# Computer Organization and Assembly Language

**Group Members:**

- **Kamran Shahzad (29)**
- **Asma Khalid      (06)**
- **Mah jabeen      (54)**

**Semester:**      3$^{rd}$

**Submitted To:**   **Sir Talha & Sir Tayyab**

**Submitted On:**   **January 31$^{st}$, 2025**

## Restaurant Management System – Documentation

### Project Overview

**Title:** Restaurant Management System (RMS)

**Language:** Assembly language

### Purpose

The primary goal of the project is to create a text-based system for managing restaurant orders. It allows the user to choose between different meal schedules (Breakfast, Lunch, Dinner), select items from a predefined list, enter quantities, and calculate the total price for the selected items.

### Objective

The objective of this project is to develop a simple yet functional Restaurant Management System that performs the following tasks:

1. **Display a welcome message** upon startup.

2. **Allow the user to select from different meal schedules**: Breakfast, Lunch, or Dinner.

3. **Allow the user to choose from various menu items** and specify the quantity.

4. **Calculate the total price** based on selected items and quantities.

5. **Handle invalid inputs** gracefully by prompting the user to enter valid choices.

6. **Exit the program** or return to the main menu after completing an order.

### Features / Core Functionality

- **User Interface**: The program is text-based and interacts with the user via simple prompts and menu choices. It displays a welcoming message, menus for Breakfast, Lunch, and Dinner, and calculates total prices.

- **Menu Options**: Users can choose from a selection of meals available for each part of the day (Breakfast, Lunch, Dinner). Each item has a set price and is represented by a number that the user must select.

- **Price Calculation**: The program calculates the total price by multiplying the price of the selected item by the quantity entered by the user.

- **Error Handling**: The system includes error handling for invalid inputs. If a user selects an invalid option, the system will prompt them to try again.

- **Exit Option**: After completing an order or choosing to exit, the user can quit the program or return to the main menu.

## System Design

The design of this program consists of several key components that interact to create a seamless user experience. Below are the core elements and their functions:

### Data Storage and Variables

- **Strings (Message Prompts)**: The program uses db (data byte) directives to store message strings displayed to the user. These strings are used to display menus, instructions, and prompts to the user.

  - a1 to a6: Welcome message.

  - a7 to a39: Menus for breakfast, lunch, and dinner, including item names and prices.

  - a19, a34, a35, a36, a37: Input prompts and results (e.g., ask the user to choose an item, display the total price).

- **Registers**: The program uses 8-bit registers (such as al, bl, bh) to handle user input and calculations. These registers are also used to store temporary data (such as the quantity of items selected and the calculated total price).

  - al is used to store user input for menu choices and quantities.

  - bl stores unit prices of selected items (e.g., bl = 5 for items priced at 50/-).

  - ax is used to hold intermediate results during calculations.

### Program Flow

1. **Welcome Message**: The user sees a welcome message from the system when they first begin the program.

2. **Main Menu**: The user is prompted to select a choice:

   - 1: View the meal schedule (Breakfast, Lunch, Dinner).

   - 2: Exit the program.

3. **Meal Selection**: The program displays the options for Breakfast, Lunch, or Dinner, and the user selects a category.

4. **Item Selection**: After selecting a meal schedule, the system displays a list of items with prices. The user selects an item and enters a quantity.

5. **Price Calculation**: Based on the user's selection, the program multiplies the item's price by the quantity and displays the total cost.

6. **Error Handling**: If the user inputs invalid choices (e.g., selecting a non-existent item), the program will display an error message and return to the main menu.

