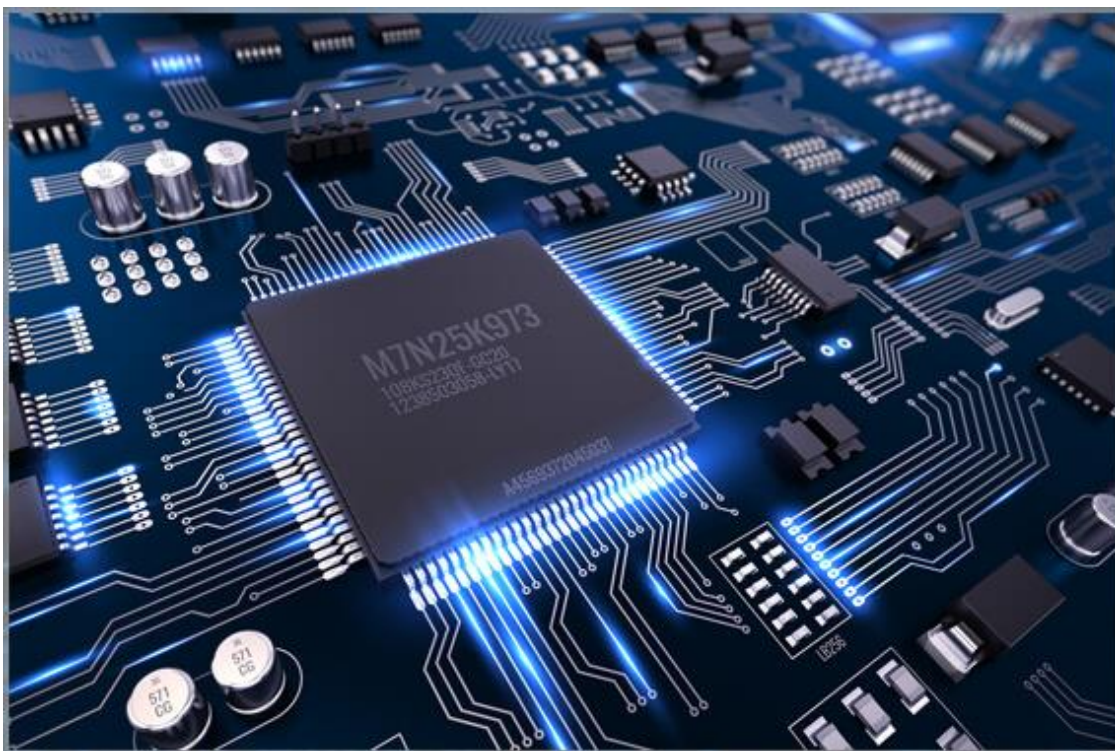


ΣΥΣΤΗΜΑΤΑ ΜΙΚΡΟΥΠΟΛΟΓΙΣΤΩΝ

5Η ΣΕΙΡΑ ΑΣΚΗΣΕΩΝ



JUNE 20, 2021

ΘΟΔΩΡΗΣ ΑΡΑΠΗΣ – EL18028
ΚΡΙΣ ΚΟΥΤΣΗ – EL18905



ΣΥΣΤΗΜΑΤΑ ΜΙΚΡΟΥΠΟΛΟΓΙΣΤΩΝ

Άσκηση 1

```
PRINT MACRO CHAR                ;MACRO to print a character
    PUSH AX
    PUSH DX
    MOV DL,CHAR                  ;Char must be defined
    MOV AH,2                     ;before translation
    INT 21H
    POP DX
    POP AX
ENDM

DATA_SEG SEGMENT
    TABLE DB 128 DUP(?)        ;Initialise TABLE of 128 bytes
    LINEFEED DB 13, 10, "$"
DATA_SEG ENDS

CODE_SEG SEGMENT
    ASSUME CS:CODE_SEG, DS:DATA_SEG
MAIN PROC FAR
START: MOV AX,DATA_SEG
      MOV DS,AX
      MOV CX,128                ;Repeat 128 times
      MOV DI,0                  ;Index of Table
STORE: MOV TABLE[DI],CL        ;Store values
      INC DI
      LOOP STORE                ;Loop 128 times

;a
      MOV AH,0
      MOV DX,0
      MOV CX,64                 ;Repeat 64 times, as the number of odd numbers
      MOV DI,1                 ;Start from 127
SUM:   MOV AL,TABLE[DI]
      ADD DX,AX
      ADD DI,2
      LOOP SUM
      MOV CX,0                  ;Initialise digit counter
      MOV AX,DX
      MOV BL,40H
      DIV BL
ADDR2: MOV DX,0
      MOV BX,10
      DIV BX                    ;Divide number with 10
      PUSH DX                   ;Save the rest to the stack
      INC CX                    ;Increase counter
      CMP AX,0                  ;If quotient is 0 then there isnt any digit left
      JNE ADDR2
