MINI-PROJECT

Course: CSE360
Course Title: Computer Architecture
section: 03
Group no: 04

Title: Resturant Management in Assembly Language

Objective:
The Restaurant management system is the most efficient way to manage a restaurant as it helps the restaurant manager and staff members to take better control of the meal ordering, billing and much more. Our job is to create and execute a basic 'Restaurant Management' program using Assembly Language. In this program, different menu options are available for the customers. The given meals and their price range acording to the quantity of selected meals are given in the menu.

Tools required:
Software : EMU8086 - MICROPROCESSOR EMULATOR
Language : Assembly

Group members:

NAME	ID
Adnan Saif Dipto	2018-1-60-157
Fatima Tanjum Tuba	2018-1-60-049
Anika Hassan Mysha	2017-1-60-097

Instructor:
Dr. Md. Nawab Yousuf Ali
Professor, Department of Computer Science & Engineering

Abstract:

This study details the steps involved in planning, creating, and testing a software system for use in a restaurant, which is commonly referred to as a restaurant management system.The restaurant management system exists to facilitate communication among various teams within a restaurant while reducing the likelihood of human mistake. The restaurant management system is a complete solution that starts with accepting customer orders for various dishes, quantities of food, and billing. In this project, we proposed to build a software project that can efficiently handle and manage various activities of a restaurant and all these activities .Restaurant management system is used to manage all the entries and data files for basic user use . This Project is developed by using basic concepts of assembly language (16 bit). The menu on a Restaurant management system should be easy to configure and set up. Our main goal of this project is to create such a Restaurant management system which will save time of both customers and restaurants management team. Restaurant management system is a project in which the main menu of different foods displays first and the customer select options for selecting the food of their own choice and customers can also select the quantity of food that they want and the prices of each food product is fixed and with the number of quantity of foods the bill will be shown on the screen by which customers can easily order food from the menu and proceed to pay their exact bill without any problem.

Keywords:

Restaurant management, Food choices, Billing system, Customer, Assembly Language.

Introduction:

In our project we are using assembly language(16 bit) . Assembly language is the most basic programming language available for any processor. Every computer, no matter how simple or complex, has at its heart exactly two things: a CPU and some memory. Together, these two things are what make it possible for your computer to run programs. A computer program is nothing more than a collection of numbers stored in memory. With assembly language, a programmer works only with operations implemented directly on the physical CPU. Assembly language lacks high-level conveniences such as variables and functions, and it is not portable between various families of processors. An assembly language is a type of programming language that translates high-level languages into machine language.It is a necessary bridge between software programs and their underlying hardware platforms.

On PCs, Assembler is normally used only under MS-DOS. When running a 32-bit, protected-mode operating system, low-level programs which directly access registers or memory locations produce protection violations. All low-level access must be made through appropriate software drivers. For MS-DOS PCs, the most popular Assembly language was Microsoft Macro Assembler, or MASM.

In Assembly Language programming, One important MASM directive is .MODEL, which determines the maximum size for a program. memory is addressed as segments, up to 64 Kbytes in length. If 16-bit addressing is used (for code or data) only a single 64K segment will be accessed. The *memory model* of a program defines how different parts of that program (code and data) access memory segments. Five memory models are supported by MASM programs: Small, Medium, Compact, Large, and Huge. In the Small model, all data fits within one 64K segment and all code fits within another single 64K segment. In the Medium model, all data fits within one 64K segment but code can be larger than 64K. arger models require larger addresses, they produce bigger and slower programs than a smaller model will. In selecting a model for a program, try to estimate the maximum amount of data storage you will need. Let us say you are writing an FFT program, using 16-bit integer math and a maximum sample size of 2048 points. Since each point requires two integers and each integer is 2 bytes long, you need 8096 bytes just to store the input data. Even if you had separate arrays for input and output data, that would still be only 16,192 bytes. As a safety margin, for temporary storage, we will double this number, to 32,384 bytes, which is only half of a 64K segment. It is more difficult to estimate the size of the code.

For Loop :

Most assembly languages do not have a for loop. However, Top/ Jump keywords are used here.

JE : (JUMP if EQUAL)

JNE: (JUMP if NOT EQUAL)

JGE: (JUMP if GREATER THAN OR EQUAL)

JMP: (basic JUMP keyword)

LOOP TOP: (for assigning variables inside loop)

While :

The initialization of the loop variable has been moved to its own line before the 'while' statement. Also, the loop variable is modified on the last line of the loop body. This is a straightforward conversion from one type of loop to another type.

Simulators :

Instruction-set simulators simulate the execution of a processor at the instruction level on a host computer. The user can see changes in the various registers, memory and flags as the program is executed. The user can single-step through a

Emulator-8086 and it's keywords :

1. General register set: (16-bit) Can access higher or lower byte

   AX: AH (8-bit) , AL (8-bit)

   BX: BH (8-bit) , BL (8-bit)

   CX: CH (8-bit) , CL (8-bit)

   DX: DH (8-bit) , DL (8-bit)

2. Addressing register set (16-bit): (no 8-bit access)
   IP = Instruction Pointer
   SP = Stack Pointer
   BP = Base Pointer
   SI = Source Index
   DI = Destination Index

RESTAURANT MANAGEMENT PROGRAM:

Using this emulator software, we have created and emulated a very basic restaurant program. We've taken different menu items, price and quantity in the .DATA segment and stored them in a particular memory location.