7. **Exit**: After completing the transaction, the user is given an option to either return to the main menu or exit the program.
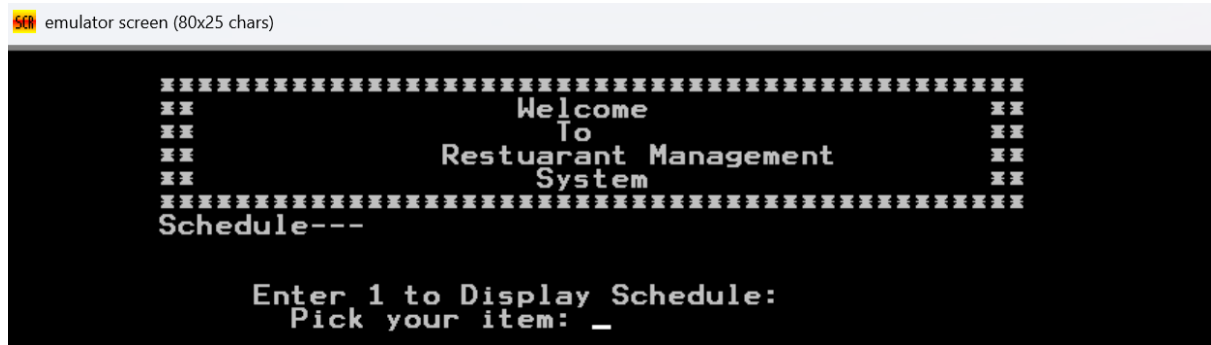
## 1. Data Section (Initialization and Messages)

The program begins by defining the **data section**, which holds various text strings used throughout the program.

```
;welcome page
a1 db 10,13,'    *******************************************  '
a2 db 10,13,'    **                Welcome              **  '
a3 db 10,13,'    **                   To                **  '
a4 db 10,13,'    **        Restuarant Management        **  '
a5 db 10,13,'    **                 System              **  '
a6 db 10,13,'    *******************************************  '
        .
```

```
SCR emulator screen (80x25 chars)

        *******************************************
        **                Welcome              **
        **                   To                **
        **        Restuarant Management        **
        **                 System              **
        *******************************************
        Schedule---

            Enter 1 to Display Schedule:
                Pick your item: _
```

a1 through a6 hold the welcome message strings.

Similarly, other strings are defined for menu prompts, item prices, and error messages.

## 2. Displaying the Welcome Message

```
056  .code
057  main proc
058        mov ax,@data
059        mov ds,ax
060
061        ;welcome page
062        mov ah,9
063        lea dx,a1
064        int 21h
```

This part loads the address of the welcome message (a1) into the dx register and calls interrupt 21h to display it on the screen.

## 3. Menu Display and User Input

The program displays menu options and waits for user input. After receiving the input, it converts the ASCII code to an integer and stores it for processing.

```
089    ;take input
090    mov ah,9
091    lea dx,a19
092    int 21h
093    mov ah,01h
094    int 21h
095    mov bh,al
096    sub bh,48
```

```
Enter 1 to Display Schedule:
        Pick your item: 1


Schedule---
  1. BreakFast
2. Lunch
3. Dinner
Enter your choice:
```

This code allows the program to accept user input and perform the necessary logic based on the choice (e.g., selecting Breakfast, Lunch, or Dinner).

## 4. Price Calculation

```
mov bl,5
lea dx,a35
mov ah,9
int 21h

mov ah,1
int 21h
sub al,48

mul bl
aam
```

```
Pick your item: 2
Enter quantity: 4
Total Price: 200/-
```

After the item and quantity are selected, the program calculates the total price using multiplication. The result is then adjusted to ASCII format for display.

## 5. Handling Invalid Inputs

```
mov ah,9
lea dx,a36
int 21h
jmp Exit
```

```
Invalid Input !! Re-run the program
        Total Price:
```

Whenever the user inputs an invalid choice (e.g., selecting a non-existent item), an error message is displayed, and the program returns to the main menu.

## 6. Error Handling

The program includes basic error handling for invalid input. This includes:

- **Invalid menu choice**: If the user selects a non-existent option, the system displays an error message and prompts them to try again.

- **Invalid item choice**: If an item number is out of range (not within the list for the selected meal), the system shows an error message and prompts the user again.

