ECE 441

Microprocessors

Instructor: Dr. Jafar Saniie

Teaching Assistant: Thomas Gonnot

Final Project Report:

MONITOR PROJECT

05/06/2013

By: Subash Luitel

Acknowledgment: I acknowledge all of the work including figures and codes belong to me and/or persons who are referenced.

_____

Table of Contents

Abstract

The main goal of the project was to design resident monitor software similar to the monitor program utilized by the SANPER-1 lab unit. This software program is meant to provide the user with a variety of commands that run on the MC68000. This will provide the user with a suitable method of testing, debugging and for implementing some hardware designs.

1 Introduction

Objective:

Design monitor program with a variety of commands namely;

- Command Interpreter
- Command programs
- Debugger programs
- Help

1.1 Problem:

The main problem is to create a way to allow users to operate hardware designs which utilize the MC68K software functionalities for operation. This problem is address by implementing the commands for the monitor program mentioned above.

1.2 Design Methodology:

Firstly, the command interpreter code is written which allows for the software program to check if a valid command has been entered. If a valid command was not entered, an error is displayed to the screen together with information requesting the user to use the HELP command allowing the user to understand how to use the instructions.

Next, once a valid command is entered, the program jumps to the code used to execute that command and if any errors were present in the syntax, an error is displayed on the screen.

Consider a case when the user attempts to divide a number by zero. Since there is no definite result for this operation, an exception handling mechanism is utilized to provide better security. When such an error occurs, the program jumps to the exception handler and displays the contents of all registers to the screen before returning back to the program. The same idea is utilized for other debugger errors mentioned above.

1.3 Technology Used:

- EASY68K software tool
- ASM
- SIM

2 Monitor Program

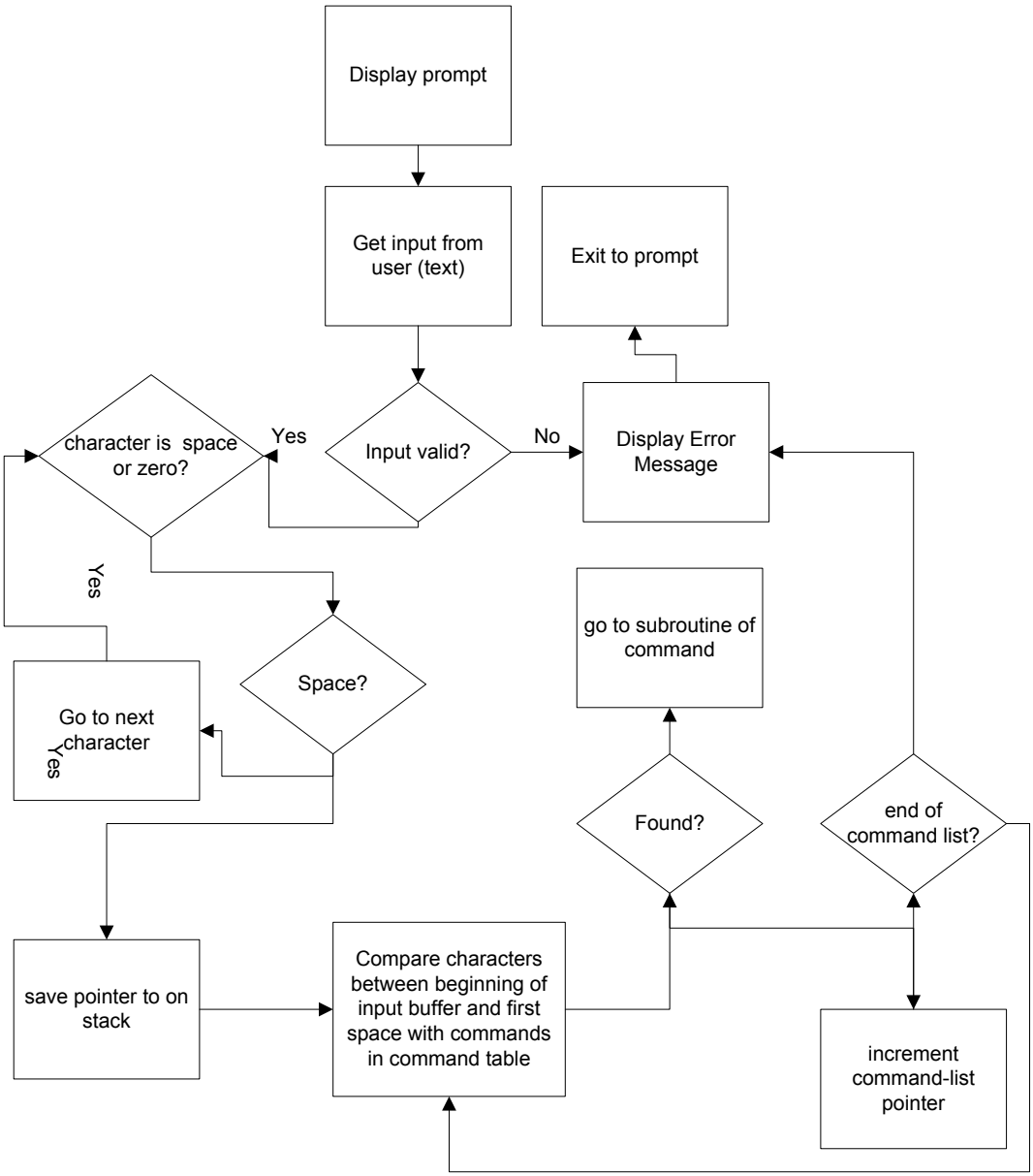
The monitor program provides the user a variety of software commands for changing memory, testing memory, debugger commands and even a help command used to provide the user with information on how to execute a certain function.

2.1 Command Interpreter

The command interpreter is a routine or section of the code used to check if the command entered was a valid command or not. If the command was valid, program flow proceeds to appropriate routine but if the command was not valid, an error message is displayed

2.1.1 Algorithm and flowchart for command interpreter

Display prompt
Get text from command line
Compare first character with space or zero
If space save pointer to on stack else get next character
Loop back to compare character
Compare characters between beginning of buff and first space with commands in command table
If found go to subroutine, else increment command-list pointer
If at end of command list, display error, else loop back to compare



2.1.2 Command Interpreter Assembly Code

```
* INTERPRETER
*-------------------------------------------------------------
INTPRT   MOVE.L   #$20,D1
         CLR.L    D2
         MOVE.L   #$41,D3
         MOVE.L   #$5A,D4
         CMP.B    (A1),D3
         BGT      UFAIL
         CMP.B    (A1),D4
         BLT      UFAIL
NXTCHI   CMP.B    (A1),D1
         BEQ      CHKCM
         ADDA     #1,A1
         CMP.B    #4,D2
```

```
          BGE       CHKCM
          ADDI      #1,D2
          BRA       NXTCHI
CHKCM     LEA       COM_TABL,A2
          LEA       INBUF,A3
          CLR.L     D3
          MOVE.L    (A3),D5
NXTCMD    CMPI.B    #NUMCMD,D3
          BGT       UFAIL
          MOVE.L    (A2),D6
          CMP.L     D5,D6
          BEQ       RUNCMD
          ADDI      #1,D3
          ADDA      #6,A2
          BRA       NXTCMD
RUNCMD    MOVEM.L   D5-D6,-(SP)       ; ARGUMENT PARSER STARTS HERE
          CLR.L     D5
          MOVE.L    #$20,D6
          LEA       EINBF,A3
GETSPC    CMPA      A1,A3
          BLE       TDONE
          CMP.B     (A1)+,D6
          BNE       GETSPC
          MOVE.L    A1,D7
          JSR       SKPUSH
          ADD       #1,D5
          BRA       GETSPC
TDONE     MOVE      D5,D7
          JSR       SKPUSH
          LEA       COM_ADD,A4
          MULU      #2,D3             ;GET DISPLACEMENT WITHIN TRANSLATION TABLE
          ADDA      D3,A4
          MOVEA     (A4),A5
          JMP       (A5)
          RTS
```

2.2 Debugger Commands

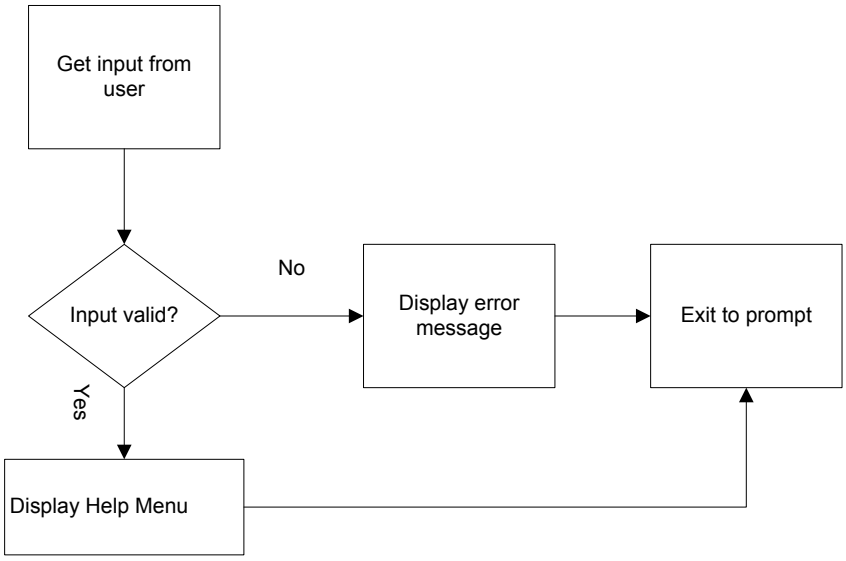
The debugger commands allow the user to actually perform different operations like memory sort, memory content changes, memory display etc. They thus provide the user with a great tool for debugging and testing.

2.2.1 HELP

Help displays all the debugging commands and format of usage with an example.

2.2.1.1 Help Algorithm and flowchart

If input is invalid:
Display Error
Exit to prompt
Display Help Menu
Exit to prompt



2.2.1.2 Debugger Command #1 Assembly Code

```
*SUBROUTINE TO DISPLAY HELP MENU
*-------------------------------------------------------------
MHELP    MOVEM.L  A1,-(SP)
         LEA      EHMNU,A3
         LEA      HLPMNU,A1
LPMHP    CMPA     A1,A3
         BLE      HMDNE
         JSR      OTPTCR
         ADDA     #$4A,A1
         BRA      LPMHP
HMDNE    MOVEM.L  (SP)+,A1
         RTS
```

Figure 2.7 Help Assembly Code

2.2.2 MDSP (Memory Display)

The command displays the contents of the specified memory address

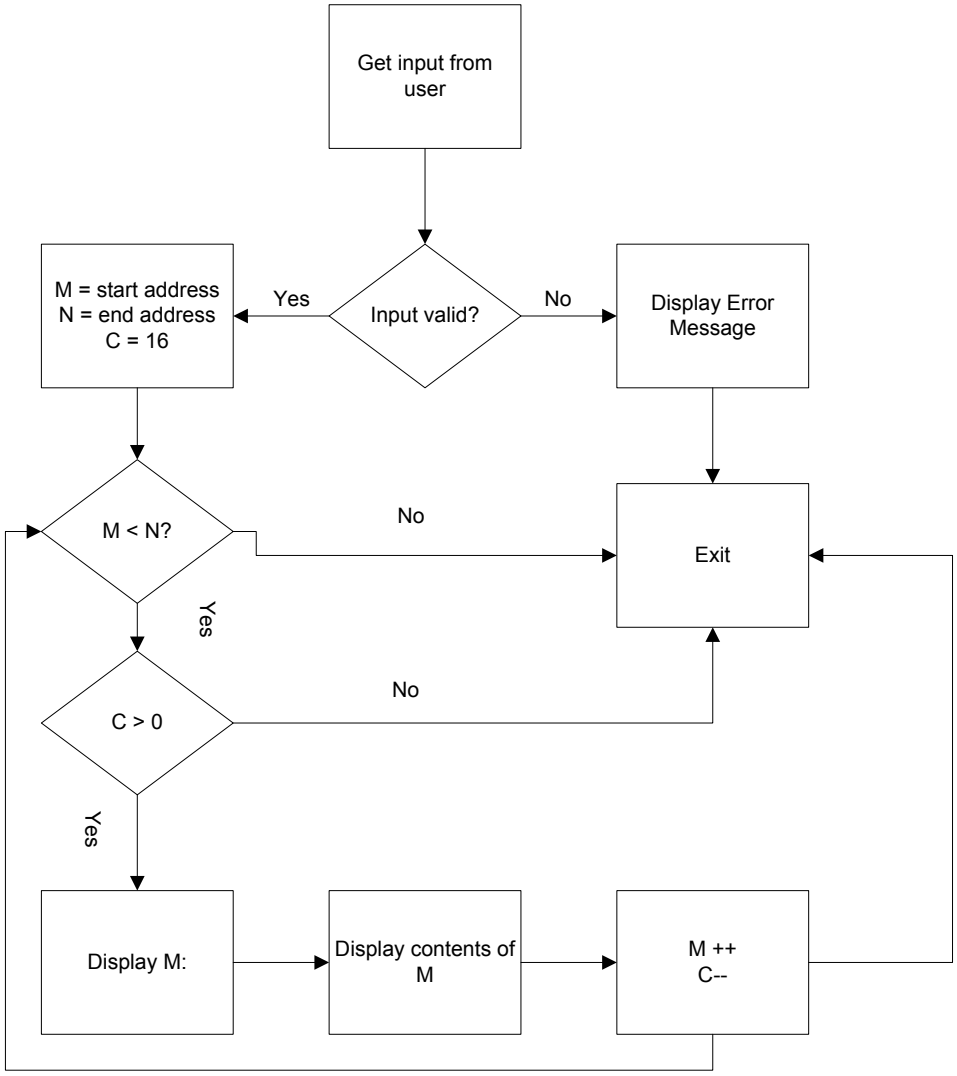
2.2.2.1 MDSP Algorithm and Flowchart

Get input from User
If input is invalid:
    Display Error
    Exit to prompt

M = start address
N = end address
While M<N:
    C= 16
    Display M
    While C>0:
        Display[M]

M++

Exit to prompt



2.2.2.1 MDSP assembly code

```
* MEMORY ADDRESS DISPLAY
* MEMORY ADDRESS AT A3
*-------------------------------------------------------------
MMADSP   MOVEM.L  D1-D2/A1-A3,-(SP) ;SAVE REGISTERS
         JSR      CLRBUF            ;CLEAR BUFFER
         LEA      INBUF,A1 ;INITIALIZE INBUF BUFFER ADDRESS
         MOVE.W   #$3030,(A1)       ;MOVE ASCII 00 INTO BUFFER
         ADDA     #2,A1             ;INCREMEENT MEMORY
         MOVE     A3,D1             ;COPY WORD INTO D1
         JSR      HX2ASC            ;CONVERT FROM HEX TO ASCII
         MOVE.L   D2,(A1)           ;MOVE CONVERTED VALUE INTO BUFFER
         ADDA     #4,A1             ;INCREMENT BUFFER ADDRESS TO NEXT FREE SPAC
```

```
          MOVE.W    #$3A20,(A1)        ;MOVE SEMICOLON : INTO BUFFER
          LEA       INBUF,A1 ;STORE BEGINNING OF BUFFER IN A1
          JSR       OTPTNC             ;OUTPUT CONTENTS OF BUFFER
          JSR       CLRBUF             ;CLEAR BUFFER
CLRDE     MOVEM.L (SP)+,D1-D2/A1-A3 ;RESTORE REGISTERS
          RTS                          ;RETURN FROM SUBROUTINE
```

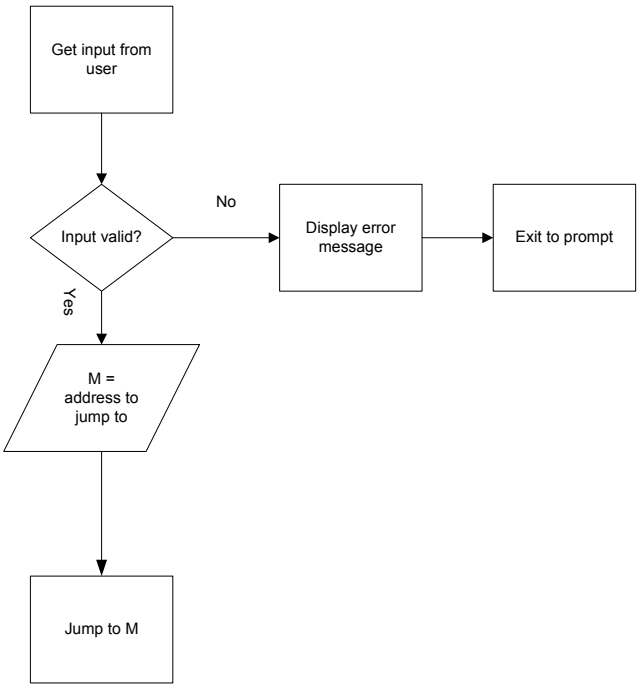
2.2.3 BKFL (Block Fill)

The command is used to fill a block with a word size value

2.2.3.1 BKFL Algorithm and Flowchart

Get input from User
If input is invalid:
    Display Error
    Exit to prompt

M = start address
N = end address
O= value to fill with
While N>M:
    [M] = [O]
    M++
Exit



2.2.3.2 BKFL Assembly Code

```
* BLOCK FILL
* TAKES TWO ADDRESSES AND A WORD AND FILLS ALL MEMORY IN RANGE
* WITH THAT WORD
* USING ADDRESS REGISTERS: START ADRESS AT A2, END ADDRESS AT
* A5, DATA IN REGISTER D4
```

```
*-----------------------------------------------------------
BLKFIL   MOVEM.L  A1,-(SP)
         LEA      INBUF,A1 ;INITILIZE BUFFER
         MOVE.B   D4,D3             ;TAKES BYTE ENTERED AND COPIES IT
         ROL.W    #8,D4             ;SO THAT IT FILLS UP A WORD
         MOVE.B   D3,D4             ;IN D4
         MOVE.W   D4,D1
         JSR      HX2ASC
BFDMA    MOVE     #$08,D5
         JSR      MMADSP
BKFLP    MOVE.W   D4,(A3)+
         MOVE.L   D2,(A1)
         JSR      OTPTNC
         JSR      DSPACE
         CMPA     A3,A5
         BLE      BFEXT
         SUBI     #1,D5
         BLE      BFNLIN
         BRA      BKFLP
BFNLIN   JSR      DNLINE
         BRA      BFDMA
BFEXT    JSR      DNLINE
         MOVEM.L  (SP)+,A1
         RTS
```

2.2.4 GOTO (Go to memory address)

The command is used to run the program from a specified address.

2.2.4.1 GOTO Algorithm and Flowchart

Get input from User
If input is invalid:
    Display Error
    Exit to prompt
M = address to goto

2.2.4.2 GOTO Assembly Code

```
GOTO     MOVE.W   D6,A0 ;MOVE ADDRESS TO REGISTER A0
         JMP  (A0)      ;JUMP TO MEMORY ADDRESS IN A0
```

2.2.5 MCHG

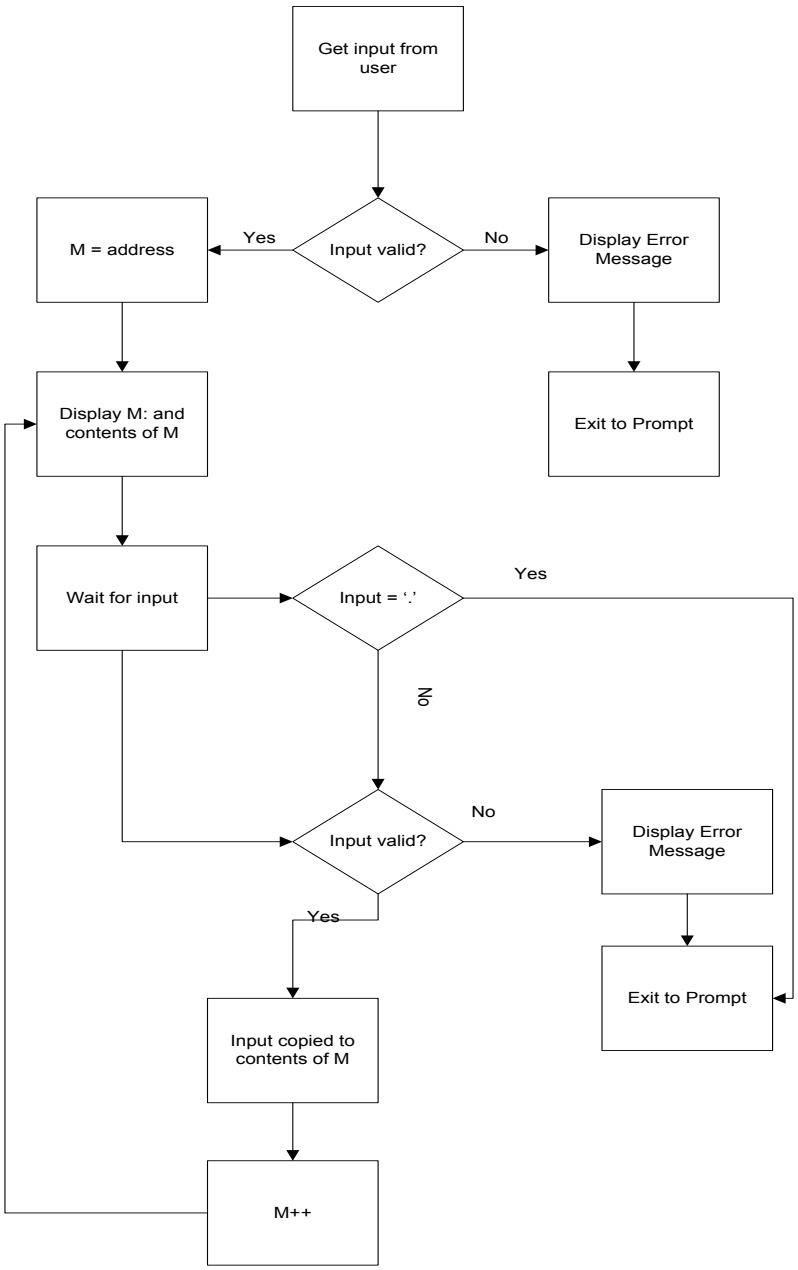
The command is used to modify the contents of memory location

2.2.5.1 MCHG Algorithm Flowchart

Get input from User
If input is invalid:
    Display Error
    Exit to prompt

M = start address

While input not equal to ".":
Display M
Wait for input
If input is not valid:
Display Error
[M] = input
M++
Exit to prompt



2.2.5.2 MCHG Code

```
* MEM MODIFY
* START ADDRESS IN A4, END ADDRESS IN A5
*-----------------------------------------------------------
```

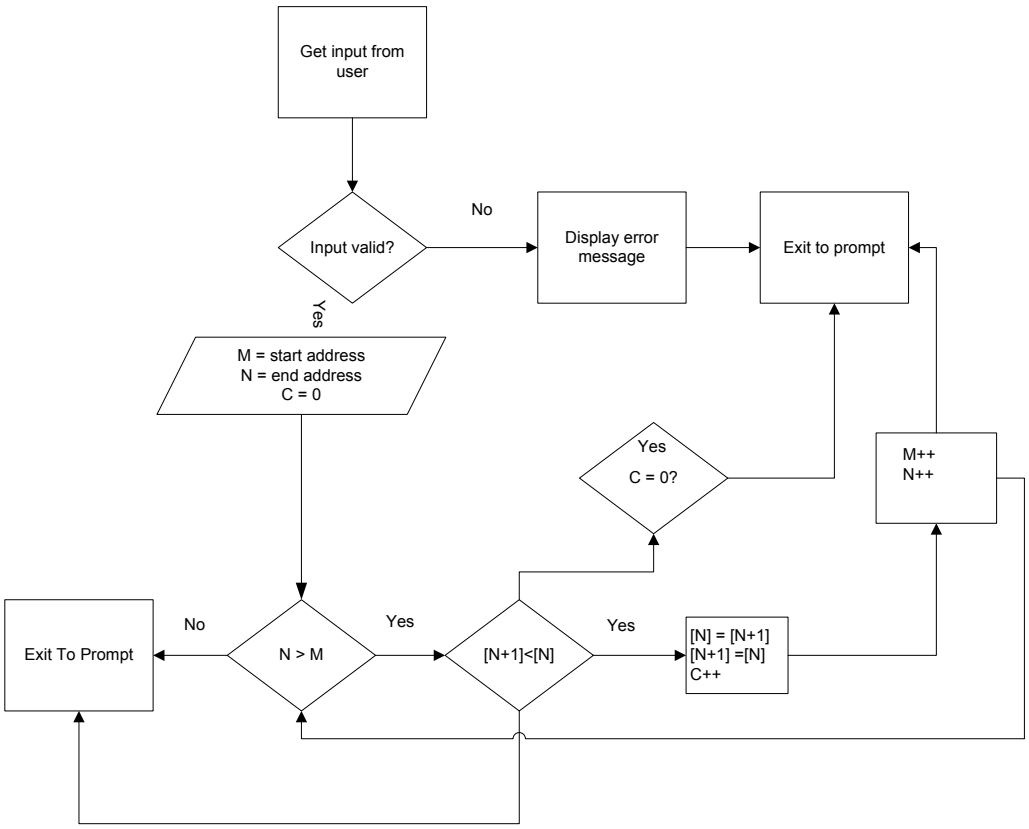
```
MEMMOD   MOVEM.L  D1/D2/D6,-(SP)
         LEA      INBUF,A1
         JSR      CLRBUF
NXTVAL   JSR      MMADSP            ;DISPLAY CURRENT ADDRESS
         MOVE.L   (A3),D1           ;MOVE CONTENTS OF ADDRESS TO D1 FOR DISPLAY
         JSR      HX2ASC            ;DISPLAY CONTENTS OF ADDRESS
         MOVE.L   D2,(A1)
         JSR      OTPTNC            ;OUTPUT THE CONTENTS OF A1
         JSR      DQNMRK            ;OUTPUT QUESTION MARK
         JSR      INPUT             ;TAKE INPUT FROM USER
         CMPI.B   #$2E,(A1)
         BEQ      MMDONE
         CMPI.B   #$30,(A1)
         BLT      UFAIL             ;ERROR MESSAGE SUBROUTINE
         CMPI.B   #$41,(A1)
         BLT      CHKNXT
         CMPI.B   #$46,(A1)
         BGT      UFAIL             ;ERROR MESSAGE SUBROUTINE
         BRA      CONTNU
CHKNXT   CMPI.B   #$39,(A1)
         BGT      UFAIL
CONTNU   MOVE.L   (A1),D2
         JSR      ASTBIN
         MOVE.W   D6,(A3)
         CLR.L    D6
         ADDA     #2,A3
         BRA      NXTVAL
         LEA      INBUF,A1
MMDONE   MOVEM.L  (SP)+,D1/D2/D6
       RTS
```

2.2.6 SRTW (Sort memory block)

The command is used to fill a block of memory

2.2.6.1 SRTW Algorithm and Flowchart

If input is invalid:
Display Error
Exit to prompt
M = start address
N = end address
C = 0
While N>M:
If [N+1] < [N]:
[temp] = [N]
[N] = [N+1]
[N+1] =[N]
C++
If C==0:
Exit to prompt
M++
N++



2.2.6.2 SRTW Assembly Code

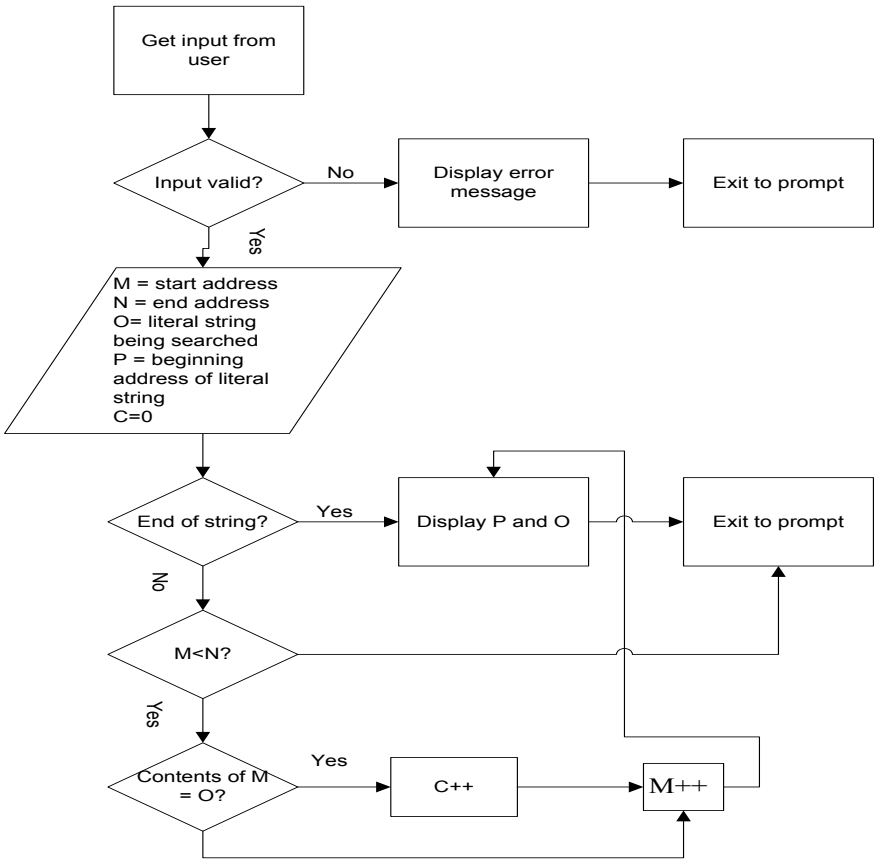
```
* SORTW
* START ADDRESS IN A3, END ADDRESS IN A2
* USES BUBBLE SORT
* A4 WILL CONTAIN A COPY OF THE START ADDRESS
*-----------------------------------------------------------
SORTW    MOVEM.L  D2-D4/A3,-(SP)
         MOVEA.L  A3,A4
RESRT    CLR.L    D2
         MOVEA.L  A4,A3
NEXTN    MOVE.B   (A3)+,D3
         MOVE.B   (A3),D4
         CMP.W    D3,D4
         BGT      CHKEND
         CMP.W    D3,D4
         BEQ      CHKEND
         ADDI     #1,D2
         SUBA     #1,A3
         MOVE.B   D4,(A3)+
         MOVE.B   D3,(A3)
CHKEND   CMPA     A5,A3
         BGE      CHKSWP
         BRA      NEXTN
CHKSWP   CMPI     #0,D2
         BGT      RESRT
         MOVEM.L  (SP)+,D2-D4/A3
         RTS
```

2.2.7 BKSH (Block Search)

Searches for a string literal in a block of memory. When literal is found, the address and string are displayed to the user

2.2.7.1 BKSH Algorithm and Flowchart

```
Get input from User
If input is invalid:
        Display Error
        Exit to prompt

M = start address
N = end address
O= address of string literal to look for
C=0
D=0
While not at end of string:
        C++
While N>M:
        If N==M:
                C++
                If D=C:
                        Display message "String found"
                        Exit
Display message "String not found"
Exit
```



2.2.7.2 BKSH Assembly Code

```
* BLKSCH
* START ADDRESS IS A3, END ADDRESS IS IN A4
* MAKE SURE THAT YOUR INPUT BUFFER DOES NOT LIE IN THE
* SEARCH ADDRESS SPACE OTHERWISE YOU WILL GET A FALSE POSITIVE
* IF STRING IS FOUND, IT AND IT LOCATION WILL BE DISPLAYED
* IF NOT, THE SUBROUTINE WILL SIMPLY EXIT WITH NO CONFIRMATION
* GIVEN TO THE USER
*-------------------------------------------------------------
BLKSCH   MOVEM.L  D2-D4/A0,-(SP)
         CLR.L    D2
         CLR.L    D3
         ADDA     #1,A0
         MOVEA    A0,A2
GETSRG   CMPI.B   #$22,(A2)+          ;CHECK IS WE HAVE REACHED THE END OF THE STRING
THAT WAS INPUT
         BEQ      SEARCH              ;IF WE HAVE REACHED THE END OF THE STRING, WE GO
TO SEARCH
         ADDI.B   #1,D2               ;INCREMENT THE CHARACTER COUNTER
         BRA      GETSRG              ;GET THE NEXT VALID CHRACTER ENTERED
SEARCH   MOVEA    A0,A2               ;RESET A2 TO ITS INITIAL POINT. WE ARE STARTING
OVER
         MOVE.L   D2,D4               ;COPY THE COUNTER VALUE FOR RESET
GETNXT   CMPA     A3,A4               ;CHECK IF WE'VE REACHED THE END OF SEARCHABLE
MEMORY
```

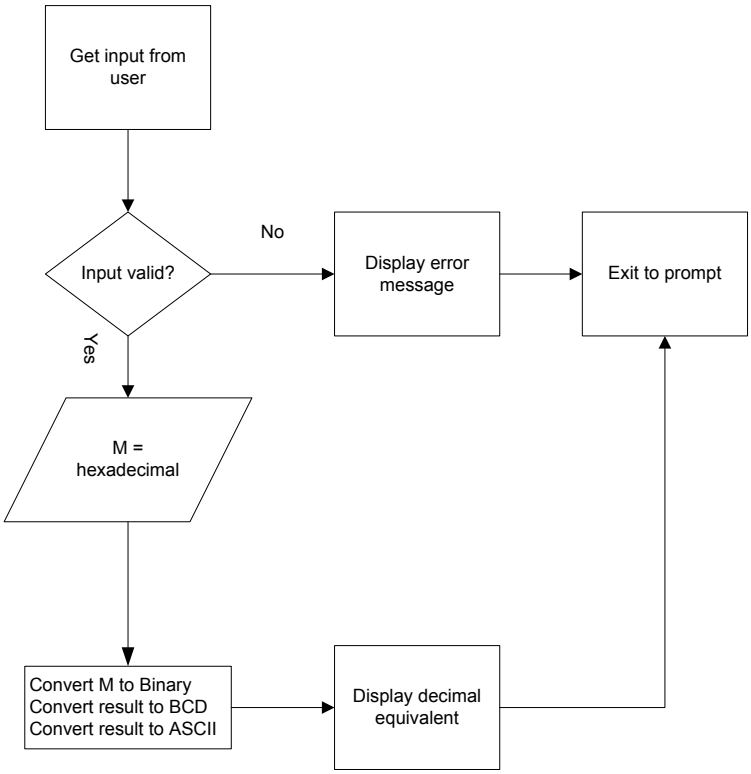
```
          BLE       SCHDNE              ;IF WE HAVE REACHED THE END THEN, RETURN FORM
SUBROUTINE
          MOVE.B    (A3)+,D3            ;MOVE THE BYTE WE WISH TO MATCH INTO D3
          CMP.B     (A2),D3             ;CHECK IF IT MATCHES WITH ONE OF THE CHARACTERS
ENTERED
          BNE       SEARCH              ;IF NOT RESET THE COUNTER
          SUBI.B    #1,D4               ;IF IT DOES, DECREMENT THE COUNTER
          BLE       DONESR              ;IF COUNTER IS ZERO, WE ARE DONE
          ADDA      #1,A2               ;OTHERWISE, INCREMENT A2 TO THE NEXT BYTE
          BRA       GETNXT              ;GET THE NEXT CHARACTER TO MATCH
DONESR    MOVE.B    #$00,(A3)           ;WE TERMINATE THE STRING FOUND WITH
          SUBA.L    D2,A3               ;SET A3 TO POINT TO THE START OF THE STRING THE E
MATCHED
          MOVEA     A3,A2               ;COPY A3 TO A2 AS PER THE MMADSP REQUIREMENTS
          JSR       MMADSP              ;CALL MMADSP TO DSIPLAY THE ADDRESS AT A2/A3
          MOVEA     A2,A1               ;COPY THE LOCATION OF THE ADDRESS OF STRING FOUND
TO A1
          JSR       OTPTCR              ;DISPLAY THE CONTENTS OF THAT ADDRESS I.E DISPLAY
THE STRING FOUND
SCHDNE    MOVEM.L   (SP)+,D2-D4/A0      ;RESTORE THE RESGISTER CONTENTS
          RTS
```

2.2.8 HXDC (Hexadecimal to Decimal)

Convert a hexadecimal number to a decimal number

2.2.8.1 HXDC Algorithm and Flowchart

Get input from User
If input is invalid:
    Display Error
    Exit to prompt
M = Hexadecimal Value
Convert M to Binary
Convert result to BCD
Convert result to ASCII
Display Result to User
Exit to prompt



2.2.8.2 HXDC Assembly Code

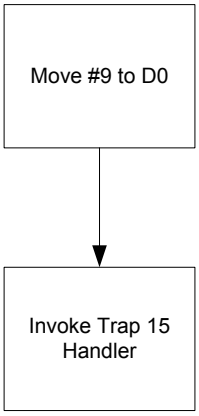
```
* HEX TO DECIMAL
* HEX VALUE READ FROM INPUT BUFFER, RESULT IN D5
*-------------------------------------------------------------
HX2DEC   JSR      ASTBIN   ; ASCII READ FROM D2 PLACED IN D6
         MOVE.L   D6,D1    ; BIN VALUE MOVED FROM D6 TO D1
         JSR      BINTBCD  ; BIN VALUE IN D1, BCD IN D4
         JSR      DECASC   ; BCD VALUE IN D4, ASCII IN D5
              RTS
```

2.2.9 EXIT

The command is used to exit the monitor program

2.2.9.1 EXIT Algorithm and Flowchart

Get input from User
If input is invalid:
Display Error
Exit to prompt
Exit program



2.2.9.2 EXIT Assembly Code

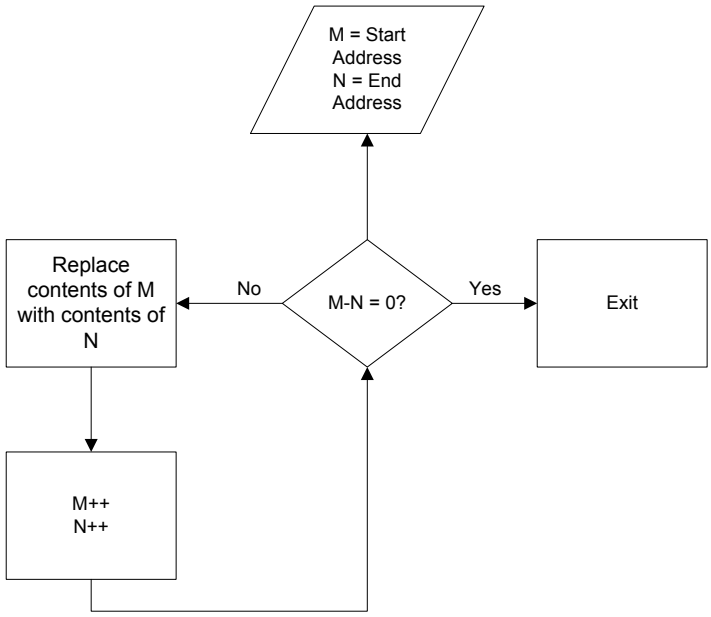
```
* EXIT SUBROUTINE
*-------------------------------------------------------------
MEXIT     MOVE        #9,D0
          TRAP        #15
```

2.2.10 BKMV (Block Move)

Used to move a block of memory to another location

2.2.10.1 BKMV Algorithm and Flowchart

M = start address
N = end address
While N>M:
[N] = [M]
M++
N++
Exit



2.2.10.2 BKMV Assembly Code

```
* BLKMOV
* START ADDRESS IS A2, END ADDRESS IS A3
* NEW BLOCK ADDRESS IS IN A4
*-------------------------------------------------------------
BLKMOV   MOVEM.L  A2-A3,-(SP)       ;SAVE REGISTERS
MVBYT    CMPA     A2,A3             ;CHECK IF WE ARE AT END OF MEMORY
         BLE      MVDONE            ;IF SO, EXIT
         MOVE.B   (A2)+,(A4)+       ;MOVE NEXT BYTE
         BRA      MVBYT             ;LOOP BACK
MVDONE   MOVEM.L  (SP)+,A2-A3       ;RESTORE REGISTERS         RTS
```

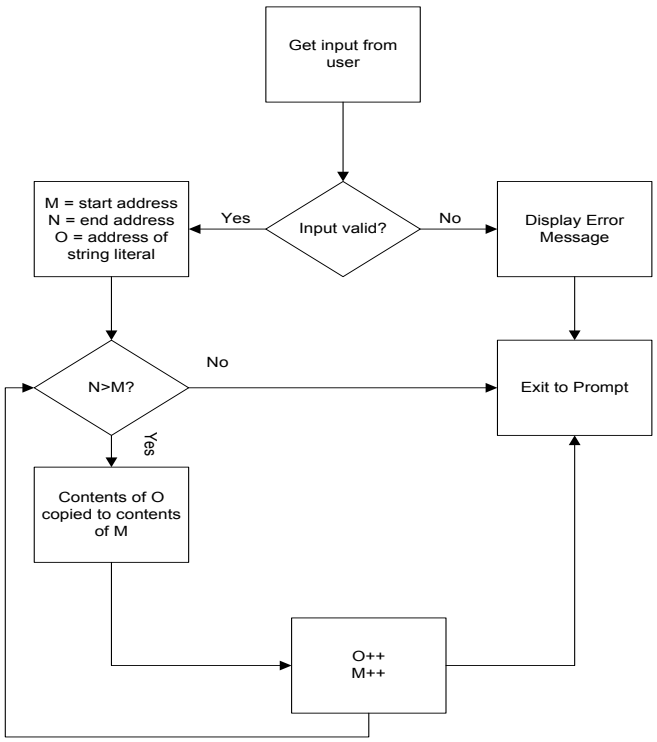
2.2.1 1 BSET (Block Set)

Used the set the contents of memory location to new value

2.2.11. 1 BSET Algorithm and Flowchart

Get input from User
If input is invalid:
    Display Error
    Exit to prompt

M = start address
N = end address
O= address of string literal
While N>M:
    [M] = [O]
    O++
    M++
Exit



2.2.11.2 BSET Assembly Code

```
* BLKSET
* START ADDRESS IS A2, END ADDRESS IS A3
* NEW BLOCK ADDRESS IS IN A4
* TODO: CHANGE THIS SO THAT IT DOES NOT RELY ON READING FROM
* THE INPUT BUFER WHICH MIGHT GIVE ERRORNEOUS OUTPUT
*-------------------------------------------------------------
BLKSET  MOVEA    A0,A2
        MOVEA    EINBF,A3
        JSR      BLKMOV
      RTS
```

2.3) Exception Handlers

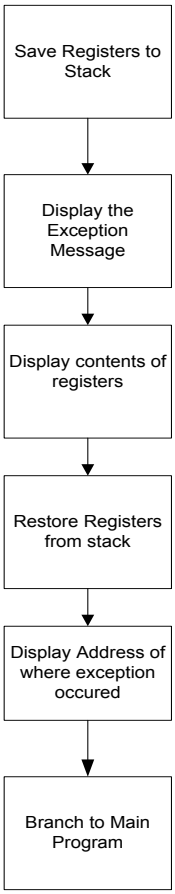
Exception handlers are used to handle exceptions in normal program. When an exception occurs, the program saves the contents in the stack, jumps to the exception routine and upon completion of the exception; it restores the contents from the stack and proceeds to normal operation.
The monitor program modified the existing MC68K exception vectors to point to custom subroutines. These subroutines implemented in the resident monitor are shown below:

2.3.1 Address Error (ADR_ERR)

Address error occurs when a program attempts to write a word or long word to an odd address

2.3.1.1 Algorithm

Display exception message
Save all registers to stack
Display contents of registers
Restore all registers
Display address at which fault occurred
Branch to main program



2.3.1.1 Assembly Code

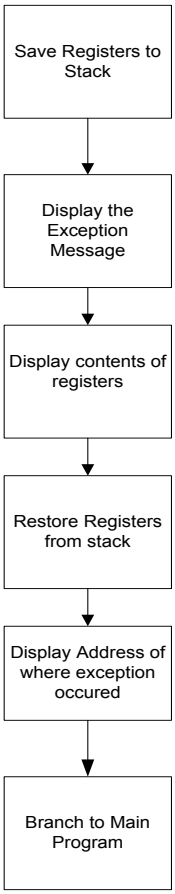
```
ADR_ERR LEA     ADEMSG,A1
        JSR     OTPTCR
        JSR     DSREGS
        MOVE.L  10(A7),D5
        MOVEM.L D0-D4/D6-D7/A0-A7,-(SP)
        JSR     CLROBF
        JSR     CLRBUF
        LEA     OUTBF,A1
        MOVE.W  D5,D1
        JSR     HX2ASC
        MOVE.L  D2,(A1)
        JSR     OTPTCR
        MOVEM.L (SP)+,D0-D4/D6-D7/A0-A7
        BRA         MAIN
```

2.3.2 Bus Error (BUS_ERR)

Occurs when program tries to access a memory location that does not exist

2.3.2.1 Algorithm and Flowchart

Display exception message
Save all registers to stack
Display contents of registers
Restore all registers
Display address at which fault occurred



2.3.2.2 Assembly Code

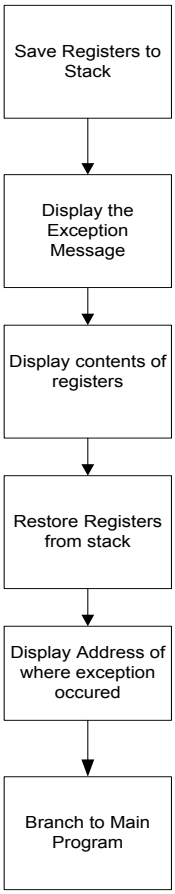
```
BUS_ERR LEA     BSEMSG,A1
        JSR     OTPTCR
        JSR     DSREGS
        MOVE.L  10(A7),D5
        MOVEM.L D0-D4/D6-D7/A0-A7,-(SP)
        JSR     CLROBF
        JSR     CLRBUF
        LEA     OUTBF,A1
        MOVE.W  D5,D1
        JSR     HX2ASC
        MOVE.L  D2,(A1)
        JSR     OTPTCR
        MOVEM.L (SP)+,D0-D4/D6-D7/A0-A7

       BRA          MAIN
```

2.3.4 Divide by Zero Error (DVZ_ERR)

Occurs when program a divide by zero occurs

2.3.4.1 Algorithm and Flowchart

Display exception message
Save all registers to stack
Display contents of registers
Restore all registers
Display address at which fault occurred
Branch to main program



2.3.4.2 Assembly Code

```
DIV_ZRO LEA     DVZMSG,A1
        JSR     OTPTCR
        JSR     DSREGS
        MOVE.L  10(A7),D5                ;SAVE EXCEPTION ADDRESS IN D5
        MOVEM.L D0-D4/D6-D7/A0-A7,-(SP)  ;
        JSR     CLROBF                   ;CLEAR OUTPUT BUFFER
        JSR     CLRBUF                   ;CLEAR INPUT BUFFER
        LEA     OUTBF,A1                 ;SET A1 TO POINT TO OUTBUF
BUFFER
        MOVE.W  D5,D1                    ;PUT EXCEPTION ADDRESS IN D1
        JSR     HX2ASC                   ;CONVERT ADDRESS IN D1 TO
ASCII
        MOVE.L  D2,(A1)                  ;LOAD VALUE INTO BUFFER
        JSR     OTPTCR                   ;OUTPUT VALUE IN BUFFER
        MOVEM.L (SP)+,D0-D4/D6-D7/A0-A7 ;RESTORE REGISTERS
        BRA     MAIN
```

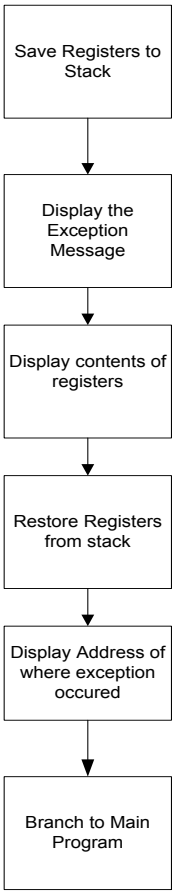
2.3.5 Illegal Instruction Error (ILS_ERR)

Occurs when program tries to execute an instruction that does not exist.

2.3.5.1 Algorithm and Flowchart

Display exception message
Save all registers to stack
Display contents of registers
Restore all registers
Display address at which fault occurred

Branch to main program



2.3.5.2 Assembly Code

```
ILL_INS LEA     ILAMSG,A1       ;INITIALIZE I/O BUFFER TO ERR MESSAG
        JSR     OTPTCR                  ;OUTPUT MESSAGE
        JSR     DSREGS                  ;DISPLAY REGISTERS
        MOVE.L  10(A7),D5               ;SAVE EXCEPTION ADDRESS IN D5
        MOVEM.L D0-D4/D6-D7/A0-A7,-(SP) ;
        JSR     CLROBF                  ;CLEAR OUTPUT BUFFER
        JSR     CLRBUF                  ;CLEAR INPUT BUFFER
        LEA     OUTBF,A1                ;SET A1 TO POINT TO OUTBUF
BUFFER
        MOVE.W  D5,D1                   ;PUT EXCEPTION ADDRESS IN D1
        JSR     HX2ASC                  ;CONVERT ADDRESS IN D1 TO
ASCII
        MOVE.L  D2,(A1)                 ;LOAD VALUE INTO BUFFER
        JSR     OTPTCR                  ;OUTPUT VALUE IN BUFFER
        MOVEM.L (SP)+,D0-D4/D6-D7/A0-A7 ;RESTORE REGISTERS
        BRA     MAIN                    ;BRANCH TO PROMPT
```

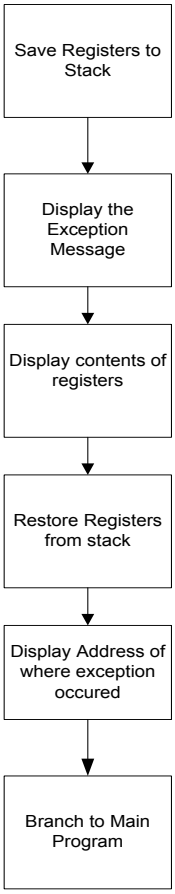
2.3.6 Line A Error (LNA_ERR)

Occurs when program tries to access reserved memory

2.3.6.1 Algorithm and Flowchart

Display exception message
Save all registers to stack
Display contents of registers

Restore all registers
Display address at which fault occurred
Branch to main program



2.3.6.2 Assembly Code

```
LINE_A  LEA     LNAMSG,A1
        JSR     OTPTCR
        JSR     DSREGS
        MOVE.L  10(A7),D5               ;SAVE EXCEPTION ADDRESS IN D5
        MOVEM.L D0-D4/D6-D7/A0-A7,-(SP) ;SAVE REGISTERS
        JSR     CLROBF                  ;CLEAR OUTPUT BUFFER
        JSR     CLRBUF                  ;CLEAR INPUT BUFFER
        LEA     OUTBF,A1                ;SET A1 TO POINT TO OUTBUF
BUFFER
        MOVE.W  D5,D1                   ;PUT EXCEPTION ADDRESS IN D1
        JSR     HX2ASC                  ;CONVERT ADDRESS IN D1 TO
ASCII
        MOVE.L  D2,(A1)                 ;LOAD VALUE INTO BUFFER
        JSR     OTPTCR                  ;OUTPUT VALUE IN BUFFER
        MOVEM.L (SP)+,D0-D4/D6-D7/A0-A7 ;RESTORE REGISTERS
      BRA     MAIN
```

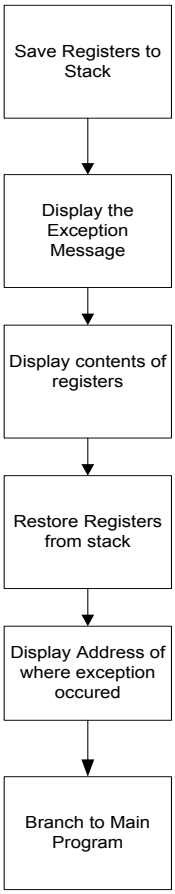
2.3.7 Line F Error (LNF_ERR)

Occurs when program tries to access reserved memory.

2.3.7.1 Algorithm and Flowchart

Display exception message

Save all registers to stack
Display contents of registers
Restore all registers
Display address at which fault occurred
Branch to main program



2.3.7.2 Assembly Code

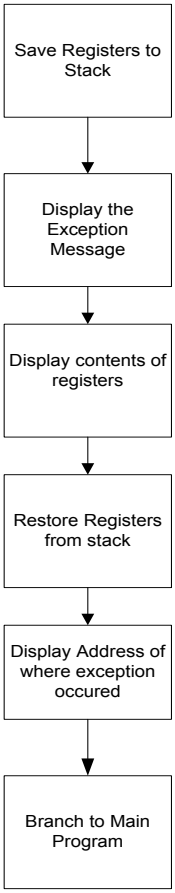
```
LINE_F  LEA     LNFMSG,A1
        JSR     OTPTCR
        JSR     DSREGS
        MOVE.L  10(A7),D5       ;SAVE EXCEPTION ADDRESS IN D5
        MOVEM.L D0-D4/D6-D7/A0-A7,-(SP) ;SAVE REGISTERS
        JSR     CLROBF                  ;CLEAR OUTPUT BUFFER
        JSR     CLRBUF                  ;CLEAR INPUT BUFFER
        LEA     OUTBF,A1                ;SET A1 TO POINT TO OUTBUF
BUFFER
        MOVE.W  D5,D1                   ;PUT EXCEPTION ADDRESS IN D1
        JSR     HX2ASC                  ;CONVERT ADDRESS IN D1 TO
ASCII
        MOVE.L  D2,(A1)                 ;LOAD VALUE INTO BUFFER
        JSR     OTPTCR                  ;OUTPUT VALUE IN BUFFER
        MOVEM.L (SP)+,D0-D4/D6-D7/A0-A7 ;RESTORE REGISTERS
          BRA       MAIN
```

2.3.8 Privilege Violation Error (PRV_ERR)

Occurs when user program tries to modify supervisor registers.

2.3.8.1 Algorithm and Flowchart

Display exception message
Save all registers to stack
Display contents of registers
Restore all registers
Display address at which fault occurred
Branch to main program



2.3.8.2 Assembly Code

```
PRV_VIO LEA     PRVMSG,A1
        JSR     OTPTCR
        JSR     DSREGS
                MOVE.L  10(A7),D5       ;SAVE EXCEPTION ADDRESS IN D5
        MOVEM.L D0-D4/D6-D7/A0-A7,-(SP) ;SAVE REGISTERS
        JSR     CLROBF                  ;CLEAR OUTPUT BUFFER
        JSR     CLRBUF                  ;CLEAR INPUT BUFFER
        LEA     OUTBF,A1                ;SET A1 TO POINT TO OUTBUF
BUFFER
        MOVE.W  D5,D1                   ;PUT EXCEPTION ADDRESS IN D1
        JSR     HX2ASC                  ;CONVERT ADDRESS IN D1 TO
ASCII
        MOVE.L  D2,(A1)                 ;LOAD VALUE INTO BUFFER
        JSR     OTPTCR                  ;OUTPUT VALUE IN BUFFER
        MOVEM.L (SP)+,D0-D4/D6-D7/A0-A7 ;RESTORE REGISTERS
        BRA     MAIN
```

3 Instruction Manual

Here is a quick guide on how to use the monitor program.

- Open Easy68K and locate the assembly file.
- Compile the source code by pushing the F9 key
- If the code compiles without errors, a pop up window will show
- Click on the button that says "Execute" to open up the simulator
- Once the simulator is open, click F9 to start running the monitior program
- A prompt should display in the I/O window
- Type "HELP" to see a list of available commands and their usage
- Choose a command to execute and type in your parameters
- Note: All commands entered must be upper case
- To exit the program, type "EXIT"

4 Discussion

The major design challenge was to take into consideration all the errors which could occur while running the program and write codes for better protection. The extra protection prevents the program from breaking when wrong commands are entered by the user. As such, this ensures not only a fully working program but a robust program.

5 Feature Suggestions

Though the resident monitor program designed worked properly, it provides the user with very limited functionality. The resident monitor can only be improved by increasing the number of debugger commands and protection through exception handlers. It would also be nice to have many of the helper subroutines such as ASCII to BCD, DECIMAL TO BCD included in the user menu.

The program code is generally modular and relies heavily on subroutines calls to perform much of the heavy lifting. However, reliance on subroutines makes the program vulnerable in such a way that if one register that is used elsewhere is modified or gets corrupted, the entire program may fail. It would be a good idea to convert all subroutines into MACROS that will reduce the need to use specific registers for particular subroutines.

6 Conclusion

The overall outcome of the project was a fairly robust monitor program with a lot of time having been spent on developing it. Many other useful functions and subroutines exist in the program code but because of time did not make it to the user menu.
I have learned of key considerations that are needed while developing an operating system for 68K family of microprocessors and could use these ideas if I were to develop a similar programs for other microprocessor architectures

7) Reference

[1] Alan Clemens. Microprocessor Systems Design

[2] Author of Book: Thomas L. Harman & David T. Hein. Digital Computers and computing

[3] ECE 441 Fall 2009 Lab Documents CD

[4] Author of Book:Panos Livadas & Christopher Ward. Computer Organization and the MC68K

```
*-----------------------------------------------------------
* Program    : ECE 441 FINAL PROJECT
* Written by : SUBASH LUITEL
* Date       : 04/30/2013
* Description: MONITOR PROGRAM TO PROVIDE SIMILAR FUNCTIONALITY
*              AS TUTOR.
*-----------------------------------------------------------

        ORG     $1500
ZERO    EQU     $00
DOLLAR  EQU     $24
NUMCMD  EQU     $0B
STACK   EQU     $3000
CR      EQU     $0D
LF      EQU     $0A
SP      EQU     $20
INBUF   DS.B    24
EINBF   DC.B    ' ',0   ;THIS SPACE IS IMPORTANT FOR THE INTERPRETATION OF
COMMANDS
OUTBF   DS.B    24
EOTBF   DC.B    ' ',0
SPACE   DC.B    ' ',0
QUNMRK  DC.B    ' ? ',0
NLINE   DC.B    '',0
*-----------------------------------------------------------
*HELP MENU
*-----------------------------------------------------------
HLPMNU  DC.B    'Available commands:
',CR,0
DHELP   DC.B    ' HELP:  Displays a list of available commands
',CR,0
DMDSP   DC.B    ' MDSP:  MDSP <address1> <address2> eg: MDSP $910 $1000<CR>
',CR,0
        DC.B    '        Outputs the address and memory contents at address
',CR,0
DMCHG   DC.B    ' MCHG:  MCHG <address>[;size] eg: MCHG $1000;W<CR>
',CR,0
        DC.B    '        Displays memory and modify data at specified address
',CR,0
DSRTW   DC.B    ' SRTW:  SRTW <address1> <address2> eg: SRTW $904 $90E<CR>
',CR,0
        DC.B    '        Sorts the data in a block of memory
',CR,0
DBKFL   DC.B    ' BKFL:  BLKF <address1> <address2> <word> eg: BLKF $800 $830
CC<CR>     ',CR,0
        DC.B    '        Fills memory from address1 to address2 with data
',CR,0
DBKSH   DC.B    ' BKSH:  BKSC <address1> <address2> "string" eg: BKSC $900
$910 "MA"<CR> ',CR,0
        DC.B    '        Used to search for a string from memory
',CR,0
DBKMV   DC.B    ' BKMV:  BKMV <address1> <address2> <address3> eg:BKMV $908
$90B $909    ',CR,0
        DC.B    '        Move (duplicate) blocks of memory
',CR,0
DHXDC   DC.B    ' HXDC:  HXDC <hex number> eg: HXDC $30<CR>
',CR,0
        DC.B    '        Convert from hexadecimal number to decimal number
',CR,0
```

```
DEXIT   DC.B    ' EXIT:  EXIT <CR>
',CR,0
        DC.B    '        Exit the program gracefully
',CR,0
DGOTO   DC.B    ' GOTO:  Display contents of the processor registers
',CR,0
        DC.B    '        GO <address> eg: GO $900<CR>
DREG    DC.B    ` DREG:  Display the contents of all the regist
',CR,0
EHMNU   DC.B    ' ',0
*-----------------------------------------------------------

*EXCEPTION MESSAGES
*-----------------------------------------------------------
ADEMSG  DC.B    'ADDRESSS ERROR!           ',CR,0
BSEMSG  DC.B    'BUS ERROR!                ',CR,0
ILAMSG  DC.B    'ILLIGAL ADDRESSS ERROR!   ',CR,0
DVZMSG  DC.B    'DIVIDE BY ZERO!           ',CR,0
PRVMSG  DC.B    'PRIVILAGE VIOLATION ERROR!',CR,0
LNAMSG  DC.B    'LINE A EMULATOR ERROR!    ',CR,0
LNFMSG  DC.B    'LINE A EMULATOR ERROR!    ',CR,0
UPROG   DC.B    'RUNNING USER DEFINED  PROG',CR,0
*-----------------------------------------------------------
DR0     DC.B    'D0 = 0000',0
DR1     DC.B    'D1 = 0000',0
DR2     DC.B    'D2 = 0000',0
DR3     DC.B    'D3 = 0000',0
DR4     DC.B    'D4 = 0000',0
DR5     DC.B    'D5 = 0000',0
DR6     DC.B    'D6 = 0000',0
DR7     DC.B    'D7 = 0000',0
AR0     DC.B    'A0 = 0000',0
AR1     DC.B    'A1 = 0000',0
AR2     DC.B    'A2 = 0000',0
AR3     DC.B    'A3 = 0000',0
AR4     DC.B    'A4 = 0000',0
AR5     DC.B    'A5 = 0000',0
AR6     DC.B    'A6 = 0000',0
US      DC.B    'US = 0000',0
SSP     DC.B    'SS = 0000',0
PCR     DC.B    'PC = 0000',0
*EXTRA LABELS
*-----------------------------------------------------------
PROMPT  DC.B    '[project441:~]$',0
INVCOM  DC.B    'WHAT!',0
ADRERR  DC.B    'ADDRESS ERROR: INVALID ADDRESING MODE',0

*-----------------------------------------------------------
COM_TABL        DC.B    'HELP ',0
                DC.B    'MDSP ',0
                DC.B    'MCHG ',0
                DC.B    'SRTW ',0
                DC.B    'BKMV ',0
                DC.B    'BKFL ',0
                DC.B    'BKSH ',0
                DC.B    'BKST ',0
                DC.B    'HXDC ',0
                DC.B    'EXIT ',0
                DC.B    'DREG ',0
```

```
                DC.B    'GOTO',0
*-------------------------------------------------------------
COM_ADD         DC.W     HELP
                DC.W     MDSP
                DC.W     MCHG
                DC.W     SRTW
                DC.W     BKMV
                DC.W     BKFL
                DC.W     BKSH
                DC.W     BKST
                DC.W     HXDC
                DC.W     EXIT
                DC.W     DREG
                DC.W     GOTO
*-------------------------------------------------------------
ADR_ERR LEA     ADEMSG,A1
        JSR     OTPTCR
        JSR     DSREGS
        MOVE.L  10(A7),D5
        MOVEM.L D0-D4/D6-D7/A0-A7,-(SP)
        JSR     CLROBF
        JSR     CLRBUF
        LEA     OUTBF,A1
        MOVE.W  D5,D1
        JSR     HX2ASC
        MOVE.L  D2,(A1)
        JSR     OTPTCR

        MOVEM.L (SP)+,D0-D4/D6-D7/A0-A7
        BRA     MAIN
BUS_ERR
        LEA     BSEMSG,A1
        JSR     OTPTCR
        JSR     DSREGS
        BRA     MAIN
ILL_INS
        LEA     ILAMSG,A1
        JSR     OTPTCR
        JSR     DSREGS
        BRA     MAIN
DIV_ZRO
        LEA     DVZMSG,A1
        JSR     OTPTCR
        JSR     DSREGS
        BRA     MAIN
PRV_VIO
        LEA     PRVMSG,A1
        JSR     OTPTCR
        JSR     DSREGS
        BRA     MAIN
LINE_A
        LEA     LNAMSG,A1
        JSR     OTPTCR
        JSR     DSREGS
        BRA     MAIN
LINE_F
        LEA     LNFMSG,A1
        JSR     OTPTCR
        JSR     DSREGS
```

```
        BRA     MAIN

* INITIALIZE EXCEPTION VECTORS
*-------------------------------------------------------------

IEXVC   MOVE.L  A5,-(SP)
        MOVEA.L #$8,A5
        MOVE.L  #BUS_ERR,(A5)
        MOVEA.L #$C,A5
        MOVE.L  #ADR_ERR,(A5)
        MOVEA.L #$10,A5
        MOVE.L  #ILL_INS,(A5)
        MOVEA.L #$14,A5
        MOVE.L  #DIV_ZRO,(A5)
        MOVEA.L #$20,A5
        MOVE.L  #PRV_VIO,(A5)
        MOVEA.L #$28,A5
        MOVE.L  #LINE_A,(A5)
        MOVEA.L #$2C,A5
        MOVE.L  #LINE_F,(A5)
        MOVE.L  (SP)+,A5
        RTS

*-------------------------------------------------------------
* DISPLAY SUBROUTINES
*-------------------------------------------------------------
* OUTPUT WITH A CARRIAGE RETURN
*-------------------------------------------------------------
OTPTCR  MOVEM.L D0-D1,-(SP)
        CMP     #0,D1
        BGT     CRNMSET
        MOVE.W  #10,D1          ;SETUP D1 TO CONTAIN MAXIMUM NUMBER OF
CHARACTERS
CRNMSET MOVE    #13,D0
        TRAP    #15
        JSR     CLRBUF
        MOVEM.L (SP)+,D0-D1
        RTS

* OUTPUT WITHOUT A CARRIAGE RETURN
*-------------------------------------------------------------
OTPTNC  MOVEM.L D0-D1,-(SP)
        CMP     #0,D1
        BGT     NCNMSET
        MOVE.W  #100,D1         ;SETUP D1 TO CONTAIN MAXIMUM NUMBER OF
CHARACTERS
NCNMSET MOVE    #14,D0
        TRAP    #15
        JSR     CLRBUF
        MOVEM.L (SP)+,D0-D1
        RTS

* CLEAR BUFFER
*-------------------------------------------------------------
CLRBUF  MOVEM.L A2/A3,-(SP)
        LEA     INBUF,A2
        LEA     EINBF,A3
CLRNXT  CMPA    A2,A3
        BLE     CLRDNE
```

```
        MOVE.W  #$00,(A2)+
        BRA     CLRNXT
CLRDNE  MOVEM.L (SP)+,A2/A3
        RTS

* CLEAR BUFFER
*-------------------------------------------------------------
CLROBF  MOVEM.L A2/A3,-(SP)
        LEA     OUTBF,A2
        LEA     EOTBF,A3
CLRNXT2 CMPA    A2,A3
        BLE     CLRDNE2
        MOVE.W  #$00,(A2)+
        BRA     CLRNXT2
CLRDNE2 MOVEM.L (SP)+,A2/A3
        RTS

* EXIT SUBROUTINE
*-------------------------------------------------------------
MEXIT   MOVE    #9,D0
        TRAP    #15

*INPUT SUBROUTINE
*-------------------------------------------------------------
INPUT   MOVEM.L D0-D7,-(SP)
        LEA     INBUF,A1
        MOVE    #2,D0
        TRAP    #15
        MOVEM.L (SP)+,D0-D7
        RTS

*SUBROUTINE TO DISPLAY PROMPT
*-------------------------------------------------------------
DPRMPT  JSR     CLRBUF
        LEA     PROMPT,A1
        JSR     OTPTNC
        LEA     INBUF,A1
        JSR     INPUT
        RTS

*SUBROUTINE TO DISPLAY ALL REGISTERS
*-------------------------------------------------------------
DSREGS  MOVEM.L D0-D7/A0-A5/A7,-(A6)
        LEA     DR0,A1
        JSR     SHWREG
        LEA     DR1,A1
        JSR     SHWREG
        LEA     DR2,A1
        JSR     SHWREG
        LEA     DR3,A1
        JSR     SHWREG
        JSR     DNLINE
        LEA     DR4,A1
        JSR     SHWREG
        LEA     DR5,A1
        JSR     SHWREG
        LEA     DR6,A1
        JSR     SHWREG
        LEA     DR7,A1
```

```
        JSR     SHWREG
        JSR     DNLINE
        LEA     AR0,A1
        JSR     SHWREG
        LEA     AR1,A1
        JSR     SHWREG
        LEA     AR2,A1
        JSR     SHWREG
        LEA     AR3,A1
        JSR     SHWREG
        JSR     DNLINE
        LEA     AR4,A1
        JSR     SHWREG
        LEA     AR5,A1
        JSR     SHWREG
        LEA     US,A1
        JSR     OTPTNC
        MOVE.L  (A6),D1
        JSR     HX2ASC
        JSR     DVALUE
        JSR     DSPACE
        LEA     SSP,A1
        JSR     OTPTNC
        MOVE.L  (A6)+,D1
        JSR     HX2ASC
        JSR     DVALUE
        JSR     DSPACE
        JSR     DNLINE
*MOVEM.L        (A6)+,D0-D7/A0-A5/A7
        BRA     MAIN

*SUBROUTINE TO DISPLAY A SPACE
*-------------------------------------------------------------
DSPACE  MOVEM.L A1,-(SP)
        LEA     SPACE,A1
        JSR     OTPTNC
        MOVEM.L (SP)+,A1
        RTS

*SUBROUTINE TO DISPLAY LINE FEED
*-------------------------------------------------------------
DLINEF  MOVEM.L A1,-(SP)
        LEA     LF,A1
        JSR     OTPTCR
        MOVEM.L (SP)+,A1
        JSR     CLRBUF
        RTS

*SUBROUTINE TO DISPLAY A QUESTION MARKS
*-------------------------------------------------------------
DQNMRK  MOVEM.L A1,-(SP)
        LEA     QUNMRK,A1
        JSR     OTPTNC
        MOVEM.L (SP)+,A1
        RTS

*SUBROUTINE TO DISPLAY A NEWLINE
*-------------------------------------------------------------
```

```
DNLINE  MOVEM.L A1,-(SP)
        LEA     NLINE,A1
        JSR     OTPTCR
        JSR     CLRBUF
        MOVEM.L (SP)+,A1
        RTS

*SUBROUTINE TO DISPLAY A VALUE
*-------------------------------------------------------------

DVALUE  MOVEM.L A1,-(SP)
        LEA     OUTBF,A1
        JSR     CLROBF
        MOVE.L  D2,(A1)
        JSR     OTPTNC
        MOVEM.L (SP)+,A1
        RTS

*SUBROUTINE TO DISPLAY HELP MENU
*-------------------------------------------------------------
MHELP   MOVEM.L A1,-(SP)
        LEA     EHMNU,A3
        LEA     HLPMNU,A1
LPMHP   CMPA    A1,A3
        BLE     HMDNE
        JSR     OTPTCR
        ADDA    #$4A,A1
        BRA     LPMHP
HMDNE   MOVEM.L (SP)+,A1
        RTS

*-------------------------------------------------------------
UFAIL   MOVEM.L A1,-(SP)
        LEA     INVCOM,A1
        JSR     OTPTCR
        MOVEM.L (SP)+,A1
        BRA     MAIN

*-------------------------------------------------------------
UFAIL2  MOVEM.L A1,-(SP)
        JSR     OTPTCR
        MOVEM.L (SP)+,A1
        BRA     MAIN

*convert a value stored in register D1 from binary to BCD
* result is stored in D4
*-------------------------------------------------------------
BINTBCD MOVEM.L D1-D2,-(SP)
        MOVE.W  #7,D2
        CLR.L   D4
NXTDIG  DIVU    #10,D1
        SWAP    D1
        OR.B    D1,D4
        ROR.L   #4,D4
        CLR.B   D1
        SWAP    D1
        DBRA    D2,NXTDIG
        MOVEM.L (SP)+,D1-D2
        RTS
```

```
* convert a value store in register D0 from BCD to binary
* DATAIS READ FROM DATA REGISTER D0, RESULT STORED IN D0D0
*-----------------------------------------------------------
BCDTBIN MOVEM.L D1,-(SP)
        ROR.W   #4,D4
        MOVE.B  D4,D1
        MULU    #10,D1
        AND.B   #0,D4
        ROR.W   #8,D4
        ROR.B   #4,D4
        ADD.W   D1,D4
        MOVEM.L (SP)+,D1
        RTS

* DATA IS READ FROM DATA REGISTER D2, RESULT STORED IN D6
*-----------------------------------------------------------
ASTBIN  MOVEM.L D2/D4-D5,-(SP)  ;SAVE REGISTERS
        CLR.L   D4
        CLR.L   D5
        CLR.L   D6
        MOVE    #4,D4           ;SETUP COUNTER
LOOP2   CMP.B   #$39,D2         ;GREATER THAN $39
        BGT     CMFRT           ;MORE,COMPARE WITH $41
        CMP.B   #$31,D2         ;COMPARE WITH $31
        BLT     FAIL2           ;LESS, BRANCH TO FAIL
        SUBI.B  #$30,D2         ;CONVERT TO HEX
        BRA     NXT2            ;SAVE BYTE TO D2
CMFRT   CMP.B   #$41,D2         ;COMPARE WITH $41
        BLT     FAIL2           ;LESS, BRANCH TO FAIL
        CMP.B   #$46,D2         ;COMPARE WITH $45
        BGT     FAIL2           ;MORE, BRANCH TO FAIL
        SUBI.B  #$37,D2         ;CONVERT TO HEX
NXT2    ROR.L   #8,D2           ;SAVE BYTE TO D2
        SUBI    #1,D4
        BGT     LOOP2
        BRA     ASCDNE
FAIL2   MOVE.B  #$00,D2
        BRA     NXT2
ASCDNE  MOVE.L  #4,D5
FIXI    OR.B    D2,D6
        ROR.W   #4,D6
        ROR.L   #8,D2
        SUBI.B  #1,D5
        BGT     FIXI
ASDONE  MOVEM.L (SP)+,D2/D4-D5  ;RESTORE REGISTERS
        RTS

* DECIMAL TO ASCII
* DECIMAL VALUE IN D4, RESULT IN D5
*-----------------------------------------------------------
DECASC  MOVEM.L D2,-(SP)
        MOVE.L  #4,D2
REPET   MOVE.B  D4,D5
        ANDI.B  #$0F,D5
        ADDI.B  #$30,D5
        ROR.L   #8,D5
        ROR.W   #4,D4
        SUBI.B  #1,D2
```

```
        BGT     REPET
        MOVEM.L (SP)+,D2
        RTS

* HEX TO DECIMAL
* HEX VALUE READ FROM INPUT IBUFFER, RESULT IN D5
*-------------------------------------------------------------
HX2DEC  JSR     ASTBIN  ; ASCII READ FROM D2 PLACED IN D6
        MOVE.L  D6,D1   ; BIN VALUE MOVED FROM D6 TO D1
        JSR     BINTBCD ; BIN VALUE IN D1, BCD IN D4
        JSR     DECASC  ; BCD VALUE IN D4, ASCII IN D5
        RTS

* MEMORY BLOCK DISPLAY
* START ADDRESS IN A3, END ADDRESS IN A4
*-------------------------------------------------------------
MMDSP   LEA     INBUF,A1        ;INITIALIZE BUFFER ADDRESS
        MOVEM.L D1/D3,-(SP)     ;SAVE REGISTERS
        CLR.L   D1              ;CLEAR D1
        CLR.L   D3              ;CLEAR D3
LPDMA   MOVE    #$10,D3         ;SET UP COUNTER FOR TEN BYTES
        JSR     MMADSP          ;DISPLAY MEMORY ADDRESS
LPDSP   MOVE.B  (A3)+,D1
        JSR     HX2ASC          ;CONVERT FROM HEX TO ASCII
        MOVE.W  D2,(A1)         ;MOVE WORD INTO BUFFER
        JSR     OTPTNC          ; OUTPUT THE CONTENTS OF A1
        JSR     DSPACE          ; DISPLAY SPACE
        CMPA    A3,A4           ; A4-A3
        BLE     MDEXT           ; IF END OF BLOCK REACHED RETURN FROM
SUBROUTINE
        SUBI    #1,D3           ;DECREMENT COUNTE
        BLE     DNULIN          ;DISPLAY NEW LINE
        BRA     LPDSP           ;LOOP BACK
DNULIN  JSR     DNLINE          ;DISPLAY NEW LINE
        BRA     LPDMA           ;LOOP BACK
MDEXT   JSR     DNLINE          ;DISPLAY NEW LINE
        MOVEM.L (SP)+,D1/D3     ;RESTORE REGISTERS
        RTS                     ;RETURN FROM SUBROUTINE

* MEMORY ADDRESS DISPLAY
* MEMORY ADDRESS AT A3
*-------------------------------------------------------------
MMADSP  MOVEM.L D1-D2/A1-A3,-(SP)
        JSR     CLRBUF
        LEA     INBUF,A1
        MOVE.W  #$3030,(A1)
        ADDA    #2,A1
        MOVE    A3,D1
        JSR     HX2ASC
        MOVE.L  D2,(A1)
        ADDA    #4,A1
        MOVE.W  #$3A20,(A1)     ; #$3A20 => :_
        LEA     INBUF,A1
        JSR     OTPTNC
        JSR     CLRBUF
CLRDE   MOVEM.L (SP)+,D1-D2/A1-A3
        RTS

* HEXADECIMAL TO ASCII
```

```
* VALUE TO BE CONVERTED IS STORED IN D1 RESULT IS IN D2
*-------------------------------------------------------
HX2ASC  MOVEM.L D3,-(SP)
        CLR.L   D2
        MOVE.L  #3,D3
H2ALUP  MOVE.B  D1,D2
        ANDI.B  #$0F,D2
        CMPI.B  #9,D2
        BLE     H2ALE9
        ADD.B   #$37,D2
        BRA     H2ANXT
H2ALE9  ADD.B   #$30,D2
H2ANXT  ROR.L   #8,D2
        ROR.W   #4,D1
        SUBI    #1,D3
        BGE     H2ALUP
        MOVEM.L (SP)+,D3
        RTS

* BLOCK FILL
* TAKES TWO ADDRESSES AND A WORD AND FILLS ALL MEMORY IN RANGE
* WITH THAT WORD
* USING ADDRESS REGISTERS: START ADRESS AT A2, END ADDRESS AT
* A5, DATA IN REGISTER D4
*-------------------------------------------------------
BLKFIL  MOVEM.L A1,-(SP)
        LEA     INBUF,A1
        MOVE.B  D4,D3           ;TAKES BYTE ENTERED AND COPIES IT
        ROL.W   #8,D4           ;SO THAT IT FILLS UP A WORD
        MOVE.B  D3,D4           ;IN D4
        MOVE.W  D4,D1
        JSR     HX2ASC
BFDMA   MOVE    #$08,D5
        JSR     MMADSP
BKFLP   MOVE.W  D4,(A3)+
        MOVE.L  D2,(A1)
        JSR     OTPTNC
        JSR     DSPACE
        CMPA    A3,A5
        BLE     BFEXT
        SUBI    #1,D5
        BLE     BFNLIN
        BRA     BKFLP
BFNLIN  JSR     DNLINE
        BRA     BFDMA
BFEXT   JSR     DNLINE
        MOVEM.L (SP)+,A1
        RTS

* MEM MODIFY
* START ADDRESS IN A4, END ADDRESS IN A5
*-------------------------------------------------------
MEMMOD  MOVEM.L D1/D2/D6,-(SP)
        LEA     INBUF,A1
        JSR     CLRBUF
        *MOVEA.L        A4,A3
NXTVAL  JSR     MMADSP          ;DISPLAY CURRENT ADDRESS
        MOVE.L  (A3),D1         ;MOVE CONTENTS OF ADDRESS TO D1 FOR DISPLAY
        JSR     HX2ASC          ;DISPLAY CONTENTS OF ADDRESS
```

```
        MOVE.L  D2,(A1)
        JSR     OTPTNC          ;OUTPUT THE CONTENTS OF A1
        JSR     DQNMRK          ;OUTPUT QUESTION MARK
        JSR     INPUT           ;TAKE INPUT FROM USER
        CMPI.B  #$2E,(A1)
        BEQ     MMDONE
        CMPI.B  #$30,(A1)
        BLT     UFAIL           ;ERROR MESSAGE SUBROUTINE
        CMPI.B  #$41,(A1)
        BLT     CHKNXT
        CMPI.B  #$46,(A1)
        BGT     UFAIL           ;ERROR MESSAGE SUBROUTINE
        BRA     CONTNU
CHKNXT  CMPI.B  #$39,(A1)
        BGT     UFAIL
CONTNU  MOVE.L  (A1),D2
        JSR     ASTBIN
        MOVE.W  D6,(A3)
        CLR.L   D6
        ADDA    #2,A3
        BRA     NXTVAL
        LEA     INBUF,A1
MMDONE  MOVEM.L (SP)+,D1/D2/D6
        RTS
* SORTW
* START ADDRESS IN A3, END ADDRESS IN A2
* USES BUBBLE SORT
* A4 WILL CONTAIN A COPY OF THE START ADDRESS
*-----------------------------------------------------------
SORTW   MOVEM.L D2-D4/A3,-(SP)
        MOVEA.L A3,A4
RESRT   CLR.L   D2
        MOVEA.L A4,A3
NEXTN   MOVE.B  (A3)+,D3
        MOVE.B  (A3),D4
        CMP.W   D3,D4
        BGT     CHKEND
        CMP.W   D3,D4
        BEQ     CHKEND
        ADDI    #1,D2
        SUBA    #1,A3
        MOVE.B  D4,(A3)+
        MOVE.B  D3,(A3)
CHKEND  CMPA    A5,A3
        BGE     CHKSWP
        BRA     NEXTN
CHKSWP  CMPI    #0,D2
        BGT     RESRT
        MOVEM.L (SP)+,D2-D4/A3
        RTS

* BLKSCH
* START ADDRESS IS $700, END ADDRESS IS $6000
* MAKE SURE THAT YOUR INPUT BUFFER DOES NOT LIE IN THE
* SEARCH ADDRESS SPACE OTHERWISE YOU WILL GET A FALSE POSITIVE
* IF STRING IS FOUND, IT AND IT LOCATION WILL BE DISPLAYED
* IF NOT, THE SUBROUTINE WILL SIMPLY EXIT WITH NO CONFIRMATION
* GIVEN TO THE USER
*-----------------------------------------------------------
```

```
BLKSCH  MOVEM.L D2-D4/A0,-(SP)
        CLR.L   D2
        CLR.L   D3
        ADDA    #1,A0
        MOVEA   A0,A2
*       MOVEA   #$700,A3          ;A3 IS THE ADDRESS WE ARE GOING TO START
OUR SEARCH FROM
*       MOVEA   #$6000,A4         ;PLACE OUR STOP ADDRESS IN A4
GETSRG  CMPI.B  #$22,(A2)+        ;CHECK IS WE HAVE REACHED THE END OF THE
STRING THAT WAS INPUT
        BEQ     SEARCH            ;IF WE HAVE REACHED THE END OF THE STRING,
WE GO TO SEARCH
        ADDI.B  #1,D2             ;INCREMENT THE CHARACTER COUNTER
        BRA     GETSRG            ;GET THE NEXT VALID CHRACTER ENTERED
SEARCH  MOVEA   A0,A2             ;RESET A2 TO ITS INITIAL POINT. WE ARE
STARTING OVER
        MOVE.L  D2,D4             ;COPY THE COUNTER VALUE FOR RESET
GETNXT  CMPA    A3,A4             ;CHECK IF WE'VE REACHED THE END OF
SEARCHABLE MEMORY
        BLE     SCHDNE            ;IF WE HAVE REACHED THE END THEN, RETURN
FORM SUBROUTINE
        MOVE.B  (A3)+,D3          ;MOVE THE BYTE WE WISH TO MATCH INTO D3
        CMP.B   (A2),D3           ;CHECK IF IT MATCHES WITH ONE OF THE
CHARACTERS ENTERED
        BNE     SEARCH            ;IF NOT RESET THE COUNTER
        SUBI.B  #1,D4             ;IF IT DOES, DECREMENT THE COUNTER
        BLE     DONESR            ;IF COUNTER IS ZERO, WE ARE DONE
        ADDA    #1,A2             ;OTHERWISE, INCREMENT A2 TO THE NEXT BYTE
        BRA     GETNXT            ;GET THE NEXT CHARACTER TO MATCH
DONESR  MOVE.B  #$00,(A3)         ;WE TERMINATE THE STRING FOUND WITH
        SUBA.L  D2,A3             ;SET A3 TO POINT TO THE START OF THE STRING
THE WE MATCHED
        MOVEA   A3,A2             ;COPY A3 TO A2 AS PER THE MMADSP
REQUIREMENTS
        JSR     MMADSP            ;CALL MMADSP TO DSIPLAY THE ADDRESS AT
A2/A3
        MOVEA   A2,A1             ;COPY THE LOCATION OF THE ADDRESS OF STRING
FOUND TO A1
        JSR     OTPTCR            ;DISPLAY THE CONTENTS OF THAT ADDRESS I.E
DISPLAY THE STRING FOUND
SCHDNE  MOVEM.L (SP)+,D2-D4/A0    ;RESTORE THE RESGISTER CONTENTS
        RTS

* BLKMOV
* START ADDRESS IS A2, END ADDRESS IS A3
* NEW BLOCK ADDRESS IS IN A4
*-----------------------------------------------------------
BLKMOV  MOVEM.L A2-A3,-(SP)
MVBYT   CMPA    A2,A3
        BLE     MVDONE
        MOVE.B  (A2)+,(A4)+
        BRA     MVBYT
MVDONE  MOVEM.L (SP)+,A2-A3
        RTS
* BLKSET
* START ADDRESS IS A2, END ADDRESS IS A3
* NEW BLOCK ADDRESS IS IN A4
* TODO: CHANGE THIS SO THAT IT DOES NOT RELY ON READING FROM
* THE INPUT BUFER WHICH MIGHT GIVE ERRORNEOUS OUTPUT
```

```
*-------------------------------------------------------------
BLKSET  MOVEA   A0,A2
        MOVEA   EINBF,A3
        JSR     BLKMOV
        RTS

* INTERPRETER
*-------------------------------------------------------------
INTPRT  MOVE.L  #$20,D1
        CLR.L   D2
        MOVE.L  #$41,D3
        MOVE.L  #$5A,D4
        CMP.B   (A1),D3
        BGT     UFAIL
        CMP.B   (A1),D4
        BLT     UFAIL
NXTCHI  CMP.B   (A1),D1
        BEQ     CHKCM
        ADDA    #1,A1
        CMP.B   #4,D2
        BGE     CHKCM
        ADDI    #1,D2
        BRA     NXTCHI
CHKCM   LEA     COM_TABL,A2
        LEA     INBUF,A3
        CLR.L   D3
        MOVE.L  (A3),D5
NXTCMD  CMPI.B  #NUMCMD,D3
        BGT     UFAIL
        MOVE.L  (A2),D6
        CMP.L   D5,D6
        BEQ     RUNCMD
        ADDI    #1,D3
        ADDA    #6,A2
        BRA     NXTCMD
RUNCMD  MOVEM.L D5-D6,-(SP)     ; ARGUMENT PARSER STARTS HERE
        CLR.L   D5
        MOVE.L  #$20,D6
        LEA     EINBF,A3
GETSPC  CMPA    A1,A3
        BLE     TDONE
        CMP.B   (A1)+,D6
        BNE     GETSPC
        MOVE.L  A1,D7
        JSR     SKPUSH
        ADD     #1,D5
        BRA     GETSPC
TDONE   MOVE    D5,D7
        JSR     SKPUSH
*       MOVEM.L (SP)+,D5-D6     ;ARGUMENT PARSER ENDS HERE
        LEA     COM_ADD,A4
        MULU    #2,D3           ;GET DISPLACEMENT WITHIN TRANSLATION TABLE
I.E 2*D3; D3=#6
        ADDA    D3,A4
        MOVEA   (A4),A5
        JMP     (A5)
        RTS
*-------------------------------------------------------------
SKPUSH
```

```
        SUBA    #2,A6
        MOVE.W  D7,(A6)
        RTS
*-------------------------------------------------------------
STKPOP  MOVE.W  (A6),D7
        ADDA    #2,A6
        RTS
*-------------------------------------------------------------
RMDLLR  MOVEM.L D2,-(SP)
        CLR.L   D1
        CMPI.B  #DOLLAR,(A2)
        BNE     UFAIL
        CMP.B   #SP,4(A2)
        BEQ     ISSPC
        CMP.B   #ZERO,4(A2)
        BEQ     ISSPC
        CLR.B   (A2)+
        BRA     NOSPC
ISSPC   CLR.B   (A2)
NOSPC   MOVE.L  #3,D2
CONTIN  MOVE.B  (A2)+,D1
        CMP.B   #ZERO,D2
        BLE     RMDDN
        SUBI.L  #1,D2
        ROL.L   #8,D1
        BRA     CONTIN
RMDDN   MOVEM.L (SP)+,D2
        RTS
*-------------------------------------------------------------
HELP    JSR     CLRBUF
        JSR     MHELP
        BRA     MAIN
*-------------------------------------------------------------
MDSP    JSR     STKPOP
        MOVEA   #DMDSP,A1
        CMP     #2,D7
        BNE     UFAIL2
        JSR     STKPOP
        MOVEA   D7,A2
        JSR     RMDLLR
        MOVE.L  D1,D2
        JSR     ASTBIN
        MOVEA.L D6,A4
        JSR     STKPOP
        MOVEA   D7,A2
        JSR     RMDLLR
        MOVE.L  D1,D2
        JSR     ASTBIN
        MOVEA.L D6,A3
        JSR     MMDSP
        JMP     MAIN
*-------------------------------------------------------------
MCHG    JSR     STKPOP
        MOVEA   #DMCHG,A1
        CMP     #1,D7
        BNE     UFAIL2
        JSR     STKPOP
        MOVEA   D7,A2
        JSR     RMDLLR
```

```
        MOVE.L  D1,D2
        JSR     ASTBIN
        MOVEA.L D6,A3
        JSR     MEMMOD
        BRA     MAIN
*-------------------------------------------------------
SRTW    JSR     STKPOP
        MOVEA   #DSRTW,A1
        CMP     #2,D7
        BNE     UFAIL2
        JSR     STKPOP
        MOVEA   D7,A2
        JSR     RMDLLR
        MOVE.L  D1,D2
        JSR     ASTBIN
        MOVEA.L D6,A5
        JSR     STKPOP
        MOVEA   D7,A2
        JSR     RMDLLR
        MOVE.L  D1,D2
        JSR     ASTBIN
        MOVEA.L D6,A3
        JSR     SORTW
        BRA     MAIN
*-------------------------------------------------------
BKMV    JSR     STKPOP
        MOVEA   #DBKMV,A1
        CMP     #3,D7
        BNE     UFAIL2
        JSR     STKPOP
        MOVEA   D7,A2
        JSR     RMDLLR
        MOVE.L  D1,D2
        JSR     ASTBIN
        MOVEA.L D6,A4
        JSR     STKPOP
        MOVEA   D7,A2
        JSR     RMDLLR
        MOVE.L  D1,D2
        JSR     ASTBIN
        MOVEA.L D6,A3
        JSR     STKPOP
        MOVEA   D7,A2
        JSR     RMDLLR
        MOVE.L  D1,D2
        JSR     ASTBIN
        MOVEA.L D6,A2
        JSR     BLKMOV
        BRA     MAIN

*-------------------------------------------------------
BKFL    JSR     STKPOP
        MOVEA   #DBKFL,A1
        CMP     #3,D7
        BNE     UFAIL2  ;
        JSR     STKPOP
        MOVEA   D7,A2
        *--------------
        MOVE.B  (A2)+,D2
```

```
        ROL.W   #8,D2
        MOVE.B  (A2),D2
        *--------------
        JSR     ASTBIN
        MOVE.L  D6,D4
        JSR     STKPOP
        MOVEA   D7,A2
        JSR     RMDLLR
        MOVE.L  D1,D2
        JSR     ASTBIN
        MOVEA.L D6,A5
        JSR     STKPOP
        MOVEA   D7,A2
        JSR     RMDLLR
        MOVE.L  D1,D2
        JSR     ASTBIN
        MOVEA.L D6,A3
        JSR     BLKFIL
        JMP     MAIN

*-------------------------------------------------------------
BKSH    JSR     STKPOP
        MOVEA   #DBKSH,A1
        CMP     #3,D7
        BNE     UFAIL2
        JSR     STKPOP
        MOVE.W  D7,A0           ;string loc
        JSR     STKPOP
        MOVEA   D7,A2
        JSR     RMDLLR
        MOVE.L  D1,D2
        JSR     ASTBIN
        MOVE.W  D6,A4           ;address 2
        JSR     STKPOP
        MOVEA   D7,A2
        JSR     RMDLLR
        MOVE.L  D1,D2
        JSR     ASTBIN
        MOVE.W  D6,A3           ;address 1
        JSR     BLKSCH
        JMP     MAIN
*-------------------------------------------------------------
BKST    JSR     STKPOP
        CMP     #2,D7
        BLT     UFAIL2
        SUBI    #2,D7
        ADD     #1,D7
        ADDA    D7,A6
        JSR     STKPOP
        MOVEA   D7,A0           ;STRING START ADDRESS
        JSR     STKPOP
        MOVEA   D7,A2
        JSR     RMDLLR
        MOVE.L  D1,D2
        JSR     ASTBIN
        MOVE.W  D6,A4           ;BLOCK LOCATION
        JMP     MAIN
*-------------------------------------------------------------
HXDC    JSR     STKPOP
```

```
        MOVEA   #DHXDC,A1
        CMP     #1,D7
        BNE     UFAIL2
        JSR     STKPOP
        MOVEA   D7,A2
        JSR     RMDLLR
        MOVE.L  D1,D2
        JSR     HX2DEC
        MOVEM.L A1,-(SP)
        JSR     CLRBUF
        LEA     INBUF,A1
        MOVE.L  D5,(A1)
        JSR     OTPTCR
        MOVEM.L (SP)+,A1
        JMP     MAIN
*-------------------------------------------------------------
EXIT    JSR     MEXIT
*-------------------------------------------------------------
DREG    JSR     STKPOP
        CMP     #0,D7
        BLT     UFAIL2
        BRA     DSREGS
        BRA     MAIN
*-------------------------------------------------------------
GOTO    JSR     STKPOP
        CMP     #0,D7
        BLT     UFAIL2
        JSR     STKPOP
        MOVEA   D7,A2
        JSR     RMDLLR
        MOVE.L  D1,D2
        JSR     ASTBIN
        MOVE.W  D6,A0
        JMP     (A0)
*-------------------------------------------------------------
SHWREG  JSR     OTPTNC
        MOVE.L  (A6)+,D1
        JSR     HX2ASC
        JSR     DVALUE
        JSR     DSPACE
        RTS


MYPROG  LEA     UPROG,A1
        JSR     OTPTCR
        BRA     MAIN

* MAIN PROGRAM
*-------------------------------------------------------------


        ORG     $1000
START:  JSR     IEXVC
        MOVE.L  #STACK,A6
*       MOVE.L  #$FFFFFFFF,A4
*       JMP     (A4)

*-------------------------
MAIN    ANDI.W  #$0700,SR
```

```
        BRA     MAIN
        JSR     DPRMPT
        JSR     INTPRT
        BRA     MAIN
*-----------------------------
        JSR     MYPROG
        BRA     MAIN
        END     START           ; last line of source
```