PES UNIVERSITY

Department of Computer Science & Engineering
Microprocessor & Computer Architecture Lab

Lab 5

UE22CS251B

Name: ANKITH GOWDA B S **SRN:** PES2UG22CS077 **SECTION:** B

1 Write an ALP to multiply 2 matrices.

Program:

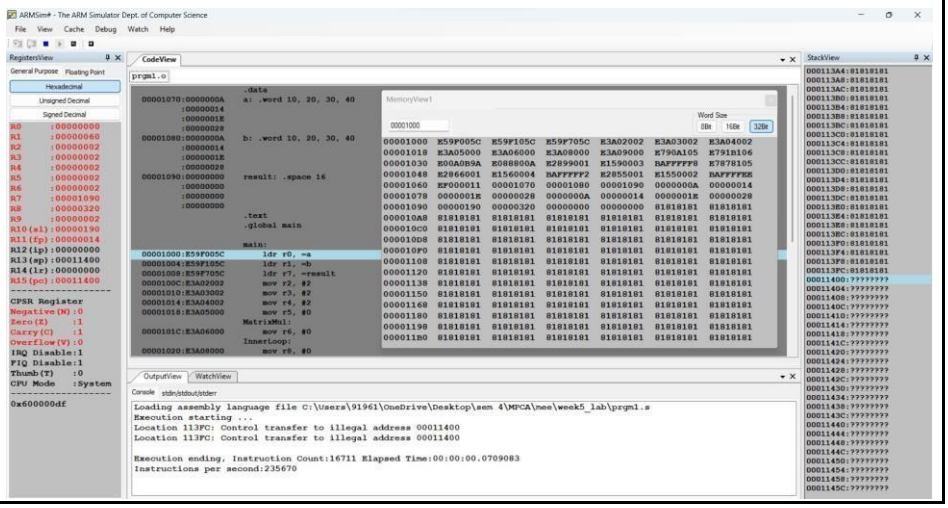
prgm1.s

week5_lab > prgm1.s

```
.data
a: .word 10, 20, 30, 40
b: .word 10, 20, 30, 40
result: .space 16
.text
.global main

main:
    ldr r0, =a
    ldr r1, =b
    ldr r7, =result
    mov r2, #2
    mov r3, #2
    mov r4, #2
    mov r5, #0
MatrixMul:
    mov r6, #0
InnerLoop:
    mov r8, #0
    mov r9, #0
InnerMul:
    ldr r10, [r0, r5, LSL #2]
    ldr r11, [r1, r6, LSL #2]
    mul r10, r10, r11
    add r8, r8, r10
    add r9, r9, #1
    cmp r9, r3
    blt InnerMul
    str r8, [r7, r5, LSL #2]
    add r6, r6, #1
    cmp r6, r4
    blt InnerLoop
    add r5, r5, #1
    cmp r5, r2
    blt MatrixMul
    SWI 0x11
```

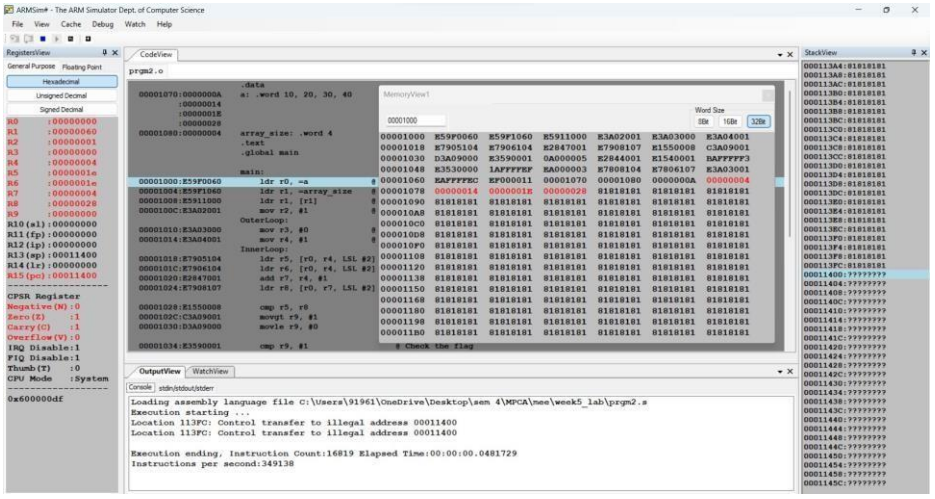
Screenshot:



2	Write an ALP using conditional ARM instructions to sort an array of numbers using Bubble Sort Algorithm. Program: prgm1.s prgm2.s prgm3_1.s week5_lab > prgm2.s

```
1   .data
2   a: .word 10, 20, 30, 40
3   array_size: .word 4
4   .text
5   .global main
6
7   main:
8       ldr r0, =a
9       ldr r1, =array_size
10      ldr r1, [r1]
11      mov r2, #1
12  OuterLoop:
13      mov r3, #0
14      mov r4, #1
15  InnerLoop:
16      ldr r5, [r0, r4, LSL #2]
17      ldr r6, [r0, r4, LSL #2]
18      add r7, r4, #1
19      ldr r8, [r0, r7, LSL #2]
20
21      cmp r5, r8
22      movgt r9, #1
23      movle r9, #0
24      cmp r9, #1
25      beq Swap
26
27      add r4, r4, #1
28      cmp r4, r1
29      blt InnerLoop
30      cmp r3, #0
31      bne OuterLoop
32
33      b BubbleSortEnd
34
35  Swap:
36      str r8, [r0, r4, LSL #2]
37      str r6, [r0, r7, LSL #2]
38      mov r3, #1
39      b InnerLoop
40
41  BubbleSortEnd:
42      SWI 0x11
```

