Computer Organization andArchitecture (EET2211)

LAB IV: Evaluate Different Arithmetic Operations and Logical operations on two32-bit data using ARM processor.

Siksha 'O' Anusandhan (Deemed to be University),Bhubaneswar

Branch: CSE		Section: 44	
S. No.	Name	Registration No.	Signature
27	E. JAGADEESWAR PATRO	2241016309	E. Jagadeeswar Patro

Marks:_____/10

Remarks:

Teacher's Signature

I. OBJECTIVE:

1. Perform Addition and Subtraction of two 32-bit numbers using data processingaddressing mode (with immediate data).
2. Perform Addition, Subtraction, and Multiplication of two 32-bit numbers usingload/store addressing mode.
3. Perform the logical operations (AND, OR, XOR, and NOT) on two 32-bit numbers using load/store addressing mode.

II. PRE-LAB

Note: For each objective in prelab describe the following points:

- Write the pseudocode.
- Write the assembly code with a description (ex. Mov ax,3000h – ax<-3000h)
- Examine & analyze the input/output of assembly code.

Objective 1:

-> Pseudocode :

1. Set the value 0x40 into register R0
2. Set the value 0x50 into R_1
3. Add R_0 and R_1 and store in R_2
4. Subtract R_1 from R_0 and store in R_3
5. Terminate program

-> Assembly Code :

```
.global _start
_start:
    mov r0, #0x40
    mov r1, #0x50
    adds r2, r0, r1
    subs r3, r0, r1
    mul r4, r0, r1
my_exit: b my_exit    // infinite loop terminates the code
```

Objective 2:

→ Pseudocode:

1. Load R_0 with 0x10100000 memory location
2. Load value at address [10100000] into R_1
3. Load value at [10100004] into R_2
4. Add R_1 and R_2 and store in [10100008]
5. Subtract R_2 from R_1 and store at [1010000C]
6. Multiply R_1 and R_2 and store at [10100010]
7. Terminate Program.

→ Assembly Code:

```
.global _start
_start:
    ldr r0, =0x10100000   // load register
    ldr r1, [r0], #4      // address +4
    ldr r2, [r0], #4
    adds r3, r1, r2
    str r3, [r0], #4
    subs r4, r1, r2
    str r4, [r0], #4
    mul r5, r1, r2
    str r5, [r0]
my_exit: b my_exit
```

Objective 3:

-> pseudocode:

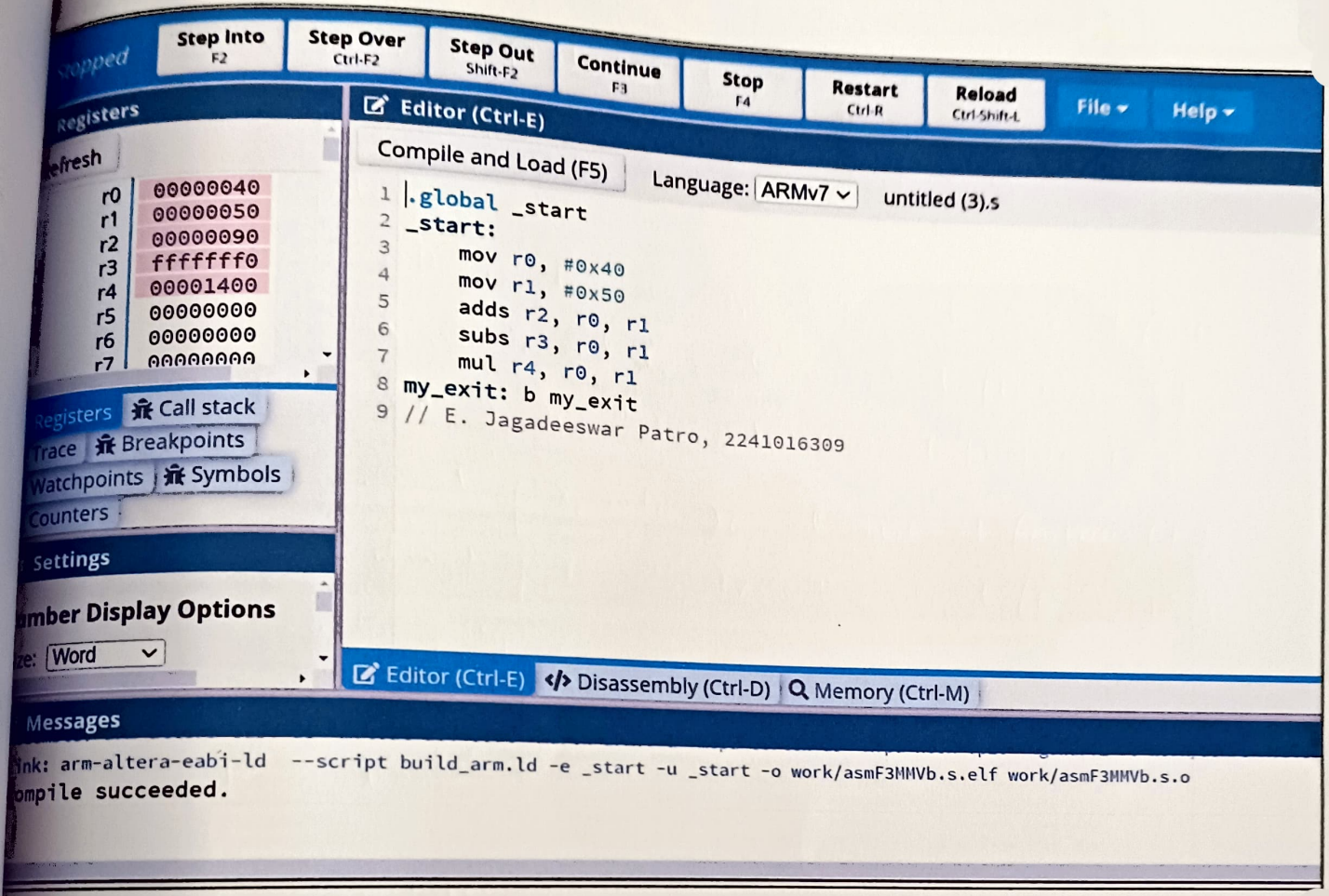
1. load r0 with [10100000] memory location
2. load r_1 from [10100000] and r_2 from [10100004] for input.
3. Perform AND operation of r_1 and r_2 and store at [10100008]
4. Perform OR on r_1 and r_2 and store at [1010000c]
5. Perform XOR on r_1 and r_2 and store at [10100010]
6. Copy compliment of r_1 into r_6 and store at [10100014]
7. Copy compliment of r_2 into r_7 and store at [10100018]
8. Terminate program.

-> Assembly Code:

```
.global _start
_start:
    ldr r0, =0x10100000
    ldr r1, [r0], #4
    ldr r2, [r0], #4
    and r3, r1, r2      // AND operation
    str r3, [r0], #4
    orr r4, r1, r2      // OR operation
    str r4, [r0], #4
    eor r5, r1, r2      // XOR operation
    str r5, [r0], #4
    mvn r6, r1          // NOT of r1
    str r6, [r0], #4
    mvn r7, r2          // NOT of r2
    str r7, [r0]
my_exit: b my_exit
```

III. LAB

Objective 1:



From this result, I have observed.....

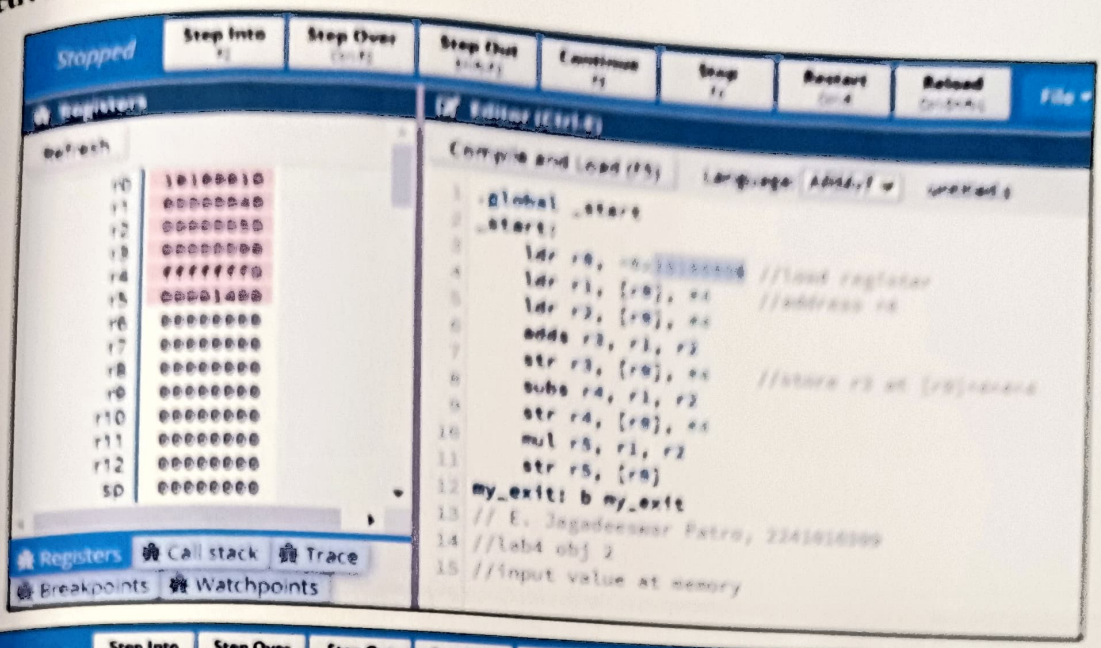
Input:

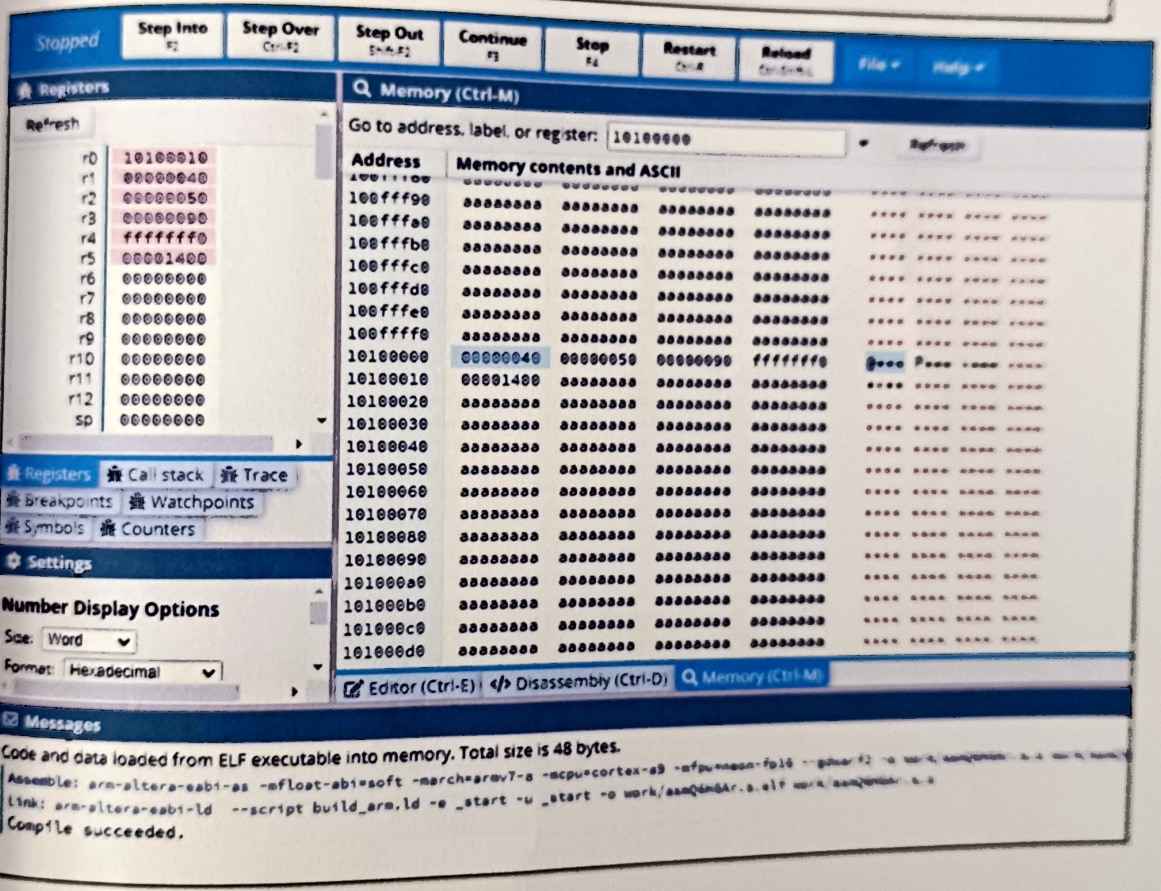
Sl. No.	Memory Location	Operand (Data)
1	r_0	0x40
2	r_1	0x50

Output:

Sl. No.	Memory Location	Operand (Data)
1	r_2	0x90
2	r_3	0xfffffff0
3	r_4	0x00001400

Objective 2:





From this result, I have observed.....

