

27/6/21

5η Ομάδα Ασκήσεων

Συστήματα Μικροϋπολογιστών

Βαρδάκης Χριστόφορος el18883
Λυμπεράκης Γεώργιος el18881

1^η ΑΣΚΗΣΗ

Στην πρώτη άσκηση καλούμαστε σε πρώτη φάση να αποθηκεύσουμε τους αριθμούς 128, 127, ..., 1 σε διαδοχικές θέσεις μνήμης μεγέθους 1 Byte.

Έπειτα, με βάσει αυτά τα δεδομένα θα υπολογίσουμε το ακέραιο μέρος του μέσου όρου των περιττών αριθμών σε δεκαδική μορφή. Δηλαδή, καλούμαστε να υπολογίσουμε την παράσταση:

$$\left[\frac{2}{N} * \sum_{i=0}^{\frac{N}{2}-1} (2i + 1) \right], \mu\epsilon N = 128$$

$$\rightarrow \left[\frac{2}{128} * \sum_{i=0}^{64-1} (2i + 1) \right] = 64.$$

Καθώς επίσης και το μέγιστο και ελάχιστο σε μέγεθος από το παραπάνω σύνολο δεδομένων, τα οποία σε ξέρουμε από πριν ότι είναι οι αριθμοί **128**, **1**. Εμείς, καλούμαστε να τους υπολογίσουμε και να τους εκτυπώσουμε στην οθόνη εκφρασμένους στο δεκαεξαδικό σύστημα αρίθμησης.

```
;EXERCISE 1 / 5TH GROUP

include "macros_lib.asm"

;-----
DATA SEGMENT
    TABLE DB 128 DUP(?)

DATA ENDS
;-----

;-----
CODE SEGMENT

    ASSUME CS: CODE, DS: DATA

MAIN PROC FAR

    MOV AX, DATA
    MOV DS, AX
```

```

MOV DI, 0
MOV CL, 128

STORE_ARRAY: ;CREATE THE ARRAY
MOV TABLE[DI], CL
INC DI
LOOP STORE_ARRAY

MOV CX, 64
MOV DI, 1
MOV AX, 0
MOV DH, 0

SUM_ODDS: ;ADD TOGETHER ALL THE ODDS NUMBER
MOV DX, TABLE[DI]
ADD AX, DX
ADD DI, 2
LOOP SUM_ODDS

MOV CL, 64
DIV CL

CALL PRINT_DEC_BCD

PRINT_LN

;AL <- MIN, AH <- MAX

MOV AL, TABLE[0]
MOV AH, TABLE[0]
MOV CX, 127
MOV DI, 1

FIND_MIN_MAX: ;FIND MIN-MAX VALUE IN THE ARRAY
CMP CX, 0
JE PRINT
MOV DL, TABLE[DI]
INC DI
DEC CX
CMP AL, DL
JNC NEW_MIN
CMP AH, DI
JNC NEW_MAX
JMP FIND_MIN_MAX

NEW_MIN:
MOV AL, DL
JMP FIND_MIN_MAX

NEW_MAX:
MOV AH, DI
JMP FIND_MIN_MAX

```

```

PRINT:      ;PRINT THE MIN-MAX FROM BEFORE

    MOV DL, AH ;PRINT MAX
    AND DL, 0F0H
    CMP AL, 0
    JE SECOND ;CHECK IF THE FIRST DIGIT IS ZERO
    MOV CL, 4 ;AND SKIP IT
    RCR DL, CL
    CALL PRINT_HEX
    MOV DL, AH
    AND DL, 0FH
    CALL PRINT_HEX

    PRINT ' '

SECOND: ;PRINT MIN
    MOV DL, AL
    AND DL, 0F0H
    JE LAST
    MOV CL, 4
    RCR DL, CL
    CALL PRINT_HEX
LAST:
    MOV DL, AL
    AND DL, 0FH
    CALL PRINT_HEX
    EXIT

MAIN ENDP

PRINT_HEX PROC NEAR ;ROUTINE FOR PRINTING HEXADECIMAL NUMBER
    CMP DL, 9
    JG ADDR1
    ADD DL, 30H
    JMP ADDR2
ADDR1:
    ADD DL, 37H
ADDR2:
    PRINT DL
    RET
PRINT_HEX ENDP

PRINT_DEC_BCD PROC NEAR ;ROUTINE FOR PRINTING DECIMAL NUMBER IN BCD
    FORMAT
    MOV CX, 0
DIGIT:
    MOV DX, 0
    MOV BX, 10D
    DIV BX
    PUSH DX
    INC CX
    CMP AX, 0
    JNE DIGIT

