

Experiment 26

Introduction to the ARM Microprocessor

Submission Template of Experimental Results

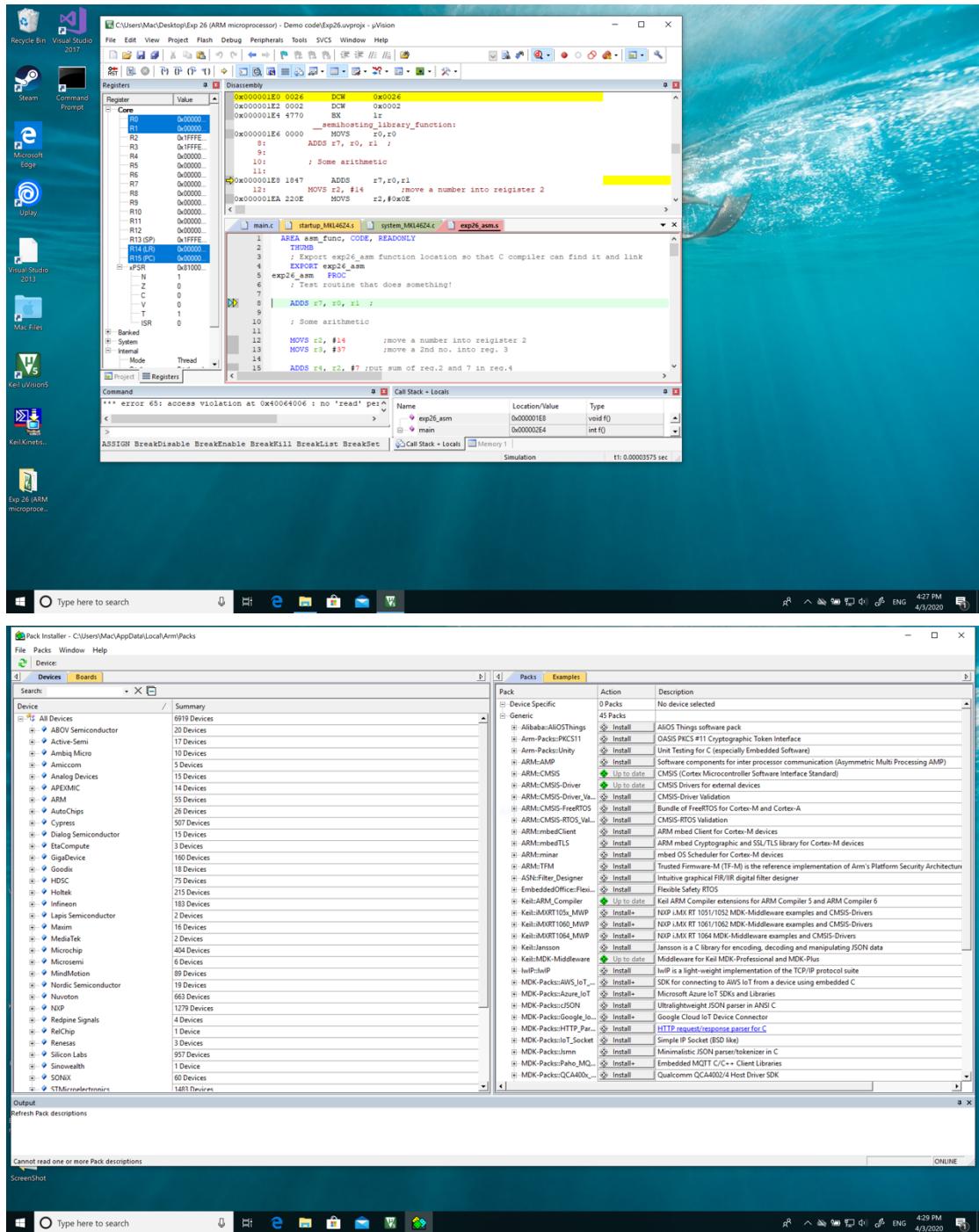
Important: Marking of all coursework is anonymous. Do not include your name, student ID number, group number, email or any other personal information in your report or in the name of the file submitted via VITAL. A penalty will be applied to submissions that do not meet this requirement.

Experimental Results and Comments (Total 90 marks):

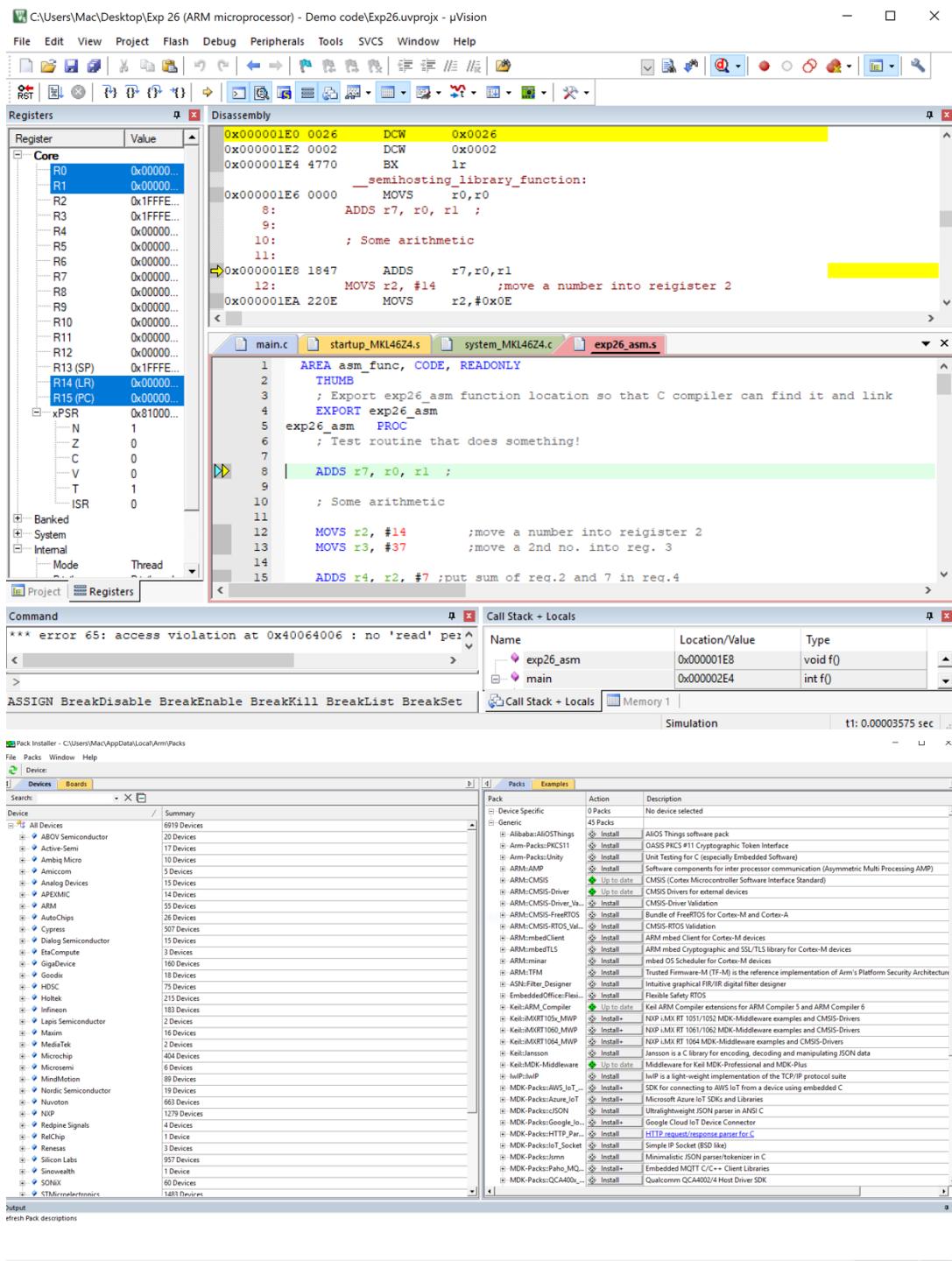
Please provide the required screenshots, photos, code, values highlighted in the script according to the following requirements (screenshots should be clear and readable):

1. Screenshot of the result of Section 4 [2 marks]

Screenshot (PrintScreen or Screenshot):



Screenshot (Snipping tool, Preview or equivalent):



2. Answer to Q1 [5 marks]

Answer:

The Second value in Hex (From the left to right)

Explanation:

In ARMv6-M Architecture Reference Manual, in **A6.7.39**

00100 **XXX** XXXXXXXX(Binary)

Opcode Rd Immediate Value

where: <Rd> The destination register.

For example: 2700(Hex) MOVS r7,#0x00

In the machine code:

The second part which is Rd identifies the register used

3. Answer to Q2 [5 marks]

Answer:

The third and fourth number in Hex (from the left to right)

Explanation:

00100 XXX XXXXXXXX

Opcode Rd Immediate Value

In ARMv6-M Architecture Reference Manual, in **A6.7.39**

Referring to MOV (immediate):

MOVS <Rd>, #<const>

where: <const> The immediate value to be placed in <Rd>. The range of permitted values is 0-255 for encoding T1.

For example:

2700(Hex)->MOVES r7,#0x00

4. Answer to Q3 [5 marks]

Answer:

Before executed ADDS r4, r2, #7, the r4 is *0x00000318*;

After executed ADDS r4, r2, #7, the r4 is *0x00000015*

Explanation:

The ADDS r4, r2, #7 means that use r2's value *0x0000000E* adds 7(decimal) which is *0x00000007*, so the result is *0x00000015* then the sum is move to the r4

5. Answer to Q4 [5 marks]

Answer:

Very important, for ADDS rx, ry, rz, if you only reverse the order of ry and rz, the result will not change. However, for SUBS rx, ry, rz, if you only reverse the order of ry and rz, the result will be different.

Explanation:

In ADDS:

ADDS r5, r2, r3 ;put sum of r2 and r3 in r5

ADDS r6, r3, r2 ;reverse the order

$r5 = r2 + r3$ and $r6 = r3 + r2$, so in add is ok to reverse ry and rz.

(Not means it can reverse rx to ry or rz)

In SUBS:

SUBS r1, r2, r3 ;subtract r3 from r2

SUBS r2, r3, r2 ;reverse the order

$r1 = r2 - r3$ and $r2 = r3 - r2$, so the result is different.

6. Answer to Q5 [5 marks]

Answer: YES

Register values table: (After executed IA+0x12, before r2 is

0xFFFFFA6)

R0	<i>0x00000063</i>
R1	<i>0x0000005A</i>
R2	<i>0x00000011</i>
R3	<i>0x0000000A</i>
R4	<i>0x0000006B</i>
R5	<i>0x0000006E</i>
R6	<i>0x0000006E</i>
R7	<i>0x00000000</i>
R8	<i>0x00000000</i>
R9	<i>0x00000000</i>
R10	<i>0x00000000</i>
R11	<i>0x00000000</i>
R12	<i>0x00000000</i>
R13	<i>0x1FFE160</i>
R14	<i>0x00000014D</i>
R15	<i>0x000001FC</i>
xPSR	<i>0x01000000</i>

Explanation:

Because it change the R2 and R3 before executing IA, so the R2 is initially set by $0x00000064$ and R3 is initially set by $0x0000000A$.

ADDS r4, r2, #7 ;put sum of reg.2 and 7 in reg.4 = $0x00000006B$

ADDS r5, r2, r3 ;put sum of r2 and r3 in r5 = $0x00000006E$

ADDS r6, r3, r2 ;put sum of r3 and r2 in r6 = $0x00000006E$

SUBS r0, r2, #1 ;subtract 1 from r2: $0x00000064 - 1 = 0x000000063$

SUBS r1, r2, r3 ;subtract r3 from r2 $0x00000064 - 0x0000000A = 0x00000005A$

SUBS r2, r3, r2 ;reverse the order r2 = $0x0000000A - 0x00000064 = 0xFFFFFA6$

MOVS r2, #0x0011 ;set bit pattern 0011 (in hex)

7. Answer to Q6 [5 marks]

Answer: YES

Let **R2** and **R3** = *0x00000000*;

Register values table:

R0	<i>0xFFFFFFFF</i>
R1	<i>0x00000000</i>
R2	<i>0x00000011</i>
R3	<i>0x00000000</i>
R4	<i>0x00000007</i>
R5	<i>0x00000000</i>
R6	<i>0x00000000</i>
R7	<i>0x00000000</i>
R8	<i>0x00000000</i>
R9	<i>0x00000000</i>
R10	<i>0x00000000</i>
R11	<i>0x00000000</i>
R12	<i>0x00000000</i>
R13	<i>0x1FFE160</i>
R14	<i>0x00000014D</i>
R15	<i>0x000001FC</i>
xPSR	<i>0x21000000</i>

Explanation:

Because we change the R2 and R3 before executing IA, so the R2 is initially set by *0x00000000* and R3 is initially set by *0x00000000*.

