Lab: 6

Familiarization with DOS Service INT 21H, BIOS Service 10H and Miscellaneous Assembly Language Problems

a) Familiarization with DOS Service INT 21H in Assembly Language Programming

DOS Service

In assembly language DOS functions and interrupts are used for input/output services. An interrupt occurs when any currently executing program is disturbed (interrupted). Interrupts are generated for a variety of reasons, usually related to peripheral devices such as keyboard, disk drive or printer. The Intel 8086 microprocessor recognizes two types of interrupts; hardware and software. Hardware interrupt is generated when a peripheral device needs attention form microprocessor. A software interrupt is a call to subroutine located in the operating system, usually an input-output routine. i.e. a software interrupt calls a built-in subroutine form the operating system usually DOS for input and output operations.

`INT 21H` is a DOS service for different purpose. This service has many functions but few of them are given below. The function no of the service is to be loaded in register AH and the other registers are loaded with the data as required before the interrupt call.

DOS Service `INT 21H`

Func No. Description

- `00H`. Terminate the current program: `INT 21H`, function `4CH` is used instead
- `01H`. Console input with echo: wait for a character from the standard input device. The character is returned in `AL` and echoed. Respond to `CTRL+BREAK`.
- `02H`. Character output: Send the character in `DL` to the standard output device.
- `05H`. Printer output: Send the character in `DL` to the parallel printer port.
- `06H`. Direct console input-output: Reads the character in `AL` if `DL=0FFH` else displays character at `DL` to the standard output device. Control characters are not filtered.
- `07H`. Console input: Wait for a character from the standard input device. The character is returned in AL, but not echoed. It does not respond to `CTRL+BREAK`.
- `08H`. Console input without echo: Wait for a character from the standard input device. Ther character is returned in `AL`, but not echoed. Respond to `CTRL+BREAK`.
- `09H`. String output: Send a string of characters to the standard output device until $ character is reached. `DX` contains the offset address of the string.
- `0AH` Read string: Read characters from the standard input device. `DX` points a location whose first byte gives the max characters allowed to enter, the next byte reserved to store the actual no of characters entered and the rest space to store the entered characters.
- `0BH` Check keyboard Status: Returns `FFH` in `AL` if an input character is available in the keyboard buffer else returns `00H` in `AL`.
- `0CH` Clear key board buffer and invoke input functions: The input functions are stored in `AL` and other registers should hold the values as required.

Assignments:

1. Write a program to display a string "Programming is Fun" in the screen using string displaying function
2. Write a program to display the same string using character displaying function (use current address operator $ to count the no of characters e.g.

```
STR DB "String to be displayed"
LEN DW   $-STR    ;Gives the length of the string STR
```

3. Write a program to read string from the user (use function that reads string) and display the string in another line. (To display the character in new line display characters `0DH` and `0AH`)
4. Write a program to read the string using the character reading function and display the string using character displaying function.
5. Write a program to read the string and convert it into upper case if it is in lower case and display the resulting string. Process the string in the memory before displaying
6. Write a program to read a string and display each word in separate lines.

b) Familiarization with BIOS Service INT 10H in Assembly Language Programming

BIOS Service

BIOS provides interrupt service 10H for video display control. `INT 10H` also has many functions like `INT 21H,` and some of them are given below. The function no of the service is to be loaded in register AH and the other registers are loaded with the data as required by the function before interrupt call.

`BIOS` Service `INT 10H`

Func. No. Description

`00H` Set video mode. Load the required mode in `AL`. This operation also clears the screen.

`01H` Set cursor size. To set cursor vertically set the register `CX` as:
`CH` (bits 4-0): starting scan line
`CL` (bits 4-0): ending scan line

`02H` Set cursor position anywhere on a screen according to row:column coordinates. Set the registers as follows:
`BH`: page number (0 is the default), `DH`: row, and `DL`: column.

`03H` Return cursor status i.e. to determine the present row, column and size to the cursor. Store the page number in `BH`.
The operation leaves `AX` and `BX` unchanged and returns these values:
`CH`: Starting scan line `CL`: Ending scan line
`DH`: Row `DL`: Column

`05H` Select the page that is to be displayed. We can create different pages and request alternating between pages.

`06H` Scroll upward of lines in a specified area of the screen. Displayed lines scroll off at the top and blank lines appear at the bottom. Setting AL to 0 caused the entire screen to scroll up, effectively clearing it. Setting a nonzero value in AL causes the number of lines to scroll up. Set the following registers as:
`AL`: Number of rows (00 for full screen) `CX`: Starting row, Column
`BH`: Attribute of pixel value `DX`: Ending row, Column

`07H` Scroll down screen. Scrolling down the screen causes the bottom lines to scroll off and blank lines to appear at the top. It works the same as function `06H`, except the fact that this operation scrolls down. Set the following registers as:
`AL`: Number of rows (00 for full screen) `CX`: Starting row, Column
`BH`: Attribute or pixel value `DX`: Ending row, Column

`08H` Read character and its attribute at cursor from the video display area. Before calling interrupt, set the page number in `BH` register.

`09H` Display a specified number of characters at cursor according to given attribute. Set the registers as:
`AL`: ASCII character `BL`: Attribute or pixel value
`BH`: Page number `CX`: Count
The count in `CX` specifies the number of times the operation is to repetitively display the character in `AL`.

`0AH` Display character at cursor. The difference with `09H` is that function `09H` sets the attribute whereas function `0AH` used the current value.
`AL`: ASCII character `BL`: Pixel value (graphics mode only)
`BH`: Page number `CX`: Count

`0BH` Set the color palette. The value in `BH` (00 or 01) determines the purpose of `BL`
`BH` = 00: Select the background color, where BL contains the color value in bits 0-3 (any of 16 colors).
`BH` = 01: Select the palette for graphics, where `BL` contains the palette (0 or 1).

`0CH` Display a selected color (background and palette) in graphics mode. Set the registers as:
`AL`: Color of the pixel `CX`: Column
`BH`: Page number `DX`: Row

`0DH` Read pixel dot to determine its color value. For this set page number in `BH`, column in `CX`, and row in `DX`. The operation returns the pixel color in `AL`.

`0EH` Monitor is used as a terminal for simple displays in test and graphics modes. For this set `AL` by the character to display, and `BL` by the foreground color.

`0FH` Get current video mode. The operation returns the values as:
`AL`: Current video mode `AH`: umber of screen columns
`BH`: Active video page

Assignments:

1. Write an assembly language program to scroll a window from row 5, column 20 to row 20, column 60 with a reverse video attributes. Then locate the cursor at row 12, column 30. And display a string as "Programming in Assembly Language is Fun".
2. Write an assembly language program that takes a string (having 60 characters at max.) as input from user, and display the string at the center of the clear screen.
3. Write an assembly language program that takes a string (having 24 characters at max.) from the user and display each character at the center of each line.
4. Write an assembly language program that takes a string (having 14 characters at max.) input from user and scroll a window of size 20×20 at the center of screen. Then display the string at the center of scrolled window. (You can choose the color attribute yourself).
5. Write a program that a string from the user and display each word in new line diagonally from upper left towards bottom right in a clear screen. If the string is "Programming in Assembly Language is Fun", it shoud be displayed as

   Programming
   in
   Assembly
   Language
   is
   Fun

c) Solving Miscellaneous Problems with Assembly Language Programming

Miscellaneous Problems

In most of the cases we have to take input from the keyboard and display the result in the screen. As you know that if you take input from the user it is in ASCII format and to display a character it should be in ASCII format again. For the characters it is ok, but what happens when you have to process numbers. In this case you have to convert the ASCII character in to numeric form when taking input and before displaying you have to convert the number into ASCII character form.

To convert a character that is read by `INT 21H` function 01 we process as follows.

```
MOV AH,01
INT 21H
SUB AH,30H ; Converts the ASCII character to binary number
```

When taking multiple digit number you have to make a loop to read characters (actually numbers) convert the characters to numbers as above and store the final number in some location in data segment.
For multiple digit number we process as follows.

```
        .DATA
NUM   DB 0       ;Space for three digit number
COUNT DB 0       ;Counter to count the no of digits entered
NEWLN DB 0DH,0AH,'$'
        .CODE
MAIN PROC FAR
        MOV AX,@DATA
        MOV DS,AX

        MOV CX,03       ;For three digit decimal number
        MOV SI,OFFSET NUM

       ;Read characters for a number and add with prev num
  L1:   MOV AH,01
        INT 21H
        CMP AL,0DH      ;Stop if the user presses return
        JE LP
        SUB AL,30H      ;Convert the digit to ASCII
        PUSH AX            ;Save the digit for further purpose
        MOV AL,10       ;Multiply NUM by ten and store it back
        MUL NUM        ;As we are using byte operation the result
```

```
        MOV NUM,AL      ;will not be greater than byte
        POP AX           ;recover back the number
        ADD NUM,AL       ;Add it with the previous result
        INC COUNT        ;Count the number of digits entered
LP:     LOOPNE L1
```

Without converting the digits entered at the same instant, the read digit are first stored in data segment then processed later. (See Abel's book page no 250). Read a string store in the memory and then convert to binary it later.
For the displaying purpose also we proceed in the same way. In this case we do the reverse process; convert the binary number to ASCII decimal format.
To display a number in memory at `NUM` into decimal format we process as follows

```
        ;Back to the ASCII format for display
        MOV CX,10
        MOV AL,NUM
        MOV AH,0
        MOV BX,0

        ;Find each decimal digit of the number and store in stack
L4:     MOV DX,0
        DIV CX      ;Divide by 10
        ADD DX,30H  ;Convert the digit to characters
        PUSH DX           ;Store the decimal digit in the stack
        INC BX
        CMP AX,0 ;Stop if the number is <= 0
        JA L4

        ;Get characters from stack and display as decimal number
        MOV AH,02
        MOV CX,BX
DISP:   POP DX
        INT 21H           ;Display the character
        LOOP DISP
```

The processing can be done directly in the data segment. (See Abel's book page no 251). Convert the binary number into ASCII decimal format and display the final ASCII string that represents the decimal number.
To convert the binary number into ASCII Hexadecimal format divide the number by 16 instead of 10 and add 30H to the remainder numbers from 0 to 9 and add 41H to the remainder numbers from 10 to 15.

Assignments:

1. Write a program to find the sum of numbers from 1 to *n*. Read n from the user and display the sum as the decimal format. (also try to display the sum in Hexadecimal format)
2. Write a program to find the sum of the following series up to the terms specified by the user and display the result in decimal format. (also try to display the sum in Hex format)
   2 x 4 + 3 x 6 + … to n terms
3. Write a program that takes a string and count the number of words in the string. Display the count in decimal format
4. Write a program that takes a string and count the no of vowels in the string. Display the count in decimal format.
5. Write a program to add ten 16-bit numbers stored in memory and store and display the result in decimal format.