



*Αριάδνη Καζδάγλη el1838, Βικέντιος Βιτάλης el18803*

*5<sup>η</sup> Σειρά Ασκήσεων Εργαστηρίου*

*Συστήματα Μικροϋπολογιστών*

*Ακαδημαϊκό Έτος 2020-2021*



Αρχικά δίνεται το περιεχόμενο του αρχείου μακροεντολών που χρησιμοποιήθηκε για τη μεταγλώττιση των προγραμμάτων.

macros.asm / emu8086 Code

```

01 ; Character display
02 PRINTCH MACRO CHAR
03     PUSH AX
04     PUSH DX
05     MOV DL, CHAR
06     MOV AH, 2
07     INT 21H
08     POP DX
09     POP AX
10 ENDM
11
12 ;Display of string
13 PRINTSTR MACRO STRING
14     PUSH AX
15     PUSH DX
16     MOV DX, OFFSET STRING
17     MOV AH, 9
18     INT 21H
19     POP DX
20     POP AX
21 ENDM
22
23 ;Change of line
24 PRINTLN MACRO
25     PUSH AX
26     PUSH DX
27     MOV DL, 13
28     MOV AH, 2
29     INT 21H
30     MOV DL, 10
31     MOV AH, 2
32     INT 21H
33     POP DX
34     POP AX
35 ENDM
36
37 ;Esxi
38 PRINTTAB MACRO
39     PUSH AX
40     PUSH DX
41     MOV DL, 9
42     MOV AH, 2
43     INT 21H
44     POP DX
45     POP AX
46 ENDM
47
48 ;Character instertion
49 READCH MACRO
50     MOV AH, 8
51     INT 21H
52 ENDM
53
54 ;Insertion and character display
55 READNPRINTCH MACRO
56     MOV AH, 1
57     INT 21H
58 ENDM
59
60 ; Output
61 EXIT MACRO
62     MOV AX, 4C00H
63     INT 21H
64 ENDM
65

```


```

0066 ; Some extra macros
0067 ;
0068 ;
0069 READ MACRO
0070     MOV AH,01
0071     INT 21H
0072 ENDM
0073 ;
0074 ;
0075 PRINT MACRO CHAR
0076     PUSH AX
0077     PUSH DX
0078     MOV DL,CHAR
0079     MOV AH,2
0080     INT 21H
0081     POP DX
0082     POP AX
0083 ENDM
0084 ;
0085 ;
0086 PRINT_STRING MACRO STRING
0087     PUSH AX
0088     PUSH DX
0089     MOV DX,OFFSET STRING
0090     MOV AH,9
0091     INT 21H
0092     POP DX
0093     POP AX
0094 ENDM
0095 ;
0096 ;
0097 READ_WITHOUT_PRINT MACRO
0098     MOV AH,08
0099     INT 21H
0100 ENDM
0101 ;
0102 ;

```

### 1η Άσκηση

Το πρόγραμμα αρχικά αποθηκεύει τους αριθμούς 128, 127, 126, ..., 3, 2, 1 σε 128 διαδοχικές 8-bit θέσεις μνήμης (δομή TABLE). Στη συνέχεια σαρώνει τον πίνακα TABLE για να βρει τους περιττούς αριθμούς (διαιρώντας τους με το 2 – μεταβλητή TWO), τους οποίους αθροίζει και μετρά και βρίσκει το μέγιστο και το ελάχιστο των αριθμών συγκρίνοντάς τους σειριακά με τον 1ο και τον τελευταίο αριθμό της σειράς αντίστοιχα. Τέλος, τυπώνει το ακέραιο μέρος του μέσου όρου (16 bit) σε δεκαδική μορφή και σε νέα γραμμή το μέγιστο και ελάχιστο με ένα κενό μεταξύ τους, σε δεκαεξαδική μορφή. Με την εκτέλεση του προγράμματος προκύπτει:



```

emulator screen (80x25 chars)
65
1h 80h

