

Austin Petersen, Ayden Dauenhauer

Prof. Wolfe

ELEN 120L Tuesday 2:15-5:00

October 24, 2023

## Report 4

### Problem 1:

The screenshot shows a debugger interface with several windows:

- Registers:** Shows CPU registers (R0-R15, PSR) and their values.
- Assembly:** Shows the assembly code for the main function and the startup\_cm4.s file.
- Memory:** A dump of memory starting at address 0x100014E4.
- Command:** The command window shows the project path and build status.

**Registers Window:**

Register	Value
R0	0x000001F6
R1	0x00000020
R2	0x00000000
R3	0x00000045
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x000001E1
R15 (PC)	0x000001E0
PSR	0x61000000

**Assembly Window:**

```
main.s: AREA main, CODE, READONLY
EXPORT __main
ENTRY

_main PROC
    LDR r1, =list ;Address of list in r1
    MOV r2, #7 ;Number of elements in list, n
    LDR r3, [r1]
    BL store
    MOV r2, #7
    MOV r0, #0
    BL sum

endless B endless
store
    PUSH {r3}
    ADD r1, #4 ;Goes next
    LDR r3, [r1]
    SUB r2, #1 ;Decrement
    CMP r2, #0 ;Compares
    BNE store
    BX lr

sum
    POP {r3}
    ADD r0, r3 ;Add current to r0 for total sum
    SUB r2, #1 ;Decrement
    CMP r2, #0 ;Compares
    BNE sum
    BX lr

ENDP
ALIGN

list DCD 69,420,6,9,-4,2,0 ;Array of n 32 bit ints
END
```

**Memory Window:**

Address	Value
0x100014E4	0000000000 0000000002 4294967292 0000000009 0000000006 000000420 0000000069
0x10001500	0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000
0x10001520	0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000
0x10001530	0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000
0x10001538	0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000
0x10001544	0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000
0x10001554	0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000
0x10001570	0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000
0x1000158C	0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000
0x100015A8	0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000
0x100015C4	0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000
0x100015E0	0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000
0x100015FC	0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000
0x10001618	0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000

## Problem 2:

**Registers**

Register	Value
R0	0x0000000D
R1	0x00000000
R2	0x0000000D
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x10001500
R14 (LR)	0x000001D1
R15 (PC)	0x000001D0
xPSR	0x61000000

**Disassembly**

```

6:          MOV r0, #7 ; p = 3
0x000001C8 F04F0007  MOV      r0,#0x07
7:          BL fib
0x000001CC F000F801  BL.W    0x000001D2
8: endless  B endless
9:
10:
11:         fib     PUSH {lr}   ;push load register to save position
12:         cmp r0, #0   ;check if value is 0
13:         BEQ endif0 ;break if 0
14:         b next    ;continue if not 0
15:
16:         endif0  mov r0, #0   ;return 0
17:         b out     ;leave
18:
19:         next    cmp r0, #1   ;check if value is 1
20:         BEQ endif1 ;break if 1
21:         b cont    ;continue if not 1
22:
23:         endif1  add r2, #1   ;return 1
24:         b out     ;leave
25:
26:         cont    push {r0}   ;stores our p value
27:         sub r0, #1   ;gets p-1
28:         BL fib    ;fib p-1
29:         pop {r0}   ;gets p again
30:         sub r0, #2   ;gets p-2
31:         BL fib    ;fib p-2
32:
33:
34:         out     pop{lr}    ;gets return possition
35:         mov r0, r2
36:         bx lr     ;returns
37:
38:
39:         ENDP
40:         ALIGN
41:
42: END

```

The screenshot shows the Registers and Disassembly windows of a debugger. The Registers window displays the ARM core registers (R0-R15, xPSR) with their current values. The Disassembly window shows the assembly code for the main function, which calculates the nth Fibonacci number. The code uses a recursive approach with base cases for 0 and 1, and a loop for n > 1. It also handles invalid input (negative n). The assembly code is annotated with comments explaining each step.

### Problem 3:

```
main.s
7         ENTRY
8
9     __main PROC
10    bl porta
11    ldr r4, =list
12    ldr r5, =coll
13    loop   bl read_jystick
14    bl decode_jystick
15    cmp r0, #0
16    movne r6, r0
17    b loop
18    endless b endless
19
20
21 ;enables the clock in GPIO A
22    porta ldr r0, =(RCC_BASE+RCC_AHB2ENR)
23    ldr r1, [r0]
24    ORR r1, #RCC_AHB2ENR_GPIOAEN
25    STR r1, [r0]
26
27    ldr r0, =(GPIOA_BASE+GPIO_MODER)
28    ldr r1, [r0]
29    mov r2, #0xFF
30    bic r1, r1, r2
31    str r1, [r0]
32
33    ldr r0, =(GPIOA_BASE+GPIO_PUPDR)
34    ldr r1, [r0]
35    mov r2, #0xCFC
36    bic r1, r1, r2
37    mov r3, #2_100010101010
38    orr r1, r3
39    str r1, [r0]
40    bx lr
41
42    read_jystick ldr r0, =(GPIOA_BASE+GPIO_IDR) ;loads IDR from gpio a in r0
43    ldr r1, [r0] ;loads the value
44    mov r2, #2_101111
45    and r0, r1, r2 ; clears all but bits 0, 1, 2, 3, 5
46    bx lr ;returns
47
48
49    decode_jystick cmp r0, #2_000001
50    beq centered
51    cmp r0, #2_000010
52    beq left
53    cmp r0, #2_000100
54    beq right
55    cmp r0, #2_001000
56    beq up
57    cmp r0, #2_100000
58    beq down
59    bx lr
60
61    center   mov r0, #'c'
62    b store
63
64    right    mov r0, #'r'
65    b store
66
67    left     mov r0, #'l'
68    b store
69
70    up      mov r0, #'u'
71    b store
72
73    down    mov r0, #'d'
74    b store
75
76    out     bx lr
77
78    store   cmp r0, r6
79    beq out ; moveq pc. lr
```

```

store      cmp r0, r6
           beq out ; moveq pc. lr
           str r0, [r4]
           add r4, #1
           cmp r4, r5
           beq endless
           bx lr

ENDP
ALIGN
AREA     myData, DATA, READWRITE
ALIGN

list      dcb 0,0,0,0,0
eoll

END

```

The screenshot shows a debugger interface with two main windows. The top window displays assembly code for the file `main.s`. The bottom window shows a memory dump titled "Memory 1".

**Assembly Code (main.s):**

```

25      STR r1, [r0]
26
27      ldr r0, =(GPIOA_BASE+GPIO_MODER)
28      ldr r1, [r0]
29      mov r2, #0xFF
30      bic r1, r1, r2
31      str r1, [r0]
32
33      ldr r0, =(GPIOA_BASE+GPIO_PUPDR)
34      ldr r1, [r0]
35      mov r2, #0x0FC
36      bic r1, r1, r2
37      mov r3, #2_100010101010
38      orr r1, r3
39      str r1, [r0]
40      bx lr
41
42      read_jystick ldr r0, =(GPIOA_BASE+GPIO_IDR) ;loads IDR from gpio a in r0
43          ldr r1, [r0] ;loads the value
44          mov r2, #2_10111
45          and r0, r1, r2 ; clears all but bits 0, 1, 2, 3, 5
46          bx lr ;returns
47
48      decode_jystick cmp r0, #2_000001
49          beg center
50          cmp r0, #2_000010
51          beg left
52          cmp r0, #2_000100
53          beg right
54          cmp r0, #2_001000
55          beg up
56          cmp r0, #2_100000
57          beg down
58          bx lr
59
60      center    mov r0, #'c'
61      center    b store
62
63      right     mov r0, #'r'
64      right     b store
65
66      left      mov r0, #'l'
67      left      b store
68
69      up        mov r0, #'u'
70      up        b store
71
72      down      mov r0, #'d'
73      down      b store

```

**Memory Dump (Memory 1):**

Address	Value
0x20000000	d1ruc.....
0x20000090	.....