ADDS r4, r2, #7 ;put sum of reg.2 and 7 in reg.4 = *0x00000007*

ADDS r5, r2, r3 ;put sum of r2 and r3 in r5 = *0x00000000*

ADDS r6, r3, r2 ;put sum of r3 and r2 in r6 = *0x00000000*

SUBS r0, r2, #1 ;subtract 1 from r2: *0x00000000 - 1 = 0xFFFFFFFF*

SUBS r1, r2, r3 ;subtract r3 from r2 r1= *0x00000000*

SUBS r2, r3, r2 ;reverse the order r2 = *0x00000000*

MOVS r2, #0x0011 ;set bit pattern 0011 (in hex)

8. Answer to Q7 [5 marks]

Answer: Logic AND

Truth Table in binary

A	B	C
000010001	100000001	000000001

Explanation:

MOV r4, r2 ;r4 = r2 = 0x00000011

ANDS r4, r3 ;AND r3=0x00000101 r4=0x00000001

000010001

100000001

AND -----

000000001

9. Answer to Q8 [5 marks]

Answer:

Order is not important for ANDS, ORRS and EORS

But for BICS the order is important

Explanation:

MOV r4, r2

ANDS r4, r3 ;AND

MOV r5, r2

ORRS r5, r3 ;OR

MOV r6, r2

EORS r6, r3 ;exclusive OR

MOV r0, r2

BICS r0, r3 ;bit clear

MOV r1, r3

BICS r1, r2 ;bit clear in reverse order

10. Answer to Q9 [5 marks]

Answer:

Before executed: PC is *0x00000218*;

After executed: PC is *0x00000216*;

Explanation:

loop1

ADDS r2, r2, #1 ;increment register 2

B loop1 ;loop always

The branch jump to the loop1

which is the *0x00000216* → ADDS r2, r2, #1 ;increment register 2

11. Answer to Q10 [7 marks]

Answer:

$$R2 = 0x00000005$$

Explanation:

Because the r2 is 0 in the first time, and the function ADDS r2, r2, #1 runs five times, so the $r2 = 0x00000005$.

12. Answer to Q11 [5 marks]

Answer:

Because my computer's system clock is almost constant during this time

Explanation:

00:19.96

0x04785BDE

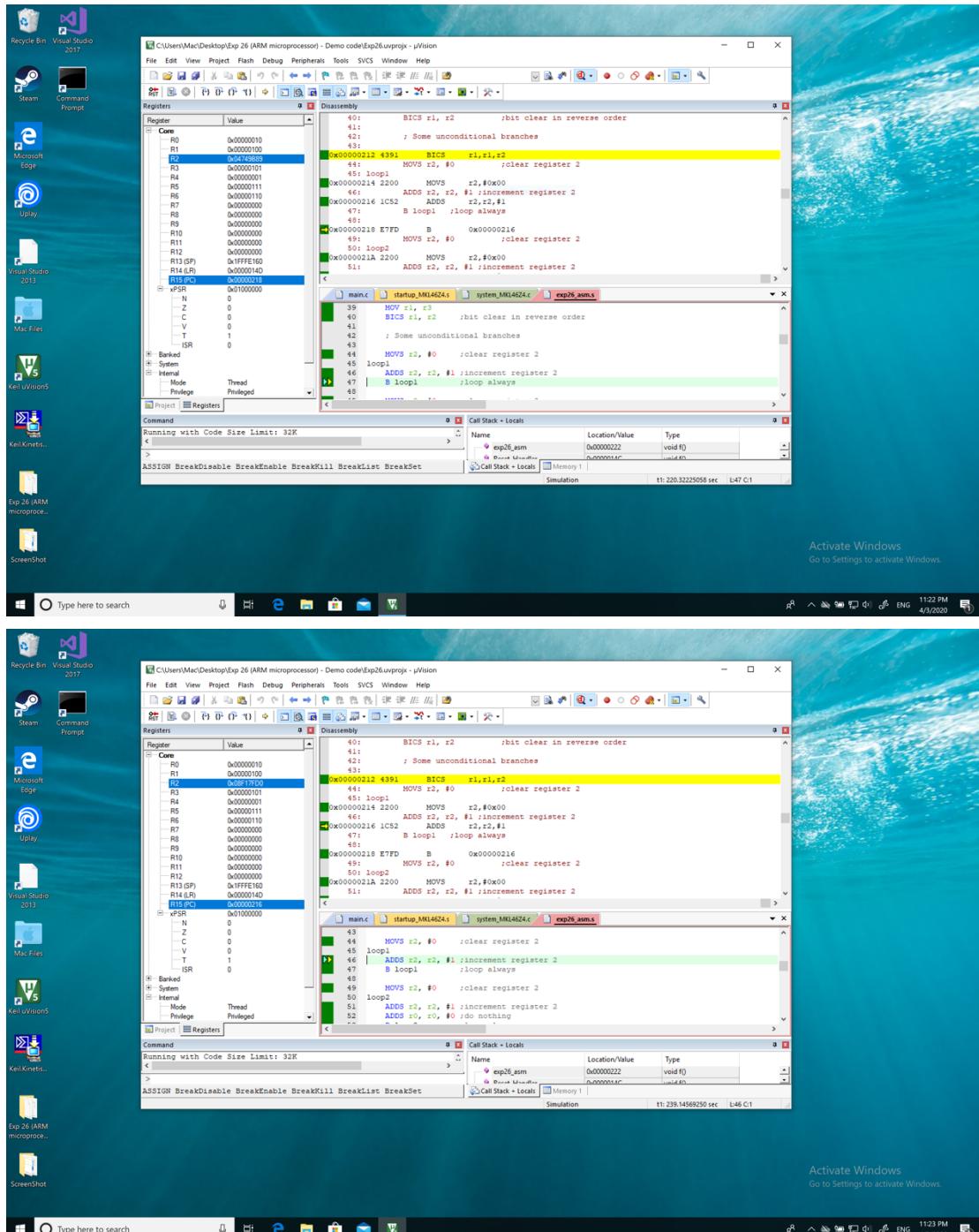
00:39.86

0x08F473E8

13. The graph of Section 8 (using MS Excel or MATLAB) [8 marks]

Screenshot (PrintScreen or Screenshot):

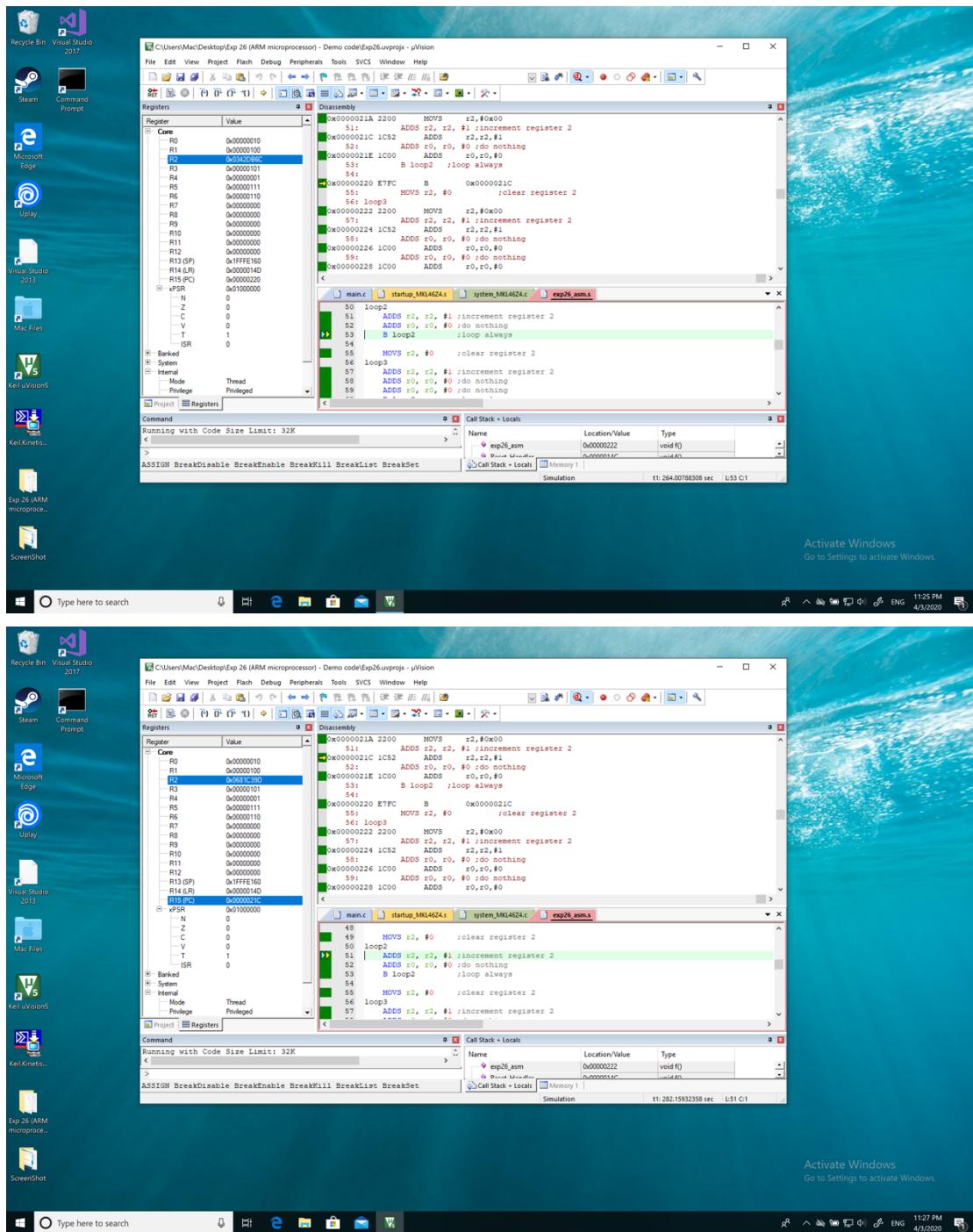
Loop1:



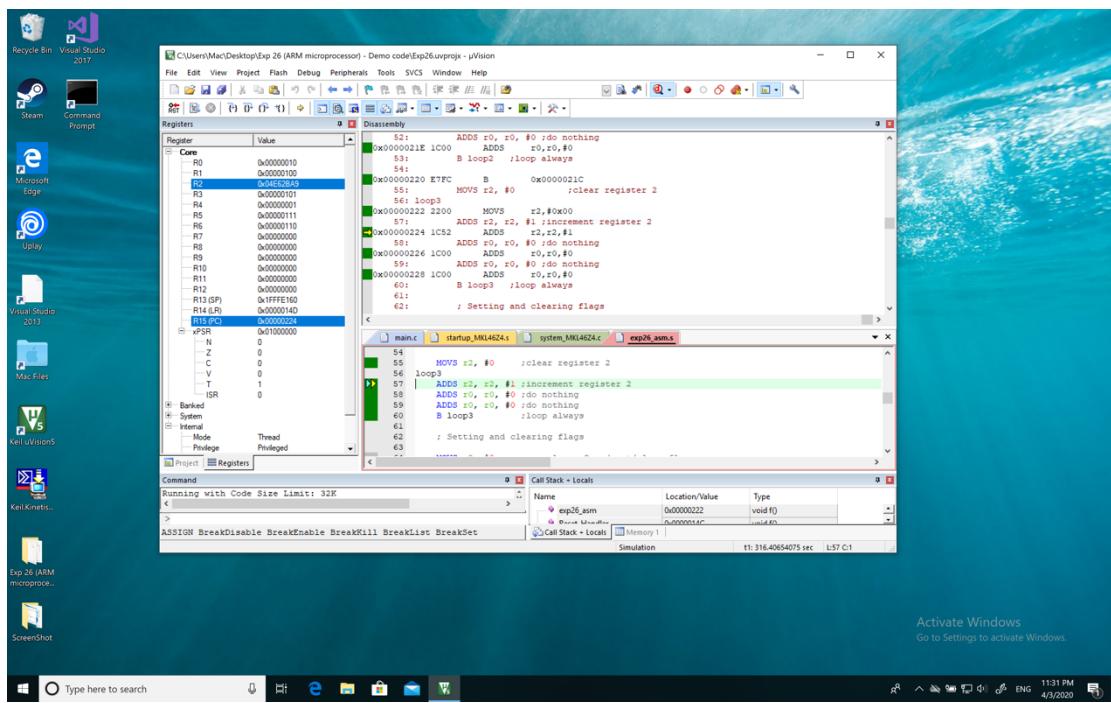
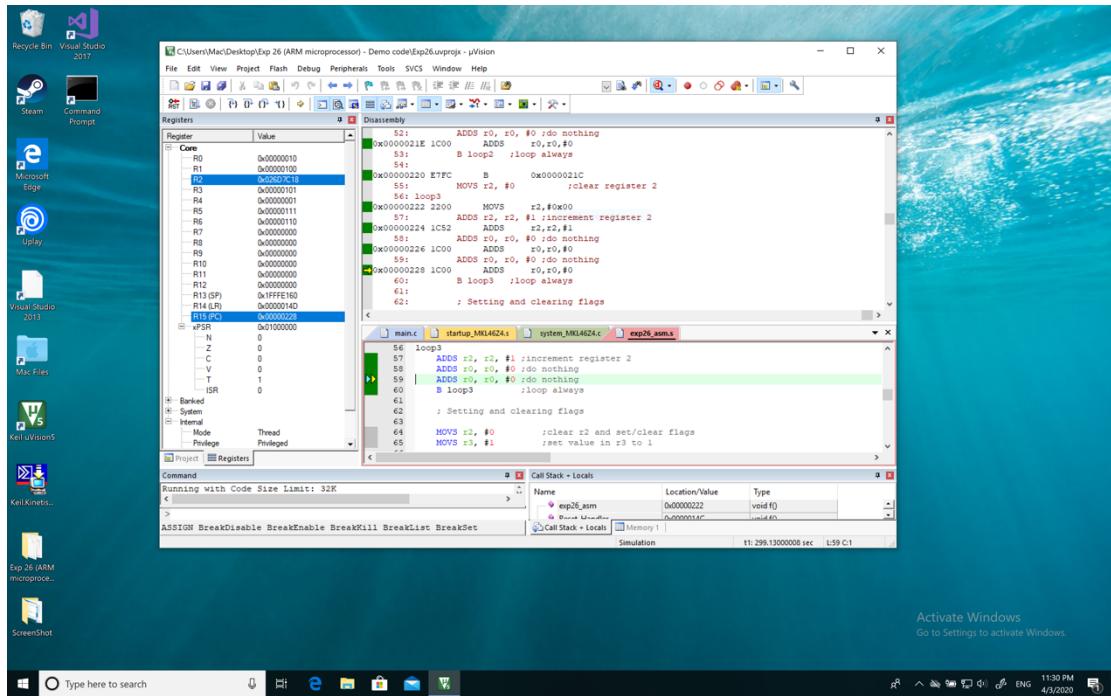
The screenshot shows the Keil µVision IDE interface with the following details:

- Project:** C:\Users\Mac\Desktop\Exp 26 (ARM microprocessor) - Demo code\Exp26.uvproj - µVision
- Registers View:** Shows the ARM register state. R2 is highlighted in blue.
- Disassembly View:** Displays assembly code for Loop1. The highlighted instruction is BICS r1, r2 at address 0x00000213.
- Code Editor:** Shows the C source code for main.c, startup_MK14624.s, and system_MK14624.c. The assembly file exp26.asm is also visible.
- Call Stack & Locals:** Shows the call stack and local variables for the current function.
- Simulation:** Shows the simulation status with time t1: 220.32225058 sec and L47 C1.
- Windows Taskbar:** Shows the Windows taskbar with various icons and the date/time: 4/3/2020 11:23 PM.

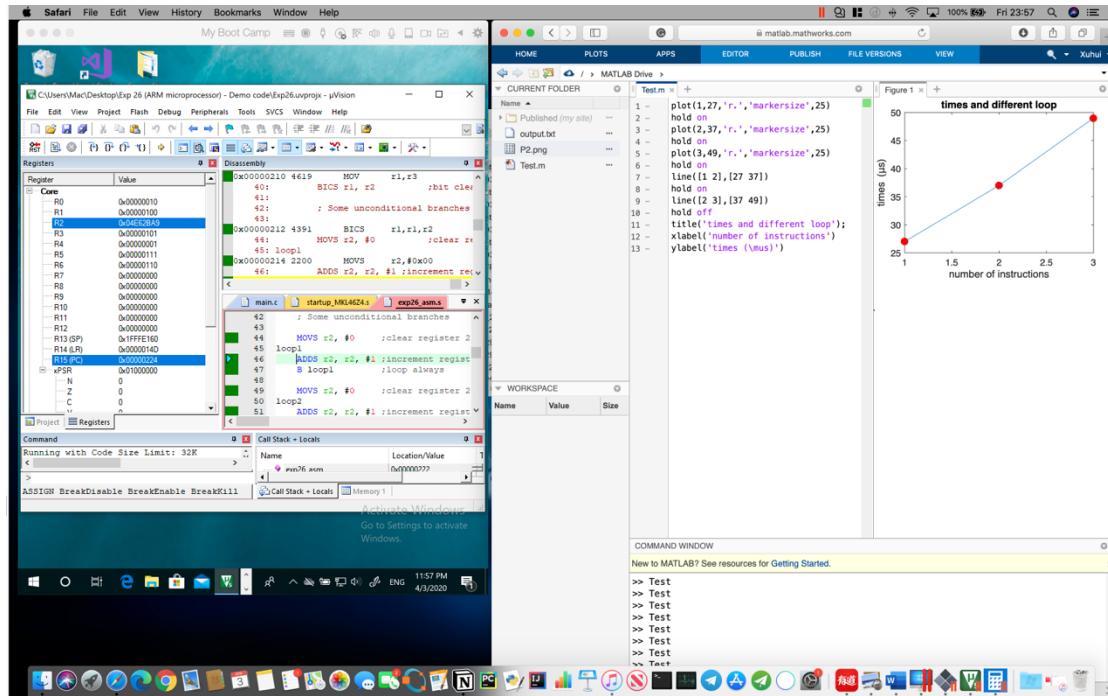
Loop2:



Loop3:



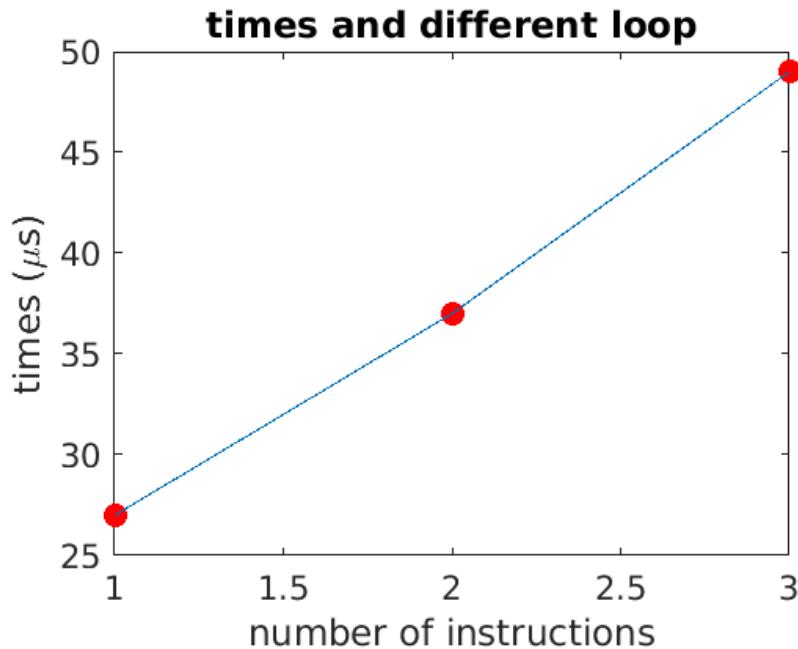
MATLAB Work



Screenshot (Snipping tool, Preview or equivalent):

The Branch is not count to the instructions (if add, the x-axis plus

one)



Comment/Explanation:

In loop1:

00:20.06
<i>0x04749B89</i>

00:39.86
<i>0x08F17FD0</i>

3764290 times/s

0.00000027 s/times

In loop2:

00:20.03
<i>0x0342DB6C</i>

00:40.04
<i>0x0681C39D</i>

2726461 times/s

0.00000037 s/times

In loop3:

00:20.06
0x026D7C18

00:40.00
0x04E62BA9

2054833 times/s

0.00000049 s/times

The more instruction added, the more run time in each loop

14. Answer to Q12 [5 marks]

Answer:

$$11\mu s$$

Explanation:

In loop1 and loop2, there are added one instruction and it led to add almost $10\mu s$, in loop2 and loop3, there are added one instruction and it led to add almost $12\mu s$. The add instruction requires one clock cycle to execute, and the branch instruction requires more clock cycles. However, all the loop has branch instruction so the time of branch instruction is offset, the one clock cycle can be calculated by using simple subtraction.

15. Answer to Q13 [4 marks]

Answer:

$$15.6/11 \text{ cycle} \sim 2 \text{ cycles}$$

Explanation:

The add instruction requires one clock cycle to execute, and the branch instruction requires more clock cycles. We know one clock

cycle is almost $11\mu s$, the total time in the loop1 is $27\mu s$, so the branch need $16\mu s$; the total time in the loop2 is $37\mu s$, $37\mu s - 2 \times \text{add instructions} = 37\mu s - 11\mu s * 2 = 15\mu s$; the total time in the loop3 is $49\mu s$, $49\mu s - 3 \times \text{add instructions} = 49\mu s - 11\mu s * 3 = 16\mu s$;

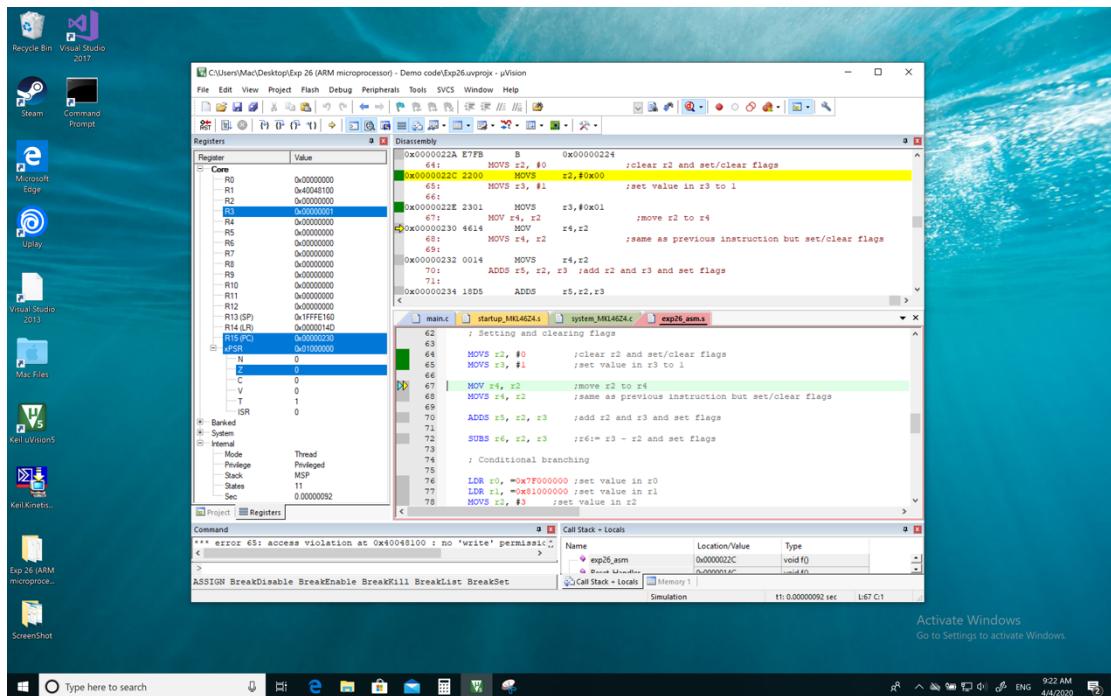
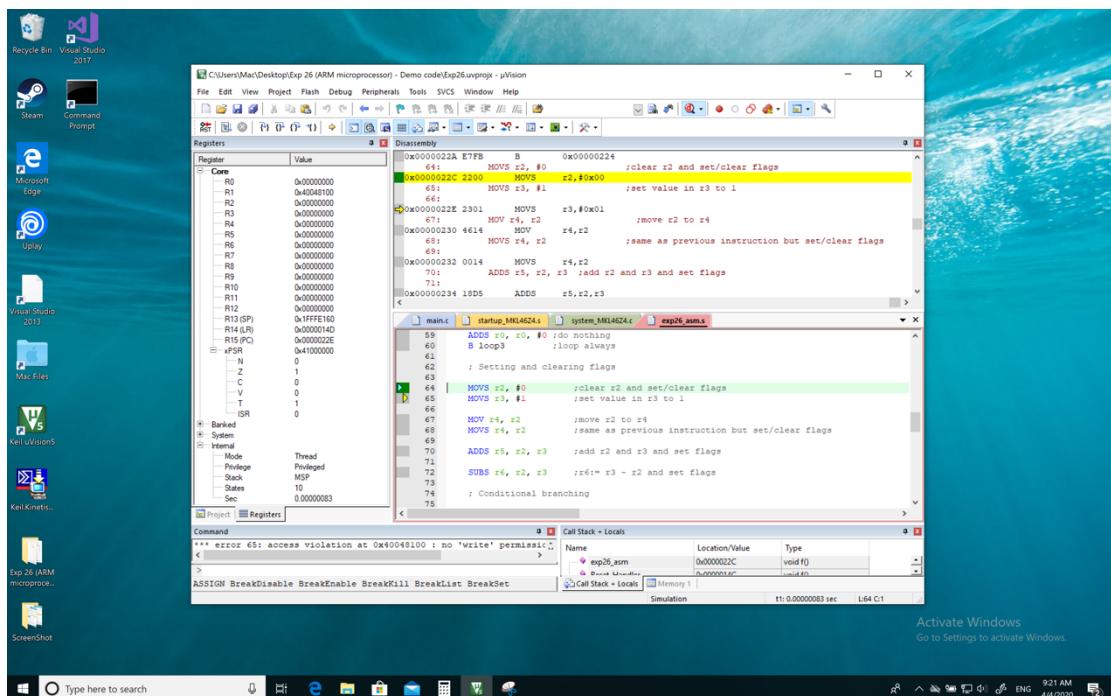
So, taking into account the error, the average value is $15.6\mu s$,

