

COAL Project

Restaurant Menu System

Overview:

Keeping in view the concepts of Assembly Language, I have developed a Restaurant Menu System. The main function of this automated Restaurant Menu System is to display the different menu options to the user from which user can select the food items to order. Based on the food items the bill is calculated at the end of the program.

Main Features:

- Display three main Menu Options.
- Display sub menus.
- Choose food Items.
- Calculate and Display Bill.

Basic Working:

Main Concepts of Assembly Language such as different Jumps, Subroutines and Interrupts have been used in the program. First of all, main menu is displayed to the user using concept of video memory. From main menu, user can go to sub menus such as sub menu for Appetizers, Main Dishes or Desserts. From there user has an option to choose between different food items, calculate bill and go to main menu or exit the program. There is a different Subroutine for each functionality. The concepts used have been explained below:

Subroutines Used:

Subroutine to clear Screen:

The first subroutine used is for clearing the console. This subroutine does not have any local variable. All registers used in this subroutine are first pushed into stack and then for clearing the screen concepts of video memory have been used.

```

mov ax, 0xb800
mov es, ax
xor di, di
mov ax, 0x0720
mov cx, 2000

cld
rep stosw

```

In order to access video memory we first load 0xB8000 into ES. We then load 0x0720 into **ax**. In which **07** is for the color black background and **20** is the ASCII for space. String instruction **stosw** is used which grabs value from **ax** and loads it into **es:di**. This is done repeatedly **cx** times because of the **rep** instruction used before it. The **cld** instruction is used to clear direction flag so that **di** is incremented by 2 each time. In the end we return from subroutine after doing all necessary pops from stack.

Subroutine to Print Main Menu:

This subroutine is for printing the main menu. It also uses the concepts of video memory. All the strings that need to be printed, along with their lengths, are first pushed into stack before calling the subroutine. After calling the subroutine, **registers** and **bp** are pushed into stack and **bp** is set to **sp** in order to access the local variables. For the first string, **di** is set to **2920**(cell number), in order to print the string in the middle of the screen, and the style byte(**ah**) is set to **03** which gives blue color to the string. For printing the string the **printstr** subroutine is called from the current subroutine.

printstr:

```

nextchar:
    mov al, [si]      ; char to print
    mov [es:di], ax   ; load char to video memory
    add di, 2         ; jmp to next cell of video memory
    add si, 1         ; jmp to next char

    loop nextchar

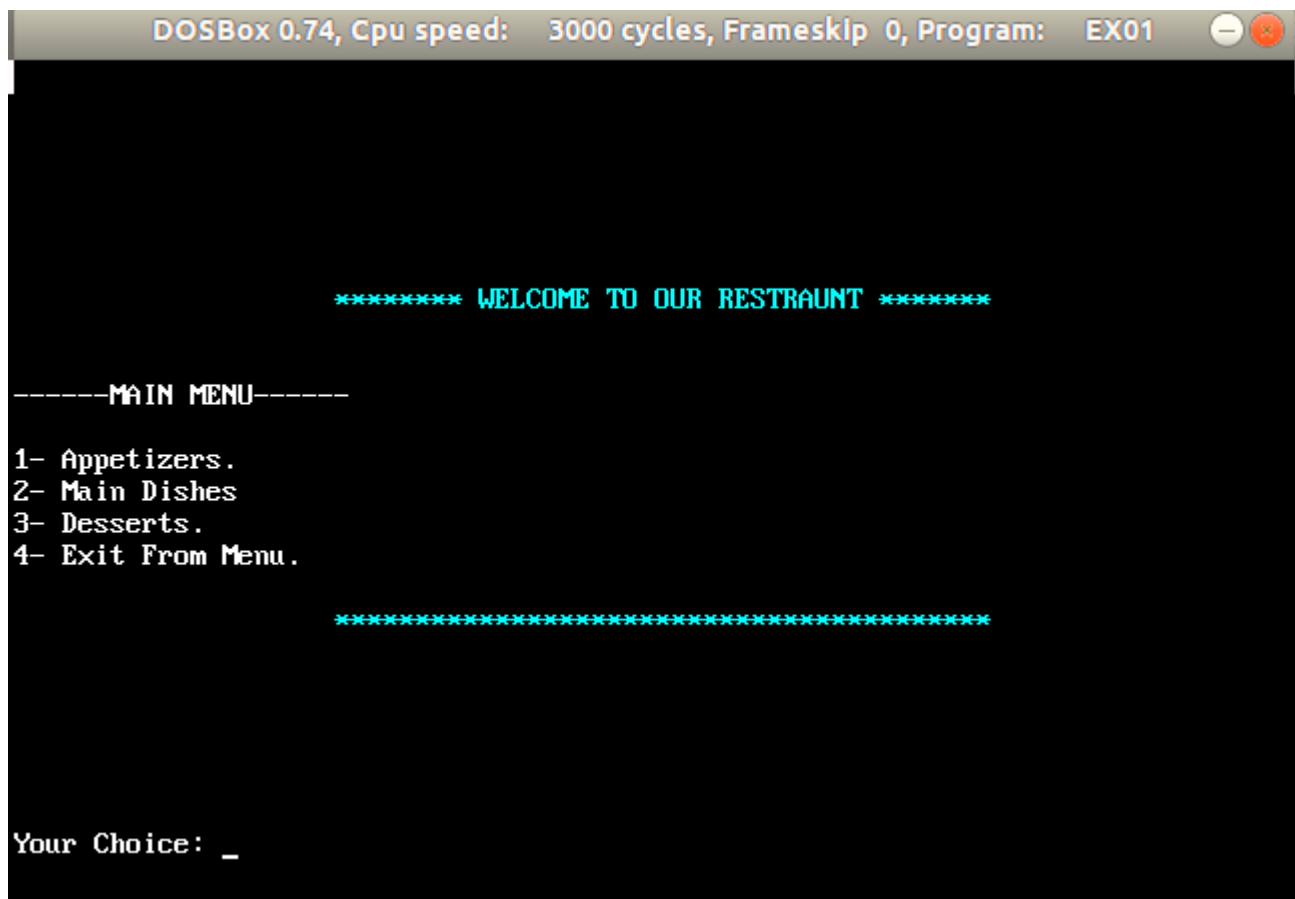
ret

