**Microprocessor and Computer Architecture Laboratory**

**UE19CS256**

**4th Semester, Academic Year 2020-21**

Date: 1/2/2021

|  |  |  |
| --- | --- | --- |
| Name: TUSHAR Y S | SRN:  PES1UG19CS545 | Section  I |

Week#\_\_\_\_2\_\_\_\_\_\_\_ Program Number: \_ 1\_\_\_\_\_

Based on the value of the number in R0, Write an ALP to store 1 in R1 if R0 is zero, Store 2 in R1 if R0 is positive, Store 3 in R1 if R0 is negative.

ARM Assembly Code:

Case 1 (zero):

.text

MOV R0,#0

CMP R0,#0

BEQ c1

BMI c2

MOV R1,#2

SWI 0x11

c1:

MOV R1,#1

SWI 0x11

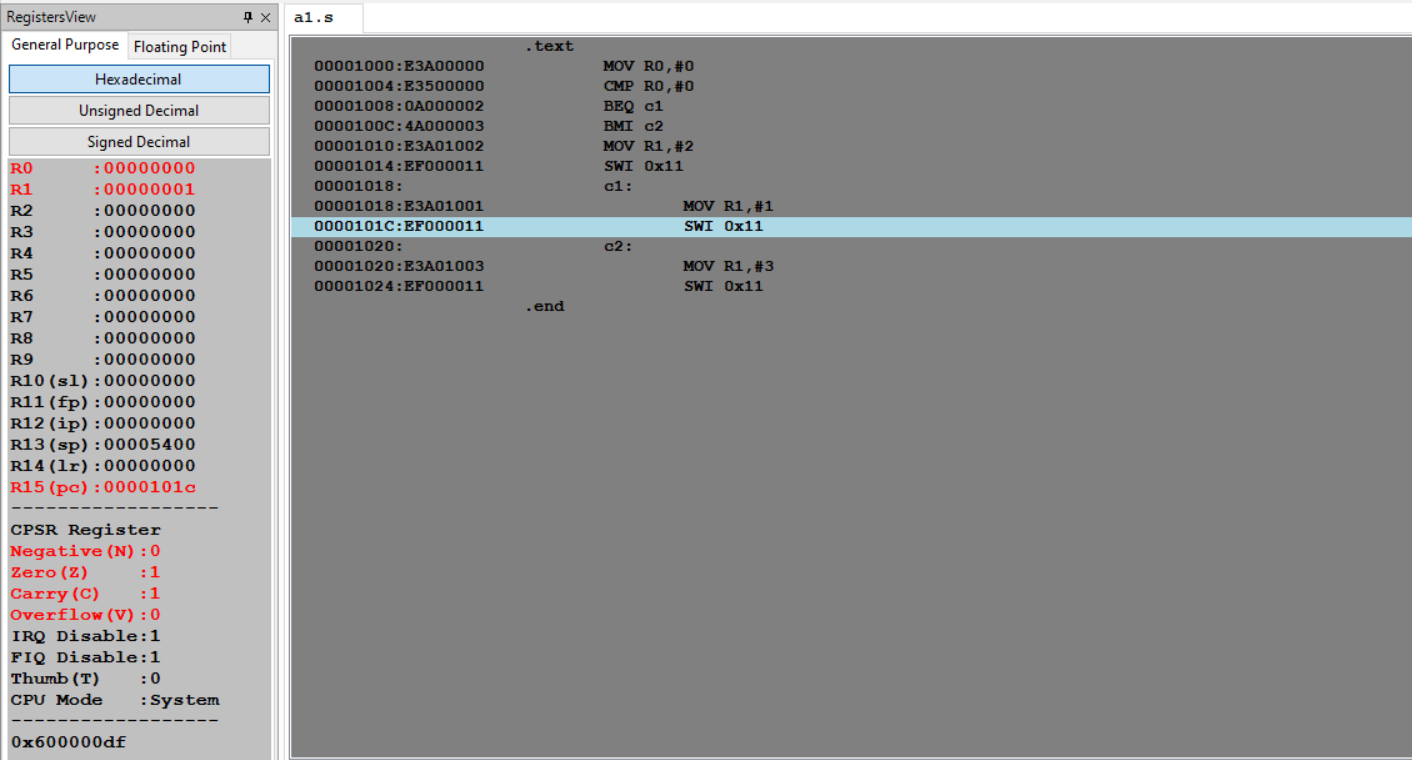
c2:

MOV R1,#3

SWI 0x11

.end

Output:



Case 2 (positive number):

.text

MOV R0,#18

CMP R0,#0

BEQ c1

BMI c2

MOV R1,#2

SWI 0x11

c1:

MOV R1,#1

SWI 0x11

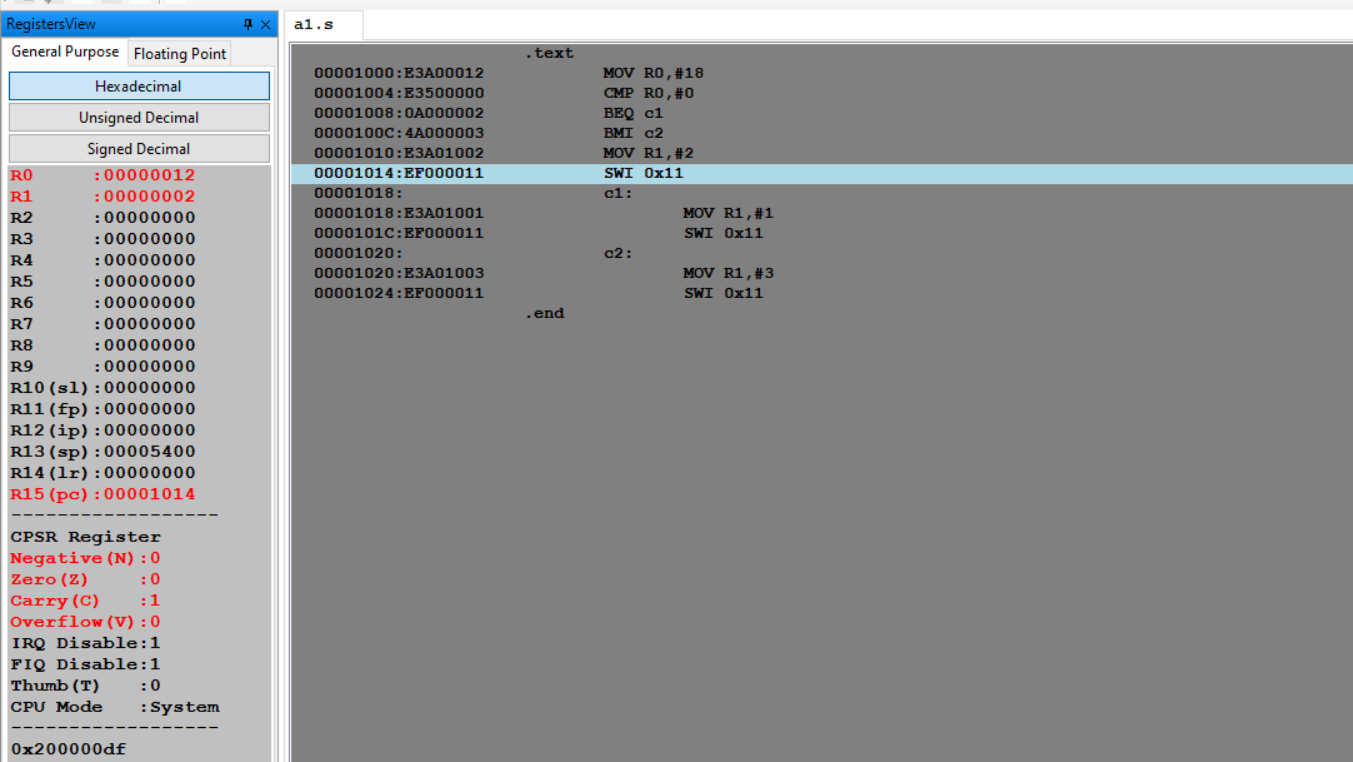
c2:

MOV R1,#3

SWI 0x11

.end

Output:



Case 3 (negative number):

.text

MOV R0,#-18

CMP R0,#0

BEQ c1

BMI c2

MOV R1,#2

SWI 0x11

c1:

MOV R1,#1

SWI 0x11

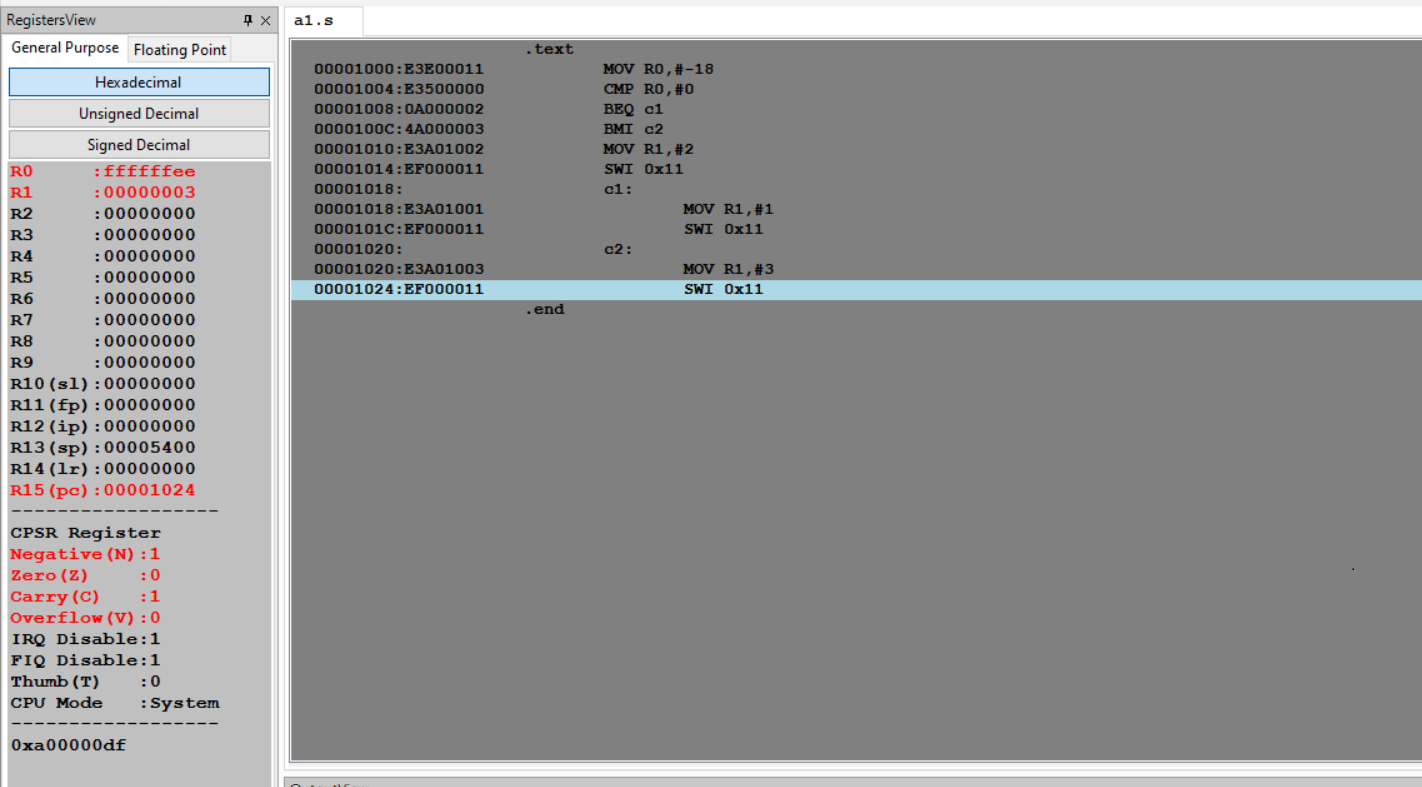
c2:

MOV R1,#3

SWI 0x11

.end

Output:



Week#\_\_\_\_2\_\_\_\_\_\_\_ Program Number: \_ 2\_\_\_\_\_

Write an ALP to compare the value of R0 and R1, add if R0 = R1, else subtract.

ARM Assembly Code:

Case 1:

.text

MOV R0,#13

MOV R1,#13

CMP R0,R1

BEQ equal

SUB R2,R0,R1

SWI 0x11

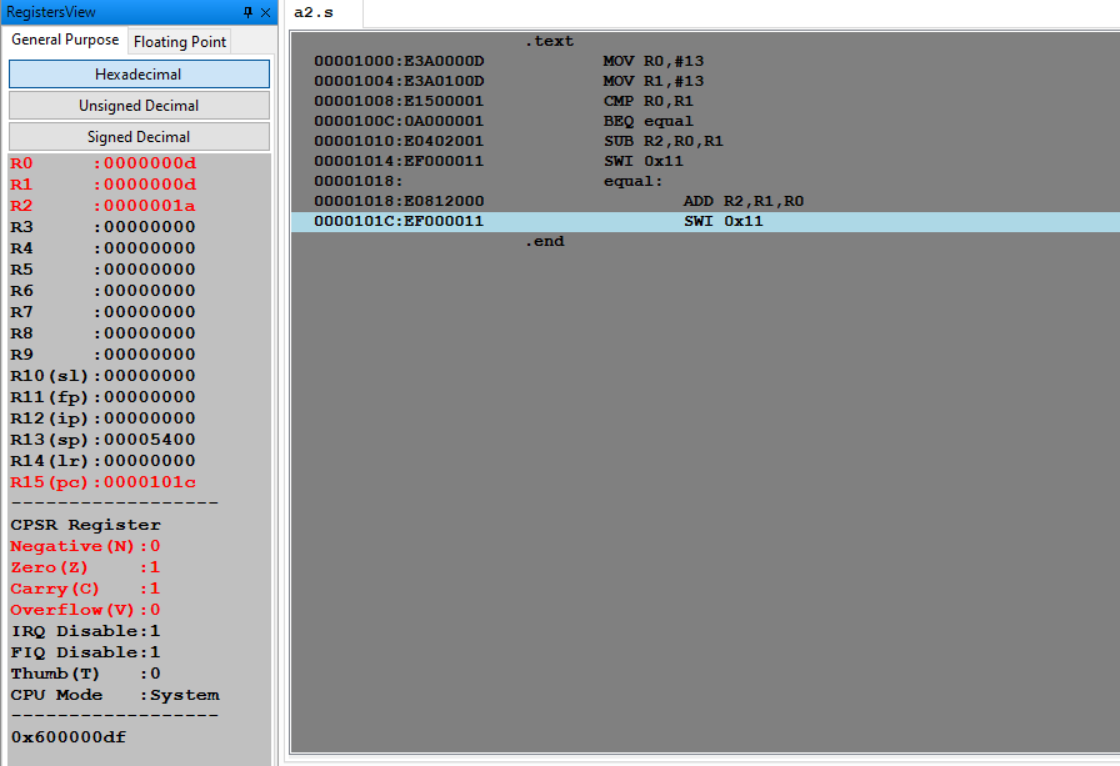
equal:

ADD R2,R1,R0

SWI 0x11

.end

Output:



Case 2:

.text

MOV R0,#10

MOV R1,#4

CMP R0,R1

BEQ equal

SUB R2,R0,R1

SWI 0x11

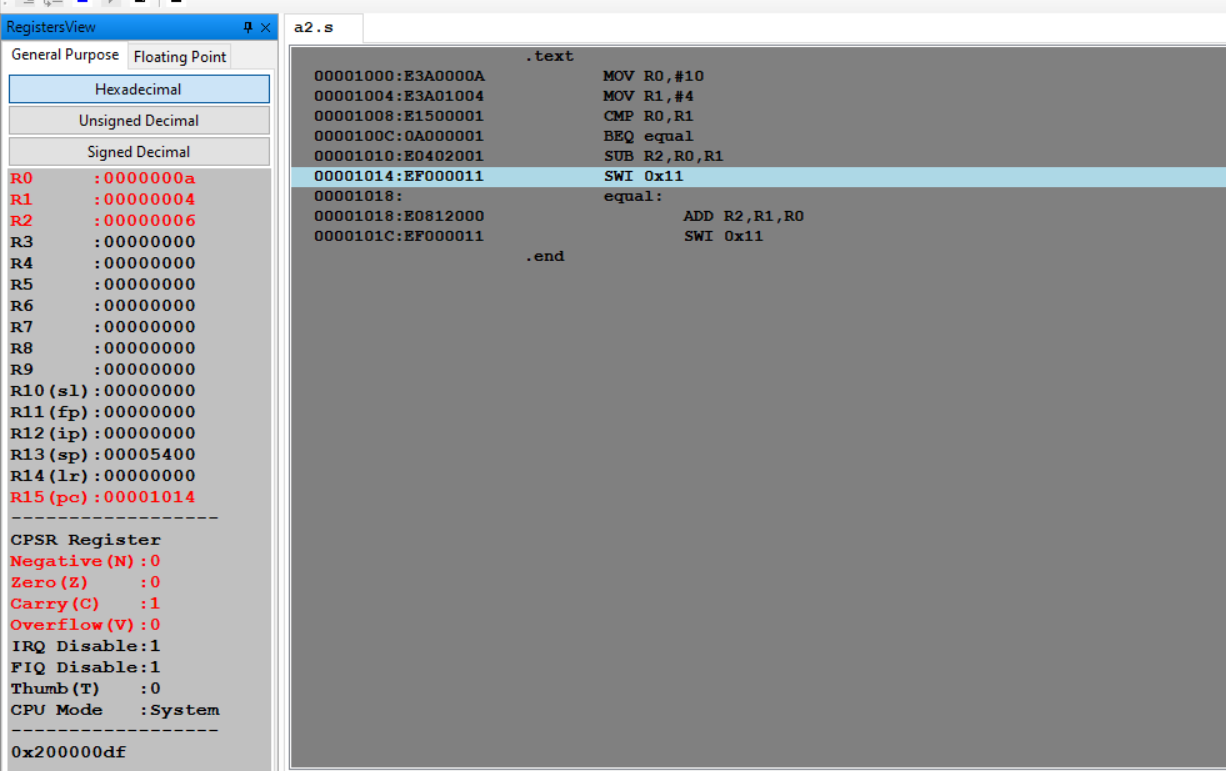
equal:

ADD R2,R1,R0

SWI 0x11

.end

Output:



Week#\_\_\_\_2\_\_\_\_\_\_\_ Program Number: \_ 3\_\_\_\_\_

Write an ALP to find the factorial of a number stored in R0. Store the value in R1 (without using LDR and STR instructions). Use only registers.

ARM Assembly Code:

.text

MOV R0,#3

MOV R1,R0

CMP R1,#1

BEQ j0

j1:

SUB R1,R1,#1

MUL R2,R1,R0

MOV R0,R2

CMP R1,#1

BNE j1

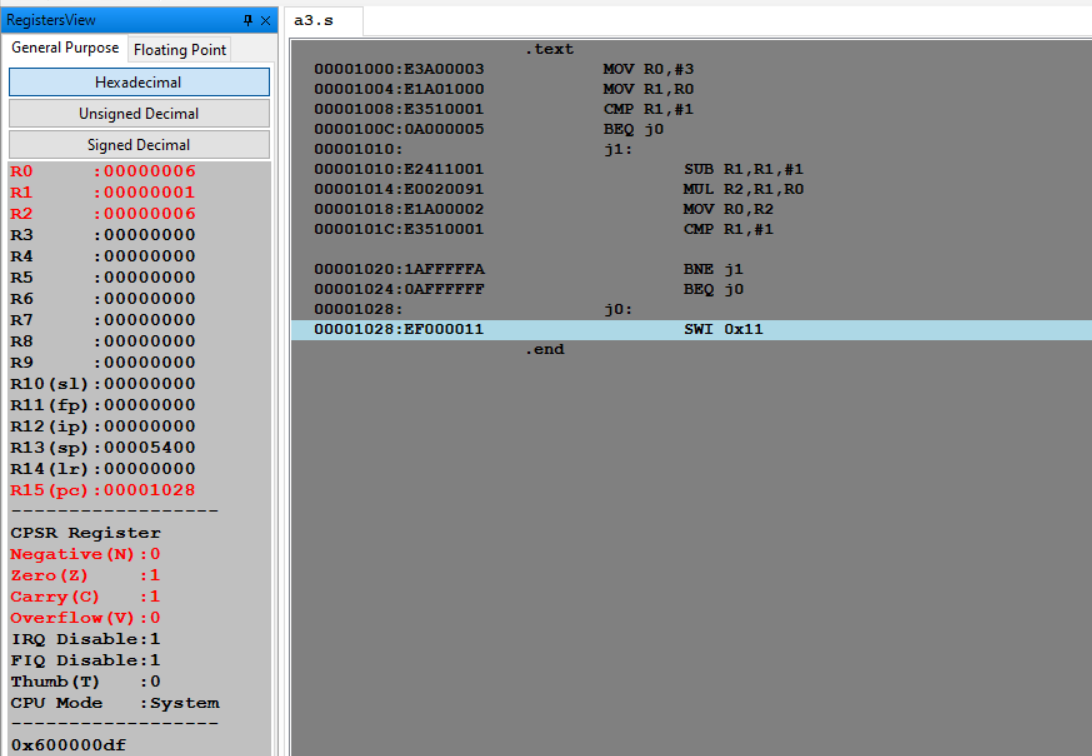
BEQ j0

j0:

SWI 0x11

.end

Output:



Week#\_\_\_\_2\_\_\_\_\_\_\_ Program Number: \_ 4a\_\_\_\_\_

Write an ALP to add two 32 bit numbers loaded from memory and store the result in memory.