Screenshot:



3	**Assignment:** i) Write a program to swap the first and last character of a given string. Example: Input: ‘dog’ Output: ‘god’ Program:

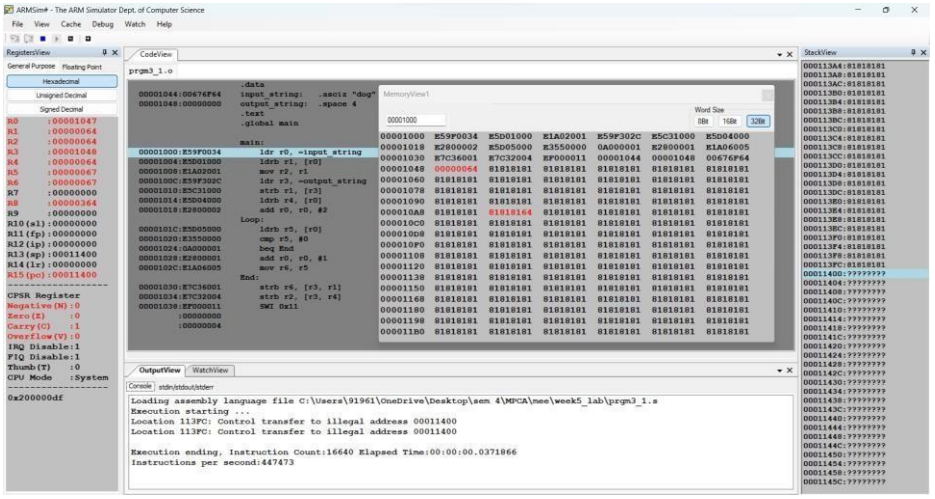
prgm1.s | prgm2.s | prgm3_1.s

week5_lab > prgm3_1.s

```
.data
input_string:   .asciz "dog"      @ Input string
output_string:  .space 4          @ Space for output string (max 4 characters including null terminator)
.text
.global main

main:
    ldr r0, =input_string      @ Load address of input string into r0
    ldrb r1, [r0]              @ Load first character of input string into r1
    mov r2, r1                 @ Copy first character to r2
    ldr r3, =output_string     @ Load address of output string into r3
    strb r1, [r3]              @ Store first character of input string into output string
    ldrb r4, [r0]              @ Load first character of input string into r4
    add r0, r0, #2             @ Move pointer to second character of input string
Loop:
    ldrb r5, [r0]              @ Load character from input string into r5
    cmp r5, #0                 @ Check if end of string
    beq End                    @ If end of string, exit loop
    add r0, r0, #1             @ Move pointer to next character
    mov r6, r5                 @ Copy current character to r6
End:
    strb r6, [r3, r1]          @ Store last character of input string into output string
    strb r2, [r3, r4]          @ Store copied first character of input string into output string
    SWI 0x11                    @ Exit the program
```

Screenshot:



ii)
Given a c Code convert it in its equivalent Arm Code.

a)x = (a + b) - c;

Program:

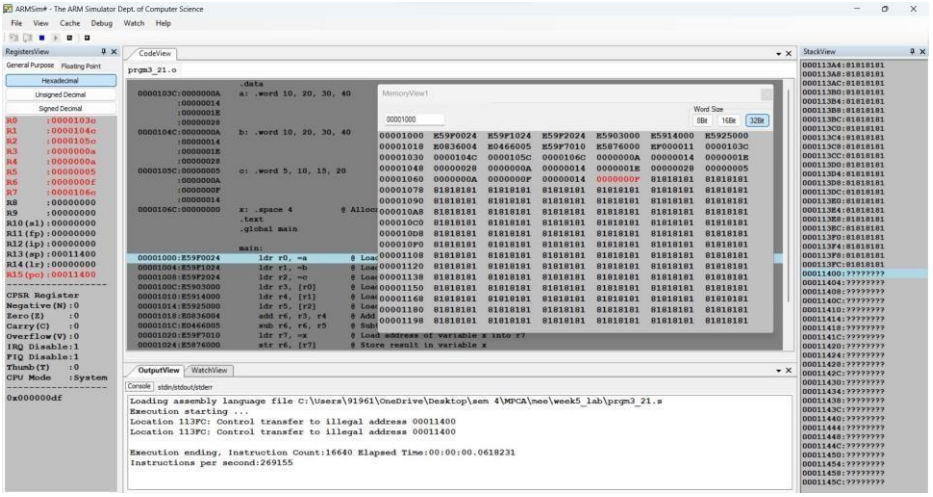
prgm1.s | prgm2.s | prgm3_1.s | prgm3_21.s