```

Ask1.asm / emu8086 Code

```

001 ; Microcomputer Systems - Roh Y [6th Semester]
002 ; 5th Series of Exercises
003 ; Kazdagli Ariadni - 03118838
004 ; Vitalis Vikentios - 03118803
005
006
007 INCLUDE macros.asm
008
009
010
011 DATA_SEG SEGMENT
012 TABLE DB 128 DUP(?)
013 AVERAGE DB ?
014 MIN DB ?
015 MAX DB ?
016 NEWLINE DB 0AH,0DH,'$'
017 DATA_SEG ENDS
018
019
020
021 CODE_SEG SEGMENT
022 ASSUME CS:CODE_SEG, DS:DATA_SEG
023
024
025
026 MAIN PROC FAR
027
028     MOV AX,DATA_SEG ; important initialization
029     MOV DS,AX
030
031
032     ; in this part data is stored
033     MOV AL,128 ; initialize counter
034     MOV DI,0 ; initialize index
035 STORE:
036     MOV [TABLE+DI],AL
037     DEC AL
038     INC DI
039     CMP DI,128
040     JNE STORE
041
042
043     ; in this part average is calculated
044     MOV DX,0 ; initialize counter to store the sum
045     MOV DI,0 ; initialize index to 0
046     MOV AX,1 ; temporary register
047 SUM:
048     MOV AL,[TABLE+DI]
049     ADD DX,AX
050 CONT:
051     ADD DI,2
052     CMP DI,128
053     JNE SUM
054 END_COUNT: ; sum of all peritton in DX
055     MOV AX,DX
056     MOV DX,1
057     MOV CX,64
058     DIV CX ; divide the sum by 64
059     MOV DX,AX
060     MOV AVERAGE,DL ; average is now calculated and stored
061
062
063     ; in this part min and max are calculated
064     MOV MAX,1 ; initializations
065     MOV MIN,128
066     MOV DI,0
067 MIN_MAX:
068     MOV AL,[TABLE+DI]
069     CMP MIN,AL ; MIN =< AL ?
070     JNA GO_MAX ; if yes, then go see for max
071     MOV MIN,AL ; else update minimum data
072 GO_MAX:
073     CMP MAX,AL ; MAX >= AL ?
074     JAE ITS_MAX ; if yes, then continue
075     MOV MAX,AL ; else update maximum data
076

```

```

077 ITS_MAX:
078     INC DI
079     CMP DI,128
080     JNE MIN_MAX
081
082
083
084     ; in this part, demanded data is printed
085     MOV CX,1
086     MOV AL,AVERAGE
087     MOV DL,10
088
089 LD:
090     MOV AH,0
091     DIV DL
092     PUSH AX
093     CMP AL,0
094     JE PRINT_AVERAGE
095     INC CX
096     JMP LD
097
098 PRINT_AVERAGE:
099     POP AX
100     MOV AL,AH
101     MOV AH,0
102     ADD AX,'0'
103     PRINT AL
104     LOOP PRINT_AVERAGE
105
106     PRINTLN NEWLINE
107
108
109
110     MOV AL,MIN                ; PRINT MINIMUM DATA
111     SAR AL,4
112     AND AL,0FH                ; isolate 4 MSB
113     CMP AL,0
114     JE NEXT_DIGIT
115     ADD AL,30H                ; ASCII code it
116     CMP AL,39H
117     JLE OK2
118     ADD AL,07H                ; if it's a letter, fix ASCII
119 OK2: PRINT AL                ; print the first hex digit
120
121 NEXT_DIGIT:
122     MOV AL,MIN
123     AND AL,0FH                ; isolate 4 LSB
124     ADD AL,30H                ; ASCII code it
125     CMP AL,39H
126     JLE OK3
127     ADD AL,07H                ; if it's a letter, fix ASCII
128 OK3: PRINT AL                ; print the second hex digit
129     PRINT 'h'
130
131     PRINT ' '
132
133
134     MOV AL,MAX                ; PRINT MAXIMUM DATA
135     SAR AL,4
136     AND AL,0FH                ; isolate 4 MSB
137     ADD AL,30H                ; ASCII code it
138     CMP AL,39H
139     JLE OK4
140     ADD AL,07H                ; if it's a letter, fix ASCII
141 OK4: PRINT AL                ; print the first hex digit
142
143     MOV AL,MAX
144     AND AL,0FH                ; isolate 4 LSB
145     ADD AL,30H                ; ASCII code it
146     CMP AL,39H
147     JLE OK5
148     ADD AL,07H                ; if it's a letter, fix ASCII
149 OK5: PRINT AL                ; print the second hex digit
150
151     PRINT 'h'
152
153
154
155
156     EXIT
157
158 MAIN     ENDP
159
160 CODE_SEG ENDS
161     END MAIN

```

## 2η Άσκηση

Το πρόγραμμα δέχεται 2 διψήφιους δεκαδικούς αριθμούς, Z και W, από το πληκτρολόγιο και τους εμφανίζει στην οθόνη. Στη συνέχεια υπολογίζει το άθροισμα Z+W και τη διαφορά τους Z-W και τα εμφανίζει σε δεκαεξαδική μορφή στην επόμενη γραμμή. Το πρόγραμμα είναι συνεχούς λειτουργίας. Παρακάτω παραθέτουμε μερικά παραδείγματα εκτέλεσης:

```

emulator screen (80x37 chars)
Z+W=43 Z-W=-B
Z=01 W=99
Z+W=64 Z-W=-62
Z=81 W=99
Z+W=B4 Z-W=-12
Z=00 W=99
Z+W=63 Z-W=-63
Z=99 W=00
Z+W=63 Z-W=63
Z=00 W=00
Z+W=0 Z-W=0
Z=45 W=80
Z+W=7D Z-W=-23
Z=80 W=45
Z+W=7D Z-W=23
Z=
    
```

ask2.asm / emu8086 Code

```

0001 INCLUDE macros.asm
0002
0003 DATA SEGMENT
0004 MSGZ DB "Z=$"
0005 MSGW DB "W=$"
0006 MSGSUM DB "Z+W=$"
0007 MSGSUB DB "Z-W=$"
0008 MSGMINUS DB "Z-W=-$"
0009 Z DB 0
0010 W DB 0
0011 TEN DB DUP<10> ;Gia tis dekades
0012 DATA ENDS
0013
0014 CODE SEGMENT
0015 ASSUME CS:CODE, DS:DATA
0016
0017 MAIN PROC FAR
0018     MOV AX,DATA
0019     MOV DS,AX
0020
0021     START: ;Construction, display and storage of Z
0022     PRINTSTR MSGZ
0023     CALL READ_DEC_DIGIT ;1st decimal digit
0024     MUL TEN
0025     LEA DI,Z ;Storage of first digit
0026     MOV [DI],AL
0027     CALL READ_DEC_DIGIT ;2nd unit digit
0028     ADD [DI],AL ;Storage of second digit
0029
0030     PRINTCH ' '
0031     ;Construction, display and storage of W
0032     PRINTSTR MSGW
0033     CALL READ_DEC_DIGIT ;1st decimal digit
0034     MUL TEN
0035     LEA DI,W ;Storage of first digit
0036     MOV [DI],AL
0037     CALL READ_DEC_DIGIT ;2nd unit digit
0038     ADD [DI],AL ;Storage of second digit
0039
0040     PRINTLN
0041     ;Summary
0042     MOV AL,[DI] ;W
0043     LEA DI,Z ;Z
0044     ADD AL,[DI] ;Sum
0045     PRINTSTR MSGSUM
0046     CALL PRINT_NUM8_HEX ;Summary display
0047
0048     PRINTCH ' '
0049     ;Subtraction
0050     MOV AL,[DI] ;Z
0051     LEA DI,W ;W
0052     MOV BL,[DI]
0053
0054     CMP AL,BL ;Z>W or W>Z ?
0055     JB MINUS
0056     SUB AL,BL ;if Z>W
0057     PRINTSTR MSGSUB
0058     JMP SHOWSUB
0059
0060     MINUS: ;if Z<W
0061     SUB BL,AL
0062     MOV AL,BL
0063     PRINTSTR MSGMINUS
0064     SHOWSUB:
0065     CALL PRINT_NUM8_HEX ;Subtraction result display
0066     PRINTLN
0067     PRINTLN
0068     JMP START
0069
0070     MAIN ENDP
0071
0072     READ_DEC_DIGIT PROC NEAR
0073     READ:
0074     READCH
0075     CMP AL,48 ;<0 ?
0076     JB READ
0077     CMP AL,57 ;>9 ?
    
```

```

076      JA READ
077      PRINTCH AL
078      SUB AL,48          ;Code ASCII
079      RET
080  READ_DEC_DIGIT ENDP
081      ;;;Print the 8-bit number in hex form in AL register
082  PRINT_NUMS_HEX PROC NEAR
083      MOV DL,AL
084      AND DL,0F0H        ;1st decimal digit
085      MOV CL,4
086      ROR DL,CL
087      CMP DL,0           ;Ignore the first 0
088      JE SKIPZERO
089      CALL PRINT_HEX
090  SKIPZERO:
091      MOV DL,AL
092      AND DL,0FH         ;2nd hex digit
093      CALL PRINT_HEX
094      RET
095  PRINT_NUMS_HEX ENDP
096      ;Print hex digit from DL register
097  PRINT_HEX PROC NEAR
098      CMP DL,9           ;0...9
099      JG LETTER
100      ADD DL,48
101      JMP SHOW
102  LETTER:
103      ADD DL,55          ;A...F
104  SHOW:
105      PRINTCH DL
106      RET
107  PRINT_HEX ENDP
108  CODE ENDS
109  END MAIN
110

```

### 3η Άσκηση

Το πρόγραμμα δέχεται έναν 12-bit δεκαεξαδικό αριθμό από τον BX και τον εμφανίζει σε δεκαδική, οκταδική και δυαδική μορφή. Ανάμεσα στα αποτελέσματα τοποθετεί ένα =, ενώ αν δοθεί ο χαρακτήρας T τερματίζεται. Το πρόγραμμα είναι συνεχούς λειτουργίας. Στο παρακάτω παράδειγμα εκτέλεσης τρέξαμε με τους αριθμούς AAA,BBB,ABC και προέκυψε το εξής αποτέλεσμα:



```

5th emulator screen (80x25 chars)
2730=101010101010=5252
3003=101110111011=5673
2748=101010111100=5274

```

Ask3.asm / emu8086 Code

```

0001 ; Microcomputer Systems - Roh Y [6th Semester]
0002 ; 5th Series of Exercises
0003 ; Kazdagli Ariadni - 03118838
0004 ; Vitalis Vikentios - 03118803
0005
0006
0007 PRINT MACRO CHAR                ;MACRO to print a character
0008     PUSH AX
0009     PUSH DX
0010     MOV DL,CHAR                ;Char must be defined
0011     MOV AH,2                    ;before translation
0012     INT 21H
0013     POP DX
0014     POP AX
0015 ENDM
0016
0017 READ MACRO
0018     MOV AH, 8                    ;MACRO to read from keyboard
0019     INT 21H                      ;The value of ASCII char is returned
0020                                ;through AL
0021 ENDM
0022
0023 NEWLINE MACRO LINEFEED            ;MACRO to print new line
0024     PUSH AX
0025     PUSH DX
0026     MOV AH,09
0027     MOV DX,OFFSET LINEFEED
0028     INT 21H
0029     POP DX
0030     POP AX
0031 ENDM
0032
0033 DATA_SEG SEGMENT
0034     LINEFEED DB 13, 10, "$"
0035 DATA_SEG ENDS
0036
0037 CODE_SEG SEGMENT
0038     ASSUME CS:CODE_SEG, DS:DATA_SEG
0039 MAIN PROC FAR
0040     MOV AX,DATA_SEG
0041     MOV DS,AX
0042
0043 START:
0044     MOV BX,0
0045     CALL HEX_KEYB                ;Get first hex digit
0046     CMP AL,'I'                   ;If it's I then end
0047     JE QUIT                      ;Else store its value to BH
0048     MOV BH,AL                    ;BX will have the 12 bit number
0049     CALL HEX_KEYB                ;Get second hex digit
0050     CMP AL,'I'
0051     JE QUIT
0052     MOV BL,AL                    ;Store its value to BL
0053     MOV CL,4
0054     ROL BL,CL                    ;Rotate the value to 4 MSB
0055     CALL HEX_KEYB                ;Get third hex digit
0056     CMP AL,'I'
0057     JE QUIT
0058     ADD BL,AL                    ;BL has the 2nd and 3rd digit
0059     MOV AX,BX
0060     MOV CX,0
0061     CALL PRINT_DEC
0062     PRINT '='
0063     CALL PRINT_BIN
0064     PRINT '='
0065     CALL PRINT_OCT
0066     NEWLINE LINEFEED
0067     JMP START
0068
0069 QUIT:
0070     HLT
0071 MAIN ENDP
0072
0073 PRINT_OCT PROC NEAR              ;Print 12bit number in octal
0074     PUSH AX
0075     MOV BX,8                     ;We have to divide number with 8
0076     MOV CX,0                     ;Initialise counter