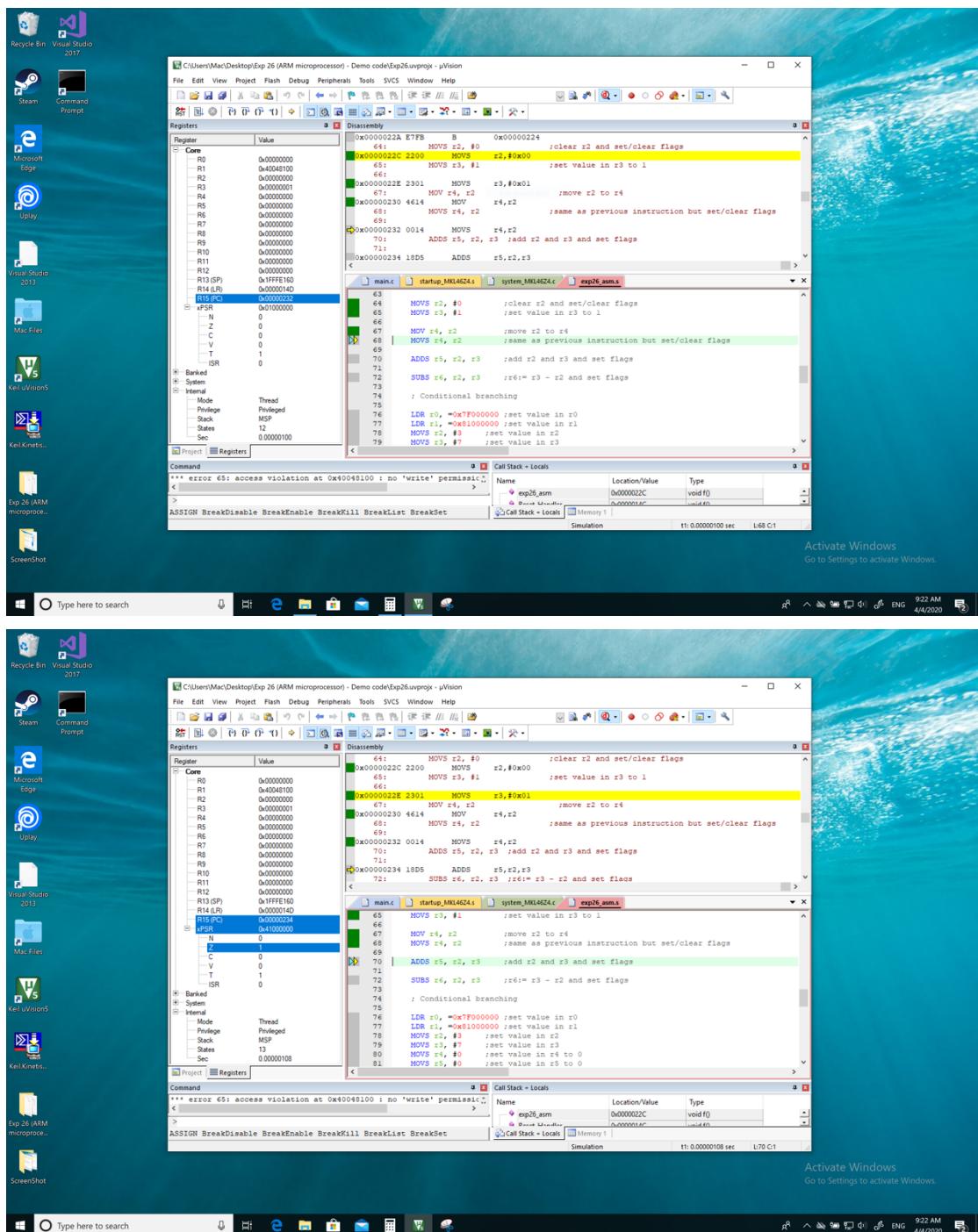
So the clock cycle is $15.6/11$

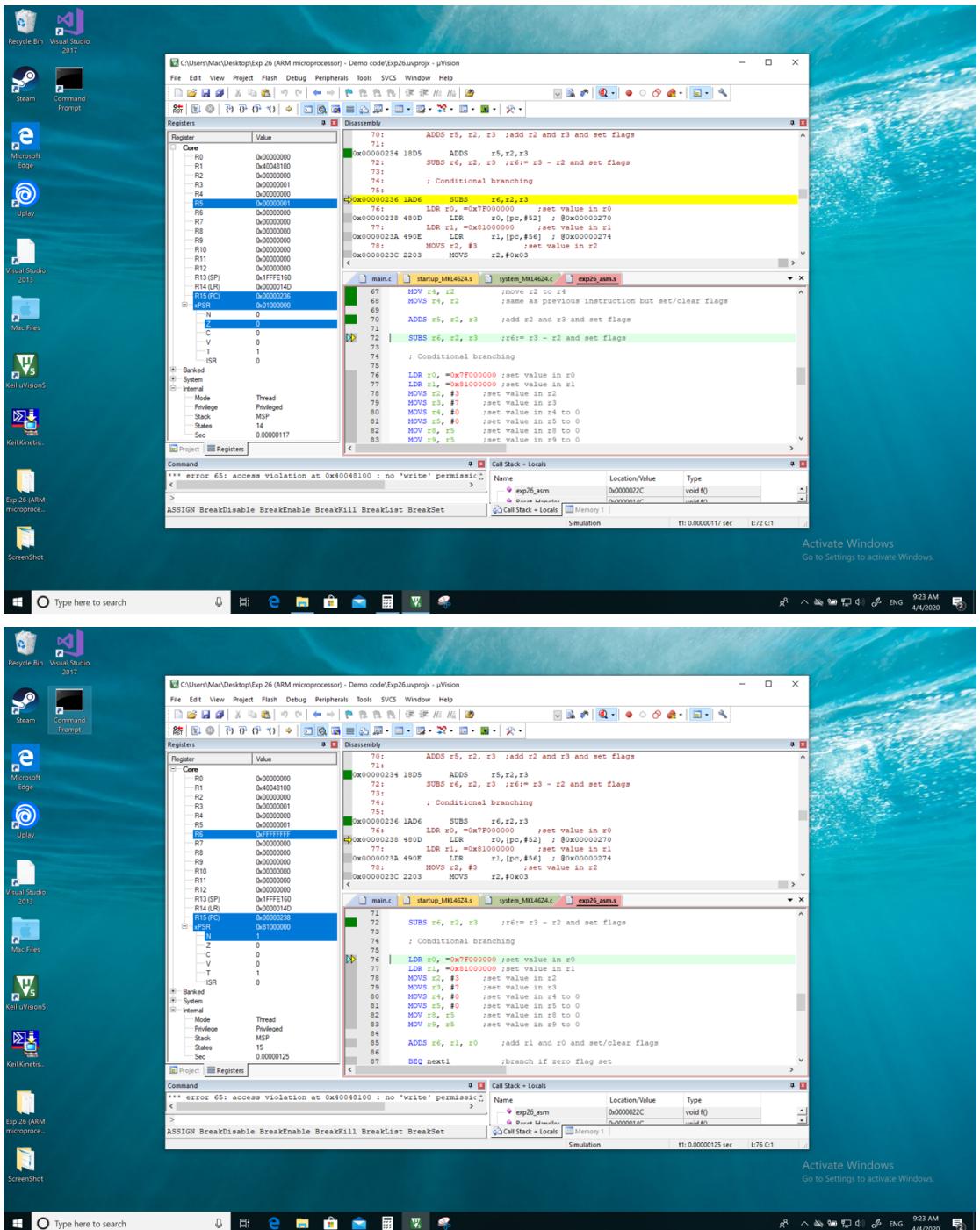
16. Screenshot of the result of Section 9 [4 marks]

Screenshot (PrintScreen or Screenshot):

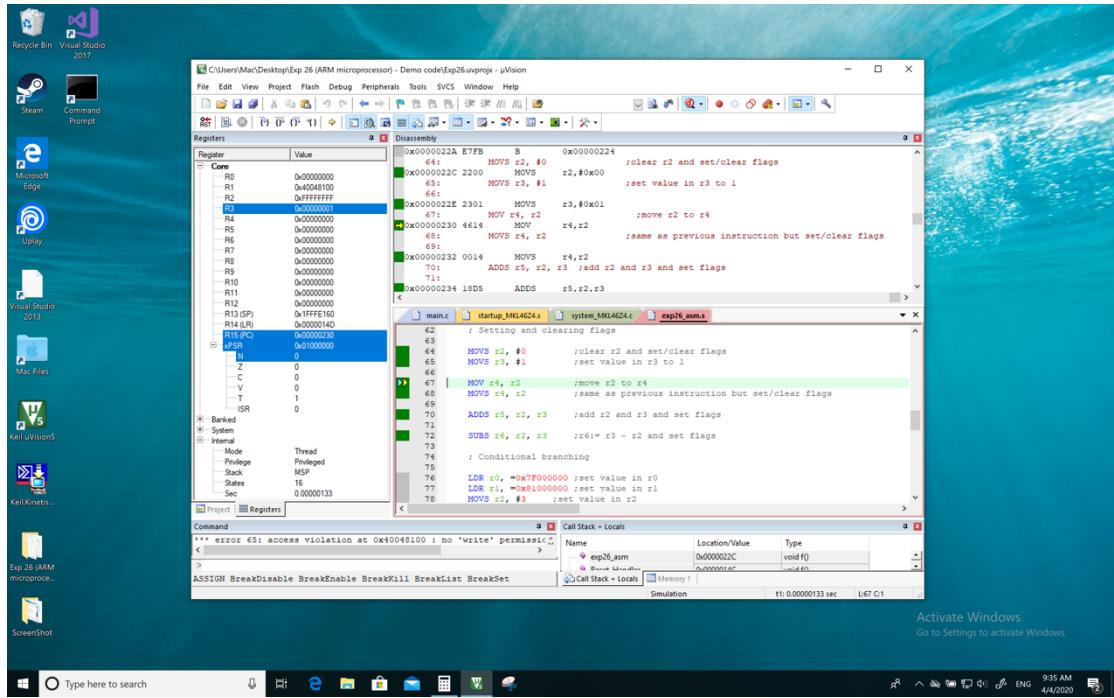
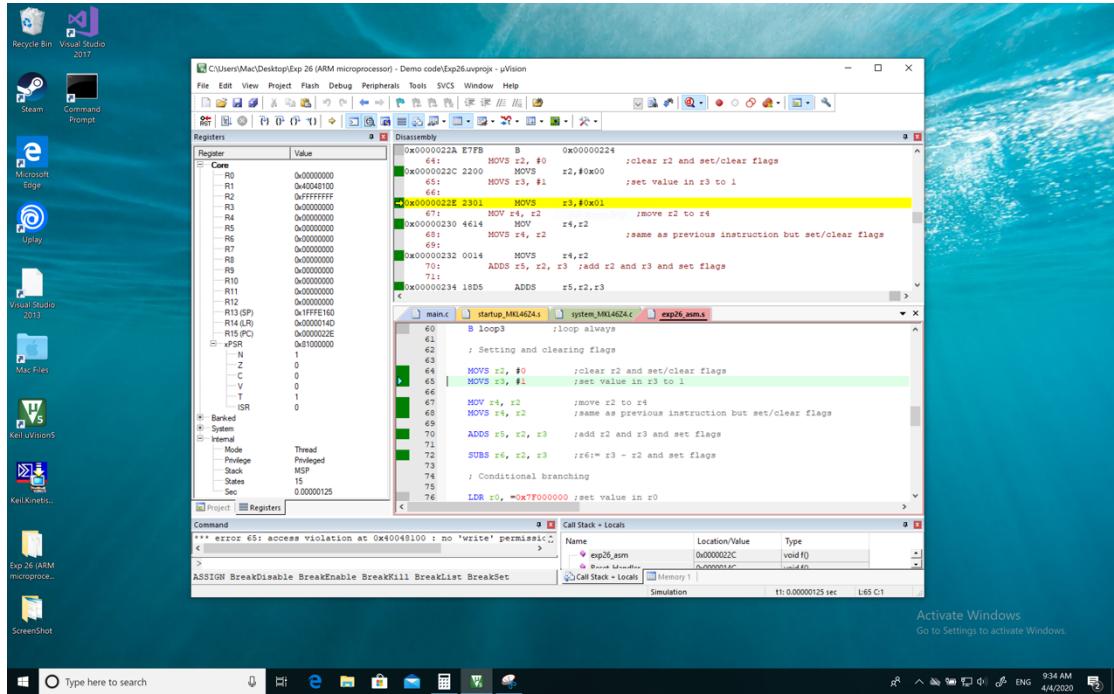
(1) R2 initial