```

```

TIGID:
    POP DX
    MPRINT_DEC
    LOOP TIGID
    RET

PRINT_DEC_BCD ENDP

CODE ENDS
;-----

END MAIN
    
```

Αρχικά, παρουσιάζουμε τη RAM όπου είναι αποθηκευμένος η ζητούμενος πίνακας.

Random Access Memory

0710:0000

update

☒ table

☐ list

0710:0000	80	7F	7E	7D	7C	7B	7A	79-78	77	76	75	74	73	72	71	ΑΔ~> i<zyxwutsr
0710:0010	70	6F	6E	6D	6C	6B	6A	69-68	67	66	65	64	63	62	61	ponmlkjihgfedcb
0710:0020	60	5F	5E	5D	5C	5B	5A	59-58	57	56	55	54	53	52	51	_ ^] \ IZYXWUTSR
0710:0030	50	4F	4E	4D	4C	4B	4A	49-48	47	46	45	44	43	42	41	PONMLKJIHGFE DCB
0710:0040	40	3F	3E	3D	3C	3B	3A	39-38	37	36	35	34	33	32	31	0?>=<;:98765432
0710:0050	30	2F	2E	2D	2C	2B	2A	29-28	27	26	25	24	23	22	21	0/_.-,+*<'8%\$#'''
0710:0060	20	1F	1E	1D	1C	1B	1A	19-18	17	16	15	14	13	12	11	▽▲↕↔↔↓↑±_ΣΠ!t
0710:0070	10	0F	0E	0D	0C	0B	0A	09-08	07	06	05	04	03	02	01	▷*¶.98....±±±±00

Επομένως, η έξοδος του προγράμματος απεικονίζεται στην παρακάτω εικόνα .



2^η ΑΣΚΗΣΗ

Στη δεύτερη άσκηση καλούμαστε να δημιουργήσουμε μια αριθμομηχανή που υποστηρίζει την πρόσθεση και την αφαίρεση διψήφιων δεκαδικών αριθμών και η οποία εκτυπώνει το αποτέλεσμα σε δεκαεξαδική μορφή

```
;EXERCISE 2 / 5TH GROUP

INCLUDE "macros_lib.asm"
;-----
DATA SEGMENT
    STR_Z DB "Z = $"
    STR_W DB "W = $"
    SUM DB "Z + W = $"
    SUBS DB "Z - W = $"

    Z1 DB 0
    Z2 DB 0

    W1 DB 0
    W2 DB 0

DATA ENDS
;-----

;COMMENT:
; Z -> [ Z1 Z2 ]
; W -> [ W1 W2 ]
;-----
CODE SEGMENT

    ASSUME CS:CODE,DS:DATA

    MAIN PROC FAR

    START:

        MOV AX,DATA
        MOV DS,AX

        PRINT_STR STR_Z
        READ ;READ THE FIRST ASCII NUMBER
        PRINT AL

        SUB AL,30H ;MAKE IT DECIMAL
        MOV Z1,AL

        READ ;READ THE SECOND ASCII NUMBER
        PRINT AL

        SUB AL,30H;MAKE IT DECIMAL
        MOV Z2,AL
```

```

PRINT ' '

PRINT_STR STR_W
READ ;READ THE THIRD ASCII NUMBER
PRINT AL

SUB AL,30H ;MAKE IT DECIMAL
MOV W1,AL

READ ;READ THE LAST ASCII NUMBER
PRINT AL

SUB AL,30H;MAKE IT DECIMAL
MOV W2,AL

PRINT_LN
PRINT_STR SUM

MOV AL,Z1
MOV BL,10
MUL BL
MOV Z1,AL ; Z1 = 10*Z1

MOV AL,W1 ; W1 = 10*W1
MOV BL,10
MUL BL
MOV W1,AL

MOV AL,Z1
ADD AL,Z2
MOV Z2,AL ;MAKE THE NUMBER 10*Z1 + Z2

MOV BL,W1
ADD BL,W2
MOV W2,BL ;MAKE THE NUMBER 10*W1 + W2

ADD AL,BL ;CALCULATE Z + W

MOV BL,AL

AND AL,0F0H ;START PRINTING THE NUMBER Z+W
CMP AL,0 ;CHECK IF THE FIRST HEXADECIMAL IS ZERO
JE L4 ; AND SKIP IT
MOV CL,4
RCR AL,CL
MOV DL,AL

CALL PRINT_HEX ;PRINT FIRST DIGIT
L4:
MOV AL,BL
AND AL,0FH
MOV DL,AL

