Συστήματα Μικρουπολογιστών

5η Ομάδα Ασκήσεων

Μοίρας Αλέξανδρος Α.Μ.: el18081

Παπαδημητρίου Ευθύμιος Α.Μ.: el18129

Ασκήσεις Προσομοίωσης

Σημείωση: Οι κώδικες όλων των παρακάτω ασκήσεων συνοδεύονται από αναλυτικά σχόλια που επεξηγούν τη λειτουργία τους.

1η Άσκηση:

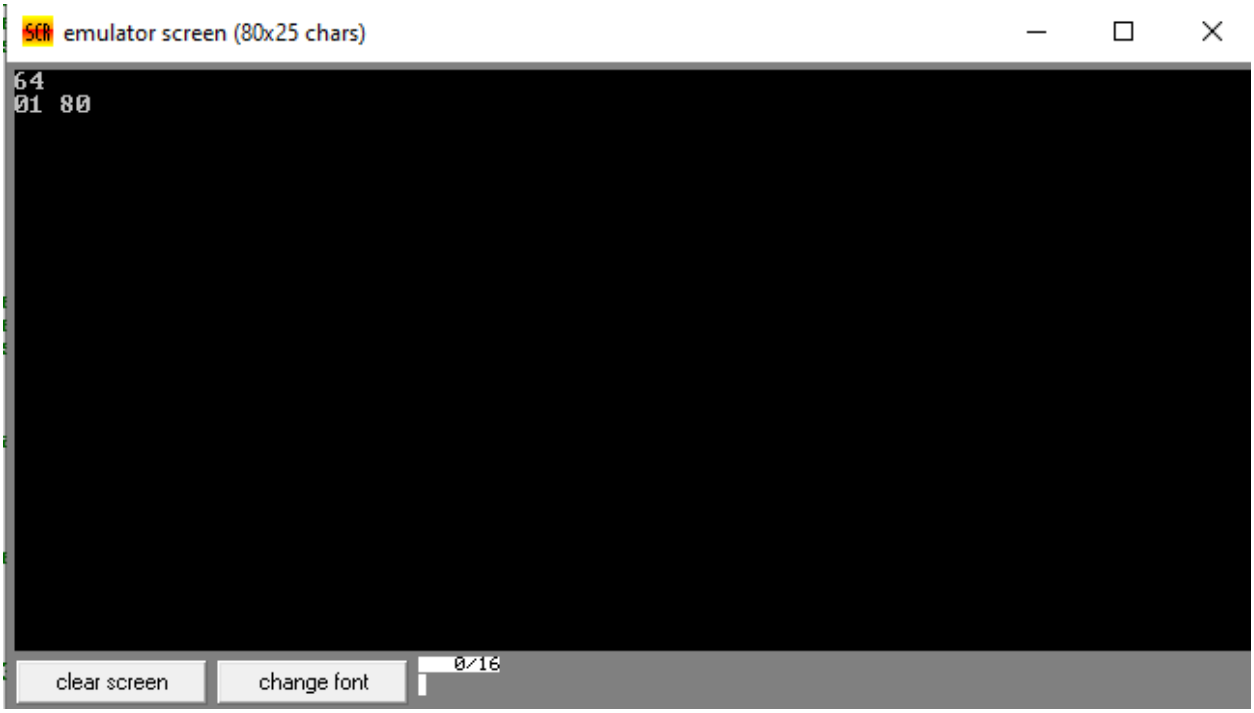
```
DATA SEGMENT
    TABLE DB 128 DUP(?) ;Allocate uninitialized memmory for table to store the nu
mbers in
    TWO DB DUP(2)       ;Store number two in memmory to use it for divisions
DATA ENDS

CODE SEGMENT
    ASSUME CS:CODE, DS:DATA

    MAIN PROC FAR
        MOV AX,DATA
        MOV DS,AX

        MOV DI,0 ;Array index
        MOV CX,128 ; # of numbers to be stored

    STORENUMS:
        MOV TABLE[DI], CL ; Store the number contained in CL in the table (CL ins
tead of CX because we store bytes)
        INC DI ; Increase the index
        LOOP STORENUMS ; Repeat until all the numbers are stored. LOOP will decre
ase the value of CX by 1 so we will store the correct number next


        MOV DX,0 ; Sum of all odd numbers in DX
        MOV BX,0 ; # of odd numbers
        MOV DI,0 ; Array index
        MOV CX,128 ; # of numbers to be checked
    AVERAGE:
        MOV AH,0
        MOV AL,TABLE[DI]
```

```
        DIV TWO ; Divide the number that is being checked by 2
        CMP AH,0 ; If it is even (remainder in AH=0) move to the next number
        JE SKIP
        MOV AL,TABLE[DI]
        MOV AH,0
        ADD DX,AX ; If it is odd add it to the sum of all odds
        INC BX ; Increase the odd number count
    SKIP:
        INC DI ; Next number (array index += 1)
        LOOP AVERAGE ; Loop until all numbers have been checked

        MOV AX,DX ; Move the sum of all odd numbers to AX in order to divide it
        MOV DX,0 ; Set DX to 0 because we will divide with 16bit source to get 16
bit result
        DIV BX ; Divide by the count of all odd numbers to get the average

    ; Print the result in decimal form
        MOV CX,0
    DIGIT:
        MOV DX,0 ;Digit counter on CX
        MOV BX,10 ;Divide by 10 to get the last digit as the remainder
        DIV BX
        PUSH DX ; and push it into the stack
        INC CX ; increase the digit counter
        CMP AX,0 ; while the quotient is not zero (there are more digits) repeat
        JNE DIGIT

    PRINT_DECIMAL_NUMBER: ;Once all digits are in the stack print them one by one
        POP DX
        ADD DL,30H
        MOV AH,2
        INT 21H
        LOOP PRINT_DECIMAL_NUMBER


        MOV DL, 0AH ; New line
        MOV AH,2
        INT 21H
        MOV DL, 0DH ; REturn cursor to the start of the line
        MOV AH,2
        INT 21H

        MOV BH,TABLE[0] ;Initialize max
        MOV BL,TABLE[0] ;Initialize min
```

```
        MOV DI,1   ; Index on second element of table in order to compare with th
e rest of the 127 numbers
        MOV CX,127 ; 127 numbers to check
    FINDMAXMIN:
        MOV AL,TABLE[DI] ; If the element checked is greater than the current max
 change the current max
        CMP BH, AL
        JNC CHANGEMAX
        CMP BL, AL ; If the element checked is less than the current min change t
he current min
        JC CHANGEMIN
    NEXT_ITER:
        INC DI ; Check the next element
        LOOP FINDMAXMIN ; Repeat until all elements are checked
        JMP PRINT_MAX_MIN ; Print the results

    CHANGEMAX:
        MOV BH, AL
        JMP NEXT_ITER

    CHANGEMIN:
        MOV BL,AL
        JMP NEXT_ITER

    PRINT_MAX_MIN:
        MOV DL, BH ; For the max
        AND DL, 0F0H ; Isolate the 4 most significant digits of the number
        MOV CL, 4
        RCR DL, CL ; Move them to the 4 least significant positions
        CALL PRINT_HEX ; Print them as one hexadecimal digit
        MOV DL, BH ; Isolate the 4 least significant digits and print them as ano
ther hex number
        AND DL, 0FH
        CALL PRINT_HEX

        MOV DL,20H ; Print " "
        MOV AH,2
        INT 21H

        MOV DL, BL ; For the min
        AND DL, 0F0H ; Isolate the 4 most significant digits of the number
        MOV CL, 4
        RCR DL, CL ; Move them to the 4 least significant positions
        CALL PRINT_HEX ; Print them as one hexadecimal digit
```

```
        MOV DL, BL ; Isolate the 4 least significant digits and print them as ano
ther hex number
        AND DL, 0FH
        CALL PRINT_HEX

        MOV AX, 4C00H ; Exit the program and return control to the operating syst
em
        INT 21H
MAIN ENDP

PRINT_HEX PROC NEAR
        CMP DL, 9 ; If the number is between 0 and 9 add the value 30H ('0'=30H).
        JG ADDR1
        ADD DL, 30H
        JMP ADDR2
    ADDR1:
        ADD DL, 37H ; else add the value 37H ('A'=41H).
    ADDR2:
        MOV AH,2
        INT 21H
        RET
PRINT_HEX ENDP
    CODE ENDS
END MAIN
```

