

Συστήματα Μικροϋπολογιστών 2020-2021

Μπούφαλης Οδυσσεύς Δημήτριος el18118

Γεώργιος Στεφανάκης el18436

5η Ομάδα Ασκήσεων

Αρχικά δίνεται το περιεχόμενο των 2 αρχείων μακροεντολών που χρησιμοποιήθηκαν για τη μεταγλώττιση των προγραμμάτων. Για κάθε άσκηση της σειράς δίνεται μια περιγραφή της λειτουργίας του προγράμματος, ένα στιγμιότυπο της εικονικής οθόνης του προσομοιωτή *emu8086* με τις εξόδους του προγράμματος μετά από ικανό αριθμό εκτελέσεων και ο πλήρης κώδικας του προγράμματος.

macros.asm

;print char

PRINTCH MACRO CHAR

PUSH AX

PUSH DX

MOV DL,CHAR

MOV AH,2

INT 21H

POP DX

POP AX

ENDM

;print string

PRINTSTR MACRO STRING

PUSH AX

PUSH DX

MOV DX,OFFSET STRING

MOV AH,9

INT 21H

POP DX

POP AX

ENDM

;print newline

PRINTLN MACRO

PUSH AX

PUSH DX

MOV DL,13

MOV AH,2

INT 21H

MOV DL,10

MOV AH,2

INT 21H

POP DX

POP AX

ENDM

;printtab

PRINTTAB MACRO

PUSH AX

PUSH DX

MOV DL,9

MOV AH,2

INT 21H

POP DX

POP AX

ENDM

```

;read a char and store it in AL
READCH MACRO
    MOV AH,8
    INT 21H
ENDM

;read a char and print it
READNPRINTCH MACRO
    MOV AH,1
    INT 21H
ENDM

;END
EXIT MACRO
    MOV AX,4C00H
    INT 21H
ENDM

```

MACROS1.asm

```

; PRINT CHAR
PRINT MACRO CHAR
    PUSH AX
    PUSH DX

    MOV DL,CHAR
    MOV AH,2
    INT 21H

    POP DX
    POP AX
ENDM PRINT

; EXIT TO DOS
EXIT MACRO
    MOV AX,4C00H
    INT 21H
ENDM EXIT

; PRINT STRING
PRNT_STR MACRO STRING
    MOV DX,OFFSET STRING
    MOV AH,9
    INT 21H
ENDM PRNT_STR

; READ ASCII CODED DIGIT
READ MACRO
    MOV AH,8
    INT 21H
ENDM READ

```

1^η 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 βρίσκει τον μέσο όρο και μετρά και βρίσκει το μέγιστο και το ελάχιστο των αριθμών.

```
PRINT MACRO CHAR
    PUSH AX
    PUSH DX ;store register values that will be changed
    MOV DL,CHAR
    MOV AH,2
    INT 21H
    POP DX
    POP AX
ENDM
```

```
NEWLINE MACRO
    PUSH BX
    MOV BL,13
    PRINT BL
    MOV BL,10 ;ASCII code for new line
    PRINT BL
    POP BX
ENDM
```

```
PRINT_BYTE MACRO BYTE
    PUSH DX
    MOV DL,BYTE
    AND DL,0F0H ;first hex digit
    SHR DL,4
    CALL PRINT_HEX_DIGIT
    MOV DL,BYTE
    AND DL,0FH ;second hex digit
    CALL PRINT_HEX_DIGIT
    POP DX
ENDM
```

```
DATA_SEG SEGMENT
    TABLE DB 128 DUP(?)
DATA_SEG ENDS
```

```
CODE_SEG SEGMENT
    ASSUME CS:CODE_SEG, DS:DATA_SEG
```

```
PRINT_DEC PROC NEAR
    PUSH AX
    PUSH BX
    MOV AH,0
    MOV BL,10 ; divide with 10
    DIV BL ; AL now contains tens and AH contains ones
    ADD AL,30H ; ASCII code of digits 0-9
    PRINT AL ; print tens
    ADD AH,30H ; ASCII code of digits 0-9
    PRINT AH ; print ones
    POP BX
    POP AX
    RET
```

```
PRINT_DEC ENDP
```

```
PRINT_HEX_DIGIT PROC NEAR
```

```
    PUSH DX
    CMP DL,9 ;hex digit in DL, compare with 9 to find whether it is
0-9 or A-F
    JG ADDR1
    SUB DL,07H ;must add 30 if hex digit is 0-9 (ASCII code)
```

```
ADDR1:
```

```
    ADD DL,37H ;must add 37 if hex digit is A-F (ASCII code)
    PRINT DL
    POP DX
    RET
```

```
PRINT_HEX_DIGIT ENDP
```

```
MAIN PROC FAR
```

```
    MOV AX,DATA_SEG
    MOV DS,AX
```

```
    MOV BL,129 ; elements
    MOV DI,0 ; index
```

```
FILL:
```

```
    DEC BL ; first element is 128
    MOV TABLE[DI],BL ; store it in the array
    INC DI ;next array element
    CMP BL,1 ;finish fill if 1
    JNE FILL
    MOV DI,0
    MOV AX,0
    INC DI ; first odd is at index 1 (127)
```

```
ADD_ODD:
```

```
    MOV BL,TABLE[DI] ; get the odd number
    MOV BH,0
    ADD AX,EBX ; add odd number to AX
    INC DI
    INC DI
    CMP DI,129
    JNE ADD_ODD
    MOV BL,64
    DIV BL ; divide with 64 to find mean value, result in AL
    CALL PRINT_DEC
    NEWLINE
```

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

```
COMPARE:
```

```
    INC DI
    CMP BL,TABLE[DI]
    JB NOT_MIN
    MOV BL,TABLE[DI] ;update min
```

```

NOT_MIN:
    CMP BH, TABLE[DI]
    JA NOT_MAX
    MOV BH, TABLE[DI] ;update max

NOT_MAX:
    CMP DI, 127
    JNE COMPARE ;finish if DI reaches 127 (array from 0 to 127)
    PRINT_BYTE BH ;max
    PRINT " "
    PRINT_BYTE BL ;min

END: MOV AX, 4C00H ;end
    INT 21H

MAIN ENDP
CODE_SEG ENDS
END MAIN

```

Στην πρώτη γραμμή βλέπουμε τον μέσο όρο των πρώτων 64 περιττών αριθμών σε δεκαδική μορφή που σωστά είναι 64. Στη δεύτερη γραμμή βλέπουμε τον μέγιστο και τον ελάχιστο αριθμό σε δεκαεξαδική μορφή που ο μέγιστος είναι ο 80H = 128 decimal και 01H = 1 decimal.

2^η 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.

```
INCLUDE macros.asm
```

```
DATA SEGMENT
    MSGZ DB "Z=$"
```

```

MSGW DB "W=$"
MSGSUM DB "Z+W=$"
MSGSUB DB "Z-W=$"
MSGMINUS DB "Z-W=-$"
Z DB 0
W DB 0
TEN DB DUP(10)
DATA ENDS

CODE SEGMENT
    ASSUME CS:CODE, DS:DATA

    MAIN PROC FAR
        MOV AX,DATA
        MOV DS,AX

        START:
            PRINTSTR MSGZ
            CALL READ_DEC_DIGIT ; read the first digit of Z
            MUL TEN ;multiply with 10 the number of the tens

            LEA DI,Z
            MOV [DI],AL ;store the numberoftens*10 of Z

            CALL READ_DEC_DIGIT ; read the second digit
            ADD [DI],AL ; add the second digit to
number_of_tens*10 and we have now the number Z ready and stored

            PRINTCH ' '

            PRINTSTR MSGW
            CALL READ_DEC_DIGIT ; read the first digit of W
            MUL TEN ; multiply with 10 the number
of tens

            LEA DI,W
            MOV [DI],AL ;store the numberoftens*10 of W
            CALL READ_DEC_DIGIT ;read the second digit of W
            ADD [DI],AL ;add the second digit to
number_of_tens*10 and we have now the number W ready and stored

            PRINTLN

            MOV AL,[DI] ; AL stores W
            LEA DI,Z ; DI has the address of Z
            ADD AL,[DI] ; Z+W
            PRINTSTR MSGSUM
            CALL PRINT_NUM8_HEX ; print the sum

            PRINTCH ' '

            MOV AL,[DI] ; AL has Z
            LEA DI,W ; DI has the address of W
            MOV BL,[DI] ; BL has W

            CMP AL,BL ; tsekare an AL<BL
            JB MINUS ;an AL<BL tote to Z-W tha einai
arnitikos

            SUB AL,BL ; an einai thetiki i
diafora apla kane afairesi kai typwse
            PRINTSTR MSGSUB ; xwris - mprosta
            JMP SHOWSUB

```

```


        MINUS:
            SUB    BU,AL                ; an i diafora einai
arnitikos arithmos kantin thetiko
            MOV    AL,BU
            PRINTSTR MSGMINUS        ; kai typwse kai ena - mprosta
        SHOWSUB:
            CALL   PRINT_NUM8_HEX      ; typwse tin diafora
            PRINTLN
            PRINTLN
            JMP    START                ; kai epanelave
MAIN ENDP

READ_DEC_DIGIT PROC NEAR
    READ:
        READCH                ; macro for read
        CMP    AL,48           ; check if it is decimal digit (48-
57)
        JB     READ
        CMP    AL,57
        JA     READ
        PRINTCH AL              ; print the digit
        SUB    AL,48           ; sub 48 in order to keep the number
in binary form
        RET
    READ_DEC_DIGIT ENDP

PRINT_NUM8_HEX PROC NEAR
    MOV    DI,AL
    AND    DI,0F0H            ; first hex digit
    MOV    CX,4
    ROR    DI,CX              ; shift the 4 msb's 4 times right
    CMP    DI,0               ; an to prwto psifio einai 0 (ara
ta 4 msb's) agnoise to
    JE     SKIPZERO
    CALL   PRINT_HEX
    SKIPZERO:
        MOV    DI,AL
        AND    DI,0FH          ; second hex digit
        CALL   PRINT_HEX
        RET
    PRINT_NUM8_HEX ENDP

PRINT_HEX PROC NEAR
    CMP    DI,9               ; an to digit einai megalytero tou 9
prosthese 55 gia na ftasoume sto 65-70 ASCII CODE pou einai to A-F
    JG     LETTER            ; an einai megalytero pigaine sto
letter pou prosthetei 55
    ADD    DI,48              ; an to digit einai <=9 tote
prosthese 48 gia na ftasoume sto 48-57 (0-9)
    JMP    SHOW               ; pigaine sto show pou kanei print
to digit
    LETTER:
        ADD    DI,55
    SHOW:
        PRINTCH DI
        RET
    PRINT_HEX ENDP
CODE ENDS
END MAIN

```

 emulator screen (80x25 chars)

```
Z=28 W=39
Z+W=43 Z-W=-B

Z=00 W=00
Z+W=0 Z-W=0

Z=99 W=99
Z+W=C6 Z-W=0

Z=00 W=99
Z+W=63 Z-W=-63


Z=99 W=00
Z+W=63 Z-W=63

Z=80 W=65
Z+W=91 Z-W=F

Z=65 W=80
Z+W=91 Z-W=-F
```

3^η Άσκηση

Το πρόγραμμα δέχεται έναν 12-bit 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.

 emulator screen (79x25 chars)

```
FFFH =4095D =7777o =111111111111B
000H =0D =0o =0B
00FH =15D =17o =1111B
F00H =3840D =7400o =11100000000B
100H =256D =400o =100000000B
800H =2048D =4000o =10000000000B
C4FH =3151D =6117o =110001001111B
```

```
INCLUDE MACROS1.ASM
```

```
DATA_SEG SEGMENT
    NEW_LINE DB 0AH,0DH,'$'
DATA_SEG ENDS
```

```
CODE_SEG SEGMENT
    ASSUME CS:CODE_SEG, DS:DATA_SEG
```

```
MAIN PROC FAR
```

```
    MOV AX,DATA_SEG
    MOV DS,AX
```



```

CALL HEX_KEYB
CMP AL, 'T'
JE QUIT
MOV BH, AL
CALL HEX_KEYB
CMP AL, 'T'
JE QUIT
MOV BL, AL
AND BL, 0FH
SAL BL, 4
CALL HEX_KEYB
CMP AL, 'T'
JE QUIT
ADD BL, AL

PUSH BX
AND BH, 0FH
CMP BH, 9
JG FIX1
ADD BH, 30H
JMP PRINTF

FIX1:
ADD BH, 37H

PRINTF:
PRINT BH

POP BX

PUSH BX
SAR BL, 4
AND BL, 0FH
CMP BL, 9
JG FIX2
ADD BL, 30H
JMP PRINTF1

FIX2:
ADD BL, 37H

PRINTF1:
PRINT BL

POP BX

PUSH BX
AND BL, 0FH
CMP BL, 9
JG FIX3
ADD BL, 30H
JMP PRINTF2

FIX3:
ADD BL, 37H

PRINTF2:
PRINT BL

PRINT 'H'

```

```

    POP     BX
    PRINT  ' '
    PRINT  '='
    CALL   PRINT_DEC

    PRINT  ' '
    PRINT  '='
    CALL   PRINT_OCT

    PRINT  ' '
    PRINT  '='
    CALL   PRINT_BIN
    PRNT_STR NEW_LINE
    JMP    MAIN

```

```

QUIT:
    EXIT

```

```

MAIN ENDP

```

```

PRINT_DEC PROC NEAR

```

```

    PUSH    AX
    PUSH    CX
    PUSH    BX
    PUSH    DX

    MOV     CX,1

    MOV     AX,BX
    MOV     BX,10

```

```

DIV1:

```

```

    MOV     DX,0
    DIV     BX
    PUSH    DX
    CMP     AX,0
    JE      PRNT_10
    INC     CX
    JMP     DIV1

```

```

PRNT_10:

```

```

    POP     DX
    ADD     DL,30H
    PRINT   DL
    LOOP    PRNT_10

```

```

    PRINT  'D'

```

```

    POP     DX
    POP     BX
    POP     CX
    POP     AX
    RET

```

```

PRINT_DEC ENDP

```

```

PRINT_OCT PROC NEAR

```

```

    PUSH    AX
    PUSH    CX

```

```

    PUSH BX
    PUSH DX

    MOV CX,1

    MOV AX,EX
    MOV EX,8

DIV2:
    MOV DX,0
    DIV BX
    PUSH DX
    CMP AX,0
    JE PRNT_8
    INC CX
    JMP DIV2

PRNT_8:
    POP DX
    ADD DL,30H
    PRINT DL
    LOOP PRNT_8

    PRINT 'o'

    POP DX
    POP BX
    POP CX
    POP AX
    RET
PRINT_OCT    ENDP

```

```

PRINT_BIN    PROC NEAR

```

```

    PUSH AX
    PUSH CX
    PUSH BX
    PUSH DX

    MOV CX,1

    MOV AX,EX
    MOV EX,2

```

```

DIV3:
    MOV DX,0
    DIV EX
    PUSH DX
    CMP AX,0
    JE PRNT_2
    INC CX
    JMP DIV3

```

```

PRNT_2:
    POP DX
    ADD DL,30H
    PRINT DL
    LOOP PRNT_2

    PRINT 'B'

```

```

    POP    DX
    POP    BX
    POP    CX
    POP    AX
    RET

PRINT_BIN    ENDP

HEX_KEYB    PROC    NEAR

IGNORE:
    READ
    CMP    AL,30H        ; if input < 30H ('0') then ignore it
    JL     IGNORE
    CMP    AL,39H        ; if input > 39H ('9') then it may be a hex
letter
    JG     CHECK_LETTER
    SUB    AL,30H        ; otherwise make it a hex number
    JMP    INPUT_OK

CHECK_LETTER:
    CMP    AL,'T'        ; if input = 'T', then return to quit
    JE     INPUT_OK
    CMP    AL,'A'        ; if input < 'A' then ignore it
    JL     IGNORE
    CMP    AL,'F'        ; if input > 'F' then ignore it
    JG     IGNORE
    SUB    AL,37H        ; otherwise make it a hex number

INPUT_OK:
    RET
HEX_KEYB    ENDP

CODE_SEG    ENDS
    END     MAIN

```

4^η Άσκηση

Σε αυτή την άσκηση πληκτρολογούμε 20 νούμερα και πεζούς χαρακτήρες (εκτός και αν δοθεί enter) και στη συνέχεια τυπώνει τους χαρακτήρες με την σειρά που δόθηκαν κάνοντας τους πεζούς χαρακτήρες κεφαλαίους και στη συνέχεια τυπώνει μια – και μετά τα νούμερα με τη σειρά που δόθηκαν.


 emulator screen (80x25 chars)

```
DATA SEGMENT
S DB 20 DUP(?)
DATA ENDS
```

```
CODE SEGMENT
    ASSUME CS:CODE, DS:DATA
```

```
PRINT MACRO CHAR
    PUSH AX
    PUSH DX ;store register values that will be changed
    MOV DI,CHAR
    MOV AH,2
    INT 21H
    POP DX
    POP AX
ENDM
```

```
NEWLINE MACRO
    PUSH BX
    MOV BL,13
    PRINT BL
    MOV BL,10 ;ASCII code for new line
    PRINT BL
    POP BX
ENDM
```

```
MAIN PROC FAR
```

```
    REPEAT:
        MOV CX,0 ;read at most 20 times
        MOV DI,0
```

```
    READ:
        CALL INPUT
        CMP AL,13
        JE BEGIN
        MOV S[DI],AL ;save input (AL)
        INC DI ; increase DI to read the next number and store it
        INC CX ; increase counter of data
        CMP CX,20 ; if we have read 20 data we stop reading
        JB READ ; loop until finish reading
```

```
    BEGIN:
        MOV DI,0
        MOV DX,CX ; save counter of data
        NEWLINE
```

```
    LETTERS:
        MOV AL,S[DI]
        CMP AL,97
        JB SKIP2
        CMP AL,122
        JA SKIP2
        SUB AL,32 ;convert to UPPERCASE
        PRINT AL ;print if it's a letter
    SKIP2:
```

```

    INC DI
    LOOP LETTERS
    PRINT "-"
    MOV CX, DX
    MOV DI, 0

NUMBERS:
    MOV AL, S[DI]
    CMP AL, 48
    JB SKIP
    CMP AL, 57
    JA SKIP
    PRINT AL ;print if it's a number
SKIP:
    INC DI
    LOOP NUMBERS
    NEWLINE
    JMP REPEAT
MAIN ENDP

INPUT PROC NEAR
IGNORE:
    MOV AH, 8
    INT 21H ;read from keyboard, result in AL
    CMP AL, 61 ;check if it is =
    JE QUIT ; if it is equal exit
    CMP AL, 13 ; if we input enter then return
    JE ENTER
    CMP AL, 48 ;ignore if not a-z(97-122) or 0-9(48-57)
    JB IGNORE
    CMP AL, 122
    JA IGNORE
    CMP AL, 57
    JBE FINISH
    CMP AL, 97
    JB IGNORE
FINISH:
    PRINT AL
    RET
ENTER:
    RET
QUIT:
    MOV AX, 4C00H ;end
    INT 21H
INPUT ENDP

CODE ENDS
END MAIN

```

5η Άσκηση

Το πρόγραμμα προσομοιώνει ένα σύστημα λήψης θερμοκρασίας που περιλαμβάνει έναν αισθητήρα θερμοκρασίας, έναν μετατροπέα από αναλογική τιμή σε ψηφιακή (ADC) και έναν υπολογιστή με τον μΕ 80x86. Υποτίθεται ότι ο αισθητήρας μετρά τη θερμοκρασία και παρέχει μία τάση στο διάστημα [0,3] Volts στον ADC. Ο ADC ψηφιοποιεί την τάση του αισθητήρα στο διάστημα [0,4095] Volts. Η ψηφιοποιημένη τάση παρέχεται ως είσοδος στον υπολογιστή, ο οποίος λαμβάνει τη θερμοκρασία μέσω μιας 16-bit θύρας εισόδου σε δυαδική μορφή των 12 bits και την απεικονίζει ως έξοδο στην οθόνη με έναν 4ψήφιο δεκαδικό αριθμό με ένα

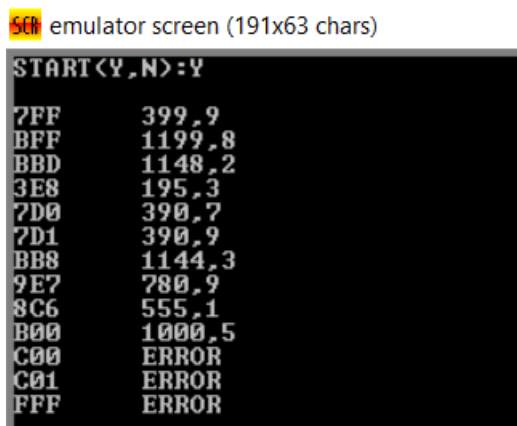
κλασματικό ψηφίο (XXXX,X) από 0,0 έως 1200,0 °C. Το σύστημα περιγράφεται από το παρακάτω σχήμα. Η θύρα εισόδου προσομοιώνεται από το πληκτρολόγιο, μέσω του οποίου εισάγονται τα δεδομένα (η τάση του ADC) ως 3 δεκαεξαδικά ψηφία. Με την εκκίνηση της εκτέλεσης του προγράμματος εμφανίζεται το μήνυμα *START(Y,N)*: και ο χρήστης επιλέγει αν αυτό θα λειτουργήσει (Y) ή θα τερματιστεί (N). Σε περίπτωση λειτουργίας, το πρόγραμμα δέχεται τα 3 ψηφία της εισόδου (μόνο έγκυρα) και εμφανίζει τη θερμοκρασία. Το πρόγραμμα είναι συνεχούς λειτουργίας, τερματίζεται οποιαδήποτε στιγμή αν δοθεί ο χαρακτήρας N και σε περίπτωση θερμοκρασίας μεγαλύτερης από 1200,0 °C εμφανίζει το μήνυμα σφάλματος *ERROR*. Το πρόγραμμα αρχικά εμφανίζει το μήνυμα εκκίνησης (*STARTPROMPT*) και τον χαρακτήρα που δίνει ο χρήστης. Κατά τη λειτουργία του, δέχεται τα 3 ψηφία της εισόδου στον AL με κλήση της ρουτίνας *HEX_KEYB* και τα ενώνει στον DX ολισθαίνοντάς τα κατάλληλα. Στη συνέχεια συγκρίνει την είσοδο με τα ψηφιοποιημένα άνω όρια των κλάδων της χαρακτηριστικής καμπύλης του αισθητήρα για να αποφασίσει σε ποιον κλάδο ανήκει και υπολογίζει τη θερμοκρασία υλοποιώντας την αντίστοιχη συνάρτηση. 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.

$$1\text{ος κλάδος: } T = \frac{800V}{4095}$$

$$2\text{ος κλάδος: } T = \frac{3200V}{4095} - 1200$$

$$\text{κλασματικός μέρος} = \frac{10 \cdot \text{υπόλοιπο}}{4095}$$

Όπου *T* 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. Επισημαίνεται ακόμη ότι οι παραπάνω διαιρέσεις αναφέρονται σε ακέραια διαίρεση και ότι τα ψηφιοποιημένα άνω όρια των κλάδων της χαρακτηριστικής καμπύλης του αισθητήρα είναι οι τιμές της τάσης εξόδου του ADC για τις οποίες παίρνουμε θερμοκρασία όχι μεγαλύτερη από τις αντίστοιχες τιμές του οριζόντιου άξονα που φαίνονται στο σχήμα (άρα για τον 1ο κλάδο το 2 μετατρέπεται σε 2047 και για τον 2ο το 3 σε 3071). Η εμφάνιση του ακέραιου μέρους γίνεται μέσω του AX με κλήση της ρουτίνας *PRINT_DEC16* που τυπώνει έναν 16-bit δεκαδικό αριθμό και ακολουθείται από την εμφάνιση του κλασματικού μέρους.

 emulator screen (191x63 chars)

START(Y,N):Y	
7FF	399,9
BFF	1199,8
BBD	1148,2
3E8	195,3
7D0	390,7
7D1	390,9
BB8	1144,3
9E7	780,9
8C6	555,1
B00	1000,5
C00	ERROR
C01	ERROR
FFF	ERROR

```
INCLUDE macros.asm
```

```
DATA SEGMENT
```

```
STARTPROMPT DB "START(Y,N):$" ;begin message
ERRORMSG DB "ERROR$" ;error message
```

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE, DS:DATA

MAIN PROC FAR

MOV AX, DATA
MOV DS, AX

PRINTSTR STARTPROMPT

START: ; read first char

READCH

CMP AL, 'N' ; if it is = N

JE FINISH ; finish

CMP AL, 'Y' ; if it is = Y

JE CONT ; continue

JMP START

CONT:

PRINTCH AL ; print first char

PRINTLN

PRINTLN

NEWTEMP:

MOV DX, 0

MOV CX, 3 ; read 3 digits

READTEMP: ; input

CALL HEX_KEYB

CMP AL, 'N' ; check if we need to terminate

JE FINISH

; put all the digits at DX

PUSH CX

DEC CL

ROL CL, 2

MOV AH, 0

ROL AX, CL

OR DX, AX

POP CX

LOOP READTEMP

PRINTTAB

MOV AX, DX

CMP AX, 2047 ; V<=2 ?

JBE BRANCH1

CMP AX, 3071 ; V<=3?

JBE BRANCH2

PRINTSTR ERRORMSG ; V>3?

PRINTLN

JMP NEWTEMP

BRANCH1: ; first branch V<=2, T=800*V div 4095

MOV BX, 800

MUL BX

MOV BX, 4095

DIV BX

JMP SHOWTEMP

BRANCH2:

; second branch 2<V<=3, T=((3200*V)div4095)-1200

MOV BX, 3200

MUL BX

MOV BX, 4095

DIV BX

SUB AX, 1200


```

SHOWTEMP:
    CALL PRINT_DEC16    ;akeraio meros (AX)
                        ;klasmatiko meros = (ypoloipo*10)
div 4095

    MOV AX,DX
    MOV BX,10
    MUL BX
    MOV BX,4095
    DIV BX

    PRINTCH ',', '
    ADD AL,48           ;ascii code
    PRINTCH AL          ;print klasmatiko meros
    PRINTLN
    JMP NEWTEMP

FINISH:
    PRINTCH AL
    EXIT

MAIN ENDP

                        ;input hex digit at AL
HEX_KEYB PROC NEAR
    READ:
        READCH
        CMP AL,'N'      ;=N ?
        JE RETURN
        CMP AL,48       ;<0 ?
        JL READ
        CMP AL,57       ;>9 ?
        JG LETTER
        PRINTCH AL
        SUB AL,48        ;ASCII code
        JMP RETURN
    LETTER:
                        ;A...F
        CMP AL,'A'      ;<A ?
        JL READ
        CMP AL,'F'      ;>F ?
        JG READ
        PRINTCH AL
        SUB AL,55        ;ASCII code
    RETURN:

    RET
HEX_KEYB ENDP

                        ;print 16 bit decimal number from AX
PRINT_DEC16 PROC NEAR
    PUSH DX

    MOV BX,10
    MOV CX,0            ;counter of digits
    GETDEC:
        MOV DX,0        ;get the digits
        DIV BX          ;number mod 10 (ypoloipo)
                        ;div with 10
        PUSH DX
        INC CL
        CMP AX,0        ;number div 10 = 0 (piliko)
        JNE GETDEC
    PRINTDEC:
                        ;print digits
    POP DX

```

```
        ADD     DI,48                ;ASCII code
        PRINTCH DI
        LOOP    PRINTDEC

        POP     DX
        RET

PRINT_DEC16 ENDP
CODE ENDS
END MAIN
```