BRAC UNIVERSITY
Inspiring Excellence

Department of Computer Science and Engineering

Course Code: CSE341	Credits: 1.5
Course Name: Microprocessors	Semester: Fall'18

Lab 03

Basic I/O, Advanced arithmetic operations and Flags

I. Topic Overview:

Instructions used to communicate with peripherals are called interrupts. There are many interrupts each dealing with a particular task. Students will familiarize themselves with the basic input output mechanisms of the assembly language. They will be looking into one of the most common interrupts, INT 21H and inquire how this very interrupt can be used for single key input, single key output and multiple characters output. Additionally, they'll perform some arithmetic operations upon taking inputs from the user.

II. Lesson Fit:

There is prerequisite to this lab. Students must have a basic idea on the following concepts:

a. Registers
b. Some basic operations such as MOV, ADD, SUB, MUL and DIV
c. Character encoding using ASCII

III. Learning Outcome:

After this lecture, the students will be able to:

a. Perform I/O operations to make their assembly programs more interactive
b. Write more dynamic assembly programs
c. Familiarize themselves with how flag register works

IV. Anticipated Challenges and Possible Solutions

a. Students may find it difficult to map ASCII values to decimal/hex

Solutions:

i. Looking at the ASCII table may help

V. Acceptance and Evaluation

Students will show their progress as they complete each problem. They will be marked according to their class performance. There may be students who might not be able to finish all the tasks, they will submit them later and give a viva to get their performance mark. A deduction of 30% marks is applicable for late submission. The marks distribution is as follows:

Code: 50%

Viva: 50%

VI. Activity Detail

a. **Hour: 1**

Discussion: Basic input and output

In order to carry out basic I/O operation, function specific values need to be kept in AH register before calling INT 21h which is used for a system call. The function dispatcher maps the value to carry out specific function.

AH = 1; single character input

AH = 2; single character output

AH = 9; string output

i. Single Key Input

mov ah,1

int 21h

The program will wait for the user to give an input. It will be saved in AL.

ii. Single Character Output

The item to be printed must be in DL

mov dl,5

mov ah,2

int 21h

The hexadecimal values of new line is 0Ah and line feed is 0Dh

iii. Displaying a String

A string is an array of characters. Previously we have defined variables and arrays.

m db "a$"

m1 db "hello$"

The values of variables and arrays are stored in the data segment of the memory. In order to fetch them we need the offset address (offset + segment). To print a string we need to store 9 in AH. While 9 is in AH, int 21h expects DX to hold the offset address of the array/variable.

LEA is the name of the instruction that provides the offset address. LEA stands for Load Effective Address. The syntax for using LEA is: LEA destination, source. Source is the memory address (the array or the variable) whose offset address will be saved in the destination. The destination is a general register. e.g. for printing the string "hello":

m1 db "hello$"

LEA DX,m1

mov AH,9

int 21h

Our first program will read a character from the keyboard and display it at the beginning of the next line.

We start by displaying a question mark:

MOV AH, 2 ;display character function

MOV DL, '?' ;character is '?'

INT 2lh ;display character

The second instructlon moves 3Fh, the ASCII code for "?", into DL.
Next we read a character:

MOV AH, l ;read character function

INT 2lh ;character in AL

Now we would like to display the character on the next line. Before doing so, the character must be saved in another register. (We'll see why in a moment.)

MOV BL,AL ;save it in BL

To move the cursor to the beginning of the next line, we must execute a carriage return and line feed. We can perform these functions by putting the ASCII codes for them in DL and executing INT 21h.

MOV AH,2 ;display character function

MOV DL,0DH ;carriage return

INT 2lh ;execute carriage return

MOV DL,0AH ;line feed

INT 2lh ;execute line feed

The reason why we had to move the input character from AL to BL is that the INT 21h function, changes AL.

Finally we are ready to display the character:

MOV DL,BL ;get character

INT 2lh ;and display it

Here is the complete program:

```
code segment
start:
; set segment registers:
    mov ax, data
    mov ds, ax
    mov es, ax

    ; add your code here

    MOV AH,2
    MOV DL,'?'
    INT 21H
    ;input a character
    MOV AH,1
    INT 21H
    MOV BL,AL
    ;go to a new line
    MOV AH, 2
    MOV DL,0DH
    INT 21H
    MOV DL,0AH
    INT 21H
    ;display character
    MOV DL,BL
    INT 21H


    mov ax, 4c00h ; exit to operating system.
    int 21h
```

Problem Task: Task 01-Task04 (Page 8)

b. **Hour: 2**

Discussion:

Check the problem tasks while the students continue with the rest.

Problem Task: Task 05 – 08 (Page 8-9)

c. **Hour: 3**

Discussion: Flag register

There is one special type of register called the flag register. It contains flags. Flags are bits of information which indicate the state of the processor. Decisions are made by the processor based on the flags. Flags are of two types:

1. **Status Flags:** The processor uses these flags to reflect the result of an operation.
2. **Control Flags:** These are used to enable/disable certain operations of the CPU.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				OF	DF	IF	TF	SF	ZF		AF		PF		CF

Bit number 10, 9 and 8 are control flags and the rest non-empty bits are status flags. The empty bits have no significance.

Status Flags

- **Carry Flag (CF):** CF = 1 when there is a carry out from the MSB during addition or a borrow into the MSB during subtraction.
- **Parity Flag (PF):** PF = 1 when the low byte of a result of an operation contains even number of ones.
- **Auxiliary Flag (AX):** AX = 1 when there is a carry out from bit number 3 to bit number 4 on addition or borrow into bit number 3 during subtraction. (Note that the bit number starts from 0)
- **Zero Flag (ZF):** ZF = 1, if the result of an operation in 0.
- **Sign Flag (SF):** SF = 1 when the MSB of a signed number is 1.
- **Overflow Flag (OF):** OF = 1 when there is an overflow. There are 3 types of overflows: - Signed Only, Unsigned Only and Both. To understand overflow, you must know by now that 1111b have 2 different meanings in the context of unsigned and signed numbers. If you are not clear about this please read the DLD book.

  An example of unsigned overflow where the maximum number of bits is 4, 1111 + 1000 = 10111

  Here the MSB gets discarded and thus data loss. As example of signed overflow where the maximum range is 8 bits. For signed numbers the MSB represents the sign. 1 for negative and 0 for positive. 0111 1111 + 0111 1111 = 1111 1110

  Both the operands start with 0 therefore both are positive numbers. Addition of 2 positive numbers has to positive. But if we look the answer, it starts with 1 therefore the number is a negative number.

  How instructions affect flags:

Instruction	Affected Flags
MOV	NONE
ADD/SUB	ALL
INC/DEC	ALL EXCEPT CF
NEG	ALL (CF =1 UNLESS RESULT IS 0 OF =1 IF THE OPERAND IS 8000H AND 80H)

It has been previously mentioned that the processor takes decisions based on the flags. In this section we will see how these decisions are made. In JAVA, decisions are made using "if" conditions. If the condition is satisfied then a portion of code runs and if not, the program jumps to another section of code. In assembly this whole system is done by two operations -- compare and jump. The jump can be a condition dependent and also independent.

Problem Task: Task 09-11 (Page 9-10)

VII. Home Tasks: All the unfinished lab tasks.

Lab 3 Activity List

Task 01

Take a character input and display it. Display the message "Please insert a character: " when taking an input.

Task 02

Perform addition/subtraction/division/multiplication by taking inputs from the user.

Note: Display appropriate messages when taking input and showing the output..

Task 03

Write instructions to do the following.

a. Read a character, and display it at the next position on the same line.

b. Read an uppercase letter (omit error checking), and display it at the next position on the same line in lower case.

Task 04

Read an uppercase letter (omit error checking), and display it at the next position on the next line in lower case.

Task 05

Write a program to:

(a) display a "?"

(b) read two decimal digits whose sum is less than 10,

(c) display them and their sum on the next line, with an appropriate message.

Sample execution:

?27

HE SUM OF 2 AND 7 IS 9

Task 06

Write a program to:

(a) prompt the user, (b) read first, middle, and last initials of a person's name, and then (c) display them down the left margin.

Sample execution:

ENTER THREE INITIALS: JFK

J

F

K

Task 07

Write a program to read one of the hex digits A-F, and display it on the next line in decimal. Sample execution:

ENTER A HEX DIGIT: C

IN DECIMAL IT IS 12

Task 08

Write a program to display a 10 x 10 solid box of asterisks.

HINT: Declare a string in the data segment that specifies the box, and display it with INT 2lh, function 9h

Task 09

ADD AL, BL. Where AL contains 80h, BL contains 80h. Identify the status of different status flags.

Task 10

Suppose that AX and BX both contain positive numbers and ADD AX, BX is executed. Show that there is a carry into the msb but no carry out of the msb if, and only if, signed over flow occurs.

Task 11

Suppose AX and BX both contain negative numbers, and ADD AX,BX is executed. Show that there is a carry out of the msb but no carry into the msb if, and only if, signed overflow occurs.