



PES UNIVERSITY

Department of Computer Science & Engineering

Microprocessor & Computer Architecture Lab

UE23CS251B

WEEK 4 submission

Name of the Student	Amit Prakash
SRN	PES1UG23AM042
Section	A
Department	CSE - AIML
Campus	RR

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:

```

week-4 > 1.s
1  .data
2  A: .byte 1,2,3,4,5,6,7,8,9,10
3
4  .text
5  MOV R0,#10
6  LDR R1,=A
7  LOOP:
8      LDRB R2,[R1],#1
9      ADD R3,R3,R2
10     SUBS R0,R0,#1
11     BNE LOOP
12
13     SWI 0x11;
14

```

Screen shot of Register set output:

ARMSim - The ARM Simulator Dept. of Computer Science

File View Cache Debug Watch Help

RegistersView 1.s

General Purpose

Hexadecimal

Unsigned Decimal

Signed Decimal

Register	Value
R0	: 0
R1	: 4138
R2	: 10
R3	: 55
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)	: 4120

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 : S

0x600000df

```

.data
00001020:      A: .byte 1,2,3,4,5,6,7,8,9,10

.text
00001000:E3A0000A  MOV R0,#10
00001004:E59F1010  LDR R1,=A
00001008:      LOOP:
00001008:E4D12001      LDRB R2,[R1],#1
0000100C:E0833002      ADD R3,R3,R2
00001010:E2500001      SUBS R0,R0,#1
00001014:1AFFFFF8      BNE LOOP
00001018:EF000011      SWI 0x11;
0000101C:00001020

```

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

Before:

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

After:

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

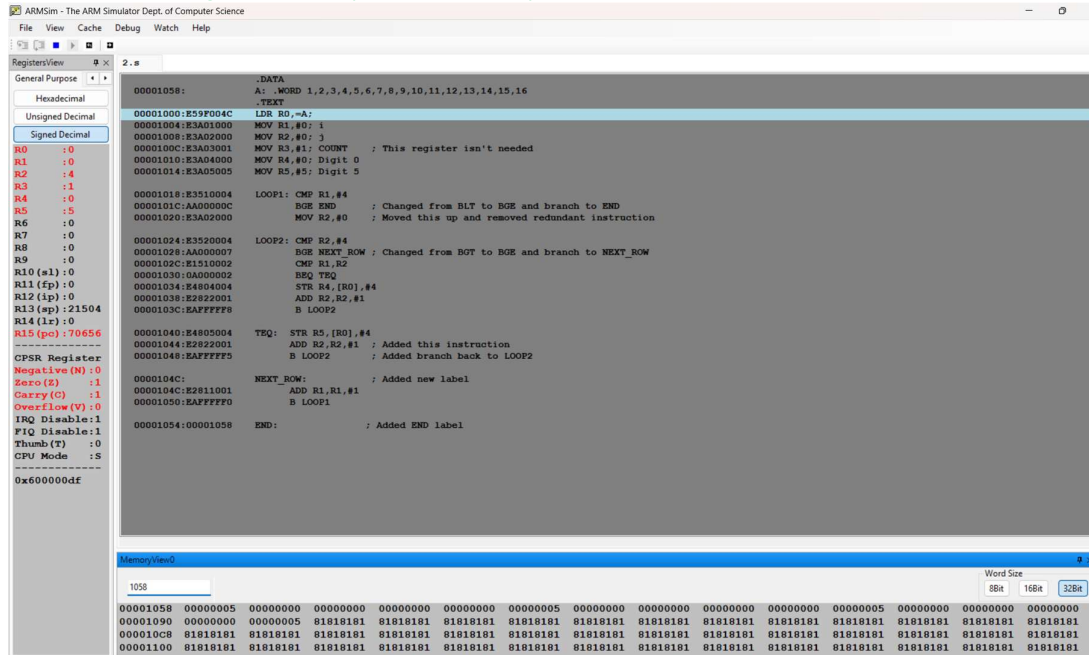
Program screen shot:

```

week-4 > 2.s
3  .TEXT
4  LDR R0,=A;
5  MOV R1,#0; i
6  MOV R2,#0; j
7  MOV R3,#1; COUNT ; This register isn't needed
8  MOV R4,#0; Digit 0
9  MOV R5,#5; Digit 5
10
11  LOOP1: CMP R1,#4
12          BGE END ; Changed from BLT to BGE and branch to END
13          MOV R2,#0 ; Moved this up and removed redundant instruction
14
15  LOOP2: CMP R2,#4
16          BGE NEXT_ROW ; Changed from BGT to BGE and branch to NEXT_ROW
17          CMP R1,R2
18          BEQ TEQ
19          STR R4,[R0],#4
20          ADD R2,R2,#1
21          B LOOP2
22
23  TEQ: STR R5,[R0],#4
24          ADD R2,R2,#1 ; Added this instruction
25          B LOOP2 ; Added branch back to LOOP2
26
27  NEXT_ROW: ; Added new label
28          ADD R1,R1,#1
29          B LOOP1
30
31  END: ; Added END label