Και η έξοδος του προγράμματος:



Πράγματι ο μέσος όρος των περιττών αριθμών είναι 64, ο ελάχιστος αριθμός 01H και ο μέγιστος ο 80H=128.

2η Άσκηση:

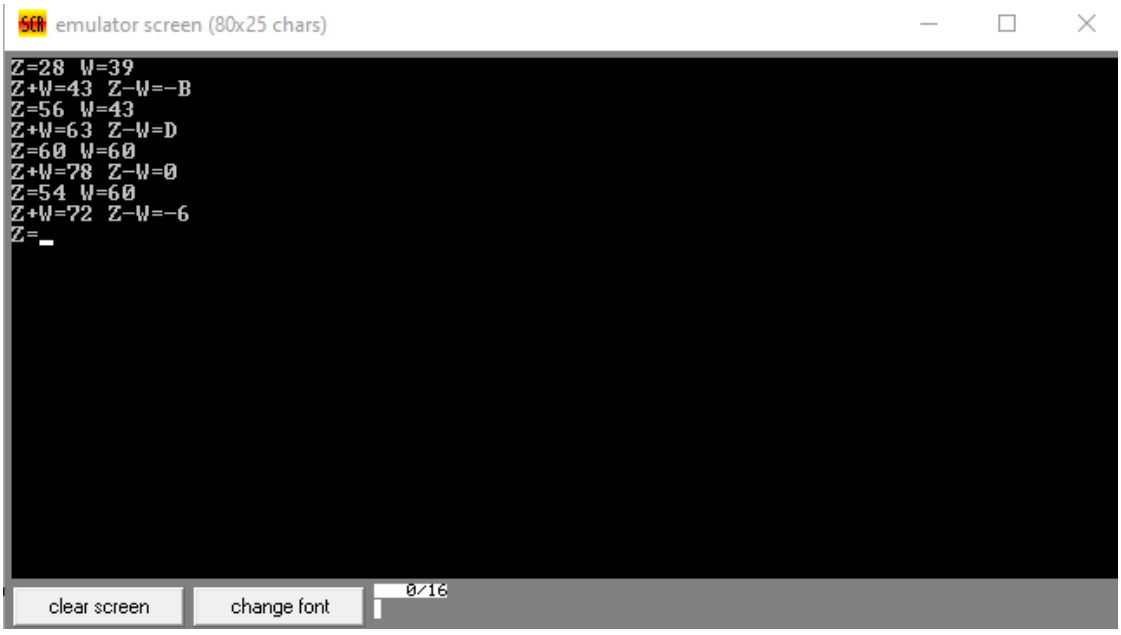
```
DATA SEGMENT
    TEN DB DUP(10)  ; Store number ten in memmory in order to use it for multipli
cations
DATA ENDS

CODE SEGMENT
ASSUME CS:CODE, DS:DATA
MAIN PROC FAR
    START:
        MOV AX, DATA
        MOV DS, AX

        MOV DL,5AH ; Print 'Z'
        CALL PRINT
        MOV DL,3DH ; Print '='
        CALL PRINT

        CALL READ_AND_PRINT ;Read and print the decades of Z
        SUB AL,30H ; Convert from ASCII to decimal
        MUL TEN  ; Multiply the number read by 10 to construct the number Z
        MOV BL,AL
        CALL READ_AND_PRINT ;Read and print least significant decimal digit of nu
mber Z
        SUB AL,30H ; Convert from ASCII to decimal
        ADD BL,AL ; Z in BL

        MOV DL,20H
        CALL PRINT

        MOV DL,57H ; Print 'W'
        CALL PRINT
        MOV DL,3DH ; Print '='
        CALL PRINT

        CALL READ_AND_PRINT ;Read and print the decades of W
        SUB AL,30H ; Convert from ASCII to decimal
        MUL TEN  ; Multiply the number read by 10 to construct the number W
        MOV BH,AL
        CALL READ_AND_PRINT ;Read and print least significant decimal digit of nu
mber Z
        SUB AL,30H ; Convert from ASCII to decimal
```

```
ADD BH, AL ; W in BH

MOV DL, 0AH ; End of line
MOV AH,2
INT 21H
MOV DL, 0DH ; Return cursor to the start of the line
MOV AH,2
INT 21H

MOV DL,5AH ; Print 'Z'
CALL PRINT
MOV DL,2BH ; Print '+'
CALL PRINT
MOV DL,57H ; Print 'W'
CALL PRINT
MOV DL,3DH ; Print '='
CALL PRINT

MOV AX,0
MOV AL, BL ; Add Z to initially 0 AL
ADD AL, BH ; Then add W to it
CALL PRINT_HEX ; Print the result

MOV DL,20H ; Print ' '
CALL PRINT

MOV DL,5AH ; Print 'Z'
CALL PRINT
MOV DL,2DH ; Print '-'
CALL PRINT
MOV DL,57H ; Print 'W'
CALL PRINT
MOV DL,3DH ; Print '='
CALL PRINT

MOV AH,0
MOV AL,BL ; Move Z in AL
SUB AL,BH ; Subtract W from Z
JNC SKIP ; If result not negative then print it
MOV DL,2DH ;else print a '-' in front
CALL PRINT
MOV AH,0 ;and calculate the opposite by moving
MOV AL,BH ; W in AL
SUB AL,BL ; and subtracting Z from W
```

```
    SKIP:
        CALL PRINT_HEX ; Print the result

        MOV DL, 0AH ; End of line
        MOV AH,2
        INT 21H
        MOV DL, 0DH ; Return cursor to the start of the line
        MOV AH,2
        INT 21H

        JMP START ; Repeat waiting for next input
MAIN ENDP

READ_AND_PRINT PROC NEAR
IGNORE:
    MOV AH,8
    INT 21H
    CMP AL,30H ;Check if the number is valid. If not read another one
    JL IGNORE
    CMP AL,39H
    JG IGNORE

    MOV DL,AL ; If it is valid print it and return
    MOV AH,2
    INT 21H

    RET
READ_AND_PRINT ENDP

PRINT PROC NEAR
    MOV AH,2
    INT 21H
    RET
PRINT ENDP

PRINT_HEX PROC NEAR ; This routine prints the number contained in AX in hexadecim
al form
    PUSH DX ; Save contents of DX because that register will be used
    PUSH BX ; Save contents of BX because that register will be used
    MOV CX,0 ;Digit counter on CX
    HEX_DIGIT:
        MOV DX,0
        MOV BX,16D ;Divide by 16 to get the last digit as the remainder
        DIV BX
        PUSH DX ; and push it into the stack
```

```
        INC CX ; increase the digit counter
        CMP AX,0 ; while the quotient is not zero (there are more digits) repeat
        JNE HEX_DIGIT

        PRINT_HEXADECIMAL_NUMBER: ;Once all digits are in the stack print them on
e by one
            POP DX
            CMP DX,9
            JG PRINT_HEX_LETTER ; If the digit is between 10 and 15 print it as a
 letter
            ADD DL,30H
        OUTPUT:
            MOV AH,2
            INT 21H
            LOOP PRINT_HEXADECIMAL_NUMBER
        POP BX ; Restore contents of BX
        POP DX ; Restore contents of DX
        RET

        PRINT_HEX_LETTER:
            ADD DL,37H
            JMP OUTPUT

PRINT_HEX ENDP

    CODE ENDS
END MAIN
```

