

Microprocessor and Computer Architecture Laboratory

UE19CS256

4th Semester, Academic Year 2020-21

Date:

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

Week# 5
1

Program Number:

Write an ALP to read from a 2D array such that

B=a[i] [j]

I. ARM Assembly Code (1).

Code:

@ ALP to implement B=a[i][j]

.data

matrix:.word 1,2,3,4,5,6,7,8,9

i:.word 1

j:.word 0

wrong_entry:.asciz "Invalid row and column"

B:.word 0

.text

ldr r1,=matrix

mov r2,#3

mov r3,#3

@ comparing the row and column to make sure it is a
valid access

ldr r4,=i

ldr r5,=j

ldr r6,[r4]

ldr r7,[r5]

cmp r6,#3

bmi valid_entry

b invalid_entry

invalid_entry:

ldr r0,=wrong_entry

swi 0x02

b end

valid_entry:

mov r10,#3 @ storing 3 for number of contents
in the row

```

mul r8,r6,r10
mov r8,r8,LSL #2
add r8,r8,r7,LSL #2
ldr r9,[r1,r8]
ldr r11,=B
str r9,[r11]

end:

swi 0x11

.end

```

II. Output Screen Shot (One Example of your choice)

Here i=1 and j=0; We are trying to access the first element in the second row of the 3x3 matrix. The element is 4 and is present in register r9.

General Purpose	Floating Point	1.s	2.s	3.s	4.s
Hexadecimal					
Unsigned Decimal					
Signed Decimal					

```

@ ALP to implement B=a[i][j]

.data
00001194:      matrix:.word 1,2,3,4,5,6,7,8,9
000011B8:      i:.word 1
000011BC:      j:.word 0
000011C0:      wrong_entry:.asciz "Invalid row and column"
000011D8:      B:.word 0

.text
00001000:E59F104C  ldr r1,=matrix
00001004:E3A02003  mov r2,#3
00001008:E3A03003  mov r3,#3

@ comparing the row and column to make sure it is a vldid access
0000100C:E59F4044  ldr r4,=i
00001010:E59F5044  ldr r5,=j
00001014:E5946000  ldr r6,[r4]
00001018:E5957000  ldr r7,[r5]
0000101C:E3560003  cmp r6,#3
00001020:4A000003  bmi valid_entry
00001024:EAF00000  b invalid_entry

00001028:      invalid_entry:
00001028:E3A00D47  ldr r0,=wrong_entry
0000102C:EF000002  swi 0x02
00001030:EA000006  b end

00001034:      valid_entry:
00001034:E3A0A003  mov r10,#3 @ storing 3 for number of contents in the row
00001038:E0080A96  mul r8,r6,r10
0000103C:E1A08108  mov r8,r8,LSL #2
00001040:E0888107  add r8,r8,r7,LSL #2
00001044:E7919008  ldr r9,[r1,r8]
00001048:E59FB010  ldr r11,=B

```

CPSR Register
Negative(N):1
Zero(Z):0
Carry(C):0
Overflow(V):0
IRQ Disable:1
FIQ Disable:1
Thumb(T):0
CPU Mode :System

0x800000df

Week# 5
2

Program Number:

Write an ALP to implement $C[k]=A[i]+B[j]$

I. ARM Assembly Code (1).

Code:

@ ALP to implement $C[k]=a[i]+b[j]$

.data

a:.word 1,2,3,4,5,6,7,8,9

b:.word 11,12,13,14,15,16,17,18,19

C:.word 0,0,0,0,0,0,0,0,0

i:.word 4

j:.word 3

k:.word 7

error:.asciz "Invalid Indexes!"

.text

ldr r1,=i

ldr r2,=j

ldr r3,=k

ldr r4,[r1]

ldr r5,[r2]

ldr r6,[r3]

@ making sure we don't access illegal memory

cmp r4,#0

bmi fail

cmp r5,#0

bmi fail

cmp r6,#0

bmi fail

cmp r4,#8

bpl fail

cmp r5,#8

bpl fail

cmp r6,#8

bpl fail

ldr r7,=a

ldr r8,=b

ldr r9,=C

ldr r10,[r7,r4,LSL #2]

ldr r11,[r8,r5,LSL #2]

add r11,r10,r11

str r11,[r9,r6,LSL #2]

b end

fail:

ldr r0,=error

swi 0x02

end:

swi 0x11

II. Output Screen Shot (One Example of your choice)

The output it stored in register r11 and is then stored into memory as can be seen at the bottom right corner.