## 7. Calculations

The total price for the selected items is calculated using the formula:

Total Price = Item Price * Quantity

Where:

- The item price is retrieved based on the selected item (e.g., Breakfast items have a price of 50/-).

- The quantity is entered by the user.

The result is displayed as an ASCII string.

## 8. Known Limitations

While the program provides basic functionality, there are some limitations:

- **No advanced features**: The system does not include user management (e.g., logging in, saving orders) or dynamic pricing (e.g., discounts, taxes).

- **Fixed menu**: The menu is hardcoded into the program and cannot be modified during runtime.

- **Single-user system**: The program assumes a single user without any support for multiple simultaneous orders.

## 9. Future Enhancements

Here are some suggestions for future enhancements:

1. **Dynamic Menu**: Allow the menu to be updated during runtime, with the ability to add or remove items.

2. **Multiple Users**: Extend the system to handle multiple orders simultaneously, with each user able to track their own selections.

3. **Graphical Interface**: Implement a more classy user interface using a graphical library (if possible in assembly).

4. **Discounts and Offers**: Implement features like discounts or promotions based on item selections or total order amount.

## 10. Conclusion

The Restaurant Management System implemented in assembly language serves as a basic but functional model for simulating meal selection, price calculation, and order management in a restaurant setting. It demonstrates the ability to handle user input, process that input, and generate a simple output, while also including error handling mechanisms.

**Code:**

```
.model large

.stack 1000h

.data


;welcome page

a1 db 10,13,'   ********************************************* '

a2 db 10,13,'   **          Welcome            ** '

a3 db 10,13,'   **             To              ** '

a4 db 10,13,'   **      Restuarant Management      ** '

a5 db 10,13,'   **           System            ** '

a6 db 10,13,'   ********************************************* '


;choose

a7 db 10,13, '    Schedule---$'

a8 db 10,13, '    Enter your choice : $ '

a19 db 10,13,'      Enter 1 to Display Schedule : '

a34 db 10,13,'      Pick your item : $ '

a35 db 10,13,'      Enter quantity : $ '

a36 db 10,13,'      Invalid Input !! Re-run the program '

a37 db 10,13,'      Total Price: $ '

a38 db 10,13,'    1.Schedule: $'
```

a39 db 10,13,'      2.Exit: $'


;schdeule list

a9 db 10,13,'      1. BreakFast$'

a10 db 10,13,'     2. Lunch$'

a11 db 10,13,'     3. Dinner$'


;Breakfast List

a12 db 10,13,'      ******    BreakFast List   ******'

a13 db 10,13,'        1.Paratha + Dal       50/-$'

a14 db 10,13,'        2.Paratha + Vegetable   50/-$'

a15 db 10,13,'        3.Paratha + fried Egg   50/-$'

a16 db 10,13,'        4.Luchi + chicken Curry  50/-$'

a17 db 10,13,'        5.Chicken Soup + Nan    50/-$'

a18 db 10,13,'        6.Nehari + Naan       50/-$'


;Lunch LIst

a20 db 10,13,'      ******    Lunch List     ******'

a21 db 10,13,'        1.Kachchi Biryani      100/-$'

a22 db 10,13,'        2.Chicken Biryani      100,-$'

a23 db 10,13,'        3.Chicken Bhuna Khichuri 100/-$'

a24 db 10,13,'        4.rice + fish curry      100/-$'

a25 db 10,13,'        5.Rice + Chicken Curry  100/-$'

a26 db 10,13,'        6.Rice + Beaf Curry      100/-$'