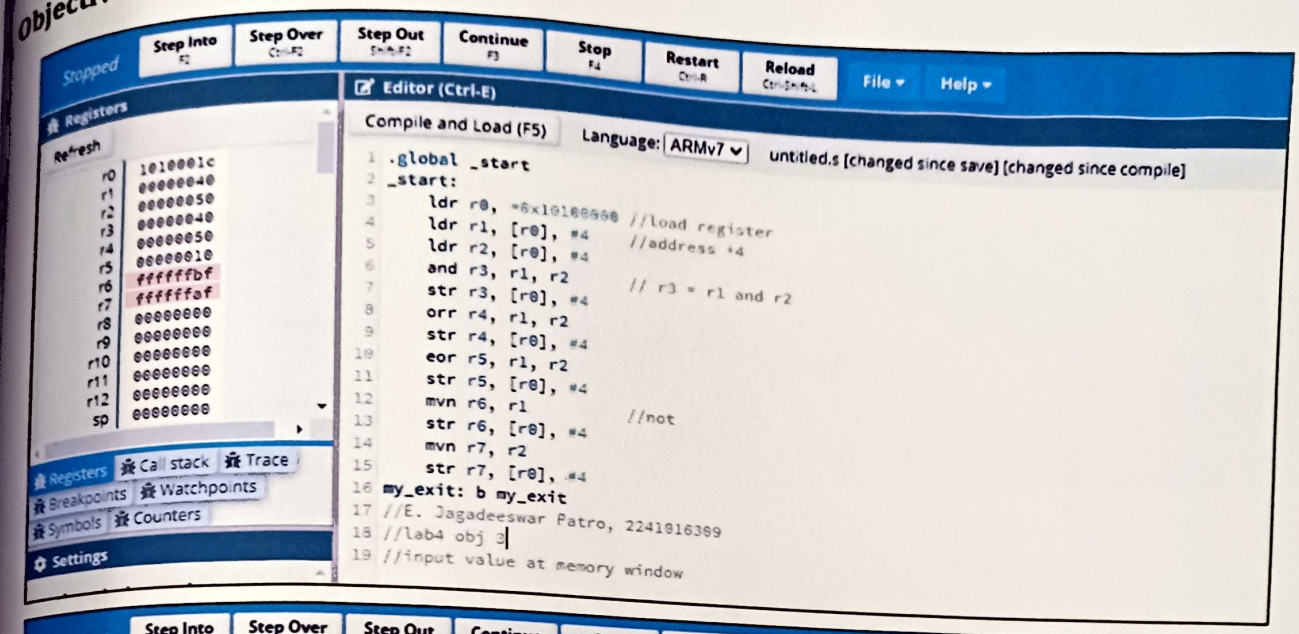
Input:

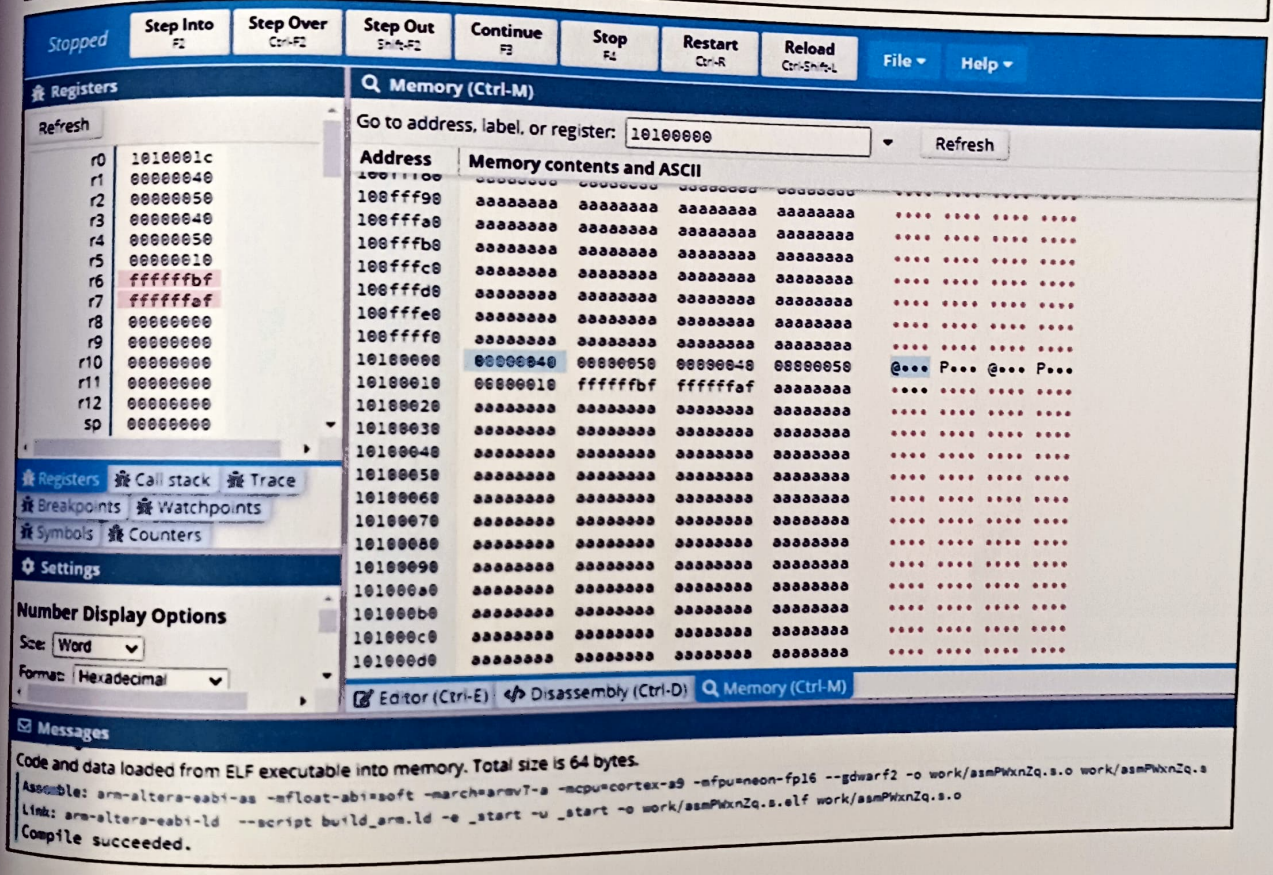
Sl. No.	Memory Location	Operand (Data)
1	10100000	0x40
2	10100004	0x50

Output:

Sl. No.	Memory Location	Operand (Data)
1	10100008	0x90
2	10100000c	0xffffffff0
3	10100010	0x00001400

Objective 3:





From this result, I have observed.....

Input:

Sl. No.	Memory Location	Operand (Data)
1	10100000	0x40
2	10100004	0x50

Output:

Sl. No.	Memory Location	Operand (Data)
1	10100008	0x40
2	1010000C	0x50
3	10100010	0x10
4	10100014	0xffffffbf
5	10100018	0xffffffaf

CONCLUSION

On successful completion of the three objectives, marks a significant achievement in demonstrating proficiency in ARM assembly language programming. Performing arithmetic operations and logical operations is successfully done using both immediate data addressing mode and load/store addressing mode.

POST LAB

Give any examples of five arithmetic and logical instructions.
Differentiate between LDR and STR instruction
Which of the following instructions is not valid
a) MOVR7.R2
b) LDR R1, =LABEL

Answers:

1) Any five examples are:-

a) ADD R1, R2, R3 : Adds values in R2 & R3 and stores in R1

b) SUB R1, R2, R3 : Performs R2 - R3 and stores in R1

c) MUL R1, R2, R3 : Performs R2 x R3 and stores in R1

d) AND R1, R2, R3 : AND between R2 and R3 is stored in R1

e) ORR R1, R2, R3 : OR between R2 and R3 is stored in R1

2)

LDR (Load)

-> Used to load data from memory into a register

-> Reads the data from the memory location specified by source address and loads into destination register.

Syntax: LDR destination, [source]

STR (store)

-> Used to store data from a register into memory.

-> Writes the data from the source register into the memory location specified by the destination address.

Syntax: STR source, [destination]

3) a) MOV R7. R2

It is incorrect as the instruction should be written as:

MOV R7, R2

in ARM7 assembly language.