

## Lab #03

# Input and Output Instruction

### Objective

- Understand the working of INT 21H instruction
- Understand program having basic Input and Output Instruction

### Theory

The processor cannot access the peripheral devices (like keyboard or monitor) directly. Microprocessor use BIOS routines or DOS routines to access peripherals. The BIOS (Basic Input/Output System) routines are store in ROM and directly run the I/O ports where DOS routine use BIOS routine to perform operation that's why has less complex than BIOS routine.

The BIOS and DOS both use INT (Interrupt) instruction, the interrupt is actually done by number which determine nature of interrupt. The format of interrupt instruction is:

**INT    interrupt number**

e.g.        INT    21H        ; Invoke DOS routine  
               INT    16H        ; Invoke BIOS routine

Some INT 21H Function

Function Number	Routine
01H	Take input with echo
02H	Display character
07H	Take input without echo
09H	Display string
4CH	Control back to DOS
0BH	Check standard input status
2AH	Get system Date
2CH	Get system Time

When INT 21H execute the DOS first see the function number in AH register, so all the function number must be placed in AH register to execute.

### **Function 01H (Single character input with echo)**

After instruction executes, the AL register get the ASCII Code of the character and if non-character key (control key) is press then than AL register get 00H. The 01H function also displays the character on screen.

```
MOV    AH, 01H        ; Move input key function to AH register
INT     21H           ; Invoke DOS interrupt
```

### **Function 07H (Single character input without echo)**

This function works as same as the function of 01H but it not display the press character on screen.

### **Function 02H (Single character output)**

This function displays the signal character whose ASCII Code value is present in DL register.

```
MOV    DL, 'A'         ; Move the ASCII code of A ( A = 41H) in DL register
MOV    AH, 02H         ; Move the single character output function into AH register
INT     21H           ; Invoke DOS interrupt
```

### **Function 09H (Display string)**

The function display the string which is terminated by \$ sign present in Data segment register. Data Segment address is store in DS register and DX register store the offset address of the string to display.

.DATA

```
MESSAGE    DB    0AH, 0DH, "INDUS UNIVERSITY$"
```

.CODE

```
MOV    AX, @DATA        ; Get address of DATA segment
MOV    DS, AX           ; Move address of DATA segment into DS register
MOV    DX, OFFSET MESSAGE ;Get offset address of MESSAGE
MOV    AH, 09H          ;Move display string function into AH register
INT     21H             ; Invoke DOS interrupt
```

### **Function 4CH (DOS exit function)**

When the program terminate (end) the control must return back to the DOS. This is done by 4CH function.

```
MOV    AH, 4CH          ; Move return to DOS function into AH register
INT     21H             ; Invoke DOS interrupt
```

## Function 0BH (Check for Character available function)

This function is used to check any available character of keyboard. This function return two values in AL register, if it's 00H then no character available and if it's FFH then a character available.

```
MOV    AH, 0BH           ; Check for any Character available
INT     21H              ; Invoke DOS interrupt
```

## Function 2AH (Get system Date function)

This function returns the Date of the system. The return values are store in varies registers that are CX (year (1980-2099)), DH (month (00-11)), DL (day (0-30)) and AL (day of the week (00 for Sunday))

```
MOV    AH, 2CH           ; Get current time of the system
INT     21H              ; Invoke DOS interrupt
```

## Function 2CH (Get system Time function)

This function return the time of the system. The return values store in varies registers that are CH (hour (0-23)), CL (minutes (0-59)), DH (second (0-59)) and DL (1/100 second (0-99))

```
MOV    AH, 2CH           ; Get current time of the system
INT     21H              ; Invoke DOS interrupt
```

Hour store in CH register, Minutes store in CL register, Second store in DH register and milli-second store in DL register.

### Tasks:

- 1- Write a program to take 4 character input from user and display user input on screen

```
01 .model small
02 .stack 100h
03 .data
04
05 A db 'enter the String : $'
06
07 MESSAGE DB 0AH, 0DH, "your input were : $"
08
09 .code
10
11 main proc
12     mov ax, @data
13     mov ds, ax
14     mov ah, 9
15     lea dx, a
16     int 21h
17     mov ah, 1
18     int 21h
19     mov ah, 1
20     int 21h
21     mov ah, 1
22     int 21h
23     mov ah, 1
24     int 21h
25
26     mov ax, @data
27     mov ds, ax
28     MOV DX, OFFSET MESSAGE
29     mov ah, 9
30     int 21h
31
32     mov dl, al
33     add dl, 61
34     sub dl, 61
35     add dl, 61
36     mov ah, 02h
37     int 21h
38     ret
```

```

38 mov dl, al
39 mov ah, 02h
40 int 21h
41 mov dl, al
42 mov ah, 02h
43 int 21h
44 mov dl, al
45 mov ah, 02h
46 int 21h
47
48     mov ah, 4ch
49     int 21h
50
51     main endp
52 end main
53
54
55

```

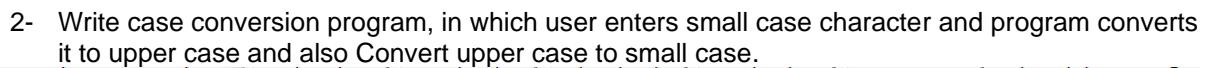
emulator: lab 03 part 01.exe\_

file math debug view external virtual devices virtual drive help

Load reload step back single step run step delay ms: 0

registers		F400:0200		F400:0200	
	H	L			
AX	01	24	F4200: FF 255 RES	BIOS DI	
BX	00	00	F4201: FF 255 RES	INT 021h	
CX	01	7D	F4202: CD 205 =	IRET	
DX	00	00	F4203: 21 033 !	ADD [BX + SI], AL	
CS	F400		F4204: CF 207 ±	ADD [BX + SI], AL	
IP	0200		F4205: 00 000 NULL	ADD [BX + SI], AL	
SS	0710		F4206: 00 000 NULL	ADD [BX + SI], AL	
SP	00FA		F4207: 00 000 NULL	ADD [BX + SI], AL	
BP	0000		F4208: 00 000 NULL	ADD [BX + SI], AL	
SI	0000		F4209: 00 000 NULL	ADD [BX + SI], AL	
DI	0000		F420A: 00 000 NULL	ADD [BX + SI], AL	
DS	0720		F420B: 00 000 NULL	ADD [BX + SI], AL	
ES	0700		F420C: 00 000 NULL	ADD [BX + SI], AL	
			F420D: 00 000 NULL	ADD [BX + SI], AL	
			F420E: 00 000 NULL	ADD [BX + SI], AL	
			F420F: 00 000 NULL	ADD [BX + SI], AL	
			F4210: 00 000 NULL	ADD [BX + SI], AL	
			F4211: 00 000 NULL	ADD [BX + SI], AL	
			F4212: 00 000 NULL	ADD [BX + SI], AL	
			F4213: 00 000 NULL	ADD [BX + SI], AL	
			F4214: 00 000 NULL	ADD [BX + SI], AL	
			F4215: 00 000 NULL	ADD [BX + SI], AL	
				...	

screen source reset aux vars debug stack flags



The screenshot shows a DOS emulator interface. On the left, the assembly code is displayed with line numbers 01 to 22. The code defines a small model, stack, and data segment, then enters a main procedure. The main procedure moves the value 1 into register AH, calls interrupt 21h, moves the value in AL into DL, adds 32 to DL, moves the value in AH into register AH, calls interrupt 21h again, and ends the procedure. Below the code, a small window shows the output screen with the characters 'Aa'.

In the center, the registers window shows the current state of the CPU registers. The registers are: AX (4C 61), BX (00 00), CX (01 11), DX (00 61), CS (F400), IP (0204), SS (0710), SP (00FA), BP (0000), SI (0000), DI (0000), DS (0700), and ES (0700).

On the right, the memory dump window shows the contents of memory starting at address F400:0204. The dump is organized into two columns. The left column shows memory addresses from F4200 to F4215, with their corresponding hex values and ASCII representations. The right column shows the BIOS interrupt vector table, starting with INT 021h pointing to IRET, followed by a series of ADD instructions for interrupts 02h through 0Ah.

At the bottom, there is a window titled 'original source code' which displays the assembly code for the main procedure, specifically lines 06 to 10.

3- Write a program that print \*\*\*\* on the output screen.(4 character password)

```

01 | model small
02 | .stack 100h
03 | .data
04 |
05 | A db 'enter the String : $'
06 |
07 | MESSAGE DB 0AH, 0DH, "your input after converting into password were : $"
08 |
09 | .code
10 |
11 | main proc
12 |     mov ax, @data
13 |     mov ds, ax
14 |     mov ah, 9
15 |     lea dx, a
16 |     int 21h
17 |
18 |     mov ah, 1
19 |     int 21h
20 |     mov dl, al
21 |     mov ah, 2
22 |     int 21h
23 |
24 |
25 |     mov ah, 1
26 |     int 21h
27 |     mov dl, al
28 |     mov ah, 2
29 |     int 21h
30 |
31 |
32 |
33 |
34 |
35 |     mov ax, @data
36 |     mov ds, ax
37 |     MOV DX, OFFSET MESSAGE
38 |     mov ah, 9
39 |     int 21h
40 |
41 |                                     ;Displaying User's input
42 |     mov dl, al
43 |     add dl, 6
44 |     mov ah, 02h
45 |     int 21h
46 |     mov dl, al
47 |     mov ah, 02h
48 |     int 21h
49 |     mov dl, al
50 |     mov ah, 02h
51 |     int 21h
52 |     mov dl, al
53 |     mov ah, 02h
54 |     int 21h
55 |
56 |     mov ah, 4ch
57 |     int 21h
58 |
59 |     main endp
60 | end main
61 |
62 |

```

line: 10

col: 8

drag a file here to open

