Name: Youssef Samuel Nachaat Id:6978 Group:2 Section:2

Final Project Microprocessors

Project 8

ATM Machine

Database:

ADDRESS	CONTENT (HEX)	CUSTOMERS
21000H	00H	1.Card = 32768
21001H	80H	Pass = 0
21002H	00H	
21003H	FFH	2.Card = 32767
21004H	7FH	Pass = 1
21005H	01H	
21006H	FEH	3.Card = 32766
21007H	7FH	Pass = 2
21008H	02H	
21009H	FDH	4.Card = 32765
2100AH	7FH	Pass = 3
2100BH	03H	
2100CH	FCH	5.Card = 32764
2100DH	7FH	Pass = 4
2100EH	04H	
2100FH	FBH	6.Card = 32763
21010H	7FH	Pass = 5
21011H	05H	
21012H	FAH	7.Card = 32762
21013H	7FH	Pass = 6
21014H	06H	
21015H	F9H	8.Card = 32761
21016H	7FH	Pass = 7
21017H	07H	
21018H	F8H	9.Card = 32760
21019H	7FH	Pass = 8
2101AH	08H	

Name: Youssef Samuel Nachaat Id:6978 Group:2 Section:2

2101BH	**F7H**	**10.Card = 32759**
2101CH	**7FH**	**Pass = 9**
2101DH	**09H**	
2101EH	**F6H**	**11.Card = 32758**
2101FH	**7FH**	**Pass = 10**
21020H	**0AH**	
21021H	**F5H**	**12.Card = 32757**
21022H	**7FH**	**Pass = 11**
21023H	**0BH**	
21024H	**F4H**	**13.Card = 32756**
21025H	**7FH**	**Pass = 12**
21026H	**0CH**	
21027H	**F3H**	**14.Card = 32755**
21028H	**7FH**	**Pass = 13**
21029H	**0DH**	
2102AH	**F2H**	**15.Card = 32754**
2102BH	**7FH**	**Pass = 14**
2102CH	**0EH**	
2102DH	**F1H**	**16.Card = 32753**
2102EH	**7FH**	**Pass = 15**
2102FH	**0FH**	
21030H	**F0H**	**17.Card = 32752**
21031H	**7FH**	**Pass = 0**
21032H	**00H**	
21033H	**EFH**	**18.Card = 32751**
21034H	**7FH**	**Pass = 1**
21035H	**01H**	
21036H	**EEH**	**19.Card = 32750**
21037H	**7FH**	**Pass = 2**
21038H	**02H**	
21039H	**EDH**	**20.Card = 32749**
2103AH	**7FH**	**Pass = 3**
2103BH	**03H**	

Name: Youssef Samuel Nachaat Id:6978 Group:2 Section:2

Code:

The code is divided into 3 parts, each part is implemented using a procedure.

1. Construction of the database. (CONST PROC)

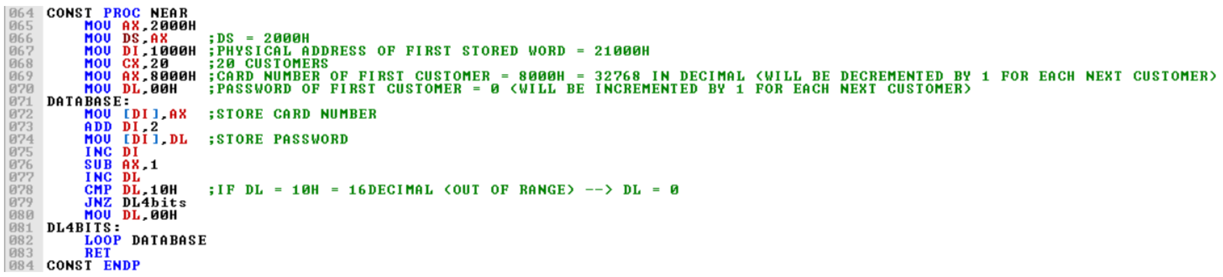
DS = 2000H, DI = 1000H, AX = 8000H, CX = 20, DL = 00H.

The customers' data are stored in memory starting from the physical address = 20000 + 1000 = 21000H. The first customer has a card number 8000H and password 00H. For the next 19 customers, the card number is decremented by 1 each time: 7FFFH, 7FFEH, … until 7FEDH, and for the passwords they are each time incremented by 1: 01H, 02H, … until the password reaches FFH and then it returns to 00H for the seventeenth customer and the last customer has a password 03H.

As required the card number is 16 bits, which means within the range from 0 to 65535 in decimal, and 0000H to FFFH in hexadecimal.

The password is 4 bits, from 0 to 15 is decimal, 0H to FH in hexadecimal.

To store the 20 customers in memory each customer takes 3 bytes, 2 bytes for the card number and 1 byte for the password, so we need 60 bytes, from 21000H to 2103BH.



```
CONST PROC NEAR
     MOV AX,2000H
     MOV DS,AX     ;DS = 2000H
     MOV DI,1000H  ;PHYSICAL ADDRESS OF FIRST STORED WORD = 21000H
     MOV CX,20     ;20 CUSTOMERS
     MOV AX,8000H  ;CARD NUMBER OF FIRST CUSTOMER = 8000H = 32768 IN DECIMAL (WILL BE DECREMENTED BY 1 FOR EACH NEXT CUSTOMER)
     MOV DL,00H    ;PASSWORD OF FIRST CUSTOMER = 0 (WILL BE INCREMENTED BY 1 FOR EACH NEXT CUSTOMER)
DATABASE:
     MOV [DI],AX   ;STORE CARD NUMBER
     ADD DI,2
     MOV [DI],DL   ;STORE PASSWORD
     INC DI
     SUB AX,1
     INC DL
     CMP DL,10H    ;IF DL = 10H = 16DECIMAL (OUT OF RANGE) --> DL = 0
     JNZ DL4bits
     MOV DL,00H
DL4BITS:
     LOOP DATABASE
     RET
CONST ENDP
```

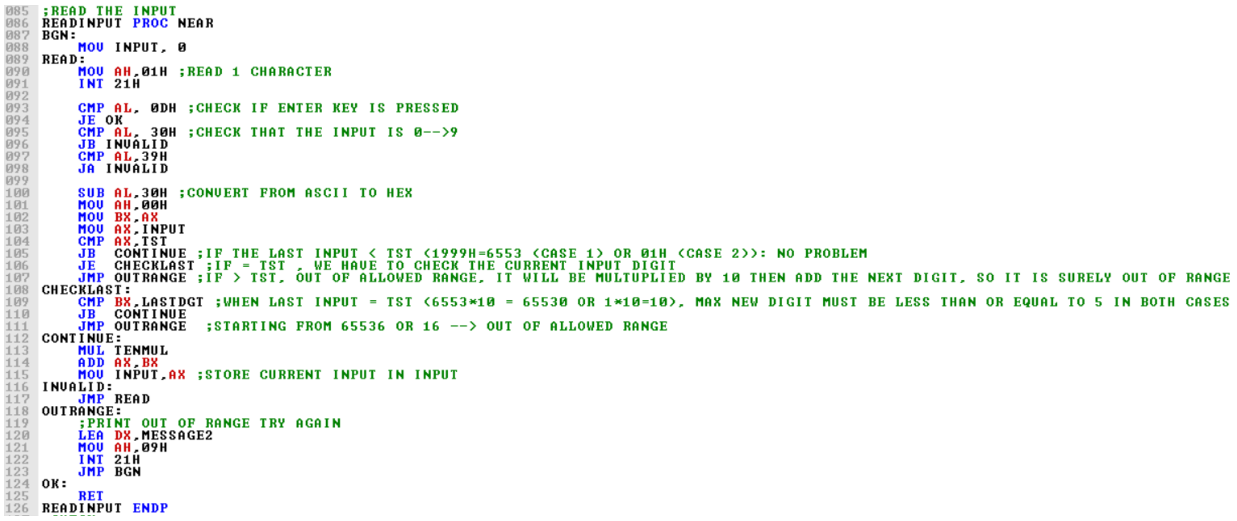
2. Reading the input. (READINPUT PROC)

The user enters the card number and the password. The input is in decimals, and a procedure called READINPUT is implemented to perform this task. For each character entered by the user, it is checked whether it is a digit from 0 to 9, then it is converted from ASCII to hexadecimal. Then, if allowed, the digit entered is shifted one place to the left by multiplying by 10 and the last result (for the first time = 0) is add to it. For each next digit the same process occurs until the user press enter button. But, how to check that the value is within the allowed range?

For the card number, which is 16 bits, in each iteration the last result is compared with the value 1999H = 6553. 6553 is a special value stored in TST, let's consider the last result entered is 7883 in decimal, so when multiplying by 10 and adding the last digit it will be surely greater than 65535 (our upper bound), so the user is prompted to re-enter the card number. What if the last result is exactly equals to 6553, we have here to check the new digit entered, if lower than or equal to 5 no problem.

For the password the same procedure is used and the value of TST here = 1H for the same purpose.

After reading both inputs from the user which are stored in memory locations labeled by CARDNUM and PASSWORD, the card number is moved to AX and password to DL to be used in the last step: Check.



```
;READ THE INPUT
READINPUT PROC NEAR
BGN:
    MOV INPUT, 0
READ:
    MOV AH,01H ;READ 1 CHARACTER
    INT 21H

    CMP AL, 0DH ;CHECK IF ENTER KEY IS PRESSED
    JE OK
    CMP AL, 30H ;CHECK THAT THE INPUT IS 0-->9
    JB INVALID
    CMP AL,39H
    JA INVALID

    SUB AL,30H ;CONVERT FROM ASCII TO HEX
    MOV AH,00H
    MOV BX,AX
    MOV AX,INPUT
    CMP AX,TST
    JB  CONTINUE ;IF THE LAST INPUT < TST (1999H=6553 (CASE 1) OR 01H (CASE 2)): NO PROBLEM
    JE  CHECKLAST ;IF = TST , WE HAVE TO CHECK THE CURRENT INPUT DIGIT
    JMP OUTRANGE ;IF > TST, OUT OF ALLOWED RANGE, IT WILL BE MULTUPLIED BY 10 THEN ADD THE NEXT DIGIT, SO IT IS SURELY OUT OF RANGE
CHECKLAST:
    CMP BX,LASTDGT ;WHEN LAST INPUT = TST (6553*10 = 65530 OR 1*10=10), MAX NEW DIGIT MUST BE LESS THAN OR EQUAL TO 5 IN BOTH CASES
    JB  CONTINUE
    JMP OUTRANGE  ;STARTING FROM 65536 OR 16 --> OUT OF ALLOWED RANGE
CONTINUE:
    MUL TENMUL
    ADD AX,BX
    MOV INPUT,AX ;STORE CURRENT INPUT IN INPUT
INVALID:
    JMP READ
OUTRANGE:
    ;PRINT OUT OF RANGE TRY AGAIN
    LEA DX,MESSAGE2
    MOV AH,09H
    INT 21H
    JMP BGN
OK:
    RET
READINPUT ENDP
```

3. Validation of input. (CHECK PROC)

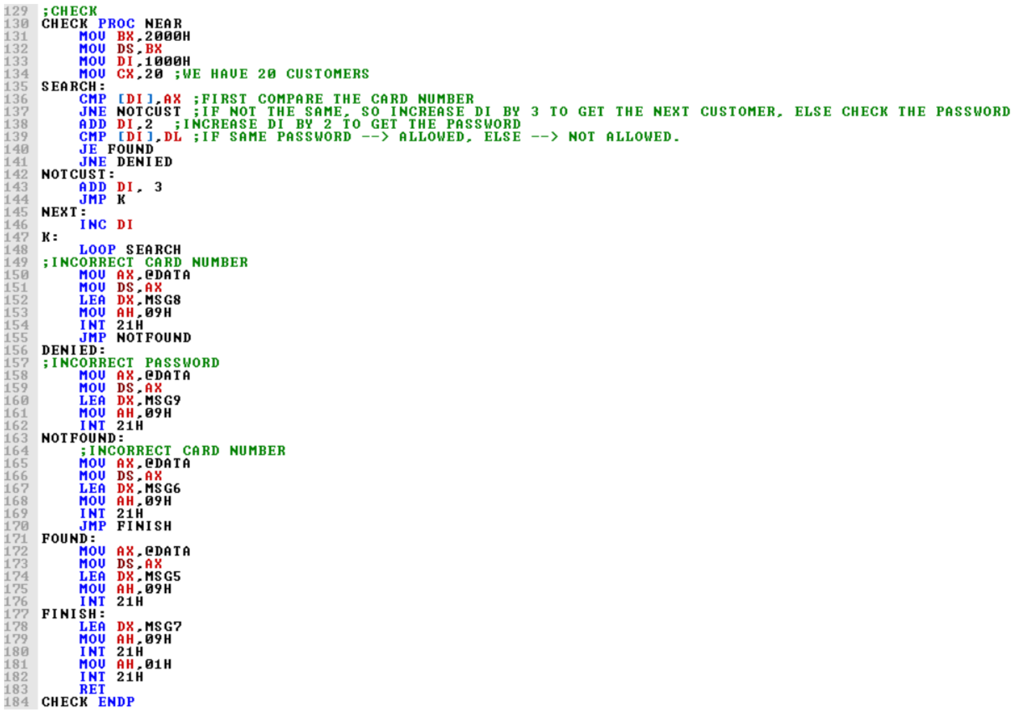
The database constructed in step 1 is now checked for the customer whose data is entered. Start from address 21000H

If the card number matches, go check for the password by incremented DI by 2, if the password also matches then the customer is found, else customer not found.

If the card number is incorrect go check for the next customer by incrementing DI by 3.

After checking the whole table, if the customer is not found, print "DENIED: 0".

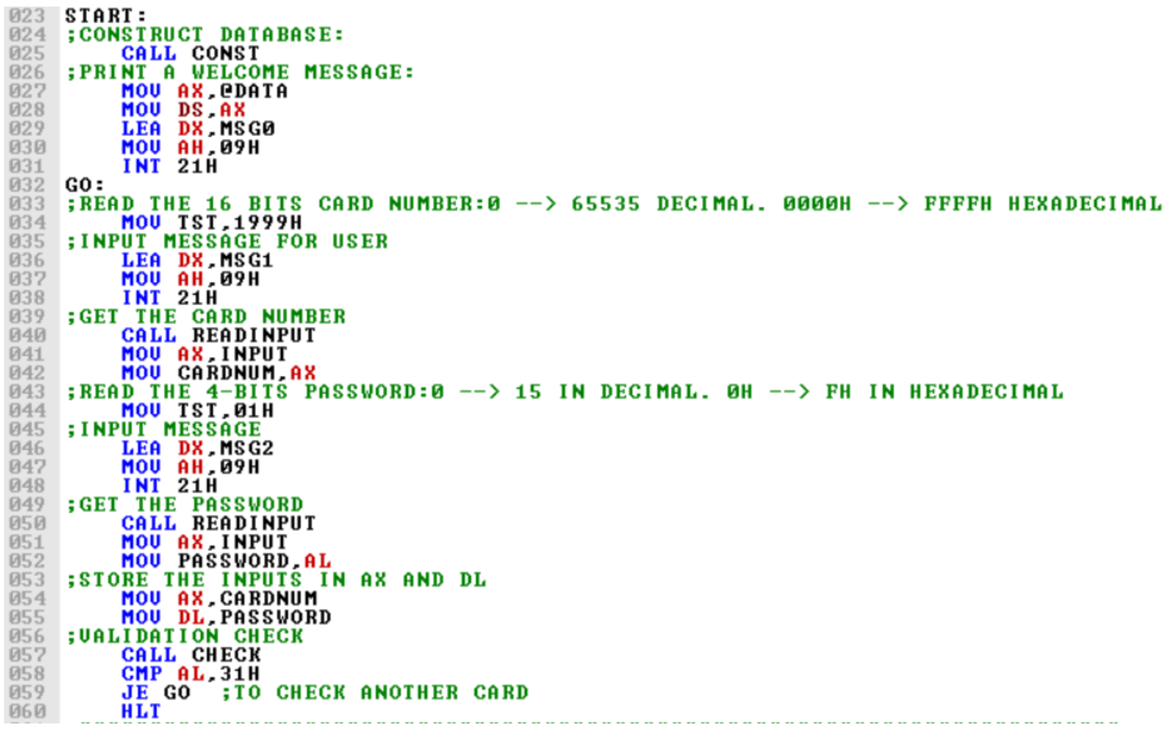
Name: Youssef Samuel Nachaat Id:6978 Group:2 Section:2



```
;CHECK
CHECK PROC NEAR
      MOV BX,2000H
      MOV DS,BX
      MOV DI,1000H
      MOV CX,20 ;WE HAVE 20 CUSTOMERS
SEARCH:
      CMP [DI],AX ;FIRST COMPARE THE CARD NUMBER
      JNE NOTCUST ;IF NOT THE SAME, SO INCREASE DI BY 3 TO GET THE NEXT CUSTOMER, ELSE CHECK THE PASSWORD
      ADD DI,2  ;INCREASE DI BY 2 TO GET THE PASSWORD
      CMP [DI],DL ;IF SAME PASSWORD --> ALLOWED, ELSE --> NOT ALLOWED.
      JE FOUND
      JNE DENIED
NOTCUST:
      ADD DI, 3
      JMP K
NEXT:
      INC DI
K:
      LOOP SEARCH
;INCORRECT CARD NUMBER
      MOV AX,@DATA
      MOV DS,AX
      LEA DX,MSG8
      MOV AH,09H
      INT 21H
      JMP NOTFOUND
DENIED:
;INCORRECT PASSWORD
      MOV AX,@DATA
      MOV DS,AX
      LEA DX,MSG9
      MOV AH,09H
      INT 21H
NOTFOUND:
      ;INCORRECT CARD NUMBER
      MOV AX,@DATA
      MOV DS,AX
      LEA DX,MSG6
      MOV AH,09H
      INT 21H
      JMP FINISH
FOUND:
      MOV AX,@DATA
      MOV DS,AX
      LEA DX,MSG5
      MOV AH,09H
      INT 21H
FINISH:
      LEA DX,MSG7
      MOV AH,09H
      INT 21H
      MOV AH,01H
      INT 21H
      RET
CHECK ENDP
```

Here is the main part of the code, first the CONST procedure is called to construct the database, then READINPUT is called two times for card number and password, and the CHECK is called at the end to validate the input.

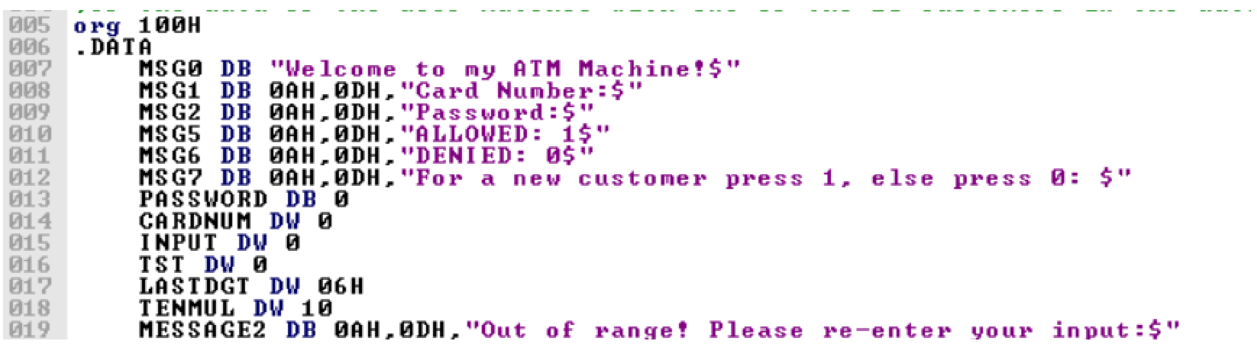
Finally, the user is asked whether to end the program or to check another customer.



```
START:
;CONSTRUCT DATABASE:
      CALL CONST
;PRINT A WELCOME MESSAGE:
      MOV AX,@DATA
      MOV DS,AX
      LEA DX,MSG0
      MOV AH,09H
      INT 21H
GO:
;READ THE 16 BITS CARD NUMBER:0 --> 65535 DECIMAL. 0000H --> FFFFH HEXADECIMAL
      MOV TST,1999H
;INPUT MESSAGE FOR USER
      LEA DX,MSG1
      MOV AH,09H
      INT 21H
;GET THE CARD NUMBER
      CALL READINPUT
      MOV AX,INPUT
      MOV CARDNUM,AX
;READ THE 4-BITS PASSWORD:0 --> 15 IN DECIMAL. 0H --> FH IN HEXADECIMAL
      MOV TST,01H
;INPUT MESSAGE
      LEA DX,MSG2
      MOV AH,09H
      INT 21H
;GET THE PASSWORD
      CALL READINPUT
      MOV AX,INPUT
      MOV PASSWORD,AL
;STORE THE INPUTS IN AX AND DL
      MOV AX,CARDNUM
      MOV DL,PASSWORD
;VALIDATION CHECK
      CALL CHECK
      CMP AL,31H
      JE GO  ;TO CHECK ANOTHER CARD
      HLT
```

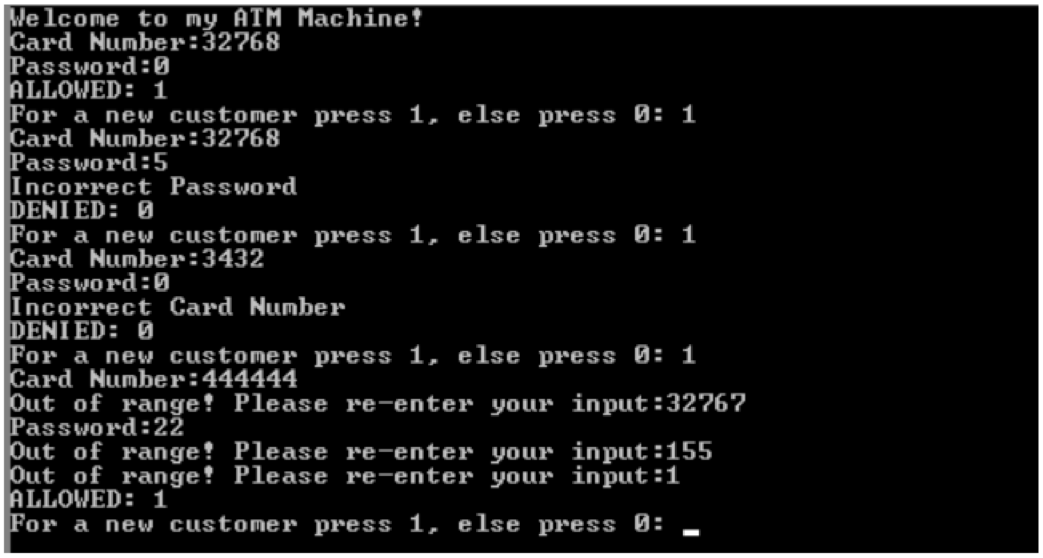
Name: Youssef Samuel Nachaat Id:6978 Group:2 Section:2

Data Segment:



```
org 100H
.DATA
    MSG0 DB "Welcome to my ATM Machine!$"
    MSG1 DB 0AH,0DH,"Card Number:$"
    MSG2 DB 0AH,0DH,"Password:$"
    MSG5 DB 0AH,0DH,"ALLOWED: 1$"
    MSG6 DB 0AH,0DH,"DENIED: 0$"
    MSG7 DB 0AH,0DH,"For a new customer press 1, else press 0: $"
    PASSWORD DB 0
    CARDNUM DW 0
    INPUT DW 0
    TST DW 0
    LASTDGT DW 06H
    TENMUL DW 10
    MESSAGE2 DB 0AH,0DH,"Out of range! Please re-enter your input:$"
```

Sample Run:



```
Welcome to my ATM Machine!
Card Number:32768
Password:0
ALLOWED: 1
For a new customer press 1, else press 0: 1
Card Number:32768
Password:5
Incorrect Password
DENIED: 0
For a new customer press 1, else press 0: 1
Card Number:3432
Password:0
Incorrect Card Number
DENIED: 0
For a new customer press 1, else press 0: 1
Card Number:444444
Out of range! Please re-enter your input:32767
Password:22
Out of range! Please re-enter your input:155
Out of range! Please re-enter your input:1
ALLOWED: 1
For a new customer press 1, else press 0: _
```