'8 0 8 0' MONITOR ROUTINES

AUTHOR: ROBERT FINDLEY

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

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 MNEMONICS USED IN THIS MANUAL ARE A COMBINATION OF THE ORIGINAL MNEMONICS SUGGESTED WHEN THE 8008 WAS FIRST MARKETED, AUGMENTED WITH ADDITIONAL MNEMONICS TO REPRESENT THE EXPANDED INSTRUCTION CAPABILITY OF THE 8080 CPU. THE MNEMONICS THAT RELATE EXCLUSIVELY TO THE 8080 UNIT HAVE BEEN CAREFULLY CHOSEN TO CLOSELY APPROXIMATE THOSE SUGGESTED BY INTEL CORPORATION. FOR THOSE READERS THAT ARE NOT FAMILIAR WITH THE MNEMONICS PRESENTED HEREIN, THE APPENDIX AT THE END OF THIS MANUAL WILL SERVE AS A CROSS REFERENCE BETWEEN THE MNEMONICS USED HERE TO THE MNEMONICS IN POPULAR USE AMONG 8080 USERS.

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 '8080' 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 ' 8080 ' 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. THAT IS, BY CAREFULLY EXAMINING THE ROUTINE DESCRIPTIONS, FLOWCHARTS AND LISTING COMMENTS, EQUIVALENT ROUTINES CAN BE WRITTEN FOR OTHER TYPES OF CPU'S.)

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 IS 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, SHOULD BE WRITTEN TO SAVE REGISTER'S B, C, D, E, H AND L AT THE START OF THE ROUTINE BY "PUSHING" THEM ONTO THE STACK AND THEN RETRIEVING THEM, USING THE "POP" INSTRUCTIONS, BEFORE RETURNING TO THE CALLING PROGRAM. THIS PRACTICE PROVIDES A GOOD, GENERAL PURPOSE INPUT ROUTINE FOR USE BY ANY PROGRAM THAT REQUIRES OPERATOR INPUT. 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. WHEN THIS ROUTINE IS CALLED THE STACK POINTER IS, AT MOST, DOWN TWO CALL LEVELS, ALLOWING UP TO 27 STACK LEVELS FOR USE BY THE USER'S ROUTINE, WHICH SHOULD BE "MORE" THAN SUFFICIENT.

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 VARIFY 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 INWHICH 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 SHOULD SAVE THE CONTENTS OF THE CPU REGISTER'S A THRU E, H AND L BY "PUSHING" THEM ONTO THE STACK AND THEN "POPPING" THEM BACK UPON RETURNING TO THE CALLING PROGRAM. AS WITH THE OPERATOR INPUT ROUTINE, THIS ALLOWS THE DISPLAY OUTPUT ROUTINE TO SERVE AS A GENERAL PURPOSE ROUTINE FOR OTHER PROGRAMS. 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 "ERR" 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). WHEN THIS ROUTINE IS CALLED THE STACK POINTER IS, AT MOST, DOWN 3 CALL LEVELS, ALLOWING UP TO 26 LEVELS FOR USE BY THIS ROUTINE.

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 VARITY OF ERROR CHECKING TECHNIQUES POSSIBLE, DEPEN-

DING 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 COMMERICAL 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

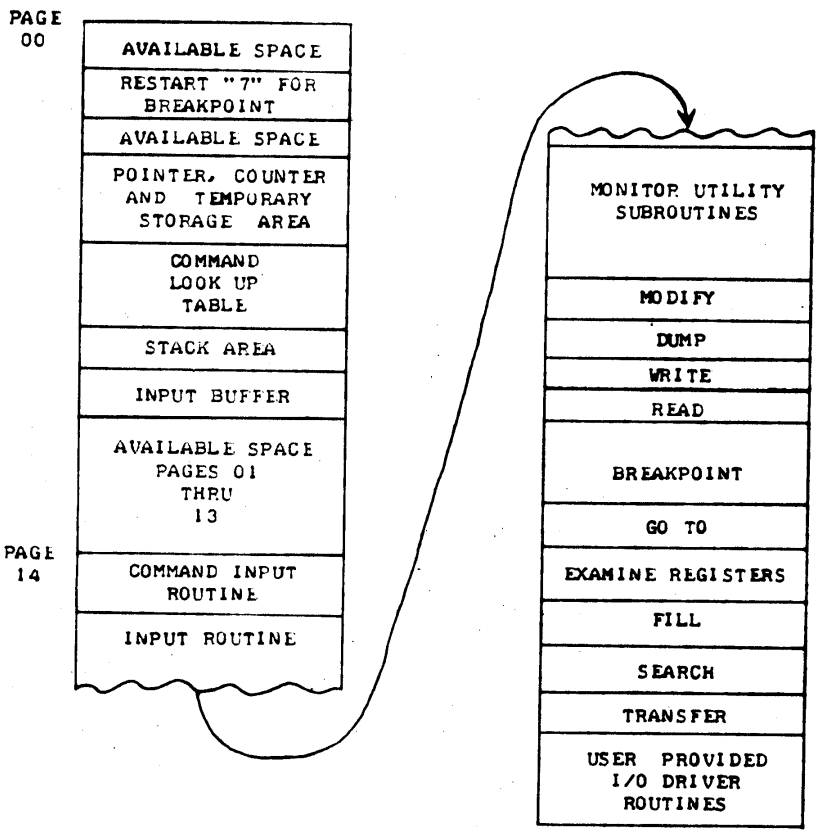
THE MONITOR PROGRAM PRESENTED IN THIS MANUAL MAKES OPTIMUM USE OF THE MEMORY BY UTILIZING EFFECTIVE PROGRAMMING TECHNIQUES WHICH TAKE ADVANTAGE OF THE '8080' INSTRUCTION SET. THE ACTUAL AMOUNT OF MEMORY USED BY THE MONITOR WILL VARY DEPENDING ON THE NUMBER OF COMMANDS ONE INCLUDES IN ONE'S VERSION AND 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 CONVENIENT (FOR EXAMPLE, THE BULK STORAGE I/O ROUTINES MAY ALREADY RESIDE IN MEMORY ON A "PROM"). PORTIONS OF PAGE 000 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 000 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 000 TO ALLOW ROOM FOR EXPANSION, AS WILL BE EXPLAINED LATER. THE PROGRAM'S "STACK" STARTS AT LOCATION 337 ON PAGE 000 AND WORKS DOWN FROM THAT POINT, ALLOWING UP TO 29 CALL LEVELS WITH THE CURRENT NUMBER OF COMMANDS DEFINED.

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 000 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 ANOTHER, 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 CPU REGISTERS AND FLAG STATUS ARE SAVED.

"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. THE CPU REGISTERS, INCLUDING THE STACK POINTER, AND THE FLAG STATUS WILL BE SET TO THE VALUES STORED IN THE "VIRTUAL" CPU CPU REGISTER STORAGE. THESE VALUES ARE ENTERED BY EITHER THE "EXAMINE REGISTER" COMMAND OR BY THE LAST "BREAKPOINT" ENCOUNTERED.

"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 AND 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	B HHH LLL
MEMORY DUMP	D HHH LLL,MMM NNN
MEMORY FILL	F HHH LLL,MMM NNN,DDD
GO TO	G 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," "L" OR "S" 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 THROUGH-OUT 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.

THIS FIRST "UTILITY" SUBROUTINE PERFORMS THE TYPE OF OPERATION FOUND IN MOST PROGRAMS WHICH STORE DOUBLE PRECISION MEMORY POINTERS IN A TABLE IN MEMORY. THIS SUBROUTINE INCREMENTS A DOUBLE PRECISION VALUE STORED

IN CONSECUTIVE MEMORY LOCATIONS. THE LISTING FOR THIS SUBROUTINE IS PRESENTED BELOW.

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

INCR, INM              /INCR CONTENTS OF MEM LOC
RFZ                    /IF NOT ZERO, RET
INL                    /PNT TO NXT LOC
INM                    /INCR 2ND HALF
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 "MCONT" OUTPUTS A "CR/LF" FOLLOWED BY THE MEMORY ADDRESS CONTAINED IN LOCATIONS 166 AND 167 ON PAGE 000. LOCATION 167, WHICH CONTAINS THE PAGE PORTION OF THE ADDRESS, IS PRINTED 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. THE MEMORY ADDRESS IS PRINTED BY CALLING THE SUBROUTINE "PRT166" WHICH SETS UP EACH HALF OF THE ADDRESS AND CALLS "OCTOUT" TO PRINT THEM. "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 SUBROUTINE LABELED "MEMPRT" PRINTS THE CONTENTS OF THE MEMORY LOCATION INDICATED BY THE POINTER AT LOCATION 166 AND 167 ON PAGE 000. THIS ROUTINE FETCHES THE MEMORY CONTENTS AND THEN CALLS "OCTOUT" PRINT IT.

```
    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, LXH 134 000      /SET PNTR TO C/R,L/F MSG
/
MSG, LAM               /FETCH CHAR TO PRINT
NDA                    /END OF MSG CHAR?
RTZ                    /YES, RET TO CALLING PGM
CAL PRINT              /NO, PRINT CHAR
INXH                   /INCR MEM PNTR
JMP MSG                /CONTINUE PRINT OUT
/
```

```
    MNEMONIC                COMMENTS
 ---------------          --------------
/
MCONT, CAL HDLN       /PRINT C/R, L/F
JMP PRT166            /PRINT ADDR TO MODIFY AND RET
/
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
/
PRT166, LXH 167 000 /SET PNTR TO LO ADDR
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
RET
/
MEMPRT, LHLD 166 000 /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 000. 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 000. 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, LAE
STA 165 000             /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
LDAD                    /FETCH NXT CHAR
CPI 254                 /CHAR = COMMA?
JFZ SGL                 /NO, ONLY ONE ENTRY
INE                     /YES, INCR INP BFR PNTR
LAE
STA 165 000             /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
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, LXH 150 000      /SET PNTR TO DIGIT TABLE
LMH                     /CLEAR TBL BY STORING 000.
INL
LMH
INL
LMH
LOOP, CAL FNUM          /CHECK FOR VALID NUMBER
JTS CKLNH               /IF NOT, CHECK CHAR CNT = 0
LDAD                    /FETCH CHAR
NDI 007                 /MASK OFF 260
LXH 150 000             /SET PNTR TO DIGIT TABLE
LBM                     /TABLE AT LOC 150 PG 00
```

```
    MNEMONIC                COMMENTS
 --------------          --------------
LMA                     /AND SHIFT OTHER NUMBERS
INL                     /UP THRU THE TABLE
LAM
LMB
INL
LMA
INE                     /INCR INP BFR PNTR
JMP LOOP                /FETCH NXT NUMBER
/
CKLNH, LTA 165 000      /FETCH ORIG INP BFR PNTR
CPE                     /IS CHAR CNT = 0?
JTZ ERR                 /YES, PRINT ERROR MSG
CAL OCT                 /FETCH FINAL OCTAL NUMBER
JFS ERR                 /IF INVALID, PRINT ERR MSG
RET                     /ELSE, RET TO CALLING PGM
/
FNUM, LDAD              /FETCH ASCII DIGIT
CPI 260                 /VALID NUMBER?
RTS                     /NO, RET WITH S FLAG SET
SUI 270                 /CHECK UPPER LIMIT BY
ADI 200                 /SETTING S FLAG TO PROPER
RET                     /STATE AND RETURN
/
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 000 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, LXH 171 000   /SET PNTR 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                  /IS 1ST HALFS EQUAL?
JTZ INCMD            /YES, RET TO CMND MODE
CONT, LLI 166        /NO, SET PNTR TO LO ADDR
JMP INCR
/
```

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 000. 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. THIS SUBROUTINE IS USED TO ENTER AT MOST 4 CHARACTERS INTO THE INPUT BUFFER. THEREFORE, IF THE INPUT BUFFER SHOULD BECOME FULL, INDICATING UP TO 32 CHARACTERS ENTERED, AN ERROR CONDITION IS ASSUMED AND THE PROGRAM JUMPS TO THE ILLEGAL ENTRY MESSAGE ROUTINE.

```
    MNEMONIC                COMMENTS
 --------------          --------------
INSPCL, CAL COLON    /PRINT COLON
LXD 340 000          /SET PNTR TO S.A. OF INP BFR
LAE
STA 165 000          /SAVE S.A. OF INP BFR
LPIN, CAL RCV        /INP CHAR
STAD                 /STORE CHAR IN INP BFR
CPI 240              /CHAR = SPACE?
JTZ LPO              /YES,
CPI 215              /NO, CHAR = C/R?
JTZ INCMD            /YES, RET TO COMMAND MODE
INE                  /NO, INCR INP BFR PNTR
JTZ ERR              /INP BFR FULL? YES, ERROR
JMP LPIN             /NO, INP NXT CHAR
LPO, LAI 340         /SET UP TEST FOR CHAR COUNT
CPE
RET                  /RET TO CALLING PGM
```

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 000 AND THEN CALLS "DCDNM" TO FETCH THE DATA BYTE, WHICH IS RETURNED IN REGISTER B.

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

ADRDTA, LEI 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
```

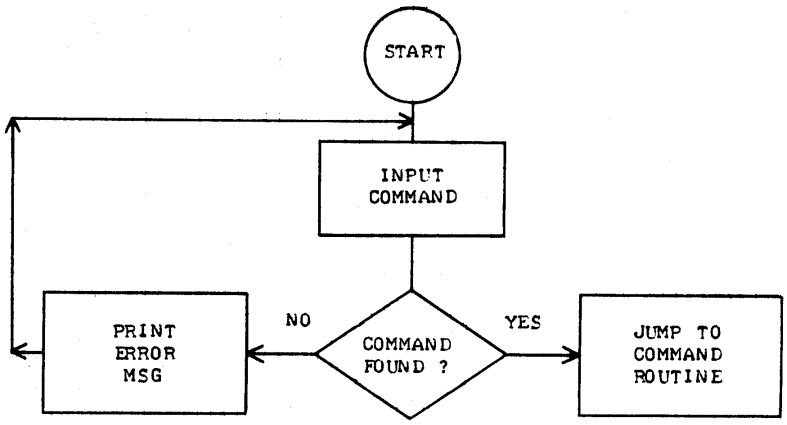
MAJOR ROUTINES FOR THE MONITOR PROGRAM

"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 ON THE FOLLOWING PAGE. AS THE FLOW CHART INDICATES, THE BASIC CONCEPT OF THIS ROUTINE IS QUITE SIMPLE AND STRAIGHT-FORWARD.

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 IDENTIFICATION 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. 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 THE PROGRAM COUNTER, THUS CAUSING THE PROGRAM TO "JUMP" TO THE DESIRED ROUTINE, AS INDICATED BY THE COMMAND. IF NO MATCH IS FOUND IN THE TABLE, AN ILLEGAL ENTRY MESSAGE IS OUTPUT TO THE DISPLAY DEVICE AND THE PROGRAM



THEN RETURNS TO THE START OF THE COMMAND INPUT ROUTINE TO RECEIVE A NEW COMMAND ENTRY. THE FORMAT FOR THE COMMAND "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   OOO = 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
```

THE "STACK POINTER" IN THIS MONITOR IS SET UP AT LOCATION 340 ON PAGE 000 INITIALLY. THE STACK CAN THUS ACCEPT UP TO 29 CALLS WITH THE CURRENT SIZE OF THE COMMAND LOOK UP TABLE, SINCE IT MAY GO FROM LOCATION 337 DOWN TO LOCATION 246, WHICH IS THE CURRENT END OF THE COMMAND LOOK UP TABLE. THE MONITOR PROGRAM ACTUALLY ONLY USES AT MOST FIVE CALL LEVELS, NOT COUNTING THOSE THAT THE USER SUPPLIED I/O ROUTINES MAY REQUIRE.