ARM Assembly Code:

.text

LDR R0,=A

LDR R1,=B

LDR R2,=C

LDR R3,[R0]

LDR R4,[R1]

ADD R5,R4,R3

STR R5,[R2]

SWI 0x11

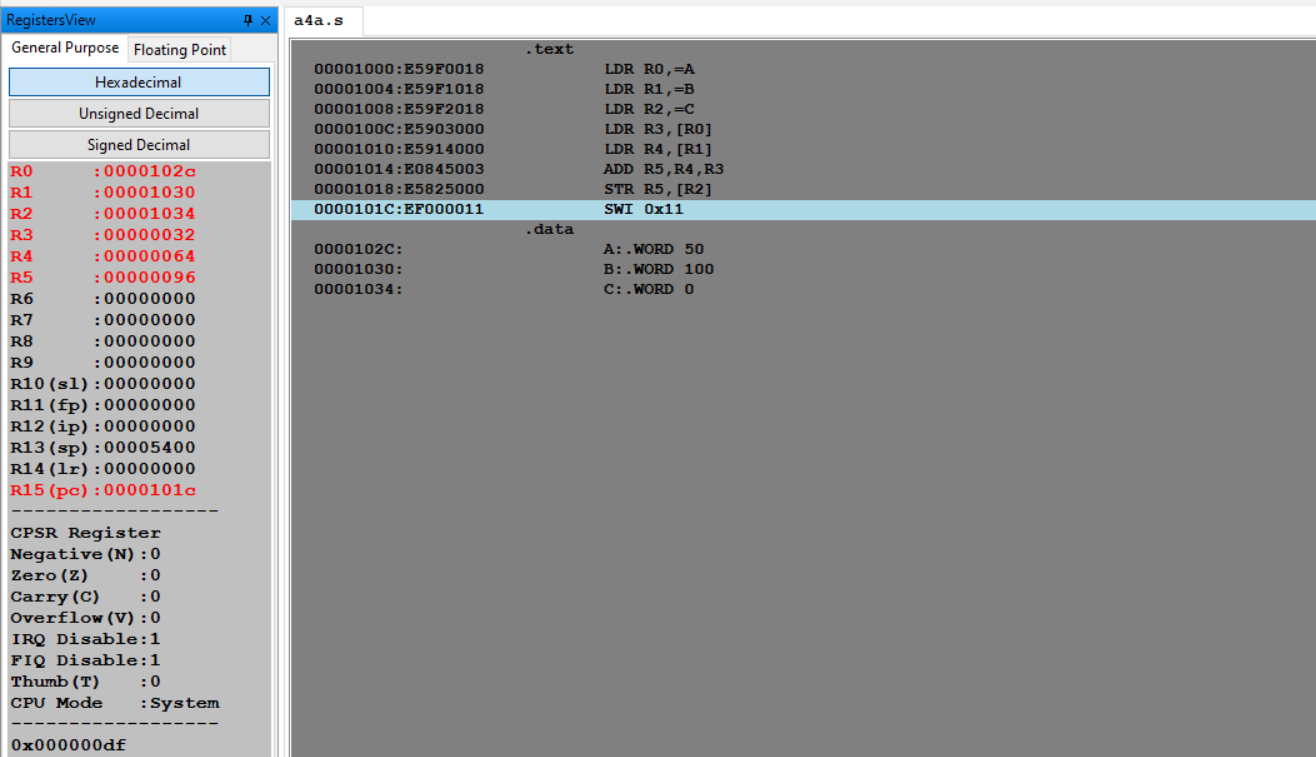
.data

A:.WORD 50

B:.WORD 100

C:.WORD 0

Output:



Week#\_\_\_\_2\_\_\_\_\_\_\_ Program Number: \_ 4b\_\_\_\_\_

Write an ALP to add two 16 bit numbers loaded from memory and store the result in memory.

ARM Assembly Code:

.text

LDR R0,=A

LDR R1,=B

LDR R2,=C

LDRH R3,[R0]

LDRH R4,[R1]

ADD R5,R4,R3

STRH R5,[R2]

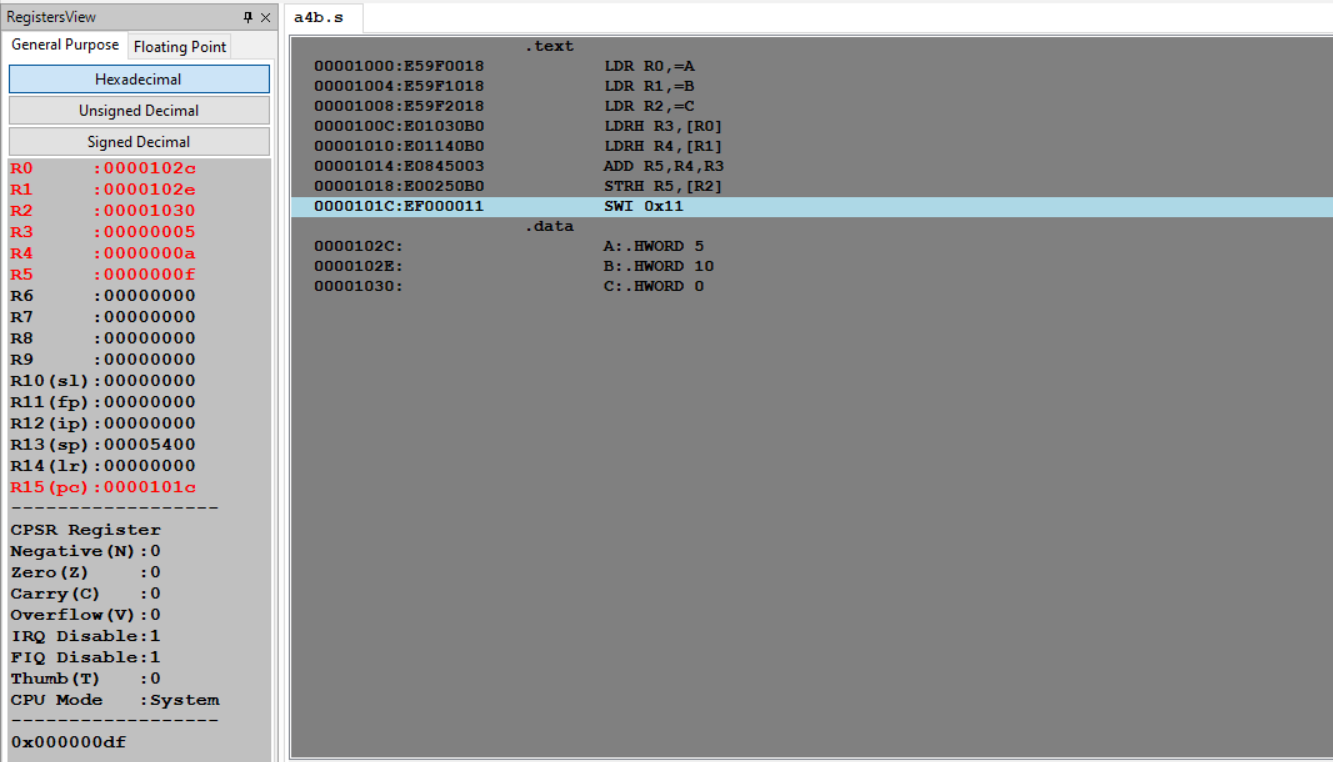
SWI 0x11

.data

A:.HWORD 5

B:.HWORD 10

C:.HWORD 0

Output:  


Week#\_\_\_\_2\_\_\_\_\_\_\_ Program Number: \_ 5a\_\_\_\_\_

Write an ALP to find GCD of two numbers (without using LDR and STR instructions). Both numbers are in registers. Use only registers.

