



PES UNIVERSITY

Department of Computer Science & Engineering

Microprocessor & Computer Architecture Lab

UE23CS251B

WEEK 4 submission

Name of the Student	Pranav Hemanth
SRN	PES1UG23CS433
Section	G
Department	CSE
Campus	RR

Department of Computer Science & Engineering
Microprocessor & Computer Architecture Lab

UE23CS251B

1 Write an ALP using ARM7TDMI to add n numbers bitwise.

..DATA

A: .byte 1,2,3,4,5,6,7,8,9,10

Program screen shot:

ARMSim_files > **ASM** lab4_q1.s

```

1  @ Write an ALP using ARM7TDMI to add n numbers bitwise using BL.
2
3  .data
4
5  size: .word 10
6  A: .byte 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
7
8  .text
9
10 init:
11     LDR R0, =A
12     LDR R1, =size
13     LDR R1, [R1]
14     MOV R3, #0
15
16 start:
17     BL add_func
18
19 end:
20     SWI 0x011
21
22 add_func:
23     CMP R1, #0
24     BXEQ LR
25
26 loop:
27     LDRB R2, [R0]
28     ADD R3, R3, R2
29     ADD R0, R0, #1
30     SUB R1, R1, #1
31     B add_func
--

```

Screen shot of Register set output:

ARMSim - The ARM Simulator

File View Cache Debug Watch Help

lab4_q1.o

Hexadecimal
Unsigned Decimal
Signed Decimal

R0 :4170
R1 :0
R2 :10
R3 :55
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):4116
R15(pc):4116

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

```

00001000:      init:
00001000:      $a:
00001000:E59F002C      LDR    r0, $d
00001004:E59F102C      LDR    r1, $d+0x4
00001008:E5911000      LDR    r1, [r1, #0]
0000100C:E3A03000      MOV    r3, #0
00001010:      start:
00001010:EB000000      B      add_func
00001014:      end:
00001014:EF000011      SWI    0x11
00001018:      add_func:
00001018:E3510000      CMP    r1, #0
0000101C:012FFF1E      BXEQ   lr
00001020:      loop:
00001020:E5D02000      LDRB   r2, [r0, #0]
00001024:E0833002      ADD    r3, r3, r2
00001028:E2800001      ADD    r0, r0, #1
0000102C:E2411001      SUB    r1, r1, #1
00001030:EAF0FF0F      B      add_func
00001034:      $d:
00001034:00001040      ANDEQ   r1, r0, r0, asr #32
00001038:0000103C      ANDEQ   r1, r0, ip, lsr r0
0000103C:      size:
0000103C:      .word 0x0000000A
00001040:      a:
00001040:      .word 0x04030201, 0x08070605, 0x00000A09
  
```

OutputView WatchView MemoryView0

0000F96

0000F94	????????	????????	????????	????????	????????	????????	????????	??????
0000FD0	????????	????????	????????	????????	????????	????????	????????	??????
0000100C	E3A03000	EB000000	EF000011	E3510000	012FFF1E	E5D02000	E0833002	E2800001
00001048	00000A09	81818181	81818181	81818181	81818181	81818181	81818181	81818181

2 Write an ALP using ARM7TDMI to generate a square given matrix with A

If (i==j) then A[i][j]=5

Otherwise A[i][j]=0

(Note:Any size of matrix can be given as input)

Considering 4X4 matrix

Example :

5	0	0	0
0	5	0	0
0	0	5	0
0	0	0	5

Jan -May 2025 LAB SUBMISSION_UE23CS251B

Before:

A:.word 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16

After:

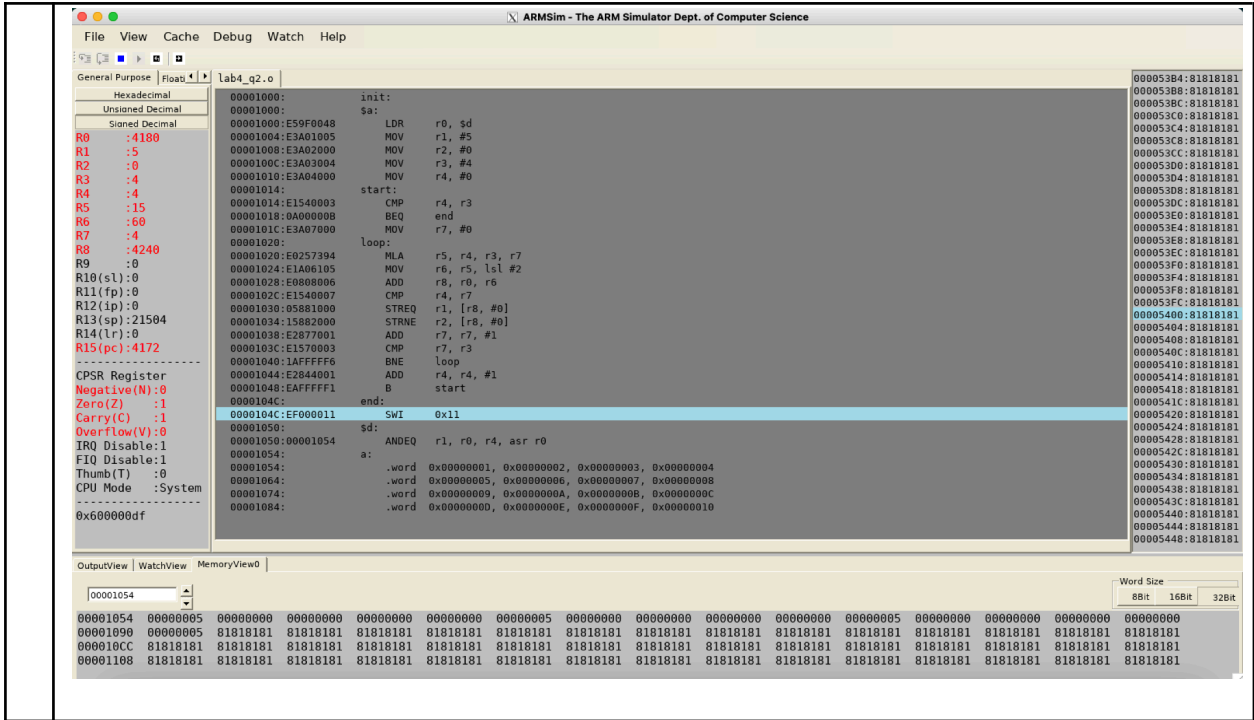
A:.word 5,0,0,0,0,5,0,0,0,0,5,0,0,0,0,5

Program screen shot:

```
ARMSim_files > ASM lab4_q2.s
7  @ Considering 4X4 matrix
8
9  @ Example :      5 0 0 0
10 @                0 5 0 0
11 @                0 0 5 0
12 @                0 0 0 0
13
14 @Before:
15 @A:.word 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16
16
17 @After:
18 @A:.word 5,0,0,0,0,5,0,0,0,0,5,0,0,0,0,5
19
20 .data
21 A: .word 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16
22
23 .text
24
25 init:
26     LDR R0, =A
27     MOV R1, #5      // Value to store in the diagonal
28     MOV R2, #0      // Value to store for non-diagonal elements
29     MOV R3, #4      // Size of the matrix (4x4)
30     MOV R4, #0      // Row index
31
32 start:
33     CMP R4, R3
34     BEQ end
35     MOV R7, #0
36
37 loop:
38     MLA R5, R4, R3, R7
39     MOV R6, R5, LSL #2
40     ADD R8, R0, R6
41     CMP R4, R7      // Check if row index == column index
42     STREQ R1, [R8]
43     STRNE R2, [R8]
44     ADD R7, R7, #1
45     CMP R7, R3
46     BNE loop
47     ADD R4, R4, #1
48     B start
49
50 end:
51     SWI 0x011
52
```

Screen shot of Register set output and memory location:

Jan -May 2025 LAB SUBMISSION_UE23CS251B



General Purpose Floati ◀ ▶

Hexadecimal

Unsigned Decimal

Signed Decimal

R0 :4180

R1 :5

R2 :0

R3 :4

R4 :4

R5 :15

R6 :60

R7 :4

R8 :4240

R9 :0

R10(sl):0

R11(fp):0

R12(ip):0

R13(sp):21504

R14(lr):0

R15(pc):4172

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

OutputView WatchView Men

OutputView WatchView MemoryView0														
00001054														
Word Size														
8Bit 16Bit														
00001054	00000005	00000000	00000000	00000000	00000000	00000005	00000000	00000000	00000000	00000000	00000005	00000000	00000000	00000000
00001090	00000005	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181
000010CC	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181
00001108	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181

3 Write an ALP using ARM7TDMI to convert hexadecimal to decimal.

Input: **0x0124**(Hex)

Conversion Process:

- 0x124 → Extract **4** → $0 * 16 + 4 = 4$
- 0x124 → Extract **2** → $2 * 16^1 + 4 = 36$
- 0x124 → Extract **1** → $1 * 16^2 + 36 = 292$

Output: **292**(Decimal)

Program screen shot:

```

ARMSim_files > asm lab4_q3.s
1  @ Write an ALP using ARM7TDMI to convert hexadecimal to decimal.
2  @ Input: 0x0124(Hex)
3  @ Conversion Process:
4  @ 0x124 → Extract 4 → 0 * 16 + 4 = 4
5  @ 0x124 → Extract 2 → 2 * 16^1 + 4 = 36
6  @ 0x124 → Extract 1 → 1* 16^2 + 36 =292
7  @ Output: 292(Decimal)
8
9  .data
10 hex: .word 0x0124
11 decimal: .word 0
12
13 .text
14
15 init:
16     LDR R0, =hex
17     LDR R0, [R0]
18     MOV R2, #0
19     MOV R3, #1
20
21 loop:
22     CMP R0, #0
23     BEQ end
24
25     AND R4, R0, #0xF @ Extract the least significant digit (last 4 bits)
26     MUL R4, R3, R4 @ Multiply the extracted digit by the current multiplier (16^0, 16^1, etc.)
27     ADD R2, R2, R4
28
29     MOV R0, R0, LSR #4 @ Shift R0 right by 4 bits to process the next digit
30
31     MOV R4, #16
32     MUL R3, R4, R3 @ Update the multiplier for the next place value (16^1, 16^2, etc.)
33
34     B loop
35
36 end:
37     LDR R1, =decimal
38     STR R2, [R1]
39     SWI 0x011

```

Screen shot of Register set output and memory location:

General Purpose

Floati

Hexadecimal

Unsigned Decimal

Signed Decimal

R0

:00000000

R1

:0000104c

R2

:00000124

R3

:00001000

R4

:00000010

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)

:0000103c

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

OutputView

WatchView

MemoryView0

0000104c

0000104C

00000124

81818181

81818181

81818181

81818181

81818181

81818181

81818181

81818181

81818181

81818181

00001088

81818181

81818181

81818181

81818181

81818181

81818181

81818181

81818181

81818181

81818181

81818181

000010C4

81818181

81818181

81818181

81818181

81818181

81818181

81818181

81818181

81818181

81818181

81818181

00001100

81818181

81818181

81818181

81818181

81818181

81818181

81818181

81818181

81818181

81818181

81818181

4 Write an ALP using ARM7TDMI, for the given matrix arranged in Column major order, find the index of an element if coordinates of a matrix is given and also find the address of the indexed element. (Using MLA instruction)

. DATA

A:WORD 1,2,3,4,5,6,7,8,9

.Index for the column major= $y \times \text{no of rows} + x$

Program screen shot:

ARMSim_files > *asm* lab4_q4.s

```

1  @ Write an ALP using ARM7TDMI, for the given matrix arranged in Column major order, find the index of an element
2  @ if coordinates of a matrix is given and also find the address of the indexed element. (Using MLA instruction)
3  @ . DATA
4  @
5  @ A: .WORD 1,2,3,4,5,6,7,8,9
6  @ .Index for the column major= y*no of rows+x
7
8  .data
9  A: .word 1,2,3,4,5,6,7,8,9
10
11 .text
12
13 init:
14     LDR R0, =A
15     MOV R1, #3           @ Number of rows
16     MOV R4, #1           @ x-coordinate (row index, assuming 1-based indexing)
17     MOV R5, #2           @ y-coordinate (column index, assuming 1-based indexing)
18
19     SUB R4, R4, #1        @ Convert x to 0-based index
20     SUB R5, R5, #1        @ Convert y to 0-based index
21
22 start:
23     MLA R6, R5, R1, R4
24     MOV R7, R6, LSL #2
25     ADD R8, R0, R7
26
27 end:
28     LDR R9, [R8]
29     SWI 0x011
30

