

# **Microprocessor and Computer Architecture**

## **UE21CS251B**

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

Date:

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

Week# \_\_\_\_2\_\_\_\_  
\_\_\_\_1\_\_\_\_

Program Number:

Title of the Program

**Write a program in ARM7TDMI-ISA to copy a block of N data items from Location A to Location B.**

- a. Use Full word (.word directive)**
- b. Use Half word(.hword directive)**
- c. Use Byte wise (.Byte directive)**

I. ARM Assembly Code

Code:

@ this program copies a block of n data(word) items from location A to location B

```
.text
ldr r0,=a
ldr r1,=b
ldmia r0!,{r2-r6}
```

```
stmia r1!,{r2-r6}
swi 0x11
```

.data

```
a:.word 10,20,30,40,50
b:.word 0,0,0,0,0
```

## II. Output Screen Shots (Three)

The output should be verified for word, half word, byte

i. word:

The screenshot displays a debugger window with the following components:

- Register List:** Shows registers R0 through R15. R15 (PC) is highlighted in red with the value 00001010.
- Assembly View:** Displays the following code:
 

```
@ this program copies a block of n data(word) items from location A to location B
.text
00001000:E3A00D47    ldr r0,=a
00001004:E59F1008    ldr r1,=b
00001008:E8B0007C    ldmbia r0!,{r2-r6}
0000100C:E8A1007C    stmbia r1!,{r2-r6}
00001010:EF000011    swi 0x11

.data
000011C0:          a:.word 10,20,30,40,50
000011D4:          b:.word 0,0,0,0,0
```
- Memory View:** Shows a memory dump starting at address 000011C0. The first row of data is highlighted in red:
 

000011C0	0000000A	00000014	0000001E	00000028	00000032	0000000A	00000014	0000001E	00000028	00000032
----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------

ii. half-word

code:

@ this program copies a block of n data(half-word)  
items from location A to location B

.text

```
ldr r0,=a
ldr r1,=b
ldmia r0!,{r2-r4}
stmia r1!,{r2-r4}
swi 0x11
```

.data

```
a:.hword 10,20,30,40,50,60
b:.hword 0,0,0,0,0,0
```

The screenshot shows a debugger window with the following components:

- General Purpose** tab selected, showing registers R0 through R15. R15 (PC) is highlighted in red.
- Hexadecimal** tab selected, showing the assembly code and its memory addresses.
- Assembly Code:**

```
@ this program copies a block of n data(half-word) items from location A to location B

.text
00001000:E59F000C    ldr r0,=a
00001004:E59F100C    ldr r1,=b
00001008:E8B0001C    ldmia r0!,{r2-r4}
0000100C:E8A1001C    stmia r1!,{r2-r4}
00001010:EF000011    swi 0x11

.data
000011A4:          a:.hword 10,20,30,40,50,60
000011B0:          b:.hword 0,0,0,0,0,0
```
- MemoryView0** tab selected, showing a memory dump starting at address 000011A4. The dump shows the data stored in memory, including the values 10, 20, 30, 40, 50, and 60.

iii. byte

code:

@ this program copies a block of n data(byte) items from location A to location B

.text

```
ldr r0,=a
ldr r1,=b
ldmia r0!,{r2-r3}
stmia r1!,{r2-r3}
swi 0x11
```

.data

a:.byte 10,20,30,40,50,60,70,80

b:.byte 0,0,0,0,0,0,0,0

The screenshot shows a debugger window with the following components:

- Registers:** A list of registers on the left. R15 (PC) is highlighted in red and shows the address 00001010.
- Disassembly:** The main pane shows assembly code for the program. The instruction at address 00001010, `swi 0x11`, is highlighted in blue.
- Memory View:** At the bottom, a memory dump is shown for address 0000101c. It displays a sequence of bytes: 0A 14 1E 28 32 3C 46 50, followed by 0A 14 1E 28 32 3C 46 50, and then a series of 81 bytes.

