VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

CSE2006 - MICROPROCESSOR AND INTERFACING

ALEENA 20BCE0200
GAURANG RASTOGI 20BCE2954
VIVIN D 20BCI0005
NEHA ELSA PAUL 20BCE2451

PROJECT REPORT ON

INVENTORY MANAGEMENT SYSTEM USING EMU8086

WINTER SEMESTER 2021-22

SUBMITTED TO
FACULTY : DR. RAJESH N
DATE : 23/04/2022
SLOT : L11+L12

Contents Page No.

1. Abstract

The primary purpose of making the Inventory Management System is to create an easy way for the shop owners to access the inventory of the store and access stocks using Emu8086.

In this Inventory Management System, help the owners of shops to maintain and keep track of each item in the store. This will also be useful, to quickly generate bills for every purchase made by a customer in the shop.

This project is implemented in the 8086 Microprocessor Emulator (emu8086) using assembly language.

2. Introduction

The juice store near my house uses only paper and pen based bills. The billing people make handwritten bills so the time to finish the bill for each person is too long. The store uses only a paper based inventory system. As Mr. Abhishek, who is the owner, is not always present in the juice store, he is quite afraid that the workers might find it hard to use a paper based system to manage inventory and do billing and is also afraid of the paper wastage in his store, so the owner of the store wishes to give his customers a soft copy of the bill, or a printed bill if they wish and know what products are available and unavailable. Mr. Abhishek approached our team to solve his problem, so we planned to make a suitable Point Of Sales (POS) and an inventory management system so that he can easily keep check of the products in his store and give his customers a soft copy or printed bill.

- The store's inventory manager can access the inventory of the store and edit stocks.
- The finished product will keep track of the products in the inventory and will generate a soft copy of the bill after complete purchase by a customer.

The product will be a computer or laptop based application created using EMU8086 as the programmes and applications written in it may be executed directly on computer hardware without the requirement for translation or interpretation. These applications and programs can run with a very minimal memory footprint and can be executed very fast.
It compiles the source code and executes it on the emulator step by step. Visual interface of EMU8086 is very easy to work with.

3. Literature Survey

S.No	Paper Title	Name of the Conference/ Journal, Year	Technology Used
1	A Machine Tool Fixture Library Management System Based on Assembly Knowledge Description	Advanced Materials Research, vol. 549, Trans Tech Publications, Ltd., July 2012, pp. 1073–1076.	● Based on Group Technology, a machine tool fixture library management system based on assembly knowledge description is put forward to maximize the reuse of fixture design knowledge and experience.
2	A Highly Interactive PC based Simulator Tool for Teaching Microprocessor Architecture and Assembly Language Programming.	Elektronika Ir Elektrotechnika, 98(2), 53-58. Retrieved from https://eejournal.ktu.lt/index.php/elt/article/view/9925 Topaloglu, T., & Gürdal, O. (2010).	● Teaching microprocessor programming in computing science is one of the challenging tasks of the instructors. ● This is mainly because of totally new and different subjects that need to be taught to the students.
3	Application of Microprocessors and Microcontrollers	IJSRD - International Journal for Scientific Research & Development| Vol. 3, Issue 03, 2015 | ISSN (online): 2321-0613	● This paper explains about the applications of microprocessor in various fields
4	An 8-bit Scientific Calculator Based Intel 8086 Virtual Machine Emulator.	The 4th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN-2013).	● A small microprocessor based system was discussed and designed using the Assembly language programming and EMU8086 virtual machine emulator.
5	An Overview of Microprocessors and Assembly Language Programming.	Advances in Interconnect Technologies: An International Journal (AITIJ) Vol .1. No .1, 2017.	● 8086 microprocessors, RISC processors, CISC processors and other special processors.
6	A Research on the Teaching Method of 80X86-Based Assembly Language Programming.	First International Workshop on Education Technology and Computer Science, March 2009.	● 80X86-based memory addressing modes and conditional instructions.

Table 1

4. Drawbacks In The Existing Work

- As the current system in place generates bills manually, the concept of maintaining stock is not possible, and the system is prone to human errors too.
- **Accessibility:** As more computers become connected to the internet, the speed of access is gradually dropping. If new technology does not emerge to solve the problem, the Internet will be flooded with error messages in the near future.
- Since an automation function isn't available in offline/open source systems, operations must be performed manually.
- Risk of computer viruses, cyber hacks and data loss.

5. Proposed Work

We shall we implementing the following features in our Inventory Management System project using Assembly Language Programming for 8086 microprocessor:

1. Login Menu
   This menu is used to move users to their respective module, based on their Username.
2. Point of Sale Menu
   This menu displays the list of all currently available items based on their stock available at the store. This then can be used by the same module to create bills for each customer.
3. Admin Menu
   This Menu allows the admin to modify stock of items that are either depleted or, has has few item that is left in stock at the shop

