Σχολή Ηλεκτρολόγων Μηχανικών
και
Μηχανικών Υπολογιστών

Συστήματα Μικροϋπολογιστών [Ροή Υ]

5η Ομάδα Ασκήσεων

Ναυσικά Αμπατζή <031 17 198>

Δημήτρης Δήμος <031 17 165>

6ο Εξάμηνο

Ιούνιος 2020

Στην παρούσα (5η) ομάδα ασκήσεων, όλες οι ασκήσεις αφορούν στη σχεδίαση κώδικα σε Assembly 8086. Στο επισυναπτόμενο αρχείο zip έχουν συμπεριληφθεί τα πέντε αρχεία κώδικα για κάθε άσκηση, συν ένα αρχείο ονόματι MACROS.asm, το οποίο χρησιμοποιείται από τους κώδικες των ασκήσεων (πλην της πρώτης).

Τα αρχεία πέραν της αναφοράς είναι τα:

- group5.ex1.asm
- group5.ex2.asm
- group5.ex3.asm
- group5.ex4.asm
- group5.ex5.asm
- MACROS.asm

Επειδή, λοιπόν, δεν υπάρχουν πολλά πράγματα να εξηγηθούν για τους κώδικες, πέραν των διευκρινιστικών σχολίων, που ήδη υπάρχουν στις αντίστοιχες σειρές, στην αναφορά παρατίθενται οι κώδικες με τα σχόλιά τους όπως και στα επισυναπτόμενα αρχεία.

Επιπλέον, στην αναφορά θα δοθεί για την άσκηση 5 μια εξήγηση της λογικής βάσει της οποίας έγιναν οι υπολογισμοί και η σύνθεση του κώδικα.

