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.

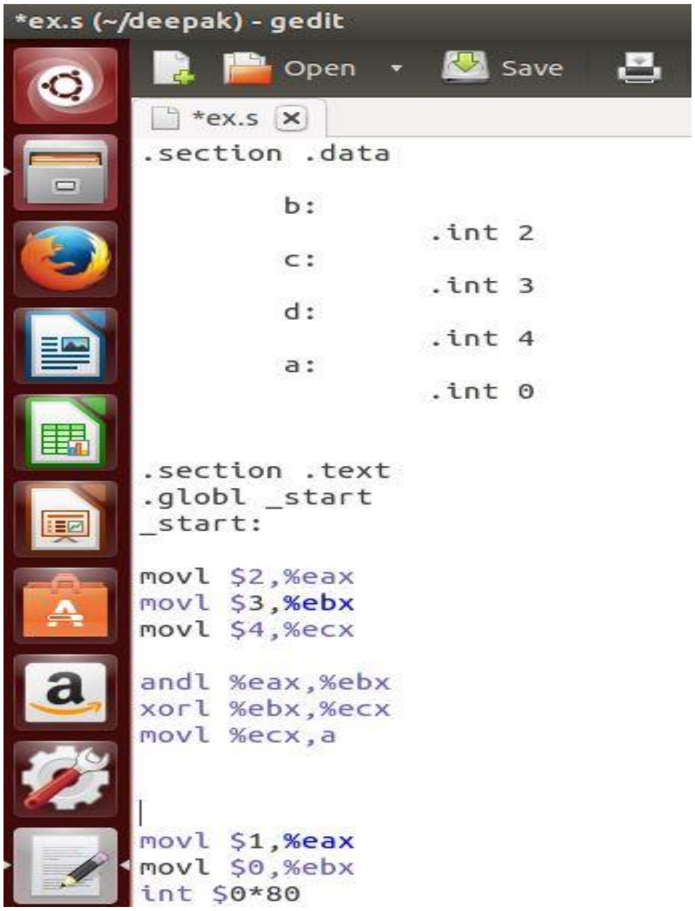
2. Consider the following source code fragment

   Int a,b,c,d;

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

   Perform multiplication and division by shift operations

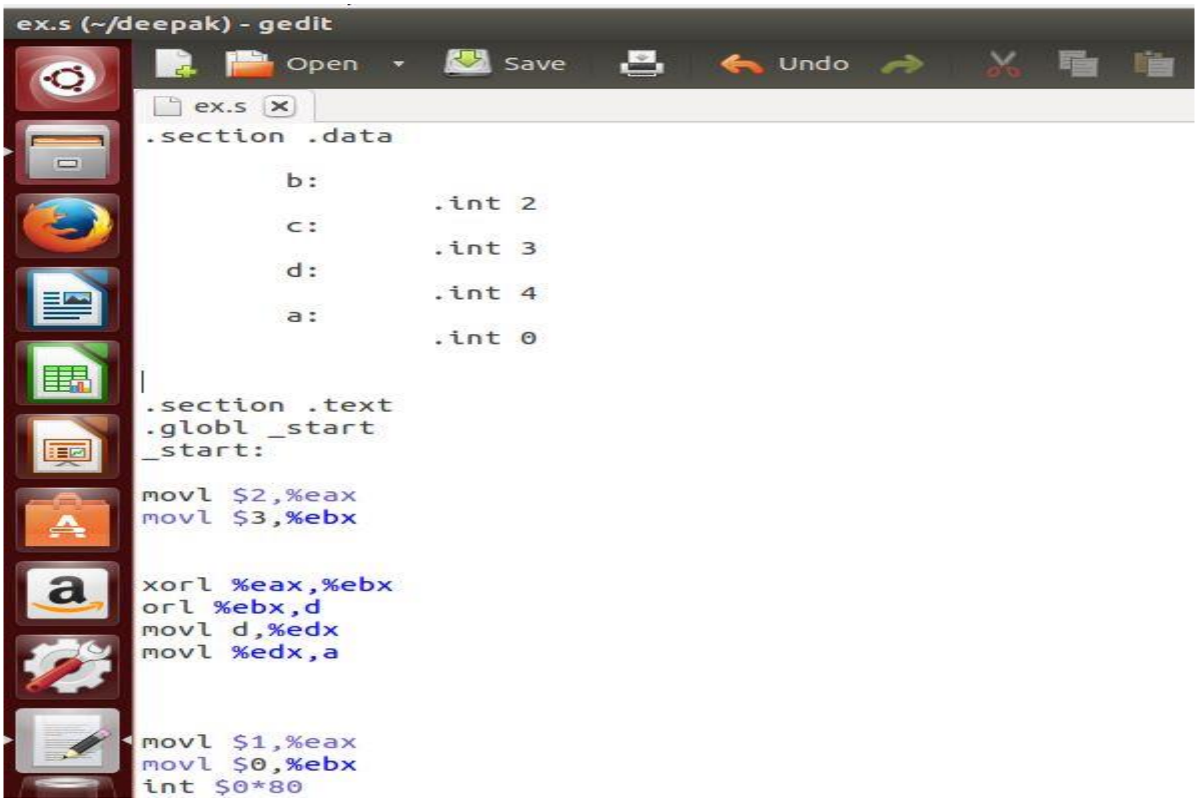
5. Calculations/Computations/Algorithms



```
.section .data

        b:
                .int 2
        c:
                .int 3
        d:
                .int 4
        a:
                .int 0


.section .text
.globl _start
_start:

movl $2,%eax
movl $3,%ebx
movl $4,%ecx

andl %eax,%ebx
xorl %ebx,%ecx
movl %ecx,a



movl $1,%eax
movl $0,%ebx
int $0*80
```

