

Microprocessor and Computer Architecture

UE22CS251B

4th Semester, Academic Year 2023-24

Date:

Name: B S Anurag Rao	SRN: PES2UG22CS121	Section B
----------------------	-----------------------	--------------

LAB # 4 Program Number: 1

Title of the Program

1. a) Write an ALP to perform Convolution using MUL instruction (Addition of multiplication of respective numbers of loc A and loc B)

b) Write an ALP to perform Convolution using MLA instruction (Addition of multiplication of respective numbers of loc A and loc B).

I. ARM Assembly Code

A) Using MUL Instruction

.data

a:

```
.word 1, 1, 1, 1
```

b:

```
.word 2, 2, 2, 2
```

size:

```
.word 4
```

result:

```
.word 0
```

```
.text
```

```
ldr r0, =a
```

```
ldr r1, =b
```

```
ldr r2, =size
```

```
ldr r2, [r2]
```

```
mov r3, #0
```

```
; r0 -> Array A
```

```
; r1 -> Array B
```

```
; r2 -> Size
```

```
; r3 -> accumulator
```

; r4, r5, r6 -> temporary registers

loop:

ldr r4, [r0], #4

ldr r5, [r1], #4

mul r6, r4, r5

add r3, r3, r6

sub r2, r2, #1

cmp r2, #0

bgt loop

ldr r0, =result

str r3, [r0]; store in result

swi 0x11; bye bye

B) Using MLA Instruction

.data

a:

.word 1, 1, 1, 1

b:

.word 2, 2, 2, 2

size:

.word 4

result:

.word 0

.text

ldr r0, =a

ldr r1, =b

ldr r2, =size

ldr r2, [r2]

mov r3, #0

; r0 -> Array A

; r1 -> Array B

; r2 -> Size

; r3 -> accumulator

; r4, r5 -> temporary registers

loop:

ldr r4, [r0], #4

```
ldr r5, [r1], #4
mla r3, r4, r5, r3
sub r2, r2, #1
cmp r2, #0
bgt loop

ldr r0, =result
str r3, [r0]; store in result
swi 0x11; bye bye
```

II. Output Screen Shots

(Two Screenshots including Register Window, Memory Window and Code Window)

A) Using MUL Instruction

General Purpose

Floating Point

convolution_muladd.s

Hexadecimal

Unsigned Decimal

Signed Decimal

R0 : 4208

R1 : 4204

R2 : 0

R3 : 8

R4 : 1

R5 : 2

R6 : 2

R7 : 0

R8 : 0

R9 : 0

R10(s1) : 0

R11(fp) : 0

R12(ip) : 0

R13(sp) : 21504

R14(lr) : 0

R15(pc) : 4152

CPSR Register

Negative(N) : 0

Zero(Z) : 1

Carry(C) : 1

Overflow(V) : 0

IRQ Disable: 1

FIQ Disable: 1

Thumb(T) : 0

CPU Mode : System

0x600000df

```

.data

0000104C:a:
.word 1, 1, 1, 1

0000105C:b:
.word 2, 2, 2, 2

0000106C:size:
.word 4

00001070:result:
.word 0

.text
00001000:E59F0034ldr r0, =a
00001004:E59F1034ldr r1, =b
00001008:E59F2034ldr r2, =size
0000100C:E5922000ldr r2, [r2]
00001010:E3A03000mov r3, #0

; r0 -> Array A
; r1 -> Array B
; r2 -> Size
; r3 -> accumulator
; r4, r5, r6 -> temporary registers

00001014:loop:
00001014:E4904004ldr r4, [r0], #4
00001018:E4915004ldr r5, [r1], #4
0000101C:E0060594mul r6, r4, r5
00001020:E0833006add r3, r3, r6
00001024:E2422001sub r2, r2, #1
00001028:E3520000cmp r2, #0
0000102C:CAFFFFFFbgt loop

00001030:E59F0010ldr r0, =result

```

OutputView

WatchView

MemoryView0

1070

00001070	00000008	81818181	81818181	81818181	81818181
000010B4	81818181	81818181	81818181	81818181	81818181
000010F8	81818181	81818181	81818181	81818181	81818181
0000113C	81818181	81818181	81818181	81818181	81818181
00001180	81818181	81818181	81818181	81818181	81818181
000011C4	81818181	81818181	81818181	81818181	81818181

B) Using MLA Instruction

General Purpose

Floating Point

convolution_mla.s

Hexadecimal

Unsigned Decimal

Signed Decimal

R0 : 4204
R1 : 4200
R2 : 0
R3 : 8
R4 : 1
R5 : 2
R6 : 0
R7 : 0
R8 : 0
R9 : 0
R10(s1) : 0
R11(fp) : 0
R12(ip) : 0
R13(sp) : 21504
R14(lr) : 0
R15(pc) : 4148

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

0x600000df

```

.data

00001048:a:
.word 1, 1, 1, 1

00001058:b:
.word 2, 2, 2, 2

00001068:size:
.word 4

0000106C:result:
.word 0

.text
00001000:E59F0030ldr r0, =a
00001004:E59F1030ldr r1, =b
00001008:E59F2030ldr r2, =size
0000100C:E5922000ldr r2, [r2]
00001010:E3A03000mov r3, #0

; r0 -> Array A
; r1 -> Array B
; r2 -> Size
; r3 -> accumulator
; r4, r5 -> temporary registers

00001014:loop:
00001014:E4904004ldr r4, [r0], #4
00001018:E4915004ldr r5, [r1], #4
0000101C:E0233594mla r3, r4, r5, r3
00001020:E2422001sub r2, r2, #1
00001024:E3520000cmp r2, #0
00001028:CAFFFFFFbgt loop

0000102C:E59F0010ldr r0, =result
00001030:E5803000str r3, [r0]; store in result

```

OutputView

WatchView

MemoryView0

106d

0000106C	00000008	81818181	81818181	81818181	81818181	81818
000010B0	81818181	81818181	81818181	81818181	81818181	81818
000010F4	81818181	81818181	81818181	81818181	81818181	81818
00001138	81818181	81818181	81818181	81818181	81818181	81818
0000117C	81818181	81818181	81818181	81818181	81818181	81818
000011C0	81818181	81818181	81818181	81818181	81818181	81818
00001204	81818181	81818181	81818181	81818181	81818181	81818
00001248	81818181	81818181	81818181	81818181	81818181	81818

Microprocessor and Computer Architecture

UE22CS251B

4th Semester, Academic Year 2023-24

Date:

Name: B S Anurag Rao	SRN: PES2UG22CS121	Section B
----------------------	-----------------------	--------------

LAB # 4 Program Number: 2

Title of the Program

Write an ALP to implement $\text{Sum}[i] += a[i][j]$

I. ARM Assembly Code

.data

matrix:

.word 10, 20, 30, 40, 50, 60, 70, 80, 90

size:

.word 9

result:


```
.word 0
```

```
.text
```

```
ldr r0, =matrix
```

```
ldr r1, =size
```

```
ldr r1, [r1]; counter
```

```
mov r5, #0
```

```
loop:
```

```
; r0 -> matrix
```

```
; r1 -> counter
```

```
; r2 -> number being read
```

```
; r5 -> accumulator
```

```
ldr r2, [r0], #4
```

```
add r5, r5, r2
```

```
sub r1, r1, #1
```

```
cmp r1, #0
```

```
bgt loop
```

```
; once loop is done, we'll store r5 in result
```

```
ldr r0, =result; we are reusing r0 which held the array  
before
```

```
str r5, [r0]
```

```
swi 0x11; bye bye
```

II. Output Screen Shots

(One Screenshot including Register Window, Memory Window and Code Window)

General Purpose

Floating Point

Hexadecimal

Unsigned Decimal

Signed Decimal

R0 : 4196

R1 : 0

R2 : 90

R3 : 0

R4 : 0

R5 : 450

R6 : 0

R7 : 0

R8 : 0

R9 : 0

R10(s1) : 0

R11(fp) : 0

R12(ip) : 0

R13(sp) : 21504

R14(lr) : 0

R15(pc) : 4140

CPSR Register

Negative(N) : 0

Zero(Z) : 1

Carry(C) : 1

Overflow(V) : 0

IRQ Disable: 1

FIQ Disable: 1

Thumb(T) : 0

CPU Mode : System

0x600000df

accumulate_matrix.s

```

.data

0000103C:matrix:
.word 10, 20, 30, 40, 50, 60, 70, 80, 90

00001060:size:
.word 9

00001064:result:
.word 0

.text
00001000:E59F0028ldr r0, =matrix
00001004:E59F1028ldr r1, =size
00001008:E5911000ldr r1, [r1]; counter
0000100C:E3A05000mov r5, #0

00001010:loop:
; r0 -> matrix
; r1 -> counter
; r2 -> number being read
; r5 -> accumulator
00001010:E4902004ldr r2, [r0], #4
00001014:E0855002add r5, r5, r2
00001018:E2411001sub r1, r1, #1
0000101C:E3510000cmp r1, #0
00001020:CAFFFFFFAbgt loop

; once loop is done, we'll store r5 in result
00001024:E59F000Cldr r0, =result; we are reusing r0
00001028:E5805000str r5, [r0]
swi 0x11; bye bye

```

OutputView

MemoryView0

WatchView

1064

^

v

00001064	000001c2	81818181	81818181	81818181	81818181	81818181
000010a8	81818181	81818181	81818181	81818181	81818181	81818181
000010ec	81818181	81818181	81818181	81818181	81818181	81818181
00001130	81818181	81818181	81818181	81818181	81818181	81818181
00001174	81818181	81818181	81818181	81818181	81818181	81818181
000011b8	81818181	81818181	81818181	81818181	81818181	81818181
000011fc	81818181	81818181	81818181	81818181	81818181	81818181

Microprocessor and Computer Architecture

UE22CS251B

4th Semester, Academic Year 2023-24

Date:

Name: B S Anurag Rao	SRN: B S Anurag Rao	Section B
----------------------	---------------------	--------------

LAB # 4 Program Number: 3

Title of the Program

Write an ALP to find the length of a given string

I. ARM Assembly Code

.data

str:

.ascii "Anurag Rao\0"

result:

.word 0

.text

ldr r0, =str

```
mov    r1, #0
```

```
; r0 -> address of the string
```

```
; r1 -> counter
```

```
; r2 -> character being read
```

```
loop:
```

```
ldrb r2, [r0], #1
```

```
cmp    r2, #0; if r2 == 0, then end of string
```

```
beq    end
```

```
add    r1, r1, #1
```

```
b      loop
```

```
end:
```

```
ldr    r0, =str
```

```
swi    0x02; just to show that we can print strings :)
```

```
ldr    r0, =result
```

```
str    r1, [r0]; store in result
```

```
swi    0x11; bye bye
```

II. Output Screen Shot

(One Screenshot including Register Window,Memory Window and Code Window, Output Window with string displayed)

Memory Window:

General Purpose Floating Point **stringlength.s**

Hexadecimal
Unsigned Decimal
Signed Decimal

R0 : 4168
R1 : 10
R2 : 0
R3 : 0
R4 : 0
R5 : 0
R6 : 0
R7 : 0
R8 : 0
R9 : 0
R10(s1) : 0
R11(fp) : 0
R12(ip) : 0
R13(sp) : 21504
R14(lr) : 0
R15(pc) : 4140

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

0x600000df

.data
0000103C:str:
.asciiz "Anurag Rao\0"

00001048:result:
.word 0
.text
00001000:E59F0028ldr r0, =str
00001004:E3A01000mov r1, #0

; r0 -> address of the string
; r1 -> counter
; r2 -> character being read

00001008:loop:
00001008:E4D02001ldrb r2, [r0], #1
0000100C:E3520000cmp r2, #0; if r2 == 0, then end of string
00001010:0A000001beq end
00001014:E2811001add r1, r1, #1
00001018:EAF0FFFAb loop

0000101C:end:
0000101C:E59F000Cldr r0, =str
00001020:EF000002swi 0x02; just to show that we can print strings :)
00001024:E59F0008ldr r0, =result
00001028:E5801000str r1, [r0]; store in result
swi 0x11; bye bye

OutputView **MemoryView0** WatchView

1030

00001030 3C 10 00 00 48 10 00 00 00 00 00 00 00 41 6E 75 72 61 67 20 52 61 6F 00 00 0A 00 00 00 8
0000105B 81 8
00001086 81 8
000010B1 8
000010DC 81 8
00001107 81 8
00001132 81 8
0000115D 81 8
00001188 81 8
000011B3 81 8
000011DE 81 8
00001200 81 8

Stdout:

Microprocessor and Computer Architecture

UE22CS251B

4th Semester, Academic Year 2023-24

Date:

Name: B S Anurag Rao	SRN: PES2UG22CS121	Section B
----------------------	-----------------------	--------------

LAB # 4 Program Number: 4

Title of the Program

Write an ALP to copy string from one location to another

I. ARM Assembly Code

.data

source:

.ascii "Anurag Rao\0"

destination:

.ascii "aaaaaaaaa\0"

newline:


```
.ascii "\n"
```

```
.text
```

```
ldr r0, =source
```

```
ldr r1, =destination
```

```
; r0 -> address of the source string
```

```
; r1 -> address of the destination string
```

```
; r2 -> character to be copied
```

```
loop:
```

```
ldrb r2, [r0], #1
```

```
cmp r2, #0; if r2 == 0, then end of string
```

```
beq end
```

```
strb r2, [r1], #1
```

```
b loop
```

```
end:
```

```
ldr r0, =source
```

```
swi 0x02
```

```
; printing the source string because it's fun to print  
strings in assembly
```

```
ldr r0, =newline
```

```
swi 0x02
```

```
ldr r0, =destination
```

```
swi 0x02; printing the destination string
```

```
swi 0x11; bye bye
```

II. Output Screen Shots

(One Screenshot including Register Window,Memory Window and Code Window,Output Window with strings displayed)

Before running the program:

The screenshot displays a debugger window with the following components:

- General Purpose:** Floating Point, Hexadecimal, Unsigned Decimal, Signed Decimal.
- Registers:** R0:0, R1:0, R2:0, R3:0, R4:0, R5:0, R6:0, R7:0, R8:0, R9:0, R10(sp):0, R11(fp):0, R12(lp):0, R13(sp):21504, R14(lr):0, R15(pc):4096.
- 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.
- Code Window:** Assembly code for stringcopy.s, including .data, .text, and .loop sections.
- Memory Window:** Memory dump starting at address 0000104C, showing the string "Anurag Rao.".

After Running The Program:

The screenshot displays the Keil uVision IDE interface. The main window shows the assembly code for a file named `stringcopy.s`. The code defines two strings in the `.data` section: `source` (containing "Anurag Rao\0") and `destination` (containing "aaaaaaaaaa\0"). It also defines a `newline` character as `\n`. The `.text` section contains the main logic: loading the addresses of `source` and `destination` into registers `r0` and `r1` respectively, then entering a loop that copies characters from `source` to `destination` until a null terminator is reached. After the loop, it prints the source string, a newline, and the destination string before exiting.

On the left side, the "Registers" window is open, showing the state of various registers. The `R15 (pc)` register is highlighted in red, indicating the current instruction pointer at address `0x4148`. Other registers like `R0` through `R14` are shown with their current values. Below the registers, the CPSR (Current Program Status Register) is displayed, showing flags for `Negative (N)`, `Zero (Z)`, `Carry (C)`, and `Overflow (V)`, along with interrupt disable and CPU mode settings.

At the bottom, the "OutputView" window is visible, showing the output of the program. It displays the string "Anurag Rao" twice, corresponding to the source and destination strings being printed in the assembly code.

Microprocessor and Computer Architecture

UE22CS251B

4th Semester, Academic Year 2023-24

Date:

Name: B S Anurag Rao	SRN: PES2UG22CS121	Section B
----------------------	-----------------------	--------------

LAB # 4

Assignment Question 1

Title of the Program

Write an ALP to find whether a given character is present in a string. If present, find how many times the given character is present in a string.

I. ARM Assembly Code

.data

search_string: .asciiz "Please give me a good grade"

search_char: .asciiz "z"

.text

mov r0, #0

ldr r1, =search_string

```

ldr  r2, =search_char
ldrb r2, [r2]

;    r0 -> number of occurrences
;    r1 -> pointer to search string
;    r2 -> character to search for
;    r3 -> character to compare (temporary)

```

loop:

```

ldrb  r3, [r1], #1
cmp   r3, #0
beq   end
cmp   r3, r2
addeq r0, r0, #1
b     loop

```

end:

```

swi 0x11

```

II. Output Screen Shots

(Two Screenshots-Character Present,Character not Present,screenshot including Register Window,Memory Window and Code Window)

Character present, count # in R0

General Purpose		Floating Point	find_char.s
Hexadecimal			
Unsigned Decimal			
Signed Decimal			
R0	: 5		.data
R1	: 4176		00001034:search_string: .asciiz "Please give me a good grade"
R2	: 101		00001050:search_char: .asciiz "e"
R3	: 0		
R4	: 0		.text
R5	: 0		00001000:E3A00000mov r0, #0
R6	: 0		00001004:E59F1020ldr r1, =search_string
R7	: 0		00001008:E59F2020ldr r2, =search_char
R8	: 0		0000100C:E5D22000ldrb r2, [r2]
R9	: 0		; r0 -> number of occurrences
R10(sl)	: 0		; r1 -> pointer to search string
R11(fp)	: 0		; r2 -> character to search for
R12(ip)	: 0		; r3 -> character to compare (temporary)
R13(sp)	: 21504		
R14(lr)	: 0		00001010:loop:
R15(pc)	: 4136		00001010:E4D13001ldrb r3, [r1], #1
-----			00001014:E3530000cmp r3, #0
CPSR Register			00001018:0A000002beq end
Negative(N)	: 0		0000101C:E1530002cmp r3, r2
Zero(Z)	: 1		00001020:02800001addeq r0, r0, #1
Carry(C)	: 1		00001024:EAffFFFF9b loop
Overflow(V)	: 0		
IRQ Disable	: 1		00001028:end:
FIQ Disable	: 1		swi 0x11

Character Not Present, Count = 0 at R0:

General Purpose		Floating Point	find_char.s
Hexadecimal			
Unsigned Decimal			
Signed Decimal			
R0	: 0		.data
R1	: 4176		00001034:search_string: .asciiz "Please give me a goo
R2	: 122		00001050:search_char: .asciiz "z"
R3	: 0		
R4	: 0		.text
R5	: 0		00001000:E3A00000mov r0, #0
R6	: 0		00001004:E59F1020ldr r1, =search_string
R7	: 0		00001008:E59F2020ldr r2, =search_char
R8	: 0		0000100C:E5D22000ldrb r2, [r2]
R9	: 0		; r0 -> number of occurrences
R10(s1)	: 0		; r1 -> pointer to search string
R11(fp)	: 0		; r2 -> character to search for
R12(ip)	: 0		; r3 -> character to compare (temporary)
R13(sp)	: 21504		
R14(lr)	: 0		00001010:loop:
R15(pc)	: 4136		00001010:E4D13001ldrb r3, [r1], #1
-----			00001014:E3530000cmp r3, #0
CPSR Register			00001018:0A000002beq end
Negative(N)	: 0		0000101C:E1530002cmp r3, r2
Zero(Z)	: 1		00001020:02800001addeq r0, r0, #1
Carry(C)	: 1		00001024:EAF9FF9b loop
Overflow(V)	: 0		
IRQ Disable	: 1		00001028:end:
FIQ Disable	: 1		swi 0x11
Thumb(T)	: 0		

Microprocessor and Computer Architecture

UE22CS251B

4th Semester, Academic Year 2023-24

Date:

Name: B S Anurag Rao	SRN: PES2UG22CS121	Section B
----------------------	-----------------------	--------------

LAB # 4

Assignment Question 2

Title of the Program

Write a program in ARM7TDMI-ISA to generate a diagonal matrix.

;Note: do not read the matrix elements.

I. ARM Assembly Code

```
.data
```

```
dimension:
```

```
.word 3
```

```
result:
```

```
.word 0, 0, 0, 0, 0, 0, 0, 0, 0
```

```

.text

ldr r0, =dimension

ldr r0, [r0]

ldr r1, =result

mov r2, #1; column

mov r3, #1; row

mov r4, #1; the number we'd be storing on the diagonal
positions

mov r5, #0; the number we'd be storing on the off-diagonal
positions

loop:

    cmp    r2, r3
    streq r4, [r1], #4
    strne r5, [r1], #4
    add    r2, r2, #1
    cmp    r2, r0
    bgt    next_row
    b      loop

next_row:

    mov r2, #1

```

```
add r3, r3, #1
cmp r3, r0
bgt end
b loop

end:
swi 0x11
```

- II. Output Screen Shots
*(One Screenshot including Register Window, Memory Window
and Code Window)*

General Purpose

Floating Point

generate_diagonal.s

Hexadecimal

Unsigned Decimal

Signed Decimal

R0 : 3

R1 : 4224

R2 : 1

R3 : 4

R4 : 1

R5 : 0

R6 : 0

R7 : 0

R8 : 0

R9 : 0

R10(s1) : 0

R11(fp) : 0

R12(ip) : 0

R13(sp) : 21504

R14(Lr) : 0

R15(pc) : 4172

CPSR Register

Negative(N) : 0

Zero(Z) : 0

Carry(C) : 1

Overflow(V) : 0

IRQ Disable: 1

FIQ Disable: 1

Thumb(T) : 0

CPU Mode : System

0x200000df

```

.data
00001058:dimension:
.word 3

0000105C:result:
.word 0, 0, 0, 0, 0, 0, 0, 0

.text
00001000:E59F0048ldr r0, =dimension
00001004:E5900000ldr r0, [r0]
00001008:E59F1044ldr r1, =result
0000100C:E3A02001mov r2, #1: column
00001010:E3A03001mov r3, #1: row
00001014:E3A04001mov r4, #1: the number we'd be storing on the diagonal positions
00001018:E3A05000mov r5, #0: the number we'd be storing on the off-diagonal positions

0000101C:loop:
0000101C:E1520003cmp r2, r3
00001020:04814004streq r4, [r1], #4
00001024:14815004strne r5, [r1], #4
00001028:E2822001add r2, r2, #1
0000102C:E1520000cmp r2, r0
00001030:CA000000bgt next_row
00001034:EAF8FF8b loop

00001038:next_row:
00001038:E3A02001mov r2, #1
0000103C:E2833001add r3, r3, #1
00001040:E1530000cmp r3, r0
00001044:CA000000bgt end
00001048:EAF8FF3b loop

0000104C:end:
swi 0x11

```

OutputView

MemoryView0

WatchView

1054

00001054

0000105C

00000003

00000001

00000000

00000000

00000000

00000000

00000001

00000000

00000000

00000000

00000001

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: B S Anurag Rao

SRN: PES2UG22CS121

Section: B

Date: 20 February 2024