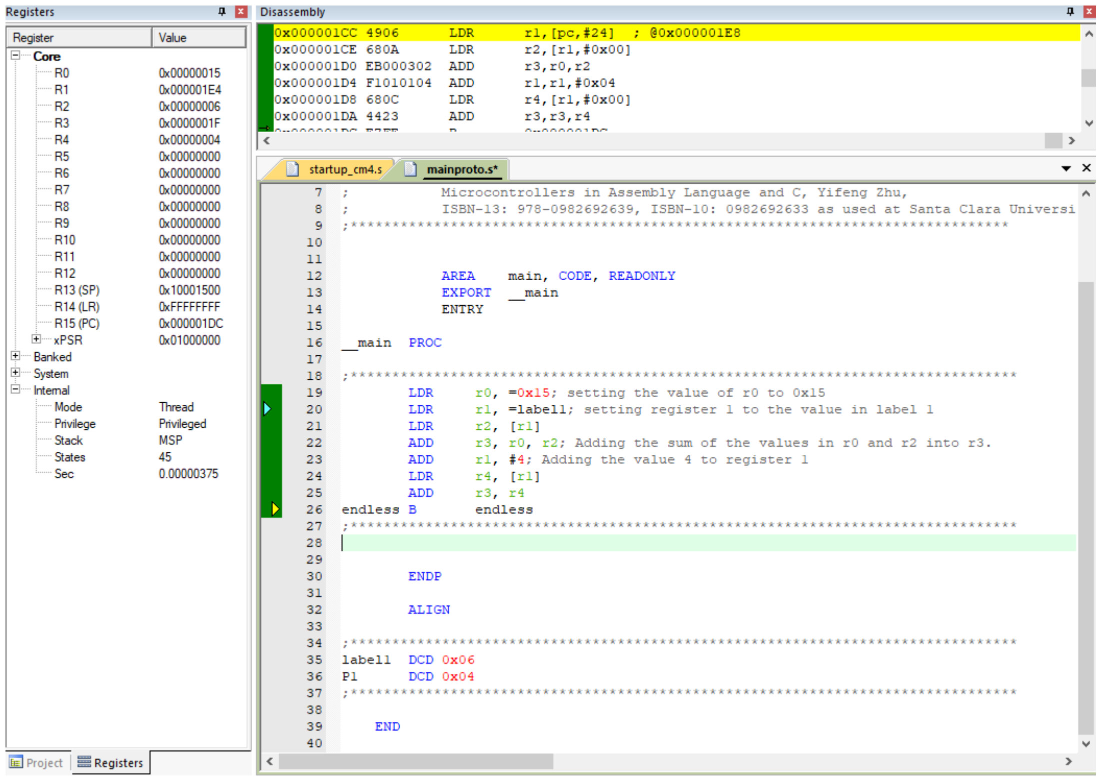
Christian Garcia

ELEN 120 Lab

October 5, 2021

Lab 1 - Introduction to ARM Assembly

Problem 1:

1. R0 = 0x00000015
2. R1 = 0x000001E0
3. R1 = 0x000001E4
4. R1 = 0x000001E4
5. R15(PC) = 0x000001D4
6. R3 = 0x0000001F
7. 

Register	Value
Core	
R0	0x00000015
R1	0x000001E4
R2	0x00000006
R3	0x0000001F
R4	0x00000004
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x10001500
R14 (LR)	0xFFFFFFFF
R15 (PC)	0x000001DC
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	45
Sec	0.00000375

```
0x000001CC 4906     LDR      r1,[pc,#24]  ; @0x000001E8
0x000001CE 680A     LDR      r2,[r1,#0x00]
0x000001D0 EB000302 ADD      r3,r0,r2
0x000001D4 F1010104 ADD      r1,r1,#0x04
0x000001D8 680C     LDR      r4,[r1,#0x00]
0x000001DA 4423     ADD      r3,r3,r4
```

```
;            Microcontrollers in Assembly Language and C, Yifeng Zhu,
;            ISBN-13: 978-0982692639, ISBN-10: 0982692633 as used at Santa Clara Universi
;*******************************************************************************


            AREA    main, CODE, READONLY
            EXPORT  __main
            ENTRY

__main  PROC

;*******************************************************************************
        LDR     r0, =0x15; setting the value of r0 to 0x15
        LDR     r1, =label1; setting register 1 to the value in label 1
        LDR     r2, [r1]
        ADD     r3, r0, r2; Adding the sum of the values in r0 and r2 into r3.
        ADD     r1, #4; Adding the value 4 to register 1
        LDR     r4, [r1]
        ADD     r3, r4
endless B       endless
;*******************************************************************************


        ENDP

        ALIGN

;*******************************************************************************
label1  DCD 0x06
P1      DCD 0x04
;*******************************************************************************

    END

```

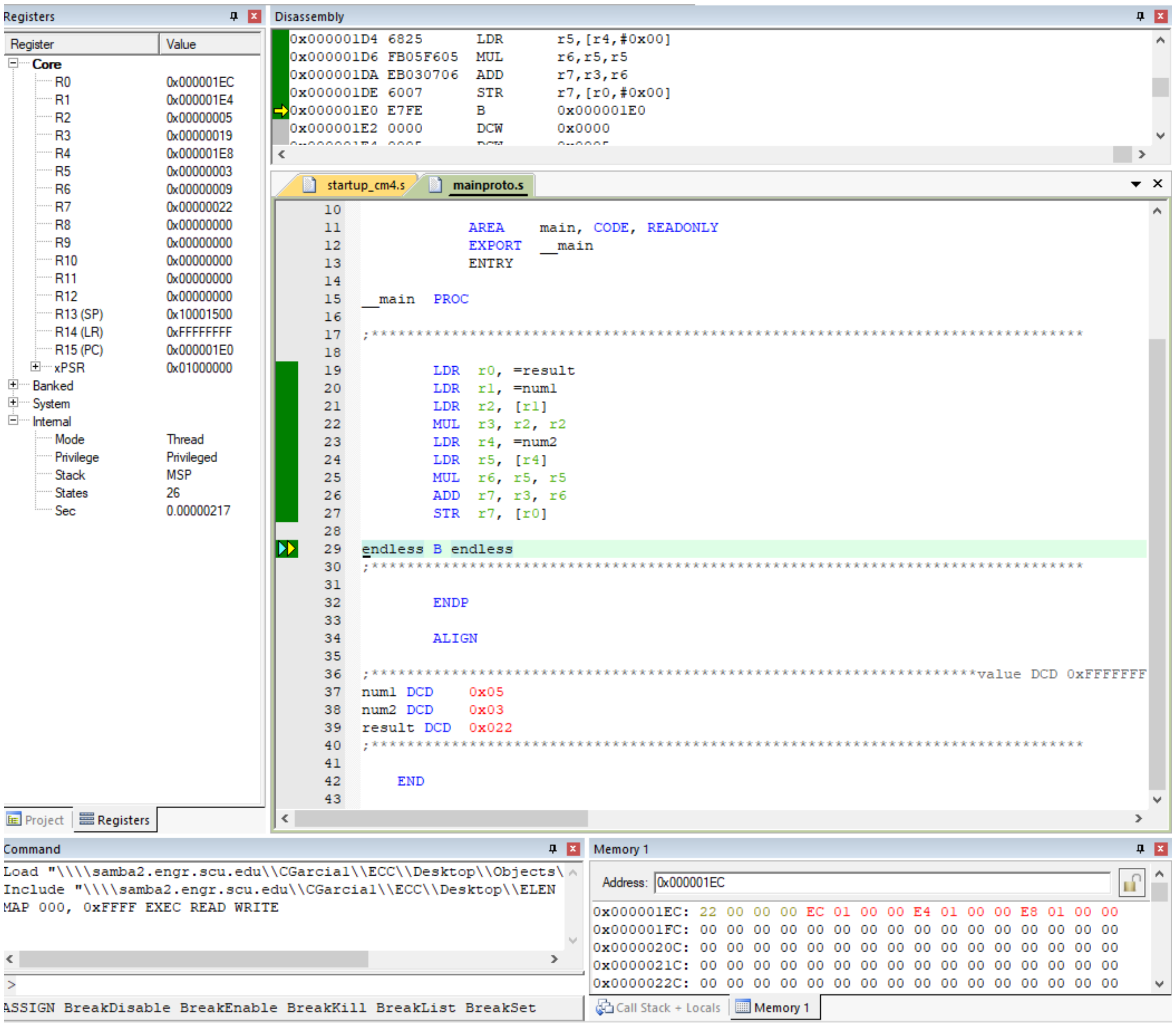
Problem 2:

1. The program gives the 2s complement number of whatever value is set.
2. R1 = 0x000001D8
3. No we cannot replace MVN with NEG because the 2s complement won't be correct since MVN gives the values logical not while NEG only multiplies the value by negative one.
4. No we cannot replace MVN with NOT because NOT is not a real instruction. So the program won't run.

Problem 3:

1. R1 = 0x000001EC
2. R2 = 0x00000003
3. The instruction is shifting each bit to the left. So the previous value for R2 0x03 or 00000011 becomes 00000110 or 0x06 doubling the value.
4. R2 = 0x00000008
5. No both instructions are doing the same thing moving the bits 1 space to the left, doubling the value.

Problem 4:



Registers

Register	Value
Core	
R0	0x000001EC
R1	0x000001E4
R2	0x00000005
R3	0x00000019
R4	0x000001E8
R5	0x00000003
R6	0x00000009
R7	0x00000022
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x10001500
R14 (LR)	0xFFFFFFFF
R15 (PC)	0x000001E0
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	26
Sec	0.00000217

Disassembly

```
0x000001D4 6825     LDR      r5,[r4,#0x00]
0x000001D6 FB05F605 MUL      r6,r5,r5
0x000001DA EB030706 ADD      r7,r3,r6
0x000001DE 6007     STR      r7,[r0,#0x00]
0x000001E0 E7FE     B        0x000001E0
0x000001E2 0000     DCW      0x0000
```

startup_cm4.s mainproto.s

```
            AREA    main, CODE, READONLY
            EXPORT  __main
            ENTRY

__main  PROC

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

        LDR  r0, =result
        LDR  r1, =num1
        LDR  r2, [r1]
        MUL  r3, r2, r2
        LDR  r4, =num2
        LDR  r5, [r4]
        MUL  r6, r5, r5
        ADD  r7, r3, r6
        STR  r7, [r0]

endless B endless
;*******************************************************************************

        ENDP

        ALIGN

;*********************************************************************value DCD 0xFFFFFFF
num1 DCD    0x05
num2 DCD    0x03
result DCD  0x022
;*******************************************************************************

    END
```

Command

```
Load "\\\\samba2.engr.scu.edu\\CGarcia1\\ECC\\Desktop\\Objects\
Include "\\\\samba2.engr.scu.edu\\CGarcia1\\ECC\\Desktop\\ELEN
MAP 000, 0xFFFF EXEC READ WRITE
```

```
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet
```

Memory 1

Address: 0x000001EC

```
0x000001EC: 22 00 00 00 EC 01 00 00 E4 01 00 00 E8 01 00 00
0x000001FC: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0000020C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0000021C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0000022C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```