```

```

CALL PRINT_HEX ;PRINT SECOND DIGIT

PRINT ' '
PRINT_STR SUBS

MOV AL,Z2
SUB AL,W2 ;CALCULATE Z - W
MOV BL,AL

CMP AL,0 ;IF THE RESULT IT'S NEGATIVE
JGE CON
PRINT '-' ;PRINT THE MINUS

NEG AL
MOV BL,AL

CON:
AND AL,0F0H
CMP AL,0 ;CHECK IF THE FIRST HEXADECIMAL IS ZERO
JE L1 ; AND SKIP IT
MOV CL,4
RCR AL,CL
MOV DL,AL
CALL PRINT_HEX ;PRINT FIRST DIGIT
L1:
MOV AL,BL
AND AL,0FH
MOV DL,AL
CALL PRINT_HEX ;PRINT SECOND DIGIT

PRINT_LN
PRINT_LN

JMP START

MAIN ENDP

PRINT_HEX PROC NEAR
CMP DL, 9
JG ADDR1
ADD DL, 30H
JMP L2

ADDR1:
ADD DL, 37H
L2:

PRINT DL
RET
PRINT_HEX ENDP

CODE ENDS
;-----
END MAIN

```


Τώρα θα παρουσιάσουμε ενδεικτικά παραδείγματα με στόχο τον έλεγχο της εξόδου του προγράμματος μας.

Οπότε,

```
SCR emulator screen (80x32 chars)
Z = 28 W = 39
Z + W = 43 Z - W = -B

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

Z = 10 W = 10
Z + W = 14 Z - W = 0

Z = 11 W = 13
Z + W = 18 Z - W = -2

Z = 50 W = 49
Z + W = 63 Z - W = 1

Z = 01 W = 99
Z + W = 64 Z - W = -62

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

Z = 20 W = 79
Z + W = 63 Z - W = -3B
```

3^η ΑΣΚΗΣΗ

Στην άσκηση αυτή υλοποιούμε τρεις ρουτίνες, οι οποίες τυπώνουν ένα 12bit αριθμό σε δεκαδική (PRINT_DEC) οκταδική (PRINT_OCT) και δυαδική (PRINT_BIN).

Για τις πρώτες δύο αυτό επιτυγχάνετε με τη διαίρεση του δεδομένου αριθμού με τη βάση και push στη στοίβα του κάθε ψηφίου, ενώ με pop γίνεται στη συνέχεια η εκτύπωση. Για την PRINT_BIN όμως η διαίρεση με μεγάλους αριθμούς δημιουργεί υπερχείλιση. Έτσι χρησιμοποιήσαμε διαφορετική υλοποίηση. Κάναμε αριστερή ολίσθηση με carry και κάθε φορά εξετάζαμε το carry. Μέχρι το carry να γίνει ένα για πρώτη φορά η ρουτίνα δεν τυπώνει τίποτα για να αποφύγουμε τα περιττά μηδενικά στην αρχή του αριθμού. Έπειτα τυπώνει κάθε φορά το carry.