week5_lab > prgm3_21.s

```
.data
a: .word 10, 20, 30, 40
b: .word 10, 20, 30, 40
c: .word 5, 10, 15, 20
x: .space 4          @ Allocate 4 bytes for variable x
.text
.global main

main:
    ldr r0, =a       @ Load address of variable a into r0
    ldr r1, =b       @ Load address of variable b into r1
    ldr r2, =c       @ Load address of variable c into r2
    ldr r3, [r0]     @ Load value of variable a into r3
    ldr r4, [r1]     @ Load value of variable b into r4
    ldr r5, [r2]     @ Load value of variable c into r5
    add r6, r3, r4   @ Add a and b, store result in r6
    sub r6, r6, r5   @ Subtract c from result, store final result in r6
    ldr r7, =x       @ Load address of variable x into r7
    str r6, [r7]     @ Store result in variable x
    SWI 0x11         @ Exit the program
```

Screenshot:



b)

z = (a << 2) |(b & 15);

Program:

prgm1.s | prgm2.s | prgm3_1.s | prgm3_21.s | prgm3_22.s

week5_lab > prgm3_22.s

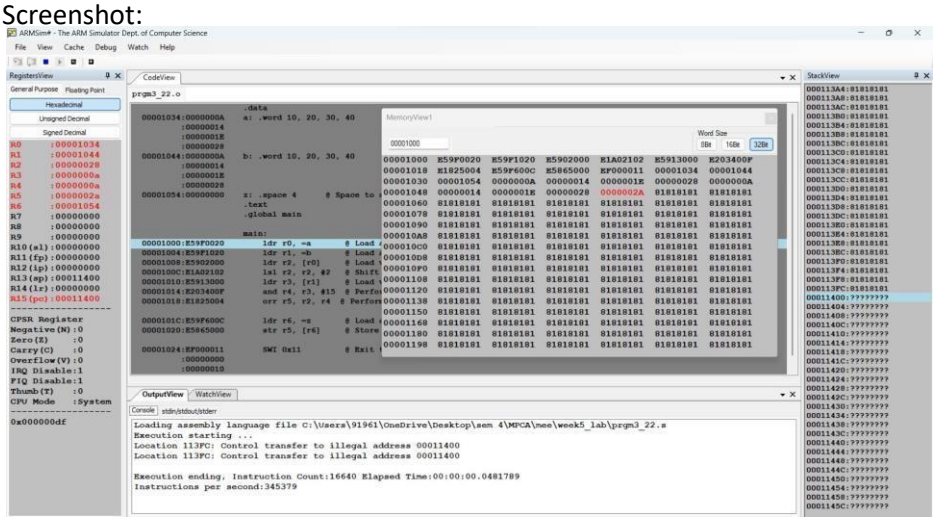
```
.data
a: .word 10, 20, 30, 40
b: .word 10, 20, 30, 40
z: .space 4      @ Space to store the result
.text
.global main

main:
    ldr r0, =a       @ Load address of 'a' into r0
    ldr r1, =b       @ Load address of 'b' into r1
    ldr r2, [r0]     @ Load value of 'a' into r2
    lsl r2, r2, #2   @ Shift left r2 by 2 (equivalent to a << 2)
    ldr r3, [r1]     @ Load value of 'b' into r3
    and r4, r3, #15  @ Perform bitwise AND of r3 with 15 (equivalent to b & 15)
    orr r5, r2, r4  @ Perform bitwise OR of r2 and r4 (equivalent to (a << 2) | (b & 15))

    ldr r6, =z       @ Load address of 'z' into r6
    str r5, [r6]     @ Store the result in memory location 'z'

    SWI 0x11         @ Exit the program
```

Screenshot:


ARMSim# - The ARM Simulator Dept. of Computer Science
File View Cache Debug Watch Help
RegistersView
General Purpose Floating Point
Hexadecimal
Unsigned Decimal
Signed Decimal
R0 :00001034
R1 :00001044
R2 :00000028
R3 :0000000a
R4 :0000000a
R5 :0000002a
R6 :00001054
R7 :00000000
R8 :00000000
R9 :00000000
R10(sl):00000000
R11(fp):00000000
R12(ip):00000000
R13(sp):00011400
R14(lr):00000000
R15(pc):00011400
CPSR Register
Negative(N):0
Zero(Z) :0
Carry(C) :0
Overflow(V):0
IRQ Disable:1
FIQ Disable:1
Thumb(T) :0
CPU Mode :System
0x000000df
CodeView
prgm3_22.o
MemoryView1
Word Size
8Bit 16Bit 32Bit
StackView
OutputView WatchView
Console stdin/stdout/stderr
Loading assembly language file C:\Users\91961\OneDrive\Desktop\sem 4\MPCA\mee\week5_lab\prgm3_22.s
Execution starting ...
Location 113FC: Control transfer to illegal address 00011400
Location 113FC: Control transfer to illegal address 00011400
Execution ending, Instruction Count:16640 Elapsed Time:00:00:00.0481789
Instructions per second:345379