```

Screen shot of Register set output and memory location:



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

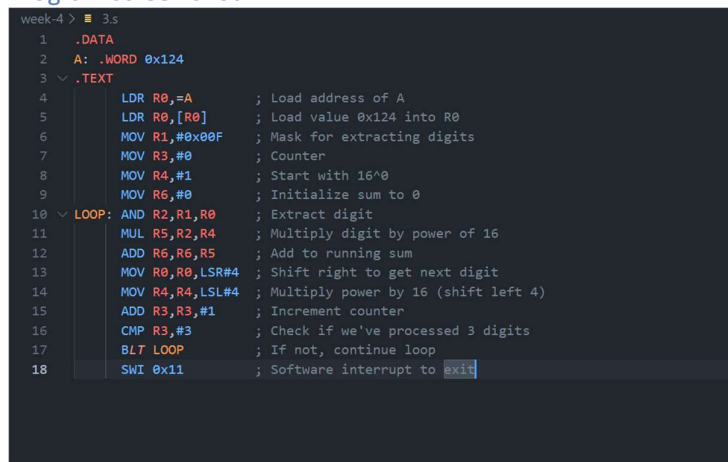
Input: **0x0124**(Hex)

Conversion Process:

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

Output: **292**(Decimal)

Program screen shot:



Screen shot of Register set output and memory location:

RegistersView

General Purpose

Hexadecimal

Unsigned Decimal

Signed Decimal

R0 : 0

R1 : 15

R2 : 1

R3 : 3

R4 : 4096

R5 : 256

R6 : 292

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 : S

0x600000df

```

00001040:
    .DATA
    A: .WORD 0x124
    .TEXT
00001000:E3A00D41    LDR R0,=A           ; Load address of A
00001004:E5900000    LDR R0,[R0]         ; Load value 0x124 into R0
00001008:E3A0100F    MOV R1,#0x00F       ; Mask for extracting digits
0000100C:E3A03000    MOV R3,#0           ; Counter
00001010:E3A04001    MOV R4,#1           ; Start with 16^0
00001014:E3A06000    MOV R6,#0           ; Initialize sum to 0
00001018:E0012000    LOOP: AND R2,R1,R0   ; Extract digit
0000101C:E0050492    MUL R5,R2,R4         ; Multiply digit by power of 16
00001020:E0866005    ADD R6,R6,R5         ; Add to running sum
00001024:E1A00220    MOV R0,R0,LSR#4      ; Shift right to get next digit
00001028:E1A04204    MOV R4,R4,LSL#4      ; Multiply power by 16 (shift left 4)
0000102C:E2833001    ADD R3,R3,#1         ; Increment counter
00001030:E3530003    CMP R3,#3           ; Check if we've processed 3 digits
00001034:BAFFFFF7    BLT LOOP             ; If not, continue loop
00001038:EF000011    SWI 0x11            ; Software interrupt to exit
0000103C:00000000

```

- 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= $y * \text{no of rows} + x$

Program screen shot:

week-4 > 4.s

```

1  .DATA
2  A: .WORD 1,2,3,4,5,6,7,8,9
3  .TEXT
4      LDR R0,=A           ; Load base address
5      MOV R1,#2           ; x coordinate
6      MOV R2,#2           ; y coordinate
7      MOV R3,#3           ; number of rows
8      MOV R6,#4           ; word size (4 bytes)
9      MLA R4,R2,R3,R1     ; Calculate index (y*rows + x)
10     MLA R5,R4,R6,R0     ; Calculate final address (base + index*4)
11     LDR R7,[R5]         ; Load value at calculated address
12     SWI 0x11

```

Screen shot of Register set output and memory location:

```

ARMSim - The ARM Simulator Dept. of Computer Science
File View Cache Debug Watch Help

RegistersView 4.s
General Purpose
Hexadecimal
Unsigned Decimal
Signed Decimal
R0 : 4136
R1 : 2
R2 : 2
R3 : 3
R4 : 8
R5 : 4168
R6 : 4
R7 : 9
R8 : 0
R9 : 0
R10 (s1) : 0
R11 (fp) : 0
R12 (ip) : 0
R13 (sp) : 21504
R14 (lr) : 0
R15 (pc) : 4128
-----
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 : S
-----
0x000000df

.DATA
00001028: A: .WORD 1,2,3,4,5,6,7,8,9
.TEXT
00001000: E59F001C LDR R0,=A ; Load base address
00001004: E3A01002 MOV R1,#2 ; x coordinate
00001008: E3A02002 MOV R2,#2 ; y coordinate
0000100C: E3A03003 MOV R3,#3 ; number of rows
00001010: E3A06004 MOV R6,#4 ; word size (4 bytes)
00001014: E0241392 MLA R4,R2,R3,R1 ; Calculate index (y*rows + x)
00001018: E0250694 MLA R5,R4,R6,R0 ; Calculate final address (base + index*4)
0000101C: E5957000 LDR R7,[R5] ; Load value at calculated address
00001020: EF000011 SWI 0x11
00001024: 00001028

```

Assignments Questions

- 5 Write an ALP using ARM7TDMI to reverse the elements stored in location A with location B
 Before:
 A: word 1,2,3,4,5,6,7,8,9,10
 After :
 A: word 10,9,8,7,6,5,4,3,2,1

Program screen shot:

```

week-4 > 5.s
1 .DATA
2 A: .WORD 1,2,3,4,5,6,7,8,9,10
3 B: .WORD 0,0,0,0,0,0,0,0,0,0
4
5 .TEXT
6 LDR R0,=A
7 LDR R1,=B
8 MOV R2,#10
9 ADD R0,R0,#36
10
11 LOOP:
12 LDR R4,[R0]
13 STR R4,[R1]
14 SUB R0,R0,#4
15 ADD R1,R1,#4
16 SUBS R2,R2,#1
17 BNE LOOP
18 LDR R0,=A
19 LDR R1,=B
20 MOV R2,#10
21
22 COPY:
23 LDR R4,[R1]
24 STR R4,[R0]
25 ADD R0,R0,#4
26 ADD R1,R1,#4
27 SUBS R2,R2,#1
28 BNE COPY
29
30 SWI 0x01

```

Screen shot of Register set output:

```

RegistersView
General Purpose
Hexadecimal
Unsigned Decimal
Signed Decimal
R0 : 4232
R1 : 4272
R2 : 0
R3 : 0
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) : 1
Carry (C) : 1
Overflow (V) : 0
IRQ Disable : 1
FIQ Disable : 1
Thumb (T) : 0
CPU Mode : S
-----
0x600000df

00001060: .DATA
00001060: A: .WORD 1,2,3,4,5,6,7,8,9,10
00001088: B: .WORD 0,0,0,0,0,0,0,0,0,0
00001000: .TEXT
00001000: E59F0048 LDR R0,=A
00001004: E59F1048 LDR R1,=B
00001008: E3A0200A MOV R2,#10
0000100C: E2800024 ADD R0,R0,#36
00001010: LOOP:
00001010: E5904000 LDR R4,[R0]
00001014: E5814000 STR R4,[R1]
00001018: E2400004 SUB R0,R0,#4
0000101C: E2811004 ADD R1,R1,#4
00001020: E2522001 SUBS R2,R2,#1
00001024: 1AFFFFF9 BNE LOOP
00001028: E59F0020 LDR R0,=A
0000102C: E59F1020 LDR R1,=B
00001030: E3A0200A MOV R2,#10
00001034: COPY:
00001034: E5914000 LDR R4,[R1]
00001038: E5804000 STR R4,[R0]
0000103C: E2800004 ADD R0,R0,#4
00001040: E2811004 ADD R1,R1,#4
00001044: E2522001 SUBS R2,R2,#1
00001048: 1AFFFFF9 BNE COPY
0000104C: SWI 0X011

Memory View
1088
Word Size
8B 16B 32B
00001088 0000000A 00000009 00000008 00000007 00000006 00000005 00000004 00000003 00000002 00000001 81818181 81818181 81818181 81818181
000010C0 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181
000010F8 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181
00001130 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181
00001168 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

```

- 6 Write an ALP using ARM7TDMI to find the largest of all the BCD digits of a given 32bit number. (hint: If R1=17845374 the largest digit is 8)

Program screen shot:

```

week-4 > 6.s
1  .DATA
2  NUM: .WORD 0X3248345
3
4  .TEXT
5  LDR R0,=NUM
6  LDR R0,[R0]
7  MOV R1,#0
8  MOV R2,#7
9
10 EXTRACT:
11 AND R3,R0,#0XF
12 CMP R3,R1
13 BLE LESS
14 MOV R1,R3
15
16 LESS:
17 MOV R0,R0,LSR #4
18 SUBS R2,R2,#1
19 BNE EXTRACT
20 SWI 0X011

```

Screen shot of Register set output:

ARMSim - The ARM Simulator Dept. of Computer Science

File View Cache Debug Watch Help

RegistersView 6.s

General Purpose

Hexadecimal

Unsigned Decimal

Signed Decimal

R0 : 0

R1 : 8

R2 : 0

R3 : 3

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 : S

0x600000df

.DATA

00001034: NUM: .WORD 0X3248345

.TEXT

00001000:E59F0028 LDR R0,=NUM

00001004:E5900000 LDR R0,[R0]

00001008:E3A01000 MOV R1,#0

0000100C:E3A02007 MOV R2,#7

00001010: EXTRACT:

00001010:E200300F AND R3,R0,#0XF

00001014:E1530001 CMP R3,R1

00001018:DA000000 BLE LESS

0000101C:E1A01003 MOV R1,R3

00001020: LESS:

00001020:E1A00220 MOV R0,R0,LSR #4

00001024:E2522001 SUBS R2,R2,#1

00001028:1AFFFFF8 BNE EXTRACT

0000102C:EF000011 SWI 0X011

00001030:00001034

Extra

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.

and

For the same transfer the block with auto-indexing.