Οι ρουτίνες αυτές φαίνονται παρακάτω.

```

;PRINT DEC-----
PRINT_DEC PROC NEAR
    PUSH BX
    MOV AH,0
    MOV AX,BX
    MOV BL,10
    MOV CX,1
LOOP_10:
    DIV BL                ; DIVIDE AX BY 10
    PUSH AX              ; SAVE AX
    CMP AL,0             ; IF 0 PRINT
    JE PRINT_DIGITS_10
    INC CX                ; INCREMENT DIGITS
    MOV AH,0
    JMP LOOP_10           ;REPEAT
PRINT_DIGITS_10:        ;PRINT
    POP DX
    MOV DL,DH
    MOV DH,0
    ADD DL,30H
    PRINT DL
    LOOP PRINT_DIGITS_10
    POP BX
    RET
ENDP PRINT_DEC

```

```

;PRINT OCT-----
PRINT_OCT PROC NEAR
    PUSH BX
    MOV AH,0
    MOV AX,BX
    MOV BL,8
    MOV CX,1
LOOP_8:
    DIV BL ; DIVIDE AX BY 6
    PUSH AX ;SAVE AX
    CMP AL,0 ; IF 0 PRINT
    JE GOOUT_8
    INC CX ; INCREMET DIGITS
    MOV AH,0
    JMP LOOP_8 ;REPEAT
GOOUT_8:
    MOV DH,AL
    PUSH DX

    POP DX
PRINT_DIGITS_8: ;PRINT
    POP DX
    MOV DL,DH
    MOV DH,0
    ADD DL,30H
    PRINT DL
    LOOP PRINT_DIGITS_8
    POP BX
    RET
ENDP PRINT_OCT

```

```

;PRINT BIN -----
PRINT_BIN PROC NEAR ;HERE WE TAKE A DIFFERENT APPROACH
    PUSH BX ;DUE TO DIVIDE OVERFLOW OF LARGE NUMBERS
    MOV AX,BX
    MOV CX,16
DISCARD_ZEROS: ;REMOVE MSB OF AH AND FRST ZEROS OF BINARY
    SHL BX,1 ; SHIFT LEFT WTH CARRY
    JC PRINT_DIGITS_2 ; IF CARRY IS 1 PRINT
    LOOP DISCARD_ZEROS
    PRINT '0'
    JMP FIN
PRINT_DIGITS_2: ;PRINT THE ONE IN CARRY
    PRINT '1'
    DEC CX ;DECREMENT ITERATOR
PRINT_LOOP:
    SHL BX,1 ; SHIFT LEFT WTH CARRY
    MOV DL,0 ; ADD CARRY TO DL
    ADC DL,0
    ADD DL,30H ;PRINT DL
    PRINT DL
    LOOP PRINT_LOOP
FIN:
    POP BX
    RET
ENDP PRINT_BIN

```

Τέλος έχουμε το πρόγραμμα που δέχεται από το πληκτρολόγιο ένα τριψήφιο δεκαεξαδικό και τον μετατρέπει σε δεκαδικό οκταδικό και δυαδικό χρησιμοποιώντας τις παραπάνω ρουτίνες.

```
;EXERCISE 3 / 5TH GROUP

INCLUDE "macros_lib.asm"
;-----
DATA SEGMENT
    NEWLINE DB 0AH,0DH,'$'
DATA ENDS
;-----

;-----
CODE SEGMENT
    ASSUME CS:CODE_SEG, DS:DASEG

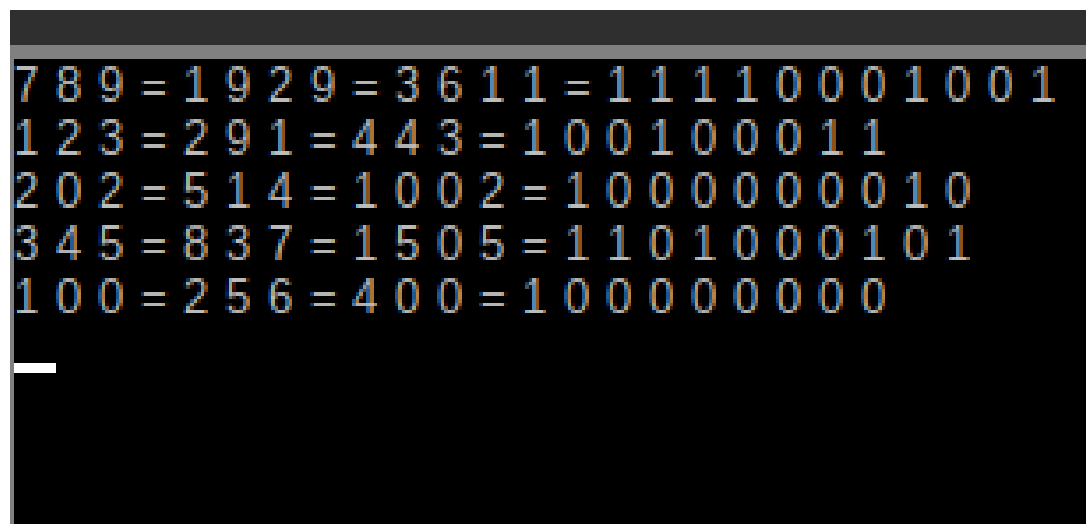
MAIN PROC FAR
    MOV AX,DATA
    MOV DS,AX

START:
    MOV DX,0
    MOV CX,3        ;ITERATOR
ADDR1:
    CALL HEX_KEYB    ;READ HEX
    CMP AL,'T'       ;IF 'T' EXIT
    JE QUIT
    ADD AL,30H        ;PRINT HEX
    PRINT AL
    SUB AL,30H        ;ROTATE CURRENT SUM 4 TIMES
    SHL DX,1
    SHL DX,1
    SHL DX,1
    SHL DX,1
    ADD DL,AL         ;ADD AL TO CURRENT SUM
    LOOP ADDR1        ;REPEAT 3 TIMES
    MOV BX,DX
    PRINT '='
    CALL PRINT_DEC    ;PRINT IN DECIMAL FORM
    PRINT '='
    CALL PRINT_OCT    ;PRINT IN OCTAL FORM
    PRINT '='
    CALL PRINT_BIN    ;PRINT IN BINARY FORM
    PRINT_STR NEWLINE

    JMP START

QUIT:
    EXIT
MAIN ENDP
```

Παράδειγμα εκτέλεσης του προγράμματος φαίνεται παρακάτω



A screenshot of a terminal window with a black background and yellow text. It displays the binary representations of the numbers 0 through 9, arranged in five rows. Each row contains a number followed by an equals sign and its 16-bit binary representation. The numbers are: 7 8 9, 1 2 3, 2 0 2, 3 4 5, and 1 0 0. The binary strings are: 1 9 2 9 = 3 6 1 1 = 1 1 1 1 0 0 0 1 0 0 1, 1 2 3 = 2 9 1 = 4 4 3 = 1 0 0 1 0 0 0 1 1, 2 0 2 = 5 1 4 = 1 0 0 2 = 1 0 0 0 0 0 0 0 1 0, 3 4 5 = 8 3 7 = 1 5 0 5 = 1 1 0 1 0 0 0 1 0 1, and 1 0 0 = 2 5 6 = 4 0 0 = 1 0 0 0 0 0 0 0 0. A white cursor is visible on the line following the last row.

```
7 8 9 = 1 9 2 9 = 3 6 1 1 = 1 1 1 1 0 0 0 1 0 0 1
1 2 3 = 2 9 1 = 4 4 3 = 1 0 0 1 0 0 0 1 1
2 0 2 = 5 1 4 = 1 0 0 2 = 1 0 0 0 0 0 0 0 1 0
3 4 5 = 8 3 7 = 1 5 0 5 = 1 1 0 1 0 0 0 1 0 1
1 0 0 = 2 5 6 = 4 0 0 = 1 0 0 0 0 0 0 0 0
_
```

4^η ΑΣΚΗΣΗ

Στην άσκηση αυτή καλούμαστε να υλοποιήσουμε πρόγραμμα που αφού διαβάσει 20 πεζούς λατινικούς χαρακτήρες ή αριθμούς, να τους τυπώνει στην οθόνη και τέλος να τυπώνει όλους τους λατινικούς μετατρέποντας τους σε κεφαλαίους έπειτα μια παύλα (-) και τέλος συγκεντρωμένους τους αριθμούς.