;Dinner LIst

a27 db 10,13,'      ******    Dinner List     ******'

a28 db 10,13,'        1.Rice + Mutton Curry   200/-$'

a29 db 10,13,'        2.Rice + Beef Curry     200/-$'

a30 db 10,13,'        3.Pulao + Mutton curry  200/-$'

a31 db 10,13,'        4.Rice + Hilsha Fish Fry200/-$'

a32 db 10,13,'        5.Bottle Gourd soup     200/-$'

a33 db 10,13,'        6.Mixed Vegetable soup  200/-$'

```asm
.code
main proc
    mov ax,@data
    mov ds,ax

    ;welcome page
    mov ah,9
    lea dx,a1
    int 21h
    ;lea dx,a2
    ;int 21h
    ;lea dx,a3
    ;int 21h
    ;lea dx,a4
    ;int 21h
    ;lea dx,a5
    ;int 21h
    ;lea dx,a6
    ;int 21h

    ;newline
    mov ah,2
    mov dl,0ah
    int 21h
    mov dl,0dh
    int 21h

    mov ah,2
    mov dl,0ah
    int 21h
    mov dl,0dh
    int 21h
```

```asm
;take input
mov ah,9
lea dx,a19
int 21h
mov ah,01h
int 21h
mov bh,al
sub bh,48

cmp bh,1
je Schedule
jmp Invalid

Schedule:
;newline
mov ah,2
mov dl,0ah
int 21h
mov dl,0dh
int 21h

mov ah,2
mov dl,0ah
int 21h
mov dl,0dh
int 21h

mov ah,9
lea dx,a7
int 21h
lea dx,a9
int 21h
```

```asm
        lea dx,a10
        int 21h
        lea dx,a11
        int 21h


        ;list choose
        lea dx,a8
        int 21h


        mov ah,1
        int 21h
        mov bh,al
        sub bh,48


        cmp bh,1
        je Breakfast


        cmp bh,2
        je Lunch


        cmp bh,3
        je Dinner


        jmp Invalid


        ;Breakfast list
Breakfast:


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

```asm
        int 21h


        mov ah,2
        mov dl,0ah
        int 21h
        mov dl,0dh
        int 21h


        ;List start
        mov ah,9
        lea dx,a12
        int 21h


        mov ah,2
        mov dl,0ah
        int 21h
        mov dl,0dh
        int 21h


        mov ah,9
        lea dx,a13
        int 21h
        lea dx,a14
        int 21h
        lea dx,a15
        int 21h
        lea dx,a16
        int 21h
        lea dx,a17
        int 21h
        lea dx,a18
        int 21h
```

```asm
;condition checking
lea dx,a34
int 21h


mov ah,1
int 21h
mov bl,al
sub bl,48


cmp bl,1
je Fifty


cmp bl,2
je Fifty


cmp bl,3
je Fifty


cmp bl,4
je Fifty


cmp bl,5
je Fifty


cmp bl,6
je Fifty


jmp Invalid



Lunch:
;newline
mov ah,2
```

```asm
        mov dl,0ah
        int 21h
        mov dl,0dh
        int 21h


        mov ah,2
        mov dl,0ah
        int 21h
        mov dl,0dh
        int 21h


;List start
        mov ah,9
        lea dx,a20
        int 21h


        mov ah,2
        mov dl,0ah
        int 21h
        mov dl,0dh
        int 21h


        mov ah,9
;lea dx,a21
;int 21h
        lea dx,a22
        int 21h
        lea dx,a23
        int 21h
        lea dx,a24
        int 21h
        lea dx,a25
        int 21h
```

```asm
        lea dx,a26
        int 21h


        ;condition checking
        lea dx,a34
        int 21h
        mov ah,1
        int 21h
        mov bl,al
        sub bl,48


        cmp bl,1
        je Hundred


        cmp bl,2
        je Hundred


        cmp bl,3
        je Hundred


        cmp bl,4
        je Hundred


        cmp bl,5
        je Hundred


        cmp bl,6
        je Hundred


        jmp Invalid


        ;for exit
        mov ah,2
```

```asm
        mov dl,0ah
        int 21h
        mov dl,0dh
        int 21h


        mov ah,9
        lea dx,a38
        int 21h


        mov ah,1
        int 21h
        mov bh,al


        cmp bh,1
        jmp Exit



Dinner:
        ;newline
        mov ah,2
        mov dl,0ah
        int 21h
        mov dl,0dh
        int 21h


        mov ah,2
        mov dl,0ah
        int 21h
        mov dl,0dh
        int 21h


        ;List start
        mov ah,9
```

```asm
        lea dx,a27
        int 21h


        mov ah,9
        mov dl,0ah
        int 21h
        mov dl,0dh
        int 21h


        mov ah,9
        lea dx,a28
        int 21h
        lea dx,a29
        int 21h
        lea dx,a30
        int 21h
        lea dx,a31
        int 21h
        lea dx,a32
        int 21h
        lea dx,a33
        int 21h