Week# \_\_\_\_2\_\_\_\_  
\_\_\_\_2\_\_\_\_

Program Number:

Title of the Program

**Write a program in ARM7TDMI-ISA to find the sum of N data items in the memory. Store the result in the memory location.**

- a. Use Full word (.word directive)**
- b. Use Half word(.hword directive)**
- c. Use Byte wise (.Byte directive)**

I.ARM Assembly Code

Code:

i. word:

@ find sum of n words

.text

ldr r0,=a

ldr r1,=sum

mov r2,#5

mov r3,#0

loop:

ldr r4,[r0]

add r3,r3,r4

add r0,r0,#4

subs r2,r2,#1

```

        bne loop
        str r3,[r1]
.data
a:.word 10,20,30,40,50
sum:.word 0

```

ii. half-word:

@ find sum of n half-words

```

.text
    ldr r0,=a
    ldr r1,=sum
    mov r2,#5
    mov r3,#0
loop:
    ldrrh r4,[r0]
    add r3,r3,r4
    add r0,r0,#2
    subs r2,r2,#1
    bne loop
    strh r3,[r1]
.data
a:.hword 10,20,30,40,50
sum:.hword 0

```

iii. byte:

@ find sum of n bytes

.text

ldr r0,=a

ldr r1,=sum

mov r2,#5

mov r3,#0

loop:

ldrb r4,[r0]

add r3,r3,r4

add r0,r0,#1

subs r2,r2,#1

bne loop

strb r3,[r1]

.data

a:.byte 10,20,30,40,50

sum:.byte 0

## II. Output Screen Shots (Three)

The output should be verified for word, half word, byte

i. word:

The screenshot displays a debugger window with the following components:

- General Purpose** tab selected, showing registers R0 through R15 (PC) and CPSR Register. R15 (PC) is highlighted in red.
- Hexadecimal** view selected for the register list.
- Assembly View** showing the following code:

```
@ find sum of n words

.text
00001000:E59F0020    ldr r0,=a
00001004:E3A01D46    ldr r1,=sum
00001008:E3A02005    mov r2,#5
0000100C:E3A03000    mov r3,#0
00001010:                loop:
00001010:E5904000    ldr r4,[r0]
00001014:E0833004    add r3,r3,r4
00001018:E2800004    add r0,r0,#4
0000101C:E2522001    subs r2,r2,#1
00001020:1AFFFFFA    bne loop
00001024:E5813000    str r3,[r1]

.data
0000116C:                a:.word 10,20,30,40,50
00001180:                sum:.word 0
```
- MemoryView0** tab selected, showing a memory dump starting at address 0000116C. The value at 00001180 is highlighted in red.

Address	Value
0000116C	0000000A
00001170	00000000
00001174	00000000
00001178	00000000
0000117C	00000000
00001180	00000000
00001184	00000000
00001188	00000000
0000118C	00000000
00001190	00000000
00001194	00000000
00001198	00000000
0000119C	00000000
000011A0	00000000
000011A4	00000000
000011A8	00000000
000011AC	00000000
000011B0	00000000
000011B4	00000000
000011B8	00000000
000011BC	00000000
000011C0	00000000
000011C4	00000000
000011C8	00000000
000011CC	00000000
000011D0	00000000
000011D4	00000000
000011D8	00000000
000011DC	00000000
000011E0	00000000
000011E4	00000000
000011E8	00000000
000011EC	00000000
000011F0	00000000
000011F4	00000000
000011F8	00000000
000011FC	00000000
00001200	00000000
00001204	00000000
00001208	00000000
0000120C	00000000
00001210	00000000
00001214	00000000
00001218	00000000
0000121C	00000000
00001220	00000000
00001224	00000000
00001228	00000000
0000122C	00000000
00001230	00000000
00001234	00000000
00001238	00000000
0000123C	00000000
00001240	00000000
00001244	00000000
00001248	00000000
0000124C	00000000
00001250	00000000
00001254	00000000
00001258	00000000
0000125C	00000000
00001260	00000000
00001264	00000000
00001268	00000000
0000126C	00000000
00001270	00000000
00001274	00000000
00001278	00000000
0000127C	00000000
00001280	00000000
00001284	00000000
00001288	00000000
0000128C	00000000
00001290	00000000
00001294	00000000
00001298	00000000
0000129C	00000000
000012A0	00000000
000012A4	00000000
000012A8	00000000
000012AC	00000000
000012B0	00000000
000012B4	00000000
000012B8	00000000
000012BC	00000000
000012C0	00000000
000012C4	00000000
000012C8	00000000
000012CC	00000000
000012D0	00000000
000012D4	00000000
000012D8	00000000
000012DC	00000000
000012E0	00000000
000012E4	00000000
000012E8	00000000
000012EC	00000000
000012F0	00000000
000012F4	00000000
000012F8	00000000
000012FC	00000000
00001300	00000000
00001304	00000000
00001308	00000000
0000130C	00000000
00001310	00000000
00001314	00000000
00001318	00000000
0000131C	00000000
00001320	00000000
00001324	00000000
00001328	00000000
0000132C	00000000
00001330	00000000
00001334	00000000
00001338	00000000
0000133C	00000000
00001340	00000000
00001344	00000000
00001348	00000000
0000134C	00000000
00001350	00000000
00001354	00000000
00001358	00000000
0000135C	00000000
00001360	00000000
00001364	00000000
00001368	00000000
0000136C	00000000
00001370	00000000
00001374	00000000
00001378	00000000
0000137C	00000000
00001380	00000000
00001384	00000000
00001388	00000000
0000138C	00000000
00001390	00000000
00001394	00000000
00001398	00000000
0000139C	00000000
000013A0	00000000
000013A4	00000000
000013A8	00000000
000013AC	00000000
000013B0	00000000
000013B4	00000000
000013B8	00000000
000013BC	00000000
000013C0	00000000
000013C4	00000000
000013C8	00000000
000013CC	00000000
000013D0	00000000
000013D4	00000000
000013D8	00000000
000013DC	00000000
000013E0	00000000
000013E4	00000000
000013E8	00000000
000013EC	00000000
000013F0	00000000
000013F4	00000000
000013F8	00000000
000013FC	00000000
00001400	00000000
00001404	00000000
00001408	00000000
0000140C	00000000
00001410	00000000
00001414	00000000
00001418	00000000
0000141C	00000000
00001420	00000000
00001424	00000000
00001428	00000000
0000142C	00000000
00001430	00000000
00001434	00000000
00001438	00000000
0000143C	00000000
00001440	00000000
00001444	00000000
00001448	00000000
0000144C	00000000
00001450	00000000
00001454	00000000
00001458	00000000
0000145C	00000000
00001460	00000000
00001464	00000000
00001468	00000000
0000146C	00000000
00001470	00000000
00001474	00000000
00001478	00000000
