Illinois Institute of Technology

ECE 441 Monitor Project

Author:
Adam Sumner

Teaching Assistant:
Boyang Wang

April 28th, 2015

Acknowledgment

I acknowledge all of the work including figures and code belongs to me and/or persons who are referenced.

Contents

List of Figures

Abstract

This project involved designing and implementing a Monitor program using the MC68000 assembly language. The program implements twelve basic debugger functions as well as two author defined functions. It is designed to handle exceptions, and is meant to be an educational piece of software for students taking ECE 441 at the Illinois Institute of Technology.

1 Introduction

The SANPER-1 ELU is a Motorola MC68000 based microcomputer designed by Dr. Jafar Saniie and Mr. Stephen Perich for use in college level computer engineering courses. For user interaction, it utilizes a monitor program called TUTOR that enables users to actively interact with the microcomputer. The design objective of this project is to re-implement the functionality of TUTOR into a student written monitor program titled MONITOR441. The program should be able to perform basic debugger functions such as memory display, memory sort, memory change, etc., and must have the ability to handle exceptions. The design constraints are:

- Code must be smaller that 3K starting from address $1000
- Stack size must be 1K starting at memory location $3000
- Macros may not be used
- Erroneous inputs should not kill the program

Twelve debugger functions must be implemented, along with two user defined debugger commands.

2 Monitor Program

The monitor program operates in a command driven environment. It acts as a typical shell, providing a user interface to access the microcomputer's services. The main program being run is a command line interpreter. Based on the input that the user enters, the interpreter determines if the input entered is valid and subsequently executes the specified command. It was

developed using the Easy68K Simulator, thus the TRAP #15 handler is used instead of the MC68000's TRAP #14 handler. The structure of how this program operates is shown in Figure 1.

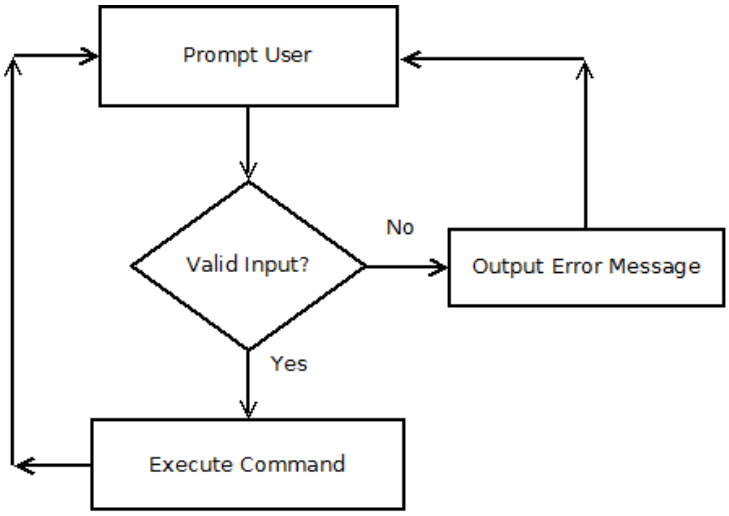

Figure 1: Structure of Monitor Program

2.1 Command Interpreter

2.1.1 Algorithm and Flowchart

The algorithm for the command interpreter uses simple string matching to determine if input is correct. The algorithm begins by outputting the message `MONITOR441>` and accepting input from the user. It then checks for the ASCII value $48 which corresponds to the letter H. This is to check for either the `HELP` command or `HXDC` command. If an H was not entered, it then checks for the ASCII value $4D which corresponds to a memory command. If this fails, then it checks for ASCII value $47, corresponding to the `GO` command. If this fails, the ASCII value $44 is tested, corresponding to the `DF` command. If this fails, it checks for $42, which signifies a `BLCK` command. If this fails, $53 is tested for the `SORTW` command. If this fails, $45 is tested for the `ECHO` command. If this fails $2E is checked for the modify register command. If all of these checks fail, the user has entered incorrect input and an error message is displayed. If any of these checks succeed, the command line interpreter jumps to the respective command's helper interpreter function.

These subroutines check for each character of the user input in order to verify the command the user entered was correct. These helper functions also serve to differentiate commands that start with the same character. The flowchart for this process is shown in Figure 2.

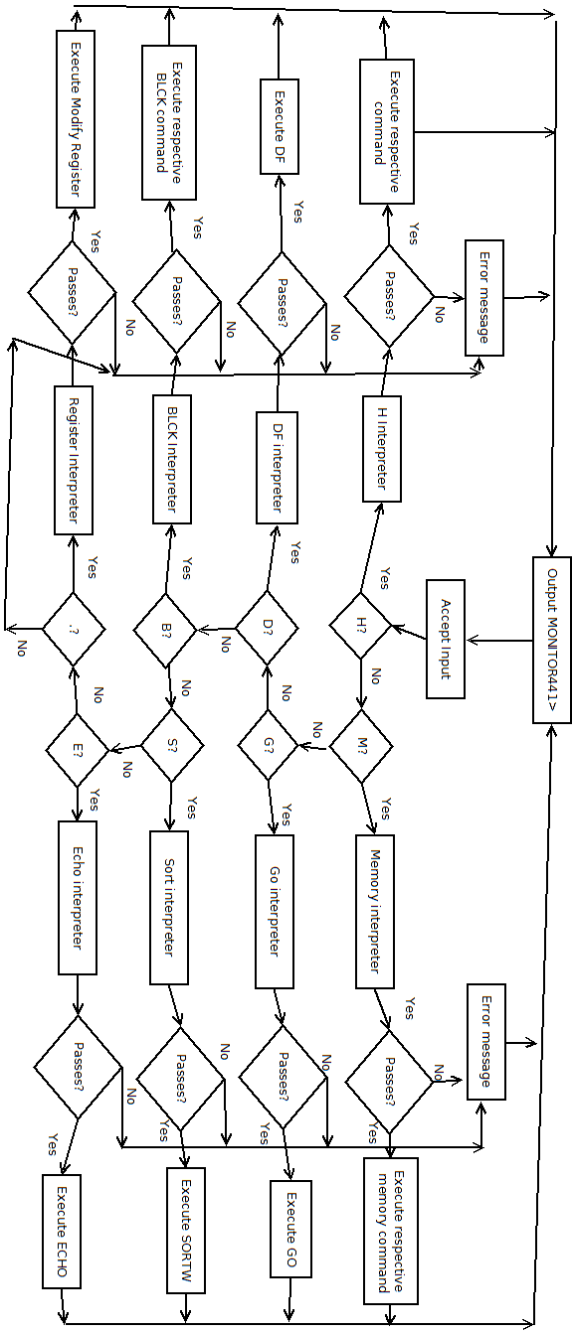

Figure 2: Flowchart for Command Line Interpreter

2.1.2 Assembly Code

```
154 SHELL:
155             PEA     *               ;save PC on Stack for DF
156             ADD.L   #4,SP           ;get original value of stack
    pointer
157             MOVE.L  SP,-8(SP)       ;save it
158             ADD.L   #-8,SP          ;update Stack position
159             MOVE    SR,-(SP)        ;save Status register for use
    with DF
160             MOVE.L  A6,-(SP)        ;temp save
161             MOVE    USP,A6          ;for use with DF command
162             MOVE.L  A6,-(SP)        ;store USP to STACK
163             ADD.L   #4,SP
164             MOVE.L  (SP),A6     ;restore original value
165             MOVE.L  -(SP),4(SP)  ;move correct value to correct
    stack position
166             ADD.L   #4,SP           ;point stack to CORRECT PLACE
167
168
169             MOVEM.L D0-D7/A0-A6,-(SP)  ;save initial values of
    registers
170             MOVEM.L D0-D7/A0-A6,-(SP)  ;unorthodox
    implementation to save registers when using DF command
171
172
173             LEA   PROMPT,A1         ;Load message
174             MOVE.W  #11,D1          ;load n bytes
175             MOVE.B  #1,D0           ;set up trap call
176             TRAP    #15
177             LEA     BUFFER,A1       ;set up storage for command
178             MOVE.B  #2,D0           ;load input trap call
179             TRAP    #15
180             CMP.B   #$48,(A1)       ;check for help/hxdc
181             BEQ     HELPORHXDC
182             CMP.B   #$4D,(A1)       ;check for memory command
183             BEQ     MEMTEST
184             CMP.B   #$47,(A1)       ;check for go
185             BEQ     GOTST
186             CMP.B   #$44,(A1)       ;check for df
187             BEQ     DFTST
188             CMP.B   #$42,(A1)       ;check for blck command
189             BEQ     BLCKTEST
190             CMP.B   #$53,(A1)       ;check for sort command
191             BEQ     SORTTEST
```

```
192                 CMP.B   #$45,(A1)     ;check for echo command
193                 BEQ     ECHOTEST
194                 CMP.B   #$2E,(A1)     ;check for modify register
    command
195                 BEQ     MODIFYREGTEST
196                 BRA     UNKNOWNCMD
197 RESTORE:        MOVEM.L (SP)+,D0-D7/A0-A6
198                 MOVEM.L (SP)+,D0-D7/A0-A6 ;double restore because of
     DF hack workaround
199                 ADD.L   #4,SP         ;account for USP, it'll fix
    itself (it shouldn't be used)
200                                       ;EASY68k simulator starts in
    supervisor mode
201                 MOVE    (SP)+,SR
202                 MOVE.L  (SP)+,D0      ;save stack cuz it'll get
    destroyed
203                 ADD.L   #4,SP         ;get rid of PC, itll fix itself
204                 MOVE.L  D0,SP
205                 CLR.L   D0            ;no longer needed
206
207                 BRA     SHELL
208 *   ------------------------------------------------------------------------

209
210 ECHOTEST:      ADD.L   #1,A1
211                 CMP.B   #$43,(A1)+    ;C?
212                 BNE     UNKNOWNCMD
213                 CMP.B   #$48,(A1)+    ;H?
214                 BNE     UNKNOWNCMD
215                 CMP.B   #$4F,(A1)+    ;O?
216                 BNE     UNKNOWNCMD
217                 CMP.B   #$20,(A1)+    ;SPACE?
218                 BEQ     ECHO
219                 BRA     ERRORSR
220 *   ------------------------------------------------------------------------

221
222
223 *   ------------------------------------------------------------------------

224
225 BLCKTEST:      ADD.L   #1,A1
```

```
226                 CMP.B   #$46,(A1)       ;BF?
227                 BEQ     BFTEST
228                 CMP.B   #$4D,(A1)       ;BMOV?
229                 BEQ     BMOVTEST
230                 CMP.B   #$54,(A1)       ;BTST?
231                 BEQ     BTSTTEST
232                 CMP.B   #$53,(A1)       ;BSCH?
233                 BEQ     BSCHTEST
234                 BRA     UNKNOWNCMD
235 *---------------------------------------------------------------------------

236
237 BSCHTEST:      ADD.L   #1,A1
238                 CMP.B   #$43,(A1)
239                 BNE     UNKNOWNCMD
240                 ADD.L   #1,A1
241                 CMP.B   #$48,(A1)
242                 BNE     UNKNOWNCMD
243                 ADD.L   #1,A1
244                 CMP.B   #$20,(A1)
245                 BNE     ERRORSR
246                 BRA     BSCH
247
248 *---------------------------------------------------------------------------

249
250 BTSTTEST:
251                 ADD.L   #1,A1
252                 CMP.B   #$53,(A1)
253                 BNE     UNKNOWNCMD
254                 ADD.L   #1,A1
255                 CMP.B   #$54,(A1)
256                 BNE     UNKNOWNCMD
257                 ADD.L   #1,A1
258                 CMP.B   #$20,(A1)
259                 BNE     ERRORSR
260                 BRA     BTST
261
262 *---------------------------------------------------------------------------

263
264 BMOVTEST:      ADD.L   #1,A1
```

```
265                 CMP.B   #$4F,(A1)
266                 BNE     UNKNOWNCMD
267                 ADD.L   #1,A1
268                 CMP.B   #$56,(A1)
269                 BNE     UNKNOWNCMD
270                 ADD.L   #1,A1
271                 CMP.B   #$20,(A1)
272                 BNE     ERRORSR
273                 BRA     BMOV
274 *
```

```
275 BFTEST:         ADD.L   #1,A1
276                 CMP.B   #$20,(A1)
277                 BNE     ERRORSR
278                 BRA     BF
279 *
```

```
280
281 DFTST:          ADD.L   #1,A1
282                 CMP.B   #$46,(A1)
283                 BNE     UNKNOWNCMD
284                 ADD.L   #1,A1
285                 CMP.B   #$00,(A1)
286                 BNE     ERRORSR
287                 BRA     DF
288 *
```

```
289
290 SORTTEST:       ADD.L   #1,A1
291                 CMP.B   #$4F,(A1)     ;O?
292                 BNE     UNKNOWNCMD
293                 ADD.L   #1,A1
294                 CMP.B   #$52,(A1)     ;R?
295                 BNE     UNKNOWNCMD
296                 ADD.L   #1,A1
297                 CMP.B   #$54,(A1)     ;T?
298                 BNE     UNKNOWNCMD
299                 ADD.L   #1,A1
300                 CMP.B   #$57,(A1)     ;W?
301                 BNE     UNKNOWNCMD
302                 ADD.L   #1,A1
303                 CMP.B   #$20,(A1)
```

```
304                 BNE       ERRORSR
305
306                 BRA       SORTW
307 *
    ------------------------------------------------------------------------------------------------------------------------

308
309 GOTST:          ADD.L     #1,A1
310                 CMP.B     #$4F,(A1)
311                 BNE       UNKNOWNCMD
312                 ADD.L     #1,A1
313                 CMP.B     #$20,(A1)+
314                 BNE       ERRORSR
315                 BRA       GO
316 *
    ------------------------------------------------------------------------------------------------------------------------

317
318 HELPORHXDC:    ADD.L     #1,A1
319                 CMP.B     #$45,(A1)     ;is it help?
320                 BEQ       HELPTST
321                 CMP.B     #$58,(A1)     ;or is it hxdc
322                 BEQ       HXDCTEST
323                 BRA       UNKNOWNCMD
324 *
    ------------------------------------------------------------------------------------------------------------------------

325
326 HELPTST:
327                 ADD.L     #1,A1    ; check next char
328                 CMP.B     #$4C,(A1) ;check for L
329                 BNE       UNKNOWNCMD
330                 ADD.L     #1,A1
331                 CMP.B     #$50,(A1)    ;check for P
332                 BNE       UNKNOWNCMD
333                 ADD.L     #1,A1    ;check for anything else
334                 CMP.B     #$00,(A1)
335                 BNE       ERRORSR
336                 BRA       HELP
337
338
339
340 *
    ------------------------------------------------------------------------------------------------------------------------
```

```
341
342 MEMTEST:       ADD.L    #1,A1
343                CMP.B    #$53,(A1)
344                BEQ      MSSPCTEST
345                CMP.B    #$44,(A1)
346                BEQ      MDSPCTEST
347                CMP.B    #$4D,(A1)
348                BEQ      MMSPCTEST
349                BRA      UNKNOWNCMD
350
351 MSSPCTEST     ADD.L    #1,A1
352                CMP.B    #$20,(A1)
353                BEQ      MEMSET
354                BRA      ERRORSR
355
356 MDSPCTEST:
357                ADD.L    #1,A1
358                CMP.B    #$53,(A1)
359                BNE      ERRORSR
360                ADD.L    #1,A1
361                CMP.B    #$50,(A1)
362                BNE      UNKNOWNCMD
363                ADD.L    #1,A1
364                CMP.B    #$20,(A1)
365                BEQ      MEMDISP
366                BRA      ERRORSR
367
368 MMSPCTEST:    ADD.L    #1,A1
369                CMP.B    #$20,(A1)
370                BEQ      MM
371                BRA      ERRORSR
372 *
```

```
373 HXDCTEST:
374                ADD.L    #1,A1
375                CMP.B    #$44,(A1)
376                BNE      UNKNOWNCMD
377                ADD.L    #1,A1
378                CMP.B    #$45,(A1)
379                BNE      UNKNOWNCMD
380                ADD.L    #1,A1
381                CMP.B    #$43,(A1)
382                BNE      UNKNOWNCMD
383                ADD.L    #1,A1
```

```
384                 CMP.B    #$20,(A1)
385                 BNE      ERRORSR
386                 BRA      HXDC
387 *
    ------------------------------------------------------------------------------

388 MODIFYREGTEST:
389                 ADD.L    #1,A1
390                 CMP.B    #$44,(A1)
391                 BEQ      MRD
392                 CMP.B    #$41,(A1)
393                 BEQ      MRA
394                 BRA      UNKNOWNCMD
395
396 *-------------------------------USER DEFINED COMMANDS
    ---------------------------------------*
397 *
    ------------------------------------------------------------------------------

398 ECHO: *What terminal DOESN'T have echo?*
399
400         MOVE.L   A1,A2    ;setup to find end of string
401 EEND:   CMP.B    #$00,(A2)+
402         BEQ      EFOUND
403         BRA      EEND
404 EFOUND:
405         SUB.L    #1,A2    ;off by one
406         SUB.L    A1,A2    ;find out how many bytes
407         MOVE.L   A2,D1    ;place it for trap function
408         MOVE.L   #0,D0
409         TRAP     #15
410
411         BRA RESTORE
```

2.2 Debugger Commands

2.2.1 Help

2.2.1.1 Algorithm and Flowchart

Help is a simple command that prints out a series of strings that display the available commands, their syntax, and a short description of each command. The syntax to invoke this command is `HELP`. The flowchart for this command is shown in Figure 3.

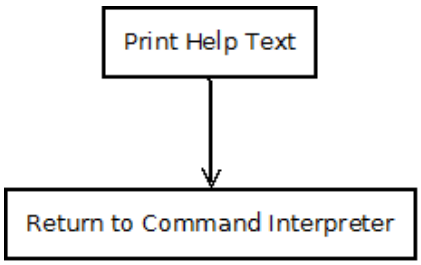

Figure 3: Flowchart for Help

2.2.1.2 Assembly Code

```
796 HELP:       LEA     HTXT,A1         ;list of commands test
797             MOVE.W  #17,D1
798             MOVE.B  #0,D0
799             TRAP    #15
800             MOVE.W  #0,D1           ;newline
801             TRAP    #15
802
803             LEA     HTXT1,A1        ;mem display command
804             MOVE.W  #75,D1
805             MOVE.B  #0,D0
806             TRAP    #15
807             LEA     HTXT1A,A1       ;mem display
808             MOVE.W  #61,D1
809             MOVE.B  #0,D0
810             TRAP    #15
811             LEA     HTXT1B,A1       ;mem display
812             MOVE.W  #20,D1
813             MOVE.B  #0,D0
814             TRAP    #15
815             MOVE.W  #0,D1           ;newline
816             TRAP    #15
817
818             LEA     HTXT2,A1        ;hxdec command text
819             MOVE.W  #75,D1
820             MOVE.B  #0,D0
821             TRAP    #15
822             MOVE.B  #0,D1           ;newline
823             TRAP    #15
824
825             LEA     HTXT3,A1        ;sort command text
826             MOVE.W  #69,D1
```

```
827             MOVE.B  #0,D0
828             TRAP    #15
829             LEA     HTXT3A,A1     ;sort command text continued
830             MOVE.W  #57,D1
831             MOVE.B  #0,D0
832             TRAP    #15
833             LEA     HTXT3B,A1     ;sort command text continued
834             MOVE.W  #20,D1
835             MOVE.B  #0,D0
836             TRAP    #15
837             LEA     HTXT3C,A1     ;sort command text continued
838             MOVE.W  #21,D1
839             MOVE.B  #0,D0
840             TRAP    #15
841             LEA     HTXT3D,A1     ;sort command text continued
842             MOVE.W  #29,D1
843             MOVE.B  #0,D0
844             TRAP    #15
845             LEA     HTXT3E,A1     ;sort command text continued
846             MOVE.W  #51,D1
847             MOVE.B  #0,D0
848             TRAP    #15
849             MOVE.B  #0,D1         ;newline
850             TRAP    #15
851
852             LEA     HTXT4,A1      ;memory modify command text
853             MOVE.W  #71,D1
854             MOVE.B  #0,D0
855             TRAP    #15
856             LEA     HTXT4A,A1     ;mem modify command text
    continued
857             MOVE.W  #69,D1
858             MOVE.B  #0,D0
859             TRAP    #15
860             LEA     HTXT4B,A1     ;mem modify command text
    continued
861             MOVE.W  #27,D1
862             MOVE.B  #0,D0
863             TRAP    #15
864             LEA     HTXT4C,A1     ;mem modify command text
    continued
865             MOVE.W  #30,D1
866             MOVE.B  #0,D0
867             TRAP    #15
```

```
868             LEA     HTXT4D,A1       ;mem modify command text
    continued
869             MOVE.W  #31,D1
870             MOVE.B  #0,D0
871             TRAP    #15
872             LEA     HTXT4E,A1       ;mem modify command text
    continued
873             MOVE.W  #36,D1
874             MOVE.B  #0,D0
875             TRAP    #15
876             MOVE.B  #0,D1
877             TRAP    #15             ;newline
878
879             LEA     HTXT5,A1        ;memory set command text
880             MOVE.W  #70,D1
881             MOVE.B  #0,D0
882             TRAP    #15
883             LEA     HTXT5A,A1       ;memory set command text
    continued
884             MOVE.W  #9,D1
885             MOVE.B  #0,D0
886             TRAP    #15
887             MOVE.B  #0,D1           ;newline
888             TRAP    #15
889
890             LEA     HTXT6,A1        ;block fill command text
891             MOVE.W  #69,D1
892             MOVE.B  #0,D0
893             TRAP    #15
894             LEA     HTXT6A,A1       ;block fill command text
895             MOVE.W  #72,D1
896             MOVE.B  #0,D0
897             TRAP    #15
898             LEA     HTXT6B,A1       ;block fill command text
899             MOVE.W  #38,D1
900             MOVE.B  #0,D0
901             TRAP    #15
902             MOVE.B  #0,D1
903             TRAP    #15             ;newline
904
905
906             LEA     HTXT7,A1        ;block move command text
907             MOVE.W  #68,D1
908             MOVE.B  #0,D0
909             TRAP    #15
```

```
910             LEA     HTXT7A,A1       ;block move command text
911             MOVE.W  #72,D1
912             MOVE.B  #0,D0
913             TRAP    #15
914             LEA     HTXT7B,A1       ;block move command text
915             MOVE.W  #24,D1
916             MOVE.B  #0,D0
917             TRAP    #15
918             MOVE.B  #0,D1           ;newline
919             TRAP    #15
920
921             LEA     HTXT8,A1        ;block test command text
922             MOVE.W  #71,D1
923             MOVE.B  #0,D0
924             TRAP    #15
925             LEA     HTXT8A,A1       ;block test command text
926             MOVE.W  #40,D1
927             MOVE.B  #0,D0
928             TRAP    #15
929             MOVE.B  #0,D1           ;newline
930             TRAP    #15
931
932             LEA     HTXT9,A1        ;block search command text
933             MOVE.W  #70,D1
934             MOVE.B  #0,D0
935             TRAP    #15
936             LEA     HTXT9A,A1       ;block search command text
937             MOVE.W  #45,D1
938             MOVE.B  #0,D0
939             TRAP    #15
940             MOVE.B  #0,D1           ;newline
941             TRAP    #15
942
943             LEA     HTXT10,A1       ;go command text
944             MOVE.W  #61,D1
945             MOVE.B  #0,D0
946             TRAP    #15
947             MOVE.B  #0,D1          ;newline
948             TRAP    #15
949
950             LEA     HTXT11,A1       ;df command text
951             MOVE.W  #56,D1
952             MOVE.B  #0,D0
953             TRAP    #15
954             MOVE.B  #0,D1
```

```
955             TRAP    #15
956
957             LEA     HTXT12,A1       ;help command text
958             MOVE.W  #66,D1
959             MOVE.B  #0,D0
960             TRAP    #15
961             MOVE.B  #0,D1           ;newline
962             TRAP    #15
963
964             LEA     HTXT13,A1       ;echo command text
965             MOVE.W  #52,D1
966             MOVE.B  #0,D0
967             TRAP    #15
968             MOVE.B  #0,D1           ;newline
969             TRAP    #15
970
971             LEA     HTXT14,A1       ;modify register command text
972             MOVE.W  #71,D1
973             MOVE.B  #0,D0
974             TRAP    #15
975             LEA     HTXT15,A1       ;modify register command text
976             MOVE.W  #63,D1
977             MOVE.B  #0,D0
978             TRAP    #15
979             MOVE.B  #0,D1           ;newline
980             TRAP    #15
981
982             BRA     RESTORE
```

2.2.2 Memory Display

2.2.2.1 Algorithm and Flowchart

Memory display is an extremely useful tool to look at blocks of memory. The syntax to call this function is `MDSP <address1> <address2`, where `<address1>` is the starting address and `<address2>` is the ending address of the memory contents to be shown. This command also displays the block of memory from `<address1>` to `<address2 +16bytes>`. The flowchart for this command is shown in Figure 4.

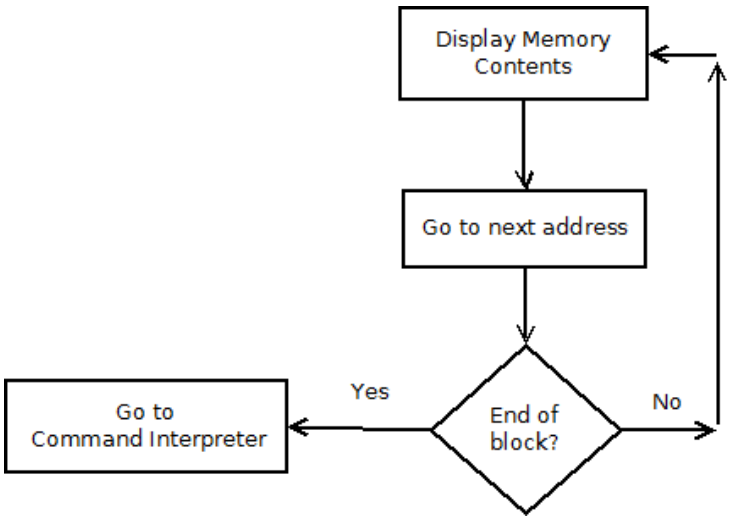

Figure 4: Flowchart for Memory Display

2.2.2.2 Assembly Code

```
1014 MEMDISP:    LEA       BUFFER,A2
1015             MOVE.L    #1,D6          ;counter for how many times to
       loop
1016             ADD.L     #5,A2          ;get first address
1017             MOVE.L    A2,A3
1018 FINDEND1:   CMP.B     #$20,(A3)+
1019             BEQ       FINDNEXT
1020             BRA       FINDEND1
1021 FINDNEXT:   MOVE.L    A3,A4
1022             MOVE.L    A3,A5
1023             SUB.L     #1,A3     ;get rid of off by one error
1024 FINDEND2:   CMP.B     #$00,(A5)+
1025             BEQ       MEMNEXT
1026             BRA       FINDEND2
1027 MEMNEXT:    SUB.L     #1,A5     ;off by one error
1028             JSR       ASCII_ADDRESS
1029             MOVE.L    D5,A6     ;put 1st address in A6
1030             MOVE.L    A4,A2
1031             MOVE.L    A5,A3
1032             JSR ASCII_ADDRESS
1033             MOVE.L    D5,A5     ;second address in A5
1034             MOVE.L    A6,A0     ;for second run through
1035             MOVE.L    A5,A1     ;see above comment
```

```
1036                ADD.L    #16,A1 ;16 byte offset
1037                MOVEM.L  A1,-(SP)
1038 DISPLOOP:     CMP.L    A6,A5
1039                BLT      SECONDLOOP
1040                MOVE.L   A6,D3
1041                JSR      HEXTOASCII
1042                SUB.L    A2,A3
1043                MOVE.L   A3,D1    ;number of ascii values to display
1044                MOVE.L   A2,A1
1045                MOVE.L   #1,D0
1046                TRAP     #15
1047                LEA      SPACE,A1
1048                MOVE.L   #1,D1
1049                TRAP     #15
1050                CLR.L    D3
1051                MOVE.B   (A6),D3
1052                JSR      HEXTOASCII
1053                SUB.L    A2,A3
1054                MOVE.L   A3,D1
1055                MOVE.L   A2,A1
1056                MOVE.L   #0,D0
1057                TRAP     #15
1058                ADD.L    #1,A6
1059                BRA      DISPLOOP
1060
1061 SECONDLOOP:
1062                MOVE.B   #0,D0
1063                MOVE.B   #0,D1
1064                TRAP     #15
1065                MOVEM.L  (SP)+,A1
1066                MOVE.L   A0,A6    ;reinit
1067                MOVE.L   A1,A5
1068                SUBI.L   #1,D6
1069                CMP.L    #$0,D6
1070                BEQ      DISPLOOP
1071                SUB.L    #4,SP    ;off by long error on stack
1072                BRA      RESTORE
```

2.2.3 HXDEC

2.2.3.1 Algorithm and Flowchart

This command will allow the user to enter a hexadecimal value (up to FFFF), and the program will return the equivalent value in decimal format. The syntax to call this function is `HXDEC <data>`. It works by extracting the

ASCII values byte by byte and determining the 16's place of each byte. The value extracted is then multiplied by its respective 16's place and added to a register that stores the total. This total must then be converted into BCD for output and then into ASCII to display it on the terminal.

2.2.3.2 Assembly Code

```
1076 HXDC:     LEA BUFFER,A2     ;load buffer
1077           ADD.L    #6,A2    ; start of number
1078           MOVE.L   A2,A3    ;set up end pointer
1079           MOVE.L   #1,D1    ;set up 16's place
1080           CLR.L    D2       ;clear total
1081           CLR.L    D3       ;temp holder for number
1082           CLR.L    D6       ;Final Value in BCD
1083           MOVE.L   #10000,D4     ;maximum 10's place of converted
       number
1084           MOVE.L   #16,D5       ;Max number of rotates needed
1085           LEA $3A00,A5
1086           LEA $3A00,A4    ;set up start pointer
1087 FINDLASTNUM:
1088           CMP.B #$00,(A3)+
1089           BEQ      CONVERTMINUS1
1090           BRA      FINDLASTNUM
1091 CONVERTMINUS1:
1092                SUB.L    #1,A3 ; cure off by 1 error
1093 CONVERT:
1094                SUB.L    #1,A3
1095                CMP      A3,A2
1096                BGT      ENDCONVERT
1097                CMP.B    #$40,(A3)
1098                BGT      HIGHHEX
1099                SUBI.B   #$30,(A3)    ;get hex value
1100                BRA      COMPUTATION
1101 HIGHHEX:       SUBI.B   #$37,(A3)   ;get hex value
1102 COMPUTATION:
1103                MOVE.B   (A3),D3
1104                MULU     D1,D3    ;get 16's place
1105               ; DIVU     #16,D3  ;get rid of off by 1 exponent error
1106                MULU     #16,D1   ;inc 16's place counter
1107                MOVE.B   D3,(A4)
1108                SUB.L    #1,A4
1109                ADD.L    D3,D2    ;store it in total for debugging
1110                CLR.L    D3       ;get rid of any numbers in there
1111                BRA      CONVERT
```

```
1112 ENDCONVERT:                    ;must convert back to ascii for
       display
1113             CLR.L    D3        ;Cleared for workability
1114             DIVU     D4,D2     ;get 10's place digit
1115             MOVE.W   D2,D3     ;extract 10's place digit to D3
1116             ROL.L    D5,D3     ;put it in its place
1117             CLR.W    D2        ;get rid of whole number
1118             SWAP     D2        ;keep remainder
1119             SUBI.L   #4,D5     ;dec rotate counter
1120             ADD.L    D3,D6     ;put it into it's place
1121             DIVU     #10,D4    ;go down a 10's place
1122             CMP.W    #0,D4     ;are we done
1123             BEQ      OUTPUTNUM
1124             BRA      ENDCONVERT
1125
1126 OUTPUTNUM:
1127             MOVE.L   D6,D3     ;put into register for conversion to
       ASCII
1128             JSR      HEXTOASCII
1129             MOVEA.L  A2,A1     ;get start of number
1130             SUBA     A2,A3     ;get how many bytes to output
1131             MOVE.L   A3,D1     ;for Trap call
1132             MOVE.L   #0,D0
1133             TRAP     #15
1134
1135             BRA RESTORE
```

2.2.4 SORTW

This command implements the most common sort algorithm for a set of data, the bubble sort. Because the user has the choice to choose between sorting the data in ascending or descending order, it also implements a "rock" sort. It works by first determining which option, ascending or descending, the user has selected. Once determined, the first data in the set is analyzed to the next immediate adjacent value in memory. If the current data is larger than the next data (assuming ascending order for example), the two words of data are swapped. This value is continuously checked against its immediate adjacent memory until it "fits" in the current state of the list. This process is repeated for n elements in a list of n words. The runtime is $\mathcal{O}(n^2)$, and the syntax for this command is `SORTW <option> <address1> <address2>`, where both `<address1>` and `<address2>` are even addresses.

2.2.4.1 Algorithm and Flowchart

dfg

2.2.4.2 Assembly Code

```
1139 SORTW:   ADD.L    #1,A1          ;increment to check for semicolon/
         dash
1140          CMP.B    #$2D,(A1)      ;check for default
1141          BEQ      DESCEND
1142          CMP.B    #$3B,(A1)+
1143          BNE      ERRORSR
1144          CMP.B    #$41,(A1)      ;is it ascending?
1145          BEQ      ASCEND
1146          CMP.B    #$44,(A1)      ;or descending?
1147          BNE      ERRORSR
1148          BRA      DESCEND
1149
1150 ASCEND:
1151          ADD.L    #1,A1     ;inc
1152          CMP.B    #$20,(A1)      ;check space
1153          BNE      ERRORSR
1154          ADD.L    #1,A1     ;start of 1st address
1155          MOVE.L   A1,A2
1156          MOVE.L   A2,A3
1157 AGETFIRSTADDRESS:
1158          CMP.B    #$00,(A3)
1159          BEQ      ERRORSR        ;incorrect syntax
1160          CMP.B    #$20,(A3)+     ;trying to find the end
1161          BEQ      AFADDCONV
1162          BRA      AGETFIRSTADDRESS
1163 AFADDCONV:
1164          SUB.L    #1,A3     ;off by one error
1165          JSR ASCII_ADDRESS      ;D5 now has that address
1166          MOVE.L D5,A4
1167          ADD.L    #1,A3     ;start of second address
1168          MOVE.L   A3,A2     ;setup for second address
1169 AGETSECADDRESS:
1170          CMP.B    #$00,(A3)+     ;trying to find the end
1171          BEQ      ASADDCONV
1172          BRA      AGETSECADDRESS
1173 ASADDCONV:
1174           SUB.L   #1,A3     ;off by one
1175           JSR     ASCII_ADDRESS
1176           MOVE.L  D5,A5
```

```
1177            MOVEA.L   A4,A6   ;CLR A6
1178
1179 ARESETLOOP: MOVE.L   A6,A4     ;reset to top of loop
1180 ACMP:       CMP.W    (A4)+,(A4)+   ;check adjacent mem
1181             BLS.S    ASWAP
1182             SUBQ.L   #2,A4
1183             CMP.L    A4,A5    ;done?
1184             BNE      ACMP      ;nope
1185             BRA      DONEASCEND   ;yep
1186 ASWAP:       MOVE.L   -(A4),D0     ;start bubbling
1187             SWAP.W   D0
1188             MOVE.L   D0,(A4)
1189             BRA      ARESETLOOP
1190
1191
1192 DESCEND:
1193          ADD.L     #1,A1    ;inc
1194          CMP.B     #$20,(A1)    ;check space
1195          BNE       ERRORSR
1196          ADD.L     #1,A1    ;start of 1st address
1197          MOVE.L    A1,A2
1198          MOVE.L    A2,A3
1199 DGETFIRSTADDRESS:
1200          CMP.B     #$00,(A3)
1201          BEQ       ERRORSR      ;incorrect syntax
1202          CMP.B     #$20,(A3)+   ;trying to find the end
1203          BEQ       DFADDCONV
1204          BRA       DGETFIRSTADDRESS
1205 DFADDCONV:
1206          SUB.L     #1,A3    ;off by one error
1207          JSR ASCII_ADDRESS     ;D5 now has that address
1208          MOVE.L D5,A4
1209          ADD.L     #1,A3    ;start of second address
1210          MOVE.L    A3,A2    ;setup for second address
1211 DGETSECADDRESS:
1212          CMP.B     #$00,(A3)+   ;trying to find the end
1213          BEQ       DSADDCONV
1214          BRA       DGETSECADDRESS
1215 DSADDCONV:
1216           SUB.L    #1,A3    ;off by one
1217           JSR      ASCII_ADDRESS
1218           MOVE.L   D5,A5
1219           MOVEA.L   A4,A6   ;CLR A6
1220
1221 DRESETLOOP: MOVE.L   A6,A4     ;reset to top of loop
```

```
1222 DCMP:          CMP.W    (A4)+,(A4)+   ;check adjacent mem
1223                BHI.S    DSWAP
1224                SUBQ.L   #2,A4
1225                CMP.L    A4,A5     ;done?
1226                BNE      DCMP       ;nope
1227                BRA      DONEDESCEND   ;yep
1228 DSWAP:          MOVE.L   -(A4),D0      ;start bubbling
1229                SWAP.W   D0
1230                MOVE.L   D0,(A4)
1231                BRA      DRESETLOOP
1232 x
1233 DONEASCEND:
1234 DONEDESCEND:
1235                BRA RESTORE
```

2.2.5 Memory Modify

This command first determines which option the user has selected. Depending on this option, it reads the address entered by the user and displays the specified amount of data currently stored in memory. The user is then prompted to enter data to store into memory. The command increments the memory location and asks for input until the user enters the '.' character. The syntax for this command is `MM <option> <address>`.

2.2.5.1 Algorithm and Flowchart

2.2.5.2 Assembly Code

```
1239 MM:      CLR.L     D2        ;used for storing values
1240          CLR.L     D6
1241 SIZECHECK:
1242           MOVE.L   A1,A3     ;set up to find end ptr
1243 ENDPTRMM:
1244           CMP.B    #$00,(A3)+
1245           BNE      ENDPTRMM
1246           SUB.L    #1,A3     ;off by one error
1247           ADD.L    #1,A1     ;inc pointer to start of specifier
1248           CMP.B    #$2D,(A1)    ;check for default
1249           BEQ      DEFAULT
1250           CMP.B    #$3B,(A1)
```

```
1251            BNE       ERRORSR
1252            ADD.L     #1,A1     ;find out which size
1253            CMP.B     #$57,(A1) ; word
1254            BEQ       WORD
1255            CMP.B     #$4C,(A1)     ;long
1256            BEQ       LONG
1257            BRA       ERRORSR
1258
1259 ************************************************
1260
1261 DEFAULT:
1262
1263            ADD.L     #2,A1         ;set up for subroutine
1264            MOVE      A1,A2         ;set up for subroutine
1265            MOVEM.L   D1/D6/A1-A3,-(SP)
1266            JSR       ASCII_ADDRESS
1267            MOVEM.L   (SP)+,D1/D6/A1-A3
1268            MOVE.L    D5,A4         ;set up address to modify
1269
1270 MODIFYLOOP:
1271            *--------Display Memory First------*
1272            MOVE.L    A4,D3         ;set up for subroutine
1273            JSR       HEXTOASCII    ;convert new address to ascii for
       output
1274            SUBA      A2,A3         ;get num of bytes to produce
1275            MOVE.L    #1,D0
1276            MOVE.L    A3,D1
1277            MOVE.L    A2,A1
1278            TRAP      #15
1279
1280            *add colon to denote containing data*
1281            MOVE.B    #$3A,(A1)
1282            MOVE.L    #1,D1     ;display only the colon
1283            MOVE.L    #1,D0
1284            TRAP      #15
1285
1286            MOVE.B    (A4),D3
1287            JSR       HEXTOASCII
1288            MOVE.L    #2,D1
1289            SUB.L     A2,A3
1290            CMP       #2,A3
1291            BEQ       FORMATGOOD
1292            SUB.L     #1,A2
1293 FORMATGOOD:
1294            MOVE.L    A2,A1
```

```
1295            MOVE.B   #1,D0
1296            TRAP     #15
1297
1298            MOVE.B   #$20,(A1)
1299            MOVE.L   #1,D1    ;space between held data and input
1300            MOVE.L   #1,D0
1301            TRAP     #15
1302
1303
1304            *------Enter Input-------*
1305            CLR.L    D3
1306            MOVE.L   #4,D6
1307            LEA      BUFFER,A1    ;set up storage for command
1308            MOVE.B   #2,D0        ;load input trap call
1309            TRAP     #15
1310            CMP.B    #$2E,(A1)
1311            BEQ      ENDLP
1312            CMP.B    #$00,(A1)
1313            BEQ      ENTER
1314
1315 PARSELOOP:
1316            CMP.B    #$00,(A1)
1317            BEQ      ENDPARSE
1318            CMP.B    #$40,(A1)
1319            BGT      HIGHHEXMM
1320            SUBI.B   #$30,(A1)    ;get hex value
1321            BRA      NEXTMMSTEP
1322 HIGHHEXMM: SUBI.B   #$37,(A1)   ;get hex value
1323 NEXTMMSTEP:
1324            MOVE.B   (A1),D2
1325            ROL.L    D6,D2
1326            SUBI.L   #4,D6
1327            ADD.L    #1,A1
1328            ADD.B    D2,D3    ;total byte stored in D3
1329            BRA      PARSELOOP
1330 ENDPARSE:
1331            MOVE.B   D3,(A4)    ;commit memory change
1332 ENTER:     ADD.L    #1,A4    ;increment address
1333            BRA      MODIFYLOOP
1334
1335 *************************************************
1336
1337 WORD:
1338
1339            ADD.L    #2,A1        ;set up for subroutine
```

```
1340            MOVE      A1,A2           ;set up for subroutine
1341            MOVEM.L   D1/D6/A1-A3,-(SP)
1342            JSR       ASCII_ADDRESS
1343            MOVEM.L   (SP)+,D1/D6/A1-A3
1344            MOVE.L    D5,A4           ;set up address to modify
1345
1346 MODIFYLOOPW:
1347            *--------Display Memory First------*
1348            MOVE.L    A4,D0
1349            DIVU      #2,D0
1350            SWAP      D0        ;check if it's an odd address
1351            CMP.W     #$00,D0
1352            BNE       ERRORSR
1353            MOVE.L    A4,D3           ;set up for subroutine
1354            MOVE.L    A4,A5           ;next byte of memory may not be
        needed
1355            ADD.L     #1,A5
1356            JSR       HEXTOASCII  ;convert new address to ascii for
        output
1357            SUBA      A2,A3           ;get num of bytes to produce
1358            MOVE.L    #1,D0
1359            MOVE.L    A3,D1
1360            MOVE.L    A2,A1
1361            TRAP      #15
1362
1363            *add colon to denote containing data*
1364            MOVE.B    #$3A,(A1)
1365            MOVE.L    #1,D1    ;display only the colon
1366            MOVE.L    #1,D0
1367            TRAP      #15
1368
1369            MOVE.B    (A4),D3
1370            JSR       HEXTOASCII
1371            MOVE.L    #2,D1
1372            SUB.L     A2,A3
1373            CMP       #2,A3
1374            BEQ       FORMATGOOD1
1375            SUB.L     #1,A2
1376 FORMATGOOD1:
1377
1378            MOVE.L    A2,A1
1379            MOVE.B    #1,D0
1380            TRAP      #15
1381
1382            MOVE.B    (A5),D3
```

```
1383            JSR         HEXTOASCII
1384            MOVE.L      #2,D1
1385            SUB.L       A2,A3
1386            CMP         #2,A3
1387            BEQ         FORMATGOOD2
1388            SUB.L       #1,A2
1389 FORMATGOOD2:
1390
1391            MOVE.L      A2,A1
1392            MOVE.B      #1,D0
1393            TRAP        #15
1394
1395
1396            MOVE.B      #$20,(A1)
1397            MOVE.L      #1,D1       ;space between held data and input
1398            MOVE.L      #1,D0
1399            TRAP        #15
1400
1401
1402            *-------Enter Input--------*
1403            CLR.L       D3
1404            MOVE.L      #12,D6
1405            LEA         BUFFER,A1   ;set up storage for command
1406            MOVE.B      #2,D0       ;load input trap call
1407            TRAP        #15
1408            CMP.B       #$2E,(A1)
1409            BEQ         ENDLP
1410            CMP.B       #$00,(A1)
1411            BEQ         ENTERW
1412
1413 PARSELOOPW:
1414            CMP.B       #$00,(A1)
1415            BEQ         ENDPARSEW
1416            CMP.B       #$40,(A1)
1417            BGT         HIGHHEXMMW
1418            SUBI.B      #$30,(A1)   ;get hex value
1419            BRA         NEXTMMSTEPW
1420 HIGHHEXMMW: SUBI.B  #$37,(A1)  ;get hex value
1421 NEXTMMSTEPW:
1422            MOVE.B      (A1),D2
1423            ROL.L       D6,D2
1424            SUBI.L      #4,D6
1425            ADD.L       #1,A1
1426            ADD.L       D2,D3       ;total byte stored in D3
1427            CLR.L       D2          ;clear for next rotate
```

```
1428            BRA         PARSELOOPW
1429 ENDPARSEW:
1430
1431            MOVE.W      D3,(A4)     ;commit memory change
1432 ENTERW:     ADD.L       #2,A4       ;increment address
1433            BRA         MODIFYLOOPW
1434
1435 *************************************************
1436
1437 LONG:
1438            ADD.L       #2,A1           ;set up for subroutine
1439            MOVE        A1,A2           ;set up for subroutine
1440            MOVEM.L     D1/D6/A1-A3,-(SP)
1441            JSR         ASCII_ADDRESS
1442            MOVEM.L     (SP)+,D1/D6/A1-A3
1443            MOVE.L      D5,A4           ;set up address to modify
1444
1445 MODIFYLOOPL:
1446            *--------Display Memory First------*
1447            MOVE.L      A4,D0
1448            DIVU        #2,D0
1449            SWAP        D0          ;check if it's an odd address
1450            CMP.W       #$00,D0
1451            BNE         ERRORSR
1452            MOVE.L      A4,D3           ;set up for subroutine
1453            MOVE.L      A4,A5           ;next byte of memory may not be
       needed
1454            ADD.L       #1,A5
1455            JSR         HEXTOASCII      ;convert new address to ascii for
       output
1456            SUBA        A2,A3           ;get num of bytes to produce
1457            MOVE.L      #1,D0
1458            MOVE.L      A3,D1
1459            MOVE.L      A2,A1
1460            TRAP        #15
1461
1462            *add colon to denote containing data*
1463            MOVE.B      #$3A,(A1)
1464            MOVE.L      #1,D1       ;display only the colon
1465            MOVE.L      #1,D0
1466            TRAP        #15
1467
1468            MOVE.B      (A4),D3
1469            JSR         HEXTOASCII
1470            MOVE.L      #2,D1
```

```
1471            SUB.L   A2,A3
1472            CMP     #2,A3
1473            BEQ     FORMATGOOD3
1474            SUB.L   #1,A2
1475 FORMATGOOD3:
1476
1477            MOVE.L  A2,A1
1478            MOVE.B  #1,D0
1479            TRAP    #15
1480
1481            MOVE.B  (A5)+,D3
1482            JSR     HEXTOASCII
1483            MOVE.L  #2,D1
1484            SUB.L   A2,A3
1485            CMP     #2,A3
1486            BEQ     FORMATGOOD4
1487            SUB.L   #1,A2
1488 FORMATGOOD4:
1489
1490            MOVE.L  A2,A1
1491            MOVE.B  #1,D0
1492            TRAP    #15
1493
1494            MOVE.B  (A5)+,D3
1495            JSR     HEXTOASCII
1496            MOVE.L  #2,D1
1497            SUB.L   A2,A3
1498            CMP     #2,A3
1499            BEQ     FORMATGOOD5
1500            SUB.L   #1,A2
1501 FORMATGOOD5:
1502
1503            MOVE.L  A2,A1
1504            MOVE.B  #1,D0
1505            TRAP    #15
1506            MOVE.B  (A5)+,D3
1507            JSR     HEXTOASCII
1508            MOVE.L  #2,D1
1509            SUB.L   A2,A3
1510            CMP     #2,A3
1511            BEQ     FORMATGOOD6
1512            SUB.L   #1,A2
1513 FORMATGOOD6:
1514
1515            MOVE.L  A2,A1
```

```
1516            MOVE.B   #1,D0
1517            TRAP     #15
```

2.2.6 Memory Set

2.2.6.1 Algorithm and Flowchart

This command is a simpler version of Memory Modify. It parses the data the user entered and stores it at one specified address. It has the syntax `MS <data> <address>`.

2.2.6.2 Assembly Code

```
 985 MEMSET:     LEA      BUFFER,A2
 986             ADD.L    #3,A2
 987             MOVE.L   A2,A3    ;set up to find end
 988 FINDEND:    CMP.B    #$00,(A3)+
 989             BEQ      MEMCONT
 990             BRA      FINDEND
 991 MEMCONT:    SUB.L    #1,A3    ;get rid of off by one erro
 992             MOVE.B   (A2)+,D1
 993             MOVE.B   (A2),D2
 994             MOVE.B   D1,D3        ;pass value to subroutine
 995             JSR      ASCII_TO_BCD
 996             MOVE.B   D3,D1        ;get converted value
 997             MOVE.B   D2,D3        ;pass value
 998             JSR      ASCII_TO_BCD
 999             MOVE.B   D3,D2        ;get returned value
1000             MOVE.B   D1,D3
1001             JSR      BCD_TO_HEX
1002             MOVE.B   D3,D1
1003             MOVE.B   D2,D3
1004             JSR      BCD_TO_HEX
1005             MOVE.B   D3,D2
1006             ROL.L    #4,D1        ;put data in correct place
1007             ADD      D1,D2        ;get combined data input
1008             ADD.L    #2,A2        ;go to start of address
1009             JSR      ASCII_ADDRESS  ;get address in workable form
1010             MOVE.L   D5,A4        ;load target address
1011             MOVE.B   D2,(A4)        ;put data in target address
1012             BRA      RESTORE                ;return to shell
```

2.2.7 Block Fill

2.2.7.1 Algorithm and Flowchart

This command requires two even addresses to be entered. It then parses the word sized data entered by the user and fills the block of memory from the first address to the second address. The syntax for this command is `BF <data> <address1> <address2>`.

2.2.7.2 Assembly Code

```
1522            TRAP    #15
1523
1524
1525            *------Enter Input--------*
1526            CLR.L   D3
1527            MOVE.L  #28,D6
1528            LEA     BUFFER,A1    ;set up storage for command
1529            MOVE.B  #2,D0        ;load input trap call
1530            TRAP    #15
1531            CMP.B   #$2E,(A1)
1532            BEQ     ENDLP
1533            CMP.B   #$00,(A1)
1534            BEQ     ENTERL
1535
1536 PARSELOOPL:
1537            CMP.B   #$00,(A1)
1538            BEQ     ENDPARSEL
1539            CMP.B   #$40,(A1)
1540            BGT     HIGHHEXMML
1541            SUBI.B  #$30,(A1)    ;get hex value
1542            BRA     NEXTMMSTEPL
1543 HIGHHEXMML: SUBI.B  #$37,(A1)   ;get hex value
1544 NEXTMMSTEPL:
1545            MOVE.B  (A1),D2
1546            ROL.L   D6,D2
1547            SUBI.L  #4,D6
1548            ADD.L   #1,A1
1549            ADD.L   D2,D3    ;total byte stored in D3
1550            CLR.L   D2       ;clear for next rotate
1551            BRA     PARSELOOPL
1552 ENDPARSEL:
1553            MOVE.L  D3,(A4)    ;commit memory change
1554 ENTERL:   ADD.L   #4,A4    ;increment address
```

```
1555            BRA       MODIFYLOOPL
1556
1557
1558 ENDLP:     BRA RESTORE
1559
1560
1561 *
```

```
1562
1563 BF:
1564          ADD.L     #1,A1     ;first byte of data
1565          MOVE.L    A1,A3     ;for end ptr
1566 BFGETENDDATA:
1567            CMP.B     #$20,(A3)+
1568            BEQ       BFNEXTADDR
1569            BRA       BFGETENDDATA
1570 BFNEXTADDR:
1571            MOVE.L    A1,A2     ;for subroutine
1572            SUB.L     #1,A3     ;off by one error
1573            JSR       ASCII_ADDRESS
1574            MOVE.L    D5,-(SP)      ;save data on the stack
```

2.2.8 Block Move

This command move a block of memory from one section to another. Both block sizes must be equal. Starting from the first address of the first block, it moves data byte by byte starting from the first address of the second block until all data has been copied. Its syntax is `BMOV <address1> <address2> <address3> <address4>`.

2.2.8.1 Algorithm and Flowchart

2.2.8.2 Assembly Code

```
1577            MOVE.L    A3,A2     ;set start ptr
1578 BFGETENDADDRONE:
1579            CMP.B     #$20,(A3)+
1580            BEQ       BFNEXTADDRTWO
1581            BRA       BFGETENDADDRONE
1582
```

```
1583 BFNEXTADDRTWO:
1584            SUB.L    #1,A3    ;off by one error
1585            JSR      ASCII_ADDRESS    ;convert address to hex
1586            MOVE.L   D5,A5        ;store address 1 in A5
1587            DIVU     #2,D5
1588            SWAP     D5
1589            CMP.W    #$00,D5
1590            BNE      ERRORSR
1591
1592            ADD.L    #1,A3    ;inc end ptr to first byte of address
1593            MOVE.L   A3,A2    ;set start ptr
1594 BFGETLASTEND:
1595            CMP.B    #$00,(A3)+
1596            BEQ      STOREDATA
1597            BRA      BFGETLASTEND
1598
1599 STOREDATA:
1600            SUB.L    #1,A3    ;off by one error
1601            JSR      ASCII_ADDRESS
1602            MOVE.L   D5,A6    ;end address in A6
1603            DIVU     #2,D5
1604            SWAP     D5
1605            CMP.W    #$00,D5
1606            BNE      ERRORSR
1607            MOVE.L   (SP)+,D5
1608
1609 DATALOOP:
1610            CMP.L    A5,A6
1611            BLT      ENDBF
1612            MOVE.W   D5,(A5)+
1613            BRA      DATALOOP
1614
1615 ENDBF:    BRA RESTORE
1616 *
```

```
1617
1618 BMOV:     ADD.L    #1,A1    ;get to start of first address
1619            MOVE.L   A1,A2    ;set up start ptr
1620            MOVE.L   A2,A3    ;set up end ptr
1621
1622 FIRSTADDRESS:
1623            CMP.B #$20,(A3)+
1624            BEQ      COMPUTEFIRSTADD
1625            BRA      FIRSTADDRESS
```

```
1626
1627 COMPUTEFIRSTADD:
1628          SUB.L    #1,A3    ;off by one error
1629          JSR      ASCII_ADDRESS
1630          MOVE.L   D5,A0    ; save 1st address
1631
1632          ADD.L    #1,A3
1633          MOVE.L   A3,A2
1634 SECONDADDRESS:
1635          CMP.B    #$20,(A3)+
1636          BEQ      COMPUTESECONDADDRESS
1637          BRA      SECONDADDRESS
1638
1639 COMPUTESECONDADDRESS:
1640          SUB.L    #1,A3    ;off by one error
1641          JSR      ASCII_ADDRESS
1642          MOVE.L   D5,A4    ;save 2nd address
1643
1644          ADD.L    #1,A3
1645          MOVE.L   A3,A2
1646 THIRDADDRESS:
1647          CMP.B    #$20,(A3)+
1648          BEQ      COMPUTETHIRDADDRESS
1649          BRA      THIRDADDRESS
1650
1651 COMPUTETHIRDADDRESS:
1652          SUB.L    #1,A3
```

2.2.9 Block Test

2.2.9.1 Algorithm and Flowchart

2.2.9.2 Assembly Code

```
1656          ADD.L    #1,A3
1657          MOVE.L   A3,A2
1658 FOURTHADDRESS:
1659          CMP.B    #$00,(A3)+
1660          BEQ      COMPUTEFOURTHADDRESS
1661          BRA      FOURTHADDRESS
1662
1663 COMPUTEFOURTHADDRESS:
```

```
1664            SUB.L    #1,A3
1665            JSR      ASCII_ADDRESS
1666            MOVE.L   D5,A6    ;save 3rd address
1667
1668
1669
1670            *Check for matching dimensions*
1671            MOVE.L   A0,D0
1672            MOVE.L   A4,D1
1673            MOVE.L   A5,D5
1674            MOVE.L   A6,D6
1675            SUB.L    D0,D1
1676            SUB.L    D5,D6
1677            CMP.L    D1,D6
1678            BNE      ERRORSR
1679            CMP.L    A0,A4
1680            BLT      ERRORSR
1681            CMP.L    A5,A6
1682            BLT      ERRORSR
1683
1684 DATATRANSFER:
1685            CMP.L    A0,A4
1686            BLT      BMOVDONE
1687            MOVE.B   (A0)+,(A5)+
1688            BRA      DATATRANSFER
1689
1690
1691
1692 BMOVDONE:
1693            BRA RESTORE
1694
1695 *-------------------------------------------------------------------------------------------------------------------------------
1696
1697 BTST:
1698          ADD.L    #1,A1    ;first byte of data
1699          MOVE.L   A1,A3    ;for end ptr
1700 BTSTGETENDDATA:
1701            CMP.B    #$20,(A3)+
1702            BEQ      BTSTNEXTADDR
1703            BRA      BTSTGETENDDATA
1704 BTSTNEXTADDR:
1705            MOVE.L   A1,A2    ;for subroutine
1706            SUB.L    #1,A3    ;off by one error
```

```
1707            JSR         ASCII_ADDRESS
1708            MOVE.L      D5,-(SP)      ;save data on the stack
1709
1710            ADD.L       #1,A3     ;inc end ptr to first byte of address
1711            MOVE.L      A3,A2     ;set start ptr
1712 BTSTGETENDADDRONE:
1713            CMP.B       #$20,(A3)+
1714            BEQ         BTSTNEXTADDRTWO
1715            BRA         BTSTGETENDADDRONE
1716
1717 BTSTNEXTADDRTWO:
1718            SUB.L       #1,A3     ;off by one error
1719            JSR         ASCII_ADDRESS    ;convert address to hex
1720            MOVE.L      D5,A5         ;store address 1 in A5
1721            MOVE.L      D5,A4         ;for second run through
1722
1723            ADD.L       #1,A3     ;inc end ptr to first byte of address
1724            MOVE.L      A3,A2     ;set start ptr
1725 BTSTGETLASTEND:
1726            CMP.B       #$00,(A3)+
1727            BEQ         STOREDATABTST
1728            BRA         BTSTGETLASTEND
1729
1730
1731 STOREDATABTST:
1732            SUB.L       #1,A3     ;off by one error
1733            JSR         ASCII_ADDRESS
1734            MOVE.L      D5,A6     ;end address in A6
1735            MOVE.L      (SP)+,D5
1736
1737 BTSTDATALOOP:
1738            CMP.L       A5,A6
1739            BLT         READ
1740            MOVE.B      D5,(A5)+
1741            BRA         BTSTDATALOOP
1742
1743
1744 READ:
1745            CMP.L       A4,A6
1746            BLT         COMPLETE
1747            CMP.B       (A4)+,D5
1748            BNE         FAIL
1749            BRA         READ
1750
1751 FAIL:
```

```
1752            LEA         BTST4,A1
1753            MOVE.L      #11,D1
1754            MOVE.L      #0,D0
1755            TRAP        #15
1756
1757            LEA         BTST1,A1
1758            MOVE.L      #1,D0
1759            MOVE.L      #20,D1
1760            TRAP        #15
1761
1762            MOVE.B      D5,D3     ;for subroutine
1763            JSR         HEXTOASCII
1764            MOVE.L          A2,A1
1765            MOVE.L      #0,D0
```

2.2.10 Block Search

2.2.10.1 Algorithm and Flowchart

2.2.10.2 Assembly Code

```
1769
1770
1771            LEA         BTST2,A1
1772            MOVE.L      #1,D0
1773            MOVE.L      #17,D1
1774            TRAP        #15
1775
1776
1777            SUB.L       #1,A4     ;go back to address that failed
1778            MOVE.B      (A4),D3
1779            JSR         HEXTOASCII  ;convert for output
1780            MOVE.L          A2,A1
1781            MOVE.L      #0,D0
1782            SUBA.L      A2,A3     ;number of bytes
1783            MOVE.L      A3,D1
1784            TRAP        #15
1785
1786            LEA         BTST5,A1
1787            MOVE.L      #27,D1
1788            MOVE.B      #1,D0
1789            TRAP        #15
```

```
1790            MOVE.L   A4,D3
1791            JSR      HEXTOASCII
1792            MOVE.L      A2,A1
1793            MOVE.L   #0,D0
1794            SUBA.L   A2,A3    ;number of bytes
1795            MOVE.L   A3,D1
1796            TRAP     #15
1797
1798
1799
1800 COMPLETE:
1801
1802            LEA      BTST3,A1
1803            MOVE.L   #18,D1
1804            MOVE.L   #0,D0
1805            TRAP     #15
1806            BRA RESTORE
1807
1808 *-------------------------------------------------------------------------------------------------------------------------------
1809
1810 BSCH:
1811            ADD.L    #1,A1    ;start of data
1812            MOVE.L   A1,A2    ;set up bac ptr
1813
1814 BSCHENDDATA:
1815            CMP.B    #$20,(A2)+
1816            BEQ      BSCHFIRSTADD
1817            BRA      BSCHENDDATA
1818
1819
1820 BSCHFIRSTADD:
1821            SUB.L    #1,A2
1822            MOVE.L   A2,A3
1823            MOVE.L   A1,A2
1824            JSR      ASCII_ADDRESS
1825            SUB.L    A1,A3    ;see how many bytes
1826            MOVE.L   A3,D6    ;store byte/word/long in D6
1827            ADD.L    #1,A2    ;set up for start of next address
1828            MOVE.L   A2,A3    ;set up for end ptr
1829            MOVE.L   D5,-(SP)     ;save data to stack
1830
1831
1832 BSCHFADDEND:
```

```
1833            CMP.B   #$20,(A3)+
1834            BEQ     BSCHSECONDADD
1835            BRA     BSCHFADDEND
1836
1837
1838 BSCHSECONDADD:
1839            SUB.L   #1,A3    ;off by one
1840            JSR     ASCII_ADDRESS
1841            MOVE.L  D5,A5    ;first address destination
1842            ADD.L   #1,A3    ;start it at next address
1843            MOVE.L  A3,A2    ; set up for next address
1844
1845
1846 BSCHSECONDFIND:
1847            CMP.B   #$00,(A3)+
1848            BEQ     TESTOP
1849            BRA     BSCHSECONDFIND
1850
1851
1852 TESTOP:
1853            SUB.L   #1,A3    ;off by one
1854            JSR     ASCII_ADDRESS
1855            MOVE.L  D5,A6    ;end address at A6
1856            MOVE.L  (SP)+,D5     ;restore data
1857            CMP.B   #2,D6
1858            BEQ     BYTEBSCH
1859            CMP.B   #4,D6
1860            BEQ     WORDBSCH
1861            CMP.B   #8,D6
1862            BEQ     LONGBSCH
1863            BRA     ERRORSR
1864
1865 BYTEBSCH:
1866            CMP.L   A5,A6
1867            BLT     ENDBSCH
1868            CMP.B   (A5)+,D5
1869            BEQ     FOUNDB
1870            BRA     BYTEBSCH
1871
1872 WORDBSCH:
1873            CMP.L   A5,A6
1874            BLT     ENDBSCH
1875            CMP.W   (A5)+,D5
1876            BEQ     FOUNDW
1877            BRA     WORDBSCH
```

```
1878
1879 LONGBSCH:
1880             CMP.L     A5,A6
1881             BLT       ENDBSCH
1882             CMP.L     (A5)+,D5
1883             BEQ       FOUNDL
1884             BRA       LONGBSCH
```

2.2.11 Go

2.2.11.1 Algorithm and Flowchart

2.2.11.2 Assembly Code

```
1891             BRA       SUCCESSTEXT
1892 FOUNDW:
1893             SUB.L     #2,A5
1894             MOVE.W    (A5),D3
1895             BRA       SUCCESSTEXT
1896 FOUNDL:
1897             SUB.L     #4,A5
1898             MOVE.L    (A5),D3
1899
1900 SUCCESSTEXT:
1901             LEA BSCH1,A1
1902             MOVE.L    #6,D1
1903             MOVE.L    #1,D0
1904             TRAP      #15
```

2.2.12 Display Formatted Registers

2.2.12.1 Algorithm and Flowchart

2.2.12.2 Assembly Code

```
1909             MOVE.L    A3,D1     ;how many bytes
1910             MOVE.L    #0,D0
1911             TRAP      #15
```

```
        LEA BSCH2,A1
        MOVE.L  #18,D1
        MOVE.L  #1,D0
        TRAP    #15

        MOVE.L  A5,D3
        JSR     HEXTOASCII
        MOVE.L  A2,A1
        SUB.L   A2,A3
        MOVE.L  A3,D1   ;how many bytes
        MOVE.L  #0,D0
        TRAP    #15


ENDBSCH:
        BRA RESTORE

* ------------------------------------------------------------------------------------------

GO:
        MOVE.L  A1,A2   ;setup for hex conversion
        MOVE.L  A2,A3
GGETEND:
        CMP.B   #$00,(A3)+
        BEQ     EXECUTE
        BRA     GGETEND

EXECUTE:
        SUB.L   #1,A3   ;off by one error
        JSR     ASCII_ADDRESS
        MOVE.L  D5,A0
        JSR     (A0)    ;go to program
        **NOTE: THE PROGRAM MUST HAVE RTS OR CONTROL WILL NOT BE RETURNED BACK TO MONITOR441!!!**
        BRA RESTORE

* ------------------------------------------------------------------------------------------

DF:     *Registers have already been saved to STACK, just need to pop them off first*
```

```
1951          *Stack looks like this*
1952
1953          *---------------*
1954          *|D0-D7/A0-A6|*
1955          *|      USP    |*
1956          *|      SR     |*
1957          *|      SSP    |*
1958          *|      PC     |*
1959          *---------------*
1960          *I should've used loops for efficiency but runtime is
          not a design constraint*
1961          *Maybe fix this in the future?*
1962
1963             *----------D0----------*
1964          LEA         RD0,A1
1965          MOVE.L      #4,D1
1966          MOVE.L      #1,D0
1967          TRAP        #15
1968          MOVE.L      (SP)+,D3
1969          JSR         HEXTOASCII
1970          MOVE.L      A2,A1
1971          SUB.L       A3,A2
1972          MOVE.L      A2,D2
1973          CMP.L       #-8,D2
1974          BEQ         D0DONTWORRY
1975 D0ACCOUNTFORZEROS:
1976            ADDI.L    #8,D2
1977            SUB.L     D2,A1
1978 D0DONTWORRY:
1979          MOVE.L      #0,D0
1980          MOVE.L      #8,D1
1981          TRAP        #15
1982
1983             *----------D1----------*
1984          LEA         RD1,A1
1985          MOVE.L      #4,D1
1986          MOVE.L      #1,D0
1987          TRAP        #15
1988          MOVE.L      (SP)+,D3
1989          JSR         HEXTOASCII
1990          MOVE.L      A2,A1
1991          SUB.L       A3,A2
1992          MOVE.L      A2,D2
1993          CMP.L       #-8,D2
1994          BEQ         D1DONTWORRY
```

```
1995 D1ACCOUNTFORZEROS:
1996             ADDI.L    #8,D2
1997             SUB.L     D2,A1
1998 D1DONTWORRY:
1999             MOVE.L    #0,D0
2000             MOVE.L    #8,D1
2001             TRAP      #15
2002
2003                 *-----------D2-----------*
2004             LEA       RD2,A1
2005             MOVE.L    #4,D1
2006             MOVE.L    #1,D0
2007             TRAP      #15
2008             MOVE.L    (SP)+,D3
2009             JSR       HEXTOASCII
2010             MOVE.L    A2,A1
2011             SUB.L     A3,A2
2012             MOVE.L    A2,D2
2013             CMP.L     #-8,D2
2014             BEQ       D2DONTWORRY
2015 D2ACCOUNTFORZEROS:
2016             ADDI.L    #8,D2
2017             SUB.L     D2,A1
2018 D2DONTWORRY:
2019             MOVE.L    #0,D0
2020             MOVE.L    #8,D1
2021             TRAP      #15
2022
2023                 *-----------D3-----------*
2024             LEA       RD3,A1
2025             MOVE.L    #4,D1
2026             MOVE.L    #1,D0
2027             TRAP      #15
2028             MOVE.L    (SP)+,D3
2029             JSR       HEXTOASCII
2030             MOVE.L    A2,A1
2031             SUB.L     A3,A2
2032             MOVE.L    A2,D2
2033             CMP.L     #-8,D2
2034             BEQ       D3DONTWORRY
2035 D3ACCOUNTFORZEROS:
2036             ADDI.L    #8,D2
2037             SUB.L     D2,A1
2038 D3DONTWORRY:
2039             MOVE.L    #0,D0
```

```
2040            MOVE.L      #8,D1
2041            TRAP        #15
2042
2043                    *-------------D4-------------*
2044            LEA         RD4,A1
2045            MOVE.L      #4,D1
2046            MOVE.L      #1,D0
2047            TRAP        #15
2048            MOVE.L      (SP)+,D3
2049            JSR         HEXTOASCII
2050            MOVE.L      A2,A1
2051            SUB.L       A3,A2
2052            MOVE.L      A2,D2
2053            CMP.L       #-8,D2
2054            BEQ         D4DONTWORRY
2055 D4ACCOUNTFORZEROS:
2056              ADDI.L    #8,D2
2057              SUB.L     D2,A1
2058 D4DONTWORRY:
2059            MOVE.L      #0,D0
2060            MOVE.L      #8,D1
2061            TRAP        #15
2062
2063             *-------------D5-------------*
2064            LEA         RD5,A1
2065            MOVE.L      #4,D1
2066            MOVE.L      #1,D0
2067            TRAP        #15
2068            MOVE.L      (SP)+,D3
2069            JSR         HEXTOASCII
2070            MOVE.L      A2,A1
2071            SUB.L       A3,A2
2072            MOVE.L      A2,D2
2073            CMP.L       #-8,D2
2074            BEQ         D5DONTWORRY
2075 D5ACCOUNTFORZEROS:
2076              ADDI.L    #8,D2
2077              SUB.L     D2,A1
2078 D5DONTWORRY:
2079            MOVE.L      #0,D0
2080            MOVE.L      #8,D1
2081            TRAP        #15
2082
2083              *-------------D6-------------*
2084            LEA         RD6,A1
```

```
2085            MOVE.L      #4,D1
2086            MOVE.L      #1,D0
2087            TRAP        #15
2088            MOVE.L      (SP)+,D3
2089            JSR         HEXTOASCII
2090            MOVE.L      A2,A1
2091            SUB.L       A3,A2
2092            MOVE.L      A2,D2
2093            CMP.L       #-8,D2
2094            BEQ         D6DONTWORRY
2095 D6ACCOUNTFORZEROS:
2096              ADDI.L    #8,D2
2097              SUB.L     D2,A1
2098 D6DONTWORRY:
2099            MOVE.L      #0,D0
2100            MOVE.L      #8,D1
2101            TRAP        #15
2102
2103              *-----------D7-----------*
2104            LEA         RD7,A1
2105            MOVE.L      #4,D1
2106            MOVE.L      #1,D0
2107            TRAP        #15
2108            MOVE.L      (SP)+,D3
2109            JSR         HEXTOASCII
2110            MOVE.L      A2,A1
2111            SUB.L       A3,A2
2112            MOVE.L      A2,D2
2113            CMP.L       #-8,D2
2114            BEQ         D7DONTWORRY
2115 D7ACCOUNTFORZEROS:
2116              ADDI.L    #8,D2
2117              SUB.L     D2,A1
2118 D7DONTWORRY:
2119            MOVE.L      #0,D0
2120            MOVE.L      #8,D1
2121            TRAP        #15
2122
2123     *-----------A0-----------*
2124            LEA         RA0,A1
2125            MOVE.L      #4,D1
2126            MOVE.L      #1,D0
2127            TRAP        #15
2128            MOVE.L      (SP)+,D3
2129            JSR         HEXTOASCII
```

```
2130            MOVE.L      A2,A1
2131            SUB.L       A3,A2
2132            MOVE.L      A2,D2
2133            CMP.L       #-8,D2
2134            BEQ         A0DONTWORRY
2135 A0ACCOUNTFORZEROS:
2136              ADDI.L    #8,D2
2137              SUB.L     D2,A1
2138 A0DONTWORRY:
2139            MOVE.L      #0,D0
2140            MOVE.L      #8,D1
2141            TRAP        #15
2142
2143                *-----------A1-----------*
2144            LEA         RA1,A1
2145            MOVE.L      #4,D1
2146            MOVE.L      #1,D0
2147            TRAP        #15
2148            MOVE.L      (SP)+,D3
2149            JSR         HEXTOASCII
2150            MOVE.L      A2,A1
2151            SUB.L       A3,A2
2152            MOVE.L      A2,D2
2153            CMP.L       #-8,D2
2154            BEQ         A1DONTWORRY
2155 A1ACCOUNTFORZEROS:
2156              ADDI.L    #8,D2
2157              SUB.L     D2,A1
2158 A1DONTWORRY:
2159            MOVE.L      #0,D0
2160            MOVE.L      #8,D1
2161            TRAP        #15
2162
2163              *-----------A2-----------*
2164            LEA         RA2,A1
2165            MOVE.L      #4,D1
2166            MOVE.L      #1,D0
2167            TRAP        #15
2168            MOVE.L      (SP)+,D3
2169            JSR         HEXTOASCII
2170            MOVE.L      A2,A1
2171            SUB.L       A3,A2
2172            MOVE.L      A2,D2
2173            CMP.L       #-8,D2
2174            BEQ         A2DONTWORRY
```

```
2175 A2ACCOUNTFORZEROS:
2176             ADDI.L    #8,D2
2177             SUB.L     D2,A1
2178 A2DONTWORRY:
2179             MOVE.L    #0,D0
2180             MOVE.L    #8,D1
2181             TRAP      #15
2182
2183               *-----------A3-----------*
2184             LEA       RA3,A1
2185             MOVE.L    #4,D1
2186             MOVE.L    #1,D0
2187             TRAP      #15
2188             MOVE.L    (SP)+,D3
2189             JSR       HEXTOASCII
2190             MOVE.L    A2,A1
2191             SUB.L     A3,A2
2192             MOVE.L    A2,D2
2193             CMP.L     #-8,D2
2194             BEQ       A3DONTWORRY
2195 A3ACCOUNTFORZEROS:
2196             ADDI.L    #8,D2
2197             SUB.L     D2,A1
2198 A3DONTWORRY:
2199             MOVE.L    #0,D0
2200             MOVE.L    #8,D1
2201             TRAP      #15
2202
2203               *-----------A4-----------*
2204             LEA       RA3,A1
2205             MOVE.L    #4,D1
2206             MOVE.L    #1,D0
2207             TRAP      #15
2208             MOVE.L    (SP)+,D3
2209             JSR       HEXTOASCII
2210             MOVE.L    A2,A1
2211             SUB.L     A3,A2
2212             MOVE.L    A2,D2
2213             CMP.L     #-8,D2
2214             BEQ       A4DONTWORRY
2215 A4ACCOUNTFORZEROS:
2216             ADDI.L    #8,D2
2217             SUB.L     D2,A1
2218 A4DONTWORRY:
2219             MOVE.L    #0,D0
```

```
2220            MOVE.L      #8,D1
2221            TRAP        #15
2222
2223            *-----------A5-----------*
2224            LEA         RA3,A1
2225            MOVE.L      #4,D1
2226            MOVE.L      #1,D0
2227            TRAP        #15
2228            MOVE.L      (SP)+,D3
2229            JSR         HEXTOASCII
2230            MOVE.L      A2,A1
2231            SUB.L       A3,A2
2232            MOVE.L      A2,D2
2233            CMP.L       #-8,D2
2234            BEQ         A5DONTWORRY
2235 A5ACCOUNTFORZEROS:
2236              ADDI.L    #8,D2
2237              SUB.L     D2,A1
2238 A5DONTWORRY:
2239            MOVE.L      #0,D0
2240            MOVE.L      #8,D1
2241            TRAP        #15
2242
2243            *-----------A6-----------*
2244            LEA         RA3,A1
2245            MOVE.L      #4,D1
2246            MOVE.L      #1,D0
2247            TRAP        #15
2248            MOVE.L      (SP)+,D3
2249            JSR         HEXTOASCII
2250            MOVE.L      A2,A1
2251            SUB.L       A3,A2
2252            MOVE.L      A2,D2
2253            CMP.L       #-8,D2
2254            BEQ         A6DONTWORRY
2255 A6ACCOUNTFORZEROS:
2256              ADDI.L    #8,D2
2257              SUB.L     D2,A1
2258 A6DONTWORRY:
2259            MOVE.L      #0,D0
2260            MOVE.L      #8,D1
2261            TRAP        #15
2262          *-----HACK---*
2263          ADD.L   #60,SP   ;should put stack in correct place
2264
```

```
2265                    *-------------USP--------------*
2266            LEA         RUS,A1
2267            MOVE.L      #4,D1
2268            MOVE.L      #1,D0
2269            TRAP        #15
2270            MOVE.L      (SP)+,D3
2271            JSR         HEXTOASCII
2272            MOVE.L      A2,A1
2273            SUB.L       A3,A2
2274            MOVE.L      A2,D2
2275            CMP.L       #-8,D2
2276            BEQ         USPDONTWORRY
2277 USPACCOUNTFORZEROS:
2278              ADDI.L    #8,D2
2279              SUB.L     D2,A1
2280 USPDONTWORRY:
2281            MOVE.L      #0,D0
2282            MOVE.L      #8,D1
2283            TRAP        #15
2284
2285                    *-------------SR--------------*
2286            LEA         RSR,A1
2287            MOVE.L      #4,D1
2288            MOVE.L      #1,D0
2289            TRAP        #15
2290            MOVE.W      (SP)+,D3
2291            MOVE.W      D3,D7    ;temp storage to restore before return
2292            JSR         HEXTOASCII
2293            MOVE.L      A2,A1
2294            SUB.L       A3,A2
2295            MOVE.L      A2,D2
2296            CMP.L       #-4,D2
2297            BEQ         SRDONTWORRY
2298 SRACCOUNTFORZEROS:
2299              ADDI.L    #4,D2
2300              SUB.L     D2,A1
2301 SRDONTWORRY:
2302            MOVE.L      #0,D0
2303            MOVE.L      #4,D1
2304            TRAP        #15
2305
2306        *-------------SS/A7--------------*
2307            LEA         RSS,A1
2308            MOVE.L      #7,D1
2309            MOVE.L      #1,D0
```

```
2310            TRAP        #15
2311            MOVE.L      (SP)+,D3
2312            JSR         HEXTOASCII
```

2.2.13 Modify Register

2.2.13.1 Algorithm and Flowchart

2.2.13.2 Assembly Code

```
413 MODIFYREGS:
414
415 MRD:
416             ADD.L       #1,A1       ;inc
417             CMP.B       #$30,(A1)
418             BEQ         MRD0
419             CMP.B       #$31,(A1)
420             BEQ         MRD1
421             CMP.B       #$32,(A1)
422             BEQ         MRD2
423             CMP.B       #$33,(A1)
424             BEQ         MRD3
425             CMP.B       #$34,(A1)
426             BEQ         MRD4
427             CMP.B       #$35,(A1)
428             BEQ         MRD5
429             CMP.B       #$36,(A1)
430             BEQ         MRD6
431             CMP.B       #$37,(A1)
432             BEQ         MRD7
433             BRA         ERRORSR
434
435 MRA:
436             ADD.L       #1,A1       ;inc
437             CMP.B       #$30,(A1)
438             BEQ         MRA0
439             CMP.B       #$31,(A1)
440             BEQ         MRA1
441             CMP.B       #$32,(A1)
442             BEQ         MRA2
443             CMP.B       #$33,(A1)
444             BEQ         MRA3
```

```
445             CMP.B   #$34,(A1)
446             BEQ     MRA4
447             CMP.B   #$35,(A1)
448             BEQ     MRA5
449             CMP.B   #$36,(A1)
450             BEQ     MRA6
451             BRA     ERRORSR
452
453
454
455
456
457 MRD0:
458             ADD.L   #1,A1
459             CMP.B   #$20,(A1)+
460             BNE     ERRORSR
461             MOVE.L  A1,A2
462             MOVE.L  A2,A3
463             JSR     MRDFINDDATA
464             SUB.L   #1,A3
465             JSR     ASCII_ADDRESS   ;convert data to hex
466             MOVE.L  D5,-(SP)        ;store it temporarily
467             ADD.L   #4,SP       ;dont lose data
468             MOVEM.L (SP)+,D0-D7/A0-A6
469             MOVEM.L (SP)+,D0-D7/A0-A6 ;double restore because of DF
    hack workaround
470             ADD.L   #4,SP       ;account for USP, it'll fix itself (
    it shouldn't be used)
471                                 ;EASY68k simulator starts in
    supervisor mode
472             MOVE    (SP)+,SR
473             ADD.L   #4,SP       ;skip saved stack
474             SUB.L   #134,SP     ;find data again
475             MOVE.L  (SP),D0
476             ADD.L   #138,SP     ;go back to original spot
477             BRA     SHELL
478
479 MRD1:
480             ADD.L   #1,A1
481             CMP.B   #$20,(A1)+
482             BNE     ERRORSR
483             MOVE.L  A1,A2
484             MOVE.L  A2,A3
485             JSR     MRDFINDDATA
486             SUB.L   #1,A3
```

```
487         JSR     ASCII_ADDRESS   ;convert data to hex
488         MOVE.L  D5,-(SP)        ;store it temporarily
489         ADD.L   #4,SP       ;dont lose data
490         MOVEM.L (SP)+,D0-D7/A0-A6
491         MOVEM.L (SP)+,D0-D7/A0-A6 ;double restore because of DF
    hack workaround
492         ADD.L   #4,SP       ;account for USP, it'll fix itself (
    it shouldn't be used)
493                                 ;EASY68k simulator starts in
    supervisor mode
494         MOVE    (SP)+,SR
495         ADD.L   #4,SP       ;skip saved stack
496         SUB.L   #134,SP     ;find data again
497         MOVE.L  (SP),D1
498         ADD.L   #138,SP     ;go back to original spot
499         BRA     SHELL
500
501 MRD2:
502         ADD.L   #1,A1
503         CMP.B   #$20,(A1)+
504         BNE     ERRORSR
505         MOVE.L  A1,A2
506         MOVE.L  A2,A3
507         JSR     MRDFINDDATA
508         SUB.L   #1,A3
509         JSR     ASCII_ADDRESS   ;convert data to hex
510         MOVE.L  D5,-(SP)        ;store it temporarily
511         ADD.L   #4,SP       ;dont lose data
512         MOVEM.L (SP)+,D0-D7/A0-A6
513         MOVEM.L (SP)+,D0-D7/A0-A6 ;double restore because of DF
    hack workaround
514         ADD.L   #4,SP       ;account for USP, it'll fix itself (
    it shouldn't be used)
515                                 ;EASY68k simulator starts in
    supervisor mode
516         MOVE    (SP)+,SR
517         ADD.L   #4,SP       ;skip saved stack
518         SUB.L   #134,SP     ;find data again
519         MOVE.L  (SP),D2
520         ADD.L   #138,SP     ;go back to original spot
521         BRA     SHELL
522
523 MRD3:
524         ADD.L   #1,A1
525         CMP.B   #$20,(A1)+
```

```
526         BNE     ERRORSR
527         MOVE.L  A1,A2
528         MOVE.L  A2,A3
529         JSR     MRDFINDDATA
530         SUB.L   #1,A3
531         JSR     ASCII_ADDRESS   ;convert data to hex
532         MOVE.L  D5,-(SP)        ;store it temporarily
533         ADD.L   #4,SP       ;dont lose data
534         MOVEM.L (SP)+,D0-D7/A0-A6
535         MOVEM.L (SP)+,D0-D7/A0-A6 ;double restore because of DF
    hack workaround
536         ADD.L   #4,SP       ;account for USP, it'll fix itself (
    it shouldn't be used)
537                                 ;EASY68k simulator starts in
    supervisor mode
538         MOVE    (SP)+,SR
539         ADD.L   #4,SP       ;skip saved stack
540         SUB.L   #134,SP     ;find data again
541         MOVE.L  (SP),D3
542         ADD.L   #138,SP     ;go back to original spot
543         BRA     SHELL
544
545 MRD4:
546         ADD.L   #1,A1
547         CMP.B   #$20,(A1)+
548         BNE     ERRORSR
549         MOVE.L  A1,A2
550         MOVE.L  A2,A3
551         JSR     MRDFINDDATA
552         SUB.L   #1,A3
553         JSR     ASCII_ADDRESS   ;convert data to hex
554         MOVE.L  D5,-(SP)        ;store it temporarily
555         ADD.L   #4,SP       ;dont lose data
556         MOVEM.L (SP)+,D0-D7/A0-A6
557         MOVEM.L (SP)+,D0-D7/A0-A6 ;double restore because of DF
    hack workaround
558         ADD.L   #4,SP       ;account for USP, it'll fix itself (
    it shouldn't be used)
559                                 ;EASY68k simulator starts in
    supervisor mode
560         MOVE    (SP)+,SR
561         ADD.L   #4,SP       ;skip saved stack
562         SUB.L   #134,SP     ;find data again
563         MOVE.L  (SP),D4
564         ADD.L   #138,SP     ;go back to original spot
```

```
565         BRA     SHELL
566
567 MRD5:
568         ADD.L   #1,A1
569         CMP.B   #$20,(A1)+
570         BNE     ERRORSR
571         MOVE.L  A1,A2
572         MOVE.L  A2,A3
573         JSR     MRDFINDDATA
574         SUB.L   #1,A3
575         JSR     ASCII_ADDRESS   ;convert data to hex
576         MOVE.L  D5,-(SP)        ;store it temporarily
577         ADD.L   #4,SP       ;dont lose data
578         MOVEM.L (SP)+,D0-D7/A0-A6
579         MOVEM.L (SP)+,D0-D7/A0-A6 ;double restore because of DF
    hack workaround
580         ADD.L   #4,SP       ;account for USP, it'll fix itself (
    it shouldn't be used)
581                                 ;EASY68k simulator starts in
    supervisor mode
582         MOVE    (SP)+,SR
583         ADD.L   #4,SP       ;skip saved stack
584         SUB.L   #134,SP     ;find data again
585         MOVE.L  (SP),D5
586         ADD.L   #138,SP     ;go back to original spot
587         BRA     SHELL
588
589 MRD6:
590         ADD.L   #1,A1
591         CMP.B   #$20,(A1)+
592         BNE     ERRORSR
593         MOVE.L  A1,A2
594         MOVE.L  A2,A3
595         JSR     MRDFINDDATA
596         SUB.L   #1,A3
597         JSR     ASCII_ADDRESS   ;convert data to hex
598         MOVE.L  D5,-(SP)        ;store it temporarily
599         ADD.L   #4,SP       ;dont lose data
600         MOVEM.L (SP)+,D0-D7/A0-A6
601         MOVEM.L (SP)+,D0-D7/A0-A6 ;double restore because of DF
    hack workaround
602         ADD.L   #4,SP       ;account for USP, it'll fix itself (
    it shouldn't be used)
603                                 ;EASY68k simulator starts in
    supervisor mode
```

```
604             MOVE    (SP)+,SR
605             ADD.L   #4,SP           ;skip saved stack
606             SUB.L   #134,SP         ;find data again
607             MOVE.L  (SP),D6
608             ADD.L   #138,SP         ;go back to original spot
609             BRA     SHELL
610
611 MRD7:
612             ADD.L   #1,A1
613             CMP.B   #$20,(A1)+
614             BNE     ERRORSR
615             MOVE.L  A1,A2
616             MOVE.L  A2,A3
617             JSR     MRDFINDDATA
618             SUB.L   #1,A3
619             JSR     ASCII_ADDRESS   ;convert data to hex
620             MOVE.L  D5,-(SP)        ;store it temporarily
621             ADD.L   #4,SP           ;dont lose data
622             MOVEM.L (SP)+,D0-D7/A0-A6
623             MOVEM.L (SP)+,D0-D7/A0-A6 ;double restore because of DF
        hack workaround
624             ADD.L   #4,SP           ;account for USP, it'll fix itself (
        it shouldn't be used)
625                                     ;EASY68k simulator starts in
        supervisor mode
626             MOVE    (SP)+,SR
627            ADD.L   #4,SP           ;skip saved stack
628             SUB.L   #134,SP         ;find data again
629             MOVE.L  (SP),D7
630             ADD.L   #138,SP         ;go back to original spot
631             BRA     SHELL
632
633 MRA0:
634             ADD.L   #1,A1
635             CMP.B   #$20,(A1)+
636             BNE     ERRORSR
637             MOVE.L  A1,A2
638             MOVE.L  A2,A3
639             JSR     MRDFINDDATA
640             SUB.L   #1,A3
641             JSR     ASCII_ADDRESS   ;convert data to hex
642             MOVE.L  D5,-(SP)        ;store it temporarily
643             ADD.L   #4,SP           ;dont lose data
644             MOVEM.L (SP)+,D0-D7/A0-A6
```

```
645         MOVEM.L (SP)+,D0-D7/A0-A6 ;double restore because of DF
    hack workaround
646         ADD.L   #4,SP       ;account for USP, it'll fix itself (
    it shouldn't be used)
647                                 ;EASY68k simulator starts in
    supervisor mode
648         MOVE    (SP)+,SR
649         ADD.L   #4,SP       ;skip saved stack
650         SUB.L   #134,SP     ;find data again
651         MOVE.L  (SP),A0
652         ADD.L   #138,SP     ;go back to original spot
653         BRA     SHELL
654 MRA1:
655         ADD.L   #1,A1
656         CMP.B   #$20,(A1)+
657         BNE     ERRORSR
658         MOVE.L  A1,A2
659         MOVE.L  A2,A3
660         JSR     MRDFINDDATA
661         SUB.L   #1,A3
662         JSR     ASCII_ADDRESS   ;convert data to hex
663         MOVE.L  D5,-(SP)        ;store it temporarily
664         ADD.L   #4,SP       ;dont lose data
665         MOVEM.L (SP)+,D0-D7/A0-A6
666         MOVEM.L (SP)+,D0-D7/A0-A6 ;double restore because of DF
    hack workaround
667         ADD.L   #4,SP       ;account for USP, it'll fix itself (
    it shouldn't be used)
668                                 ;EASY68k simulator starts in
    supervisor mode
669         MOVE    (SP)+,SR
670        ADD.L   #4,SP       ;skip saved stack
671         SUB.L   #134,SP     ;find data again
672         MOVE.L  (SP),A1
673         ADD.L   #138,SP     ;go back to original spot
674         BRA     SHELL
675
676 MRA2:
677         ADD.L   #1,A1
678         CMP.B   #$20,(A1)+
679         BNE     ERRORSR
680         MOVE.L  A1,A2
681         MOVE.L  A2,A3
682         JSR     MRDFINDDATA
683         SUB.L   #1,A3
```

```
684         JSR     ASCII_ADDRESS   ;convert data to hex
685         MOVE.L  D5,-(SP)        ;store it temporarily
686         ADD.L   #4,SP       ;dont lose data
687         MOVEM.L (SP)+,D0-D7/A0-A6
688         MOVEM.L (SP)+,D0-D7/A0-A6 ;double restore because of DF
    hack workaround
689         ADD.L   #4,SP       ;account for USP, it'll fix itself (
    it shouldn't be used)
690                                 ;EASY68k simulator starts in
    supervisor mode
691         MOVE    (SP)+,SR
692         ADD.L   #4,SP       ;skip saved stack
693         SUB.L   #134,SP     ;find data again
694         MOVE.L  (SP),A2
695         ADD.L   #138,SP     ;go back to original spot
696         BRA     SHELL
697
698 MRA3:
699         ADD.L   #1,A1
700         CMP.B   #$20,(A1)+
701         BNE     ERRORSR
702         MOVE.L  A1,A2
703         MOVE.L  A2,A3
704         JSR     MRDFINDDATA
705         SUB.L   #1,A3
706         JSR     ASCII_ADDRESS   ;convert data to hex
707         MOVE.L  D5,-(SP)        ;store it temporarily
708         ADD.L   #4,SP       ;dont lose data
709         MOVEM.L (SP)+,D0-D7/A0-A6
710         MOVEM.L (SP)+,D0-D7/A0-A6 ;double restore because of DF
    hack workaround
711         ADD.L   #4,SP       ;account for USP, it'll fix itself (
    it shouldn't be used)
712                                 ;EASY68k simulator starts in
    supervisor mode
713         MOVE    (SP)+,SR
714         ADD.L   #4,SP       ;skip saved stack
715         SUB.L   #134,SP     ;find data again
716         MOVE.L  (SP),A3
717         ADD.L   #138,SP     ;go back to original spot
718         BRA     SHELL
719
720 MRA4:
721         ADD.L   #1,A1
722         CMP.B   #$20,(A1)+
```

```
723         BNE     ERRORSR
724         MOVE.L  A1,A2
725         MOVE.L  A2,A3
726         JSR     MRDFINDDATA
727         SUB.L   #1,A3
728         JSR     ASCII_ADDRESS   ;convert data to hex
729         MOVE.L  D5,-(SP)        ;store it temporarily
730         ADD.L   #4,SP       ;dont lose data
731         MOVEM.L (SP)+,D0-D7/A0-A6
732         MOVEM.L (SP)+,D0-D7/A0-A6 ;double restore because of DF
    hack workaround
733         ADD.L   #4,SP       ;account for USP, it'll fix itself (
    it shouldn't be used)
734                                 ;EASY68k simulator starts in
    supervisor mode
735         MOVE    (SP)+,SR
736         ADD.L   #4,SP       ;skip saved stack
737         SUB.L   #134,SP     ;find data again
738         MOVE.L  (SP),A4
739         ADD.L   #138,SP     ;go back to original spot
740         BRA     SHELL
741
742 MRA5:
743         ADD.L   #1,A1
744         CMP.B   #$20,(A1)+
745         BNE     ERRORSR
746         MOVE.L  A1,A2
747         MOVE.L  A2,A3
748         JSR     MRDFINDDATA
749         SUB.L   #1,A3
750         JSR     ASCII_ADDRESS   ;convert data to hex
751         MOVE.L  D5,-(SP)        ;store it temporarily
752         ADD.L   #4,SP       ;dont lose data
753         MOVEM.L (SP)+,D0-D7/A0-A6
754         MOVEM.L (SP)+,D0-D7/A0-A6 ;double restore because of DF
    hack workaround
755         ADD.L   #4,SP       ;account for USP, it'll fix itself (
    it shouldn't be used)
756                                 ;EASY68k simulator starts in
    supervisor mode
757         MOVE    (SP)+,SR
758        ADD.L   #4,SP       ;skip saved stack
759         SUB.L   #134,SP     ;find data again
760         MOVE.L  (SP),A5
761         ADD.L   #138,SP     ;go back to original spot
```

```
762             BRA         SHELL
763
764 MRA6:
765             ADD.L       #1,A1
766             CMP.B       #$20,(A1)+
767             BNE         ERRORSR
768             MOVE.L      A1,A2
769             MOVE.L      A2,A3
770             JSR         MRDFINDDATA
771             SUB.L       #1,A3
772             JSR         ASCII_ADDRESS    ;convert data to hex
773             MOVE.L      D5,-(SP)         ;store it temporarily
774             ADD.L       #4,SP        ;dont lose data
775             MOVEM.L     (SP)+,D0-D7/A0-A6
776             MOVEM.L     (SP)+,D0-D7/A0-A6 ;double restore because of DF
        hack workaround
777             ADD.L       #4,SP        ;account for USP, it'll fix itself (
        it shouldn't be used)
778                                      ;EASY68k simulator starts in
        supervisor mode
779             MOVE        (SP)+,SR
780             ADD.L       #4,SP        ;skip saved stack
781             SUB.L       #134,SP      ;find data again
782             MOVE.L      (SP),A6
783             ADD.L       #138,SP      ;go back to original spot
784             BRA         SHELL
785
786 MRDFINDDATA:
787             CMP.B       #$00,(A3)+
788             BEQ         GOBACK
789             BRA         MRDFINDDATA
790 GOBACK:    RTS
791
792
793             BRA RESTORE
```

2.2.14 Echo

2.2.14.1 Algorithm and Flowchart

2.2.14.2 Assembly Code

```
398 ECHO: *What terminal DOESN'T have echo?*
399
400         MOVE.L  A1,A2    ;setup to find end of string
401 EEND:   CMP.B   #$00,(A2)+
402         BEQ     EFOUND
403         BRA     EEND
404 EFOUND:
405         SUB.L   #1,A2    ;off by one
406         SUB.L   A1,A2    ;find out how many bytes
407         MOVE.L  A2,D1    ;place it for trap function
408         MOVE.L  #0,D0
409         TRAP    #15
410
411         BRA RESTORE
```

2.3 Subroutines

2.3.1 Hexadecimal to ASCII

2.3.1.1 Algorithm

2.3.1.2 Assembly Code

```
2516 BCD_TO_HEX:     *Number passed via D3 accepts BYTE ONLY*
2517                 MOVE.L  D3,D4
2518                 MOVE.L  D3,D5
2519                 ANDI.L  #240,D4 ;upper byte
2520                 ANDI.L  #15,D5  ;lower byte
2521                 ROR.L   #4,D4   ;get bits into correct place
2522                 MULU    #10,D4  ;multiply by its tens place
2523                 CLR.L   D3
2524                 ADD.L   D4,D3
2525                 ADD.L   D5,D3
2526                 RTS
2527 *
```

```
2528 ASCII_ADDRESS:   *Address to be converted from ascii to hex
     passed through A2 and A3*
2529                  *Returned in D5
                      *
2530                 CLR.L   D3
```

```
2531                    CLR.L    D5
2532                    MOVE.L    A2,D1
2533                    MOVE.L    A3,D0
2534                    SUB.L    D1,D0    ;store the difference in D0
2535                    MOVE.L   #0,D4    ;set up 10's place counter
2536                    SUBI.L   #1,D0
2537 PLACECOUNTER:      CMP      #0,D0
2538                    BEQ      CONVERTADDRESS
2539                    ADDI.L   #4,D4
2540                    SUBI.L   #1,D0
2541                    BRA      PLACECOUNTER
2542 CONVERTADDRESS    CMP      A2,A3
2543                    BEQ      ADDRESSDONE
2544                    CLR.L    D3
2545                    MOVE.B   (A2)+,D3
2546                    MOVEM.L  A2-A3/D0-D2/D4-D5,-(SP)    ;so regs dont
        get destroyed
2547                    JSR      ASCII_TO_BCD
2548                    JSR      BCD_TO_HEX
2549                    MOVEM.L  (SP)+,A2-A3/D0-D2/D4-D5
2550                    ROL.L     D4,D3
2551                    SUBI.L   #4,D4
2552                    ADD.L     D3,D5    ;get total
2553                    BRA      CONVERTADDRESS
2554 ADDRESSDONE       RTS
```

2.3.2 ASCII to Hexadecimal

2.3.2.1 Algorithm

2.3.2.2 Assembly Code

```
2487        MOVE.L  #13,D0
2488        TRAP  #15
2489        MOVEM.L  (SP)+,A1/D0
2490        MOVE.L #$01000000,SP
2491        BRA SHELL
2492
2493
2494
2495 ****************************COMMON SUBROUTINES NEEDED
        *********************************
```

```
2496
2497
2498 *
```

```
2499 ERRORSR:          LEA     ERROR,A1      ;load message
2500                   MOVE.W  #44,D1
2501                   MOVE.L  #0,D0
2502                   TRAP    #15
2503                   BRA     RESTORE
2504 *
```

```
2505 ASCII_TO_BCD:    *Number passed via D3 byte size only(to be
         expected)*
2506                   CMP #$46,D3
2507                   BGT ERRORSR
2508                   CMP #$40,D3
2509                   BGT UPPER
2510                   SUBI.L  #$30,D3
2511                   RTS
2512 UPPER:           SUBI.L  #$31,D3 ;If ASCII number is A-F
2513                   RTS
```

2.3.3 BCD to Hexadecimal

2.3.3.1 Algorithm

2.3.3.2 Assembly Code

```
2475 FLERR:
2476     MOVEM.L A1/D0,-(SP)
2477     LEA FLERR_TEXT,A1
2478     MOVE.L #13,D0
2479     TRAP #15
2480     MOVEM.L (SP)+,A1/D0
2481     MOVE.L #$01000000,SP
2482     BRA SHELL
2483
2484 CHKERR:
2485     MOVEM.L A1/D0,-(SP)
```

2.3.4 ASCII to BCD

2.3.4.1 Algorithm

2.3.4.2 Assembly Code

```
2464        BRA SHELL
2465
2466 ALERR:
2467        MOVEM.L A1/D0,-(SP)
2468        LEA ALERR_TEXT,A1
2469        MOVE.L #13,D0
2470        TRAP #15
2471        MOVEM.L (SP)+,A1/D0
2472        MOVE.L #$01000000,SP
```

2.4 Exception Handlers

The Monitor441 program uses custom exception handlers. They are loaded using the source code:

```
134         *Load custom exceptions*
135         LEA BERR,A1 ;init exception handlers
136         MOVE.L A1,$8
137         LEA AERR,A1
138         MOVE.L A1,$C
139         LEA IERR,A1
140         MOVE.L A1,$10
141         LEA ZERR,A1
142         MOVE.L A1,$14
143         LEA CHKERR,A1
144         MOVE.L  A1,$18
145         LEA PERR,A1
146         MOVE.L A1,$20
147         LEA ALERR,A1
148         MOVE.L A1,$28
149         LEA FLERR,A1
150         MOVE.L A1,$2C
151         MOVEM.L (SP)+,D0-D2/A1  ;restore any preset values
```

2.4.1 Bus Error Exception

2.4.1.1 Algorithm and Flowchart

2.4.1.2 Assembly Code

```
2320            SUB.L     D2,A1
2321 SSDONTWORRY:
2322            MOVE.L    #0,D0
2323            MOVE.L    #8,D1
2324            TRAP      #15
2325
2326        *-----------PC-----------*
2327            LEA       RPC,A1
2328            MOVE.L    #4,D1
2329            MOVE.L    #1,D0
2330            TRAP      #15
2331            MOVE.L    (SP)+,D3
2332            JSR       HEXTOASCII
2333            MOVE.L    A2,A1
2334            SUB.L     A3,A2
2335            MOVE.L    A2,D2
2336            CMP.L     #-8,D2
2337            BEQ       PCDONTWORRY
2338 PCACCOUNTFORZEROS:
2339              ADDI.L  #8,D2
2340              SUB.L   D2,A1
2341 PCDONTWORRY:
2342            MOVE.L    #0,D0
2343            MOVE.L    #8,D1
2344            TRAP      #15
2345
2346     *---DF HACK RESTORE---*
2347     MOVE.W    D7,-(SP)
2348     ADD.L     #-72,SP
2349     MOVEM.L   (SP)+,D0-D7/A0-A6
2350     ADD.L     #12,SP   ;go back to original value
2351     MOVE.W    (SP)+,SR
2352
2353          BRA SHELL
2354
2355 *
```

2.4.2 Address Error Exception

2.4.2.1 Algorithm and Flowchart

2.4.2.2 Assembly Code

```
2359 *-----------------------------------------------------------------------------

2360
2361 BERR:
2362            MOVEM.L  A1-A3/D0-D1,-(SP)
2363            LEA      BERR_TEXT,A1
2364            MOVE.L   #13,D0
2365            TRAP     #15
2366            LEA      SSW,A1
2367            MOVE.L   #14,D0
2368            TRAP     #15
2369            MOVE.W   (20,SP),D3
2370            JSR      HEXTOASCII
2371            SUB.L    #4,A3
2372            MOVEA.L  A3,A1
2373            MOVE.L   #4,D1
2374            MOVE.L   #0,D0
2375            TRAP     #15
2376            LEA      BA,A1
2377            MOVE.L   #14,D0
2378            TRAP     #15
2379            MOVE.L   (22,SP),D3
2380            JSR      HEXTOASCII
2381            SUB.L    #8,A3
2382            MOVEA.L  A3,A1
2383            MOVE.L   #8,D1
2384            MOVE.L   #0,D0
2385            TRAP     #15
2386            LEA      IR,A1
2387            MOVE.L   #14,D0
2388            TRAP     #15
2389            MOVE.W   (26,SP),D3
2390            JSR      HEXTOASCII
```

```
2391            SUB.L     #4,A3
2392            MOVEA.L   A3,A1
2393            MOVE.L    #4,D1
2394            MOVE.L    #0,D0
2395            TRAP      #15
2396            MOVEM.L   (SP)+,A1-A3/D0-D1
```

2.4.3 Illegal Instruction Error Exception

2.4.3.1 Algorithm and Flowchart

2.4.3.2 Assembly Code

```
2398            BRA       SHELL
2399
2400 AERR:
2401            MOVEM.L   A1-A3/D0-D1,-(SP)
2402            LEA       AERR_TEXT,A1
2403            MOVE.L    #13,D0
2404            TRAP      #15
2405            LEA       SSW,A1
```

2.4.4 Privilege Violation Error Exception

2.4.4.1 Algorithm and Flowchart

2.4.4.2 Assembly Code

```
2407            TRAP      #15
2408            MOVE.W    (20,SP),D3
2409            JSR       HEXTOASCII
2410            SUB.L     #4,A3
2411            MOVEA.L   A3,A1
2412            MOVE.L    #4,D1
2413            MOVE.L    #0,D0
2414            TRAP      #15
```

2.4.5 Divide by Zero Error Exception

2.4.5.1 Algorithm and Flowchart

2.4.5.2 Assembly Code

```
2416            MOVE.L    #14,D0
2417            TRAP      #15
2418            MOVE.L    (22,SP),D3
2419            JSR       HEXTOASCII
2420            SUB.L     #8,A3
2421            MOVEA.L   A3,A1
2422            MOVE.L    #8,D1
2423            MOVE.L    #0,D0
```

2.4.6 A Line Emulator Error Exception

2.4.6.1 Algorithm and Flowchart

2.4.6.2 Assembly Code

```
2425            LEA       IR,A1
2426            MOVE.L    #14,D0
2427            TRAP      #15
2428            MOVE.W    (26,SP),D3
2429            JSR       HEXTOASCII
2430            SUB.L     #4,A3
2431            MOVEA.L   A3,A1
2432            MOVE.L    #4,D1
```

2.4.7 F Line Emulator Error Exception

2.4.7.1 Algorithm and Flowchart

2.4.7.2 Assembly Code

```
2434            TRAP      #15
2435            MOVEM.L   (SP)+,A1-A3/D0-D1
2436            MOVE.L    #$01000000,SP     ;reset stack
2437            BRA       SHELL
2438
2439 IERR:
2440        MOVEM.L  A1/D0,-(SP)
2441        LEA  IERR_TEXT,A1
```

2.4.8 Check Instruction Error Exception

2.4.8.1 Algorithm and Flowchart

2.4.8.2 Assembly Code

```
2443        TRAP  #15
2444        MOVEM.L  (SP)+,A1/D0
2445        MOVE.L  #$01000000,SP
2446        BRA  SHELL
2447
2448 PERR:
2449        MOVEM.L  A1/D0,-(SP)
2450        LEA  PERR_TEXT,A1
```

2.5 User Instruction Manual Exception Handlers

2.5.0.3 Algorithm and Flowchart

2.5.0.4 Assembly Code

3 Discussion

4 Feature Suggestions

5 Conclusion

References

[1] test