Experiment 26

Introduction to the ARM Microprocessor

Submission Template of Experimental Results

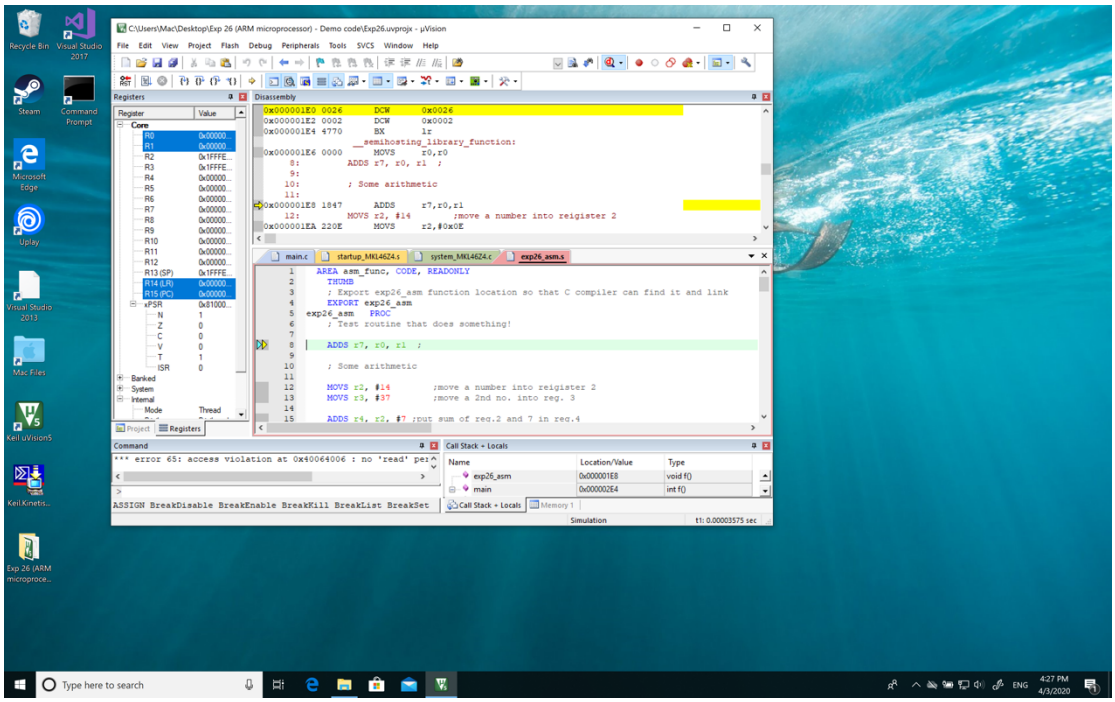
Important: Marking of all coursework is anonymous. Do not include your name, student ID number, group number, email or any other personal information in your report or in the name of the file submitted via VITAL. A penalty will be applied to submissions that do not meet this requirement.

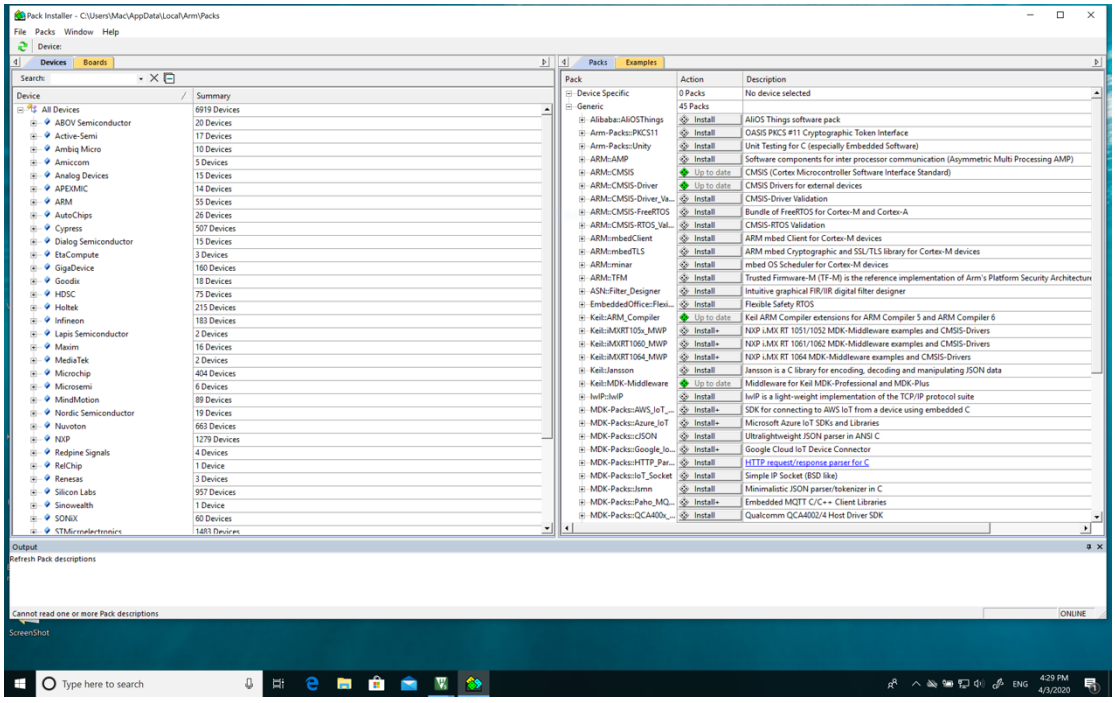
Experimental Results and Comments (Total 90 marks):

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

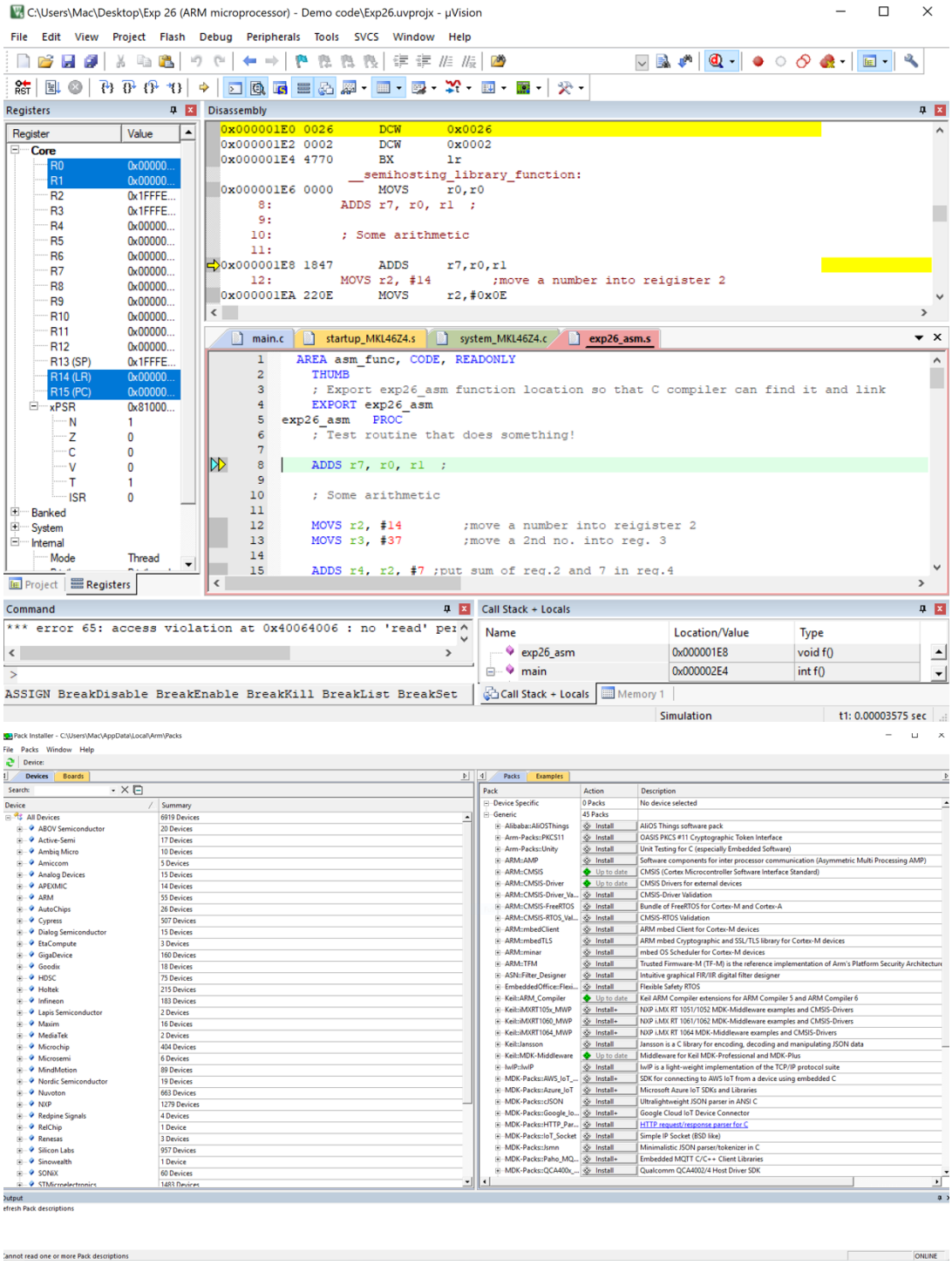
1. Screenshot of the result of Section 4 [2 marks]

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 Vaule

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 Vaule

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 *0xFFFFFFA6*)

R0	*0x00000063*
R1	*0x0000005A*
R2	*0x00000011*
R3	*0x0000000A*
R4	*0x0000006B*
R5	*0x0000006E*
R6	*0x0000006E*
R7	*0x00000000*
R8	*0x00000000*
R9	*0x00000000*
R10	*0x00000000*
R11	*0x00000000*
R12	*0x00000000*
R13	*0x1FFFE160*
R14	*0x00000014D*
R15	*0x000001FC*
xPSR	*0x01000000*

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 = *0x0000006B*

ADDS r5, r2, r3 ;put sum of r2 and r3 in r5 = *0x0000006E*

ADDS r6, r3, r2 ;put sum of r3 and r2 in r6 = *0x0000006E*

SUBS r0, r2, #1 ;subtract 1 from r2: *0x00000064* **- 1 =** *0x00000063*

SUBS r1, r2, r3 ;subtract r3 from r2 *0x00000064 - 0x0000000A = 0x0000005A*

SUBS r2, r3, r2 ;reverse the order r2 = *0x0000000A - 0x00000064 = 0xFFFFFFA6*

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	*0xFFFFFFFF*
R1	*0x00000000*
R2	*0x00000011*
R3	*0x00000000*
R4	*0x00000007*
R5	*0x00000000*
R6	*0x00000000*
R7	*0x00000000*
R8	*0x00000000*
R9	*0x00000000*
R10	*0x00000000*
R11	*0x00000000*
R12	*0x00000000*
R13	*0x1FFFE160*
R14	*0x00000014D*
R15	*0x000001FC*
xPSR	*0x21000000*

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:

