ''/L1''
0445  00567  57146240         DATA  ''/L2''
0446  00570  57146340         DATA  ''/L3''
0447  00571  57140340         DATA  ''/LC''
0448  00572  57141540         DATA  ''/LM''
0449  00573  57041540         DATA  ''/DM''
0450  00574  57102340         DATA  ''/HS''
0451  00575  57232340         DATA  ''/SS''
0452  00576  57232240         DATA  ''/SR''
0453  00577  57042040         DATA  ''/DP''
0454  00600  57040140         DATA  ''/DA''
0455  00601  57040240         DATA  ''/DB''
0456  00602  57046140         DATA  ''/D1''
0457  00603  57046240         DATA  ''/D2''
0458  00604  57046340         DATA  ''/D3''
0459  00605  57040340         DATA  ''/DC''
0460  00606  57042740         DATA  ''/Dw''
0461  00607  57232740         DATA  ''/SW''
0462  00610  57152040         DATA  ''/MP''
0463  00611  57230440         DATA  ''/SD''
0464  00612  57031440         DATA  ''/CL''
0465  00613  57171340         DATA  ''/OK''
0466  00614  57020340         DATA  ''/BC''
0467         00615       TABL BES   0
0468                     *--------- TABLE OF STARTING ADDRESSES
0469  00615  00000220*         DATA  LP
0470  00616  00000211*         DATA  LA
0471  00617  00000213*         DATA  LB
0472  00620  00000251*         DATA  L1
0473  00621  00000254*         DATA  L2
0474  00622  00000257*         DATA  L3
0475  00623  00000215*         DATA  LC
0476  00624  00000225*         DATA  LM
0477  00625  00000106*         DATA  DM
0478  00626  00000140*         DATA  HS
0479  00627  00000171*         DATA  SS
0480  00630  00000174*         DATA  SR
0481  00631  00000136*         DATA  DP
0482  00632  00000067*         DATA  DA
```

```
0483  00633  00000075*          DATA DB
0484  00634  00000120*          DATA D1
0485  00635  00000132*          DATA D2
0486  00636  00000134*          DATA D3
0487  00637  00000077*  REFD    DATA DC    REFERENCE FOR /DW
0488  00640  00000101*          DATA DW
0489  00641  00000203*          DATA SW
0490  00642  00000003*          DATA MP
0491  00643  00000151*          DATA SD
0492  00644  00000262*          DATA CL
0493  00645  00000000           DATA $DATA
0494  00646  00000001*          DATA BC
0495         00647              THER BES  0
0496                            *--------- ROUTINE TO STORE REGISTERS
0497                            *--------- BEFORE RETURNING TO BACKGROUND
0498                            *--------- FOR OUTPUT
0499  00647  00000000           FIX  ZZZ  **
0500  00650  00300657*          STA  AA
0501  00651  00400660*          STB  BB
0502  00652  13300661*          STI  XX1,1
0503  00653  23300662*          STI  XX2,2
0504  00654  33300663*          STI  XX3,3
0505  00655  14340000           PIE  '40000,0
0506  00656  01100376*          BRU  RTOR
0507  00657  00000000  AA       HLT
0508  00660  00000000  BB       HLT
0509  00661  00000000  XX1      HLT
0510  00662  00000000  XX2      HLT
0511  00663  00000000  XX3      HLT
0512                            *--------- OUTPUT INTERRUPT ROUTINE
0513  00664  00000000  OINT     ZZZ  **
0514  00665  00300303*          STA  TMP
0515  00666  00100664*          LAA  OINT
0516  00667  01200304*          SPB  COMP
0517  00670  04340000           PID  '40000,0
0518  00671  00100657*          LAA  AA
0519  00672  00200660*          LBA  BB
0520  00673  13200661*          LIX  XX1,1
0521  00674  23200662*          LIX  XX2,2
0522  00675  33200663*          LIX  XX3,3
0523  00676  03500000           SMP  $OPUT   EXECUTE OUTPUTO
0524  00677  01100000           BRU  $OUT    YES
0525  00700  01700001           AOP  1
0526  00701  01100700*          BRU  *-1
0527  00702  01400705*          IMS  HIGH
0528  00703  03600647*          PIR  FIX
0529  00704  41100647*          BRU* FIX
0530  00705  00000000  HIGH     HLT
0531                            END
      00706  77777757
      00707  77700000
      00710  77777774
      00711  77777746
      00712  50000000
```

00713	0C0CCC40
00714	77777777
00715	77777770
0C716	40C00000
00717	77777776
0072C	00000240
00721	67C00000
00722	60CCC000
00723	56000000
00724	00C0377
00725	00000257
00726	43305000
00727	77777772
00730	77777773
00731	00000001
0C732	54CC0000
0C733	01777777
00734	00000026
00735	77777775
0C736	00000200
00737	00000100
0C740	00000037

```
0002                                *          JUNE 22,1968.  PART II
0003            00000236                NAME GOON,GOCN
0004            00000200                NAME ACK,ACK
0005            00000462                NAME TRAP,TRAP
0006            00000461                NAME EXEQ,EXEC
0007            00000753                NAME A,A
0008            00000754                NAME B,B
0009            00000755                NAME X0,X0
0010            00000756                NAME X1,X1
0011            00000757                NAME X2,X2
0012            00000760                NAME X3,X3
0013            00000331                NAME INP,INP
0014            00000717                NAME CHK1,CHK1
0015            00000747                NAME EADD,EADD
0016            00000761                NAME PC,PC
0017            00000130                NAME IN,IN
0018            00000315                NAME OUT,OUT
0019            00000330                NAME OPUT,OPUT
0020            00000750                NAME LB,LB
0021            00000751                NAME UB,UB
0022            00000146                NAME DATA,DATA
0023            00000636                NAME FLOW,FLOW
0024            00000234                NAME MT,MT
0025            00000206                NAME FILL,FILL
0026            00000235                NAME RCNT,RCNT
0027            00000000                NAME U0,U0
0028  00000  00001110*    U0      DATA ='77700000 HIGH ADD OF THIS PROG
0029                              *--------- BRANCH INSTRUCTIONS
0030  00001  00100752*    BRU     LAA   BOX
0031  00002  01200712*            SPB   CHCK
0032  00003  00100747*            LAA   EADD
0033  00004  00300761*            STA   PC
0034  00005  01100575*            BRU   ADVC&1
0035  00006  00100753*    BAN     LAA   A
0036  00007  02300001*            BAN   BRU
0037  00010  01100574*            BRU   ADVC
0038  00011  00100753*    BAP     LAA   A
0039  00012  02400001*            BAP   BRU
0040  00013  01100574*            BRU   ADVC
0041  00014  00100753*    BAZ     LAA   A
0042  00015  02200001*            BAZ   BRU
0043  00016  01100574*            BRU   ADVC
0044  00017  00100636*    BOF     LAA   FLOW
0045  00020  02400574*            BAP   ADVC
0046  00021  00000003            CLA
0047  00022  00300636*            STA   FLOW   RESET OVERFLOW IND.
0048  00023  01100001*            BRU   BRU
0049  00024  00100752*    SPB     LAA   BOX
0050  00025  01200712*            SPB   CHCK
0051  00026  01400761*            IMS   PC
0052  00027  00100761*            LAA   PC
0053  00030  40300747*            STA*  EADD
0054  00031  00100747*            LAA   EADD
0055  00032  00300761*            STA   PC
```

```
0056  00033  01100574*          BRU   ADVC
0057  00034  00100752*   IIB    LAA   BOX
0058  00035  00025010           RSA   21
0059  00036  02300052*          BAN   *&12    ERROR MESSAGE
0060  00037  02200574*          BAZ   ADVC
0061  00040  10000001           TAI   ,1
0062  00041  00101110*          LAA   ='77700000
0063  00042  13000755*          MOA   X1-1,1
0064  00043  00501064*          AMA   =1
0065  00044  02701053*          MAA   ='77777
0066  00045  10300755*          STA   X1-1,1
0067  00046  02200574*          BAZ   ADVC
0068  00047  00100752*          LAA   BOX
0069  00050  01200717*          SPB   CHK1
0070  00051  01100003*          BRU   BRU&2
0071  00052  13201054*          LIX   =-3,1
0072  00053  00100060*          LAA   ME5
0073  00054  01100000           BRU   $MSS
0074  00055  11160411           DATA  ''INDIRECT IIB''
      00056  22050324
      00057  40111102
0075  00060  10000060*   ME5    DAC   *,1
0076                     *
0077                     *---------- INPUT/OUTPUT BLOCK
0078                     *
0079  00061  01200444*   IO     SPB   UTST    TTY0
0080  00062  00100752*          LAA   BOX
0081  00063  02701107*          MAA   ='60000
0082  00064  13201054*          LIX   =-3,1
0083  00065  11500312*          CMA   TAB&3,1
0084  00066  01100070*          BRU   *&2
0085  00067  51100315*          BRU*  TAB1&3,1
0086  00070  13400065*          IIB   *-3,1
0087  00071  00100752*   MIP    LAA   BOX
0088  00072  01400761*          IMS   PC
0089  00073  02400077*          BAP   *&4          INDIRECT0
0090  00074  40100761*          LAA*  PC           YES
0091  00075  01200712*          SPB   CHCK
0092  00076  01100113*          BRU   NOW
0093  00077  00100761*          LAA   PC            NO
0094  00100  00300747*          STA   EADD
0095  00101  01100113*          BRU   NOW
0096  00102  00100263*   AIP    LAA   IT
0097  00103  00300747*          STA   EADD
0098  00104  00100752*          LAA   BOX
0099  00105  02701106*          MAA   ='4000       GET MERGE FLAG
0100  00106  02200113*          BAZ   NOW          NO MERGE
0101  00107  00100753*          LAA   A            MERGE
0102  00110  02701105*          MAA   ='77777400   MASK OFF LAST CHAR
0103  00111  00300753*          STA   A
0104  00112  01100114*          BRU   *&2
0105  00113  43300747*   NOW    STI*  EADD,0
0106  00114  00100752*          LAA   BOX
0107  00115  02701104*          MAA   ='14000      GET WAIT AND MERGE
```

```
0108  00116  00501103*          AMA   ='17200C1   CONSTRUCT AIP
0109  00117  00300133*          STA   IN&3
0110  00120  033C0000           STI   $XFLG,0  RESET EXECUTE
0111  00121  00100430*          LAA   COMD
0112  00122  01501066*          CMA   ='2000000   READER
0113  00123  01100125*          BRU   *&2         NO
0114  00124  01100166*          BRU   READ        YES
0115  00125  00101043*          LAA   =-1
0116  00126  00300331*          STA   INP   SET INPUT INDICATOR
0117  00127  01100000           BRU   $RTOR
