



PES UNIVERSITY

Department of Computer Science & Engineering

Microprocessor & Computer Architecture Lab

UE23CS251B

WEEK 3 submission

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

UE23CS251B

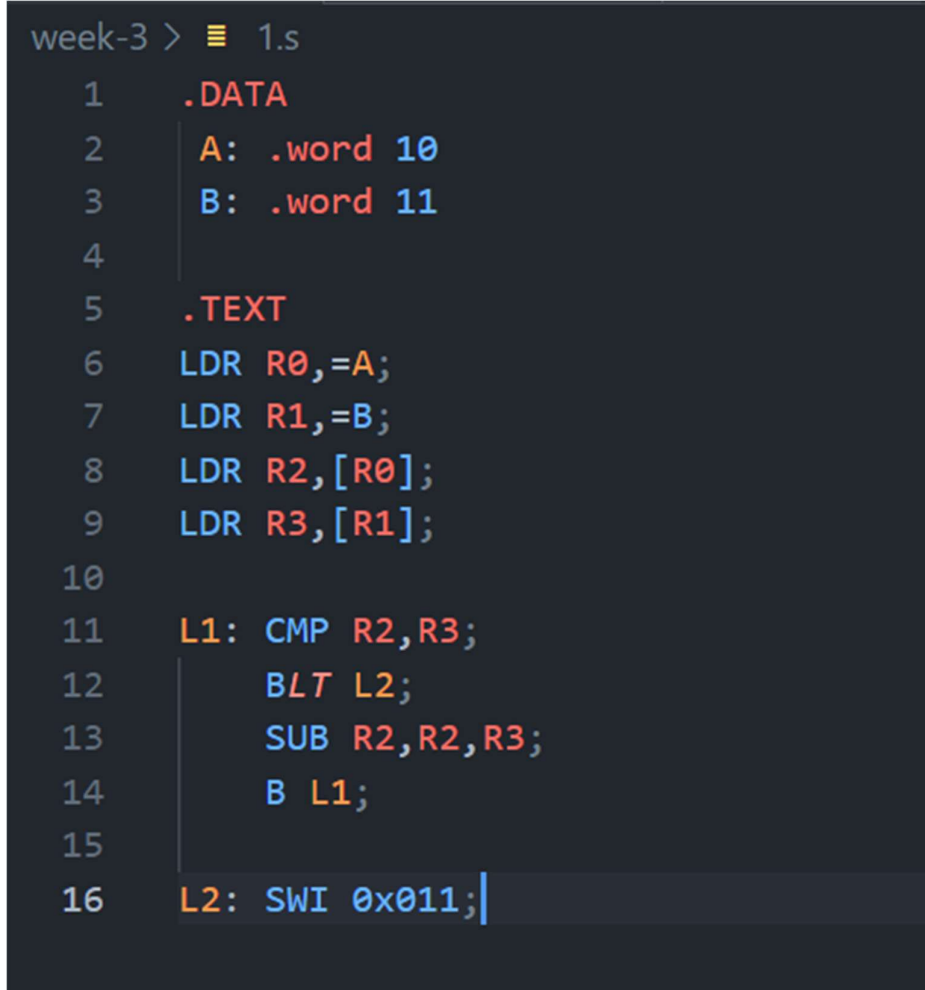
1 Write an ALP using ARM7TDMI to find the remainder of a number.(ie 10/3, remainder is 1)

.DATA

A: .word 10

B: .word 3

Program screen shot:

A screenshot of a text editor showing an ARM7TDMI assembly program. The editor has a dark background with light-colored text. The program is written in assembly language and is numbered from 1 to 16. The code defines two data words, A and B, and then performs a division operation to find the remainder of 10 divided by 3. The program uses the LDR, CMP, BLT, SUB, B, and SWI instructions. The program is saved as 1.s.

```
week-3 > 1.s
1  .DATA
2  A: .word 10
3  B: .word 11
4
5  .TEXT
6  LDR R0,=A;
7  LDR R1,=B;
8  LDR R2,[R0];
9  LDR R3,[R1];
10
11 L1: CMP R2,R3;
12     BLT L2;
13     SUB R2,R2,R3;
14     B L1;
15
16 L2: SWI 0x011;
```

Screen shot of Register set output:

The screenshot shows the ARMSim interface. The 'RegistersView' window on the left displays the state of the ARM7TDMI registers. The 'Disassembly' window on the right shows the assembly code for the file '1.s'.

RegistersView:

- General Purpose: Signed Decimal
- R0: 4140
- R1: 4144
- R2: 10
- R3: 11
- 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): 4128
- 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: S
- 0x800000df

Disassembly (1.s):

```

.DATA
0000102C:      A: .word 10
00001030:      B: .word 11

.TEXT
00001000:E59F001C  LDR R0,=A;
00001004:E59F101C  LDR R1,=B;
00001008:E5902000  LDR R2,[R0];
0000100C:E5913000  LDR R3,[R1];

00001010:E1520003  L1: CMP R2,R3;
00001014:BA000001      BLT L2;
00001018:E0422003  SUB R2,R2,R3;
0000101C:EAF0FFFB  B L1;

00001020:      L2: SWI 0x011;
  
```

- 2 Write an ALP using ARM7TDMI to search for an element in an array of 16 bit each using Linear search technique

.DATA

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

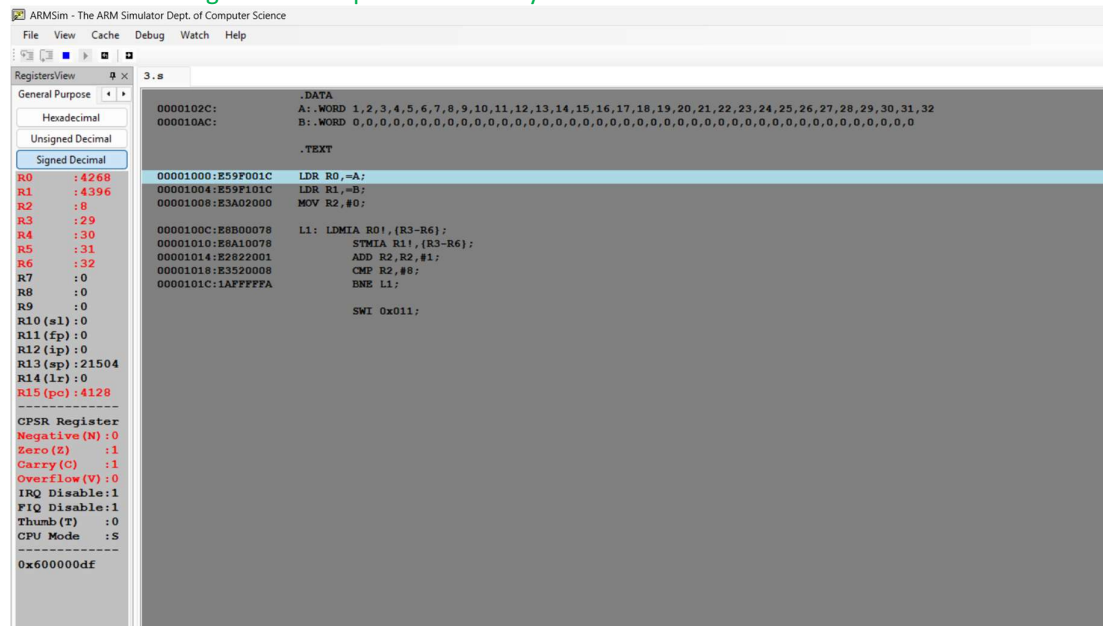
Program screen shot:

```
week-3 > ≡ 2.s
1  .DATA
2  A: .HWORD 1,2,3,4,5,6,7,8,9
3  K: .HWORD 3
4  I: .HWORD 0
5
6  .TEXT
7  LDR R0,=A;
8  LDR R1,=K;
9  LDRH R5,[R1];
10 LDR R4,=I;
11
12 MOV R2,#0 ;Index
13 L1: LDRH R3,[R0];
14     ADD R0,R0,#2;
15     CMP R3,R5;
16     BEQ L2;
17     ADD R2,R2,#1;
18     CMP R2,#8;
19     BLT L1;
20     MOV R2,#-1;
21
22 L2: STRH R2,[R4];
23
24 SWI 0x011
```

Screen shot of Register set output and memory location:


```
week-3 > ≡ 3.s  
1 .DATA  
2 A:.WORD 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,2  
3 B:.WORD 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
4  
5 .TEXT  
6  
7 LDR R0,=A;  
8 LDR R1,=B;  
9 MOV R2,#0;  
10  
11 L1: LDmia R0!,{R3-R6};  
12 STMIA R1!,{R3-R6};  
13 ADD R2,R2,#1;  
14 CMP R2,#8;  
15 BNE L1;  
16  
17 SWI 0x011;
```

Screen shot of Register set output and memory location:



- | | |
|---|---|
| 4 | Write an ALP using ARM7TDMI, for the given matrix arranged in row 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) |
|---|---|

1	00000000	1 2 3	A(0,0)	A(0,1)	A(0,2)
2	00000001	A = 4 5 6 =>	A(1,0)	A(1,1)	A(1,2)
3		7 8 9	A(2,0)	A(2,1)	A(2,2)
4					
5					
6					
7					
8	00000020				
9					

Row Major Order for a 3 x 3 matrix :

A [2, 2] = Row Num. x No. of elements per row + Column Num.

$2 \times 3 + 2 = 8^{\text{th}}$ location.

- Since each location is of 4 bytes, the address of the 3rd element in the 2nd row is
- row is
- $8 \times 4 = 32$.
- This is achieved in ARM processor, using MLA instruction as follows.

MLA	R1,	R2,	R3,	R4
Mnemonic	Dst	Row #	# of elements per row	col #

* To get consecutive data elements in a @D Array.

Program screen shot:

```

week-3 > 4.s
1  .data
2  A: .word 1,2,3,4,5,6,7,8,9
3
4  .text
5  LDR R0,=A      @ Load base address of array A
6  MOV R9,#4      @ Size of each element (4 bytes)
7  MOV R1,#2      @ Column number (0-based)
8  MOV R2,#2      @ Row number (0-based)
9  MOV R3,#3      @ Number of columns in matrix
10
11  MLA R4,R2,R3,R1 @ R4 = (row * num_columns) + col
12  @ R4 now contains the index
13
14  @ Calculate address
15  MUL R5,R4,R9    @ R5 = index * 4
16  ADD R6,R0,R5    @ R6 = base_address + offset
17
18  @ Now:
19  @ R4 contains the index of the element
20  @ R6 contains the memory address of the element
21

```

Screen shot of Register set output and memory location:

The screenshot shows the ARMSim - The ARM Simulator interface. On the left, the RegistersView is open, displaying the state of various registers. The General Purpose registers (R0-R15) are shown in hexadecimal and decimal. R0 is 4132, R1 is 2, R2 is 2, R3 is 3, R4 is 8, R5 is 32, R6 is 4164, R7 is 0, R8 is 0, R9 is 4, R10 (s1) is 0, R11 (fp) is 0, R12 (ip) is 0, R13 (sp) is 21504, R14 (lr) is 0, and R15 (pc) is 70656. The CPSR Register is also shown with fields like Negative (N), Zero (Z), Carry (C), Overflow (V), IRQ Disable, FIQ Disable, Thumb (T), and CPU Mode (S). The main window displays the assembly code for a program named 4.s. The code includes a data section with a word array A containing values 1 through 9, and a text section with instructions to load the base address of array A, calculate the index of an element, and calculate its memory address. Comments explain the purpose of each instruction.

```

RegistersView
General Purpose
Hexadecimal
Unsigned Decimal
Signed Decimal
R0 : 4132
R1 : 2
R2 : 2
R3 : 3
R4 : 8
R5 : 32
R6 : 4164
R7 : 0
R8 : 0
R9 : 4
R10 (s1) : 0
R11 (fp) : 0
R12 (ip) : 0
R13 (sp) : 21504
R14 (lr) : 0
R15 (pc) : 70656
-----
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

4.s
.data
A: .word 1,2,3,4,5,6,7,8,9

.text
00001000:E59F0018 LDR R0,=A @ Load base address of array A
00001004:E3A09004 MOV R9,#4 @ Size of each element (4 bytes)
00001008:E3A01002 MOV R1,#2 @ Column number (0-based)
0000100C:E3A02002 MOV R2,#2 @ Row number (0-based)
00001010:E3A03003 MOV R3,#3 @ Number of columns in matrix

00001014:E0241392 MLA R4,R2,R3,R1 @ R4 = (row * num_columns) + col
@ R4 now contains the index

@ Calculate address
00001018:E0050994 MUL R5,R4,R9 @ R5 = index * 4
0000101C:E0806005 ADD R6,R0,R5 @ R6 = base_address + offset

@ Now:
@ R4 contains the index of the element
@ R6 contains the memory address of the element

```

Assignments Questions

- 5 a)Write an ALP using ARM7TDMI to perform Convolution using MUL instruction (Addition of multiplication of respective numbers of loc A and loc B)
 b Write an ALP using ARM7TDMI to perform Convolution using MLA instruction (Addition of multiplication of respective numbers of loc A and loc B).

Program screen shot:

The screenshot shows the ARM7TDMI assembly code for a convolution program. The code is written in a dark-themed editor. It starts with a comment 'week-3 > 5.s'. The code includes a text section with instructions to initialize the accumulator R6 to 0, load values from arrays A and B into registers R3 and R4, multiply them, accumulate the result in R6, decrement a loop counter, and branch back to the start of the loop if the counter is not zero. Finally, it stores the result in array C and ends the program.

```

week-3 > 5.s
6 .text
11 MOV R6, #0 @ Initialize accumulator R6 to 0
12
13 L1:
14 LDR R3, [R0], #4 @ Load value from array A into R3 a
15 LDR R4, [R1], #4 @ Load value from array B into R4 a
16 MUL R5, R3, R4 @ Multiply R3 and R4, store result
17 ADD R6, R6, R5 @ Accumulate the result in R6
18 SUBS R10, R10, #1 @ Decrement loop counter
19 BNE L1 @ If counter is not zero, branch to
20
21 STR R6, [R2] @ Store the final result in array C
22
23 @ End of program

```


Screen shot of Register set output:

The screenshot shows the ARM Simulator interface. On the left, the 'RegistersView' panel displays the state of the registers. The 'General Purpose' register set is selected, and the 'Signed Decimal' view is chosen. The registers R0 through R15 are listed with their current values. R0 is 0, R1 is 0, R2 is 4196, R3 is 5, R4 is 1, R5 is 5, R6 is 35, R7 is 0, R8 is 0, R9 is 0, R10 (s1) is 0, R11 (fp) is 0, R12 (sp) is 0, R13 (sp) is 21504, R14 (lr) is 0, and R15 (pc) is 70656. Below the registers, the CPSR Register is shown with various flags set to 0 or 1, and the CPU Mode is set to S.

The main window displays the assembly code for the program. The code is organized into sections: .data and .text. The .data section defines three arrays: A (1, 2, 3, 4, 5), B (5, 4, 3, 2, 1), and C (0, 0, 0, 0, 0). The .text section contains the main logic of the program, including loading addresses, initializing the accumulator, and a loop that calculates the product of elements from arrays A and B, storing the result in array C. The program ends with a comment indicating the end of the program.

Program screen shot:

The screenshot shows the program code in the ARM Simulator. The code is organized into sections: .data and .text. The .data section defines three arrays: A (1, 2, 3, 4, 5), B (5, 4, 3, 2, 1), and C (0, 0, 0, 0, 0). The .text section contains the main logic of the program, including loading addresses, initializing the accumulator, and a loop that calculates the product of elements from arrays A and B, storing the result in array C. The program ends with a comment indicating the end of the program.

Screen shot of Register set output:

The screenshot shows the ARMSim - The ARM Simulator interface. The left pane displays the Register View for the 5-2.s file, showing registers R0 through R15 and CPSR. The right pane displays the Assembly View for the same file, showing the assembly code.

Register View (5-2.s):

Register	Value
R0	:0
R1	:4
R2	:4192
R3	:5
R4	:5
R5	:0
R6	:55
R7	:0
R8	:0
R9	:0
R10 (s1)	:0
R11 (fp)	:0
R12 (ip)	:0
R13 (sp)	:21504
R14 (lr)	:0
R15 (pc)	:70656

CPSR Register:

Field	Value
Negative (N)	:0
Zero (Z)	:1
Carry (C)	:1
Overflow (V)	:0
IRQ Disable	:1
FIQ Disable	:1
Thumb (T)	:0
CPU Mode	:S

Assembly View (5-2.s):

```

.data
00001038:      A: .word 1, 2, 3, 4, 5
0000104C:      B: .word 1, 2, 3, 4, 5
00001060:      C: .word 0, 0, 0, 0, 0

.text
00001000:E3A0A005      MOV R10, #5           @ Number of elements in arrays A and B
00001004:E59F0020      LDR R0, =A           @ Load address of array A into R0
00001008:E59F1020      LDR R1, =B           @ Load address of array B into R1
0000100C:E59F2020      LDR R2, =C           @ Load address of array C into R2
00001010:E3A06000      MOV R6, #0           @ Initialize accumulator R6 to 0

00001014:      L1:
00001014:E4903004      LDR R3, [R0], #4      @ Load value from array A into R3 and increment R0 by 4
00001018:E4914004      LDR R4, [R1], #4      @ Load value from array B into R4 and increment R1 by 4
0000101C:E0266493      MLA R6, R3, R4, R6     @ Multiply R3 and R4, add to R6, and store result in R6
00001020:E25AA001      SUBS R10, R10, #1      @ Decrement loop counter
00001024:1AFFFFFA      BNE L1               @ If counter is not zero, branch to L1

00001028:E5826000      STR R6, [R2]          @ Store the final result in array C

@ End of program

```

- 6 Write an ALP using ARM7TDMI to find the sum of all the BCD digits of a given 32 bit number. (hint:788 =7+8+8)

Program screen shot:

The screenshot shows the ARM7TDMI assembly code for finding the sum of BCD digits. The code is written in a text editor with a dark background.

```

week-3 > 6.s
5  .text
6  .global main
7  main:
8      LDR R0, =Number
9      LDR R1, [R0]
10     MOV R2, #0
11     MOV R3, #8
12
13     L1:
14     AND R4, R1, #0xF
15     ADD R2, R2, R4
16     MOV R1, R1, LSR #4
17     SUBS R3, R3, #1
18     BNE L1
19
20     LDR R0, =Sum
21     STR R2, [R0]
22

```

Screen shot of Register set output:

The screenshot displays the ARMSim - The ARM Simulator interface. The top menu bar includes File, View, Cache, Debug, Watch, and Help. Below the menu is a toolbar with icons for file operations and execution. The main window is divided into two panes. The left pane, titled 'RegistersView', shows the 'General Purpose' register set. The 'Signed Decimal' tab is selected, displaying the following values: R0: 0, R1: 0, R2: 23, 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, and R15 (pc): 70656. Below the register list, the CPSR Register status is shown: Negative (N): 0, Zero (Z): 1, Carry (C): 1, Overflow (V): 0, IRQ Disable: 1, FIQ Disable: 1, Thumb (T): 0, and CPU Mode: S. The right pane, titled '6.s', shows the assembly code for a program. The code includes a data section with a variable 'Number' at address 00001034 and a text section with a 'main' function starting at address 00001000. The assembly instructions are as follows: 00001000: E59F0024 LDR R0, =Number; 00001004: E5901000 LDR R1, [R0]; 00001008: E3A02000 MOV R2, #0; 0000100C: E3A03008 MOV R3, #8; 00001010: L1: AND R4, R1, #0xF; 00001014: E0822004 ADD R2, R2, R4; 00001018: E1A01221 MOV R1, R1, LSR #4; 0000101C: E2533001 SUBS R3, R3, #1; 00001020: 1AFFFFFA BNE L1; 00001024: E59F0004 LDR R0, =Sum; 00001028: STR R2, [R0].