0000147C	00000000
00001480	00000000
00001484	00000000
00001488	00000000
0000148C	00000000
00001490	00000000
00001494	00000000
00001498	00000000
0000149C	00000000
000014A0	00000000
000014A4	00000000
000014A8	00000000
000014AC	00000000
000014B0	00000000
000014B4	00000000
000014B8	00000000
000014BC	00000000
000014C0	00000000
000014C4	00000000
000014C8	00000000
000014CC	00000000
000014D0	00000000
000014D4	00000000
000014D8	00000000
000014DC	00000000
000014E0	00000000
000014E4	00000000
000014E8	00000000
000014EC	00000000
000014F0	00000000
000014F4	00000000
000014F8	00000000
000014FC	00000000
00001500	00000000
00001504	00000000
00001508	00000000
0000150C	00000000
00001510	00000000
00001514	00000000
00001518	00000000
0000151C	00000000
00001520	00000000
00001524	00000000
00001528	00000000
0000152C	00000000
00001530	00000000
00001534	00000000
00001538	00000000
0000153C	00000000
00001540	00000000
00001544	00000000
00001548	00000000
0000154C	00000000
00001550	00000000
00001554	00000000
00001558	00000000
0000155C	00000000
00001560	00000000
00001564	00000000
00001568	00000000
0000156C	00000000
00001570	00000000
00001574	00000000
00001578	00000000
0000157C	00000000
00001580	00000000
00001584	00000000
00001588	00000000
0000158C	00000000
00001590	00000000
00001594	00000000
00001598	00000000
0000159C	00000000
000015A0	00000000
000015A4	00000000
000015A8	00000000
000015AC	00000000
000015B0	00000000
000015B4	00000000
000015B8	00000000
000015BC	00000000
000015C0	00000000
000015C4	00000000
000015C8	00000000
000015CC	00000000
000015D0	00000000
000015D4	00000000
000015D8	00000000
000015DC	00000000
000015E0	00000000
000015E4	00000000
000015E8	00000000
000015EC	00000000
000015F0	00000000
000015F4	00000000
000015F8	00000000
000015FC	00000000
00001600	00000000
00001604	00000000
00001608	00000000
0000160C	00000000
00001610	00000000
00001614	00000000
00001618	00000000
0000161C	00000000
00001620	00000000
00001624	00000000
00001628	00000000
0000162C	00000000
00001630	00000000
00001634	00000000
00001638	00000000
0000163C	00000000
00001640	00000000
00001644	00000000
00001648	00000000
0000164C	00000000
00001650	00000000
00001654	00000000
00001658	00000000
0000165C	00000000
00001660	00000000
00001664	00000000
00001668	00000000
0000166C	00000000
00001670	00000000
00001674	00000000
00001678	00000000
0000167C	00000000
00001680	00000000
00001684	00000000
00001688	00000000
0000168C	00000000
00001690	00000000
00001694	00000000
00001698	00000000
0000169C	00000000
000016A0	00000000
000016A4	00000000
000016A8	00000000
000016AC	00000000
000016B0	00000000
000016B4	00000000
000016B8	00000000
000016BC	00000000
000016C0	00000000
000016C4	00000000
000016C8	00000000
000016CC	00000000
000016D0	00000000
000016D4	00000000
000016D8	00000000
000016DC	00000000
000016E0	00000000
000016E4	00000000
000016E8	00000000
000016EC	00000000
000016F0	00000000
000016F4	00000000
000016F8	00000000
000016FC	00000000
00001700	00000000
00001704	00000000
00001708	00000000
0000170C	00000000
00001710	00000000
00001714	00000000
00001718	00000000
0000171C	00000000
00001720	00000000
00001724	00000000
00001728	00000000
0000172C	00000000
00001730	00000000
00001734	00000000
00001738	00000000
0000173C	00000000
00001740	00000000
00001744	00000000
00001748	00000000
0000174C	00000000
00001750	00000000
00001754	00000000
00001758	00000000
0000175C	00000000
00001760	00000000
00001764	00000000
00001768	00000000
0000176C	00000000
00001770	00000000
00001774	00000000
00001778	00000000
0000177C	00000000
00001780	00000000
00001784	00000000
00001788	00000000
0000178C	



ii. half-word:

[illegible]

iii. byte:

[illegible]

Week#   2    
  3  

Program Number:

Title of the Program

**Write a program in ARM7TDMI-ISA to find the sum of N natural numbers. Store the result in the memory location.**

I. ARM Assembly Code

Code:

@ sum of n natural numbers

.text

ldr r0,=n

ldr r1,[r0]

mov r2,#0

loop:

add r2,r2,r1

subs r1,r1,#1

bne loop

ldr r3,=sum

str r2,[r3]

.data

n:.word 10

sum:.word 0

## II. Output Screen Shots (One)

General Purpose | Floating | 3.s | 4.s | 5.s | 6.s |

Hexadecimal  
Unsigned Decimal  
Signed Decimal

R0 :000010dc  
R1 :00000000  
R2 :00000037  
R3 :000010e0  
R4 :00000000  
R5 :00000000  
R6 :00000000  
R7 :00000000  
R8 :00000000  
R9 :00000000  
R10(sl):00000000  
R11(fp):00000000  
R12(ip):00000000  
R13(sp):00005400  
R14(lr):00000000  
R15(pc):00001020  
-----  
CPSR Register  
Negative(N):0  
Zero(Z) :1

@ sum of n natural numbers

```
.text
00001000:E59F0018    ldr r0,=n
00001004:E5901000    ldr r1,[r0]
00001008:E3A02000    mov r2,#0
0000100C:           loop:
0000100C:E0822001    add r2,r2,r1
00001010:E2511001    subs r1,r1,#1
00001014:1AFFFFFC    bne loop
00001018:E59F3004    ldr r3,=sum
0000101C:E5832000    str r2,[r3]

.data
000010DC:           n:.word 10
000010E0:           sum:.word 0
```

OutputView | WatchView | MemoryView0

000010dc

000010DC	0000000A	00000037	00000018	00000000	0000000A	0000001E
00001114	81818181	81818181	81818181	81818181	81818181	81818181
0000114C	81818181	81818181	81818181	81818181	81818181	81818181
00001184	81818181	81818181	81818181	81818181	81818181	81818181
000011BC	81818181	81818181	81818181	81818181	81818181	81818181

Week#   2    
  4  

Program Number:

Title of the Program

**Write a program in ARM7TDMI-ISA to find the product of two 32bit numbers using barrel shifter.**

### I. ARM Assembly Code

Code:@ find product of two 32-bit numbers using barrel shifter  
(here we multiply 24 and 33)

.text

```
ldr r0,=a
ldr r1,[r0]
mov r2,#33
ldr r3,=res
add r1,r1,r1,LSL #5
str r1,[r3]
```

.data

```
a:.word 24
res:.word 0
```

## II. Output Screen Shot (One)

General Purpose | Floating | 4.s | 5.s | 6.s |

Hexadecimal  
Unsigned Decimal  
Signed Decimal

R0 :000010b4  
R1 :00000318  
R2 :00000021  
R3 :000010b8  
R4 :00000000  
R5 :00000000  
R6 :00000000  
R7 :00000000  
R8 :00000000  
R9 :00000000  
R10(sl):00000000  
R11(fp):00000000  
R12(ip):00000000  
R13(sp):00005400  
R14(lr):00000000  
R15(pc):00001018  
-----  
CPSR Register  
Negative(N):0  
Zero(Z) :0

@ find product of two 32-bit numbers using barrel shi

```
.text
00001000:E59F0010    ldr r0,=a
00001004:E5901000    ldr r1,[r0]
00001008:E3A02021    mov r2,#33
0000100C:E59F3008    ldr r3,=res
00001010:E0811281    add r1,r1,r1,LSL #5
00001014:E5831000    str r1,[r3]

.data
000010B4:           a:.word 24
000010B8:           res:.word 0
```

OutputView | WatchView | MemoryView0 |

000010b4

000010B4	00000018	00000318	0000000A	0000001E	00000014	00000000
000010EC	81818181	81818181	81818181	81818181	81818181	81818181
00001124	81818181	81818181	81818181	81818181	81818181	81818181
0000115C	81818181	81818181	81818181	81818181	81818181	81818181
00001194	81818181	81818181	81818181	81818181	81818181	81818181

Week# 2  
5

Program Number:

Title of the Program

**Convert the following statement in C language into an ALP using ARM7TDMI - ISA.**

**IF([A]==[B]) then C=[A]+[B];**

**ELSE IF ([B]==[C]) D=[A]-[B];**

**ELSE E=[A]\*[B]**

**Where A,B C, D & E are memory locations.**

I. ARM Assembly Code

Code:

@ if([A]==[B]) then [C]=[A]+[B]

@ else if [B]==[C] then [D]=[A]-[B]

@ else [E]=[A]\*[B]

.text

ldr r0,=a

ldr r1,=b

ldr r2,=c

ldr r3,[r0]

ldr r4,[r1]

ldr r5,[r2]

cmp r3,r4

```
beq case1
cmp r4,r5
beq case2
b default
case1:
    add r6,r3,r4
    str r6,[r2]
    b end
case2:
    ldr r6,=d
    sub r7,r3,r4
    str r7,[r6]
    b end
default:
    ldr r6,=e
    mul r7,r3,r4
    str r7,[r6]
end:
```

.data

```
a:.word 10
b:.word 30
c:.word 20
d:.word 0
e:.word 0
```



## II. Output Screen Shot (One)

case1: when a==b

The screenshot displays a debugger interface with two main sections: the top section for assembly code and the bottom section for memory view.

**Top Section: Assembly Code**

The top section shows assembly code with a comment block at the top:

```
@ if([A]==[B]) then [C]=[A]+[B]
@ else if [B]==[C] then [D]=[A]-[B]
@ else [E]=[A]*[B]
```

Below the comments, the code is organized into two blocks: `.text` and `case1:`.

**case1: Block:**

```
00001000:E59F004C    ldr r0,=a
00001004:E59F104C    ldr r1,=b
00001008:E59F204C    ldr r2,=c
0000100C:E5903000    ldr r3,[r0]
00001010:E5914000    ldr r4,[r1]
00001014:E5925000    ldr r5,[r2]
00001018:E1530004    cmp r3,r4
0000101C:0A000002    beq case1
00001020:E1540005    cmp r4,r5
00001024:0A000003    beq case2
00001028:EA000006    b default
```

**case2: Block:**

```
0000102C:                case2:
0000102C:E0836004    add r6,r3,r4
00001030:E5826000    str r6,[r2]
00001034:EA000006    b end
00001038:                case2:
00001038:E59F6020    ldr r6,=d
0000103C:E0437004    sub r7,r3,r4
```

**Bottom Section: Memory View**

The bottom section shows a memory view with a search bar containing `00001068`. The memory view displays a table of memory addresses and their corresponding values:

Address	Value	Address	Value	Address	Value
00001068	0000000A	0000000A	00000014	00000000	00000000
000010A0	81818181	81818181	81818181	81818181	81818181
000010D8	81818181	81818181	81818181	81818181	81818181
00001110	81818181	81818181	81818181	81818181	81818181
00001148	81818181	81818181	81818181	81818181	81818181
00001180	81818181	81818181	81818181	81818181	81818181

case2: when b==c

General Purpose   Floating   5.s

Register	Value
R0	:00001068
R1	:0000106c
R2	:00001070
R3	:00000032
R4	:0000001e
R5	:0000001e
R6	:00001074
R7	:00000014
R8	:00000000
R9	:00000000
R10(sl)	:00000000
R11(fp)	:00000000
R12(ip)	:00000000
R13(sp)	:00005400
R14(lr)	:00000000
R15(pc)	:00001044

-----  
CPSR Register  
Negative(N):0  
Zero(Z) :1

```
00001028:EA000006    b default
0000102C:                case1:
0000102C:E0836004    add r6,r3,r4
00001030:E5826000    str r6,[r2]
00001034:EA000006    b end
00001038:                case2:
00001038:E59F6020    ldr r6,=d
0000103C:E0437004    sub r7,r3,r4
00001040:E5867000    str r7,[r6]
00001044:EA000002    b end
00001048:                default:
00001048:E59F6014    ldr r6,=e
0000104C:E0070493    mul r7,r3,r4
00001050:E5867000    str r7,[r6]
00001054:                end:

.data
00001068:                a:.word 50
0000106C:                b:.word 30
00001070:                c:.word 30
00001074:                d:.word 0
00001078:                e:.word 0
```

OutputView   WatchView   MemoryView0

00001068

00001068	00000032	0000001E	0000001E	00000014	00000000
000010A0	81818181	81818181	81818181	81818181	81818181
000010D8	81818181	81818181	81818181	81818181	81818181
00001110	81818181	81818181	81818181	81818181	81818181
00001148	81818181	81818181	81818181	81818181	81818181
00001180	81818181	81818181	81818181	81818181	81818181

default:

General Purpose

Floatin

5.s

6.s

Hexadecimal

Unsigned Decimal

Signed Decimal

R0

:00001094

▲

R1

:00001098

R2

:0000109c

R3

:0000000a

R4

:0000001e

R5

:00000014

R6

:000010a4

R7

:0000012c

R8

:00000000

R9

:00000000

R10(sl)

:00000000

R11(fp)

:00000000

R12(ip)

:00000000

R13(sp)

:00005400

R14(lr)

:00000000

R15(pc)

:00001054

-----

CPSR Register

Negative(N)

:0

Zero(Z)

:0

▼

00001028:EA000006

b default

0000102C:

case1:

0000102C:E0836004

add r6,r3,r4

00001030:E5826000

str r6,[r2]

00001034:EA000006

b end

00001038:

case2:

00001038:E59F6020

ldr r6,=d

0000103C:E0437004

sub r7,r3,r4

00001040:E5867000

str r7,[r6]

00001044:EA000002

b end

00001048:

default:

00001048:E59F6014

ldr r6,=e

0000104C:E0070493

mul r7,r3,r4

00001050:E5867000

str r7,[r6]

00001054:

end:

.data

00001094:

a:.word 10

00001098:

b:.word 30

0000109C:

c:.word 20

000010A0:

d:.word 0

000010A4:

e:.word 0

OutputView

WatchView

MemoryView0

00001094

▲

▼

00001094

0000000A

0000001E

00000014

00000000

0000012C

00000005

000010CC

81818181

81818181

81818181

81818181

81818181

81818181

00001104

81818181

81818181

81818181

81818181

81818181

81818181

0000113C

81818181

81818181

81818181

81818181

81818181

81818181

00001174

81818181

81818181

81818181

81818181

81818181

81818181

Week#   2    
  6  

Program Number:

Title of the Program

**Write a program in ARM7TDMI-ISA to find the factorial of a number.**

I. ARM Assembly Code

Code:

@ factorial of n

.text

ldr r0,=n

ldr r1,=fact

ldr r2,[r0]

mov r3,#1

loop:

mov r4,r3

mul r3,r4,r2

subs r2,r2,#1

bne loop

str r3,[r1]

.data

n:.word 5

fact:.word 1

## II. Output Screen Shot (One)

General Purpose | Floating | 6. S

Hexadecimal  
Unsigned Decimal  
Signed Decimal

R0 :0000102c  
R1 :00001030  
R2 :00000000  
R3 :00000078  
R4 :00000078  
R5 :00000000  
R6 :00000000  
R7 :00000000  
R8 :00000000  
R9 :00000000  
R10(sl):00000000  
R11(fp):00000000  
R12(ip):00000000  
R13(sp):00005400  
R14(lr):00000000  
R15(pc):00001024

-----  
CPSR Register  
Negative(N):0  
Zero(Z) :1

@ factorial of n

.text  
00001000:E59F001C ldr r0,=n  
00001004:E59F101C ldr r1,=fact  
00001008:E5902000 ldr r2,[r0]  
0000100C:E3A03001 mov r3,#1  
00001010: loop:  
00001010:E1A04003 mov r4,r3  
00001014:E0030294 mul r3,r4,r2  
00001018:E2522001 subs r2,r2,#1  
0000101C:1AFFFFFB bne loop  
00001020:E5813000 str r3,[r1]

.data  
0000102C: n:.word 5  
00001030: fact:.word 1

OutputView | WatchView | MemoryView0

0000102c

0000102C	00000005	00000078	81818181	81818181	81818181	81818181
00001064	81818181	81818181	81818181	81818181	81818181	81818181
0000109C	81818181	81818181	81818181	81818181	81818181	81818181
000010D4	81818181	81818181	81818181	81818181	81818181	81818181
0000110C	81818181	81818181	81818181	81818181	81818181	81818181

### **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: 25/01/2023