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 6'TH INSTRUCTION (LDI 012) IN THE COMMAND INPUT ROUTINE. THE ACTUAL OP-

ERATING PORTION OF THE COMMAND INPUT ROUTINE AND, THUS, THE MONITOR PROGRAM ITSELF, STARTS AT THE INSTRUCTION LABELED "INCMD."

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

315                     /MODIFY
107
015
304                     /DUMP
235
015
327                     /BULK WRITE
301
015
322                     /BULK READ
323
015
302                     /BREAKPOINT
331
015
307                     /GO TO
041
016
330                     /EXAMINE REGISTERS
062
016
306                     /FILL MEM
005
017
323                     /SEARCH
022
017
324                     /TRANSFER
061
017
/
/COMMMAND INPUT ROUTINE
/
ORG 014 000
INCMD, LXS 340 000   /SET STACK POINTER
LXH 130 000          /SET PNTR TO HEADING MSG
CAL MSG              /PRINT C/R, L/F, >
CAL CDIN             /INPUT COMMAND FM KYBD
LTA 340 000          /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
/
```

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

FOUND, INL              /ADV CMND TBL PNTR
LEM                     /FETCH CMND LO ADDR
INL
LDM                     /FETCH CMND PAGE ADDR
XCHG                    /SET UP JUMP ADDR
PCHL                    /JUMP TO COMMAND RTN
```

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.

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.

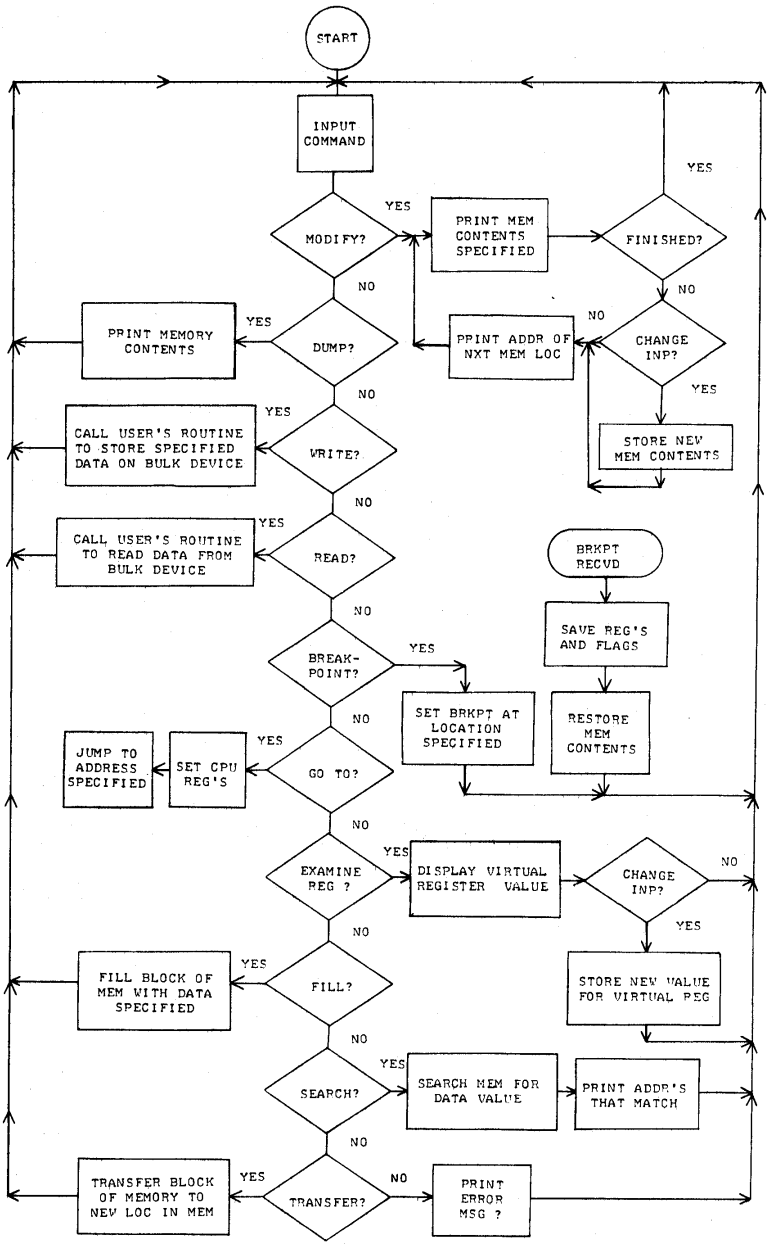
THE FLOW CHART FOR THE INPUT ROUTINE IS PRESENTED ON PAGE 21. THE READER MAY REFER TO THIS DURING THE FOLLOWING DISCUSSION.

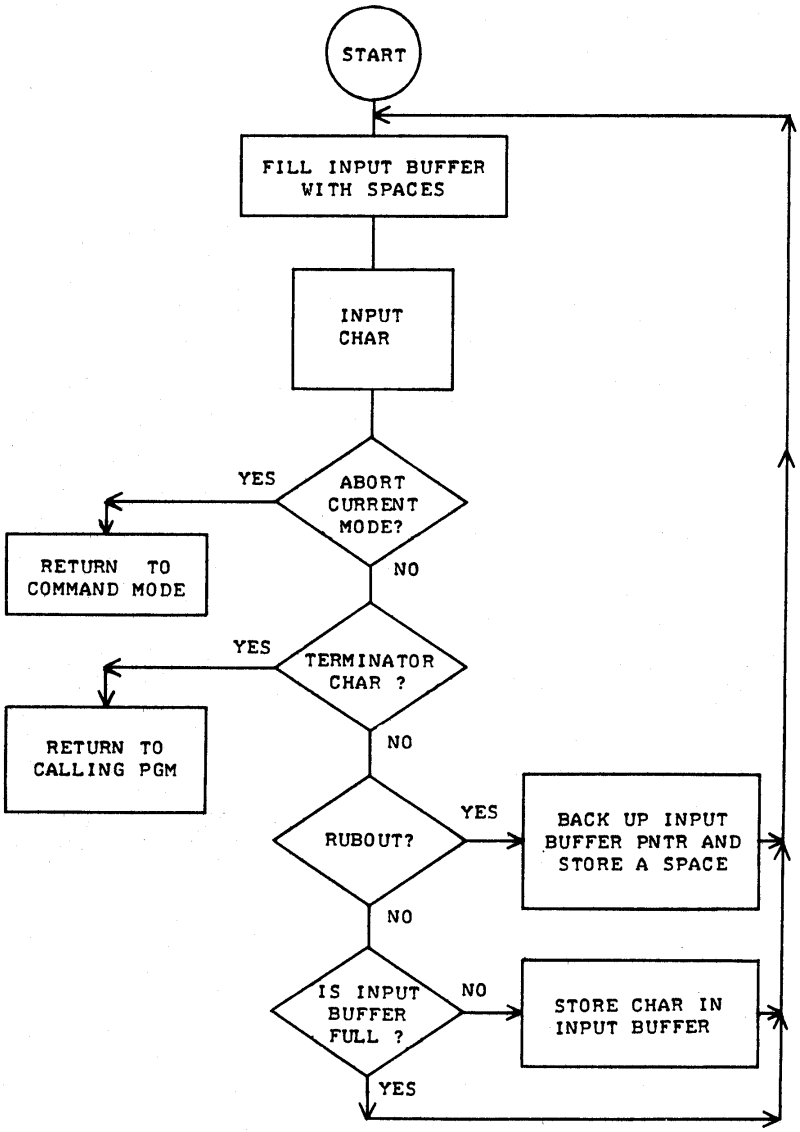
THE FIRST OPERATION PERFORMED BY THIS ROUTINE IS TO "CLEAR OUT" THE INPUT BUFFER AREA. THIS IS ACCOMPLISHED BY FILLING THE INPUT BUFFER AREA WITH THE ASCII CODE FOR A SPACE, '240' OCTAL. THE START ADDRESS OF THE INPUT BUFFER IS THEN SET UP TO BEGIN STORING CHARACTERS AS THEY ARE ENTERED VIA THE "RCV" 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


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



INPUT ROUTINE FLOW CHART

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 ERASE ONE CHARACTER ENTRY FROM THE INPUT BUFFER. AN OPERATOR MAY ERASE 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 IS SHOWN BELOW. THE START OF THIS ROUTINE IS AT THE INSTRUCTION LABELED "CDIN."

```
   MNEMONIC              COMMENTS
-------------        --------------
/
CDIN, LLI 340        /SET PNTR TO START OF INP BFR
SP1, LMI 240         /FILL INP BFR WITH SPACES
INL                  /INCR INP BFR PNTR
JFZ SP1              /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 DE-

SIRES 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.

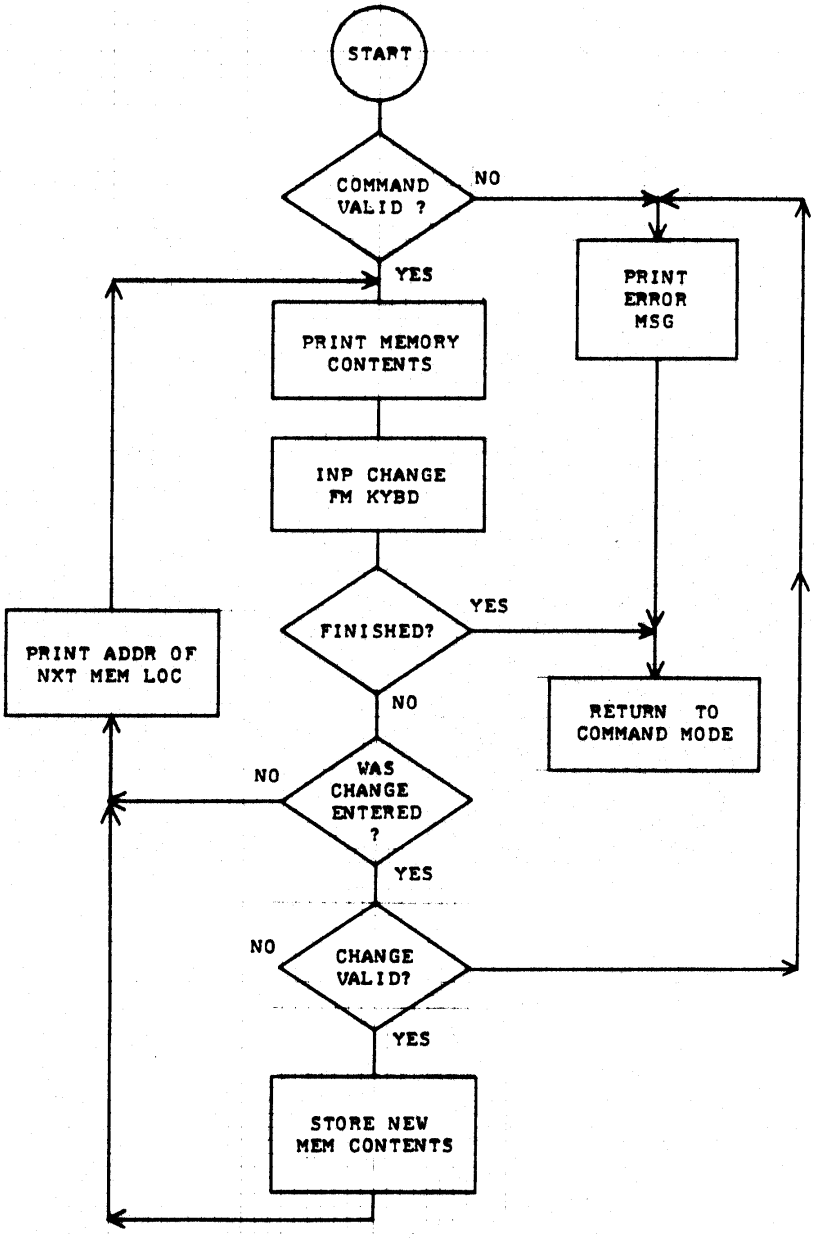
THE "MODIFY" ROUTINE

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 000. THE "MODIFY" ROUTINE 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, LEI 342       /SET INP BFR PNTR
CAL OCTNM             /FETCH ADDR TO MODIFY
CAL SPAC              /PRINT SPACE
MOD1, CAL MEMPRT      /PRINT CONTENTS OF MEM LOC
CAL INSPCL            /INP MODIFICATION
JTZ NXLOC             /NO, SET UP NXT LOC
LEA                   /YES, SAVE INP PNTR
CAL DCDNM             /CONVERT TO OCTAL NUMBER
LAB                   /SAVE OCTAL NUMBER
LHLD 166 000          /SET PNTR TO MEM LOC
LMA                   /LOAD MEM WITH NEW VALUE
NXLOC, LXH 166 000    /SET PNTR TO MEM ADDR STRAGE
CAL INCR              /INCR MEM ADDR
CAL MCONT             /PRINT NXT ADDR TO MODIFY
JMP MOD1
```



MEMORY "MODIFY" ROUTINE FLOW CHART

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.

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 000. 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.

