Experiment 3

3 Basic Input Output

Introduction

The aim of this experiment is to introduce the use of input/output through the DOS interrupt.

Objectives:

- INT Instruction
- Keyboard access using DOS INT 21H function calls 01, 02 and 08.
- OFFSET operator
- String reading and display using the DOS INT 21H function call 09and 0AH
- Some basic assembly instructions

3.1 The INT Instruction

You might have noticed in the first experiments that we made use of INT 21H to exit the program. In fact, INT 21H is a built in program available as part of the BIOS. This built in program can perform a multitude of functions. However, INT 21H is not the only available interrupt. There are over 50 different interrupts available when using DOS. Of these, the programmer will only use a few. Each interrupt though, has a number of functions which select the individual task that the function has to do. The INT instruction calls a DOS interrupt (like a function) to perform a special task.

For example, there is just one interrupt for accessing the mouse INT 33h, but there are separate functions available to see if a button has been clicked, to see how far the mouse has moved, to display or hide a mouse pointer … etc.

Example:

From table 3.1 it can be seen that interrupt 21h function 02, outputs a character on the screen. So to output the character "!" on the screen, use the following code:

```
MOV AH,02   ; To select function 2, move the number 2 to AH
MOV DL,"!"  ; Move character to output in DL
INT 21h     ; Finally when all registers are set the interrupt is called
```

Function 4CH is the function which terminates the program and returns the user to the operating system.

3.2 Basic Input Output

The following table summarizes the main I/O functions used through INT 21H. These functions are mainly used to read a character or a string from the keyboard, which could stand for input data to a program, and display characters or strings, which also could be results, or an output, of a program. Input and output can only be a character or a string.

Function	**Input in**	**Output in**	**Effect**
01H	AH	AL	Reads a character with echo on the screen.
02H / 06H	AH, Character in DL	No output	Displays a character on the screen
08H	AH	AL	Reads a character without echo.
09H	AH	No output	Displays a string terminated by a '$' sign
0AH	AH	Offset in DX	Reads a string of characters from the keyboard

Table 3.1: Simple I/O DOS function calls

3.2.1 Character Input Output Functions

3.2.1.1 Displaying a Character

DOS functions 02 and 06 are used for a single character display. Function 09 is used to display a string. Both functions, 02 and 06, are identical except that function 02 can be interrupted by a control break (Ctrl-Break), while function 06 cannot. To display a single character ASCII character at the current cursor position, use the following sequence of instructions:

```
MOV AH, 06H                ;(Or: 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 or the character itself written between single quotes.

The following displays the number '2' using its ASCII code:

```
MOV AH,  06H
MOV DL,  32H        ; or MOV DL, '2'
INT 21H
```

3.2.1.2 Reading a Character

These include reading a single character with or without echo preformed respectively by functions 01 and 08.

Function 01H

To read single character and have it echoed, or displayed, on the screen, use the following code:

```
MOV AH, 01H
INT 21H
```

After this code, AL contains now the ASCII code of the character read from the keyboard.

Function 08H

If the character is to be read without echo, such as reading a password, use the following code:

```
MOV AH, 08H
INT 21H
```

AL contains now the ASCII code of the character read

3.2.2 Dealing with Strings

Dealing with strings is done through functions 09 and 0AH, respectively for displaying and reading. However, we need to introduce a few useful tools before getting to write programs that deal with strings.

- The Offset Operator
- LEA Instruction
- Array Indexing
- Keyboard Buffer declaration

3.2.2.1 The Offset Operator

The offset operator returns the number of bytes between the label and the beginning of its segment, or its effective address. Since it produces a 16-bit immediate value, the destination must be a 16-bit operand. The statement: **offset label,** returns the offset address portion of label, which can be saved in a 16-bit register or memory location.

Examples

1. After executing the following instruction,

```
MOV   BX, OFFSET COUNT          ; Let BX point to count
```

the BX register contains the address of the variable count. BX may thus be used as a pointer to the count variable.

2. The following example illustrates the use of the OFFSET operator:

```
.DATA
        BLIST       DB     10H, 20H, 30H, 40H
        WLIST       DW     1000H, 2000H, 3000H
.CODE
        MOV   DI,   OFFSET BLIST          ; DI = 0000
        MOV   BX, OFFSET BLIST+1          ; BX = 0001
        MOV   SI,   OFFSET WLIST+2        ; SI = 0006
```

3.2.2.2 The LEA Instruction

The MOV instruction when dealing with Offsets, as in the above examples, may be replaced by the following instruction:

```
LEA BX, COUNT     ; equivalent to MOV   BX, Offset COUNT
```

3.2.2.3 Array Indexing

When dealing with strings, we often need to access specific items within the string. Therefore, some sort of indexing is necessary. Strings are considered as arrays indexed starting from 0. To access the Nth character in the string KeybBuf, use KeybBuf [N-1]

Usually the BX register is used as an index, and helps accessing a specific character within a string:

```
        MOV BX, Position             ; Position could be a number
        MOV AL, KeybBuffer[BX]
```

3.2.2.4 String Display Function

Function 09 is used to display a string of characters ended with a ‘$’ sign. The following code displays the string MESSAGE defined as:

```
.DATA
        MESSAGE DB ‘This is the Message to be displayed’, ‘$’

.CODE
        LEA DX, MESSAGE          ; or MOV DX, OFFSET MESSAGE
        MOV AH, 09H
        INT 21H
```

A. Reading a String

Reading a string is accomplished by function 0AH. This function accepts a string of text entered at the keyboard and copies that string into a memory buffer. Function 0AH is invoked with DX pointing to an input buffer, whose size should be at least three bytes longer than the largest input string anticipated.

String Declaration

Strings, like arrays, are declared using the DUP operator. To declare an array **MyStr** of 20 bytes:

MyStr DB 20 DUP (?)

Keyboard Buffer Declaration

Before invoking function 0AH, you must set the first byte of the buffer with the maximum number of characters to be expected in the buffer. After returning from function 0AH, the second byte of the buffer will contain the number of characters actually read form the keyboard (**Table 3.3**).

Buffer Length	Actual Length								

Figure 3.1: Keyboard buffer structure

Function 0AH	Read from Keyboard
Entry	AH = 0AH ; DX = address of keyboard input buffer First byte of buffer contains the size of the buffer (up to 255)
Exit	Second byte of buffer contains the number of characters read. Reading operation continues until buffer full, or a carriage return (CR = 0DH) is typed.

Table 3. 2: Functions 0AH of DOS interrupt

Example

This example shows the use of function 0AH. The user is supposed to enter the word "**hello**". We prepare a buffer to enter a string of maximum 10 characters.

Buffer:

KeybBuffer DB 11, 12 DUP (?)

11	**?**	?	?	?	?	?	?	?	?	?	?	?

The first number (**11**) indicates the length of the buffer including the Carriage return key entered after the string.

Code:

```
        LEA DX,   KeybBuffer
        MOV AH, 0AH
        INT 21H
```

Input:

The user enters through the keyboard the word "**hello**"

Output:

11	**05**	68	65	6C	6C	6F	**0D**	?	?	?	?	?

- Bold characters are returned by function 0Ah
- The second number (**05)** indicates the number of characters effectively read from the keyboard.
- The eighth number (**0Dh)** corresponding to the carriage return is not shown on the screen.

3.3 Some Basic Assembly Instructions

3.3.1.1 The LOOP instruction

In this experiment, the FOR loop equivalent in assembly language is introduced. The LOOP instruction uses the CX register as a counter.

Syntax:

```
        MOV CX, Count
  next: …..
        …..
        ….
        LOOP next
```

Effect:

- decrements CX but does not change any flags
- if CX is not zero after the decrement, control is transferred to the *destination* label
- This is a SHORT jump only

3.3.1.2 The Unconditional Jump

This is mainly used to perform unconditional jumps such as infinite loops. The syntax is:

```
  here: …..
        …..
        ….
        JMP here
```

3.4 Lab Work

3.4.1 Part I

Write and test the following programs.

Program 1

; This program displays the characters A B C, using INT 21H function 02.

```
.MODEL SMALL
.STACK  200
.DATA
        X EQU 'B'
        Y DB   43H
.CODE
        MOV AX, @DATA
        MOV DS, AX              ; Assume Data Segment

        MOV AH, 02              ; Load Function 02
        MOV DL, 'A'             ; Load Character To Be Displayed
        INT 21H                 ; Call Interrupt 21h

        MOV DL, X               ; Load Character To Be Displayed
        INT 21H                 ; Call Interrupt 21h

        MOV DL, Y               ; Load Character To Be Displayed
        INT 21H                 ; Call Interrupt 21h

        MOV AH, 4CH             ; Exit to DOS
        INT 21H
END
```

Program 2

; This program displays a string terminated by a $ sign using INT 21H function 09H.

```
.MODEL SMALL

.DATA
        MESSAGE DB 'This is the message to be displayed', '$'
.STACK  200
.CODE
        MOV AX, @DATA
        MOV DS,  AX
        MOV DX,  OFFSET MESSAGE
        MOV AH,  09H
        INT 21H
.EXIT
END
```

Program 3

```
.MODEL SMALL
.STACK  200
.DATA
        MESSAGE DB ‘Enter a character: ’, ’$’
        MESSAGE2 DB ‘The character you typed is: ’, 0DH,  0AH, ’$’
.CODE
.STARTUP
        LEA DX, MESSAGE
        MOV AH, 09H
        INT 21H         ; Display message

        MOV AH, 02      ; Function 02H, display character
        MOV DL, AL      ; Load character to be displayed
        INT 21H         ;

        LEA DX, MESSAGE
        MOV AH, 09H
        INT 21H

        MOV AH, 08H     ; Function read character without echo.
        INT 21H         ; Character read is returned in AL register. No echo on the display.

        MOV BL, AL      ;Save character read in BL register
        LEA DX, MESSAGE2
        MOV AH, 09H     ;Display MESSAGE2
        INT 21H

        MOV AH, 02      ; Function 02H, display character
        MOV DL, BL      ; Load character to be displayed from BL
        INT 21H
.EXIT
END
```

3.4.2 Part II

1. Write a program that displays a string on the screen then asks the user to enter a number between 1 and 9. The program should then display the character whose position is indicated by the number entered by the user.

2. Write a program that reads a string from the keyboard then displays the string back on the screen together with the number of characters in the string.

3. Write a program that reads a string from the keyboard, reverses the string, then displays it back on the screen.