

## Laboratory 4

Title of the Laboratory Exercise: Controlling execution flow using conditional instructions

### 1. Introduction and Purpose of Experiment

Students will be able to perform control flow operations using conditional instructions

### 2. Aim and Objectives

**Aim** To develop assembly language program to perform control flow operations using conditional instructions.

#### Objectives

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

- Identify the appropriate assembly language instruction for the given conditional operations
- Perform all conditional 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

Develop an assembly language program to perform the following

1. Print all even numbers in 'n' natural numbers
2. Print all odd numbers in 'n' natural numbers
3. Compute GCD for the given two natural numbers
4. Compute LCM for the given two natural numbers
5. Develop an assembly language program to generate the first n numbers in Fibonacci series.

### 5. Calculations/Computations/Algorithms



```

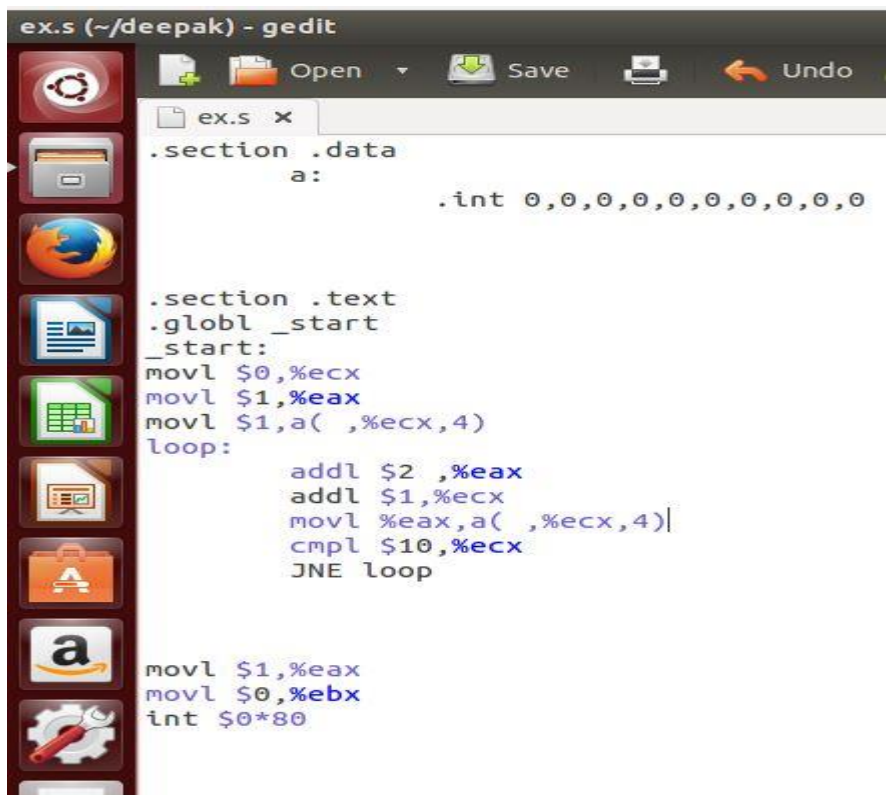
ex.s (~/deepak) - gedit
ex.s x
.section .data
a:
.int 0,0,0,0,0,0,0,0,0,0

.section .text
.globl _start
_start:
movl $0,%ecx
movl $0,%eax
movl $0,a(,%ecx,4)
loop:
    addl $2,%eax
    addl $1,%ecx
    movl %eax,a(,%ecx,4)
    cmpl $10,%ecx
    JNE loop

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

```

Fig 1 Program to Print all even numbers in 'n' natural numbers



```

ex.s (~/deepak) - gedit
ex.s x
.section .data
a:
.int 0,0,0,0,0,0,0,0,0,0

.section .text
.globl _start
_start:
movl $0,%ecx
movl $1,%eax
movl $1,a(,%ecx,4)
loop:
    addl $2,%eax
    addl $1,%ecx
    movl %eax,a(,%ecx,4)
    cmpl $10,%ecx
    JNE loop

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

```

Fig 2 Program to Print all odd numbers in 'n' natural numbers

```

lab4.s x
1
2
3      .int 64
4      b :
5      .int 36
6      gcd :
7      .int 0
8      lcm :
9      .int 0
10 .section .text
11 .globl _start
12 _start :
13
14 movl $64, %eax
15 movl $36, %ebx
16 loop:
17     divl %ebx
18     movl %ebx, %eax
19     movl %edx, %ebx
20     movl $0, %edx
21     cmpl $0, %ebx
22     JNE loop
23     movl %eax, gcd
24
25 movl a, %eax
26 mull b
27 divl gcd
28 movl %eax, lcm
29
30 movl $1, %eax
31 movl $0, %ebx
32 int $0x80

```

Fig 3 Program to Print Lcm and Gcd of any two numbers

```

ex.s (~/.deepak) - gedit
1
2
3      .section .data
4      a:
5      .int 0,0,0,0,0,0,0,0,0,0
6
7      .section .text
8      .globl _start
9      _start:
10     movl $2,%edx
11     movl $1,%ecx
12     movl $1,a(,%ecx,4)
13     addl $1,%ecx
14     movl $1,a(,%ecx,4)
15     loop:
16         subl $1,%ecx
17         movl a(,%ecx,4),%eax
18         addl $1,%ecx
19         addl a(,%ecx,4),%eax
20         addl $1,%ecx
21         movl %eax,a(,%ecx,4)
22         addl $1,%edx
23         cmpl $10,%edx
24         JNE loop
25
26     movl $1,%eax
27     movl $0,%ebx
28     int $0x80

```

Fig 4 program to generate the first n numbers in Fibonacci series.

## 6. Presentation of Results

```
Breakpoint 1 at 0x804809b: file ex.s, line 22.  
(gdb) run  
Starting program: /home/mplab/deepak/ex  
  
Breakpoint 1, loop () at ex.s:22  
22      movl $1,%eax  
(gdb) print a@10  
$1 = {0, 2, 4, 6, 8, 10, 12, 14, 16, 18}  
(gdb)
```

Fig 5 Result of Program to Print all even numbers in 'n' natural numbers

```
Breakpoint 1 at 0x804809b: file ex.s, line 22.  
(gdb) run  
Starting program: /home/mplab/deepak/ex  
  
Breakpoint 1, loop () at ex.s:22  
22      movl $1,%eax  
(gdb) print a@10  
$1 = {1, 3, 5, 7, 9, 11, 13, 15, 17, 19}  
(gdb)
```

Fig 6 Result of Program to Print all odd numbers in 'n' natural numbers

```
Starting program: /home/mplab/deepak/ex  
  
Breakpoint 1, loop () at ex.s:24  
24      movl $1,%eax  
(gdb) print a@10  
$1 = {0, 1, 1, 2, 3, 5, 8, 13, 21, 34}  
(gdb)
```

Fig 7 Result of program to generate the first n numbers in Fibonacci series.

```
(gdb) print lcm  
$2 = 576  
(gdb) print gcd  
$3 = 4  
(gdb)
```

Fig 8 Result of Program to Print Lcm and Gcd of any two numbers

## 1. Conclusions

Execution Flow can be controlled by using conditional instructions, which includes a `cmp` instruction followed by a jump instruction, a `cmp` instruction compares the two operands and updates the flag register, this is then used with jump instruction to go to some other part of the program, using this we can form looping structures to do stuff like print  $n$  natural numbers, sum of them and some basic programs like LCM and GCD of two numbers, even functions can be emulated in assembly by using such structures.

## 2. Comments

### 1. Limitations of Experiments

Although looping structures can be formed using the `cmp`, `jcc` instructions but recursive structures are complex to form using just these instructions.

### 2. Limitations of Results

None

### 3. Learning happened

We learnt the use of compare, unconditional jump and conditional jump instructions to form looping structures and conditional statements.

### 4. Recommendations

Since a program can contain numerous loop labels, each label should be carefully names, and the programmer must keep track of which parts of the program jump to where, else there might be chances of forming infinite loops.

Signature and date

Marks