Το πρόγραμμα πρέπει να είναι συνεχούς λειτουργίας και να τερματίζει όταν δεχθεί τον χαρακτήρα ENTER. Το πρόγραμμα που εκτελεί την παραπάνω λειτουργία φαίνεται παρακάτω:

```
;EXERCISE 4 / 5TH GROUP
INCLUDE "macros_lib.asm"

;-----
DATA SEGMENT
    SYMBOLS DB 20 DUP(?)
    NEWLINE DB 0AH,0DH,'$'
DATA ENDS
;-----

CODE SEGMENT
    ASSUME CS:CODE_SEG, DS:DATA_SEG

MAIN PROC FAR

    MOV AX,DATA
    MOV DS,AX

AGAIN:
    MOV CX,20          ; INITIALIZE COUNTER
    MOV DI,0           ; INITIALIZE ITERATOR
INPUT:
    CALL GET_CHAR      ; GET CHARACTER (ACCEPTABLE ONLY)
    CMP AL,0DH         ; EXIT IF ENTER IS PRESSED
    JE END_OF_PROG
    MOV [SYMBOLS+DI],AL
    PRINT [SYMBOLS+DI] ; PRINT CHARACTER
    INC DI
    LOOP INPUT

OUTPUT:
    PRINT_STR NEWLINE  ; new line

    MOV CX,20
    MOV DI,0

PRINT_LETTERS:
    MOV AL,[SYMBOLS+DI]
    CMP AL,'a'         ; CHECK IF NON-CAPITAL ASCII
    JL NOT_A_LETTER
    CMP AL,'z'
    ; ... (rest of the code is cut off in the image)
```

```

SUB AL,20H           ; MAKE CAPITAL
PRINT AL            ; PRINT CAPITAL

NOT_A_LETTER:
INC DI
LOOP PRINT_LETTERS
PRINT '-'
MOV CX,20
MOV DI,0
PRINT_NUMS:
MOV AL,[SYMBOLS+DI]
CMP AL,30H           ; IF ASCII
JL NOT_A_NUMBER
CMP AL,39H
JG NOT_A_NUMBER
PRINT AL             ; PRINT DIGITS

NOT_A_NUMBER:
INC DI
LOOP PRINT_NUMS
PRINT_STR NEWLINE    ; NEW LINE
JMP AGAIN

END_OF_PROG:
EXIT
MAIN ENDP

GET_CHAR PROC NEAR
IGNORE:
READ
CMP AL,0DH
JE INPUT_OK
CMP AL,30H           ;IF INPUT < 30H ('0') IGNORE
JL IGNORE
CMP AL,39H           ; IF INPUT > 39H ('9') CHECK IF ASCII
JG MAYBE_ASCII
JMP INPUT_OK

MAYBE_ASCII:
CMP AL,'a'           ; IF INPUT < 'a' IGNORE
JL IGNORE
CMP AL,'z'           ; IF INPUT > 'z' IGNORE
JG IGNORE

INPUT_OK:
RET
GET_CHAR ENDP

CODE ENDS
;-----
END MAIN

```

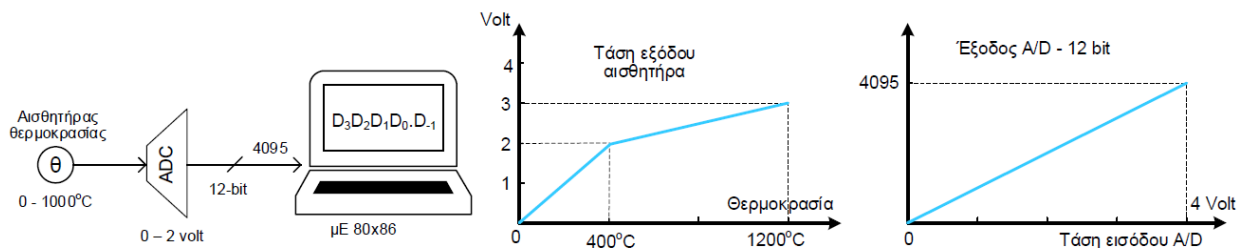
Παράδειγμα της εκτέλεσης του προγράμματος φαίνεται παρακάτω

```
u s 5 4 0 s d k 2 9 0 d j 5 0 m n s 9 2
USSDKDJ MNS - 5 4 0 2 9 0 5 0 9 2
g b 5 4 8 j 0 s 0 a 9 2 j j 1 3 6 d h 4
GBJSAJJ DH - 5 4 8 0 0 9 2 1 3 6 4
8 9 a w h 2 8 d j 2 8 9 2 f 0 2 e 9 d 9
AWHDJ FED - 8 9 2 8 2 8 9 2 0 2 9 9
```

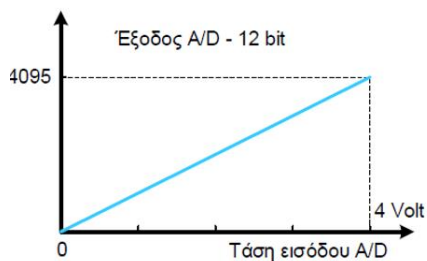

5^η ΑΣΚΗΣΗ

Στην τελευταία άσκηση δημιουργούμε ένα πρόγραμμα το οποίο παρακολουθεί και απεικονίζει θερμοκρασίες στο εύρος 0°C με 1200°C στην οθόνη του υπολογιστικού συστήματος ,εκφρασμένες σε δεκαδική μορφή με ακρίβεια ενός δεκαδικού.

Σχηματικά η διάταξη παρουσιάζεται στο παρακάτω σχήμα όπου μας δίνεται και οι δύο χαρακτηριστικές καμπύλες εξόδου του αισθητήρα και του μετατροπέα αναλογικού και ψηφιακού σήματος.

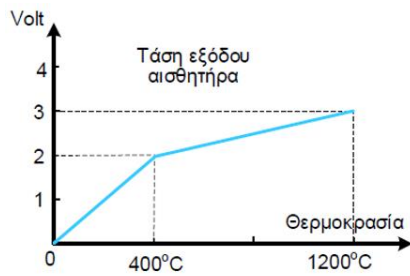


Πιο συγκεκριμένα, από τις παραπάνω χαρακτηριστικές λαμβάνουμε την εξής πληροφορία.



$$y_1 = x_1 * \frac{4095}{4} \text{ or } x_1 = \frac{4}{4095} * y_1$$

with $0 \leq x_1 \leq 4$



$$y_2 = \begin{cases} x_2 * \frac{1}{200} & \text{if } 0 \leq x_2 \leq 400 \\ x_2 * \frac{1}{800} + 1.5 & \text{if } 400 \leq x_2 \leq 1200 \end{cases}$$

Εμείς έχουμε σαν **είσοδο** το σήμα y_1 το οποίο είναι οι 3 δεκαεξαδικοί αριθμοί που τους εισάγει ο χρήστης και η επιθυμητή **έξοδος** μας είναι το σήμα x_2 . Επιπρόσθετα, ισχύει ότι $x_1 \equiv y_2$ και άρα μπορούμε να συνδυάσουμε τις παραπάνω εκφράσεις των χαρακτηριστικών παίρνοντας τις παρακάτω σχέσεις.

$$x_2 = \begin{cases} \frac{800}{4095} * y_1 & \text{if } 0 \leq y_1 \leq 2047 \\ \frac{3200}{4095} * y_1 - 1200 & \text{if } 2047 \leq y_1 \leq 3071 \end{cases}$$

Τέλος, πριν παρουσιάσουμε τον απαιτούμενο κώδικα, για τον υπολογισμό του κλασματικού μέρους έχουμε την παρακάτω σχέση

$$.D_{-1} \dots = \frac{Remainder * 10}{4095}$$

```

;EXERCISE 5 / 5TH GROUP

INCLUDE "macros_lib.asm"
;-----
DATA SEGMENT
    MSG1 DB 'START (Y,N) : $'
    MSG2 DB 0AH,0DH,'VOLTAGE (VOLT) | TEMPRATURE (CELSIUS)$'

    ERROR DB 'ERROR$'

DATA ENDS
;-----
;-----
CODE SEGMENT
    ASSUME CS:CODE,DS:DATA

MAIN PROC FAR
    MOV AX,DATA
    MOV DS,AX
    PRINT_STR MSG1

START:

    READ ;START THE PROGRAM AND WAIT FOR WHAT
    CMP AL,'Y'; THE USER WANTS TO DO (START OR STOP)
    JE CON
    CMP AL,'N'
    JE QUIT
    JMP START

CON:
    PRINT 'Y'
    PRINT_LN
    PRINT_STR MSG2
    PRINT_LN
    PRINT_LN

READ_VOL: ; READ THE FIRST 12 BITS (IN FORM OF 3 HEXADECIMAL)

    MOV BH,16D
    MOV AX,0

    CALL HEX_KEYB ;READ FIRST HEXADECIMAL
    MUL BH
    MUL BH ;SHIFT THE NUMBER 8 TIMES

    MOV DX,AX

    CALL HEX_KEYB ;READ SECOND HEXADECIMAL
    MUL BH ;SHIFT THE NUMBER 4 TIMES
    ADD DX,AX