        ;condition checking
        lea dx,a34
        int 21h


        mov ah,1
        int 21h
        mov bl,al
        sub bl,48


        cmp bl,1
```

```asm
        je Twohundred

        cmp bl,2
        je TwoHundred

        cmp bl,3
        je TwoHundred

        cmp bl,4
        je TwoHundred

        cmp bl,5
        je TwoHundred

        cmp bl,6
        je TwoHundred

        jmp Invalid


Fifty:
        mov bl,5
        lea dx,a35
        mov ah,9
        int 21h

        mov ah,1
        int 21h
        sub al,48

        mul bl
        aam
```

```asm
        mov cx,ax
        add ch,48
        add cl,48


        lea dx,a37
        mov ah,9
        int 21h


        mov ah,2
        mov dl,ch
        int 21h


        mov dl,cl
        int 21h


        mov dl,'0'
        int 21h


        mov dl,47
        int 21h
        mov dl,45
        int 21h



        ;for exit or main menu
        mov ah,2
        mov dl,0ah
        int 21h
        mov dl,0dh
        int 21h


        mov ah,9
        lea dx,a38
```

```
int 21h


mov ah,9
lea dx,a39
int 21h


mov ah,9
lea dx,a8
int 21h


mov ah,1
int 21h
sub al,48


cmp al,1
je Schedule


cmp al,2
je Exit


jmp Invalid


Hundred:
mov bl,10
lea dx,a35
mov ah,9
int 21h


mov ah,1
int 21h
sub al,48


mul bl
```

```asm
aam

mov cx,ax
add ch,48
add cl,48

lea dx,a37
mov ah,9
int 21h

mov ah,2
mov dl,ch
int 21h

mov dl,cl
int 21h

mov dl,'0'
int 21h

mov dl,47
int 21h
mov dl,45
int 21h

;for exit
mov ah,2
mov dl,0ah
int 21h
mov dl,0dh
int 21h

mov ah,9
```

```asm
        lea dx,a38

        int 21h


        mov ah,9

        lea dx,a39

        int 21h


        mov ah,9

        lea dx,a8

        int 21h


        mov ah,1

        int 21h

        sub al,48


        cmp al,1

        je Schedule


        cmp al,2

        je Exit


        jmp Invalid


TwoHundred:

        mov bl,20

        lea dx,a35

        mov ah,9

        int 21h


        mov ah,1

        int 21h

        sub al,48
```

```asm
mul bl
aam

mov cx,ax
add ch,48
add cl,48

lea dx,a37
mov ah,9
int 21h

mov ah,2
mov dl,ch
int 21h

mov dl,cl
int 21h

mov dl,'0'
int 21h

mov dl,47
int 21h
mov dl,45
int 21h

;for exit
mov ah,2
mov dl,0ah
int 21h
mov dl,0dh
int 21h
```

```
mov ah,9

lea dx,a38

int 21h


mov ah,9

lea dx,a39

int 21h


mov ah,9

lea dx,a8

int 21h


mov ah,1

int 21h

sub al,48


cmp al,1

je Schedule


cmp al,2

je Exit


jmp Invalid

Invalid:

;newline

mov ah,2

mov dl,0ah

int 21h

mov dl,0dh

int 21h


mov ah,2
```

```
        mov dl,0ah
        int 21h
        mov dl,0dh
        int 21h


        mov ah,9
        lea dx,a36
        int 21h
        jmp Exit
        Exit:
        mov ah,4ch
        int 21h
        main endp
end main
```

# Final Output: