

# COAL LAB MANUAL

**Submitted to:**

Dr. Tauqir

Teacher assistant : Sir Shahzad

**Submitted By:**

Komal Shehzadi

Registration no : 2016-CS-178

**Section:**

Section B

**Department:**

Department of Computer Science

**Institute:**

University of Engineering and Technology Lahore (Main Campus)



**Date of Submission:**

March 29, 2018

## LAB 4

### Task 1:

**Take a digit input from user and check whether it is even or odd.**

The description inside the “// ‘ ‘ // “ are the comments to make the code more clear. These are not the part of program.

**// PROGRAM //**

.model

**// Data Segment//**

**// All variables defined here //**

.Data

msg DB 10, 13, 'Enter A NUMBER \$'

msg1 DB 10, 13, 'number is even!\$'

msg2 DB 10, 13, 'number is odd! \$'

msg3 DB 10, 13, 'do you want to continue(y/n) for yes or no?', 10, 13, '\$'

**// Code Segment //**

**// all the functions to be performed written here //**

.Code

**// Basic commands to use the variables in data segment //**

**// Compulsory!! //**

mov AX, @data

mov DS, AX

**// Jump on label p1 //**

JMP p1

**// Label p1 //**

**// in label p1 we are taking single digit input from keyboard**

**And checking its remainder from 2 to check the even or odd digit if remainder is 0, it is even digit, control jumps on label p2 otherwise, it is odd digit, it will jump on p3 //**

**p1:**

**LEA DX, msg**

**mov ah, 9**

**int 21h**

**mov ah, 1**

**int 21h**

**mov bl, 2**

**div bl**

**CMP ah, 0**

**JE p2**

**JNE p3**

**// Label p2 //**

**// it will display that the number is even and then the control jumps on label p4 //**

**p2:**

**LEA DX, msg1**

**mov ah, 9**

**int 21h**

**jmp p4**

**// Label p3 //**

**// it will display that the number is odd and then control jumps on label p4 //**

p3:

```
LEA DX, msg2
mov ah, 9
int 21h
jmp p4
```

**// Label p4 //**

**// this will prompt the user for another input or just exit the program  
by entering, 'y' to continue for another input at this control will again  
jump on p1 label and if user enters 'n' control will jump on exit label  
//**

p4:

```
LEA DX, msg3
mov Ah, 9
int 21h
mov ah, 1
int 21h
CMP al, 'y'
JE p1
JNE exit
```

**// Label exit //**

**// Exits the program //**

exit:

```
.exit
```

## Commands:

**.model:** it is used to tell the assembler that program is in protected mode.

**.code:** it contains all the codes and commands.

**.data:** it contains all the definitions and declarations of variables.

**Labels:** these are the customized names of specific code segment to call that code segment anywhere in program by its name

LABEL\_NAME:

---

Write the code here.

---

**MOV:** Moves the contents of source instruction in destination and the syntax is as follows:

Mov Destination, Source

**INT 21H:** Call to DOS interrupt Handler.

INT 21H

**LEA:** Load effective address of a memory block in a register it moves the starting address of a memory block in a register.

LEA Source, Destination

**JMP:** JMP command is used for unconditional jump on a label.

JMP LABEL\_NAME

**CMP:** CMP is used to compare the contents of two register or variables and give a value in result if source is less than , greater than or equal to destination then -1, 1 or 0 is returned by CMP respectively.

CMP Source, Destination

**JNE:** in the result of CMP command this label will be executed if the contents of source and destination are not equal and by this we can jump on a label

JNE LABEL\_NAME

**JLE:** In the result of CMP command this label will be executed if the contents of source is less or equal to destination by this we can jump on a label

JLE LABEL\_NAME

**JE:** In the result of CMP command this label will be executed if the contents of source and destination are equal and by this we can jump on a label

JE LABEL\_NAME

**JL:** In the result of CMP command this label will be executed if the contents of source is less than destination by this we can jump on a label

JL LABEL\_NAME

**JG: :** In the result of CMP command this label will be executed if the contents of source is greater than destination by this we can jump on a label

JG LABEL\_NAME

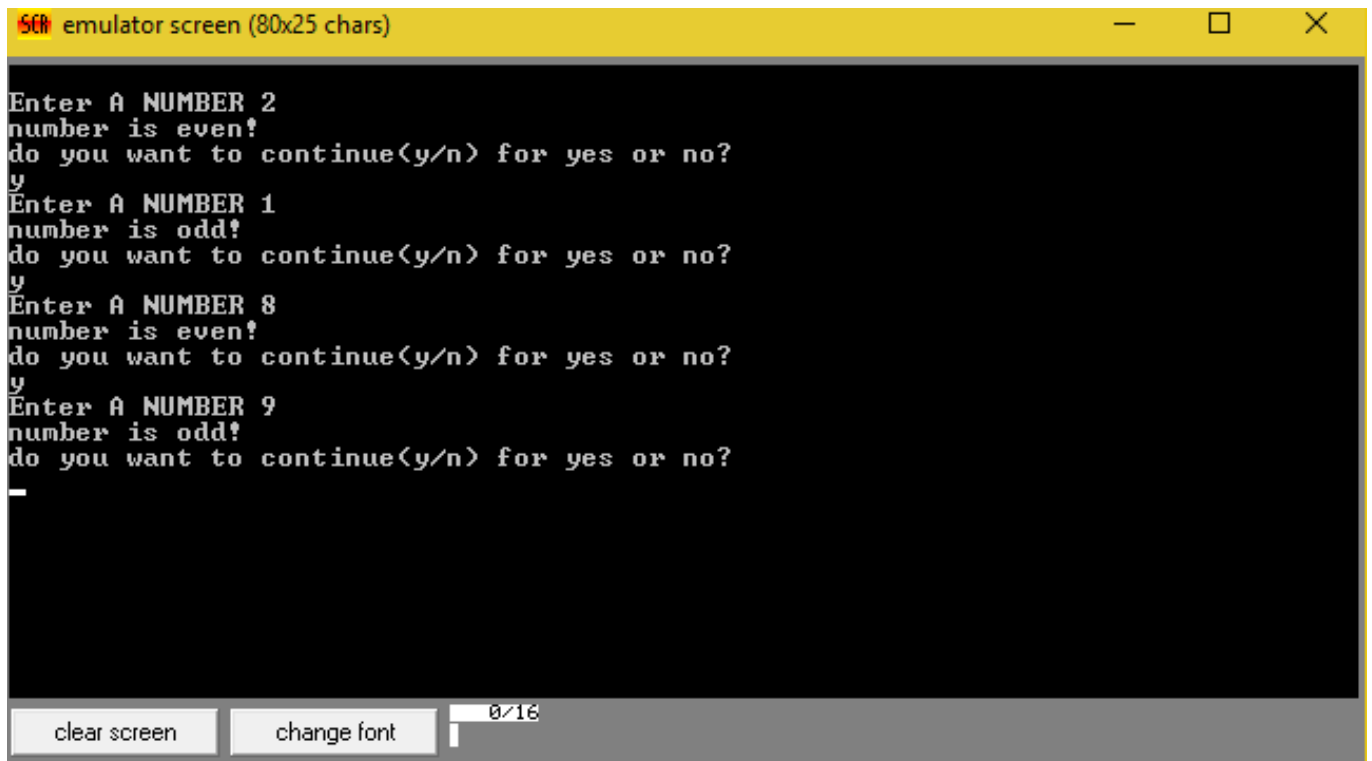
**JGE:** In the result of CMP command this label will be executed if the contents of source is greater or equal to destination by this we can jump on a label

JGE LABEL\_NAME

**.exit:** Command to exit the program.

## OUTPUT:

The following output screen will appear when the above code is run enter the number and it will tell either the number is even or odd and then prompt for continue or exit.



```
emulator screen (80x25 chars)
Enter A NUMBER 2
number is even!
do you want to continue(y/n) for yes or no?
y
Enter A NUMBER 1
number is odd!
do you want to continue(y/n) for yes or no?
y
Enter A NUMBER 8
number is even!
do you want to continue(y/n) for yes or no?
y
Enter A NUMBER 9
number is odd!
do you want to continue(y/n) for yes or no?
-
```

## Description:

In this lab we take a digit input from user and check whether it is even or odd and display the result on screen. For this purpose we have used number of labels to take input and display the desired even or odd output and then again prompt for next input or exit the program. In data segment the string variables are stored. Where we have to display the string variable move 9 in AH register. And load the string variable to be displayed in DX register by LEA command. And all the other tasks are done as explained in code.



## TASK 2 : Home Task

**Take a character input from user and convert it to its opposite case means lower to upper and upper to lower case.**

The description inside the “// ‘ ‘ // “ are the comments to make the code more clear. These are not the part of program.

**// PROGRAM //**

.model

**// DATA SEGMENT //**

**// All the required variables and string messages to be displayed on screen are declared here //**

.data

msg DB 10, 13, 'enter a alphabet character from keyboard \$'

msg1 DB 0ah, 0dh, 'the upper case of letter is \$'

msg2 DB 0ah, 0dh, 'the lower case of letter is \$'

msg3 DB 0ah, 0dh, 'enter 1 to continue and 0 to exit: \$'

**// CODE SEGMENT //**

**// All the codes are written here to perform the action //**

.code

```
mov AX, @data
```

```
mov DS, AX
```

```
jmp p0
```

```
// LABEL p0 //
```

```
// Takes input from user //
```

```
p0:
```

```
LEA DX, msg
```

```
mov ah, 9
```

```
int 21h
```

```
jmp p1
```

```
// LABEL p1 //
```

```
// Compares the input character with suitable hexadecimal ascii  
values to check whether it is upper case or lower case then jump on  
suitable label to convert it in its opposite case //
```

```
p1:
```

```
mov ah, 1
```

```
int 21h
```

mov bl, al

Cmp bl, 91

JL p3

CMP bl, 96

JG p2

**// LABEL p3 //**

**// it converts lower case letter in upper case and shows the screen message "the upper case of letter is " and then display the upper case letter. And then jumps on label p4 //**

p2:

LEA DX, msg1

mov ah, 9

int 21h

sub bl, 32

mov ah, 2

mov dl, 0ah

int 21h

mov dl, 0dh

int 21h

```
mov dl, bl
```

```
int 21h
```

```
jmp p4
```

```
// LABEL p3 //
```

```
// it converts upper case letter in lower case and shows the screen  
message "the lower case of letter is " and then display the lower case  
letter. And then jumps on label p4 //
```

```
p3:
```

```
LEA DX, msg2
```

```
mov ah, 9
```

```
int 21h
```

```
add bl, 32
```

```
mov ah, 2
```

```
mov dl, 0ah
```

```
int 21h
```

```
mov dl, 0dh
```

```
int 21h
```

```
mov dl, bl
```

```
int 21h
```

jmp p4

**// LABEL p4 //**

**// It prompts screen for user to either continue for next character or  
exit the program //**

p4:

Lea dx, msg3

mov ah, 9

int 21h

mov ah, 1

int 21h

cmp al, 49

JE p0

JNE exit

**// exit Label //**

**// To exit the program //**

exit:

.exit

## Commands:

**.model:** it is used to tell the assembler that program is in protected mode.

**.code:** it contains all the codes and commands.

**.data:** it contains all the definitions and declarations of variables.

**Labels:** these are the customized names of specific code segment to call that code segment anywhere in program by its name

LABEL\_NAME:

---

Write the code here.

---

**MOV:** Moves the contents of source instruction in destination and the syntax is as follows:

Mov Destination, Source

**INT 21H:** Call to DOS interrupt Handler.

INT 21H

**LEA:** Load effective address of a memory block in a register it moves the starting address of a memory block in a register.

LEA Source, Destination

**JMP:** JMP command is used for unconditional jump on a label.

JMP LABEL\_NAME

**CMP:** CMP is used to compare the contents of two register or variables and give a value in result if source is less than , greater than or equal to destination then -1, 1 or 0 is returned by CMP respectively.

CMP Source, Destination

**JNE:** in the result of CMP command this label will be executed if the contents of source and destination are not equal and by this we can jump on a label

JNE LABEL\_NAME

**JLE:** In the result of CMP command this label will be executed if the contents of source is less or equal to destination by this we can jump on a label

JLE LABEL\_NAME

**JE:** In the result of CMP command this label will be executed if the contents of source and destination are equal and by this we can jump on a label

JE LABEL\_NAME

**JL:** In the result of CMP command this label will be executed if the contents of source is less than destination by this we can jump on a label

JL LABEL\_NAME

**JG: :** In the result of CMP command this label will be executed if the contents of source is greater than destination by this we can jump on a label

JG LABEL\_NAME

**JGE:** In the result of CMP command this label will be executed if the contents of source is greater or equal to destination by this we can jump on a label

JGE LABEL\_NAME

**Sub:** It is used to subtract destination from source and store the result of computation in destination.

Sub Destination, Source

**Add:** It is used to add source in destination and store the result of computation in destination.

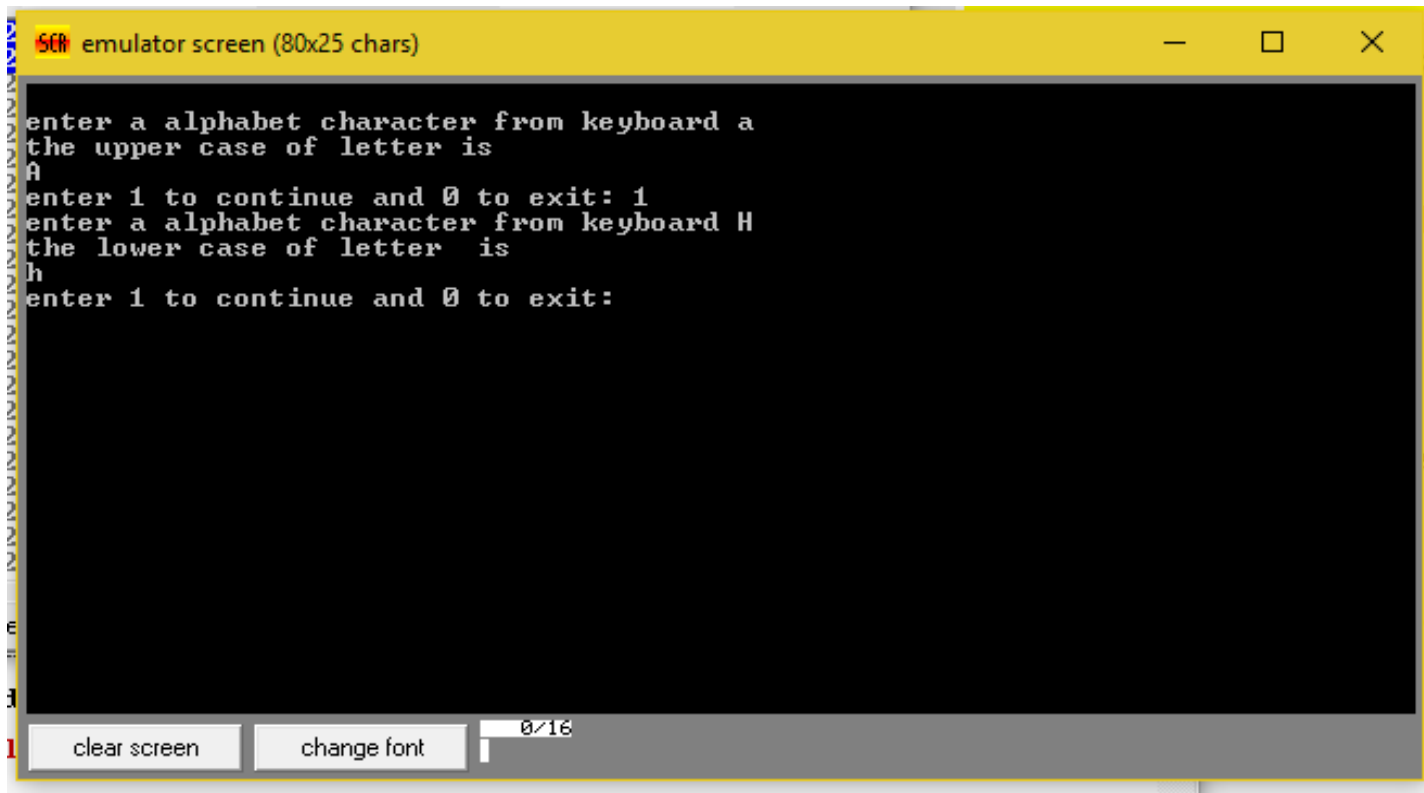
Add Destination, Source

**.exit:** command to exit the program.



## OUTPUT:

Output screen looks like this when input is taken and required output is displayed so the above code works like this.



The screenshot shows a window titled "emulator screen (80x25 chars)". The screen displays the following text:

```
enter a alphabet character from keyboard a
the upper case of letter is
A
enter 1 to continue and 0 to exit: 1
enter a alphabet character from keyboard H
the lower case of letter is
h
enter 1 to continue and 0 to exit:
```

At the bottom of the window, there are two buttons: "clear screen" and "change font". To the right of these buttons is a small input field containing "0/16".

## Description:

In this home task we have to take input character from user and convert it in its opposite case .Upper to lower and lower to upper case. For this purpose I have used multiple labels and compared the hexadecimal ascii values of upper case and lower case with the input character and then generated the output character upper and lower case for lower and upper case

respectively and then again prompt the screen to continue or exit as shown in output screen above.