Ethan Crawford

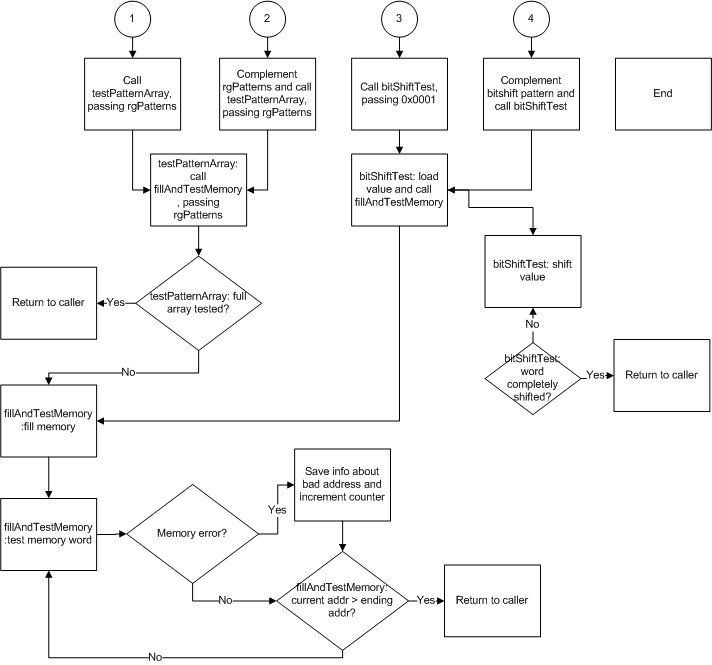
CSS 422

February 24, 2009

Homework Problem Set #5

CSS 422: Homework assignment #5: Write a memory test program.

Flow chart:



Listfile:

00000400 Starting Address

Assembler used: EASy68K Editor/Assembler v4.1.1

Created On: 2/24/2009 2:24:53 AM

00000000 1 \*----------------------------------------------------------------------------

00000000 2 \* Program : Homework 5

00000000 3 \* Written by : Ethan Crawford

00000000 4 \* Course : CSS 422

00000000 5 \* Date : February 24, 2009

00000000 6 \* Description: Implements a memory test program.

00000000 7 \*----------------------------------------------------------------------------

00000000 8

00000000 9 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

00000000 10 \*

00000000 11 \* Beginning of code segment.

00000000 12 \*

00000000 13 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

00000000 14 OPT CRE

00000000 15

00000400 16 START ORG $400

00000400 2E7C 000A0000 17 MAIN MOVEA.L #lStackPtr,A7 \* Init stack pointer

00000406 42B9 00000516 18 CLR.L lBadAddr

0000040C 4279 0000051A 19 CLR.W wBadCount

00000412 4279 0000051C 20 CLR.W wDataWritten

00000418 4279 0000051E 21 CLR.W wDataRead

0000041E 22

0000041E 23 \* fill and test memory with rgPatterns

0000041E 2F3C 00000512 24 MOVE.L #rgPatterns,-(A7) \* Push rgPatterns onto the stack

00000424 4EB9 00000496 25 JSR testPatternArray

0000042A 588F 26 ADDQ.L #4,A7 \* Deallocate parameter

0000042C 27

0000042C 28 \* Complement rgPatterns and fill and test memory with rgPatterns

0000042C 307C 0512 29 MOVEA.W #rgPatterns,A0 \* Initialize patterns array pointer

00000430 3210 30 NOTP MOVE.W (A0),D1

00000432 4641 31 NOT.W D1 \* Complement the word

00000434 3081 32 MOVE.W D1,(A0) \* Store the complemented word

00000436 5488 33 ADDQ.L #2,A0 \* Move to the next word

00000438 5202 34 ADDQ.B #1,D2 \* Increment counter

0000043A B43C 0002 35 CMP.B #cPatterns,D2 \* Check counter

0000043E 6DF0 36 BLT NOTP \* Loop again if patterns remain

00000440 37

00000440 2F3C 00000512 38 MOVE.L #rgPatterns,-(A7) \* Push rgPatterns onto the stack

00000446 4EB9 00000496 39 JSR testPatternArray

0000044C 588F 40 ADDQ.L #4,A7 \* Deallocate parameter

0000044E 41

0000044E 42 \* Bit-shift wBitShift 16 times and fill and test memory

0000044E 7201 43 MOVE.L #1,D1

00000450 2F01 44 MOVE.L D1,-(A7) \* Push wShiftValue onto the stack

00000452 4EB9 0000046C 45 JSR bitShiftTest

00000458 588F 46 ADDQ.L #4,A7 \* Deallocate parameter

0000045A 47

0000045A 48 \* Complement bitshift pattern and fill and test memory

0000045A 4641 49 NOT.W D1

0000045C 2F01 50 MOVE.L D1,-(A7) \* Push wShiftValue onto the stack

0000045E 4EB9 0000046C 51 JSR bitShiftTest

00000464 588F 52 ADDQ.L #4,A7 \* Deallocate parameter

00000466 53

00000466 4EF9 0000050C 54 JMP EXIT \* Exit the program

0000046C 55

0000046C 56 \* void bitShift(word wShiftValue)

0000046C 57 bitShiftTest

0000046C 48E7 6000 58 MOVEM.L D1-D2,-(A7)

00000470 4281 59 CLR.L D1 \* Clear loop counter

00000472 242F 000C 60 MOVE.L 12(A7),D2 \* Load bitshift value

00000476 2F02 61 BSHIFT MOVE.L D2,-(A7) \* Push bitshifted value onto the stack

00000478 62

00000478 4EB9 000004C0 63 JSR fillAndTestMemory

0000047E 588F 64 ADDQ.L #4,A7 \* Deallocate parameter

00000480 D179 0000051A 65 ADD.W D0,wBadCount \* Add the return value to wBadCount

00000486 66

00000486 E35A 67 ROL.W #1,D2 \* Bitshift D0

00000488 5201 68 ADDQ.B #1,D1 \* Increment counter

0000048A B23C 0010 69 CMP.B #bWordSize,D1 \* Check counter

0000048E 6DE6 70 BLT BSHIFT \* Loop again if patterns remain

00000490 4CDF 0006 71 MOVEM.L (A7)+,D1-D2

00000494 4E75 72 RTS

00000496 73

00000496 74 \* void testPatternArray(word\* rgPatterns)

00000496 75 testPatternArray

00000496 48E7 C080 76 MOVEM.L A0/D0-D1,-(A7)

0000049A 4281 77 CLR.L D1 \* Clear loop counter

0000049C 206F 0010 78 MOVEA.L 16(A7),A0 \* Get rgPatterns

000004A0 3218 79 PL MOVE.W (A0)+,D1 \* Put the nth pattern into D0

000004A2 80

000004A2 2F01 81 MOVE.L D1,-(A7) \* Push wPattern onto the stack

000004A4 4EB9 000004C0 82 JSR fillAndTestMemory \* Call the subroutine

000004AA 588F 83 ADDQ.L #4,A7 \* Deallocate parameter

000004AC D179 0000051A 84 ADD.W D0,wBadCount \* Add the return value to wBadCount

000004B2 85

000004B2 5201 86 ADDQ.B #1,D1 \* Increment counter

000004B4 B23C 0002 87 CMP.B #cPatterns,D1 \* Check counter

000004B8 6DE6 88 BLT PL \* Loop again if patterns remain

000004BA 4CDF 0103 89 MOVEM.L (A7)+,A0/D0-D1

000004BE 4E75 90 RTS

000004C0 91

000004C0 92 \* int fillAndTestMemory(word wPattern)

000004C0 93 fillAndTestMemory

000004C0 48E7 6060 94 MOVEM.L A1/A2/D1/D2,-(A7)

000004C4 222F 0014 95 MOVE.L 20(A7),D1 \* Get wPattern

000004C8 4280 96 CLR.L D0 \* Clear return value

000004CA 97

000004CA 98 \* Fill memory block

000004CA 227C 00001000 99 MOVE.L #lStartAddr,A1 \* Init start

000004D0 247C 0003FFFF 100 MOVE.L #lEndAddr,A2 \* Init end

000004D6 3281 101 FMB MOVE.W D1,(A1) \* Fill word

000004D8 5489 102 ADDQ.L #2,A1 \* Advance address another word

000004DA B3CA 103 CMPA.L A2,A1 \* Loop while lStartAddr <= lEndAddr

000004DC 6FF8 104 BLE FMB

000004DE 105

000004DE 106 \* Test memory block

000004DE 227C 00001000 107 MOVE.L #lStartAddr,A1 \* Init start again

000004E4 108

000004E4 3411 109 TMB MOVE.W (A1),D2 \* Get memory value

000004E6 B441 110 CMP.W D1,D2 \* Compare retrieved value to wPattern

000004E8 6700 0016 111 BEQ TMBEQ \* If the values are equal, go to the next word

000004EC 112

000004EC 5280 113 ADDQ.L #1,D0 \* Else, increment the return value and store interesting data

000004EE 23C9 00000516 114 MOVE.L A1,lBadAddr \* Store bad address

000004F4 33C1 0000051C 115 MOVE.W D1,wDataWritten \* Store data written

000004FA 33C2 0000051E 116 MOVE.W D2,wDataRead \* Store data read

00000500 117

00000500 5489 118 TMBEQ ADDQ.L #2,A1 \* Advance address another word

00000502 B3CA 119 CMPA.L A2,A1 \* Loop while lStartAddr <= lEndAddr

00000504 6FDE 120 BLE TMB

00000506 121

00000506 122 \* Epilogue

00000506 4CDF 0606 123 MOVEM.L (A7)+,A1/A2/D1/D2

0000050A 4E75 124 RTS

0000050C 125

0000050C 103C 0009 126 EXIT MOVE.B #9,D0

00000510 4E4F 127 TRAP #15 Halt Simulator

00000512 128

00000512 129 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

00000512 130 \*

00000512 131 \* EQUates section

00000512 132 \*

00000512 133 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

00000512 =00001000 134 lStartAddr EQU $00001000 \* starting memory address

00000512 =0003FFFF 135 lEndAddr EQU $0003FFFF \* ending memory address

00000512 =000A0000 136 lStackPtr EQU $000A0000 \* stack pointer address

00000512 =00000010 137 bWordSize EQU 16 \* Number of bits in a word

00000512 =00000002 138 cPatterns EQU 2 \* Patterns to test and complement

00000512= FFFF AAAA 139 rgPatterns DC.W $FFFF,$AAAA \* Patterns to test and complement

00000516 140 lBadAddr DS.L 1 \* Address of latest bad memory location

0000051A 141 wBadCount DS.W 1 \* Number of memory test errors found

0000051C 142 wDataWritten DS.W 1 \* Data written to latest bad memory location

0000051E 143 wDataRead DS.W 1 \* Data read from latest bad memory location

00000520 144 END START

No errors detected

No warnings generated

SYMBOL TABLE INFORMATION

Symbol-name Value

-------------------------

BITSHIFTTEST 46C

BSHIFT 476

BWORDSIZE 10

CPATTERNS 2

EXIT 50C

FILLANDTESTMEMORY 4C0

FMB 4D6

LBADADDR 516

LENDADDR 3FFFF

LSTACKPTR A0000

LSTARTADDR 1000

MAIN 400

NOTP 430

PL 4A0

RGPATTERNS 512

START 400

TESTPATTERNARRAY 496

TMB 4E4

TMBEQ 500

WBADCOUNT 51A

WDATAREAD 51E

WDATAWRITTEN 51C