Department of Computer Science & Engineering (AI & ML)
Microprocessor & Computer Architecture Lab

Lab 1 Programs

UE23CS251B

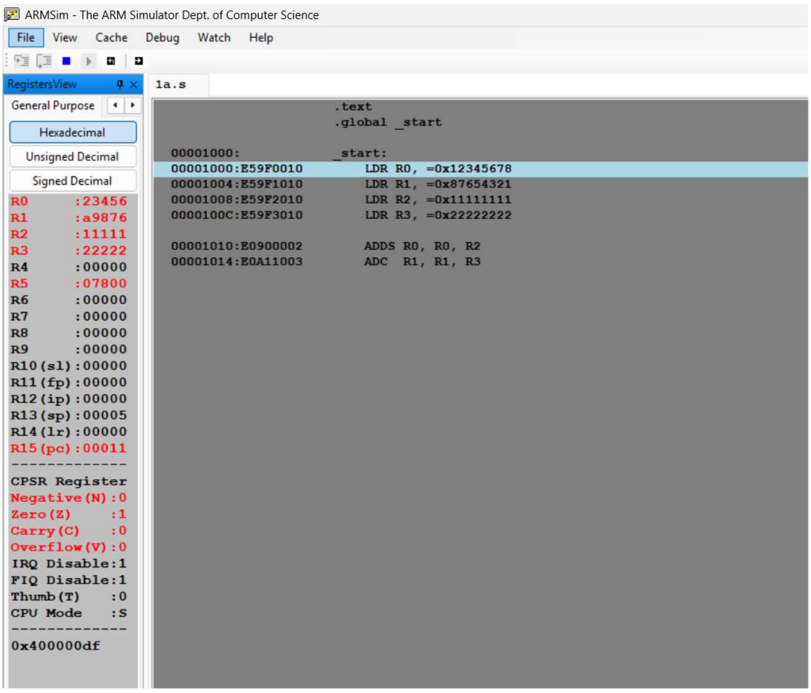
1 Write an ALP to perform Addition for of two numbers of size
a)64 bit
b)128 bit
save the result in register (reuse the register to store the result)

Hint: For 128 bit given below

```
  R3 R2 R1 R0
 +R7 R6 R5 R4
```

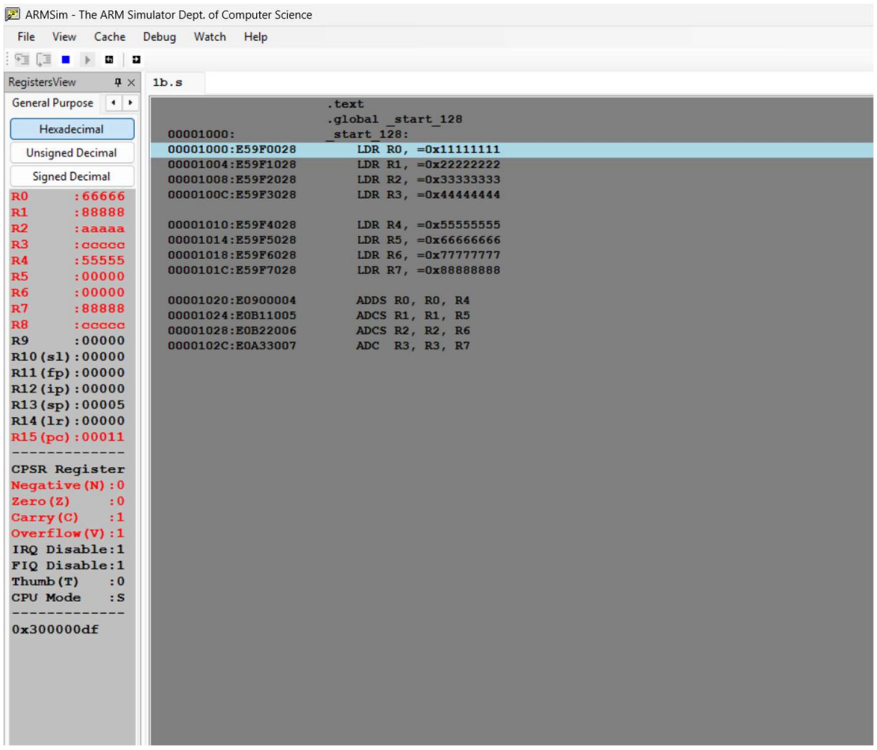
a)

```
.text
.global _start

_start:
   LDR R0, =0x12345678
   LDR R1, =0x87654321
   LDR R2, =0x11111111
   LDR R3, =0x22222222

   ADDS R0, R0, R2
   ADC  R1, R1, R3
```



