

Austin Petersen, Ayden Dauenhauer

Prof. Wolfe

ELEN 120L Tuesday 2:15-5:00

October 24, 2023

## Report 4

### Problem 1:

The screenshot displays the Keil uVision IDE with an ARM assembly project. The main window shows the following assembly code:

```
1 AREA main, CODE, READONLY
2 EXPORT __main
3 ENTRY __main
4
5 __main PROC
6
7     LDR r1, =list ;Address of list in r1
8     MOV r2, #7 ;Number of elements in list, n
9     LDR r3, [r1]
10    BL store
11    MOV r2, #7
12    MOV r0, #0
13    BL sum
14
15    endless B endless
16
17 store    PUSH {r3}
18         ADD r1, #4 ;Goes next
19         LDR r3, [r1]
20         SUB r2, #1 ;Decrements
21         CMP r2, #0 ;Compares
22         BNE store
23         BX lr
24
25 sum      POP {r3}
26         ADD r0, r3 ;Add current to r0 for total sum
27         SUB r2, #1 ;Decrements
28         CMP r2, #0 ;Compares
29         BNE sum
30         BX lr
31
32 ENDP
33 ALIGN
34
35 list    DCD 69,420,6,9,-4,2,0 ;Array of n 32 bit ints
36
37 END
```

The bottom window shows the command line with the following text:

```
Load "Z:\\ELEN_120L\\Lab4\\Lab4.1\\Objects\\Lab4.1"
Include "Z:\\ELEN_120L\\Lab4\\Lab4.1\\..\\..\\vdefault.ini"
MAP 000, 0xFFFF EXEC READ WRITE
```

The Memory window shows the address 0x100014E4 and a list of memory values, including 0x100014E4: 0000000000 0000000000 4294967292 0000000009 0000000006 0000000420 0000000069.

## Problem 2:

Registers

Register	Value
Core	
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
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	1010
Sec	0.00008417

Disassembly

main.s

startup\_cm4.s

```

1      AREA main, CODE, READONLY
2      EXPORT __main
3      ENTRY
4
5      __main PROC
6      MOV r0, #7 ; p = 3
7      BL fib
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
35     out    pop{lr}       ;gets return position
36           mov r0, r2
37           bx lr         ;returns
38
39           ENDP
40           ALIGN
41
42     END

```

### Problem 3:

```
main.s startup_stm32476xx.s
7      ENTRY
8
9      _main PROC
10         bl porta
11         ldr r4, =list
12         ldr r5, =eoll
13     loop    bl read_jstick
14           bl decode_jstick
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, #0xCFF
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_jstick ldr r0, =(GPIOA_BASE+GPIO_IDR) ;loads IDR from gpio a in r0
43               ldr r1, [r0] ;loads the value
44               mov r2, #2_1011111
45               and r0, r1, r2 ; clears all but bits 0, 1, 2, 3, 5
46               bx lr ;returns
47
48
49     decode_jstick cmp r0, #2_000001
50                   beq center
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 code editor with two tabs: 'main.s\*' and 'startup\_stm32476xx.s'. The code is written in assembly for an STM32 microcontroller. It includes comments in English to explain the functionality of the code.

```

25      STR r1, [r0]
26
27      ldr r0, =(GPIOA_BASE+GPIO_MODER)
28      ldr r1, [r0]
29      mov r2, #0xCFF
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_jstick 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_jstick cmp r0, #2_000001
50               beq center
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'

```

At the bottom of the editor, there is a 'Memory 1' window showing the address 0x20000000. The memory content is displayed as follows:

Address	Value
0x20000000	dlruc.....
0x20000004	.....