ARM Assembly Code:

.text

MOV R0,#100

MOV R1,#5

MOV R2,R0

MOV R3,R1

L1:

CMP R2,R3

BEQ OUT

BLT L2

SUB R2,R2,R3

CMP R3,R2

BEQ OUT

BNE L1

L2:

SUB R3,R3,R2

CMP R3,R2

BEQ OUT

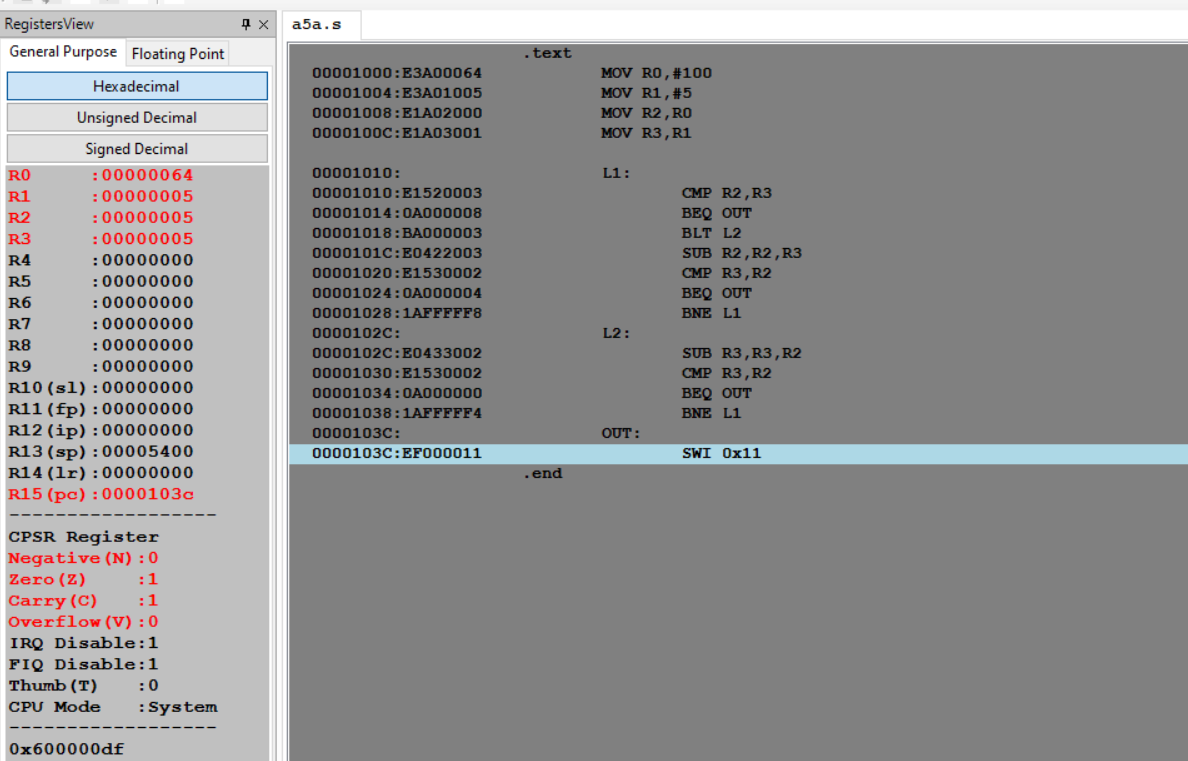
BNE L1

OUT:

SWI 0x11

.end

Output:



Week#\_\_\_\_2\_\_\_\_\_\_\_ Program Number: \_ 5b\_\_\_\_\_

Write an ALP to find the GCD of given numbers (both numbers in memory) Store result in memory.

ARM Assembly Code:

Ex 1 (A=B):

.text

LDR R0,=A

LDR R1,=B

LDR R2,[R0]

LDR R3,[R1]

L1:

CMP R2,R3

BEQ OUT

BLT L2

SUB R2,R2,R3

CMP R3,R2

BEQ OUT

BNE L1

L2:

SUB R3,R3,R2

CMP R3,R2

BEQ OUT

BNE L1

OUT:

SWI 0x11

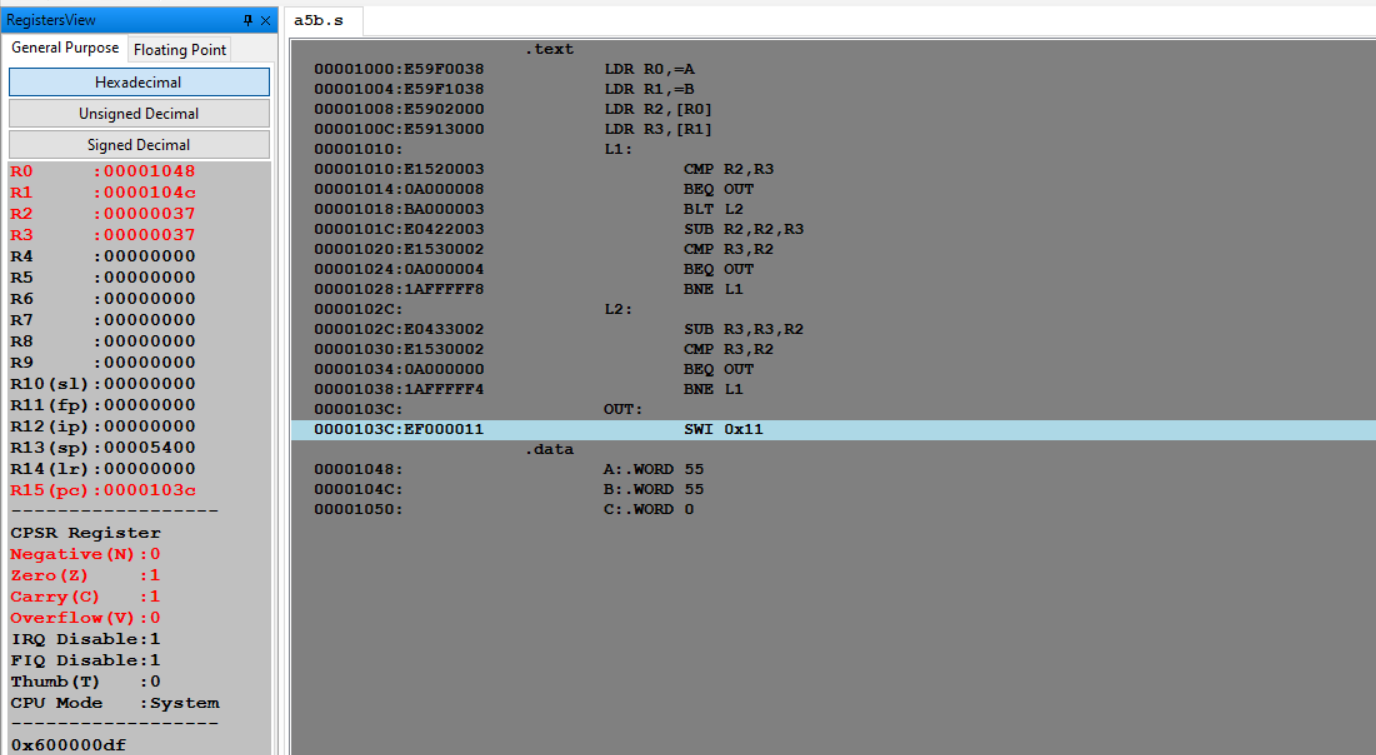
.data

A:.WORD 55

B:.WORD 55

C:.WORD 0

Output:



Ex 2 (A>B):

.text

LDR R0,=A

LDR R1,=B

LDR R2,[R0]

LDR R3,[R1]

L1:

CMP R2,R3

BEQ OUT

BLT L2

SUB R2,R2,R3

CMP R3,R2

BEQ OUT

BNE L1

L2:

SUB R3,R3,R2

CMP R3,R2

BEQ OUT

BNE L1

OUT:

SWI 0x11

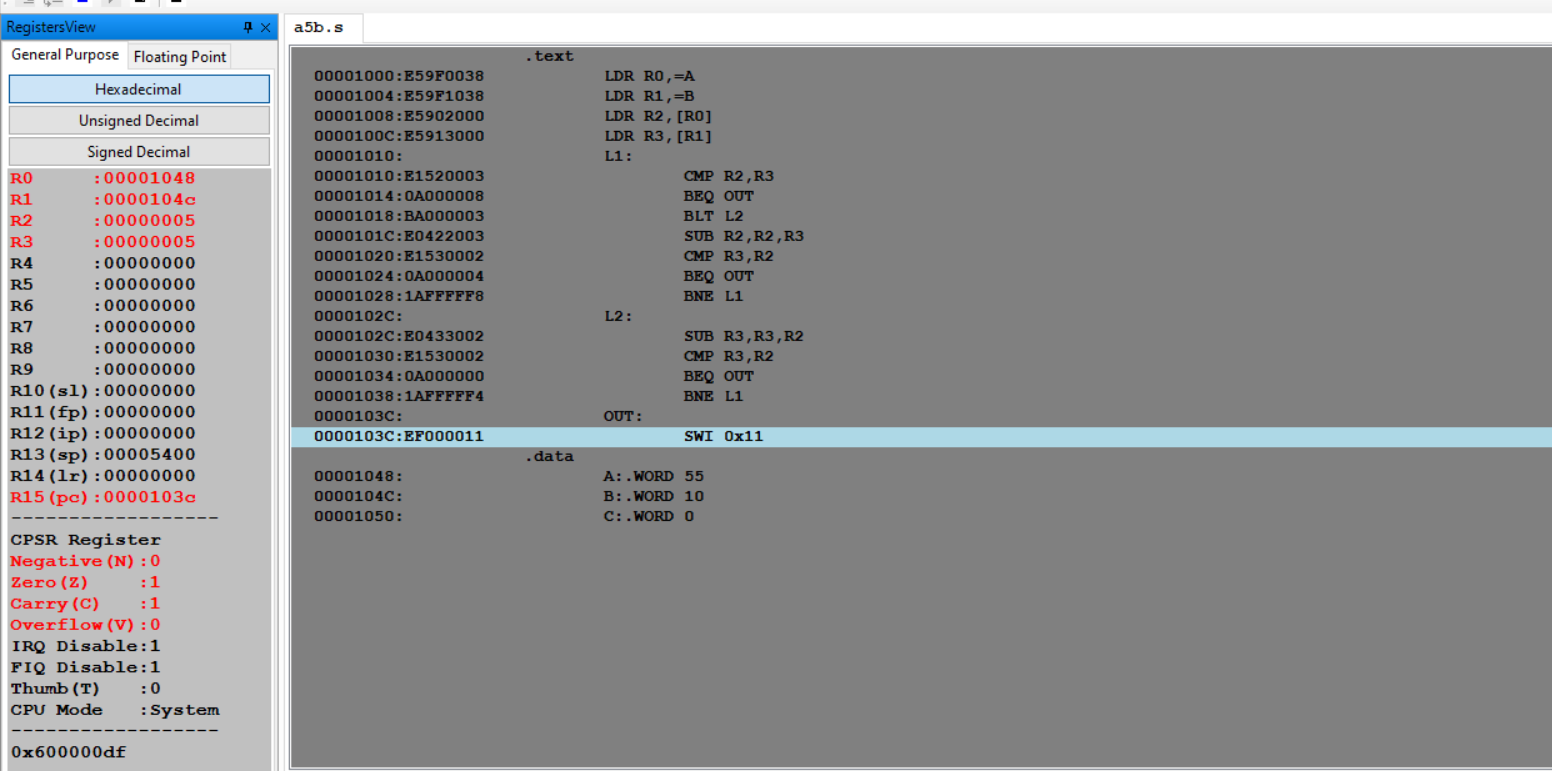
.data

A:.WORD 55

B:.WORD 10

C:.WORD 0

Output:



Ex 3 (A<B):

.text

LDR R0,=A

LDR R1,=B

LDR R2,[R0]

LDR R3,[R1]

L1:

CMP R2,R3

BEQ OUT

BLT L2

SUB R2,R2,R3

CMP R3,R2

BEQ OUT

BNE L1

L2:

SUB R3,R3,R2

CMP R3,R2

BEQ OUT

BNE L1

OUT:

SWI 0x11

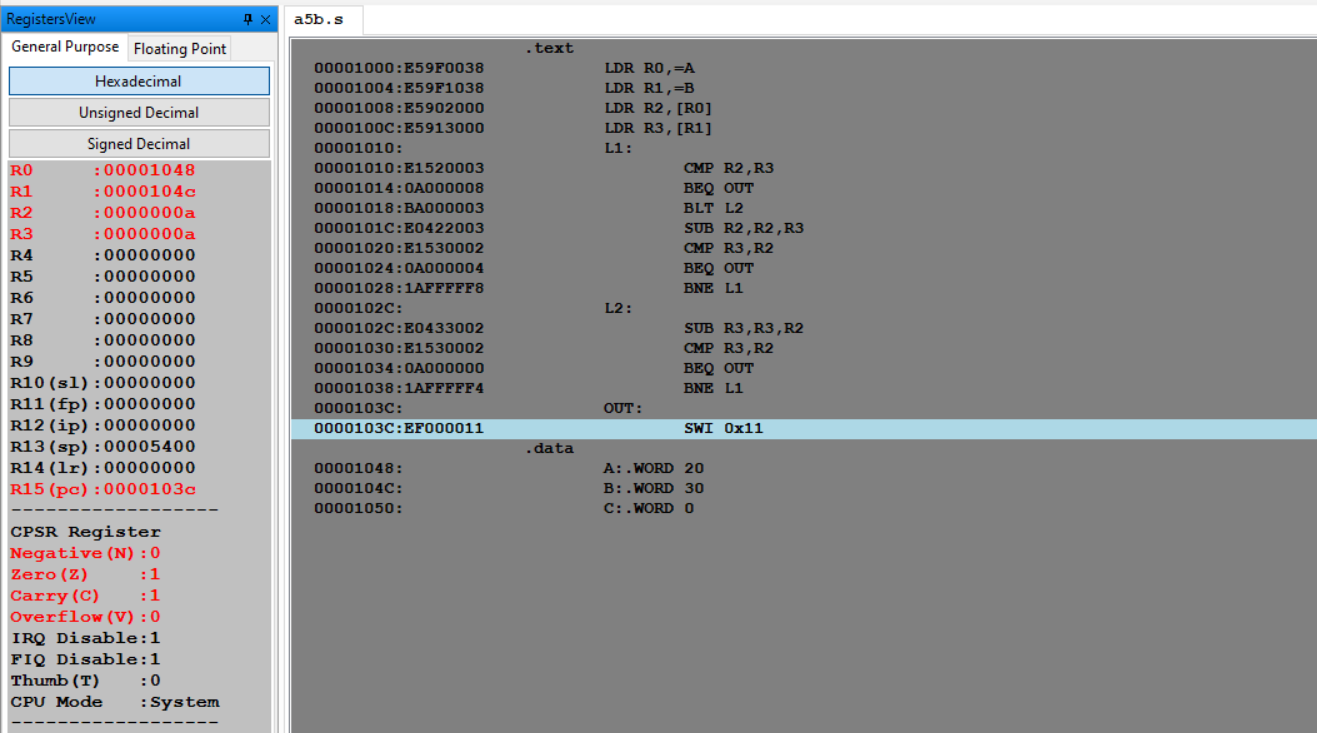
.data

A:.WORD 20

B:.WORD 30

C:.WORD 0

Output:



Week#\_\_\_\_2\_\_\_\_\_\_\_ Program Number: \_ 6a\_\_\_\_\_

Write an ALP to add an array of ten 32 bit numbers from memory.

ARM Assembly Code:

.text

LDR R0,=A

LDR R1,=B

LDR R4,[R1]

MOV R3,#0

L:

LDR R2,[R0],#4

ADD R3,R3,R2

SUB R4,R4,#1

CMP R4,#0

BEQ OUT

BNE L

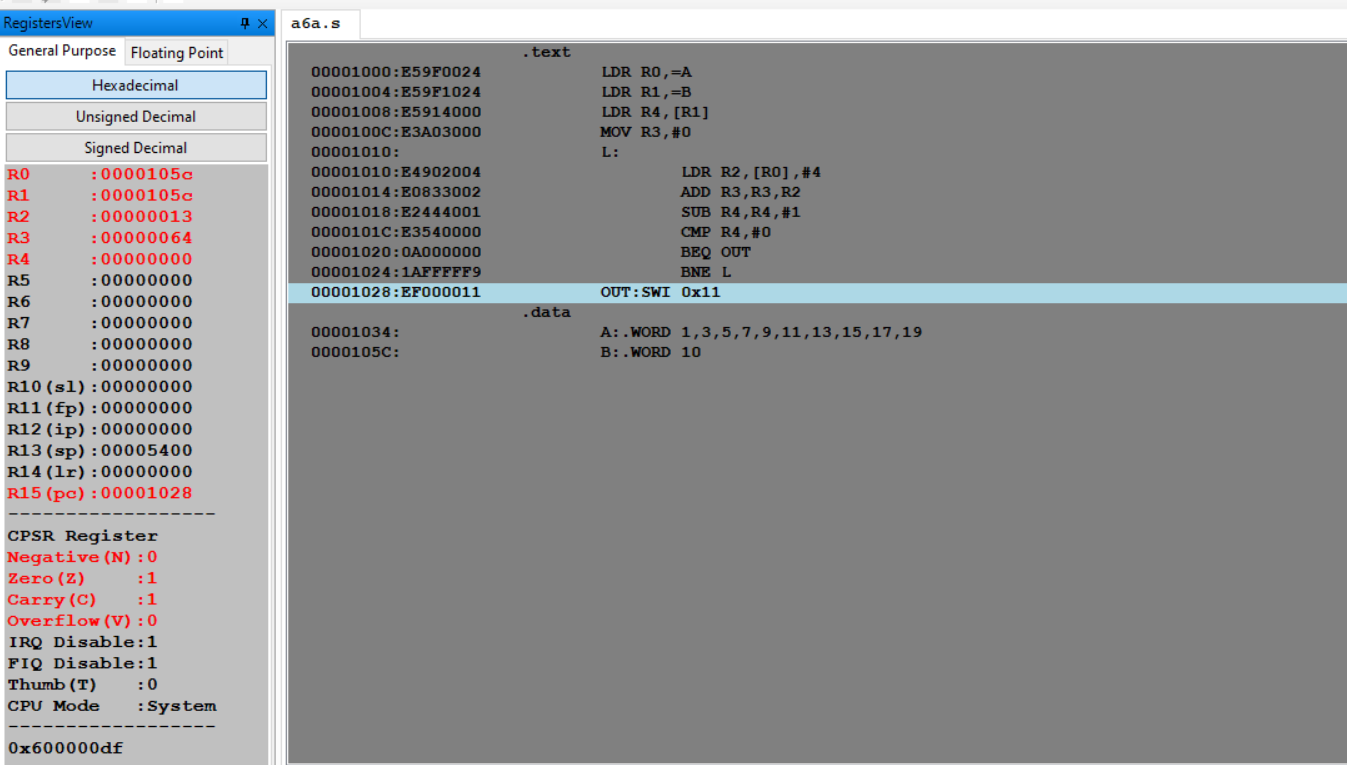
OUT:SWI 0x11

.data

A:.WORD 1,3,5,7,9,11,13,15,17,19

B:.WORD 10

Output:



Week#\_\_\_\_2\_\_\_\_\_\_\_ Program Number: \_ 6b\_\_\_\_\_

Write an ALP to add array of ten 8 bit numbers taking data from memory location stored as byte data (use .byte to store the data instead of .word)

ARM Assembly Code:

.text

LDR R0,=A

LDR R1,=B

LDRB R4,[R1]

MOV R3,#0

L1:

LDRB R2,[R0]

ADD R3,R3,R2

SUB R4,R4,#1

CMP R4,#0

BEQ L2

ADD R0,R0,#1

BNE L1

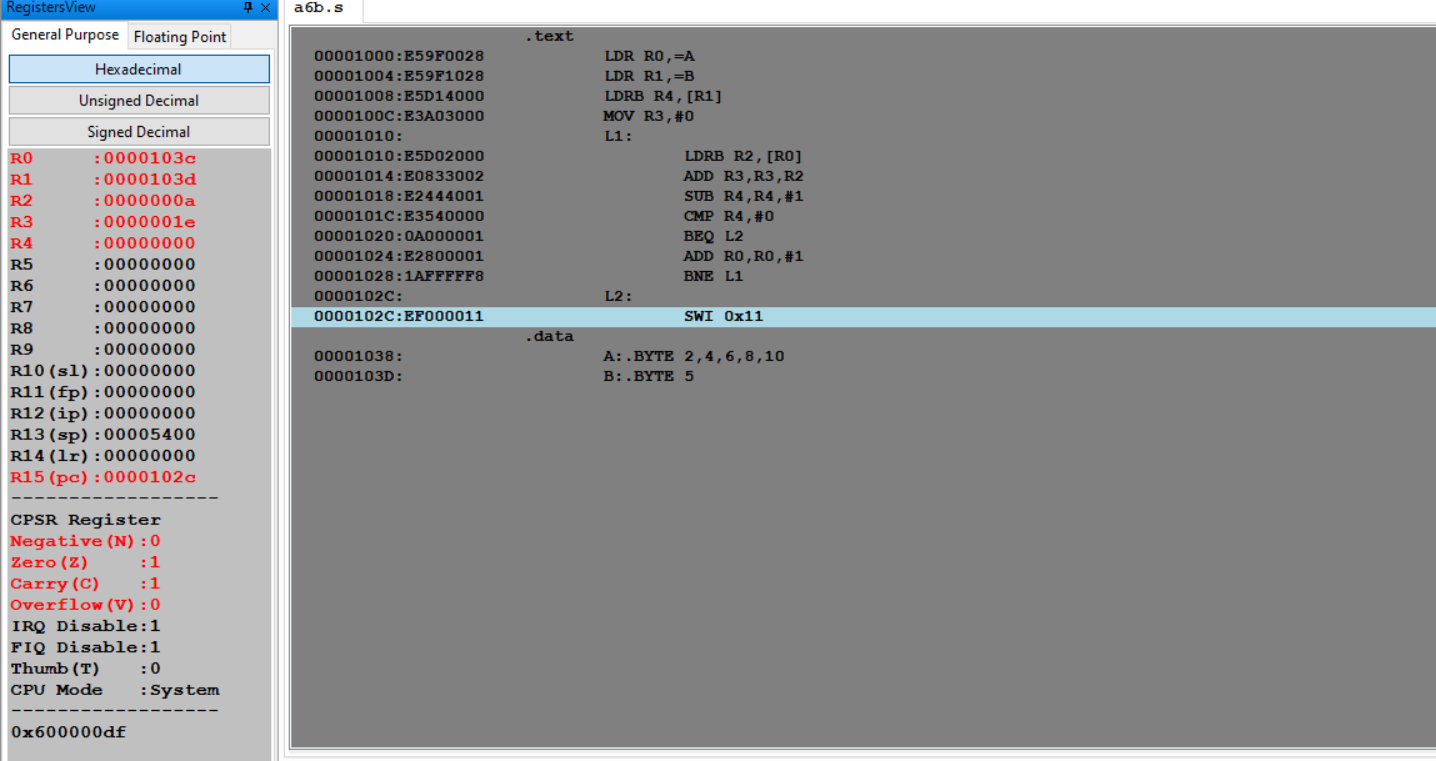
L2:

SWI 0x11

.data

A:.BYTE 2,4,6,8,10

B:.BYTE 5

Output:  


Week#\_\_\_\_2\_\_\_\_\_\_\_ Program Number: \_ 7\_\_\_\_\_

Write an ALP to multiply using barrel shifter.