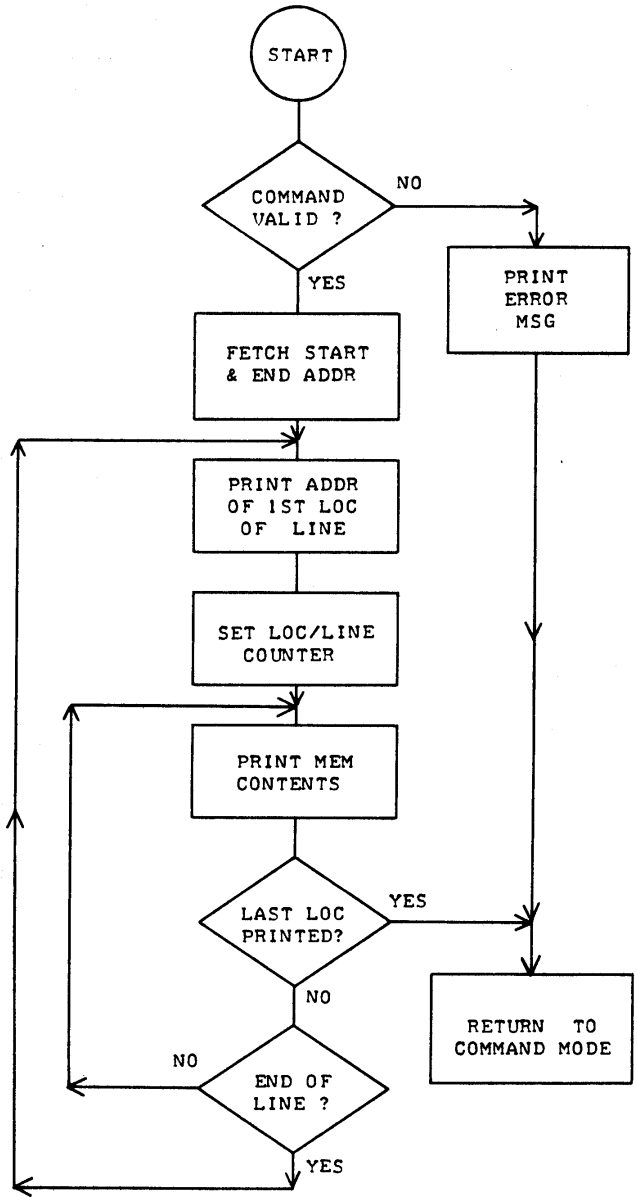
THE DETAILED LISTING FOR THE "DUMP" ROUTINE IS GIVEN BELOW WITH THE FLOW CHART PRESENTED ON THE FOLLOWING PAGE.

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

MDUMP, LEI 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
DCM                     /DECR 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



MEMORY "DUMP" ROUTINE FLOW CHART

WRITE ROUTINE. THE ADDRESSES ARE ALSO CONTAINED IN THE DATA AREA ON PAGE 000, 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.

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

/
WRITE, LEI 342          /SET PNTR TO INP BFR
CAL OCTNM               /FETCH START AND END ADDR
LHLD 170 000
XCHG                    /SET END ADDR
LHLD 166 000            /SET START ADDR
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.

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 START ADDRESS OF THE SECOND ROUTINE "BRK" 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 TWO 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 INSTRUCTION LABELED "BRK." THIS FIRST BOUTINE IS LABELED "BREAK."

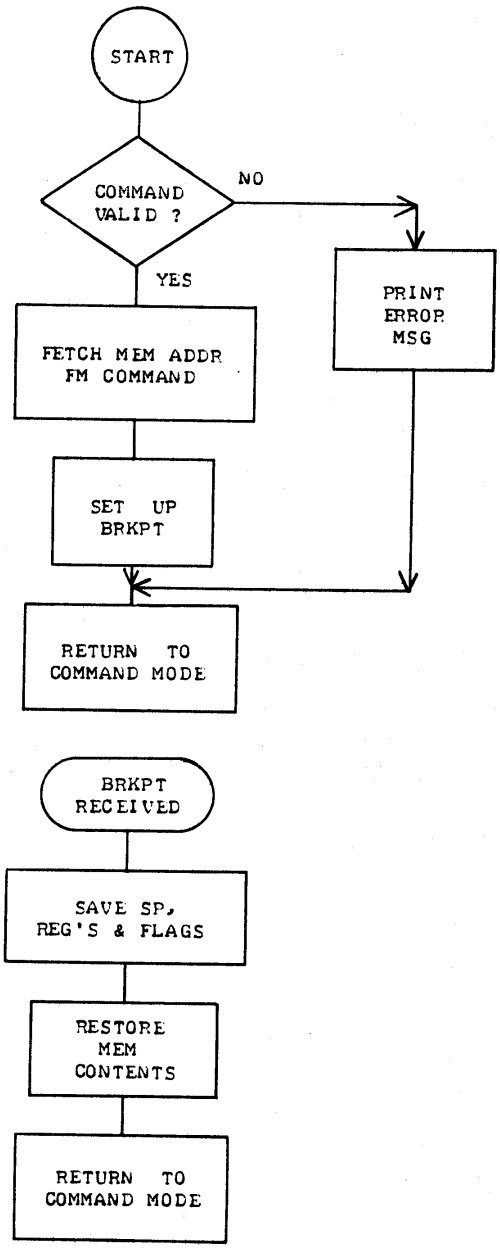
THE SUBROUTINE LABELED "ANLYZ" IS USED BY BOTH THE BREAKPOINT ROUTINE AND THE "GO TO" ROUTINE. FOR THE BREAKPOINT ROUTINE, IT SIMPLY FETCHES THE ADDRESS AT WHICH THE BREAKPOINT IS TO BE LOCATED. HOWEVER, FOR THE "GO TO" ROUTINE IT ALSO SETS UP THE JUMP INSTRUCTION USED TO JUMP TO THE DESIGNATED ADDRESS. SETTING UP THIS JUMP INSTRUCTION WILL NOT HAVE ANY ADVERSE AFFECT ON THE BREAKPOINT ROUTINE, EVEN THOUGH IT IS NOT REQUIRED.

THE SECOND ROUTINE SHOWN ON THE FLOW CHART IS THE ROUTINE WHICH IS ENTERED AT THE TIME THE BREAKPOINT IS REACHED. THE CPU REGISTERS, STACK POINTER AND FLAG STATUS ARE STORED IN THE "VIRTUAL" CPU REGISTER STORAGE TABLE ON PAGE 000. THE READER WILL NOTE THAT IN ORDER TO SAVE THE STACK POINTER IT IS NECESSARY TO FIRST "PUSH" THE CURRENT FLAG STATUS ONTO THE STACK USED BY THE PROGRAM BEING EXECUTED AND THEN "ADD" THE STACK POINTER TO THE H AND L REGISTERS WHERE IT MAY BE LOADED DIRECTLY INTO THE STACK POINTER STORAGE LOCATION. THE CONSTANT WHICH IS ADDED TO THE STACK POINTER ADJUSTS IT TO ITS VALUE AT THE TIME THE BREAKPOINT WAS ENCOUNTERED. THE FLAG STATUS IS THEN "POPPED" BACK TO ITS ORIGINAL CONTENTS FOR STORAGE IN THE TABLE ON PAGE 000. AFTER THE REGISTERS ARE STORED, THE BREAKPOINT ROUTINE THEN RESTORES THE ORIGINAL INSTRUCTION AT THE BREAKPOINT LOCATION TO ITS ORIGINAL CONTENTS AND RETURNS TO THE COMMAND INPUT ROUTINE.

THE LISTINGS FOR THE BREAKPOINT ROUTINES ARE PRESENTED NEXT.

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

BREAK, CAL ANLYZ      /SET UP ADDRESS OF BP
LAM                   /SAVE ORIG CONTENTS OF BP
LMI 377               /INSERT BP RESTART INSTR.
XCHG                  /SAVE BP ADDR
LXH 070 000           /SET PNTR TO RST 7 LOC
LMI 303               /STORE JUMP INSTR
INL
LMI 005               /**** STORE BRK LO ADDR
INL
LMI 016               /**** STORE BRK PG ADDR
INL
LME                   /STORE BP ORIG LOW ADDR
INL
LMD                   /STORE BP ORIG PG ADDR
INL
LMA                   /STORE ORIG BP INSTRUCTION
JMP INCMD
```



THE "BREAKPOINT" ROUTINES FLOW CHART

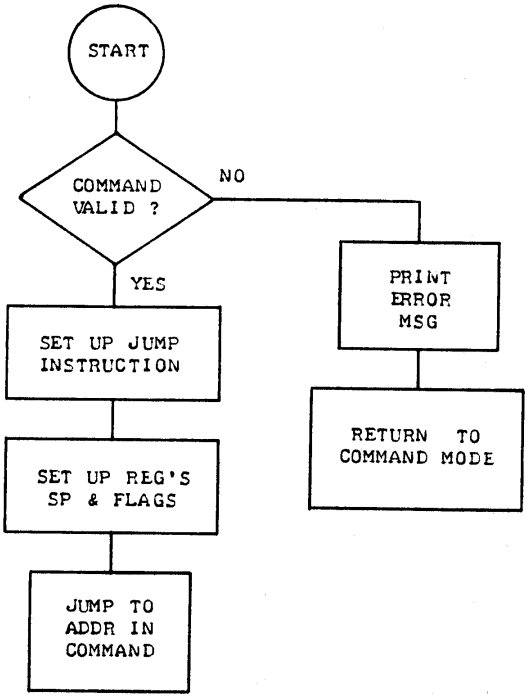
```
    MNEMONIC                COMMENTS
 --------------          --------------
/
ANLYZ, LEI 342          /SET PNTR TO BUFF SA
CAL OCTNM               /FETCH ADDR INTO 166, 167
LAI 303                 /SET JUMP INSTR FOR GOTO
STA 155 000
LHLD 166 000            /GET BP ADDR
SHLD 156 000            /STORE BP ADDR IN JUMP INSTR
RET
/
BRK, SHLD 206 000       /SAVE H & L
PUSS                    /SAVE STATUS & REG A
LXH 004 000             /SET UP TO SAVE SP
DADS                    /MOVE SP TO REG'S H & L
SHLD 176 000            /SAVE ORIG SP
POPS                    /RESTORE STATUS
LXS 206 000             /SET SP TO REG STRAGE
PUSD                    /SAVE REG'S D & E
PUSB                    /SAVE REG'S B & C
PUSS                    /SAVE STATUS WORD & REG A
LHLD 073 000            /SET BP PNTR
LTA 075 000             /FETCH ORIG INSTR
LMA                     /RESTORE ORIG BKPNT INSTR
JMP INCMD               /BACK TO MONITOR
```

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. THE "ANLYZ" SUBROUTINE, DESCRIBED IN THE "BREAKPOINT" ROUTINE, IS CALLED TO FETCH THE ADDRESS FROM THE COMMAND ENTERED AND SET UP THE JUMP INSTRUCTION WHICH WILL BE EXECUTED AT THE END OF THIS ROUTINE. BEFORE JUMPING TO THE ADDRESS INDICATED, THE "GO TO" ROUTINE WILL SET THE CPU REGISTERS, STACK POINTER AND FLAG STATUS TO THE VALUES STORED IN THE "VIRTUAL" CPU REGISTER TABLE ON PAGE 000. THE VALUES STORED IN THIS TABLE ARE SET BY EITHER THE "EXAMINE REGISTER" ROUTINE, TO BE PRESENTED NEXT, OR BY THE LAST BREAKPOINT ENCOUNTERED. THUS, THE "GO TO" ROUTINE CAN BE USED TO CONTINUE EXECUTING A PROGRAM BEING DEBUGGED AT THE POINT OF THE LAST BREAKPOINT. SINCE THE "GO TO" ROUTINE RESTORES THE CPU REGISTERS AND FLAGS, THE PROGRAM CAN BE ENTERED AT THE LAST BREAKPOINT AS THOUGH IT WAS NEVER INTERRUPTED BY THE BREAKPOINT. THE "GO TO" ROUTINE STARTS AT THE LOCATION LABELED "GOTO."

THE LISTING AND FLOW CHART FOR THE "GO TO" ROUTINE ARE PRESENTED NEXT.