Fig 1 program to perform *a= (b AND c) XOR d*

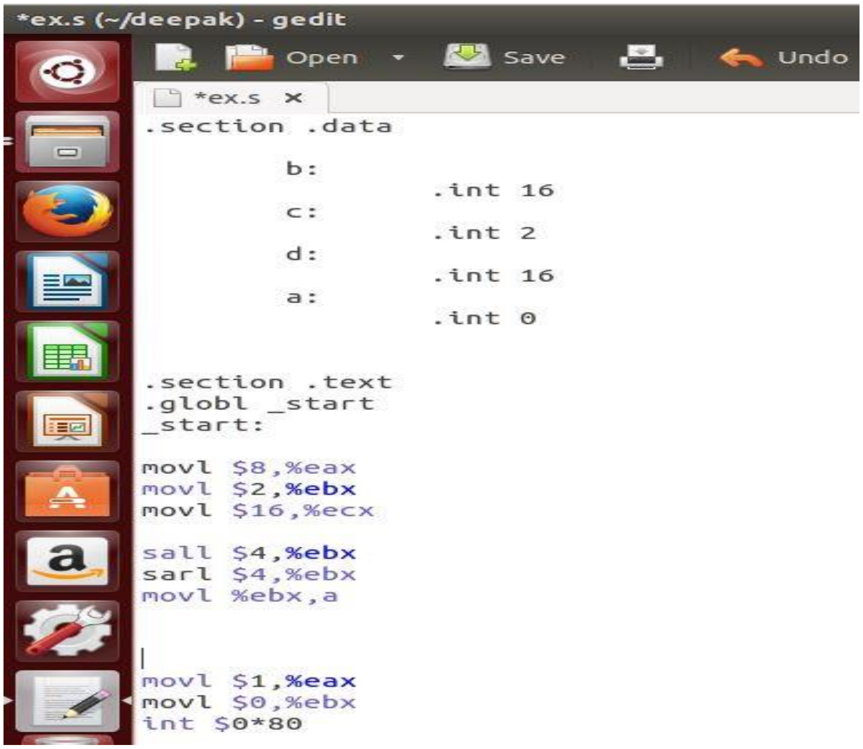
In this algorithm as the question suggests, it uses all the inputs as general purpose registers. By using "mov" command the assigned values of b, c, d are moved to the registers eax, ebx, ecx , respectively. As per the given expression a= (b AND c) XOR d , according to "Bodmas rule" we first perform And operation on b and c which are in the registers eax abd ebx the xor operation from this result to d containing register .And then we move the answer to memory a.



```
.section .data

        b:
                .int 2
        c:
                .int 3
        d:
                .int 4
        a:
                .int 0

.section .text
.globl _start
_start:

movl $2,%eax
movl $3,%ebx

xorl %eax,%ebx
orl %ebx,d
movl d,%edx
movl %edx,a


movl $1,%eax
movl $0,%ebx
int $0*80
```

Fig 2 program to perform *a=(b XOR c) OR d;*

In this algorithm as the question suggests, it uses all the inputs as general purpose registers. By using "mov" command the assigned values of b, c, are moved to the registers eax, ebx, , respectively. And d in memory As per the given expression a=(b XOR c) OR d;, according to "Bodmas rule" we first perform xor operation on b and c which are in the registers eax abd ebx the or operation from this result to d .And then we move the answer to memory a.

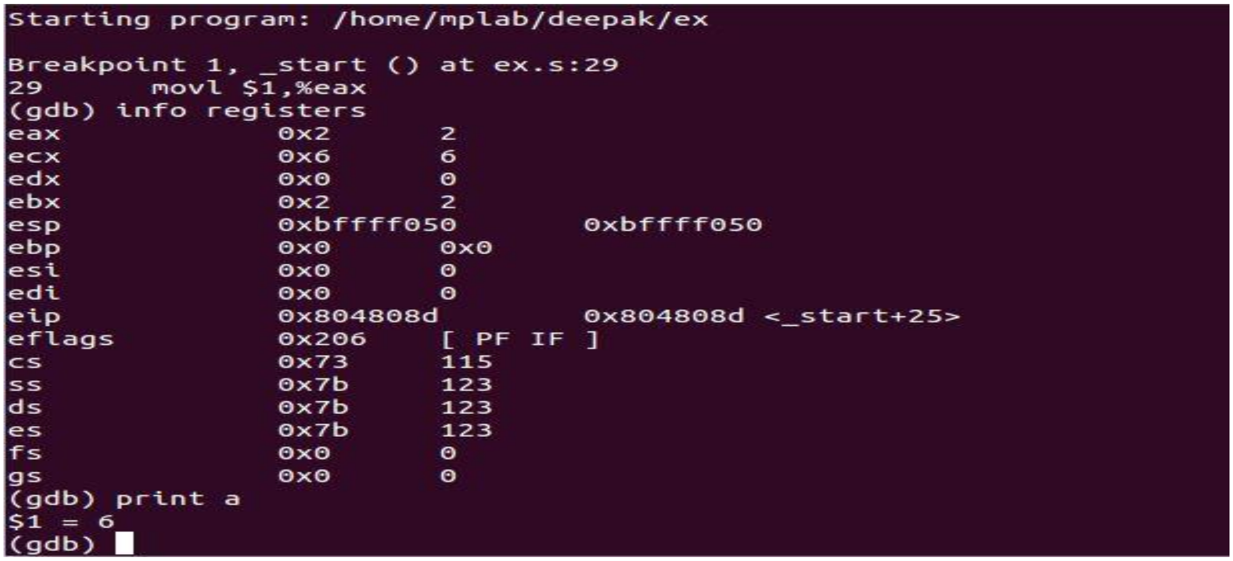


```
.section .data

        b:
                .int 16
        c:
                .int 2
        d:
                .int 16
        a:
                .int 0


.section .text
.globl _start
_start:

movl $8,%eax
movl $2,%ebx
movl $16,%ecx

sall $4,%ebx
sarl $4,%ebx
movl %ebx,a


movl $1,%eax
movl $0,%ebx
int $0*80
```

Fig 3 program to *A = (b*c) / d; by shift operation*

*In this algorithm as the question suggests, it uses all the inputs as general purpose registers. By using “mov” command the assigned values of b, c, are moved to the registers eax, ebx, ecx, respectively. As per the given expression A = (b*c) / d;, according to “Bodmas rule” we first perform sal(shift arithmetic left) operation on b and c which are in the registers eax abd ebx the sar(shift arithmetic right) operation from this result to d containing register .And then we move the answer to memory a.*

