PROJECT REPORT

RESTURANT BILLING SYSTEM

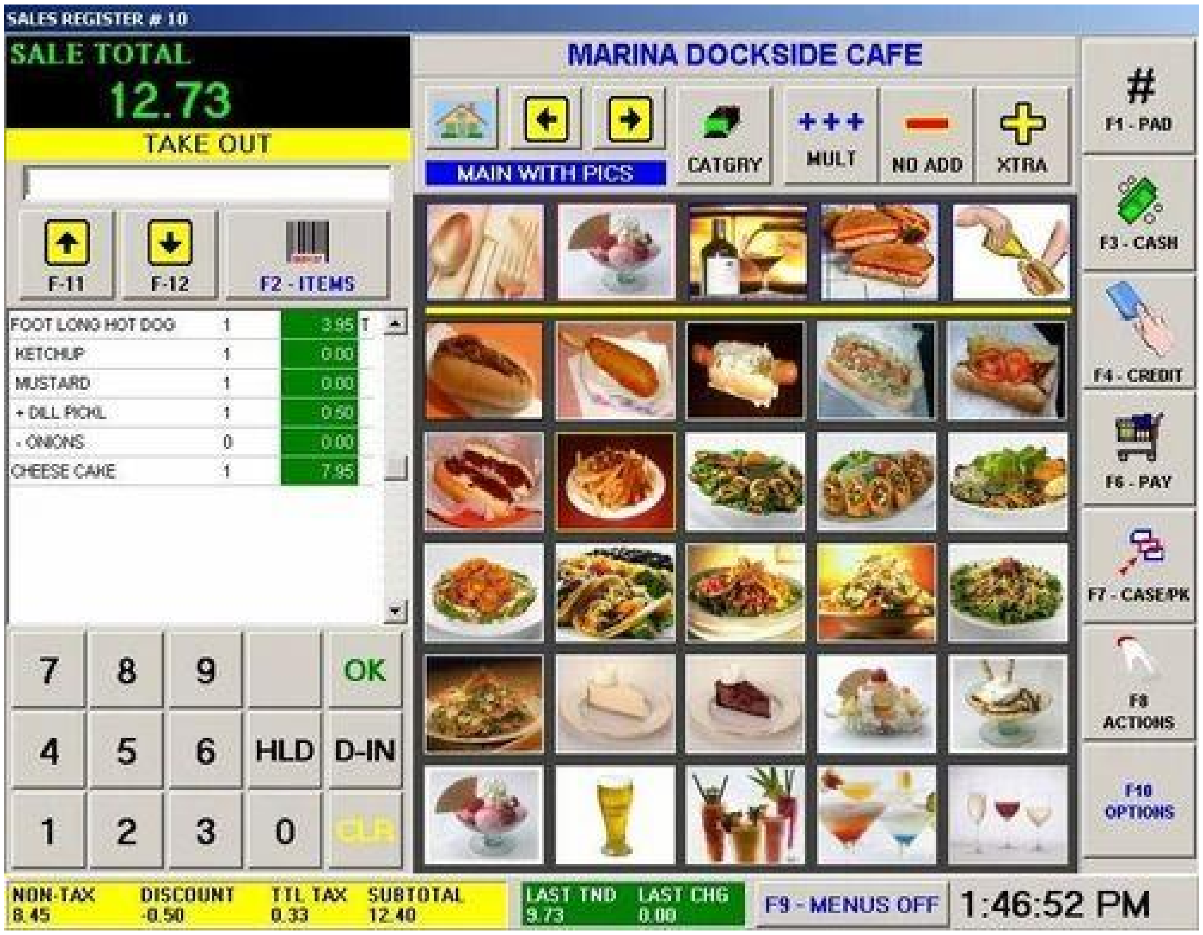
University of Engineering and Technology
Department of Computer Engineering

Submitted by:

- Qurat ul ain (2021-CE-02)
- Haseeba Yasin (2021-CE-54)
- Rabia Khanum (2021-CE-56)
- Sumbal Ijaz (2021-CE-05)

Submitted to:

Raja Muzammal Munir



SALE TOTAL
12.73

FOOT LONG HOT DOG	1	3.95	T
KETCHUP	1	0.00	
MUSTARD	1	0.00	
+ DILL PICKL	1	0.50	
- ONIONS	0	0.00	
CHEESE CAKE	1	7.95	

NON-TAX	DISCOUNT	TTL TAX	SUBTOTAL	LAST TND	LAST CHG
8.45	-0.50	0.33	12.40	9.73	0.00

ABSTRACT

The main goal of this project is to develop a billing system for a restaurant. This application is designed to administer its users and customers. RBS is a billing system, made for the effective utilization of modern technology in the organization. It is an automated software that can handle a lot of information about the restaurant’s menu, price, order history, reservation data. It has the capability to process bills and gather information of billing history. It is designed for the sole purpose of efficiency, speed and accuracy

Table of Contents

INTRODUCTION

Restaurant Billing System is a computer based billing system with user friendly interface which automatically manages the billing process of the customer very easily taking only a short period of time. The system can large amount of data and also generates bill for the customer. Billing history, reservation information and staff information can also be obtained with the use of RBS. It is an automated desktop based software which has a simple design and very easy to use also. This project's main focus is on proper management of information regarding the staffs, billing and reservation records. It is also specialized in automatically processing the customer bills and discounts.

The proposed system either does not require paper work or very few paper works are required. All the data is fetched into the computer immediately and various bills can be generated through computers. Since all the data is kept in a database, no data of the organization can be destroyed. Moreover, works become very easy because there is no need to keep data on papers.

OBJECTIVE:

The motive and objective to digitalize the restaurant menu and billing system is to save time, resources, reduce human error to minimum and making the menu system more environmentally friendly

METHODOLOGY:

1. For entering the data of one user, a function is used. Its purpose is to show menu and ask for order. After selecting menu item, it will ask for quantity and then after pressing ok it will show the total cost.
2. After showing the cost it will show another option which is re-order. For re-order we have to press 1 or press any key to exit.
3. There are total 3 items. If the user enters 4 then the system will say if you want to add this item press 1 and press 2 to get back to menu. After pressing 1 it will ask for food name and price and then it will return to the menu and also show the added item.

TOOL /SOFTWARE:

Another advantage of 8086 instruction set is that it is much smaller, and thus easier to learn. Emu8086 has a much easier syntax than any of the major assemblers, but will still generate a program that can be executed on any computer that runs 8086 machine code; a great combination for beginners!

FEASIBILITY:

The feasibility study is carried out to test whether the proposed system is worth being implemented. Feasibility study is a test of system proposed regarding its work ability, its impact on the organization to meet user needs and effective use of resources. It is usually carried out by a small number of people who are familiar with the information system techniques, understand the part of the business or organization that will be involved or affected by the project and are skilled in the system analysis and design process.

The key consideration involved in the feasibility study are:

- Technical feasibility
- Economic feasibility
- Operational feasibility
- Schedule feasibility

CODE:

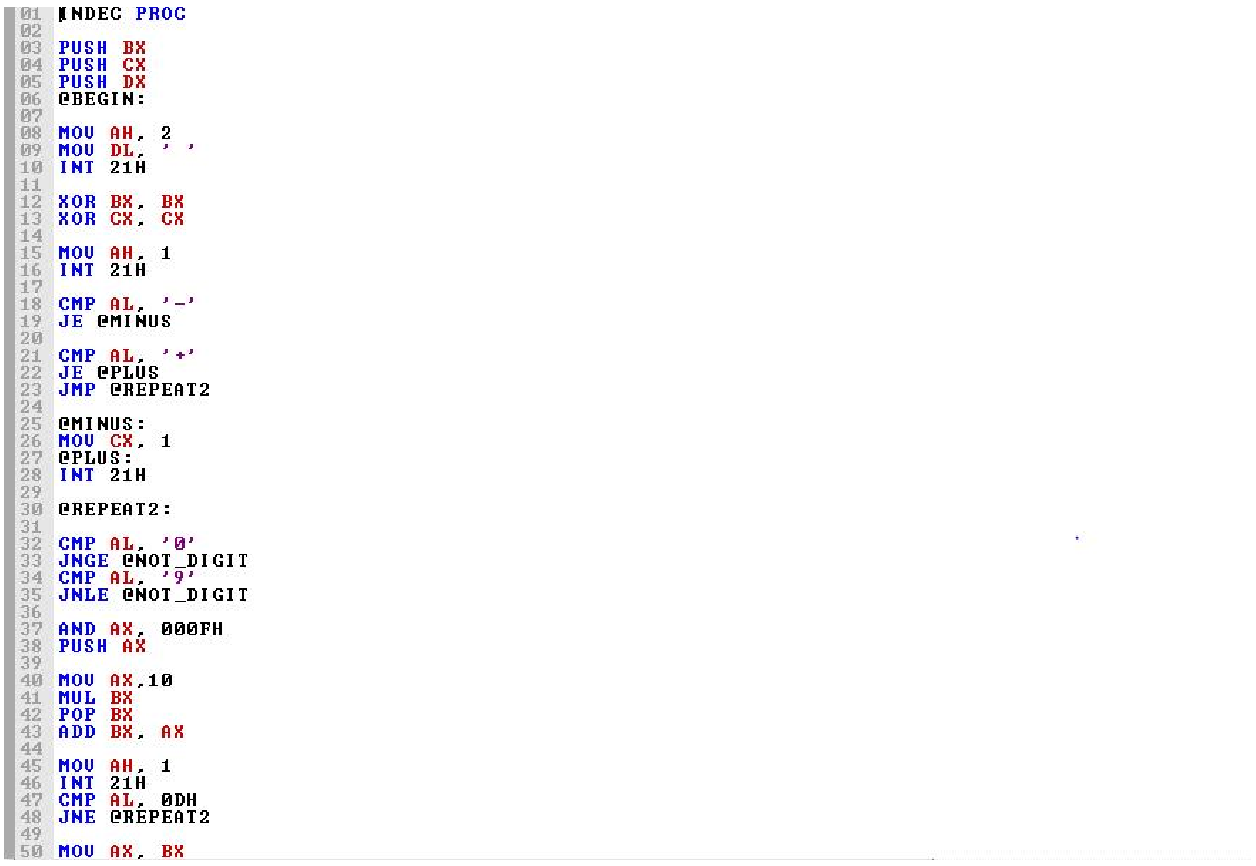
INDEC.ASM:

```
INDEC PROC


PUSH BX

PUSH CX

PUSH DX

@BEGIN:
```

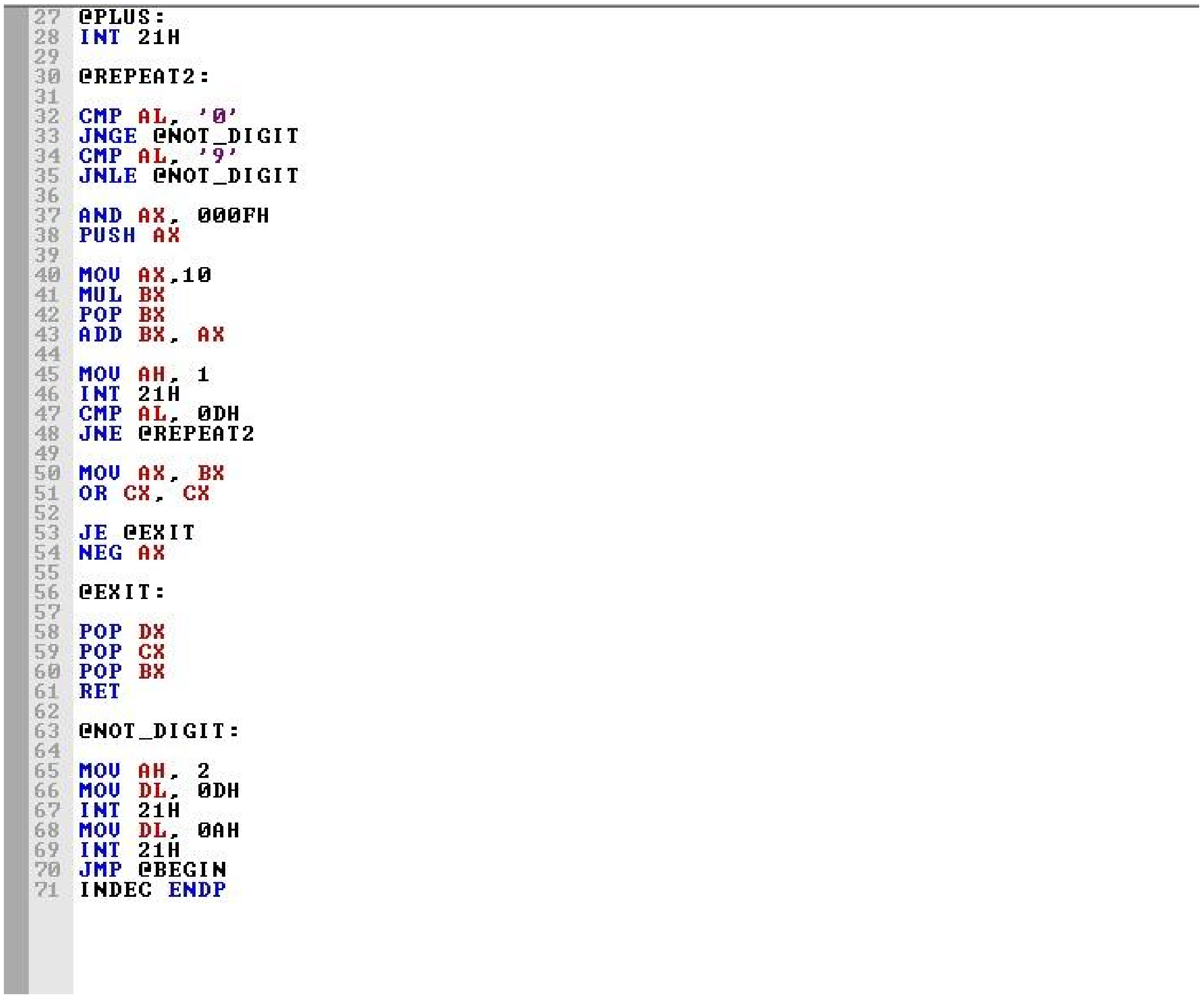
```
MOV AH, 2
MOV DL, ' '
INT 21H

XOR BX, BX
XOR CX, CX

MOV AH, 1
INT 21H

CMP AL, '-'
JE @MINUS

CMP AL, '+'
JE @PLUS
JMP @REPEAT2

@MINUS:
MOV CX, 1
@PLUS:
INT 21H

@REPEAT2:
```

```
CMP AL, '0'
JNGE @NOT_DIGIT
CMP AL, '9'
JNLE @NOT_DIGIT

AND AX, 000FH
PUSH AX

MOV AX,10
MUL BX
POP BX
ADD BX, AX

MOV AH, 1
INT 21H
CMP AL, 0DH
JNE @REPEAT2

MOV AX, BX
OR CX, CX

JE @EXIT
```

```
NEG AX

@EXIT:

POP DX
POP CX
POP BX
RET

@NOT_DIGIT:

MOV AH, 2
MOV DL, 0DH
INT 21H
MOV DL, 0AH
INT 21H
JMP @BEGIN
INDEC ENDP
```



```
INDEC PROC

PUSH BX
PUSH CX
PUSH DX
@BEGIN:

MOV AH, 2
MOV DL, ' '
INT 21H

XOR BX, BX
XOR CX, CX

MOV AH, 1
INT 21H

CMP AL, '-'
JE @MINUS

CMP AL, '+'
JE @PLUS
JMP @REPEAT2

@MINUS:
MOV CX, 1
@PLUS:
INT 21H

@REPEAT2:

CMP AL, '0'
JNGE @NOT_DIGIT
CMP AL, '9'
JNLE @NOT_DIGIT

AND AX, 000FH
PUSH AX

MOV AX,10
MUL BX
POP BX
ADD BX, AX

MOV AH, 1
INT 21H
CMP AL, 0DH
JNE @REPEAT2

MOV AX, BX
```



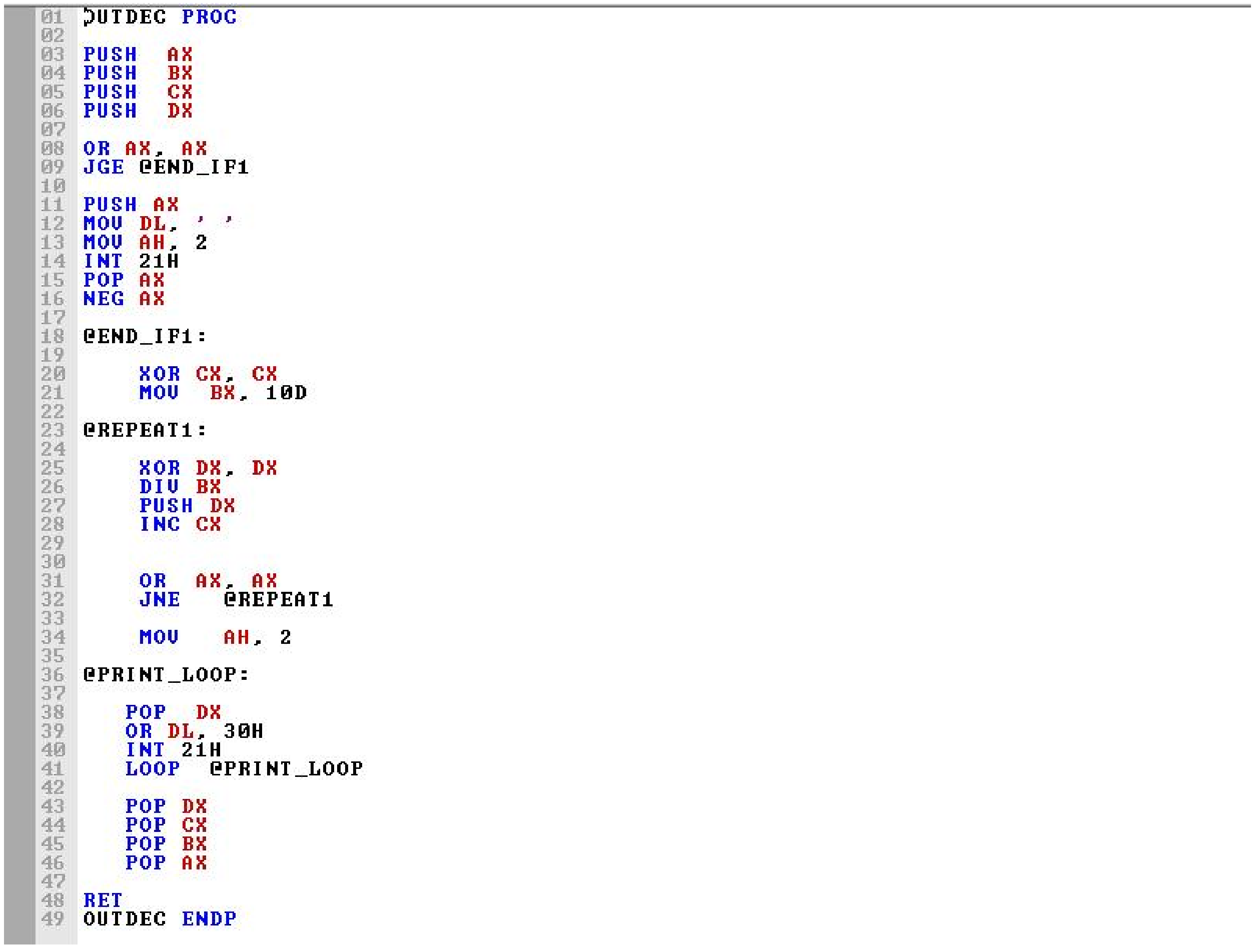
```
@PLUS:
INT 21H

@REPEAT2:

CMP AL, '0'
JNGE @NOT_DIGIT
CMP AL, '9'
JNLE @NOT_DIGIT

AND AX, 000FH
PUSH AX

MOV AX,10
MUL BX
POP BX
ADD BX, AX

MOV AH, 1
INT 21H
CMP AL, 0DH
JNE @REPEAT2

MOV AX, BX
OR CX, CX

JE @EXIT
NEG AX

@EXIT:

POP DX
POP CX
POP BX
RET

@NOT_DIGIT:

MOV AH, 2
MOV DL, 0DH
INT 21H
MOV DL, 0AH
INT 21H
JMP @BEGIN
INDEC ENDP
```

OUTDEC.ASM:

```
OUTDEC PROC


PUSH  AX

PUSH  BX

PUSH  CX

PUSH  DX
```

```
OR AX, AX

JGE @END_IF1


PUSH AX

MOV DL, ' '

MOV AH, 2

INT 21H

POP AX

NEG AX


@END_IF1:


   XOR CX, CX

   MOV  BX, 10D


@REPEAT1:


   XOR DX, DX

   DIV BX

   PUSH DX

   INC CX
```

```
    OR  AX, AX
    JNE   @REPEAT1

    MOV   AH, 2

@PRINT_LOOP:

   POP  DX
   OR DL, 30H
   INT 21H
   LOOP  @PRINT_LOOP

   POP DX
   POP CX
   POP BX
   POP AX

RET
OUTDEC ENDP
```



```
OUTDEC PROC

PUSH  AX
PUSH  BX
PUSH  CX
PUSH  DX

OR AX, AX
JGE @END_IF1

PUSH AX
MOV DL, ' '
MOV AH, 2
INT 21H
POP AX
NEG AX

@END_IF1:

    XOR CX, CX
    MOV  BX, 10D

@REPEAT1:

    XOR DX, DX
    DIV BX
    PUSH DX
    INC CX


    OR  AX, AX
    JNE   @REPEAT1

    MOV   AH, 2

@PRINT_LOOP:

   POP  DX
   OR DL, 30H
   INT 21H
   LOOP  @PRINT_LOOP

   POP DX
   POP CX
   POP BX
   POP AX

RET
OUTDEC ENDP
```

RESTURANT.ASM:

```
.model small

.stack 100h

.data


   m0 dw "        !!!!!!Welcome in our project!!!!!!$"

   m1 dw 10,13,10,13, "Which menu do you want ??please select:$"

   m2 dw 10,13,10,13, "1.Rice 100/- 2.Vegetable 50/- 3.Soup 20/- $"

   m3 dw 10,13,10,13, "Select the menu number:$"
```

```
m8 dw 10,13,10,13, "SORRY!!!There is no more than 3 item,if u
want,u can add one$"
m9 dw 10,13,10,13, "Enter Food name:$"
m10 dw 10,13,10,13,"          Price:$"
m4 dw 10,13,10,13, "To add press 1 or press 2 to get back menu :$"
m5 dw 10,13,10,13, "Enter quantity:$"
m6 dw 10,13,10,13, "Total price: $"
m7 dw 10,13,10,13, "      **THANK YOU**$"
m11 dw "4.$"
m12 dw "/-$"
m13 dw 10,13,10,13, " Re-odrer : Press <1>$",
m14 dw 10,13,10,13, " Exit : Press Any key$"
q dw 0
r dw 0
v db 0
s dw 0
rprice dw 100
vprice dw 50
sprice dw 20
nprice dw 0
```

```
   var1 db 100 dup('$')

.code
   main proc

      mov ax,@data
      mov ds,ax

      mov ah,9
      Lea dx,m0
      int 21h

      start:
      cmp v,0
      jg start1

      mov ah,9
      Lea dx,m1
      int 21h

      menu:
```

```
mov ah,9
Lea dx,m2
int 21h

mov ah,9
Lea dx,m3
int 21h

mov ah,1
int 21h

cmp al,31h
je rice_
cmp al,32h
je veg_
cmp al,33h
je soup_
```

```
menuadd:
inc v

mov ah,9
Lea dx,m8
int 21h

mov ah,9
Lea dx,m4
int 21h

mov ah,1
int 21h
cmp al,32h
je menu

mov ah,9
Lea dx,m9
int 21h

mov si,offset var1
```

```
l1:
mov ah,1
int 21h
cmp al,13
je print
mov [si],al
inc si
jmp l1

print:

call price

start1:

mov ah,9
Lea dx,m2
int 21h

mov ah,9
```

```
Lea dx,m11
int 21h

mov dx,offset var1
mov ah,9
int 21h

mov ah,2
mov dl,' '
int 21h

xor ax,ax
mov ax,nprice
call outdec

mov ah,9
Lea dx,m12
int 21h
```

```
mov ah,9

Lea dx,m3

int 21h


mov ah,1

int 21h


cmp al,31h

je rice_

cmp al,32h

je veg_

cmp al,33h

je soup_


newmenu_:


   mov ah,9

   Lea dx,m5

   int 21h


   xor ax,ax
   call indec


   mul nprice


   mov bx,ax


   jmp totalprice



veg_:


   mov ah,9
   Lea dx,m5
   int 21h


   xor ax,ax


   call indec


   mul vprice
```

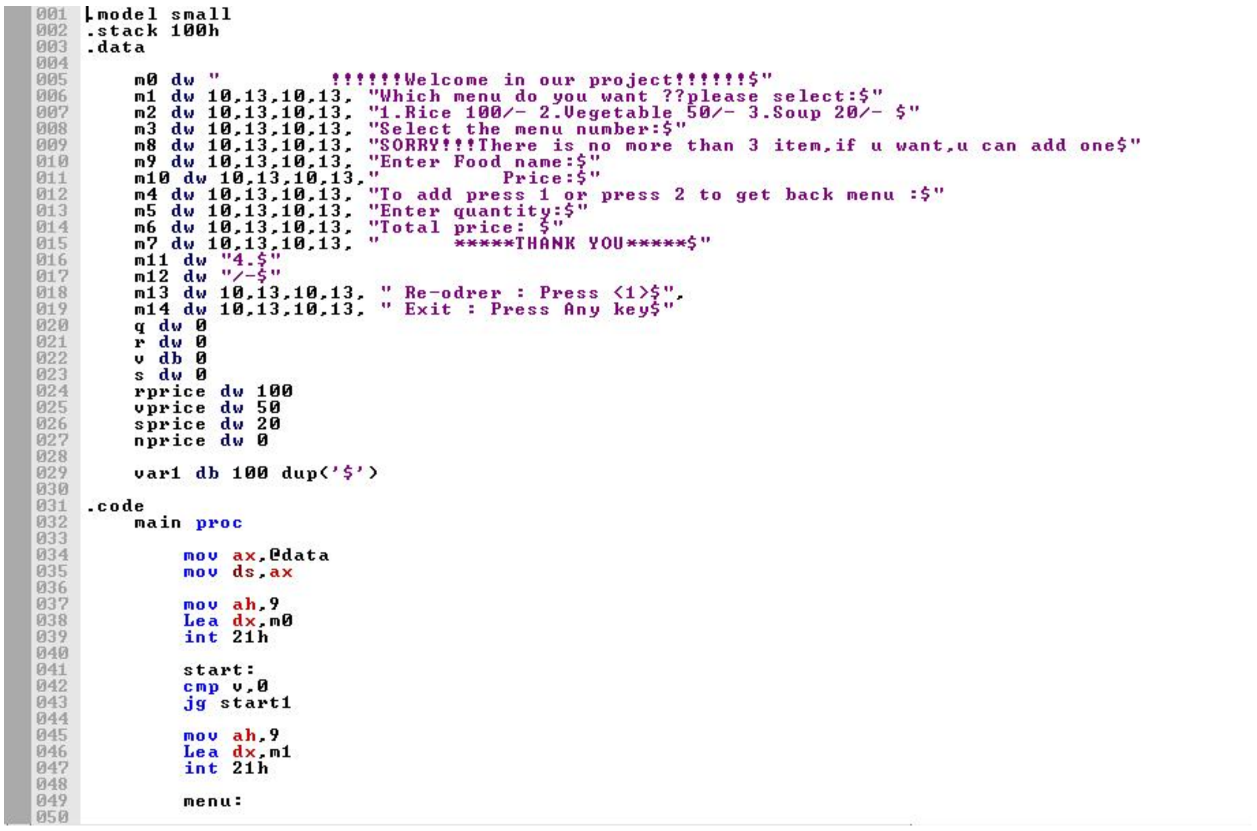
```
    mov bx,ax

    jmp totalprice

rice_:

    mov ah,9
    Lea dx,m5
    int 21h

    xor ax,ax

    call indec

    mul rprice

    mov bx,ax

    jmp totalprice
```

```
soup_:
   mov ah,9
   Lea dx,m5
   int 21h

   xor ax,ax

   call indec

   mul sprice

   mov bx,ax

   jmp totalprice

price:

   mov ah,9
   Lea dx,m10
   int 21h
```

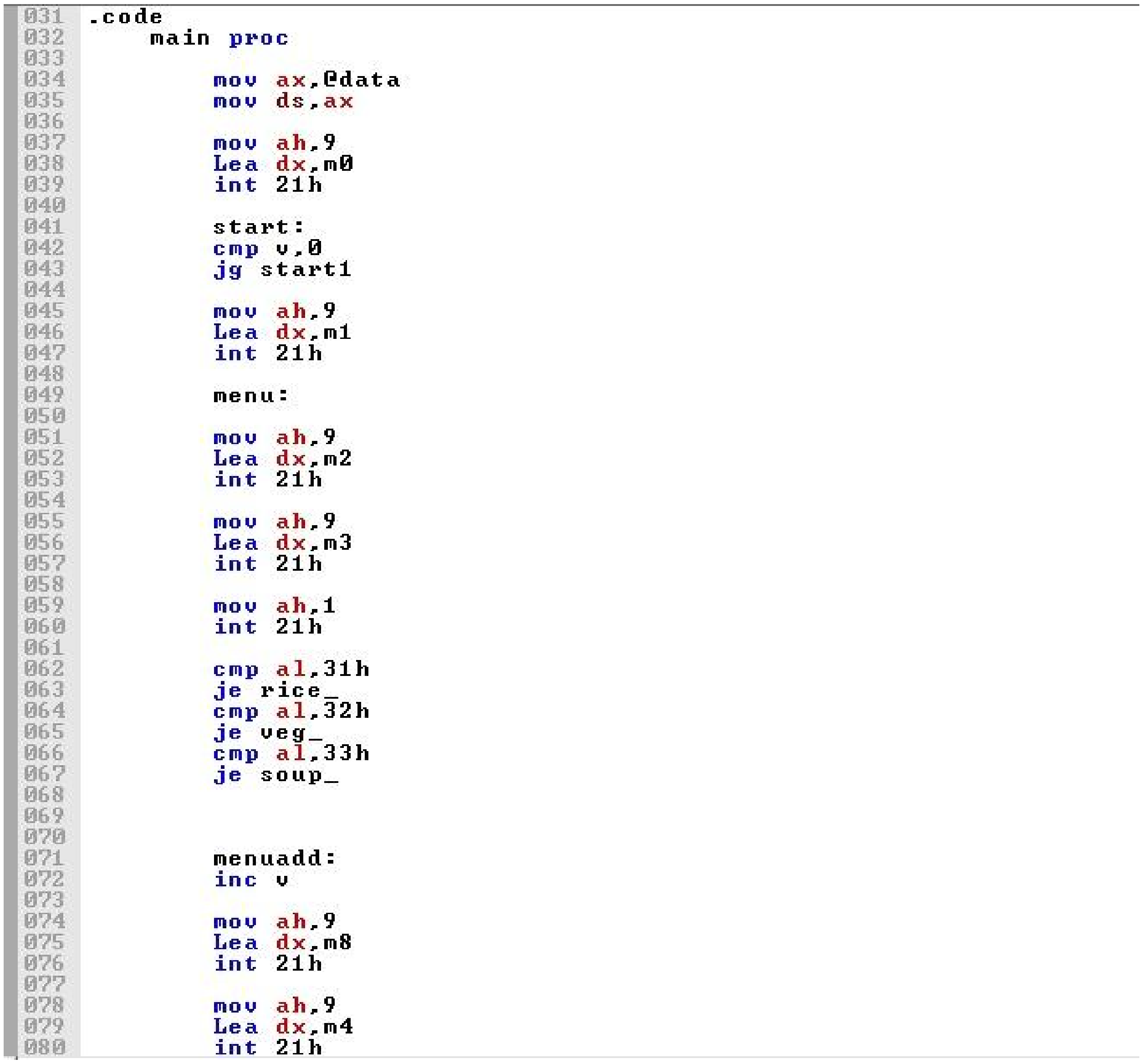
```
mov ax,0
 mov bx,0
  mov cx,0
   mov dx,0

input:
    and ax,000Fh
    push ax
    mov ax,10
    mul bx
    mov bx,ax
    pop ax
    add bx,ax

    mov ah,1
    int 21h

    cmp al,0Dh
    jne input

add nprice,bx
```

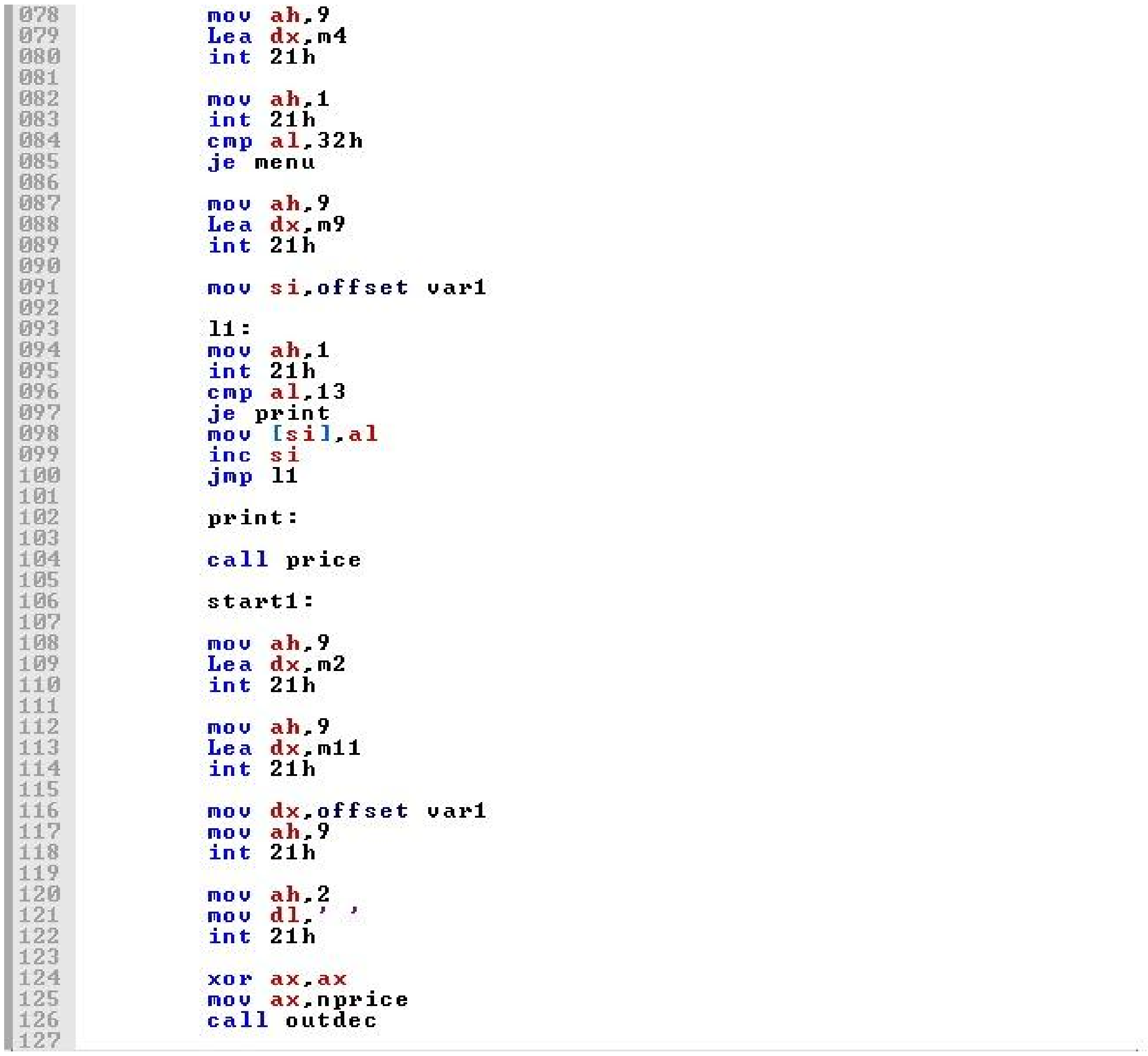
```
    ret



totalprice:

    mov ah,9
    Lea dx,m6
    int 21h

    xor ax,ax


    mov ax,bx
    call outdec

    mov ah,9
    Lea dx,m13
    int 21h

    mov ah,9
```

```
        Lea dx,m14
        int 21h

        mov ah,1
        int 21h

        cmp al,31h
        je start

        mov ah,9
        Lea dx,m7
        int 21h



    mov ah,4ch
    int 21h

    main endp
include indec.asm
include outdec.asm
end main
```

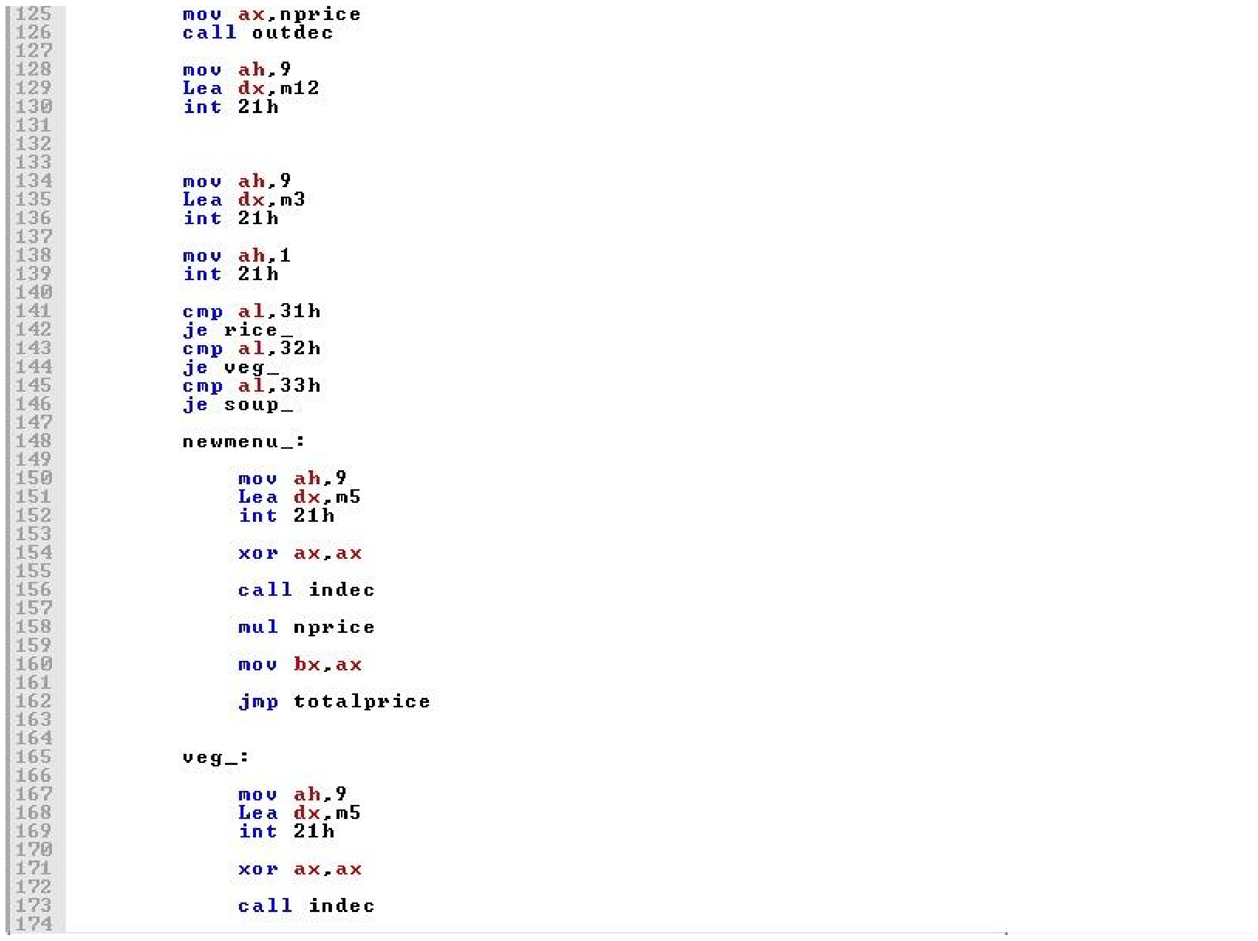


```
.model small
.stack 100h
.data

    m0 dw "          !!!!!!Welcome in our project!!!!!!$"
    m1 dw 10,13,10,13, "Which menu do you want ??please select:$"
    m2 dw 10,13,10,13, "1.Rice 100/- 2.Vegetable 50/- 3.Soup 20/- $"
    m3 dw 10,13,10,13, "Select the menu number:$"
    m8 dw 10,13,10,13, "SORRY!!!There is no more than 3 item,if u want,u can add one$"
    m9 dw 10,13,10,13, "Enter Food name:$"
    m10 dw 10,13,10,13,"           Price:$"
    m4 dw 10,13,10,13, "To add press 1 or press 2 to get back menu :$"
    m5 dw 10,13,10,13, "Enter quantity:$"
    m6 dw 10,13,10,13, "Total price: $"
    m7 dw 10,13,10,13, "       *****THANK YOU*****$"
    m11 dw "4.$"
    m12 dw "/-$"
    m13 dw 10,13,10,13, " Re-odrer : Press <1>$",
    m14 dw 10,13,10,13, " Exit : Press Any key$"
    q dw 0
    r dw 0
    v db 0
    s dw 0
    rprice dw 100
    vprice dw 50
    sprice dw 20
    nprice dw 0

    var1 db 100 dup('$')

.code
    main proc

        mov ax,@data
        mov ds,ax

        mov ah,9
        Lea dx,m0
        int 21h

        start:
        cmp v,0
        jg start1

        mov ah,9
        Lea dx,m1
        int 21h

        menu:
```

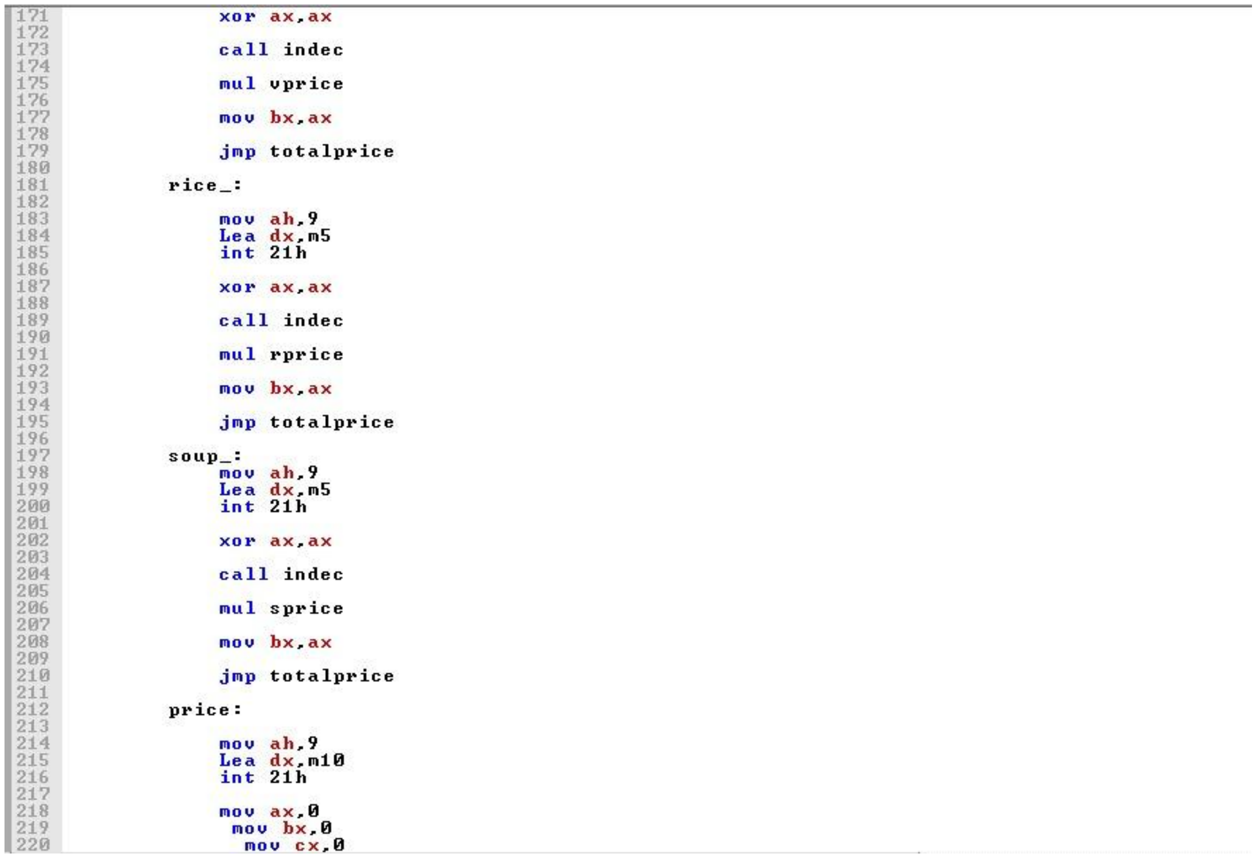

```
.code
     main proc

          mov ax,@data
          mov ds,ax

          mov ah,9
          Lea dx,m0
          int 21h

          start:
          cmp v,0
          jg start1

          mov ah,9
          Lea dx,m1
          int 21h

          menu:

          mov ah,9
          Lea dx,m2
          int 21h

          mov ah,9
          Lea dx,m3
          int 21h

          mov ah,1
          int 21h

          cmp al,31h
          je rice_
          cmp al,32h
          je veg_
          cmp al,33h
          je soup_



          menuadd:
          inc v

          mov ah,9
          Lea dx,m8
          int 21h

          mov ah,9
          Lea dx,m4
          int 21h
```



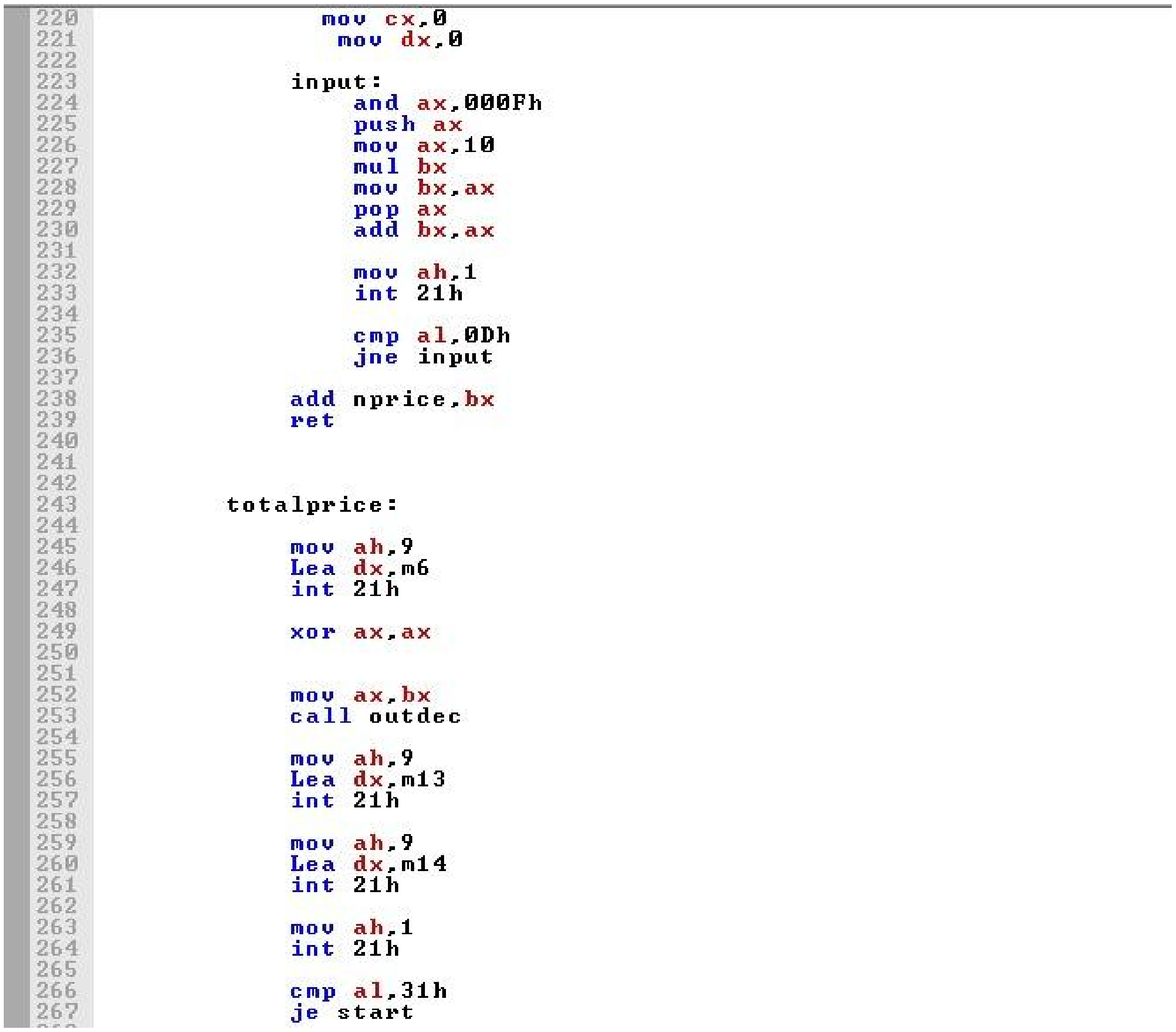
```
mov ah,9
Lea dx,m4
int 21h

mov ah,1
int 21h
cmp al,32h
je menu

mov ah,9
Lea dx,m9
int 21h

mov si,offset var1

l1:
mov ah,1
int 21h
cmp al,13
je print
mov [si],al
inc si
jmp l1

print:

call price

start1:

mov ah,9
Lea dx,m2
int 21h

mov ah,9
Lea dx,m11
int 21h

mov dx,offset var1
mov ah,9
int 21h

mov ah,2
mov dl,' '
int 21h

xor ax,ax
mov ax,nprice
call outdec
```



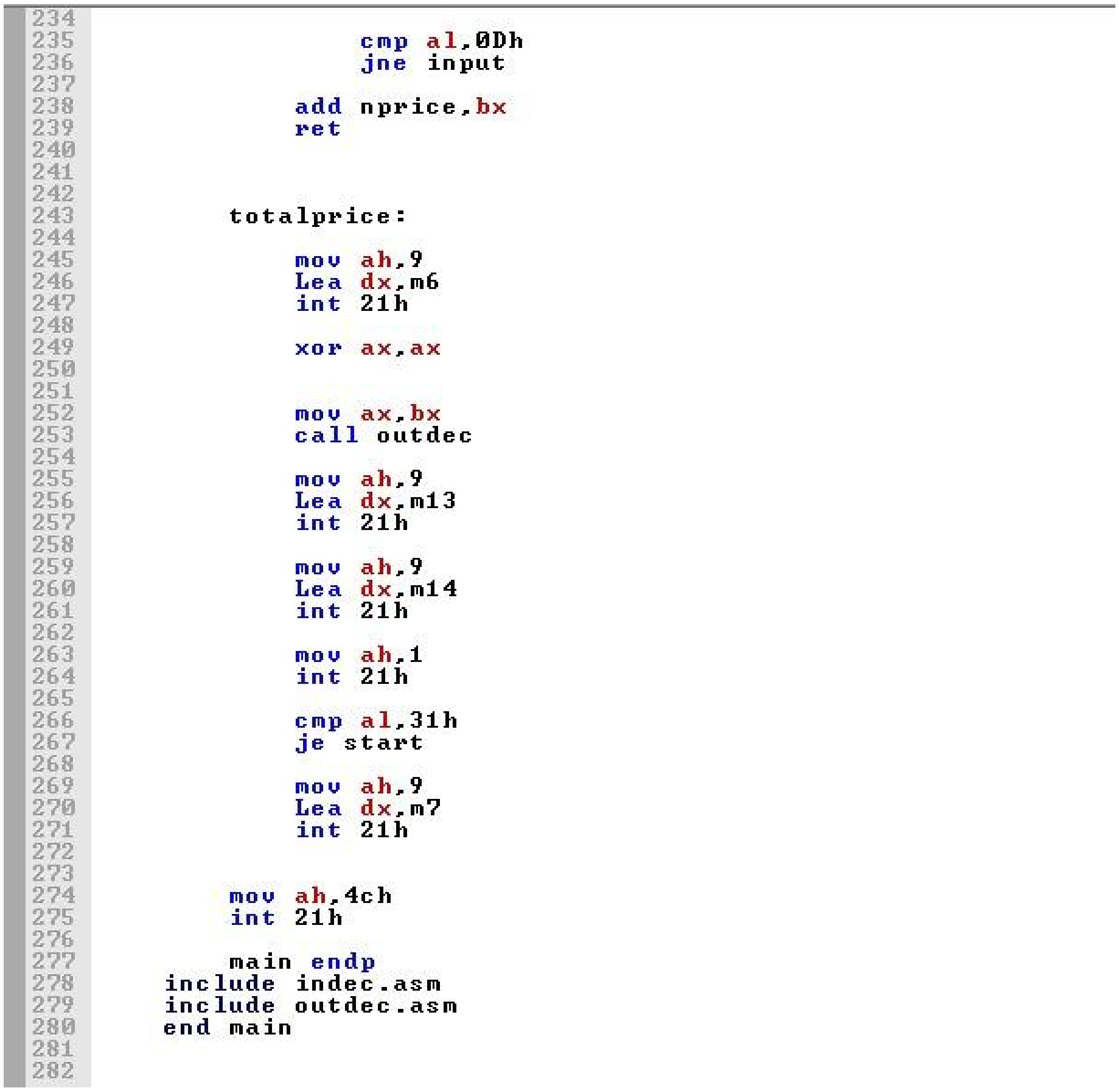
```
        mov ax,nprice
        call outdec

        mov ah,9
        Lea dx,m12
        int 21h



        mov ah,9
        Lea dx,m3
        int 21h

        mov ah,1
        int 21h

        cmp al,31h
        je rice_
        cmp al,32h
        je veg_
        cmp al,33h
        je soup_

        newmenu_:

            mov ah,9
            Lea dx,m5
            int 21h

            xor ax,ax

            call indec

            mul nprice

            mov bx,ax

            jmp totalprice


        veg_:

            mov ah,9
            Lea dx,m5
            int 21h

            xor ax,ax

            call indec

```



```
            xor ax,ax

            call indec

            mul vprice

            mov bx,ax

            jmp totalprice

        rice_:

            mov ah,9
            Lea dx,m5
            int 21h

            xor ax,ax

            call indec

            mul rprice

            mov bx,ax

            jmp totalprice

        soup_:
            mov ah,9
            Lea dx,m5
            int 21h

            xor ax,ax

            call indec

            mul sprice

            mov bx,ax

            jmp totalprice

        price:

            mov ah,9
            Lea dx,m10
            int 21h

            mov ax,0
             mov bx,0
              mov cx,0
```

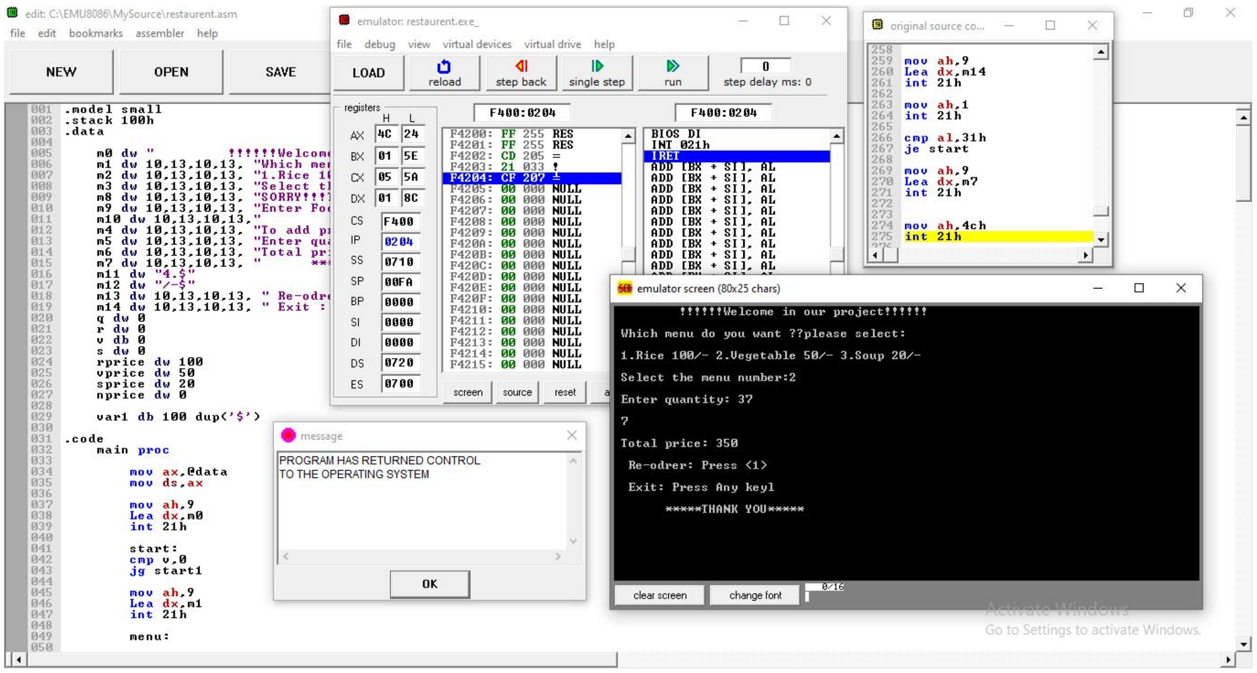


```
                    mov cx,0
                     mov dx,0

                input:
                    and ax,000Fh
                    push ax
                    mov ax,10
                    mul bx
                    mov bx,ax
                    pop ax
                    add bx,ax

                    mov ah,1
                    int 21h

                    cmp al,0Dh
                    jne input

                add nprice,bx
                ret


        totalprice:

                mov ah,9
                Lea dx,m6
                int 21h

                xor ax,ax


                mov ax,bx
                call outdec

                mov ah,9
                Lea dx,m13
                int 21h

                mov ah,9
                Lea dx,m14
                int 21h

                mov ah,1
                int 21h

                cmp al,31h
                je start
```



```
                    cmp al,0Dh
                    jne input

                add nprice,bx
                ret



            totalprice:

                mov ah,9
                Lea dx,m6
                int 21h

                xor ax,ax


                mov ax,bx
                call outdec

                mov ah,9
                Lea dx,m13
                int 21h

                mov ah,9
                Lea dx,m14
                int 21h

                mov ah,1
                int 21h

                cmp al,31h
                je start

                mov ah,9
                Lea dx,m7
                int 21h


            mov ah,4ch
            int 21h

            main endp
        include indec.asm
        include outdec.asm
        end main
```

FINAL OUTPUT:


edit: C:\EMU8086\MySource\restaurent.asm

```
.model small
.stack 100h
.data

    m0 dw "          !!!!!!Welcome
    m1 dw 10,13,10,13, "Which men
    m2 dw 10,13,10,13, "1.Rice 10
    m3 dw 10,13,10,13, "Select th
    m8 dw 10,13,10,13, "SORRY!!!!
    m9 dw 10,13,10,13, "Enter Foo
    m10 dw 10,13,10,13,"
    m4 dw 10,13,10,13, "To add pr
    m5 dw 10,13,10,13, "Enter qua
    m6 dw 10,13,10,13, "Total pri
    m7 dw 10,13,10,13, "        ***
    m11 dw "4.$"
    m12 dw "/-$"
    m13 dw 10,13,10,13, " Re-odre
    m14 dw 10,13,10,13, " Exit :
    q dw 0
    r dw 0
    v db 0
    s dw 0
    rprice dw 100
    vprice dw 50
    sprice dw 20
    nprice dw 0

    var1 db 100 dup('$')

.code
    main proc

        mov ax,@data
        mov ds,ax

        mov ah,9
        Lea dx,m0
        int 21h

        start:
        cmp v,0
        jg start1

        mov ah,9
        Lea dx,m1
        int 21h

        menu:
```

emulator: restaurent.exe_

registers	H	L
AX	4C	24
BX	01	5E
CX	05	5A
DX	01	8C
CS	F400	
IP	0204	
SS	0710	
SP	00FA	
BP	0000	
SI	0000	
DI	0000	
DS	0720	
ES	0700	

```
F400:0204
F4200: FF 255 RES
F4201: FF 255 RES
F4202: CD 205 =
F4203: 21 033 !
F4204: CF 207 ╧
F4205: 00 000 NULL
F4206: 00 000 NULL
F4207: 00 000 NULL
F4208: 00 000 NULL
F4209: 00 000 NULL
F420A: 00 000 NULL
F420B: 00 000 NULL
F420C: 00 000 NULL
F420D: 00 000 NULL
F420E: 00 000 NULL
F420F: 00 000 NULL
F4210: 00 000 NULL
F4211: 00 000 NULL
F4212: 00 000 NULL
F4213: 00 000 NULL
F4214: 00 000 NULL
F4215: 00 000 NULL
```

```
F400:0204
BIOS DI
INT 021h
IRET
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
```

original source co...

```
mov ah,9
Lea dx,m14
int 21h

mov ah,1
int 21h

cmp al,31h
je start

mov ah,9
Lea dx,m7
int 21h


mov ah,4ch
int 21h
```

message

PROGRAM HAS RETURNED CONTROL
TO THE OPERATING SYSTEM

emulator screen (80x25 chars)

```
        !!!!!!Welcome in our project!!!!!!
Which menu do you want ??please select:
1.Rice 100/- 2.Vegetable 50/- 3.Soup 20/-
Select the menu number:2
Enter quantity: 37
7
Total price: 350
 Re-odrer: Press <1>
 Exit: Press Any key1
     *****THANK YOU*****
```

CONCLUSION:

The documentation includes all necessary information on the structure and the coding of the program created for Restaurant Billing system. Creating the program was an overwhelming task that required a lot of analyzing, research work and personal skills. Creating this report has been a great experience and numerous facts have been learned since the required tasks were very challenging. Tasks such as creating a system to a restaurant, needed research work as well as personal skills. Creating proper design and smooth flow of operation was a very tiring task that consumed a lot of time

Applications

1. Improved accuracy.
2. It saves you time.
3. It allows customer to track menu items easily.

References

https://youtu.be/juk4lp80KLQ

https://youtu.be/zM1GEM8DALY

https://www.nibizsoft.com/26-restaurant-billing-project-assembly-language-programming-x86-emu8086/