1η Άσκηση

```
; ---------------------- MACROS --------------------------
; 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
; ------------------------------------------------------------



DATA_SEG    SEGMENT
    NEW_LINE  DB 0AH,0DH,'$'
    QUIT_LINE DB "Time to quit...",0AH,0DH,'$'
DATA_SEG    ENDS



CODE_SEG    SEGMENT
    ASSUME CS:CODE_SEG, DS:DATA_SEG

MAIN PROC FAR          ; we consider the first digit to be the MS 4-bit

   MOV AX,DATA_SEG     ; important segment arrangements
   MOV DS,AX


   CALL HEX_KEYB       ; get hex digit
   CMP AL,'Q'          ; if it's 'Q' end it
   JE QUIT
   MOV DH,AL           ; save it in DH
   CALL HEX_KEYB       ; get then next hex digit
   CMP AL,'Q'          ; check for 'Q'
   JE QUIT
   MOV DL,AL           ; save it in DL

   CLC                 ; clear carry, ready for left-slide
   SHL DH,4            ; slide DH by 4 and the OR it with DL
   OR DL,DH


   ; from now on we print the different forms of DL
   PUSH DX
```

```
   ; print the hex form of input number
   SAR DH,4          ; manipulate the MS bits
   AND DH,0FH
   ADD DH,30H        ; ASCII code them
   CMP DH,39H
   JLE PRINT_H
   ADD DH,07H        ; if it's a letter, then correct the ASCII by adding
07H
PRINT_H:
   PRINT DH          ; and print their hex form


   AND DL,0FH        ; isolate 4 LS bits
   ADD DL,30H        ; ASCII code them
   CMP DL,39H
   JLE PRINT_L
   ADD DL,07H        ; if it's a letter, then correct the ASCII by adding
07H
PRINT_L:
   PRINT DL          ; print their hex form

   PRINT 'h'


   POP DX            ; restore DX
   PRINT '='
   CALL PRINT_DEC
   PRINT '='
   CALL PRINT_OCT
   PRINT '='
   CALL PRINT_BIN
   PRNT_STR NEW_LINE
   JMP MAIN


QUIT:
    PRNT_STR QUIT_LINE
    EXIT
MAIN ENDP



; ----------------------- ROUTINES ---------------------------

; PRINT DECIMAL FORM OF DL
PRINT_DEC PROC NEAR     ; input: DL
                        ; prints its decimal form

    PUSH AX     ; save registers
    PUSH CX
    PUSH DX

    MOV CX,1    ; initialize digit counter

    MOV AL,DL
    MOV DL,10

LD:
    MOV AH,0    ; divide number by 10
    DIV DL
    PUSH AX     ; save quotient
    CMP AL,0    ; if quot = 0, start printing
    JE PRNT_10
    INC CX      ; increase counter (aka digits number)
    JMP LD      ; repeat dividing quotients by 10

PRNT_10:
    POP AX      ; get digit
    MOV AL,AH
    MOV AH,0
    ADD AX,'0'  ; ASCII coded
    PRINT AL    ; print
    LOOP PRNT_10   ; repeat till no more digits

    PRINT 'd'

    POP DX
    POP CX      ; restore registers
```

```
    POP AX
    RET
PRINT_DEC ENDP



; PRINT OCTAL FORM OF DL
PRINT_OCT PROC NEAR     ; input: DL
                        ; prints its octal form

    PUSH AX     ; save registers
    PUSH CX
    PUSH DX


    MOV CX,1    ; initialize digit counter

    MOV AL,DL
    MOV DL,8

LO:
    MOV AH,0    ; divide number by 8
    DIV DL
    PUSH AX     ; save quotient
    CMP AL,0    ; if quot = 0, start printing
    JE PRNT_8
    INC CX      ; increase counter (aka digits number)
    JMP LO      ; repeat dividing quotients by 8

PRNT_8:
    POP AX      ; get digit
    MOV AL,AH
    MOV AH,0
    ADD AX,'0'  ; ASCII coded
    PRINT AL    ; print
    LOOP PRNT_8   ; repeat till no more digits

    PRINT 'o'

    POP DX      ; restore registers
    POP CX
    POP AX
    RET
PRINT_OCT ENDP



; PRINT BINARY FORM OF DL
PRINT_BIN PROC NEAR     ; input: DL
                        ; prints its binary form

    PUSH AX     ; save registers
    PUSH CX
    PUSH DX


    MOV CX,1    ; initialize digit counter

    MOV AL,DL
    MOV DL,2

LB:
    MOV AH,0    ; divide number by 2
    DIV DL
    PUSH AX     ; save quotient
    CMP AL,0    ; if quot = 0, start printing
    JE PRNT_2
    INC CX      ; increase counter (aka digits number)
    JMP LB      ; repeat dividing quotients by 2

PRNT_2:
    POP AX      ; get digit
    MOV AL,AH
    MOV AH,0
    ADD AX,'0'  ; ASCII coded
    PRINT AL    ; print
    LOOP PRNT_2   ; repeat till no more digits
```

```
    PRINT 'b'

    POP DX      ; restore registers
    POP CX
    POP AX
    RET
PRINT_BIN ENDP



; READ A HEX DIGIT (ONLY HEX DIGITS ARE ACCEPTED) --> AL
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,'Q'        ; if input = 'Q', 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
```

2η Άσκηση

```
INCLUDE MACROS.ASM


DATA_SEG    SEGMENT
    TABLE   DB 256 DUP(?)
    AVERAGE DB ?
    MIN     DB ?
    MAX     DB ?
    NEWLINE DB 0AH,0DH,'$'
DATA_SEG    ENDS


CODE_SEG    SEGMENT
    ASSUME CS:CODE_SEG, DS:DATA_SEG


MAIN PROC FAR

    MOV AX,DATA_SEG  ; important initialization
    MOV DS,AX


    ; in this part data is stored
    MOV AL,254   ; initialize counter
    MOV DI,0     ; initialize index
STORE:
    MOV [TABLE+DI],AL
    DEC AL
    INC DI
    CMP DI,256
    JNE STORE


    ; in this part average is calculated
    MOV DX,0    ; initialize counter to store the sum
    MOV DI,0    ; initialize index to 0
    MOV AX,0    ; temporary register
SUM:
    MOV AL,[TABLE+DI]
    ADD DX,AX
CONT:
    ADD DI,2
    CMP DI,256
    JNE SUM
END_COUNT:      ; sum of all evens in DX
    MOV AX,DX
    MOV DX,0
    MOV CX,128
    DIV CX      ; divide the sum by 128
    MOV DX,AX
    MOV AVERAGE,DL  ; average is now calculated and stored


    ; in this part min and max are calculated
    MOV MAX,0   ; initializations
    MOV MIN,255
    MOV DI,0

MIN_MAX:
    MOV AL,[TABLE+DI]
    CMP MIN,AL          ; MIN =< AL ?
    JNA GO_MAX          ; if yes, then go see for max
    MOV MIN,AL          ; else update minimum data
GO_MAX:
    CMP MAX,AL          ; MAX >= AL ?
    JAE ITS_MAX         ; if yes, then continue
    MOV MAX,AL          ; else update maximum data
ITS_MAX:
    INC DI
    CMP DI,256
```

```
    JNE MIN_MAX


    ; in this part, demanded data is printed
    MOV AL,AVERAGE     ; PRINT AVERAGE
    SAR AL,4
    AND AL,0FH   ; isolate 4 MSB
    ADD AL,30H   ; ASCII code it
    PRINT AL     ; print the first hex digit

    MOV AL,AVERAGE
    AND AL,0FH   ; isolate 4 LSB
    ADD AL,30H   ; ASCII code it
    CMP AL,39H
    JLE OK1
    ADD AL,07H   ; if it's a letter, fix ASCII
OK1:
    PRINT AL     ; print the second hex digit
    PRINT 'h'

    PRNT_STR NEWLINE



    MOV AL,MIN          ; PRINT MINIMUM DATA
    SAR AL,4
    AND AL,0FH   ; isolate 4 MSB
    CMP AL,0
    JE NEXT_DIGIT
    ADD AL,30H   ; ASCII code it
    CMP AL,39H
    JLE OK2
    ADD AL,07H   ; if it's a letter, fix ASCII
OK2:
    PRINT AL     ; print the first hex digit
NEXT_DIGIT:
    MOV AL,MIN
    AND AL,0FH   ; isolate 4 LSB
    ADD AL,30H   ; ASCII code it
    CMP AL,39H
    JLE OK3
    ADD AL,07H   ; if it's a letter, fix ASCII
OK3:
    PRINT AL     ; print the second hex digit
    PRINT 'h'

    PRINT ' '

    MOV AL,MAX           ; PRINT MAXIMUM DATA
    SAR AL,4
    AND AL,0FH   ; isolate 4 MSB
    ADD AL,30H   ; ASCII code it
    CMP AL,39H
    JLE OK4
    ADD AL,07H   ; if it's a letter, fix ASCII
OK4:
    PRINT AL     ; print the first hex digit

    MOV AL,MAX
    AND AL,0FH   ; isolate 4 LSB
    ADD AL,30H   ; ASCII code it
    CMP AL,39H
    JLE OK5
    ADD AL,07H   ; if it's a letter, fix ASCII
OK5:
    PRINT AL     ; print the second hex digit
    PRINT 'h'



    EXIT

MAIN    ENDP

CODE_SEG    ENDS
    END MAIN
```

3η Άσκηση

```
INCLUDE MACROS.ASM


DATA_SEG    SEGMENT
    FIRST   DB ?
    SECOND  DB ?
    NEWLINE 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

    MOV BX,0          ; initialization
    CALL HEX_KEYB
    MOV BH,AL
    SAL BH,4
    CALL HEX_KEYB
    MOV BL,AL
    OR BL,BH
    MOV FIRST,BL      ; first number in FIRST


    CALL HEX_KEYB
    MOV BH,AL
    SAL BH,4
    CALL HEX_KEYB
    MOV BL,AL
    OR BL,BH
    MOV SECOND,BL     ; second number in SECOND

    ; print in the form: x=... y=...
    PRINT 'x'
    PRINT '='

    MOV DL,FIRST
    CALL PRINT_HEX        ; print its hex form

    PRINT ' '
    PRINT 'y'
    PRINT '='

    MOV DL,SECOND
    CALL PRINT_HEX        ; print its hex form

    PRNT_STR NEWLINE



    ; calculate sum and difference
    MOV AH,0
    MOV BH,0
    MOV AL,FIRST
    MOV BL,SECOND
    ADD AX,BX          ; add them


    PRINT 'x'
    PRINT '+'
    PRINT 'y'
    PRINT '='

    MOV DX,AX
    CALL PRINT_DEC_2      ; print the sum from DX
```

```
    PRINT ' '
    PRINT 'x'
    PRINT '-'
    PRINT 'y'
    PRINT '='

    MOV AH,0
    MOV BH,0
    MOV AL,FIRST
    MOV BL,SECOND

    CMP AL,BL
    JAE NO_MINUS      ; check for the sign. If FIRST < SECOND then
    PRINT '-'         ; we do SECOND-FIRST
    SUB BX,AX         ; and put a '-'
    MOV DL,BL
    JMP OK_DIF
NO_MINUS:
    SUB AX,BX         ; sub them
    MOV DL,AL
OK_DIF:

    CALL PRINT_DEC    ; print the difference from DL


    EXIT
MAIN ENDP



; --------- AUXILIARY ROUTINES -----------
; READ A HEX DIGIT (ONLY HEX DIGITS ARE ACCEPTED) --> AL
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,'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



; PRINT THE NUMBER IN DL
PRINT_HEX PROC NEAR

    PUSH AX

    MOV AL,DL
    SAR AL,4
    AND AL,0FH   ; isolate 4 MSB
    ADD AL,30H   ; ASCII code it
    CMP AL,39H
    JLE NEX
    ADD AL,07H   ; if it's a letter, fix ASCII
NEX:
    CMP AL,'0'
    JE DONT_PRINT_IT
    PRINT AL     ; print the first hex digit

DONT_PRINT_IT:
    MOV AL,DL
    AND AL,0FH   ; isolate 4 LSB
    ADD AL,30H   ; ASCII code it
    CMP AL,39H
```

```
    JLE OK
    ADD AL,07H   ; if it's a letter, fix ASCII
OK:
    PRINT AL     ; print the second hex digit

    POP AX
    RET
PRINT_HEX ENDP



; PRINT DECIMAL FORM OF DL
PRINT_DEC PROC NEAR     ; input: DL
                        ; prints its decimal form

    PUSH AX     ; save registers
    PUSH CX
    PUSH DX

    MOV CX,1    ; initialize digit counter

    MOV AL,DL
    MOV DL,10

LD:
    MOV AH,0    ; divide number by 10
    DIV DL
    PUSH AX     ; save
    CMP AL,0    ; if quot = 0, start printing
    JE PRNT_10
    INC CX      ; increase counter (aka digits number)
    JMP LD      ; repeat dividing quotients by 10

PRNT_10:
    POP AX      ; get digit
    MOV AL,AH
    MOV AH,0
    ADD AX,'0'     ; ASCII coded
    PRINT AL       ; print
    LOOP PRNT_10   ; repeat till no more digits


    POP DX
    POP CX      ; restore registers
    POP AX
    RET
PRINT_DEC ENDP



; PRINT DECIMAL FORM OF DX
PRINT_DEC_2 PROC NEAR   ; input: DX
                        ; prints its decimal form

    PUSH AX     ; save registers
    PUSH BX
    PUSH CX
    PUSH DX

    MOV CX,1    ; initialize digit counter

    MOV AX,DX
    MOV BX,10

LD_2:
    MOV DX,0
    DIV BX
    PUSH DX        ; save
    CMP AX,0       ; if quot = 0, start printing
    JE PRNT_10_2
    INC CX         ; increase counter (aka digits number)
    JMP LD_2       ; repeat dividing quotients by 10

PRNT_10_2:
    POP DX         ; get digit
    MOV AL,DL
    MOV AH,0
    ADD AX,'0'        ; ASCII coded
    PRINT AL          ; print
```

```
    LOOP PRNT_10_2   ; repeat till no more digits


    POP DX
    POP CX      ; restore registers
    POP BX
    POP AX
    RET
PRINT_DEC_2 ENDP


CODE_SEG    ENDS
    END MAIN
```

4η Άσκηση

```
INCLUDE MACROS.ASM


DATA_SEG    SEGMENT
    SYMBOLS DB 16 DUP(?)
    NEWLINE DB 0AH,0DH,'$'
    ENTER   DB "Enter was pressed. Program now exits...",0AH,0DH,'$'
DATA_SEG    ENDS


CODE_SEG    SEGMENT
    ASSUME CS:CODE_SEG, DS:DATA_SEG


MAIN PROC FAR

    MOV AX,DATA_SEG
    MOV DS,AX


AGAIN:
    MOV CX,16             ; initialize counter
    MOV DI,0              ; initialize array index
INPUT:
    CALL GET_CHAR         ; get acceptable character
    CMP AL,0DH            ; if it's enter then exit
    JE END_OF_PROG
    MOV [SYMBOLS+DI],AL
    INC DI
    LOOP INPUT



    MOV CX,16
    MOV DI,0
OUTPUT:
    PRINT [SYMBOLS+DI]    ; print all characters
    INC DI
    LOOP OUTPUT


    PRNT_STR NEWLINE      ; new line


    MOV CX,16
    MOV DI,0
PRINT_NUMS:
    MOV AL,[SYMBOLS+DI]
    CMP AL,30H             ; we check if it is an ASCII coded digit
    JL  NOT_A_NUMBER
    CMP AL,39H
    JG NOT_A_NUMBER
    PRINT AL               ; print only the digits

NOT_A_NUMBER:
    INC DI
    LOOP PRINT_NUMS

    PRINT '-'

    MOV CX,16
    MOV DI,0
PRINT_LETTERS:
    MOV AL,[SYMBOLS+DI]
    CMP AL,'A'             ; we check if it is an ASCII coded capital letter
    JL  NOT_A_LETTER
    CMP AL,'Z'
    JG NOT_A_LETTER
    ADD AL,20H             ; make it a non-capital letter
    PRINT AL               ; print only the letters as non capitals
```

```
NOT_A_LETTER:
    INC DI
    LOOP PRINT_LETTERS


    PRNT_STR NEWLINE     ; new line
    JMP AGAIN

END_OF_PROG:
    PRNT_STR ENTER
    EXIT
MAIN ENDP



; --------- AUXILIARY ROUTINES -----------

; READ A DIGIT OR CHARACTER BETWEEN 'A' AND 'Z' --> AL
GET_CHAR PROC NEAR
IGNORE:
    READ
    CMP AL,0DH
    JE INPUT_OK
    CMP AL,30H        ; if input < 30H ('0') then ignore it
    JL IGNORE
    CMP AL,39H        ; if input > 39H ('9') then it may be a capital letter
    JG MAYBE_LETTER
    JMP INPUT_OK

MAYBE_LETTER:
    CMP AL,'A'        ; if input < 'A' then ignore it
    JL IGNORE
    CMP AL,'Z'        ; if input > 'Z' then ignore it
    JG IGNORE

INPUT_OK:
    RET
GET_CHAR ENDP



; PRINT THE NUMBER IN DL
PRINT_HEX PROC NEAR

    PUSH AX

    MOV AL,DL
    SAR AL,4
    AND AL,0FH   ; isolate 4 MSB
    ADD AL,30H   ; ASCII code it
    CMP AL,39H
    JLE NEX
    ADD AL,07H   ; if it's a letter, fix ASCII
NEX:
    CMP AL,'0'
    JE DONT_PRINT_IT
    PRINT AL     ; print the first hex digit

DONT_PRINT_IT:
    MOV AL,DL
    AND AL,0FH   ; isolate 4 LSB
    ADD AL,30H   ; ASCII code it
    CMP AL,39H
    JLE OK
    ADD AL,07H   ; if it's a letter, fix ASCII
OK:
    PRINT AL     ; print the second hex digit

    POP AX
    RET
PRINT_HEX ENDP



CODE_SEG    ENDS
    END MAIN
```

5η Άσκηση

Για την υλοποίηση του κώδικα ακολουθήσαμε την εξής λογική:

Από το διάγραμμα (A/D bits – Voltage) συμπεραίνουμε ότι η σχέση μεταξύ εξόδου του μετατροπέα και της τάσης εισόδου του είναι γραμμική. Θα βόλευε πολύ στους υπολογισμούς αν δουλεύαμε με την τάση σε κλίμακα mV ώστε να έχουμε περισσότερα ψηφία για επεξεργασία στους καταχωρητές μας, χωρίς δεκαδικά.

Από το εν λόγω διάγραμμα, λοιπόν, διαπιστώνουμε πως:

$$(input\ bits) = \frac{4095}{2000} mVolt \Rightarrow mVolt = \frac{400}{819}\ (input\ bits)$$

Ακόμη, από το διάγραμμα Volt – θερμοκρασία παρατηρούμε τρεις γραμμικές περιοχές αντιστοίχισης τάσης με θερμοκρασία. Με εφαρμογή απλών μαθηματικών καταλήγουμε στα τρία ευθύγραμμα τμήματα που ανήκουν στις ευθείες με εξισώσεις:

- mVolt = 1000/500 T => Temperature = ½ mVolt
- mVolt = 800/200 T – 1000 => Temperature = 250+mVolt/4
- mVolt = 2/3 T + 1333 => Temperature = 3/2 V – 2000

Για να καταλάβουμε σε ποια περιοχή του γραφήματος βρισκόμαστε, κοιτάμε:

- 0 < mV < 1000 => 0< input < 2047.5
- 1000 < mV < 1800 => 2047.4 < input < 3685.5
- 1800 < mV < 2000 => 3685.5 < input < 4095

Στον κώδικα, λοιπόν, αφού γίνει η επιλογή περιοχής σωστά, σύμφωνα με τα παραπάνω, η θερμοκρασία υπολογίζεται για κάθε περιοχή με τον αντίστοιχο τύπο:

- Temperature = 200/819 * input, (περιοχή 1)
- Temperature = 250+100/819 * input, (περιοχή 2)
- Temperature = 200/273 * input - 2000, (περιοχή 3)

Η υλοποίηση του κώδικα βασίστηκε στα παραπάνω. Παρατίθεται παρακάτω:

```
INCLUDE MACROS.ASM



PRINT_DEC MACRO
    PUSH AX

    ADD DL, 30H
    MOV AH,2
    INT 21H

    POP AX
ENDM

DATA_SEG    SEGMENT
    START_MSG DB "START (Y, N):",0AH,0DH,'$'
    END_MSG   DB "Program will now end...",0AH,0DH,'$'
    ERROR_MSG DB "ERROR",0AH,0DH,'$'
    FIRST     DB ?
    SECOND    DB ?
    THIRD     DB ?
    TEMP      DB "The temperature is: ",'$'
    NEWLINE   DB 0AH,0DH,'$'
DATA_SEG    ENDS

CODE_SEG    SEGMENT
    ASSUME CS:CODE_SEG, DS:DATA_SEG

MAIN PROC FAR

    ; define DATA SEGMENT
    MOV AX,DATA_SEG
    MOV DS,AX

    PRNT_STR START_MSG   ; ask user what he wants
    CALL YES_OR_NO       ; put answer in AL

    ; act accordingly (N===>end program)
    CMP AL,'N'
    JE END_ALL

REPEAT:
    ; we get the 3-HEX-digit input
    CALL GET_INPUT
    CMP AL,'N'
    JE END_ALL
    MOV FIRST,AL

    CALL GET_INPUT
    CMP AL,'N'
    JE END_ALL
    MOV SECOND,AL

    CALL GET_INPUT
    CMP AL,'N'
    JE END_ALL
    MOV THIRD,AL

    ; put input in AX
    CALL TIDY_INPUT

    ; calculate A/D Converter output Volts (AX)
    MOV BX,10
    MUL BX
    MOV BX,20475
    DIV BX
    MOV BX,AX           ; Volts = (INPUT * 10 / 20475)
                        ; BX = quotient
                        ; DX = remainder

    ; check for error
    CMP BX,2
    JGE ERROR_PART      ; if quotient >= 2, then we have
                        ; a temperature over 999.9 degrees ===> ERROR



    ; which region are we in?
    CALL TIDY_INPUT     ; AX <=== input
```

```
    CMP AX,2047          ;       0 < (AID) < 2047.5  ===> 0<temp<500 deg
    JLE FIRST_REGION
    CMP AX,3685          ; 2047.5 < (AID) < 3685.5  ===> 500<temp<700 deg
    JLE SECOND_REGION
                         ; otherwise third class    ===> 700<temp<1000 deg

THIRD_REGION:      ; T = (200*input)/273 - 2000
    MOV CX,2000
    MUL CX         ; DX:AX = 2000*input
    MOV CX,273
    DIV CX         ; AX = (2000*input)/273
    MOV CX,20000
    SUB AX,CX      ; AX = (2000*input)/273 - 20000
                   ; result is multiplied by 10
                   ; to keep last digit as the decimal afterwards
    CALL PRINT_TEMPERATURE
    JMP REPEAT

FIRST_REGION:      ; T = (200*input)/819
    MOV CX,2000    ; mul by 2000 (to keep last digit as the decimal
afterwards)
    MUL CX
    MOV CX,819
    DIV CX         ; result in AX
    CALL PRINT_TEMPERATURE
    JMP REPEAT

SECOND_REGION:     ; T = 250 + (100*input)/819
    MOV CX,1000
    MUL CX         ; DX:AX = input*1000
    MOV CX,819
    DIV CX         ; AX = (1000*input)/819
    MOV CX,2500
    ADD AX,CX      ; AX = 2500 + (1000*input)/819
                   ; result is multiplied by 10
                   ; to keep last digit as the decimal afterwards
    CALL PRINT_TEMPERATURE
    JMP REPEAT

    ; if temperature exceeds 999.9 degrees
    ERROR_PART:
    PRNT_STR  ERROR_MSG
    JMP REPEAT
    ; -------------------------------

END_ALL:
    PRNT_STR END_MSG
    EXIT
MAIN ENDP

; --------- AUXILIARY ROUTINES -----------

; routine get a Y or a N (ignore all there characters)
YES_OR_NO PROC NEAR
IGNORE:
    READ
    CMP AL,'Y'
    JE GOT_IT
    CMP AL,'N'
    JE GOT_IT
    JMP IGNORE
GOT_IT:
    RET
YES_OR_NO   ENDP

; routine to input a hex number
GET_INPUT 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 GOT_INPUT

CHECK_LETTER:
    CMP AL,'N'         ; if input = 'N', then return to quit
    JE GOT_INPUT
```

```
    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

GOT_INPUT:
    RET
GET_INPUT ENDP

; routine to put input in AX
TIDY_INPUT PROC NEAR
    PUSH BX

    MOV AH,FIRST      ; AH = 0000XXXX

    MOV AL,SECOND     ; AL = 0000YYYY
    SAL AL,4
    AND AL,0F0H       ; AL = YYYY0000

    MOV BL,THIRD
    AND BL,0FH        ; BL = 0000ZZZZ
    OR  AL,BL         ; AL = YYYYZZZZ
                      ; AX = 0000XXXX YYYYZZZZ (FULL NUMBER)
    POP BX
    RET
TIDY_INPUT ENDP

; routine to print temperature (it's in AX)
PRINT_TEMPERATURE PROC NEAR
    PUSH AX
    PRNT_STR TEMP
    POP AX

    MOV CX,0          ; initialize counter
SPLIT:
    MOV DX,0
    MOV BX,10
    DIV BX            ; take the last decimal digit
    PUSH DX           ; save it
    INC CX
    CMP AX,0
    JNE SPLIT         ; continue, till we split the whole number

    DEC CX
    CMP CX,0
    JNE PRNT_
    PRINT '0'
    JMP ONLY_DECIMAL

PRNT_:
    POP DX            ; print the digits we saved in reverse
    PRINT_DEC
    LOOP PRNT_

ONLY_DECIMAL:
    PRINT '.'         ; the last digit is the decimal
    POP DX
    PRINT_DEC

    PRINT ' '
    PRINT 0F8H
    PRINT 'C'
    PRNT_STR NEWLINE

    RET
PRINT_TEMPERATURE ENDP

CODE_SEG    ENDS
    END MAIN
```

Τελευταίο παρατίθεται το αρχείο MACROS.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
```