```



```

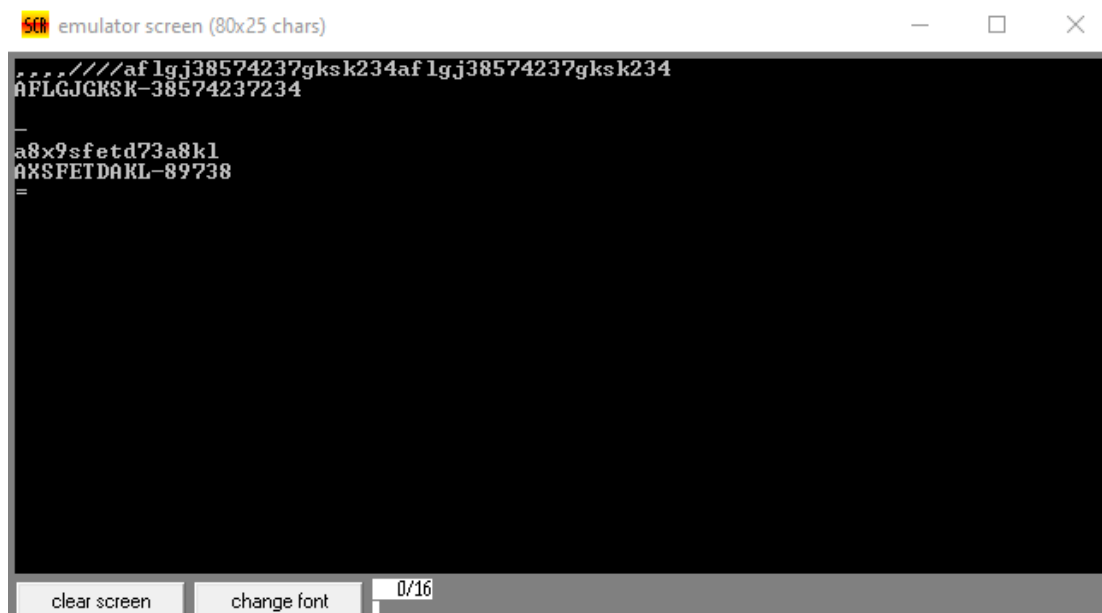
076 GETOCT:
077     MOV DX,0           ;DX will have the remainder of the division
078     DIV BX             ;Divide number with 8
079     PUSH DX            ;Save the remainder to the stack
080     INC CL             ;Increase counter
081     CMP AX,0           ;If quotient is 0 then there isnt any digit left
082     JNE GETOCT
083 PRINTOCT:
084     POP AX             ;Read one digit from the stack
085     ADD AL,48           ;Calculate ASCII code and print the
086     PRINT AL           ;corresponding character on the screen
087     LOOP PRINTOCT
088     POP AX
089     RET
090 PRINT_OCT ENDP
091
092 PRINT_BIN PROC NEAR   ;Print 12bit number in binary
093     PUSH AX
094     MOV BX,2           ;We have to divide number with 2
095     MOV CX,0           ;We follow the same process as above
096     GETBIN:            ;but instead of dividing by 8
097     MOV DX,0           ;we divide number with 2
098     DIV BX
099     PUSH DX
100     INC CL
101     CMP AX,0
102     JNE GETBIN
103 PRINTBIN:
104     POP AX
105     ADD AL,48           ;Calculate ASCII code and print the
106     PRINT AL           ;corresponding character on the screen
107     LOOP PRINTBIN
108     POP AX
109     RET
110 PRINT_BIN ENDP
111
112 PRINT_DEC PROC NEAR   ;Print 12bit number in decimal
113     PUSH AX
114     MOV BX,10          ;We have to divide number with 10
115     MOV CX,0           ;We follow the same process as above
116     GETDEC:            ;but instead of dividing by 2
117     MOV DX,0           ;we divide number with 10
118     DIV BX
119     PUSH DX
120     INC CX
121     CMP AX,0
122     JNE GETDEC
123 PRINTDEC:
124     POP DX
125     ADD DX,48           ;Calculate ASCII code and print
126     PRINT DL           ;the corresponding character on the screen
127     LOOP PRINTDEC
128     POP AX
129     RET
130 PRINT_DEC ENDP
131
132
133
134 HEX_KEYB PROC NEAR    ;Read a hex digit from the keyboard
135                       ;and store it in AL
136 IGNORE:
137     READ              ;Read from keyboard
138     CMP AL,'T'         ;If char is T then end routine
139     JE ADDR2
140     CMP AL,30H         ;Check if char is a digit
141     JL IGNORE          ;If it isn't then wait for a valid input
142     CMP AL,39H         ;Check if char is a digit greater than 9
143     JG ADDR1           ;If it is then check if its A,B,C,D,E or F
144     SUB AL,30H         ;Export the correct number from its ASCII value
145     JMP ADDR2
146
147 ADDR1:
148     CMP AL,'A'         ;Check if char is a valid hex digit
149     JL IGNORE          ;If it isn't then wait for a valid input
150     CMP AL,'F'
151     JG IGNORE
152     SUB AL,37H         ;Export the correct number from its ASCII value
153     RET
154 HEX_KEYB ENDP
155 CODE_SEG ENDS
156 END MAIN

```

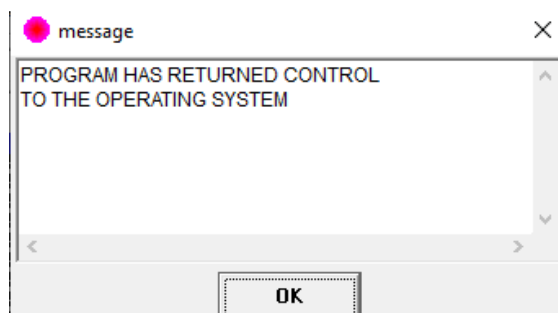
#### 4η Άσκηση

Το πρόγραμμα δέχεται έως 20 πεζούς αγγλικούς (a-z) και αριθμητικούς (0-9) χαρακτήρες από το πληκτρολόγιο και τους εμφανίζει στην οθόνη σε μία γραμμή τον έναν δίπλα στον άλλο. Στην συνέχεια, με τη συμπλήρωση 20 χαρακτήρων ή όταν δοθεί ENTER, εμφανίζει στην επόμενη γραμμή την ίδια σειρά χαρακτήρων με τους αγγλικούς χαρακτήρες κεφαλαίους (A-Z). Το πρόγραμμα είναι συνεχούς λειτουργίας και τερματίζεται αν δοθεί =.

Επίσης ανάμεσα στους χαρακτήρες και στα νούμερα προστίθεται μία παύλα '-' όπως φαίνεται παρακάτω:



Στην περίπτωση που δοθεί '=' παρακάτω φαίνεται το μήνυμα τερματισμού:



Ask4.asm / emu8086 Code

```

0001 ; Microcomputer Systems - Roh Y [6th Semester]
0002 ; 5th Series of Exercises
0003 ; Kazdagli Ariadni - 03118838
0004 ; Vitalis Vikentios - 03118803
0005
0006
0007 INCLUDE macros.asm
0008
0009
0010
0011 DATA_SEG SEGMENT
0012     STRING DB 20 DUP(?)
0013     NEWLINE DB 0AH,0DH,'$'
0014 DATA_SEG ENDS
0015
0016 CODE_SEG SEGMENT
0017     ASSUME CS:CODE_SEG, DS:DATA_SEG
0018
0019 MAIN PROC FAR
0020
0021
0022     MOV AX,DATA_SEG
0023     MOV DS,AX
0024
0025 START:
0026
0027     MOV CX,20
0028     MOV DI,0
0029
0030 INITIALIZATION:
0031     MOV [STRING+DI],0
0032     INC DI
0033     ;Initialization with
0034     ;0s in case we press ENTER mid-way and to delete everything from last run
0035     LOOP INITIALIZATION
0036
0037     MOV CX,20
0038     MOV DI,0
0039     MOV DX,0
0040
0041 INPUT:
0042     CALL GET_CHAR
0043     CMP AL,0DH
0044     JE AUXILIARY
0045     MOV [STRING+DI],AL
0046     INC DI
0047     LOOP INPUT
0048
0049 AUXILIARY:
0050     MOV CX,DX
0051     MOV DI,0
0052 DISPLAY:
0053     PRINT [STRING+DI]
0054     INC DI
0055     LOOP DISPLAY
0056
0057     PRINTLN NEWLINE
0058
0059     MOV CX,DX
0060     INC CX
0061     MOV DI,0
0062
0063
0064
0065 CHARACTERS:
0066     MOV AL,[STRING+DI]
0067     INC DI
0068     DEC CX
0069     CMP CX,00H
0070     JZ CONTINUE
0071     CMP AL,'a'
0072     JL CHARACTERS
0073     CMP AL,'z'
0074     JG CHARACTERS
0075     SUB AL,20H
0076     PRINT AL

```

```

0777 JMP CHARACTERS
0778
0779
0800 CONTINUE: ;Now for the numbers
0801 PRINT ','
0802 MOV DI,0
0803 MOV CX,DX
0804 INC CX
0805
0806
0807 NUMBERS:
0808 MOV AL,[STRING+DI]
0809 INC DI
0810 DEC CX
0811 CMP CX,00H
0812 JZ CONTINUE2 ;If char is num then print it
0813 CMP AL,30H
0814 JL NUMBERS
0815 CMP AL,39H
0816 JG NUMBERS
0817 PRINT AL
0818 JMP NUMBERS
0819
0820 CONTINUE2:
0821 PRINTLN NEWLINE
0822 JMP START
0823
0824 QUIT:
0825 EXIT
0826 MAIN ENDP
0827
0828
0829 GET_CHAR PROC NEAR
0830 IGNORE:
0831 READ
0832 CMP AL,3DH ;If char is = then quit
0833 JE QUIT
0834 CMP AL,0DH ;Char must be either ENTER or Number or small letter
0835 JE OK
0836 CMP AL,30H
0837 JL IGNORE
0838 CMP AL,39H
0839 JG CHECK_LETTER
0840 JMP OK
0841
0842 CHECK_LETTER:
0843 CMP AL,'a'
0844 JL IGNORE
0845 CMP AL,'z'
0846 JG IGNORE
0847
0848 OK:
0849 INC DX ;Num of chars read
0850 RET
0851 GET_CHAR ENDP
0852
0853 CODE_SEG ENDS
0854 END MAIN

```

## 5η Άσκηση

Για την υλοποίηση προγραμματιστικά των συναρτήσεων χρησιμοποιήθηκε η εντολή DIV που δίνει πηλίκο, άρα τα αποτελέσματα των υλοποιήσεων αυτών είναι τα ακέραια μέρη των ζητούμενων αριθμών και αποθηκεύονται στον AX. Από το υπόλοιπο της διαίρεσης, που αρχικά τοποθετείται στον DX, προκύπτει το μονοψήφιο κλασματικό μέρος των αριθμών. Οι συναρτήσεις των 2 κλάδων και ο τρόπος υπολογισμού των κλασματικών μερών φαίνονται παρακάτω:

Πρώτος κλάδος:  $T = 800V/4095$

Δεύτερο κλάδος  $T = (3200V/4095) - 1200$

Κλασματικό μέρος =  $(10 * \text{υπόλοιπο}) / 4095$

T είναι η ζητούμενη θερμοκρασία και V η τάση εξόδου του ADC. Επισημαίνεται ότι το κλασματικό μέρος είναι ίδιο και για τους 2 κλάδους, αφού έχουν τον ίδιο διαιρέτη στη συνάρτησή τους.

```

500 emulator screen (80x25 chars)
START<Y,N>:Y
7FF      399,9
BFF      1199,8
C00      ERROR
3E8      195,3
BB8      1144,3
B00      1000,5
FFF      ERROR
C01      ERROR

```

ask5.asm / emu8086 Code

```

0001 INCLUDE macros.asm
0002
0003 DATA SEGMENT
0004     STARTPROMPT DB "START<Y,N>:$";Starting message
0005     ERRORMSG DB "ERROR$";Error message
0006
0007 ENDS
0008
0009 CODE SEGMENT
0010     ASSUME CS:CODE, DS:DATA
0011
0012     MAIN PROC FAR
0013         MOV AX,DATA
0014         MOV DS,AX
0015
0016         PRINTSI STARTPROMPT
0017     START:      ;Instertion of starting character
0018         READCH
0019         CMP AL,'N'      ;= N ?
0020         JE FINISH      ;EndΓ
0021         CMP AL,'Y'      ;= Y ?
0022         JE CONT        ;Function
0023         JMP START
0024
0025     CONT:
0026         PRINTCH AL      ;Display of starting character
0027         PRINTLN
0028         PRINTLN
0029     NEWTEMP:
0030         MOV DX,0
0031         MOV CX,3      ;3 Hex digits
0032     READTEMP:   ;Input
0033         CALL HEX_KEYB  ;Insertion of digit
0034         CMP AL,'N'      ;Check for ending
0035         JE FINISH
0036         ;Union of digits in DX register
0037         PUSH CX
0038         DEC CL          ;Shifting
0039         ROL CL,2
0040         MOV AH,0
0041         ROL AX,CL        ;Left shift 8, 4, 0 digits
0042         OR DX,AX         ;Addition of digit in the number
0043         POP CX
0044         LOOP READTEMP
0045
0046     PRINTTAB
0047     MOV AX,DX
0048     CMP AX,2047          ;U<=2 ?
0049     JBE BRANCH1
0050     CMP AX,3071          ;U<=3 ?
0051     JBE BRANCH2
0052     PRINTSI ERRORMSG    ;U>3
0053     PRINTLN
0054     JMP NEWTEMP
0055
0056     BRANCH1:            ;1st branch: U<=2, T=(800*U) div 4095
0057         MOV BX,800
0058         MUL BX
0059         MOV BX,4095
0060         DIV BX
0061         JMP SHOWTEMP
0062
0063     BRANCH2:            ;2nd branch: 2<U<=3, T=((3200*U) div 4095)-1200
0064         MOV BX,3200
0065         MUL BX
0066         MOV BX,4095
0067         DIV BX
0068         SUB AX,1200
0069     SHOWTEMP:
0070         CALL PRINT_DEC16 ;Display of whole part in (AX) register
0071         ;Fraction part = ( remainder *10 ) div 4095
0072         MOV AX,DX
0073         MOV BX,10
0074         MUL BX
0075         MOV BX,4095
0076         DIV BX
0077         PRINTCH ','      ;comma

```

```

076         ADD AL,48             ;Code ASCII
077         PRINTCH AL           ;Display of fraction part
078         PRINTLN
079         JMP NEWTEMP
080
081     FINISH:
082         PRINTCH AL
083         EXIT
084 MAIN ENDP
085
086 HEX_KEYB PROC NEAR           ;Instert hex digit in AL register
087     READ:
088         READCH
089         CMP AL,'N'            ;=N ?
090         JE RETURN
091         CMP AL,48             ;<0 ?
092         JL READ
093         CMP AL,57             ;>9 ?
094         JG LETTER
095         PRINTCH AL
096         SUB AL,48             ;Code ASCII
097         JMP RETURN
098     LETTER:
099         CMP AL,'A'            ;A...F
100         JL READ               ;<A ?
101         CMP AL,'F'            ;>F ?
102         JG READ
103         PRINTCH AL
104         SUB AL,55             ;Code ASCII
105     RETURN:
106
107         RET
108 HEX_KEYB ENDP
109
110 PRINT_DEC16 PROC NEAR       ;Display of 16-bit hex numver from AX register
111     PUSH DX
112
113     MOV BX,10                ;Decimal => divisions with 10
114     MOV CX,0                 ;Digit counter
115     GETDEC:
116         MOV DX,0             ;Digit export
117         DIV BX                ;Number mod 10 (remainder)
118         PUSH DX              ;Division with 10
119         INC CL                ;Temporary storage
120         CMP AX,0              ;Number div 10 = 0 ? < quotient >
121         JNE GETDEC
122     PRINTDEC:
123         POP DX                ;Digit display
124         ADD DL,48              ;Code ASCII
125         PRINTCH DL
126         LOOP PRINTDEC
127
128     POP DX
129     RET
130 PRINT_DEC16 ENDP
131 CODE ENDS
132 END MAIN
133

```