The screenshot displays a debugger window with the following components:

- General Purpose** tab selected, showing a list of registers (R0 to R15) and their values. R15 (PC) is highlighted in red with the value 00001064.
- Assembly View** showing the following code:

```
0000102C:4A00000D    bmi fail
00001030:E3540008    cmp r4,#8
00001034:5A00000B    bpl fail
00001038:E3550008    cmp r5,#8
0000103C:5A000009    bpl fail
00001040:E3560008    cmp r6,#8
00001044:5A000007    bpl fail

00001048:E59F7030    ldr r7,=a
0000104C:E59F8030    ldr r8,=b
00001050:E59F9030    ldr r9,=c

00001054:E797A104    ldr r10,[r7,r4,LSL #2]
00001058:E798B105    ldr r11,[r8,r5,LSL #2]
0000105C:E08AB00B    add r11,r10,r11
00001060:E789B106    str r11,[r9,r6,LSL #2]
00001064:EA000001    b end

00001068:                fail:
00001068:E59F001C    ldr r0,=error
0000106C:EF000002    swi 0x02

00001070:                end:
00001070:                swi 0x11
```
- OutputView** tab selected, showing a memory dump starting at address 00001174. The first few bytes are 00000000, 00000000, 00000000, 00000000, 00000000, 00000000, 00000000, 00000000, 00000013.

Week# 5
3

Program Number:

Write an ALP to implement Sum[i] +=a[i][j]

I. ARM Assembly Code (1).

Code:

@ ALP to calculate the sum of elements of each row in the matrix

.data

a:.word 1,2,3,4,5,6,7,8,9

Sum:.word 0,0,0

.text

ldr r1,=a

ldr r2,=Sum

mov r3,#3 @ stores number of elements in a row

mov r7,#3 @ number of rows traversed

mov r4,#0 @ stores current pointer

mov r5,#0 @ non-array sum

loop1:

loop2:

ldr r6,[r1],#4

add r5,r5,r6

subs r3,r3,#1

bne loop2

str r5,[r2],#4

```

mov r3,#3
subs r7,r7,#1
beq end
b loop1

```

```

end:
    swi 0x11
.end

```

II. Output Screen Shot (One Example of your choice)

The screenshot displays a debugger window with the following components:

- General Purpose Register Window:** Shows registers R0 through R15. R15 (PC) is highlighted in red and contains the value 0000103c. The CPSR Register is also visible, showing Negative(N):0 and Zero(Z):1.
- Assembly Code Window:** Displays the assembly code with addresses and comments. The code includes instructions for loading registers, moving values, subtracting, branching, and storing data. Comments explain the purpose of certain registers and operations.
- OutputView Window:** Shows the output of the program, which is the hexadecimal value 000010c0.
- MemoryView0 Window:** Shows a memory dump starting at address 000010c0, with values 00000006, 00000015, 0000002D, and others.

Week# 5
4

Program Number:

Write an ALP to implement $c[k] = a[i] * b[j]$

I. ARM Assembly Code (1).

Code:

@ ALP to implement $c[k]=a[i]*b[j]$

.data

a:.word 1,2,3,4,5,6,7,8,9

b:.word 9,8,7,6,5,4,3,2,1

c:.word 0,0,0,0,0,0,0,0,0

i:.word 4

j:.word 6

k:.word 2

.text

ldr r1,=a

ldr r2,=b

ldr r3,=c

ldr r4,=i

ldr r5,=j

ldr r6,=k

ldr r4,[r4]

ldr r5,[r5]

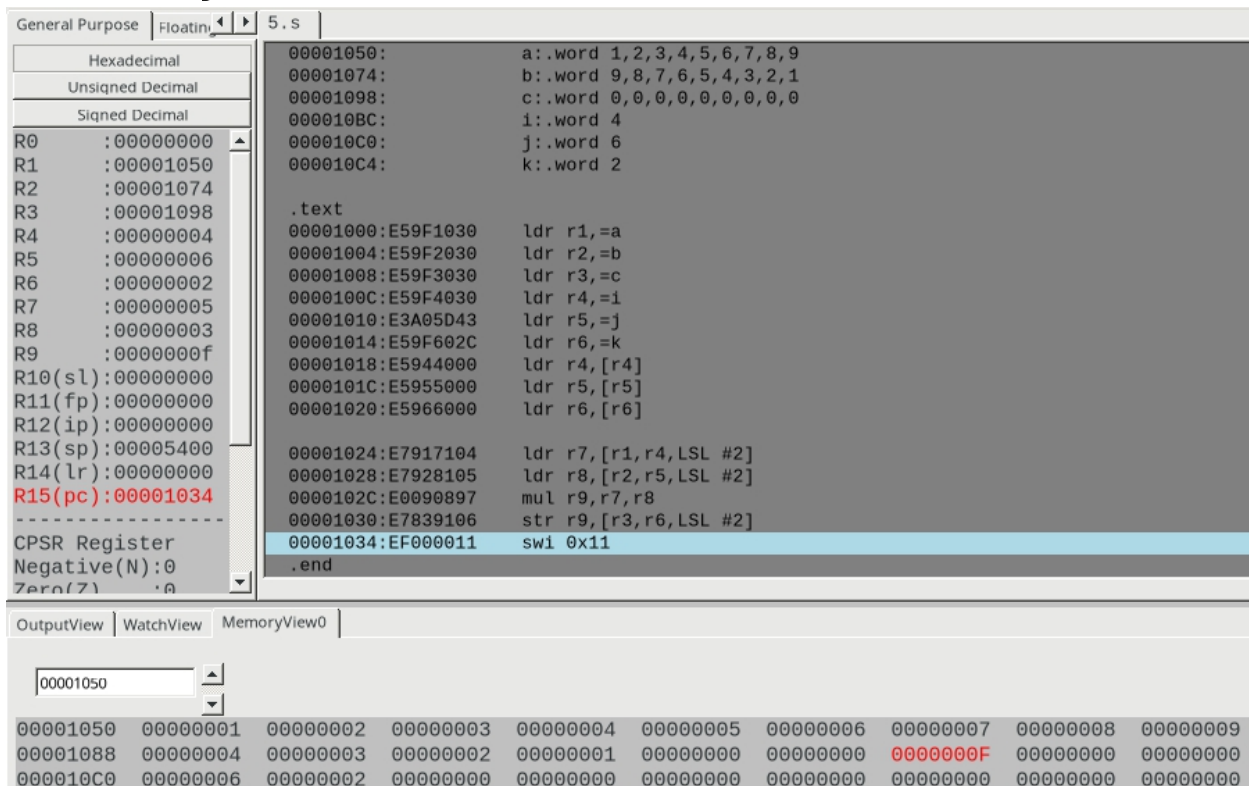
ldr r6,[r6]

ldr r7,[r1,r4,LSL #2]

ldr r8,[r2,r5,LSL #2]

```
mul r9,r7,r8
str r9,[r3,r6,LSL #2]
swi 0x11
.end
```

II. Output Screen Shot (One Example of your choice)



Week# 5
5

Program Number:

Write an ALP to implement
 $C[i][j]=a[i][j]+b[i][j]$

1. ARM Assembly Code (1).

Code:

@ ALP to implement $C[i][j]=a[i][j]+b[i][j]$ (basically matrix addition)

.data

a:.word 1,2,3,4,5,6,7,8,9

b:.word 1,2,3,4,5,6,7,8,9

C:.word 0,0,0,0,0,0,0,0,0

.text

ldr r1,=a

ldr r2,=b

ldr r3,=C

mov r4,#3 @ stores the number of columns in the matrix

mov r5,#3 @ stores the number of rows in the matrix

loop1:

loop2:

ldr r6,[r1],#4

add r7,r7,r6

ldr r6,[r2],#4

add r7,r7,r6

str r7,[r3],#4

```

mov r7,#0
subs r4,r4,#1
bne loop2

```

```

mov r4,#3
subs r5,r5,#1
beq end
b loop1

```

```

end:
swi 0x11
.end

```

2. Output Screen Shot (One Example of your choice)

The screenshot shows a debugger window with the following components:

- General Purpose** tab selected, showing a list of registers (R0 to R15) and their values in hexadecimal. R15 (pc) is highlighted in red.
- CPSR Register** section showing **Negative(N):0** and **Zero(Z):1**.
- OutputView** tab selected, showing the assembly code with addresses and comments. The code is as follows:


```

00001004:E59F2040 ldr r2,=b
00001008:E59F3040 ldr r3,=C
0000100C:E3A04003 mov r4,#3 @ stores the number of columns in the matrix
00001010:E3A05003 mov r5,#3 @ stores the number of rows in the matrix

00001014: loop1:
00001014: loop2:
00001014:E4916004 ldr r6,[r1],#4
00001018:E0877006 add r7,r7,r6
0000101C:E4926004 ldr r6,[r2],#4
00001020:E0877006 add r7,r7,r6
00001024:E4837004 str r7,[r3],#4
00001028:E3A07000 mov r7,#0
0000102C:E2544001 subs r4,r4,#1
00001030:1AFFFFF7 bne loop2

00001034:E3A04003 mov r4,#3
00001038:E2555001 subs r5,r5,#1
0000103C:0A000000 beq end
00001040:EAffFFF3 b loop1

00001044: end:
00001044:EF000011 swi 0x11
.end

```
- MemoryView0** tab selected, showing a memory dump starting at address 0000109C.

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: 18/03/2023