```
    MNEMONIC                COMMENTS
 --------------          --------------
/
GOTO, CAL ANLYZ         /SET UP ADDR OF GOTO
LXS 176 000             /SET SP TO REG STRAGE
POPH                    /FETCH SP IN H & L
POPS                    /SET UP STATUS AND REG A
POPB                    /SET UP REG'S B & C
POPD                    /SET UP REG'S D & E
SPHL                    /SET UP SP
LHLD 206 000            /SET UP REG H & L
JMP 155 000             /START PROGRAM
```



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 000 AT LOCATIONS 176 THRU 207. THE "VIRTUAL" CPU REGISTERS AND FLAG STATUS ARE ASSIGNED THE FOLLOWING LOCATIONS IN THE CPU REGISTER TABLE.

LOCATION	REGISTER
000 176	STACK POINTER LO ADDR
000 177	STACK POINTER PG ADDR
000 200	FLAG STATUS BYTE
000 201	REGISTER A
000 202	REGISTER C
000 203	REGISTER B
000 204	REGISTER E
000 205	REGISTER D
000 206	REGISTER L
000 207	REGISTER H

THE CONTENTS OF THE "VIRTUAL" CPU REGISTERS AND THE SETTING OF THE

FLAG STATUS MAY BE MODIFIED BY ENTERING THE REVISION AFTER THE CURRENT VALUE IS DISPLAYED, IN A MANNER SIMILAR TO THE "MODIFY" ROUTINE. THE REGISTERS ARE MODIFIED BY ENTERING A THREE DIGIT OCTAL NUMBER. THE STACK POINTER REQUIRES AN ADDRESS ENTRY AND THE FLAG STATUS IS DISPLAYED AND ITS MODIFICATIONS ARE ENTERED BY INPUTTING THE FIRST LETTER OF THE FLAG NAME. THE DEFINITION OF THE BIT POSITIONS IN THE FLAG STATUS BYTE ARE GIVEN IN THE FOLLOWING TABLE. THE LETTER IN PARENTHESIS INDICATES THE DESIGNATION USED TO DISPLAY AND MODIFY THE FLAG STATUS.

B7 = SIGN FLAG (S)
B6 = ZERO FLAG (Z)
B5 = ALWAYS "0"
B4 = AUXILIARY CARRY FLAG (A)
B3 = ALWAYS "0"
B2 = PARITY FLAG (P)
B1 = ALWAYS "1"
B0 = CARRY FLAG (C)

THE ROUTINE STARTS BY FETCHING THE REGISTER DESIGNATION FROM THE INPUT BUFFER AND, USING THE SUBROUTINE "TBLCK," SEARCHES THE TABLE LABELED "RGTBL" FOR A MATCH WITH THE REGISTER DESIGNATION. THIS SUBROUTINE OPERATES IN THE SAME MANNER AS THE COMMAND INPUT ROUTINE, CHECKING EVERY OTHER LOCATION IN THE "RGTBL" FOR A MATCH. THE ONLY REAL DIFFERENCE BEING THAT THE END OF THE TABLE IS DETERMINED BY AN ALL ZERO BYTE. IF A MATCH IS FOUND, ONE OF THREE ROUTINES ARE ENTERED TO DISPLAY AND MODIFY THE REGISTER. IF NO MATCH IS FOUND, AN ILLEGAL ENTRY ERROR MESSAGE IS DISPLAYED AND THE ROUTINE RETURNS TO THE COMMAND MODE.

IF THE REGISTER DESIGNATED IS ONE OF THE CPU GENERAL REGISTERS OR THE ACCUMULATOR, THE NEXT LOCATION IN THE "RGTBL" IS USED TO INDICATE THE LOCATION AT WHICH THE DESIGNATED REGISTER IS STORED IN THE "VIRTUAL" CPU REGISTER TABLE. THE CURRENT VALUE OF THE REGISTER IS PRINTED, AND THE "INSPCL" SUBROUTINE IS CALLED TO ENTER ANY MODIFICATION THAT MAY BE DESIRED. IF A MODIFICATION IS ENTERED, THE "DCDNM" SUBROUTINE DECODES THE OCTAL NUMBER FROM THE ENTRY AND THIS VALUE IS STORED IN THE PROPER LOCATION IN THE "VIRTUAL" CPU REGISTER TABLE.

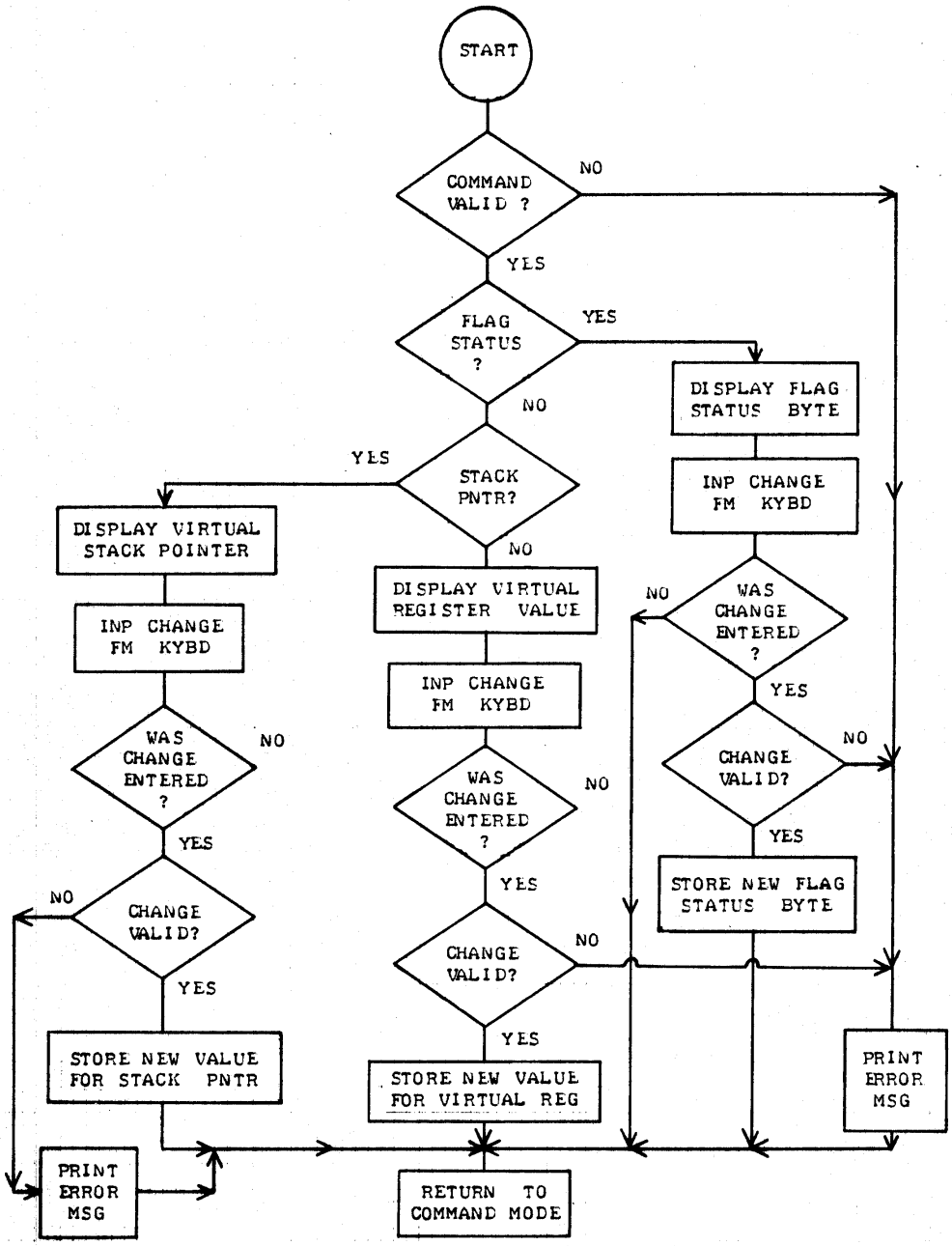
IF THE STACK POINTER IS DESIGNATED IN THE COMMAND, THE PROGRAM JUMPS TO A ROUTINE SIMPLY LABELED "S" WHICH DISPLAYS THE ADDRESS STORED FOR THE STACK POINTER, IN THE STANDARD FORMAT OF THE PAGE PORTION FOLLOWED BY THE LOW ADDRESS PORTION, AND THEN CALLS THE "INSPCL" SUBROUTINE TO ENTER THE MODIFICATION. IF A MODIFICATION IS ENTERED, THE PAGE AND LOW ADDRESS PORTIONS MUST BE SEPARATED BY A CHARACTER OTHER THAN A SPACE SINCE "INSPCL" ACCEPTS A SPACE AS A TERMINATOR CHARACTER. THEREFORE, A COMMA IS SPECIFIED AS THE CHARACTER WHICH MUST SEPARATE THE PAGE AND LOW ADDRESS ENTRY FOR THE STACK POINTER MODIFICATION, ALTHOUGH ANY CHARACTER OTHER THAN A SPACE OR CARRIAGE RETURN WILL WORK. THE MODIFICATION IS CONVERTED TO TWO OCTAL NUMBERS AND STORED IN THE "VIRTUAL" CPU REGISTER TABLE AT LOCATIONS 176 AND 177.

IF THE FLAG STATUS IS DESIGNATED, THE ROUTINE LABELED "F" IS ENTERED AND THE FLAG DESIGNATION CHARACTERS (INDICATED IN PARENTHESIS ABOVE) ARE DISPLAYED FOR EACH FLAG BIT WHICH IS SET TO A "1" IN THE FLAG BYTE. THE "INSPCL" SUBROUTINE IS THEN CALLED TO INPUT ANY MODIFICATION TO THE FLAG STATUS. THE MODIFICATIONS ARE MADE BY ENTERING THE FLAG DESIGNATION CHARACTERS FOR THE FLAGS WHICH ARE TO BE SET TO A "1." THE FLAGS WHICH ARE NOT ENTERED IN THE MODIFICATION ENTRY WILL BE SET TO A "0." WHEN A MODIFICATION IS ENTERED, THE "FTBL" IS SEARCHED BY THE "TBLCK" SUBROUTINE TO DETERMINE WHICH FLAG DESIGNATIONS HAVE BEEN ENTERED AND A

NEW FLAG STATUS BYTE IS FORMED. ONCE FORMED, THE FLAG STATUS BYTE IS CHECKED FOR AN ILLEGAL SET UP. THAT IS, IF THE ZERO FLAG IS A "1," THE PARITY FLAG MUST ALSO BE A "1" AND THE SIGN FLAG MUST BE A "0." THIS CONDITION IS TESTED AND IF FOUND TO BE IN ERROR, THE ILLEGAL ENTRY ERROR MESSAGE IS DISPLAYED. IF THE ENTRY IS VALID, THE NEW FLAG STATUS BYTE IS STORED AT LOCATION 200 IN THE "VIRTUAL" CPU REGISTER TABLE.

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

```
    MNEMONIC                COMMENTS
    --------                --------
XREG, LXH 341 000     /SET INP BFR PNTR
LXD RGTBL             /SET REG TABLE PNTR
CAL TBLCK             /SEARCH FOR REG DESIGNATED
CPI 306               /FLAG STATUS?
JTZ F                 /YES, PRINT FLAGS
CPI 323               /STACK POINTER?
JTZ S                 /YES, PRINT ADDRESS
LLI 164               /SET PNTR TO TEMP STRAGE
INXD                  /INCR REG TBL PNTR
LDAD                  /FETCH REG STRAGE PNTR
LMA                   /SAVE REG STRAGE PNTR
LLA                   /SET PNTR TO REG VALUE
CAL SPAC              /PRINT SPACE
LAM                   /FETCH CURRENT REG VALUE
CAL OCTOUT            /PRINT CURRENT REG VALUE
CAL INSPCL            /INP MODIFICATION
JTZ INCMD             /NO ENTRY, RET TO CMND
LEA                   /SAVE INP BFR PNTR
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
/
F, CAL SPAC           /PRINT SPACE
LLI 200               /SET REG TBL PNTR
LAM                   /FETCH FLAG BYTE
LXD FTBL              /SET PNTR TO FLAG TBL
INXD                  /ADV PNTR TO BIT WORDS
LBA
PRTBIT, LDAD          /FETCH FLAG BIT
NDA                   /END OF TABLE?
JTZ INFLG             /YES, INP FLAG CHANGES
NDB                   /IS BIT SET?
JTZ TRYNX             /NO, TRY NEXT BIT
DCXD
LDAD                  /FETCH CHAR FOR FLAG
CAL PRINT             /PRINT CHAR FOR FLAG
INXD                  /RESET TBL PNTR
TRYNX, INXD           /ADV REG TBL PNTR
INXD
JMP PRTBIT
INFLG, CAL INSPCL     /INPUT CHANGES
JTZ INCMD             /NO ENTRY, RET TO CMND
LLA                   /SET INP BFR PNTR
LBI 002               /SET BASIC FLAG BYTE
LAM                   /FETCH FLAG CHAR FM INP
```



THE "EXAMINE REGISTER" ROUTINE FLOW CHART

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

    FLAG, LXD FTBL          /SET PNTR TO FLAG TBL
    CAL TBLCK               /SEARCH TBL FOR FLAG CHAR
    INXD                    /FETCH BIT WORD
    LDAD
    ORB                     /ADD BIT TO BASIC BYTE
    LBA                     /SAVE FLAG BYTE
    INL
    LAM                     /FETCH NEXT CHAR FM INP
    CPI 240                 /CHAR = SPACE?
    JFZ FLAG                /NO, MORE STATUS INP
    LAB                     /FETCH NEW STATUS WORD
    NDI 100                 /IS ZERO SET?
    JTZ OK                  /NO, WORD IS O.K.
    LAB                     /YES, CK S AND P FLAGS
    NDI 204                 /SEPARATE S AND P FM OTHERS
    XRI 004                 /S = 0? P = 1?
    JFZ ERR                 /NO, ILLEGAL SET UP
    OK, LLI 200             /SET PNTR TO STATUS BYTE
    LMB                     /SAVE NEW BYTE
    JMP INCMD               /RET TO COMMAND MODE
    /
    S, CAL SPAC             /PRINT A SPACE
    LXH 177 000             /SET PNTR TO SP PG ADDR
    LAM
    CAL OCTOUT              /PRINT SP PG ADDR
    CAL SPAC                /PRINT A SPACE
    LLI 176                 /SET PNTR TO SP LO ADDR
    LAM
    CAL OCTOUT              /PRINT SP LO ADDR
    CAL INSPCL              /INP MODIFICATION
    JTZ INCMD               /NO ENTRY, RET TO CMND
    LEA
    STA 165 000             /SAVE INP PNTR
    CAL OCTPR               /CONVERT ADDR INP
    LXH 176 000             /SET SP STRAGE
    LMB                     /STORE SP LO ADDR
    INL
    LMC                     /STORE SP PG ADDR
    JMP INCMD               /RET TO CMND MODE
    /
    TBLCK, LDAD             /FETCH CHAR
    NDA                     /END OF TABLE?
    JTZ ERR                 /YES, INVALID INPUT
    CPM                     /CHAR MATCH?
    RTZ                     /YES, RET TO CALLING PGM
    INXD                    /NO, ADV TBL PNTR
    INXD
    JMP TBLCK               /CONT. SEARCH
    /
    RGTBL, 301              /REG A CHAR
    201                     /REG A STRAGE
    302                     /REG B CHAR
    203                     /REG B STRAGE
    303                     /REG C CHAR
    202                     /REG C STRAGE
    304                     /REG D CHAR
    205                     /REG D STRAGE
```

```
    MNEMONIC                COMMENTS
 --------------          --------------
305                     /REG E CHAR
204                     /REG E STRAGE
310                     /REG H CHAR
207                     /REG H STRAGE
314                     /REG L CHAR
206                     /REG L STRAGE
306                     /FLAG STATUS CHAR
200                     /FLAG STATUS STRAGE
323                     /STACK PNTR CHAR
176                     /STACK PNTR STRAGE
000
/
FTBL, 323               /SIGN CHAR
200                     /SIGN BIT
332                     /ZERO CHAR
100                     /ZERO BIT
301                     /AUXILIARY CARRY CHAR
020                     /AUXILIARY CARRY BIT
320                     /PARITY CHAR
004                     /PARITY BIT
303                     /CARRY CHAR
001                     /CARRY BIT
000
000
```

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

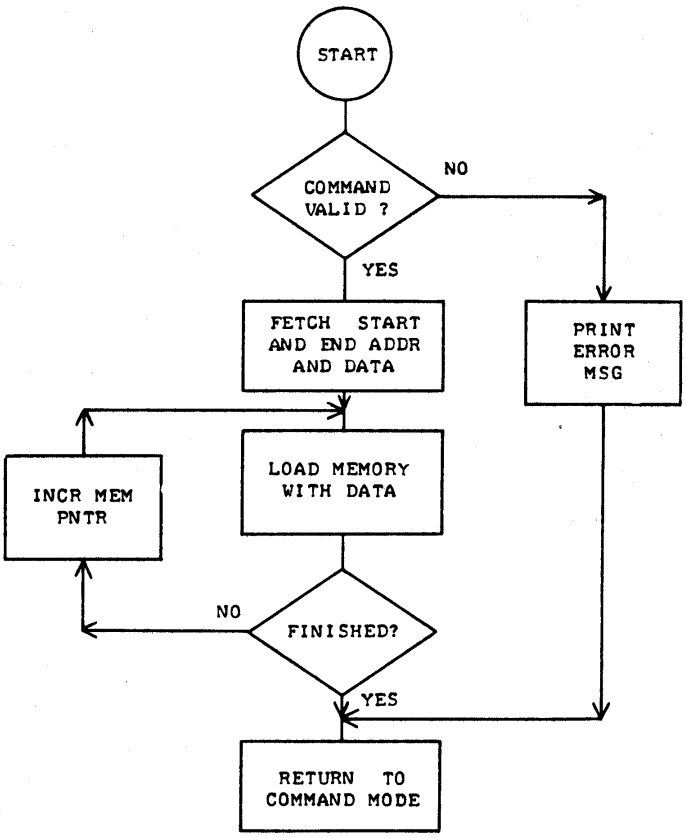
THE MEMORY "FILL" ROUTINE IS USED TO FILL A BLOCK OF MEMORY WITH A SPECIFIC 8 BIT DATA VALUE. THIS ROUTINE IS USEFUL IN "ZEROING" A BLOCK OF MEMORY BEFORE EXECUTING A PROGRAM TO DETERMINE WHETHER THAT PROGRAM IS WRITING INTO THE SECTION OF MEMORY "ZEROED" OUT OR NOT. AS THE READER WILL SEE FROM THE LISTING, 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 BELOW.

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

FILL, CAL ADRDTA     /INP ADDR AND DATA FM BFR
FL1, LHLD 166 000    /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 000, 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.

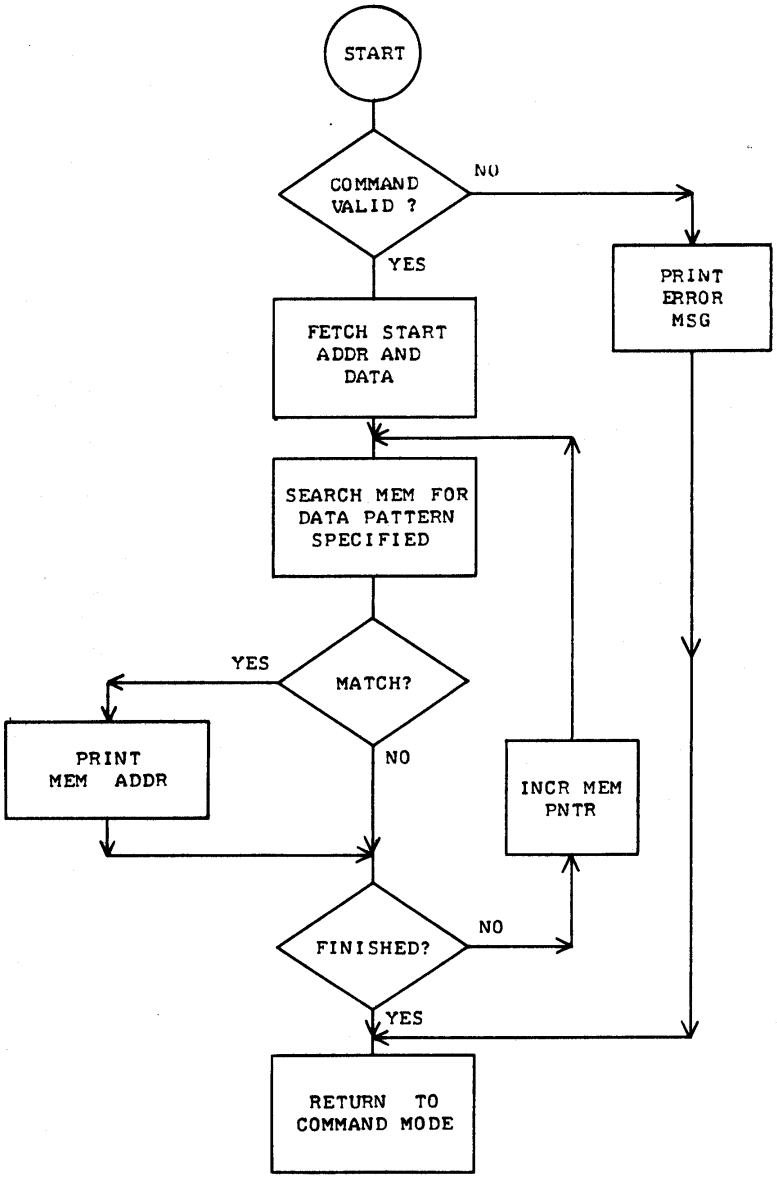
```
   MNEMONIC              COMMENTS
   --------              --------

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
LHLD 166 000         /SET PNTR TO MEM
CPM                  /DATA EQUAL SRCH DATA
CTZ MCONT            /YES, PRINT ADDR
CAL CKEND            /DONE? YES, RET TO CMND MODE
JMP SH1              /NO, CONTINUE SEARCH
/
```

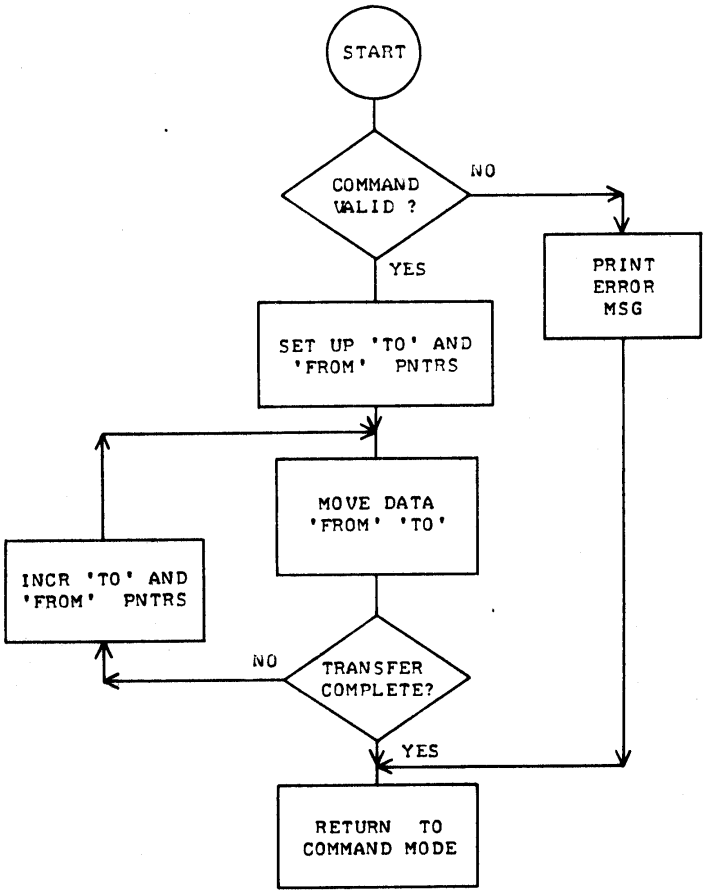
THE "TRANSFER" ROUTINE

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, LEI 342       /SET PNTR TO ADDR INP
CAL OCTNM             /FETCH 'FROM' ADDR
LLI 166               /SET PNTR TO ADDR INP
LBE                   /SAVE INP BFR PNT
LXD 172 000           /SAVE 'FROM' IN TEMP STRAGE
SVSA, CAL SWAP        /MOVE ADDR TO TEMP STRGE
LAI 172               /IS XFR COMPLETE?
CPL
JFZ SVSA              /NO, CONTINUE MOVE
INB
```

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

LEB                     /RESTORE INP BFR PNTR
CAL OCTNM               /INP 'TO' ADDR
LLI 172                 /SET PNTR TO TEMP STRGE
LXD 166 000             /SET PNTR TO TEMP STRAGE
TF1, CAL SWAP           /XFR 'FROM' PNTR
LAI 176
CPL                     /XFR COMPLETE?
JFZ TF1                 /NO, CONTINUE
LEB                     /FETCH 'TO' PNTR
LDC
TF2, LHLD 166 000       /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
INXH              /INCR 'FROM' PNTR
STAD                    /STORE BYTE IN NEW LOC
INXD              /INCR 'TO' PNTR
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 000 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  303 000 000    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            000
000 134  215            215                  /CAR. RET.
000 135  212            212                  /LINE FEED
000 136  000            000
000 137                 /
000 137                 /LOC. 137 THRU 147 AVAILABLE FOR USER
000 137                 /
000 137                 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                 /GO TO JUMP INSTRUCTION
000 155                 /
000 155  303 000 000    JMP 000 000          /GO TO ROUTINE 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                  /"VIRTUAL" STK PNTR LO ADDR
000 177  000            000                  /"VIRTUAL" STK PNTR PG ADDR
000 200  000            000                  /FLAG STATUS BYTE
000 201  000            000                  /VIRTUAL CPU REG "A"
000 202  000            000                  /VIRTUAL CPU REG "C"
000 203  000            000                  /VIRTUAL CPU REG "B"
000 204  000            000                  /VIRTUAL CPU REG "E"
000 205  000            000                  /VIRTUAL CPU REG "D"
000 206  000            000                  /VIRTUAL CPU REG "L"
000 207  000            000                  /VIRTUAL CPU REG "H"
000 210                 /
```

```
000 210                 /COMMAND LOOK UP TABLE
000 210                 /
000 210   315           315                 /MODIFY
000 211   150           150
000 212   015           015
000 213   304           304                 /DUMP
000 214   275           275
000 215   015           015
000 216   327           327                 /BULK WRITE
000 217   343           343
000 220   015           015
000 221   322           322                 /BULK READ
000 222   371           371
000 223   015           015
000 224   302           302                 /BREAKPOINT
000 225   377           377
000 226   015           015
000 227   307           307                 /GO TO
000 230   220           220
000 231   016           016
000 232   330           330                 /EXAMINE REGISTERS
000 233   257           257
000 234   016           016
000 235   306           306                 /FILL MEM
000 236   005           005
000 237   017           017
000 240   323           323                 /SEARCH
000 241   022           022
000 242   017           017
000 243   324           324                 /TRANSFER
000 244   061           061
000 245   017           017
                        /
                        /LOC. 246 THRU 337 RESERVED FOR
                        /MONITOR "PUSH-POP" STACK
                        /
                        /LOC. 340 THRU 377 - INPUT BUFFER
                        /
                        /PAGES 01 THRU 13 AVAILABLE
                        /FOR USER'S PROGRAMS
                        /
                        ORG 014 000
014 000   061 340 000   INCMD, LXS 340 000   /SET STACK POINTER
014 003   041 130 000   LXH 130 000          /SET PNTR TO READING MSG
014 006   315 145 014   CAL MSG              /PRINT C/R, L/F, >
014 011   315 057 014   CAL CDIN             /INPUT COMMAND FM KYBD
014 014   072 340 000   LTA 340 000          /FETCH COMMAND CHAR
014 017   026 012       LDI 012              /SET CMND NMBR CNTR
014 021   056 210       LLI 210              /SET CMND TABLE PNTR
014 023   276           CKCMD, CPM           /IS CMND CHAR FOUND IN TBL?
014 024   312 051 014   JTZ FOUND            /YES, PROCESS COMMAND
014 027   054           INL                  /NO,ADVANCE CMND TBL PNTR
014 030   054           INL
014 031   054           INL
014 032   025           DCD                  /IS LAST CMND CHECKED?
014 033   302 023 014   JFZ CKCMD            /NO, CHECK NEXT
014 036   315 142 014   ERR, CAL HDLN        /YES, PRINT C/R, L/F
014 041   076 311       LAI 311              /ILLEGAL ENTRY CODE
014 043   315 300 017   CAL PRINT            /PRINT ERROR MSG
014 046   303 000 014   JMP INCMD            /INP NEXT COMMAND
```

```
014 051                     /
014 051  054                FOUND, INL          /ADV CMND TBL PNTR
014 052  136                LEM                 /FETCH CMND LO ADDR
014 053  054                INL
014 054  126                LDM                 /FETCH CMND PAGE ADDR
014 055  353                XCHG                /SET UP JUMP ADDR
014 056  351                PCHL                /JUMP TO COMMAND RTN
014 057                     /
014 057  056 340            CDIN, LLI 340       /SET PNTR TO START OF INP BFR
014 061  066 240            SP1, LMI 240        /FILL INP BFR WITH SPACES
014 063  054                INL                 /INCR INP BFR PNTR
014 064  302 061 014        JFZ SP1             /DONE? NO, STORE MORE SPACES
014 067  056 340            LLI 340             /SET INP BFR PNTR
014 071  315 200 017        IN2, CAL RCV        /INP CHAR FM INP DEVICE
014 074  376 204            CPI 204             /CHAR = CNT'L D?
014 076  312 000 014        JTZ INCMD           /YES, RET TO COMMAND MODE
014 101  376 215            CPI 215             /CHAR = CAR RET?
014 103  310                RTZ                 /YES, RET TO CALLING PGM
014 104  376 214            CPI 214             /CHAR = CNT'L L?
014 106  310                RTZ                 /YES, RET TO CALLING PGM
014 107  376 377            CPI 377             /CHAR = RUBOUT?
014 111  312 126 014        JTZ BDCR            /YES, DELETE CHAR FM INP BFR
014 114  054                INL                 /IS INP BFR FULL?
014 115  055                DCL
014 116  312 071 014        JTZ IN2             /YES, DON'T STORE CHAR
014 121  167                LMA                 /NO, STORE CHARACTER
014 122  054                INL                 /INCR INP BFR PNTR
014 123  303 071 014        JMP IN2             /INP NEXT CHAR
014 126                     /
014 126  076 340            BDCR, LAI 340       /SET ACC TO INP BFR S.A.
014 130  275                CPL                 /ANY CHARACTERS YET?
014 131  312 071 014        JTZ IN2             /NO, CONTINUE INPUT
014 134  055                DCL                 /YES, BACK UP INP BFR PNTR
014 135  066 240            LMI 240             /STORE SPACE OVER LAST CHAR
014 137  303 071 014        JMP IN2             /CONTINUE INPUT
014 142                     /
014 142  041 134 000        HDLN, LXH 134 000   /SET PNTR TO C/R,L/F MSG
014 145                     /
014 145  176                MSG, LAM            /FETCH CHAR TO PRINT
014 146  247                NDA                 /END OF MSG CHAR?
014 147  310                RTZ                 /YES, RET TO CALLING PGM
014 150  315 300 017        CAL PRINT           /NO, PRINT CHAR
014 153  043                INXH                /INCR MEM PNTR
014 154  303 145 014        JMP MSG             /CONTINUE PRINT OUT
014 157                     /
014 157  173                OCTNM, LAE
014 160  062 165 000        STA 165 000         /SAVE INP BFR PNTR
014 163  315 237 014        CAL OCTPR           /CONVERT 1ST OCTAL PAIR
014 166  056 166            LLI 166             /SET PNTR TO LO ADDR STRAGE
014 170  160                LMB                 /SAVE LO HALF OF LO ADDR
014 171  054                INL
014 172  161                LMC                 /SAVE PG HALF OF LO ADDR
014 173  032                LDAD                /FETCH NXT CHAR
014 174  376 254            CPI 254             /CHAR = COMMA?
014 176  302 211 014        JFZ SGL             /NO, ONLY ONE ENTRY
014 201  034                INE                 /YES, INCR INP BFR PNTR
014 202  173                LAE
014 203  062 165 000        STA 165 000         /SAVE INP BFR PNTR
014 206  315 237 014        CAL OCTPR           /CONVERT 2ND OCTAL PAIR
014 211  056 170            SGL, LLI 170        /SET PNTR TO HI ADDR STRAGE
```

```
014 213  160           LMB                 /SAVE LO HALF OF HI ADDR
014 214  054           INL
014 215  161           LMC                 /SAVE PG HALF OF HI ADDR
014 216  171           LAC
014 217  056 167       LLI 167             /IS HI ADDR < LO ADDR?
014 221  276           CPM
014 222  332 036 014   JTC ERR             /YES, PRINT ERROR
014 225  300           RFZ                 /IF PG HALF NOT =, RET
014 226  054           INL                 /ELSE, CHECK LO HALF
014 227  176           LAM
014 230  056 166       LLI 166             /IS HI ADDR < LO ADDR?
014 232  276           CPM
014 233  332 036 014   JTC ERR             /YES, PRINT ERROR MSG
014 236  311           RET                 /NO, RET TO CALLING PGM
014 237                /
014 237  315 244 014   OCTPR, CAL DCDNM    /DECODE 1ST OCTAL NUMBER
014 242  110           LCB                 /SAVE OCTAL NUMBER
014 243  034           INE                 /INCR INP BFR PNTR
014 244                /                    FALL THRU TO DECODE 2ND NMBR
014 244                /
014 244  041 150 000   DCDNM, LXH 150 000  /SET PNTR TO DIGIT TABLE
014 247  164           LMH                 /CLEAR TBL BY STORING 000.
014 250  054           INL
014 251  164           LMH
014 252  054           INL
014 253  164           LMH
014 254  315 321 014   LOOP, CAL FNUM      /CHECK FOR VALID NUMBER
014 257  372 303 014   JTS CKLNH           /IF NOT, CHECK CHAR CNT = 0
014 262  032           LDAD                /FETCH CHAR
014 263  346 007       NDI 007             /MASK OFF 260
014 265  041 150 000   LXH 150 000         /SET PNTR TO DIGIT TABLE
014 270  106           LBM                 /TABLE AT LOC 150 PG 00
014 271  167           LMA                 /AND SHIFT OTHER NUMBERS
014 272  054           INL                 /UP THRU THE TABLE
014 273  176           LAM
014 274  160           LMB
014 275  054           INL
014 276  167           LMA
014 277  034           INE                 /INCR INP BFR PNTR
014 300  303 254 014   JMP LOOP            /FETCH NXT NUMBER
014 303                /
014 303  072 165 000   CKLNH, LTA 165 000  /FETCH ORIG INP BFR PNTR
014 306  273           CPE                 /IS CHAR CNT = 0?
014 307  312 036 014   JTZ ERR             /YES, PRINT ERROR MSG
014 312  315 337 014   CAL OCT             /FETCH FINAL OCTAL NUMBER
014 315  362 036 014   JFS ERR             /IF INVALID, PRINT ERR MSG
014 320  311           RET                 /ELSE, RET TO CALLING PGM
014 321                /
014 321  032           FNUM, LDAD          /FETCH ASCII DIGIT
014 322  376 260       CPI 260             /VALID NUMBER?
014 324  370           RTS                 /NO, RET WITH S FLAG SET
014 325  326 270       SUI 270             /CHECK UPPER LIMIT BY
014 327  306 200       ADI 200             /SETTING S FLAG TO PROPER
014 331  311           RET                 /STATE AND RETURN
014 332                /
014 332  064           INCR, INM           /INCR CONTENTS OF MEM LOC
014 333  300           RFZ                 /IF NOT ZERO, RET
014 334  054           INL                 /PNT TO NXT LOC
014 335  064           INM                 /INCR 2ND HALF
014 336  311           RET                 /RET TO CALLING PGM
```

```
014 337                     /
014 337  056 152           OCT, LLI 152          /SET PNTR TO 3RD DIGIT
014 341  176               LAM
014 342  376 004           CPI 004               /IS 3RD DIGIT > 3?
014 344  360               RFS                   /YES, RET WITH S FLAG RESET
014 345  346 003           NDI 003               /CLEAR CARRY
014 347  017               RRC                   /POSITION DIGIT
014 350  017               RRC
014 351  107               LBA                   /SAVE IN REG B
014 352  055               DCL                   /DECR PNTR
014 353  176               LAM                   /FETCH NEXT DIGIT
014 354  007               RLC                   /POSITION DIGIT
014 355  007               RLC
014 356  007               RLC
014 357  200               ADB                   /ADD TO REG B
014 360  055               DCL                   /DECR PNTR
014 361  206               ADM
014 362  107               LBA                   /SAVE FINAL NUMBER
014 363  076 200           LAI 200               /SET S FLAG TO INDICATE
014 365  247               NDA                   /THAT THE NUMBER IS VALID
014 366  311               RET                   /RET TO CALLING PGM
014 367                     /
014 367  157               OCTOUT, LLA           /SAVE OCTAL NUMBER TO PRINT
014 370  007               RLC                   /POSITION HUNDRED'S DIGIT
014 371  007               RLC
014 372  346 003           NDI 003               /MASK OFF OTHER BITS
014 374  366 260           ORI 260               /FORM ASCII CODE
014 376  315 300 017       CAL PRINT             /PRINT DIGIT
015 001  175               LAL                   /FETCH OCTAL NUMBER
015 002  017               RRC                   /POSITION TEN'S DIGIT
015 003  017               RRC
015 004  017               RRC
015 005  346 007           NDI 007               /MASK OFF OTHER DIGITS
015 007  366 260           ORI 260               /FORM ASCII CODE
015 011  315 300 017       CAL PRINT             /PRINT DIGIT
015 014  175               LAL                   /FETCH OCTAL NUMBER
015 015  346 007           NDI 007               /MASK OFF OTHER DIGITS
015 017  366 260           ORI 260               /FORM ASCII CODE
015 021  303 300 017       JMP PRINT             /PRINT DIGIT AND RET
015 024                     /
015 024  076 272           COLON, LAI 272        /SET ASCII CODE FOR :
015 026  303 300 017       JMP PRINT             /PRINT COLON AND RET
015 031                     /
015 031  041 167 000       PRT166, LXH 167 000   /SET PNTR TO LO ADDR
015 034  176               LAM                   /FETCH PG ADDR
015 035  346 077           NDI 077
015 037  315 367 014       CAL OCTOUT            /PRINT PAGE ADDR
015 042  315 053 015       CAL SPAC              /PRINT A SPACE
015 045  056 166           LLI 166               /SET PNTR TO LO ADDR
015 047  176               LAM                   /FETCH LO ADDR
015 050  315 367 014       CAL OCTOUT            /PRINT LO ADDR
015 053                     /                     /FALL THRU TO PRINT SPACE
015 053                     /
015 053  076 240           SPAC, LAI 240         /SET ASCII CODE FOR SPACE
015 055  303 300 017       JMP PRINT             /PRINT SPACE AND RET
015 060                     /
015 060  041 171 000       CKEND, LXH 171 000    /SET PNTR HI ADDR
015 063  176               LAM                   /FETCH 2ND HALF
015 064  056 167           LLI 167               /SET PNTR TO 2ND HALF LO ADDR
015 066  276               CPM                   /2ND HALFS EQUAL?
```

```
015 067  302 102 015  JFZ CONT              /NO, CONTINUE PROCESS
015 072  054          INL
015 073  176          LAM                   /FETCH 1ST HALF HI ADDR
015 074  056 166      LLI 166               /SET PNTR TO 1ST HALF LO ADDR
015 076  276          CPM                   /IS 1ST HALFS EQUAL?
015 077  312 000 014  JTZ INCMD             /YES, RET TO CMND MODE
015 102  056 166      CONT, LLI 166         /NO, SET PNTR TO LO ADDR
015 104  303 332 014  JMP INCR
015 107               /
015 107  036 342      MODIFY, LEI 342       /SET INP BFR PNTR
015 111  315 157 014  CAL OCTNM             /FETCH ADDR TO MODIFY
015 114  315 053 015  CAL SPAC              /PRINT SPACE
015 117  315 226 015  MOD1, CAL MEMPRT      /PRINT CONTENTS OF MEM LOC
015 122  315 163 015  CAL INSPCL            /INP MODIFICATION
015 125  312 141 015  JTZ NXLOC             /NO, SET UP NXT LOC
015 130  137          LEA                   /YES, SAVE INP PNTR
015 131  315 244 014  CAL DCDNM             /CONVERT TO OCTAL NUMBER
015 134  170          LAB                   /SAVE OCTAL NUMBER
015 135  052 166 000  LHLD 166 000          /SET PNTR TO MEM LOC
015 140  167          LMA                   /LOAD MEM WITH NEW VALUE
015 141  041 166 000  NXLOC, LXH 166 000    /SET PNTR TO MEM ADDR STRAGE
015 144  315 332 014  CAL INCR              /INCR MEM ADDR
015 147  315 155 015  CAL MCONT             /PRINT NXT ADDR TO MODIFY
015 152  303 117 015  JMP MOD1
015 155               /
015 155  315 142 014  MCONT, CAL HDLN       /PRINT C/R, L/F
015 160  303 031 015  JMP PRT166            /PRINT ADDR TO MODIFY AND RET
015 163               /
015 163  315 024 015  INSPCL, CAL COLON     /PRINT COLON
015 166  021 340 000  LXD 340 000           /SET PNTR TO S.A. OF INP BFR
015 171  173          LAE
015 172  062 165 000  STA 165 000           /SAVE S.A. OF INP BFR
015 175  315 200 017  LPIN, CAL RCV         /INP CHAR
015 200  022          STAD                  /STORE CHAR IN INP BFR
015 201  376 240      CPI 240               /CHAR = SPACE?
015 203  312 222 015  JTZ LPO               /YES,
015 206  376 215      CPI 215               /NO, CHAR = C/R?
015 210  312 000 014  JTZ INCMD             /YES, RET TO COMMAND MODE
015 213  034          INE                   /NO, INCR INP BFR PNTR
015 214  312 036 014  JTZ ERR               /INP BFR FULL? YES, ERROR
015 217  303 175 015  JMP LPIN              /NO, INP NXT CHAR
015 222  076 340      LPO, LAI 340          /SET UP TEST FOR CHAR COUNT
015 224  273          CPE
015 225  311          RET                   /RET TO CALLING PGM
015 226               /
015 226  052 166 000  MEMPRT, LHLD 166 000  /SET PNTR TO MEM LOC
015 231  176          LAM                   /FETCH CURRENT MEM CONTENTS
015 232  303 367 014  JMP OCTOUT            /PRINT CONTENTS AND RET
015 235               /
015 235               /MEA - MEMORY DUMP
015 235               /
015 235  036 342      MDUMP, LEI 342        /SET PNTR TO INP BFR
015 237  315 157 014  CAL OCTNM             /FETCH MEM DUMP LIMITS
015 242  315 142 014  CAL HDLN              /PRINT C/R, L/F
015 245  315 155 015  MDMP1, CAL MCONT      /PRINT ADDR OF 1ST LOC
015 250  315 053 015  CAL SPAC              /PRINT SPACE
015 253  056 164      MDMP2, LLI 164        /SET PNTR TO TEMP STRAGE
015 255  066 020      LMI 020               /SAVE LOC PER LINE CNTR
015 257  315 226 015  OUTAGN, CAL MEMPRT    /PRINT MEM CONTENTS
```

```
015 262   315 060 015   CAL CKEND          /CHECK FOR LAST LOC PRTD
015 265   315 053 015   CAL SPAC           /PRINT SPACE
015 270   056 164       LLI 164            /SET PNTR TO L/L CNTR
015 272   065           DCM                /DECR CNTR. CNTR = 0?
015 273   312 245 015   JTZ MDMP1          /YES, START NEW LINE
015 276   303 257 015   JMP OUTAGN         /NO, PRINT MORE CONTENTS
015 301                 /
015 301   036 342       WRITE, LEI 342     /SET PNTR TO INP BFR
015 303   315 157 014   CAL OCTNM          /FETCH START AND END ADDR
015 306   052 170 000   LHLD 170 000
015 311   353           XCHG               /SET END ADDR
015 312   052 166 000   LHLD 166 000       /SET START ADDR
015 315   315 340 017   CAL PUNCH          /GO TO USER BULK WRITE RTN
015 320   303 000 014   JMP INCMD          /RET TO COMMAND MODE
015 323                 /
015 323   315 240 017   RDBULK, CAL READ   /GO TO USER BULK READ RTN
015 326   303 000 014   JMP INCMD          /RET TO COMMAND MODE
015 331                 /
015 331   315 364 015   BREAK, CAL ANLYZ   /SET UP ADDRESS OF BP
015 334   176           LAM                /SAVE ORIG CONTENTS OF BP
015 335   066 377       LMI 377            /INSERT BP RESTART INSTR.
015 337   353           XCHG               /SAVE BP ADDR
015 340   041 070 000   LXH 070 000        /SET PNTR TO RST 7 LOC
015 343   066 303       LMI 303            /STORE JUMP INSTR
015 345   054           INL
015 346   066 005       LMI 005            /**** STORE BRK LO ADDR
015 350   054           INL
015 351   066 016       LMI 016            /**** STORE BRK PG ADDR
015 353   054           INL
015 354   163           LME                /STORE BP ORIG LOW ADDR
015 355   054           INL
015 356   162           LMD                /STORE BP ORIG PG ADDR
015 357   054           INL
015 360   167           LMA                /STORE ORIG BP INSTRUCTION
015 361   303 000 014   JMP INCMD
015 364                 /
015 364   036 342       ANLYZ, LEI 342     /SET PNTR TO BUFF SA
015 366   315 157 014   CAL OCTNM          /FETCH ADDR INTO 166, 167
015 371   076 303       LAI 303            /SET JUMP INSTR FOR GOTO
015 373   062 155 000   STA 155 000
015 376   052 166 000   LHLD 166 000       /GET BP ADDR
016 001   042 156 000   SHLD 156 000       /STORE BP ADDR IN JUMP INSTR
016 004   311           RET
016 005                 /
016 005   042 206 000   BRK, SHLD 206 000  /SAVE H & L
016 010   365           PUSS               /SAVE STATUS & REG A
016 011   041 004 000   LXH 004 000        /SET UP TO SAVE SP
016 014   071           DADS               /MOVE SP TO REG'S H & L
016 015   042 176 000   SHLD 176 000       /SAVE ORIG SP
016 020   361           POPS               /RESTORE STATUS
016 021   061 206 000   LXS 206 000        /SET SP TO REG STRAGE
016 024   325           PUSD               /SAVE REG'S D & E
016 025   305           PUSB               /SAVE REG'S B & C
016 026   365           PUSS               /SAVE STATUS WORD & REG A
016 027   052 073 000   LHLD 073 000       /SET BP PNTR
016 032   072 075 000   LTA 075 000        /FETCH ORIG INSTR
016 035   167           LMA                /RESTORE ORIG BKPNT INSTR
016 036   303 000 014   JMP INCMD          /BACK TO MONITOR
016 041                 /
```

```
016 041   315 364 015   GOTO, CAL ANLYZ      /SET UP ADDR OF GOTO
016 044   061 176 000   LXS 176 000          /SET SP TO REG STRAGE
016 047   341           POPH                 /FETCH SP IN H & L
016 050   361           POPS                 /SET UP STATUS AND REG A
016 051   301           POPB                 /SET UP REG'S B & C
016 052   321           POPD                 /SET UP REG'S D & E
016 053   371           SPHL                 /SET UP SP
016 054   052 206 000   LHLD 206 000         /SET UP REG H & L
016 057   303 155 000   JMP 155 000          /START PROGRAM
016 062               /
016 062   041 341 000   XREG, LXH 341 000    /SET INP BFR PNTR
016 065   021 346 016   LXD RGTBL            /SET REG TABLE PNTR
016 070   315 332 016   CAL TBLCK            /SEARCH FOR REG DESIGNATED
016 073   376 306       CPI 306              /FLAG STATUS?
016 075   312 143 016   JTZ F                /YES, PRINT FLAGS
016 100   376 323       CPI 323              /STACK POINTER?
016 102   312 261 016   JTZ S                /YES, PRINT ADDRESS
016 105   056 164       LLI 164              /SET PNTR TO TEMP STRAGE
016 107   023           INXD                 /INCR REG TBL PNTR
016 110   032           LDAD                 /FETCH REG STRAGE PNTR
016 111   167           LMA                  /SAVE REG STRAGE PNTR
016 112   157           LLA                  /SET PNTR TO REG VALUE
016 113   315 053 015   CAL SPAC             /PRINT SPACE
016 116   176           LAM                  /FETCH CURRENT REG VALUE
016 117   315 367 014   CAL OCTOUT           /PRINT CURRENT REG VALUE
016 122   315 163 015   CAL INSPCL           /INP MODIFICATION
016 125   312 000 014   JTZ INCMD            /NO ENTRY, RET TO CMND
016 130   137           LEA                  /SAVE INP BFR PNTR
016 131   315 244 014   CAL DCDNM            /YES, DECODE OCTAL NUMBER
016 134   056 164       LLI 164              /SET PNTR TO TEMP STRAGE
016 136   156           LLM                  /FETCH REG TBL PNTR
016 137   160           LMB                  /STORE NEW REG VALUE
016 140   303 000 014   JMP INCMD            /RET TO COMMAND MODE
016 143               /
016 143   315 053 015   F, CAL SPAC          /PRINT SPACE
016 146   056 200       LLI 200              /SET REG TBL PNTR
016 150   176           LAM                  /FETCH FLAG WORD
016 151   021 371 016   LXD FTBL             /SET PNTR TO FLAG TBL
016 154   023           INXD                 /ADV PNTR TO BIT WORDS
016 155   107           LBA
016 156   032           PRTBIT, LDAD         /FETCH FLAG BIT
016 157   247           NDA                  /END OF TABLE?
016 160   312 202 016   JTZ INFLG            /YES, INP FLAG CHANGES
016 163   240           NDB                  /IS BIT SET?
016 164   312 175 016   JTZ TRYNX            /NO, TRY NEXT BIT
016 167   033           DCXD
016 170   032           LDAD                 /FETCH CHAR FOR FLAG
016 171   315 300 017   CAL PRINT            /PRINT CHAR FOR FLAG
016 174   023           INXD                 /RESET TBL PNTR
016 175   023           TRYNX, INXD          /ADV REG TBL PNTR
016 176   023           INXD
016 177   303 156 016   JMP PRTBIT
016 202   315 163 015   INFLG, CAL INSPCL    /INPUT CHANGES
016 205   312 000 014   JTZ INCMD            /NO ENTRY, RET TO CMND
016 210   157           LLA                  /SET INP BFR PNTR
016 211   006 002       LBI 002              /SET BASIC FLAG BYTE
016 213   176           LAM                  /FETCH FLAG CHAR FM INP
016 214   021 371 016   FLAG, LXD FTBL       /SET PNTR TO FLAG TBL
016 217   315 332 016   CAL TBLCK            /SEARCH TBL FOR FLAG CHAR
```

```
016 222  023          INXD              /FETCH BIT WORD
016 223  032          LDAD
016 224  260          ORB               /ADD BIT TO BASIC BYTE
016 225  107          LBA               /SAVE FLAG BYTE
016 226  054          INL
016 227  176          LAM               /FETCH NEXT CHAR FM INP
016 230  376 240       CPI 240           /CHAR = SPACE?
016 232  302 214 016   JFZ FLAG          /NO, MORE STATUS INP
016 235  170          LAB               /FETCH NEW STATUS WORD
016 236  346 100       NDI 100           /IS ZERO SET?
016 240  312 253 016   JTZ OK            /NO, WORD IS O.K.
016 243  170          LAB               /YES, CK S AND P FLAGS
016 244  346 204       NDI 204           /SEPARATE S AND P FM OTHERS
016 246  356 004       XRI 004           /S = 0? P = 1?
016 250  302 036 014   JFZ ERR           /NO, ILLEGAL SET UP
016 253  056 200       OK, LLI 200       /SET PNTR TO STATUS BYTE
016 255  160          LMB               /SAVE NEW BYTE
016 256  303 000 014   JMP INCMD         /RET TO COMMAND MODE
016 261               /
016 261  315 053 015   S, CAL SPAC       /PRINT A SPACE
016 264  041 177 000   LXH 177 000       /SET PNTR TO SP PG ADDR
016 267  176          LAM
016 270  315 367 014   CAL OCTOUT        /PRINT SP PG ADDR
016 273  315 053 015   CAL SPAC          /PRINT A SPACE
016 276  056 176       LLI 176           /SET PNTR TO SP LO ADDR
016 300  176          LAM
016 301  315 367 014   CAL OCTOUT        /PRINT SP LO ADDR
016 304  315 163 015   CAL INSPCL        /INP MODIFICATION
016 307  312 000 014   JTZ INCMD         /NO ENTRY, RET TO CMND
016 312  137          LEA
016 313  062 165 000   STA 165 000       /SAVE INP PNTR
016 316  315 237 014   CAL OCTPR         /CONVERT ADDR INP
016 321  041 176 000   LXH 176 000       /SET SP STRAGE
016 324  160          LMB               /STORE SP LO ADDR
016 325  054          INL
016 326  161          LMC               /STORE SP PG ADDR
016 327  303 000 014   JMP INCMD         /RET TO CMND MODE
016 332               /
016 332  032          TBLCK, LDAD       /FETCH CHAR
016 333  247          NDA               /END OF TABLE?
016 334  312 036 014   JTZ ERR           /YES, INVALID INPUT
016 337  276          CPM               /CHAR MATCH?
016 340  310          RTZ               /YES, RET TO CALLING PGM
016 341  023          INXD              /NO, ADV TBL PNTR
016 342  023          INXD
016 343  303 332 016   JMP TBLCK         /CONT. SEARCH
016 346               /
016 346  301          RGTBL, 301        /REG A CHAR
016 347  201          201               /REG A STRAGE
016 350  302          302               /REG B CHAR
016 351  203          203               /REG B STRAGE
016 352  303          303               /REG C CHAR
016 353  202          202               /REG C STRAGE
016 354  304          304               /REG D CHAR
016 355  205          205               /REG D STRAGE
016 356  305          305               /REG E CHAR
016 357  204          204               /REG E STRAGE
016 360  310          310               /REG H CHAR
016 361  207          207               /REG H STRAGE
```

```
016 362  314          314                 /REG L CHAR
016 363  206          206                 /REG L STRAGE
016 364  306          306                 /FLAG STATUS CHAR
016 365  200          200                 /FLAG STATUS STRAGE
016 366  323          323                 /STACK PNTR CHAR
016 367  176          176                 /STACK PNTR STRAGE
016 370  000          000
016 371               /
016 371  323          FTBL, 323           /SIGN CHAR
016 372  200          200                 /SIGN BIT
016 373  332          332                 /ZERO CHAR
016 374  100          100                 /ZERO BIT
016 375  301          301                 /AUXILIARY CARRY CHAR
016 376  020          020                 /AUXILIARY CARRY BIT
016 377  320          320                 /PARITY CHAR
017 000  004          004                 /PARITY BIT
017 001  303          303                 /CARRY CHAR
017 002  001          001                 /CARRY BIT
017 003  000          000
017 004  000          000
017 005               /
017 005  315 050 017  FILL, CAL ADRDTA     /INP ADDR AND DATA FM BFR
017 010  052 166 000  FL1, LHLD 166 000    /SET UP MEM PNTR
017 013  160          LMB                 /FILL MEM LOC WITH DATA
017 014  315 060 015  CAL CKEND            /DONE? YES, RET TO CMND MODE
017 017  303 010 017  JMP FL1              /NO, CONTINUE WITH FILL
017 022               /
017 022               /SEARCH ROUTINE
017 022               /
017 022  315 050 017  SEARCH, CAL ADRDTA   /INP ADDR AND DATA FM BFR
017 025  056 165      LLI 165             /SET PNTR TO SAVE DATA
017 027  160          LMB                 /SAVE SEARCH DATA IN MEM
017 030  056 165      SH1, LLI 165        /SET PNTR TO SRCH DATA
017 032  176          LAM                 /FETCH SEARCH DATA
017 033  052 166 000  LHLD 166 000        /SET PNTR TO MEM
017 036  276          CPM                 /DATA EQUAL SRCH DATA
017 037  314 155 015  CTZ MCONT           /YES, PRINT ADDR
017 042  315 060 015  CAL CKEND           /DONE? YES, RET TO CMND MODE
017 045  303 030 017  JMP SH1              /NO, CONTINUE SEARCH
017 050               /
017 050  036 342      ADRDTA, LEI 342     /SET PNTR TO ADDR INP
017 052  315 157 014  CAL OCTNM           /INP START AND END ADDR
017 055  034          INE                 /INCR TO DATA POSITION
017 056  303 244 014  JMP DCDNM           /FETCH DATA FM INP BFR
017 061               /
017 061               /TRANSFER ROUTINE
017 061               /
017 061  036 342      TRNSFR, LEI 342     /SET PNTR TO ADDR INP
017 063  315 157 014  CAL OCTNM           /FETCH 'FROM' ADDR
017 066  056 166      LLI 166             /SET PNTR TO ADDR INP
017 070  103          LBE                 /SAVE INP BFR PNT
017 071  021 172 000  LXD 172 000         /SAVE 'FROM' IN TEMP STRAGE
017 074  315 146 017  SVSA, CAL SWAP      /MOVE ADDR TO TEMP STRGE
017 077  076 172      LAI 172             /IS XFR COMPLETE?
017 101  275          CPL
017 102  302 074 017  JFZ SVSA            /NO, CONTINUE MOVE
017 105  004          INB
017 106  130          LEB                 /RESTORE INP BFR PNTR
017 107  315 157 014  CAL OCTNM           /INP 'TO' ADDR
017 112  056 172      LLI 172             /SET PNTR TO TEMP STRGE
```

```
017 114   021 166 000   LXD 166 000          /SET PNTR TO TEMP STRAGE
017 117   315 146 017   TF1, CAL SWAP        /XFR 'FROM' PNTR
017 122   076 176        LAI 176
017 124   275           CPL                  /XFR COMPLETE?
017 125   302 117 017   JFZ TF1              /NO, CONTINUE
017 130   130           LEB                  /FETCH 'TO' PNTR
017 131   121           LDC
017 132   052 166 000   TF2, LHLD 166 000    /SET 'FROM' PNTR
017 135   315 146 017   CAL SWAP             /SWAP MEM CONTENTS
017 140   315 060 015   CAL CKEND            /DONE? YES, RET TO CMND MODE
017 143   303 132 017   JMP TF2              /NO, CONTINUE XFR
017 146                 /
017 146   176           SWAP, LAM            /FETCH BYTE TO XFR
017 147   043           INXH                 /INCR 'FROM' PNTR
017 150   022           STAD                 /STORE BYTE IN NEW LOC
017 151   023           INXD                 /INCR 'TO' PNTR
017 152   311           RET
017 153                 /
017 200                 RCV,                 /USER DEFINED INPUT ROUTINE
                                             /FOR OPERATOR INPUT DEVICE
                        /
017 240                 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 MEMORU 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 ENTERED 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 ENTERING 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 MAY 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" (ILLEGAL 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" COMMAND IS ENTERED BY TYPING IN THE FOLLOWING COMMAND:

B 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 BREAKPOINT COMMAND SETS A POINT IN A PROGRAM BEING TESTED AT WHICH THE CONTENTS OF THE CPU REGISTERS, THE STACK POINTER AND THE FLAG STATUS ARE TO BE STORED FOR EXAMINATION BY THE PROGRAMMER. 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 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 "377" 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.

THE "GO TO" COMMAND

THE "GO TO" COMMAND IS INITIATED BY TYPING IN THE FOLLOWING COMMAND ENTRY:

G HHH LLL (CTRL/L)

WHERE "HHH LLL" REPRESENTS THE MEMORY ADDRESS AT WHICH PROGRAM OPERATION IS TO COMMENCE. THE CONTENTS OF THE CPU REGISTERS, STACK POINTER AND THE FLAG STATUS ARE SET UP WITH THE VALUES ENTERED BY THE OPERATOR USING THE "X" COMMAND OR WITH THE VALUES STORED AT THE TIME THE LAST BREAKPOINT WAS ENCOUNTERED BEFORE ACTUALLY JUMPING TO THE ADDRESS DESIGNATED IN THE COMMAND.

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)
XS (CTRL/L)
XF (CTRL/L)

WHERE THE LETTER FOLLOWING THE "X" INDICATES THE "VIRTUAL" CPU REGISTER TO BE DISPLAYED, OR THE STACK POINTER (S) OR THE FLAG STATUS (F). 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. FOR THE CPU REGISTERS, THE CONTENTS WILL BE DISPLAYED IN THE FOLLOWING FORMAT:

XR DDD:

WHERE "DDD" IS THE CURRENT VALUE STORED FOR THE CPU REGISTER INDICATED. 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.

FOR ANY OF THE COMMANDS LISTED ABOVE, IF THE OPERATOR SHOULD TYPE IN A MODIFICATION AND THEN DECIDE THAT IT IS NOT DESIRABLE TO MAKE THE CHANGE, THE OPERATOR MAY ENTER A "C/R" TO RETURN TO THE COMMAND MODE, IN WHICH CASE THE ORIGINAL VALUE WILL REMAIN UNCHANGED.

THE "XS" COMMAND CAUSES THE CURRENT CONTENTS STORED FOR THE STACK POINTER TO BE DISPLAYED IN THE FOLLOWING FORMAT:

XS HHH LLL:

WHERE "HHH LLL" INDICATE THE PAGE AND LOW ADDRESS PORTIONS, RESPECTIVELY, OF THE VIRTUAL STACK POINTER. IF IT IS NOT DESIRED TO CHANGE THIS VALUE, THE OPERATOR DEPRESSES THE SPACE BAR AND THE PROGRAM RETURNS TO THE COMMAND MODE.

IF THE OPERATOR DESIRES TO CHANGE THE CONTENTS OF THE VIRTUAL STACK POINTER, THE MODIFICATION MUST BE ENTERED IN THE FOLLOWING FORMAT AND TERMINATE IT BY ENTERING A SPACE.

XS HHH LLL:MMM,NNN

WHERE "MMM" IS THE PAGE PORTION AND "NNN" IS THE LOW ADDRESS PORTION OF THE ADDRESS. THE PAGE AND LOW ADDRESS MUST BE SEPARATED BY A COMMA WHEN ENTERING THIS MODIFICATION. IF A SPACE IS USED, THE VALUE STORED FOR THE STACK POINTER WILL MOST LIKELY NOT BE THE VALUE DESIRED.

THE "XF" COMMAND CAUSES THE STATUS OF THE CPU FLAGS, AS THEY WHERE

WHEN THE LAST BREAKPOINT WAS ENCOUNTERED, TO BE DISPLAYED IN THE FOLLOWING MANNER.

EACH FLAG IS ASSIGNED AN IDENTIFICATION LETTER, WHICH IS THE FIRST LETTER OF THE FLAG NAME. THIS ASSIGNMENT IS AS FOLLOWS:

S = SIGN FLAG
Z = ZERO FLAG
A = AUXILIARY CARRY FLAG
P = PARITY FLAG
C = CARRY FLAG

WHEN THE COMMAND "XF" IS ENTERED, EACH FLAG WHICH HAS A VALUE OF "1" STORED IN THE VIRTUAL STORAGE AREA WILL BE INDICATED BY THE CORRESPONDING LETTER BEING PRINTED ON THE DISPLAY DEVICE. FOR EXAMPLE, IF AT THE TIME THE LAST BREAKPOINT WAS ENCOUNTERED THE SIGN, PARITY AND CARRY FLAGS WERE "1" AND THE ZERO AND AUXILIARY CARRY FLAGS WERE "0" THE PROGRAM WOULD OUTPUT THE FOLLOWING:

XF SPC:

IF THE OPERATOR DESIRES TO SPECIFY CERTAIN FLAGS TO BE SET THE NEXT TIME A "GO TO" IS PERFORMED, THE LETTERS INDICATING WHICH FLAGS ARE TO HAVE A VALUE OF "1" SHOULD BE ENTERED AND THE INPUT TERMINATED BY A "SPACE" CHARACTER. ANY FLAG WHICH IS NOT ENTERED AT THIS TIME WILL BE SET TO A VALUE OF "0," REGARDLESS OF THE SETTING INDICATED BY THE PROGRAM. IN THE ABOVE EXAMPLE, IF IT IS DESIRED TO CHANGE THE SETTING TO HAVE THE ZERO, PARITY AND CARRY FLAGS SET TO "1" AND THE SIGN AND AUXILIARY CARRY SET TO "0" THE FOLLOWING ENTRY SHOULD BE MADE:

XF SPC:ZPC (SPACE)

THERE IS ONE RESTRICTION ON THE SETTING OF THE FLAG STATUS. IF THE ZERO FLAG IS "1," THE SIGN FLAG MUST BE "0" AND THE PARITY FLAG MUST BE "1." IF THE ENTRY DOES NOT FOLLOW THIS RESTRICTION, THE ENTRY WILL BE IGNORED AND THE ILLEGAL ENTRY ERROR MESSAGE WILL BE DISPLAYED.

IF IT IS NOT DESIRED TO CHANGE THE SETTING OF THE FLAGS, THE OPERATOR MAY SIMPLY ENTER A SPACE AND THE PROGRAM WILL RETURN TO THE COMMAND MODE.

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. FINALLY, BEFORE PUTTING THE PROGRAM IN "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 IN THE INITIAL PROGRAMMING.

THERE ARE SEVERAL IMPORTANT ADVANTAGES TO HAVING A PROGRAM SUCH AS THE MONITOR PROGRAM ON PROMS. FIRST, AS MENTIONED ABOVE, IT ALLOWS ON-LINE CAPABILITY SECONDS AFTER THE SYSTEM IS TURNED ON. IT ALSO PREVENTS A PROGRAM BEING DEBUGGED FROM "WIPING OUT" THE MONITOR PROGRAM, 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.

APPENDIX

MNEMONIC FOR THIS ASSEMBLER	POPULAR EQUIVALENTS	COMMENTS
LAA	MOV A,A	LOAD REG A TO A (MOVE REG A TO A)
LBA	MOV B,A	
LCA	MOV C,A	
.	.	
.	.	
LMA	MOV M,A	LOAD REG A TO MEMORY (MOVE A TO MEMORY)
.	.	
.	.	THE "LOAD" CLASS IS EQUIVALENT TO THE
.	.	"MOVE" CLASS FOR THE ENTIRE GROUP OF
LAM	MOV A,M	SIMILAR TYPE "LOAD"/"MOVE" INSTRUCTIONS
ADA	ADD A	ADD REG A TO THE ACCUMULATOR
ADM	ADD M	ADD MEMORY LOCATION TO THE ACCUMULATOR
ACA	ADC A	ADD (W/CARRY) REG A TO ACCUMULATOR
ACM	ADC M	ADD (W/CARRY) MEMORY LOC. TO ACC.
SUA	SUB A	SUBTRACT REG A FROM THE ACCUMULATOR
SBA	SBB A	SUBTRACT (W/BORROW) REG A FROM THE ACC.
NDA	ANA A	LOGICAL "AND" OPERATION REG A WITH ACC.
XRA	XRA A	LOGICAL "EXCLUSIVE OR" REG A WITH ACC.
ORA	ORA A	LOGICAL "OR" REG A WITH ACCUMULATOR
CPA	CMP A	COMPARE REGISTER A WITH ACCUMULATOR
CPB	CMP B	COMPARE REGISTER B WITH ACCUMULATOR
.	.	THE ENTIRE GROUP OF ARITHMETIC AND LOGICAL
.	.	INSTRUCTIONS BETWEEN THE ACCUMULATOR AND
.	.	OTHER CPU REGISTERS OR MEMORY LOCATIONS
CPM	CMP M	HAS THE SAME MNEMONIC FORMAT.
INA	INR A	INCREMENT THE ACCUMULATOR
DCM	DCR M	DECREMENT THE MEMORY LOCATION
ADI DDD	ADI DDD	ADD IMMEDIATE (NOTHING NEW HERE!)
ACI DDD	ACI DDD	ADD (W/CARRY) IMMEDIATE
SUI DDD	SUI DDD	SUBTRACT IMMEDIATE
SBI DDD	SUI DDD	SUBTRACT (W/BORROW)
NDI DDD	ANI DDD	LOGICAL "AND" IMMEDIATE
XRI DDD	XRI DDD	LOGICAL "EXCLUSIVE OR" IMMEDIATE
ORI DDD	ORI DDD	LOGICAL "OR" IMMEDIATE
CPI DDD	CPI DDD	COMPARE IMMEDIATE
LAI DDD	MVI A,DDD	LOAD REGISTER IMMEDIATE
INP PPP	IN PPP	INPUT PORT # "PPP" (RANGE 0 - 377 OCTAL)
OUT PPP	OUT PPP	OUTPUT PORT # "PPP" (RANGE 0 - 377 OCTAL)
HLT	HLT	CPU HALT INSTRUCTION
NOP	NOP	CPU "NO OPERATION" INSTRUCTION
DIN	DI	DISABLE INTERRUPT
EIN	EI	ENABLE INTERRUPT
RLC	RLC	ROTATE LEFT
RRC	RRC	ROTATE RIGHT
RAL	RAL	ROTATE LEFT (THROUGH CARRY)
RAR	RAR	ROTATE RIGHT (THROUGH CARRY)
LXB ADDR	LXI B,ADDR	LOAD REG PAIR B&C IMMEDIATE
LXD ADDR	LXI D,ADDR	LOAD REG PAIR D&E IMMEDIATE
LXH ADDR	LXI H,ADDR	LOAD REG PAIR H&L IMMEDIATE
LXS ADDR	LXI SP,ADDR	LOAD STACK POINTER IMMEDIATE
STA ADDR	STA ADDR	STORE ACCUMULATOR DIRECT
LTA ADDR	LDA ADDR	LOAD ACC DIRECT (NOTE THE DIFFERENCE!)

MNEMONIC FOR THIS ASSEMBLER	POPULAR EQUIVALENTS	COMMENTS
DAA	DAA	DECIMAL ADJUST ACCUMULATOR
CMA	CMA	COMPLEMENT THE ACCUMULATOR
STC	STC	SET THE CARRY
CMC	CMC	COMPLEMENT THE CARRY
RST X	RST X	RESTART INSTRUCTION, X = RESTART VECTOR
POPB	POP B	POP PAIR FROM STACK TO REG PAIR B&C
POPD	POP D	POP PAIR FROM STACK TO REG PAIR D&E
POPH	POP H	POP PAIR FROM STACK TO REG PAIR H&L
POPS	POP PSW	RESTORE CONDITION FLAGS
PUSB	PUSH B	PUSH REG PAIR B&C TO STACK
PUSD	PUSH D	PUSH REG PAIR D&E TO STACK
PUSH	PUSH H	PUSH REG PAIR H&L TO STACK
PUSS	PUSH PSW	PUSH CONDITION OF FLAGS TO STACK
XCHG	XCHG	EXCHANGE REG PAIRS H&L WITH D&E
XTHL	XTHL	EXCHANGE REG PAIR H&L WITH STACK
SPHL	SPHL	LOAD THE STACK POINTER FROM H&L
PCHL	PCHL	LOAD THE PROGRAM COUNTER FROM H&L
DADB	DAD B	DOUBLE ADD BETWEEN REG PAIRS B&C AND H&L
DADD	DAD D	DOUBLE ADD BETWEEN REG PAIRS D&E AND H&L
DADH	DAD H	DOUBLE ADD H&L WITH ITSELF
DADS	DAD SP	DOUBLE ADD BETWEEN H&L AND STACK
STAB	STAX B	STORE ACC AT ADDRESS IN B&C
STAD	STAX D	STORE ACC AT ADDRESS IN D&E
LDAB	LDAX B	LOAD ACC FROM ADDRESS IN B&C
LDAD	LDAX D	LOAD ACC FROM ADDRESS IN D&E
INXB	INX B	INCREMENT THE B&C REGISTER PAIR
INXD	INX D	INCREMENT THE D&E REGISTER PAIR
INXH	INX H	INCREMENT THE H&L REGISTER PAIR
INXS	INX SP	INCREMENT THE STACK POINTER
DCXB	DCX B	DECREMENT THE B&C REGISTER PAIR
DCXD	DCX D	DECREMENT THE D&E REGISTER PAIR
DCXH	DCX H	DECREMENT THE H&L REGISTER PAIR
DCXS	DCX SP	DECREMENT THE STACK POINTER
SHLD ADDR	SHLD ADDR	STORE H&L AT THE INDICATED ADDRESS
LHLD ADDR	LHLD ADDR	LOAD H&L FROM THE INDICATED ADDRESS
RTZ	RZ	RETURN IF ZERO FLAG IS TRUE
RFZ	RNZ	RETURN IF ZERO FLAG IS FALSE
RTC	RC	RETURN IF CARRY FLAG IS TRUE
RFC	RNC	RETURN IF CARRY FLAG IS FALSE
RTP	RPE	RETURN IF PARITY FLAG IS TRUE (EVEN PARITY)
RFP	RPO	RETURN IF PARITY FLAG IS FALSE (ODD PARITY)
RTS	RM	RETURN IF SIGN FLAG IS TRUE (MINUS VALUE)
RFS	RP	RETURN IF SIGN FLAG IS FALSE (PLUS VALUE)
RET	RET	UNCONDITIONAL RETURN
CAL ADDR	CALL ADDR	CALL SUBROUTINE AT ADDRESS
JMP ADDR	JMP ADDR	JUMP TO ROUTINE AT ADDRESS
CTZ	CZ	CONDITIONAL CALLS SAME FORMAT AS CONDI-
CFZ	CNZ	TIONAL RETURN COMMANDS SHOWN ABOVE
JTZ	JZ	CONDITIONAL JUMPS SAME FORMAT AS CONDI-
JFZ	JNZ	TIONAL RETURN COMMANDS SHOWN ABOVE

PUBLICATIONS FROM SCELBI COMPUTER CONSULTING, INC.

MACHINE LANGUAGE PROGRAMMING FOR THE '8008'
(AND SIMILAR MICROCOMPUTERS) $19.95

ASSEMBLER PROGRAMS FOR THE '8008' $17.95

AN '8008' EDITOR PROGRAM $14.95

'8008' MONITOR ROUTINES $11.95

AN '8080' ASSEMBLER PROGRAM $17.95

AN '8080' EDITOR PROGRAM $14.95

'8080' MONITOR ROUTINES $11.95

THE ABOVE PUBLICATIONS MAY BE ORDERED DIRECTLY FROM:

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