Christian Garcia

Edward Ghazarossian

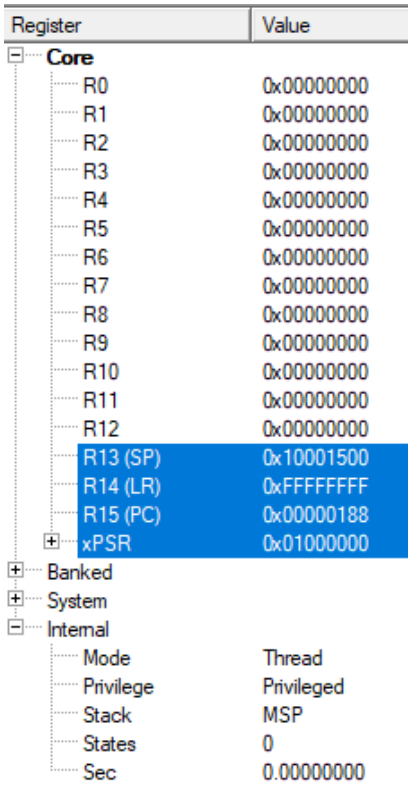
ELEN 120 Lab

26 October 2021

Lab 4: Stacks and Subroutines

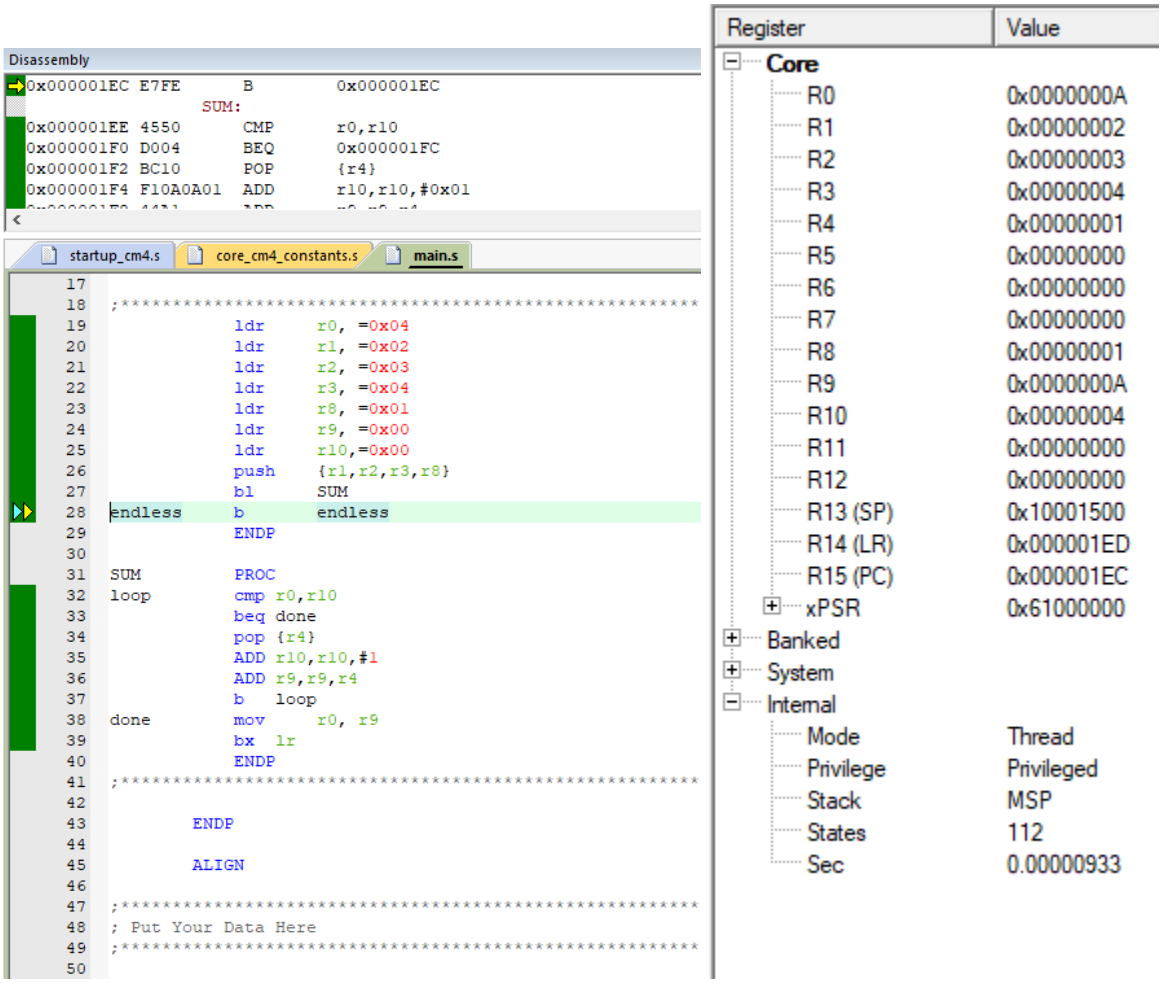
Problem 1:

Before:



Register	Value
Core	
R0	0x00000000
R1	0x00000000
R2	0x00000000
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)	0xFFFFFFFF
R15 (PC)	0x00000188
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	0
Sec	0.00000000

After:

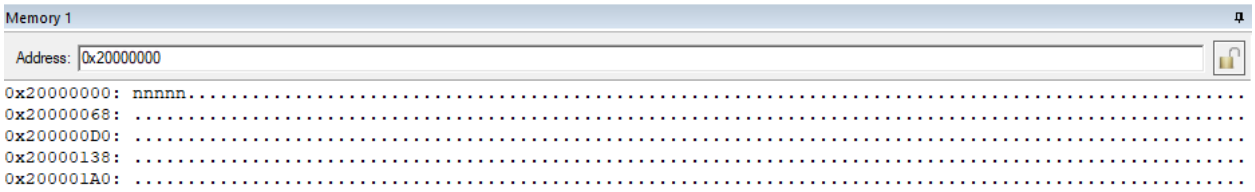

Disassembly
0x000001EC E7FE B 0x000001EC
SUM:
0x000001EE 4550 CMP r0,r10
0x000001F0 D004 BEQ 0x000001FC
0x000001F2 BC10 POP {r4}
0x000001F4 F10A0A01 ADD r10,r10,#0x01
startup_cm4.s
core_cm4_constants.s
main.s
17
18 ;***
19 ldr r0, =0x04
20 ldr r1, =0x02
21 ldr r2, =0x03
22 ldr r3, =0x04
23 ldr r8, =0x01
24 ldr r9, =0x00
25 ldr r10,=0x00
26 push {r1,r2,r3,r8}
27 bl SUM
28 endless b endless
29 ENDP
30
31 SUM PROC
32 loop cmp r0,r10
33 beq done
34 pop {r4}
35 ADD r10,r10,#1
36 ADD r9,r9,r4
37 b loop
38 done mov r0, r9
39 bx lr
40 ENDP
41 ;***
42
43 ENDP
44
45 ALIGN
46
47 ;***
48 ; Put Your Data Here
49 ;***
50
Register
Value
Core
R0 0x0000000A
R1 0x00000002
R2 0x00000003
R3 0x00000004
R4 0x00000001
R5 0x00000000
R6 0x00000000
R7 0x00000000
R8 0x00000001
R9 0x0000000A
R10 0x00000004
R11 0x00000000
R12 0x00000000
R13 (SP) 0x10001500
R14 (LR) 0x000001ED
R15 (PC) 0x000001EC
xPSR 0x61000000
Banked
System
Internal
Mode Thread
Privilege Privileged
Stack MSP
States 112
Sec 0.00000933

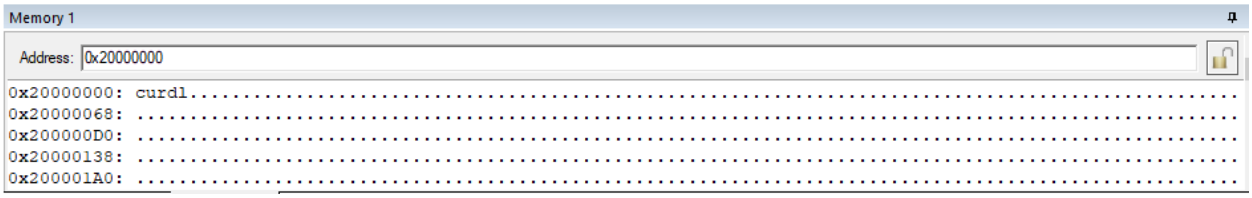
Problem 3:

```
        INCLUDE core_cm4_constants.s
        INCLUDE stm32l476xx_constants.s

                        AREA    main, CODE, READONLY
                        EXPORT      __main
                        ENTRY

__mainPROC
                        bl              CONFIG
                        ldr     r3, =buffer
                        mov     r5, #4
                        mov     r6, #'n'
loop            bl              RJSTICK
                        bl              DJSTICK
                        cmp     r0, r6
                        mov     r6, r0
                        beq     loop
                        cmp     r0, #'n'
                        beq             loop
                        str             r0, [r3]
                        bleq    subfunction
                        cmp             r5, #0
                        add     r3, #1
                        beq     endless

subfunction             sub r5, #1
                        bx lr

endless         b               endless

                        ENDP

;*****************************************************************************
***
CONFIG                  PROC

                        ldr     r0, =(RCC_BASE+RCC_AHB2ENR)
                        ldr     r1, [r0]
                        orr     r1, #RCC_AHB2ENR_GPIOAEN
                        str     r1, [r0]

                        ldr     r0, =(GPIOA_BASE+GPIO_MODER)
                        ldr     r1, [r0]
                        bic             r1, r1, #GPIO_MODER_MODER0
```

```
                bic             r1, r1, #GPIO_MODER_MODER1
                bic             r1, r1, #GPIO_MODER_MODER2
                bic             r1, r1, #GPIO_MODER_MODER3
                bic             r1, r1, #GPIO_MODER_MODER5
                str     r1, [r0]

                ldr     r0, =(GPIOA_BASE+GPIO_PUPDR)
                ldr     r1, [r0]
                bic             r1, r1, #GPIO_PUPDR_PUPDR0_0
                bic             r1, r1, #GPIO_PUPDR_PUPDR1_0
                bic             r1, r1, #GPIO_PUPDR_PUPDR2_0
                bic             r1, r1, #GPIO_PUPDR_PUPDR3_0
                bic             r1, r1, #GPIO_PUPDR_PUPDR5_0
                orr             r1, r1, #GPIO_PUPDR_PUPDR0_1
                orr             r1, r1, #GPIO_PUPDR_PUPDR1_1
                orr             r1, r1, #GPIO_PUPDR_PUPDR2_1
                orr             r1, r1, #GPIO_PUPDR_PUPDR3_1
                orr             r1, r1, #GPIO_PUPDR_PUPDR5_1
                str     r1, [r0]

                bx              lr

                ENDP

RJSTICK         PROC

                ldr     r0, =(GPIOA_BASE+GPIO_IDR)
                ldr     r1, [r0]
                AND     r1, #0x0000002F

                bx              lr

                ENDP

DJSTICK         PROC

                mov     r0, r1
                bic     r0, #0xFFFFFFFE
                teq     r0, #0x00000001
                beq     CENTER

                mov     r0, r1
                bic     r0, #0xFFFFFFFD
                teq     r0, #0x00000002
                beq     LEFT
```

```
                mov     r0, r1
                bic     r0, #0xFFFFFFFB
                teq     r0, #0x00000004
                beq     RIGHT

                mov     r0, r1
                bic     r0, #0xFFFFFFF7
                teq     r0, #0x00000008
                beq     UP

                mov     r0, r1
                bic     r0, #0xFFFFFFDF
                teq     r0, #0x00000020
                beq     DOWN

                mov     r0, #'n'
                bx lr

CENTER          mov     r0,#'c'
                bx      lr
LEFT    mov     r0,#'l'
                bx      lr
RIGHT           mov     r0,#'r'
                bx      lr
UP              mov     r0,#'u'
                bx      lr
DOWN            mov     r0,#'d'
                bx      lr

                ENDP

;*****************************************************************************
***

                ALIGN
                AREA    myData, DATA, READWRITE
                ALIGN

;*****************************************************************************
***
buffer  DCB     'n', 'n', 'n', 'n', 'n'
;*****************************************************************************
***

        END
```


Memory 1
Address: 0x20000000
0x20000000: nnnnn.....
0x20000068: .....
0x200000D0: .....
0x20000138: .....
0x200001A0: .....


Memory 1
Address: 0x20000000
0x20000000: curdl.....
0x20000068: .....
0x200000D0: .....
0x20000138: .....
0x200001A0: .....