Και η έξοδος του προγράμματος για διάφορες εισόδους:



3η Άσκηση:

```
DATA SEGMENT
    SIXTEEN DB DUP(16)  ; Store number sixteen in memmory in order to use it for
multiplications
DATA ENDS

CODE SEGMENT
ASSUME CS:CODE, DS:DATA
MAIN PROC FAR
    START:
        MOV AX,DATA
        MOV DS,AX
        CALL HEX_KEYB ;Read the most significant digit
        MOV AH, 0
        MUL SIXTEEN ;Multiply with 16*16 to start forming the 3digit hexadecimal
number
        MUL SIXTEEN
        MOV BX, AX ; Number will be stored in BX

        CALL HEX_KEYB ;Read the next digit
        MOV AH, 0
        MUL SIXTEEN ;Multiply with 16 following the same logic
        ADD BX, AX ; Add to the number

        CALL HEX_KEYB ;Read the next digit
        MOV AH, 0
        ADD BX, AX ; Add to the number

        MOV AX, BX ; Number in AX in order to print it in hexadecimal form
        CALL PRINT_HEX

        MOV DL,3DH ; Print '='
        CALL PRINT

        MOV AX, BX ; Number in AX in order to print it in decimal form
        CALL PRINT_DEC

        MOV DL,3DH ; Print '='
        CALL PRINT

        MOV AX, BX ; Number in AX in order to print it in octal form
        CALL PRINT_OCT

        MOV DL,3DH ; Print '='
```

