'8008'

MONITOR ROUTINES

SCELBI COMPUTER CONSULTING, INC.
1322 REAR - BOSTON POST ROAD
MILFORD, CT. 06460

'8 0 0 8' MONITOR ROUTINES

AUTHOR: ROBERT FINDLEY

I M P O R T A N T N O T I C E

ERRATA FOR THE

'8008' MONITOR ROUTINES

OOPS!! IN GENERATING THE LISTING USED FOR THIS MANUAL, AN INSTRUCTION WAS LEFT OUT OF THE "INSPCL" SUBROUTINE, ON PAGES 16 AND 49, WHICH SETS THE INPUT BUFFER POINTER TO PAGE 000. THIS ERROR MAY BE CORRECTED WITHOUT RE-ASSEMBLING THE LISTING IN THE BACK OF THE BOOK BY SIMPLY HAVING THE OPERATOR INPUT ROUTINE, DESCRIBED ON PAGE 5, SET THE VALUE OF REGISTER H TO ZERO BEFORE RETURNING TO THE CALLING PROGRAM. THIS WILL NOT AFFECT THE OPERATION OF THE OTHER ROUTINE (CDIN) WHICH ALSO CALLS THE OPERATOR INPUT ROUTINE, SINCE THEY BOTH INPUT CHARACTERS TO THE INPUT BUFFER WHICH IS LOCATED ON PAGE 000.

IF THE PROGRAM IS TO BE RE-ASSEMBLED, TO BE ORIGINED ON A DIFFERENT PAGE OR TO MAKE REVISIONS TO THE PROGRAM, THE "INSPCL" SUBROUTINE SHOULD INCLUDE AN "LHI 000" INSTRUCTION AS THE SECOND INSTRUCTION OF THE SUBROUTINE. THE REVISED LISTING SHOULD APPEAR AS FOLLOWS:

```
INSPCL, LLI 340
LHI 000
LPIN, CAL RCV
  ...
  ...
```

INTRODUCTION

THE MONITOR PROGRAM IS A PROGRAM WHICH ENABLES THE COMPUTER OPERATOR TO UTILIZE A COMPUTER SYSTEM WITH GREATER EFFICIENCY AND EFFECTIVENESS, BY TAKING ADVANTAGE OF THE INHERENT POWER OF THE COMPUTER. BASICALLY, THE MONITOR PROGRAM ALLOWS THE OPERATOR TO CONTROL THE COMPUTER BY DIRECTING IT TO EXECUTE PROGRAMS STORED IN MEMORY, OPERATE PERIPHERAL DEVICES FOR STORING AND RETRIEVING PROGRAMS AND DATA, AND EXAMINE AND/OR MODIFY MEMORY LOCATIONS, EITHER ONE AT A TIME OR IN BLOCKS. THE PROGRAMMER WILL FIND ITS ABILITY TO INTERRUPT A PROGRAM BEING DEBUGGED AT VARIOUS POINTS AND EXAMINE THE CONTENTS OF MEMORY LOCATIONS AND "CPU REGISTERS AND STATUS FLAGS" AT THAT POINT IN THE PROGRAM IS A FUNCTION THAT IS AS POWERFUL A DEBUGGING TOOL AS A GOOD OSCILLOSCOPE IS FOR THE HARDWARE TROUBLESHOOTER.

THERE ARE SEVERAL FACTORS WHICH DETERMINE THE ABILITY TO OPERATE A COMPUTER SYSTEM ' EFFECTIVELY .' ONE OF THESE FACTORS IS TO BE ABLE TO CONTROL ITS OPERATION FROM A SINGLE LOCATION. THE MOST COMMON METHOD IS TO CONTROL THE COMPUTER FROM ITS 'FRONT PANEL'. THIS IS NORMALLY A MYRIAD OF SWITCHES AND LAMPS WHICH ENABLE THE OPERATOR TO LOAD AND EXAMINE MEMORY LOCATIONS, EXECUTE PROGRAMS STORED IN MEMORY AND, IN SOME OF THE MORE SOPHISTICATED FRONT PANELS, PERFORM SEVERAL PROGRAM DEBUGGING FUNCTIONS. USING THE FRONT PANEL TO OPERATE THE COMPUTER IS AN EXCELLENT WAY TO INTRODUCE THE BEGINNER TO THE BASICS OF THE COMPUTER'S OPERATION, BECAUSE IT GIVES HIM FIRST-HAND EXPERIENCE IN THE CONCEPTS OF LOADING MEMORY WITH A PROGRAM, STEPPING THROUGH THE PROGRAM AND SEEING HOW THE COMPUTER PROGRESSES FROM ONE INSTRUCTION TO ANOTHER. THAT'S FINE, FOR THE BEGINNER! BUT ONCE THE 'THRILL' OF WATCHING THE COMPUTER STEP THROUGH ONE OR TWO PROGRAMS IS GONE (ESPECIALLY SINCE THEY HAD TO BE LOADED SEVERAL TIMES TO GET THEM IN CORRECTLY), EVEN THE BEGINNER FINDS OPERATING THROUGH THE FRONT PANEL SLOW, CUMBERSOME AND OFTEN ANNOYING.

AN ALTERNATIVE METHOD IS TO HAVE THE COMPUTER AID IN THESE BASIC FUNCTIONS BY PROGRAMMING IT TO UTILIZE A MORE CONVENIENT 'CONTROL' DEVICE, NAMELY A KEYBOARD AND DISPLAY DEVICE. THE KEYBOARD ENTRY IS BY FAR A FASTER AND MORE ACCURATE MEANS OF ENTERING MEMORY ADDRESSES AND DATA THAN THAT OF TOGGLING THEM IN THROUGH THE FRONT PANEL SWITCHES. AND DISPLAYING THE INFORMATION AS OCTAL DIGITS ON AN ALPHANUMERIC DISPLAY, WHETHER IT BE A TTY PRINTER OR VIDEO DISPLAY, IS MUCH EASIER TO READ THAN DECODING THE BINARY PRESENTATION OF MEMORY ADDRESS AND CONTENTS ON THE FRONT PANEL INDICATORS. MAKING USE OF THESE DEVICES IMPROVES THE SYSTEM FROM THE 'HUMAN ENGINEERING' STANDPOINT, SINCE THEY GIVE THE OPERATOR A FORM OF COMMUNICATION WITH THE COMPUTER THAT IS MORE CONVENTIONAL THAN FLIPPING SWITCHES AND WATCHING LIGHTS. THIS BRINGS UP THE SECOND FACTOR IN OPERATING AN EFFECTIVE COMPUTER SYSTEM. THAT FACTOR IS USING A COMPUTER PROGRAM TO PERFORM AS MANY OF THE TASKS AS POSSIBLE WHICH THE COMPUTER IS CAPABLE OF PERFORMING FASTER AND MORE ACCURATELY THAN THE OPERATOR COULD EVER DREAM OF PERFORMING.

SINCE THE PROGRAM WILL BE OCCUPYING SPACE IN MEMORY, IT IS NECESSARY TO EVALUATE THE TYPE OF FUNCTIONS IT IS TO PERFORM AND CHOOSE THE ONES WHICH WILL BE OF GREATEST IMPORTANCE TO THE OPERATOR. FIRST, THE FUNCTIONS OF THE FRONT PANEL SHOULD BE REPLACED. ONE OF THESE FUNCTIONS IS THE EXAMINATION AND MODIFICATION OF MEMORY CONTENTS, FOR LOADING AND REVISING PROGRAMS AND DATA IN MEMORY. AN EXPANSION OF THIS WILL ALSO BE PROGRAMMED, THAT OF DISPLAYING A LARGE BLOCK OF MEMORY AT ONE TIME. THIS IS QUITE VALUABLE FOR CHECKING THAT A PROGRAM HAS BEEN LOADED CORRECTLY AND, IN DEBUGGING, TO EXAMINE LARGE DATA STORAGE AREAS.

THE NEXT FUNCTION THAT WOULD GENERALLY FOLLOW WOULD BE TO DIRECT THE OPERATION OF A STORAGE DEVICE TO STORE AND RETRIEVE THE CONTENTS OF A BLOCK OF MEMORY FOR SAVING PROGRAMS OR DATA. THIS WILL SAVE A LOT OF TIME IN THAT A LARGE PROGRAM WOULD NOT HAVE TO BE ENTERED THROUGH THE KEYBOARD EVERY TIME IT IS DESIRED TO USE IT. INSTEAD, IT CAN BE READ FROM THE BULK STORAGE DEVICE DIRECTLY INTO MEMORY TAKING ADVANTAGE OF ITS SPEED AND ACCURACY, AS OPPOSED TO KEYBOARD ENTRY. THIS PORTION OF THE PROGRAM WILL HAVE TO BE CUSTOMIZED TO THE USER'S SPECIFIC STORAGE DEVICE, AS WILL BE DESCRIBED LATER.

NOW THAT THE ABILITY TO ENTER, MODIFY AND STORE A PROGRAM HAS BEEN ESTABLISHED, THE NEXT LOGICAL PROGRESSION WOULD BE TO ENABLE THE OPERATOR TO START EXECUTION OF A PROGRAM FROM THE KEYBOARD. AT THIS POINT, A REQUIREMENT FOR DEBUGGING PROGRAMS MUST BE CONSIDERED.

IN THE PROCESS OF DEBUGGING A PROGRAM, IT MAY BE DESIRED TO SET THE INITIAL VALUES OF SPECIFIC CPU REGISTERS BEFORE JUMPING TO THE START OF A ROUTINE BEING WORKED ON. THIS CAN BE ACCOMPLISHED BY USING A SEPARATE FUNCTION TO SET UP THE VALUES TO BE PLACED IN THE CPU REGISTERS AT THE TIME THE PROGRAM IS ENTERED, VIA THE 'GO TO' FUNCTION.

AS A COMPLIMENTARY FUNCTION OF GO TO, THE MONITOR SHOULD BE ABLE TO SET A 'BREAKPOINT.' A BREAKPOINT IS A POINT IN A PROGRAM AT WHICH THE PROGRAMMER DESIRES TO STOP EXECUTION AND CHECK THE PROGRESS OF THE PROGRAMS OPERATION. THE BREAKPOINT FUNCTION REPLACES THE INSTRUCTION AT THE POINT IN QUESTION WITH A JUMP TO THE BREAKPOINT ROUTINE. WHEN THE BREAKPOINT IS REACHED, THE COMPUTER RETURNS CONTROL TO THE MONITOR WHERE THE BREAKPOINT ROUTINE WILL SAVE THE CONTENTS OF THE CPU REGISTERS AND THE STATUS FLAGS IN A TABLE IN MEMORY WHICH THE PROGRAMMER MAY REFER TO IN CHECKING THE OPERATION OF THE PROGRAM.

THESE FUNCTIONS ARE A GOOD BASE FOR SETTING UP A MONITOR PROGRAM, SINCE THEY PROVIDE THE OPERATOR WITH AN ASSORTMENT OF FUNCTIONS WHICH ARE COMMON TO THE OPERATION OF ANY COMPUTER SYSTEM. FROM THIS BASE, THE MONITOR CAN BE EXPANDED TO INCLUDE OPERATIONS OF SPECIFIC APPLICATION TO ONES OWN SET UP. SEVERAL POSSIBILITIES ARE PRESENTED AS PART OF THIS MONITOR PROGRAM. THESE FUNCTIONS INCLUDE FILLING A BLOCK OF MEMORY WITH A SPECIFIC DATA VALUE, SEARCHING MEMORY FOR A DATA PATTERN AND SHIFTING BLOCKS OF DATA FROM ONE SECTION OF MEMORY TO ANOTHER.

THE PURPOSE OF THE MANUAL IS TO PRESENT THE READER WITH A MONITOR PROGRAM WHICH CAN BE USED AS IS, OR MODIFIED OR EXPANDED TO CREATE A REAL "OPERATING SYSTEM" FOR ONE'S OWN COMPUTER SYSTEM. THE MONITOR PROGRAM CAN BE AN INVALUABLE ASSET TO ANY COMPUTER SYSTEM. ITS ABILITY TO PERFORM MANY OF THE REQUIRED 'CONVENIENCE' FUNCTIONS NEEDED TO CONTROL A COMPUTER SYSTEM ALONG WITH THE POWER IT AFFORDS THE PROGRAMMER IN DEBUGGING PROGRAMS MAKES IT A 'MUST' FOR THE SERIOUS COMPUTER OWNER.

THE BASIC FUNCTIONS AND CAPABILITIES OF A "MONITOR" PROGRAM

GENERALLY, A MONITOR PROGRAM CONSISTS OF A VARIETY OF COMMANDS WHICH ENABLE THE COMPUTER OPERATOR TO CONTROL THE OPERATION OF THE COMPUTER AND ITS RELATED PERIPHERAL DEVICES. THIS IS ACHIEVED BY ENTERING COMMANDS ON A KEYBOARD DEVICE WHICH DIRECT THE COMPUTER TO DISPLAY AND/OR MODIFY THE CONTENTS OF MEMORY LOCATIONS, PERFORM DATA STORAGE AND RETRIEVAL ON AVAILABLE 'BULK' STORAGE PERIPHERALS AND EXECUTE OTHER PROGRAMS WHICH ARE STORED IN THE COMPUTER'S MEMORY. THE MEMORY ADDRESS, OR ADDRESSES, AFFECTED BY THE COMMAND IS GENERALLY SPECIFIED IN THE COMMAND INPUT. THE NUMBER OF DIFFERENT COMMANDS ONE SETS UP IN A MONITOR PROGRAM WILL DEPEND ON THE AMOUNT OF MEMORY DESIRED TO DEDICATE TO THE MONITOR PROGRAM, SINCE IT MUST RESIDE IN MEMORY, AND ON THE NUMBER OF PERIPHERALS IT IS DESIRED TO CONTROL WITH THE MONITOR.

THE SPECIFIC I/O (INPUT/OUTPUT) DEVICES USED TO OPERATE THE MONITOR PROGRAM WILL NATURALLY VARY FROM ONE SYSTEM TO ANOTHER. FOR THIS REASON THE I/O PORTION OF THE MONITOR IS SET UP TO CALL 'USER PROVIDED' I/O DRIVER ROUTINES TO PERFORM THE ACTUAL INPUTTING AND OUTPUTTING OF COMMANDS AND DATA IN RESPONSE TO THE COMMANDS. THE REQUIREMENTS OF THE I/O DRIVERS WILL BE DESCRIBED IN THE NEXT SECTION. THIS APPROACH ENABLES THE READER TO "CUSTOMIZE" THE MONITOR PROGRAM TO THE SPECIFIC DEVICES AVAILABLE ON ONE'S COMPUTER SYSTEM WITHOUT CHANGING THE INSTRUCTIONS OF THE MONITOR PROGRAM PRESENTED HEREIN.

THE MONITOR PROGRAM PRESENTED IN THIS MANUAL IS CAPABLE OF PERFORMING THE FUNCTIONS MENTIONED WHILE OPERATING IN AN '8008' BASED MINICOMPUTER SYSTEM WITH AT LEAST 1.5K BYTES OF MEMORY. IF A SHORTER VERSION IS DESIRED, THE FUNCTIONS DEEMED LESS VALUABLE TO THE USER CAN BE DELETED. EACH FUNCTION AND ITS ASSOCIATED ROUTINE(S) IS EXPLAINED IN DETAIL TO ENABLE THE READER TO UNDERSTAND THE OPERATION OF THE PROGRAM. MANY OF THE ROUTINES DESCRIBED MAY BE APPLICABLE TO OTHER TYPES OF FUNCTIONS WHICH ONE MAY DESIRE TO INCLUDE IN ONE'S MONITOR PROGRAM. OR, THEY MAY BE UTILIZED IN DEVELOPING OTHER PROGRAMS. AS EACH ROUTINE IS PRESENTED A DETAILED, HIGHLY COMMENTED LISTING IS PROVIDED. A COMPLETE ASSEMBLED LISTING OF THE MONITOR PROGRAM IS THEN PRESENTED, TO WHICH THE READER MAY ADD THE CUSTOM I/O DRIVER ROUTINES AND IMPLEMENT THE MONITOR PROGRAM ON AN '8008' BASED SYSTEM. (READERS THAT DESIRE TO IMPLEMENT THIS PROGRAM ON OTHER TYPES OF SYSTEMS SHOULD FIND THE INFORMATION CONTAINED IN THIS MANUAL OF CONSIDERABLE VALUE. FOR EXAMPLE, IMPLEMENTING SUCH A PROGRAM ON AN '8080' BASED SYSTEM WOULD REQUIRE THE MERE TRANSLATION OF THE SOURCE LISTING TO THE EQUIVALENT '8080' INSTRUCTIONS.)

I/O (INPUT/OUTPUT) CONSIDERATIONS FOR THE MONITOR PROGRAM

BEFORE DISCUSSING THE ACTUAL ROUTINES WHICH MAKE UP THE MONITOR PROGRAM, IT IS NECESSARY TO MENTION SEVERAL POINTS ABOUT THE CHARACTER SET USED AND DESCRIBE THE REQUIREMENTS FOR THE I/O PROGRAMMING.

THE CHARACTER CODE USED BY THE MONITOR PROGRAM FOR ENTERING COMMANDS AND OUTPUTTING CHARACTERS TO THE DISPLAY DEVICE IS ASSUMED TO BE "ASCII" ENCODED CHARACTERS. THE "ASCII" CHARACTER SET CONSIST OF A 7-BIT CODE WHICH IS CAPABLE OF DEFINING UP TO 128 "CHARACTERS." THE MONITOR PROGRAM DESCRIBED HEREIN UTILIZES A SUBSET OF THIS CODE CONSISTING OF 31 DIFFERENT CHARACTERS - 15 "UPPER CASE" LETTERS OF THE ALPHABET,

THE NUMERALS 0 - 9, AND SEVERAL SYMBOLS AND PUNCTUATION MARKS. OFTEN, WHEN COMMUNICATING WITH AN ASCII ENCODED I/O DEVICE, AN 8'TH BIT IS ADDED TO THE SEVEN BIT ASCII CODE. THIS 8'TH BIT IS OFTEN REFERRED TO AS THE "PARITY" BIT BECAUSE IT CAN BE USED TO SERVE AS AN ERROR DETECTING BIT. MANY I/O DEVICES ARE DESIGNED TO OPERATE WITH EIGHT BITS OF INFORMATION, REGARDLESS OF WHETHER OR NOT "PARITY" ERROR CHECKING METHODS ARE BEING UTILIZED. THE MONITOR PROGRAM DESCRIBED HEREIN ASSUMES THAT THE "PARITY" POSITION IS ALWAYS IN A LOGIC ONE STATE. THE "ASCII" CHARACTER CODES USED BY THE MONITOR ARE PRESENTED BELOW ALONG WITH THE CODES FOR OTHER "ASCII" CHARACTERS GENERALLY PROVIDED BY "ASCII" ENCODED DEVICES. FOR I/O DEVICES WHICH DO NOT OPERATE WITH THE "ASCII" CHARACTER SET, THE PROBLEM OF CODE CONVERSION IS EASILY TAKEN CARE OF BY PROGRAMMING THE I/O DRIVER TO MAKE THE NECESSARY CONVERSION BETWEEN THE ASCII CODE DEFINED HERE TO THE CODE UTILIZED BY THE DEVICE.

CHARACTERS SYMBOLIZED	BINARY CODE	OCTAL REP	CHARACTERS SYMBOLIZED	BINARY CODE	OCTAL REP
A	11 000 001	301	!	10 100 001	241
B	11 000 010	302	"	10 100 010	242
C	11 000 011	303	#	10 100 011	243
D	11 000 100	304	$	10 100 100	244
E	11 000 101	305	%	10 100 101	245
F	11 000 110	306	&	10 100 110	246
G	11 000 111	307	'	10 100 111	247
H	11 001 000	310	(	10 101 000	250
I	11 001 001	311	)	10 101 001	251
J	11 001 010	312	*	10 101 010	252
K	11 001 011	313	+	10 101 011	253
L	11 001 100	314	,	10 101 100	254
M	11 001 101	315	-	10 101 101	255
N	11 001 110	316	.	10 101 110	256
O	11 001 111	317	/	10 101 111	257
P	11 010 000	320	0	10 110 000	260
Q	11 010 001	321	1	10 110 001	261
R	11 010 010	322	2	10 110 010	262
S	11 010 011	323	3	10 110 011	263
T	11 010 100	324	4	10 110 100	264
U	11 010 101	325	5	10 110 101	265
V	11 010 110	326	6	10 110 110	266
W	11 010 111	327	7	10 110 111	267
X	11 011 000	330	8	10 111 000	270
Y	11 011 001	331	9	10 111 001	271
Z	11 011 010	332	:	10 111 010	272
[	11 011 011	333	;	10 111 011	273
\	11 011 100	334	<	10 111 100	274
]	11 011 101	335	=	10 111 101	275
↑	11 011 110	336	>	10 111 110	276
←	11 011 111	337	?	10 111 111	277
SPACE	11 100 000	240	@	11 000 000	300
CTRL D	10 000 100	204	CTRL N	10 001 110	216
CTRL I	10 001 001	211	CTRL S	10 010 011	223
LINE FEED	10 001 010	212	CTRL T	10 010 100	224
CTRL L	10 001 100	214	CTRL U	10 010 101	225
CAR-RET	10 001 101	215	RUB OUT	11 111 111	377

74 CHARACTER ASCII SUBSET

THE I/O PORTION OF THE MONITOR PROGRAM HAS BEEN CAREFULLY STRUCTURED TO REMAIN SEPARATE FROM THE ACTUAL OPERATING ROUTINES OF THE MONITOR PROGRAM. THIS ALLOWS THE USER TO INCORPORATE WHATEVER I/O DRIVER ROUTINES MAY BE REQUIRED FOR THE SPECIFIC DEVICES AVAILABLE WITHOUT DISTURBING THE LOGIC OF THE OPERATING PROGRAM. THE USER MUST SIMPLY FOLLOW THE RULES TO BE PRESENTED NEXT WHEN FORMING THE I/O ROUTINES TO GUARANTEE THAT THE I/O DRIVER WILL PROVIDE THE NECESSARY FUNCTION WHILE MAINTAINING THE INTEGRETY OF THE OPERATING PROGRAM. IF, FOR EXAMPLE, THE PRINTER DEVICE TO BE USED IN ONE'S SYSTEM REQUIRES BAUDOT CODE, RATHER THAN ASCII, THE PRINTER OUTPUT ROUTINE MUST MAKE THE CONVERSION FROM THE ASCII CODE SENT BY THE PROGRAM TO THE EQUIVALENT BAUDOT CODE EXPECTED BY THE PRINTER.

THERE ARE FOUR SEPARATE I/O DRIVER ROUTINES REQUIRED BY THE MONITOR PROGRAM AS PRESENTED. THESE ROUTINES SHOULD BE PREPARED AS SUBROUTINES WHICH WILL BE CALLED BY THE OPERATING PROGRAM. TWO OF THE ROUTINES ARE USED TO COMMUNICATE BETWEEN COMPUTER AND OPERATOR FOR ENTERING COMMANDS AND DATA AND DISPLAYING THE COMMANDS AS ENTERED AND ALSO THE RESULTANT OUTPUT AS REQUESTED BY THE COMMAND. THE OTHER TWO ROUTINES WILL CONTROL THE STORAGE AND RETRIEVAL OF DATA ON THE SYSTEM 'BULK' STORAGE DEVICE. THE REQUIREMENTS FOR THESE I/O ROUTINES, AS FAR AS THIS MONITOR PROGRAM IS CONCERNED, ARE PRESENTED BELOW.

OPERATOR INPUT

THE OPERATOR INPUT ROUTINE WHEN CALLED MUST INPUT A SINGLE CHARACTER FROM A DEVICE, SUCH AS A KEYBOARD, AND RETURN TO THE OPERATING PROGRAM WITH THE ASCII CODE FOR THE INPUTTED CHARACTER IN THE ACCUMULATOR REGISTER OF THE CPU. THIS ROUTINE, CREATED BY THE USER, IS FREE TO USE CPU REGISTERS "A" THRU "E" FOR ITS PROCESSING. IF REGISTERS "H" AND "L" MUST BE USED (TO POINT TO A CONVERSION TABLE, FOR EXAMPLE) THEIR CONTENTS MUST BE SAVED AND THEN RESTORED TO THEIR ORIGINAL VALUE PRIOR TO RETURNING TO THE CALLING PROGRAM. THE OPERATOR INPUT ROUTINE IS REFERRED TO IN THE MONITOR PROGRAM BY THE LABEL "RCV." THERE ARE TWO POINTS IN THIS MONITOR PROGRAM WHERE "CAL RCV" IS USED TO SIGNIFY A CALL TO THE "OPERATOR INPUT" SUBROUTINE. ONE IS AT THE INSTRUCTION LABELED "IN2" IN THE "INPUT" ROUTINE (TO BE PRESENTED LATER). THE OTHER LOCATION WHICH CALLS THIS ROUTINE IS THE LOCATION LABELED "LPIN" IN THE "INSPCL" SUBROUTINE.

AN ADDITIONAL FUNCTION WHICH THE USER SHOULD PROVIDE IN THE "OPERATOR INPUT" SUBROUTINE IS THE CAPABILITY TO "ECHO" THE CHARACTER RECEIVED FROM THE INPUT DEVICE TO THE DISPLAY DEVICE. THAT IS, WHEN A CHARACTER IS ENTERED ON THE KEYBOARD IT IS GENERALLY DESIRED TO HAVE THAT CHARACTER DISPLAYED FOR THE OPERATOR TO VERIFY THE ENTRY. FOR EXAMPLE, IF THE OPERATOR INPUT IS COMING FROM AN ELECTRONIC KEYBOARD WHICH IS COMPLETELY SEPARATE FROM THE DISPLAY DEVICE, IT WOULD BE REQUIRED TO HAVE THE "RCV" ROUTINE OUTPUT THE CHARACTER CODE TO THE DISPLAY DEVICE AS EACH CHARACTER IS RECEIVED. OR, ONE MIGHT HAVE A SYSTEM IN WHICH THE INPUT DEVICE IS COORDINATED WITH THE DISPLAY DEVICE, SUCH AS A TELETYPE MACHINE OR TELEVISION-TYPE-WRITER, WHICH MAY BE COUPLED WITH A HARDWARE INTERFACE TO AUTOMATICALLY ECHO THE KEYBOARD INPUT TO THE DISPLAY DEVICE. IN THIS CASE, THE "RCV" SUBROUTINE WOULD HAVE TO ENABLE THE INTERFACE TO ECHO THE CHARACTERS WHEN ENTERED.

DISPLAY OUTPUT

THE DISPLAY OUTPUT ROUTINE IS DISTINCT FROM THE "ECHO" ROUTINE DESCRIBED IN THE OPERATOR INPUT ROUTINE ABOVE (ALTHOUGH, IN MANY CASES, THE "ECHO" FUNCTION OF THE "RCV" SUBROUTINE MAY SIMPLY BE OBTAINED BY CALLING THIS DISPLAY OUTPUT ROUTINE AS IT IS DEFINED HERE!) THE DISPLAY OUTPUT ROUTINE WHEN CALLED BY THE MONITOR PROGRAM MUST OUTPUT THE ASCII ENCODED CHARACTER CONTAINED IN THE ACCUMULATOR AT THE TIME THE ROUTINE IS CALLED TO THE DISPLAY DEVICE. THE ROUTINE IS FREE TO USE CPU REGISTERS "B" THRU "E" FOR PROCESSING. THE CALLING ROUTINE EXPECTS THE ACCUMULATOR AND REGISTERS "H" AND "L" TO CONTAIN THE ORGINAL INFORMATION WHEN THE SUBROUTINE IS EXITED. THE DISPLAY OUTPUT SUBROUTINE IS REFERENCED IN THE MONITOR PROGRAM BY A "CAL PRINT" INSTRUCTION. THERE ARE FIVE ROUTINES WHICH USE THE "CAL PRINT" COMMAND. THE "ERROR" ROUTINE USES THE "PRINT" SUBROUTINE TO OUTPUT ERROR MESSAGES TO THE OPERATOR. THE DISPLAY OUTPUT ROUTINE IS ALSO CALLED BY THE SUBROUTINES LABELED "MSG" (TO PRINT VARIOUS MESSAGES), "OCTOUT" (FOR PRINTING 3 DIGIT OCTAL NUMBERS), "COLON" (TO PRINT A :) AND "SPAC" (TO PRINT A SPACE).

BULK STORAGE INPUT

THE BULK STORAGE INPUT ROUTINE WHEN CALLED MUST INPUT DATA FROM THE BULK STORAGE DEVICE. THE FORMAT FOR READING THE DATA AND DETERMINING WHERE THE DATA IS TO BE STORED IS ENTIRELY LEFT UP TO THE USER PROVIDED BULK INPUT ROUTINE. THE ONLY FUNCTION OF THE MONITOR PROGRAM FOR THIS COMMAND IS TO ALLOW THE INITIATION OF A BULK INPUT VIA THE KEYBOARD AND TO RETURN TO THE MONITOR PROGRAM UPON COMPLETION OF THE INPUT SEQUENCE. THEREFORE, THE BULK STORAGE INPUT ROUTINE IS FREE TO USE ALL THE CPU REGISTERS WHILE PERFORMING ITS DATA INPUT. THE BULK STORAGE INPUT ROUTINE IS REFERENCED BY THE INSTRUCTION "CAL READ" WHICH IS LOCATED IN THE BULK READ ROUTINE OF THE MONITOR PROGRAM.

BULK STORAGE OUTPUT

THE BULK STORAGE OUTPUT ROUTINE WHEN CALLED MUST OUTPUT THE DATA INDICATED TO THE BULK STORAGE DEVICE. THE DATA TO BE STORED IS DELINEATED BY REGISTERS "L" AND "H" FOR THE LOW AND PAGE ADDRESS, RESPECTIVELY, FOR THE START ADDRESS AND REGISTERS "E" AND "D" FOR THE LOW AND PAGE ADDRESS, RESPECTIVELY, FOR THE ENDING ADDRESS OF THE BLOCK OF DATA TO BE OUTPUT. AS WITH THE BULK INPUT ROUTINE, THE ACTUAL FORMAT AND PROCEDURE FOR OUTPUTTING THE DATA IS ENTIRELY CONTROLLED BY THIS ROUTINE. THE MONITOR PROGRAM SIMPLY SETS UP THE REGISTERS DESIGNATING THE LIMITS OF THE BLOCK TO BE OUTPUT. THIS BULK STORAGE OUTPUT ROUTINE IS CALLED BY THE BULK WRITE ROUTINE BY THE INSTRUCTION "CAL PUNCH."

I/O INTEGRITY CONSIDERATIONS

THE OPTION OF PERFORMING ERROR CHECKS ON THE TRANSMISSION OF DATA TO AND FROM THE PERIPHERAL DEVICES IS LEFT TO THE USER. THIS IS DONE BECAUSE THERE ARE A VARIETY OF ERROR CHECKING TECHNIQUES POSSIBLE, DEPENDING ON THE TYPE OF DEVICE BEING USED IN THE SYSTEM. FOR EXAMPLE, A USER WITH A PAPER TAPE READER SYSTEM MAY ELECT TO PROVIDE FOR PARITY

CHECKING TECHNIQUES. SUCH TECHNIQUES MAY BE INPLEMENTED USING "EVEN" OR "ODD" PARITY CONVENTIONS DEPENDING ON THE TYPE OF DEVICE, OR EVEN THE USER'S PREFERENCE. ANOTHER TYPE OF I/O DEVICE, SUCH AS A COMMERCIAL MAGNETIC TAPE, OR DISC UNIT, MAY HAVE AUTOMATIC "BLOCK" ERROR CHECKING CAPABILITIES, IN WHICH CASE THE USER WOULD WANT TO HAVE THE APPROPRIATE I/O ROUTINE TEST FOR ERROR CONDITIONS AND TAKE APPROPROATE ACTION. THE USER MAY ELECT, IF ERROR CHECKING CAPABILITIES ARE IMPLEMENTED, TO ADD ADDITIONAL ROUTINES THAT PRESENT ERROR MESSAGES TO THE OPERATOR, OR THAT DIRECT THE OPERATION OF "ERROR CORRECTING" TECHNIQUES. IN ANY EVENT, SUCH TECHNIQUES ARE OUTSIDE THE SCOPE IF THIS PARTICULAR PUBLICATION AND WILL BE LEFT TO THE USER TO IMPLEMENT AS DESIRED.

MEMORY UTILIZATION OF THE MONITOR PROGRAM

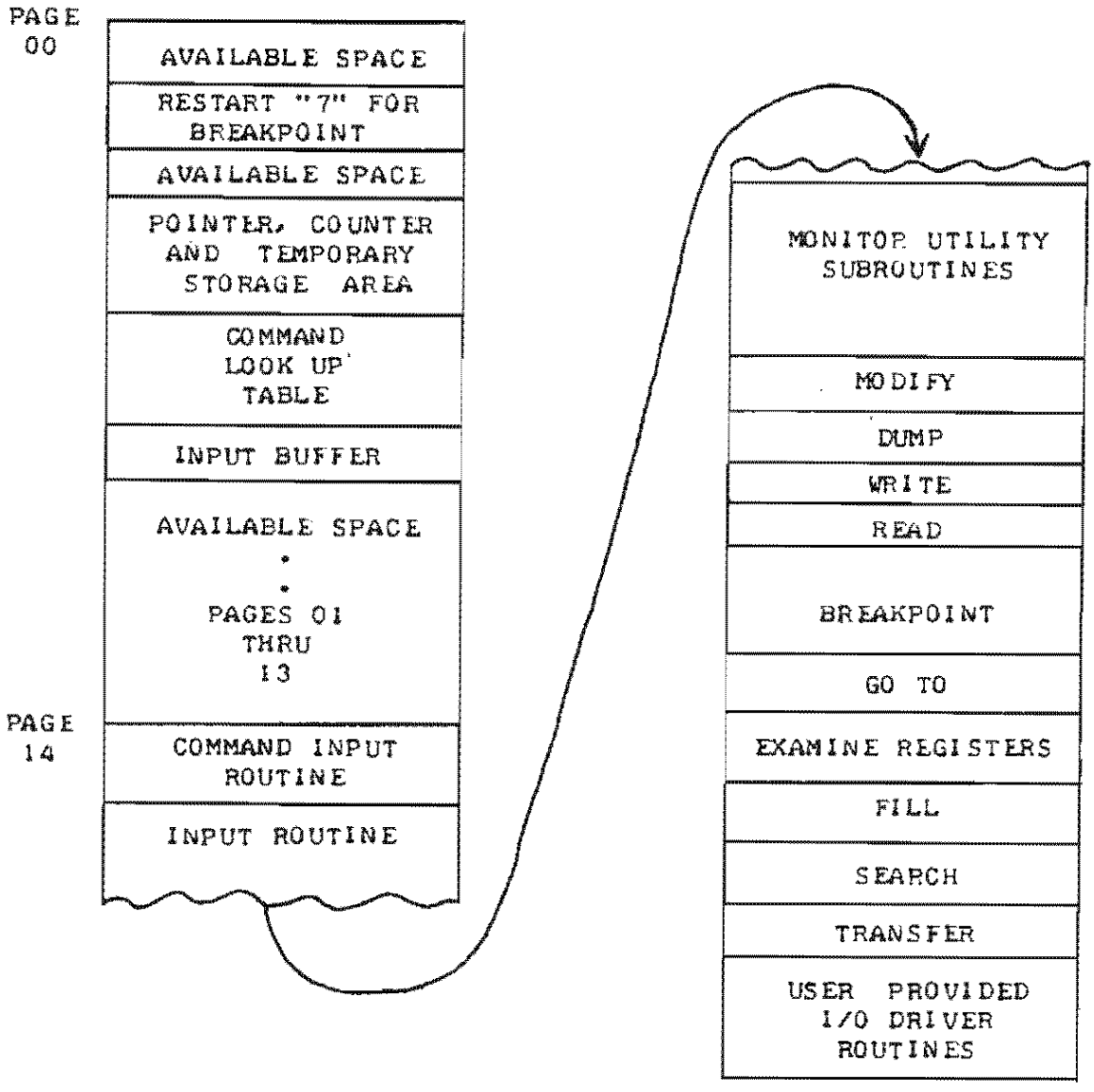
THE MONITOR PROGRAM PRESENTED IN THIS MANUAL MAKES OPTIMUM USE OF THE MEMORY BY UTILIZING EFFECTIVE PROGRAMMING TECHNIQUES WHICH TAKE ADVANTAGE OF THE '8008' INSTRUCTION SET. THE ACTUAL AMOUNT OF MEMORY USED BY THE MONITOR WILL VARY DEPENDING ON THE NUMBER OF COMMANDS ONE INCLUDES IN ONE'S VERSION AND ON THE AMOUNT OF PROGRAMMING REQUIRED TO CONTROL THE PERIPHERAL DEVICES. THE MEMORY USAGE FOR THE VERSION PRESENTED IN THIS MANUAL IS AS FOLLOWS.

THE OPERATING PORTION OF THE PROGRAM RESIDES IN PAGES 14 THROUGH PART OF PAGE 17. THE USER PROVIDED I/O ROUTINES MAY BE PLACED ON THE REMAINDER OF PAGE 17, OR, IF MORE ROOM IS REQUIRED, THE USER MAY PUT THE I/O ROUTINES WHEREVER THEY WILL BE MOST CONVIENENT (FOR EXAMPLE, THE BULK STORAGE I/O ROUTINES MAY ALREADY RESIDE IN MEMORY ON A "PROM"). PORTIONS OF PAGE 00 ARE USED AS A "SCRATCH PAD" AREA FOR THE STORAGE OF POINTERS, COUNTERS AND TEMPORARY DATA BY THE MONITOR PROGRAM. THERE IS ALSO A SECTION ON PAGE 00 WHICH CONTAINS "CANNED" MESSAGES AND THE LAST 40 OCTAL LOCATIONS ARE USED AS THE INPUT BUFFER FOR STORING THE COMMANDS AND DATA ENTERED ON THE KEYBOARD INPUT DEVICE. ONE OF THE RESTART LOCATIONS (LOCATION 070) IS USED BY THE BREAKPOINT ROUTINE TO ALLOW A SINGLE RESTART INSTRUCTION TO BE USED TO SET A BREAKPOINT IN A PROGRAM BEING DEBUGGED. THE LOOK-UP TABLE FOR THE COMMAND ROUTINE HAS BEEN SET UP ON PAGE 00 TO ALLOW ROOM FOR EXPANSION, AS WILL BE EXPLAINED LATER.

THE LOCATION OF THE OPERATING PORTION OF THE MONITOR PROGRAM FOR A SPECIFIC USER'S SYSTEM SHOULD BE IN THE UPPER PORTION OF THE AVAILABLE MEMORY. THIS ARRANGEMENT HAS BEEN FOUND TO BE MOST ADVANTAGEOUS FOR A MONITOR PROGRAM, AS IT LEAVES THE LOWER PORTION OF THE MEMORY OPEN TO BE USED FOR PROGRAM DEVELOPMENT. THE MEMORY MAP FOR THIS MONITOR PROGRAM AS ORIGINED IN THIS MANUAL IS PRESENTED ON THE FOLLOWING PAGE. THE EXACT LOCATIONS USED FOR THE TEMPORARY STORAGE ON PAGE 00 WILL BE DETAILED IN THE ASSEMBLED LISTING.

MONITOR COMMANDS

THE MONITOR PROGRAM IS ESSENTIALLY A COLLECTION OF FUNCTIONS WHICH ENABLE THE OPERATOR OR PROGRAMMER TO CONTROL THE OVER-ALL OPERATION OF THE COMPUTER. THESE FUNCTIONS ARE INITIATED BY THE OPERATOR ENTERING "COMMANDS" ON THE "OPERATOR INPUT DEVICE." EACH COMMAND DIRECTS THE MONITOR PROGRAM TO THE APPROPRIATE ROUTINE TO PERFORM THE FUNCTION INDICATED. THE FORMAT FOR ENTERING EACH COMMAND MAY VARY FROM ONE TO AN-



OTHER, DEPENDING ON WHETHER THE COMMAND REQUIRES MEMORY ADDRESSES OR DATA TO BE SPECIFIED. THE FOLLOWING IS A SUMMARY OF THE VARIOUS COMMANDS PRESENTED IN THIS MONITOR PROGRAM ALONG WITH A BRIEF DESCRIPTION OF THE OPERATION EACH PERFORMS.

"BREAKPOINT" (B) - USED TO EXAMINE THE OPERATION OF A PROGRAM IN MEMORY AT THE LOCATION SPECIFIED IN THE COMMAND. WHEN THE PROGRAM REACHES THE "BREAKPOINT," CONTROL RETURNS TO THE MONITOR PROGRAM AND THE CONTENTS OF THE SPECIFIED CPU REGISTERS AND FLAG STATUS ARE SAVED. TWO TYPES OF BREAKPOINTS ARE POSSIBLE. A TYPE "1" BREAKPOINT SAVES THE VALUES OF CPU REGISTERS A, B AND C AND THE FLAG STATUS. A TYPE "2" BREAKPOINT SAVES THE VALUES OF CPU REGISTERS D, E, H AND L AND THE FLAG STATUS.

"MEMORY DUMP" (D) - OUTPUTS THE CONTENTS OF THE MEMORY LOCATIONS SPECIFIED TO THE DISPLAY DEVICE.

"MEMORY FILL" (F) - FILLS THE MEMORY LOCATIONS SPECIFIED WITH THE DATA INDICATED IN THE COMMAND.

"GO TO" (G) - STARTS EXECUTION OF A PROGRAM BY JUMPING TO THE ADDRESS SPECIFIED IN THE COMMAND. TWO TYPES OF GO TO COMMANDS ARE POSSIBLE. A TYPE "1" GO TO COMMAND WILL SET THE CONTENTS OF CPU REGISTERS A, B AND C WITH PRE-DETERMINED VALUES BEFORE JUMPING TO THE PROGRAM. A TYPE "2" GO TO COMMAND WILL SET UP REGISTERS D, E, H AND L.

"MEMORY MODIFY" (M) - DISPLAYS THE CONTENTS OF THE MEMORY LOCATION SPECIFIED. THE OPERATOR MAY THEN CHANGE THE CONTENTS BY ENTERING THE DESIRED VALUE, AFTER WHICH THE NEXT LOCATION WILL BE DISPLAYED, OR CONTINUE ON TO DISPLAY THE NEXT LOCATION WITHOUT CHANGING THE PREVIOUS ONE, OR RETURN TO THE COMMAND MODE.

"BULK READ" (R) - CALLS THE USER PROVIDED BULK STORAGE INPUT ROUTINE TO READ DATA IN FROM THE BULK STOAGE DEVICE.

"SEARCH" (S) - SEARCHES THE MEMORY LOCATIONS SPECIFIED FOR THE 8 BIT DATA PATTERN ENTERED IN THE COMMAND AND PRINTS THE MEMORY ADDRESSES OF EACH LOCATION THAT MATCHES.

"TRANSFER" (T) - TRANSFERS THE CONTENTS OF THE SECTION OF MEMORY SPECIFIED TO THE SECTION OF MEMORY INDICATED BY THE THIRD ADDRESS SPECIFIED IN THE COMMAND.

"BULK WRITE" (W) - CALLS THE USER PROVIDED BULK STORAGE OUTPUT ROUTINE TO WRITE A SPECIFIED BLOCK OF MEMORY OUT TO THE BULK STORAGE DEVICE.

"EXAMINE REG'S" (X) - DISPLAYS THE CONTENTS OF THE SPECIFIED "VIRTUAL" CPU REGISTER OR FLAG STATUS. THE "VIRTUAL" CPU REGISTERS AND FLAG STATUS IS THEIR ACTUAL CONTENTS AT THE TIME A "BREAKPOINT" IS ENCOUNTERED, OR, AT THE TIME A "GO TO" IS ISSUED. THE VALUE OF THE "VIRTUAL" CPU REGISTERS (BUT NOT THE FLAG STATUS) MAY BE ALTERED BY THIS COMMAND.

EACH OF THE COMMANDS ARE ENTERED BY THE OPERATOR ENTERING THE LETTER ILLUSTRATED IN THE PARENTHESIS FOLLOWED BY WHATEVER DATA IS REQUIRED TO DEFINE THE ACTION TO BE TAKEN. MOST OF THE COMMANDS REQUIRE THE SPECIFICATION IF EITHER COMMAND TYPE, MEMORY ADDRESS (OR ADDRESSES), OR DATA, OR A COMBINATION OF THESE TO DEFINE THE EXACT OPERATION OF THE COMMAND. THE FORMAT FOR ENTERING EACH COMMAND IS SUMMARIZED ON THE FOLLOWING PAGE.

COMMAND	COMMAND FORMAT
BREAKPOINT (TYPE 1)	B1 HHH LLL
BREAKPOINT (TYPE 2)	B2 HHH LLL
MEMORY DUMP	D HHH LLL,MMM NNN
MEMORY FILL	F HHH LLL,MMM NNN,DDD
GO TO (TYPE 1)	G1 HHH LLL
GO TO (TYPE 2)	G2 HHH LLL
MEMORY MODIFY	M HHH LLL
BULK READ	R
SEARCH	S HHH LLL,MMM NNN,DDD
TRANSFER	T HHH LLL,MMM NNN,YYY ZZZ
BULK WRITE	W HHH LLL,MMM NNN
EXAMINE REGISTER	XP

WHERE "HHH LLL", "MMM NNN", AND "YYY ZZZ" INDICATE MEMORY ADDRESS'S AFFECTED BY THE COMMANDS, "DDD" IS THE DATA VALUE USED IN THE COMMAND AND "P" IS THE REGISTER DESIGNATION IN THE EXAMINE REGISTER COMMAND. "P" IS REPLACED BY THE LETTERS "A" THRU "E", "H" OR "L" TO INDICATE THE "VIRTUAL" CPU REGISTER TO BE EXAMINED OR THE LETTER "F" TO INDICATE THE FLAG STATUS IS TO BE DISPLAYED.

THE MEMORY ADDRESS AND DATA INFORMATION SHOWN ABOVE USES GROUPS OF THREE OCTAL DIGITS TO SPECIFY THE COMMAND'S OPERATION. EACH GROUP HAS A POSSIBLE RANGE OF VALUES FROM 000 TO 377. MEMORY ADDRESSES ARE SPECIFIED BY TWO GROUPS, THE FIRST GROUP BEING THE HIGH, OR PAGE, ADDRESS, WHILE THE SECOND GROUP DEFINES THE LOW PORTION OF THE ADDRESS. THE DATA VALUE IS SPECIFIED BY A SINGLE THREE DIGIT GROUPING. THIS NOTATION WAS CHOSEN BECAUSE IT IS A GENERALLY ACCEPTED FORMAT FOR REPRESENTING 8-BIT BINARY INFORMATION, WHICH SHOULD BE FAMILIAR TO MOST MICROCOMPUTER USER'S. IT SHOULD BE NOTED THAT WHEN ENTERING A COMMAND, LEADING ZEROS MAY BE DELETED, HOWEVER, EACH GROUP MUST BE REPRESENTED BY AT LEAST ONE DIGIT. THAT IS, IF THE MEMORY LOCATION 000 000 IS TO BE MODIFIED, THE COMMAND MAY BE ENTERED USING ONE OF THE FOLLOWING FORMS.

```
M 000 000
   OR
M 0 0
```

THE MONITOR PROGRAM

GENERAL UTILITY SUBROUTINES

THERE ARE A GROUP OF SUBROUTINES USED BY THE MAJOR ROUTINES OF THE MONITOR PROGRAM WHICH PERFORM MANY OF THE COMMON TASKS REQUIRED BY THESE ROUTINES. SUCH SMALL SEQUENCES OF INSTRUCTIONS ARE REFERRED TO AS "UTILITY" SUBROUTINES BECAUSE OF THEIR BROAD, GENERAL USAGE THROUGHOUT THIS PROGRAM. THESE SUBROUTINES ARE PRESENTED IN THIS SECTION TO POINT OUT IMPORTANT FACTORS RELATING TO THEIR OPERATION SO THAT THE READER MAY HAVE A GOOD UNDERSTANDING OF THE SUBROUTINES WHICH FORM THE BASE OF THE MONITOR PROGRAM. ALTHOUGH THESE SUBROUTINES WERE WRITTEN FOR THE MONITOR PROGRAM, THE READER MAY FIND MANY OF THEM USEFUL IN APPLYING THEM TO OTHER PROGRAMS ONE MAY DEVELOP.

THE FIRST GROUP OF "UTILITY" SUBROUTINES PERFORM THE TYPE OF OPERATIONS FOUND IN ALMOST ANY PROGRAM. THESE OPERATIONS INCLUDE INCREMENTING THE MEMORY POINTER IN REGISTER PAIR "H" AND "L," INCREMENTING A DOUBLE PRECISION VALUE STORED IN MEMORY AND SWITCHING THE CONTENTS OF REGISTERS "H" AND "L" WITH THE CONTENTS OF REGISTERS "D" AND "E," RESPECTIVELY. THESE SUBROUTINES ARE QUITE BASIC BUT ARE NEVER-THE-LESS IMPORTANT FOR MAINTAINING EFFICIENT USE OF MEMORY. AN ADDITIONAL SUBROUTINE IS INCLUDED HERE LABELED "SETUP" WHICH SETS THE MEMORY POINTER REGISTERS "H" AND "L" TO THE CONTENTS OF MEMORY LOCATIONS 167 AND 166 ON PAGE 00, RESPECTIVELY. THIS SUBROUTINE IS USED TO SET THE MEMORY POINTER TO THE MEMORY LOCATION CURRENTLY BEING OPERATED ON BY THE COMMAND.

```
    MNEMONIC                COMMENTS
 --------------          --------------

INMEM, INL              /INCR LO ADDR
RFZ                     /IF NON ZERO, RET
INH                     /ELSE, INCR PG ADDR
RET                     /RET TO CALLING PGM
/
INCR, ADI 001           /INCR CONTENTS OF MEM LOC
LMA                     /RESTORE MEM CONTENTS
RFC                     /IF NO CARRY, RET
INL                     /ELSE, FETCH NXT LOC
LAM
ADI 001                 /INCR MEM CONTENTS
LMA                     /RESTORE MEM CONTENTS
RET                     /RET TO CALLING PGM
/
SWITCH, LCH             /SWITCH THE PNTR IN
LHD                     /REG'S H AND L WITH
LDC                     /THE PNTR IN REG'S D AND E
LCL
LLE
LEC
RET                     /RET TO CALLING PGM
/
SETUP, LHI 000
LLI 166                 /SET PNTR TO 00 166
LCM                     /FETCH LO ADDR
INL
LHM                     /FETCH PG ADDR
LLC                     /SET PNTR TO MEM LOC
RET                     /RET TO CALLING PGM
```

THE NEXT GROUP OF SUBROUTINES PRESENTED BELOW ARE USED TO OUTPUT VARIOUS MESSAGES TO THE DISPLAY OUTPUT DEVICE. THREE OF THESE MESSAGE PRINTOUT ROUTINES OUTPUT A FIXED MESSAGE TO THE PRINTER. THE ROUTINE LABELED "SPAC" OUTPUTS A SPACE CHARACTER (ASCII CODE '240') AND THE ROUTINE "COLON" OUTPUTS A COLON (ASCII CODE '272') BY LOADING THE RESPECTIVE CODES IN THE ACCUMULATOR AND CALLING THE DISPLAY OUTPUT ROUTINE. "HDLN" SETS UP A POINTER TO THE "CANNED" MESSAGE "CARRIAGE-RETURN/LINE-FEED" AND THEN FALLS THROUGH TO THE SUBROUTINE "MSG" TO PRINT THE "CR-LF" COMBINATION. THE "MSG" SUBROUTINE OUTPUTS A STRING OF CHARACTERS STORED IN MEMORY TO THE DISPLAY DEVICE UNTIL A "ZERO" BYTE IS ENCOUNTERED. THE PROGRAM CALLING "MSG" SIMPLY SETS REGISTERS "H" AND "L" TO THE START ADDRESS OF THE MESSAGE TO BE PRINTED AND CALLS "MSG." THIS SUBROUTINE MAY BE OF USE TO THE READER IN DEVELOPING PROGRAMS WHICH REQUIRE

THE PRINTOUT OF "CANNED MESSAGES." THE SUBROUTINE LABELED "PRT166" OUTPUTS THE MEMORY ADDRESS CONTAINED IN LOCATIONS 166 AND 167 ON PAGE 00. LOCATION 167, WHICH CONTAINS THE HIGH PORTION OF THE ADDRESS, IS PRINTED FIRST FOLLOWED BY A SPACE AND THEN THE LOW PORTION, CONTAINED IN LOCATION 166. THIS IS USED BY SEVERAL ROUTINES, SUCH AS THE "MODIFY," "DUMP" AND "SEARCH" ROUTINES, TO PRINT THE AFFECTIVE MEMORY ADDRESSES. THIS ROUTINE CALLS THE SUBROUTINE "OCTOUT" TO PRINT EACH THREE DIGIT OCTAL NUMBER. "OCTOUT" SEPARATES EACH DIGIT FROM THE 8-BIT BYTE, FORMS THE ASCII CODE FOR THE DIGIT AND CALLS THE DISPLAY OUTPUT ROUTINE TO PRINT IT. THE FINAL SUBROUTINE, LABELED "MEMPRT," PRINTS THE CONTENTS OF THE MEMORY LOCATION INDICATED BY THE POINTER AT LOCATION 166 AND 167 ON PAGE 00. THIS ROUTINE USES THE SUBROUTINE "SETUP" TO SET THE MEMORY POINTER AND THEN CALLS "OCTOUT" TO PRINT THE MEMORY CONTENTS.

MNEMONIC	COMMENTS
SPAC, LAI 240	/SET ASCII CODE FOR SPACE
JMP PRINT	/PRINT SPACE AND RET
/	
COLON, LAI 272	/SET ASCII CODE FOR :
JMP PRINT	/PRINT COLON AND RET
/	
HDLN, LLI 134	/SET PNTR TO C/R, L/F MSG
LHI 000	/FALL THRU TO PRINT MSG
/	
MSG, LAM	/FETCH CHAR TO PRINT
NDA	/END OF MSG CHAR?
RTZ	/YES, RET TO CALLING PGM
CAL PRINT	/NO, PRINT CHAR
CAL INMEM	/INCR MSG PNTR
JMP MSG	/CONTINUE PRINT OUT
/	
PRT166, LLI 167	/SET PNTR TO PG ADDR
LHI 000	/OF LO ADDR STORED
LAM	/FETCH PG ADDR
NDI 077	
CAL OCTOUT	/PRINT PAGE ADDR
CAL SPAC	/PRINT A SPACE
LLI 166	/SET PNTR TO LO ADDR
LAM	/FETCH LO ADDR
CAL OCTOUT	/PRINT LO ADDR
/	/FALL THRU TO PRINT SPACE
/	
OCTOUT, LLA	/SAVE OCTAL NUMBER TO PRINT
RLC	/POSITION HUNDRED'S DIGIT
RLC	
NDI 003	/MASK OFF OTHER BITS
ORI 260	/FORM ASCII CODE
CAL PRINT	/PRINT DIGIT
LAL	/FETCH OCTAL NUMBER
RRC	/POSITION TEN'S DIGIT
RRC	
RRC	
NDI 007	/MASK OFF OTHER DIGITS
ORI 260	/FORM ASCII CODE
CAL PRINT	/PRINT DIGIT
LAL	/FETCH OCTAL NUMBER
NDI 007	/MASK OFF OTHER DIGITS
ORI 260	/FORM ASCII CODE
JMP PRINT	/PRINT DIGIT AND RET

```
    MNEMONIC                COMMENTS
 ---------------          --------------

/
MEMPRT, CAL SETUP     /SET PNTR TO MEM LOC
LAM                   /FETCH CURRENT MEM CONTENTS
JMP OCTOUT            /PRINT CONTENTS AND RET
/
```

THE READER SHOULD NOW UNDERSTAND THAT THE MONITOR PROGRAM IS CONTROLLED BY THE OPERATOR ENTERING COMMANDS ON THE OPERATOR INPUT DEVICE. ONCE THE COMMAND IS ENTERED AND RECOGNIZED, THE COMPUTER JUMPS TO THE MAJOR ROUTINE TO PERFORM THE DESIGNATED FUNCTION. WHEN THE MAJOR ROUTINE IS ENTERED, IT MAY BE NECESSARY TO RETRIEVE MORE INFORMATION FROM THE INPUT BUFFER IN ORDER TO PROCESS THE COMMAND. THE ADDITIONAL DATA IS ALMOST ALWAYS IN THE FORM OF OCTAL DIGITS WHICH SPECIFY MEMORY ADDRESSES OR DATA. THIS INFORMATION IS STORED IN THE INPUT BUFFER AS A STRING OF ASCII CHARACTERS AND MUST BE TRANSLATED INTO ITS EQUIVALENT BINARY VALUE(S) BEFORE THE MAJOR ROUTINE CAN USE IT. SINCE THIS FUNCTION IS A COMMON PROCESS THE FOLLOWING ASCII TO OCTAL AND OCTAL TO BINARY CONVERSION SUBROUTINES ARE USED TO PERFORM THE TRANSLATION. THE SUBROUTINE "OCTNM" READS IN A MEMORY ADDRESS, CONVERTS IT TO THE BINARY VALUE AND STORES IT IN LOCATIONS 166 AND 167 ON PAGE 00. IF A SECOND ADDRESS FOLLOWS THE FIRST IN THE INPUT BUFFER, THE SECOND ADDRESS WILL BE CONVERTED TO BINARY AND STORED IN LOCATIONS 170 AND 171 ON PAGE 00. IF THERE IS NO SECOND ADDRESS, THE FIRST ADDRESS WILL BE STORED AGAIN IN LOCATIONS 170 AND 171. THE TWO ADDRESSES THUS STORED ARE THEN CHECKED AGAINST EACH OTHER TO DETERMINE THAT THE FIRST IS LESS THAN OR EQUAL TO THE SECOND. IF NOT, AN ERROR MESSAGE IS PRINTED AND CONTROL RETURNS TO THE COMMAND MODE. ALSO, AS THE CONVERSION IS BEING PERFORMED, THE INPUT IS CHECKED FOR POSSIBLE ERRORS, SUCH AS INVALID OCTAL NUMBERS (I.E. 8,9) OR INVALID ENTRIES (I.E. ONLY ONE THREE DIGIT GROUP DEFINING AN ADDRESS). IF SUCH ERRORS ARE FOUND, AN ERROR MESSAGE IS PRINTED AND CONTROL RETURNS TO THE COMMAND MODE. THE ACTUAL ASCII TO OCTAL ("DCDNM") AND OCTAL TO BINARY ("OCT") ROUTINES ARE IN THE FORM OF SUBROUTINES TO ALLOW THEM TO BE CALLED SEPARATELY WHEN REQUIRED.

```
    MNEMONIC                COMMENTS
 ---------------          --------------

OCTNM, LEL            /SAVE INP BFR PNTR
CAL OCTPR             /CONVERT 1ST OCTAL PAIR
LLI 166               /SET PNTR TO LO ADDR STRAGE
LMB                   /SAVE LO HALF OF LO ADDR
INL
LMC                   /SAVE PG HALF OF LO ADDR
LLE                   /RESTORE INP BFR PNTR
LAM                   /FETCH NXT CHAR
CPI 254               /CHAR = COMMA?
JFZ SGL               /NO, ONLY ONE ENTRY
INL                   /YES, INCR INP BFR PNTR
LEL                   /SAVE INP BFR PNTR
CAL OCTPR             /CONVERT 2ND OCTAL PAIR
SGL, LLI 170          /SET PNTR TO HI ADDR STRAGE
LMB                   /SAVE LO HALF OF HI ADDR
INL
LMC                   /SAVE PG HALF OF HI ADDR
LAC
```

```
    MNEMONIC                COMMENTS
 ---------------          ---------------
LLI 167                 /IS HI ADDR < LO ADDR?
CPM
JTC ERR                 /YES, PRINT ERROR
RFZ                     /IF PG HALF NOT =, RET
INL                     /ELSE, CHECK LO HALF
LAM
LLI 166                 /IS HI ADDR < LO ADDR?
CPM
JTC ERR                 /YES, PRINT ERROR MSG
RET                     /NO, RET TO CALLING PGM
/
OCTPR, CAL DCDNM        /DECODE 1ST OCTAL NUMBER
LCB                     /SAVE OCTAL NUMBER
INE                     /INCR INP BFR PNTR
/                        FALL THRU TO DECODE 2ND NMBR
/
DCDNM, LLI 150          /SET PNTR TO DIGIT STRAGE TBL
LMH                     /CLEAR TBL BY STORING 000.
INL
LMH
INL
LMH
LLE                     /RESET INP BFR PNTR
LOOP, CAL FNUM          /CHECK FOR VALID NUMBER
JTS CKLNH               /IF NOT, CHECK CHAR CNT = 0
LAM                     /FETCH CHAR
LDL                     /SAVE INP BFR PNTR
NDI 007                 /MASK OFF 260
LLI 150                 /STORE OCTAL NUMBER IN
LBM                     /TABLE AT LOC 150 PG 00
LMA                     /AND SHIFT OTHER NUMBERS
INL                     /UP THRU THE TABLE
LAM
LMB
INL
LMA
LLD                     /RESTORE AND INCR INP BFR PNTR
INL
JMP LOOP                /FETCH NXT NUMBER
/
CKLNH, LAL
CPE                     /IS CHAR CNT = 0?
JTZ ERR                 /YES, PRINT ERROR MSG
LEL                     /NO, SAVE INP BFR PNTR
CAL OCT                 /FETCH FINAL OCTAL NUMBER
JFS ERR                 /IF INVALID, PRINT ERR MSG
RET                     /ELSE, RET TO CALLING PGM
/
FNUM, LAM               /IS CHAR A VALID NUMBER?
CPI 260
RTS                     /NO, RET WITH S FLAG SET
SUI 270                 /CHECK UPPER LIMIT BY
ADI 200                 /SETTING S FLAG TO PROPER
RET                     /STATE AND RETURN
/
```

```
    MNEMONIC                COMMENTS
 --------------          --------------

OCT, LLI 152            /SET PNTR TO 3RD DIGIT
LAM
CPI 004                 /IS 3RD DIGIT > 3?
RFS                     /YES, RET WITH S FLAG RESET
NDI 003                 /CLEAR CARRY
RRC                     /POSITION DIGIT
RRC
LBA                     /SAVE IN REG B
DCL                     /DECR PNTR
LAM                     /FETCH NEXT DIGIT
RLC                     /POSITION DIGIT
RLC
RLC
ADB                     /ADD TO REG B
DCL                     /DECR PNTR
ADM
LBA                     /SAVE FINAL NUMBER
LAI 200                 /SET S FLAG TO INDICATE
NDA                     /THAT THE NUMBER IS VALID
RET                     /RET TO CALLING PGM
```

THE NEXT SUBROUTINE TO BE PRESENTED IS LABELED "CKEND." THIS SUBROUTINE IS UTILIZED BY A NUMBER OF MAJOR ROUTINES WHICH OPERATE ON A GROUP OF MEMORY LOCATIONS, SUCH AS THE "DUMP," "FILL" AND "SEARCH" ROUTINES. THE BASIC FUNCTION OF THIS ROUTINE IS TO COMPARE THE VALUES OF THE POINTERS STORED IN THE DATA AREA ON PAGE 00 AT LOCATIONS 166 THRU 171 WHICH WERE INITIALLY SET UP BY INPUTTING THE COMMAND. AS EACH LOCATION IS OPERATED ON, THE TWO POINTERS ARE CHECKED TO DETERMINE IF THEY ARE EQUAL, INDICATING THE OPERATION IS COMPLETE. IF THEY ARE NOT EQUAL, THE POINTER AT LOCATION 166 AND 167 IS INCREMENTED AND THE PROCESSING IS CONTINUED. WHEN THEY BECOME EQUAL, THE PROGRAM RETURNS TO THE COMMAND MODE.

```
    MNEMONIC                COMMENTS
 --------------          --------------

CKEND, LHI 000
LLI 171                 /SET PNTR TO HI ADDR
LAM                     /FETCH 2ND HALF
LLI 167                 /SET PNTR TO 2ND HALF LO ADDR
CPM                     /2ND HALFS EQUAL?
JFZ CONT                /NO, CONTINUE PROCESS
INL
LAM                     /FETCH 1ST HALF HI ADDR
LLI 166                 /SET PNTR TO 1ST HALF LO ADDR
CPM                     /1ST HALFS EQUAL?
JTZ INCMD               /YES, RET TO CMND MODE
LLI 166                 /NO, SET PNTR TO LO ADDR
LAM
JMP INCR                /INCR LO ADDR AND RET
```

THERE ARE SEVERAL ROUTINES IN THE MONITOR PROGRAM WHICH REQUIRE THE INPUT OF ADDITIONAL INFORMATION BY THE OPERATOR AFTER THE COMMAND HAS BEEN ENTERED. FOR EXAMPLE, THE MEMORY "MODIFY" ROUTINE DISPLAYS THE CONTENTS OF A MEMORY LOCATION AND THEN WAITS FOR THE OPERATOR TO INPUT EITHER A MODIFICATION TO THE MEMORY CONTENTS OR A COMMAND TO DISPLAY THE NEXT LOCATION OR RETURN TO THE COMMAND MODE. THE FORMAT FOR THIS ENTRY, AS WILL BE DETAILED LATER, IS TERMINATED BY EITHER A SPACE OR A CARRIAGE RETURN. SINCE THE SPACE IS NOT DEFINED AS A TERMINATING CHARACTER IN THE INPUT ROUTINE, WHICH WILL BE PRESENTED SHORTLY, THE FOLLOWING INPUT ROUTINE IS USED TO ENTER THE MODIFICATIONS FOR THE "MODIFY" AND ALSO THE "EXAMINE REGISTER" COMMAND. THIS SUBROUTINE IS LABELED "INSPCL." THIS ROUTINE CALLS THE OPERATOR INPUT ROUTINE TO FETCH THE CHARACTERS ENTERED AT THE KEYBOARD. WHEN A SPACE IS ENTERED, THE SUBROUTINE RETURNS TO THE CALLING PROGRAM WITH THE MODIFICATION STORED IN THE INPUT BUFFER ON PAGE 00. IF NO MODIFICATION HAS BEEN ENTERED, THE MEMORY POINTER (REG'S H & L) WILL INDICATE THE START ADDRESS OF THE INPUT BUFFER. OTHERWISE, IT WILL INDICATE THE LOCATION IN THE INPUT BUFFER WHICH CONTAINS THE TERMINATING "SPACE" CHARACTER. WHEN A CARRIAGE RETURN IS RECEIVED, THE "INSPCL" SUBROUTINE RETURNS TO THE COMMAND MODE.

```
    MNEMONIC              COMMENTS
 --------------        --------------

INSPCL, LLI 340      /SET PNTR TO S.A. OF INP BFR
LPIN, CAL RCV        /INP CHAR
LMA                  /STORE CHAR IN INP BFR
CPI 240              /CHAR = SPACE?
RTZ                  /YES, RET TO CALLING PGM
CPI 215              /NO, CHAR = C/R?
JTZ INCMD            /YES, RET TO COMMAND MODE
INL                  /NO, INCR INP BFR PNTR
JTZ ERR              /INP BFR FULL? YES, ERROR
JMP LPIN             /NO, INP NXT CHAR
```

THE SUBROUTINE LABELED "ADRDTA" IS USED BY SEVERAL OF THE ROUTINES WHICH REQUIRE THE SPECIFICATION OF A PAIR OF MEMORY ADDRESSES FOLLOWED BY A DATA BYTE, SUCH AS THE "FILL" AND "SEARCH" ROUTINES. THIS SUBROUTINE CALLS "OCTNM" TO FETCH THE ADDRESSES FROM THE INPUT BUFFER AND STORES THEM IN BINARY FORM IN THE DATA STORAGE AREA ON PAGE 00 AND THEN CALLS "DCDNM" TO FETCH THE DATA BYTE, WHICH IS RETURNED IN REGISTER B.

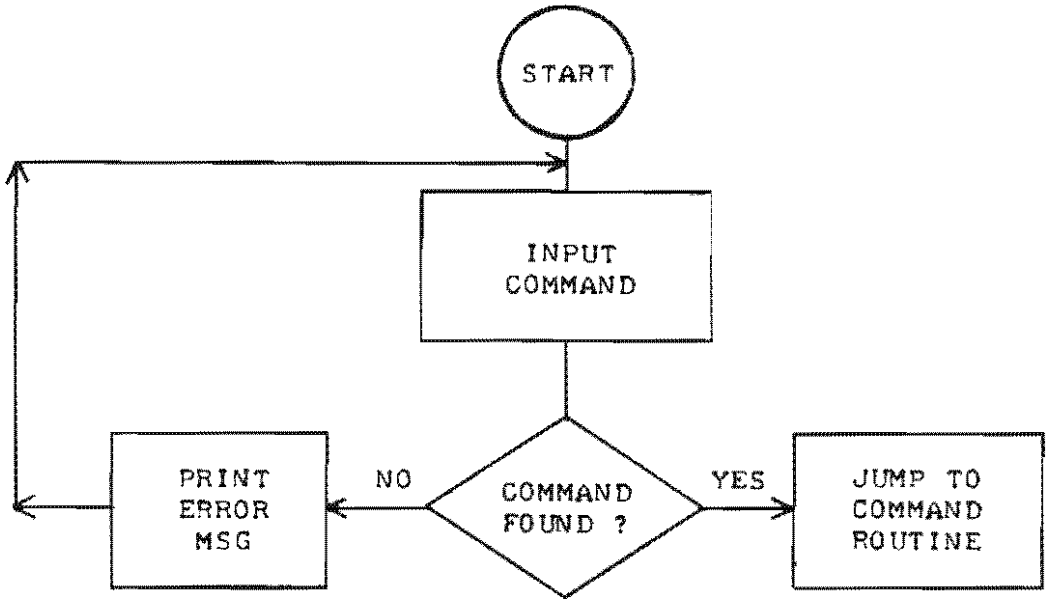
```
    MNEMONIC              COMMENTS
 --------------        --------------

ADRDTA, LLI 342      /SET PNTR TO ADDR INP
CAL OCTNM            /INP START AND END ADDR
INE                  /INCR TO DATA POSITION
JMP DCDNM            /FETCH DATA FM INP BFR
```

"COMMAND" INPUT ROUTINE

THIS SECTION DESCRIBES THE MAJOR OPERATING ROUTINES USED IN THE MONITOR PROGRAM PRESENTED HEREIN. THE FIRST SUCH ROUTINE IN THIS CATEGORY IS DESIGNATED THE "COMMAND INPUT ROUTINE." THE COMMAND INPUT ROUTINE IS SET UP WITH A VERY GENERAL FORMAT WHICH MAY BE APPLIED TO OTHER PROGRAMS THAT REQUIRE A COMMAND "LOOK UP" OPERATION. ESSENTIALLY, THE COMMAND INPUT ROUTINE ACCEPTS A COMMAND INPUT FROM THE OPERATOR INPUT DEVICE AND DIRECTS THE COMPUTER TO THE START ADDRESS OF THE ROUTINE WHICH PERFORMS THE ASSOCIATED OPERATION. THE COMMAND INPUT ROUTINE IS EASILY EXPANDABLE TO ACCOMODATE THE ADDITION OF OTHER FUNCTIONS THE USER MAY DESIRE TO INCLUDE IN THE MONITOR PROGRAM. THE BASIC OPERATING PORTION OF THIS ROUTINE IS THE SAME REGARDLESS OF THE NUMBER OF COMMANDS THERE ARE IN THE PROGRAM. TO CHANGE THE NUMBER OF COMMANDS AVAILABLE, ONE MERELY ADDS THE INFORMATION REQUIRED TO THE COMMAND "LOOK UP TABLE" AND INCREASES THE COMMAND COUNTER TO INDICATE THE TOTAL NUMBER OF COMMANDS.

THE FLOW CHART FOR THE COMMAND INPUT ROUTINE IS ILLUSTRATED BELOW. AS THE FLOW CHART INDICATES, THE BASIC CONCEPT OF THIS ROUTINE IS QUITE SIMPLE AND STRAIGHT-FORWARD.



FLOW CHART - COMMAND INPUT ROUTINE

THE COMMAND INPUT ROUTINE STARTS BY DISPLAYING A "COMMAND MODE" SYMBOL ON THE DISPLAY DEVICE. THIS SYMBOL (DEFINED AS A ">" MARK) INDICATES TO THE OPERATOR THAT THE MONITOR PROGRAM IS CURRENTLY IN THE COMMAND MODE. THE OPERATOR INPUT ROUTINE (TO BE DESCRIBED NEXT) IS THEN CALLED TO INPUT THE COMMAND FROM THE OPERATOR INPUT DEVICE. AFTER THE OPERATOR ENTERS THE COMMAND, THE COMMAND LOOK UP TABLE IS SEARCHED FOR A MATCH WITH THE FIRST CHARACTER IN THE COMMAND NOW STORED IN THE INPUT BUFFER. THIS CHARACTER IS ASSUMED TO BE ONE OF THE COMMAND IDENTIFICA-

TION LETTERS, AS DESCRIBED PREVIOUSLY. THE LOOK UP TABLE IS SEARCHED BY COMPARING THE CHARACTER ENTERED TO EVERY THIRD BYTE OF THE COMMAND "LOOK UP" TABLE. THE FORMAT FOR THE "LOOK UP" TABLE IS ILLUSTRATED BELOW.

```
BYTE  N    XXX = ASCII CODE FOR A COMMAND CHARACTER
BYTE N+1   YYY = LOW ADDR OF ASSOC COMMAND ROUTINE
BYTE N+2   ZZZ = PAGE ADDR OF ASSOC COMMAND ROUTINE
BYTE N+3   MMM = ASCII CODE FOR A COMMAND CHARACTER
BYTE N+4   NNN = LOW ADDR OF ASSOC COMMAND ROUTINE
BYTE N+5   000 = PAGE ADDR OF ASSOC COMMAND ROUTINE
BYTE N+6   AAA = ASCII CODE FOR A COMMAND CHARACTER
                  .
                  .
                  .
  REPEAT SEQUENCE TO END OF COMMAND LOOK UP TABLE
```

IF A MATCH IS FOUND BETWEEN THE CHARACTER ENTERED AND AN ENTRY IN THE COMMAND LOOK UP TABLE, THE ADDRESS IN THE SUCCEEDING TWO BYTES OF THE COMMAND LOOK UP TABLE ARE OBTAINED AND TRANSFERRED TO TWO SPECIAL LOCATIONS ON PAGE 00. THESE LOCATIONS FORM THE SECOND AND THIRD BYTES OF A "JUMP" INSTRUCTION WHICH IS THEN EXECUTED TO JUMP TO THE COMMAND ROUTINE AS SPECIFIED IN THE COMMAND JUST RECEIVED. IF, HOWEVER, THERE IS NO MATCH FOUND IN THE LOOK UP TABLE, THIS IS ASSUMED TO BE AN ERROR CONDITION AND AN ERROR MESSAGE IS OUTPUT TO THE DISPLAY DEVICE. THE PROGRAM THEN RETURNS TO THE START OF THE COMMAND INPUT ROUTINE TO RECEIVE A NEW COMMAND ENTRY.

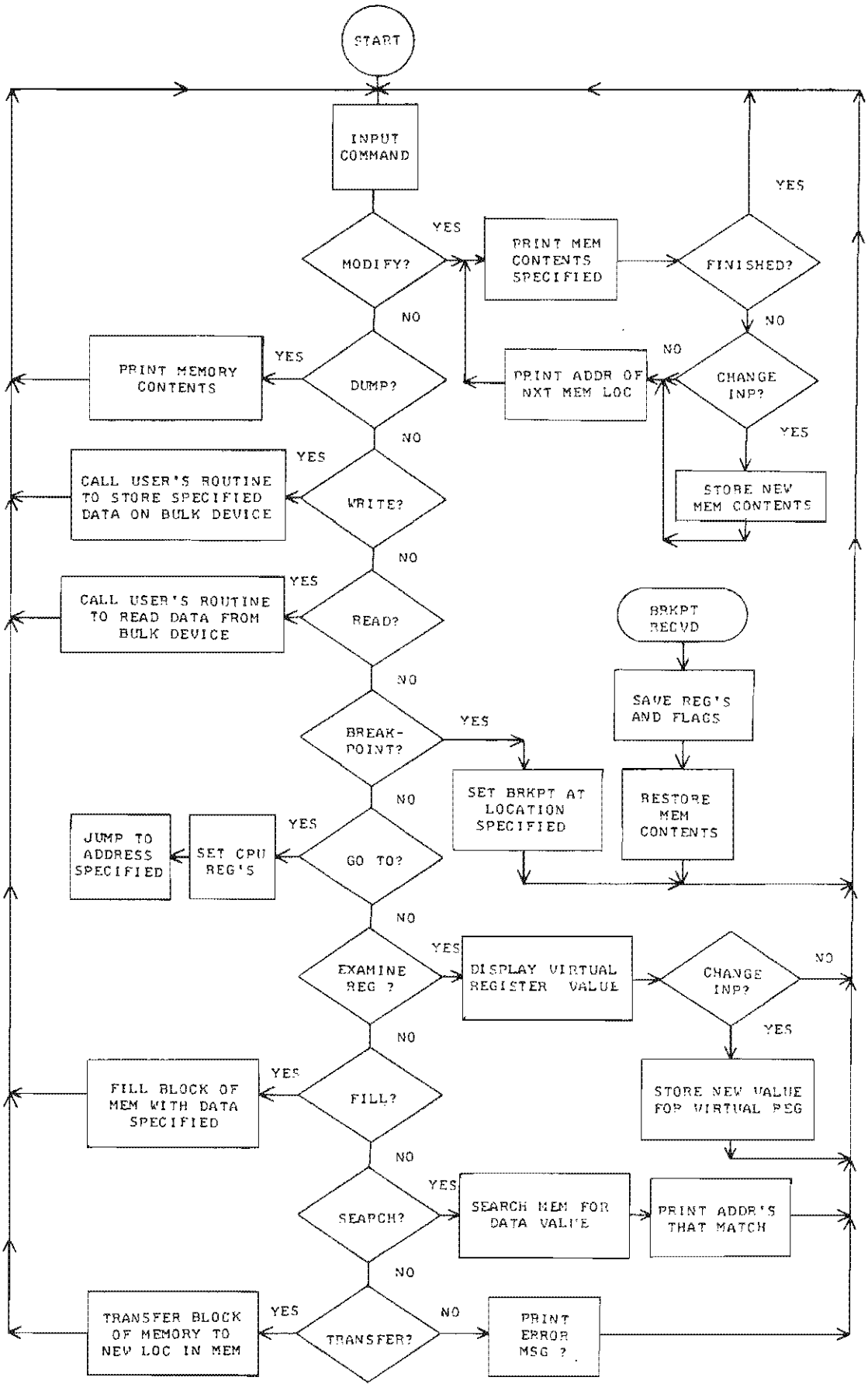
THE LISTING FOR THE COMMAND "LOOK UP" TABLE FOLLOWED BY THE COMMAND INPUT ROUTINE FOR THIS MONITOR PROGRAM IS PRESENTED BELOW. THE COMMAND "LOOK UP" TABLE RESIDES ON PAGE 00 STARTING AT LOCATION 210. THIS LOCATION ALLOWS EXPANSION OF THE LOOK UP TABLE BY SIMPLY ADDING THE ASCII CODE FOR THE IDENTIFYING CHARACTER FOR THE COMMAND TO BE ADDED, FOLLOWED BY THE LOW AND PAGE PORTION OF THE START ADDRESS OF THE NEW COMMAND, AS EXPLAINED ABOVE. THEN SIMPLY INCREMENT THE "IMMEDIATE" PORTION OF THE 7'TH INSTRUCTION (LDI 011) IN THE COMMAND INPUT ROUTINE. THE ACTUAL OPERATING PORTION OF THE COMMAND INPUT ROUTINE AND, THUS, THE MONITOR PROGRAM ITSELF, STARTS AT THE INSTRUCTION LABELED "INCMD."

```
    MNEMONIC                COMMENTS
 -------------            -------------

/COMMAND LOOK UP TABLE
/
315                   /MODIFY
150
015
304                   /DUMP
275
015
327                   /BULK WRITE
343
015
322                   /BULK READ
371
015
302                   /BREAKPOINT
377
015
```

```
    MNEMONIC                    COMMENTS
 ---------------            ---------------
307                     /GO TO
220
016
330                     /EXAMINE REGISTERS
257
016
306                     /FILL MEM
005
017
323                     /SEARCH
022
017
324                     /TRANSFER
061
017
/
/COMMMAND INPUT ROUTINE
/
ORG 014 000
INCMD, LHI 000          /SET PNTR TO HEADING MSG
LLI 130
CAL MSG                 /PRINT C/R, L/F, >
CAL CDIN                /INPUT COMMAND FM KYBD
LLI 340
LAM                     /FETCH COMMAND CHAR
LDI 012                 /SET CMND NMBR CNTR
LLI 210                 /SET CMND TABLE PNTR
CKCMD, CPM              /IS CMND CHAR FOUND IN TBL?
JTZ FOUND               /YES, PROCESS COMMAND
INL                     /NO,ADVANCE CMND TBL PNTR
INL
INL
DCD                     /IS LAST CMND CHECKED?
JFZ CKCMD               /NO, CHECK NEXT
ERR, CAL HDLN           /YES, PRINT C/R, L/F
LAI 311                 /ILLEGAL ENTRY CODE
CAL PRINT               /PRINT ERROR MSG
JMP INCMD               /INP NEXT COMMAND
/
FOUND, INL              /ADV CMND TBL PNTR
LDM                     /FETCH CMND LO ADDR
INL
LCM                     /FETCH CMND PG ADDR
LLI 156                 /SET PNTR TO JMP INSTR.
LHI 000
LMD                     /LOAD LO ADDR OF CMND
INL
LMC                     /LOAD PG ADDR OF CMND
LLE
JMP 155 000             /JUMP TO CMND ROUTINE
```

A FLOW CHART OF THE ENTIRE MONITOR PROGRAM IN THIS MANUAL IS PRESENTED ON THE FOLLOWING PAGE. IT CAN ACTUALLY BE THOUGHT OF AS A MORE DETAILED VERSION OF THE COMMAND INPUT ROUTINE FLOW CHART, SINCE IT DEFINES EACH COMMAND THAT IS SEARCHED FOR IN THE COMMAND INPUT ROUTINE. THE READER MAY DESIRE TO REFER TO THIS FLOW CHART FROM TIME-TO-TIME TO SEE HOW VARIOUS FUNCTIONS OF THE PROGRAM RELATE TO EACH OTHER.


START
INPUT COMMAND
MODIFY?
YES
PRINT MEM CONTENTS SPECIFIED
FINISHED?
YES
NO
NO
DUMP?
YES
PRINT MEMORY CONTENTS
PRINT ADDR OF NXT MEM LOC
NO
CHANGE INP?
YES
STORE NEW MEM CONTENTS
NO
WRITE?
YES
CALL USER'S ROUTINE TO STORE SPECIFIED DATA ON BULK DEVICE
NO
READ?
YES
CALL USER'S ROUTINE TO READ DATA FROM BULK DEVICE
BRKPT RECVD
SAVE REG'S AND FLAGS
NO
BREAK-POINT?
YES
SET BRKPT AT LOCATION SPECIFIED
RESTORE MEM CONTENTS
NO
GO TO?
YES
SET CPU REG'S
JUMP TO ADDRESS SPECIFIED
NO
EXAMINE REG ?
YES
DISPLAY VIRTUAL REGISTER VALUE
CHANGE INP?
NO
YES
STORE NEW VALUE FOR VIRTUAL REG
NO
FILL?
YES
FILL BLOCK OF MEM WITH DATA SPECIFIED
NO
SEARCH?
YES
SEARCH MEM FOR DATA VALUE
PRINT ADDR'S THAT MATCH
NO
TRANSFER?
YES
TRANSFER BLOCK OF MEMORY TO NEW LOC IN MEM
NO
PRINT ERROR MSG ?

INPUT ROUTINE

THE INPUT ROUTINE IN THIS MONITOR PROGRAM IS USED TO INPUT COMMANDS FROM THE OPERATOR INPUT DEVICE. THE ROUTINE ACCEPTS INPUTS FROM AN EXTERNAL DEVICE BY CALLING THE "RCV" SUBROUTINE AND STORES THE CHARACTERS IN THE INPUT BUFFER RESIDING ON PAGE 00 UNTIL A TERMINATING CHARACTER IS RECEIVED. THE ROUTINE ALLOWS THE CORRECTION OF INDIVIDUAL CHARACTERS ENTERED AND THE CAPABILITY TO ABORT THE CURRENT INPUT AND RETURN TO THE COMMAND MODE.

THE FLOW CHART FOR THE INPUT ROUTINE IS PRESENTED ON THE FOLLOWING PAGE. THE READER MAY REFER TO THIS DURING THE FOLLOWING DISCUSSION.

THE FIRST OPERATION PERFORMED BY THIS ROUTINE IS TO "CLEAR OUT" THE INPUT BUFFER AREA. THIS IS ACCOMPLISHED BY FILLING THE INPUT BUFFER AREA WITH THE ASCII CODE FOR A SPACE, '240' OCTAL. THE START ADDRESS OF THE INPUT BUFFER IS THEN SET UP TO BEGIN STORING CHARACTERS AS THEY ARE ENTERED VIA THE "RCV" ROUTINE. AS EACH CHARACTER IS ENTERED, IT IS RETURNED TO THE INPUT ROUTINE IN THE ACCUMULATOR. THE CHARACTER IS THEN TESTED TO DETERMINE IF IT IS ONE OF THE "CONTROL" CHARACTERS.

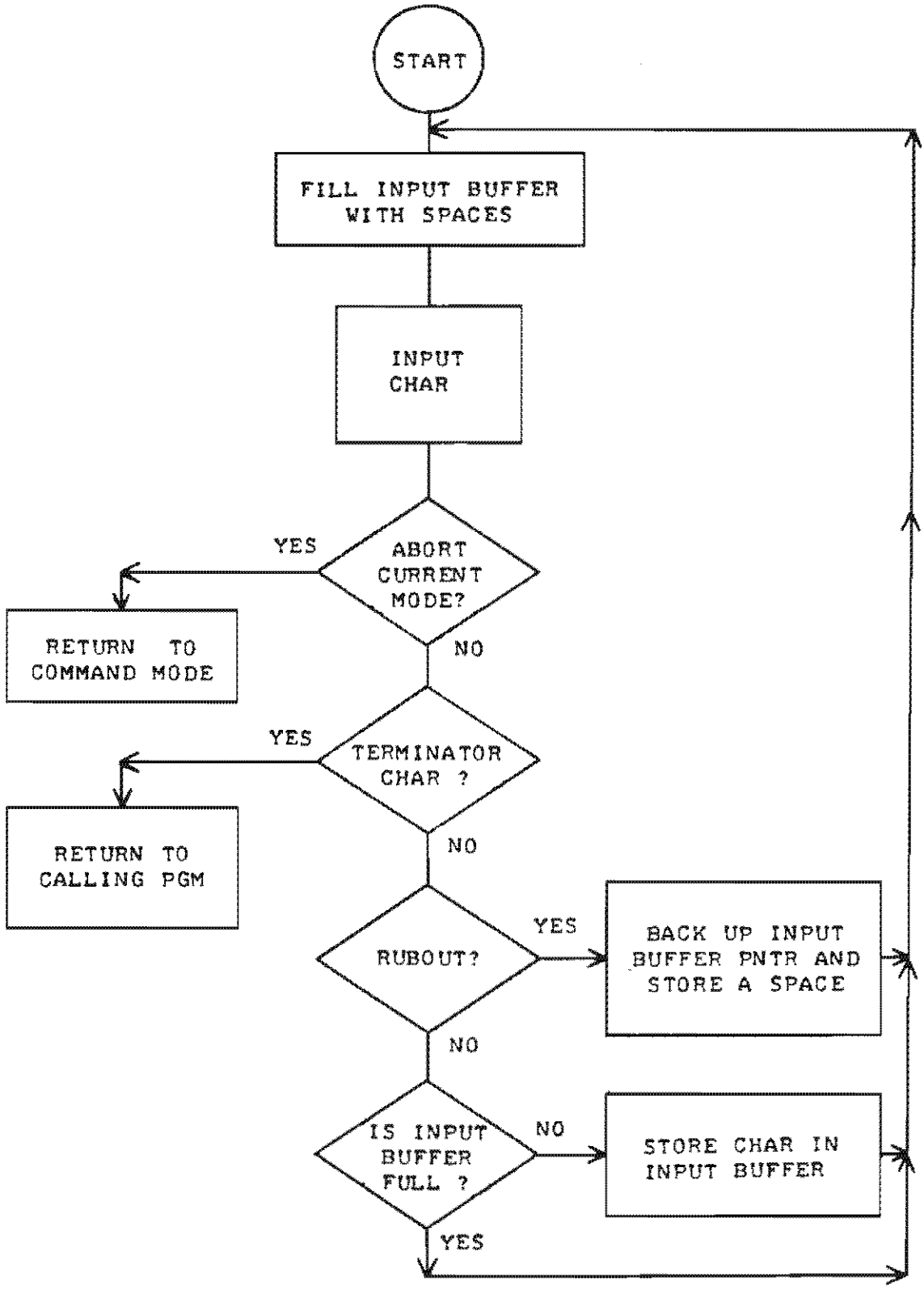
THE FIRST CONTROL CHARACTER TESTED FOR IS THE "CONTROL/D," ASCII CODE 204 OCTAL. THIS IS GENERALLY ENTERED BY SIMULTANEOUSLY DEPRESSING THE "CONTROL" KEY AND THE "D" ON AN ASCII ENCODED KEYBOARD. RECEIPT OF "CONTROL D" INDICATES THE OPERATOR WISHES TO ABORT THE CURRENT INPUT AND START A NEW COMMAND INPUT.

IF THE CHARACTER IS NOT A "CONTROL/D," THE ROUTINE TESTS FOR ONE OF TWO POSSIBLE "TERMINATING" CHARACTERS. THESE CHARACTERS ARE A CARRIAGE RETURN, ASCII CODE 215 OCTAL, AND A "CONTROL/L," ASCII CODE 214 OCTAL. THE REASON FOR PROVIDING TWO TERMINATING CHARACTERS IS TO ALLOW THE OPTION OF EITHER CAUSING THE DISPLAY DEVICE TO PERFORM A CARRIAGE RETURN WHEN THE TERMINATING CHARACTER IS ENTERED, OR, TO MAINTAIN THE POSITION OF THE DISPLAY DEVICE AT THE END OF THE CURRENT LINE OF INPUT, AS IS THE CASE WITH THE FIRST COMMAND INPUT FOR THE "MODIFY" ROUTINE AND AFTER ENTERING THE "EXAMINE REGISTER" COMMAND.

THE FINAL CONTROL CHARACTER TESTED FOR BY THE INPUT ROUTINE IS THE ASCII CODE 377 OCTAL, WHICH IS ASSIGNED TO THE "RUBOUT" OR "DELETE" FUNCTION. RECEIPT OF THIS CHARACTER INDICATES TO THE INPUT ROUTINE THAT THE PREVIOUS CHARACTER ENTERED BY THE OPERATOR IS TO BE DELETED FROM THE INPUT BUFFER. THIS IS ACCOMPLISHED BY BACKING UP THE INPUT BUFFER POINTER ONE LOCATION AND INSERTING THE CODE FOR A "SPACE" TO EFFECTIVELY "ERACE" ONE CHARACTER ENTRY FROM THE INPUT BUFFER. AN OPERATOR MAY ERACE MORE THAN ONE CHARACTER BY USING THE "RUBOUT" FUNCTION SEVERAL TIMES IN SUCCESSION.

IF NONE OF THE PREVIOUSLY MENTIONED "CONTROL" CHARACTERS ARE FOUND BY THE INPUT ROUTINE, THE CODE FOR THE CHARACTER ENTERED WILL BE STORED IN THE INPUT BUFFER AND THE INPUT BUFFER POINTER WILL BE ADVANCED. THIS PROCESS WILL CONTINUE AS LONG AS CHARACTERS ARE ENTERED FROM THE OPERATOR INPUT DEVICE. HOWEVER, ONCE THE INPUT BUFFER IS FILLED, NO FURTHER STORAGE WILL TAKE PLACE, PREVENTING THE OPERATOR FROM INADVERTANTLY ENTERING TOO MANY CHARACTERS AND OVERFLOWING ONTO PAGE 01. THE INPUT BUFFER IS CAPABLE OF HOLDING 32 CHARACTERS WHICH IS LONGER THAN ANY OF THE INPUTS REQUIRED BY THIS MONITOR PROGRAM.

THE LISTING FOR THE INPUT ROUTINE FOLLOWS THE FLOWCHART. THE START OF THIS ROUTINE IS AT THE INSTRUCTION LABELED "CDIN."



INPUT ROUTINE FLOW CHART

```
    MNEMONIC                COMMENTS
/---------------          ---------------
/
CDIN, LLI 340          /SET PNTR TO START OF INP BFR
SPI, LMI 240           /FILL INP BFR WITH SPACES
INL                    /INCR INP BFR PNTR
JFZ SPI                /DONE? NO, STORE MORE SPACES
LLI 340                /SET INP BFR PNTR
IN2, CAL RCV           /INP CHAR FM INP DEVICE
CPI 204                /CHAR = CNT'L D?
JTZ INCMD              /YES, RET TO COMMAND MODE
CPI 215                /CHAR = CAR RET?
RTZ                    /YES, RET TO CALLING PGM
CPI 214                /CHAR = CNT'L L?
RTZ                    /YES, RET TO CALLING PGM
CPI 377                /CHAR = RUBOUT?
JTZ BDCR               /YES, DELETE CHAR FM INP BFR
INL                    /IS INP BFR FULL?
DCL
JTZ IN2                /YES, DON'T STORE CHAR
LMA                    /NO, STORE CHARACTER
INL                    /INCR INP BFR PNTR
JMP IN2                /INP NEXT CHAR
/
BDCR, LAI 340          /SET ACC TO INP BFR S.A.
CPL                    /ANY CHARACTERS YET?
JTZ IN2                /NO, CONTINUE INPUT
DCL                    /YES, BACK UP INP BFR PNTR
LMI 240                /STORE SPACE OVER LAST CHAR
JMP IN2                /CONTINUE INPUT
/
```

IT SHOULD BE EASY TO SEE THAT THE READER MAY ELECT TO ASSIGN DIFFERENT CHARACTERS TO OPERATE AS "CONTROL" CHARACTERS IN THE INPUT ROUTINE. THIS IS READILY ACCOMPLISHED BY CHANGING THE IMMEDIATE PORTION OF THE "CPI" INSTRUCTIONS IN THE INPUT ROUTINE. FOR EXAMPLE, IF THE USER DESIRES TO HAVE THE CODE FOR "CONTROL O" (217 OCTAL) SERVE AS THE CONTROL CHARACTER FOR THE "RUBOUT" FUNCTION INSTEAD OF 377 OCTAL, THE USER SIMPLY SUBSTITUTES "217" FOR "377" IN THE "CPI" INSTRUCTION USED TO TEST FOR THE "RUBOUT."

ADDITIONALLY, IF THE USER DESIRES TO ADD OTHER TYPES OF "CONTROL" FUNCTIONS TO THE INPUT ROUTINE, IT COULD BE READILY DONE BY ADDING "CPI" INSTRUCTIONS FOLLOWED BY APPROPRIATE CONDITIONAL "JUMPS" TO USER PROVIDED ROUTINES TO PERFORM THE DESIRED OPERATION.

THE "MODIFY" ROUTINE

THE "MODIFY" ROUTINE IS USED TO DISPLAY AND, IF DESIRED, MODIFY THE CONTENTS OF MEMORY LOCATIONS FOR THE PURPOSE OF LOADING PROGRAMS USING THE KEYBOARD AS THE ENTRY DEVICE, OR CHANGING THE INSTRUCTIONS IN A PROGRAM OR EXAMINING AND REVISING DATA STORED IN MEMORY. THIS ROUTINE DISPLAYS ONE LOCATION AT A TIME, ALLOWING THE OPERATOR TO ENTER CHANGES OR CONTINUE TO DISPLAY THE NEXT LOCATION OR TERMINATE THE OPERATION. THE "MODIFY" ROUTINE PERFORMS IN THE FOLLOWING MANNER.

FIRST, THE ADDRESS ENTERED IN THE COMMAND IS CONVERTED AND STORED IN THE DATA AREA AT LOCATION 166 AND 167 ON PAGE 00. THE "MODIFY" ROU-

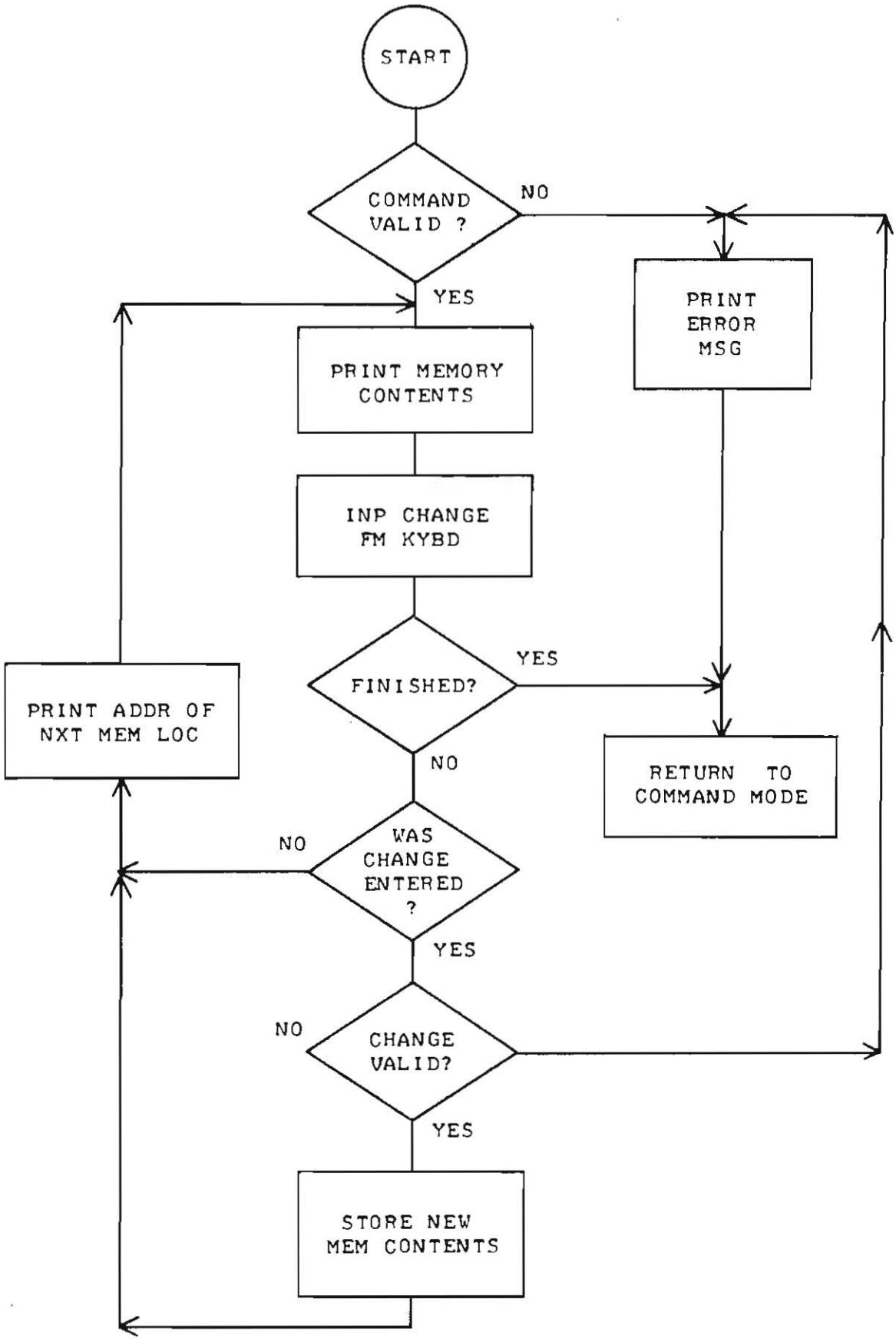
TINE THEN PRINTS THE CONTENTS OF THE DESIGNATED MEMORY LOCATION AND CALLS THE "INSPCL" SUBROUTINE TO ALLOW THE OPERATOR TO ENTER THE MODIFICATION. IF A "MOD" IS ENTERED, THE "DCDNM" SUBROUTINE IS CALLED TO DECODE THE NUMBER FROM THE INPUT BUFFER WHICH IS THEN STORED AS THE NEW CONTENTS OF THE SPECIFIED MEMORY LOCATION. WHEN THIS IS COMPLETE, OR IF NO MODIFICATION WAS ENTERED, THE ADDRESS STORED FOR THIS COMMAND WILL BE INCREMENTED AND THIS NEW ADDRESS WILL BE PRINTED ON A NEW LINE ON THE DISPLAY DEVICE. THE PROGRAM THEN LOOPS BACK TO PRINT AND MODIFY THE CONTENTS OF THIS LOCATION. THE LOOP IS TERMINATED BY THE OPERATOR ENTERING A CARRIAGE RETURN OR AN INVALID OCTAL NUMBER FOR THE MODIFICATION.

THE LISTING FOR THIS "MODIFY" ROUTINE IS PRESENTED BELOW AND THE FLOW CHART OF ITS OPERATION FOLLOWS ON THE NEXT PAGE.

```
    MNEMONIC                COMMENTS
 ---------------          ---------------
MODIFY, LLI 342         /SET INP BFR PNTR
CAL OCTNM               /FETCH ADDR TO MODIFY
CAL SPAC                /PRINT SPACE
MOD1, CAL MEMPRT        /PRINT CONTENTS OF MEM LOC
CAL COLON               /PRINT COLON
CAL INSPCL              /INP MODIFICATION
LAI 340                 /WAS MOD ENTERED?
CPL
JTZ NXLOC               /NO, SET UP NXT LOC
LEA                     /YES, SAVE INP PNTR
CAL DCDNM               /CONVERT TO OCTAL NUMBER
LAB                     /SAVE OCTAL NUMBER
LLI 166                 /SET PNTR TO MEM ADDR STRAGE
LEM                     /FETCH MEM PNTR
INL
LDM
CAL SWITCH              /SET PNTR TO MEM LOC
LMA                     /LOAD MEM WITH NEW VALUE
NXLOC, LHI 000          /SET PNTR TO PG 00
LLI 166                 /SET PNTR TO MEM ADDR STRAGE
LAM                     /FETCH LO HALF
CAL INCR                /INCR MEM ADDR
CAL MCONT               /PRINT NXT ADDR TO MODIFY
JMP MOD1
/
MCONT, CAL HDLN         /PRINT C/R, L/F
JMP PRT166              /PRINT ADDR TO MODIFY AND RET
/
```

THE "DUMP" ROUTINE

THE MEMORY "DUMP" ROUTINE ENABLES THE OPERATOR TO EXAMINE A LARGE BLOCK OF MEMORY LOCATIONS WITH A SINGLE COMMAND ENTRY, AS OPPOSED TO HAVING TO ENTER A CHARACTER IN BETWEEN THE COMPUTER DISPLAYING EACH LOCATION, AS REQUIRED BY THE "MODIFY" ROUTINE. THIS ROUTINE WILL DISPLAY AS MANY LOCATIONS AS DEFINED BY THE START AND END ADDRESSES SPECIFIED IN THE COMMAND.



MEMORY "MODIFY" ROUTINE FLOW CHART

AFTER CONVERTING AND STORING THE ADDRESSES SPECIFIED IN THE COMMAND BY CALLING THE "OCTNM" SUBROUTINE, THE "DUMP" ROUTINE PRINTS THE ADDRESS OF THE FIRST LOCATION TO BE DISPLAYED. A COUNTER IS THEN SET UP WHICH INDICATES THE NUMBER OF LOCATIONS TO BE PRINTED ON THE CURRENT LINE. THIS COUNTER IS SET FOR 20 OCTAL LOCATIONS PER LINE IN THIS PROGRAM AND IS TEMPORARILY STORED ON PAGE 00. THE CONTENTS OF THE MEMORY LOCATIONS ARE THEN PRINTED UNTIL EITHER THE LOCATION PER LINE COUNTER REACHES ZERO OR THE LAST LOCATION SPECIFIED HAS BEEN PRINTED. WHEN THE L/L COUNTER REACHES ZERO, THE L/L COUNTER IS SET TO 20 AGAIN AND A NEW LINE IS STARTED WITH THE ADDRESS OF THE NEXT LOCATION PRINTED FIRST FOLLOWED BY THE CONTENTS OF THE NEXT 20 OCTAL LOCATIONS. THIS ROUTINE RETURNS TO THE COMMAND MODE WHEN THE LAST LOCATION SPECIFIED IN THE COMMAND HAS BEEN PRINTED.

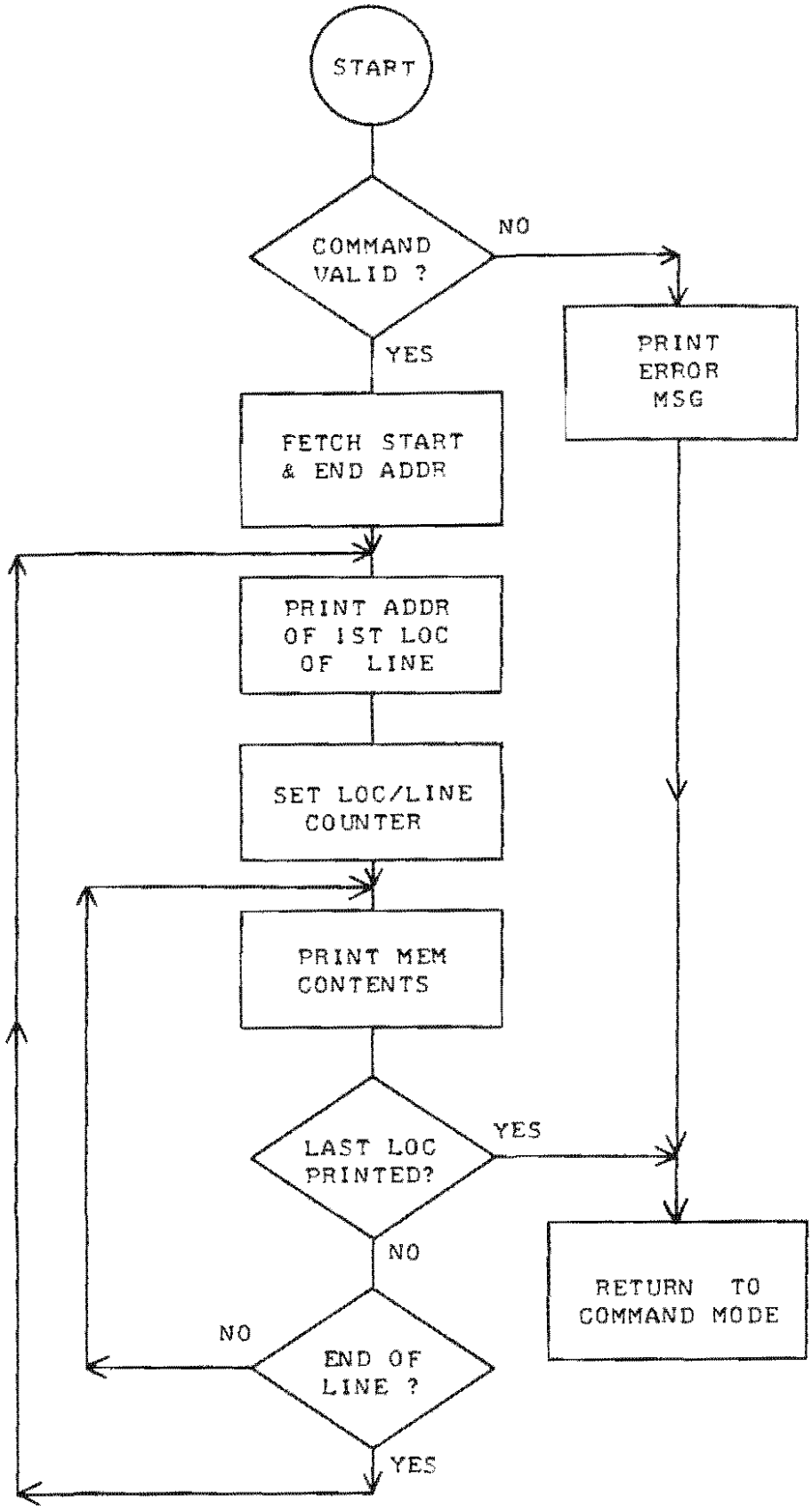
THE DETAILED LISTING FOR THE "DUMP" ROUTINE IS GIVEN BELOW WITH THE FLOW CHART PRESENTED ON THE FOLLOWING PAGE.

```
    MNEMONIC                COMMENTS
 ---------------          --------------

MDUMP, LLI 342          /SET PNTR TO INP BFR
CAL OCTNM               /FETCH MEM DUMP LIMITS
CAL HDLN                /PRINT C/R, L/F
MDMP1, CAL MCONT        /PRINT ADDR OF 1ST LOC
CAL SPAC                /PRINT SPACE
MDMP2, LLI 164          /SET PNTR TO TEMP STRAGE
LMI 020                 /SAVE LOC PER LINE CNTR
OUTAGN, CAL MEMPRT      /PRINT MEM CONTENTS
CAL CKEND               /CHECK FOR LAST LOC PRTD
CAL SPAC                /PRINT SPACE
LLI 164                 /SET PNTR TO L/L CNTR
LBM                     /FETCH CNTR
DCB                     /DECR CNTR
LMB                     /SAVE CNTR. CNTR = 0?
JTZ MDMP1               /YES, START NEW LINE
JMP OUTAGN              /NO, PRINT MORE CONTENTS
```

THE "BULK WRITE" ROUTINE

THE "BULK WRITE" ROUTINE PRESENTED IN THIS MONITOR PROGRAM SIMPLY PROVIDES A SET UP FUNCTION FOR THE USER PROVIDED BULK WRITE OUTPUT ROUTINE. THE PURPOSE OF THIS FUNCTION IS TO PROVIDE A MEANS OF STORING THE CONTENTS OF MEMORY (PROGRAMS OR BLOCKS OF DATA) ON A BULK STORAGE DEVICE VIA A COMMAND FROM THE MONITOR PROGRAM. THE USER'S BULK WRITE ROUTINE IS CALLED BY THIS ROUTINE WITH THE START AND END ADDRESSES OF THE MEMORY LOCATIONS, AS SPECIFIED IN THE COMMAND, STORED IN REGISTERS H AND L FOR THE START LOCATION AND REGISTERS D AND E FOR THE ENDING LOCATION. THIS IS DONE TO MAKE THE INFORMATION READILY AVAILABLE TO THE USER'S BULK WRITE ROUTINE. THE ADDRESSES ARE ALSO CONTAINED IN THE DATA AREA ON PAGE 00, LOCATIONS 166 THRU 171. THE SHORT LISTING FOR THIS ROUTINE IS GIVEN NEXT FOLLOWED BY SOME SUGGESTIONS FOR THE USER'S BULK WRITE OUTPUT ROUTINE.



MEMORY "DUMP" ROUTINE FLOW CHART

```
    MNEMONIC                COMMENTS
 --------------          --------------

WRITE, LLI 342          /SET PNTR TO INP BFR
CAL OCTNM               /FETCH START AND END ADDR
LLI 166                 /SET REG'S H AND L WITH
LCM                     /THE START ADDR AND
INL                     /REG'S D AND E WITH
LBM                     /THE END ADDR OF THE
INL                     /BLOCK OF MEM TO BE
LEM                     /WRITTEN TO THE BULK
INL                     /STORAGE DEVICE.
LDM
LHB
LLC
CAL PUNCH               /GO TO USER BULK WRITE RTN
JMP INCMD               /RET TO COMMAND MODE
```

NOTES AND SUGGESTIONS FOR THE USER PROVIDED BULK STORAGE ROUTINES

WHEN CREATING A BULK STORAGE OUTPUT ROUTINE, ONE SHOULD KEEP SEVERAL FACTORS IN MIND. FIRST, THE DEVICE BEING USED TO STORE THE DATA WILL HAVE TO BE CONSIDERED WHEN DEFINING THE FORMAT FOR STORING THE DATA. FOR EXAMPLE, IF A PAPER TAPE SYSTEM IS USED, THE OUTPUT ROUTINE SHOULD PRECEED THE DATA WITH A SEQUENCE OF "LEADER/TRAILER" CODE, TO GIVE THE READER A PLACE TO START WHEN READING THE TAPE BACK, FOLLOWED BY ADDRESSING INFORMATION AND THEN THE DATA FROM THE SPECIFIED MEMORY LOCATIONS. THE SEQUENCE CAN BE TERMINATED BY EITHER LEADER/TRAILER OR AN "END-OF-DATA" CODE AND THEN LEADER/TRAILER. THE LEADER/TRAILER CODE SHOULD BE A CODE WHICH IS UNIQUE TO THE OTHER DATA CODES TRANSMITTED AND SHOULD PROVIDE ENOUGH LEADER AND TRAILER TO ALLOW EASE OF HANDLING. THE ADDRESSING INFORMATION CAN BE BOTH THE START AND END ADDRESSES OR ONLY THE START ADDRESS WITH THE "END-OF-DATA" CODE OR TRAILER SIGNALING THE END OF THE DATA ON THE TAPE. A SIMILAR FORMAT MAY BE USED FOR A MAGNETIC TAPE SYSTEM.

ANOTHER FACTOR TO CONSIDER IS WHETHER ADDITIONAL INFORMATION IS NEEDED TO EFFECTIVELY USE THE STORAGE DEVICE. FOR EXAMPLE, A DISC UNIT MAY REQUIRE THE SPECIFICATION OF TRACK AND/OR SECTOR NUMBER TO STORE THE DATA. OR, THERE MAY BE SEVERAL DEVICES ON THE SYSTEM WHICH CAN BE USED FOR STORING THE DATA. THIS INFORMATION CAN EASILY BE DEFINED AT THE TIME THE COMMAND IS ENTERED, SINCE THE COMMAND IN STILL AVAILABLE IN THE INPUT BUFFER AREA WHEN THE BULK STORAGE ROUTINES ARE CALLED. SUPPOSE THERE ARE TWO TAPE UNITS ASSOCIATED WITH THE COMPUTER SYSTEM. ONE WILL BE REFERRED TO AS UNIT "A" AND THE OTHER AS UNIT "B." ONE COULD SELECT EITHER TAPE UNIT "A" OR "B" AT THE TIME THE READ OR WRITE COMMAND IS ENTERED BY INCLUDING A LETTER AT THE END OF THE COMMAND WHICH DESIGNATES THE TAPE UNIT TO BE USED. THE FORMAT FOR THE COMMAND MIGHT LOOK LIKE THE FOLLOWING:

W HHH LLL,XXX YYY,A OR R,B

FOR THESE COMMANDS, THE BULK WRITE ROUTINE WOULD WRITE TO TAPE UNIT "A" AND THE BULK READ WOULD CALL UPON TAPE UNIT "B" TO RECEIVE THE DATA. THE USER PROVIDED BULK STORAGE ROUTINES WOULD SIMPLY HAVE TO LOOK IN THE INPUT BUFFER AREA FOR THE UNIT DESIGNATION TO DETERMINE WHICH IS TO BE USED.

ANOTHER POSSIBILITY WOULD BE TO INCLUDE A "DISPLACEMENT" ADDRESS IN THE BULK READ COMMAND. THAT IS, WHEN THE ADDRESS INFORMATION IS READ IN FROM THE STORAGE DEVICE, THE "DISPLACEMENT" ADDRESS WOULD BE "ADDED" TO THE ADDRESS RECEIVED. THIS NEW ADDRESS WOULD BE USED AS THE POINTER INDICATING WHERE TO STORE THE DATA AS IT IS RECEIVED. THUS, DATA THAT WAS WRITTEN TO THE BULK STORAGE FROM PAGE 01 COULD BE READ BACK AND STORED IN PAGE 03, FOR EXAMPLE, BY SPECIFYING A "DISPLACEMENT" ADDRESS OF 002 000.

ABOVE ALL, THE IMPORTANT FACTOR IN WRITING THE BULK STORAGE ROUTINES IS THAT THE DATA WRITTEN BY THE BULK WRITE ROUTINE MUST BE IN A FORMAT THAT CAN BE READ IN BY THE ROUTINE CALLED BY THE BULK READ ROUTINE, DISCUSSED NEXT.

THE "BULK READ" ROUTINE

THE "BULK READ" ROUTINE PRESENTED HERE SIMPLY CALLS THE USER PROVIDED BULK STORAGE READ ROUTINE TO READ IN THE DATA AVAILABLE AT THE SYSTEM BULK STORAGE DEVICE. THE ONLY REAL FUNCTION IT PERFORMS IS THAT OF PROVIDING A MEANS OF ACCESSING THE BULK INPUT DEVICE BY A COMMAND FROM THE KEYBOARD AND ALLOWING A RETURN TO THE MONITOR WHEN THE OPERATION IS COMPLETE.

```
    MNEMONIC                COMMENTS
-------------------     ---------------

RDBULK, CAL READ        /GO TO USER BULK READ RTN
JMP INCMD               /RET TO COMMAND MODE
```

THE ROUTINES PRESENTED TO THIS POINT REQUIRE ONLY 1/2 K OF MEMORY FOR THE OPERATING PORTION, NOT INCLUDING THE USER'S I/O ROUTINES AND OMITTING THE "ADRDTA" SUBROUTINE WHICH HAS NOT BEEN CALLED AS YET. THE USER WITH A LIMITED AMOUNT OF MEMORY MAY DESIRE TO END THE MONITOR PROGRAM HERE, SINCE THE ROUTINES INCLUDED ARE SUFFICIENT TO BE USED AS A SMALL SYSTEM MONITOR. FOR THOSE WITH AN ABUNDANCE OF MEMORY, THE FOLLOWING ROUTINES WILL BE FOUND TO BE VERY HELPFUL IN PROGRAM DEVELOPMENT AND GENERAL SYSTEM OPERATION.

THE "BREAKPOINT" ROUTINE

ONE OF THE MOST DIFFICULT TASKS IN OPERATING A COMPUTER SYSTEM IS THAT OF DEBUGGING PROGRAMS. FINDING OUT EXACTLY WHAT IS HAPPENING TO THIS REGISTER OR THAT MEMORY LOCATION WHEN A NEW PROGRAM IS BEING TRIED OUT CAN BE VERY TIME CONSUMING IF ONE DOES NOT HAVE THE PROPER TOOLS TO AID IN THE PROCESS. ONE "TOOL" THAT CAN BE VERY EFFECTIVE IS A "BREAKPOINT" PROGRAM. A "BREAKPOINT" CAN BE SET AT A PARTICULAR POINT IN A PROGRAM WHICH, WHEN ENCOUNTERED, WILL STOP EXECUTION OF THE PROGRAM, RETURN TO THE MONITOR AND SAVE THE CONTENTS OF THE CPU REGISTERS AND FLAG STATUS AT THE TIME THE BREAKPOINT WAS REACHED. THE PROGRAMMER MAY THEN EXAMINE THE CPU REGISTER'S CONTENTS AND THE CPU FLAG STATUS AND ALSO THE CONTENTS OF MEMORY LOCATIONS, WHICH WILL CONTAIN THEIR VALUES AT THE TIME THE BREAKPOINT WAS ENCOUNTERED. THE BREAKPOINT ROUTINE PRESENTED HERE PERFORMS THIS FUNCTION.

THIS BREAKPOINT ROUTINE IS WRITTEN TO STORE THE CPU REGISTERS IN TWO SEPARATE GROUPS. THE REASON BEING THAT THE 8008 INSTRUCTION SET DOES NOT PROVIDE FOR THE STORAGE OF REGISTERS IN MEMORY UNLESS REGISTERS H AND L HAVE BEEN SET TO POINT TO THE STORAGE LOCATION. THEREFORE, AT LEAST TWO REGISTER VALUES MUST BE SACRIFICED WHEN A BREAKPOINT IS ENCOUNTERED. THIS ROUTINE ALLOWS THE SPECIFICATION OF TWO TYPES OF BREAKPOINTS. A "TYPE 1" BREAKPOINT WILL SAVE THE VALUES OF REGISTERS A, B AND C AND A "TYPE 2" BREAKPOINT WILL SAVE THE VALUES OF REGISTERS D, E, H AND L.

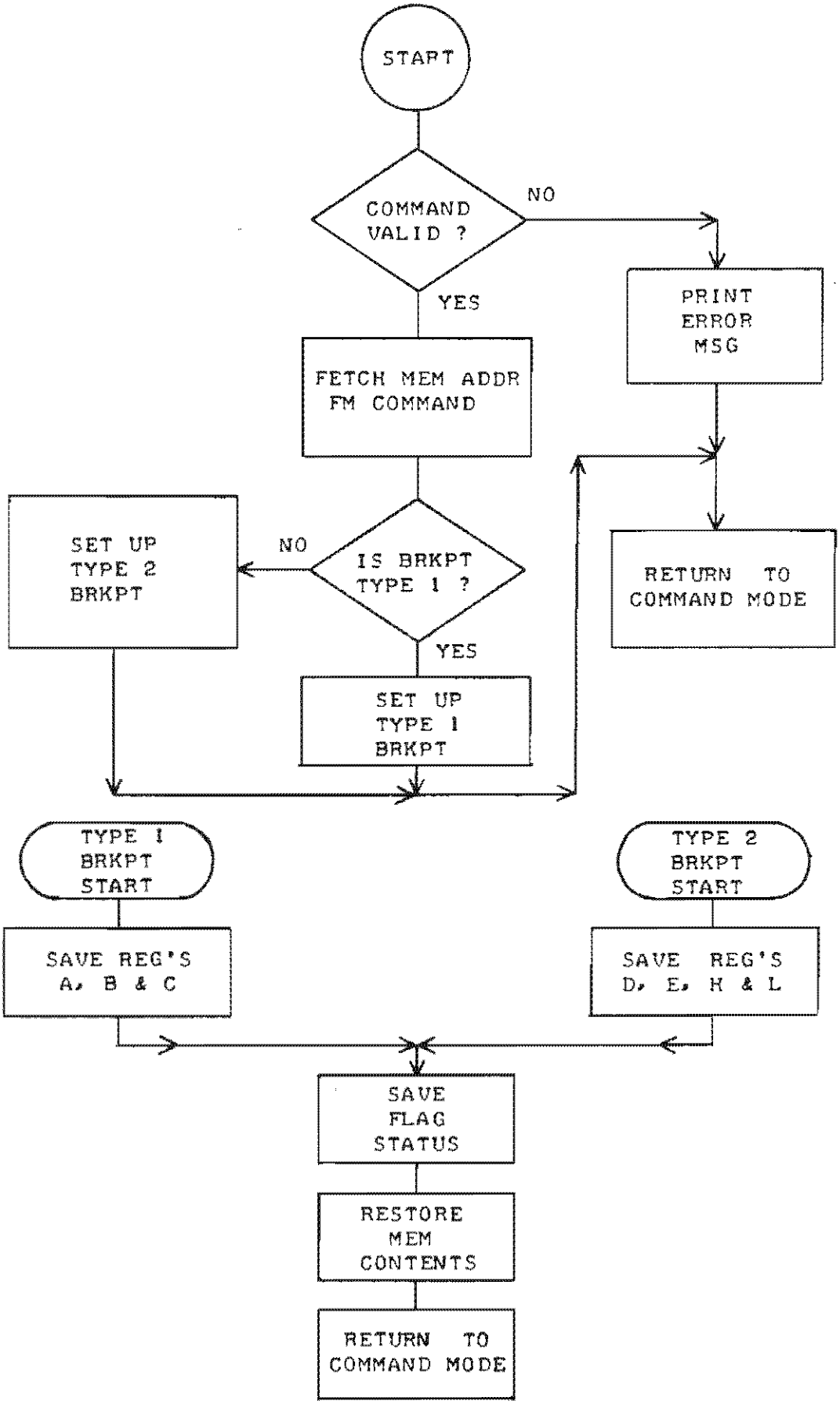
AS NOTED IN THE FLOW CHART ON THE FOLLOWING PAGE, THE BREAKPOINT ROUTINE IS ACTUALLY MADE UP OF TWO SEPARATE ROUTINES. THE FIRST ROUTINE SETS UP THE BREAKPOINT BY STORING A "RESTART 7" INSTRUCTION AT THE LOCATION SPECIFIED IN THE COMMAND AND SAVING THE CONTENTS OF THAT LOCATION SO THAT IT WILL BE RESTORED BACK TO ITS ORIGINAL VALUE AFTER THE BREAKPOINT IS PERFORMED. THE "TYPE" OF BREAKPOINT (1 OR 2) IS THEN DETERMINED FROM THE COMMAND AND THE START ADDRESS FOR THAT TYPE (THE ADDRESS OF "BRK1" FOR A TYPE "1" BREAKPOINT, "BRK2" FOR A TYPE "2" BREAKPOINT) IS STORED AS THE SECOND AND THIRD BYTES OF A JUMP INSTRUCTION AT THE "RESTART 7" LOCATION, PAGE 00 LOCATION 070. IT IS IMPORTANT TO NOTE THAT SHOULD THE BREAKPOINT ROUTINE BE ORIGINED IN A DIFFERENT LOCATION THAN THE ASSEMBLED VERSION PRESENTED IN THIS MANUAL, THE FOUR INSTRUCTIONS WHICH HAVE THE COMMENTS STARTING WITH FOUR ASTERISK'S (****) MUST HAVE THE IMMEDIATE PORTION OF THE INSTRUCTION CHANGED TO INDICATE THE NEW LOW ADDRESS AND PAGE ADDRESS OF THE INSTRUCTIONS LABELED "BRK1" AND "BRK2." THIS FIRST ROUTINE IS LABELED "BREAK."

THE SECOND ROUTINE SHOWN ON THE FLOW CHART IS THE ROUTINE WHICH IS ENTERED AT THE TIME THE BREAKPOINT IS REACHED. IF A TYPE "1" BREAKPOINT WAS SET, THE CONTENTS OF REGISTER'S A, B AND C WILL BE STORED IN THE "VIRTUAL" CPU REGISTER TABLE (PAGE 00 LOCATION 200 THRU 206). FOR A TYPE "2" BREAKPOINT, THE CONTENTS OF REGISTER'S D, E, H AND L WILL BE STORED. THE EXPERIENCED PROGRAMMER WILL OBSERVE THAT IN STORING THESE REGISTERS, ONLY INSTRUCTIONS WHICH DO NOT AFFECT THE CONDITION OF THE FLAG STATUS OF THE CPU ARE USED. THUS, ONCE THE REGISTER VALUES ARE SAFELY STORED IN THE "VIRTUAL" CPU REGISTER TABLE, THE CONDITION OF THE FLAG STATUS MAY BE TESTED AND A SPECIAL BYTE IS FORMED AND STORED AT PAGE 00 LOCATION 207 WHICH INDICATES WHICH FLAGS WERE SET AT THE TIME THE BREAKPOINT WAS REACHED. THE FOLLOWING BITS WILL BE SET TO A "1" FOR A TRUE CONDITION OF THE RESPECTIVE FLAGS. BIT 0 INDICATES THE CONDITION OF THE CARRY FLAG, BIT 3 INDICATES THE CONDITION OF THE ZERO FLAG, BIT 6 INDICATES THE CONDITION OF THE PARITY FLAG AND BIT 7 INDICATES THE CONDITION OF THE SIGN FLAG. THE FINAL STEP IN THE BREAKPOINT PROCESS, BEFORE RETURNING TO THE COMMAND MODE, IS TO RESTORE THE INSTRUCTION AT THE BREAKPOINT LOCATION TO ITS ORIGINAL CONTENTS.

THE LISTINGS FOR THE BREAKPOINT ROUTINES ARE PRESENTED NEXT.

```
    MNEMONIC                COMMENTS
 ---------------          ---------------

BREAK, CAL ANLYZ      /SET UP ADDRESS OF BP
LLE
LHD
JTZ B1                /DETERMINE IF B1 OR B2
CPI 262
JFZ ERR               /ERROR IF NEITHER
/
```



THE "BREAKPOINT" ROUTINE FLOW CHART

```
    MNEMONIC                COMMENTS
 --------------          --------------
B2, CAL SETBK          /SET UP BP RST COMMAND
LMI 176                /**** STORE BP2 FND LA
INL
LMI 016                /**** STORE BP2 FND PG
JMP FINBK              /TO REST OF BP SET UP RTN
/
B1, CAL SETBK          /SET UP BP RST COMMAND
LMI 112                /**** STORE BP1 FND LA
INL
LMI 016                /**** STORE BP1 FND PG
FINBK, INL
LME                    /STORE BP ORIG LOW ADDR
INL
LMD                    /STORE BP ORIG PG ADDR
INL
LMA                    /STORE ORIG BP INSTRUCTION
JMP INCMD
/
ANLYZ, LLI 343         /SET PNTR TO BUFF SA
CAL OCTNM              /FETCH ADDR INTO 166, 167
LLI 341                /RESTORE BUFF SA
LAM                    /GET BP 1 OR 2 COMMAND
LLI 166                /GET BP LOW ADDR
LEM                    /INTO "E"
INL
LDM                    /AND BP PAGE
LLI 156                /PNTR TO JUMP COMMAND
LME
INL                    /SET UP JUMP ADDRESS
LMD
CPI 261                /DETERMINE IF CMND 1 OR 2
RET
/
SETBK, LAM             /SAVE ORIG CONTENTS OF BP
LMI 075                /INSERT BP RESTART INSTR
LEL                    /CHANGE POINTERS
LDH
LHI 000                /SET PAGE 00
LLI 070                /SET PNTR TO RST 1 LOC
LMI 104                /STORE JUMP INSTRUCTION
INL
RET
/
BRK1, LHI 000          /SET PAGE 00
LLI 200                /CPU REGISTER STORAGE LOCS
LMA                    /SAVE ACCUMULATOR
LLI 201
LMB                    /AND CPU REGS B & C
LLI 202
BRKCOM, LMC            /WITHOUT DISTURBING FLAGS
FLAGT, LAH             /SET UP TEMP REGS
LBA
LCA
JFC NOC                /TEST FOR CARRY FLAG
LAI 001                /SET 1 IN "A" IF CARRY TRUE
NOC, JFZ NOZ           /TEST FOR ZERO FLAG
LBI 010                /SET 1 IN "B" IF ZERO TRUE
```

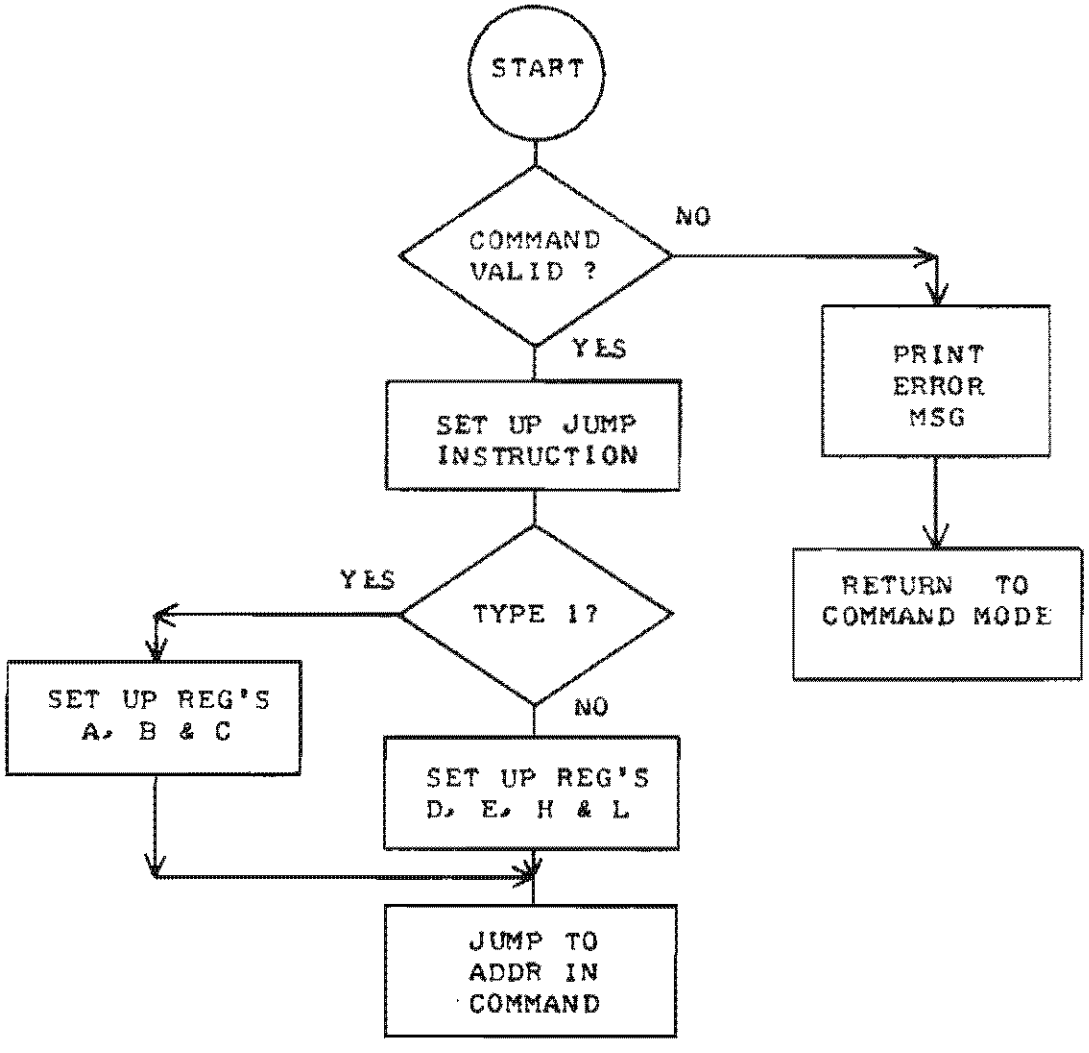
```
    MNEMONIC                COMMENTS
 --------------          --------------

NOZ, JFP NOP            /TEST FOR PARITY FLAG
LCI 100                 /SET 1 IN "C" IF PARITY "T"
NOP, JFS NOS            /TEST FOR SIGN FLAG
ADI 200                 /SET MSB IF SIGN TRUE
NOS, ADB
ADC                     /FORM FLAG STATUS BYTE
LLI 207
LMA                     /STORE FLAG STATUS
LLI 073                 /PNTR TO ORIG BP LOW ADDR
LEM                     /GET ORIG LOC OF BP
INL
LDM                     /AND ORIG PG OF BP
INL
LAM                     /AND ORIG BP INSTRUCTION
LLE                     /SET UP ORIGINAL
LHD                     /BREAK POINT POINTERS
LMA                     /RESTORE ORIG BKPNT INSTR
JMP INCMD               /BACK TO MONITOR
/
BRK2, LBH               /SAVE ORIG VALUE OF H & L
LCL
LHI 000                 /SET PNTR TO PAGE 00
LLI 203                 /CPU REGISTER STORAGE LOCS
LMD                     /SAVE REGS D AND E
LLI 204                 /AS WELL AS ORIG H AND L
LME
LLI 205                 /WITHOUT DISTURBING FLAGS
LMB
LLI 206
JMP BRKCOM              /TO REST OF BREAKPT RTN
```

THE "GO TO" ROUTINE

THE "GO TO" ROUTINE PROVIDES A MEANS OF INITIATING EXECUTION OF A PROGRAM IN MEMORY BY DIRECTING THE MONITOR TO JUMP TO A SPECIFIED ADDRESS. AFTER FETCHING THE ADDRESS FROM THE COMMAND, THE "GO TO" ROUTINE DETERMINES WHICH "TYPE" OF GO TO IS REQUESTED. THAT IS, THE "GO TO" FUNCTION ALLOWS THE SETTING OF A GROUP OF CPU REGISTERS BEFORE JUMPING TO THE PROGRAM. THE TWO GROUPS ARE THE SAME AS THOSE FOR THE BREAKPOINT ROUTINE. A TYPE "1" "GO TO" WILL SET THE VALUES OF REGISTERS A, B AND C FROM THE "VIRTUAL" CPU REGISTER TABLE WHILE A TYPE "2" "GO TO" WILL SET THE VALUES OF REGISTERS D, E, H AND L. THE VALUES IN THE "VIRTUAL" CPU REGISTER TABLE ARE SET UP BY EITHER THE "BREAKPOINT" ROUTINE OR BY THE "EXAMINE REGISTER" ROUTINE TO BE PRESENTED NEXT. THE "GO TO" ROUTINE STARTS AT THE LOCATION LABELED "GOTO." THE LISTING AND FLOW CHART ARE PRESENTED ON THE NEXT PAGE. THE READER WILL NOTE THAT THE "ANLYZ" SUBROUTINE OF THE BREAKPOINT ROUTINE IS ALSO USED BY "GO TO" TO FETCH THE START OF EXECUTION ADDRESS AND FORM THE JUMP INSTRUCTION WHICH IS THE FINAL STEP IN THE "GO TO" ROUTINE.

```
    MNEMONIC                COMMENTS
 ---------------          ---------------
GOTO, CAL ANLYZ         /SET UP ADDR OF GOTO
JTZ G01                 /TO SET UP CPU REGS A,B,C
CPI 262
JFZ ERR                 /ERROR IF NOT G1 OR G2
/
G02, LLI 203            /SET UP CPU REGS D,E,H & L
LDM
INL
LEM
GOCOM, INL
LBM
INL
LCM
LLC
LHB
JMP 155 000
/
G01, LLI 200            /SET UP CPU REGS A,B,C
LAM
JMP GOCOM
```



THE "GO TO" ROUTINE FLOW CHART

THE "EXAMINE REGISTER" ROUTINE

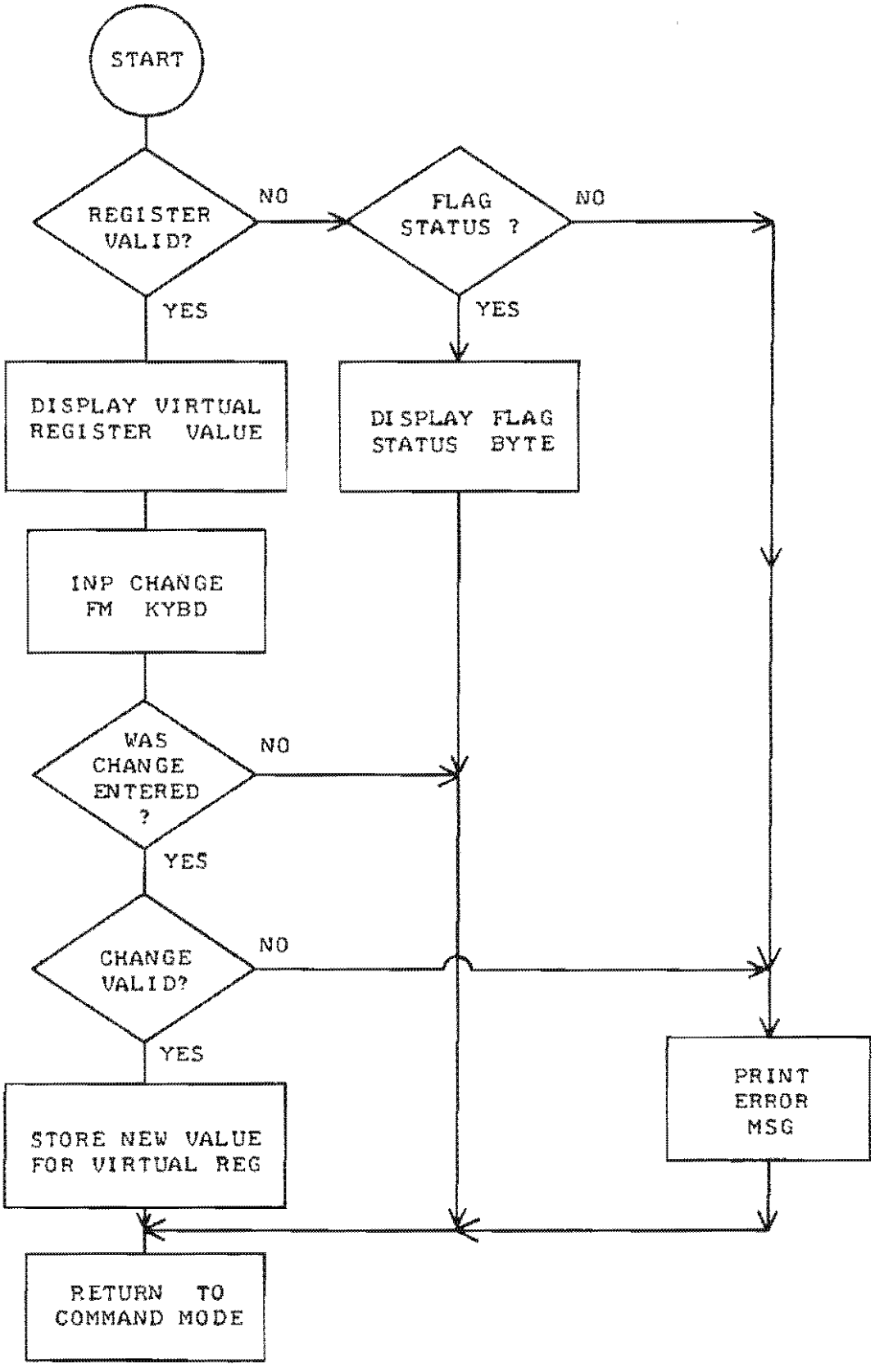
THE "EXAMINE REGISTER" ROUTINE ALLOWS ONE TO EXAMINE THE CONTENTS OF THE "VIRTUAL" CPU REGISTERS AND THE FLAG STATUS WHICH ARE STORED IN A TABLE ON PAGE 00 AT LOCATIONS 200 THRU 207. THE CONTENTS OF REGISTERS A THRU E ARE STORED IN LOCATIONS 200 THRU 204 RESPECTIVELY, REGISTER H IN 205, REGISTER L IN 206 AND THE FLAG STATUS BYTE IN 207. THE CONTENTS OF THE "VIRTUAL" REGISTERS MAY BE MODIFIED BY ENTERING THE REVISION AFTER THE CURRENT VALUE IS PRINTED. HOWEVER, THE FLAG STATUS IS DISPLAYED SOLELY TO ALLOW EXAMINATION OF THE STATE OF THE FLAGS AT THE TIME THE LAST BREAKPOINT WAS EXECUTED.

THIS ROUTINE STARTS BY FETCHING THE REGISTER DESIGNATION FROM THE INPUT BUFFER AND DETERMINING WHICH IS SPECIFIED. IF A "VIRTUAL" CPU REGISTER IS SPECIFIED, A POINTER IS FORMED TO INDICATE WHICH LOCATION IN THE TABLE IS TO BE DISPLAYED. THE CURRENT VALUE IS PRINTED, FOLLOWED BY A COLON, AND THEN THE "INSPCL" SUBROUTINE IS CALLED TO INPUT ANY CHANGES THE OPERATOR MAY DESIRE TO MAKE TO THE VALUE STORED FOR THAT REGISTER. IF NO MODIFICATION IS ENTERED, THE ROUTINE SIMPLY RETURNS TO THE COMMAND MODE AND THE ORIGINAL CONTENT IS MAINTAINED. IF A MODIFICATION IS ENTERED, THE "DCDNM" SUBROUTINE CONVERTS THE INPUT TO BINARY FORM AND THE NEW VALUE IS STORED IN THE TABLE. IF THE FLAG STATUS IS REQUESTED, THE VALUE CURRENTLY STORED AT LOCATION 207 ON PAGE 00 WILL BE PRINTED AND THE ROUTINE AUTOMATICALLY RETURNS TO THE COMMAND MODE. IF THE REGISTER DESIGNATION IS NOT VALID, THE ILLEGAL ENTRY ERROR MESSAGE IS DISPLAYED.

THE DETAILED LISTING FOR THE "EXAMINE REGISTER" ROUTINE IS PRESENTED BELOW AND THE FLOW CHART IS ON THE FOLLOWING PAGE.

```
    MNEMONIC                COMMENTS
 --------------          --------------

XREG, LLI 341          /SET INP BFR PNTR
LAM                    /FETCH REG LETTER
RGAGN, CPI 301         /IS REG VALID?
JTC ERR                /NO, PRINT ERROR
CPI 306                /YES, IS REG A THRU E?
JFC FHL                /NO, TRY H, L OR F
SUI 101                /SET UP REG TBL PNTR
XCOM, LLI 164          /SAVE TBL PNTR IN TEMP STRAGE
LMA
LLA                    /SET PNTR TO REG TBL LOC
CAL SPAC               /PRINT SPACE
LAM                    /FETCH CURRENT REG VALUE
CAL OCTOUT             /PRINT CURRENT REG VALUE
CAL COLON              /PRINT COLON
CAL INSPCL             /INP MODIFICATION
LEI 340                /SET INP BFR PNTR
LAL
CPE                    /WAS MOD ENTERED?
JTZ INCMD              /NO, RET TO COMMAND MODE
CAL DCDNM              /YES, DECODE OCTAL NUMBER
LLI 164                /SET PNTR TO TEMP STRAGE
LLM                    /FETCH REG TBL PNTR
LMB                    /STORE NEW REG VALUE
JMP INCMD              /RET TO COMMAND MODE
```



THE "EXAMINE REGISTER" ROUTINE FLOW CHART

```
    MNEMONIC                COMMENTS
 ---------------          ---------------
/
FHL, CPI 310            /IS REG = H?
JFZ LORF                /NO, TRY L OR F
LAI 205                 /YES, SET REG TBL PNTR
JMP XCOM                /INP MOD TO REG VALUE
/
LORF, CPI 314           /IS REG = L?
JFZ F                   /NO, TRY F
LAI 206                 /YES, SET REG TBL PNTR
JMP XCOM                /INP MOD TO REG VALUE
/
F, CPI 306              /IS REG = F, FOR FLAGS?
JFZ ERR                 /NO, PRINT ERROR
CAL SPAC                /PRINT SPACE
LLI 207                 /SET REG TBL PNTR
LAM                     /FETCH FLAG WORD
CAL OCTOUT              /PRINT FLAG WORD
JMP INCMD               /RET TO COMMAND MODE
```

THE THREE ROUTINES JUST PRESENTED ARE ALL INTER-RELATED IN ONE WAY OR ANOTHER. THE "EXAMINE REGISTER" ROUTINE SETS UP THE VALUES TO BE LOADED IN THE CPU REGISTERS AT THE TIME THE "GO TO" OPERATION IS PERFORMED. THE "GO TO" ROUTINE MAY START THE EXECUTION OF A PROGRAM WHICH WILL EVENTUALLY REACH A "BREAKPOINT" WHICH RETURNS TO THE "BREAKPOINT" ROUTINE TO STORE THE CPU REGISTER VALUES AND THE FLAG STATUS, WHICH, IN TURN MAY BE EXAMINED BY THE "EXAMINE REGISTER" ROUTINE. THIS COORDINATION BETWEEN THESE ROUTINES MAKES THE INCLUSION OF THESE ROUTINES, AS A GROUP, A CONVENIENT POINT TO COMPLETE ONE'S MONITOR PROGRAM. THE OPERATING PORTION OF THE MONITOR PROGRAM PRESENTED TO THIS POINT OCCUPIES SLIGHTLY MORE THAN 3/4 K BYTES OF MEMORY. SO, IF ONE FEELS THAT THE ROUTINES PRESENTED THUS FAR WILL BE SUFFICIENT FOR ONE'S MONITOR PROGRAM, THE PROGRAM CAN BE ENDED HERE AND USED TO GIVE THE OPERATOR THE NECESSARY BASICS FOR A GOOD "OPERATING SYSTEM" AND "PROGRAM DEBUGGING" MONITOR PROGRAM. THE FOLLOWING ROUTINES ARE PRESENTED TO GIVE THE READER AN IDEA FOR OTHER TYPES OF "CONVENIENCE" ROUTINES THAT MAY BE ADDED.

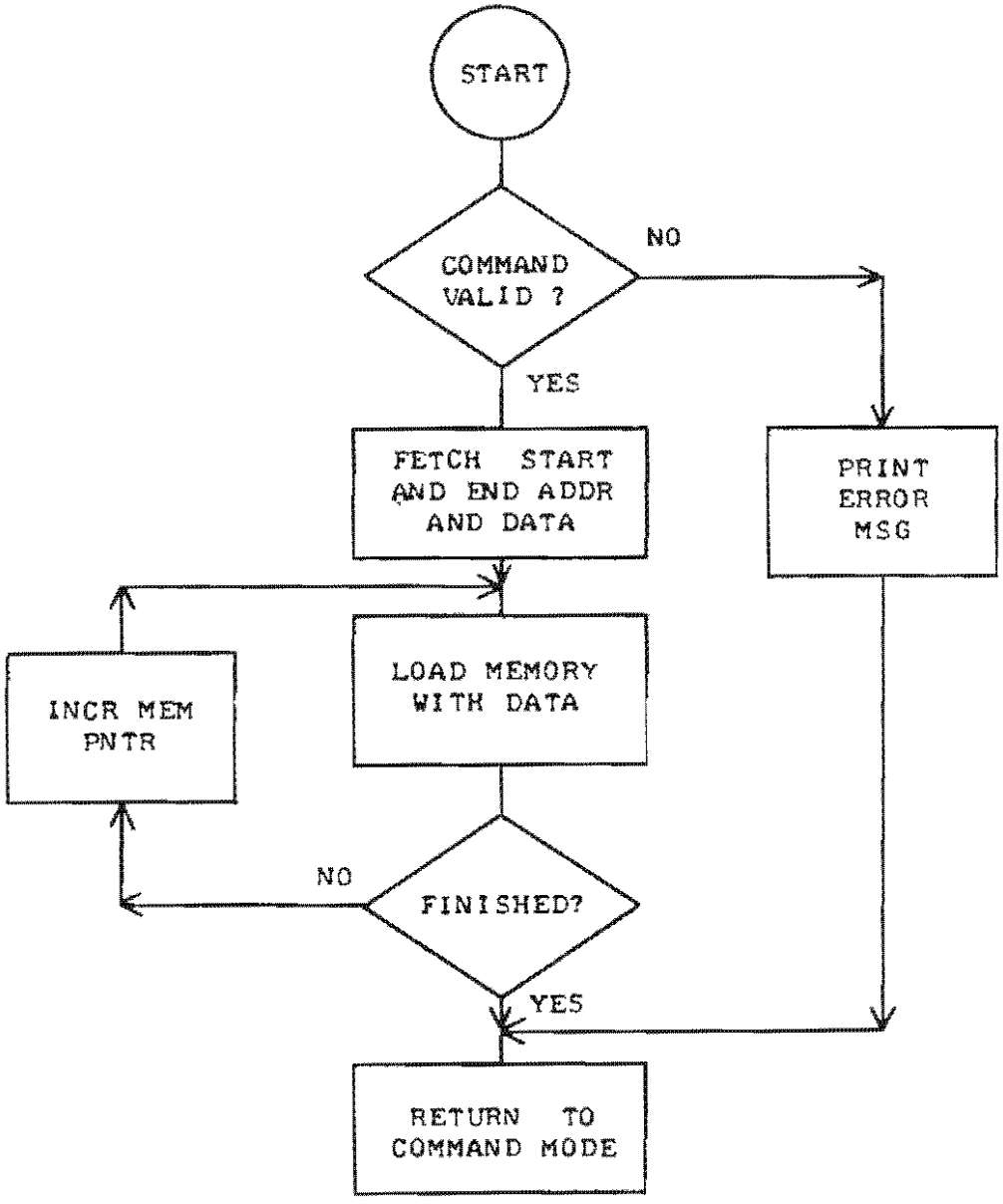
THE "FILL" ROUTINE

THE MEMORY "FILL" ROUTINE IS USED TO FILL A BLOCK OF MEMORY WITH A SPECIFIC 8 BIT DATA VALUE. THIS ROUTINE IS USEFUL IN "ZEROING" A BLOCK OF MEMORY BEFORE EXECUTING A PROGRAM TO DETERMINE WHETHER THAT PROGRAM IS WRITING INTO THE SECTION OF MEMORY "ZEROED" OUT OR NOT. AS THE READER WILL SEE FROM THE LISTING, THIS PROGRAM MAKES VERY EFFECTIVE USE OF SUBROUTINES TO PERFORM ITS FUNCTION. THE "ADRDTA" SUBROUTINE FETCHES THE PERTAINENT INFORMATION FROM THE INPUT BUFFER. THE "SETUP" SUBROUTINE SETS THE MEMORY POINTER TO THE MEMORY LOCATION TO RECEIVE THE DATA BYTE, AND THE "CKEND" SUBROUTINE DETERMINES WHEN THE FINAL LOCATION HAS BEEN LOADED.

THE PROGRAM LISTING AND FLOW CHART FOR THE "FILL" ROUTINE IS PRESENTED ON THE NEXT PAGE.

```
    MNEMONIC                    COMMENTS
 --------------              --------------

FILL, CAL ADRDTA        /INP ADDR AND DATA FM BFR
FL1, CAL SETUP          /SET UP MEM PNTR
LMB                     /FILL MEM LOC WITH DATA
CAL CKEND               /DONE? YES, RET TO CMND MODE
JMP FL1                 /NO, CONTINUE WITH FILL
```



THE MEMORY "FILL" ROUTINE FLOW CHART

THE "SEARCH" ROUTINE

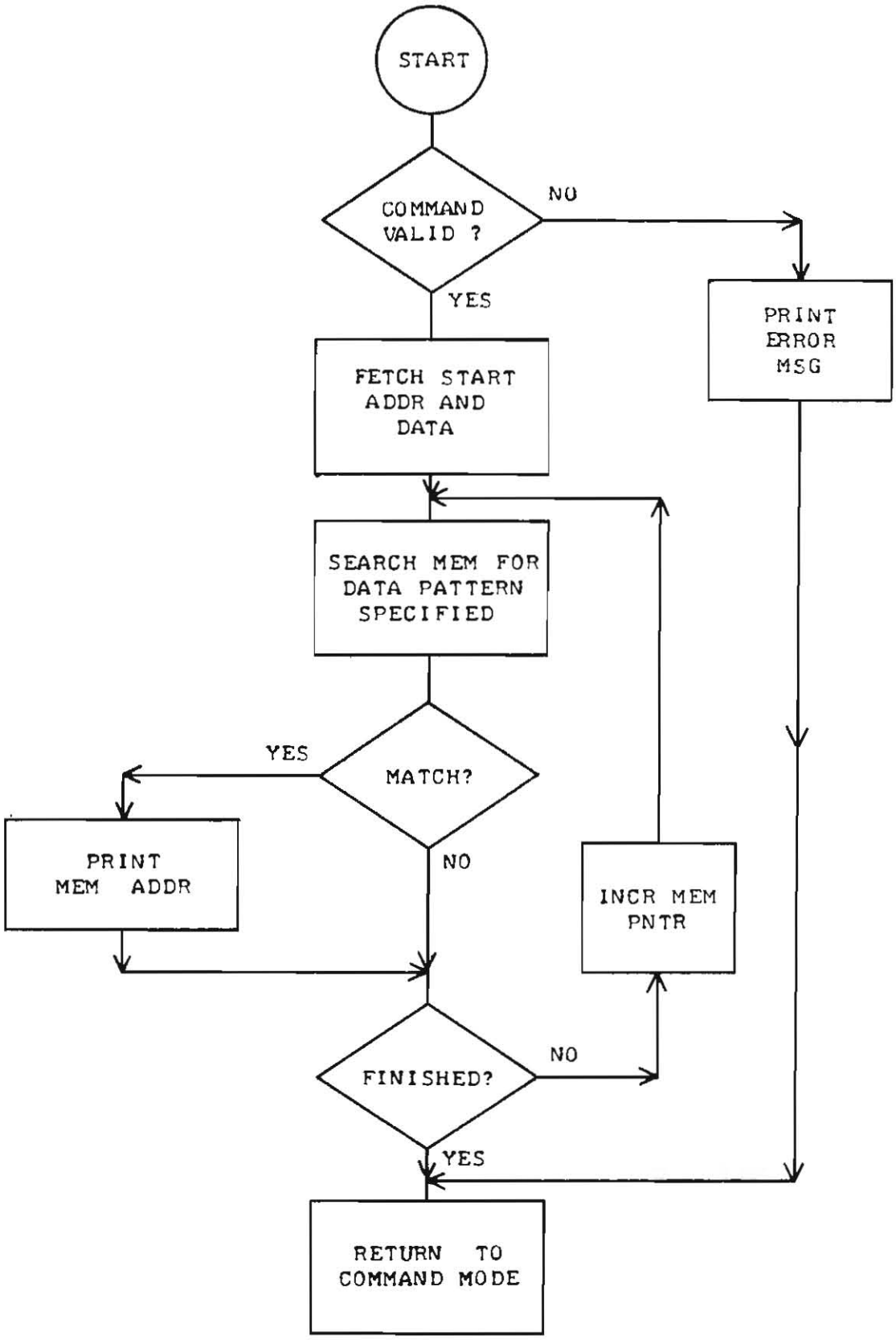
THE MEMORY "SEARCH" ROUTINE IS USED TO SEARCH THE CONTENTS OF A SPECIFIED BLOCK OF MEMORY FOR AN 8 BIT DATA PATTERN ENTERED IN THE COMMAND. EACH TIME IT FINDS A BYTE WHICH MATCHES THE PATTERN, THE ADDRESS OF THE MATCHING BYTE IS PRINTED ON THE DISPLAY DEVICE. THE ROUTINE FETCHES THE ADDRESS BLOCK AND SEARCH DATA FROM THE INPUT BUFFER BY CALLING THE "ADRDTA" SUBROUTINE. THE BLOCK OF DATA IS SEARCHED BY COMPARING EACH LOCATION IN THE BLOCK TO THE DATA PATTERN ENTERED AND, IF A MATCH IS FOUND, THE "MCONT" SUBROUTINE, WHICH PRINTS A CARRIAGE RETURN, LINE FEED FOLLOWED BY THE MEMORY ADDRESS STORED AT LOCATION 166 ON PAGE 00, IS CALLED TO PRINT THE MEMORY ADDRESS WHICH CONTAINS THE MATCH. THE PROCESS CONTINUES UNTIL THE LAST LOCATION SPECIFIED IN THE COMMAND IS SEARCHED. ONCE AGAIN THE EFFECTIVENESS OF GOOD GENERAL SUBROUTINES IS EVIDENCED BY THE BREVITY OF THIS ROUTINE. THE DETAILED LISTING IS SHOWN BELOW AND THE FLOW CHART ON THE NEXT PAGE.

MNEMONIC	COMMENTS
SEARCH, CAL ADRDTA	/INP ADDR AND DATA FM BFR
LLI 165	/SET PNTR TO SAVE DATA
LMB	/SAVE SEARCH DATA IN MEM
SH1, LLI 165	/SET PNTR TO SRCH DATA
LAM	/FETCH SEARCH DATA
CAL SETUP	/FETCH CONTENTS OF MEM
CPM	/DATA EQUAL SRCH DATA?
CTZ MCONT	/YES, PRINT ADDR
CAL CKEND	/DONE? YES, RET TO CMND MODE
JMP SH1	/NO, CONTINUE SEARCH

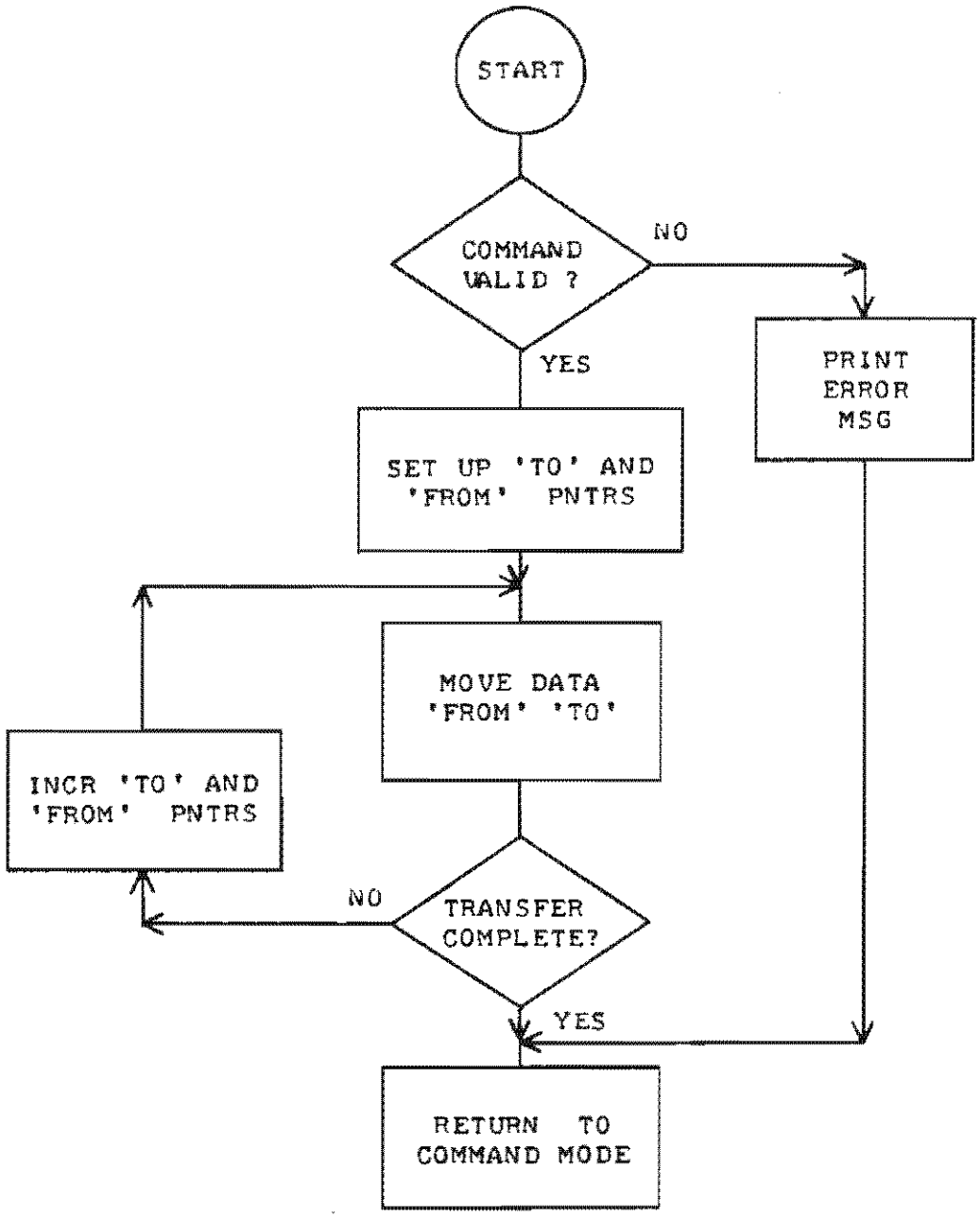
THE "TRANSFER" ROUTINE

THE "TRANSFER" ROUTINE ALLOWS THE OPERATOR TO TRANSFER A BLOCK OF MEMORY FROM ONE SECTION OF MEMORY TO ANOTHER, BY SIMPLY SPECIFYING THE START AND END ADDRESS OF THE BLOCK TO BE MOVED, FOLLOWED BY THE START ADDRESS OF THE SECTION TO RECEIVE THE MEMORY CONTENTS IN THE COMMAND. THE "TRANSFER" ROUTINE THEN SETS UP A "FROM" POINTER AND A "TO" POINTER WHICH ARE USED TO TRANSFER THE THE DATA "FROM" THE ORIGINAL LOCATION "TO" THE NEW LOCATION. THIS ROUTINE USES A SUBROUTINE CALLED "SWAP" NOT ONLY DURING THE ACTUAL TRANSFER OF THE DATA BUT ALSO TO TEMPORARILY SAVE THE ADDRESSES AS THEY ARE READ IN FROM THE INPUT BUFFER. THIS COMMAND CAN BE USEFUL IN SAVING A BLOCK OF DATA IN ONE SECTION OF MEMORY BEFORE USING THE ORIGINAL DATA AREA AGAIN. AFTER THE SECOND USAGE, THE TWO BLOCKS WILL BE AVAILABLE FOR EXAMINATION AND/OR COMPARISION. ANOTHER POSSIBLE APPLICATION IS TO RE-ORIGIN A PROGRAM FROM ONE AREA OF MEMORY TO ANOTHER. OF COURSE, THE JUMP AND CALL INSTRUCTIONS WOULD HAVE TO BE CHANGED TO INDICATE THE NEW ADDRESSES, BUT THIS CAN BE ASSISTED BY USING THE "SEARCH" ROUTINE TO LOCATE THE JUMP AND CALL INSTRUCTIONS WITHIN THE PROGRAM. THIS METHOD OF MOVING PROGRAMS CAN BE EFFECTIVE FOR PROGRAMS WHICH ARE NOT TOO LONG, AS OPPOSED TO RE-ASSEMBLING THE PROGRAM.

THE FLOW CHART AND LISTING FOR THE "TRANSFER" ROUTINE ARE PRESENTED FOLLOWING THE "SEARCH" ROUTINE FLOW CHART.



THE "SEARCH" ROUTINE FLOW CHART



THE MEMORY "TRANSFER" ROUTINE FLOW CHART

```
    MNEMONIC                COMMENTS
 ---------------          ---------------
TRNSFR, LLI 342         /SET PNTR TO ADDR INP
CAL OCTNM               /FETCH 'FROM' ADDR
LLI 166                 /SET PNTR TO ADDR INP
LBE                     /SAVE INP BFR PNT
LEI 172                 /SAVE 'FROM' IN TEMP STRGE
LDH
SVSA, CAL SWAP          /MOVE ADDR TO TEMP STRGE
LAI 172                 /IS XFR COMPLETE?
CPL
JFZ SVSA                /NO, CONTINUE MOVE
```

```
    MNEMONIC                COMMENTS
-------------------     -------------
INB
LLB                     /RESTORE INP BFR PNTR
CAL OCTNM               /INP 'TO' ADDR
LLI 176                 /SET PNTR TO SAVE 'TO' ADDR
LMB                     /SAVE LO ADDR
INL
LMC                     /SAVE PG ADDR
LLI 172                 /SET PNTR TO TEMP STRGE
LEI 166                 /SET TO MOVE 'FROM' BACK
LDH
TF1, CAL SWAP           /XFR 'FROM'
LAI 176
CPL                     /XFR COMPLETE?
JFZ TF1                 /NO, CONTINUE
LEM                     /FETCH 'TO' PNTR
INL
LDM
TF2, CAL SETUP          /SET 'FROM' PNTR
CAL SWAP                /SWAP MEM CONTENTS
CAL CKEND               /DONE? YES, RET TO CMND MODE
JMP TF2                 /NO, CONTINUE XFR
/
SWAP, LAM               /FETCH BYTE TO XFR
CAL INMEM               /INCR 'FROM' PNTR
CAL SWITCH              /CHANGE PNTRS
LMA                     /STORE BYTE IN NEW LOC
CAL INMEM               /INCR 'TO' PNTR
JMP SWITCH              /CHANGE PNTRS AND RET
```

PUTTING IT ALL TOGETHER - THE ASSEMBLED MONITOR PROGRAM

AND AFTER ALL IS SAID AND DONE, HERE IT IS! THE MONITOR PROGRAM PRESENTED IN ITS FINAL ASSEMBLED FORM. THE ROUTINES DISCUSSED ARE NOW LISTED WITH THEIR ADDRESSES AND MACHINE CODE TO PROVIDE THE READER WITH A MONITOR PROGRAM THAT SIMPLY REQUIRES THE ADDITION OF THE I/O DRIVERS (DETAILED PREVIOUSLY) TO TURN ONE'S COMPUTER SYSTEM INTO A HIGHLY FUNCTIONAL "OPERATING SYSTEM!"

THE FIRST PART OF THE LISTING SHOWS THE LOCATIONS ON PAGE 00 WHICH ARE USED BY THE MONITOR FOR STORING POINTERS, COUNTERS, TEMPORARY DATA, THE COMMAND LOOK UP TABLE AND THE INPUT BUFFER. THE READER WILL NOTE THAT SEVEN OF THE EIGHT RESTART LOCATIONS ARE AVAILABLE FOR THE USER'S PROGRAMS.

THE OPERATING PORTION OF THE MONITOR PROGRAM HAS BEEN ORIGINED ON PAGES 14 THROUGH THE FIRST HALF OF PAGE 17, WITH THE EXPECTED STARTING LOCATIONS OF THE USER PROVIDED I/O DRIVERS ON THE SECOND HALF OF PAGE 17. THE READER MAY DESIRE TO RE-ORIGIN THE OPERATING PORTION TO THE UPPER SECTION OF THE MEMORY AVAILABLE IN ONE'S SYSTEM.

THE START OF EXECUTION ADDRESS FOR THE MONITOR PROGRAM, AS LISTED, IS AT PAGE 14 LOCATION 000.

```
000 000                 ORG 000 070
000 070                 /
000 070  104 000 000    JMP 000 000          /JUMP INSTRUCTION FOR BRKPT
000 073  000            000                  /BRKPT LOCATION - LOW ADDR
000 074  000            000                  /BRKPT LOCATION - PG ADDR
000 075  000            000                  /ORIG. BRKPT INSTRUCTION
000 076                 /
000 076                 /LOC. 076 THRU 127 AVAILABLE FOR USER
000 076                 /
000 076                 /MONITOR MESSAGE TABLE
000 076                 /
000 076                 ORG 000 130
000 130  215            215                  /CAR. RET.
000 131  212            212                  /LINE FEED
000 132  276            276                  />
000 133  000 207        000
000 134  215 000        215                  /CAR. RET.
000 135  212 215        212                  /LINE FEED
000 136  000 212        000
000 137  000           /
000 138                 /LOC. 138 THRU 147 AVAILABLE FOR USER
000 138                 /
000 138                 ORG 000 150
000 150                 /
000 150  000            000                  /DIGIT STORAGE
000 151  000            000                  /FOR OCTAL NUMBER
000 152  000            000                  /SUBROUTINE
000 153  000            000                  /AVAILABLE
000 154  000            000                  /AVAILABLE
000 155                 /
000 155                 /COMMAND AND GO TO JUMP INSTRUCTION
000 155                 /
000 155  104 000 000    JMP 000 000          /CMND RTN FILLS IN ADDR
000 160                 /
000 160  000            000                  /AVAILABLE
000 161  000            000                  /AVAILABLE
000 162  000            000                  /AVAILABLE
000 163  000            000                  /AVAILABLE
000 164  000            000                  /TEMP STORAGE
000 165  000            000                  /TEMP STORAGE
000 166  000            000                  /LOW ADDRESS - LOW PORTION
000 167  000            000                  /LOW ADDRESS - PAGE PORTION
000 170  000            000                  /HIGH ADDRESS - LOW PORTION
000 171  000            000                  /HIGH ADDRESS - PAGE PORTION
000 172  000            000                  /TEMP STORAGE
000 173  000            000                  /TEMP STORAGE
000 174  000            000                  /TEMP STORAGE
000 175  000            000                  /TEMP STORAGE
000 176  000            000                  /TEMP STORAGE
000 177  000            000                  /TEMP STORAGE
000 200                 /
000 200  000            000                  /VIRTUAL CPU REG "A"
000 201  000            000                  /VIRTUAL CPU REG "B"
000 202  000            000                  /VIRTUAL CPU REG "C"
000 203  000            000                  /VIRTUAL CPU REG "D"
000 204  000            000                  /VIRTUAL CPU REG "E"
000 205  000            000                  /VIRTUAL CPU REG "H"
000 206  000            000                  /VIRTUAL CPU REG "L"
000 207  000            000                  /FLAG STATUS BYTE
```

```
000 210                 /COMMAND LOOK UP TABLE
000 210                 /
000 210  315            315                     /MODIFY
000 211  150            150
000 212  015 74         015
000 213  304            304                     /DUMP
000 214  275            275
000 215  015 74         015
000 216  327            327                     /BULK WRITE
000 217  343            343
000 220  015 74         015
000 221  322            322                     /BULK READ
000 222  371            371
000 223  015 74         015
000 224  302            302                     /BREAKPOINT
000 225  377            377
000 226  015 74         015
000 227  307            307                     /GO TO
000 230  220            220
000 231  016 75         016
000 232  330            330                     /EXAMINE REGISTERS
000 233  257            257
000 234  016 75         016
000 235  306            306                     /FILL MEM
000 236  005            005
000 237  017 76         017
000 240  323            323                     /SEARCH
000 241  022            022
000 242  017 76         017
000 243  324            324                     /TRANSFER
000 244  061            061
000 245  017 76         017
000 246                 /
000 246                 /LOC. 246 THRU 337 AVAILABLE FOR
000 246                 /USER OR TO EXPAND COMMAND TABLE
000 246                 /
000 246                 /LOC. 340 THRU 377 - INPUT BUFFER
000 246                 /
000 246                 /PAGES 01 THRU 13 (72) AVAILABLE
000 246                 /FOR USER'S PROGRAMS
000 246                 /
000 246                 /
000 246                 /COMMMAND INPUT ROUTINE
000 246                 /
000 246                 ORG 014 000
73 014 000  056 000     INCMD, LHI 000          /SET PNTR TO HEADING MSG
   014 002  066 130     LLI 130
   014 004  106 155 014 CAL MSG                 /PRINT C/R, L/F, >
   014 007  106 066 014 CAL CDIN                /INPUT COMMAND FM KYBD
   014 012  066 340     LLI 340
   014 014  307         LAM                     /FETCH COMMAND CHAR
   014 015  036 012     LDI 012                 /SET CMND NMBR CNTR
   014 017  066 210     LLI 210                 /SET CMND TABLE PNTR
   014 021  277         CKCMD, CPM              /IS CMND CHAR FOUND IN TBL?
   014 022  150 047 014 JTZ FOUND               /YES, PROCESS COMMAND
   014 025  060         INL                     /NO,ADVANCE CMND TBL PNTR
   014 026  060         INL
   014 027  060         INL
   014 030  031         DCD                     /IS LAST CMND CHECKED?
   014 031  110 021 014 JFZ CKCMD               /NO, CHECK NEXT
```

```
014 034   106 151 014   ERR, CAL HDLN      /YES, PRINT C/R, L/F
014 037   006 311       LAI 311            /ILLEGAL ENTRY CODE
014 041   106 300 017   CAL PRINT          /PRINT ERROR MSG
014 044   104 000 014   JMP INCMD          /INP NEXT COMMAND
014 047                 /
014 047   060           FOUND, INL         /ADV CMND TBL PNTR
014 050   337           LDM                /FETCH CMND LO ADDR
014 051   060           INL
014 052   327           LCM                /FETCH CMND PG ADDR
014 053   066 156       LLI 156            /SET PNTR TO JMP INSTR.
014 055   056 000       LHI 000
014 057   373           LMD                /LOAD LO ADDR OF CMND
014 060   060           INL
014 061   372           LMC                /LOAD PG ADDR OF CMND
014 062   364           LLE
014 063   104 155 000   JMP 155 000        /JUMP TO CMND ROUTINE
014 066                 /
014 066   066 340       CDIN, LLI 340      /SET PNTR TO START OF INP BFR
014 070   076 240       SP1, LMI 240       /FILL INP BFR WITH SPACES
014 072   060           INL                /INCR INP BFR PNTR
014 073   110 070 014   JFZ SP1            /DONE? NO, STORE MORE SPACES
014 076   066 340       LLI 340            /SET INP BFR PNTR
014 100   106 200 017   IN2, CAL RCV       /INP CHAR FM INP DEVICE
014 103   074 204       CPI 204            /CHAR = CNT'L D?
014 105   150 000 014   JTZ INCMD          /YES, RET TO COMMAND MODE
014 110   074 215       CPI 215            /CHAR = CAR RET?
014 112   053           RTZ                /YES, RET TO CALLING PGM
014 113   074 214       CPI 214            /CHAR = CNT'L L?
014 115   053           RTZ                /YES, RET TO CALLING PGM
014 116   074 377       CPI 377            /CHAR = RUBOUT?
014 120   150 135 014   JTZ BDCR           /YES, DELETE CHAR FM INP BFR
014 123   060           INL                /IS INP BFR FULL?
014 124   061           DCL
014 125   150 100 014   JTZ IN2            /YES, DON'T STORE CHAR
014 130   370           LMA                /NO, STORE CHARACTER
014 131   060           INL                /INCR INP BFR PNTR
014 132   104 100 014   JMP IN2            /INP NEXT CHAR
014 135                 /
014 135   006 340       BDCR, LAI 340      /SET ACC TO INP BFR S.A.
014 137   276           CPL                /ANY CHARACTERS YET?
014 140   150 100 014   JTZ IN2            /NO, CONTINUE INPUT
014 143   061           DCL                /YES, BACK UP INP BFR PNTR
014 144   076 240       LMI 240            /STORE SPACE OVER LAST CHAR
014 146   104 100 014   JMP IN2            /CONTINUE INPUT
014 151                 /
014 151   066 134       HDLN, LLI 134      /SET PNTR TO C/R, L/F MSG
014 153   056 000       LHI 000            /FALL THRU TO PRINT MSG
014 155                 /
014 155   307           MSG, LAM           /FETCH CHAR TO PRINT
014 156   240           NDA                /END OF MSG CHAR?
014 157   053           RTZ                /YES, RET TO CALLING PGM
014 160   106 300 017   CAL PRINT          /NO, PRINT CHAR
014 163   106 171 014   CAL INMEM          /INCR MSG PNTR
014 166   104 155 014   JMP MSG            /CONTINUE PRINT OUT
014 171                 /
014 171   060           INMEM, INL         /INCR LO ADDR
014 172   013           RFZ                /IF NON ZERO, RET
014 173   050           INH                /ELSE, INCR PG ADDR
014 174   007           RET                /RET TO CALLING PGM
014 175                 /
```

```
014 175                         /
014 175  346                    OCTNM, LEL          /SAVE INP BFR PNTR
014 176  106 250 014            CAL OCTPR           /CONVERT 1ST OCTAL PAIR
014 201  066 166                LLI 166             /SET PNTR TO LO ADDR STRAGE
014 203  371                    LMB                 /SAVE LO HALF OF LO ADDR
014 204  060                    INL
014 205  372                    LMC                 /SAVE PG HALF OF LO ADDR
014 206  364                    LLE                 /RESTORE INP BFR PNTR
014 207  307                    LAM                 /FETCH NXT CHAR
014 210  074 254                CPI 254             /CHAR = COMMA?
014 212  110 222 014            JFZ SGL             /NO, ONLY ONE ENTRY
014 215  060                    INL                 /YES, INCR INP BFR PNTR
014 216  346                    LEL                 /SAVE INP VFR PNTR
014 217  106 250 014            CAL OCTPR           /CONVERT 2ND OCTAL PAIR
014 222  066 170                SGL, LLI 170        /SET PNTR TO HI ADDR STRAGE
014 224  371                    LMB                 /SAVE LO HALF OF HI ADDR
014 225  060                    INL
014 226  372                    LMC                 /SAVE PG HALF OF HI ADDR
014 227  302                    LAC
014 230  066 167                LLI 167             /IS HI ADDR < LO ADDR?
014 232  277                    CPM
014 233  140 034 014            JTC ERR             /YES, PRINT ERROR
014 236  013                    RFZ                 /IF PG HALF NOT =, RET
014 237  060                    INL                 /ELSE, CHECK LO HALF
014 240  307                    LAM
014 241  066 166                LLI 166             /IS HI ADDR < LO ADDR?
014 243  277                    CPM
014 244  140 034 014            JTC ERR             /YES, PRINT ERROR MSG
014 247  007                    RET                 /NO, RET TO CALLING PGM
014 250                         /
014 250  106 255 014            OCTPR, CAL DCDNM    /DECODE 1ST OCTAL NUMBER
014 253  321                    LCB                 /SAVE OCTAL NUMBER
014 254  040                    INE                 /INCR INP BFR PNTR
014 255                         /                    FALL THRU TO DECODE 2ND NMBR
014 255                         /
014 255  066 150                DCDNM, LLI 150      /SET PNTR TO DIGIT STRAGE TBL
014 257  375                    LMH                 /CLEAR TBL BY STORING 000.
014 260  060                    INL
014 261  375                    LMH
014 262  060                    INL
014 263  375                    LMH
014 264  364                    LLE                 /RESET INP BFR PNTR
014 265  106 332 014            LOOP, CAL FNUM      /CHECK FOR VALID NUMBER
014 270  160 315 014            JTS CKLNH           /IF NOT, CHECK CHAR CNT = 0
014 273  307                    LAM                 /FETCH CHAR
014 274  336                    LDL                 /SAVE INP BFR PNTR
014 275  044 007                NDI 007             /MASK OFF 260
014 277  066 150                LLI 150             /STORE OCTAL NUMBER IN
014 301  317                    LBM                 /TABLE AT LOC 150 PG 00
014 302  370                    LMA                 /AND SHIFT OTHER NUMBERS
014 303  060                    INL                 /UP THRU THE TABLE
014 304  307                    LAM
014 305  371                    LMB
014 306  060                    INL
014 307  370                    LMA
014 310  363                    LLD                 /RESTORE AND INCR INP BFR PNTR
014 311  060                    INL
014 312  104 265 014            JMP LOOP            /FETCH NXT NUMBER
014 315                         /
```

```
014 315  306          CKLNH, LAL
014 316  274          CPE                  /IS CHAR CNT = 0?
014 317  150 034 014  JTZ ERR              /YES, PRINT ERROR MSG
014 322  346          LEL                  /NO, SAVE INP BFR PNTR
014 323  106 355 014  CAL OCT              /FETCH FINAL OCTAL NUMBER
014 326  120 034 014  JFS ERR              /IF INVALID, PRINT ERR MSG
014 331  007          RET                  /ELSE, RET TO CALLING PGM
014 332               /
014 332  307          FNUM, LAM            /IS CHAR A VALID NUMBER?
014 333  074 260      CPI 260
014 335  063          RTS                  /NO, RET WITH S FLAG SET
014 336  024 270      SUI 270              /CHECK UPPER LIMIT BY
014 340  004 200      ADI 200              /SETTING S FLAG TO PROPER
014 342  007          RET                  /STATE AND RETURN
014 343               /
014 343  004 001      INCR, ADI 001        /INCR CONTENTS OF MEM LOC
014 345  370          LMA                  /RESTORE MEM CONTENTS
014 346  003          RFC                  /IF NO CARRY, RET
014 347  060          INL                  /ELSE, FETCH NXT LOC
014 350  307          LAM
014 351  004 001      ADI 001              /INCR MEM CONTENTS
014 353  370          LMA                  /RESTORE MEM CONTENTS
014 354  007          RET                  /RET TO CALLING PGM
014 355               /
014 355  066 152      OCT, LLI 152         /SET PNTR TO 3RD DIGIT
014 357  307          LAM
014 360  074 004      CPI 004              /IS 3RD DIGIT > 3?
014 362  023          RFS                  /YES, RET WITH S FLAG RESET
014 363  044 003      NDI 003              /CLEAR CARRY
014 365  012          RRC                  /POSITION DIGIT
014 366  012          RRC
014 367  310          LBA                  /SAVE IN REG B
014 370  061          DCL                  /DECR PNTR
014 371  307          LAM                  /FETCH NEXT DIGIT
014 372  002          RLC                  /POSITION DIGIT
014 373  002          RLC
014 374  002          RLC
014 375  201          ADB                  /ADD TO REG B
014 376  061          DCL                  /DECR PNTR
014 377  207          ADM
015 000  310          LBA                  /SAVE FINAL NUMBER
015 001  006 200      LAI 200              /SET S FLAG TO INDICATE
015 003  240          NDA                  /THAT THE NUMBER IS VALID
015 004  007          RET                  /RET TO CALLING PGM
015 005               /
015 005  325          SWITCH, LCH          /SWITCH THE PNTR IN
015 006  353          LHD                  /REG'S H AND L WITH
015 007  332          LDC                  /THE PNTR IN REG'S D AND E
015 010  326          LCL
015 011  364          LLE
015 012  342          LEC
015 013  007          RET                  /RET TO CALLING PGM
015 014               /
015 014  360          OCTOUT, LLA          /SAVE OCTAL NUMBER TO PRINT
015 015  002          RLC                  /POSITION HUNDRED'S DIGIT
015 016  002          RLC
015 017  044 003      NDI 003              /MASK OFF OTHER BITS
015 021  064 260      ORI 260              /FORM ASCII CODE
015 023  106 300 017  CAL PRINT            /PRINT DIGIT
```

```
015 026   306           LAL                    /FETCH OCTAL NUMBER
015 027   012           RRC                    /POSITION TEN'S DIGIT
015 030   012           RRC
015 031   012           RRC
015 032   044 007       NDI 007                /MASK OFF OTHER DIGITS
015 034   064 260       ORI 260                /FORM ASCII CODE
015 036   106 300 017   CAL PRINT              /PRINT DIGIT
015 041   306           LAL                    /FETCH OCTAL NUMBER
015 042   044 007       NDI 007                /MASK OFF OTHER DIGITS
015 044   064 260       ORI 260                /FORM ASCII CODE
015 046   104 300 017   JMP PRINT              /PRINT DIGIT AND RET
015 051                 /
015 051   006 272       COLON, LAI 272         /SET ASCII CODE FOR :
015 053   104 300 017   JMP PRINT              /PRINT COLON AND RET
015 056                 /
015 056   066 167       PRT166, LLI 167        /SET PNTR TO PG ADDR
015 060   056 000       LHI 000                /OF LO ADDR STORED
015 062   307           LAM                    /FETCH PG ADDR
015 063   044 077       NDI 077
015 065   106 014 015   CAL OCTOUT             /PRINT PAGE ADDR
015 070   106 101 015   CAL SPAC               /PRINT A SPACE
015 073   066 166       LLI 166                /SET PNTR TO LO ADDR
015 075   307           LAM                    /FETCH LO ADDR
015 076   106 014 015   CAL OCTOUT             /PRINT LO ADDR
015 101                 /                      /FALL THRU TO PRINT SPACE
015 101                 /
015 101   006 240       SPAC, LAI 240          /SET ASCII CODE FOR SPACE
015 103   104 300 017   JMP PRINT              /PRINT SPACE AND RET
015 106                 /
015 106   056 000       SETUP, LHI 000
015 110   066 166       LLI 166                /SET PNTR TO 00 166
015 112   327           LCM                    /FETCH LO ADDR
015 113   060           INL
015 114   357           LHM                    /FETCH PG ADDR
015 115   362           LLC                    /SET PNTR TO MEM LOC
015 116   007           RET                    /RET TO CALLING PGM
015 117                 /
015 117   056 000       CKEND, LHI 000
015 121   066 171       LLI 171                /SET PNTR TO HI ADDR
015 123   307           LAM                    /FETCH 2ND HALF
015 124   066 167       LLI 167                /SET PNTR TO 2ND HALF LO ADDR
015 126   277           CPM                    /2ND HALFS EQUAL?
015 127   110 142 015   JFZ CONT               /NO, CONTINUE PROCESS
015 132   060           INL
015 133   307           LAM                    /FETCH 1ST HALF HI ADDR
015 134   066 166       LLI 166                /SET PNTR TO 1ST HALF LO ADDR
015 136   277           CPM                    /IS 1ST HALFS EQUAL?
015 137   150 000 014   JTZ INCMD              /YES, RET TO CMND MODE
015 142   066 166       CONT, LLI 166          /NO, SET PNTR TO LO ADDR
015 144   307           LAM                    /INCR LO ADDR
015 145   104 343 014   JMP INCR
015 150                 /
015 150   066 342       MODIFY, LLI 342        /SET INP BFR PNTR
015 152   106 175 014   CAL OCTNM              /FETCH ADDR TO MODIFY
015 155   106 101 015   CAL SPAC               /PRINT SPACE
015 160   106 266 015   MOD1, CAL MEMPRT       /PRINT CONTENTS OF MEM LOC
015 163   106 051 015   CAL COLON              /PRINT COLON
015 166   106 241 015   CAL INSPCL             /INP MODIFICATION
015 171   006 340       LAI 340                /WAS MOD ENTERED?
015 173   276           CPL
```

```
015 174   150 215 015   JTZ NXLOC          /NO, SET UP NXT LOC
015 177   340           LEA                /YES, SAVE INP PNTR
015 200   106 255 014   CAL DCDNM          /CONVERT TO OCTAL NUMBER
015 203   301           LAB                /SAVE OCTAL NUMBER
015 204   066 166       LLI 166            /SET PNTR TO MEM ADDR STRAGE
015 206   347           LEM                /FETCH MEM PNTR
015 207   060           INL
015 210   337           LDM
015 211   106 005 015   CAL SWITCH         /SET PNTR TO MEM LOC
015 214   370           LMA                /LOAD MEM WITH NEW VALUE
015 215   056 000       NXLOC, LHI 000     /SET PNTR TO PG 00
015 217   066 166       LLI 166            /SET PNTR TO MEM ADDR STRAGE
015 221   307           LAM                /FETCH LO HALF
015 222   106 343 014   CAL INCR           /INCR MEM ADDR
015 225   106 233 015   CAL MCONT          /PRINT NXT ADDR TO MODIFY
015 230   104 160 015   JMP MOD1
015 233               /
015 233   106 151 014   MCONT, CAL HDLN     /PRINT C/R, L/F
015 236   104 056 015   JMP PRT166         /PRINT ADDR TO MODIFY AND RET
015 241               /
015 241   066 340       INSPCL, LLI 340    /SET PNTR TO S.A. OF INP BFR
015 243   106 ~~000~~ 014   LPIN, CAL RCV  /INP CHAR
015 246   370           LMA                /STORE CHAR IN INP BFR
015 247   074 240       CPI 240            /CHAR = SPACE?
015 251   053           RTZ                /YES, RET TO CALLING PGM
015 252   074 215       CPI 215            /NO, CHAR = C/R?
015 254   150 000 014   JTZ INCMD          /YES, RET TO COMMAND MODE
015 257   060           INL                /NO, INCR INP BFR PNTR
015 260   150 034 014   JTZ ERR            /INP BFR FULL? YES, ERROR
015 263   104 243 015   JMP LPIN           /NO, INP NXT CHAR
015 266               /
015 266   106 106 015   MEMPRT, CAL SETUP  /SET PNTR TO MEM LOC
015 271   307           LAM                /FETCH CURRENT MEM CONTENTS
015 272   104 014 015   JMP OCTOUT         /PRINT CONTENTS AND RET
015 275               /
015 275   066 342       MDUMP, LLI 342     /SET PNTR TO INP BFR
015 277   106 175 014   CAL OCTNM          /FETCH MEM DUMP LIMITS
015 302   106 151 014   CAL HDLN           /PRINT C/R, L/F
015 305   106 233 015   MDMP1, CAL MCONT   /PRINT ADDR OF 1ST LOC
015 310   106 101 015   CAL SPAC           /PRINT SPACE
015 313   066 164       MDMP2, LLI 164     /SET PNTR TO TEMP STRAGE
015 315   076 020       LMI 020            /SAVE LOC PER LINE CNTR
015 317   106 266 015   OUTAGN, CAL MEMPRT /PRINT MEM CONTENTS
015 322   106 117 015   CAL CKEND          /CHECK FOR LAST LOC PRTD
015 325   106 101 015   CAL SPAC           /PRINT SPACE
015 330   066 164       LLI 164            /SET PNTR TO L/L CNTR
015 332   317           LBM                /FETCH CNTR
015 333   011           DCB                /DECR CNTR
015 334   371           LMB                /SAVE CNTR. CNTR = 0?
015 335   150 305 015   JTZ MDMP1          /YES, START NEW LINE
015 340   104 317 015   JMP OUTAGN         /NO, PRINT MORE CONTENTS
015 343               /
015 343   066 342       WRITE, LLI 342     /SET PNTR TO INP BFR
015 345   106 175 014   CAL OCTNM          /FETCH START AND END ADDR
015 350   066 166       LLI 166            /SET REG'S H AND L WITH
015 352   327           LCM                /THE START ADDR AND
015 353   060           INL                /REG'S D AND E WITH
015 354   317           LBM                /THE END ADDR OF THE
015 355   060           INL                /BLOCK OF MEM TO BE
```

```
015 356   347          LEM                 /WRITTEN TO THE BULK
015 357   060          INL                 /STORAGE DEVICE.
015 360   337          LDM
015 361   351          LHB
015 362   362          LLC
015 363   106 340 017  CAL PUNCH           /GO TO USER BULK WRITE RTN
015 366   104 000 014  JMP INCMD           /RET TO COMMAND MODE
015 371                /
015 371   106 240 017  RDBULK, CAL READ     /GO TO USER BULK READ RTN
015 374   104 000 014  JMP INCMD           /RET TO COMMAND MODE
015 377                /
015 377   106 050 016  BREAK, CAL ANLYZ     /SET UP ADDRESS OF BP
016 002   364          LLE
016 003   353          LHD
016 004   150 027 016  JTZ B1              /DETERMINE IF B1 OR B2
016 007   074 262      CPI 262
016 011   110 034 014  JFZ ERR             /ERROR IF NEITHER
016 014                /
016 014   106 075 016  B2, CAL SETBK        /SET UP BP RST COMMAND
016 017   076 176      LMI 176             /**** STORE BP2 FND LA
016 021   060          INL
016 022   076 016      LMI 016             /**** STORE BP2 FND PG
016 024   104 037 016  JMP FINBK           /TO REST OF BP SET UP RTN
016 027                /
016 027   106 075 016  B1, CAL SETBK        /SET UP BP RST COMMAND
016 032   076 112      LMI 112             /**** STORE BP1 FND LA
016 034   060          INL
016 035   076 016      LMI 016             /**** STORE BP1 FND PG
016 037   060          FINBK, INL
016 040   374          LME                 /STORE BP ORIG LOW ADDR
016 041   060          INL
016 042   373          LMD                 /STORE BP ORIG PG ADDR
016 043   060          INL
016 044   370          LMA                 /STORE ORIG BP INSTRUCTION
016 045   104 000 014  JMP INCMD
016 050                /
016 050   066 343      ANLYZ, LLI 343      /SET PNTR TO BUFF SA
016 052   106 175 014  CAL OCTNM           /FETCH ADDR INTO 166, 167
016 055   066 341      LLI 341             /RESTORE BUFF SA
016 057   307          LAM                 /GET BP 1 OR 2 COMMAND
016 060   066 166      LLI 166             /GET BP LOW ADDR
016 062   347          LEM                 /INTO "E"
016 063   060          INL
016 064   337          LDM                 /AND BP PAGE
016 065   066 156      LLI 156             /PNTR TO JUMP COMMAND
016 067   374          LME
016 070   060          INL                 /SET UP JUMP ADDRESS
016 071   373          LMD
016 072   074 261      CPI 261             /DETERMINE IF CMND 1 OR 2
016 074   007          RET
016 075                /
016 075   307          SETBK, LAM          /SAVE ORIG CONTENTS OF BP
016 076   076 075      LMI 075             /INSERT BP RESTART INSTR
016 100   346          LEL                 /CHANGE POINTERS
016 101   335          LDH
016 102   056 000      LHI 000             /SET PAGE 00
016 104   066 070      LLI 070             /SET PNTR TO RST 1 LOC
016 106   076 104      LMI 104             /STORE JUMP INSTRUCTION
016 110   060          INL
016 111   007          RET
```

```
016 112                         /
016 112  056 000                BRK1, LHI 000          /SET PAGE 00
016 114  066 200                LLI 200                /CPU REGISTER STORAGE LOCS
016 116  370                    LMA                    /SAVE ACCUMULATOR
016 117  066 201                LLI 201
016 121  371                    LMB                    /AND CPU REGS B & C
016 122  066 202                LLI 202
016 124  372                    BRKCOM, LMC            /WITHOUT DISTURBING FLAGS
016 125  305                    FLAGT, LAH             /SET UP TEMP REGS
016 126  310                    LBA
016 127  320                    LCA
016 130  100 135 016            JFC NOC                /TEST FOR CARRY FLAG
016 133  006 001                LAI 001                /SET 1 IN "A" IF CARRY TRUE
016 135  110 142 016            NOC, JFZ NOZ           /TEST FOR ZERO FLAG
016 140  016 010                LBI 010                /SET 1 IN "B" IF ZERO TRUE
016 142  130 147 016            NOZ, JFP NOP           /TEST FOR PARITY FLAG
016 145  026 100                LCI 100                /SET 1 IN "C" IF PARITY "T"
016 147  120 154 016            NOP, JFS NOS           /TEST FOR SIGN FLAG
016 152  004 200                ADI 200                /SET MSB IF SIGN TRUE
016 154  201                    NOS, ADB
016 155  202                    ADC                    /FORM FLAG STATUS BYTE
016 156  066 207                LLI 207
016 160  370                    LMA                    /STORE FLAG STATUS
016 161  066 073                LLI 073                /PNTR TO ORIG BP LOW ADDR
016 163  347                    LEM                    /GET ORIG LOC OF BP
016 164  060                    INL
016 165  337                    LDM                    /AND ORIG PG OF BP
016 166  060                    INL
016 167  307                    LAM                    /AND ORIG BP INSTRUCTION
016 170  364                    LLE                    /SET UP ORIGINAL
016 171  353                    LHD                    /BREAK POINT POINTERS
016 172  370                    LMA                    /RESTORE ORIG BKPNT INSTR
016 173  104 000 014            JMP INCMD              /BACK TO MONITOR
016 176                         /
016 176  315                    BRK2, LBH              /SAVE ORIG VALUE OF H & L
016 177  326                    LCL
016 200  056 000                LHI 000                /SET PNTR TO PAGE 00
016 202  066 203                LLI 203                /CPU REGISTER STORAGE LOCS
016 204  373                    LMD                    /SAVE REGS D AND E
016 205  066 204                LLI 204                /AS WELL AS ORIG H AND L
016 207  374                    LME
016 210  066 205                LLI 205                /WITHOUT DISTURBING FLAGS
016 212  371                    LMB
016 213  066 206                LLI 206
016 215  104 124 016            JMP BRKCOM             /TO REST OF BREAKPT RTN
016 220                         /
016 220  106 050 016            GOTO, CAL ANLYZ        /SET UP ADDR OF GOTO
016 223  150 251 016            JTZ G01                /TO SET UP CPU REGS A,B,C
016 226  074 262                CPI 262
016 230  110 034 014            JFZ ERR                /ERROR IF NOT G1 OR G2
016 233                         /
016 233  066 203                G02, LLI 203           /SET UP CPU REGS D,E,H & L
016 235  337                    LDM
016 236  060                    INL
016 237  347                    LEM
016 240  060                    GOCOM, INL
016 241  317                    LBM
016 242  060                    INL
016 243  327                    LCM
```

```
016 244  362          LLC
016 245  351          LHB
016 246  104 155 000  JMP 155 000
016 251               /
016 251  066 200      GO1, LLI 200        /SET UP CPU REGS A,B,C
016 253  307          LAM
016 254  104 240 0167 JMP GOCOM
016 257               /
016 257  066 341      XREG, LLI 341       /SET INP BFR PNTR
016 261  307          LAM                 /FETCH REG LETTER
016 262  074 301      RGAGN, CPI 301      /IS REG VALID?
016 264  140 034 0147 JTC ERR             /NO, PRINT ERROR
016 267  074 306      CPI 306             /YES, IS REG A THRU E?
016 271  100 340 0167 JFC FHL             /NO, TRY H, L OR F
016 274  024 101      SUI 101             /SET UP REG TBL PNTR
016 276  066 164      XCOM, LLI 164       /SAVE TBL PNTR IN TEMP STRAGE
016 300  370          LMA
016 301  360          LLA                 /SET PNTR TO REG TBL LOC
016 302  106 101 0157 CAL SPAC            /PRINT SPACE
016 305  307          LAM                 /FETCH CURRENT REG VALUE
016 306  106 014 0157 CAL OCTOUT          /PRINT CURRENT REG VALUE
016 311  106 051 0157 CAL COLON           /PRINT COLON
016 314  106 241 0157 CAL INSPCL          /INP MODIFICATION
016 317  046 340      LEI 340             /SET INP BFR PNTR
016 321  306          LAL
016 322  274          CPE                 /WAS MOD ENTERED?
016 323  150 000 0147 JTZ INCMD           /NO, RET TO COMMAND MODE
016 326  106 255 0147 CAL DCDNM           /YES, DECODE OCTAL NUMBER
016 331  066 164      LLI 164             /SET PNTR TO TEMP STRAGE
016 333  367          LLM                 /FETCH REG TBL PNTR
016 334  371          LMB                 /STORE NEW REG VALUE
016 335  104 000 0147 JMP INCMD           /RET TO COMMAND MODE
016 340               /
016 340  074 310      FHL, CPI 310        /IS REG = H?
016 342  110 352 0167 JFZ LORF            /NO, TRY L OR F
016 345  006 205      LAI 205             /YES, SET REG TBL PNTR
016 347  104 276 0167 JMP XCOM            /INP MOD TO REG VALUE
016 352               /
016 352  074 314      LORF, CPI 314       /IS REG = L?
016 354  110 364 0167 JFZ F               /NO, TRY F
016 357  006 206      LAI 206             /YES, SET REG TBL PNTR
016 361  104 276 0167 JMP XCOM            /INP MOD TO REG VALUE
016 364               /
016 364  074 306      F, CPI 306          /IS REG = F, FOR FLAGS?
016 366  110 034 0147 JFZ ERR             /NO, PRINT ERROR
016 371  106 101 0157 CAL SPAC            /PRINT SPACE
016 374  066 207      LLI 207             /SET REG TBL PNTR
016 376  307          LAM                 /FETCH FLAG WORD
016 377  106 014 0157 CAL OCTOUT          /PRINT FLAG WORD
017 002  104 000 0147 JMP INCMD           /RET TO COMMAND MODE
017 005               /
017 005  106 050 0177 FILL, CAL ADRDTA     /INP ADDR AND DATA FM BFR
017 010  106 106 0157 FL1, CAL SETUP       /SET UP MEM PNTR
017 013  371          LMB                 /FILL MEM LOC WITH DATA
017 014  106 117 0157 CAL CKEND           /DONE? YES, RET TO CMND MODE
017 017  104 010 0177 JMP FL1             /NO, CONTINUE WITH FILL
017 022               /
```

```
017 022                          /
017 022  106 050 01/776  SEARCH, CAL ADRDTA   /INP ADDR AND DATA FM BFR
017 025  066 165         LLI 165              /SET PNTR TO SAVE DATA
017 027  371             LMB                  /SAVE SEARCH DATA IN MEM
017 030  066 165         SH1, LLI 165         /SET PNTR TO SRCH DATA
017 032  307             LAM                  /FETCH SEARCH DATA
017 033  106 106 01/5    CAL SETUP            /FETCH CONTENTS OF MEM
017 036  277             CPM                  /DATA EQUAL SRCH DATA
017 037  152 233 01/5    CTZ MCONT            /YES, PRINT ADDR
017 042  106 117 01/5    CAL CKEND            /DONE? YES, RET TO CMND MODE
017 045  104 030 01/776  JMP SH1              /NO, CONTINUE SEARCH
017 050                          /
017 050  066 342         ADRDTA, LLI 342      /SET PNTR TO ADDR INP
017 052  106 175 01/4    CAL OCTNM            /INP START AND END ADDR
017 055  040             INE                  /INCR TO DATA POSITION
017 056  104 255 01/4    JMP DCDNM            /FETCH DATA FM INP BFR
017 061                          /
017 061  066 342         TRNSFR, LLI 342      /SET PNTR TO ADDR INP
017 063  106 175 01/4    CAL OCTNM            /FETCH 'FROM' ADDR
017 066  066 166         LLI 166              /SET PNTR TO ADDR INP
017 070  314             LBE                  /SAVE INP BFR PNT
017 071  046 172         LEI 172              /SAVE 'FROM' IN TEMP STRGE
017 073  335             LDH
017 074  106 154 01/776  SVSA, CAL SWAP       /MOVE ADDR TO TEMP STRGE
017 077  006 172         LAI 172              /IS XFR COMPLETE?
017 101  276             CPL
017 102  110 074 01/776  JFZ SVSA             /NO, CONTINUE MOVE
017 105  010             INB
017 106  361             LLB                  /RESTORE INP BFR PNTR
017 107  106 175 01/4    CAL OCTNM            /INP 'TO' ADDR
017 112  066 176         LLI 176              /SET PNTR TO SAVE 'TO' ADDR
017 114  371             LMB                  /SAVE LO ADDR
017 115  060             INL
017 116  372             LMC                  /SAVE PG ADDR
017 117  066 172         LLI 172              /SET PNTR TO TEMP STRGE
017 121  046 166         LEI 166              /SET TO MOVE 'FROM' BACK
017 123  335             LDH
017 124  106 154 01/776  TF1, CAL SWAP        /XFR 'FROM'
017 127  006 176         LAI 176
017 131  276             CPL                  /XFR COMPLETE?
017 132  110 124 01/776  JFZ TF1              /NO, CONTINUE
017 135  347             LEM                  /FETCH 'TO' PNTR
017 136  060             INL
017 137  337             LDM
017 140  106 106 01/5    TF2, CAL SETUP       /SET 'FROM' PNTR
017 143  106 154 01/776  CAL SWAP             /SWAP MEM CONTENTS
017 146  106 117 01/5    CAL CKEND            /DONE? YES, RET TO CMND MODE
017 151  104 140 01/776  JMP TF2              /NO, CONTINUE XFR
017 154                          /
017 154  307             SWAP, LAM            /FETCH BYTE TO XFR
017 155  106 171 01/4    CAL INMEM            /INCR 'FROM' PNTR
017 160  106 005 01/5    CAL SWITCH           /CHANGE PNTRS
017 163  370             LMA                  /STORE BYTE IN NEW LOC
017 164  106 171 01/4    CAL INMEM            /INCR 'TO' PNTR
017 167  104 005 01/5    JMP SWITCH           /CHANGE PNTRS AND RET
017 172
```

```
                                RCV,        /USER DEFINED INPUT ROUTINE
                                            /FOR OPERATOR INPUT DEVICE
                                /
017 240     (ROM)               READ,       /USER DEFINED INPUT ROUTINE
                                            /FOR BULK STORAGE DEVICE
                                /
017 300                         PRINT,      /USER DEFINED OUTPUT ROUTINE
                                            /FOR DISPLAY DEVICE
                                /
017 340                         PUNCH,      /USER DEFINED OUTPUT ROUTINE
                                            /FOR BULK STORAGE DEVICE
```

OPERATING THE MONITOR PROGRAM

AS A REVIEW OF THE MONITOR PROGRAM FUNCTIONS AND, ALSO, TO SERVE AS AN OPERATOR'S GUIDE, THE OPERATION OF EACH OF THE MONITOR COMMANDS WILL NOW BE DESCRIBED.

THE "MODIFY" COMMAND

THE "MODIFY" COMMAND IS INITIATED BY TYPING IN THE "M" COMMAND FOLLOWED BY THE ADDRESS TO BE MODIFIED, IN THE FOLLOWING FORMAT:

M HHH LLL (CTRL/L)

WHERE "HHH" IS THE PAGE ADDRESS AND "LLL" IS THE LOW ADDRESS (IN OCTAL) OF THE RAM MEMORY ADDRESS WHERE ONE DESIRES TO BEGIN EXAMINING AND/OR MODIFYING THE CONTENTS OF MEMORY LOCATIONS. THE OPERATOR SHOULD NOTE THAT A SPACE SHOULD BE INSERTED BETWEEN THE "M" AND THE PAGE ADDRESS AS WELL AS BETWEEN THE PAGE ADDRESS AND THE LOW ADDRESS WHEN ENTERING THE COMMAND STRING.

WHEN THE OPERATOR DEPRESSES THE "CTRL/L" COMBINATION TO EXECUTE THE "M" COMMAND, THE FOLLOWING WILL OCCUR. THE OUTPUT DEVICE WILL DISPLAY THE FOLLOWING INFORMATION:

HHH LLL XXX:

THE "XXX" IS THE CURRENT CONTENTS OF THE MEMORY LOCATION SPECIFIED. THE PROGRAM WILL THEN WAIT FOR THE OPERATOR TO SELECT EITHER A "MODIFY" OPTION, OR TAKE THE OPTION OF NOT MODIFYING THE CURRENT LOCATION BEING DISPLAYED BUT CONTINUE TO DISPLAY THE NEXT LOCATION, OR TERMINATE THE "M" SEQUENCE. TO ELECT TO MODIFY THE CONTENTS OF THE MEMORY LOCATION BEING DISPLAYED, THE OPERATOR SIMPLY TYPES IN THE DESIRED OCTAL CONTENTS IMMEDIATELY FOLLOWING THE ":" SIGN AND THEN DEPRESSES THE "SPACE" BAR. THE NUMBER INTERED WILL BECOME THE NEW VALUE FOR THE MEMORY LOCATION AND THE PROGRAM WILL PROCEED TO DISPLAY THE ADDRESS AND CONTENTS OF THE NEXT SEQUENTIAL MEMORY LOCATION.

IF THE OPERATOR DOES NOT WISH TO MODIFY THE CONTENTS OF A LOCATION, BUT DOES DESIRE TO EXAMINE THE CONTENTS OF THE NEXT MEMORY LOCATION, THEN IT IS ONLY NECESSARY TO DEPRESS THE "SPACE" BAR. THE PROGRAM WILL PROCEED TO DISPLAY THE MEMORY ADDRESS AND CONTENTS OF THE NEXT MEMORY LOCATION.

IF THE OPERATOR DESIRES TO TERMINATE THE "MODIFY" PROCESS, THEN THE "CARRIAGE RETURN" IS ENTERED AND THE PROGRAM WILL RETURN TO THE MONITOR COMMAND MODE AND DISPLAY THE ">" MONITOR "READY" CHARACTER.

IT IS IMPORTANT TO NOTE THAT WHEN ELECTING TO MODIFY A MEMORY LOCATION, THE "SPACE" CHARACTER MUST BE ENTERED AFTER ENTERING THE OCTAL NUMBER THAT IS TO BE THE NEW VALUE IN THE MEMORY LOCATION! THIS WILL CAUSE THE NEW VALUE TO BE PLACED IN THE MEMORY LOCATION AND AUTOMATICALLY CAUSE THE NEXT LOCATION IN MEMORY TO BE DISPLAYED. HITTING THE "C/R" IMMEDIATELY AFTER INTERING A NEW VALUE FOR A MEMORY LOCATION WILL CAUSE THE PROGRAM TO RETURN TO THE MONITOR AND WILL N O T RESULT IN THE VALUE BEING PLACED IN MEMORY! THIS FORMAT ALLOWS THE OPERATOR TO ELECT NOT TO CHANGE A MEMORY LOCATION EVEN AFTER HAVING TYPED IN A VALUE. IF, HOWEVER, THE RULE IS NOT REMEMBERED, THE OPERATOR MEY INADVERTENTLY FAIL TO INSERT THE DESIRED CHANGES.

CORRECTING ERRORS WHEN IN THE MONITOR COMMAND MODE

IF THE OPERATOR MAKES A TYPING MISTAKE WHILE ENTERING A COMMAND SEQUENCE TO THE MONITOR, THE CURRENT COMMAND CAN BE ERASED BY ENTERING THE CHARACTER "CONTROL/D." THIS WILL CAUSE THE PROGRAM TO GO BACK TO THE INITIAL "READY" CONDITION (">" DISPLAYED) TO AWAIT A NEW ENTRY. IF ONLY ONE OR TWO CHARACTERS ARE ENTERED IN ERROR, THE "RUBOUT" CHARACTER MAY BE ENTERED TO DELETE ONE CHARACTER TO THE LEFT FOR EACH RUBOUT ENTERED.

SHOULD THE OPERATOR INADVERTENTLY ENTER AN INVALID COMMAND OR COMMAND SEQUENCE, THE PROGRAM WILL CAUSE THE LETTER "I" (ILLIGAL COMMAND) TO BE PRINTED.

THE MEMORY "DUMP" COMMAND

THE MONITOR MEMORY "DUMP" COMMAND IS INITIATED BY TYPING IN THE "D" COMMAND IN THE FOLLOWING FORMAT:

D HHH LLL,MMM NNN (CTRL/L)

WHERE "HHH" AND "LLL" SIGNIFIES THE STARTING ADDRESS (OCTAL) AND "MMM" AND "NNN" INDICATE THE ENDING ADDRESS OF THE BLOCK OF MEMORY THAT ONE DESIRES TO HAVE DISPLAYED. WHEN THE "CTRL/L" (OR, "C/R" MAY BE USED) IS ENTERED, THE PROGRAM WILL PROCEED TO DISPLAY THE CONTENTS OF THE MEMORY LOCATIONS SPECIFIED. THE OUTPUT FORMAT WILL BE THE FOLLOWING:

```
HHH LLL  XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX
HHH+020  XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX
HHH+040  XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX
```

EACH LINE PRINTED STARTS WITH THE ADDRESS OF THE FIRST LOCATION DISPLAYED FOLLOWED BY THE CONTENTS OF THE NEXT 20 (OCTAL) LOCATIONS IN MEMORY. THE PROCESS CONTINUES UNTIL THE LAST LOCATION SPECIFIED IN THE COMMAND HAS BEEN PRINTED.

THE "WRITE" COMMAND

THE "WRITE" COMMAND IS INITIATED BY THE OPERATOR ENTERING THE "W" COMMAND IN THE FOLLOWING FORMAT:

W HHH LLL, MMM NNN (CTRL/L)

WHERE "HHH" AND "LLL" INDICATE THE START ADDRESS AND "MMM" AND "NNN" INDICATE THE ENDING ADDRESS OF THE BLOCK TO BE WRITTEN TO THE BULK STORAGE DEVICE. NATURALLY, THE OPERATOR MUST MAKE WHATEVER PREPARATIONS ARE NECESSARY FOR THE BULK STORAGE DEVICE TO RECEIVE THE DATA BEFORE THE COMMAND IS ISSUED (BY ENTERING THE "CTRL/L" (OR "C/R")). AT THE CONCLUSION OF THE DATA TRANSFER, IT IS ASSUMED THAT THE BULK STORAGE OUTPUT ROUTINE WILL RETURN TO THE MONITOR COMMAND MODE.

THE "READ" COMMAND

THE "READ" COMMAND IS INITIATED BY THE OPERATOR ENTERING THE "R" COMMAND IN THE FOLLOWING FORMAT:

R (CTRL/L)

THE ISSUANCE OF THIS COMMAND CALLS THE BULK STORAGE INPUT ROUTINE TO BEGIN READING IN THE DATA FROM THE BULK STORAGE DEVICE. ADDRESSING INFORMATION IS ASSUMED TO BE EITHER SET UP BY THE BULK STORAGE INPUT ROUTINE OR RECEIVED FROM THE DATA AS IT IS READ IN. THE OPERATOR MUST SET UP THE BULK STORAGE DEVICE PRIOR TO ENTERING THIS COMMAND OR AS IS REQUIRED BY THE BULK INPUT ROUTINE.

THE "BREAKPOINT" COMMAND

THE MONITOR "BREAKPOINT" COMMANDS ARE INITIATED BY TYPING IN ONE OF THE FOLLOWING COMMANDS:

B1 HHH LLL (CTRL/L)
OR
B2 HHH LLL (CTRL/L)

WHERE "HHH LLL" DESIGNATES THE MEMORY ADDRESS AT WHICH THE BREAKPOINT IS TO BE INSERTED.

N O T I C E

IN CASES WHERE A BREAKPOINT IS TO BE INSERTED IN A MULTI-BYTE INSTRUCTION, SUCH AS "IMMEDIATE," "JUMP" OR "CALL" INSTRUCTIONS, THE ADDRESS INDICATED MUST BE THE ADDRESS OF THE FIRST BYTE IN THE INSTRUCTION!

THE TWO TYPES OF BREAKPOINT INSTRUCTIONS, "B1" AND "B2" REFER TO THE OPTION OF HAVING THE STATUS OF THE "A," "B" AND "C" CPU REGISTERS (B1) OR THE "D," "E," "H" AND "L" CPU REGISTERS (B2) SAVED IN THE VIRTUAL CPU REGISTER LOCATIONS, ALONG WITH THE FLAG STATUS, AT THE TIME THE BREAKPOINT IS ENCOUNTERED. THUS, THE OPERATOR MAY INSERT A BREAKPOINT IN A

PROGRAM BEING TESTED TO ASCERTAIN WHETHER PROGRAM OPERATION IS ACTUALLY REACHING A CERTAIN POINT, OR TO VALIDATE THE STATUS OF THE SELECTED CPU REGISTERS AT GIVEN POINTS WITHIN A PROGRAM UNDER DEVELOPMENT. WHEN THE PROGRAM BEING TESTED REACHES THE ADDRESS AT WHICH A BREAKPOINT HAS BEEN INSERTED, CONTROL WILL REVERT TO THE MONITOR A N D THE ORIGINAL INSTRUCTION IN THE PROGRAM WILL BE RESTORED AT THE BREAKPOINT ADDRESS!

C A U T I O N

WHEN UTILIZING THE BREAKPOINT FACILITY THERE ARE SEVERAL CONSIDERATIONS THAT THE OPERATOR MUST KEEP IN MIND:

1. THE PROGRAM BEING TESTED MAY NEVER REACH THE SELECTED BREAKPOINT ADDRESS IN WHICH CASE THE OPERATOR MAY HAVE TO MANUALLY STOP THE PROGRAM AND RESTART THE MONITOR PROGRAM. IF THIS OCCURS, THE OPERATOR SHOULD USE THE "MODIFY" FUNCTION TO REMOVE THE "BREAKPOINT" INSTRUCTION FROM THE LOCATION THAT IT WAS INSERTED (WHICH WILL APPEAR AS AN "075" CODE) AND RESTORE THE ORIGINAL INSTRUCTION CODE TO THE PROGRAM UNDER TEST. THE OPERATOR WOULD MOST LIKELY THEN CONTINUE TO "DEBUG" THE PROGRAM BY SELECTING A BREAKPOINT AT SOME OTHER LOCATION.

2. ONLY ONE BREAKPOINT SHOULD BE ESTABLISHED AT ONE TIME. ATTEMPTING TO ESTABLISH MORE THAN ONE BREAKPOINT WILL RESULT IN THE FIRST BREAKPOINT ENCOUNTERED BEING RESTORED WITH THE INSTRUCTION CODE CONTAINED IN THE ORIGINAL PROGRAM AT THE LAST POINT AT WHICH A BREAKPOINT WAS ESTABLISHED. THIS MIGHT NOT BE APPROPRIATE.

3. A TYPE "1" BREAKPOINT SHOULD NOT BE CHANGED TO A TYPE "2" BREAKPOINT (OR VICE-VERSA) UNTIL THE BREAKPOINT HAS ACTUALLY BEEN ENCOUNTERED. ATTEMPTING TO DO SO WILL RESULT IN AN "075" CODE BEING INCORRECTLY RESTORED TO THE ORIGINAL BREAKPOINT.

IT SHOULD BE APPARENT, THAT IF ONE DESIRES TO EXAMINE ALL THE CPU REGISTERS AT A GIVEN POINT IN A PROGRAM'S OPERATION, ONE WILL NEED TO OPERATE THE PROGRAM TWICE - ONCE WITH A "B1 BREAKPOINT ESTABLISHED, AND ONCE WITH A "B2" BREAKPOINT ESTABLISHED AT THE SAME ADDRESS.

SINCE THE "VIRTUAL" CPU REGISTERS ARE ONLY UPDATED WHEN A BREAKPOINT IS REACHED (OR WHEN THE OPERATOR SPECIFICALLY SETS THEM UP) IT IS POSSIBLE TO REVIEW THE STATUS OF THE TWO GROUPS OF CPU REGISTERS AT SEVERAL DIFFERENT POINTS IN A PROGRAM. FOR INSTANCE, ONE COULD SET UP A "B1" TYPE BREAKPOINT AT LOCATION "A," HAVE THE BREAKPOINT ENCOUNTERED AND THE ASSOCIATED "A," "B" AND "C" CPU REGISTERS SAVED IN THE VIRTUAL LOCATIONS, THEN INSERT A TYPE "B2" BREAKPOINT AT LOCATION "B," HAVE IT ENCOUNTERED, AND THEN REVIEW THE STATUS OF THE CPU REGISTERS USING THE "X" TYPE COMMANDS. ONE COULD CONTINUE, SAY, TO INSERT AND ENCOUNTER MORE TYPE "B2" BREAKPOINTS WHILE STILL SAVING THE ORIGINAL "A," "B" AND "C" VALUES FOR REVIEW. (PARTICULARLY VALUABLE FOR THOSE THAT HAVE SHORT MEMORIES WHEN WORKING ON DEBUGGING A COMPLEX PROGRAM!)

THE "GO TO" COMMAND

THE "GO TO" COMMANDS ARE INITIATED BY TYPING IN ONE OF THE FOLLOWING COMMANDS:

G1 HHH LLL (CTRL/L)
OR
G2 HHH LLL (CTRL/L)

WHERE "HHH LLL" REPRESENTS THE MEMORY ADDRESS AT WHICH PROGRAM OPERATION IS TO COMMENCE WITH THE "A," "B" AND "C" REGISTERS FOR "G1" OR THE "D," "E," "H" AND "L" REGISTERS FOR "G2" INITIALIZED TO THE VALUES RESIDING IN THE VIRTUAL CPU REGISTER STORAGE LOCATIONS. IN MANY CASES, WHERE THE OPERATOR DOES NOT CARE WHAT THE STATUS OF THE CPU REGISTERS ARE WHEN PROGRAM OPERATION BEGINS, THE SELECTION OF THE "G1" OR "G2" TYPE "GO TO" COMMAND IS PURELY ARBITRARY. HOWEVER, WHEN DESIRED, THE OPERATOR MAY SET UP EITHER GROUP OF CPU REGISTERS TO CONTAIN SPECIFIC VALUES (USING THE "X" COMMAND) PRIOR TO EXECUTING THE "GO TO" COMMAND. THOSE VALUES WILL BE PLACED IN THE CPU REGISTERS WHEN THE "GO TO" COMMAND IS EXECUTED AND THE PROGRAM WILL THEN JUMP TO COMMENCE PROGRAMMED OPERATION AT THE ADDRESS SPECIFIED IN THE "GO TO" COMMAND. (NOTE THAT SINCE A BREAKPOINT IS ENCOUNTERED A F T E R A "GO TO" COMMAND HAS BEEN EXECUTED, SETTING UP THE DESIRED VALUES IN CPU REGISTERS FOR A "GO TO" COMMAND WILL NOT EFFECT THE BREAKPOINT PROCESS OF "SAVING" THE CONTENTS OF A GROUP OF CPU REGISTERS WHEN A BREAKPOINT IS ENCOUNTERED.)

THE "EXAMINE REGISTER" COMMAND

THE "EXAMINE REGISTER" COMMANDS ARE INITIATED BY TYPING IN ONE OF THE FOLLOWING COMMANDS:

XA (CTRL/L)
XB (CTRL/L)
XC (CTRL/L)
XD (CTRL/L)
XE (CTRL/L)
XH (CTRL/L)
XL (CTRL/L)
XF (CTRL/L)

WHERE THE LETTER FOLLOWING THE "X" INDICATES THE "VIRTUAL" CPU REGISTER TO BE DISPLAYED. THE "CTRL/L" MUST BE USED IN THIS COMMAND AS THE TERMINATING CHARACTER TO MAINTAIN THE DISPLAY DEVICE AT THE POSITION FOLLOWING THE "XR" COMMAND. THE CONTENTS OF THE SPECIFIED REGISTER WILL BE DISPLAYED IN THE FOLLOWING FORMAT:

XR XXX:

FOR ALL BUT THE "XF" COMMAND, THE OPERATOR THEN HAS THE CHOICE OF MODIFYING, OR NOT MODIFYING, THE CONTENTS OF THE VIRTUAL REGISTER. IF IT IS NOT DESIRED TO MODIFY THE CONTENTS AS DISPLAYED, THE OPERATOR SIMPLY DEPRESSES THE SPACE BAR AND THE PROGRAM RETURNS TO THE MONITOR COMMAND MODE.

IF IT IS DESIRED TO MODIFY THE CONTENTS OF A VIRTUAL REGISTER, THE OPERATOR TYPES IN THE DESIRED OCTAL VALUE AND DEPRESSES THE SPACE BAR.

IF THE OPERATOR SHOULD TYPE IN A NEW OCTAL VALUE AND THEN DECIDE THAT IT IS NOT DESIRABLE TO CHANGE THE ORIGINAL VALUE, THE OPERATOR MAY STRIKE THE "C/R" KEY TO RETURN TO THE COMMAND MODE, IN WHICH CASE THE ORIGINAL VALUE WILL REMAIN UNCHANGED.

THE "XF" COMMAND CAUSES THE STATUS OF THE CPU FLAGS (AS THEY WERE

WHICH THE LAST BREAKPOINT WAS ENCOUNTERED) TO BE DISPLAYED ACCORDING TO THE FOLLOWING ARRANGEMENT.

B7 B6 B5 B4 B3 B2 B1 B0

THE FOUR FLAGS CONNECTED WITH THE CPU HAVE BEEN ASSIGNED TO THE FOLLOWING POSITIONS IN THE EIGHT BIT GROUP.

B7 = SIGN FLAG
B6 = PARITY FLAG
B3 = ZERO FLAG
B0 = CARRY FLAG

THE FLAG WAS SET IF THE CORRESPONDING BIT POSITION HAS A VALUE OF "1." SINCE THE FLAG STATUS IS DISPLAYED AS AN OCTAL VALUE, THE OPERATOR MUST INTERPRET THE OCTAL CODE DISPLAYED TO DETERMINE THE SETTING OF EACH CPU FLAG. FOR INSTANCE, IF THE VALUE "300" WAS DISPLAYED IT WOULD MEAN THE SIGN AND PARITY FLAGS WERE "SET" AND THE ZERO AND CARRY FLAGS WERE IN THE CLEARED CONDITION. THE VALUE "011" WOULD INDICATE THAT THE SIGN AND PARITY FLAGS WERE IN THE ZERO STATE (FALSE) WHILE THE ZERO AND CARRY FLAGS WERE TRUE (IN THE ONE CONDITION). THE VALUE "201" WOULD BE INTERPRETED TO INDICATE THAT THE SIGN AND CARRY FLAGS WERE SET WHILE THE PARITY AND ZERO FLAGS WERE NOT.

THE "FILL" COMMAND

THE "FILL" COMMAND IS INITIATED BY TYPING IN THE "F" COMMAND IN THE FOLLOWING FORMAT:

F HHH LLL,MMM NNN,DDD (CTRL/L)

WHERE "HHH LLL" IS THE START ADDRESS AND "MMM NNN" IS THE END ADDRESS OF THE SECTION OF MEMORY THAT IS TO BE FILLED WITH THE DATA BYTE "DDD." WHEN THE CTRL/L (OR C/R) IS ENTERED, THE PROGRAM WILL PROCEED TO LOAD THE MEMORY LOCATIONS SPECIFIED WITH THE 8 BIT DATA BYTE ENTERED IN THE COMMAND. AT THE CONCLUSION, THE PROGRAM RETURNS TO THE MONITOR COMMAND MODE.

THE "SEARCH" COMMAND

THE SEARCH COMMAND IS INITIATED BY TYPING IN THE "S" COMMAND IN THE FOLLOWING FORMAT:

S HHH LLL,MMM NNN,DDD (CTRL/L)

WHERE "HHH LLL" SIGNIFIES THE START ADDRESS AND "MMM NNN" INDICATE THE ENDING ADDRESS OF THE BLOCK OF MEMORY TO BE SEARCHED FOR THE DATA PATTERN "DDD." WHEN THE OPERATOR ENTERS THE CTRL/L (OR C/R), THE PROGRAM BEGINS SEARCHING THE DESIGNATED MEMORY LOCATIONS FOR THE DATA PATTERN SPECIFIED IN THE COMMAND AND EACH TIME A MATCH IS FOUND, THE ASSOCIATED MEMORY ADDRESS IS OUTPUT TO THE DISPLAY DEVICE, PRECEEDED BY A C/R, L/F COMBINATION TO START EACH ADDRESS OUTPUT ON A NEW LINE. THE PROGRAM RETURNS TO THE COMMAND MODE WHEN THE ENTIRE BLOCK HAS BEEN SEARCHED.

THE "TRANSFER" COMMAND

THE "TRANSFER" COMMAND IS INITIATED BY TYPING IN THE "T" COMMAND IN THE FOLLOWING FORMAT:

T HHH LLL,MMM NNN,YYY ZZZ (CTRL/L)

WHERE "HHH LLL" SPECIFIES THE START ADDRESS AND "MMM NNN" THE END ADDRESS OF THE BLOCK OF MEMORY THAT IS TO BE TRANSFERED TO THE SECTION OF MEMORY WHICH STARTS AT LOCATION "YYY ZZZ." WHEN THE CTRL/L (OR C/R) IS ENTERED, THE PROGRAM BEGINS THE TRANSFER BY FETCHING THE CONTENTS OF THE MEMORY LOCATION "HHH LLL" AND STORES THAT VALUE IN THE LOCATION "YYY ZZZ." THE CONTENTS OF "HHH LLL+1" IS THEN TRANSFERRED TO "YYY ZZZ+1" AND SO ON, UNTIL THE CONTENTS OF THE LAST LOCATION "MMM NNN" HAS BEEN TRANSFERRED. THE PROGRAM THEN RETURNS TO THE COMMAND MODE.

PUTTING THE MONITOR PROGRAM ON "PROMS"

ONCE THE MONITOR PROGRAM PRESENTED IN THIS MANUAL HAS BEEN "CUSTOMIZED" TO THE READER'S PARTICULAR SYSTEM, BY MODIFYING OR EXPANDING THE PROGRAM TO MEET THE REQUIREMENTS OF ONE'S SYSTEM, IT CAN BE EASILY ADAPTED FOR PERMANENT STORAGE ON "PROMS" TO ALLOW THE COMPUTER TO BE "ON-LINE" ONCE THE POWER IS TURNED ON BY SIMPLY JUMPING TO THE START ADDRESS OF THE MONITOR PROGRAM. THIS IS MADE POSSIBLE BY HAVING ALL TEMPORARY DATA STORED IN THE FIRST 256 LOCATIONS OF RAM MEMORY. IF ONE IS TO PUT THE MONITOR PROGRAM ON "PROMS" THERE ARE SEVERAL FACTS THAT MUST BE BROUGHT OUT. FIRST, THE PROGRAM SHOULD BE LOCATED IN THE UPPER-MOST SECTION OF MEMORY THAT THE SYSTEM IS CAPABLE OF ADDRESSING. NEXT, THE COMMAND LOOK UP TABLE AND CANNED MESSAGES SHOULD BE MOVED TO BE INCLUDED IN THE PROM SECTION OF THE PROGRAM. THIS REQUIRES THAT THE POINTERS TO THESE TWO AREAS, IN THE "COMMAND INPUT" ROUTINE AND THE "HDLN" SUBROUTINE, BE CHANGED TO INDICATE THE NEW START ADDRESSES. ALSO, IN THE COMMAND INPUT ROUTINE, WHEN THE START ADDRESS OF THE COMMAND TO BE EXECUTED IS STORED AT LOCATIONS 156 AND 157, THE PROGRAM SHOULD ALSO STORE THE "104" PORTION OF THE JUMP INSTRUCTION AT LOCATION 155, TO SET UP THE JUMP INSTRUCTION PROPERLY WHEN THE FIRST COMMAND IS ENTERED. AND FINALLY, BEFORE PUTTING THE PROGRAM ON "PROMS," MAKE SURE THAT EACH FUNCTION IS CHECKED OUT T H O R O U G H L Y, THEREBY, DECREASING THE LIKELYHOOD THAT THE PROMS WILL HAVE TO BE RE-PROGRAMMED TO CORRECT SOMETHING THAT WAS OVERLOOKED ON THE INITIAL PROGRAMMING.

HAVING THIS TYPE OF PROGRAM ON PROM HAS SEVERAL IMPORTANT ADVANAGES. AS MENTIONED ABOVE, IT ALLOWS IMMEDIATE "ON-LINE" CAPABILITY. IT ALSO PREVENTS A PROGRAM BEING DEBUGGED FROM "WIPING IT OUT," SHOULD THE NEW PROGRAM HAVE A NEVER-ENDING LOOP IN IT WHICH TRIES TO STORE SOME DATA IN EVERY MEMORY LOCATION THE COMPUTER CAN ACCESS. FINALLY, THE SUBROUTINES OF THE MONITOR PROGRAM WILL ALWAYS BE AVAILABLE FOR OTHER PROGRAMS TO CALL AS THEY REQUIRE.

THE MONITOR PROGRAM IS AN EXTREMELY USEFUL TOOL, AS ANYONE WILL ATTEST TO THAT HAS WORKED ON A COMPUTER WITH AND WITHOUT A MONITOR. IT IS HOPED THAT THIS MONITOR PROGRAM WILL GET THE READER OFF ON THE RIGHT FOOT TOWARDS TRANSFORMING ONE'S COMPUTER SYSTEM FROM A BOX THAT MERELY BLINKS ITS LIGHTS TO A FULLY FUNCTIONAL OPERATING SYSTEM THAT WILL PERFORM MANY OF THE TASKS EXPECTED OF IT.