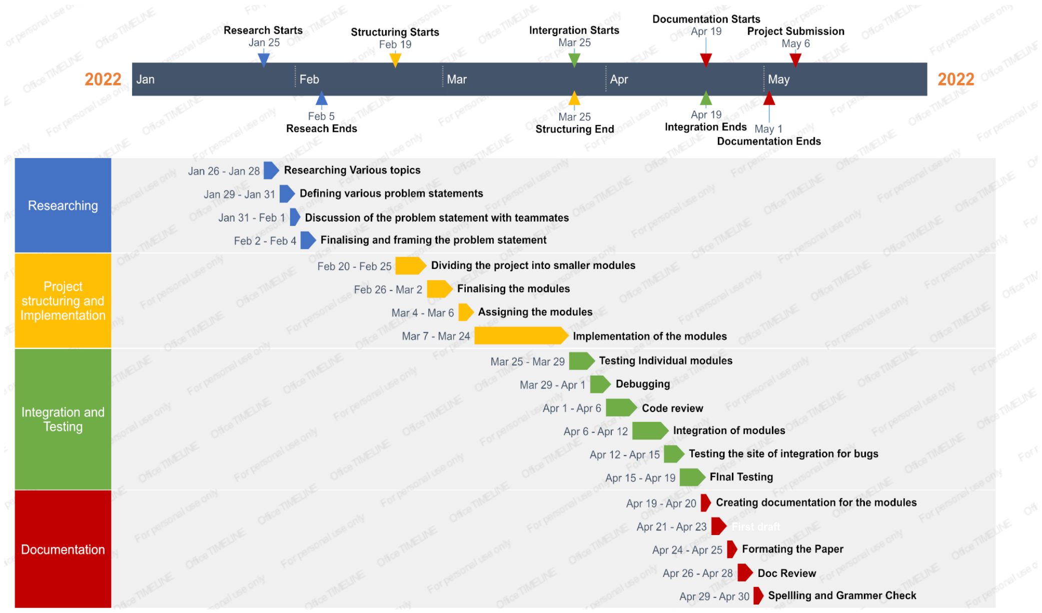
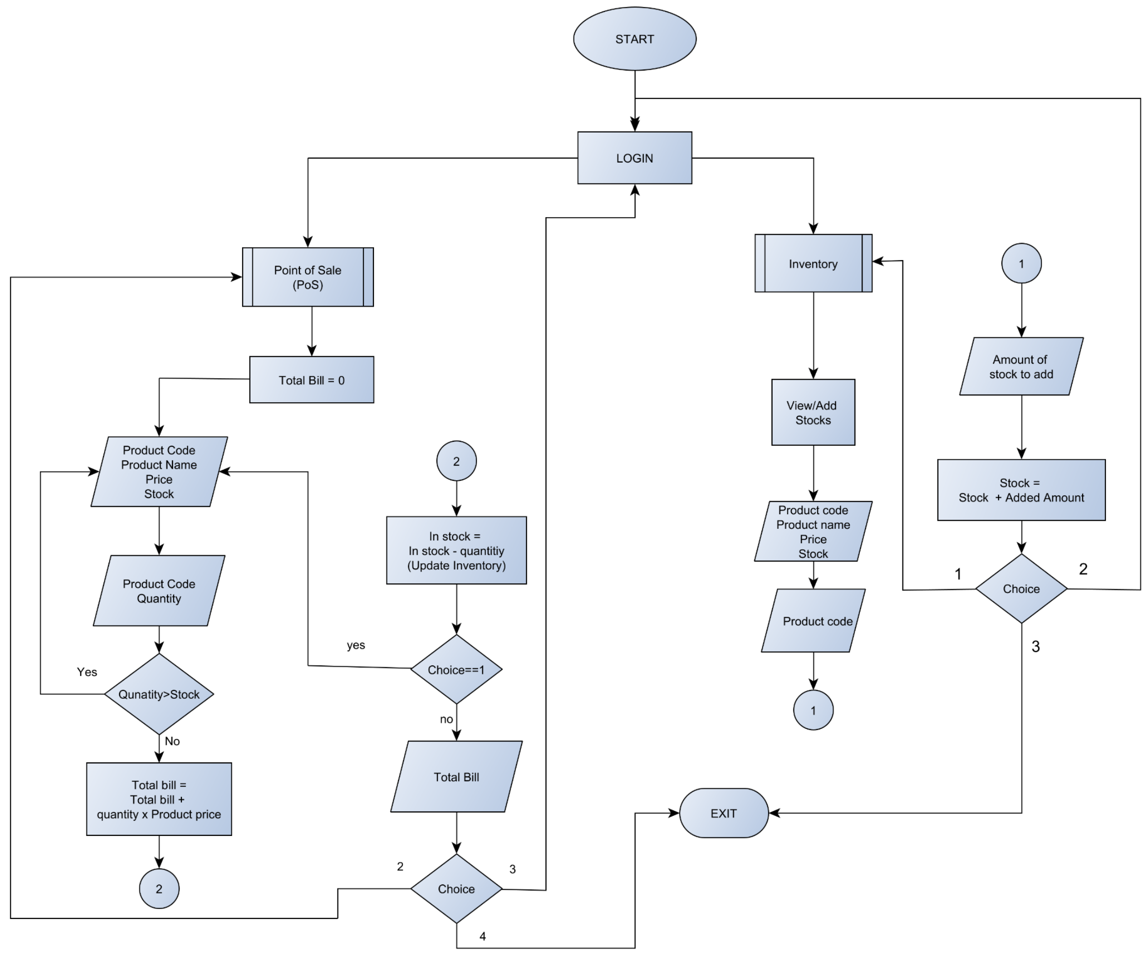


Fig 1. Proposed timeline for Product Completion.

6. Flowchart



START
LOGIN
Point of Sale (PoS)
Total Bill = 0
Product Code Product Name Price Stock
Product Code Quantity
Qunatity>Stock
Yes
No
Total bill = Total bill + quantity x Product price
2
In stock = In stock - quantitiy (Update Inventory)
Choice==1
yes
no
Total Bill
Choice
2
3
4
Inventory
View/Add Stocks
Product code Product name Price Stock
Product code
1
Amount of stock to add
Stock = Stock + Added Amount
Choice
1
2
3
EXIT

Fig 2. Flowchart of the Product

From Fig 2. We have divided the workload of each the project into 3 main modules namely, Login module,

7. Implementation

We have implemented different options of the menu in different parts of the code and it is executed on EMU8086. The different option available in the menu are:

1. **Login**

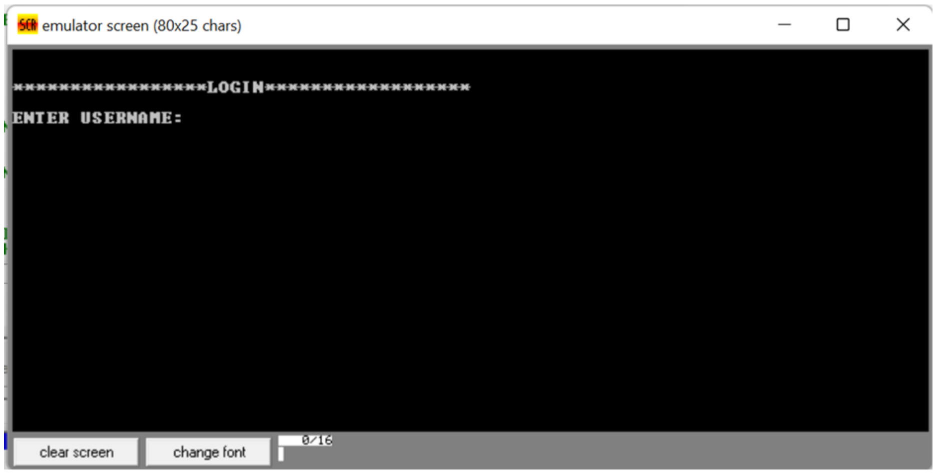
   This Module takes input from the user, a username and a password. When the username and the corresponding password matches with pos's credentials , the user would be navigated to the POS Module. The same happens with the admin user, where the username and password should match with the admin's credentials. When the username and the password doesn't meet the above criteria, we clear the screen.

2. **Point of Sale(POS)**
   This module displays the available items for purchase, along with the amount of stock remaining in the inventory. The module then asks for the product code and then the quantity of the chosen item. When this chosen stock is greater than the available stock an error would be thrown, and the quantity will need to be imputed again. A variable that stores the product of the price of the chosen item, and the quantity. The program branches here into 4 options. Option 1 is to continue with the bill, hence the total will not be set to zero and the same bill for the same customer is continued. Option 2, the bill for the current customer is printed, and then total is set to zero for use by the next customer, by looping to the start of this module. Option 3, will print the total for that particular bill, and move back to the login module. Option 4, will print the total bill, and exit the program.

3. **Admin- Inventory**
   The admin module opens up the option for the admin to add stocks for the items. The module displays the available items for purchase, along with the amount of stock remaining in the inventory. A choice would be given to the user for entering the product code such that the owner can add the stocks for that item. The quantity is to be entered such that the stock for that item can be updated. After the update, the user would be given three options from which the user can decide to either continue updating the stock or return back to the login page or exit the program.

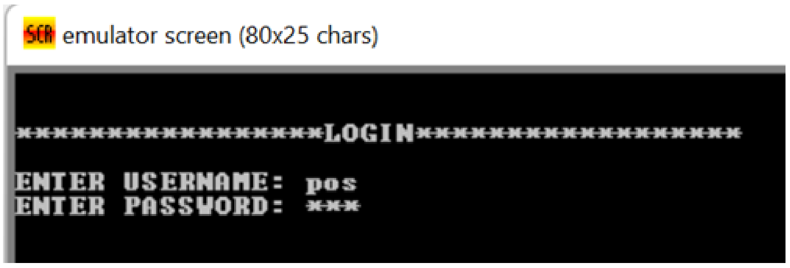
8. Screenshots Of The Prototype

Login Module:

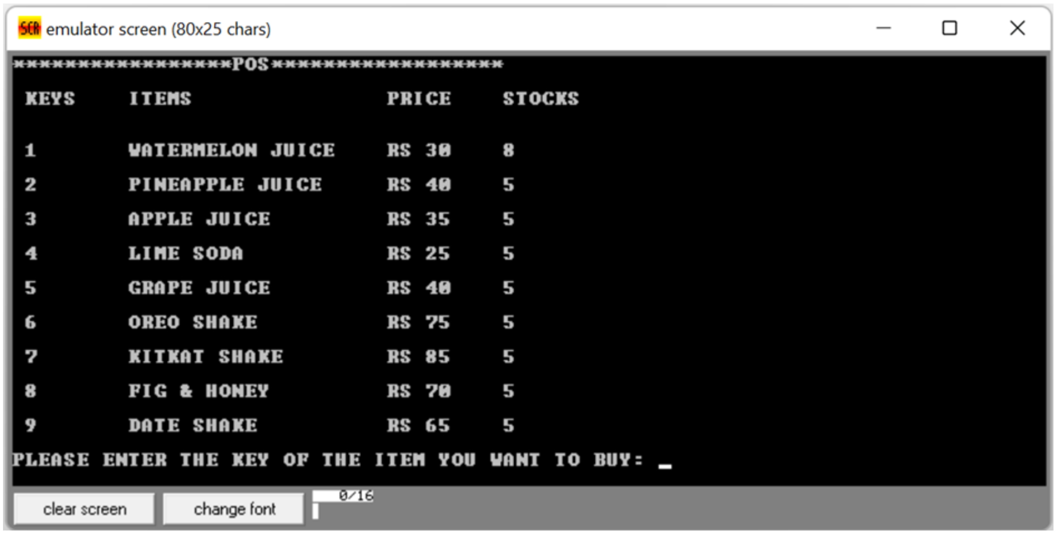


```
*****************LOGIN******************
ENTER USERNAME:
```

POS Module:

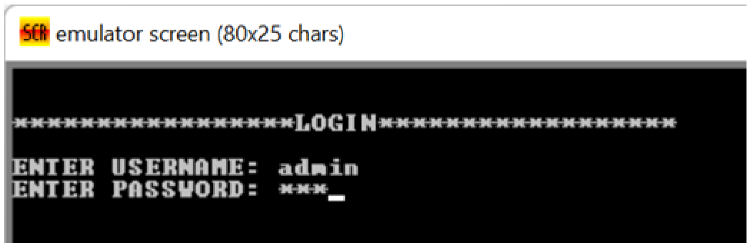

emulator screen (80x25 chars)

```
******************LOGIN*******************

ENTER USERNAME: pos
ENTER PASSWORD: ***
```

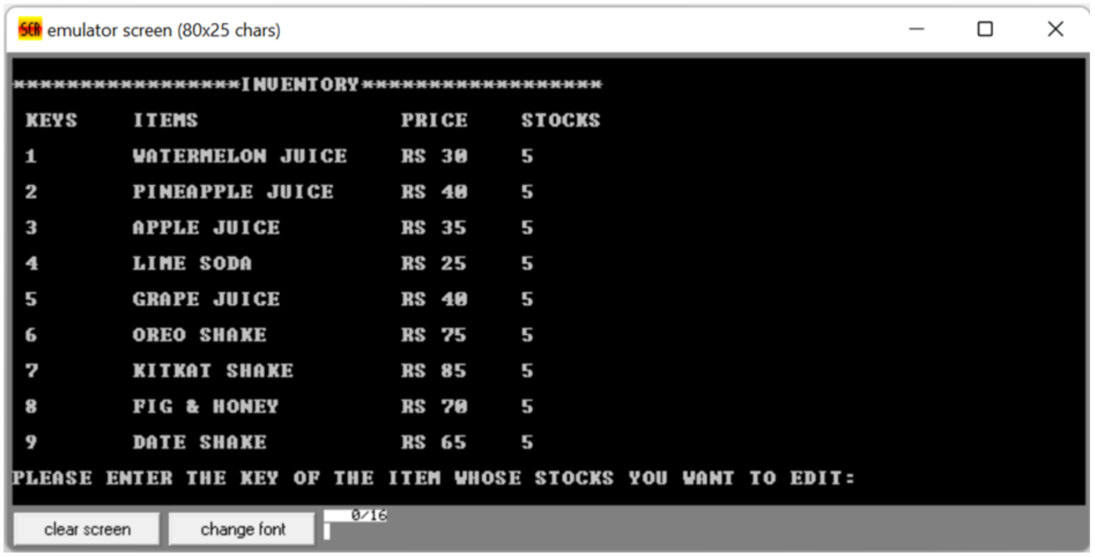

emulator screen (80x25 chars)

```
*****************POS******************
 KEYS    ITEMS               PRICE    STOCKS

 1       WATERMELON JUICE    RS 30    8

 2       PINEAPPLE JUICE     RS 40    5

 3       APPLE JUICE         RS 35    5

 4       LIME SODA           RS 25    5

 5       GRAPE JUICE         RS 40    5

 6       OREO SHAKE          RS 75    5

 7       KITKAT SHAKE        RS 85    5

 8       FIG & HONEY         RS 70    5

 9       DATE SHAKE          RS 65    5
PLEASE ENTER THE KEY OF THE ITEM YOU WANT TO BUY: _
```

Admin Module:


emulator screen (80x25 chars)

```
******************LOGIN*******************

ENTER USERNAME: admin
ENTER PASSWORD: ***_
```


emulator screen (80x25 chars)

```
*****************INVENTORY******************
 KEYS    ITEMS               PRICE    STOCKS
 1       WATERMELON JUICE    RS 30    5
 2       PINEAPPLE JUICE     RS 40    5
 3       APPLE JUICE         RS 35    5
 4       LIME SODA           RS 25    5
 5       GRAPE JUICE         RS 40    5
 6       OREO SHAKE          RS 75    5
 7       KITKAT SHAKE        RS 85    5
 8       FIG & HONEY         RS 70    5
 9       DATE SHAKE          RS 65    5
PLEASE ENTER THE KEY OF THE ITEM WHOSE STOCKS YOU WANT TO EDIT:
```

9. Results

The login module would take the user to their respective window where the items would be displayed. The point of sale accurately generates the bill according to the user's discretion and the admin module successfully adds the stocks to the items in the store.

10. Conclusion

We have successfully implemented our project in Emulator 8086 using assembly language. We have successfully implemented a basic working inventory management system using references and books as our guidance.

Some future scopes for the project includes

1. Adding item and removing item to the Inventory
2. Optimize and introduce Modularity to the code base

11. References

[1]. Cao, Y., Yang, D., & Bai, Y. (2012). A Machine Tool Fixture Library Management System Based on Assembly Knowledge Description. In Advanced Materials Research (Vol. 549, pp. 1073-1076). Trans Tech Publications Ltd.
https://www.eejournal.ktu.lt/index.php/elt/article/view/9925

[2]. Topaloglu, T., & Gürdal, O. (2010). A highly interactive PC based simulator tool for teaching microprocessor architecture and assembly language programming. Elektronika ir Elektrotechnika, 98(2), 53-58.
https://eejournal.ktu.lt/index.php/elt/article/view/9925

[3]. Al Zaman, M. A., & Monira, N. J. AN OVERVIEW OF MICROPROCESSORS AND ASSEMBLY LANGUAGE PROGRAMMING.
https://www.researchgate.net/publication/324417574_An_Overview_of_Microprocessors_and_Assembly_Language_Programming

[4]. Gao, F., Wang, J., & Zhang, J. (2009, March). A research on the teaching method of 80X86-based assembly language programming. In 2009 First International Workshop on Education Technology and Computer Science (Vol. 2, pp. 964-966). IEEE.
https://ieeexplore.ieee.org/abstract/document/4959192

[5]. Raymond, O. U., Kuyoro'Shade, O., Adekunle, Y. A., & Awodele, O. (2013). Application of microprocessors. International Journal of Emerging Technology and Advanced Engineering, 3(4), 488-493.
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.413.7858&rep=rep1&type=pdf

[6]. Al-Haija, Q. A., Al-Abdulatif, S., & Al-Ghofaily, M. (2013). An 8-Bit Scientific Calculator Based Intel 8086 Virtual Machine Emulator. Procedia Computer Science, 21, 506-511. https://www.scientific.net/AMR.549.1073

[7]. https://www.academia.edu/36143631/Writing_Assembly_Language_Program

[8].https://www.academia.edu/35898641/AN_OVERVIEW_OF_MICROPROCESSORS _AND_ ASSEMBLY_LANGUAGE_PROGRAMMING

[9]. https://www.sciencedirect.com/topics/engineering/assembly-language-program

[10].https://www.scribd.com/document/426963618/Microprocessor-8086-Research-Paper

12.Appendix (Code)

```
.MODEL SMALL
.STACK 100H
.DATA

;DECLARED STRINGS

UNAME DB 10,10,13, '******LOGIN*******',10,10,13,'ENTER
USERNAME: $'

PWORD DB 10,13,'ENTER PASSWORD: $'

TEXT DB 10,10,13,'******INVENTORY*******$'


INTRO DB 10,10,13,'******POS*******$'



ENTER DB 10,13,'PLEASE ENTER THE KEY OF THE ITEM YOU WANT TO
BUY: $'

ENTER1 DB 10,13, 'PLEASE ENTER THE KEY OF THE ITEM WHOSE STOCKS
YOU WANT TO EDIT: $'

INFO DB 10,13,' KEYS    ITEMS               PRICE    STOCKS$'

WATERMELON DB 10, ' 1       WATERMELON JUICE    RS 30    $'

PINEAPPLE DB 10,13,' 2       PINEAPPLE JUICE     RS 40    $'

APPLE DB 10,13,' 3       APPLE JUICE         RS 35    $'

LIMESODA DB 10,13,' 4       LIME SODA           RS 25    $'

GRAPE DB 10,13,' 5       GRAPE JUICE         RS 40    $'

OREO DB 10,13,' 6       OREO SHAKE          RS 75    $'

KITKAT DB 10,13,' 7       KITKAT SHAKE        RS 85    $'

FIGNHONEY DB 10,13,' 8       FIG & HONEY         RS 70    $'

DATESHAKE DB 10,13,' 9       DATE SHAKE          RS 65    $'

E_QUANTITY DB 10,13,'ENTER QUANTITY: $'
```

```
AGAIN DB 10,13,'(1.CONTINUE BILL || 2.CHECK OUT || 3.BACK TO
LOGIN || 4.EXIT)', 10,13, 'ENTER CHOICE: $'
AGAIN1 DB 10,13, '(1.ADD MORE STOCKS || 2.BACK TO LOGIN ||
3.EXIT): ',10,13, 'ENTER CHOICE: $'

ER_MSG DB 10,13,'ERROR INPUT$'
ER_MSG1 DB 10, 13, 'NOT ENOUGH STOCK$'
ER_MSG2 DB 10, 13, 'TOO MUCH STOCK$'

CHOICE DB 10,13,'ENTER YOUR CHOICE:$'

FT DB 10,13,'TOTAL AMOUNT IS :$'

ERR DB 0DH,0AH,'WRONG INPUT! START FROM THE BEGINNING $'

ERR2 DB 0DH,0AH,'WRONG INPUT.$'

R DB 0DH,0AH,'PRESENT AMOUNT IS : $'

ERASK DB 10,13,'START FROM THE BEGINNING $'

;DECLARED VARIABLES
A DW ?
B DW ?
C DW ?
S DW 0,'$'

I1 DB 53, '$'
I2 DW 53, '$'
I3 DW 53, '$'
I4 DW 53, '$'
I5 DW 53, '$'
I6 DW 53, '$'
I7 DW 53, '$'
I8 DW 53, '$'
I9 DW 53, '$'
TEMP DW ?
CHO DB ?

USER DB 10 DUP('$')
PASS DB 25 DUP('$')

NL DB 0DH,0AH,'$'                ;NEW LINE

ADMINU DB "admin", "$"

ADMINP DB "123", "$"
```

```
POSUN DB "pos", "$"

POSPW DB "456", "$"

AST DB "*$"

.CODE
     MOV AX, @DATA
     MOV DS, AX
     MOV ES, AX

LOGIN:

    MOV AH,06H ;CLEAR SCREEN INSTRUCTION
    MOV AL,00H ;NUMBER OF LINES TO SCROLL
    MOV BH,07H ;DISPLAY ATTRIBUTE - COLOR
    MOV CH,00D ;START ROW
    MOV CL,00D ;START COL
    MOV DH,25D ;END OF ROW
    MOV DL,80D ;END OF COL
    INT 10H

    ;MOVE CURSOR TO MIDDLE

    MOV AH,02H ;MOVE CURSOR INSTRUCTION
    MOV BH,00H ;PAGE 0
    MOV DH,00D ;ROW
    MOV DL,0D  ;COLUMN
    INT 10H

     LEA DX,UNAME                 ;ASK FOR USERNAME
     MOV AH,9
     INT 21H

     MOV AH, 0AH                  ;TAKE USERNAME INPUT
     LEA DX,USER
     INT 21H

     LEA SI, USER
     INC SI
     INC SI

     LEA DI, ADMINU               ;COMPARE USERNAME
     MOV CX,0005H
     CLD
     REPE CMPSB
```

```
JNZ POSCHECK

ADPASSCHECK:
LEA DX,PWORD                ;ASK FOR PASSWORD
MOV AH,9
INT 21H

MOV CX,03H
LEA SI,PASS

READ:
MOV AH,07H
INT 21H
MOV BYTE PTR[SI],AL

LEA DX, AST                 ;DISPLAY ASTERISK
MOV AH,9
INT 21H

INC SI
DEC CX
JNZ READ


LEA SI,PASS

LEA DI, ADMINP              ;COMPARE PASSWORD
MOV CX,0003H
CLD
REPE CMPSB


MOV AH,01H
INT 21H

JZ ADMIN
JNZ LOGIN


POSCHECK:
LEA SI, USER
INC SI
INC SI

LEA DI, POSUN              ;COMPARE USERNAME
MOV CX,0003H
```

```
     CLD
     REPE CMPSB

     JNZ LOGIN


     POSPASSCHECK:
     LEA DX,PWORD                ;ASK FOR PASSWORD
     MOV AH,9
     INT 21H

     MOV CX,03H
     LEA SI,PASS

     READ2:
     MOV AH,07H
     INT 21H
     MOV BYTE PTR[SI],AL

     LEA DX, AST                 ;DISPLAY ASTERISK
     MOV AH,9
     INT 21H

     INC SI
     DEC CX
     JNZ READ2

     LEA SI,PASS

     LEA DI, POSPW               ;COMPARE PASSWORD
     MOV CX,0003H
     CLD
     REPE CMPSB

     MOV AH,01H
     INT 21H

     JZ POS
     JNZ LOGIN




ADMIN:

     LEA DX,TEXT                ;ASK FOR USERNAME
     MOV AH,9
```

```
INT 21H


LEA DX,NL                   ;PRINT A NEW LINE
MOV AH,9
INT 21H

LEA DX,INFO                 ;PRINT INFO STRING
MOV AH,9
INT 21H

LEA DX,NL                   ;PRINT A NEW LINE
MOV AH,9
INT 21H

LEA DX,WATERMELON              ;PRINT WATERMELON STRING
MOV AH,9
INT 21H

LEA DX, I1                     ;PRINT STOCK
MOV AH,9
INT 21H

LEA DX,NL                   ;PRINT A NEW LINE
MOV AH,9
INT 21H


LEA DX,PINEAPPLE              ;PRINT PINEAPPLE MALE STRING
MOV AH,9
INT 21H

LEA DX, I2                     ;PRINT STOCK
MOV AH,9
INT 21H

LEA DX,NL                   ;PRINT A NEW LINE
MOV AH,9
INT 21H


LEA DX,APPLE            ;PRINT APPLE STRING
MOV AH,9
INT 21H

LEA DX, I3                    ;PRINT STOCK
MOV AH,9
```

```
INT 21H

LEA DX,NL                  ;PRINT A NEW LINE
MOV AH,9
INT 21H


LEA DX,LIMESODA             ;PRINT LIMESODA STRING
MOV AH,9
INT 21H

LEA DX, I4                    ;PRINT STOCK
MOV AH,9
INT 21H


LEA DX,NL                  ;PRINT A NEW LINE
MOV AH,9
INT 21H


LEA DX,GRAPE           ;PRINT GRAPE STRING
MOV AH,9
INT 21H

LEA DX, I5                    ;PRINT STOCK
MOV AH,9
INT 21H

LEA DX,NL                  ;PRINT A NEW LINE
MOV AH,9
INT 21H


LEA DX,OREO         ;PRINT OREO STRING
MOV AH,9
INT 21H

LEA DX, I6                    ;PRINT STOCK
MOV AH,9
INT 21H

LEA DX,NL                  ;PRINT A NEW LINE
MOV AH,9
INT 21H

LEA DX,KITKAT              ;PRINT KITKAT STRING
```

```
MOV AH,9
INT 21H

LEA DX, I7                     ;PRINT STOCK
MOV AH,9
INT 21H

LEA DX,NL                    ;PRINT A NEW LINE
MOV AH,9
INT 21H


LEA DX,FIGNHONEY               ;PRINT FIGNHONEY STRING
MOV AH,9
INT 21H

LEA DX, I8                      ;PRINT STOCK
MOV AH,9
INT 21H

LEA DX,NL                    ;PRINT A NEW LINE
MOV AH,9
INT 21H


LEA DX,DATESHAKE                ;PRINT DATESHAKE STRING
MOV AH,9
INT 21H

LEA DX, I9                      ;PRINT STOCK
MOV AH,9
INT 21H

LEA DX,NL                    ;PRINT A NEW LINE
MOV AH,9
INT 21H

LEA DX,ENTER1                 ;PRINT ENTER STRING
MOV AH,9
INT 21H


MOV AH,1                     ;TAKE AN INPUT & SAVED TO AL
INT 21H
```

```
     CMP AL,49                  ;IF AL=1 GO TO WATERMELONB
LABEL
     JE WATERMELONC

     CMP AL,50                  ;IF AL=2 GO TO PINEAPPLEB LABEL
     JE PINEAPPLEC

     CMP AL,51                  ;IF AL=3 GO TO APPLEB LABEL
     JE APPLEC

     CMP AL,52                  ;IF AL=4 GO TO LIMESODAB LABEL
     JE LIMESODAC

     CMP AL,53                  ;IF AL=5 GO TO GRAPEB LABEL
     JE GRAPEC

     CMP AL,54                  ;IF AL=6 GO TO OREOB LABEL
     JE OREOC

     CMP AL,55                  ;IF AL=7 GO TO KITKATB LABEL
     JE KITKATC

     CMP AL,56                  ;IF AL=8 GO TO FIGNHONEYB LABEL
     JE FIGNHONEYC

     CMP AL,57                  ;IF AL=9 GO TO DATESHAKEB LABEL
     JE DATESHAKEC

WATERMELONC:

MOV A,30                        ;PRICE OF A PRODUCT IS MOVED TO
A
LEA DX, I1
MOV TEMP, DX
JMP QUANTITY1

PINEAPPLEC:

MOV A,40
LEA DX, I2
MOV TEMP, DX
JMP QUANTITY1

APPLEC:

MOV A,35
LEA DX, I3
```

```
MOV TEMP, DX
JMP QUANTITY1

LIMESODAC:

MOV A,25
LEA DX, I4
MOV TEMP, DX
JMP QUANTITY1

GRAPEC:

MOV A,40
LEA DX, I5
MOV TEMP, DX
JMP QUANTITY1

OREOC:

MOV A,75
LEA DX, I6
MOV TEMP, DX
JMP QUANTITY1

KITKATC:

MOV A,85
LEA DX, I7
MOV TEMP, DX
JMP QUANTITY1

FIGNHONEYC:

MOV A,70
LEA DX, I8
MOV TEMP, DX
JMP QUANTITY1

DATESHAKEC:

MOV A,65
LEA DX, I9
MOV TEMP, DX
JMP QUANTITY1


QUANTITY1:
    LEA DX,E_QUANTITY           ;PRINT ENTER QUANTITY STRING
    MOV AH,9
    INT 21H


    MOV AH, 01H                  ;INPUTING QUANTITY
    INT 21H
    MOV AH,00H

    CMP AL,48
    JL QUANTITY1

    CMP AL,57
    JG QUANTITY1


    SUB AL,48

    MOV SI, [TEMP]
    MOV DX, [SI]
    ADD DX, AX
    CMP DL,57                      ;IF DL>9, PRINT ERROR
MESSAGE
    JG ERRORSTOCK1
    MOV [SI], DX

    LEA DX, AGAIN1                 ;ASKING FOR CHOICE
    MOV AH, 09
    INT 21H

    MOV AH, 01H
    INT 21H

    CMP AL,49
    JE ADMIN

    CMP AL,50
    JE LOGIN

    CMP AL, 51
    JE END


POS:
     LEA DX,INTRO                   ;PRINT INTRO STRING
     MOV AH,9
     INT 21H



     LEA DX,NL                      ;PRINT A NEW LINE
     MOV AH,9
     INT 21H

     JMP BEGINTOP                   ;DIRECTLY GO TO BEGINTOP LABEL
WHERE USER WILL GIVE INPUT

 ERROR121:

     LEA DX,ER_MSG                  ;PRINT ERROR MESSAGE
     MOV AH,9
     INT 21H
                                    ;IF USER GIVES AN ERROR THEN
USER WILL BE ASKED TO INPUT AGAIN
     LEA DX,ERASK
     MOV AH,9
     INT 21H


 BEGINTOP:


     LEA DX,INFO                    ;PRINT INFO STRING
     MOV AH,9
     INT 21H

     LEA DX,NL                      ;PRINT A NEW LINE
     MOV AH,9
     INT 21H
     INT 21H

     LEA DX,WATERMELON                 ;PRINT WATERMELON STRING
     MOV AH,9
     INT 21H

     LEA DX, I1                        ;PRINT STOCK
     MOV AH,9
     INT 21H

     LEA DX,NL                      ;PRINT A NEW LINE
     MOV AH,9
```

```
INT 21H


LEA DX,PINEAPPLE              ;PRINT PINEAPPLE MALE STRING
MOV AH,9
INT 21H

LEA DX, I2                     ;PRINT STOCK
MOV AH,9
INT 21H

LEA DX,NL                   ;PRINT A NEW LINE
MOV AH,9
INT 21H


LEA DX,APPLE            ;PRINT APPLE STRING
MOV AH,9
INT 21H

LEA DX, I3                    ;PRINT STOCK
MOV AH,9
INT 21H

LEA DX,NL                   ;PRINT A NEW LINE
MOV AH,9
INT 21H


LEA DX,LIMESODA              ;PRINT LIMESODA STRING
MOV AH,9
INT 21H

LEA DX, I4                     ;PRINT STOCK
MOV AH,9
INT 21H


LEA DX,NL                   ;PRINT A NEW LINE
MOV AH,9
INT 21H


LEA DX,GRAPE            ;PRINT GRAPE STRING
MOV AH,9
INT 21H
```

```
LEA DX, I5                      ;PRINT STOCK
MOV AH,9
INT 21H

LEA DX,NL                    ;PRINT A NEW LINE
MOV AH,9
INT 21H


LEA DX,OREO           ;PRINT OREO STRING
MOV AH,9
INT 21H

LEA DX, I6                      ;PRINT STOCK
MOV AH,9
INT 21H

LEA DX,NL                    ;PRINT A NEW LINE
MOV AH,9
INT 21H

LEA DX,KITKAT               ;PRINT KITKAT STRING
MOV AH,9
INT 21H

LEA DX, I7                     ;PRINT STOCK
MOV AH,9
INT 21H

LEA DX,NL                    ;PRINT A NEW LINE
MOV AH,9
INT 21H


LEA DX,FIGNHONEY               ;PRINT FIGNHONEY STRING
MOV AH,9
INT 21H

LEA DX, I8                      ;PRINT STOCK
MOV AH,9
INT 21H

LEA DX,NL                    ;PRINT A NEW LINE
MOV AH,9
INT 21H
```

```
     LEA DX,DATESHAKE                ;PRINT DATESHAKE STRING
     MOV AH,9
     INT 21H

     LEA DX, I9                     ;PRINT STOCK
     MOV AH,9
     INT 21H

     LEA DX,NL                  ;PRINT A NEW LINE
     MOV AH,9
     INT 21H


     LEA DX,ENTER               ;PRINT ENTER STRING
     MOV AH,9
     INT 21H


     MOV AH,1                   ;TAKE AN INPUT & SAVED TO AL
     INT 21H


     CMP AL,49                  ;IF AL=1 GO TO WATERMELONB
LABEL
     JE WATERMELONB

     CMP AL,50                  ;IF AL=2 GO TO PINEAPPLEB LABEL
     JE PINEAPPLEB

     CMP AL,51                  ;IF AL=3 GO TO APPLEB LABEL
     JE APPLEB

     CMP AL,52                  ;IF AL=4 GO TO LIMESODAB LABEL
     JE LIMESODAB

     CMP AL,53                  ;IF AL=5 GO TO GRAPEB LABEL
     JE GRAPEB

     CMP AL,54                  ;IF AL=6 GO TO OREOB LABEL
     JE OREOB

     CMP AL,55                  ;IF AL=7 GO TO KITKATB LABEL
     JE KITKATB

     CMP AL,56                  ;IF AL=8 GO TO FIGNHONEYB LABEL
     JE FIGNHONEYB
     CMP AL,57                  ;IF AL=9 GO TO DATESHAKEB LABEL
     JE DATESHAKEB



     JMP ERROR121               ;IF WRONG KEYWORD IS PRESSED
THEN THE SHOPLIST WILL SHOW AGAIN


WATERMELONB:

MOV A,30
LEA DX, I1
MOV TEMP, DX
JMP QUANTITY

PINEAPPLEB:

MOV A,40
LEA DX, I2
MOV TEMP, DX
JMP QUANTITY

APPLEB:

MOV A,35
LEA DX, I3
MOV TEMP, DX
JMP QUANTITY

LIMESODAB:

MOV A,25
LEA DX, I4
MOV TEMP, DX
JMP QUANTITY

GRAPEB:

MOV A,40
LEA DX, I5
MOV TEMP, DX
JMP QUANTITY

OREOB:
```

```
MOV A,75
LEA DX, I6
MOV TEMP, DX
JMP QUANTITY

KITKATB:

MOV A,85
LEA DX, I7
MOV TEMP, DX
JMP QUANTITY

FIGNHONEYB:

MOV A,70
LEA DX, I8
MOV TEMP, DX
JMP QUANTITY

DATESHAKEB:

MOV A,65
LEA DX, I9
MOV TEMP, DX
JMP QUANTITY

;AFTER MOVING PRICE PROGRAM WILL JUMP TO QUANTITY LABEL

QUANTITY:


    LEA DX,E_QUANTITY             ;PRINT ENTER QUANTITY STRING
    MOV AH,9
    INT 21H

    JMP MULTI            ;PROGRAM WILL GO TO MULTI LABEL WHERE
THE PRICE WILL BE MILTIPLIED WITH THE AMOUNT


ASK:


    LEA DX,AGAIN                  ;PRINT AGAIN IF USER WANTS TO
BUY MORE
    MOV AH,9
    INT 21H
    MOV AH,1                    ;TAKES THE INPUT OF YES OR NO
    INT 21H

    MOV CHO,AL

    CMP AL,49                   ;IF YES, THEN AGAIN GO TO
SHOPLIST MENU AND BUY AGAIN
    JE BEGINTOP


    CMP AL,50
    JGE OUTPUT2                  ;IF NO, PROGRAM WILL GIVE THE
TOTAL OUTPUT

    LEA DX,ER_MSG
    MOV AH,9                    ;IF ANY WRONG INPUT, PRINT
ERROR MESSAGE AND AGAIN ASK TO BUY AGAIN
    INT 21H

    JMP ASK



ERROR:

    LEA DX,ER_MSG               ;PRINT ERROR MESSAGE
    MOV AH,9
    INT 21H

    JMP QUANTITY                ;JUMP TO QUANTITY LABEL

ERRORSTOCK:
    LEA DX,ER_MSG1               ;PRINT ERROR MESSAGE
    MOV AH,9
    INT 21H

    JMP BEGINTOP

ERRORSTOCK1:
    LEA DX,ER_MSG2               ;PRINT ERROR MESSAGE
    MOV AH,9
    INT 21H


    LEA DX, AGAIN1                 ;ASKING FOR CHOICE
    MOV AH, 09
    INT 21H
    MOV AH, 01H
    INT 21H

    CMP AL,49
    JE ADMIN

    CMP AL,50
    JE LOGIN

    CMP AL, 51
    JE END

    JMP ADMIN

MULTI:

INDEC3 PROC                             ;INDEC3 IS FOR TAKING INPUT
FOR MULTIPLY WITH THE GIVEN AMOUNT

    PUSH BX                             ;TAKE VALUES INTO STACK
    PUSH CX
    PUSH DX



    XOR BX,BX                            ;HOLDS TOTAL

    XOR CX,CX                            ;SIGN


    MOV AH,1                             ;TAKE CHARACTER IN AL
    INT 21H



    REPEAT4:

    CMP AL,48                            ;IF AL<0, PRINT ERROR
MESSAGE
    JL ERROR

    CMP AL,57                            ;IF AL>9, PRINT ERRIR
MESSAGE
    JG ERROR
    AND AX,00FH                         ;CONVERT TO DIGIT
    PUSH AX                             ;SAVE ON STACK

    MOV AX,10                           ;GET 10
    MUL BX                              ;AX=TOTAL * 10
    POP BX                              ;GET DIGIT BACK
    ADD BX,AX                           ;TOTAL = TOTAL X 10 +DIGIT


    MOV AH,1
    INT 21H

    CMP AL,0DH                          ;CARRIAGE RETURN
    JNE REPEAT4                         ;IF NO CARRIEGE RETURN THEN
MOVE ON

    MOV AX,BX                           ;STORE IN AX
    MOV SI,[TEMP]
    MOV DX,[SI]
    SUB DX,AX
    CMP DL,48
    JL ERRORSTOCK                       ;IF DL<0, PRINT ERROR
MESSAGE
    MOV [SI],DX



    JMP MUL_

    POP DX                              ;RESTORE REGISTERS
    POP CX
    POP BX
    RET                                 ;AND RETURN



INDEC3 ENDP                             ;END OF INDEC3

ADD_:


    ;SECOND VALUE STORED IN B
    MOV B,AX



    XOR AX,AX                            ;CLEAR AX
    MOV AX,B                         ;MOV B TO AX
    ADD A,AX                         ;ADD A WITH AX


    MOV AX,A                         ;MOV A TO AX

    PUSH AX                          ;TAKE AX INTO STACK


    JMP END

SUB_:


    ;SECOND VALUE STORED IN B
    MOV B,AX

    LEA DX,R                         ;PRINT PRESENT AMOUNT
STRING
    MOV AH,9
    INT 21H


    XOR AX,AX                        ;CLEAR AX

    MOV AX,B                         ;MOV B TO AX

    PUSH AX

    ADD S,AX

    JMP OUTPUT

MUL_:


    ;SECOND VALUE STORED IN B

    MUL A                            ;MULTIPLY A WITH AX


    PUSH AX                          ;TAKE AX INTO STACK

    MOV A,AX
    JMP SUB_                    ;JUMP TO INP1UT_SUB



    JMP OUTPUT

INPUT_ADD:

INDEC1 PROC                          ;INDEC PROC1 IS FOR ADDING
THE PRESENT AMOUNTS INTO TOTAL

    PUSH BX                          ;TAKE THE VALUES IN STACK
    PUSH CX
    PUSH DX


    BEGIN1:


    XOR BX,BX                        ;HOLDS TOTAL

    XOR CX,CX                        ;SIGN


    MOV AH,1                         ;TAKE CHARACTER IN AL
    INT 21H


    REPEAT2:
                                     ;IF AL<0, PRINT ERROR
MESSAGE
    CMP AL,48
    JL ERROR

    CMP AL,57                        ;IF AL>9, PRINT ERROR
MESSAGE
    JG ERROR


    AND AX,00FH                      ;CONVERT TO DIGIT
    PUSH AX                          ;SAVE ON STACK

    MOV AX,10                        ;GET 10
    MUL BX                           ;AX=TOTAL * 10
    POP BX                           ;GET DIGIT BACK
    ADD BX,AX                        ;TOTAL = TOTAL X 10 +DIGIT
    MOV AH,1                        ;TAKE VALUE INTO AL
    INT 21H

    CMP AL,0DH                      ;CARRIAGE RETURN
    JNE REPEAT2                     ;NO KEEP GOING

    MOV AX,BX                       ;STORE IN AX


    JMP ADD_                        ;JUMP TO ADD_ TO STORE THE
TOTAL VALUE

    POP DX                          ;RESTORE REGISTERS
    POP CX
    POP BX
    RET                             ;AND RETURN



INDEC1 ENDP


OUTPUT:

;OUTDEC PROC IS FOR GIVING THE OUTPUT OF THE PRESENT AMOUNT

OUTDEC PROC


    PUSH AX                         ;SAVE REGISTERS
    PUSH BX
    PUSH CX
    PUSH DX

    XOR CX,CX                       ;CX COUNTS DIGITS
    MOV BX,10D                      ;BX HAS DIVISOR

    REPEAT1:

    XOR DX,DX                       ;PREP HIGH WORD
    DIV BX                          ;AX = QUOTIENT,
DX=REMAINDER

    PUSH DX                         ;SAVE REMAINDER ON STACK
    INC CX                          ;COUNT = COUNT +1
    OR AX,AX                         ;QUOTIENT = 0?
    JNE REPEAT1                      ;NO, KEEP GOING

    MOV AH,2                         ;PRINT CHAR FUNCTION

    PRINT_LOOP:

    POP DX                           ;DIGIT IN DL
    OR DL,30H                        ;CONVERT TO CHAR
    INT 21H                          ;PRINT DIGIT
    LOOP PRINT_LOOP                  ;LOOP UNTILL DONE

    POP DX
    POP CX                           ;RESTORE REGISTERS
    POP BX
    POP AX

    JMP ASK

    RET
    OUTDEC ENDP

OUTPUT2:

    LEA DX,FT                        ;PRINT FINAL TOTAL
    MOV AH,9
    INT 21H

    XOR AX,AX                        ;CLEAR AX

    MOV AX,S                         ;SET AX INTO 0


    ;OUTDEC2 IS FOR GIVING THE TOTAL OUTPUT OF THE AMOUNT


OUTDEC2 PROC


    PUSH AX                          ;SAVE REGISTERS
    PUSH BX
    PUSH CX
    PUSH DX

    XOR CX,CX                        ;CX COUNTS DIGITS
    MOV BX,10D                       ;BX HAS DIVISOR
    REPEAT12:

    XOR DX,DX                       ;PREP HIGH WORD
    DIV BX                          ;AX = QUOTIENT,
DX=REMAINDER

    PUSH DX                         ;SAVE REMAINDER ON STACK
    INC CX                          ;COUNT = COUNT +1

    OR AX,AX                        ;QUOTIENT = 0?
    JNE REPEAT12                    ;NO, KEEP GOING

    MOV AH,2                        ;PRINT CHAR FUNCTION

    PRINT_LOOP2:

    POP DX                          ;DIGIT IN DL
    OR DL,30H                       ;CONVERT TO CHAR
    INT 21H                         ;PRINT DIGIT
    LOOP PRINT_LOOP2                ;LOOP UNTILL DONE

    POP DX
    POP CX                          ;RESTORE REGISTERS
    POP BX
    POP AX


    OUTDEC2 ENDP


    MOV AH,01H
    INT 21H


    MOV AL,CHO
    MOV S,0

    CMP AL, 50
    JE POS

    CMP AL,51
    JE LOGIN

    CMP AL,52
    JE END
```

```
END:
    MOV AH, 4CH
    INT 21H
```