For each test case within each problem, just replace the array of numbers or array of strings and strings with what you want and it should work. For the submitted code, I used the default cases. I included some extra tests just to show the code works for other more nuanced cases and as a way for myself to check. Feel free to investigate or ignore them.

P1 Code:

```
;******************** (C) Andrew Wolfe ********************************************
; @file    HW7 Problem 1
; @author  Andrew Wolfe
; @date    2023
;**********************************************************************************


                        AREA    main, CODE, READONLY
                        EXPORT          __main
                        ENTRY

__main PROC
                        ldr r4, =5;test to make sure subroutine preserved register values are indeed
preserved
                        ldr r5, =5
                        ldr r6, =5
                        ldr r7, =5
                        ldr r8, =5
                        ldr r9, =5
                        ldr r10, =5
                        ldr r11, =5
                        ldr                 r0, =array          ; Pointer to array of string pointers
                        ldr                 r1, =endofarray                     ; end of array
                        push {r11,r10,r9,r8,r7,r6,r5,r4}
                        bl                  mysort                  ; Call sorting routine
                        pop {r4,r5,r6,r7,r8,r9,r10,r11}
endless         b               endless

                ENDP


                                        ; registers 4-11 must be preserved while in subroutines which is
why they are pushed and popped


mysort  PROC
                        ldr r4, =0
                        push{lr}
loop0           bl      compare_and_swap
                        ldr r0, =array
                        add r4, #4                      ;increments r0 up to the next value after a full
loop
                        add r0, r4
                        cmp r1, r0
                        bne     loop0
                        pop{lr}
end                     bx                  lr
                        ENDP
```

```
compare_and_swap    PROC
                    mov r8, #1     ;counter
                    mov r9, #0                  ;r9 and r6 are used for constants in the switching
process
                    mov r6, #0
                    ldr r2, [r0]
                    add r0, #4
                    cmp r0, r1
                    beq back     ; check in case we've reached the end
loop1           ldr r3, [r0]
                    mov r6, #-4    ;used for offset
                    cmp r3, r2
                    blt swap
                    add r0, #4
                    add r8, #1
back            cmp r1, r0
                    bne      loop1
                    bx lr
swap            mov r7, r2                  ;swap with offset of r8 times r6(-4)
                    mov r2, r3
                    mov r3, r7
                    mul r9, r8, r6
                    str r2, [r0, r9]
                    str r3, [r0]
                    b back
                    ENDP

                    ALIGN
                    AREA mydata, DATA, READONLY


array           DCD             9,2,5,1,8,6,7,0,3,4
endofarray


                    END
```

P2 Code:

```
;******************** (C) Andrew Wolfe ******************************************
; @file    HW7 Problem 2
; @author  Andrew Wolfe
; @date    2023
;*******************************************************************************


                    AREA    main, CODE, READONLY
                    EXPORT          __main
                    ENTRY

__main PROC
```

```
                ldr r4, =5 ;demonstrate values are conserved
                ldr r5, =5
                ldr r6, =5
                ldr r7, =5
                ldr r8, =5
                ldr r9, =5
                ldr r10, =5
                ldr r11, =5
                ldr     r0, =strarray      ; Pointer to array of string pointers
                ldr     r1, =endofarray
                push{r11,r10,r9,r8,r7,r6,r5,r4}
                bl      mysort             ; Call sorting routine
                pop{r4,r5,r6,r7,r8,r9,r10,r11}
endless     b       endless

            ENDP



mysort  PROC
                ldr r4, =0
                push{lr}
loop0       bl  compare_and_swap
                ldr r0, =strarray
                add r4, #4                 ;increments r0 up to the next value after a full
loop
                add r0, r4
                cmp r1, r0
                bne     loop0
                pop{lr}
end             bx      lr
                ENDP

compare_and_swap    PROC
                mov r8, #1    ;counter
                mov r9, #0          ;r9 and r6 are used for constants in the switching
process
                mov r7, #0
                mov r10, #0
                ldr r5, =0xffffff00
                add r0, #4
                cmp r0, r1
                beq back    ; check in case we've reached the end
                sub r0, #4
                mov r6, #-4   ;used for offset
loop1       b compare
continue    add r0, #4
                add r8, #1
                mul r9, r8, r6
                add r9, #4
                ldr r5, =0xffffff00
                mov r10, #0
                mov r7, #0
back        cmp r1, r0
```

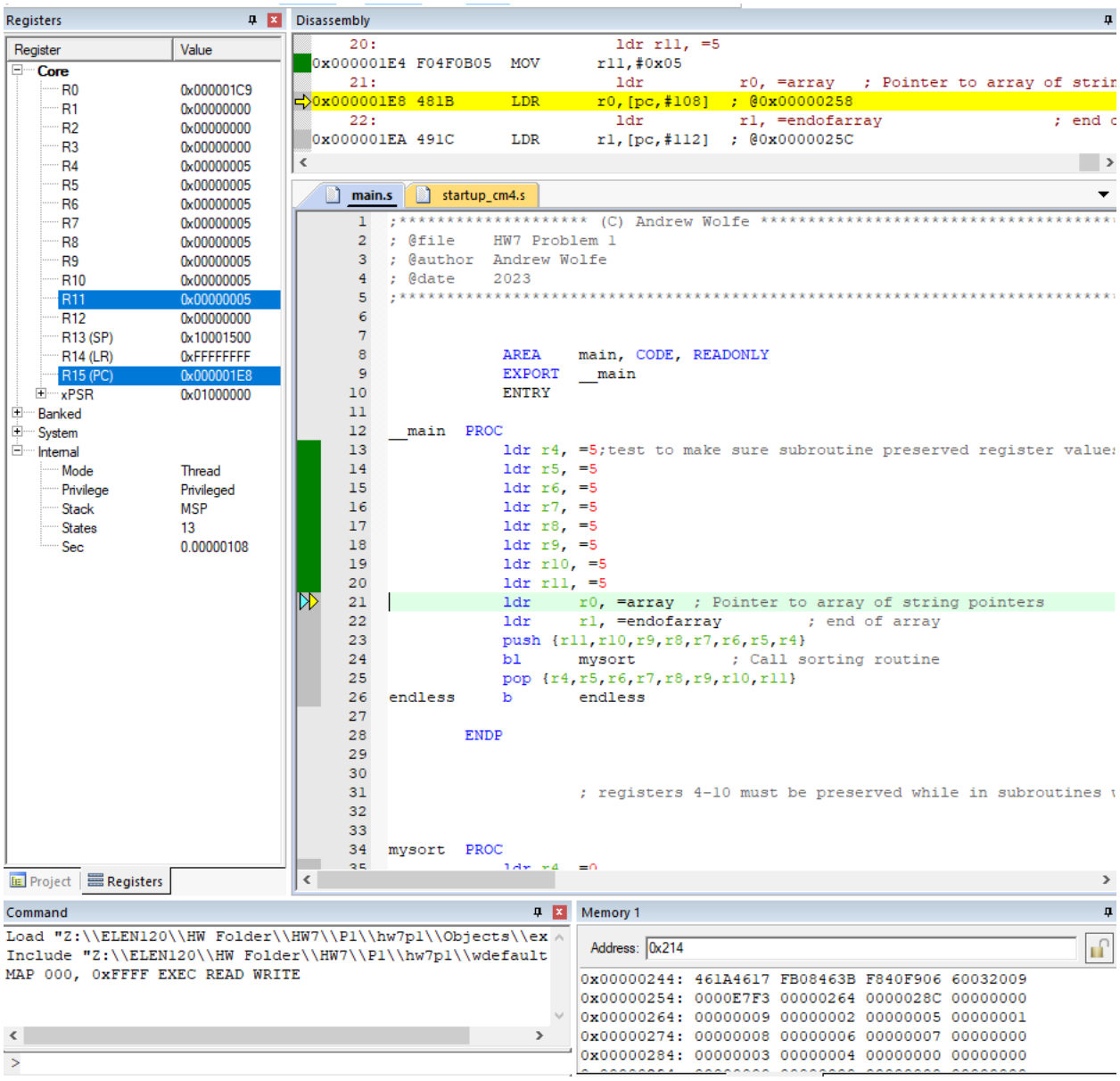
```
                bne     loop1
                bx lr
swap            mul r9, r8, r6
                add r9, #4
                ldr r2, [r0, r9]
                ldr r3, [r0]
                mov r7, r2              ;swap with offset of r8 times r6(-4)
                mov r2, r3
                mov r3, r7
                mov r7, #0
                str r2, [r0, r9]
                str r3, [r0]
                b back
compare         cmp r8, #1
                beq continue
                ldr r2, [r0, r9]
                ldr r3, [r0]
                add r2, r10
                add r3, r10
                ldr r2, [r2]
                ldr r3, [r3]
                bic r2, r5
                bic r3, r5
                b ridcapitals
return2         cmp r3, r2
                blt swap
                bne continue
                cmp r3, #0                      ;case for when we are at the end and r3=r2 (they
are the same string)
                beq continue
                ror r5, #24
                add r7, #1
                cmp r7, #4
                beq nextword
                b compare
nextword        add r10, #4 ;if first four letters are the same, move to next 32-bit string (thats what I mean
by word). It's not the actual next word in the strarray
                mov r7, #0
                b compare
ridcapitals push {r10}
                mov r10, #0x5b5b5b5b ;check if r2 is capital
                bic r10, r5
                cmp r2, r10
                blt action1
return0         mov r10, #0x5b5b5b5b ;check if r3 is capital
                bic r10, r5
                cmp r3, r10
                blt action2
return1         pop {r10}
                b return2
action1         mov r10, #0x20202020 ;if r2 is capital convert to lower case
                bic r10, r5
                add r2, r10
                b return0
action2         mov r10, #0x20202020 ;if r3 is capital convert to lower case
                bic r10, r5
```

```
                        add r3, r10
                        b return1
                        ENDP

                        ALIGN
                        AREA mydata, DATA, READONLY

                        AREA mydata, DATA, READONLY
str1 DCB "First string",0
str2 DCB "Second string",0
str3 DCB "So, do I really need a third string",0
str4 DCB "Tetraphobia is the fear of the number 4",0
str5 DCB "A is for apple",0
str6 DCB "Z is called \'zed\' in Canada",0
str7 DCB "M is for middle",0
strarray DCD    str1, str2, str3, str4, str5, str6, str7
endofarray
                END
```

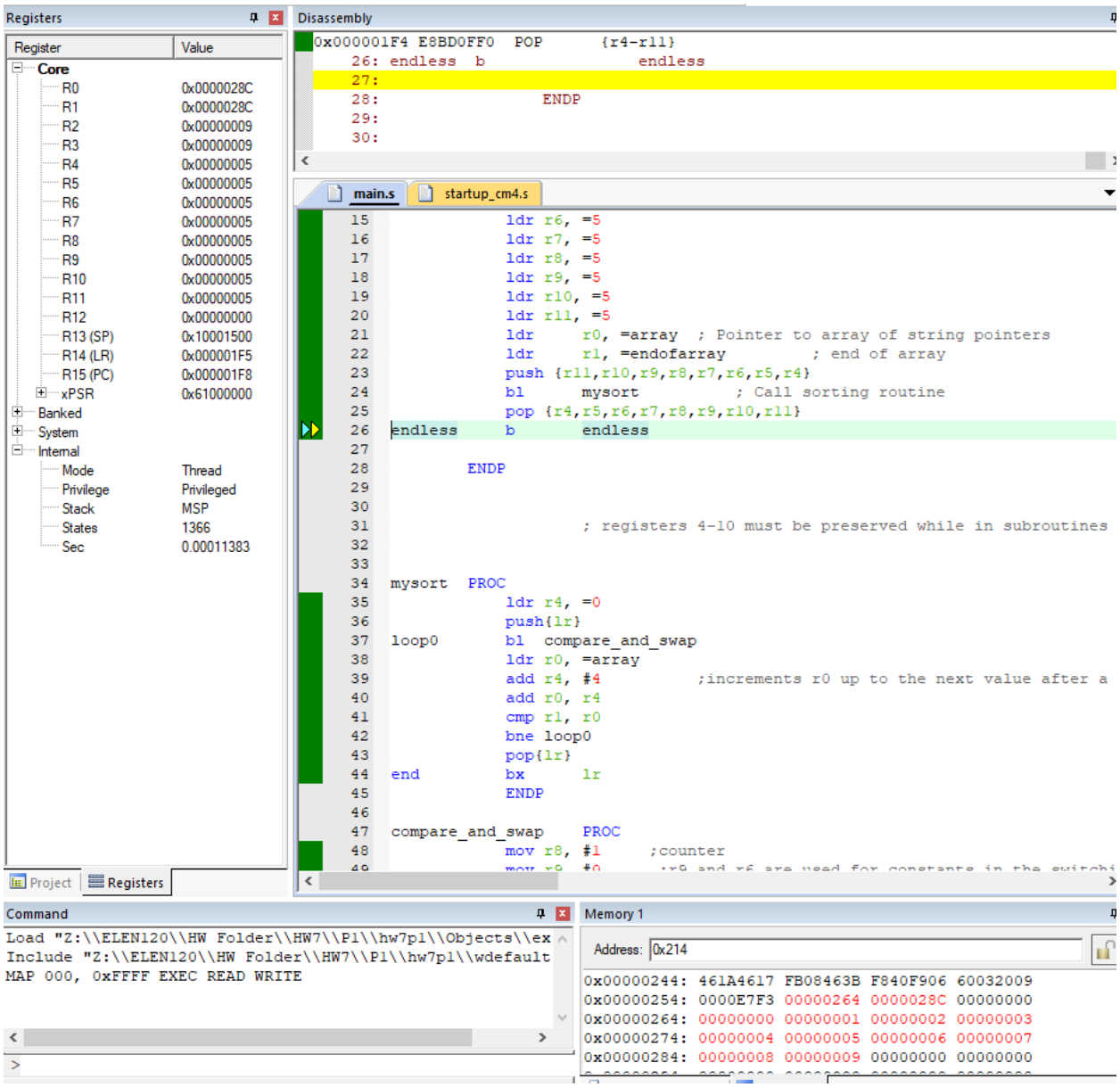
P1

TEST 1

Before:

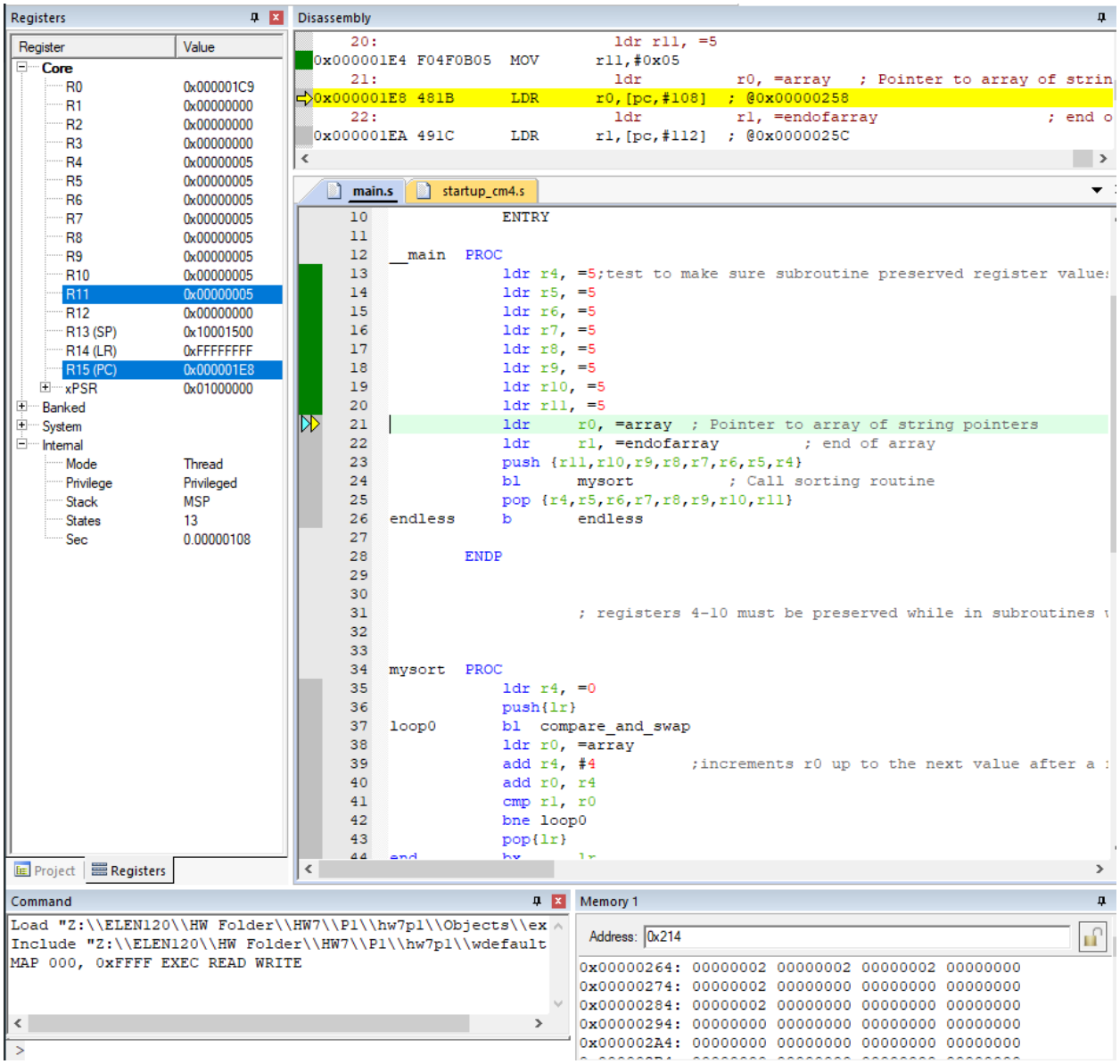


After:

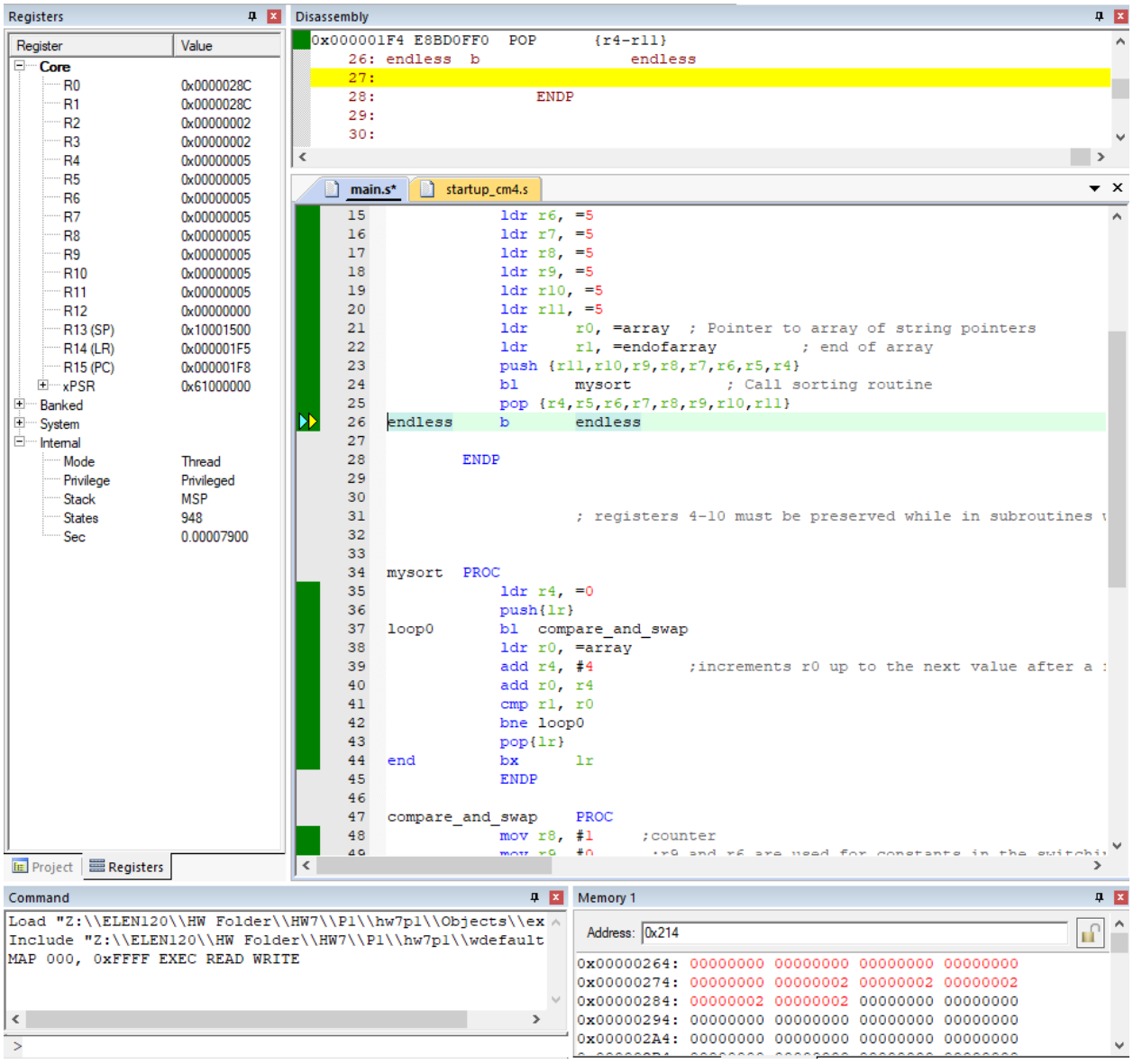

Registers
Register
Value
Core
R0 0x0000028C
R1 0x0000028C
R2 0x00000009
R3 0x00000009
R4 0x00000005
R5 0x00000005
R6 0x00000005
R7 0x00000005
R8 0x00000005
R9 0x00000005
R10 0x00000005
R11 0x00000005
R12 0x00000000
R13 (SP) 0x10001500
R14 (LR) 0x000001F5
R15 (PC) 0x000001F8
xPSR 0x61000000
Banked
System
Internal
Mode Thread
Privilege Privileged
Stack MSP
States 1366
Sec 0.00011383
Project
Registers
Disassembly
0x000001F4 E8BD0FF0 POP {r4-r11}
26: endless b endless
27:
28: ENDP
29:
30:
main.s
startup_cm4.s
15 ldr r6, =5
16 ldr r7, =5
17 ldr r8, =5
18 ldr r9, =5
19 ldr r10, =5
20 ldr r11, =5
21 ldr r0, =array ; Pointer to array of string pointers
22 ldr r1, =endofarray ; end of array
23 push {r11,r10,r9,r8,r7,r6,r5,r4}
24 bl mysort ; Call sorting routine
25 pop {r4,r5,r6,r7,r8,r9,r10,r11}
26 endless b endless
27
28 ENDP
29
30
31 ; registers 4-10 must be preserved while in subroutines
32
33
34 mysort PROC
35 ldr r4, =0
36 push{lr}
37 loop0 bl compare_and_swap
38 ldr r0, =array
39 add r4, #4 ;increments r0 up to the next value after a
40 add r0, r4
41 cmp r1, r0
42 bne loop0
43 pop{lr}
44 end bx lr
45 ENDP
46
47 compare_and_swap PROC
48 mov r8, #1 ;counter
Command
Load "Z:\\ELEN120\\HW Folder\\HW7\\P1\\hw7p1\\Objects\\ex
Include "Z:\\ELEN120\\HW Folder\\HW7\\P1\\hw7p1\\wdefault
MAP 000, 0xFFFF EXEC READ WRITE
Memory 1
Address: 0x214
0x00000244: 461A4617 FB08463B F840F906 60032009
0x00000254: 0000E7F3 00000264 0000028C 00000000
0x00000264: 00000000 00000001 00000002 00000003
0x00000274: 00000004 00000005 00000006 00000007
0x00000284: 00000008 00000009 00000000 00000000

TEST 2

Before:

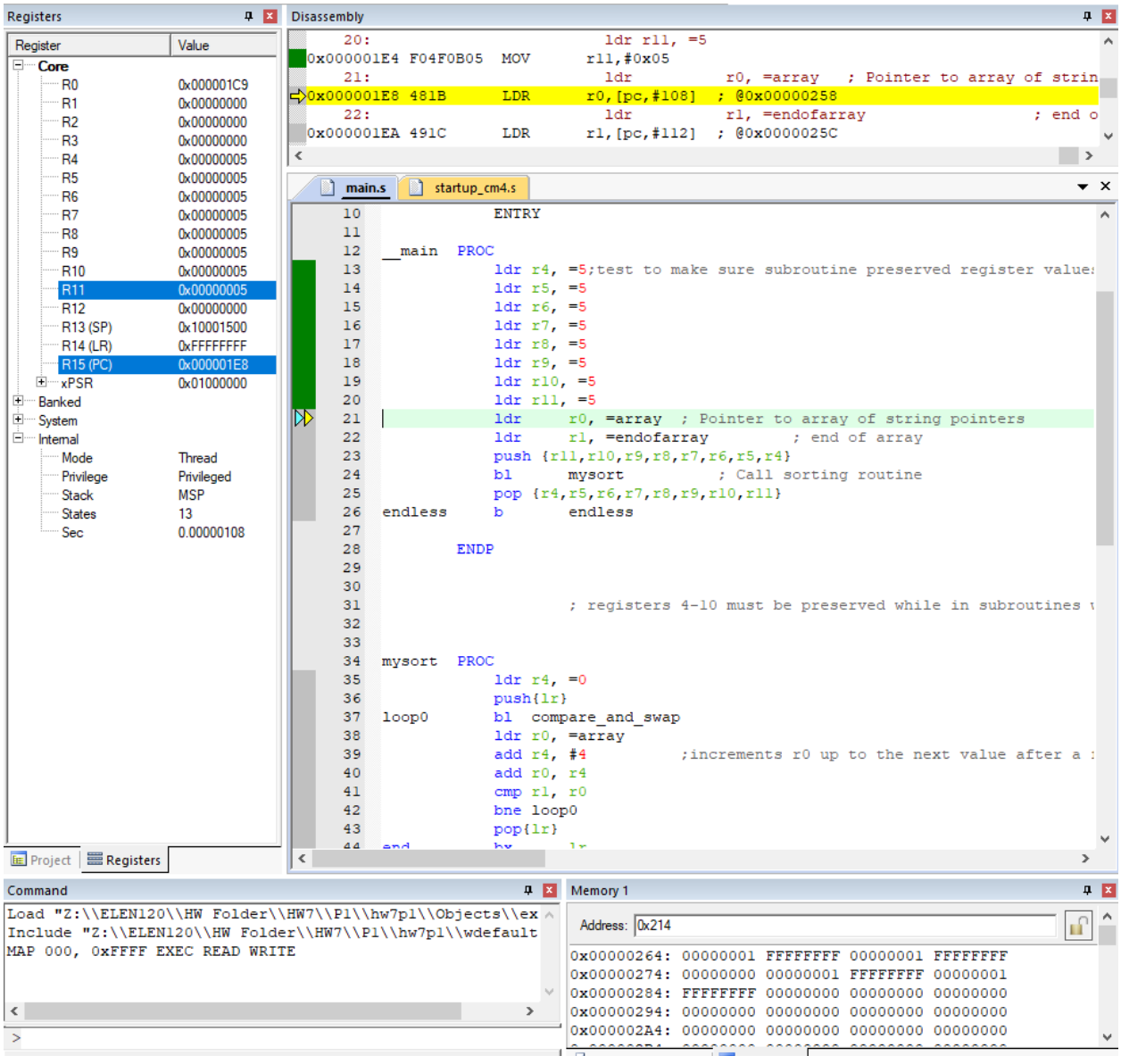


After:

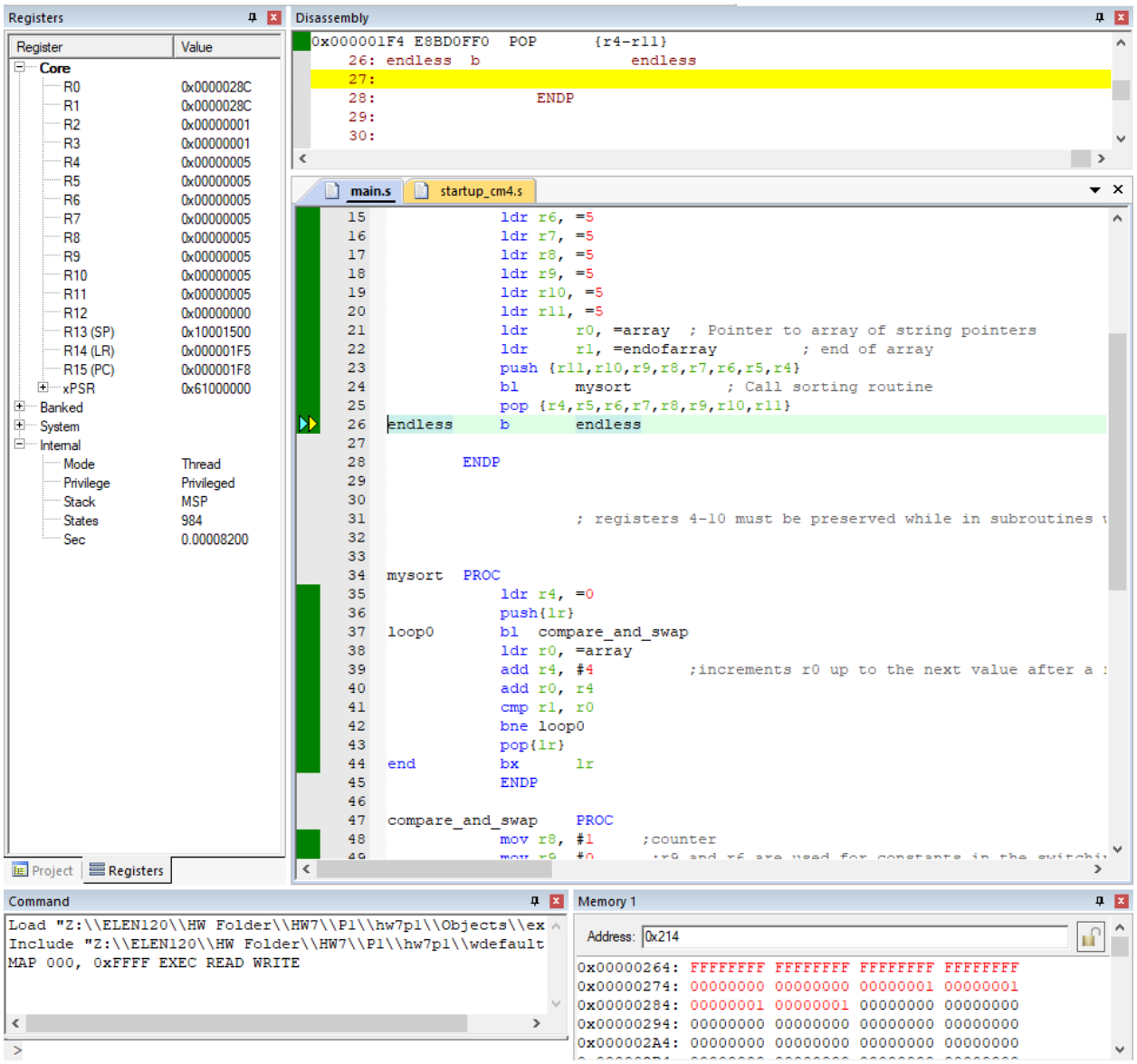

Registers
Register
Value
Core
R0 0x0000028C
R1 0x0000028C
R2 0x00000002
R3 0x00000002
R4 0x00000005
R5 0x00000005
R6 0x00000005
R7 0x00000005
R8 0x00000005
R9 0x00000005
R10 0x00000005
R11 0x00000005
R12 0x00000000
R13 (SP) 0x10001500
R14 (LR) 0x000001F5
R15 (PC) 0x000001F8
xPSR 0x61000000
Banked
System
Internal
Mode Thread
Privilege Privileged
Stack MSP
States 948
Sec 0.00007900
Project
Registers
Disassembly
0x000001F4 E8BD0FF0 POP {r4-r11}
26: endless b endless
27:
28: ENDP
29:
30:
main.s*
startup_cm4.s
15 ldr r6, =5
16 ldr r7, =5
17 ldr r8, =5
18 ldr r9, =5
19 ldr r10, =5
20 ldr r11, =5
21 ldr r0, =array ; Pointer to array of string pointers
22 ldr r1, =endofarray ; end of array
23 push {r11,r10,r9,r8,r7,r6,r5,r4}
24 bl mysort ; Call sorting routine
25 pop {r4,r5,r6,r7,r8,r9,r10,r11}
26 endless b endless
27
28 ENDP
29
30
31 ; registers 4-10 must be preserved while in subroutines
32
33
34 mysort PROC
35 ldr r4, =0
36 push{lr}
37 loop0 bl compare_and_swap
38 ldr r0, =array
39 add r4, #4 ;increments r0 up to the next value after a
40 add r0, r4
41 cmp r1, r0
42 bne loop0
43 pop{lr}
44 end bx lr
45 ENDP
46
47 compare_and_swap PROC
48 mov r8, #1 ;counter
Command
Load "Z:\\ELEN120\\HW Folder\\HW7\\P1\\hw7p1\\Objects\\ex
Include "Z:\\ELEN120\\HW Folder\\HW7\\P1\\hw7p1\\wdefault
MAP 000, 0xFFFF EXEC READ WRITE
Memory 1
Address: 0x214
0x00000264: 00000000 00000000 00000000 00000000
0x00000274: 00000000 00000002 00000002 00000002
0x00000284: 00000002 00000002 00000000 00000000
0x00000294: 00000000 00000000 00000000 00000000
0x000002A4: 00000000 00000000 00000000 00000000

TEST 3

Before:

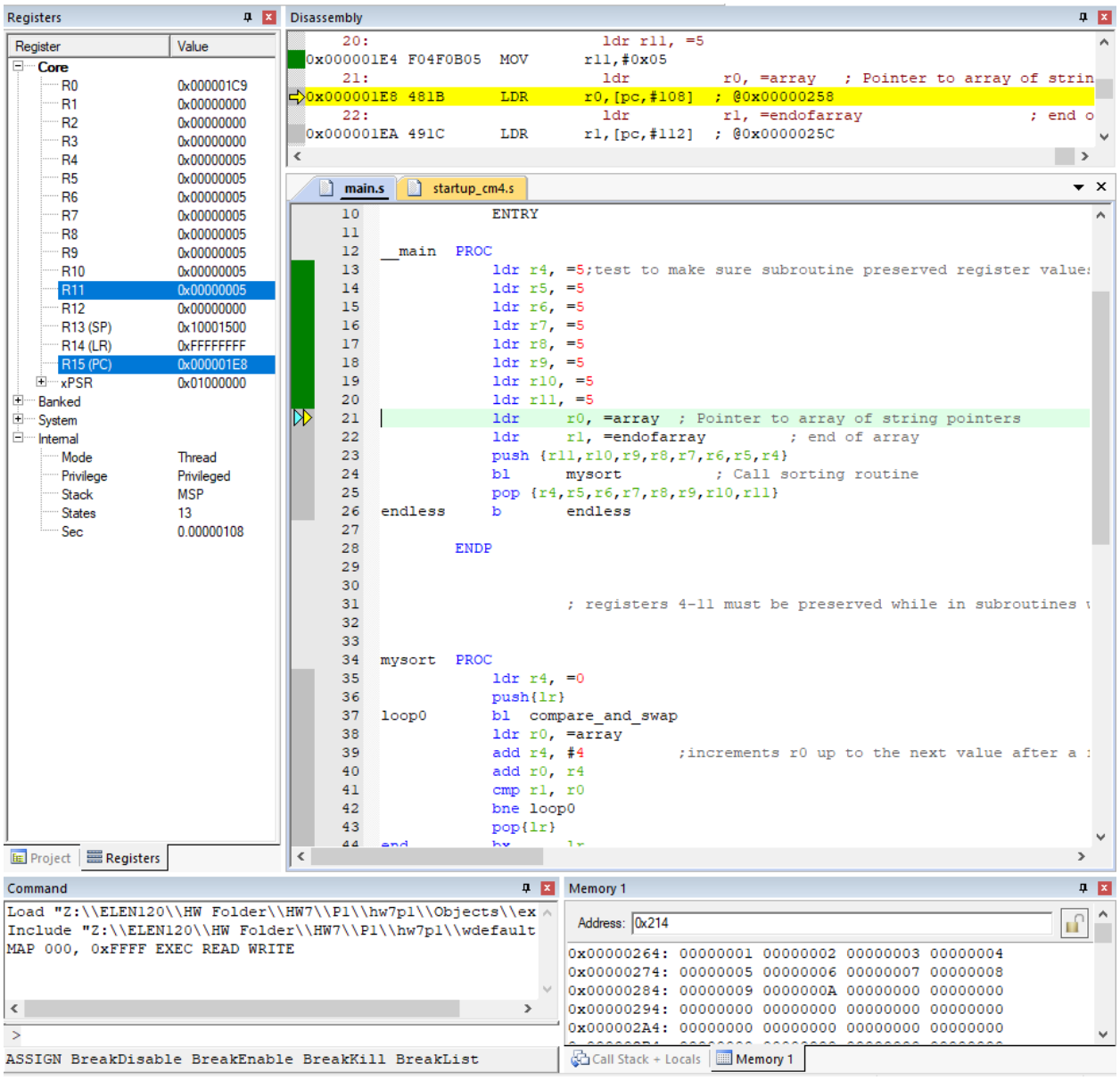


After:

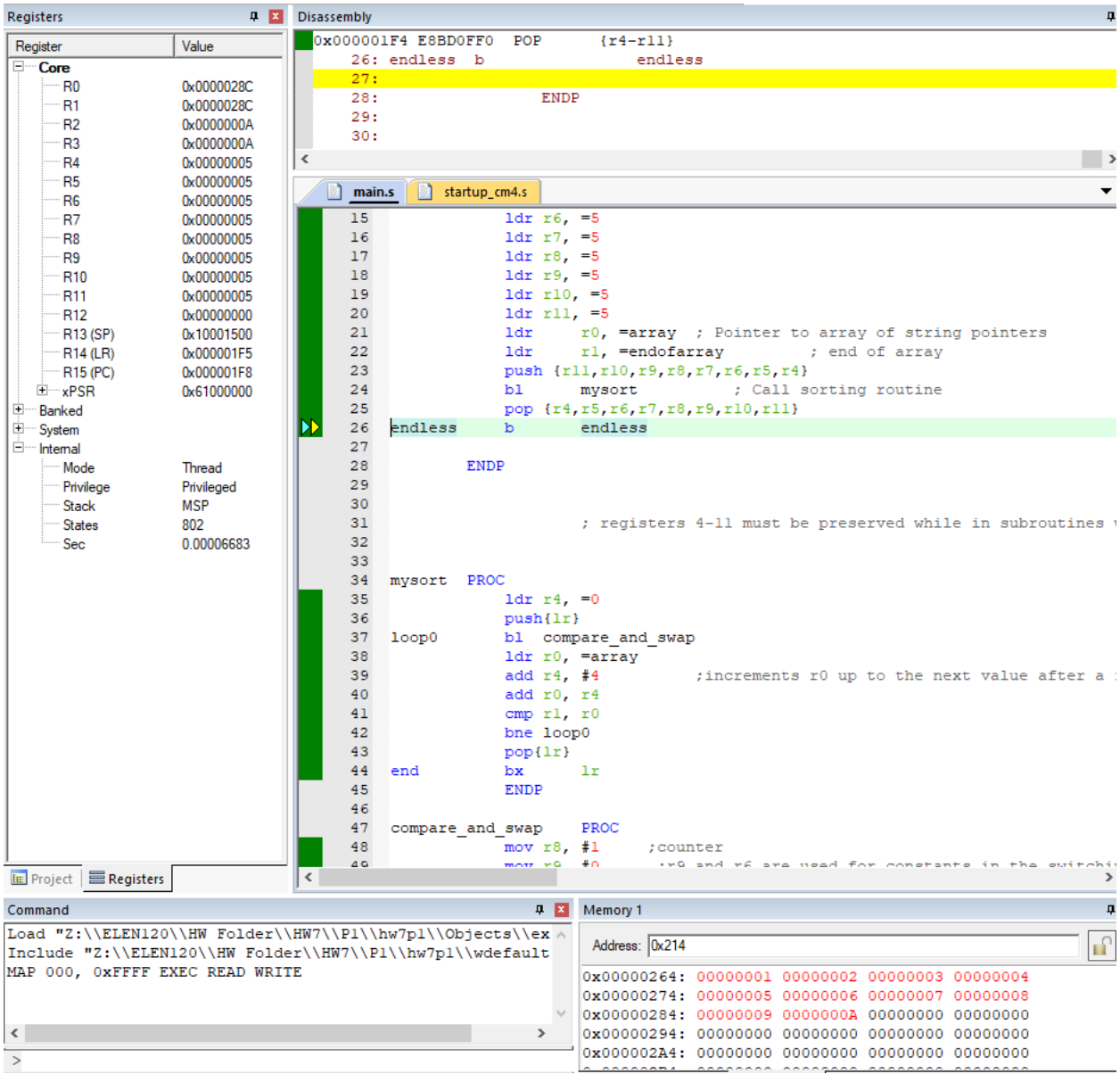

Registers
Register
Value
Core
R0 0x0000028C
R1 0x0000028C
R2 0x00000001
R3 0x00000001
R4 0x00000005
R5 0x00000005
R6 0x00000005
R7 0x00000005
R8 0x00000005
R9 0x00000005
R10 0x00000005
R11 0x00000005
R12 0x00000000
R13 (SP) 0x10001500
R14 (LR) 0x000001F5
R15 (PC) 0x000001F8
xPSR 0x61000000
Banked
System
Internal
Mode Thread
Privilege Privileged
Stack MSP
States 984
Sec 0.00008200
Project
Registers
Disassembly
0x000001F4 E8BD0FF0 POP {r4-r11}
26: endless b endless
27:
28: ENDP
29:
30:
main.s
startup_cm4.s
15 ldr r6, =5
16 ldr r7, =5
17 ldr r8, =5
18 ldr r9, =5
19 ldr r10, =5
20 ldr r11, =5
21 ldr r0, =array ; Pointer to array of string pointers
22 ldr r1, =endofarray ; end of array
23 push {r11,r10,r9,r8,r7,r6,r5,r4}
24 bl mysort ; Call sorting routine
25 pop {r4,r5,r6,r7,r8,r9,r10,r11}
26 endless b endless
27
28 ENDP
29
30
31 ; registers 4-10 must be preserved while in subroutines
32
33
34 mysort PROC
35 ldr r4, =0
36 push{lr}
37 loop0 bl compare_and_swap
38 ldr r0, =array
39 add r4, #4 ;increments r0 up to the next value after a
40 add r0, r4
41 cmp r1, r0
42 bne loop0
43 pop{lr}
44 end bx lr
45 ENDP
46
47 compare_and_swap PROC
48 mov r8, #1 ;counter
Command
Load "Z:\\ELEN120\\HW Folder\\HW7\\P1\\hw7p1\\Objects\\ex
Include "Z:\\ELEN120\\HW Folder\\HW7\\P1\\hw7p1\\wdefault
MAP 000, 0xFFFF EXEC READ WRITE
Memory 1
Address: 0x214
0x00000264: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
0x00000274: 00000000 00000000 00000001 00000001
0x00000284: 00000001 00000001 00000000 00000000
0x00000294: 00000000 00000000 00000000 00000000
0x000002A4: 00000000 00000000 00000000 00000000

TEST 4

Before:



After:


Registers
Register
Value
Core
R0 0x0000028C
R1 0x0000028C
R2 0x0000000A
R3 0x0000000A
R4 0x00000005
R5 0x00000005
R6 0x00000005
R7 0x00000005
R8 0x00000005
R9 0x00000005
R10 0x00000005
R11 0x00000005
R12 0x00000000
R13 (SP) 0x10001500
R14 (LR) 0x000001F5
R15 (PC) 0x000001F8
xPSR 0x61000000
Banked
System
Internal
Mode Thread
Privilege Privileged
Stack MSP
States 802
Sec 0.00006683
Project
Registers
Disassembly
0x000001F4 E8BD0FF0 POP {r4-r11}
26: endless b endless
27:
28: ENDP
29:
30:
main.s
startup_cm4.s
15 ldr r6, =5
16 ldr r7, =5
17 ldr r8, =5
18 ldr r9, =5
19 ldr r10, =5
20 ldr r11, =5
21 ldr r0, =array ; Pointer to array of string pointers
22 ldr r1, =endofarray ; end of array
23 push {r11,r10,r9,r8,r7,r6,r5,r4}
24 bl mysort ; Call sorting routine
25 pop {r4,r5,r6,r7,r8,r9,r10,r11}
26 endless b endless
27
28 ENDP
29
30
31 ; registers 4-11 must be preserved while in subroutines
32
33
34 mysort PROC
35 ldr r4, =0
36 push{lr}
37 loop0 bl compare_and_swap
38 ldr r0, =array
39 add r4, #4 ;increments r0 up to the next value after a
40 add r0, r4
41 cmp r1, r0
42 bne loop0
43 pop{lr}
44 end bx lr
45 ENDP
46
47 compare_and_swap PROC
48 mov r8, #1 ;counter
Command
Load "Z:\\ELEN120\\HW Folder\\HW7\\P1\\hw7p1\\Objects\\ex
Include "Z:\\ELEN120\\HW Folder\\HW7\\P1\\hw7p1\\wdefault
MAP 000, 0xFFFF EXEC READ WRITE
Memory 1
Address: 0x214
0x00000264: 00000001 00000002 00000003 00000004
0x00000274: 00000005 00000006 00000007 00000008
0x00000284: 00000009 0000000A 00000000 00000000
0x00000294: 00000000 00000000 00000000 00000000
0x000002A4: 00000000 00000000 00000000 00000000

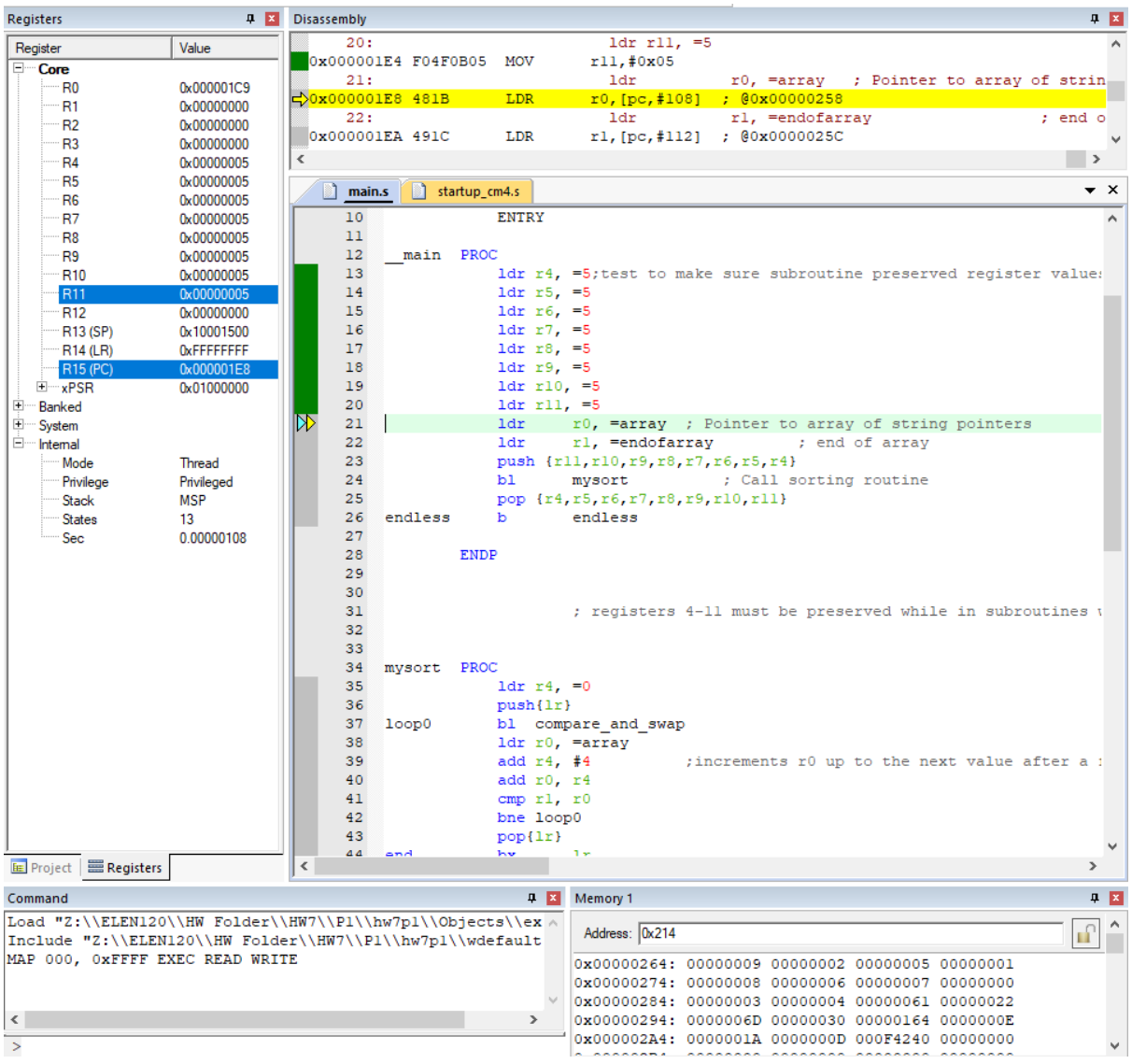
TEST 5 (extra test)

!THIS IS NOT A PART OF THE STANDARD RUBRIC!

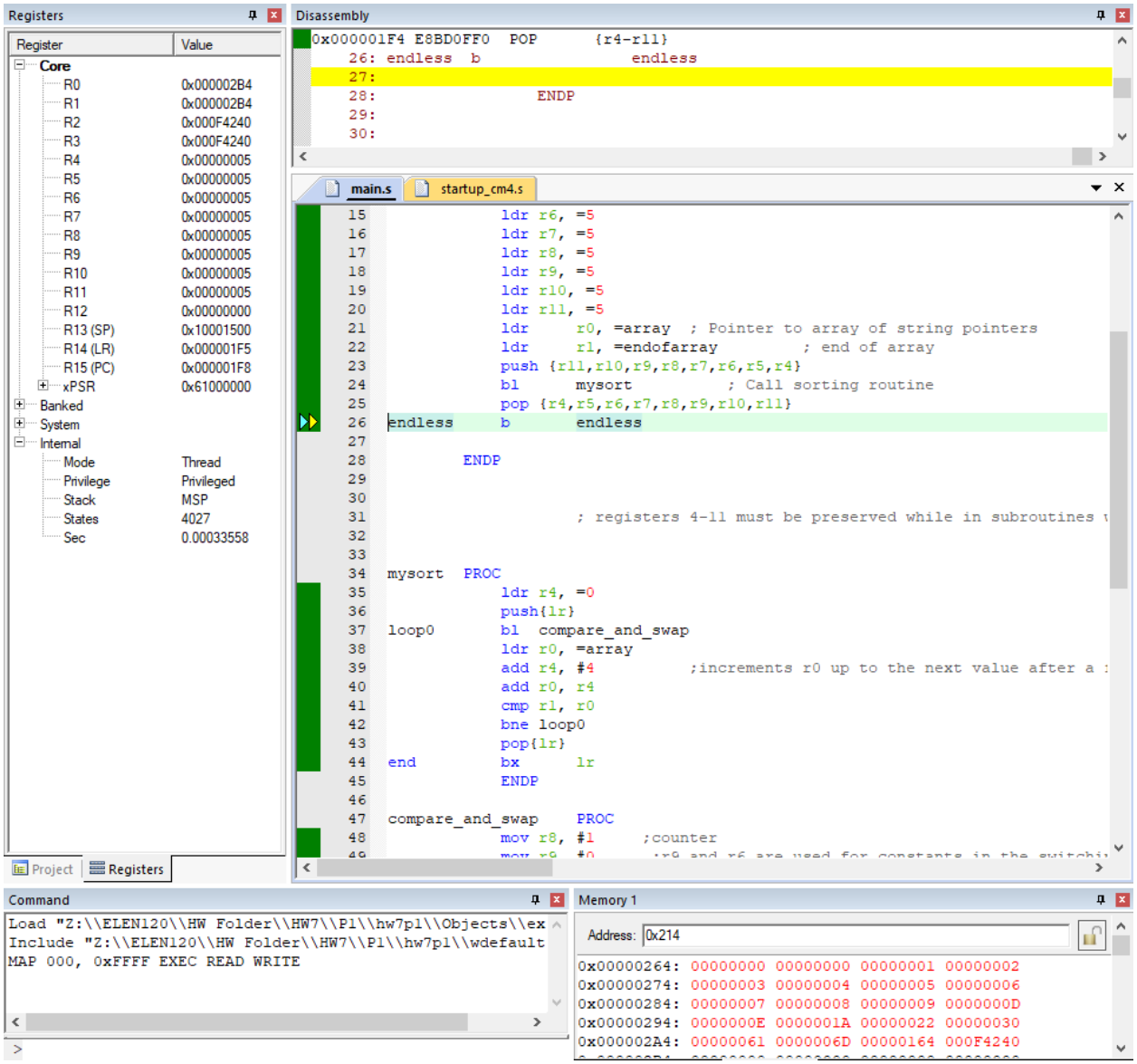
This test shows that the code is scalable to an array of numbers with any n number of elements and not just 10. In this example I use 20 numbers which are:

9,2,5,1,8,6,7,0,3,4,97,34,109,48,356,14,26,13,1000000,0 in that order

Before:



After:

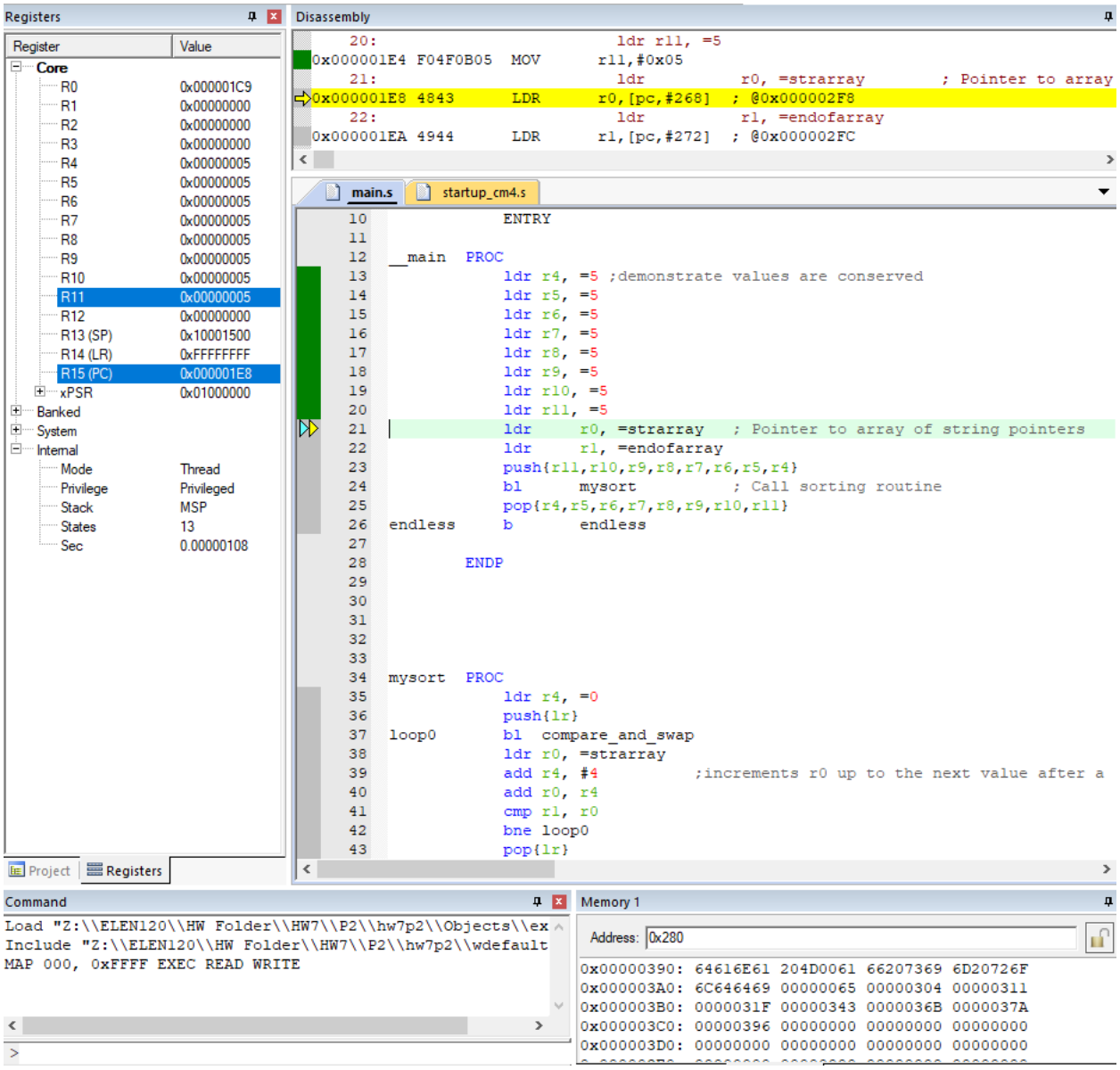


P2

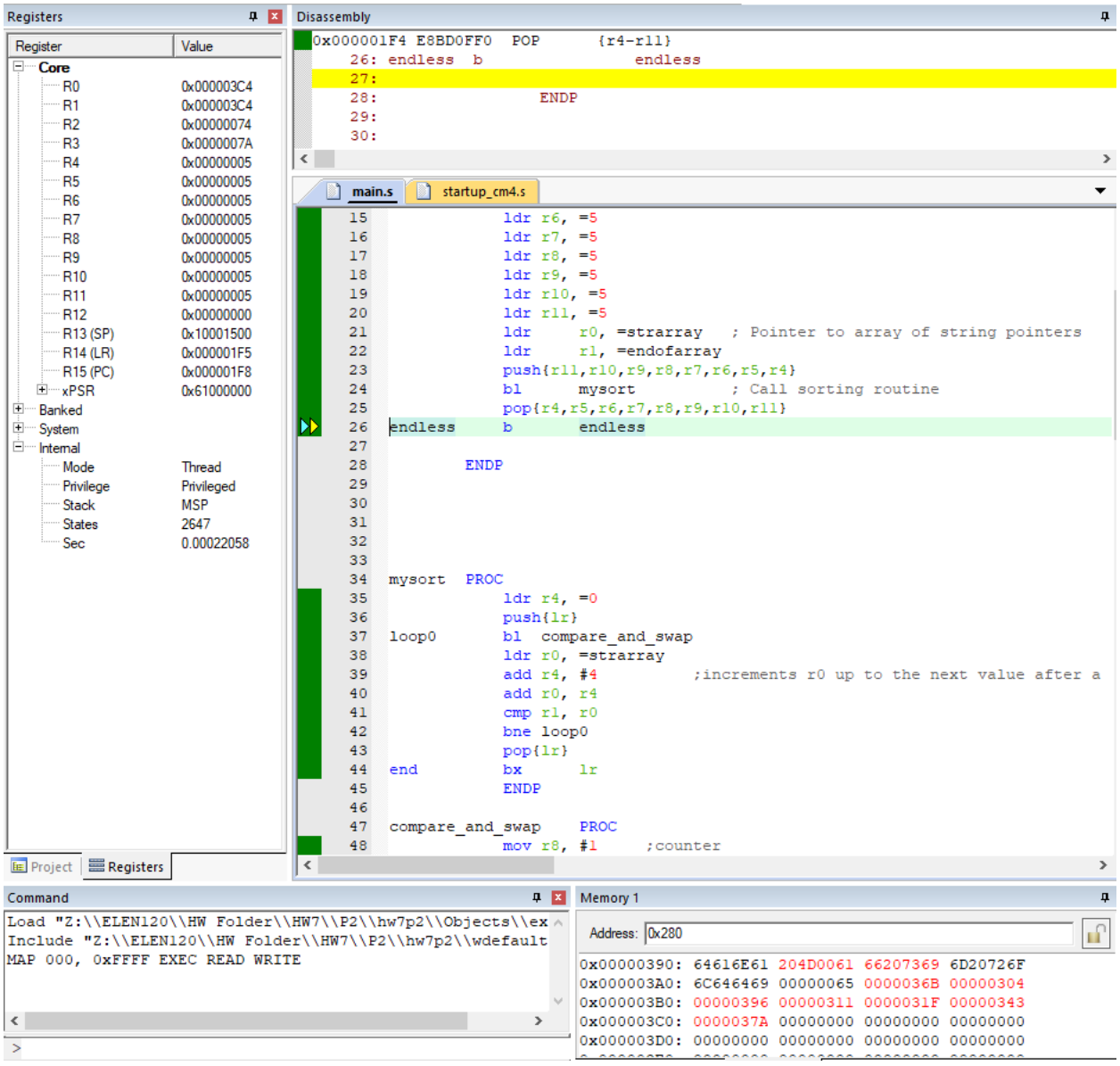
TEST 1

Words	Start Addresses (in hex)	Alphabetical order	What addresses should be (in hex)	What they are (in hex)
"First string",0	304	"A is for apple",0	36b	36b
"Second string",0	311	"First string",0	304	304
"So, do I really need a third string",0	31f	"M is for middle",0	396	396
"Tetraphobia is the fear of the number 4",0	343	"Second string",0	311	311
"A is for apple",0	36b	"So, do I really need a third string",0	31f	31f
"Z is called \'zed\' in Canada",0	37a	"Tetraphobia is the fear of the number 4",0	343	343
"M is for middle",0	396	"Z is called \'zed\' in Canada",0	37a	37a

Before:


Registers
Register
Value
Core
R0 0x000001C9
R1 0x00000000
R2 0x00000000
R3 0x00000000
R4 0x00000005
R5 0x00000005
R6 0x00000005
R7 0x00000005
R8 0x00000005
R9 0x00000005
R10 0x00000005
R11 0x00000005
R12 0x00000000
R13 (SP) 0x10001500
R14 (LR) 0xFFFFFFFF
R15 (PC) 0x000001E8
xPSR 0x01000000
Banked
System
Internal
Mode Thread
Privilege Privileged
Stack MSP
States 13
Sec 0.00000108
Project
Registers
Disassembly
20: ldr r11, =5
0x000001E4 F04F0B05 MOV r11,#0x05
21: ldr r0, =strarray ; Pointer to array
0x000001E8 4843 LDR r0,[pc,#268] ; @0x000002F8
22: ldr r1, =endofarray
0x000001EA 4944 LDR r1,[pc,#272] ; @0x000002FC
main.s
startup_cm4.s
10 ENTRY
12 __main PROC
13 ldr r4, =5 ;demonstrate values are conserved
14 ldr r5, =5
15 ldr r6, =5
16 ldr r7, =5
17 ldr r8, =5
18 ldr r9, =5
19 ldr r10, =5
20 ldr r11, =5
21 ldr r0, =strarray ; Pointer to array of string pointers
22 ldr r1, =endofarray
23 push{r11,r10,r9,r8,r7,r6,r5,r4}
24 bl mysort ; Call sorting routine
25 pop{r4,r5,r6,r7,r8,r9,r10,r11}
26 endless b endless
28 ENDP
34 mysort PROC
35 ldr r4, =0
36 push{lr}
37 loop0 bl compare_and_swap
38 ldr r0, =strarray
39 add r4, #4 ;increments r0 up to the next value after a
40 add r0, r4
41 cmp r1, r0
42 bne loop0
43 pop{lr}
Command
Load "Z:\\ELEN120\\HW Folder\\HW7\\P2\\hw7p2\\Objects\\ex
Include "Z:\\ELEN120\\HW Folder\\HW7\\P2\\hw7p2\\wdefault
MAP 000, 0xFFFF EXEC READ WRITE
Memory 1
Address: 0x280
0x00000390: 64616E61 204D0061 66207369 6D20726F
0x000003A0: 6C646469 00000065 00000304 00000311
0x000003B0: 0000031F 00000343 0000036B 0000037A
0x000003C0: 00000396 00000000 00000000 00000000
0x000003D0: 00000000 00000000 00000000 00000000

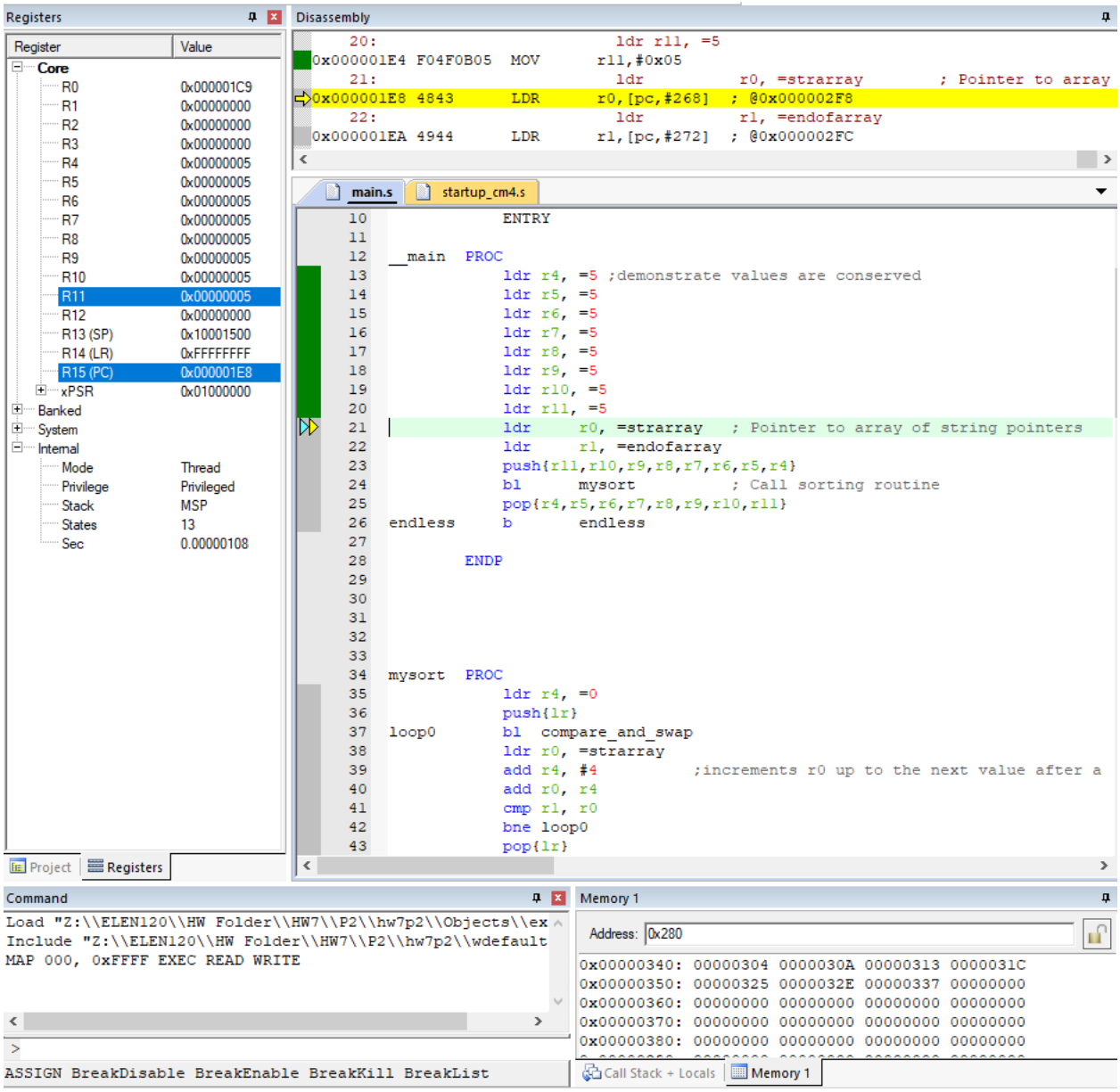
After:


Registers
Register
Value
Core
R0 0x000003C4
R1 0x000003C4
R2 0x00000074
R3 0x0000007A
R4 0x00000005
R5 0x00000005
R6 0x00000005
R7 0x00000005
R8 0x00000005
R9 0x00000005
R10 0x00000005
R11 0x00000005
R12 0x00000000
R13 (SP) 0x10001500
R14 (LR) 0x000001F5
R15 (PC) 0x000001F8
xPSR 0x61000000
Banked
System
Internal
Mode Thread
Privilege Privileged
Stack MSP
States 2647
Sec 0.00022058
Project
Registers
Disassembly
0x000001F4 E8BD0FF0 POP {r4-r11}
26: endless b endless
27:
28: ENDP
29:
30:
main.s
startup_cm4.s
15 ldr r6, =5
16 ldr r7, =5
17 ldr r8, =5
18 ldr r9, =5
19 ldr r10, =5
20 ldr r11, =5
21 ldr r0, =strarray ; Pointer to array of string pointers
22 ldr r1, =endofarray
23 push{r11,r10,r9,r8,r7,r6,r5,r4}
24 bl mysort ; Call sorting routine
25 pop{r4,r5,r6,r7,r8,r9,r10,r11}
26 endless b endless
27
28 ENDP
29
30
31
32
33
34 mysort PROC
35 ldr r4, =0
36 push{lr}
37 loop0 bl compare_and_swap
38 ldr r0, =strarray
39 add r4, #4 ;increments r0 up to the next value after a
40 add r0, r4
41 cmp r1, r0
42 bne loop0
43 pop{lr}
44 end bx lr
45 ENDP
46
47 compare_and_swap PROC
48 mov r8, #1 ;counter
Command
Load "Z:\\ELEN120\\HW Folder\\HW7\\P2\\hw7p2\\Objects\\ex
Include "Z:\\ELEN120\\HW Folder\\HW7\\P2\\hw7p2\\wdefault
MAP 000, 0xFFFF EXEC READ WRITE
Memory 1
Address: 0x280
0x00000390: 64616E61 204D0061 66207369 6D20726F
0x000003A0: 6C646469 00000065 0000036B 00000304
0x000003B0: 00000396 00000311 0000031F 00000343
0x000003C0: 0000037A 00000000 00000000 00000000
0x000003D0: 00000000 00000000 00000000 00000000

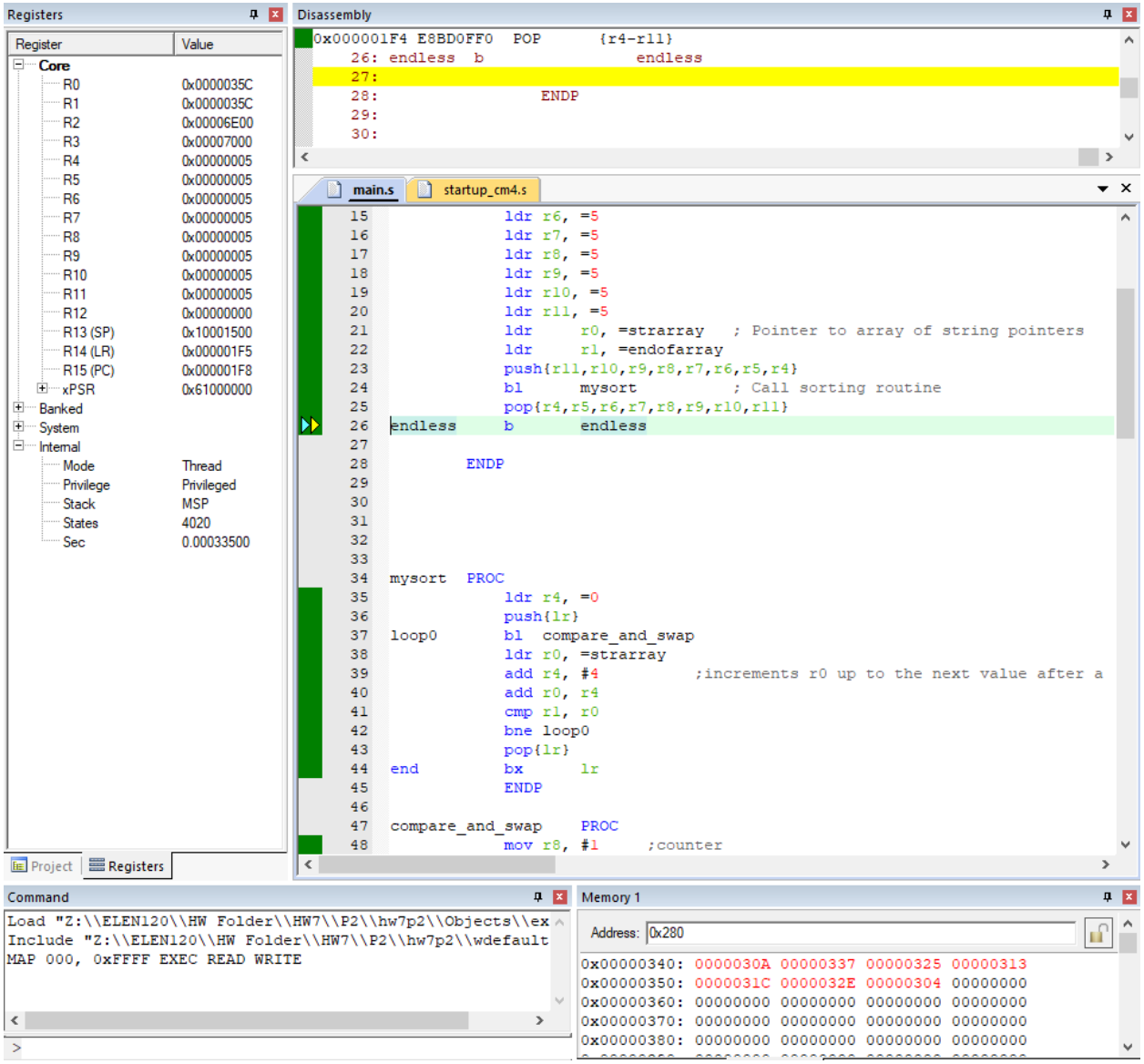
TEST 2

Words	Start Addresses (in hex)	Alphabetical order	What addresses should be (in hex)	What they are (in hex)
"Apple",0	304	"Aardvark",0	30a	30a
"Aardvark",0	30a	"Acronym",0	337	337
"Airplane",0	313	"Air ball",0	325	325
"Airwaves",0	31c	"Airplane",0	313	313
"Air ball",0	325	"Airwaves",0	31c	31c
"Antibody",0	32e	"Antibody",0	32e	32e
"Acronym",0	337	"Apple",0	304	304

Before:


Registers
Register
Value
Core
R0 0x000001C9
R1 0x00000000
R2 0x00000000
R3 0x00000000
R4 0x00000005
R5 0x00000005
R6 0x00000005
R7 0x00000005
R8 0x00000005
R9 0x00000005
R10 0x00000005
R11 0x00000005
R12 0x00000000
R13 (SP) 0x10001500
R14 (LR) 0xFFFFFFFF
R15 (PC) 0x000001E8
xPSR 0x01000000
Banked
System
Internal
Mode Thread
Privilege Privileged
Stack MSP
States 13
Sec 0.00000108
Project
Registers
Disassembly
20: ldr r11, =5
0x000001E4 F04F0B05 MOV r11,#0x05
21: ldr r0, =strarray ; Pointer to array
0x000001E8 4843 LDR r0,[pc,#268] ; @0x000002F8
22: ldr r1, =endofarray
0x000001EA 4944 LDR r1,[pc,#272] ; @0x000002FC
main.s
startup_cm4.s
10 ENTRY
12 __main PROC
13 ldr r4, =5 ;demonstrate values are conserved
14 ldr r5, =5
15 ldr r6, =5
16 ldr r7, =5
17 ldr r8, =5
18 ldr r9, =5
19 ldr r10, =5
20 ldr r11, =5
21 ldr r0, =strarray ; Pointer to array of string pointers
22 ldr r1, =endofarray
23 push{r11,r10,r9,r8,r7,r6,r5,r4}
24 bl mysort ; Call sorting routine
25 pop{r4,r5,r6,r7,r8,r9,r10,r11}
26 endless b endless
28 ENDP
34 mysort PROC
35 ldr r4, =0
36 push{lr}
37 loop0 bl compare_and_swap
38 ldr r0, =strarray
39 add r4, #4 ;increments r0 up to the next value after a
40 add r0, r4
41 cmp r1, r0
42 bne loop0
43 pop{lr}
Command
Load "Z:\\ELEN120\\HW Folder\\HW7\\P2\\hw7p2\\Objects\\ex
Include "Z:\\ELEN120\\HW Folder\\HW7\\P2\\hw7p2\\wdefault
MAP 000, 0xFFFF EXEC READ WRITE
ASSIGN BreakDisable BreakEnable BreakKill BreakList
Memory 1
Address: 0x280
0x00000340: 00000304 0000030A 00000313 0000031C
0x00000350: 00000325 0000032E 00000337 00000000
0x00000360: 00000000 00000000 00000000 00000000
0x00000370: 00000000 00000000 00000000 00000000
0x00000380: 00000000 00000000 00000000 00000000
Call Stack + Locals
Memory 1

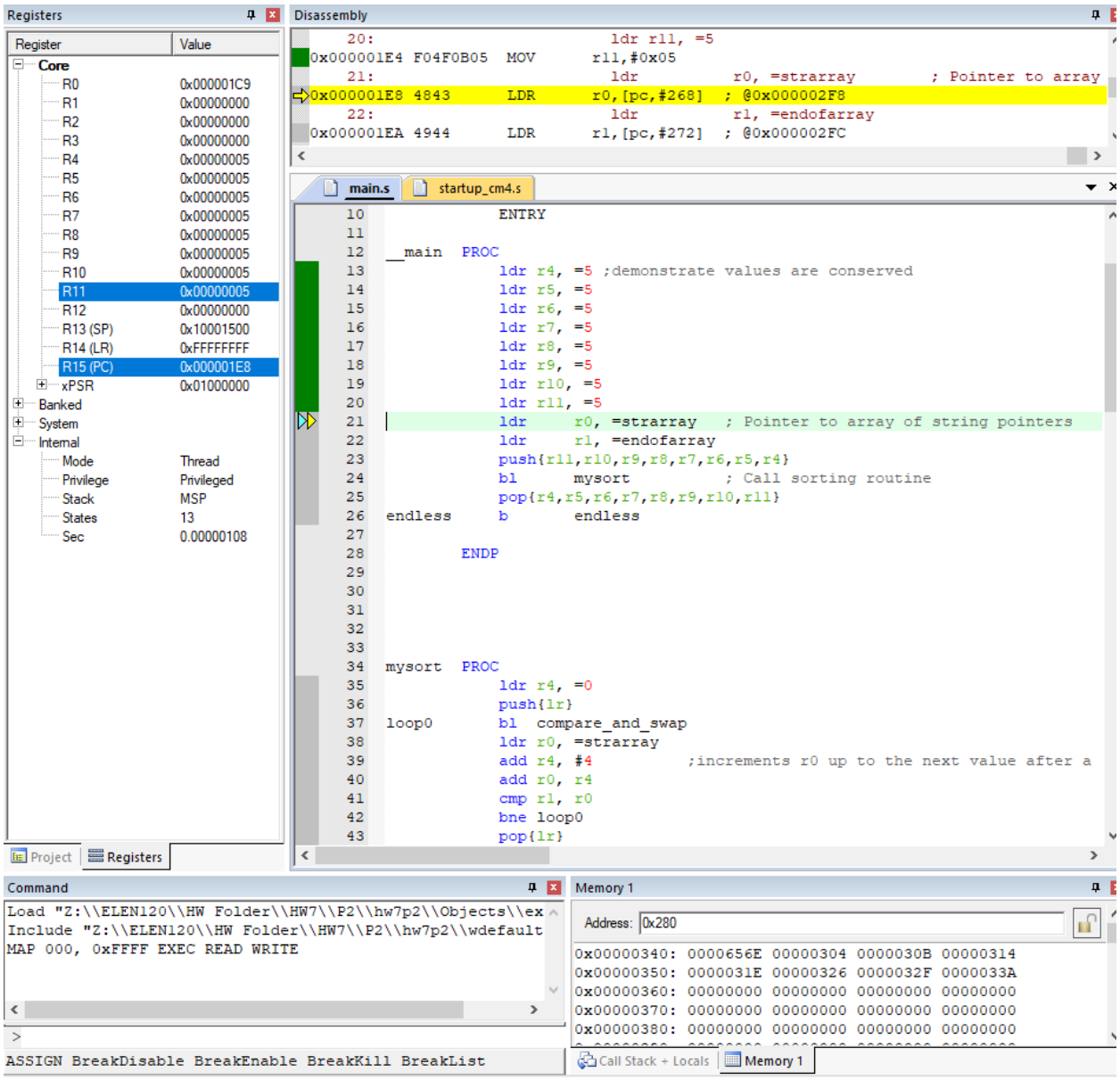
After:


Registers
Register
Value
Core
R0 0x0000035C
R1 0x0000035C
R2 0x00006E00
R3 0x00007000
R4 0x00000005
R5 0x00000005
R6 0x00000005
R7 0x00000005
R8 0x00000005
R9 0x00000005
R10 0x00000005
R11 0x00000005
R12 0x00000000
R13 (SP) 0x10001500
R14 (LR) 0x000001F5
R15 (PC) 0x000001F8
xPSR 0x61000000
Banked
System
Internal
Mode Thread
Privilege Privileged
Stack MSP
States 4020
Sec 0.00033500
Project
Registers
Disassembly
0x000001F4 E8BD0FF0 POP {r4-r11}
26: endless b endless
27:
28: ENDP
29:
30:
main.s
startup_cm4.s
15 ldr r6, =5
16 ldr r7, =5
17 ldr r8, =5
18 ldr r9, =5
19 ldr r10, =5
20 ldr r11, =5
21 ldr r0, =strarray ; Pointer to array of string pointers
22 ldr r1, =endofarray
23 push{r11,r10,r9,r8,r7,r6,r5,r4}
24 bl mysort ; Call sorting routine
25 pop{r4,r5,r6,r7,r8,r9,r10,r11}
26 endless b endless
27
28 ENDP
29
30
31
32
33
34 mysort PROC
35 ldr r4, =0
36 push{lr}
37 loop0 bl compare_and_swap
38 ldr r0, =strarray
39 add r4, #4 ;increments r0 up to the next value after a
40 add r0, r4
41 cmp r1, r0
42 bne loop0
43 pop{lr}
44 end bx lr
45 ENDP
46
47 compare_and_swap PROC
48 mov r8, #1 ;counter
Command
Load "Z:\\ELEN120\\HW Folder\\HW7\\P2\\hw7p2\\Objects\\ex
Include "Z:\\ELEN120\\HW Folder\\HW7\\P2\\hw7p2\\wdefault
MAP 000, 0xFFFF EXEC READ WRITE
Memory 1
Address: 0x280
0x00000340: 0000030A 00000337 00000325 00000313
0x00000350: 0000031C 0000032E 00000304 00000000
0x00000360: 00000000 00000000 00000000 00000000
0x00000370: 00000000 00000000 00000000 00000000
0x00000380: 00000000 00000000 00000000 00000000

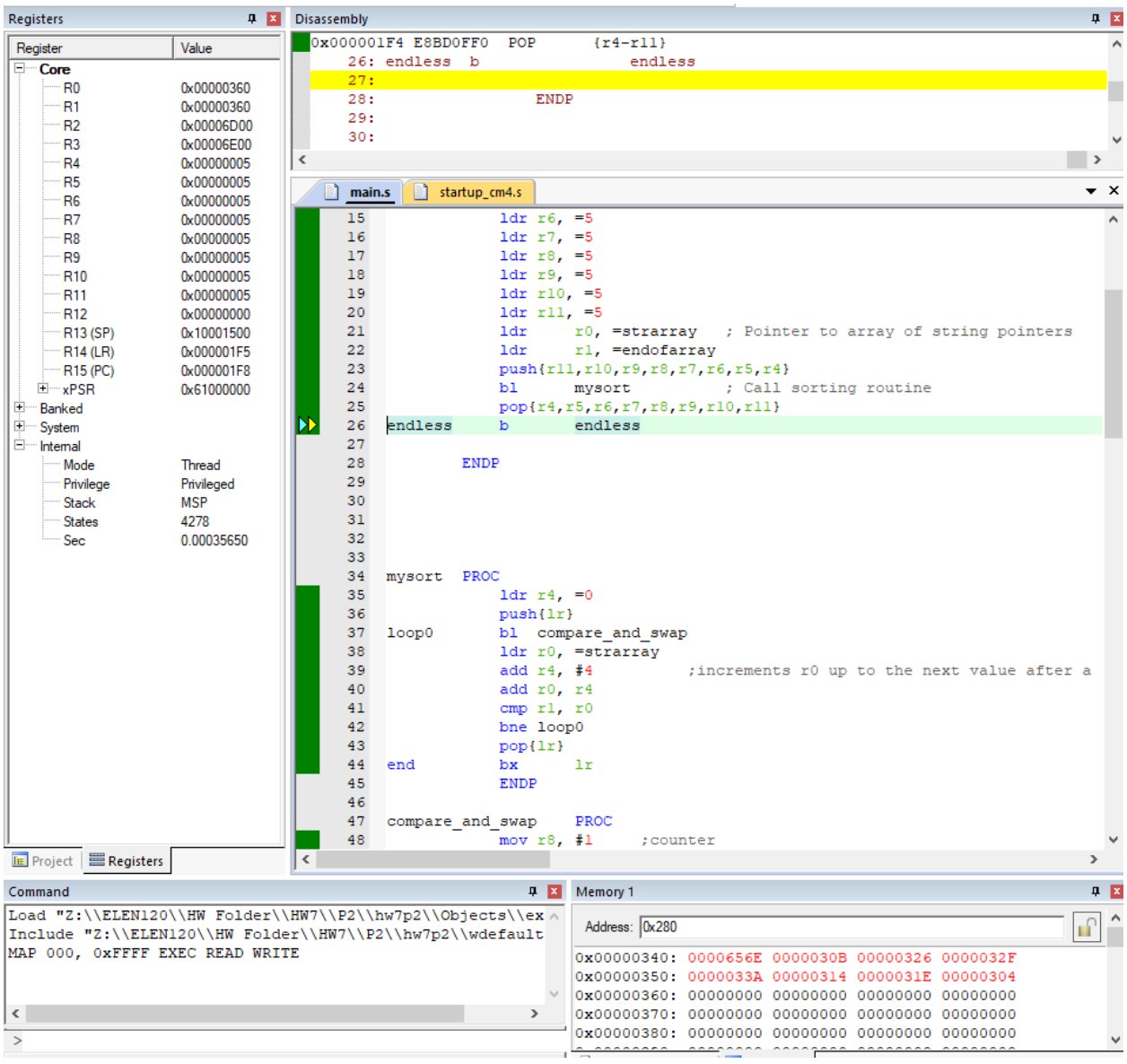
TEST 3

Words	Start Addresses (in hex)	Alphabetical order	What addresses should be (in hex)	What they are (in hex)
"Andrew",0	304	"aardvark",0	30b	30b
"aardvark",0	30b	"air ball",0	326	326
"airplanes",0	314	"Air Canada",0	32f	32f
"America",0	31e	"airplane",0	33a	33a
"air ball",0	326	"airplanes",0	314	314
"Air Canada",0	32f	"America",0	31e	31e
"airplane",0	33a	"Andrew",0	304	304

Before:


Registers
Register
Value
Core
R0 0x000001C9
R1 0x00000000
R2 0x00000000
R3 0x00000000
R4 0x00000005
R5 0x00000005
R6 0x00000005
R7 0x00000005
R8 0x00000005
R9 0x00000005
R10 0x00000005
R11 0x00000005
R12 0x00000000
R13 (SP) 0x10001500
R14 (LR) 0xFFFFFFFF
R15 (PC) 0x000001E8
xPSR 0x01000000
Banked
System
Internal
Mode Thread
Privilege Privileged
Stack MSP
States 13
Sec 0.00000108
Project
Registers
Disassembly
20: ldr r11, =5
0x000001E4 F04F0B05 MOV r11,#0x05
21: ldr r0, =strarray ; Pointer to array
0x000001E8 4843 LDR r0,[pc,#268] ; @0x000002F8
22: ldr r1, =endofarray
0x000001EA 4944 LDR r1,[pc,#272] ; @0x000002FC
main.s
startup_cm4.s
10 ENTRY
12 __main PROC
13 ldr r4, =5 ;demonstrate values are conserved
14 ldr r5, =5
15 ldr r6, =5
16 ldr r7, =5
17 ldr r8, =5
18 ldr r9, =5
19 ldr r10, =5
20 ldr r11, =5
21 ldr r0, =strarray ; Pointer to array of string pointers
22 ldr r1, =endofarray
23 push{r11,r10,r9,r8,r7,r6,r5,r4}
24 bl mysort ; Call sorting routine
25 pop{r4,r5,r6,r7,r8,r9,r10,r11}
26 endless b endless
28 ENDP
34 mysort PROC
35 ldr r4, =0
36 push{lr}
37 loop0 bl compare_and_swap
38 ldr r0, =strarray
39 add r4, #4 ;increments r0 up to the next value after a
40 add r0, r4
41 cmp r1, r0
42 bne loop0
43 pop{lr}
Command
Load "Z:\\ELEN120\\HW Folder\\HW7\\P2\\hw7p2\\Objects\\ex
Include "Z:\\ELEN120\\HW Folder\\HW7\\P2\\hw7p2\\wdefault
MAP 000, 0xFFFF EXEC READ WRITE
ASSIGN BreakDisable BreakEnable BreakKill BreakList
Memory 1
Address: 0x280
0x00000340: 0000656E 00000304 0000030B 00000314
0x00000350: 0000031E 00000326 0000032F 0000033A
0x00000360: 00000000 00000000 00000000 00000000
0x00000370: 00000000 00000000 00000000 00000000
0x00000380: 00000000 00000000 00000000 00000000
Call Stack + Locals
Memory 1

After:


Registers
Register
Value
Core
R0 0x00000360
R1 0x00000360
R2 0x00006D00
R3 0x00006E00
R4 0x00000005
R5 0x00000005
R6 0x00000005
R7 0x00000005
R8 0x00000005
R9 0x00000005
R10 0x00000005
R11 0x00000005
R12 0x00000000
R13 (SP) 0x10001500
R14 (LR) 0x000001F5
R15 (PC) 0x000001F8
xPSR 0x61000000
Banked
System
Internal
Mode Thread
Privilege Privileged
Stack MSP
States 4278
Sec 0.00035650
Project
Registers
Disassembly
0x000001F4 E8BD0FF0 POP {r4-r11}
26: endless b endless
27:
28: ENDP
29:
30:
main.s
startup_cm4.s
15 ldr r6, =5
16 ldr r7, =5
17 ldr r8, =5
18 ldr r9, =5
19 ldr r10, =5
20 ldr r11, =5
21 ldr r0, =strarray ; Pointer to array of string pointers
22 ldr r1, =endofarray
23 push{r11,r10,r9,r8,r7,r6,r5,r4}
24 bl mysort ; Call sorting routine
25 pop{r4,r5,r6,r7,r8,r9,r10,r11}
26 endless b endless
27
28 ENDP
29
30
31
32
33
34 mysort PROC
35 ldr r4, =0
36 push{lr}
37 loop0 bl compare_and_swap
38 ldr r0, =strarray
39 add r4, #4 ;increments r0 up to the next value after a
40 add r0, r4
41 cmp r1, r0
42 bne loop0
43 pop{lr}
44 end bx lr
45 ENDP
46
47 compare_and_swap PROC
48 mov r8, #1 ;counter
Command
Load "Z:\\ELEN120\\HW Folder\\HW7\\P2\\hw7p2\\Objects\\ex
Include "Z:\\ELEN120\\HW Folder\\HW7\\P2\\hw7p2\\wdefault
MAP 000, 0xFFFF EXEC READ WRITE
Memory 1
Address: 0x280
0x00000340: 0000656E 0000030B 00000326 0000032F
0x00000350: 0000033A 00000314 0000031E 00000304
0x00000360: 00000000 00000000 00000000 00000000
0x00000370: 00000000 00000000 00000000 00000000
0x00000380: 00000000 00000000 00000000 00000000

TEST 4 (extra test)

!THIS IS NOT A PART OF THE STANDARD RUBRIC!
Shows that code works with all sorts of different cases from all capital words, special characters, and more than 7 words. Example used here is:

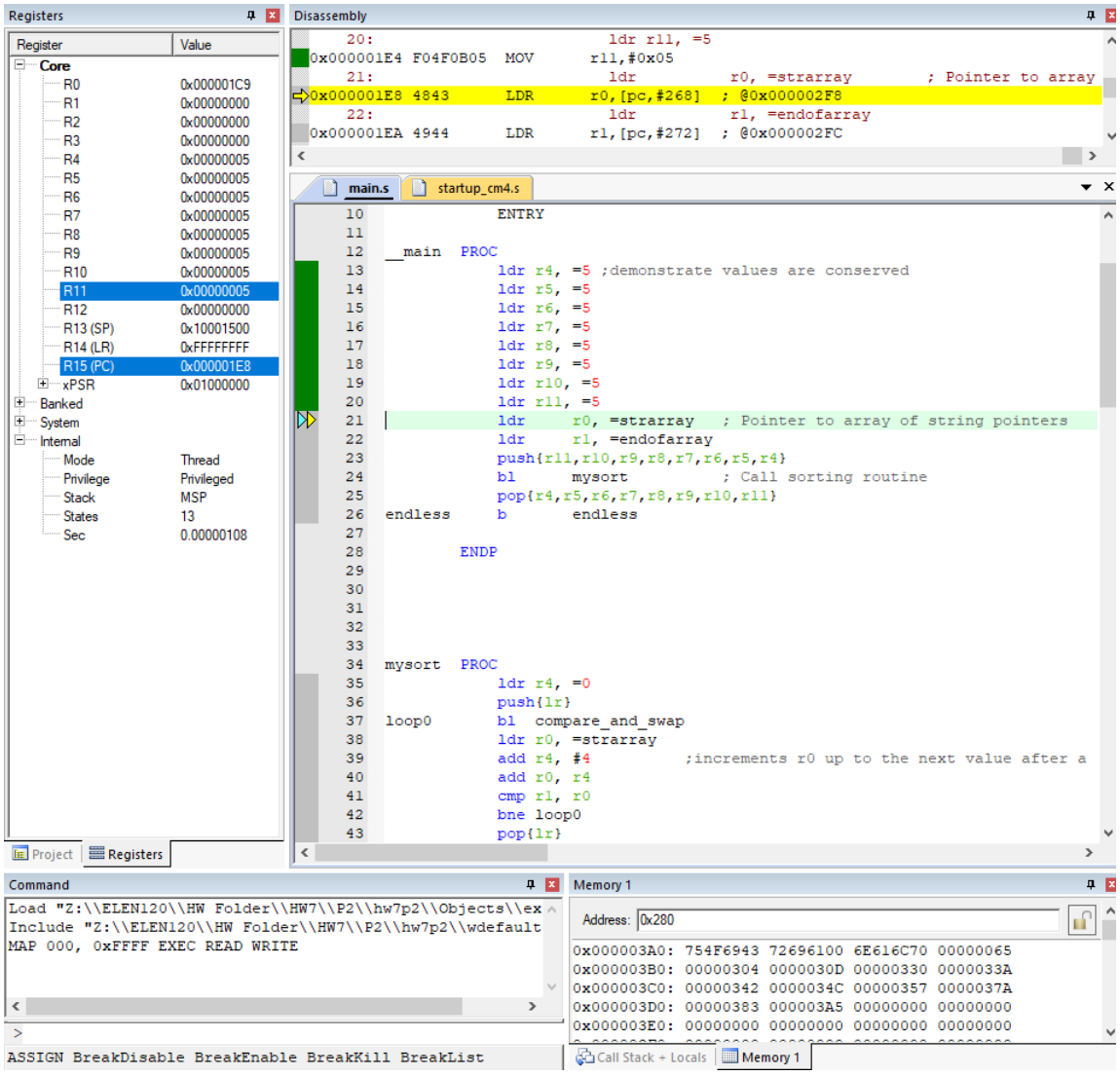
```
str1 DCB "$Andrew$",0
str2 DCB "supercalifragilisticexpialidocious",0
str3 DCB "airplanes",0
str4 DCB "America",0
str5 DCB "#air ball",0
str6 DCB "Air Canada",0
str7 DCB "SUPERCALIFRAGILISTICEXPIALIDOCIOUZ",0
str8 DCB "aardvark",0
str9 DCB "SuPeRcAlIfRaGiLIsTiCeXpIaLiDoCiOu",0
str10 DCB "airplane",0
strarray        DCD    str1, str2, str3, str4, str5, str6, str7, str8, str9, str10
```

(note the hashtag character is one less than the dollar sign)

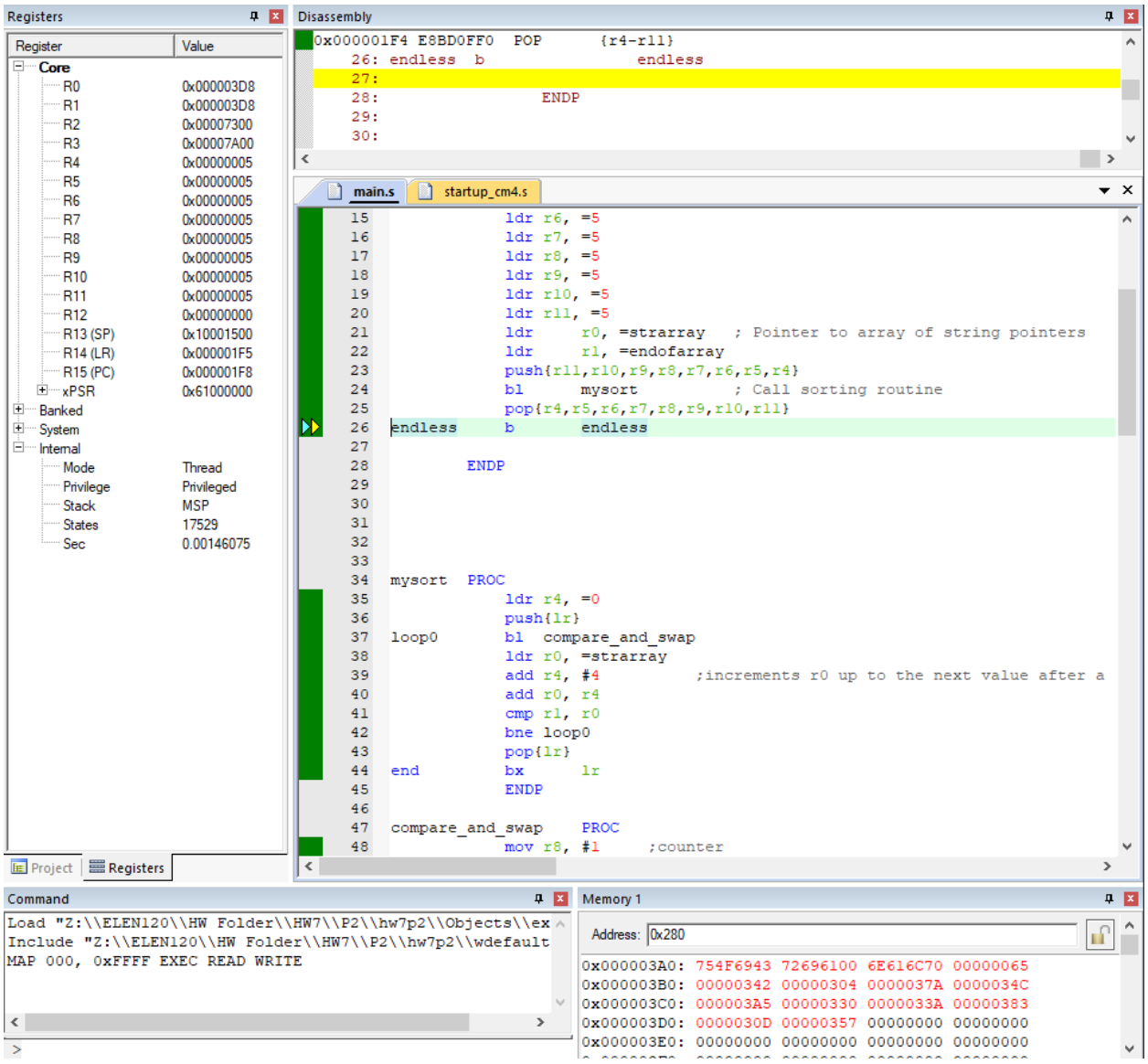
Words	Start Addresses (in hex)	Alphabetical order	What addresses should be (in hex)	What they are (in hex)
"$Andrew$",0	304	"#air ball",0	342	342
"supercalifragilisticexpialidocious",0	30d	"$Andrew$",0	304	304
"airplanes",0	330	"aardvark",0	37a	37a
"America",0	33a	"Air Canada",0	34c	34c
"#air ball",0	342	"airplane",0	3a5	3a5
"Air Canada",0	34c	"airplanes",0	330	330
"SUPERCALIFRAGILISTICEXPIALIDOCIOUZ",0	357	"America",0	33a	33a
"aardvark",0	37a	"SuPeRcAlIfRaGiLIsTiCeXpIaLiDoCiOu",0	383	383

"SuPeRcAlIfRaGiLIsTiCeXpIaLiDoCiOu",0	383	"supercalifragilisticexpialidocious",0	30d	30d
"airplane",0	3a5	"SUPERCALIFRAGILISTICEXPIALIDOCIOUZ",0	357	357

Before:



After:


Registers
Register
Value
Core
R0 0x000003D8
R1 0x000003D8
R2 0x00007300
R3 0x00007A00
R4 0x00000005
R5 0x00000005
R6 0x00000005
R7 0x00000005
R8 0x00000005
R9 0x00000005
R10 0x00000005
R11 0x00000005
R12 0x00000000
R13 (SP) 0x10001500
R14 (LR) 0x000001F5
R15 (PC) 0x000001F8
xPSR 0x61000000
Banked
System
Internal
Mode Thread
Privilege Privileged
Stack MSP
States 17529
Sec 0.00146075
Project
Registers
Disassembly
0x000001F4 E8BD0FF0 POP {r4-r11}
26: endless b endless
27:
28: ENDP
29:
30:
main.s
startup_cm4.s
15 ldr r6, =5
16 ldr r7, =5
17 ldr r8, =5
18 ldr r9, =5
19 ldr r10, =5
20 ldr r11, =5
21 ldr r0, =strarray ; Pointer to array of string pointers
22 ldr r1, =endofarray
23 push{r11,r10,r9,r8,r7,r6,r5,r4}
24 bl mysort ; Call sorting routine
25 pop{r4,r5,r6,r7,r8,r9,r10,r11}
26 endless b endless
27
28 ENDP
29
30
31
32
33
34 mysort PROC
35 ldr r4, =0
36 push{lr}
37 loop0 bl compare_and_swap
38 ldr r0, =strarray
39 add r4, #4 ;increments r0 up to the next value after a
40 add r0, r4
41 cmp r1, r0
42 bne loop0
43 pop{lr}
44 end bx lr
45 ENDP
46
47 compare_and_swap PROC
48 mov r8, #1 ;counter
Command
Load "Z:\\ELEN120\\HW Folder\\HW7\\P2\\hw7p2\\Objects\\ex
Include "Z:\\ELEN120\\HW Folder\\HW7\\P2\\hw7p2\\wdefault
MAP 000, 0xFFFF EXEC READ WRITE
Memory 1
Address: 0x280
0x000003A0: 754F6943 72696100 6E616C70 00000065
0x000003B0: 00000342 00000304 0000037A 0000034C
0x000003C0: 000003A5 00000330 0000033A 00000383
0x000003D0: 0000030D 00000357 00000000 00000000
0x000003E0: 00000000 00000000 00000000 00000000

Note regarding my sorting method:
My selection sort does not work exactly as normal selection sort works. Instead of searching and finding the smallest value and then swapping it with the first position and incrementing and so on, it swaps anytime it finds a value smaller than the current lowest position and then increments and so on. As a result the code is technically slower (still O(n^2)) than normal selection sort but it uses less memory as you don't ever have to store the lowest value and its address. This is still by definition selection sort because every number is placed into its correct position by being swapped there. The only difference is that the unsorted section of the array is moved around a bit. I only did this because I remembered selection sort incorrectly and thought you were SUPPOSED to switch everytime you found a value lower. If I had time, I would change it, but I do not and as said previously it is still by definition selection sort; just a little more inefficient. Wolfe also never stated efficiency was something to be considered (even though it's standard practice to consider it). All in all, the time complexity is still O(n^2) based on how many loops it does and it sorts the numbers/words by swapping the lowest number/word of the unsorted array into whatever position into the sorted part of the array it belongs (same as selection sort).

Visualization of what I'm saying:

List and steps	Normal way	The way I did it in this HW
(11,25,12,22,63)	(11,25,12,22,13)	(11,25,12,22,13)
Step 1; checks position1	(11,25,12,22,13)	(11,25,12,22,13)
Step 2; checks position2	(11,12,25,22,13)	(11,12,25,22,13)
Step 3; checks position3 (the difference between the two methods happens here)	(11,12,13,22,25)	(11,12,13,25,22)
Step 4; checks position4	(11,12,13,22,25)	(11,12,13,22,25)