```

SI contains the address of characters in the string and the loop continues **CX** times. **CX** contains the length of the string. On each iteration value of **ax** is displayed on **[es:di]** location in the video memory.

The same procedure is done for printing all the strings pushed into stack before calling the `main_menu` subroutine. For each string we first load the string into **SI** and the length into **CX** and then call the `printstr` subroutine.

Screenshot of Output obtained from calling `main_menu` subroutine:



```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: EX01

***** WELCOME TO OUR RESTAUNT *****

-----MAIN MENU-----
1- Appetizers.
2- Main Dishes
3- Desserts.
4- Exit From Menu.

*****

Your Choice: _
```

Subroutine to Calculate Length of String:

For printing the length of the string, this subroutine is used. Before calling this subroutine we first push **ds** into stack. We also push the string whose length needs to be computed. Inside the body of the subroutine **les** (load **ES**) instruction is used that loads into **DI** the value **from bp+4** (address of string we pushed onto stack before calling subroutine) and **DS** is loaded into **ES**.

```

les di, [bp+4]    ; load DI from BP+4 and ES from BP+6
mov cx, 0xffff

xor al, al
repne scasb

mov ax, 0xffff
sub ax, cx        ; find how many times CX was decremented

dec ax

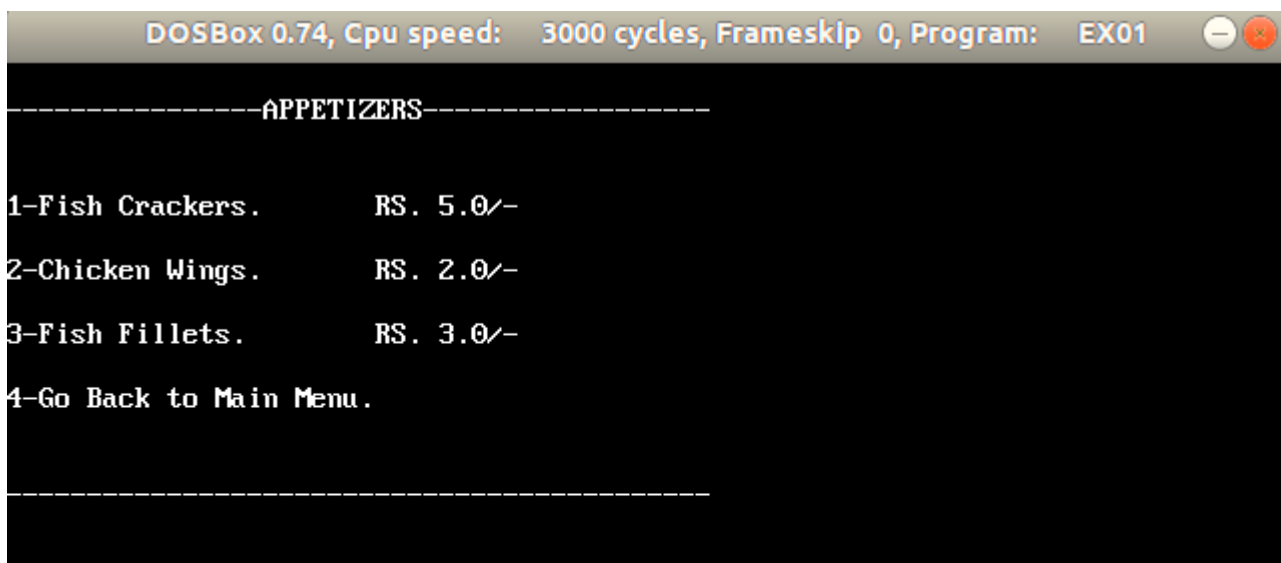
```

The ***repne scasb*** instruction is used which compare the value of **[es:di]** with **al** repeatedly until the values become equal and also the value of **cx** is decremented repeatedly and **di** is incremented by **2**. Initially **al** is set to **0** and **cx** is set to maximum possible length. Since **cx** is decremented on each repetition so at the end we subtract **cx** from **0xffff** which results in the total length of the string but since the string have a null terminator so we decrement the original length by one. Finally, we return from the function and **AX** contains the total length.

Printing Sub Menus:

For printing different sub menus, based on user choice, 3 different labels i.e Appetizers, Main Dishes, Desserts are used. Depending on the choice of the user we jump into any of the above labels from main of the program. Code logic is same under all three labels but the menu items are different for each. All we do here is display all the strings (menu items) and than take user input to choose between items or go back to main menu. Once the user choose the item , we then call the calculate_bill subroutine.

Screenshot of output for sub menu have been attached below:



MS-DOS Functions used:

MS-DOS provides user with many function. Each function having a different number.

For displaying string (INT 21h Function 09h):

In order to write strings to standard output, the following MS-DOS function have been used.

```
lea dx,[item1]
mov ah,09h
int 21h
```

For this code to work fine, the string must be terminated by a '\$' character. **DS** must point to the string's segment, and **DX** must contain the string's offset. **LEA** (load effective address) is used to load offset of string to **dx**.

The above code is used repeatedly for printing all menu items on console.

For taking user input(INT 21h Function 01h):

The following MS-DOS function is used to take input character from user and echo it:

```
mov ah,01h
int 21h
```

Subroutine for inserting new line:

In this subroutine two MS-DOS functions have been used. One for carriage return and the other for adding a new line. Both are explained below:

For Adding new Line(INT 21h Function 0Ah):

The following MS-DOS function is used to move to next output line:

```
mov ah,2      ; insert newline
mov dl, 0AH
int 21h
```

For Carriage return(INT 21h Function 0Dh):

The following MS-DOS function is used to move to the left most column:

```
mov dl, 0DH    ; carriage return
int 21h
```

