Microprocessor and Assembly Programming Laboratory

B.Tech. III Semester

Name : KAUSHAL VASHISTH

Roll Number :18ETCS002147

Department : Computer Science and Engineering

Faculty of Engineering & Technology

Ramaiah University of Applied Sciences

NAME: Kaushal '

Ramaiah University of Applied Sciences

Private University Established in Karnataka State by Act No. 15 of 2013

Faculty	Engineering & Technology
Programme	B. Tech. in Computer Science and Engineering
Year/Semester	2018/3rd Semester
Name of the Laboratory	Microprocessor and Assembly Programming Laboratory
Laboratory Code	

List of Experiments

1. Data transfer operations	
2. Arithmetic operations	
3. Logical operations	
4. Controlling execution flow using conditional instructions	
5. String manipulation	
6. Searching an element in an array	
7. Sorting an array	
8. Interfacing	
9. Interfacing	

NAME: Kaushal Vashisth ROLL NO:- 18ETCS002147

No.	Lab Experiment	Viva (6)	Results (7)	Documentation (7)	Total Marks (20)
1	Data transfer operations				
2	Arithmetic operations				
3	Logical operations				
4	Controlling execution flow using conditional instructions				
5	String manipulation				
6	Searching an element in an array				
7	Sorting an array				
8	Interfacing				
9	Interfacing				
10	Lab Internal Test conducted along the lines of SEE and valued for 50 Marks and reduced for 20 Marks				
	Total Marks				

NAME: Kaushal Vashisth ROLL NO:- 18ETCS002147

Laboratory 3

Title of the Laboratory Exercise: Logical operations

1. Introduction and Purpose of Experiment

   Students will be able to perform all logical operations using assembly instructions.

2. Aim and Objectives

   Aim

   To develop assembly language program to perform all logical operations

   Objectives

   At the end of this lab, the student will be able to

   - Identify the appropriate assembly language instruction for the given logical operations
   - Perform all logical operations using assembly language instructions
   - Get familiar with assembly language program by developing simple programs

3. Experimental Procedure

   1. Write algorithm to solve the given problem
   2. Translate the algorithm to assembly language code
   3. Run the assembly code in GNU assembler
   4. Create a laboratory report documenting the work

4. Questions:

   1. Consider the following source code fragment

      int a,b,c,d;

      a= (b AND c) XOR d;

      a=(b XOR c) OR d;

      Assume that *b, c, d* are in registers. Develop an assembly language program to perform this assignment statements. Assume that *b, c* are in registers and *d* in memory. Develop an assembly language program to perform this assignment statements.

NAME: Kaushal Vashisth ROLL NO:- 18ETCS002147

2. Consider the following source code fragment

int a ,b, c, d;

*A = (b*c) / d;*

Perform multiplication and division by shift operations

5. Calculations/Computations/Algorithms:-

Algorithms:-

Q1:-

Ans:-

Step1:- define “a”, “b”, “c”, “d” in section data with values a=1;b=2;c=3;d=4;

Step2:- start

for operation *a= (b AND c) XOR d:--*

Step3:-move value of variable c to eax general purpose register.

Step4:- perform “and” operation between b and register “eax”.

Step5:- move value of variable “d” to “ebx” general purpose register.

Step6:- perform “xor” operation between register “eax” and register “ebx”.

Step7:- store the value of ebx register in a.

for operation *a= (b XOR c) OR d:--*

Step8:-move value of variable c to “ecx” general purpose register.

Step9:- perform “xor” operation between “b” and register “ecx”.

Step10:- move value of variable d to “esi” general purpose register.

Step11:- perform “or” operation between register “ecx” and register “esi”.

Step12:- store the value of “esi” register in a.

Step 13:- stop

NAME: Kaushal Vashisth ROLL NO:- 18ETCS002147

Q2:-

Ans:-

Step1:- define “a”, “b” in section data

Step2:- start

Step3:-move b to eax general purpose register.

Step4:- use command “sall $3 ,%ebx ” to multiply values by 8.

Step5:- move value of “eax” to “esi” general purpose register.

Step6:- use command “sarl $4 ,%ebx ” to multiply values by 16.

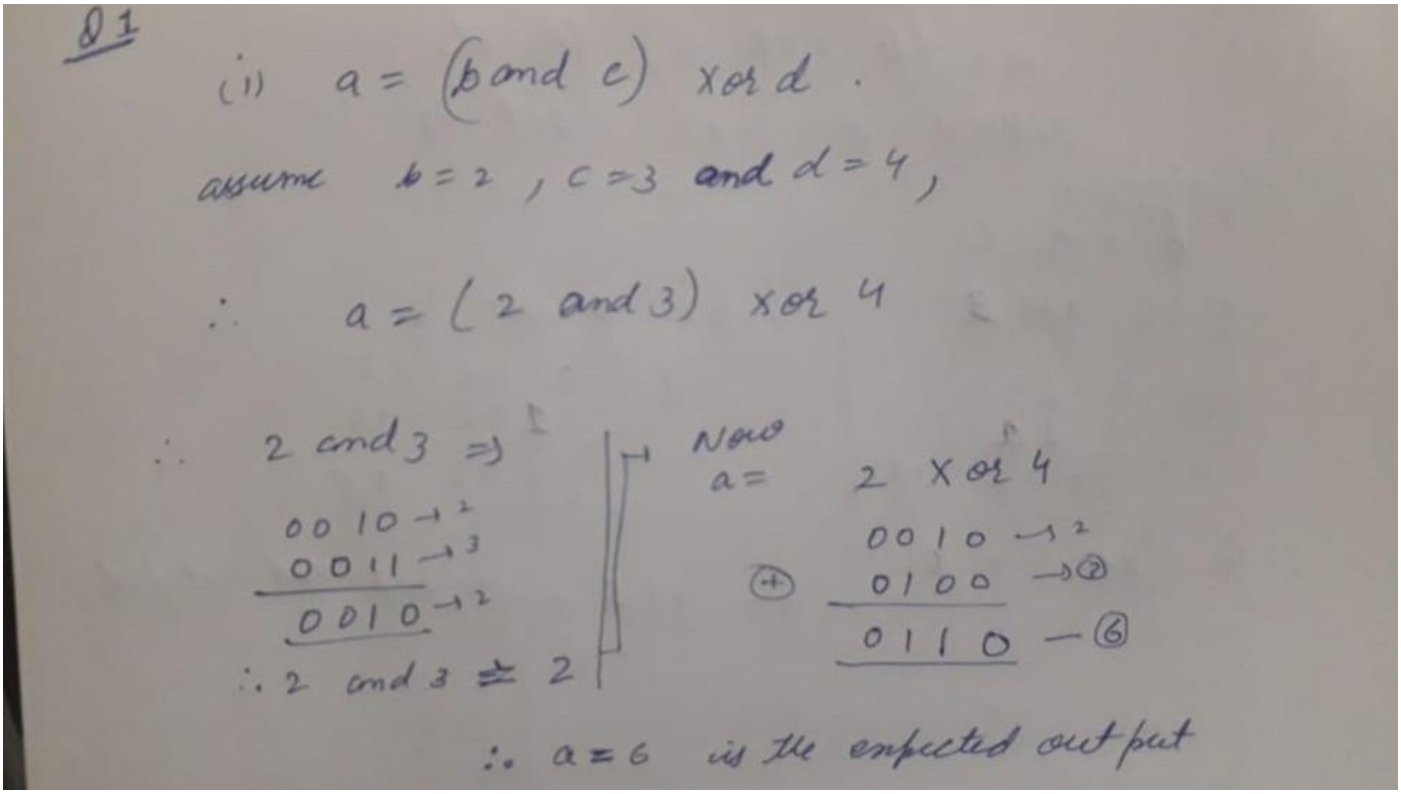
Step7:- :- store the value in a.

Step8:- stop

Manual Calculations for expected ouput:-

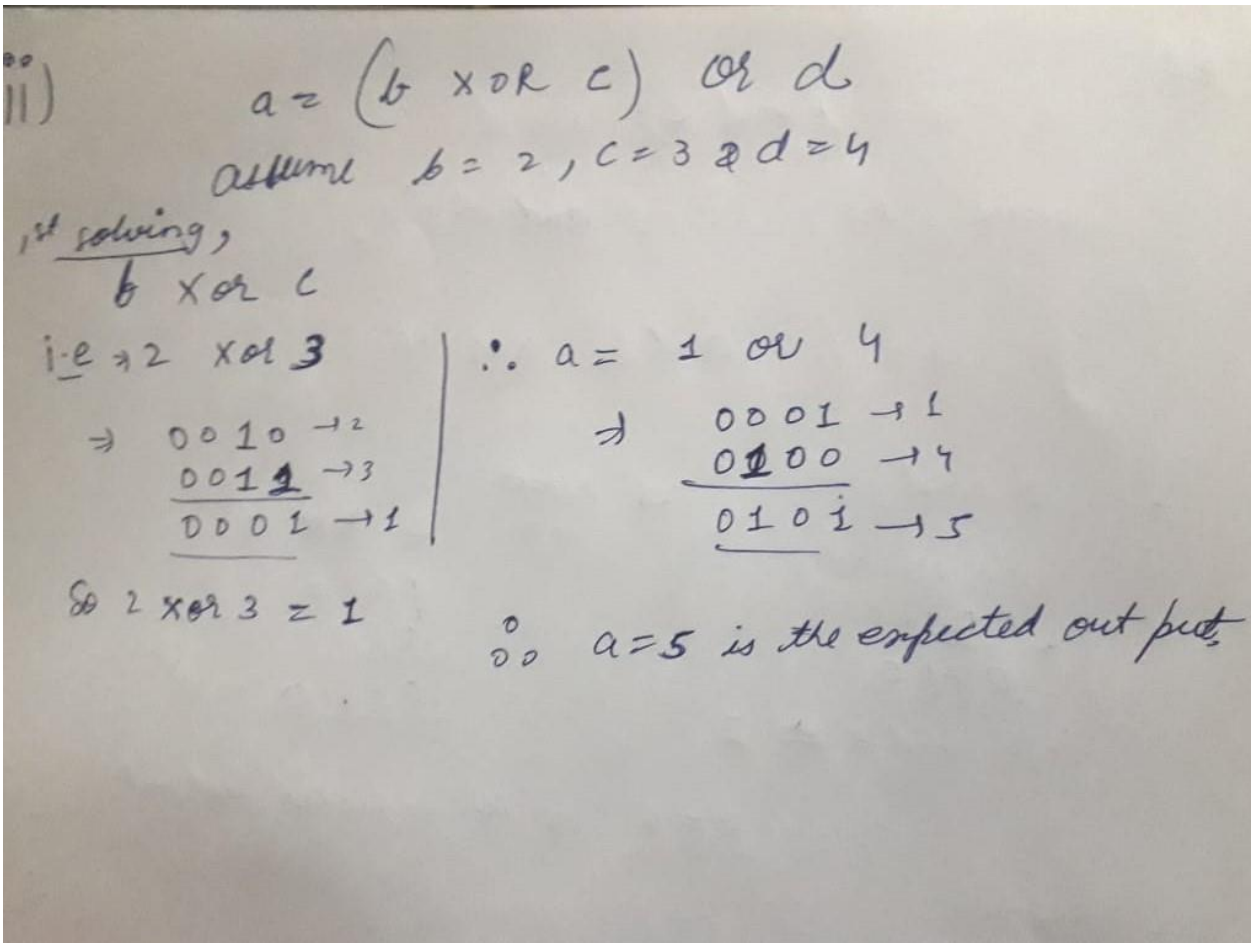
Question 1:-

Part 1:-



NAME: Kaushal Vashisth ROLL NO:- 18ETCS002147

Part2:-



6. Presentation of Results

```
1   .section .data
2       a:
3           .int 1
4       b:
5           .int 2
6       c:
7           .int 3
8       d:
9           .int 4
10  .section .text
11  .globl _start
12  _start:
13
14  movl c,%eax
15  andl b,%eax
16  movl d,%ebx
17  xorl %eax,%ebx
18
19  movl %ebx,a
20
21  movl c,%ecx
22  xorl b,%ecx
23  movl d,%esi
24  orl %ecx,%esi
25
26  movl %esi,a
27
28  movl $1,%eax
29  movl $0,%ebx
30  int $0*80
```

Figure1:- answer code for question 1

```
1   .section .data
2       b:
3           .int 30
4       a:
5           .int 1
6
7   .section .text
8   .globl _start
9   _start:
10
11  movl b,%eax
12  sall $3,%eax
13
14  movl %eax,%esi
15  sarl $4,%esi
16  movl %esi,a
17
18  movl $1,%eax
19  movl $0,%ebx
20  int $0*80
21
22
23
```

figure 2: answer code for question 2

NAME: Kaushal Vashisth ROLL NO:- 18ETCS002147

Program 2:-

*A = (b*c) / d;*

a= (b AND c) XOR d;

ans==

```
Breakpoint 1, _start () at lab2.s:21
21      movl c,%ecx
(gdb) info registers
eax            0x2      2
ecx            0x0      0
edx            0x0      0
ebx            0x6      6
esp            0xbffff040       0xbffff040
ebp            0x0      0x0
esi            0x0      0
edi            0x0      0
eip            0x804808d        0x804808d <_start+25>
eflags         0x206    [ PF IF ]
cs             0x73     115
ss             0x7b     123
ds             0x7b     123
es             0x7b     123
fs             0x0      0
gs             0x0      0
(gdb) print a
$1 = 6
(gdb)
```

Q2:- a=(b XOR c) OR d;

Ans:--

NAME: Kaushal Vashisth ROLL NO:- 18ETCS002147

```
Continuing.

Breakpoint 2, _start () at lab2.s:28
28      movl $1,%eax
(gdb) info registers
eax            0x2      2
ecx            0x1      1
edx            0x0      0
ebx            0x6      6
esp            0xbffff040       0xbffff040
ebp            0x0      0x0
esi            0x5      5
edi            0x0      0
eip            0x80480a7        0x80480a7 <_start+51>
eflags         0x206    [ PF IF ]
cs             0x73     115
ss             0x7b     123
ds             0x7b     123
es             0x7b     123
fs             0x0      0
gs             0x0      0
(gdb) print a
$2 = 5
(gdb)
```

Answer 2:-

```
Breakpoint 3, _start () at lab3.s:18
18      movl $1,%eax
(gdb) print a
$2 = 15
(gdb) info registers
eax            0xf0     240
ecx            0x0      0
edx            0x0      0
ebx            0x0      0
esp            0xbffff050       0xbffff050
ebp            0x0      0x0
esi            0xf      15
edi            0x0      0
eip            0x8048087        0x8048087 <_start+19>
eflags         0x206    [ PF IF ]
cs             0x73     115
ss             0x7b     123
ds             0x7b     123
es             0x7b     123
fs             0x0      0
gs             0x0      0
(gdb) print a
$3 = 15
(gdb)
```

7. Analysis and Discussions:-

   Logical operations are successfully executed in the GNU assembler :-

   The learning happened in this process can be summarized as follows:-

   - The processor instruction set provides the instructions AND, OR, XOR, TEST, and NOT Boolean logic, which tests, sets, and clears the bits according to the need of the program.
   - The format for these instructions –

Sr.No.	Instruction	Format
1	AND	AND operand1, operand2
2	OR	OR operand1, operand2
3	XOR	XOR operand1, operand2
4	TEST	TEST operand1, operand2
5	NOT	NOT operand1

   - The first in all the cases could be either in register or in memory. The second operand could be either in register/memory or an immediate (constant) value. However, memory-to-memory operations are not possible.

However the second question has some limitations like:-

   - Since division by zero is not possible hence, the program will show error for this specific case.

NAME: Kaushal Vashisth ROLL NO:- 18ETCS002147

8. Conclusions :-

   It can be concluded that the manually calculated outputs are equivalent to the outputs given by the GNU compiler.

   Also, by this lab the it can said that the concepts of Logical operations are thoroughly revised.

Signature and date Marks