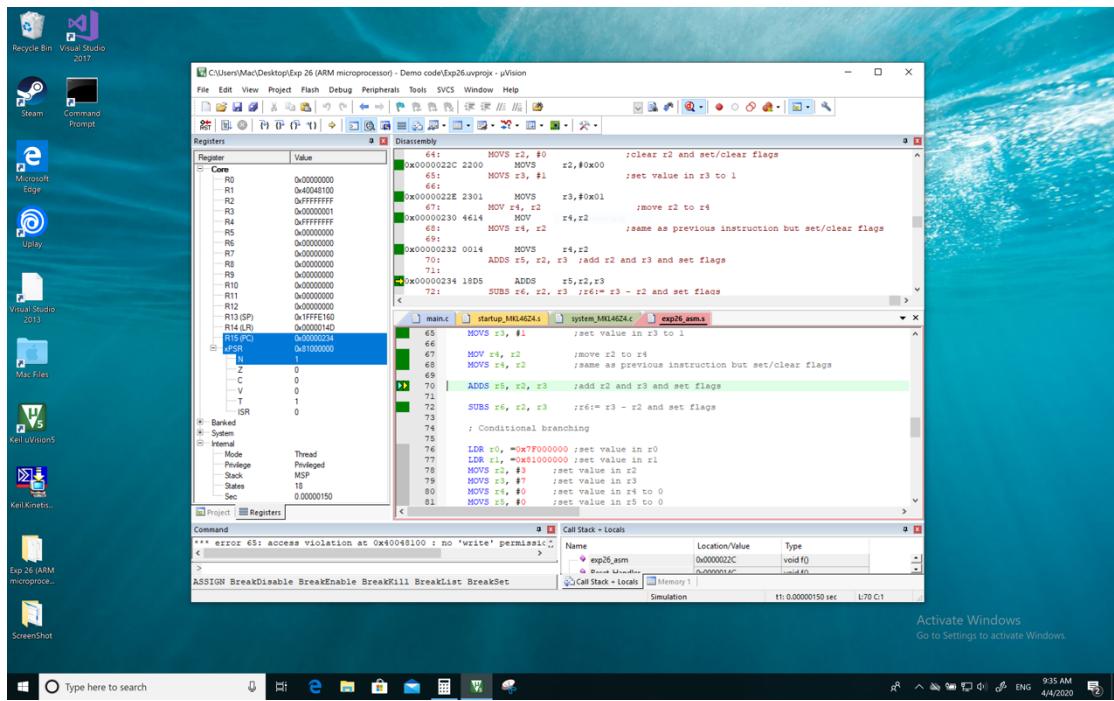
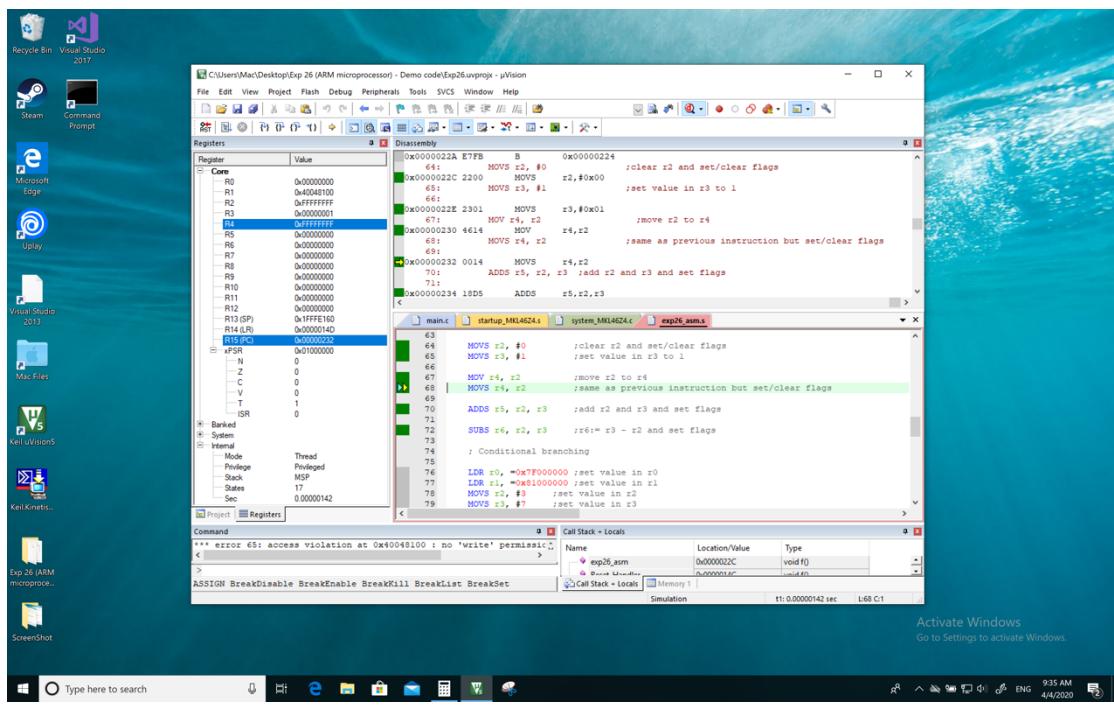


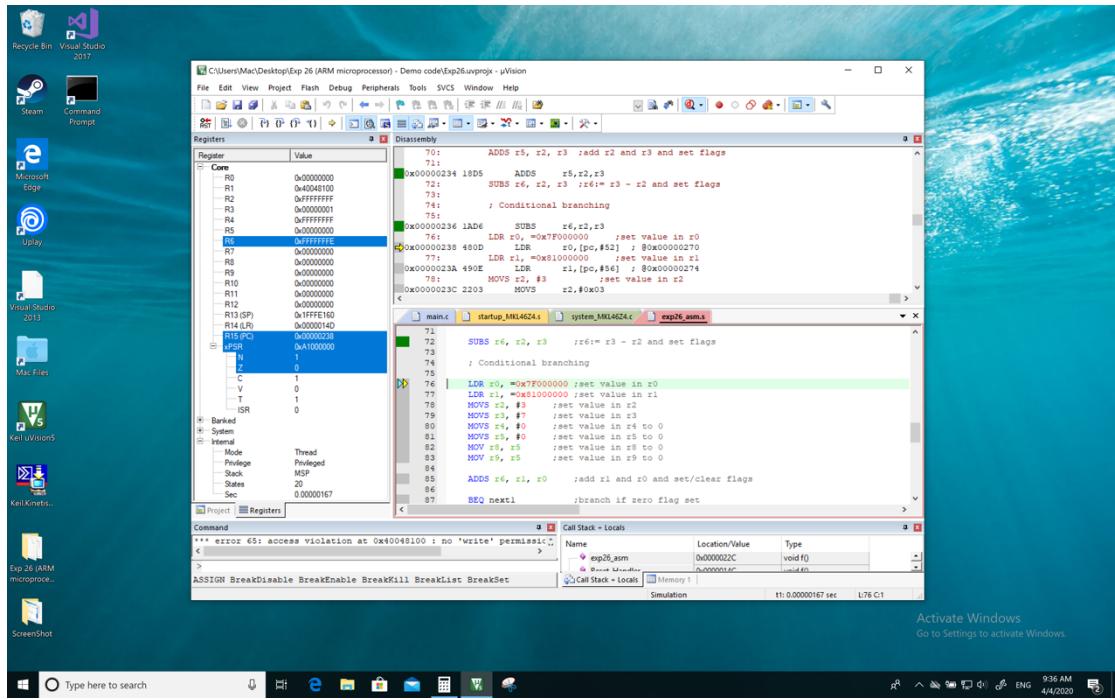
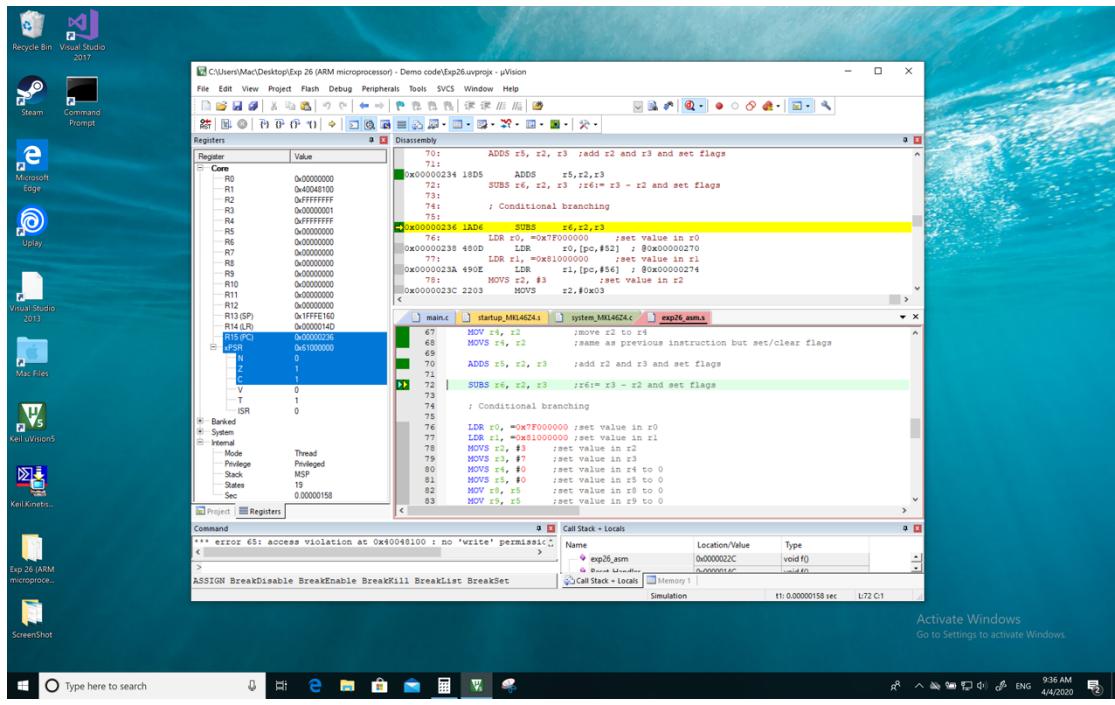




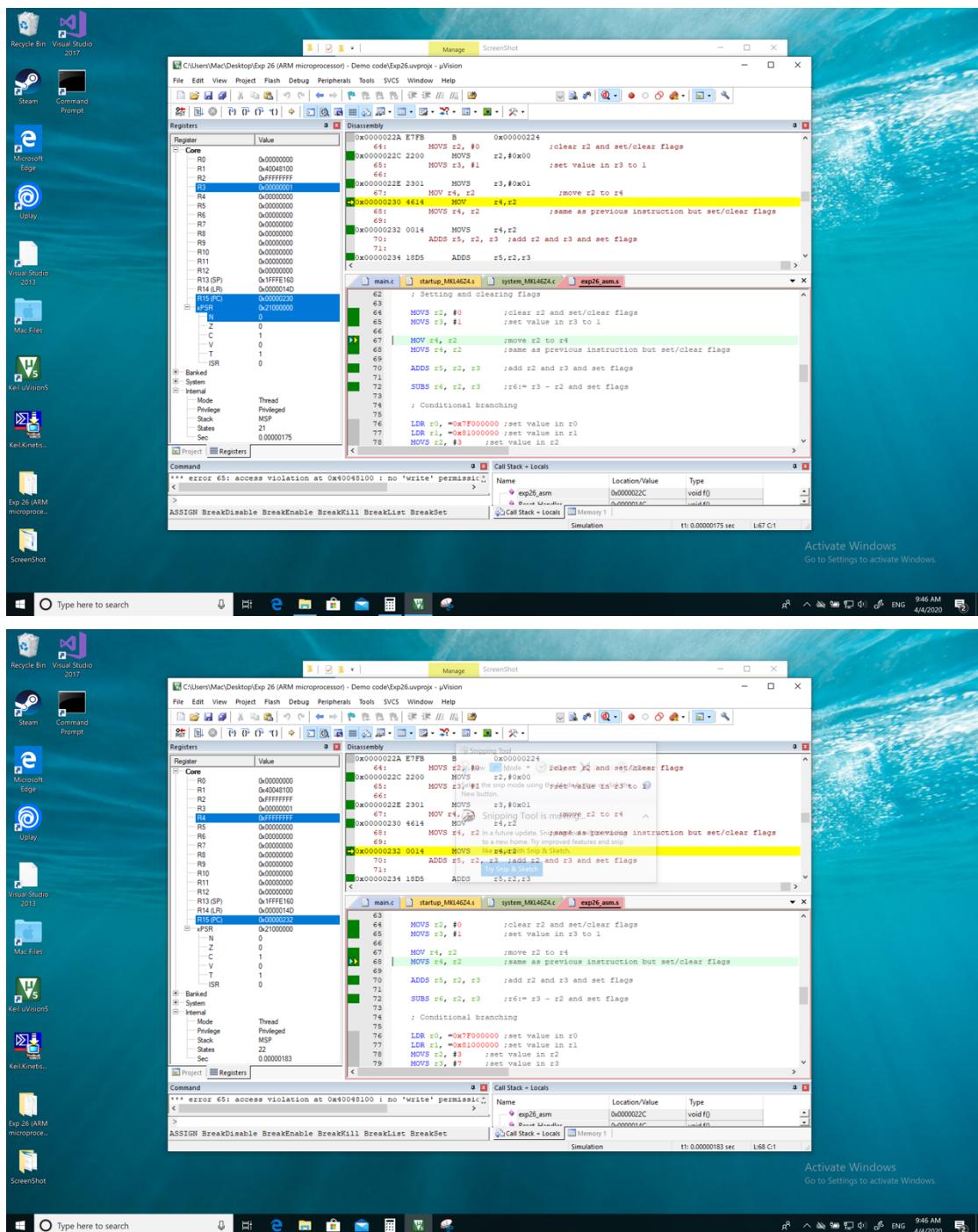
(2) R2 to FFFFFFFF

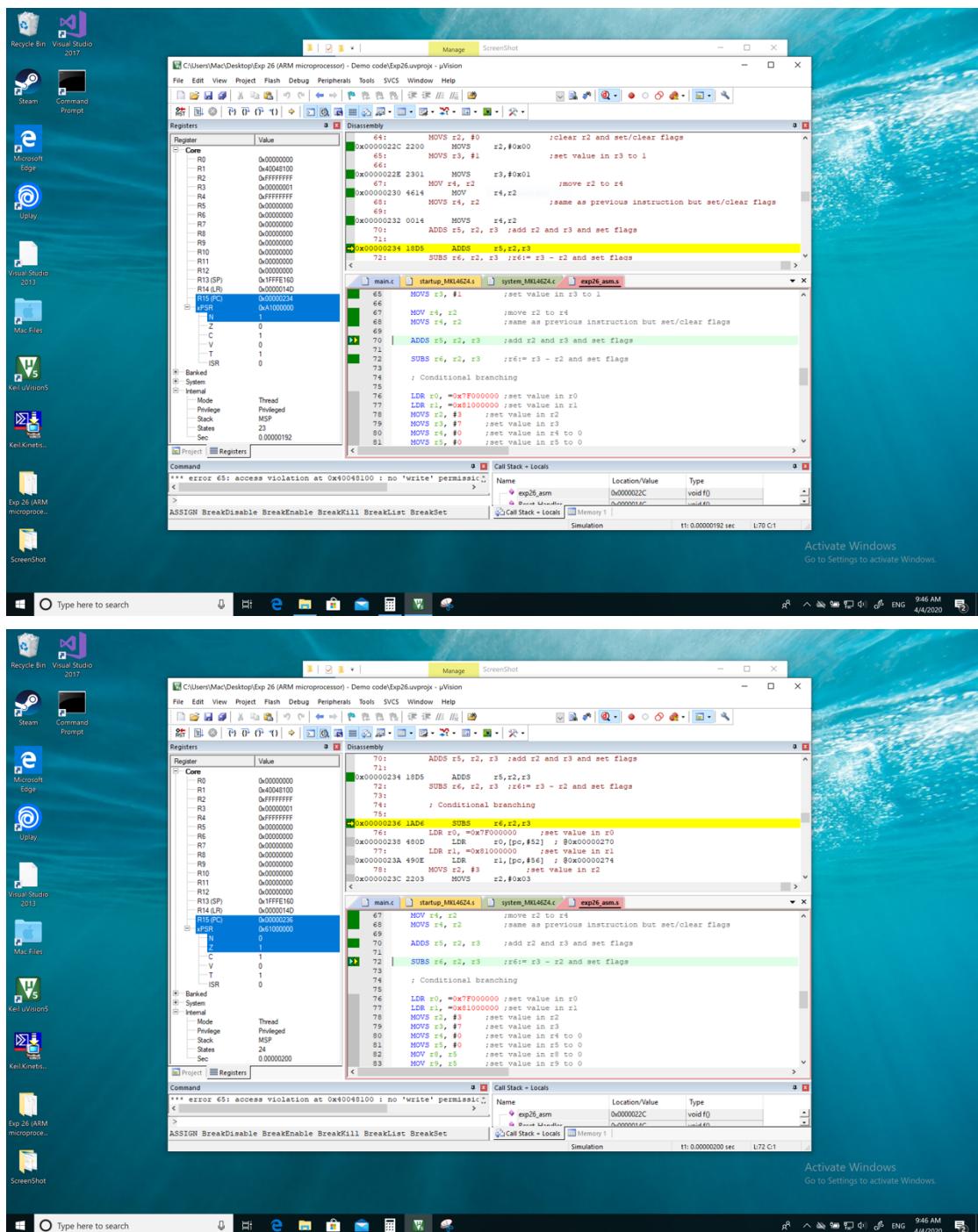


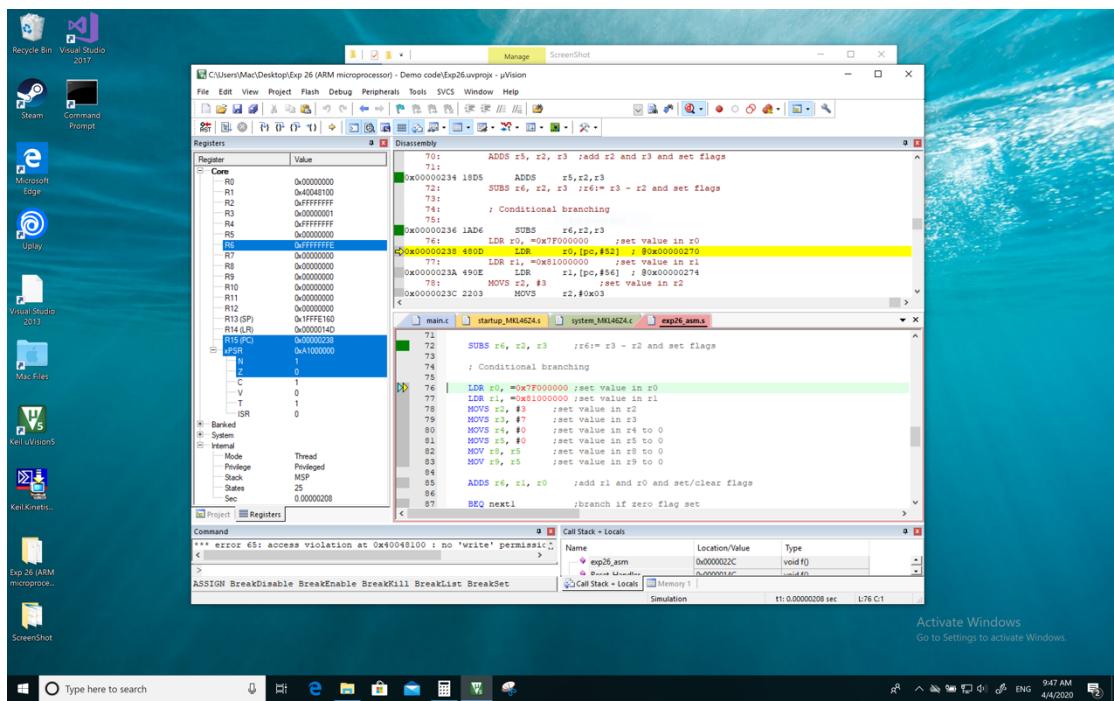




(3) R2 to 7FFFFFFF







Screenshot (Snipping tool, Preview or equivalent):

(1) R2 initial

C:\Users\Mac\Desktop\Exp 26 (ARM microprocessor) - Demo code\Exp26.uvproj - µVision

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

Registers

Register	Value
R0	0x00000000
R1	0x40048100
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13(GP)	0xFFFFE160
R14(LR)	0x0000002E
vFSR	0x41000000
N	0
Z	1
C	0
V	0
T	1
ISR	0

Banked

System

Internal

- Mode Thread
- Privileged
- Stack
- Stack SP
- States 10
- Sec 0.00000083

Project Registers

Command

```
*** error 65: access violation at 0x40048100 : no 'write' permission.
< >
> ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet
```

Call Stack + Locals

Name	Location/Value	Type
exp26_ssm	0x0000002C	void()
Break Monitor	0x0000014C	void()

Simulation t1: 0.00000083 sec L64 C1

C:\Users\Mac\Desktop\Exp 26 (ARM microprocessor) - Demo code\Exp26.uvproj - µVision

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

Registers

Register	Value
R0	0x00000000
R1	0x40048100
R2	0x00000000
R3	0x00000001
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13(GP)	0xFFFFE160
R14(LR)	0x0000002D
vFSR	0x10000000
N	0
Z	0
C	0
V	0
T	1
ISR	0

Banked

System

Internal

- Mode Thread
- Privileged
- Stack
- Stack SP
- States 11
- Sec 0.00000092

Project Registers

Command

```
*** error 65: access violation at 0x40048100 : no 'write' permission.
< >
> ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet
```

Call Stack + Locals

Name	Location/Value	Type
exp26_ssm	0x0000002C	void()
Break Monitor	0x0000014C	void()

Simulation t1: 0.00000092 sec L67 C1

C:\Users\Mac\Desktop\Exp 26 (ARM microprocessor) - Demo code\Exp26.uvproj - µVision

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

Registers

Register	Value
R0	0x00000000
R1	0x40048100
R2	0x00000000
R3	0x00000001
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13(GP)	0xFFFFE160
R14(LR)	0x0000002D
vFSR	0x10000000
N	0
Z	0
C	0
V	0
T	1
ISR	0

Banked

System

Internal

- Mode Thread
- Privileged
- Stack
- Stack SP
- States 12
- Sec 0.00000100

Project Registers

Command

```
*** error 65: access violation at 0x40048100 : no 'write' permission.
< >
> ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet
```

Call Stack + Locals

Name	Location/Value	Type
exp26_ssm	0x0000002C	void()
Break Monitor	0x0000014C	void()

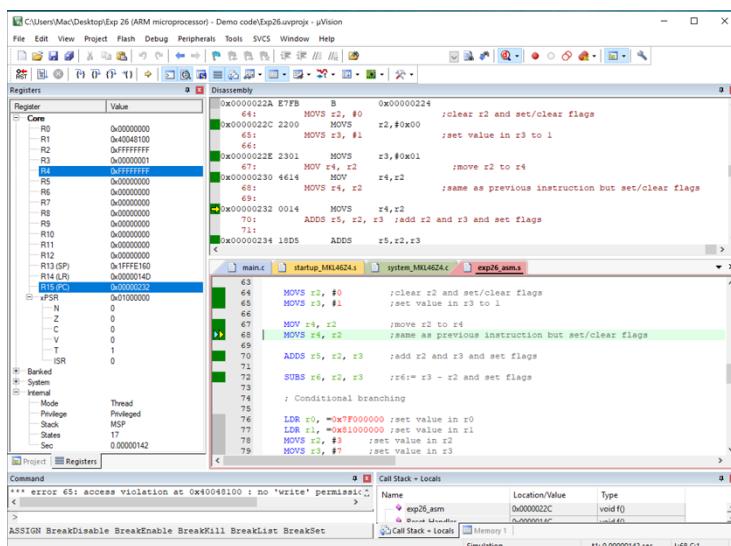
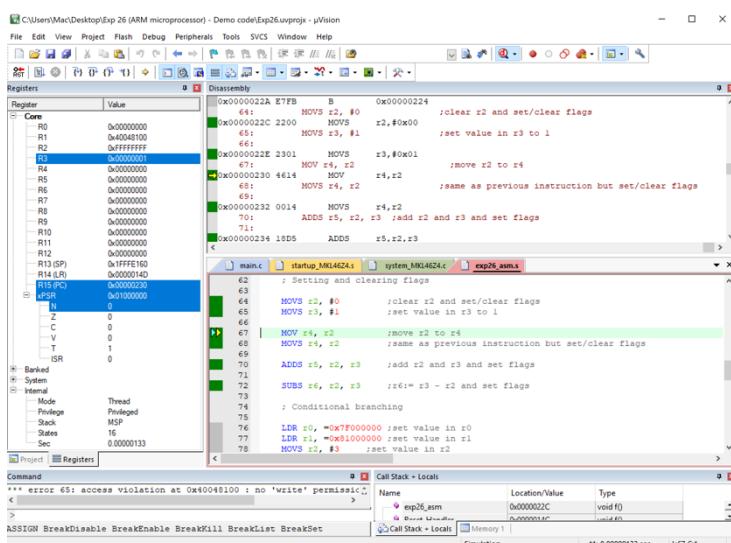
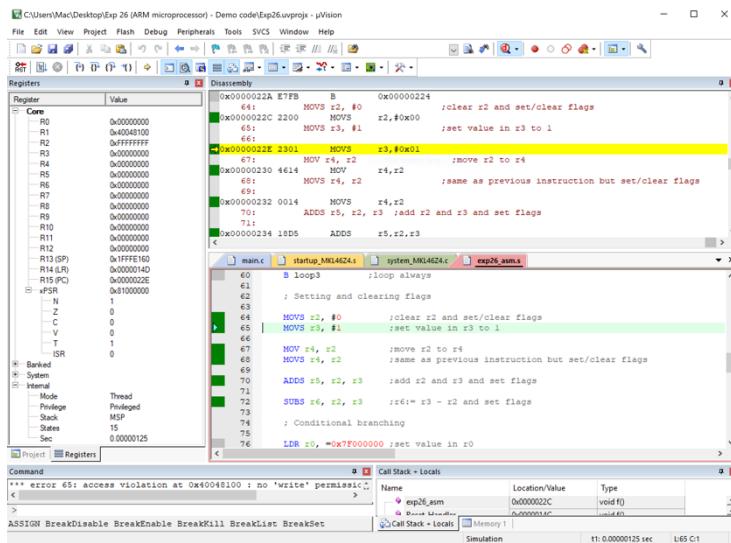
Simulation t1: 0.00000100 sec L68 C1

(2) R2 to FFFFFFFF

```

C:\Users\Mac\Desktop\Exp 26 (ARM microprocessor) - Demo code\Exp26.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers Disassembly
Registers
Register Value
Core
R0 0x00000000
R1 0x40048100
R2 0x00000000
R3 0x00000001
R4 0x00000000
R5 0x00000000
R6 0x00000000
R7 0x00000000
R8 0x00000000
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 (SP) 0xFFFFE160
R14 (LR) 0x0000014D
R15 (PC) 0x00000234
+FSR N 0
Z 1
C 0
V 0
T 1
ISR 0
Banked
System
Internal
Mode Thread
Privilege Privileged
Stack MSP
States 13
Sec 0.00000108
Project Registers
Command
*** error 65: access violation at 0x00048100 : no 'write' permission.
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet
Call Stack + Locals
Name Location/Value Type
exp26_arm 0x0000022C void(f)
+ BreakList 0x000002C void(f)
Call Stack + Locals Memory 1 Simulation t1:0.00000108 sec L70 C1
C:\Users\Mac\Desktop\Exp 26 (ARM microprocessor) - Demo code\Exp26.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers Disassembly
Registers
Register Value
Core
R0 0x00000000
R1 0x40048100
R2 0x00000000
R3 0x00000001
R4 0x00000000
R5 0x00000000
R6 0x00000000
R7 0x00000000
R8 0x00000000
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 (SP) 0xFFFFE160
R14 (LR) 0x0000014D
R15 (PC) 0x00000236
+FSR N 0
Z 0
C 0
V 0
T 1
ISR 0
Banked
System
Internal
Mode Thread
Privilege Privileged
Stack MSP
States 14
Sec 0.00000117
Project Registers
Command
*** error 65: access violation at 0x00048100 : no 'write' permission.
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet
Call Stack + Locals
Name Location/Value Type
exp26_arm 0x0000022C void(f)
+ BreakList 0x000002C void(f)
Call Stack + Locals Memory 1 Simulation t1:0.00000117 sec L72 C1
C:\Users\Mac\Desktop\Exp 26 (ARM microprocessor) - Demo code\Exp26.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers Disassembly
Registers
Register Value
Core
R0 0x00000000
R1 0x40048100
R2 0x00000000
R3 0x00000001
R4 0x00000000
R5 0x00000000
R6 0xFFFFFFF
R7 0x00000000
R8 0x00000000
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 (SP) 0xFFFFE160
R14 (LR) 0x0000014D
R15 (PC) 0x00000238
+FSR N 0
Z 0
C 0
V 0
T 1
ISR 0
Banked
System
Internal
Mode Thread
Privilege Privileged
Stack MSP
States 15
Sec 0.00000125
Project Registers
Command
*** error 65: access violation at 0x00048100 : no 'write' permission.
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet
Call Stack + Locals
Name Location/Value Type
exp26_arm 0x0000022C void(f)
+ BreakList 0x000002C void(f)
Call Stack + Locals Memory 1 Simulation t1:0.00000125 sec L76 C1

```



C:\Users\Mac\Desktop\Exp 26 (ARM microprocessor) - Demo code\Exp26.uvproj - µVision

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

Registers Disassembly

Register	Value
R0	0x00000000
R1	0x40048100
R2	0xFFFFFFFF
R3	0x00000001
R4	0xFFFFFFFF
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13(SP)	0x1FFE160
R14(LR)	0x0000014D
R15(PC)	0x00000024
N	1
Z	0
C	1
V	0
T	1
ISR	0

```

    .L1:    MOVW r2, #0           ;clear r2 and set/clear flags
    .L2:    MOVS r2, #0x0000      ;set value in r2 to 1
    .L3:    MOVW r3, #1           ;set value in r3 to 1
    .L4:    MOVS r3, #0           ;set value in r3 to 0
    .L5:    ADDS r2, r2, r3      ;move r2 to r2
    .L6:    MOVS r2, #0x0001      ;move r2 to r2
    .L7:    ADDS r2, r2, r3      ;move r2 to r2
    .L8:    MOVS r2, r3          ;move r2 to r2
    .L9:    ADDS r2, r2, r3      ;move r2 to r2
    .L10:   MOVS r2, r3          ;move r2 to r2
    .L11:   SUBS r2, r2, r3      ;move r2 to r2
    .L12:   MOVS r2, r3          ;move r2 to r2
    .L13:   ADDS r2, r2, r3      ;move r2 to r2
    .L14:   MOVS r2, r3          ;move r2 to r2
    .L15:   ADDS r2, r2, r3      ;move r2 to r2
    .L16:   MOVS r2, r3          ;move r2 to r2
    .L17:   ADDS r2, r2, r3      ;move r2 to r2
    .L18:   MOVS r2, r3          ;move r2 to r2
    .L19:   ADDS r2, r2, r3      ;move r2 to r2
    .L20:   MOVS r2, r3          ;move r2 to r2
    .L21:   ADDS r2, r2, r3      ;move r2 to r2
    .L22:   MOVS r2, r3          ;move r2 to r2
    .L23:   ADDS r2, r2, r3      ;move r2 to r2
    .L24:   MOVS r2, r3          ;move r2 to r2
    .L25:   ADDS r2, r2, r3      ;move r2 to r2
    .L26:   MOVS r2, r3          ;move r2 to r2
    .L27:   ADDS r2, r2, r3      ;move r2 to r2
    .L28:   MOVS r2, r3          ;move r2 to r2
    .L29:   ADDS r2, r2, r3      ;move r2 to r2
    .L30:   MOVS r2, r3          ;move r2 to r2
    .L31:   ADDS r2, r2, r3      ;move r2 to r2
    .L32:   MOVS r2, r3          ;move r2 to r2
    .L33:   ADDS r2, r2, r3      ;move r2 to r2
    .L34:   MOVS r2, r3          ;move r2 to r2
    .L35:   ADDS r2, r2, r3      ;move r2 to r2
    .L36:   MOVS r2, r3          ;move r2 to r2
    .L37:   ADDS r2, r2, r3      ;move r2 to r2
    .L38:   MOVS r2, r3          ;move r2 to r2
    .L39:   ADDS r2, r2, r3      ;move r2 to r2
    .L40:   MOVS r2, r3          ;move r2 to r2
    .L41:   ADDS r2, r2, r3      ;move r2 to r2
    .L42:   MOVS r2, r3          ;move r2 to r2
    .L43:   ADDS r2, r2, r3      ;move r2 to r2
    .L44:   MOVS r2, r3          ;move r2 to r2
    .L45:   ADDS r2, r2, r3      ;move r2 to r2
    .L46:   MOVS r2, r3          ;move r2 to r2
    .L47:   ADDS r2, r2, r3      ;move r2 to r2
    .L48:   MOVS r2, r3          ;move r2 to r2
    .L49:   ADDS r2, r2, r3      ;move r2 to r2
    .L50:   MOVS r2, r3          ;move r2 to r2
    .L51:   ADDS r2, r2, r3      ;move r2 to r2
    .L52:   MOVS r2, r3          ;move r2 to r2
    .L53:   ADDS r2, r2, r3      ;move r2 to r2
    .L54:   MOVS r2, r3          ;move r2 to r2
    .L55:   ADDS r2, r2, r3      ;move r2 to r2
    .L56:   MOVS r2, r3          ;move r2 to r2
    .L57:   ADDS r2, r2, r3      ;move r2 to r2
    .L58:   MOVS r2, r3          ;move r2 to r2
    .L59:   ADDS r2, r2, r3      ;move r2 to r2
    .L60:   MOVS r2, r3          ;move r2 to r2
    .L61:   ADDS r2, r2, r3      ;move r2 to r2
    .L62:   MOVS r2, r3          ;move r2 to r2
    .L63:   ADDS r2, r2, r3      ;move r2 to r2
    .L64:   MOVS r2, r3          ;move r2 to r2
    .L65:   ADDS r2, r2, r3      ;move r2 to r2
    .L66:   MOVS r2, r3          ;move r2 to r2
    .L67:   ADDS r2, r2, r3      ;move r2 to r2
    .L68:   MOVS r2, r3          ;move r2 to r2
    .L69:   ADDS r2, r2, r3      ;move r2 to r2
    .L70:   MOVS r2, r3          ;move r2 to r2
    .L71:   ADDS r2, r2, r3      ;move r2 to r2
    .L72:   MOVS r2, r3          ;move r2 to r2
    .L73:   ADDS r2, r2, r3      ;move r2 to r2
    .L74:   MOVS r2, r3          ;move r2 to r2
    .L75:   ADDS r2, r2, r3      ;move r2 to r2
    .L76:   LDR r0, =0x7F000000 ;set value in r0
    .L77:   LDR r1, =0x10000000 ;set value in r1
    .L78:   MOVS r2, #3           ;set value in r2
    .L79:   MOVS r3, #7           ;set value in r3
    .L80:   MOVS r4, #0           ;set value in r4 to 0
    .L81:   MOVS r5, #0           ;set value in r5 to 0
    .L82:   MOVS r6, #0           ;set value in r6 to 0
    .L83:   MOVS r9, #0           ;set value in r9 to 0

```

main.c startup_MN14Z4.s system_MN14Z4.c exp26_asm.s

Call Stack + Locals

Name	Location/Value	Type
exp26_asm	0x0000002C	void f()
main	0x00000000	void f()

Call Stack + Locals

Name	Location/Value	Type
exp26_asm	0x0000002C	void f()
main	0x00000000	void f()

Memory 1 Simulation t1: 0.00000150 sec L70 C1

Registers Disassembly

Register	Value
R0	0x00000000
R1	0x40048100
R2	0xFFFFFFFF
R3	0x00000001
R4	0xFFFFFFFF
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13(SP)	0x1FFE160
R14(LR)	0x00000266
R15(PC)	0x00000000
N	0
Z	1
C	1
V	0
T	1
ISR	0

```

    .L1:    ADDS r5, r2, r3      ;add r2 and r3 and set flags
    .L2:    MOVS r5, r2, r3      ;move r2 to r5
    .L3:    ADDS r6, r2, r3      ;add r2 and r3 and set flags
    .L4:    MOVS r6, r2, r3      ;move r2 to r6
    .L5:    ADDS r6, r2, r3      ;add r2 and r3 and set flags
    .L6:    MOVS r6, r2, r3      ;move r2 to r6
    .L7:    ADDS r6, r2, r3      ;add r2 and r3 and set flags
    .L8:    MOVS r6, r2, r3      ;move r2 to r6
    .L9:    ADDS r6, r2, r3      ;add r2 and r3 and set flags
    .L10:   MOVS r6, r2, r3      ;move r2 to r6
    .L11:   ADDS r6, r2, r3      ;add r2 and r3 and set flags
    .L12:   MOVS r6, r2, r3      ;move r2 to r6
    .L13:   ADDS r6, r2, r3      ;add r2 and r3 and set flags
    .L14:   MOVS r6, r2, r3      ;move r2 to r6
    .L15:   ADDS r6, r2, r3      ;add r2 and r3 and set flags
    .L16:   MOVS r6, r2, r3      ;move r2 to r6
    .L17:   ADDS r6, r2, r3      ;add r2 and r3 and set flags
    .L18:   MOVS r6, r2, r3      ;move r2 to r6
    .L19:   ADDS r6, r2, r3      ;add r2 and r3 and set flags
    .L20:   MOVS r6, r2, r3      ;move r2 to r6
    .L21:   ADDS r6, r2, r3      ;add r2 and r3 and set flags
    .L22:   MOVS r6, r2, r3      ;move r2 to r6
    .L23:   ADDS r6, r2, r3      ;add r2 and r3 and set flags
    .L24:   MOVS r6, r2, r3      ;move r2 to r6
    .L25:   ADDS r6, r2, r3      ;add r2 and r3 and set flags
    .L26:   MOVS r6, r2, r3      ;move r2 to r6
    .L27:   ADDS r6, r2, r3      ;add r2 and r3 and set flags
    .L28:   MOVS r6, r2, r3      ;move r2 to r6
    .L29:   ADDS r6, r2, r3      ;add r2 and r3 and set flags
    .L30:   MOVS r6, r2, r3      ;move r2 to r6
    .L31:   ADDS r6, r2, r3      ;add r2 and r3 and set flags
    .L32:   MOVS r6, r2, r3      ;move r2 to r6
    .L33:   ADDS r6, r2, r3      ;add r2 and r3 and set flags
    .L34:   MOVS r6, r2, r3      ;move r2 to r6
    .L35:   ADDS r6, r2, r3      ;add r2 and r3 and set flags
    .L36:   MOVS r6, r2, r3      ;move r2 to r6
    .L37:   ADDS r6, r2, r3      ;add r2 and r3 and set flags
    .L38:   MOVS r6, r2, r3      ;move r2 to r6
    .L39:   ADDS r6, r2, r3      ;add r2 and r3 and set flags
    .L40:   MOVS r6, r2, r3      ;move r2 to r6
    .L41:   ADDS r6, r2, r3      ;add r2 and r3 and set flags
    .L42:   MOVS r6, r2, r3      ;move r2 to r6
    .L43:   ADDS r6, r2, r3      ;add r2 and r3 and set flags
    .L44:   MOVS r6, r2, r3      ;move r2 to r6
    .L45:   ADDS r6, r2, r3      ;add r2 and r3 and set flags
    .L46:   MOVS r6, r2, r3      ;move r2 to r6
    .L47:   ADDS r6, r2, r3      ;add r2 and r3 and set flags
    .L48:   MOVS r6, r2, r3      ;move r2 to r6
    .L49:   ADDS r6, r2, r3      ;add r2 and r3 and set flags
    .L50:   MOVS r6, r2, r3      ;move r2 to r6
    .L51:   ADDS r6, r2, r3      ;add r2 and r3 and set flags
    .L52:   MOVS r6, r2, r3      ;move r2 to r6
    .L53:   ADDS r6, r2, r3      ;add r2 and r3 and set flags
    .L54:   MOVS r6, r2, r3      ;move r2 to r6
    .L55:   ADDS r6, r2, r3      ;add r2 and r3 and set flags
    .L56:   MOVS r6, r2, r3      ;move r2 to r6
    .L57:   ADDS r6, r2, r3      ;add r2 and r3 and set flags
    .L58:   MOVS r6, r2, r3      ;move r2 to r6
    .L59:   ADDS r6, r2, r3      ;add r2 and r3 and set flags
    .L60:   MOVS r6, r2, r3      ;move r2 to r6
    .L61:   ADDS r6, r2, r3      ;add r2 and r3 and set flags
    .L62:   MOVS r6, r2, r3      ;move r2 to r6
    .L63:   ADDS r6, r2, r3      ;add r2 and r3 and set flags
    .L64:   MOVS r6, r2, r3      ;move r2 to r6
    .L65:   ADDS r6, r2, r3      ;add r2 and r3 and set flags
    .L66:   MOVS r6, r2, r3      ;move r2 to r6
    .L67:   ADDS r6, r2, r3      ;add r2 and r3 and set flags
    .L68:   MOVS r6, r2, r3      ;move r2 to r6
    .L69:   ADDS r6, r2, r3      ;add r2 and r3 and set flags
    .L70:   MOVS r6, r2, r3      ;move r2 to r6
    .L71:   ADDS r6, r2, r3      ;add r2 and r3 and set flags
    .L72:   MOVS r6, r2, r3      ;move r2 to r6
    .L73:   ADDS r6, r2, r3      ;add r2 and r3 and set flags
    .L74:   MOVS r6, r2, r3      ;move r2 to r6
    .L75:   ADDS r6, r2, r3      ;add r2 and r3 and set flags
    .L76:   LDR r0, =0x7F000000 ;set value in r0
    .L77:   LDR r1, =0x10000000 ;set value in r1
    .L78:   MOVS r2, #3           ;set value in r2
    .L79:   MOVS r3, #7           ;set value in r3
    .L80:   MOVS r4, #0           ;set value in r4 to 0
    .L81:   MOVS r5, #0           ;set value in r5 to 0
    .L82:   MOVS r6, #0           ;set value in r6 to 0
    .L83:   MOVS r9, #0           ;set value in r9 to 0

```

main.c startup_MN14Z4.s system_MN14Z4.c exp26_asm.s

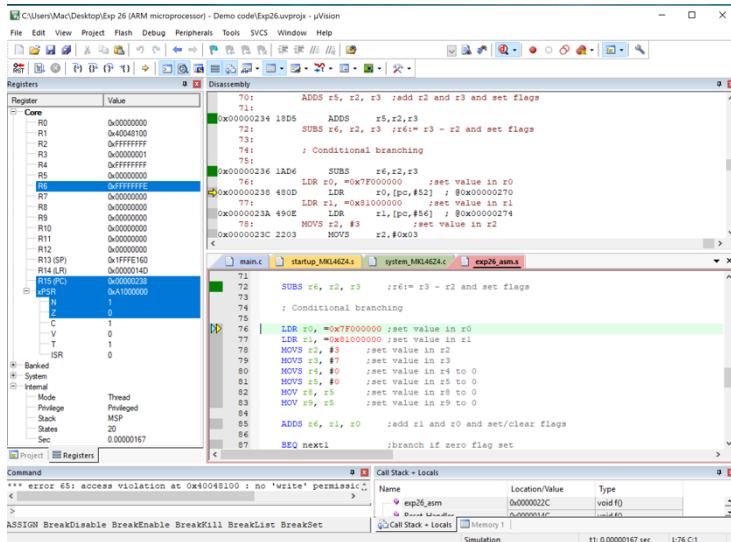
Call Stack + Locals

Name	Location/Value	Type
exp26_asm	0x0000002C	void f()
main	0x00000000	void f()

Call Stack + Locals

Name	Location/Value	Type
exp26_asm	0x0000002C	void f()
main	0x00000000	void f()

Memory 1 Simulation t1: 0.00000158 sec L72 C1



Comment/Explanation:

(1) R2 initial

MOVS r2, #0 ;clear r2 and set/clear flags

R2	R3	R4	R5	R6	N	Z	C	V
0	0	0	0	0	0	1	0	0

MOVS r3, #1 ;set value in r3 to 1

R2	R3	R4	R5	R6	N	Z	C	V
0	1	0	0	0	0	0	0	0

MOV r4, r2 ;move r2 to r4

R2	R3	R4	R5	R6	N	Z	C	V
0	1	0	0	0	0	0	0	0

MOVS r4, r2 ;same as previous instruction but set/clear flags

R2	R3	R4	R5	R6	N	Z	C	V
0	1	0	0	0	0	1	0	0

ADDS r5, r2, r3 ;add r2 and r3 and set flags

R2	R3	R4	R5	R6	N	Z	C	V
0	1	0	1	0	0	0	0	0

SUBS r6, r2, r3 ;r6:= r3 - r2 and set flags

R2	R3	R4	R5	R6	N	Z	C	V
0	1	0	1	FFFFFF	1	0	0	0

R4	R5	R6	NF
0	0	1	1

(2) R2 to FFFFFFFF

MOVS r3, #1 ;set value in r3 to 1

R2	R3	R4	R5	R6	N	Z	C	V
FFFFFF	1	0	0	0	0	0	0	0

MOV r4, r2 ;move r2 to r4

R2	R3	R4	R5	R6	N	Z	C	V
FFFFFFFF	1	FFFFFFFF	0	0	0	0	0	0

MOVS r4, r2 ;same as previous instruction but set/clear flags

R2	R3	R4	R5	R6	N	Z	C	V
FFFFFFFF	1	FFFFFFFF	0	0	1	0	0	0

ADDS r5, r2, r3 ;add r2 and r3 and set flags

R2	R3	R4	R5	R6	N	Z	C	V
FFFFFFFF	1	FFFFFFFF	0	0	0	1	1	0

SUBS r6, r2, r3 ;r6:= r3 - r2 and set flags

R2	R3	R4	R5	R6	N	Z	C	V
FFFFFFFF	1	FFFFFFFF	0	FFFFFFE	1	0	1	0

17. Answer to Q14 [4 marks]

Answer:

BEQ, BCS, BPL, BVC

Explanation:

BEQ next1 ;branch if zero flag set

In the last op, the zero flag is 1 so branch;

BCS next3 ;branch if carry flag set

In the last op, the carry flag is 1 so branch;

BMI next5 ;branch if negative flag set

In the last op, the negative flag is 0 so not branch;

BPL next6 ;branch if negative flag clear

In the last op, the negative flag is 0 so branch;

BVC last ;branch if overflow flag clear

In the last op, the overflow flag is 0 so branch

18. Answer to Q15 [4 marks]

Answer:**Yes****Explanation:****The excepted process:****In the first term:**

```
LDR r0, =0x7F000000 ;set value in r0
```

```
LDR r1, =0x81000000 ;set value in r1
```

```
MOVS r2, #3 ;set value in r2
```

```
MOVS r3, #7 ;set value in r3
```

```
MOVS r4, #0 ;set value in r4 to 0
```

```
MOVS r5, #0 ;set value in r5 to 0
```

```
MOV r8, r5 ;set value in r8 to 0
```

```
MOV r9, r5 ;set value in r9 to 0
```

R0	R1	R2	R3	R4	R5	R6	R7	R8	R9
0x7F000000	0x81000000	0x00000003	0x00000007	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

ADDS r6, r1, r0 ;add r1 and r0 and set/clear flags

R6 = 0x00000000 → Z=1 C=1 other flags is 0

BNE next2 ;branch if zero flag clear → Omit because the zero flag not clear

BCS next3 ;branch because carry flag set

BCC next4 ;Not branch because carry flag is not clear

ADD r5, r3 ;add if carry flag set

BMI next5 ;not branch because negative flag is not set

ADD r8, r2 ;add if negative flag clear

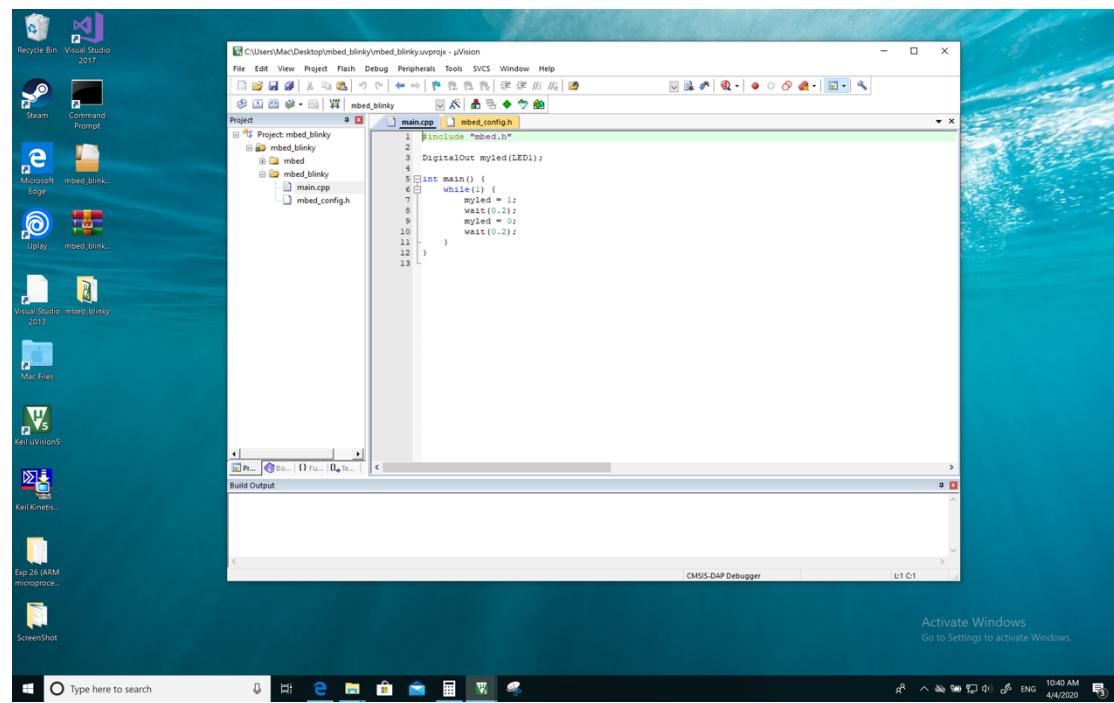
BPL next6 ;branch because negative flag clear

```
BVS next7           ;not branch because overflow flag is not set
ADD r9, r2          ;add if overflow flag clear
BVC last            ;branch because overflow flag is clear
```

So, the expected is the same as the experiment which is BEQ next1, BCS next3, BMI next5, BPL next6, BVC last is executed.

19. Screenshot of the result of Section 12 part IV [2 marks]

Screenshot (PrintScreen or Screenshot):



Screenshot (Snipping tool, Preview or equivalent):

The screenshot shows the µVision IDE interface with the following details:

- Title Bar:** C:\Users\Mac\Desktop\mbed_blinky\mbed_blinky.uvprojx - µVision
- Menu Bar:** File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, Help
- Toolbar:** Includes icons for Open, Save, Build, Run, and others.
- Project Explorer:** Shows the project structure under "mbed_blinky".
- Code Editor:** Displays the main.cpp file with the following code:

```
1 #include "mbed.h"
2
3 DigitalOut myled(LED1);
4
5 int main() {
6     while(1) {
7         myled = 1;
8         wait(0.2);
9         myled = 0;
10        wait(0.2);
11    }
12 }
```
- Build Output:** Shows the following error message:

```
Load "C:\Users\Mac\Desktop\mbed_blinky\BUILD\mbed_blinky.axf"
^
*** error 56: cannot open file
Error: Flash Download failed - Could not load file 'C:\Users\Mac\Desktop\mbed_blinky\BUILD\mbed_blinky.axf'
Flash Load finished at 10:41:20
```
- Status Bar:** CMSIS-DAP Debugger, L:6 C:4