```

Screen shot of Register set output and memory location:

	Assignments Questions
5	<p>Write an ALP using ARM7TDMI to reverse the elements stored in location A with location B</p> <p>Before: A:.word 1,2,3,4,5,6,7,8,9,10</p> <p>After : A:.word 10,9,8,7,6,5,4,3,2,1</p> <p>Solution 1:</p> <p>Program screen shot:</p> <pre> ARMSim_files > ASM lab4_q5a.s 1 @ Write an ALP using ARM7TDMI to reverse the elements stored in location A with location B 2 @ Before: 3 @ A:.word 1,2,3,4,5,6,7,8,9,10 4 @ After : 5 @ A:.word 10,9,8,7,6,5,4,3,2,1 6 7 .data 8 A: .word 1,2,3,4,5,6,7,8,9,10 9 B: .word 0,0,0,0,0,0,0,0,0,0 10 size: .word 10 11 12 .text 13 14 init: 15 LDR R0, =A 16 LDR R1, =size 17 LDR R1, [R1] 18 LDR R2, =B 19 20 ADD R2, R2, R1, LSL #2 @ Move R2 to the end of array B (size * 4) 21 SUB R2, R2, #4 @ Adjust R2 to point to the last element of B 22 23 start: 24 CMP R1, #0 25 BEQ end @ If size (R1) == 0, we're done 26 27 LDR R3, [R0], #4 @ Load current element from A into R3 and increment R0 28 STR R3, [R2], #-4 @ Store R3 into B and decrement R2 to move backward 29 SUB R1, R1, #1 @ Decrement the size counter 30 31 B start 32 33 end: 34 SWI 0x011 35 </pre>

Screen shot of Register set output:

General Purpose

Floati

Hexadecimal

Unsigned Decimal

Signed Decimal

R0

:4200

R1

:0

R2

:4196

R3

:10

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)

:0

R15(pc)

:4144

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

OutputView WatchView MemoryView0															
00000F80															
00000F80	????????	????????	????????	????????	????????	????????	????????	????????	????????	????????	????????	????????	????????	????????	????????
00000FEC	????????	????????	????????	????????	????????	E59F002C	E59F102C	E5911000	E59F2028	E0822101	E2422004	E3510000	0A000003	E4903004	E4023004
00001028	E2411001	EAFFFFF9	EF000011	00001040	00001090	00001068	00000001	00000002	00000003	00000004	00000005	00000006	00000007	00000008	00000009
00001064	0000000A	0000000A	00000009	00000008	00000007	00000006	00000005	00000004	00000003	00000002	00000001	0000000A	81818181	81818181	81818181

Solution 2:

Program screen shot:

Jan -May 2025 LAB SUBMISSION_UE23CS251B

ARMSim_files > **ASM** lab4_q5b.s

```
1  @ Write an ALP using ARM7TDMI to reverse the elements stored in location A with location B
2  @ Before:
3  @ A:.word 1,2,3,4,5,6,7,8,9,10
4  @ After :
5  @ A:.word 10,9,8,7,6,5,4,3,2,1
6
7  .data
8  A:   .word 1,2,3,4,5,6,7,8,9,10
9  B:   .word 0,0,0,0,0,0,0,0,0,0
10 size: .word 10
11
12 .text
13
14 init:
15     LDR R0, =A
16     LDR R1, =size
17     LDR R1, [R1]
18     LDR R2, =B
19
20     SUB R3, R1, #1      @ Set R3 to (size - 1), used as the reverse index
21
22 start:
23     CMP R3, #0          @ Check if reverse index (R3) is less than 0
24     BLT end             @ If R3 < 0, we are done
25
26     LDR R4, [R0], #4
27     STR R4, [R2, R3, LSL #2] @ Store R4 into B at offset (R3 * 4)
28     SUB R3, R3, #1
29
30     B start
31
32 end:
33     SWI 0x011
34
```

Screen shot of Register set output:

Jan -May 2025 LAB SUBMISSION_UE23CS251B

ARMSim_files > **ASM** lab4_q6.s

```
1  @ Write an ALP using ARM7TDMI to find the largest of all the BCD digits of a given 32bit number.
2  @ (hint:If R1=17845374 the largest digit is 8
3
4  .data
5  num: .word 0x17845374 @ 32-bit number
6
7  .text
8
9  init:
10     LDR R0, =num
11     LDR R0, [R0]
12     MOV R1, #0
13     MOV R2, #8 @ number of digits in 32bit number
14     MOV R3, #0
15
16  loop:
17     AND R3, R0, #0xF @ Extract the lsb (last 4 bits)
18     CMP R3, R1
19     MOVHI R1, R3
20
21     MOV R0, R0, LSR #4 @ Shift R0 right by 4 bits
22     SUB R2, R2, #1
23     CMP R2, #0
24     BNE loop
25
26  end:
27     MOV R7, #1
28     SWI 0x011
29
```

Screen shot of Register set output:

a) Write an ALP using ARM7TDMI to copy a block 400 bytes of data from location A to location B if the rate of data transfer rate is 40 bytes, LDM and STM instructions.

b) For the same transfer the block with auto-indexing.

Program screen shot:

Jan -May 2025 LAB SUBMISSION_UE23CS251B

```
ARMSim_files > ASM lab4_q7a.s
1  @ Write an ALP using ARM7TDMI to copy a block 400 bytes of data from location A to location B
2  @if the rate of data transfer rate is 40 bytes, LDM and STM instructions.
3  @ and
4  @ For the same transfer the block with auto-indexing.
5
6  @ Using LDM and STM Instructions
7
8  .data
9  A: .skip 400      @ 400 bytes of data at location A (source)
10 B: .skip 400      @ 400 bytes of data at location B (destination)
11
12 .text
13 init:
14     LDR R0, =A
15     LDR R1, =B
16     MOV R2, #40    @ Set the data transfer rate to 40 bytes (one LDM/STM operation)
17     MOV R3, #10    @ Set the number of transfers
18
19 loop:
20     LDMIA R0!, {R4-R11} @ Load 8 registers (40 bytes) from source (A) into R4-R11, incrementing R0
21     STMIA R1!, {R4-R11} @ Store the 8 registers (40 bytes) to destination (B), incrementing R1
22     SUBS R3, R3, #1    @ Decrement the transfer counter
23     CMP R3, #0
24     BNE loop
25
26 end:
27     SWI 0x11
28
```

Screen shot of Register set output:

Solution B:

Program screen shot:

```
ARMSim_files > ASM lab4_q7b.s
1  @ Write an ALP using ARM7TDMI to copy a block 400 bytes of data from location A to location B
2  @if the rate of data transfer rate is 40 bytes, LDM and STM instructions.
3  @ and
4  @ For the same transfer the block with auto-indexing.
5
6  @ Using Auto-Indexing (with LDM and STM)
7
8  .data
9  A: .skip 400      @ 400 bytes of data at location A (source)
10 B: .skip 400      @ 400 bytes of data at location B (destination)
11
12 .text
13 init:
14     LDR R0, =A
15     LDR R1, =B
16     MOV R2, #40    @ Set the data transfer rate to 40 bytes (one LDM/STM operation)
17     MOV R3, #10    @ Set the number of transfers
18
19 loop:
20     LDMIA R0!, {R4-R11} @ Load 8 registers (40 bytes) from source (A) into R4-R11, incrementing R0
21     STMIA R1!, {R4-R11} @ Store the 8 registers (40 bytes) to destination (B), incrementing R1
22     SUBS R3, R3, #1
23     CMP R3, #0
24     BNE loop
25
26 end:
27     SWI 0x11
28
```

Screen shot of Register set output:

