

# Microprocessor and Computer Architecture

**UE21CS251B**

**4th Semester, Academic Year 2022-23**

Date:

Name: Nihal T M	SRN: PES2UG21CS333	Section: F
-----------------	--------------------	------------

Week#   3    
  1  

Program Number:

Title of the Program

**Generate Fibonacci Series and store them in an array.**

I. ARM Assembly Code

code:

@ generate fibonacci series and store them in an array

.text

ldr r2,=fib

mov r0,#0

mov r1,#1

str r0,[r2]

add r2,r2,#4

str r1,[r2]

```

mov r4,#8
loop:
    add r3,r0,r1
    add r2,r2,#4
    str r3,[r2]
    mov r0,r1
    mov r1,r3
    subs r4,r4,#1
    bne loop

```

.data

```
fib:.word 0,0,0,0,0,0,0,0,0,0
```

## II. Output Screen Shots (One)

Output is along memory location:  
0000112c

The screenshot shows a debugger window with the following components:

- General Purpose:** Floating point, 1.s, 2.s, 3a.s, 3b.s, 4.s, 5.s
- Hexadecimal:** Unsigned Decimal, Signed Decimal
- Registers:** R0 to R15 (PC) are listed. R15 (PC) is highlighted at 00001034.
- CPSR Register:** Negative(N):0, Zero(Z):1, etc.
- Assembly Code:**

```

@ generate fibonacci series and store them in an array
.text
00001000:E59F2030    ldr r2,=fib
00001004:E3A00000    mov r0,#0
00001008:E3A01001    mov r1,#1
0000100C:E5820000    str r0,[r2]
00001010:E2822004    add r2,r2,#4
00001014:E5821000    str r1,[r2]
00001018:E3A04008    mov r4,#8
0000101C:           loop:
0000101C:E0803001    add r3,r0,r1
00001020:E2822004    add r2,r2,#4
00001024:E5823000    str r3,[r2]
00001028:E1A00001    mov r0,r1
0000102C:E1A01003    mov r1,r3
00001030:E2544001    subs r4,r4,#1
00001034:1AFFFFF8    bne loop

.data
0000112C:           fib:.word 0,0,0,0,0,0,0,0,0,0

```
- MemoryView:** Shows a memory dump starting at 0000112C. The first row shows 0000112C to 00001133 with values 00000000, 00000001, 00000001, 00000002, 00000003, 00000005, 00000008, 0000000D, 00000015, 00000022.

Week#   3    
  2  

Program Number:

Title of the Program

**Write an ALP to find smallest number in an array of n 32-bit numbers**

I. ARM Assembly Code

code:

@ find the smallest nummber in an array os 32 bit numbers

.text

ldr r4,=arr

ldr r1,[r4],#4

mov r2,#9

loop:

ldr r3,[r4]

cmp r3,r1

bmi swap

add r4,r4,#4

subs r2,r2,#1

bne loop

b end

swap:

mov r1,r3

b loop

end:

mov r0,r1

swi 0x02

swi 0x00

.data

arr:.word 5,2,5,3,10,7,9,5,1,2

## II. Output Screen Shots (One)

data array present in 000010f0 memory:

1 is the smallest element in the array is stored in r0 at the end of the program.

The screenshot displays a debugger window with two main panes. The top pane shows assembly code with addresses, hex values, and mnemonics. The bottom pane shows a memory view of the array 'arr' starting at address 000010f0.

**Assembly Code:**

```
.text
00001000:E59F4034    ldr r4,=arr
00001004:E4941004    ldr r1,[r4],#4
00001008:E3A02009    mov r2,#9
0000100C:                loop:
0000100C:E5943000    ldr r3,[r4]
00001010:E1530001    cmp r3,r1
00001014:4A000003    bmi swap
00001018:E2844004    add r4,r4,#4
0000101C:E2522001    subs r2,r2,#1
00001020:1AFFFFF9    bne loop
00001024:EA000001    b end

00001028:                swap:
00001028:E1A01003    mov r1,r3
0000102C:EAF00006    b loop

00001030:                end:
00001030:E1A00001    mov r0,r1
00001034:EF000002    swi 0x02
00001038:EF000000    swi 0x00

.data
000010F0:                arr:.word 5,2,5,3,10,7,9,5,1,2
```

**Memory View:**

Address	Value
000010F0	00000005
000010F1	00000002
000010F2	00000005
000010F3	00000003
000010F4	0000000A
000010F5	00000007
000010F6	00000009
000010F7	00000005
000010F8	00000006
000010F9	00000001
000010FA	00000002

Week#     3      
    3    

Program Number:

Title of the Program

**To perform Convolution using MUL instruction (Addition of multiplication of respective numbers of loc A and loc B)**

I. ARM Assembly Code

code:

@ Perform convolution using mul instruction

.text

ldr r0,=a

ldr r1,=b

mov r2,#6

loop:

ldr r3,[r0],#4

ldr r4,[r1],#4

mul r5,r3,r4

add r6,r5,r6

subs r2,r2,#1

bne loop

swi 0x11

.data

a:.word 1,2,3,4,5,6

b:.word 1,2,3,4,5,6

## II. Output Screen Shot (One)

output is present in r6 register and is displayed in unsigned decimal to enable ease of correction

The screenshot displays a debugger window with the following components:

- General Purpose** tab selected, showing register values in **Unsigned Decimal** format.
- Registers:**
  - R0: 4296
  - R1: 4320
  - R2: 0
  - R3: 6
  - R4: 6
  - R5: 36
  - R6: 91
  - R7: 0
  - R8: 0
  - R9: 0
  - R10(sl): 0
  - R11(fp): 0
  - R12(ip): 0
  - R13(sp): 21504
  - R14(lr): 0
  - R15(pc): 4132
- CPSR Register:** Negative(N): 0
- Assembly View:**
  - Comment: @ Perform convolution using mul instruction
  - Code:

```
.text
00001000:E59F0020    ldr r0,=a
00001004:E59F1020    ldr r1,=b
00001008:E3A02006    mov r2,#6
0000100C:           loop:
0000100C:E4903004    ldr r3,[r0],#4
00001010:E4914004    ldr r4,[r1],#4
00001014:E0050493    mul r5,r3,r4
00001018:E0856006    add r6,r5,r6
0000101C:E2522001    subs r2,r2,#1
00001020:1AFFFFF9    bne loop
00001024:EF000011    swi 0x11

.data
000010B0:           a: .word 1,2,3,4,5,6
000010C8:           b: .word 1,2,3,4,5,6
```
- Memory View:**
  - Address: 000010b0
  - Memory dump:

000010B0	00000001	00000002	00000003	00000004	00000005	00000006
000010E8	00000003	00000004	00000005	00000006	00000001	00000002

Week#   3    
  4  

Program Number:

Title of the Program

**To perform Convolution using MLA instruction (Addition of multiplication of respective numbers of loc A and loc B).**

I. ARM Assembly Code

code:

@ doing the same as 3a but by using mla instruction

.text

ldr r0,=a

ldr r1,=b

mov r2,#6

loop:

ldr r3,[r0],#4

ldr r4,[r1],#4

mla r5,r3,r4,r5

subs r2,r2,#1

bne loop

swi 0x11

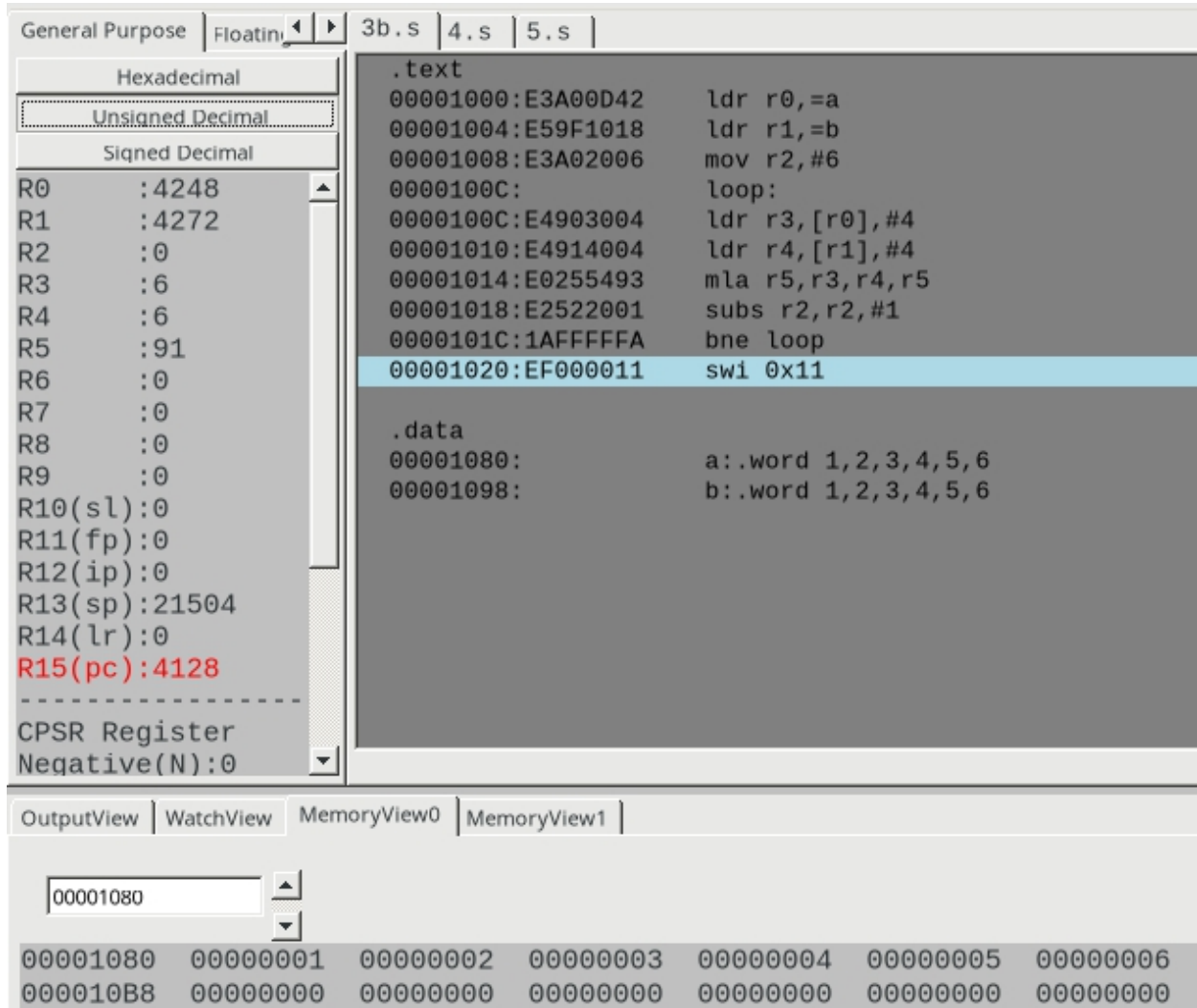
.data

a:.word 1,2,3,4,5,6

b:.word 1,2,3,4,5,6

## II. Output Screen Shot (One)

output is present in r5 register and is displayed in decimal to enable ease of correction.





Week# 3  
5

Program Number:

Title of the Program

**Write an ALP to find mul (add( a,b),c)**

I. ARM Assembly Code

code:

@ program to find mul(add(a,b),c)

.data

a:.word 10

b:.word 15

c:.word 7

res:.word 0

.text

ldr r1,=a

ldr r2,=b

ldr r3,=c

ldr r4,[r1]

ldr r5,[r2]

ldr r6,[r3]

add r7,r4,r5

mul r8,r7,r6

ldr r9,=res

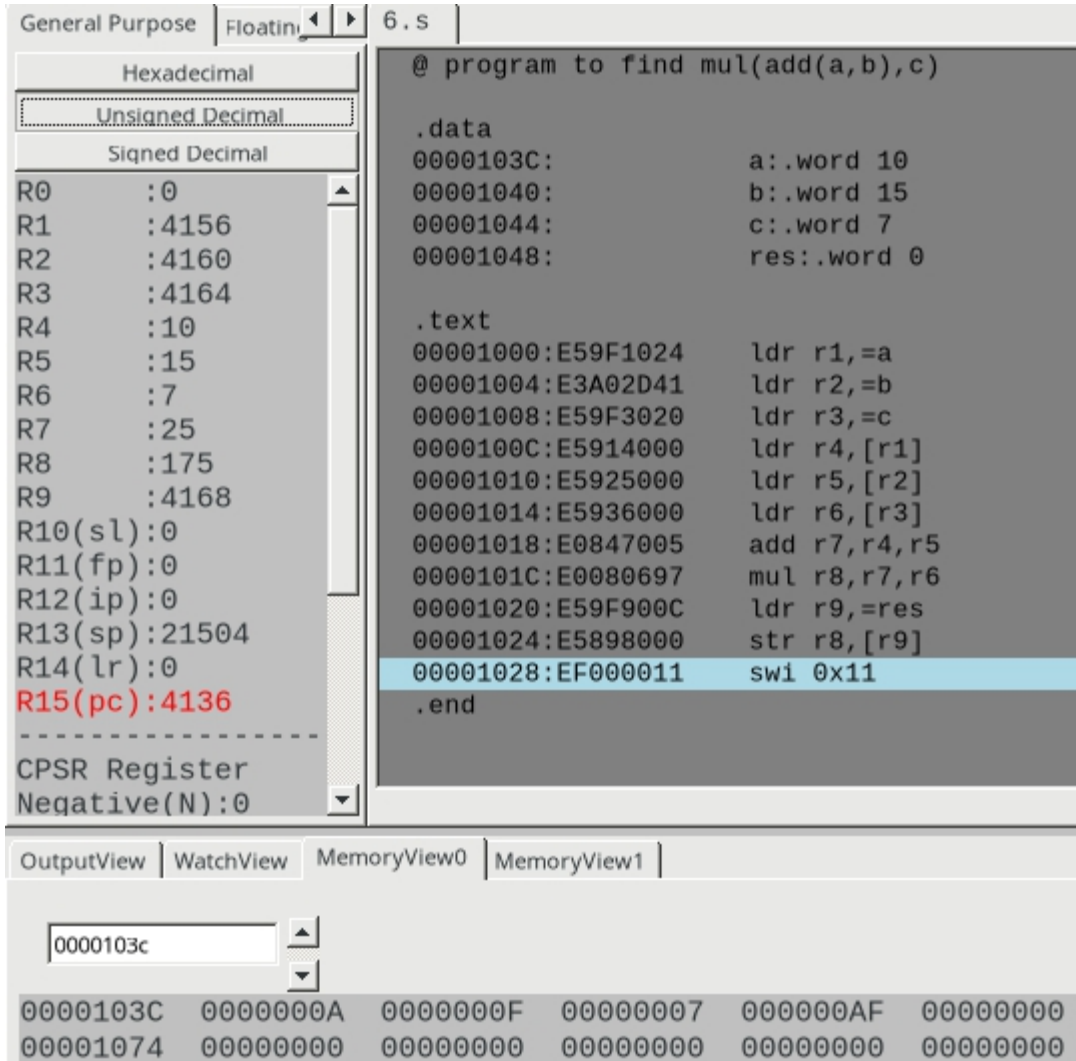
str r8,[r9]

swi 0x11

.end

## II. Output Screen Shot (One)

output is present in r8 register and is displayed in decimal. It is also stored into memory.



Week#   3    
  6  

Program Number:

Title of the Program

**Write an ALP to find factorial using subroutine**

I. ARM Assembly Code

code:

@ Find the factorial of a number using subroutine

.text

ldr r0,=n

mov r1,#1

ldr r3,[r0]

bl fact

swi 0x11

fact:

mov r2,r1

mul r1,r2,r3

subs r3,r3,#1

bne fact

mov pc,lr

.data

n:.word 5

## II. Output Screen Shot (One)

output is present in r1 register and it is in decimal:

The screenshot displays a debugger window with the following components:

- General Purpose** tab is selected. The **Floatin** dropdown is set to **4.s** and **5.s**.
- Hexadecimal** tab is selected for the register view.
- Unsigned Decimal** tab is selected for the CPSR Register view.
- Signed Decimal** tab is selected for the Negative(N) view.
- Registers:** R0: 4180, R1: 120, R2: 120, R3: 0, R4: 0, R5: 0, R6: 0, R7: 0, R8: 0, R9: 0, R10(sl): 0, R11(fp): 0, R12(ip): 0, R13(sp): 21504, R14(lr): 4112, R15(pc): 4112 (highlighted in red).
- CPSR Register:** Negative(N): 0.
- Assembly Code:** The code is for finding the factorial of a number using a subroutine. The current instruction is `swi 0x11` at address `00001010:EF000011`.
- Memory View:** The **MemoryView0** tab is selected, showing a memory dump starting at address `00001054`.

```
@ Find the factorial of a number using subroutine

.text
00001000:E59F0020    ldr r0,=n
00001004:E3A01001    mov r1,#1
00001008:E5903000    ldr r3,[r0]
0000100C:EB000000    bl fact
00001010:EF000011    swi 0x11

00001014:          fact:
00001014:E1A02001    mov r2,r1
00001018:E0010392    mul r1,r2,r3
0000101C:E2533001    subs r3,r3,#1
00001020:1AFFFFFB    bne fact
00001024:E1A0F00E    mov pc,lr

.data
00001054:          n:.word 5

MemoryView0
00001054  00000005  0000002F  00000000  00000000  00000000  00000000
0000108C  00000000  00000000  00000000  00000000  00000000  00000000
```

Week#   3    
  7  

Program Number:

Title of the Program

**Write an ALP to perform multiplication using shift method (without using MUL)**

I. ARM Assembly Code

code:

@ Perform multiplication using shift method without using the mul operation

@ here let us multiply the data num with the number 135

.text

```
ldr r0,=num
ldr r1,[r0]
rsb r2,r1,r1,LSL #3
mov r3,r1,LSL #7
add r4,r2,r3
ldr r5,=res
str r4,[r5]
swi 0x11
```

.data

```
num:.word 47
res:.word 0
```

## II. Output Screen Shot (One)

output is present in r4 register and is displayed in decimal

The screenshot displays a debugger window with the following components:

- General Purpose Register View:** A list of registers (R0 to R15) and the CPSR register. R15 (PC) is highlighted in red and shows the value 4124. The CPSR register shows Negative(N):0.
- Assembly View:** A list of assembly instructions with their addresses and hex values. The instruction at address 0000101C is highlighted in blue: `swi 0x11`.
- OutputView:** A section at the bottom showing memory addresses and their corresponding values. The first row shows address 00001028 with value 0000002F. The second row shows address 00001060 with value 00000000.

```
@ Perform multiplication using shift method without using the mul operation
@ here let us multiply the data num with the number 135

.text
00001000:E59F0018    ldr r0,=num
00001004:E5901000    ldr r1,[r0]
00001008:E0612181    rsb r2,r1,r1,LSL #3
0000100C:E1A03381    mov r3,r1,LSL #7
00001010:E0824003    add r4,r2,r3
00001014:E59F5008    ldr r5,=res
00001018:E5854000    str r4,[r5]
0000101C:EF000011    swi 0x11

.data
00001028:                num:.word 47
0000102C:                res:.word 0
```

Address	Value
00001028	0000002F
00001060	00000000

### **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:

Name: Nihal T M

SRN: PES2UG21CS333

Section: F

Date: 31/01/2023