The .MAIN PROC from CODE segment will take all the data from .DATA and store them in the primary accumulator (AX).

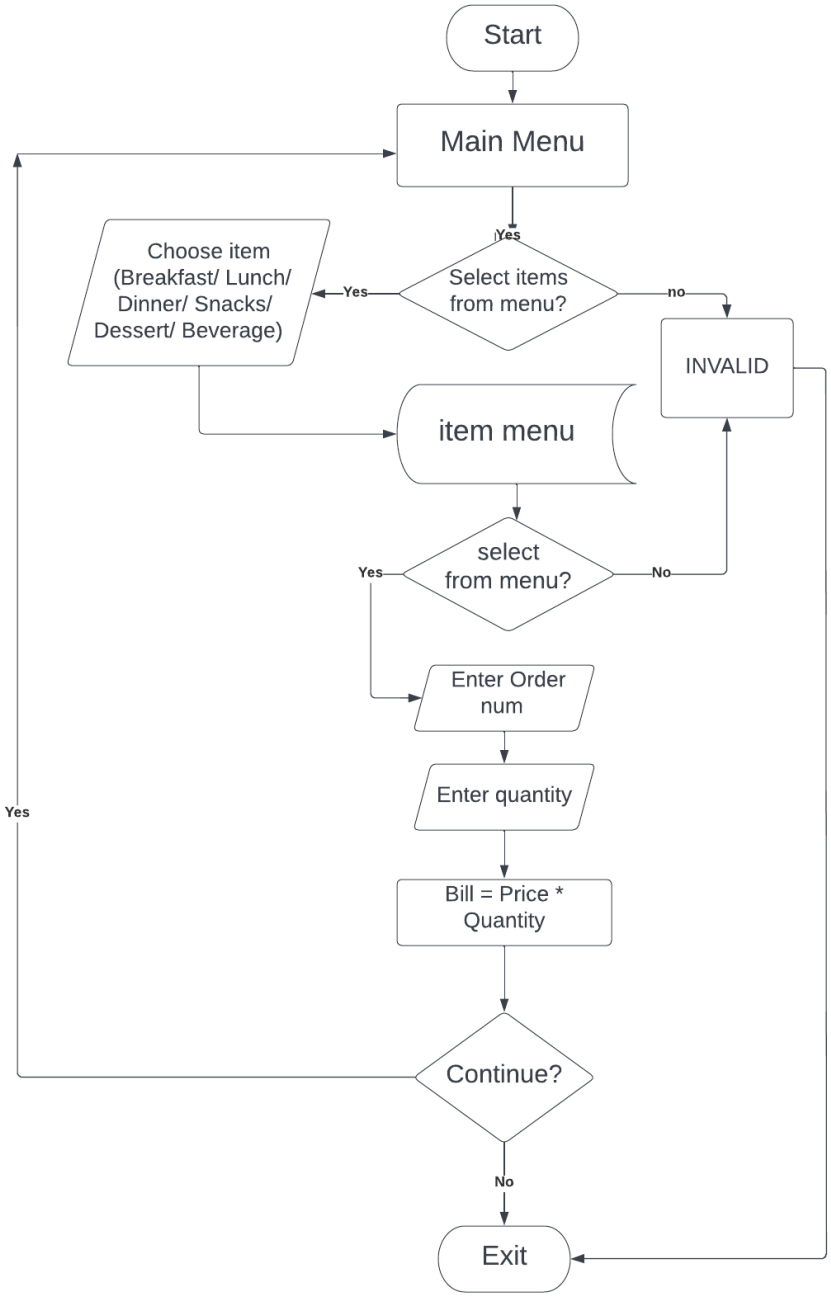
```
.CODE
MAIN PROC
MOV AX,@DATA  ;storing all data items in an accumulator
MOV DS,AX ;keeping them as default segment
TOP: ;keyword
```

We also used ASCII codes from the ascii chart as Assembly language doesn't directly take string values unlike C/C++ or Python.

Ascii codes from table that we used:

09 = TAB (for making horizontal table)
10 = play
13 = pause
48 = "0" (initially)

Flowchart:



Data segment:

(main menu)

```
.DATA
M1 DB 10,13,10,13,'                    ****Welcome to Our Restaurants****$',10,13
M2 DB 10,13,10,13,'Enter your Choise $'

M3 DB 10,13,'   **                  1.Breakfast Menue          **$'
M4 DB 10,13,'   **                  2.Lunce Menue              **$'
M$5 DB 10,13,'   **                  3.Dinner Menue             **$'
M5 DB 10,13,'   **                  4.Snacks                   **$'
M6 DB 10,13,'   **                  5.Sweat Meat               **$'
M7 DB 10,13,'   **                  6.Drinks                   **$'
```