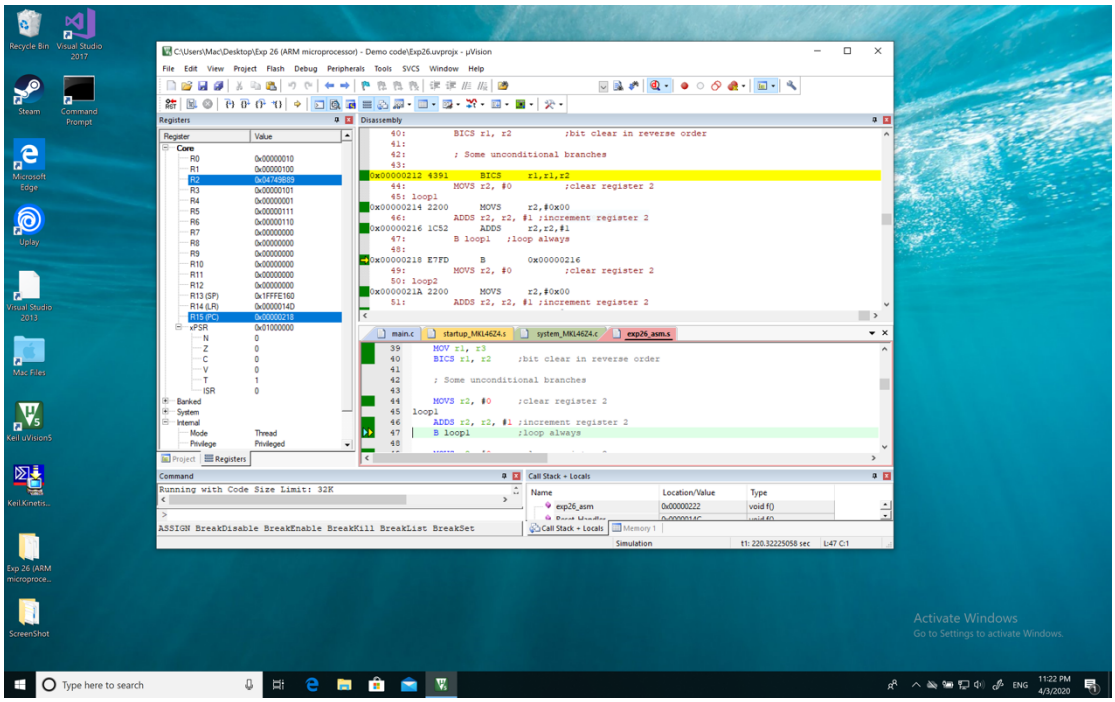
00:19.96
0x04785BDE

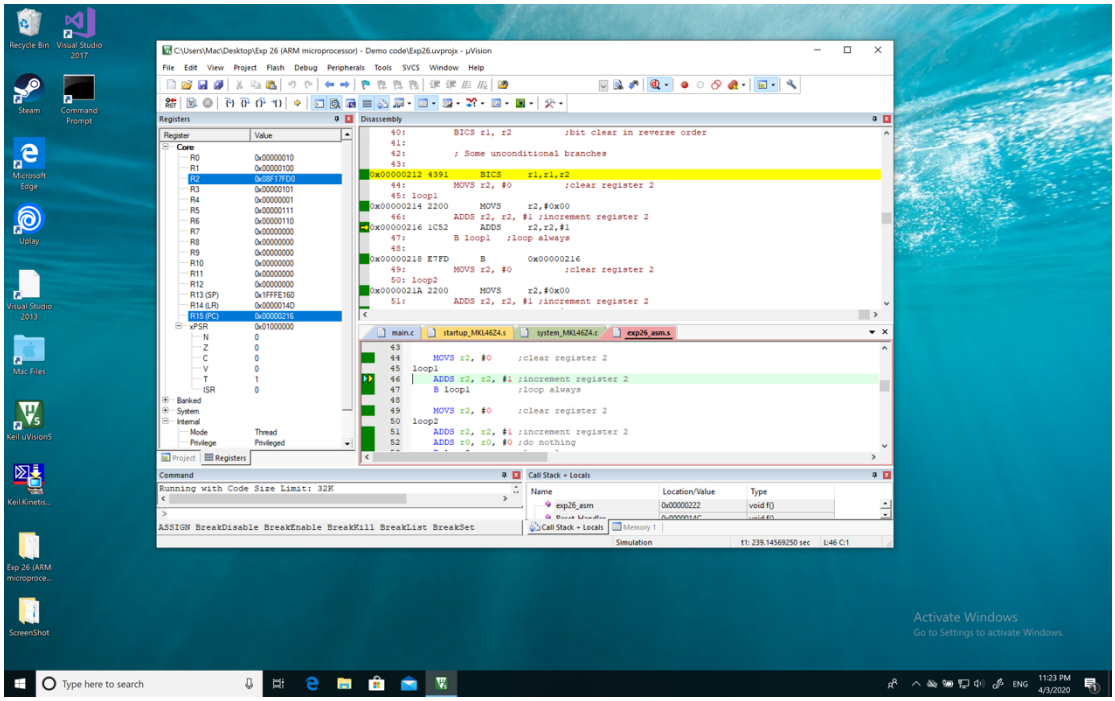
00:39.86
0x08F473E8

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

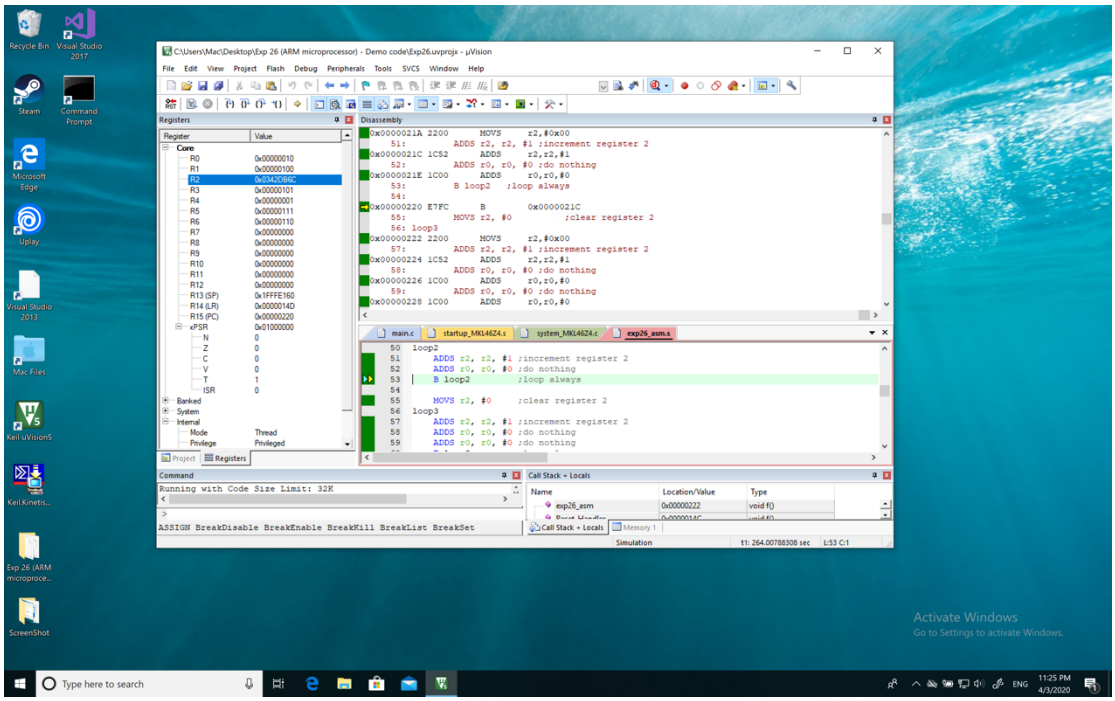
Screenshot (PrintScreen or Screenshot):

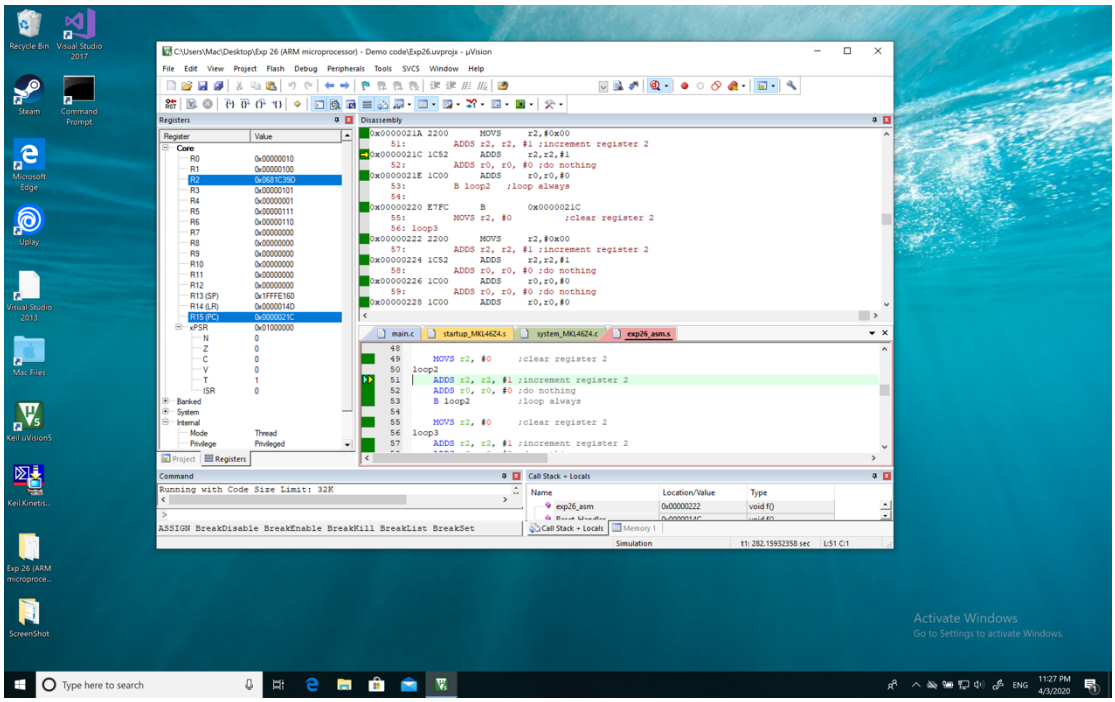
Loop1:



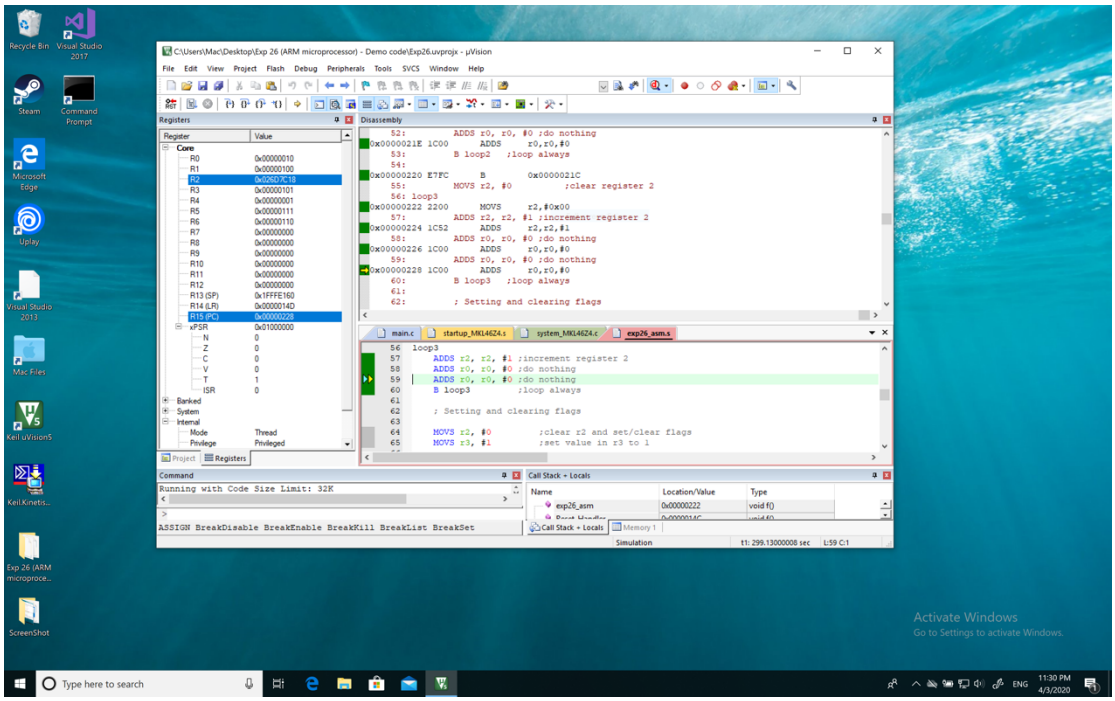


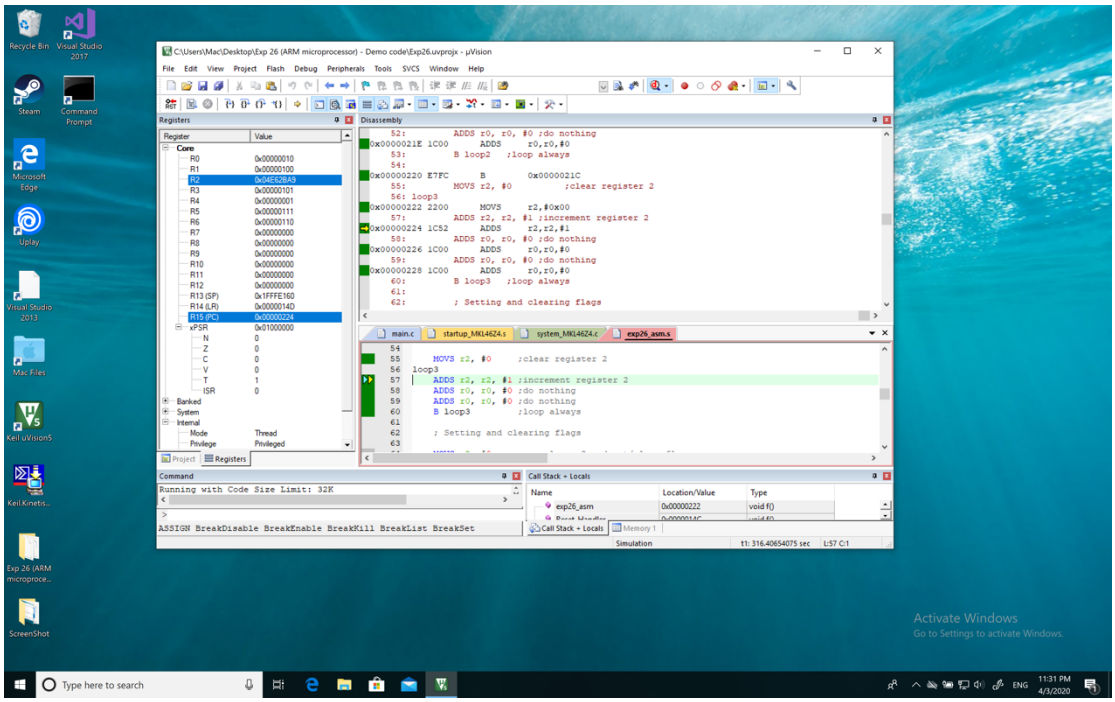
Loop2:


C:\Users\Mac\Desktop\Exp 26 (ARM microprocessor) - Demo code\Exp26.uvprojx - µVision
Running with Code Size Limit: 32K
Activate Windows
Go to Settings to activate Windows.
Type here to search
11:25 PM
4/3/2020

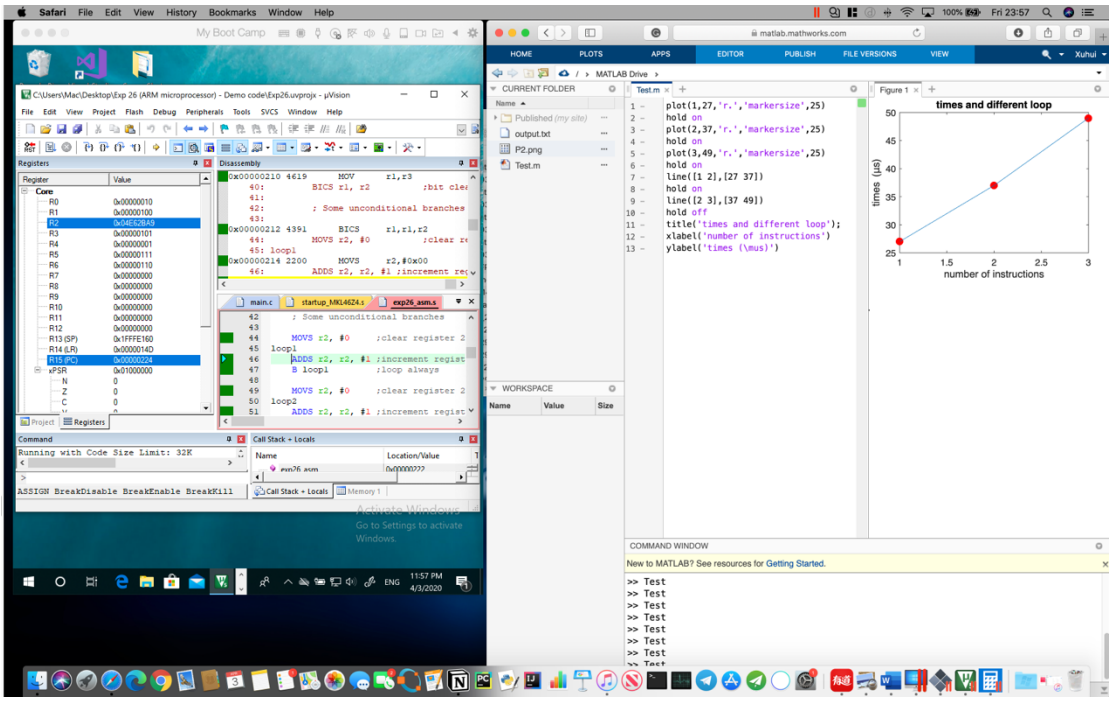

C:\Users\Mac\Desktop\Exp 26 (ARM microprocessor) - Demo code\Exp26.uvprojx - µVision
Running with Code Size Limit: 32K
Activate Windows
Go to Settings to activate Windows.
Type here to search
11:27 PM
4/3/2020

Loop3:



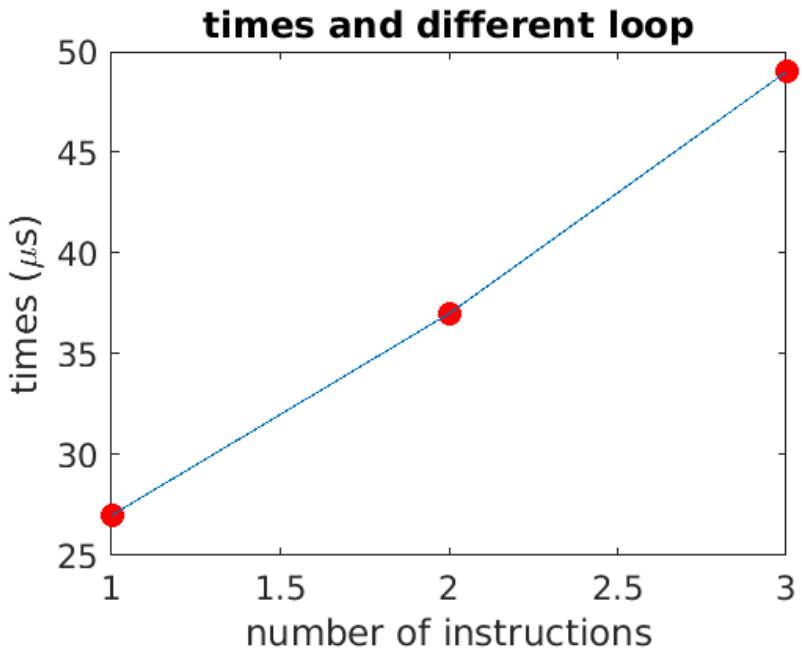


MATLAB Work



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
0x04749B89

00:39.86
0x08F17FD0

3764290 times/s

0.00000027 s/times

In loop2:

00:20.03
0x0342DB6C

00:40.04
0x0681C39D

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:

$$\mathbf{11}\boldsymbol{\mu s}$$

Explanation:

In loop1 and loop2, there are added one instruction and it led to add almost 10$\boldsymbol{\mu s}$, in loop2 and loop3, there are added one instruction and it led to add almost 12$\boldsymbol{\mu s}$. 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:

$$\mathbf{15.6/11\ cycle} \sim \mathbf{2}\ \boldsymbol{cycles}$$

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 11μs, the total time in the loop1 is 27 μs, so the branch need 16 μs; the total time in the loop2 is 37 μs, 37 μs - 2*add instructions = 37 μs - 11 μs * 2 = 15 μs; the total time in the loop3 is 49 μs, 49 μs - 3*add instructions = 49 μs - 11 μs * 3 = 16 μs;

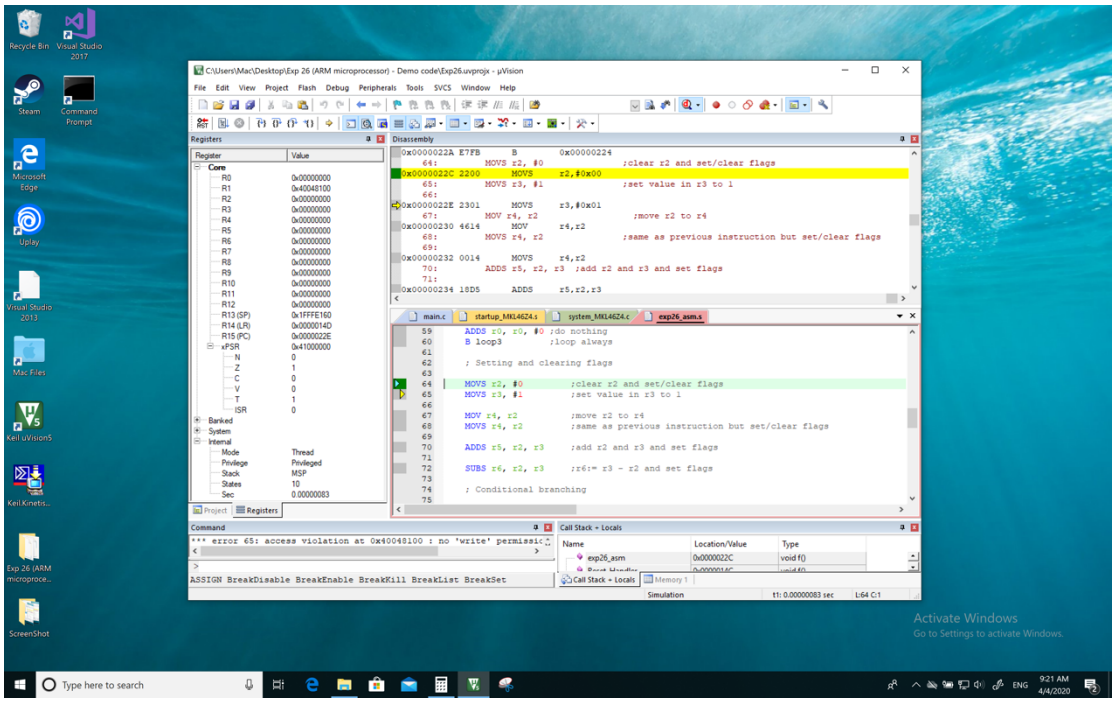
So, taking into account the error, the average value is 15.6 μs,

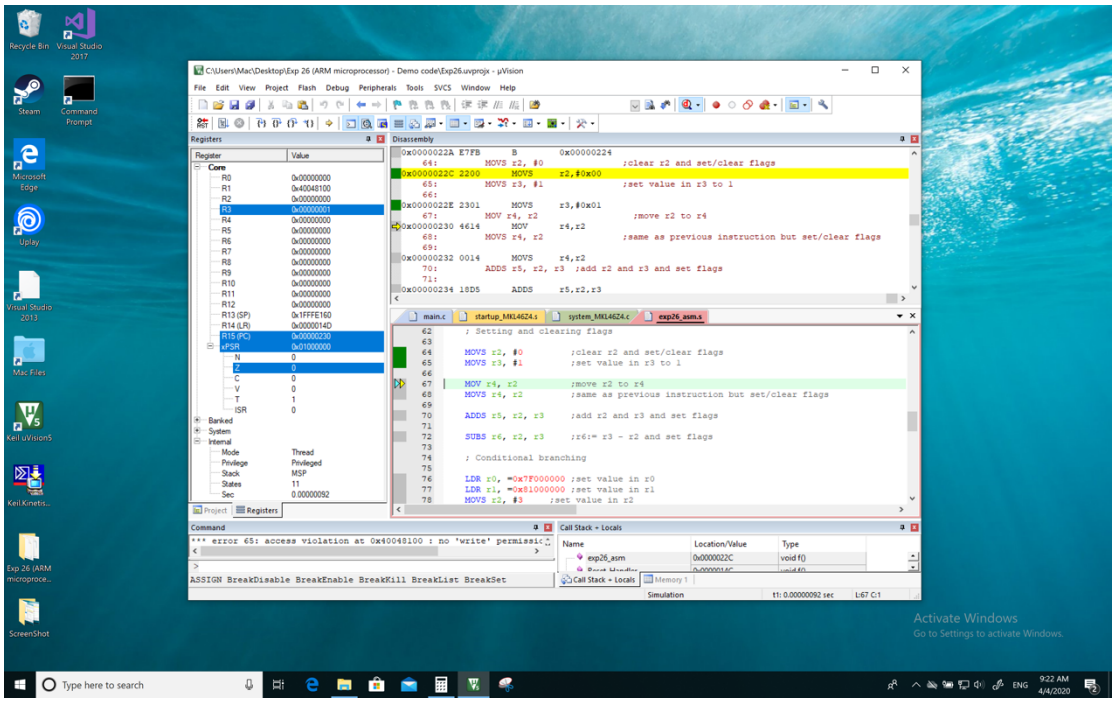
So the clock cycle is 15.6/11

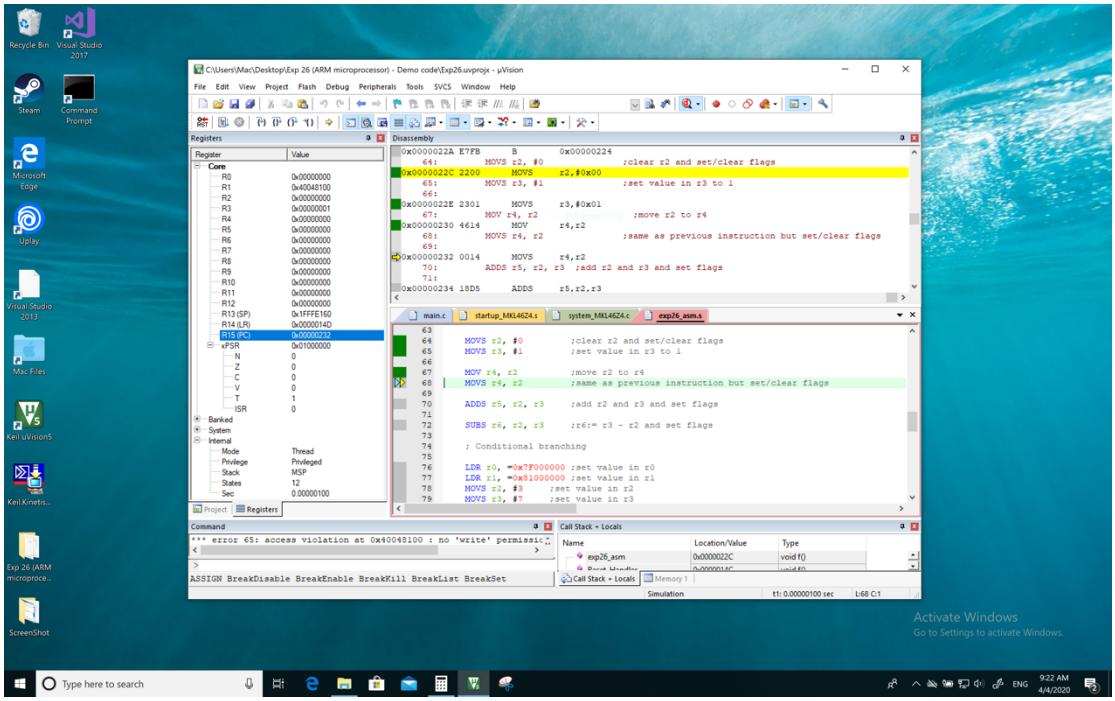
16. Screenshot of the result of Section 9 [4 marks]

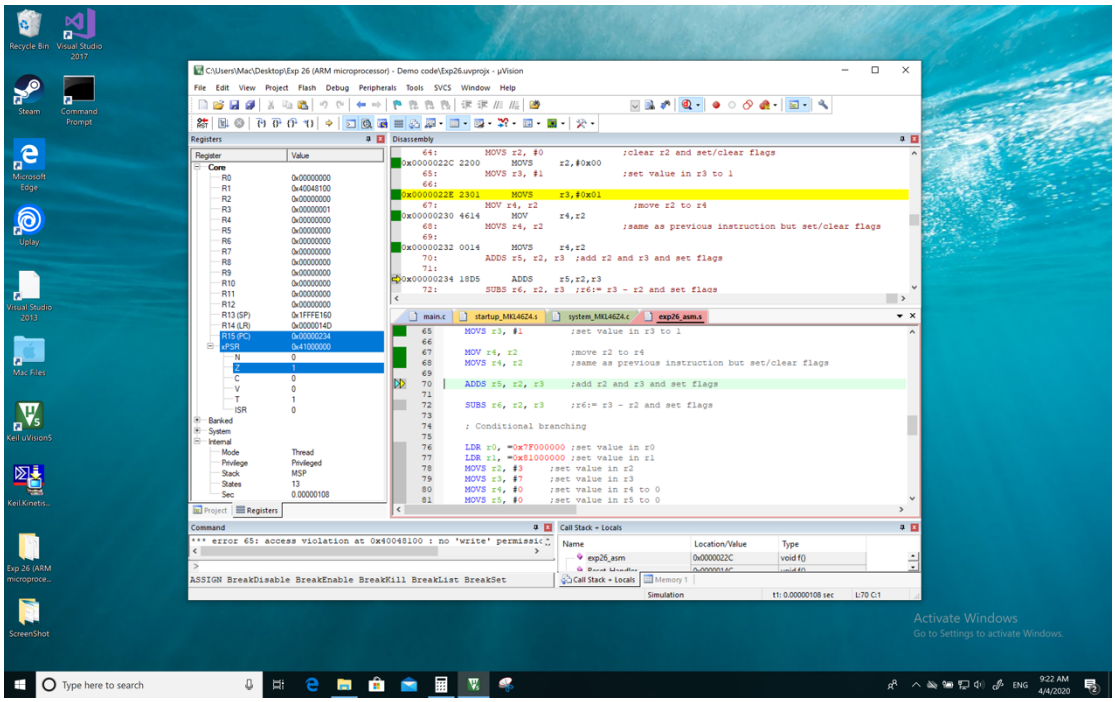
Screenshot (PrintScreen or Screenshot):

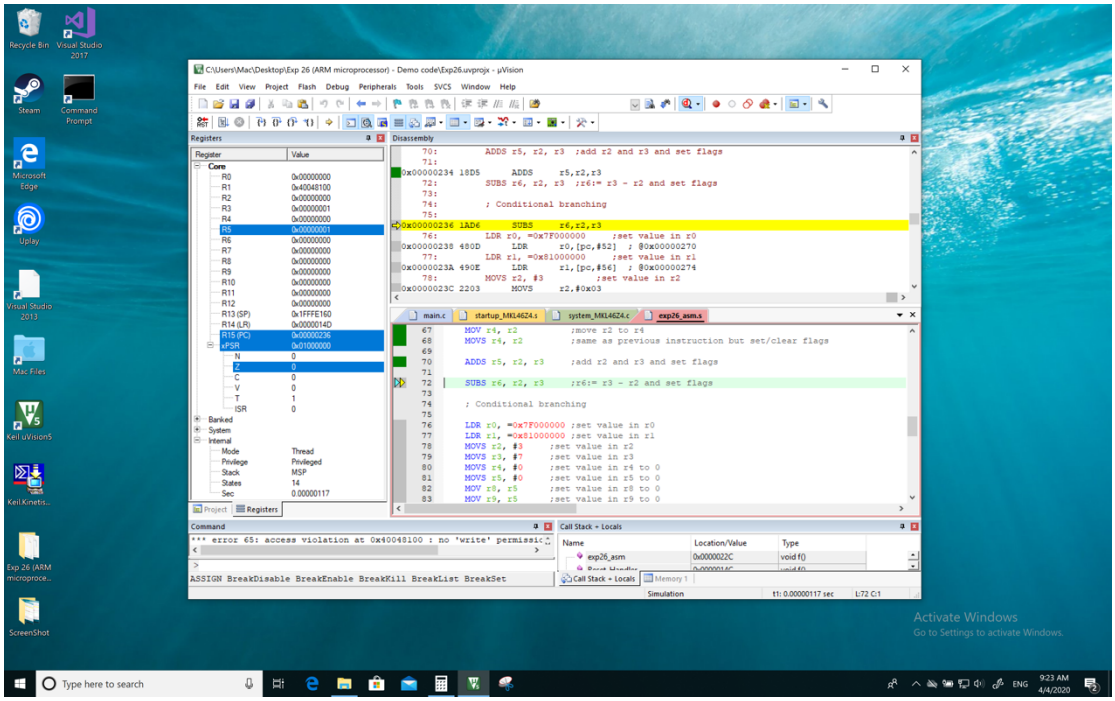
(1) R2 initial

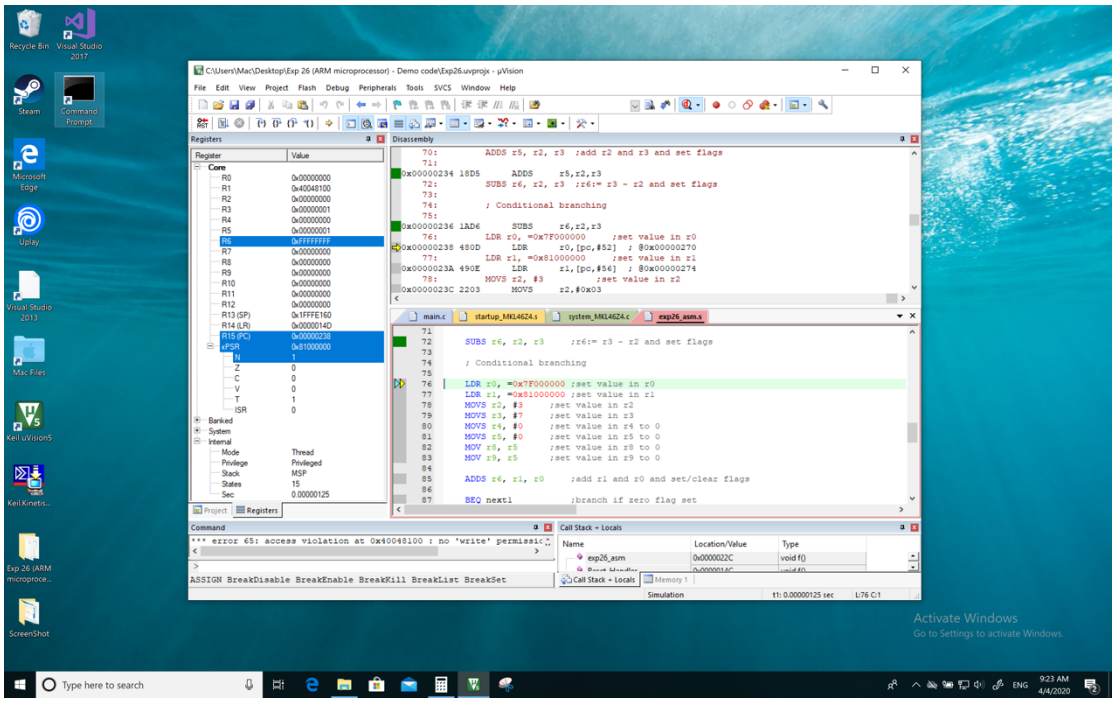




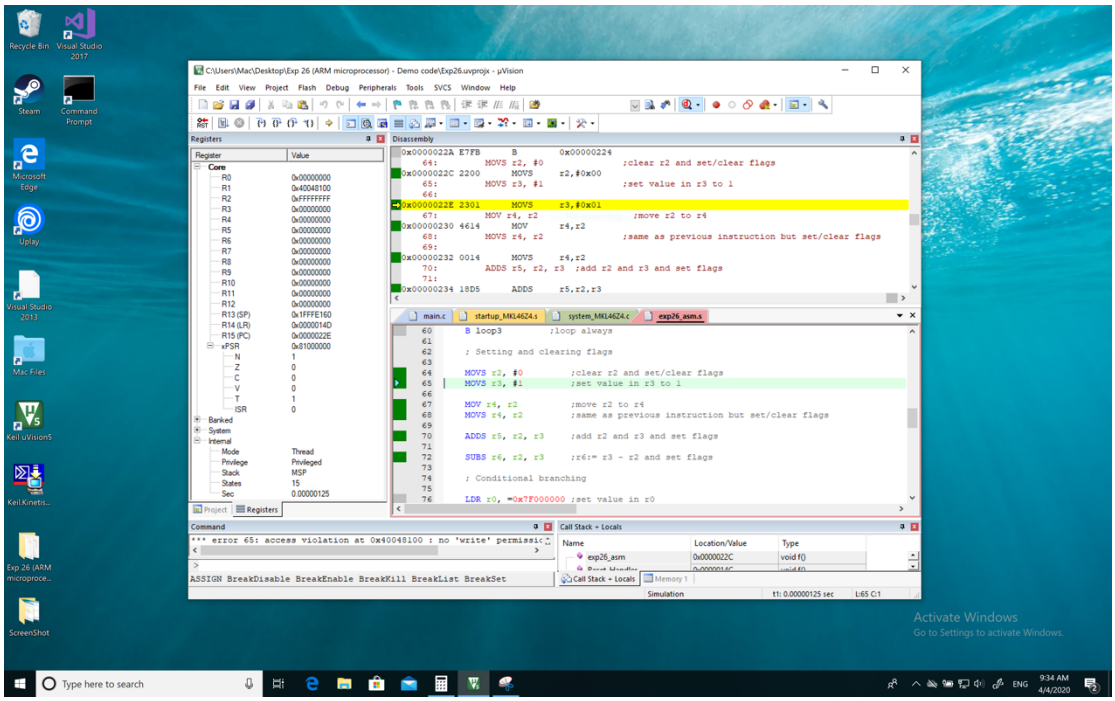

C:\Users\Mac\Desktop\Exp 26 (ARM microprocessor) - Demo code\Exp26.uvprojx - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Disassembly
Command
*** error 65: access violation at 0x40048100 : no 'write' permissic
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet
Call Stack + Locals
Activate Windows
Type here to search

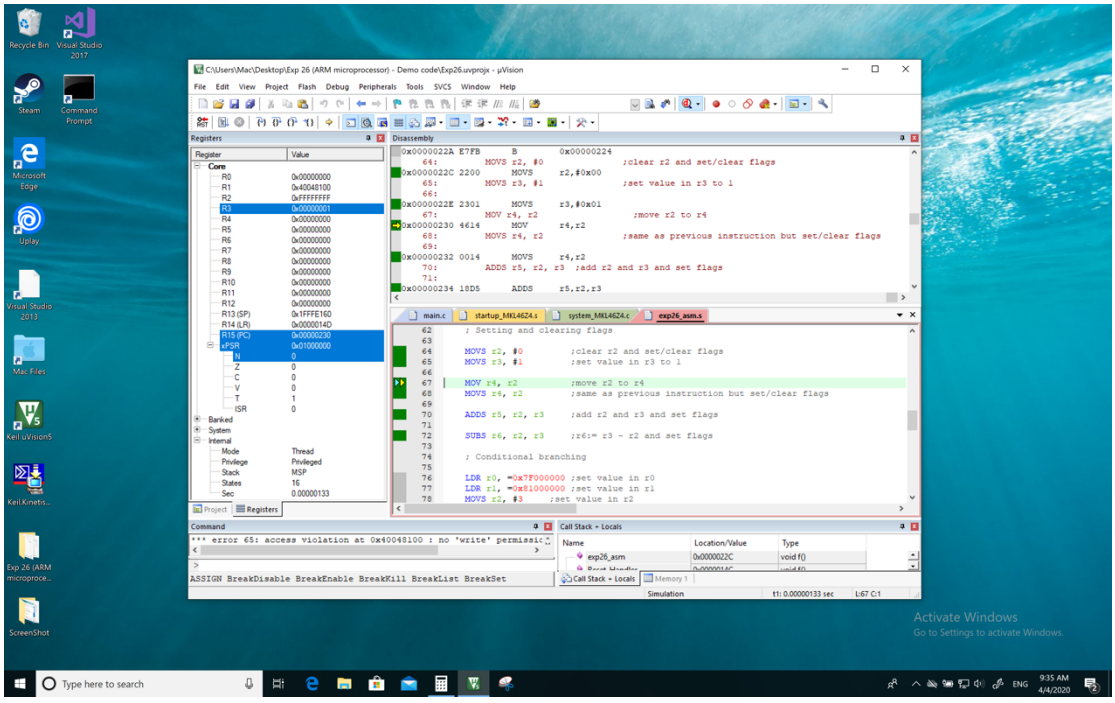

C:\Users\Mac\Desktop\Exp 26 (ARM microprocessor) - Demo code\Exp26.uvprojx - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Disassembly
Command
*** error 65: access violation at 0x40048100 : no 'write' permissic
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet
Call Stack + Locals
Activate Windows
Type here to search

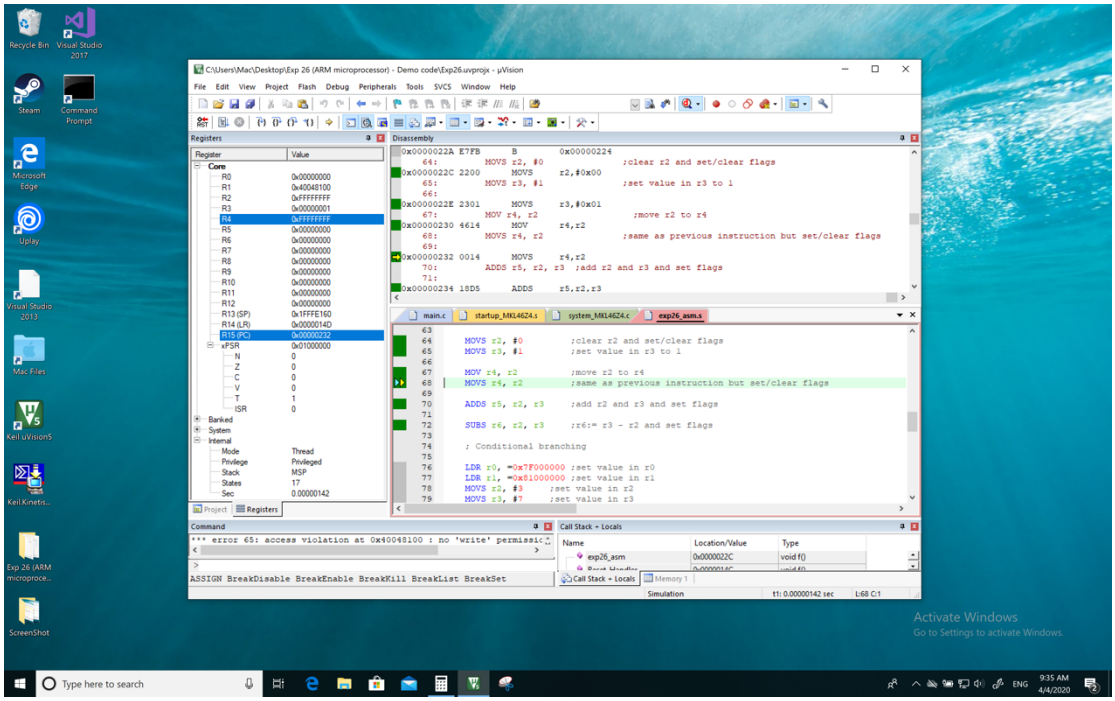


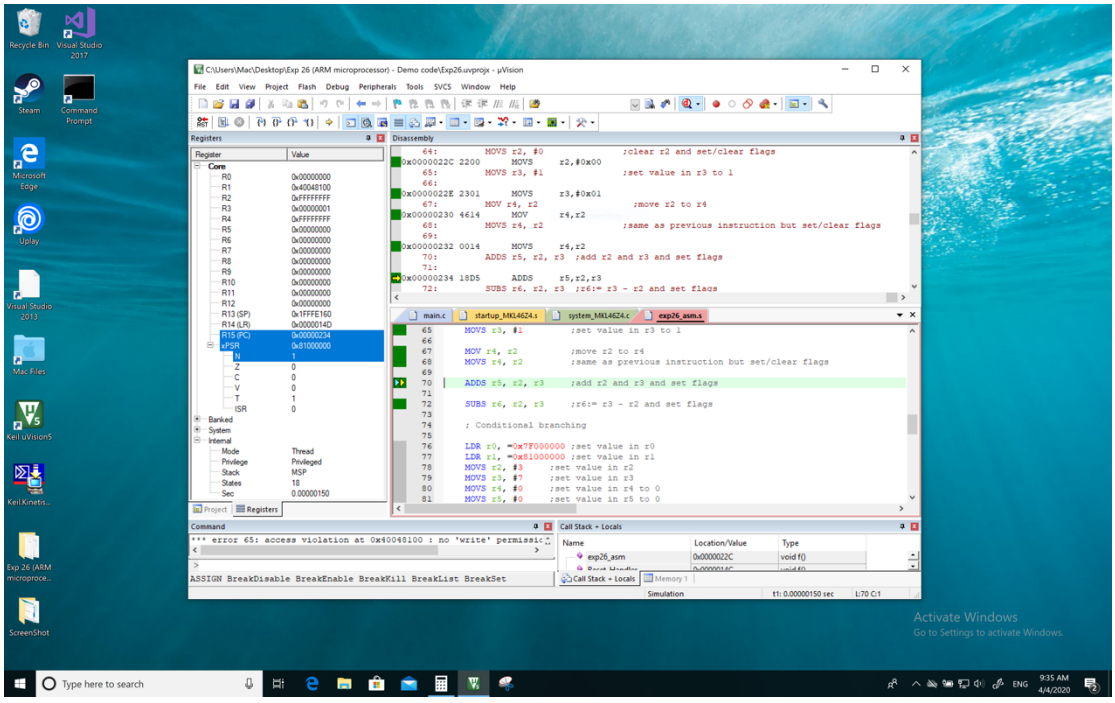


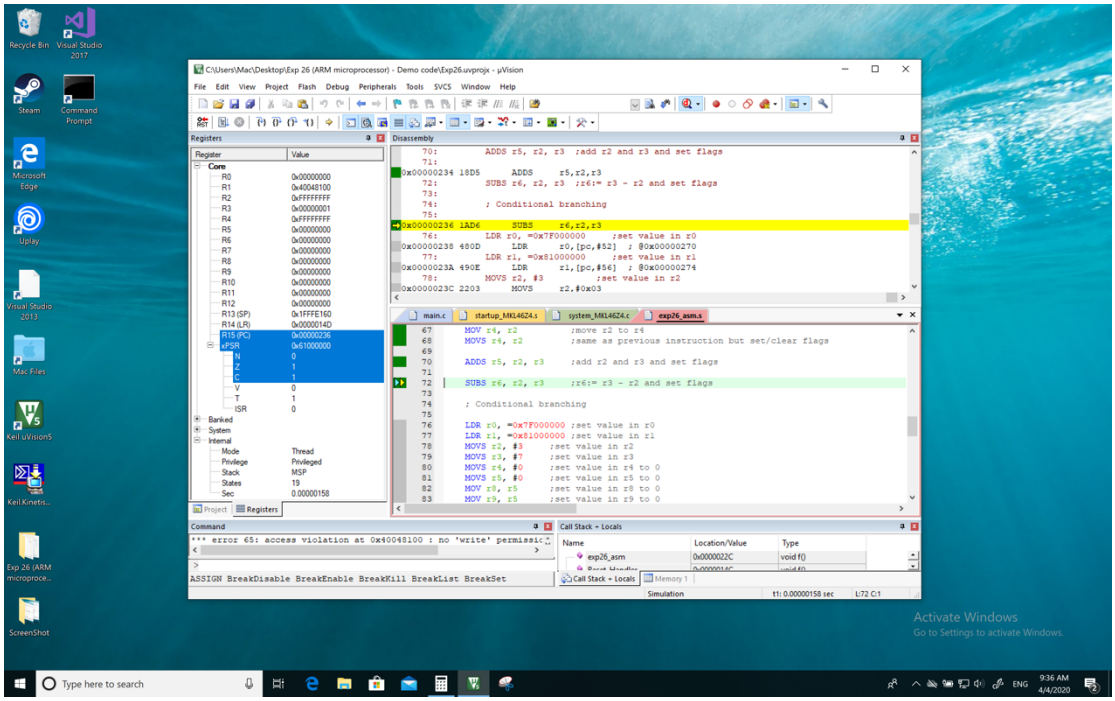
(2) R2 to FFFFFFFF

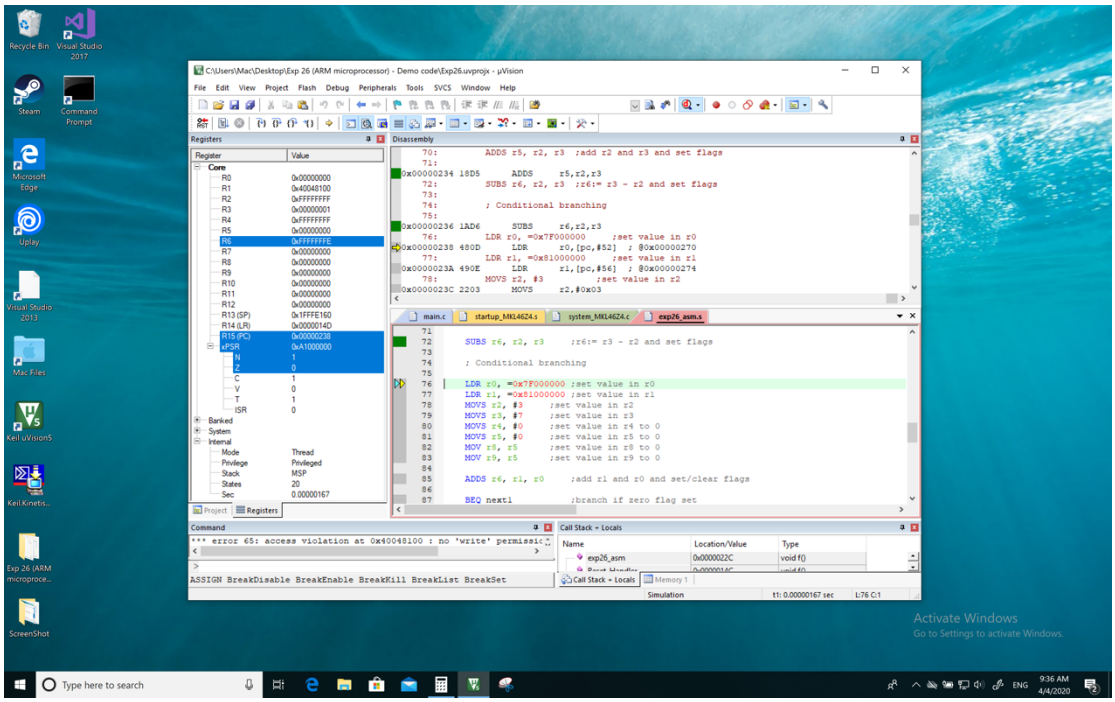

C:\Users\Mac\Desktop\Exp 26 (ARM microprocessor) - Demo code\Exp26.uvprojx - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Disassembly
Command
Call Stack + Locals
*** error 65: access violation at 0x40048100 : no 'write' permissic
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet
Activate Windows
Go to Settings to activate Windows.
Type here to search


C:\Users\Mac\Desktop\Exp 26 (ARM microprocessor) - Demo code\Exp26.uvprojx - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Disassembly
Command
Call Stack + Locals
*** error 65: access violation at 0x40048100 : no 'write' permissic
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet
Activate Windows
Go to Settings to activate Windows.
Type here to search

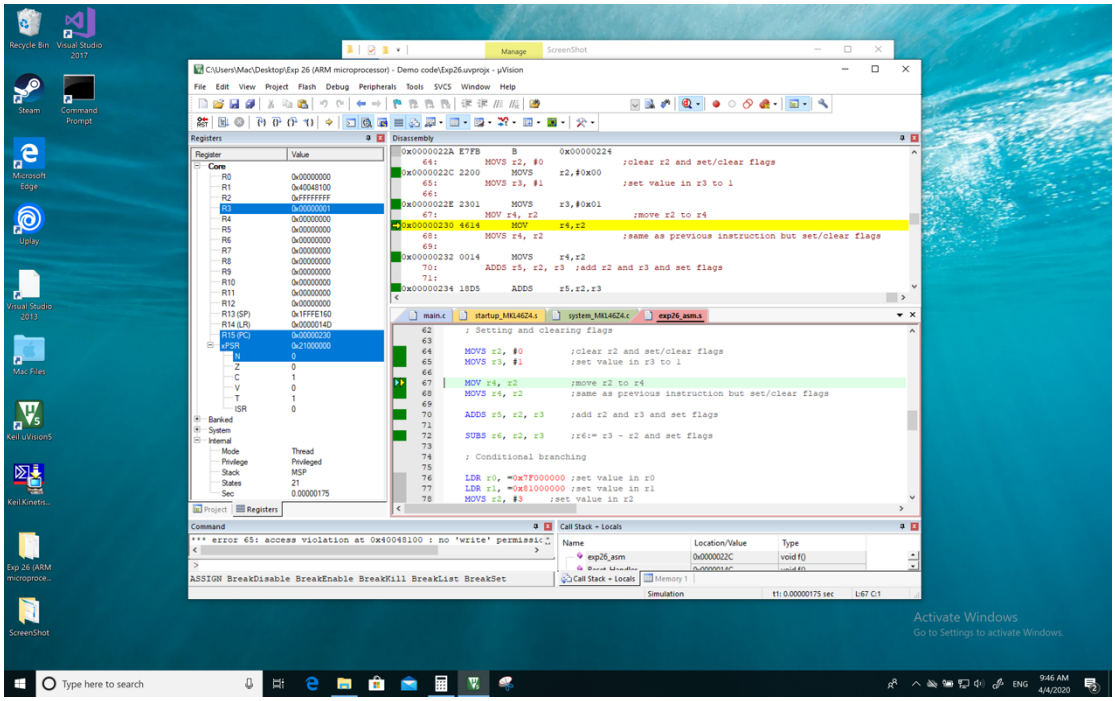

C:\Users\Mac\Desktop\Exp 26 (ARM microprocessor) - Demo code\Exp26.uvprojx - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Disassembly
Command
Call Stack + Locals
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet
Activate Windows
Go to Settings to activate Windows.
Type here to search

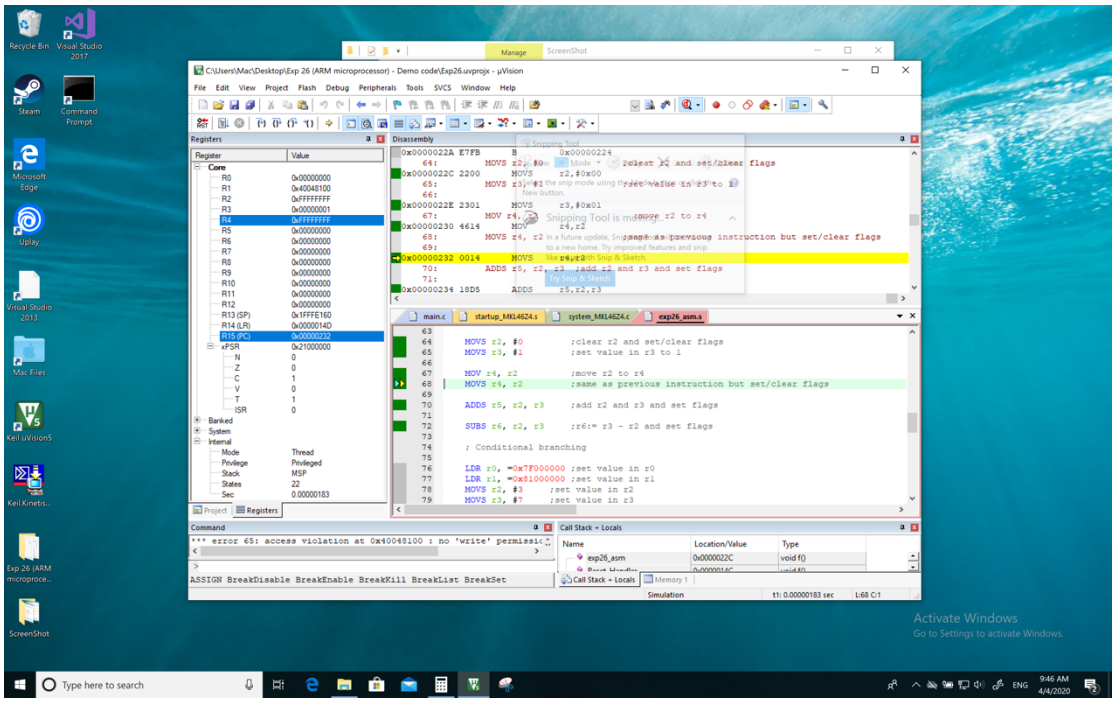

C:\Users\Mac\Desktop\Exp 26 (ARM microprocessor) - Demo code\Exp26.uvprojx - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Disassembly
Command
Call Stack + Locals
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet
Activate Windows
Go to Settings to activate Windows.
Type here to search

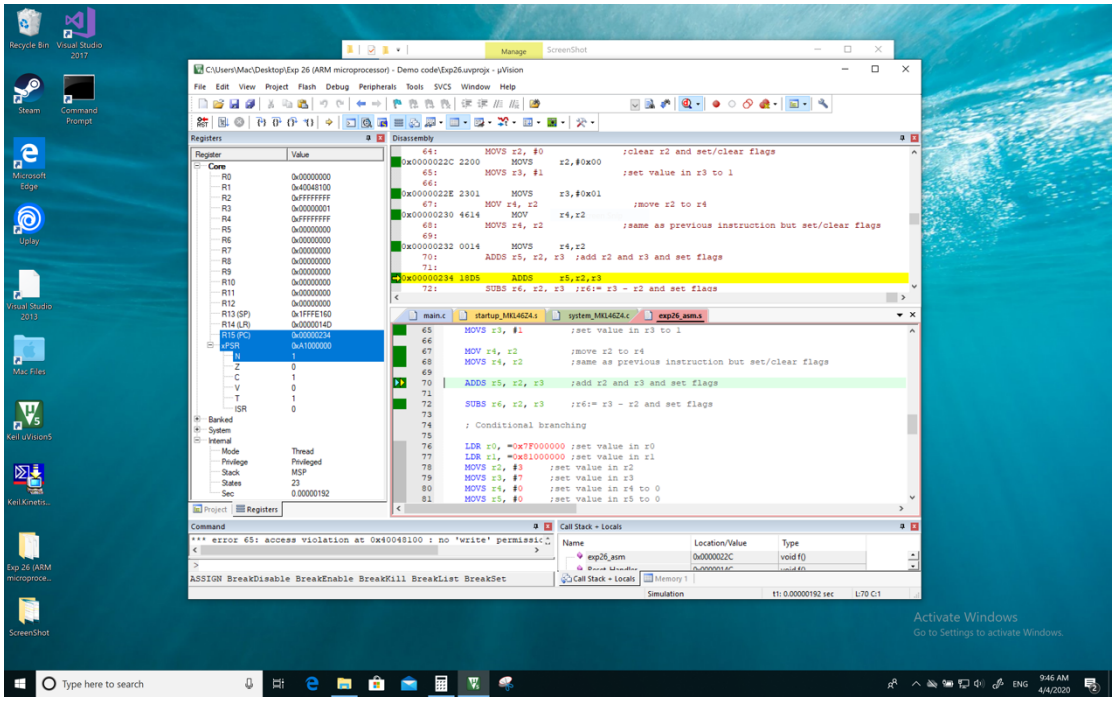


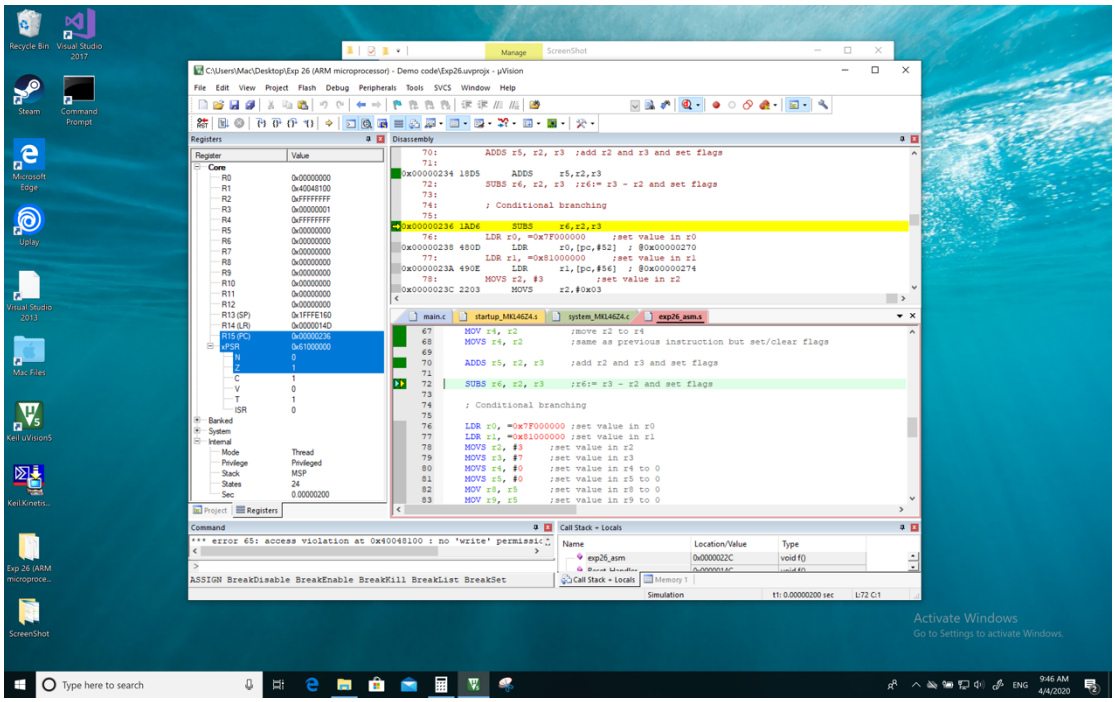


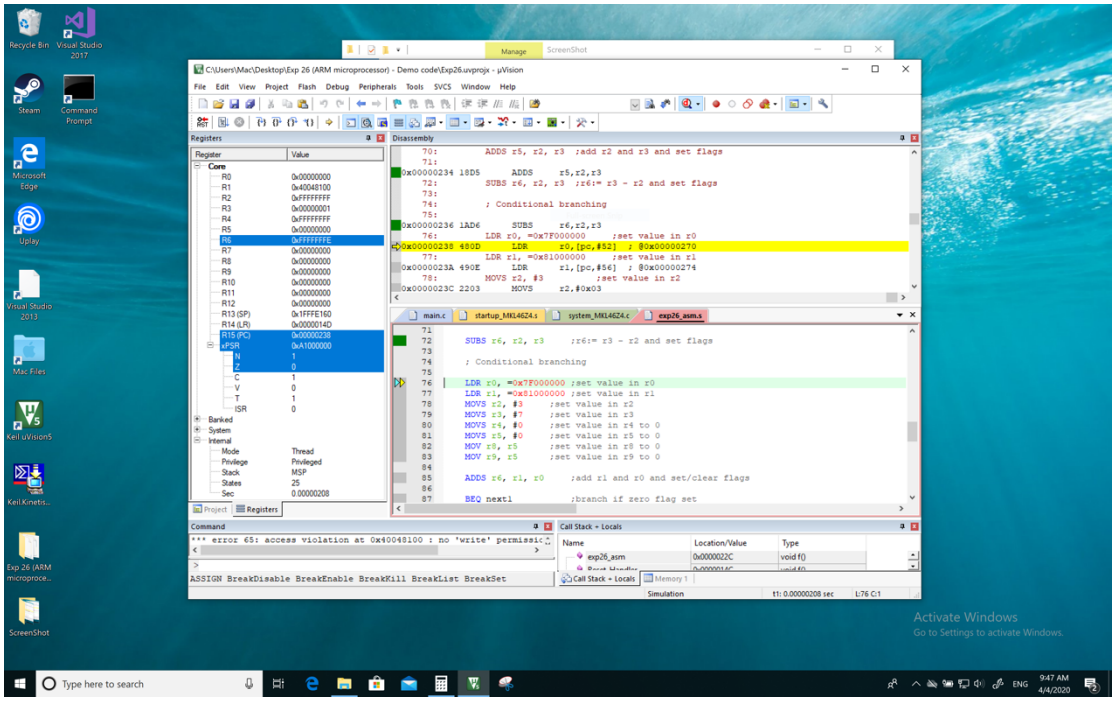
(3) R2 to 7FFFFFFF


C:\Users\Mac\Desktop\Exp 26 (ARM microprocessor) - Demo code\Exp26.uvprojx - µVision
*** error 65: access violation at 0x40048100 : no 'write' permissic
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet
Activate Windows
Go to Settings to activate Windows.
Type here to search


C:\Users\Mac\Desktop\Exp 26 (ARM microprocessor) - Demo code\Exp26.uvprojx - µVision
*** error 65: access violation at 0x40048100 : no 'write' permissic
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet
Activate Windows
Go to Settings to activate Windows.
Type here to search

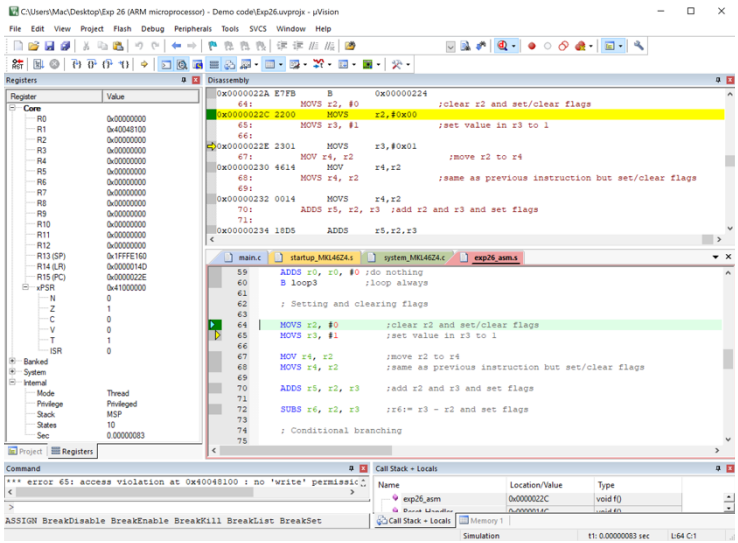

C:\Users\Mac\Desktop\Exp 26 (ARM microprocessor) - Demo code\Exp26.uvprojx - µVision
Activate Windows
Go to Settings to activate Windows.

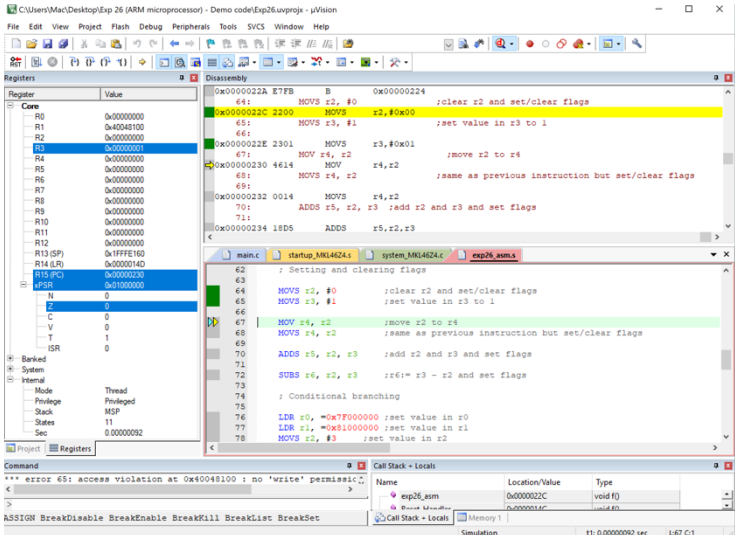

C:\Users\Mac\Desktop\Exp 26 (ARM microprocessor) - Demo code\Exp26.uvprojx - µVision
Activate Windows
Go to Settings to activate Windows.

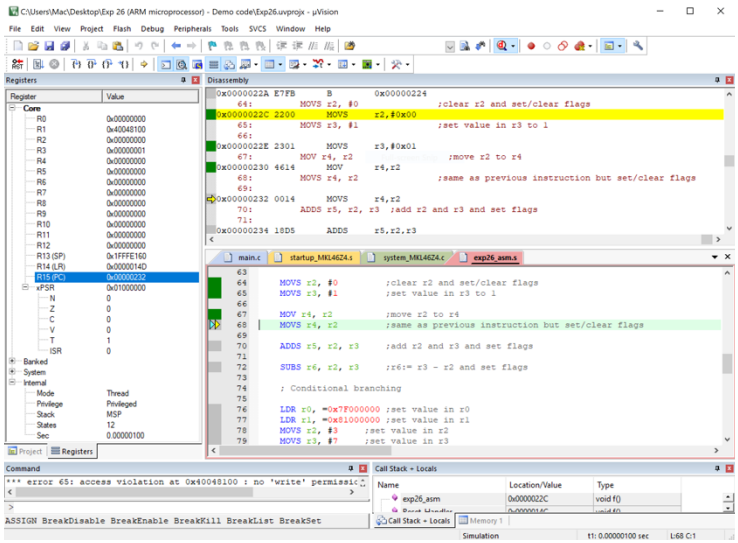


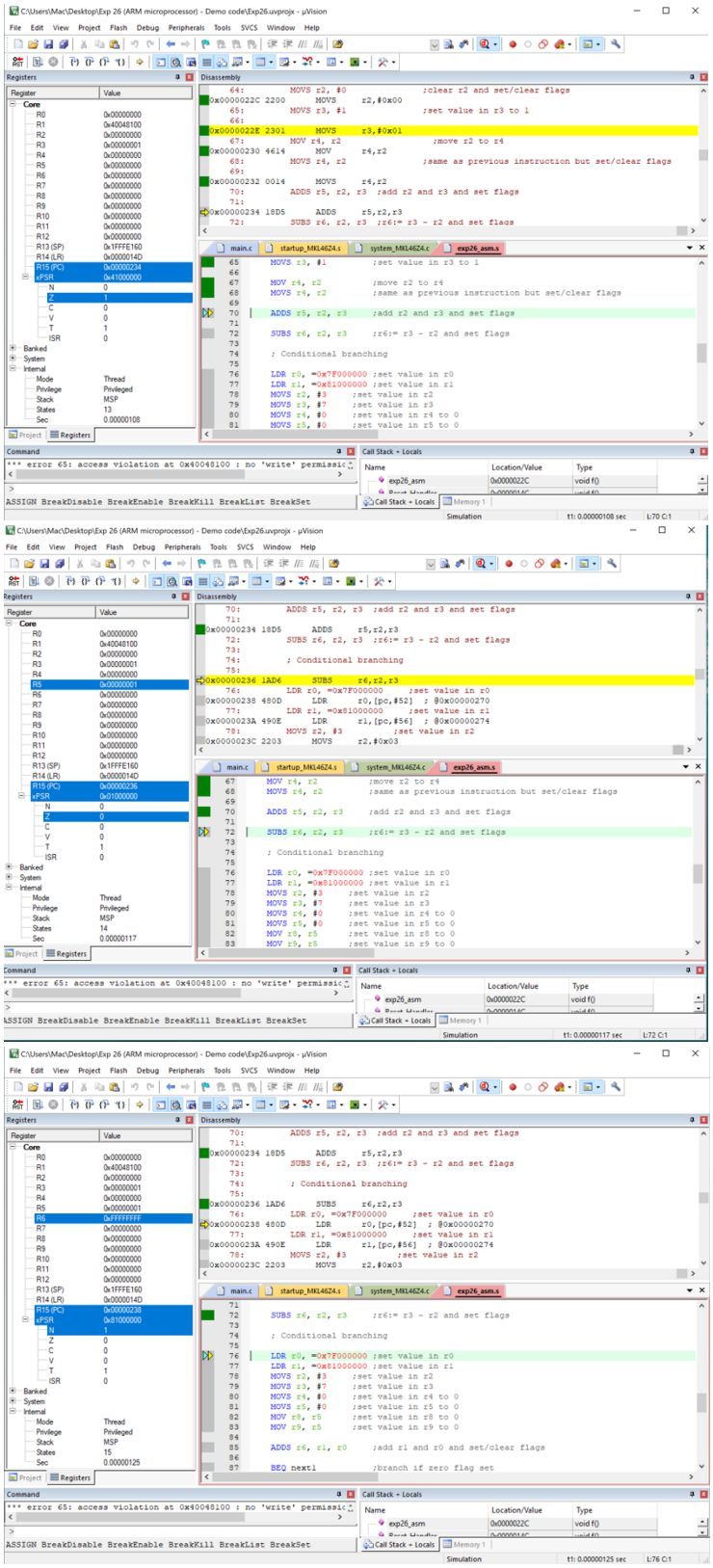
Screenshot (Snipping tool, Preview or equivalent):

(1) R2 initial

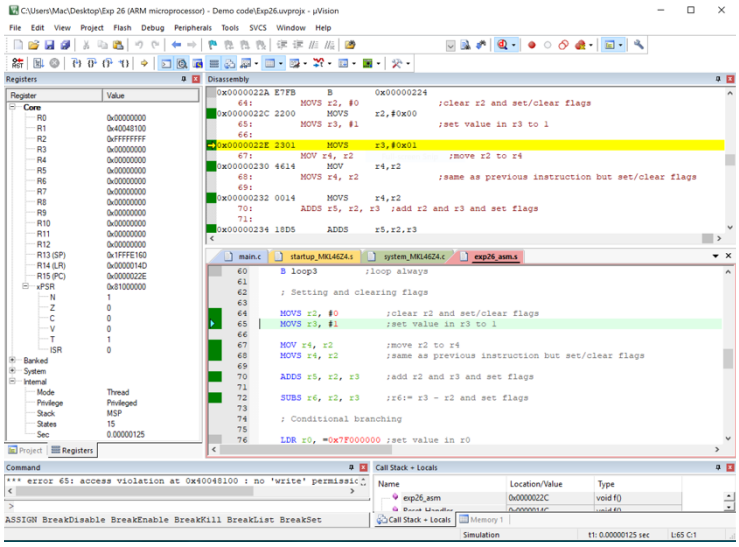


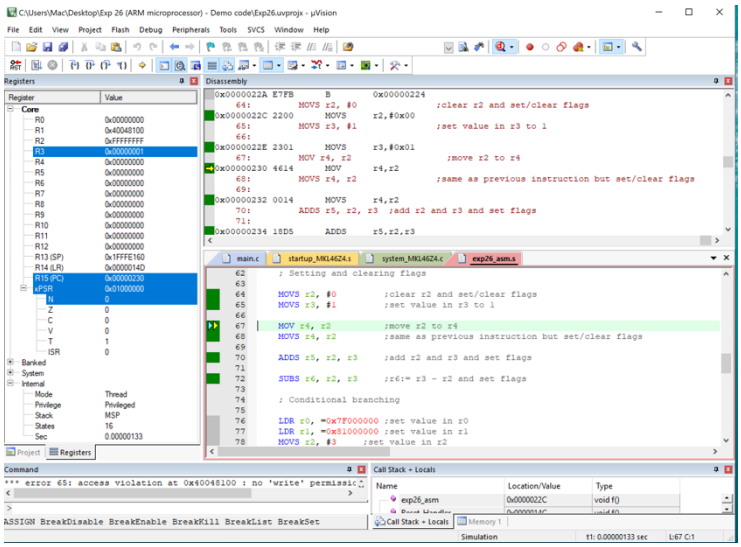


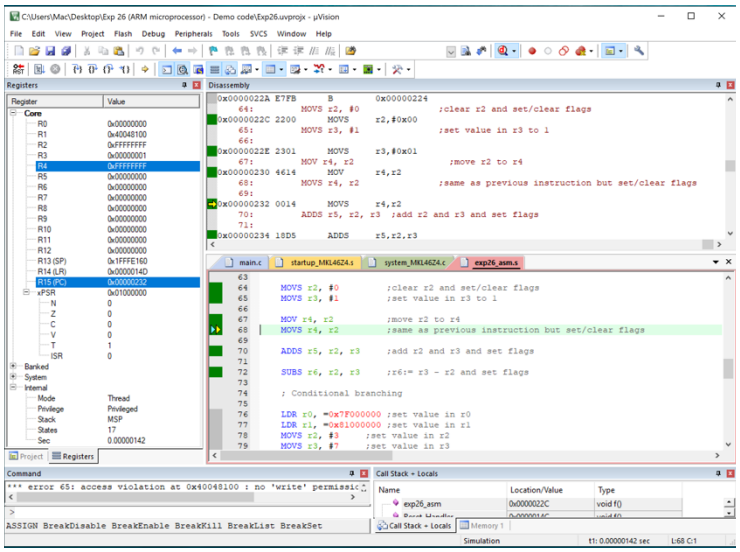


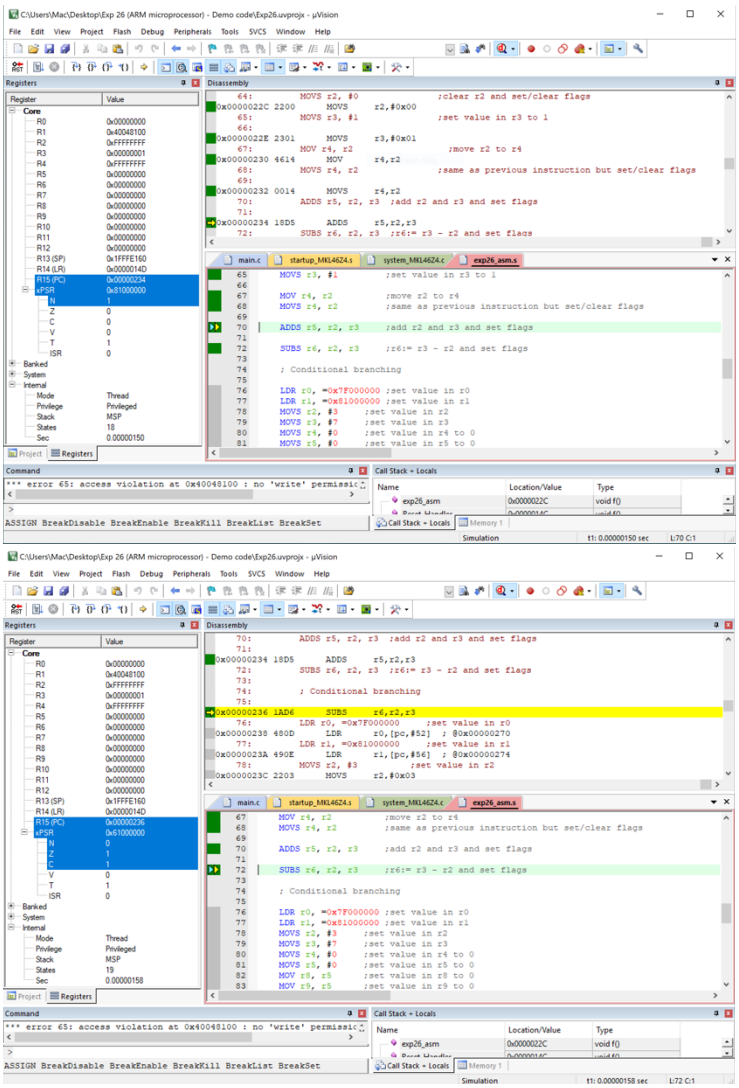


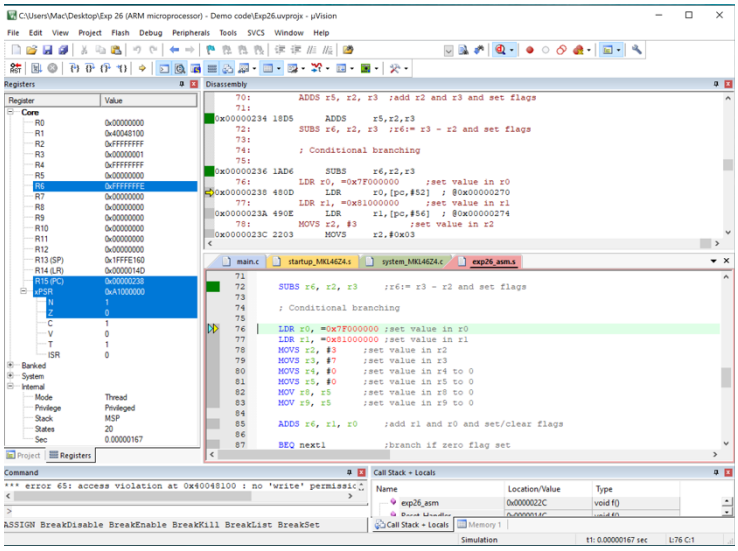
(2) R2 to FFFFFFFF


C:\Users\Mac\Desktop\Exp 26 (ARM microprocessor) - Demo code\Exp26.uvprojx - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Disassembly
main.c
startup_MKL46Z4.s
system_MKL46Z4.c
exp26_asm.s
Command
Call Stack + Locals
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet


C:\Users\Mac\Desktop\Exp 26 (ARM microprocessor) - Demo code\Exp26.uvprojx - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Disassembly
main.c
startup_MKL46Z4.s
system_MKL46Z4.c
exp26_asm.s
Command
Call Stack + Locals
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet


C:\Users\Mac\Desktop\Exp 26 (ARM microprocessor) - Demo code\Exp26.uvprojx - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Disassembly
main.c
startup_MKL46Z4.s
system_MKL46Z4.c
exp26_asm.s
Command
Call Stack + Locals
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet


C:\Users\Mac\Desktop\Exp 26 (ARM microprocessor) - Demo code\Exp26.uvprojx - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Disassembly
main.c
startup_MKL46Z4.s
system_MKL46Z4.c
exp26_asm.s
Command
*** error 65: access violation at 0x40048100 : no 'write' permissic
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet
Call Stack + Locals
Simulation
t1: 0.00000150 sec
L:70 C:1
C:\Users\Mac\Desktop\Exp 26 (ARM microprocessor) - Demo code\Exp26.uvprojx - µVision
t1: 0.00000158 sec
L:72 C:1



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	FFFFFFFF	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
FFFFFFFF	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	FFFFFFFE	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

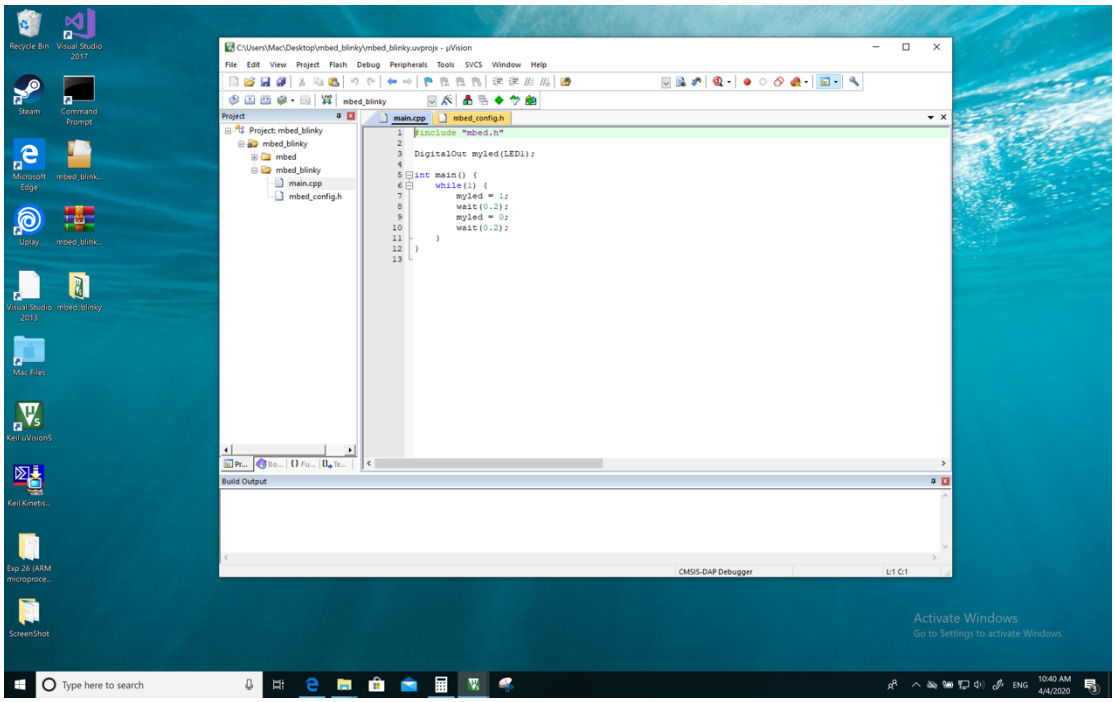
```
ADDS r6, r1, r0          ;add r1 and r0 and set/clear flags
    R6 = 0x00000000 → Z=1 C=1 other flags is 0
BNE next2                ;branch if zero flag clear → Omit because the zero flag not
                          clear
    BCS next3            ;branch because carry flag set
BCC next4                ;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
```

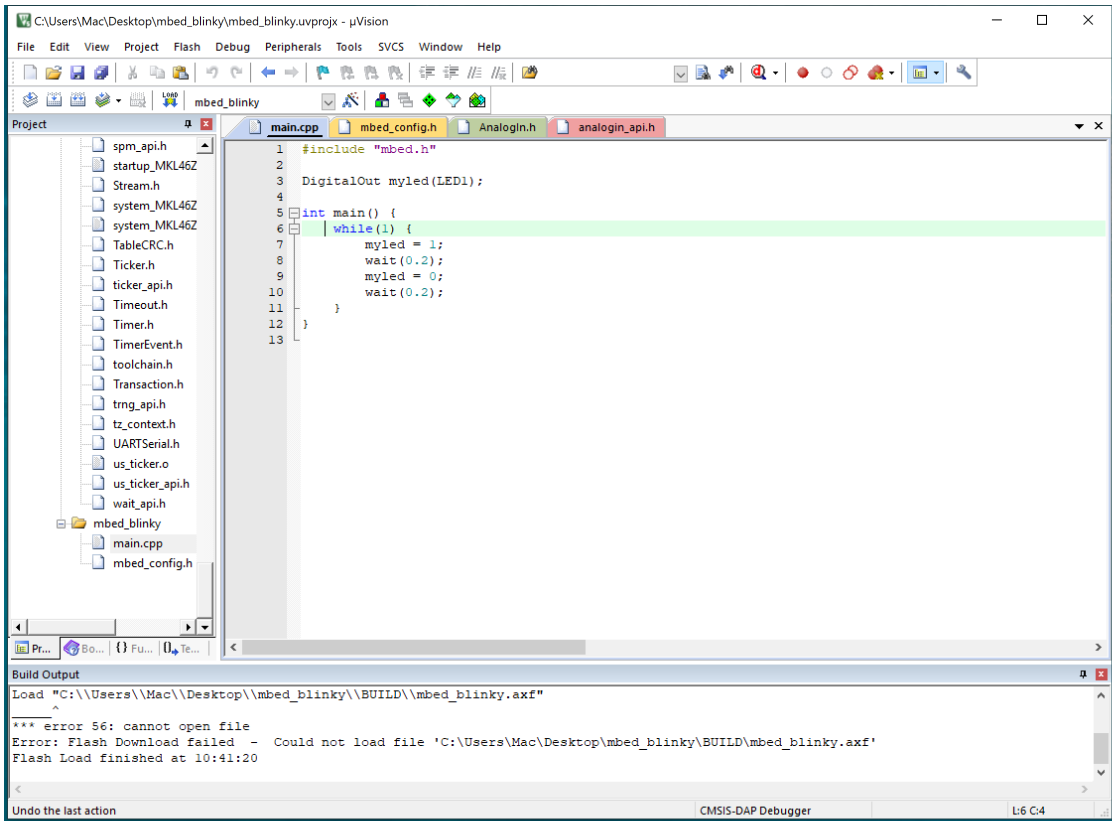
So, the expected is the same as the experiment which is BEQ next1, BCS next3, BMI next5, BPL next6, BVC last is executed.

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

Screenshot (PrintScreen or Screenshot):



Screenshot (Snipping tool, Preview or equivalent):


C:\Users\Mac\Desktop\mbed_blinky\mbed_blinky.uvprojx - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
mbed_blinky
Project
spm_api.h
startup_MKL46Z
Stream.h
system_MKL46Z
system_MKL46Z
TableCRC.h
Ticker.h
ticker_api.h
Timeout.h
Timer.h
TimerEvent.h
toolchain.h
Transaction.h
trng_api.h
tz_context.h
UARTSerial.h
us_ticker.o
us_ticker_api.h
wait_api.h
mbed_blinky
main.cpp
mbed_config.h
main.cpp
mbed_config.h
AnalogIn.h
analogin_api.h
#include "mbed.h"
DigitalOut myled(LED1);
int main() {
while(1) {
myled = 1;
wait(0.2);
myled = 0;
wait(0.2);
}
}
Build Output
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
Undo the last action
CMSIS-DAP Debugger
L:6 C:4