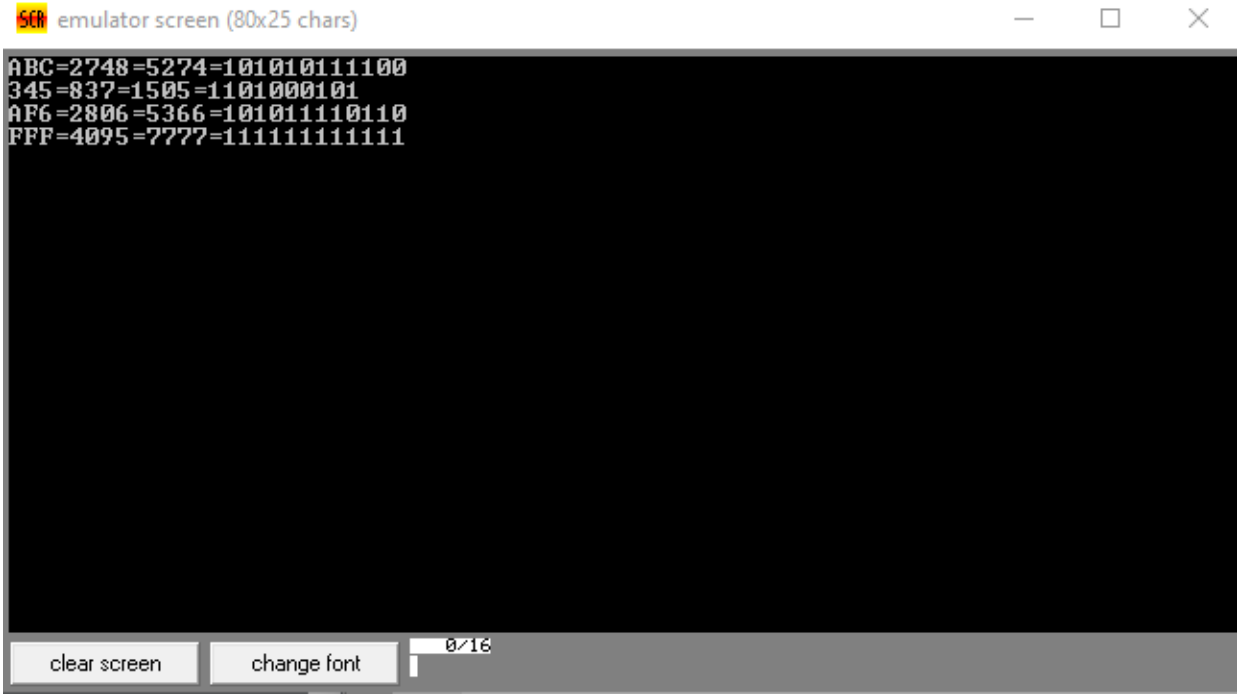
```
        CALL PRINT

        MOV AX, BX ; Number in AX in order to print it in binary form
        CALL PRINT_BIN

        MOV DL, 0AH ; End of line
        MOV AH,2
        INT 21H
        MOV DL, 0DH ; Return cursor to the start of the line
        MOV AH,2
        INT 21H

        JMP START ; Repeat for next input
MAIN ENDP

PRINT_HEX PROC NEAR ; This routine prints the number contained in AX in hexadecim
al form
    PUSH DX ; Save contents of DX because that register will be used
    PUSH BX ; Save contents of BX because that register will be used
    MOV CX,0 ;Digit counter on CX
    HEX_DIGIT:
        MOV DX,0
        MOV BX,16D ;Divide by 16 to get the last digit as the remainder
        DIV BX
        PUSH DX ; and push it into the stack
        INC CX ; increase the digit counter
        CMP AX,0 ; while the quotient is not zero (there are more digits) repeat
        JNE HEX_DIGIT

        PRINT_HEXADECIMAL_NUMBER: ;Once all digits are in the stack print them on
e by one
            POP DX
            CMP DX,9
            JG PRINT_HEX_LETTER ; If the digit is between 10 and 15 print it as a
 letter
            ADD DL,30H
        OUTPUT:
            MOV AH,2
            INT 21H
            LOOP PRINT_HEXADECIMAL_NUMBER
        POP BX ; Restore contents of BX
        POP DX ; Restore contents of DX
        RET

        PRINT_HEX_LETTER:
```

```
            ADD DL,37H
            JMP OUTPUT

PRINT_HEX ENDP

PRINT_DEC PROC NEAR ; This routine prints the number contained in AX in decimal f
orm
    PUSH DX ; Save contents of DX because that register will be used
    PUSH BX ; Save contents of BX because that register will be used
    MOV CX,0 ;Digit counter on CX
    DEC_DIGIT:
        MOV DX,0
        MOV BX,10D ;Divide by 10 to get the last digit as the remainder
        DIV BX
        PUSH DX ; and push it into the stack
        INC CX ; increase the digit counter
        CMP AX,0 ; while the quotient is not zero (there are more digits) repeat
        JNE DEC_DIGIT

    PRINT_DECIMAL_NUMBER: ;Once all digits are in the stack print them one by one
        POP DX
        ADD DL,30H
        MOV AH,2
        INT 21H
        LOOP PRINT_DECIMAL_NUMBER
        POP BX ; Restore contents of BX
        POP DX ; Restore contents of DX
        RET
PRINT_DEC ENDP

PRINT_OCT PROC NEAR ; This routine prints the number contained in AX in octal for
m
    PUSH DX ; Save contents of DX because that register will be used
    PUSH BX ; Save contents of BX because that register will be used