35\*R0

ARM Assembly Code:

.text

MOV R0,#10

MOV R1,R0,LSL #5

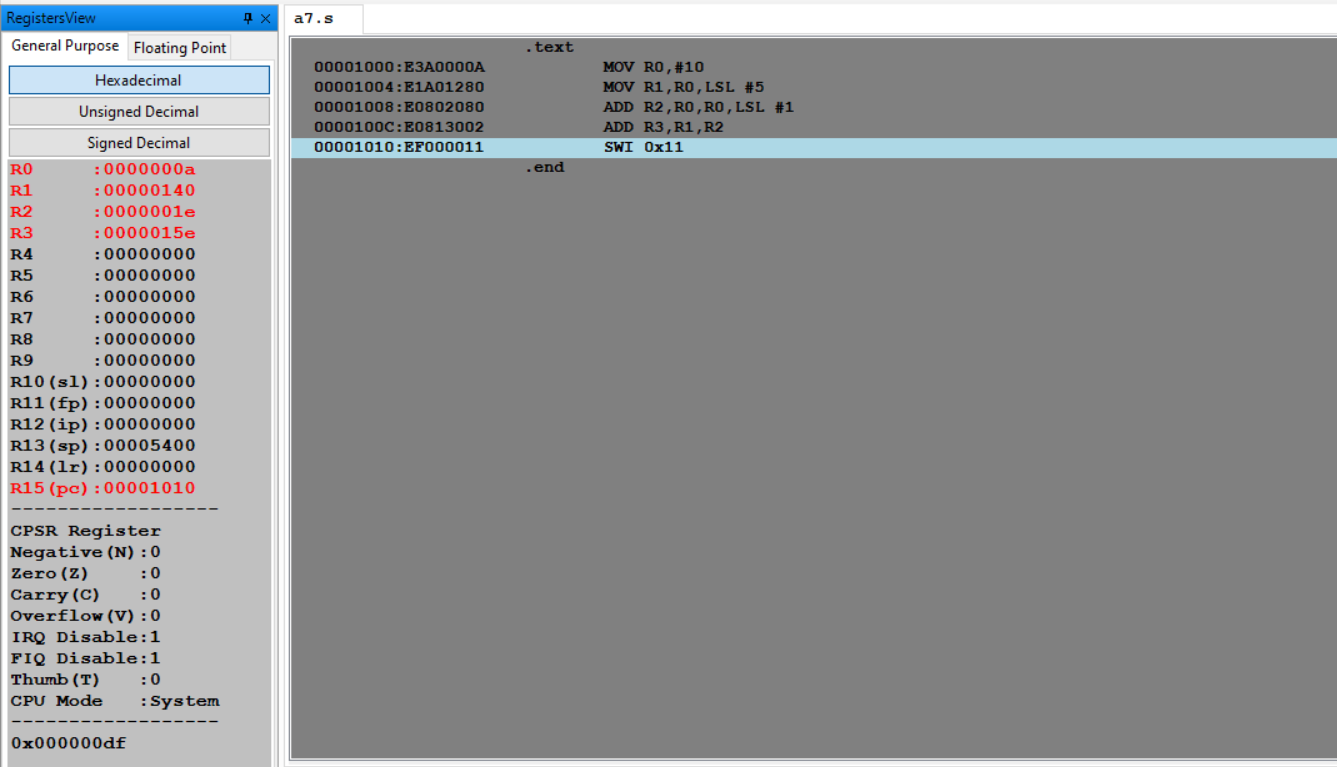
ADD R2,R0,R0,LSL #1

ADD R3,R1,R2

SWI 0x11

.end

Output:



Week#\_\_\_\_2\_\_\_\_\_\_\_ Program Number: \_ 8\_\_\_\_\_

Write an ALP to evaluate the expression (A+B) + (3\*B), where A and B are memory location.

\* Use LSL instruction for multiplication

ARM Assembly Code:

.text

LDR R0,=A

LDR R1,=B

LDR R2,[R0]

LDR R3,[R1]

ADD R5,R2,R3,LSL #2

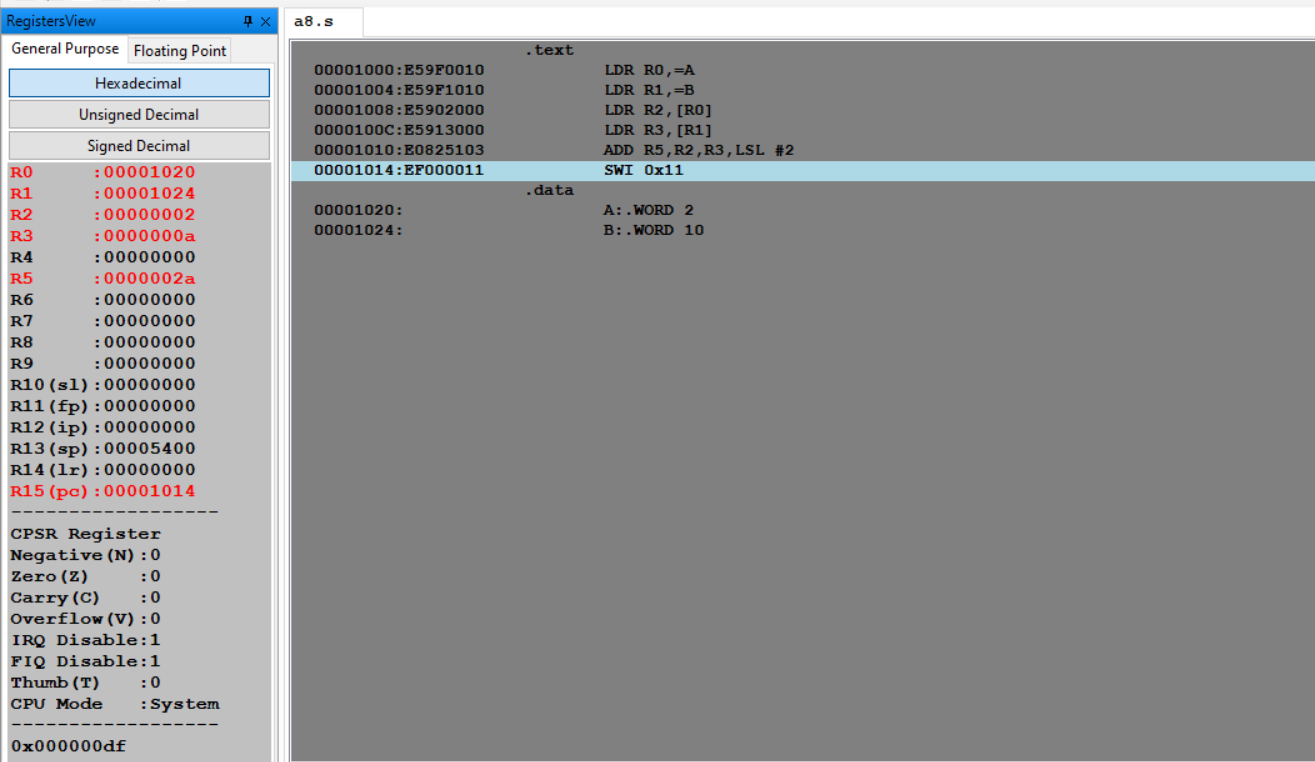
SWI 0x11

.data

A:.WORD 2

B:.WORD 10

Output:



**Disclaimer:**

* The programs and output submitted is duly written, verified and executed by me.
* I have not copied from any of my peers nor from the external resource such as internet.
* If found plagiarized, I will abide with the disciplinary action of the University.

Signature: Tushar

Name: TUSHAR Y S

SRN: PES1UG19CS545

Section: I

Date: 1/2/2021