ADDR3: POP DX                   ;Read one digit from the stack
      ADD DX,30H                ;Calculate ASCII code and print
      PRINT DL                  ;the corresponding character on the screen
      LOOP ADDR3
      PUSH AX
      PUSH DX
      MOV AH,09
      MOV DX,OFFSET LINEFEED
      INT 21H
      POP DX
      POP AX

;b
      MOV DI,0
      MOV BH,TABLE[DI]          ;BH for max, BL for min
      MOV BL,TABLE[DI]
      MOV CX,127                ;127 because we start the LOOP from TABLE[1]

MAX_MIN:
      INC DI
      CMP BH,TABLE[DI]          ;If BH<TABLE[DI] -> CF=1
      JC MAX
      CMP TABLE[DI],BL          ;If TABLE[DI]<BL -> CF=1
      JC MIN
      LOOP MAX_MIN
L:     MOV DL,BH
      CALL PRINT_8BIT_HEX
      PRINT ' '
      MOV DL,BL
      CALL PRINT_8BIT_HEX
      JMP QUIT

MAX:   MOV BH,TABLE[DI]          ;Store new MAX
      JMP L
MIN:   MOV BL,TABLE[DI]          ;Store new MIN
      JMP L

QUIT:  HLT
MAIN ENDP
```

```

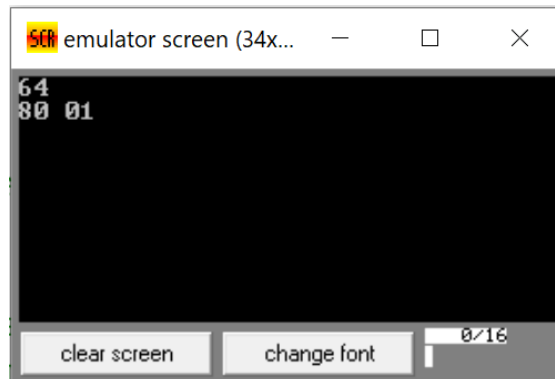
PRINT_8BIT_HEX PROC NEAR ;Print 8bit number in hex
    MOV AL,DL
    AND DL,0F0H ;Isolate 4 MSB
    MOV CL,4
    ROR DL,CL
    CALL PRINT_HEX
    MOV DL,AL
    AND DL,0FH ;Isolate 4 LSB
    CALL PRINT_HEX
    RET
PRINT_8BIT_HEX ENDP

PRINT_HEX PROC NEAR
    CMP DL, 9 ;If number is between 0 and 9 add 30H
    JG ADDR4
    ADD DL, 30H
    JMP ADDR5
ADDR4: ADD DL, 37H ;Else add 37H
ADDR5: PRINT DL ;Print character
    RET
PRINT_HEX ENDP

CODE_SEG ENDS
END MAIN

```

Παρακάτω φαίνεται η λειτουργία του προγράμματος:



Άσκηση 2

```
PRINT MACRO CHAR                ;MACRO to print a character
    PUSH AX
    PUSH DX
    MOV DL, CHAR                ;Char must be defined
    MOV AH, 2                  ;before translation
    INT 21H
    POP DX
    POP AX
ENDM

READ MACRO                      ;MACRO to read from keyboard
    MOV AH, 8                  ;The value of ASCII char is returned
    INT 21H                    ;through AL
ENDM

PRINT_STR MACRO STRING          ;MACRO to print string
    PUSH DX
    PUSH AX
    MOV DX, OFFSET STRING
    MOV AH, 9
    INT 21H
    POP AX
    POP DX
ENDM

NEWLINE MACRO LINEFEED          ;MACRO to print new line
    PUSH AX
    PUSH DX
    MOV AH, 09
    MOV DX, OFFSET LINEFEED
    INT 21H
    POP DX
    POP AX
ENDM

DATA_SEG SEGMENT
    LINEFEED DB 13, 10, "$"
    MSG1 DB 'Z=$'
    MSG2 DB ' W=$'
    MSG3 DB 'Z+W=$'
    MSG4 DB ' Z-W=$'
    MSG5 DB ' Z-W=-$'
DATA_SEG ENDS

CODE_SEG SEGMENT
    ASSUME CS:CODE_SEG, DS:DATA_SEG
MAIN PROC FAR
    MOV AX, DATA_SEG
    MOV DS, AX

START:
; <1>
    CALL DEC_KEYB              ;Get tens of Z
    MOV BH, AL                 ;Store tens of Z to BH
    CALL DEC_KEYB              ;Get units of Z
    MOV BL, AL                 ;Store units of Z to BL
    CALL DEC_KEYB              ;Get tens of W
    MOV DH, AL                 ;Store tens of W to DH
    CALL DEC_KEYB              ;Get units of W
    MOV DL, AL                 ;Store units of W to DL
    PRINT_STR MSG1
    PRINT BH
    SUB BH, 30H                ;Subtract 30 to get the initial number
    PRINT BL
    SUB BL, 30H
    PRINT_STR MSG2
    PRINT DH
    SUB DH, 30H
    PRINT DL
    SUB DL, 30H
```

```

; <2>
NEWLINE LINEFEED
MOV AL, 10          ; Multiply tens of Z with 10 to
MUL BH              ; store the whole number in BL
ADD BL, AL
MOV AL, 10
MUL DH              ; Same with W in DL
ADD DL, AL
MOV AL, DL          ; Save W in AL
PUSH AX             ; Push AL because it will change
ADD DL, BL          ; Store Z+W in DL
PRINT_STR MSG3
CALL PRINT_8BIT_HEX
POP AX              ; Retrieve W
CMP BL, AL          ; If Z < W then jump to NEGATIVE
JC NEGATIVE
SUB BL, AL          ; Else store Z-W in BL
MOV DL, BL          ; and move it to DL to print it
PRINT_STR MSG4
CALL PRINT_8BIT_HEX
NEWLINE LINEFEED
NEWLINE LINEFEED
JMP START           ; Get new data
NEGATIVE:
SUB AL, BL          ; Store W-Z in AL to get the absolute value
MOV DL, AL          ; and move it to DL to print it
PRINT_STR MSG5
CALL PRINT_8BIT_HEX
NEWLINE LINEFEED
NEWLINE LINEFEED
JMP START           ; Get new data

MAIN ENDP

PRINT_8BIT_HEX PROC NEAR ; Print 8bit number in hex
    MOV AL, DL
    AND DL, 0F0H         ; Isolate 4 MSB
    MOV CL, 4
    ROR DL, CL
    CALL PRINT_HEX
    MOV DL, AL
    AND DL, 0FH          ; Isolate 4 LSB
    CALL PRINT_HEX
    RET
PRINT_8BIT_HEX ENDP

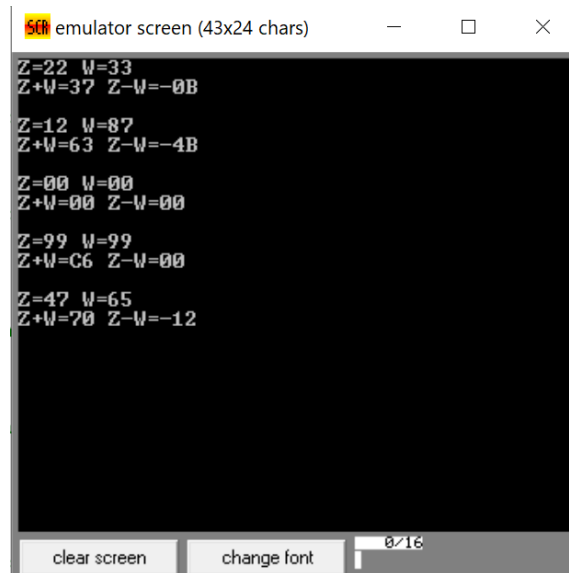
PRINT_HEX PROC NEAR
    CMP DL, 9            ; If number is between 0 and 9 add 30H
    JG ADDR4
    ADD DL, 30H
    JMP ADDR5
ADDR4:
    ADD DL, 37H          ; Else add 37H
ADDR5:
    PRINT DL             ; Print character
    RET

DEC_KEYB PROC NEAR
IGNORE:
    READ                 ; Read char from keyboard
    CMP AL, 30H          ; Check if char is decimal digit
    JL IGNORE            ; If it isn't then wait for a valid input
    CMP AL, 39H
    JG IGNORE
    RET
DEC_KEYB ENDP

CODE_SEG ENDS
END MAIN

```

Παρακάτω φαίνεται η λειτουργία του προγράμματος:



```
scf emulator screen (43x24 chars)
Z=22 W=33
Z+W=37 Z-W=-0B
Z=12 W=87
Z+W=63 Z-W=-4B
Z=00 W=00
Z+W=00 Z-W=00
Z=99 W=99
Z+W=C6 Z-W=00
Z=47 W=65
Z+W=70 Z-W=-12
clear screen change font 0/16
```

Άσκηση 3

```
PRINT MACRO CHAR                ;MACRO to print a character
    PUSH AX
    PUSH DX
    MOV DL,CHAR                  ;Char must be defined
    MOV AH,2                    ;before translation
    INT 21H
    POP DX
    POP AX
ENDM

READ MACRO                       ;MACRO to read from keyboard
    MOV AH, 8                   ;The value of ASCII char is returned
    INT 21H                     ;through AL
ENDM

NEWLINE MACRO LINEFEED           ;MACRO to print new line
    PUSH AX
    PUSH DX
    MOV AH,09
    MOV DX,OFFSET LINEFEED
    INT 21H
    POP DX
    POP AX
ENDM

DATA_SEG SEGMENT
    LINEFEED DB 13, 10, "$"
DATA_SEG ENDS

CODE_SEG SEGMENT
    ASSUME CS:CODE_SEG, DS:DATA_SEG
MAIN PROC FAR
    MOV AX,DATA_SEG
    MOV DS,AX

START:
    MOV BX,0
    CALL HEX_KEYB                ;Get first hex digit
    CMP AL, 'T'                  ;If it's T then end
    JE QUIT                      ;Else store its value to BH
    MOV BH,AL                    ;BX will have the 12 bit number
    CALL HEX_KEYB                ;Get second hex digit
    CMP AL, 'T'
    JE QUIT
    MOV BL,AL                    ;Store its value to BL
    MOV CL,4                     ;Rotate the value to 4 MSB
    ROL BL,CL                    ;Get third hex digit
    CALL HEX_KEYB
    CMP AL, 'T'
    JE QUIT
    ADD BL,AL                    ;BL has the 2nd and 3rd digit
    MOV AX,BX
    MOV CX,0
    CALL PRINT_DEC
    PRINT '='
    CALL PRINT_BIN
    PRINT '='
    CALL PRINT_OCT
    NEWLINE LINEFEED
    JMP START

QUIT:
    HLT
MAIN ENDP
```

```

PRINT_OCT PROC NEAR
    PUSH AX
    MOV BX,8
    MOV CX,0
GETOCT:
    MOV DX,0
    DIV BX
    PUSH DX
    INC CL
    CMP AX,0
    JNE GETOCT
PRINTOCT:
    POP AX
    ADD AL,48
    PRINT AL
    LOOP PRINTOCT
    POP AX
    RET
PRINT_OCT ENDP

PRINT_BIN PROC NEAR
    PUSH AX
    MOV BX,2
    MOV CX,0
GETBIN:
    MOV DX,0
    DIV BX
    PUSH DX
    INC CL
    CMP AX,0
    JNE GETBIN
PRINTBIN:
    POP AX
    ADD AL,48
    PRINT AL
    LOOP PRINTBIN
    POP AX
    RET
PRINT_BIN ENDP

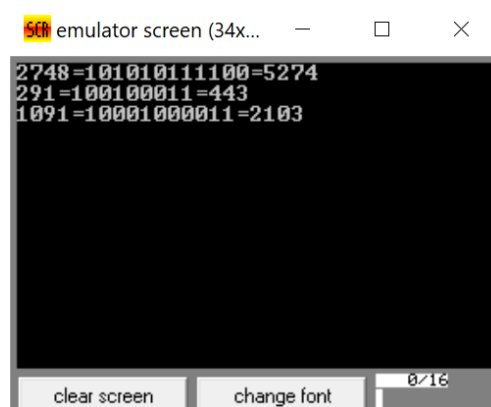
PRINT_DEC PROC NEAR
    PUSH AX
    MOV BX,10
    MOV CX,0
GETDEC:
    MOV DX,0
    DIV BX
    PUSH DX
    INC CX
    CMP AX,0
    JNE GETDEC
PRINTDEC:
    POP DX
    ADD DL,48
    PRINT DL
    LOOP PRINTDEC
    POP AX
    RET
PRINT_DEC ENDP

HEX_KEYB PROC NEAR
    IGNORE:
        READ
        CMP AL, 'T'
        JE ADDR2
        CMP AL, 30H
        JL IGNORE
        CMP AL, 39H
        JG ADDR1
        SUB AL,30H
        JMP ADDR2
    ADDR1:
        CMP AL, 'A'
        JL IGNORE
        CMP AL, 'F'
        JG IGNORE
        SUB AL,37H
    ADDR2:
        RET
HEX_KEYB ENDP
CODE_SEG ENDS
END MAIN

```

;Print 12bit number in octal
 ;We have to divide number with 8
 ;Initialise counter
 ;DX will have the remainder of the division
 ;Divide number with 8
 ;Save the remainder to the stack
 ;Increase counter
 ;If quotient is 0 then there isnt any digit left
 ;Read one digit from the stack
 ;Calculate ASCII code and print the
 ;corresponding character on the screen
 ;Print 12bit number in binary
 ;We have to divide number with 2
 ;We follow the same process as above
 ;but instead of dividing by 8
 ;we divide number with 2
 ;Calculate ASCII code and print the
 ;corresponding character on the screen
 ;Print 12bit number in decimal
 ;We have to divide number with 10
 ;We follow the same process as above
 ;but instead of dividing by 2
 ;we divide number with 10
 ;Calculate ASCII code and print
 ;the corresponding character on the screen
 ;Read a hex digit from the keyboard
 ;and store it in AL
 ;Read from keyboard
 ;If char is T then end routine
 ;Check if char is a digit
 ;If it isn't then wait for a valid input
 ;Check if char is a digit greater than 9
 ;If it is then check if its A,B,C,D,E or F
 ;Export the correct number from its ASCII value
 ;Check if char is a valid hex digit
 ;If it isn't then wait for a valid input
 ;Export the correct number from its ASCII value

Παρακάτω φαίνεται η λειτουργία του προγράμματος για εισόδους ABC, 123, 443 αντίστοιχα. Επιλέξαμε να μην εκτιπώνουμε την είσοδο (που είναι σε δεκαεξαδική μορφή) επειδή δεν διευκρινιζόταν στην εκφώνηση. Αν θέλαμε να το εκτυπώσουμε απλά θα κάναμε print τους χαρακτήρες που εισάγουμε και στην συνέχεια ένα “=”.



```
emulator screen (34x...  
2748=101010111100=5274  
291=100100011=443  
1091=10001000011=2103  
clear screen change font 0/16
```

Άσκηση 4

```
PRINT MACRO CHAR                ;MACRO to print a character
    PUSH AX
    PUSH DX
    MOV DL,CHAR                  ;Char must be defined
    MOV AH,2                     ;before translation
    INT 21H
    POP DX
    POP AX
ENDM

READ MACRO                      ;MACRO to read from keyboard
    MOV AH, 8                   ;The value of ASCII char is returned
    INT 21H                     ;through AL
ENDM

NEWLINE MACRO LINEFEED          ;MACRO to print new line
    PUSH AX
    PUSH DX
    MOV AH,09
    MOV DX,OFFSET LINEFEED
    INT 21H
    POP DX
    POP AX
ENDM

DATA_SEG SEGMENT
    LINEFEED DB 13, 10, "$"
    TABLE DB 20 DUP(?)        ;Table to store the characters from keyboard
DATA_SEG ENDS

CODE_SEG SEGMENT
    ASSUME CD:CODE_SEG, DS:DATA_SEG
MAIN PROC FAR
    MOV AX,DATA_SEG
    MOV DS,AX

START:
    MOV DI,0                    ;Initialise table pointer
    MOV CL,0                    ;Initialise counter
GETCHAR:
    READ                        ;Read from keyboard
    CMP AL,61                   ;If we typed '=' then end
    JE QUIT
    CMP AL,13                   ;If we pressed enter then go
    JE PRINT_ANS               ;go to printing process
    CMP AL,48                   ;ASCII values less than 48 don't
    JB GETCHAR                 ;have a value that we want
    CMP AL,122                  ;ASCII values greater than 122 don't
    JA GETCHAR                 ;have a value that we want
    CMP AL,57                   ;If ASCII value is between 48 and 57 then
    JBE SAVECHAR               ;char is a number, so store it in the table
    CMP AL,97                   ;If ASCII value is between 58 and 96 then
    JB GETCHAR                 ;we don't have a value that we want
                                ;Else the value is between 97 and 122, so
                                ;char is a case letter of the alphabet
```

```

SAVECHAR:
    PRINT AL
    MOV TABLE[DI],AL
    INC DI
    INC CL
    MOV DL,CL
    CMP CL,20
    JB GETCHAR
PRINT_ANS:
    NEWLINE LINEFEED
    MOV DI,0
GETCAP:
    MOV AL,TABLE[DI]
    CMP AL,97
    JB NEXTCAP
    CMP AL,122
    JA NEXTCAP
    SUB AL,32
    PRINT AL
NEXTCAP:
    INC DI
    LOOP GETCAP
    PRINT '-'
    MOV CL,DL
    MOV DI,0
GETNUM:
    MOV AL,TABLE[DI]
    CMP AL,48
    JB NEXNUM
    CMP AL,57
    JA NEXNUM
    PRINT AL
NEXNUM:
    INC DI
    LOOP GETNUM
    NEWLINE LINEFEED
    NEWLINE LINEFEED
    JMP START
QUIT:
    HLT
MAIN ENDP
CODE_SEG ENDS
END MAIN

```

;Store char in table
 ;Increase table pointer
 ;Increase counter
 ;DL will be used to restore CL's value
 ;Check if we typed 20 characters
 ;If not then read next char

 ;Initialise table pointer

 ;Getting case letters from the
 ;table and making them capital

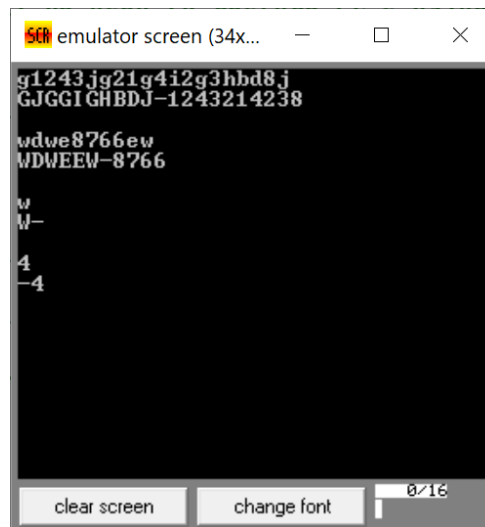
 ;Subtracting 32 makes this transformation
 ;Print capital letter

 ;Increase pointer

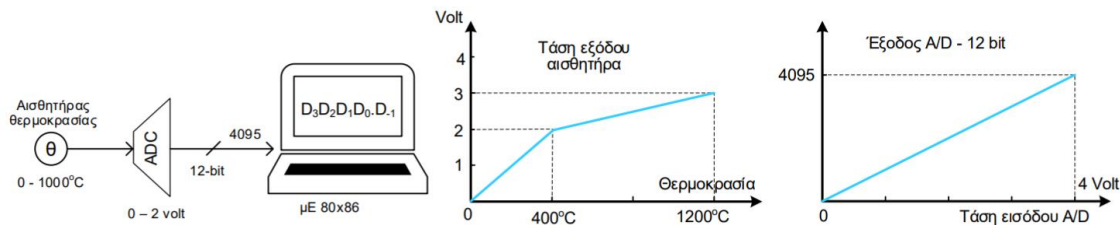
 ;Restoring CL's value because
 ;it's needed in GETNUM's loop

 ;Getting all numbers from
 ;table and printing them

Παρακάτω φαίνεται η λειτουργία του προγράμματος, αρχικά με 20 χαρακτήρες (χωρίς να πατήσουμε ENTER), και στις υπόλοιπες 3 εισόδους πατώντας ENTER.



Άσκηση 5



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 $\mu\text{E } 80\text{x}86$. 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.

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

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

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

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.

```
PRINT_STR MACRO STRING      ;MACRO to print string
    PUSH DX
    PUSH AX
    MOV DX, OFFSET STRING
    MOV AH, 9
    INT 21H
    POP AX
    POP DX
ENDM

READ MACRO                  ;MACRO to read from keyboard
    MOV AH, 8                ;The value of ASCII char is returned
    INT 21H                  ;through AL
ENDM

PRINT MACRO CHAR            ;MACRO to print a character
    PUSH AX
    PUSH DX
    MOV DL, CHAR              ;Char must be defined
    MOV AH, 2                ;before translation
    INT 21H
    POP DX
    POP AX
ENDM

NEWLINE MACRO LINEFEED      ;MACRO to print new line
    PUSH AX
    PUSH DX
    MOV AH, 09
    MOV DX, OFFSET LINEFEED
    INT 21H
    POP DX
    POP AX
ENDM

DATA SEGMENT
    LINEFEED DB 13, 10, "$"
    STARTPROMPT DB "START(Y,N):$" ;Starting message
    ERRORMSG DB "ERROR$"
ENDS

CODE SEGMENT
    ASSUME CS:CODE, DS:DATA
MAIN PROC FAR
    MOV AX, DATA
    MOV DS, AX
    PRINT_STR STARTPROMPT
```

| | | |
|--------------------|--|---|
| START: | | |
| READ | | ;Wait until Y or N is typed |
| CMP AL,'N' | | ;If we typed N |
| JE QUIT | | ;then end |
| CMP AL,'Y' | | ;If we typed Y |
| JE CONT | | ;then go to main process |
| JMP START | | |
| CONT: | | |
| PRINT AL | | ;Print Y |
| NEWLINE LINEFEED | | |
| NEWLINE LINEFEED | | |
| NEWTEMP: | | |
| MOV DX,0 | | |
| MOV CX,3 | | ;Initialise for 3 hex digits |
| READTEMP: | | ;Input |
| CALL HEX_KEYB | | ;Read input from keyboard |
| CMP AL,'N' | | |
| JE QUIT | | |
| PUSH CX | | ;DX will have the 12bit input |
| DEC CL | | |
| ROL CL,2 | | |
| MOV AH,0 | | |
| ROL AX,CL | | ;Rotate left 8, 4, 0 digits |
| OR DX,AX | | ;Add digit to reg |
| POP CX | | |
| LOOP READTEMP | | |
| PRINT ', ' | | |
| MOV AX,DX | | |
| CMP AX,2047 | | ;U<=2 ? |
| JBE BRANCH1 | | |
| CMP AX,3071 | | ;U<=3 ? |
| JBE BRANCH2 | | |
| PRINT_STR ERRORMSG | | ;U>3 |
| NEWLINE LINEFEED | | |
| JMP NEWTEMP | | |
| BRANCH1: | | ;1st branch: U<=2, T=(800*U) div 4095 |
| MOV BX,800 | | |
| MUL BX | | |
| MOV BX,4095 | | |
| DIV BX | | |
| JMP SHOWTEMP | | |
| BRANCH2: | | ;2nd branch: 2<U<=3, T=((3200*U) div 4095)-1200 |
| MOV BX,3200 | | |
| MUL BX | | |
| MOV BX,4095 | | |
| DIV BX | | |
| SUB AX,1200 | | |
| SHOWTEMP: | | |
| CALL PRINT_DEC16 | | ;Show integet |
| MOV AX,DX | | ;Fractional part = (remainder*10) div 4095 |
| MOV BX,10 | | |
| MUL BX | | |
| MOV BX,4095 | | |
| DIV BX | | |
| PRINT ', ' | | |
| ADD AL,48 | | ;Calculate ASCII code and print |
| PRINT AL | | ;fractional part on the screen |
| NEWLINE LINEFEED | | |
| JMP NEWTEMP | | |
| QUIT: | | |
| PRINT AL | | |
| HLT | | |
| MAIN ENDP | | |

```

HEX_KEYB PROC NEAR                ;Routine to get hex values from keyboard
READ1:
    READ
    CMP AL,'N'
    JE RETURN
    CMP AL,48                      ;If ASCII value is between 48
    JL READ1                      ;and 57 then char is a number
    CMP AL,57
    JG LETTER
    PRINT AL
    SUB AL,48                      ;Remove ASCII code to get the correct number
    JMP RETURN
LETTER:
    CMP AL,'A'                    ;A...F
    JL READ1                      ;<A ?
    CMP AL,'F'                    ;>F ?
    JG READ1
    PRINT AL
    SUB AL,55                      ;ASCII code
RETURN:
    RET
HEX_KEYB ENDP

PRINT_DEC16 PROC NEAR             ;Routine to print fractional number
    PUSH DX
    MOV BX,10                     ;We have to divide number with 10
    MOV CX,0                      ;Initialise counter
GETDEC:
    MOV DX,0                     ;DX will have the remainder of the division
    DIV BX                       ;Divide number with 10
    PUSH DX                      ;Save the remainder to the stack
    INC CL                      ;Increase counter
    CMP AX,0                     ;If quotient is 0 then there isnt any digit left
    JNE GETDEC
PRINTDEC:
    POP DX                       ;Read one digit from the stack
    ADD DL,48                    ;Calculate ASCII code and print the
    PRINT DL                     ;corresponding character on the screen
    LOOP PRINTDEC
    POP DX
    RET
PRINT_DEC16 ENDP
CODE ENDS
END MAIN

```

Παρακάτω φαίνεται η λειτουργία του προγράμματος (με είσοδο N τερματίζει):