    MOV CX,0 ;Digit counter on CX
    OCT_DIGIT:
        MOV DX,0
        MOV BX,8D ;Divide by 8 to get the last digit as the remainder
        DIV BX
        PUSH DX ; and push it into the stack
        INC CX ; increase the digit counter
        CMP AX,0 ; while the quotient is not zero (there are more digits) repeat
        JNE OCT_DIGIT

    PRINT_OCTAL_NUMBER: ;Once all digits are in the stack print them one by one
```

```
        POP DX
        ADD DL,30H
        MOV AH,2
        INT 21H
        LOOP PRINT_OCTAL_NUMBER
        POP BX ; Restore contents of BX
        POP DX ; Restore contents of DX
        RET
PRINT_OCT ENDP

PRINT_BIN PROC NEAR ; This routine prints the number contained in AX in binary fo
rm
    PUSH DX ; Save contents of DX because that register will be used
    PUSH BX ; Save contents of BX because that register will be used
    MOV CX,0 ;Digit counter on CX
    BIN_DIGIT:
        MOV DX,0
        MOV BX,2D ;Divide by 2 to get the last digit as the remainder
        DIV BX
        PUSH DX ; and push it into the stack
        INC CX ; increase the digit counter
        CMP AX,0 ; while the quotient is not zero (there are more digits) repeat
        JNE BIN_DIGIT

    PRINT_BINARY_NUMBER: ;Once all digits are in the stack print them one by one
        POP DX
        ADD DL,30H
        MOV AH,2
        INT 21H
        LOOP PRINT_BINARY_NUMBER
        POP BX ; Restore contents of BX
        POP DX ; Restore contents of DX
        RET
PRINT_BIN ENDP

HEX_KEYB PROC NEAR ; Routine reads a hex digit and returns it as binary in AL reg
ister (Page 21 mP11_80x86_programs.pdf)
    IGNORE:

        MOV AH,8; Read from keyboard
        INT 21H
        CMP AL, 'T' ; If the character T is pressed exit the program and return c
ontrol to the operating system
        JE EXIT
        CMP AL,30H ; Check if the character read is a hex digit
```

```
        JL IGNORE ; If not ignore it and read another
        CMP AL,39H
        JG ADDR1
        SUB AL,30H ; Extract the actual number converting the ASCII code ('0'=30)
        JMP ADDR2
    ADDR1:
        CMP AL,'A'
        JL IGNORE
        CMP AL,'F'
        JG IGNORE
        SUB AL,37H ; Convert HEX ASCII to a pure number ('A'=41, 41H-37H=0AH=10D)
    ADDR2:
        RET
    EXIT:
        MOV AX, 4C00H ; Return control to the operating system
        INT 21H

HEX_KEYB ENDP

PRINT PROC NEAR
    MOV AH,2
    INT 21H
    RET
PRINT ENDP

    CODE ENDS
END MAIN
```

Και η έξοδος του προγράμματος για διάφορες εισόδους:



4η Άσκηση:

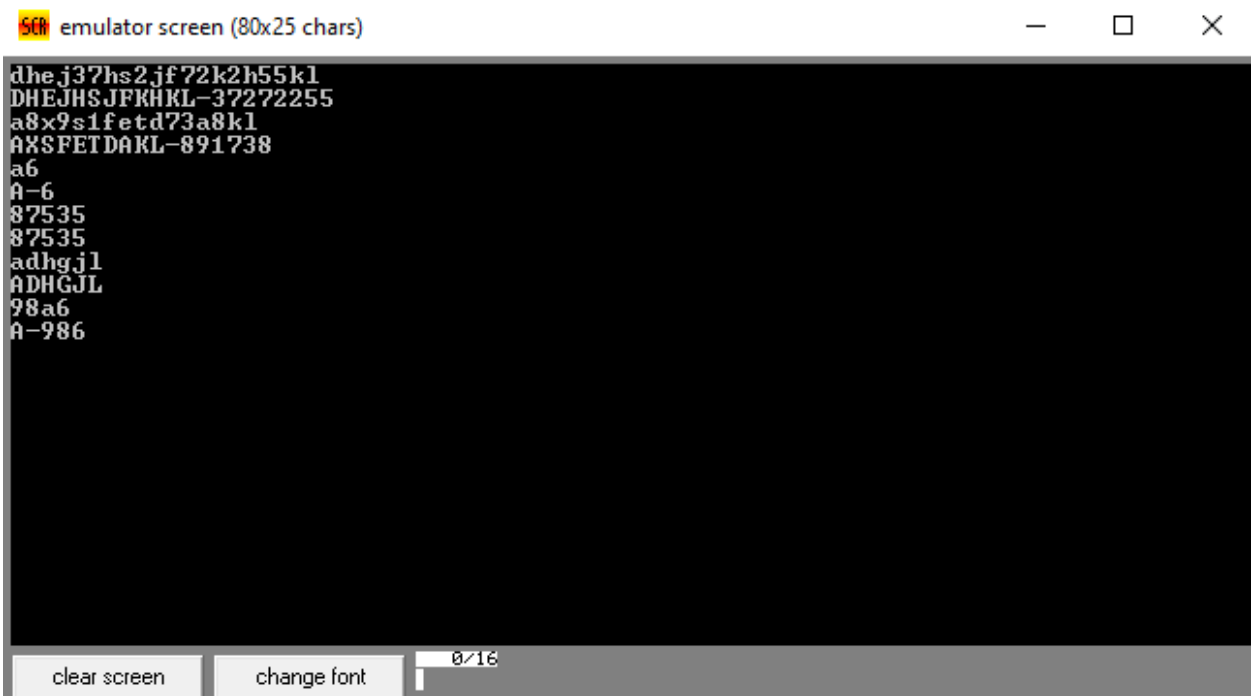
```
DATA SEGMENT
    LETTERS DB 20 DUP(?) ; Table to store the letters read
    NUMBERS DB 20 DUP(?) ; Table to store the numbers read

    LETTER_INDEX DW DUP(0) ; Index to iterate through letter table
    NUMBER_INDEX DW DUP(0) ; Index to iterate through number table
DATA ENDS

CODE SEGMENT
ASSUME CS:CODE, DS:DATA
MAIN PROC FAR
    START:
        MOV AX,DATA
        MOV DS,AX

        MOV CX,20 ; A maximum of 20 characters will be read
        MOV LETTER_INDEX,0 ; Set the index of the letter table to 0
        MOV NUMBER_INDEX,0 ; Set the index of the number table to 0
    INPUT:
        CALL READ ; Read characters
        LOOP INPUT ; Loop 20 times or until ENTER is pressed and the READ routine
 breaks the loop

        MOV DL, 0AH ; End of line
        MOV AH,2
        INT 21H
        MOV DL, 0DH ;  Move cursor to the start of the line
        MOV AH,2
        INT 21H

    CALL PRINT_LETTERS ; Print all the letters read capitalized
    CALL PRINT_DASH ; Print a dash between letters and numbers (will only be prin
ted if both letters and numbers were read)
    CALL PRINT_NUMBERS ; Print all the numbers read


    MOV DL, 0AH ; End of line
    MOV AH,2
    INT 21H
    MOV DL, 0DH ; Move cursor to the start of the line
    MOV AH,2
    INT 21H
    JMP START ; Restart and wait next input
```

```
MAIN ENDP

READ PROC NEAR ; Routine for reading a lower-case letter or number from keyboard.
    PUSH DX ; Temporarily store DX
    MOV BX,0
IGNORE:
    MOV AH,8; Διάβασε τον χαρακτήρα από το πληκτρολόγιο
    INT 21H
    CMP AL, 3DH ; Check if the character read is '=' in which case stop the progr
am and return control to the OS
    JZ EQUAL_KEY
    CMP AL, 0DH ; Check if ENTER was pressed in which case stop reading character
s
    JZ ENTER_KEY
    CMP AL,30H ; Check if character is a digit
    JL IGNORE ; If not, ignore it and read another
    CMP AL,39H
    JG LETTER
    JMP NUMBER
LETTER:
    CMP AL,'a' ; Check if the character read is a valid lower-case letter
    JL IGNORE
    CMP AL,'z'
    JG IGNORE
    MOV DL, AL
    CALL PRINT ; If yes print it
    MOV DI,LETTER_INDEX
    MOV LETTERS[DI], AL ; Store it in the letter table in the position where LETT
ER_INDEX is pointing
    INC LETTER_INDEX ; Increase the LETTER_INDEX
    POP DX
    RET
NUMBER:
    MOV DL,AL
    CALL PRINT
    MOV DI, NUMBER_INDEX
    MOV NUMBERS[DI], AL ; If it is a number store it in the number table in the p
osition where NUMBER_INDEX is pointing
    INC NUMBER_INDEX ; Increase the NUMBER_INDEX
    POP DX ; Restore contents of DX
    RET
ENTER_KEY:
    POP DX ; Restore contents of DX
    MOV CX,1 ; Set CX to 1 in order to break the loop that reads characters
```

```
    RET
EQUAL_KEY:
    MOV AX, 4C00H ; Return control to the OS
    INT 21H

READ ENDP

PRINT PROC NEAR
    MOV AH,2
    INT 21H
    RET
PRINT ENDP

PRINT_LETTERS PROC NEAR
    MOV CX, LETTER_INDEX ; Move LETTER_INDEX to CX in order to print all the lett
ers using loop
    CMP CX,0 ; If the LETTER_INDEX is 0 (there are no letters) return
    JE END_LETTERS
    MOV DI,0
    PRINT_LETTER:
        MOV DL, LETTERS[DI] ; else iterate through the letter table and move curr
ent letter to DL
        SUB DL, 20H ; Capitalize the letter
        CALL PRINT ;and print it
        INC DI
        LOOP PRINT_LETTER ; Continue until all letters read are printed
    END_LETTERS:
        RET
PRINT_LETTERS ENDP

PRINT_DASH PROC NEAR ; Print the '-' only if both numbers and letters are read
    CMP LETTER_INDEX, 0 ; If the LETTER_INDEX is 0 (there are no letters) return
without printing '-'
    JE NO_DASH
    CMP NUMBER_INDEX, 0 ; If the NUMBER_INDEX is 0 (there are no numbers) return
without printing '-'
    JE NO_DASH

    MOV DL,2DH ; If both letters and numbers were read print the '-'
    MOV AH,2
    INT 21H

    NO_DASH:
        RET
PRINT_DASH ENDP
```

```
PRINT_NUMBERS PROC NEAR
    MOV CX, NUMBER_INDEX ; Move NUMBER_INDEX to CX in order to print all the numb
ers using loop
    CMP CX,0 ; If the NUMBER_INDEX is 0 (there are no numbers) return
    JE END_NUMBERS
    MOV DI,0
    PRINT_NUM:
        MOV DL, NUMBERS[DI] ; else iterate through the number table and move curr
ent number to DL
        CALL PRINT ;and print it
        INC DI
        LOOP PRINT_NUM ; Continue until all numbers read are printed
    END_NUMBERS:
        RET
PRINT_NUMBERS ENDP
    CODE ENDS
END MAIN
```

Και η έξοδος του προγράμματος για διάφορες εισόδους:



Για εισόδους με λιγότερους από 20 χαρακτήρες πατήσαμε το ENTER για να γίνει η εκτύπωση όπως ζητείται από την εκφώνηση.

5η Άσκηση:

```
DATA SEGMENT
    START_MSG DB "START(Y,N) :$" ; Message to be printed at the start of the prog
ram
    ERROR_MSG DB "ERROR$" ; Error message for temperatures above 1200 degrees
    SIXTEEN DB DUP(16) ; Store number sixteen in memmory in order to use it for m
ultiplications
DATA ENDS

CODE SEGMENT
ASSUME CS:CODE, DS:DATA
MAIN PROC FAR

        MOV AX,DATA
        MOV DS,AX

        LEA DX,START_MSG  ; Print the start message
        MOV AH,09H
        INT 21H
        CALL READ_YN

    START:
        MOV DL, 0AH ; New line
        MOV AH,2
        INT 21H
        MOV DL, 0DH ; Move cursor to the start of the line
        MOV AH,2
        INT 21H

        CALL HEX_KEYB ;Read the most significant digit
        MOV AH, 0
        MUL SIXTEEN ;Multiply with 16*16 to start forming the 3digit hexadecimal
number
        MUL SIXTEEN
        MOV BX, AX ; Add to the number

        CALL HEX_KEYB ;Read the next digit
        MOV AH, 0
        MUL SIXTEEN ;Multiply with 16 following the same logic
        ADD BX, AX ; Add to the number

        CALL HEX_KEYB ;Read the least significant digit
        MOV AH, 0
        ADD BX, AX ; Add to the number
```

```
        MOV AH,2
        MOV DL,3AH ;Print a ":" character next to the hexadecimal number you read
        INT 21H
        MOV DL,09H ;and then a tab next to it
        INT 21H

        ;MOV DX,0
        MOV AX, BX ;Move the number to AX to use it in the following multiplicati
ons and divisions
        CMP AX, 2048 ;If the given number is below 2048 we are in the first part
of the temperature-voltage curve
        JL LINE_PART1
        CMP AX, 3072 ;IF it is between 2048 and 3071 we are in the second part
        JL LINE_PART2

    ERROR:      ; Else the temperature is above 1200 degrees and we print an erro
r message
        LEA DX,ERROR_MSG ;Load the address of the message and print it until you
find '$'
        MOV AH,09H
        INT 21H
        JMP START ;Then get a new reading

    LINE_PART1: ; T=reading*(4/4095)*(400/2) => T=reading*(800/4095)
        MOV BX, 8000 ; Multiply with 8000 instead of 800 to keep the first digit
of the fractional part
        MUL BX ;Result is stored in DX:AX
        MOV BX, 4095 ;Divide with 4095
        DIV BX ; Quotient goes in AX and remainder in DX
        MOV DX,0 ;The remainder is of no use to us (all the digits we care about
are in the quotient)
        MOV BX, 10 ; so we make DX 0 to divide the quotient with 10
        DIV BX ; And now the integer part is in AX and the first digit of the rem
ainder in DX

        CALL PRINT_RESULT ;Print the result
        JMP START ; Go to the start of the program to get a new reading

    LINE_PART2: ; T=(reading*(4/4095)-2)*800+400 => T=reading*(3200/4095)-1200
        MOV BX, 32000 ;Following the same logic as before we multiply with 32000
instead of 3200
        MUL BX
        MOV BX, 4095 ;Divide by 4095
        DIV BX
```

```
        SUB AX, 12000 ;Sub 12000 instead of 1200
        MOV DX,0
        MOV BX, 10 ;Divide by 10 to seperate integer part from fractional digit
        DIV BX

        CALL PRINT_RESULT ;Print the result
        JMP START ; Go to the start of the program to get a new reading

MAIN ENDP


HEX_KEYB PROC NEAR ; Routine reads a hex digit and returns it as binary in AL reg
ister (Page 21 mP11_80x86_programs.pdf)
    IGNORE:

        MOV AH,8; Read from keyboard
        INT 21H
        CMP AL, 'N' ; If the character N is pressed exit the program and return c
ontrol to the operating system
        JE EXIT
        CMP AL,30H ; Check if the character read is a hex digit
        JL IGNORE ; If not ignore it and read another
        CMP AL,39H
        JG ADDR1
        MOV AH,2 ;Print the hex digit read on the screen
        MOV DL,AL
        INT 21H
        SUB AL,30H ; Extract the actual number converting the ASCII code ('0'=30)
        JMP ADDR2
    ADDR1:
        CMP AL,'A'
        JL IGNORE
        CMP AL,'F'
        JG IGNORE
        MOV AH,2
        MOV DL,AL
        INT 21H
        SUB AL,37H ; Convert HEX ASCII to a pure number ('A'=41, 41H-37H=0AH=10D)
    ADDR2:
        RET
    EXIT:
        MOV AX, 4C00H ; Return control to the operating system
        INT 21H

HEX_KEYB ENDP
```

```
READ_YN PROC NEAR ; This routine reads 'Y' or 'N' and determines whether the prog
ram will continue or exit
    IGNORE1:
        MOV AH,8; Read a character from the keyboard
        INT 21H
        CMP AL, 'N' ; If it is the character 'N' exit the program
        JE EXIT1
        CMP AL, 'Y' ; If it is neither 'N' or 'Y' read a new character
        JNZ IGNORE1
        MOV DL,AL ; If it is 'Y' print it and return to the main program
        MOV AH,2
        INT 21H
        RET
    EXIT1:
        MOV AX, 4C00H ; Return control to the operating system
        INT 21H
READ_YN ENDP

PRINT_DEC PROC NEAR ; This routine prints the number contained in AX in decimal f
orm
    PUSH DX ; Save contents of DX because that register will be used
    PUSH BX ; Save contents of BX because that register will be used
    MOV CX,0 ;Digit counter on CX
    DEC_DIGIT:
        MOV DX,0
        MOV BX,10D ;Divide by 10 to get the last digit as the remainder
        DIV BX
        PUSH DX ; and push it into the stack
        INC CX ; increase the digit counter
        CMP AX,0 ; while the quotient is not zero (there are more digits) repeat
        JNE DEC_DIGIT

    PRINT_DECIMAL_NUMBER: ;Once all digits are in the stack print them one by one
        POP DX
        ADD DL,30H
        MOV AH,2
        INT 21H
        LOOP PRINT_DECIMAL_NUMBER
        POP BX ; Restore contents of BX
        POP DX ; Restore contents of DX
        RET
PRINT_DEC ENDP

PRINT_RESULT PROC NEAR
```

```
    CALL PRINT_DEC ;Print the integer part
    MOV BX,DX ;Save contents of DX because we use the register in the later print
    MOV DL,46 ;Print the "."
    MOV AH,2
    INT 21H
    MOV AX,BX ;Move the fractional digit on AX and print it
    CALL PRINT_DEC
    RET
PRINT_RESULT ENDP
CODE ENDS
END MAIN
```

Και η έξοδος του προγράμματος για διάφορες εισόδους: