Course Title: Microprocessors and Assembly Language Lab (CSE-4504)

Department of Computer Science and Engineering (CSE)
Islamic University of Technology (IUT), Gazipur

Lab # 02

Understanding 8086 I/O Instructions and Conditional JUMP Instructions using Assembly Language in EMU8086.

Objective:

To understand the basic 8086 I/O instructions using Assembly Language Program in EMU8086.

Theory:

- **8086 Input / Output Instructions**

CPU communicates with the peripherals through I/O registers called I/O ports. There are two instructions, IN and OUT, that access the port directly. These instructions are used when very fast I/O is essential; for example, in a game program. However, most applications programs do not use IN and OUT because

i. Port addresses vary among computer models and
ii. It's much easier to program I/O with the **service routines** provided by the manufacturer.

There are two categories of I/O service routines:

i. The BIOS (Basic Input/Output System) routines.
ii. The DOS (Disk Operating System) routines.

The BIOS routines are stored in ROM and interact directly with the I/O ports. The DOS routines carry out more complex tasks: for example, printing a character string; actually they use the BIOS routines to perform direct I/O operations.

The INT Instruction

To invoke a DOS or BIOS routine, the **INT** (interrupt) instruction is used. It has the format

INT interrupt_number

Where **interrupt_number** is a number that specifies a routine. For example, INT 16h invokes a BIOS routine that performs keyboard input. In the following, we use a particular DOS routine, INT 21h.

INT 21h

INT 21h may be used to invoke a large number of DOS functions; a particular function is requested by placing a function number in the AH register and invoking INT 21h. Here we are interested in the following functions:

Function Number	Routine
1	single-key input
2	single-key output
9	character string output

INT 21h functions expect input values to be in certain registers and return output values in other registers. These are listed as we describe each function.

Function 1:
Single-key Input
Input: AH=1
Output: AL = ASCII code if character key is pressed
= 0 if non-character key is pressed.

To invoke the routine, execute these instructions:

```
MOV AH, 1   ; input key function
INT 21h     ; ASCII code in AL
```

The processor will wait for the user to hit a key if necessary. If a character key is pressed, AL gets its ASCII code; the character is also displayed on the screen. If any other key is pressed, such as an arrow key, F1-F10 and so on, AL will contain 0.

Because INT 21h, function 1 doesn't prompt the user for input, he or she might not know whether the computer is waiting for input or is occupied by some computation. The next function can be used to generate an output prompt.

Function 2:
Single-key Output
Input: AH=2
DL = ASCII code of the display character or control character
Output: AL = ASCII code of the display character or control character

To display a character with this function, we put its ASCII code in DL. For example, the following instructions cause a question mark to appear on the screen:

```
MOV AH, 2   ; display character function
MOV DL, '?' ; character is '?'
INT 21h     ; display character
```

After the character is displayed, the cursor advances to the next position on the line. Function 2 may also be used to perform control functions. If DL contains the ASCII codes of a control character, INT 21h causes the control function to be performed. The principle control characters are as follows:

ASCII code	Symbol	Fucntion
7	BEL	Beep
8	BS	backspace
9	HT	tab
A	LF	line feed (new line)
D	CR	carriage return (start of a line)

- **Conditional Control Transfer Instruction**

Conditional jumps transfer control to another address depending on the values of the flags in the flag register..

The jump condition often provided by the CMP instruction:

CMP *destination, source*

Condition	Instruction	Condition	Instruction
Jump if zero flag ZF=1	**JZ *zero***	Jump if zero flag ZF=0	**JNZ *notzero***
Jump if greater	**JG *greater***	Jump if greater than or equal	**JGE *notless***
Jump if less	**JL *less***	Jump if less than or equal	**JLE *notgreater***
Jump if Below	**JB *smaller***	Jump if carry flag CF=1	**JC *carry***

Assembly Language Program Example:

```
ORG 0100h

MAIN PROC

; display prompt
MOV AH, 2
MOV DL, '?'
INT 21h

; input a character
MOV AH, 1
INT 21h
MOV BL, AL

; go to a new line with carriage return
MOV AH, 2
MOV DL, 0DH
INT 21h
MOV DL, 0AH
INT 21h

; display character
MOV DL, BL
INT 21h

; return to DOS
MOV AH, 4CH
INT 21H

MAIN ENDP
END MAIN

RET
```

```
org 100h

START:      mov cl, 03h

LABEL_JNZ:  dec cl
            jnz LABEL_JNZ
            mov bl, 04h
            mov al, 04h

LABEL_JZ:   dec al
            dec bl
            xor bl, al
            jz LABEL_JZ
            mov bl, 02h
            mov al, 06h

LABEL_JG:       dec al
            cmp al, bl
            jg LABEL_JG
            mov bl, 06h
            mov al, 00h

LABEL_JL:   inc al
            cmp al, bl
            jl LABEL_JL

ret
```

Tasks to do:

1. Write an assembly language program that inputs a single letter and shows the same letter in it's opposite case in a new line. (Lower-case to Upper-case or vice-versa).

   Sample Input / Output:

Input:	a	Input:	Z
Output:	A	Output:	z

2. Write an assembly language program that inputs a single letter and shows the next 5 (five) letters in opposite case of input (Lower-case to Upper-case or vice-versa) in a row of a new line and also shows the previous 5 (five) letters in the next line in opposite case of input (Lower-case to Upper-case or vice-versa).

   Sample Input / Output:

Input:	a	Input:	Z
Output:	BCDEF	Output:	abcde
	ZYXWV		yxwvu