0118  00130  00101043*  IN      LAA   =-1
0119  00131  003C0000           STA   $XFLG  SET EXECUTE
0120  00132  40100747*          LAA*  EADD
0121  00133  00000000           DATA  0   CONTAINS AIP
0122  00134  01100136*          BRU   *&2  NO SKIP
0123  00135  01400761*          IMS   PC   SKIP
0124  00136  01400761*          IMS   PC
0125  00137  40300747*          STA*  EADD
0126  00140  01400331*          IMS   INP         RESET INPUT IND
0127  00141  00000022           NOP
0128  00142  02701102*          MAA   ='777  GET CHAR. READ IN
0129  00143  01501100*          CMA   ='257   / 0
0130  00144  01100146*          BRU   *&2
0131  00145  011C0150*          BRU   *&3
0132  00146  03600147*  DATA    PIR   *&1
0133  00147  00000605*          DATA  ENT-2
0134  00150  00100165*          LAA   MEND
0135  00151  13201101*          LIX   =-8,1
0136  00152  01200000           SPB   $MESS
0137  00153  00101100*          LAA   ='257    /
0138  00154  01100000           BRU   $SLSH
0139  00155  57402205           DATA  ''/ READ IN - IF CATA TYPE  OK ''
      00156  01044011
      00157  164C5540
      00160  11064004
      00161  01240140
      00162  24312005
      00163  40421713
      00164  42404040
0140  00165  10000165*  MEND    DAC   *,1
0141  00166  03500234*  READ    SMP   MT          BUFFER EMPTYO
0142  00167  01100221*          BRU   AMIP         NO
0143  00170  00101043*          LAA   =-1
0144  00171  00300236*          STA   GOON    SET FILL BUFFER
0145  00172  00100177*          LAA   ME9
0146  00173  13201043*          LIX   =-1,1
0147  00174  012C0000           SPB   $MESS
0148  00175  011C0000           BRU   $CRLF
0149  00176  23200305           DATA  ''SPCE''
0150  00177  10000177*  ME9     DAC   *,1
0151  C020C  00101077*  ACK     LAA   =-20  FILL BUFFER
0152  00201  00300235*          STA   RCNT        POINTER & FLAG
0153  00202  013C0001           CEU   1
0154  00203  02000000           DATA  '2000000  ENABLE RDR - DISABLE KYBD
```

```
0155  00204  01100202*          BRU   *-2
0156  00205  011C0000           BRU   $RTOR
0157  00206  13200235*   FILL   LIX   RCNT,1      ENTRY FROM INTERRUPT
0158  00207  417600C1           MIP*  1
0159  00210  10000263*          DAC   BUFF,1
0160  00211  01100207*          BRU   *-2
0161  00212  01400235*          IMS   RCNT         INCREMENT PTR
0162  00213  011C0000           BRU   $RTOR        NOT FULL
0163  00214  013C0001           CEU   1      FULL - ENABLE KYBD
0164  00215  010CC0C0           DATA  '1000000
0165  00216  01100214*          BRU   *-2
0166  00217  00101077*          LAA   =-20
0167  00220  00300234*          STA   MT     INITIALIZE PTR
0168  00221  00101043*   AMIP   LAA   =-1
0169  00222  003C0000           STA   $XFLG  RESET EXECUTE
0170  00223  13200234*          LIX   MT,1
0171  00224  10100263*          LAA   BUFF,1
0172  00225  431C0747*          AAM*  EADD         CHAR READ IN
0173  00226  01400234*          IMS   MT           INCREMENT PTR
0174  00227  00000022           NOP
0175  00230  00100752*          LAA   BOX
0176  00231  02701065*          MAA   ='10000      GET WAIT FLG
0177  00232  02200573*          BAZ   ADVC-1       SKIP MODE
0178  00233  01100574*          BRU   ADVC         WAIT MODE
0179  00234  00CC0000    MT     HLT                PTR FOR EMPTYING
0180  00235  00CC0000    RCNT   HLT                 PTR FOR FILLING
0181  00236  00CC0000    GOON   HLT
0182         00263         BUFF   BES   20
0183  00263  00000753*    IT     DATA  A
0184  00264  00100752*    MOP    LAA   BOX
0185  00265  01400761*          IMS   PC
0186  00266  02400272*          BAP   *&4
0187  00267  40100761*          LAA*  PC
0188  00270  01200712*          SPB   CHCK
0189  00271  01100274*          BRU   *&3
0190  00272  00100761*          LAA   PC
0191  00273  00300747*          STA   EADD
0192  00274  00101043*          LAA   =-1
0193  00275  003C0330*          STA   OPUT
0194  00276  00100752*          LAA   BOX
0195  00277  02701065*          MAA   ='1CC00  GET WAIT FLAG
0196  00300  00501076*          AMA   ='1700001    AOP
0197  00301  00300320*          STA   OUT&3
0198  00302  03300000           STI   $XFLG,0  RESET EXECUTE
0199  00303  40100747*          LAA*  EADD
0200  003C4  012C0000           SPB   $FIX
0201  00305  00100263*   AOP    LAA   IT
0202  003C6  01100273*          BRU   MOP&7
0203  003C7  CCCC0000    TAB    DATA  '00000
0204  0031C  0CC20000           DATA  '20000
0205  00311  0C040000           DATA  '40000
0206  00312  00000305*    TAB1   DATA  AOP
0207  0C313  00000102*          DATA  AIP
0208  00314  00000264*          DATA  MOP
```

```
0209  00315  00101043*  OUT   LAA   =-1
0210  00316  00300000        STA   $XFLG  SET EXECUTE
0211  00317  00100000        LAA   $AA
0212  00320  00000000        DATA  0   CONTAINS AOP
0213  00321  01100323*       BRU   *&2   NO SKIP
0214  00322  01400761*       IMS   PC   SKIP
0215  00323  01400761*       IMS   PC
0216  00324  01400330*       IMS   OPUT   RESET OUT IND.
0217  00325  00000022        NOP
0218  00326  03600327*       PIR   *&1
0219  00327  00000605*       DATA  ENT-2
0220  00330  00000000  OPUT  HLT
0221  00331  00000000  INP   HLT
0222                        *--------- CEU,TEU,SNS
0223  00332  00100752*  CTS   LAA   BOX
0224  00333  02701075*        MAA   ='70000   LOOK AT 4TH DIGIT
0225  00334  01501074*        CMA   ='20000
0226  00335  01100402*        BRU   CEU
0227  00336  01100521*        BRU   ERR   TEU ILLEGAL
0228  00337  00100752*  SNS   LAA   BOX
0229  00340  02701073*        MAA   ='2000  LOOK AT GROUP 0
0230  00341  02400352*        BAP   ME1&1
0231  00342  23201052*  SSER  LIX   =-4,2
0232  00343  00100351*        LAA   ME1
0233  00344  01100000        BRU   $MSS
0234  00345  11141405        DATA  ''ILLEGAL SWITCH''
      00346  07011440
      00347  23271124
      00350  03104040
0235  00351  10000351*  ME1   DAC   *,1
0236  00352  00100000        LAA   $SHLT   GROUP 0
0237  00353  02300573*        BAN   ADVC-1  SKIP
0238  00354  02200573*        BAZ   ADVC-1  SKIP
0239  00355  00100752*        LAA   BOX  YES
0240  00356  00020016        LSL   16
0241  00357  13201052*        LIX   =-4,1
0242  00360  23201072*        LIX   =0,2
0243  00361  02300366*        BAN   *&5
0244  00362  00001016        LSL   1
0245  00363  23400364*        IIB   *&1,2
0246  00364  13400361*        IIB   *-3,1
0247  00365  01100573*  BLOW  BRU   ADVC-1  SKIP
0248  00366  23300401*        STI   SHFT,2   NO. OF SHIFTS
0249  00367  04400401*        IAM   SHFT
0250  00370  00011016        LSL   9
0251  00371  00501071*        AMA   ='16   LSL
0252  00372  00300375*        STA   *&3
0253  00373  00100000        LAA   $CTL
0254  00374  02701070*        MAA   ='74000000  GET SENSE SWITCHES
0255  00375  00000000        HLT       SHIFT INSTRUCTION
0256  00376  02300574*        BAN   ADVC   SWITCH SET
0257  00377  00100401*        LAA   SHFT
0258  00400  01100362*        BRU   BLOW-3
0259  00401  00000000  SHFT  HLT
```

```
0260                          *--------- CEU - COMMAND NOT EXECUTED
0261                          *--------- UNTIL AIP OR MIP
0262  00402  01200444*   CEU  SPB  UTST   TTYO
0263  00403  01200431*        SPB  MIOD
0264  00404  01501067*        CMA  ='1000000
0265  00405  01100420*        BRU  CERR
0266  00406  01100413*        BRU  *&5
0267  00407  01501066*        CMA  ='2000000
0268  00410  01100420*        BRU  CERR
0269  00411  01100413*        BRU  *&2
0270  00412  01100521*        BRU  ERR
0271  00413  00300430*        STA  COMD
0272  00414  00100752*        LAA  BOX
0273  00415  02701065*        MAA  ='10000   GET WAIT
0274  00416  02200573*        BAZ  ADVC-1  SKIP - NO WAIT
0275  00417  01100574*        BRU  ADVC
0276  00420  13201052*   CERR LIX  =-4,1
0277  00421  00100427*        LAA  ME8
0278  00422  01100000         BRU  $MSS
0279  00423  11141405         DATA ''ILLEGAL CEU DATA''
      00424  07011440
      00425  03052540
      00426  04012401
0280  00427  10000427*   ME8  DAC  *,1
0281  00430  00000000    COMD HLT
0282                          *--------- SUBROUTINE TO RETURN WITH
0283                          *           COMMAND DATA IN A
0284  00431  00000000    MIOD ZZZ  **
0285  00432  00100752*        LAA  BOX
0286  00433  01400761*        IMS  PC
0287  00434  02400441*        BAP  *&5
0288  00435  40100761*        LAA* PC
0289  00436  01200712*        SPB  CHCK
0290  00437  40100747*        LAA* EADD
0291  00440  41100431*        BRU* MIOD
0292  00441  40100761*        LAA* PC
0293  00442  41100431*        BRU* MIOD
0294  00443  00000000    TEMP HLT
0295                          *--------- SUBROUTINE TO TEST UNIT NO.
0296  00444  00000000    UTST ZZZ  **
0297  00445  00100752*        LAA  BOX
0298  00446  02701050*        MAA  ='77  GET UNIT NO
0299  00447  01501064*        CMA  ='1  TTYO
0300  00450  01100452*        BRU  *&2
0301  00451  41100444*        BRU* UTST  YES
0302  00452  13201054*        LIX  =-3,1  NO - ERROR
0303  00453  00100460*        LAA  ME3
0304  00454  01100000         BRU  $MSS
0305  00455  27221716         DATA ''WRONG UNIT''
      00456  07402516
      00457  11244040
0306  00460  10000460*   ME3  DAC  *,1
0307                          *--------- EXECUTION PROGRAM
0308  00461  00000000    EXEQ ZZZ  **
```

```
0309                                *--------- INSTRUCTION TRAP
0310                                *--------- PON,POF,PID,PIE,PIR,AND TEU
0311                                *--------- ARE TREATED AS ILLEGAL
0312  00462  03500000    TRAP  SMP   $RFLG   INTERRUPT REQUESTO
0313  00463  01100000          BRU   $RTRN    YES
0314  00464  00101043*         LAA   =-1
0315  00465  00300000          STA   $XFLG    SET EXECUTE
0316  00466  03300532*         STI   TRP,0   SET TRAPPED INSTR.
0317  00467  00100761*         LAA   PC
0318  00470  01200717*         SPB   CHK1
0319  00471  40100761*         LAA*  PC
0320  00472  00300752*   TRP4  STA   BOX     RETURNS FROM EXU
0321  00473  00017015          RSL   15
0322  00474  02701050*         MAA   ='77   GET 2ND   3RD DIGITS
0323  00475  10000001          TAI   ,1
0324  00476  02200762*         BAZ   AUG   AUGMENTED INSTRUCTION
0325  00477  01501063*         CMA   ='11
0326  00500  01101011*         BRU   RTP1
0327  00501  01100001*         BRU   BRU
0328  00502  01501062*         CMA   ='21
0329  00503  51100521*         BRU*  TBLE-'12,1
0330  00504  01100521*         BRU   ERR    ERROR
0331  00505  01501061*         CMA   ='35
0332  00506  51100520*         BRU*  TBLE-'13,1
0333  00507  01101011*         BRU   RTP1    SMP
0334  00510  01501060*         CMA   ='44
0335  00511  01100521*         BRU   ERR
0336  00512  01101011*         BRU   RTP1    IAM
0337  00513  01501057*         CMA   ='56
0338  00514  01100521*         BRU   ERR
0339  00515  01101013*         BRU   RTRP    NEG
0340  00516  01501056*         CMA   ='61
0341  00517  51100476*         BRU*  TAL1-'57,1
0342  00520  01101035*         BRU   SAL  AMX
0343  00521  13201051*   ERR   LIX   =-5,1
0344  00522  00100531*         LAA   ME4
0345  00523  01100000          BRU   $MSS
0346  00524  40111414          DATA  '' ILLEGAL INSTRUCTION''
      00525  05070114
      00526  40111623
      00527  24222503
      00530  24111716
0347  00531  10000531*   ME4   DAC   *,1
0348  00532  00000000    TRP   HLT
0349  00533  00000024*   TBLE  DATA  SPB
0350  00534  00000332*         DATA  CTS    CEU,TEU,SNS
0351  00535  00001011*         DATA  RTP1    IMS
0352  00536  00001011*         DATA  RTP1    CMA
0353  00537  00001030*         DATA  EXU
0354  00540  00000061*         DATA  IO      AOP,AIP,MOP,MIP
0355  00541  00001013*         DATA  RTRP    CNS
0356  00542  00000014*         DATA  BAZ
0357  00543  00000006*         DATA  BAN
0358  00544  00000011*         DATA  BAP
```

```
0359  00545  0CC000017*          DATA BOF
0360  00546  000001011*          DATA RTP1   MEA
0361  00547  00C001011*          DATA RTP1   MAA
0362  00550  00C001011*          DATA RTP1   MCA
0363  00551  000001011*          DATA RTP1   AAM
0364  00552  000001035*          DATA SAL    LIX
0365  00553  000001035*          DATA SAL    STI
0366  00554  000000034*          DATA IIB
0367  00555  00C001025*  TAL1    DATA LCS
0368  00556  000001013*          DATA RTRP   RNA
0369  00557  001C0753*  XQ      LAA  A
0370  00560  002C0754*          LBA  B
0371  00561  132C0756*          LIX  X1,1
0372  00562  232C0757*          LIX  X2,2
0373  00563  332C0760*          LIX  X3,3
0374  00564  025C0565*          BOF  *&1
0375  00565  035C1010*          SMP  CSBI   CSB IND
0376  00566  011C0637*          BRU  CSBX   LAST INSTR WAS CSB
0377  00567  016C0752*          EXU  BOX
0378  00570  011C0574*          BRU  ADVC   ADVANCE PC BY 1
0379  00571  011C0573*          BRU  *&2    ADVANCE PC BY 2
0380  00572  014C0761*          IMS  PC     ADVANCE PC BY 3
0381  00573  014C0761*          IMS  PC
0382  00574  014C0761*  ADVC    IMS  PC
0383  00575  014C0532*          IMS  TRP
0384  00576  011C0605*          BRU  ENT-2
0385  00577  003C0753*          STA  A
0386  00600  004C0754*          STB  B
0387  00601  133C0756*          STI  X1,1
0388  00602  233C0757*          STI  X2,2
0389  00603  333C0760*          STI  X3,3
0390  00604  025C0633*          BOF  OFLW
0391  00605  0CCC0003           CLA
0392  00606  003C1010*          STA  CSBI   RESET CSB IND.
0393  00607  033C0000    ENT     STI  $XFLG,0  RESET EXECUTE
0394  00610  035C0000           SMP  $SNGL   SKIPS IF NOT IN SING. CYC.
0395  00611  411C0461*          BRU* EXEQ
0396  00612  035C0000           SMP  $SHLT  SKIPS NOT IN HALT MODE
0397  00613  011C0615*          BRU  *&2
0398  00614  011C0462*          BRU  EXEQ&1
0399  00615  00C10000           LAA  $CTL
0400  00616  027C1053*          MAA  ='77777   GET ADDRESS
0401  00617  015C0761*          CMA  PC
0402  00620  011C0462*          BRU  EXEQ&1
0403  00621  011C0623*          BRU  *&2
0404  00622  011C0462*          BRU  EXEQ&1
0405  00623  132C1052*          LIX  =-4,1
0406  00624  001C0632*          LAA  ME7
0407  00625  011C0000           BRU  $MSS
0408  00626  031C1624           DATA ''CONTROL HALT AT''
      00627  221C1440
      00630  100C1424
      00631  400C244C
0409  00632  100C0632*  ME7     DAC  *,1
```

```
0410  00633  00101043*  OFLW LAA  =-1
0411  00634  00300636*       STA  FLOW
0412  00635  01100605*       BRU  ENT-2
0413  00636  00000000  FLOW HLT
0414  00637  00200650*  CSBX LBA  CSBL  PREVIOUS B
0415  00640  00100752*       LAA  BOX
0416  00641  00300644*       STA  *&3
0417  00642  00100753*       LAA  A
0418  00643  00000007       CSB
0419  00644  00000000       HLT       INSTRUCTION STORED
0420  00645  01100574*       BRU  ADVC
0421  00646  01100573*       BRU  ADVC-1
0422  00647  01100572*       BRU  ADVC-2
0423  00650  00000000  CSBL HLT
0424                       *--------- FIND EFFECTIVE ADDRESS
0425                       *--------- DATA TO BE TESTED ARRIVES IN B
0426                       *--------- EFFECTIVE ADDRESS RETURNED IN A
0427  00651  00000000  IND  ZZZ  **
0428  00652  00101051*       LAA  =-5
0429  00653  00300703*       STA  CNT   COUNTER
0430  00654  03501042*       SMP  XINS  SKIP IF NOT STI,AMX,LIX
0431  00655  01100704*       BRU  SLA  STORE, LOAD, OR ADD X
0432  00656  00000004  RET1 TBA
0433  00657  02701055*       MAA  ='30000000  GET INDEX BITS
0434  00660  00025015       RSL  21
0435  00661  10000002       TBI  ,1   ADDRESS PORTION
0436  00662  00500711*       AMA  BASE MAKE INSTR. TO ADD
0437  00663  00300664*       STA  *&1  TO MODIFY WITH INDEX
0438  00664  00000000       HLT       AMX  SOFTWARE XI,1
0439  00665  13300747*       STI  EADD,1
0440  00666  00000004       TBA
0441  00667  02300671*       BAN  YES   INDIRECT0
0442  00670  41100651*       BRU* IND   NO
0443  00671  40200747*  YES  LBA* EADD
0444  00672  01400703*       IMS  CNT
0445  00673  01100654*       BRU  RET1-2
0446  00674  13201054*  ERCR LIX  =-3,1
0447  00675  00100702*       LAA  ME6
0448  00676  01100000       BRU  $MSS
0449  00677  65401116       DATA ''5 INDIRECTS''
      00700  04112205
      00701  0324234C
0450  00702  10000702*  ME6  DAC  *,1
0451  00703  00000000  CNT  HLT
0452  00704  00400004  SLA  TBA
0453  00705  02300657*       BAN  RET1&1   INDIRECT0
0454  00706  00000003       CLA
0455  00707  00301042*       STA  XINS   RESET INDICATOR
0456  00710  41100651*       BRU* IND
0457  00711  16100755*  BASE AMX  X0,1
0458                       *--------- CHECK EFFECTIVE ADDRESS
0459                       *--------- WORD TO BE CHECKED IN A
0460  00712  00000000  CHCK ZZZ  **
0461  00713  00000005       TAB
```

```
0462  00714  01200651*          SPB   IND
0463  00715  00100747*          LAA   EADD
0464  00716  01100726*          BRU   *&8
0465  00717  00000000     CHK1 ZZZ   **  WORD RETURNED IN B
0466  00720  00000005          TAB
0467  00721  00100717*          LAA   CHK1
0468  00722  00300712*          STA   CHCK
0469  00723  00000004          TBA
0470  00724  02701053*          MAA   ='77777   GET ADDRESS
0471  00725  00300747*          STA   EADD
0472  00726  01500750*          CMA   LB    LOWER BOUND OF ALLOWABLE CORE
0473  00727  01100734*          BRU   CORE    ERROR
0474  00730  00000022          NOP
0475  00731  01500751*          CMA   UB    UPPER BOUND
0476  00732  00000022          NOP
0477  00733  41100712*          BRU*  CHCK
0478  00734  13201052*     CORE LIX   =-4,1    4 WORD MESSAGE
0479  00735  00100746*          LAA   ME
0480  00736  01200000          SPB   $MESS
0481  00737  13201051*          LIX   =-5,1
0482  00740  00200747*          LBA   EADD
0483  00741  01100000          BRU   $TWO    OUTPUT ADDRESS
0484  00742  11141405          DATA  ''ILLEGAL ADDRESS''
      00743  07011440
      00744  01040422
      00745  05232340
0485  00746  10000746*     ME   DAC   *,1
0486  00747  00000000     EADD HLT
0487  00750  00000000     LB   HLT       BOUNDS OF AREA FOR
0488  00751  00000000     UB   HLT       TERMINAL OPERATOR'S PROG.
0489  00752  00000000     BOX  HLT     HOLDS INSTR  TO EXECUTE
0490  00753  00000000     A    HLT
0491  00754  00000000     B    HLT
0492  00755  00000000     X0   HLT
0493  00756  00000000     X1   HLT
0494  00757  00000000     X2   HLT
0495  00760  00000000     X3   HLT
0496  00761  00000000     PC   HLT
0497                      *---------- AUGMENTED INSTRUCTIONS
0498  00762  00100752*     AUG  LAA   BOX
0499  00763  02701050*          MAA   ='77  LAST TWO DIGITS
0500  00764  02201016*          BAZ   HLT
0501  00765  01501047*          CMA   ='7
0502  00766  01101013*          BRU   RTRP
0503  00767  01100777*          BRU   CSB
0504  00770  01501046*          CMA   ='23
0505  00771  01101013*          BRU   RTRP
0506  00772  01100521*          BRU   ERR
0507  00773  01501045*          CMA   ='32
0508  00774  01100521*          BRU   ERR
0509  00775  01101013*          BRU   RTRP  NOR
0510  00776  01100521*          BRU   ERR
0511  00777  00101043*     CSB  LAA   =-1
0512  01000  00301010*          STA   CSBI    CSB INDICATOR
```

```
0513  01001  01400761*          IMS  PC
0514  01002  00200754*          LBA  B
0515  01003  00400650*          STB  CSBL
0516  01004  00000007           CSB
0517  01005  00000022           NOP
0518  01006  00400754*          STB  B
0519  01007  01100607*          BRU  ENT
0520  01010  00000000     CSBI HLT
0521  01011  00100752*     RTP1 LAA  BOX
0522  01012  01200712*          SPB  CHCK
0523  01013  00101043*     RTRP LAA  =-1
0524  01014  00300532*          STA  TRP  RESET TRAPPED INST.
0525  01015  01100557*          BRU  XQ   NO MEM. CHECK
0526  01016  13201044*     HLT  LIX  =-2,1
0527  01017  01400761*          IMS  PC
0528  01020  00101024*          LAA  ME2
0529  01021  01100000           BRU  $MSS
0530  01022  10011424           DATA ''HALT AT''
      01023  40012440
0531  01024  10001024*     ME2  DAC  *,1
0532                       *--------- CONTROL SWITCHES
0533  01025  00100000     LCS  LAA  $CTL
0534  01026  00300753*          STA  A
0535  01027  01100574*          BRU  ADVC
0536                       *--------- EXECUTE
0537  01030  00100752*     EXU  LAA  BOX
0538  01031  03301010*          STI  CSBI,0   RESET CSB IND.
0539  01032  01200712*          SPB  CHCK
0540  01033  40100747*          LAA* EADD
0541  01034  01100472*          BRU  TRP4
0542                       *--------- LOAD/ADD/STORE INDEX INSTRUCTIONS
0543  01035  00101043*     SAL  LAA  =-1  STI,AMX,LIX
0544  01036  00301042*          STA  XINS   INDICATOR
0545  01037  00100752*          LAA  BOX
0546  01040  01200712*          SPB  CHCK
0547  01041  01101013*          BRU  RTRP  RESET TRAPPED INSTR.
0548  01042  00000000     XINS HLT
0549                            END
      01043  77777777
      01044  77777776
      01045  00000032
      01046  00000023
      01047  00000007
      01050  00000077
      01051  77777773
      01052  77777774
      01053  00077777
      01054  77777775
      01055  30000000
      01056  00000061
      01057  00000056
      01060  00000044
      01061  00000035
      01062  00000021
```

01063	0C000011
01064	00000001
01065	00010000
01066	02C00000
01067	01C00000
01070	74C00000
01071	00000016
01072	00000000
01073	00002000
01074	00020000
01075	00070000
01076	01700001
01077	77777754
01100	00000257
01101	77777770
01102	00000777
01103	01720001
01104	00014000
01105	77777400
01106	00004000
01107	00060000
01110	77700000

APPENDIX D

OPERATING INSTRUCTIONS

The remote terminal relocatable object tape contains all programs needed. It contains the initialization program as well as the two subprograms which are actually the remote terminal program.

To run the remote terminal, the A and B registers are loaded with the lower and upper bounds of the memory area available to the terminal operator and the control switches are loaded with the starting address of the program which is to time-share the computer with the terminal program. If the object tape were loaded at location X, the program counter is loaded with X and start is pressed twice.

The initialization routine stores the limits for the terminal operator's program and loads SPB INT and SPB OINT into '116 and '117 respectively. The keyboard is then enabled and the input and output interrupts are connected. An AOP is then executed to clear the buffer so that the output interrupt can be raised. The input interrupt is then enabled and control is transferred to the address specified by the control switches.

Before the operator uses the terminal, he should key in the command /MP to determine the area in core which he is allowed to use. The command /CL should be issued before proceeding in order to initialize the terminal program.

APPENDIX E – BRIEF DESCRIPTION OF THE COMPUTER

The SEL 840A is a 24-bit binary computer with 1.75 microsecond cycle time.

The NRC 840A has an 8K memory, two arithmetic registers (A and B) and three index registers.

840A INSTRUCTIONS *

MNE-MONIC	OP CODE	FUNCTION	MNE-MONIC	OP CODE	FUNCTION
AAM	31	Add (A) to (M)	HLT	00-00	Halt
AIP	172	(Unit) to A			
AMA	05	Add (M) to (A)	IAB	00-06	(A) to B; (B) to A
AMX	61	Add (M) to (X)	IAM	44	(A) to M; (M) to A
AOP	170	(A) to Unit	IIB	34	(X)+1, Branch if not 0
ASC	00-20	Comp. A sign	IMS	14	(M)+1; Skip if 0
BAN	23	Branch if (A) neg.	LAA	01	(M) to A
BAP	24	Branch if (A) pos.	LBA	02	(M) to B
BAZ	22	Branch if (A)=0	LCS	57	Switches to A
BOF	25	Branch on O'FLOW	LIX	32	(M) to X
BRU	11	Branch to M	LSA	00-11	Left Shift A, Arith.
			LSL	00-16	Left Shift A, Log.
CEU	130	Command Ext. Unit			
CLA	00-03	Clear (A) to 0	MAA	27	(M) AND (A)
CMA	15	Compare (A) and (M)	MEA	26	(M) Exclusive OR (A)
CNS	20	Convert No. System	MIP	176	(Unit) to M
CSB	00-07	Copy B sign	MOA	30	(M) OR (A)
			MOP	174	(M) to Unit

DIV	10	Divide	MPY	07	Multiply
EAB	21-03	(EA) to EB	NEG	56	2's comp. (A)
EAD	45	EAU add	NOP	00-22	No operation
EBA	21-02	(EB) to EA	NOR	00-32	Normalize (A) and (B)
EDP	21-12	EAU D.P. mode			
EDV	50	EAU Divide	PID	0.43	P.I. Disable
EFP	21-14	EAU F.P. mode	PIE	1.43	P.I. Enable
EFU	21-11	Un-normalize F.P.	PIR	36	P.I. Return
EIA	21-01	(EA) to EB, (EB) to EA	POF	63	Protect Bit Off
ELL	54	Load LSH of (EA)	PON	62	Protect Bit On
ELN	51	Load (M) in EA			
ELO	52	Load (M) in EA	RNA	60	Round (A) by (B)
EMU	47	EAU Multiply	RSA	00-10	Right shift A, Arith.
ENO	21-00	EAU Normalize	RSL	00-15	Right shift A, Log.
EPS	21-06	Skip if (EA) pos.			
ESL	55	Store LSH of (EA)	SAS	00-21	Skip on A sign
ESN	21-07	Skip if (EA) neg.	SMA	06	Subtract (M) from (A)
ESO	21-10	Skip on EAU O'FLOW	SMP	35	Skip if (M) pos.
ESP	21-13	EAU S.P. mode	SNS	134	Skip No. Switch
ESR	00-23	Skip if EAU not ready	SPB	12	Store Place & Branch
EST	53	Store (EA)	STA	03	(A) to M
ESU	46	EAU Subtract	STB	04	(B) to M
ESZ	21-05	Skip if (EA)=0	STI	33	(X) to M
EXU	16	Execute (M)			
			TAB	00-05	(A) to B
FLA	00-13	Full Left Shift, Arith.	TAI	00-01	(A) to X
FLL	00-17	Full Left Shift, Log.	TBA	00-04	(B) to A
FRA	00-12	Full Right Shift, Arith	TBI	00-02	(B) to X
FRL	00-14	Full Left Rotate, Log.	TEU	132	Test Ext. Unit

* [1], pg. F-1