```

```

CALL HEX_KEYB ;READ THIRD HEXADECIMAL
MOV AH,0
ADD AX,BX      ; LINK THE 3 NUMBERS

PRINT_TAB     ;PREPARE THE USER FOR THE RESULT
PRINT_TAB
PRINT '|'
PRINT_TAB

CMP AX,2047D ;CHECK IN WHICH BRANCH YOUR VOLTAGE IS
JBE BR1
CMP AX,3071D
JBE BR2
PRINT_STR ERROR ; IF YOU ARE HERE THEN YOU HAVE A WRONG INPUT
PRINT_LN
JMP READ_VOL ;GO TO THE NEXT INPUT

BR1:
MOV BX,800D ;BRANCH 1 ->CALCULATE THE EXPRESSION
MUL BX
MOV BX,4095D
DIV BX
JMP PRINT

BR2:
MOV BX,3200D ;BRANCH 1 ->CALCULATE THE EXPRESSION
MUL BX
MOV BX,4095D
DIV BX
SUB AX,1200D

PRINT: ;NOW IT'S THE TIME FOR PRINTING

PUSH DX ;SAVE IS STACK THE REMINDER OF THE ABOVE DIVISION
MOV CX,0

DIGIT: ;ALGORITHM FOR PRINTING IN BCD FORM THE NUMBER
MOV DX,0
MOV BX,10D
DIV BX
PUSH DX
INC CX
CMP AX,0
JNE DIGIT
MOV AX,DX

TIGID: ;NOW POP THE NUMBERS FROM THE STACK
POP DX
MPRINT_DEC ;AND PRINT THEM FROM THE MSB TO LSB
LOOP TIGID

PRINT '.' ;BE READY FOR THE FRACTIONAL PART

```

```

    POP DX
    MOV AX,DX

    MOV BX,10 ;CODE FOR CALCULATING THE FRACTIONAL PART
    MUL BX
    MOV BX,4095
    DIV BX
    MOV DL,AL
    MPRINT_DEC ;PRINT IT

    PRINT_LN ;GO TO THE NEXT LINE

    JMP READ_VOL ; LET'S READ THE NEXT 12BIT

QUIT: ;IF THE USER HAS PRESS THE BUTTON "N" AT ANY MOMENT
    EXIT ;THEN HE GOES HERE AND THE PROGRAM RETURN THE CONTROL
        ;BACK TO THE OPERATING SYSTEM

MAIN ENDP

HEX_KEYB PROC NEAR ;ROUTINE WHICH READ THE ASCII NUMBER AND
                    ;CONVERT IT IN HEXADECIMAL SYSTEM

    PUSH DX

IGNORE1:
    READ
    CMP AL, 'N'
    JE QUIT1

    CMP AL,30H
    JL IGNORE1
    CMP AL,39H
    JG ADDR11
    PUSH AX

PRINT AL
    POP AX
    SUB AL,30H
    JMP ADDR22

ADDR11:
    CMP AL,'A'
    JL IGNORE1
    CMP AL,'F'
    JG IGNORE1
    PUSH AX
    PRINT AL
    POP AX
    SUB AL,37H

```

```

ADDR22:
    POP DX
    RET

QUIT1:
    EXIT

HEX_KEYB ENDP

CODE ENDS
;-----
    END MAIN

```

Κλείνοντας, θα παρουσιάσουμε ενδεικτικά μερικά παραδείγματα από τη χρήση του προγράμματος.

```

START <Y,N>: Y
VOLTAGE <VOLT> : TEMPRATURE <CELSIUS>
000           : 0.0
FFF           : ERROR
FFA           : ERROR
DDD           : ERROR
CCC           : ERROR
BBB           : 1146.6
BFF           : 1199.8
AFB           : 996.6
EAE           : ERROR
700           : 350.0
001           : 0.1
002           : 0.3
AAA           : 933.3
F00           : ERROR
A00           : 800.4

```

clear screen change font 0/16