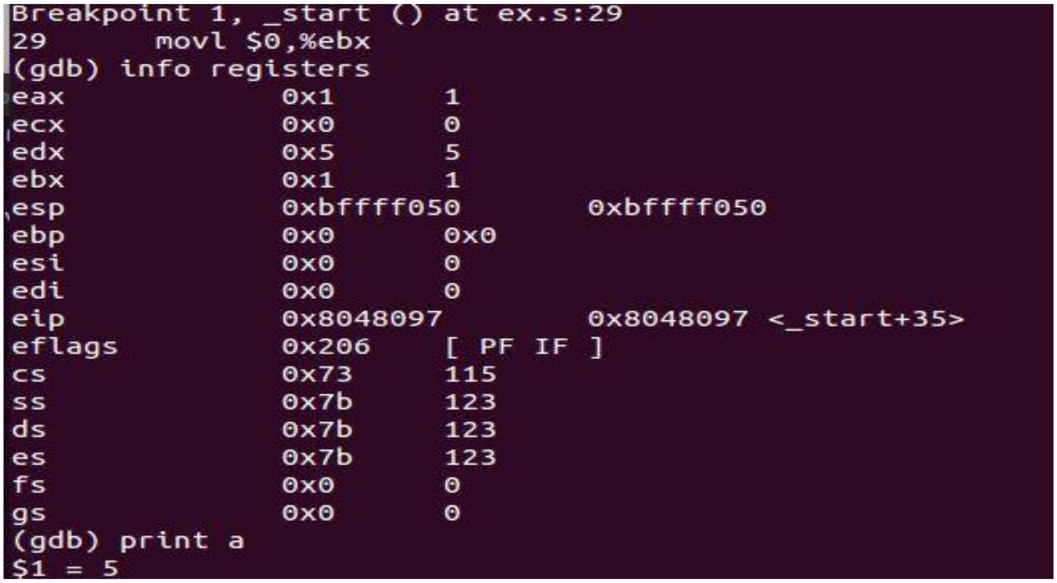
6. Presentation of Results



```
Starting program: /home/mplab/deepak/ex

Breakpoint 1, _start () at ex.s:29
29      movl $1,%eax
(gdb) info registers
eax            0x2      2
ecx            0x6      6
edx            0x0      0
ebx            0x2      2
esp            0xbffff050       0xbffff050
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) 
```

Fig 4 RESULT for *a= (b AND c) XOR d*

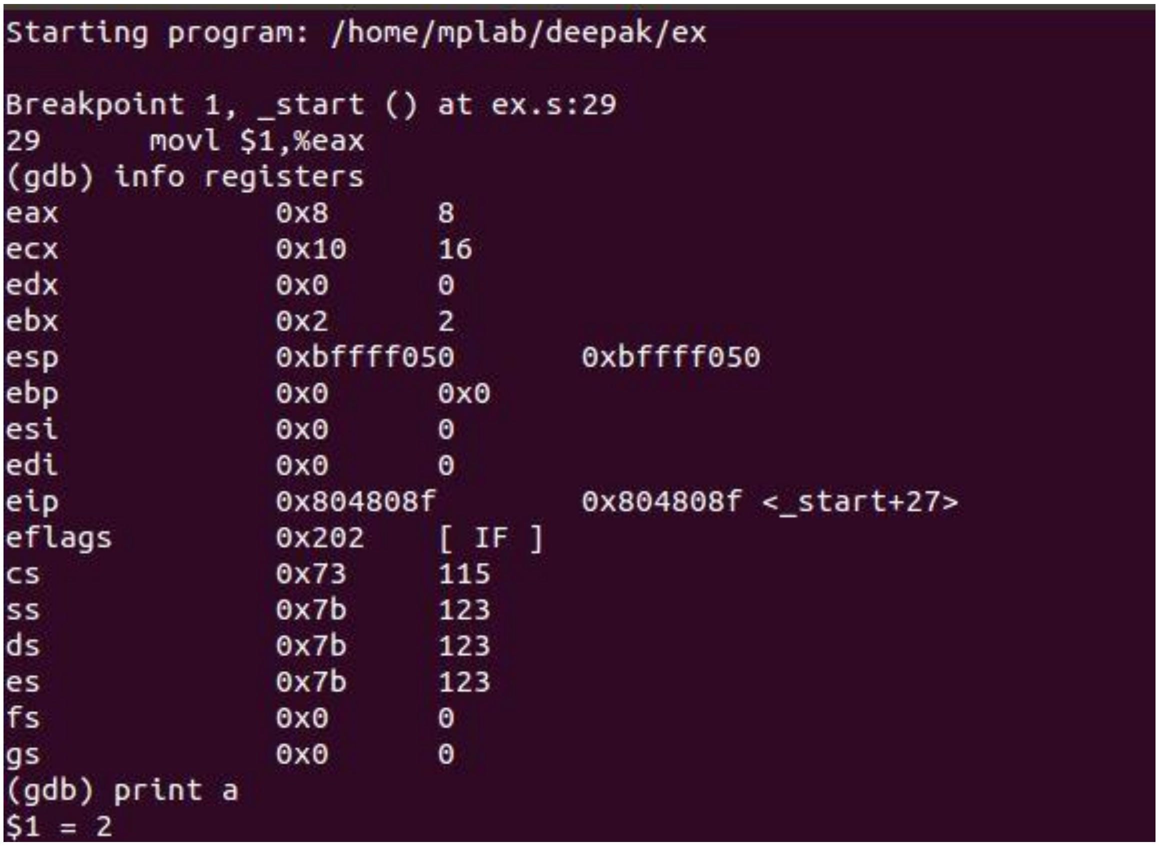
The answer to this question is 6 which is stored in memory region a , so as we print a we can see the answer 6



```
Breakpoint 1, _start () at ex.s:29
29      movl $0,%ebx
(gdb) info registers
eax            0x1      1
ecx            0x0      0
edx            0x5      5
ebx            0x1      1
esp            0xbffff050       0xbffff050
ebp            0x0      0x0
esi            0x0      0
edi            0x0      0
eip            0x8048097        0x8048097 <_start+35>
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 = 5
```

Fig 5 Result of *a=(b XOR c) OR d;*

The answer to this question is 5 which is stored in memory region a , so as we print a we can see the answer 5



```
Starting program: /home/mplab/deepak/ex

Breakpoint 1, _start () at ex.s:29
29      movl $1,%eax
(gdb) info registers
eax            0x8      8
ecx            0x10     16
edx            0x0      0
ebx            0x2      2
esp            0xbffff050       0xbffff050
ebp            0x0      0x0
esi            0x0      0
edi            0x0      0
eip            0x804808f        0x804808f <_start+27>
eflags         0x202    [ IF ]
cs             0x73     115
ss             0x7b     123
ds             0x7b     123
es             0x7b     123
fs             0x0      0
gs             0x0      0
(gdb) print a
$1 = 2
```

Fig 6 Result of *A = (b*c) / d; by shift operation*

The answer to this question is 2 which is stored in memory region a , so as we print a we can see the answer 2

7. Analysis and Discussions

   ➢ Learn to Identify the appropriate assembly language instruction for the given logical operation

   ➢ Used to Perform all logical operations using assembly language instruction

   s ➢ Understand different data types and memory used

   ➢ Get familiar with assembly language program by developing simple programs

8. Conclusions

   From the given two programs we can conclude that all logical operations can be used in assembly level programming for performing operations which helps to perform large program.

9. Comments

   1. Limitations of Experiments

      We don’t find any Limitations of Experiments

2. Limitations of Results

We don’t find any Limitations of Experiments

3. Learning happened

Learned to use assembly language instruction with all logical operation of a assembly level programming, and also learn the data types used in the programming and how the data can be stored in register etc..

4. Recommendations

We are supposed to carefully do operations by taking consideration of whether particular bit of answer can be stored in which general purpose register by giving thought to this we can overcome the limitations.

Signature and date

Marks