(Breakfast)

```
M8 DB 10,13,10,13,'***Choise your food from the menu***$'

;BREAKFAST
M9 DB 10,13,'   **            1.Tanduri Roti          10/-                    **$' ;breakfast
M10 DB 10,13,'   **            2.Nan                   10/-                    **$'
M11 DB 10,13,'   **            3.Parata                10/-                    **$'
M12 DB 10,13,'   **            4.Dal                   10/-                    **$'
M13 DB 10,13,'   **            5.Mixed Vegetables      20/-                    **$'
M14 DB 10,13,'   **            6.Halwa                 20/-                    **$'
M15 DB 10,13,'   **            7.Luchi                 10/-                    **$'
M16 DB 10,13,'   **            8.Fried Egg             20/-                    **$'
M17 DB 10,13,'   **            9.Goats Feet            60/-                    **$'
```

(Lunch)

```
;LUNCH

M25 DB 10,13,'   **  1.Kachchi Birani(Kabab+Egg)                  90/-
M26 DB 10,13,'   **  2.Chicken Birani(Kabab+Egg)                  90/-
M27 DB 10,13,'   **  3.Plain Polao                                30/-
M28 DB 10,13,'   **  4.Chicken Bhuna Khichuri(with Kabab+Egg)     90/-
M29 DB 10,13,'   **  5.Mutton Bhuna Khichuri(with Kabab+Egg)      90/-
M30 DB 10,13,'   **  6.White Rice                                 10/-
M31 DB 10,13,'   **  7.Dried Fish                                 30/-
M32 DB 10,13,'   **  8.Lobstar Big/Small                          30/-
M33 DB 10,13,'   **  9.Koi Fish                                   30/-
```

(Dinner)

```
;DINNER

M18 DB 10,13,'   **  1.Chicken Roast              60/-
M19 DB 10,13,'   **  2.Chicken Bhuna Khichuri     80/-
M20 DB 10,13,'   **  3.Mutton Bhuna Khichuri      80/-
M21 DB 10,13,'   **  4.Chicken Liver/Kolija       40/-
M22 DB 10,13,'   **  5.Beef Kolija                50/-
M23 DB 10,13,'   **  6.Chicken kebab (x2)         70/-
M34 DB 10,13,'   **  7.Hilsha Fish                60/-
M35 DB 10,13,'   **  8.Rui Fish                   60/-
M36 DB 10,13,'   **  9.Molay/Kaski Fish           60/-
```

(Snacks and dessert)

```
;SNACKS

M41 DB 10,13,'   **  1.Daal puri         8/-                    **$'
M42 DB 10,13,'   **  2.Shami/Jali Kabab 80/-                    **$'
M43 DB 10,13,'   **  3.Singara          5/-                    **$'
M44 DB 10,13,'   **  4.Fried peanut     5/-                    **$'


;DESSERT
M45 DB 10,13,'   **  1.Faluda 50/-                             **$'
M46 DB 10,13,'   **  2.Puding 50/-                             **$'
M47 DB 10,13,'   **  3.Firni  50/-                             **$'
M48 DB 10,13,'   **  4.Curd   50/-                             **$'
```

(Beverage)

```
;BEVERAGE
M49 DB 10,13,'   **  1.Shoft Drinks 8/-                         **$'
M50 DB 10,13,'   **  2.Laschi       6/-                         **$'
M51 DB 10,13,'   **  3.Borhani      9/-                         **$'
M52 DB 10,13,'   **  4.Labang       9/-                         **$'
M53 DB 10,13,'   **  5.Coffee       7/-                         **$'
M54 DB 10,13,'   **  6.Tea          5/-                         **$'
```

(Invalid item)

```
;INVALID (IF ITEM IS NOT ON MENU)
M55 DB 10,13,10,13,'***&&INVALID ENTRY&&***$'
M56 DB 10,13,'       ***&&Try Again&&***$'
```

(Billing)

```
;CHOOSE FROM MENU
M57 DB 10,13,10,13,'Enter your order: $' ;ALL
M58 DB 10,13,'Quantity: $'              ;BR/L/DN/S/DSRT/BEV
M59 DB 10,13,'Total Price: $'       ; ITEM * QUANTITY
```

For printing all the menu items from DATA:

```
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


CMP BH,4
JE SNACKS


CMP BH,5
JE DESSERT

CMP BH,6
JE BEVERAGE


JMP INVALID
```

How the program works:

1. Customers will enter the restaurant and choose their desired items on the menu.
2. The items are stored as string in ‘data’ segment will will be called in the main proc.
3. Loop will start, and the customer will choose any item from the menu. If the item is outside the main menu, it will show ‘invalid’ on screen and the program will stop.
4. After choosing a particular option from the main menu, the program will allocate the items from their memory location and display them.
5. Each meal comes with a price and quantity. Customer (user) will manually choose the amount of meal (quantity). Finally the total bill will be calculated (Bill = Price * Quantity) which will be calculated through the operator in Assembly language (ADD/SUB/MUL/DIV)

6. If the customer wants to continue , they will have to choose the option given in the output. And it will take them back to the main menu. If not, the program will stop the loop and thus exit.
