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.

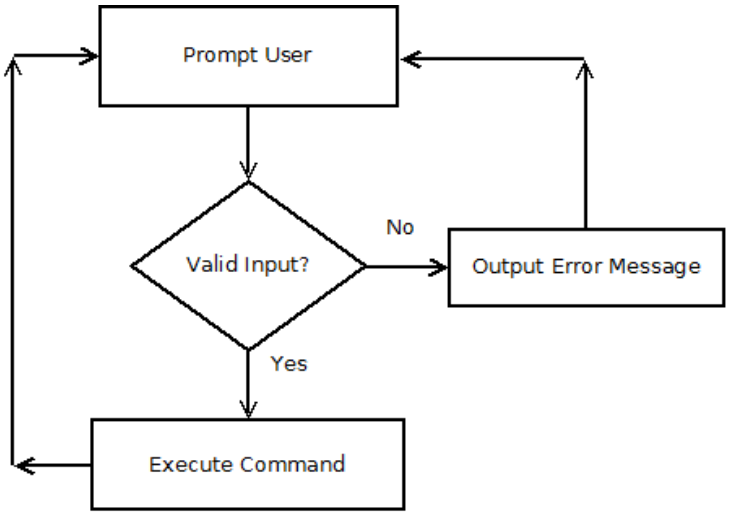

Figure 1: Structure of Monitor Program

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.

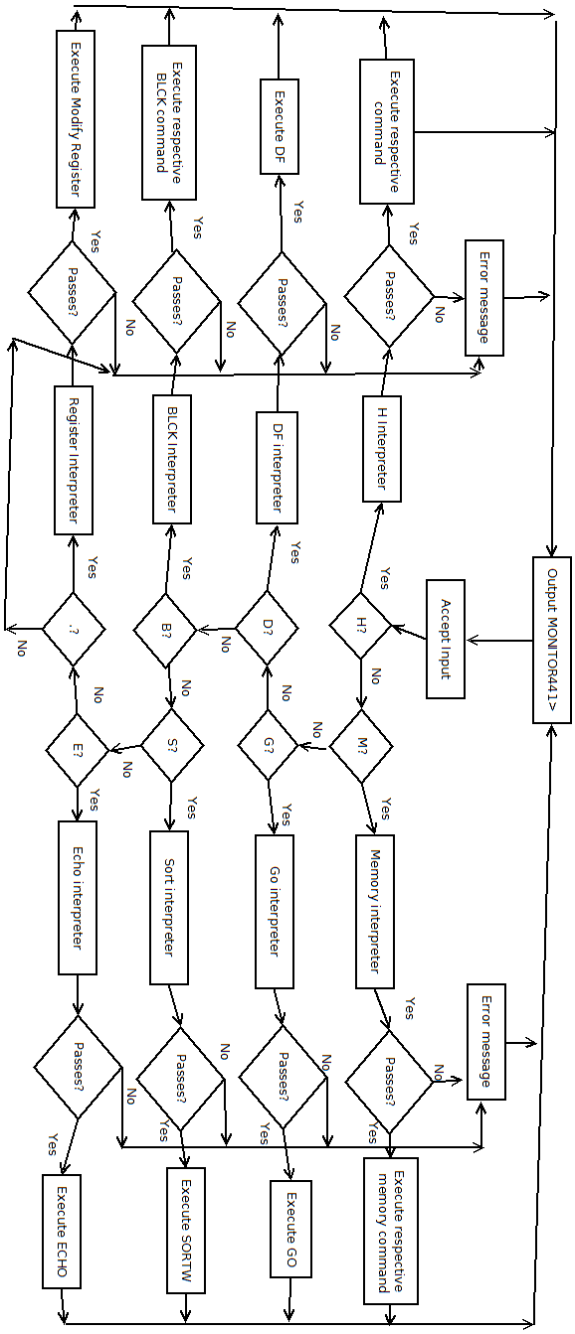

Figure 2: Flowchart for Command Line Interpreter

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     ERRORSR
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     ERRORSR
213                 CMP.B   #$48,(A1)+    ;H?
214                 BNE     ERRORSR
215                 CMP.B   #$4F,(A1)+    ;O?
216                 BNE     ERRORSR
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     ERRORSR
235 *---------------------------------------------------------------------------

236
237 BSCHTEST:      ADD.L   #1,A1
238                 CMP.B   #$43,(A1)
239                 BNE     ERRORSR
240                 ADD.L   #1,A1
241                 CMP.B   #$48,(A1)
242                 BNE     ERRORSR
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     ERRORSR
254                 ADD.L   #1,A1
255                 CMP.B   #$54,(A1)
256                 BNE     ERRORSR
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      ERRORSR
267                 ADD.L    #1,A1
268                 CMP.B    #$56,(A1)
269                 BNE      ERRORSR
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      ERRORSR
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      ERRORSR
293                 ADD.L    #1,A1
294                 CMP.B    #$52,(A1)     ;R?
295                 BNE      ERRORSR
296                 ADD.L    #1,A1
297                 CMP.B    #$54,(A1)     ;T?
298                 BNE      ERRORSR
299                 ADD.L    #1,A1
300                 CMP.B    #$57,(A1)     ;W?
301                 BNE      ERRORSR
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       ERRORSR
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       ERRORSR
324 *
    ------------------------------------------------------------------------------

325
326 HELPTST:
327                 ADD.L     #1,A1    ; check next char
328                 CMP.B     #$4C,(A1) ;check for L
329                 BNE       ERRORSR
330                 ADD.L     #1,A1
331                 CMP.B     #$50,(A1)    ;check for P
332                 BNE       ERRORSR
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      ERRORSR
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      ERRORSR
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      ERRORSR
377                ADD.L    #1,A1
378                CMP.B    #$45,(A1)
379                BNE      ERRORSR
380                ADD.L    #1,A1
381                CMP.B    #$43,(A1)
382                BNE      ERRORSR
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      ERRORSR
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.

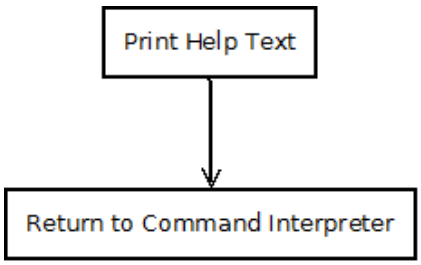

Figure 3: Flowchart for Help

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.

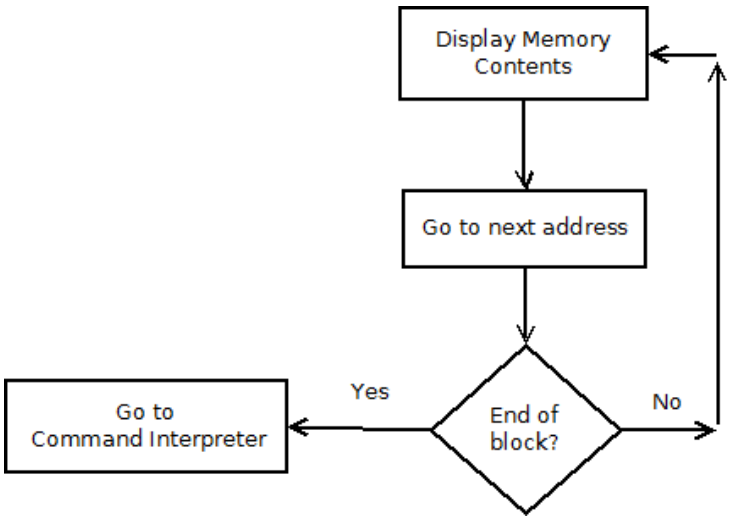

Figure 4: Flowchart for Memory Display

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

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

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

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            MOVE.L  A2,A1
1290            MOVE.B  #1,D0
1291            TRAP    #15
1292
1293            MOVE.B  #$20,(A1)
1294            MOVE.L  #1,D1    ;space between held data and input
1295            MOVE.L  #1,D0
1296            TRAP    #15
1297
1298
1299            *------Enter Input--------*
1300            CLR.L   D3
1301            MOVE.L  #4,D6
1302            LEA     BUFFER,A1    ;set up storage for command
1303            MOVE.B  #2,D0        ;load input trap call
1304            TRAP    #15
1305            CMP.B   #$2E,(A1)
1306            BEQ     ENDLP
1307            CMP.B   #$00,(A1)
1308            BEQ     ENTER
1309
1310 PARSELOOP:
1311            CMP.B   #$00,(A1)
1312            BEQ     ENDPARSE
1313            CMP.B   #$40,(A1)
1314            BGT     HIGHHEXMM
1315            SUBI.B  #$30,(A1)    ;get hex value
1316            BRA     NEXTMMSTEP
1317 HIGHHEXMM: SUBI.B  #$37,(A1)   ;get hex value
1318 NEXTMMSTEP:
1319            MOVE.B  (A1),D2
1320            ROL.L   D6,D2
1321            SUBI.L  #4,D6
1322            ADD.L   #1,A1
1323            ADD.B   D2,D3    ;total byte stored in D3
1324            BRA     PARSELOOP
1325 ENDPARSE:
```

```
1326            MOVE.B   D3,(A4)    ;commit memory change
1327 ENTER:    ADD.L    #1,A4     ;increment address
1328            BRA      MODIFYLOOP
1329
1330 *************************************************
1331
1332 WORD:
1333
1334            ADD.L    #2,A1          ;set up for subroutine
1335            MOVE     A1,A2          ;set up for subroutine
1336            MOVEM.L  D1/D6/A1-A3,-(SP)
1337            JSR      ASCII_ADDRESS
1338            MOVEM.L  (SP)+,D1/D6/A1-A3
1339            MOVE.L   D5,A4          ;set up address to modify
1340
1341 MODIFYLOOPW:
1342            *--------Display Memory First------*
1343            MOVE.L   A4,D0
1344            DIVU     #2,D0
1345            SWAP     D0        ;check if it's an odd address
1346            CMP.W    #$00,D0
1347            BNE      ERRORSR
1348            MOVE.L   A4,D3          ;set up for subroutine
1349            MOVE.L   A4,A5          ;next byte of memory may not be
       needed
1350            ADD.L    #1,A5
1351            JSR      HEXTOASCII   ;convert new address to ascii for
       output
1352            SUBA     A2,A3          ;get num of bytes to produce
1353            MOVE.L   #1,D0
1354            MOVE.L   A3,D1
1355            MOVE.L   A2,A1
1356            TRAP     #15
1357
1358            *add colon to denote containing data*
1359            MOVE.B   #$3A,(A1)
1360            MOVE.L   #1,D1    ;display only the colon
1361            MOVE.L   #1,D0
1362            TRAP     #15
1363
1364            MOVE.B   (A4),D3
1365            JSR      HEXTOASCII
1366            MOVE.L   #2,D1
1367            MOVE.L   A2,A1
1368            MOVE.B   #1,D0
```

```
1369            TRAP    #15
1370
1371            MOVE.B  (A5),D3
1372            JSR     HEXTOASCII
1373            MOVE.L  #2,D1
1374            MOVE.L  A2,A1
1375            MOVE.B  #1,D0
1376            TRAP    #15
1377
1378
1379            MOVE.B  #$20,(A1)
1380            MOVE.L  #1,D1    ;space between held data and input
1381            MOVE.L  #1,D0
1382            TRAP    #15
1383
1384
1385            *------Enter Input-------*
1386            CLR.L   D3
1387            MOVE.L  #12,D6
1388            LEA     BUFFER,A1    ;set up storage for command
1389            MOVE.B  #2,D0        ;load input trap call
1390            TRAP    #15
1391            CMP.B   #$2E,(A1)
1392            BEQ     ENDLP
1393            CMP.B   #$00,(A1)
1394            BEQ     ENTERW
1395
1396 PARSELOOPW:
1397            CMP.B   #$00,(A1)
1398            BEQ     ENDPARSEW
1399            CMP.B   #$40,(A1)
1400            BGT     HIGHHEXMMW
1401            SUBI.B  #$30,(A1)    ;get hex value
1402            BRA     NEXTMMSTEPW
1403 HIGHHEXMMW: SUBI.B  #$37,(A1)   ;get hex value
1404 NEXTMMSTEPW:
1405            MOVE.B  (A1),D2
1406            ROL.L   D6,D2
1407            SUBI.L  #4,D6
1408            ADD.L   #1,A1
1409            ADD.L   D2,D3    ;total byte stored in D3
1410            CLR.L   D2       ;clear for next rotate
1411            BRA     PARSELOOPW
1412 ENDPARSEW:
1413
```

```
1414            MOVE.W  D3,(A4)     ;commit memory change
1415 ENTERW:    ADD.L    #2,A4     ;increment address
1416            BRA     MODIFYLOOPW
1417
1418 *************************************************
1419
1420 LONG:
1421            ADD.L   #2,A1       ;set up for subroutine
1422            MOVE    A1,A2       ;set up for subroutine
1423            MOVEM.L D1/D6/A1-A3,-(SP)
1424            JSR     ASCII_ADDRESS
1425            MOVEM.L (SP)+,D1/D6/A1-A3
1426            MOVE.L  D5,A4       ;set up address to modify
1427
1428 MODIFYLOOPL:
1429            *--------Display Memory First------*
1430            MOVE.L  A4,D0
1431            DIVU    #2,D0
1432            SWAP    D0      ;check if it's an odd address
1433            CMP.W   #$00,D0
1434            BNE     ERRORSR
1435            MOVE.L  A4,D3       ;set up for subroutine
1436            MOVE.L  A4,A5       ;next byte of memory may not be
       needed
1437            ADD.L   #1,A5
1438            JSR     HEXTOASCII  ;convert new address to ascii for
       output
1439            SUBA    A2,A3       ;get num of bytes to produce
1440            MOVE.L  #1,D0
1441            MOVE.L  A3,D1
1442            MOVE.L  A2,A1
1443            TRAP    #15
1444
1445            *add colon to denote containing data*
1446            MOVE.B  #$3A,(A1)
1447            MOVE.L  #1,D1    ;display only the colon
1448            MOVE.L  #1,D0
1449            TRAP    #15
1450
1451            MOVE.B  (A4),D3
1452            JSR     HEXTOASCII
1453            MOVE.L  #2,D1
1454            MOVE.L  A2,A1
1455            MOVE.B  #1,D0
1456            TRAP    #15
```

```
1457
1458            MOVE.B   (A5)+,D3
1459            JSR      HEXTOASCII
1460            MOVE.L   #2,D1
1461            MOVE.L   A2,A1
1462            MOVE.B   #1,D0
1463            TRAP     #15
1464
1465            MOVE.B   (A5)+,D3
1466            JSR      HEXTOASCII
1467            MOVE.L   #2,D1
1468            MOVE.L   A2,A1
1469            MOVE.B   #1,D0
1470            TRAP     #15
1471            MOVE.B   (A5)+,D3
1472            JSR      HEXTOASCII
1473            MOVE.L   #2,D1
1474            MOVE.L   A2,A1
1475            MOVE.B   #1,D0
1476            TRAP     #15
1477
1478            MOVE.B   #$20,(A1)
1479            MOVE.L   #1,D1    ;space between held data and input
1480            MOVE.L   #1,D0
1481            TRAP     #15
1482
1483
1484            *------Enter Input-------*
1485            CLR.L    D3
1486            MOVE.L   #28,D6
1487            LEA      BUFFER,A1    ;set up storage for command
1488            MOVE.B   #2,D0        ;load input trap call
1489            TRAP     #15
1490            CMP.B    #$2E,(A1)
1491            BEQ      ENDLP
1492            CMP.B    #$00,(A1)
1493            BEQ      ENTERL
1494
1495 PARSELOOPL:
1496            CMP.B    #$00,(A1)
1497            BEQ      ENDPARSEL
1498            CMP.B    #$40,(A1)
1499            BGT      HIGHHEXMML
1500            SUBI.B   #$30,(A1)    ;get hex value
1501            BRA      NEXTMMSTEPL
```

```
1502 HIGHHEXMML: SUBI.B  #$37,(A1)  ;get hex value
1503 NEXTMMSTEPL:
1504          MOVE.B  (A1),D2
1505          ROL.L   D6,D2
1506          SUBI.L  #4,D6
1507          ADD.L   #1,A1
1508          ADD.L   D2,D3   ;total byte stored in D3
1509          CLR.L   D2      ;clear for next rotate
1510          BRA     PARSELOOPL
1511 ENDPARSEL:
1512          MOVE.L  D3,(A4)   ;commit memory change
1513 ENTERL:  ADD.L   #4,A4   ;increment address
1514          BRA     MODIFYLOOPL
1515
1516
1517 ENDLP:   BRA RESTORE
```

2.2.6 Memory Set

2.2.6.1 Algorithm and Flowchart

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

2.2.7.2 Assembly Code

```
1522 BF:
1523            ADD.L     #1,A1    ;first byte of data
1524            MOVE.L    A1,A3    ;for end ptr
1525 BFGETENDDATA:
1526             CMP.B    #$20,(A3)+
1527             BEQ      BFNEXTADDR
1528             BRA      BFGETENDDATA
1529 BFNEXTADDR:
1530             MOVE.L   A1,A2    ;for subroutine
1531             SUB.L    #1,A3    ;off by one error
1532             JSR      ASCII_ADDRESS
1533             MOVE.L   D5,-(SP)     ;save data on the stack
1534
1535             ADD.L    #1,A3    ;inc end ptr to first byte of address
1536             MOVE.L   A3,A2    ;set start ptr
1537 BFGETENDADDRONE:
1538             CMP.B    #$20,(A3)+
1539             BEQ      BFNEXTADDRTWO
1540             BRA      BFGETENDADDRONE
1541
1542 BFNEXTADDRTWO:
1543             SUB.L    #1,A3    ;off by one error
1544             JSR      ASCII_ADDRESS   ;convert address to hex
1545             MOVE.L   D5,A5        ;store address 1 in A5
1546             DIVU     #2,D5
1547             SWAP     D5
```

```
1548            CMP.W    #$00,D5
1549            BNE      ERRORSR
1550
1551            ADD.L    #1,A3    ;inc end ptr to first byte of address
1552            MOVE.L   A3,A2    ;set start ptr
1553 BFGETLASTEND:
1554            CMP.B    #$00,(A3)+
1555            BEQ      STOREDATA
1556            BRA      BFGETLASTEND
1557
1558 STOREDATA:
1559            SUB.L    #1,A3    ;off by one error
1560            JSR      ASCII_ADDRESS
1561            MOVE.L   D5,A6    ;end address in A6
1562            DIVU     #2,D5
1563            SWAP     D5
1564            CMP.W    #$00,D5
1565            BNE      ERRORSR
1566            MOVE.L   (SP)+,D5
1567
1568 DATALOOP:
1569            CMP.L    A5,A6
1570            BLT      ENDBF
1571            MOVE.W   D5,(A5)+
1572            BRA      DATALOOP
1573
1574 ENDBF:    BRA RESTORE
```

2.2.8 Block Move

2.2.8.1 Algorithm and Flowchart

2.2.8.2 Assembly Code

```
1577 BMOV:     ADD.L    #1,A1    ;get to start of first address
1578            MOVE.L   A1,A2    ;set up start ptr
1579            MOVE.L   A2,A3    ;set up end ptr
1580
1581 FIRSTADDRESS:
1582            CMP.B #$20,(A3)+
1583            BEQ      COMPUTEFIRSTADD
1584            BRA      FIRSTADDRESS
```

```
1585
1586 COMPUTEFIRSTADD:
1587          SUB.L    #1,A3    ;off by one error
1588          JSR      ASCII_ADDRESS
1589          MOVE.L   D5,A0    ; save 1st address
1590
1591          ADD.L    #1,A3
1592          MOVE.L   A3,A2
1593 SECONDADDRESS:
1594          CMP.B    #$20,(A3)+
1595          BEQ      COMPUTESECONDADDRESS
1596          BRA      SECONDADDRESS
1597
1598 COMPUTESECONDADDRESS:
1599          SUB.L    #1,A3    ;off by one error
1600          JSR      ASCII_ADDRESS
1601          MOVE.L   D5,A4    ;save 2nd address
1602
1603          ADD.L    #1,A3
1604          MOVE.L   A3,A2
1605 THIRDADDRESS:
1606          CMP.B    #$20,(A3)+
1607          BEQ      COMPUTETHIRDADDRESS
1608          BRA      THIRDADDRESS
1609
1610 COMPUTETHIRDADDRESS:
1611          SUB.L    #1,A3
1612          JSR      ASCII_ADDRESS
1613          MOVE.L   D5,A5    ;save 3rd address
1614
1615          ADD.L    #1,A3
1616          MOVE.L   A3,A2
1617 FOURTHADDRESS:
1618          CMP.B    #$00,(A3)+
1619          BEQ      COMPUTEFOURTHADDRESS
1620          BRA      FOURTHADDRESS
1621
1622 COMPUTEFOURTHADDRESS:
1623          SUB.L    #1,A3
1624          JSR      ASCII_ADDRESS
1625          MOVE.L   D5,A6    ;save 3rd address
1626
1627
1628
1629          *Check for matching dimensions*
```

```
1630            MOVE.L   A0,D0
1631            MOVE.L   A4,D1
1632            MOVE.L   A5,D5
1633            MOVE.L   A6,D6
1634            SUB.L    D0,D1
1635            SUB.L    D5,D6
1636            CMP.L    D1,D6
1637            BNE      ERRORSR
1638            CMP.L    A0,A4
1639            BLT      ERRORSR
1640            CMP.L    A5,A6
1641            BLT      ERRORSR
1642
1643 DATATRANSFER:
1644            CMP.L    A0,A4
1645            BLT      BMOVDONE
1646            MOVE.B   (A0)+,(A5)+
1647            BRA      DATATRANSFER
1648
1649
1650
1651 BMOVDONE:
1652            BRA RESTORE
```

2.2.9 Block Test

2.2.9.1 Algorithm and Flowchart

2.2.9.2 Assembly Code

```
1656 BTST:
1657          ADD.L    #1,A1    ;first byte of data
1658          MOVE.L   A1,A3    ;for end ptr
1659 BTSTGETENDDATA:
1660            CMP.B    #$20,(A3)+
1661            BEQ      BTSTNEXTADDR
1662            BRA      BTSTGETENDDATA
1663 BTSTNEXTADDR:
1664            MOVE.L   A1,A2    ;for subroutine
1665            SUB.L    #1,A3    ;off by one error
1666            JSR      ASCII_ADDRESS
1667            MOVE.L   D5,-(SP)     ;save data on the stack
```

```
1668
1669            ADD.L    #1,A3    ;inc end ptr to first byte of address
1670            MOVE.L   A3,A2    ;set start ptr
1671 BTSTGETENDADDRONE:
1672            CMP.B    #$20,(A3)+
1673            BEQ      BTSTNEXTADDRTWO
1674            BRA      BTSTGETENDADDRONE
1675
1676 BTSTNEXTADDRTWO:
1677            SUB.L    #1,A3    ;off by one error
1678            JSR      ASCII_ADDRESS    ;convert address to hex
1679            MOVE.L   D5,A5        ;store address 1 in A5
1680            MOVE.L   D5,A4        ;for second run through
1681
1682            ADD.L    #1,A3    ;inc end ptr to first byte of address
1683            MOVE.L   A3,A2    ;set start ptr
1684 BTSTGETLASTEND:
1685            CMP.B    #$00,(A3)+
1686            BEQ      STOREDATABTST
1687            BRA      BTSTGETLASTEND
1688
1689
1690 STOREDATABTST:
1691            SUB.L    #1,A3    ;off by one error
1692            JSR      ASCII_ADDRESS
1693            MOVE.L   D5,A6    ;end address in A6
1694            MOVE.L   (SP)+,D5
1695
1696 BTSTDATALOOP:
1697            CMP.L    A5,A6
1698            BLT      READ
1699            MOVE.B   D5,(A5)+
1700            BRA      BTSTDATALOOP
1701
1702
1703 READ:
1704            CMP.L    A4,A6
1705            BLT      COMPLETE
1706            CMP.B    (A4)+,D5
1707            BNE      FAIL
1708            BRA      READ
1709
1710 FAIL:
1711            LEA      BTST4,A1
1712            MOVE.L   #11,D1
```

```
1713            MOVE.L  #0,D0
1714            TRAP    #15
1715
1716            LEA     BTST1,A1
1717            MOVE.L  #1,D0
1718            MOVE.L  #20,D1
1719            TRAP    #15
1720
1721            MOVE.B  D5,D3    ;for subroutine
1722            JSR     HEXTOASCII
1723            MOVE.L      A2,A1
1724            MOVE.L  #0,D0
1725            SUBA.L  A2,A3    ;number of bytes
1726            MOVE.L  A3,D1
1727            TRAP    #15
1728
1729
1730            LEA     BTST2,A1
1731            MOVE.L  #1,D0
1732            MOVE.L  #17,D1
1733            TRAP    #15
1734
1735
1736            SUB.L   #1,A4    ;go back to address that failed
1737            MOVE.B  (A4),D3
1738            JSR     HEXTOASCII  ;convert for output
1739            MOVE.L      A2,A1
1740            MOVE.L  #0,D0
1741            SUBA.L  A2,A3    ;number of bytes
1742            MOVE.L  A3,D1
1743            TRAP    #15
1744
1745            LEA     BTST5,A1
1746            MOVE.L  #27,D1
1747            MOVE.B  #1,D0
1748            TRAP    #15
1749            MOVE.L  A4,D3
1750            JSR     HEXTOASCII
1751            MOVE.L      A2,A1
1752            MOVE.L  #0,D0
1753            SUBA.L  A2,A3    ;number of bytes
1754            MOVE.L  A3,D1
1755            TRAP    #15
1756
1757
```

```
1758
1759 COMPLETE:
1760
1761         LEA     BTST3,A1
1762         MOVE.L  #18,D1
1763         MOVE.L  #0,D0
1764         TRAP    #15
1765         BRA RESTORE
```

2.2.10 Block Search

2.2.10.1 Algorithm and Flowchart

2.2.10.2 Assembly Code

```
1769 BSCH:
1770         ADD.L   #1,A1    ;start of data
1771         MOVE.L  A1,A2    ;set up bac ptr
1772
1773 BSCHENDDATA:
1774         CMP.B   #$20,(A2)+
1775         BEQ     BSCHFIRSTADD
1776         BRA     BSCHENDDATA
1777
1778
1779 BSCHFIRSTADD:
1780         SUB.L   #1,A2
1781         MOVE.L  A2,A3
1782         MOVE.L  A1,A2
1783         JSR     ASCII_ADDRESS
1784         SUB.L   A1,A3    ;see how many bytes
1785         MOVE.L  A3,D6    ;store byte/word/long in D6
1786         ADD.L   #1,A2    ;set up for start of next address
1787         MOVE.L  A2,A3    ;set up for end ptr
1788         MOVE.L  D5,-(SP)     ;save data to stack
1789
1790
1791 BSCHFADDEND:
1792         CMP.B   #$20,(A3)+
1793         BEQ     BSCHSECONDADD
1794         BRA     BSCHFADDEND
1795
```

```
1796
1797 BSCHSECONDADD:
1798            SUB.L    #1,A3    ;off by one
1799            JSR      ASCII_ADDRESS
1800            MOVE.L   D5,A5    ;first address destination
1801            ADD.L    #1,A3    ;start it at next address
1802            MOVE.L   A3,A2    ; set up for next address
1803
1804
1805 BSCHSECONDFIND:
1806            CMP.B    #$00,(A3)+
1807            BEQ      TESTOP
1808            BRA      BSCHSECONDFIND
1809
1810
1811 TESTOP:
1812            SUB.L    #1,A3    ;off by one
1813            JSR      ASCII_ADDRESS
1814            MOVE.L   D5,A6    ;end address at A6
1815            MOVE.L   (SP)+,D5     ;restore data
1816            CMP.B    #2,D6
1817            BEQ      BYTEBSCH
1818            CMP.B    #4,D6
1819            BEQ      WORDBSCH
1820            CMP.B    #8,D6
1821            BEQ      LONGBSCH
1822            BRA      ERRORSR
1823
1824 BYTEBSCH:
1825            CMP.L    A5,A6
1826            BLT      ENDBSCH
1827            CMP.B    (A5)+,D5
1828            BEQ      FOUNDB
1829            BRA      BYTEBSCH
1830
1831 WORDBSCH:
1832            CMP.L    A5,A6
1833            BLT      ENDBSCH
1834            CMP.W    (A5)+,D5
1835            BEQ      FOUNDW
1836            BRA      WORDBSCH
1837
1838 LONGBSCH:
1839            CMP.L    A5,A6
1840            BLT      ENDBSCH
```

```
1841            CMP.L    (A5)+,D5
1842            BEQ      FOUNDL
1843            BRA      LONGBSCH
1844
1845
1846
1847 FOUNDB:
1848            SUB.L    #1,A5
1849            MOVE.B   (A5),D3
1850            BRA      SUCCESSTEXT
1851 FOUNDW:
1852            SUB.L    #2,A5
1853            MOVE.W   (A5),D3
1854            BRA      SUCCESSTEXT
1855 FOUNDL:
1856            SUB.L    #4,A5
1857            MOVE.L   (A5),D3
1858
1859 SUCCESSTEXT:
1860            LEA BSCH1,A1
1861            MOVE.L   #6,D1
1862            MOVE.L   #1,D0
1863            TRAP     #15
1864
1865            JSR      HEXTOASCII
1866            MOVE.L   A2,A1
1867            SUB.L    A2,A3
1868            MOVE.L   A3,D1    ;how many bytes
1869            MOVE.L   #0,D0
1870            TRAP     #15
1871
1872            LEA BSCH2,A1
1873            MOVE.L   #18,D1
1874            MOVE.L   #1,D0
1875            TRAP     #15
1876
1877            MOVE.L   A5,D3
1878            JSR      HEXTOASCII
1879            MOVE.L   A2,A1
1880            SUB.L    A2,A3
1881            MOVE.L   A3,D1    ;how many bytes
1882            MOVE.L   #0,D0
1883            TRAP     #15
1884
1885
```

```
1886 ENDBSCH:
1887         BRA RESTORE
```

2.2.11 Go

2.2.11.1 Algorithm and Flowchart

2.2.11.2 Assembly Code

```
1891 GO:
1892         MOVE.L   A1,A2    ;setup for hex conversion
1893         MOVE.L   A2,A3
1894 GGETEND:
1895         CMP.B    #$00,(A3)+
1896         BEQ      EXECUTE
1897         BRA      GGETEND
1898
1899 EXECUTE:
1900         SUB.L    #1,A3    ;off by one error
1901         JSR      ASCII_ADDRESS
1902         MOVE.L   D5,A0
1903         JSR      (A0)     ;go to program
1904         **NOTE: THE PROGRAM MUST HAVE RTS OR CONTROL WILL NOT BE
       RETURNED BACK TO MONITOR441!!!**
1905         BRA RESTORE
```

2.2.12 Display Formatted Registers

2.2.12.1 Algorithm and Flowchart

2.2.12.2 Assembly Code

```
1909 DF:    *Registers have already been saved to STACK, just need to
      pop them off first*
1910        *Stack looks like this*
1911
1912        *---------------*
1913        *|D0-D7/A0-A6|*
```

```
1914            *|      USP     |*
1915            *|      SR      |*
1916            *|      SSP     |*
1917            *|      PC      |*
1918            *---------------*
1919            *I should've used loops for efficiency but runtime is
        not a design constraint*
1920            *Maybe fix this in the future?*
1921
1922                *----------D0----------*
1923            LEA         RD0,A1
1924            MOVE.L      #4,D1
1925            MOVE.L      #1,D0
1926            TRAP        #15
1927            MOVE.L      (SP)+,D3
1928            JSR         HEXTOASCII
1929            MOVE.L      A2,A1
1930            SUB.L       A3,A2
1931            MOVE.L      A2,D2
1932            CMP.L       #-8,D2
1933            BEQ         D0DONTWORRY
1934 D0ACCOUNTFORZEROS:
1935              ADDI.L    #8,D2
1936              SUB.L     D2,A1
1937 D0DONTWORRY:
1938            MOVE.L      #0,D0
1939            MOVE.L      #8,D1
1940            TRAP        #15
1941
1942                *----------D1----------*
1943            LEA         RD1,A1
1944            MOVE.L      #4,D1
1945            MOVE.L      #1,D0
1946            TRAP        #15
1947            MOVE.L      (SP)+,D3
1948            JSR         HEXTOASCII
1949            MOVE.L      A2,A1
1950            SUB.L       A3,A2
1951            MOVE.L      A2,D2
1952            CMP.L       #-8,D2
1953            BEQ         D1DONTWORRY
1954 D1ACCOUNTFORZEROS:
1955              ADDI.L    #8,D2
1956              SUB.L     D2,A1
1957 D1DONTWORRY:
```

```
1958            MOVE.L      #0,D0
1959            MOVE.L      #8,D1
1960            TRAP        #15
1961
1962                    *-------------D2-------------*
1963            LEA         RD2,A1
1964            MOVE.L      #4,D1
1965            MOVE.L      #1,D0
1966            TRAP        #15
1967            MOVE.L      (SP)+,D3
1968            JSR         HEXTOASCII
1969            MOVE.L      A2,A1
1970            SUB.L       A3,A2
1971            MOVE.L      A2,D2
1972            CMP.L       #-8,D2
1973            BEQ         D2DONTWORRY
1974 D2ACCOUNTFORZEROS:
1975              ADDI.L    #8,D2
1976              SUB.L     D2,A1
1977 D2DONTWORRY:
1978            MOVE.L      #0,D0
1979            MOVE.L      #8,D1
1980            TRAP        #15
1981
1982                    *-------------D3-------------*
1983            LEA         RD3,A1
1984            MOVE.L      #4,D1
1985            MOVE.L      #1,D0
1986            TRAP        #15
1987            MOVE.L      (SP)+,D3
1988            JSR         HEXTOASCII
1989            MOVE.L      A2,A1
1990            SUB.L       A3,A2
1991            MOVE.L      A2,D2
1992            CMP.L       #-8,D2
1993            BEQ         D3DONTWORRY
1994 D3ACCOUNTFORZEROS:
1995              ADDI.L    #8,D2
1996              SUB.L     D2,A1
1997 D3DONTWORRY:
1998            MOVE.L      #0,D0
1999            MOVE.L      #8,D1
2000            TRAP        #15
2001
2002                    *-------------D4-------------*
```

```
2003            LEA         RD4,A1
2004            MOVE.L      #4,D1
2005            MOVE.L      #1,D0
2006            TRAP        #15
2007            MOVE.L      (SP)+,D3
2008            JSR         HEXTOASCII
2009            MOVE.L      A2,A1
2010            SUB.L       A3,A2
2011            MOVE.L      A2,D2
2012            CMP.L       #-8,D2
2013            BEQ         D4DONTWORRY
2014 D4ACCOUNTFORZEROS:
2015              ADDI.L    #8,D2
2016              SUB.L     D2,A1
2017 D4DONTWORRY:
2018            MOVE.L      #0,D0
2019            MOVE.L      #8,D1
2020            TRAP        #15
2021
2022            *-----------D5-----------*
2023            LEA         RD5,A1
2024            MOVE.L      #4,D1
2025            MOVE.L      #1,D0
2026            TRAP        #15
2027            MOVE.L      (SP)+,D3
2028            JSR         HEXTOASCII
2029            MOVE.L      A2,A1
2030            SUB.L       A3,A2
2031            MOVE.L      A2,D2
2032            CMP.L       #-8,D2
2033            BEQ         D5DONTWORRY
2034 D5ACCOUNTFORZEROS:
2035              ADDI.L    #8,D2
2036              SUB.L     D2,A1
2037 D5DONTWORRY:
2038            MOVE.L      #0,D0
2039            MOVE.L      #8,D1
2040            TRAP        #15
2041
2042              *-----------D6-----------*
2043            LEA         RD6,A1
2044            MOVE.L      #4,D1
2045            MOVE.L      #1,D0
2046            TRAP        #15
2047            MOVE.L      (SP)+,D3
```

```
2048            JSR         HEXTOASCII
2049            MOVE.L      A2,A1
2050            SUB.L       A3,A2
2051            MOVE.L      A2,D2
2052            CMP.L       #-8,D2
2053            BEQ         D6DONTWORRY
2054 D6ACCOUNTFORZEROS:
2055              ADDI.L    #8,D2
2056              SUB.L     D2,A1
2057 D6DONTWORRY:
2058            MOVE.L      #0,D0
2059            MOVE.L      #8,D1
2060            TRAP        #15
2061
2062              *-----------D7-----------*
2063            LEA         RD7,A1
2064            MOVE.L      #4,D1
2065            MOVE.L      #1,D0
2066            TRAP        #15
2067            MOVE.L      (SP)+,D3
2068            JSR         HEXTOASCII
2069            MOVE.L      A2,A1
2070            SUB.L       A3,A2
2071            MOVE.L      A2,D2
2072            CMP.L       #-8,D2
2073            BEQ         D7DONTWORRY
2074 D7ACCOUNTFORZEROS:
2075              ADDI.L    #8,D2
2076              SUB.L     D2,A1
2077 D7DONTWORRY:
2078            MOVE.L      #0,D0
2079            MOVE.L      #8,D1
2080            TRAP        #15
2081
2082      *-----------A0-----------*
2083            LEA         RA0,A1
2084            MOVE.L      #4,D1
2085            MOVE.L      #1,D0
2086            TRAP        #15
2087            MOVE.L      (SP)+,D3
2088            JSR         HEXTOASCII
2089            MOVE.L      A2,A1
2090            SUB.L       A3,A2
2091            MOVE.L      A2,D2
2092            CMP.L       #-8,D2
```

```
2093            BEQ         A0DONTWORRY
2094 A0ACCOUNTFORZEROS:
2095              ADDI.L    #8,D2
2096              SUB.L     D2,A1
2097 A0DONTWORRY:
2098            MOVE.L      #0,D0
2099            MOVE.L      #8,D1
2100            TRAP        #15
2101
2102                *-------------A1-------------*
2103            LEA         RA1,A1
2104            MOVE.L      #4,D1
2105            MOVE.L      #1,D0
2106            TRAP        #15
2107            MOVE.L      (SP)+,D3
2108            JSR         HEXTOASCII
2109            MOVE.L      A2,A1
2110            SUB.L       A3,A2
2111            MOVE.L      A2,D2
2112            CMP.L       #-8,D2
2113            BEQ         A1DONTWORRY
2114 A1ACCOUNTFORZEROS:
2115              ADDI.L    #8,D2
2116              SUB.L     D2,A1
2117 A1DONTWORRY:
2118            MOVE.L      #0,D0
2119            MOVE.L      #8,D1
2120            TRAP        #15
2121
2122              *-------------A2-------------*
2123            LEA         RA2,A1
2124            MOVE.L      #4,D1
2125            MOVE.L      #1,D0
2126            TRAP        #15
2127            MOVE.L      (SP)+,D3
2128            JSR         HEXTOASCII
2129            MOVE.L      A2,A1
2130            SUB.L       A3,A2
2131            MOVE.L      A2,D2
2132            CMP.L       #-8,D2
2133            BEQ         A2DONTWORRY
2134 A2ACCOUNTFORZEROS:
2135              ADDI.L    #8,D2
2136              SUB.L     D2,A1
2137 A2DONTWORRY:
```

```
2138            MOVE.L      #0,D0
2139            MOVE.L      #8,D1
2140            TRAP        #15
2141
2142              *-----------A3-----------*
2143            LEA         RA3,A1
2144            MOVE.L      #4,D1
2145            MOVE.L      #1,D0
2146            TRAP        #15
2147            MOVE.L      (SP)+,D3
2148            JSR         HEXTOASCII
2149            MOVE.L      A2,A1
2150            SUB.L       A3,A2
2151            MOVE.L      A2,D2
2152            CMP.L       #-8,D2
2153            BEQ         A3DONTWORRY
2154 A3ACCOUNTFORZEROS:
2155              ADDI.L    #8,D2
2156              SUB.L     D2,A1
2157 A3DONTWORRY:
2158            MOVE.L      #0,D0
2159            MOVE.L      #8,D1
2160            TRAP        #15
2161
2162              *-----------A4-----------*
2163            LEA         RA3,A1
2164            MOVE.L      #4,D1
2165            MOVE.L      #1,D0
2166            TRAP        #15
2167            MOVE.L      (SP)+,D3
2168            JSR         HEXTOASCII
2169            MOVE.L      A2,A1
2170            SUB.L       A3,A2
2171            MOVE.L      A2,D2
2172            CMP.L       #-8,D2
2173            BEQ         A4DONTWORRY
2174 A4ACCOUNTFORZEROS:
2175              ADDI.L    #8,D2
2176              SUB.L     D2,A1
2177 A4DONTWORRY:
2178            MOVE.L      #0,D0
2179            MOVE.L      #8,D1
2180            TRAP        #15
2181
2182            *-----------A5-----------*
```

```
2183            LEA         RA3,A1
2184            MOVE.L      #4,D1
2185            MOVE.L      #1,D0
2186            TRAP        #15
2187            MOVE.L      (SP)+,D3
2188            JSR         HEXTOASCII
2189            MOVE.L      A2,A1
2190            SUB.L       A3,A2
2191            MOVE.L      A2,D2
2192            CMP.L       #-8,D2
2193            BEQ         A5DONTWORRY
2194 A5ACCOUNTFORZEROS:
2195              ADDI.L    #8,D2
2196              SUB.L     D2,A1
2197 A5DONTWORRY:
2198            MOVE.L      #0,D0
2199            MOVE.L      #8,D1
2200            TRAP        #15
2201
2202            *-----------A6-----------*
2203            LEA         RA3,A1
2204            MOVE.L      #4,D1
2205            MOVE.L      #1,D0
2206            TRAP        #15
2207            MOVE.L      (SP)+,D3
2208            JSR         HEXTOASCII
2209            MOVE.L      A2,A1
2210            SUB.L       A3,A2
2211            MOVE.L      A2,D2
2212            CMP.L       #-8,D2
2213            BEQ         A6DONTWORRY
2214 A6ACCOUNTFORZEROS:
2215              ADDI.L    #8,D2
2216              SUB.L     D2,A1
2217 A6DONTWORRY:
2218            MOVE.L      #0,D0
2219            MOVE.L      #8,D1
2220            TRAP        #15
2221        *-----HACK---*
2222        ADD.L   #60,SP   ;should put stack in correct place
2223
2224                    *-----------USP-----------*
2225            LEA         RUS,A1
2226            MOVE.L      #4,D1
2227            MOVE.L      #1,D0
```

```
2228            TRAP        #15
2229            MOVE.L      (SP)+,D3
2230            JSR         HEXTOASCII
2231            MOVE.L      A2,A1
2232            SUB.L       A3,A2
2233            MOVE.L      A2,D2
2234            CMP.L       #-8,D2
2235            BEQ         USPDONTWORRY
2236 USPACCOUNTFORZEROS:
2237              ADDI.L    #8,D2
2238              SUB.L     D2,A1
2239 USPDONTWORRY:
2240            MOVE.L      #0,D0
2241            MOVE.L      #8,D1
2242            TRAP        #15
2243
2244                    *-----------SR-----------*
2245            LEA         RSR,A1
2246            MOVE.L      #4,D1
2247            MOVE.L      #1,D0
2248            TRAP        #15
2249            MOVE.W      (SP)+,D3
2250            MOVE.W      D3,D7    ;temp storage to restore before return
2251            JSR         HEXTOASCII
2252            MOVE.L      A2,A1
2253            SUB.L       A3,A2
2254            MOVE.L      A2,D2
2255            CMP.L       #-4,D2
2256            BEQ         SRDONTWORRY
2257 SRACCOUNTFORZEROS:
2258              ADDI.L    #4,D2
2259              SUB.L     D2,A1
2260 SRDONTWORRY:
2261            MOVE.L      #0,D0
2262            MOVE.L      #4,D1
2263            TRAP        #15
2264
2265        *-----------SS/A7-----------*
2266            LEA         RSS,A1
2267            MOVE.L      #7,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         SSDONTWORRY
2277 SSACCOUNTFORZEROS:
2278              ADDI.L    #8,D2
2279              SUB.L     D2,A1
2280 SSDONTWORRY:
2281            MOVE.L      #0,D0
2282            MOVE.L      #8,D1
2283            TRAP        #15
2284
2285        *-----------PC-----------*
2286            LEA         RPC,A1
2287            MOVE.L      #4,D1
2288            MOVE.L      #1,D0
2289            TRAP        #15
2290            MOVE.L      (SP)+,D3
2291            JSR         HEXTOASCII
2292            MOVE.L      A2,A1
2293            SUB.L       A3,A2
2294            MOVE.L      A2,D2
2295            CMP.L       #-8,D2
2296            BEQ         PCDONTWORRY
2297 PCACCOUNTFORZEROS:
2298              ADDI.L    #8,D2
2299              SUB.L     D2,A1
2300 PCDONTWORRY:
2301            MOVE.L      #0,D0
2302            MOVE.L      #8,D1
2303            TRAP        #15
2304
2305     *---DF HACK RESTORE---*
2306     MOVE.W     D7,-(SP)
2307     ADD.L      #-72,SP
2308     MOVEM.L    (SP)+,D0-D7/A0-A6
2309     ADD.L      #12,SP   ;go back to original value
2310     MOVE.W     (SP)+,SR
2311
2312            BRA SHELL
```

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 HEXTOASCII: *Result returned in address buffer from A2 to A3,
       HEX number passed*
2517                *through D3*
2518                CLR.L   D4
2519                CLR.L   D5
2520                MOVE.L  #$3A00,A2
2521                MOVE.L  #$3A32,A3
2522 KEEP_CONVERTING:
2523                MOVE.B  D3,D4
2524                MOVE.B  D3,D5
2525                LSR.L   #8,D3    ;prepare for next byte
2526                ANDI.L  #15,D4    ;get lower byte
2527                ANDI.L  #240,D5 ;get upper byte'
2528                ROR     #4,D5    ;move D5 into position
2529                CMP.B     #$9,D4
2530                BGT     A_TO_F
2531                ADDI.L  #$30,D4
2532                BRA     NEXTHEX
2533 A_TO_F:       ADDI.L  #$37,D4
2534 NEXTHEX:      CMP.B     #$9,D5
2535                BGT     A_TO_F2
2536                ADDI.L  #$30,D5
2537                BRA     PUTBUFFER
```

```
2538 A_TO_F2:       ADDI.L   #$37,D5
2539 PUTBUFFER:     MOVE.B   D4,-(A3)
2540                MOVE.B   D5,-(A3)
2541                CMP       A2,A3
2542                BNE       KEEP_CONVERTING
2543 RID_ZEROS:     CMP.B    #$30,(A2)
2544                BEQ       ADD
2545                BRA       HEXASCIIDONE
2546 ADD:           ADD.L    #1,A2    ;increment to find start of string
2547                BRA       RID_ZEROS
2548 HEXASCIIDONE:
2549                MOVE.L   #$3A32,A3     ;end of original number
2550                CMP.L    A2,A3
2551                BEQ       ZEROS
2552                BRA       NOTZEROS
2553 ZEROS          SUB.L    #8,A2
2554 NOTZEROS      RTS
```

2.3.2 ASCII to Hexadecimal

2.3.2.1 Algorithm

2.3.2.2 Assembly Code

```
2487 ASCII_ADDRESS:    *Address to be converted from ascii to hex
         passed through A2 and A3*
2488                   *Returned in D5
                                *
2489                   CLR.L    D3
2490                   CLR.L    D5
2491                   MOVE.L    A2,D1
2492                   MOVE.L    A3,D0
2493                   SUB.L    D1,D0    ;store the difference in D0
2494                   MOVE.L   #0,D4    ;set up 10's place counter
2495                   SUBI.L   #1,D0
2496 PLACECOUNTER:    CMP       #0,D0
2497                   BEQ       CONVERTADDRESS
2498                   ADDI.L   #4,D4
2499                   SUBI.L   #1,D0
2500                   BRA       PLACECOUNTER
2501 CONVERTADDRESS   CMP       A2,A3
2502                   BEQ       ADDRESSDONE
```

```
2503                    CLR.L    D3
2504                    MOVE.B   (A2)+,D3
2505                    MOVEM.L  A2-A3/D0-D2/D4-D5,-(SP)    ;so regs dont
        get destroyed
2506                    JSR      ASCII_TO_BCD
2507                    JSR      BCD_TO_HEX
2508                    MOVEM.L  (SP)+,A2-A3/D0-D2/D4-D5
2509                    ROL.L      D4,D3
2510                    SUBI.L   #4,D4
2511                    ADD.L      D3,D5    ;get total
2512                    BRA      CONVERTADDRESS
2513 ADDRESSDONE       RTS
```

2.3.3 BCD to Hexadecimal

2.3.3.1 Algorithm

2.3.3.2 Assembly Code

```
2475 BCD_TO_HEX:        *Number passed via D3 accepts BYTE ONLY*
2476                    MOVE.L   D3,D4
2477                    MOVE.L   D3,D5
2478                    ANDI.L   #240,D4 ;upper byte
2479                    ANDI.L   #15,D5  ;lower byte
2480                    ROR.L    #4,D4   ;get bits into correct place
2481                    MULU     #10,D4  ;multiply by its tens place
2482                    CLR.L    D3
2483                    ADD.L    D4,D3
2484                    ADD.L    D5,D3
2485                    RTS
```

2.3.4 ASCII to BCD

2.3.4.1 Algorithm

2.3.4.2 Assembly Code

```
2464 ASCII_TO_BCD:    *Number passed via D3 byte size only(to be
         expected)*
2465                  CMP #$46,D3
2466                  BGT ERRORSR
2467                  CMP #$40,D3
2468                  BGT UPPER
2469                  SUBI.L  #$30,D3
2470                  RTS
2471 UPPER:           SUBI.L  #$31,D3 ;If ASCII number is A-F
2472                  RTS
```

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 BERR:
2321            MOVEM.L  A1-A3/D0-D1,-(SP)
2322            LEA      BERR_TEXT,A1
2323            MOVE.L   #13,D0
2324            TRAP     #15
2325            LEA      SSW,A1
2326            MOVE.L   #14,D0
2327            TRAP     #15
2328            MOVE.W   (20,SP),D3
2329            JSR      HEXTOASCII
2330            SUB.L    #4,A3
2331            MOVEA.L  A3,A1
2332            MOVE.L   #4,D1
2333            MOVE.L   #0,D0
2334            TRAP     #15
2335            LEA      BA,A1
2336            MOVE.L   #14,D0
2337            TRAP     #15
2338            MOVE.L   (22,SP),D3
2339            JSR      HEXTOASCII
2340            SUB.L    #8,A3
2341            MOVEA.L  A3,A1
2342            MOVE.L   #8,D1
2343            MOVE.L   #0,D0
2344            TRAP     #15
2345            LEA      IR,A1
2346            MOVE.L   #14,D0
2347            TRAP     #15
2348            MOVE.W   (26,SP),D3
2349            JSR      HEXTOASCII
2350            SUB.L    #4,A3
2351            MOVEA.L  A3,A1
2352            MOVE.L   #4,D1
2353            MOVE.L   #0,D0
2354            TRAP     #15
2355            MOVEM.L  (SP)+,A1-A3/D0-D1
2356            MOVE.L   #$01000000,SP     ;reset stack
2357            BRA      SHELL
```

2.4.2 Address Error Exception

2.4.2.1 Algorithm and Flowchart

2.4.2.2 Assembly Code

```
2359 AERR:
2360            MOVEM.L  A1-A3/D0-D1,-(SP)
2361            LEA      AERR_TEXT,A1
2362            MOVE.L   #13,D0
2363            TRAP     #15
2364            LEA      SSW,A1
2365            MOVE.L   #14,D0
2366            TRAP     #15
2367            MOVE.W   (20,SP),D3
2368            JSR      HEXTOASCII
2369            SUB.L    #4,A3
2370            MOVEA.L  A3,A1
2371            MOVE.L   #4,D1
2372            MOVE.L   #0,D0
2373            TRAP     #15
2374            LEA      BA,A1
2375            MOVE.L   #14,D0
2376            TRAP     #15
2377            MOVE.L   (22,SP),D3
2378            JSR      HEXTOASCII
2379            SUB.L    #8,A3
2380            MOVEA.L  A3,A1
2381            MOVE.L   #8,D1
2382            MOVE.L   #0,D0
2383            TRAP     #15
2384            LEA      IR,A1
2385            MOVE.L   #14,D0
2386            TRAP     #15
2387            MOVE.W   (26,SP),D3
2388            JSR      HEXTOASCII
2389            SUB.L    #4,A3
2390            MOVEA.L  A3,A1
2391            MOVE.L   #4,D1
2392            MOVE.L   #0,D0
2393            TRAP     #15
2394            MOVEM.L  (SP)+,A1-A3/D0-D1
2395            MOVE.L   #$01000000,SP     ;reset stack
2396            BRA      SHELL
```

2.4.3 Illegal Instruction Error Exception

2.4.3.1 Algorithm and Flowchart

2.4.3.2 Assembly Code

```
2398 IERR:
2399        MOVEM.L  A1/D0,-(SP)
2400        LEA  IERR_TEXT,A1
2401        MOVE.L  #13,D0
2402        TRAP  #15
2403        MOVEM.L  (SP)+,A1/D0
2404        MOVE.L  #$01000000,SP
2405        BRA  SHELL
```

2.4.4 Privilege Violation Error Exception

2.4.4.1 Algorithm and Flowchart

2.4.4.2 Assembly Code

```
2407 PERR:
2408        MOVEM.L  A1/D0,-(SP)
2409        LEA  PERR_TEXT,A1
2410        MOVE.L  #13,D0
2411        TRAP  #15
2412        MOVEM.L  (SP)+,A1/D0
2413        MOVE.L  #$01000000,SP
2414        BRA  SHELL
```

2.4.5 Divide by Zero Error Exception

2.4.5.1 Algorithm and Flowchart

2.4.5.2 Assembly Code

```
2416 ZERR:
2417       MOVEM.L  A1/D0,-(SP)
2418       LEA  ZERR_TEXT,A1
2419       MOVE.L  #13,D0
2420       TRAP  #15
2421       MOVEM.L  (SP)+,A1/D0
2422       MOVE.L #$01000000 ,SP
2423       BRA  SHELL
```

2.4.6 A Line Emulator Error Exception

2.4.6.1 Algorithm and Flowchart

2.4.6.2 Assembly Code

```
2425 ALERR:
2426       MOVEM.L  A1/D0,-(SP)
2427       LEA  ALERR_TEXT,A1
2428       MOVE.L  #13,D0
2429       TRAP  #15
2430       MOVEM.L  (SP)+,A1/D0
2431       MOVE.L #$01000000 ,SP
2432       BRA  SHELL
```

2.4.7 F Line Emulator Error Exception

2.4.7.1 Algorithm and Flowchart

2.4.7.2 Assembly Code

```
2434 FLERR:
2435       MOVEM.L  A1/D0,-(SP)
2436       LEA  FLERR_TEXT,A1
2437       MOVE.L  #13,D0
2438       TRAP  #15
2439       MOVEM.L  (SP)+,A1/D0
2440       MOVE.L #$01000000 ,SP
2441       BRA  SHELL
```

2.4.8 Check Instruction Error Exception

2.4.8.1 Algorithm and Flowchart

2.4.8.2 Assembly Code

```
2443 CHKERR:
2444        MOVEM.L  A1/D0,-(SP)
2445        LEA  CHKERR_TEXT,A1
2446        MOVE.L  #13,D0
2447        TRAP  #15
2448        MOVEM.L  (SP)+,A1/D0
2449        MOVE.L  #$01000000 ,SP
2450        BRA  SHELL
```

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