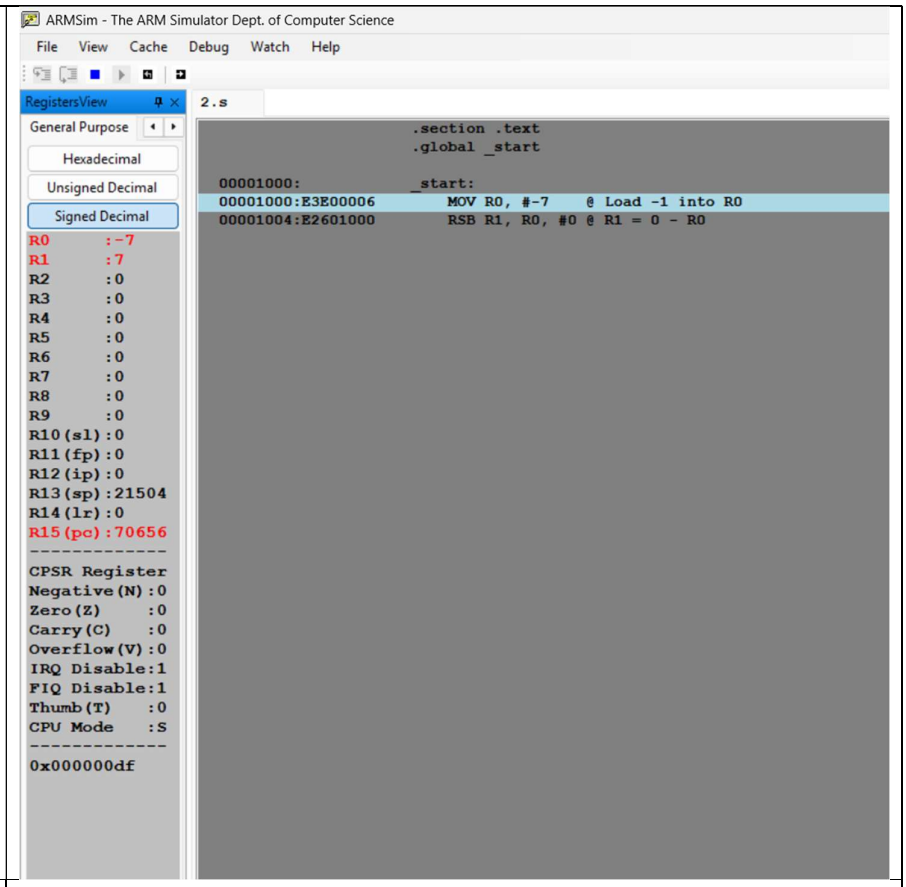
b)

```
.text
.global _start_128
_start_128:
   LDR R0, =0x11111111
   LDR R1, =0x22222222
   LDR R2, =0x33333333
   LDR R3, =0x44444444

   LDR R4, =0x55555555
   LDR R5, =0x66666666
   LDR R6, =0x77777777
   LDR R7, =0x88888888

   ADDS R0, R0, R4
   ADCS R1, R1, R5
   ADCS R2, R2, R6
   ADC  R3, R3, R7
```



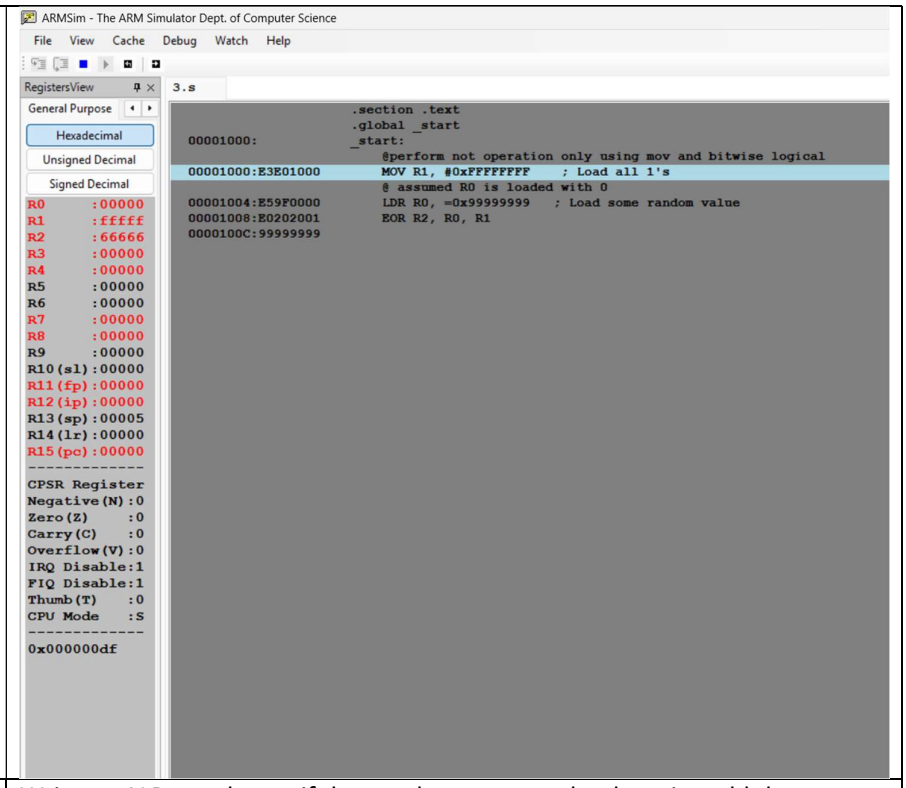
2 Write an ALP to perform 2’complement only using mov and RSB instruction.

```
.section .text
.global _start

_start:
   MOV R0, #-7    @ Load -1 into R0
   RSB R1, R0, #0 @ R1 = 0 - R0
```



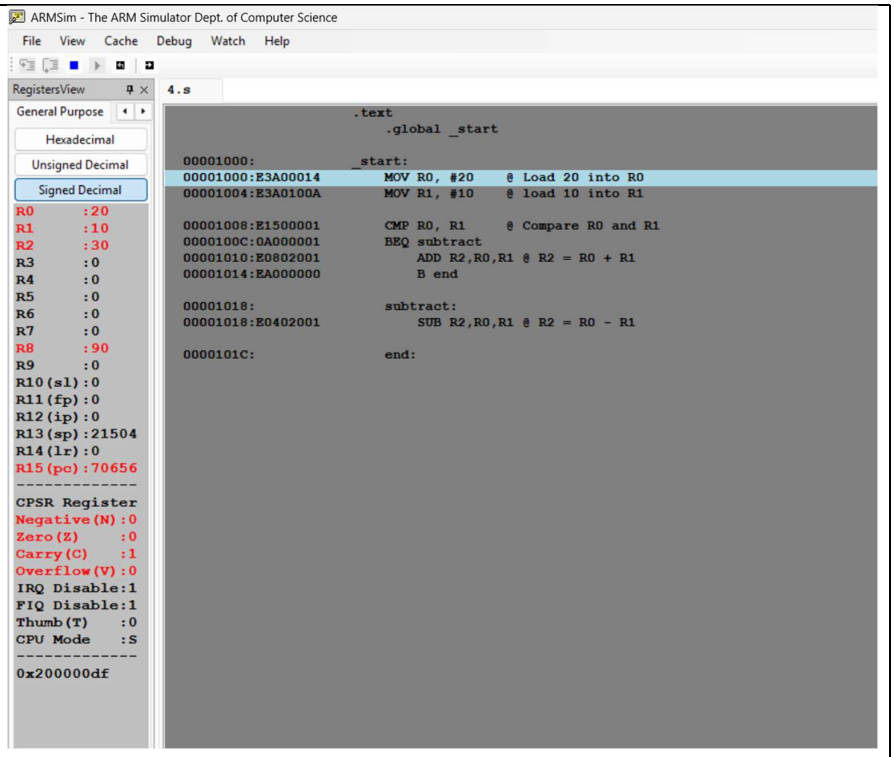
3 Write an ALP to perform not operation only using mov and bitwise logical instructions.

```
.section .text
.global _start
_start:
   @perform not operation only using mov and bitwise logical
   MOV R1, #0xFFFFFFFF    ; Load all 1's
   @ assumed R0 is loaded with 0
   LDR R0, =0x99999999   ; Load some random value
   EOR R2, R0, R1
```



4 Write an ALP to subtract if the numbers are equal, otherwise add them.

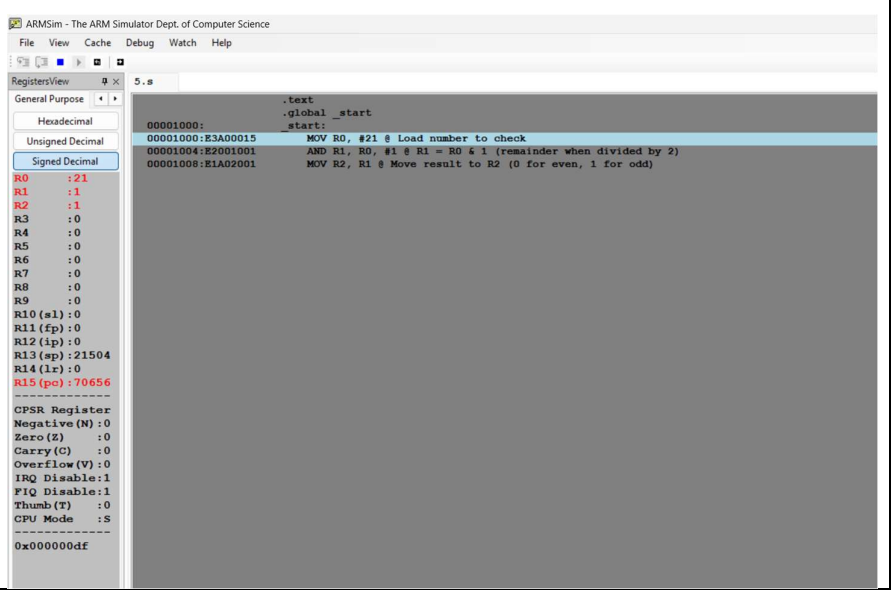
```
.text
   .global _start

_start:
   MOV R0, #20    @ Load 20 into R0
   MOV R1, #10    @ load 10 into R1

   CMP R0, R1     @ Compare R0 and R1
   BEQ subtract
      ADD R2,R0,R1 @ R2 = R0 + R1
      B end

   subtract:
      SUB R2,R0,R1 @ R2 = R0 - R1

   end:
```



5 Check if a given number is even or odd.

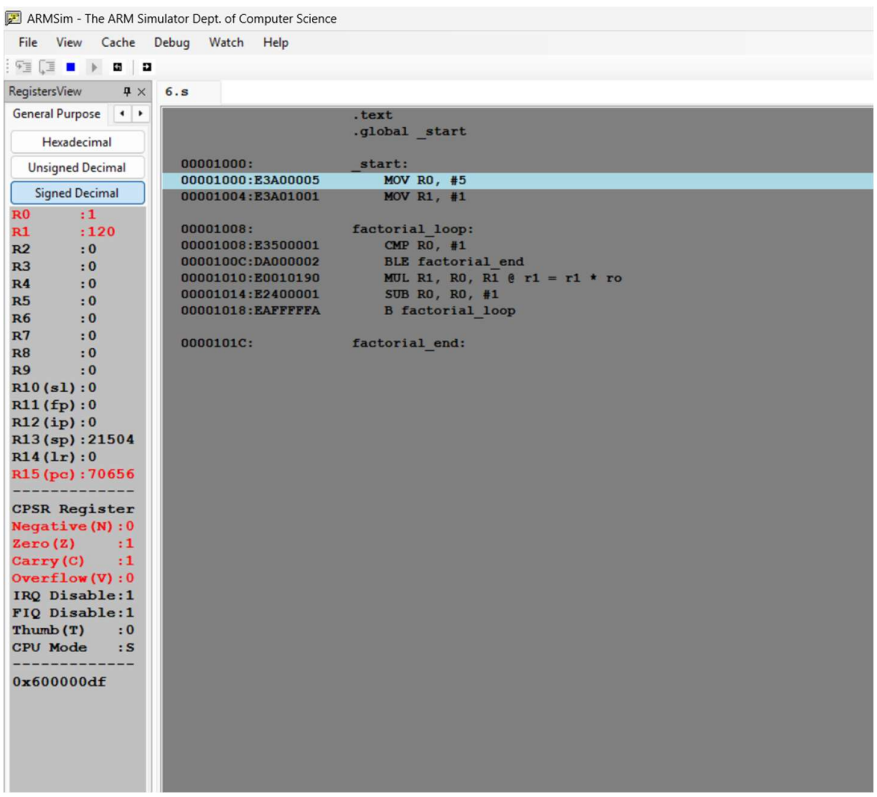
Note: at the end of the program execution R2 contains 0 if number is even, otherwise R2 contains 1.

```
.text
.global _start
_start:
   MOV R0, #21 @ Load number to check
   AND R1, R0, #1 @ R1 = R0 & 1 (remainder when divided by 2)
   MOV R2, R1 @ Move result to R2 (0 for even, 1 for odd)
```



6 Write a program to find the factorial of a given number.

```
.text
.global _start

_start:
   MOV R0, #5
   MOV R1, #1

factorial_loop:
   CMP R0, #1
   BLE factorial_end
   MUL R1, R0, R1 @ r1 = r1 * ro
   SUB R0, R0, #1
   B factorial_loop

factorial_end:
```



7 Write a program to find GCD of two numbers.

```
@to find gcd of 2 numbers
.text
.global _start

_start:
   MOV R0, #80   @ Load 20 into R0
   MOV R1, #90  @ load 10 into R1
```

```
gcd_loop:
   CMP R0, #0     @ Compare R0 and R1
   BEQ gcd_end

   CMP R0,R1
   BLT swap
   SUB R0,R0,R1
   B gcd_loop

swap:
MOV R2,R0
MOV R0,R1
MOV R1,R2
B gcd_loop

gcd_end:
```