**              DEPARTMENT OF COMPUTER A yellow and black logo

Description automatically generated&**

**SOFTWARE    ENGINEERING**

**COLLEGE OF E&ME, NUST, RAWALPINDI**

**Microprocessor and Microcontroller Based Design B**

**DE- 44 Synd-B**

**SUBMITTED TO:**

**Dr Saghir Khan**

**SUBMITTED BY:**

**Waseem Ghulam (409600)**

**Ameer Hamza      (409662)**

**Muskan Fatima    (414875)**

**Waqar Ahmed      (398417)**

**Lab 01: Introduction to Assembly Language & Basic IO Operations**

**Objectives:**

The objectives of this lab session is to get familiar with 80x86 based microprocessors architecture, few assembly language instructions and some IO operations using INT 21H.

**Task:**

1. **Observe and write down the contents of registers AX, BX, CX and DX after the complete code is run?**

AX: H = 4C, L = 00

BX: H = 00, L = 00

CX: H = 00, L = 08

DX: H = 00, L = 00

**A screenshot of a computer

Description automatically generated**

1. **Do the contents of any register change as the code is run step by step? If yes, what change is observed and in which registers?**

At start content of AX is 00 for both H and L register but as the code progresses H becomes 4C and L becomes 00. Other than that IP and CS also changes continuously on each step. And at the end SP also changes.

**Input and Output operations of 8086/8088 Assembly Language**

Now you will be introduced with how to perform input and output operations using DOS **INT 21H** function calls and get another step further in learning the structure of assembly language by using:

1. Variable declaration using: **DB**
2. Offset operator using **OFFSET**
3. Familiarization with friendly Emulator, **Emu8086**.

The instruction INT 21h stands for “Call Interrupt no. 21”. For the time being let’s take interrupts like a function or a subroutine placed in memory and being called from another program. The parameters to that function or subroutine are passed by using different CPU registers mostly AH, AL or DX. Functionality of that subroutine depends upon those parameters; Typical examples include taking input, showing output, mouse control ,VGA memory control etc. some of them are tabulated as under.

|  |  |  |  |
| --- | --- | --- | --- |
| **Function** | **Value of** | **Output in** | **Functionality** |
| **call No.** | **AH** |  |  |
|  |  |  |  |
| 01h | 01h | AL | Reads a character from keyboard, stores it in AL and |
|  |  |  | display it (echo it ) on screen. |
|  |  |  |  |
| 02h | 02h | Screen | Display the content of register DL on screen in ASCII |
|  |  |  | form. |
|  |  |  |  |
| 09h | 09h | Screen | Display the string characters addressed by DX to the |
|  |  |  | screen |
|  |  |  |  |
| 0Ah | 0Ah | Offset in | Read a string of characters from keyboard. |
|  |  | DX |  |
|  |  |  |  |

**DOS Function 02h:**

To display a single ASCII character at the current cursor position, use the following sequence of instructions.

**MOV** AH, 02H

**MOV** DL, Character Code

**INT** 21H

The Character Code may be the ASCII code of the character taken from the ASCII table (provided along with) or the character itself written between quotes.

The following code displays number 5 using its ASCII code:

**MOV AH, 02H**

**MOV DL, 35H**

**INT 21H**

**Question:**

**What happens if we replace 35H with just 35?**

It will display ‘#’ because Ascii value of 35 decimal is ‘#’

**Question:**

**What is the result of the code below?**

**MOV AH, 02H**

**MOV DL, ‘5’**

**INT 21H**

It will display ‘5’ as 5 is written in single quotes and it will simply interpret 5 as ascii.

**Function 01H**:

To read single character and have it echoed (displayed) on the screen, use the following

code:

**MOV AH, 01H**

**INT 21**

**Question:**

**What register contains the ASCII code of the character read from the keyboard?**

The 01H takes input from keyboard stores it in AL register.

**DOS Functions 09**:

This function is used to display a string of characters ending with a ‘$’ sign. The following code displays the string MESSAGE defined as:

**.DATA**

MESSAGE **DB** ‘This is the Message to be displayed’,

**.CODE**

**MOV** DX, OFFSET MESSAGE

**MOV** AH, 09H

INT 21H

Or

**.DATA**

MESSAGE **DB** ‘This is the Message to be displayed’, ‘$’

**.CODE**

**LEA** DX, MESSAGE

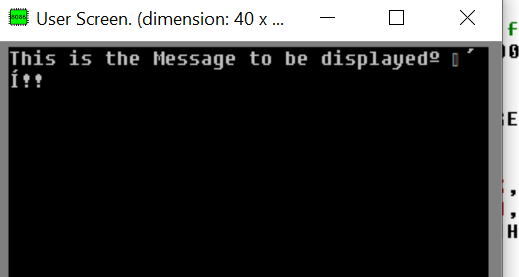
**MOV** AH, 09H

**INT** 21H

**Question:**

**what is the importance of “$”.**

The program starts reading from start of string and goes until it encounters ‘$’. If there is no ‘$’ at the end the program will go beyond the end of string and will print garbage value beyond the string.



**Does using any other register in place of DX (in function 09h) give the same result? If not, what can be the reason? Additionally, what does the LEA**

**DX** is required because **INT 21H, function 09H** expects the string address to be in **DX**. Using any other register will result in incorrect behavior or failure to display the string. As the programming is expecting address of string in the DX register.

**LEA** instruction is used to load address of memory location into a register rather than actual value stored at that address. In our case the LEA stores the address of message to DX register.

**TASK :**

**Define three strings containing your name, degree and department and display them on screen. The strings should be displayed on three different lines. Paste your code and the screenshot of output.**

; COM file is loaded at CS:0100h

ORG 100h

.DATA

MESSAGE DB 'Waseem Ghulam', 0Dh, 0Ah, 'DE-44', 0Dh, 0Ah, 'Computer Engineering', 0Dh, 0Ah, '$'

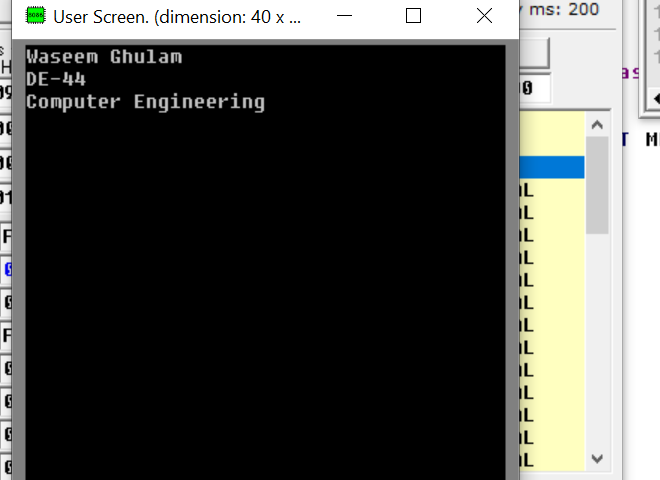
.CODE

MOV DX, OFFSET MESSAGE

MOV AH, 09H

INT 21H

RET



––