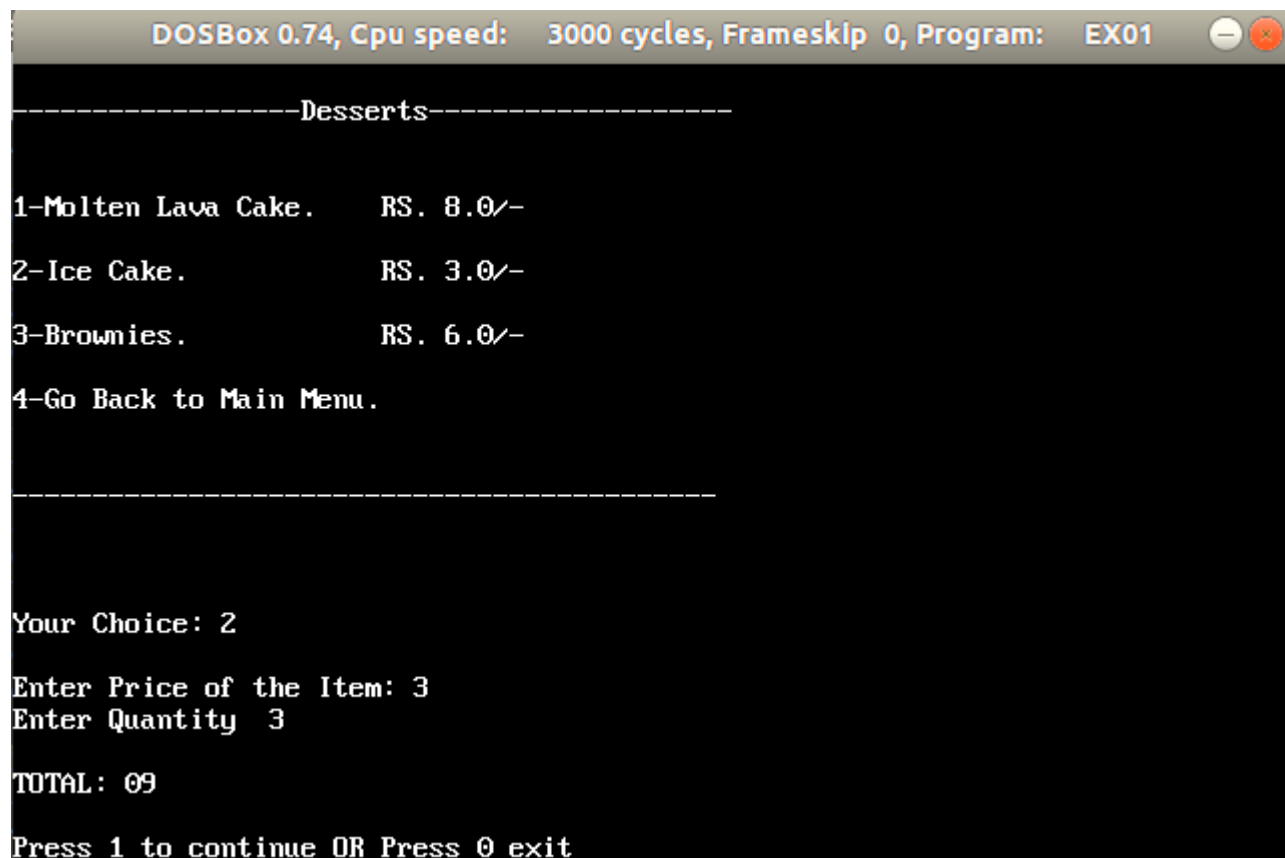
Subroutine to calculate bill:

This Subroutine simply asks the user to enter the quantity of the food item. The entered value is converted from ascii and multiplied with the price. It is then added in total and converted back to ascii for displaying on console the console. User is given an option to continue with menu or exit from program. If more items are chose, they are also added to total by calling the calculate_bill subroutine again.

```
sub al,30H      ; convert from ascii
mov [val2],al
mul byte[val1]  ; multiply price with quantity
mov [total],al  ; store in total for later adding more prices to it

add ah,30H      ;convert back to ascii for printing
```

Screenshot of Output from calculate_bill subroutine:



```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: EX01

-----Desserts-----

1-Molten Lava Cake.    RS. 8.0/-
2-Ice Cake.            RS. 3.0/-
3-Brownies.            RS. 6.0/-
4-Go Back to Main Menu.

-----

Your Choice: 2
Enter Price of the Item: 3
Enter Quantity 3
TOTAL: 09
Press 1 to continue OR Press 0 exit_
```

Subroutine to Exit from the program:

This subroutine simply prints a 'Thank you' message and exit from the program. It is only called when the user chooses an option to exit.

exit:

call clrscr

lea dx,[bye] ; display thank you msg

mov ah,9

int 21h

call newline

call newline

mov ah,0x4c ;terminate from program

int 21h