ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΜ&ΜΥ
Εργαστήριο Μικροϋπολογιστών

4η Εργαστηριακή Άσκηση
Ακ. έτος 2011-2012

Ομάδα C07:

Ελένη Ευαγγελάτου Α.Μ.: 03108050
Γρηγόρης Λύρας Α.Μ.: 03109687
Βασιλεία Φραγκιαδάκη Α.Μ.: 03108026

15 Ιανουαρίου 2012

Άσκηση (i)

Σε αυτή την άσκηση ζητείται αρχικά να φτιάξουμε δύο ρουτίνες, μία που να επιστρέφει τους ascii τιμές του δεκαεξαδικού αριθμού που υπάρχει στον al και μία που να επιστρέφει τον αριθμό στον al, όταν δέχεται ως έισοδο τις ascii τιμές στον ax (στον ah ο ascii των τεσσάρων MSB bits και στον al των τεσσάρων LSB bits). Γι' αυτό το σκοπό στην πρώτη περίπτωση απομονώνουμε ανά τετράδες τα ψηφία και ανάλογα με το αν ο αριθμός είναι μεγαλύτερος ή μικρότερος του 10 επιστρέφουμε την αντίστοιχη ascii τιμή. Στη δεύτερη ρουτίνα βρίσκουμε από την ascii τιμή τον αριθμό που αντιστοιχεί και αφού πολλαπλασιάσουμε τον MSB αριθμό επί 16, τον προσθέτουμε στον LSB αριθμό. Έπειτα φτιάξαμε μια βιβλιοθήκη -ένα αρχείο "lib.inc" - που περιλαμβάνει όλες τις ρουτίνες που χρειαζόμαστε στο πρόγραμμα, βάζοντας τις κατάλληλες δηλώσεις των ρουτινών αλλά και τις αντίστοιχες εντολές στο κυρίως πρόγραμμα ώστε να συμπεριλάβουμε τη βιβλιοθήκη. Ακόμη συμπεριλάβαμε το αρχείο "macros.txt" όπου βάλαμε τις μακροεντολές που χρειαστήκαμε. Στο κυρίως πρόγραμμα διαβάζουμε τους δεκαεξαδικούς αριθμούς που δίνει ο χρήστης, μετρώντας κάθε φορά τα ψηφία ώστε να μπορούμε να "φτιάξουμε" τον αριθμό που αντιστοιχεί σ' αυτή την παράσταση, ακόμα διαβάζουμε την πράξη που θα εκτελέσουμε (πρόσθεση ή αφαίρεση) και σταματάμε με το <enter>. Στην πρόσθεση μπορεί να έχουμε υπερχείλιση, γεγονός που ελέγχουμε και στην αφαίρεση χρειάζεται να ελέγξουμε ποιο από τα δύο ορίσματα είναι μεγαλύτερο (σε unsigned παράσταση) ώστε σε κάθε περίπτωση να εκτυπώσουμε ή όχι το "-" και να αφαιρέσουμε από το μεγαλύτερο το μικρότερο αριθμό.

Κυρίως κώδικας:

```
1  DEFINE_ASCII MACRO
2      LOCAL GRAMMA,PSIFIO,SYNEXEIA,GRAMMA2,PSIFIO2,SYNEXEIA2
3      LOCAL SKIP_ASCII
4      JMP SKIP_ASCII
5
6      ASCII PROC NEAR
7      PUSH BX
8      MOV AH,AL        ; SAVE TON AL
9      AND AL,0FH       ; APOMONWSH TWN 4 LSB BITS
10     CMP AL,09H
11     JG  GRAMMA       ; AN EINAI GRAMMA
12     JMP PSIFIO       ; AN EINAI ARITHMOS
13
14 GRAMMA:
15     ADD AL,37H       ; ASCII GIA KEFALAIA
16     JMP SYNEXEIA
17
18 PSIFIO:
19     ADD AL,30H
20
21 SYNEXEIA:
22     MOV BL,AL        ; SAVE TON AL
23     AND AH,0F0H      ; APOMONWSH TWN 4 MSB BITS
24     SHR AH,4         ; OLIS8HSH 4 8ESEIS DEXIA GIA NA PARW TA 4 MSB BITS
25     CMP AH,09H
26     JG GRAMMA2
27     JMP PSIFIO2
28
29 GRAMMA2:             ; AH ->ASCII 4 MSB BITS
30     ADD AH,37H
31     JMP SYNEXEIA2
32
33 PSIFIO2:
34     ADD AH,30H
35
36 SYNEXEIA2:
37     MOV AL,BL        ; AL ->ASCII TWN 4 LSB BITS
38     POP BX
39 RET
40 ASCII ENDP
41
42 SKIP_ASCII:
43 DEFINE_ASCII ENDM
44
45 DEFINE_NUMBER MACRO
46     LOCAL CONT,CONT1
47     LOCAL SKIP_NUMBER
48     JMP SKIP_NUMBER
```

```
49
50      NUMBER PROC NEAR
51      PUSH BX
52      PUSH CX
53      PUSH DX
54      SUB AL,30H      ; AN O AL EINAI ASCII ARITHMOU
55      CMP AL,09H
56      JLE CONT
57      SUB AL,07H      ; ASCII KEFALAIOU
58      CMP AL,09H
59      JLE CONT
60      SUB AL,20H      ; ASCII MIKROU GRAMMATOS
61
62  CONT:
63      SUB AH,30H      ; ANTISTOIXA GIA TON AH
64      CMP AH,09H
65      JLE CONT1
66      SUB AH,07H
67      CMP AH,09H
68      JLE CONT1
69      SUB AH,20H
70
71  CONT1:
72
73      SHL AH,4        ; POLLAPLASIASMOS TWN 4 MSB ME 16
74      ADD AL,AH       ; AL ->PROS8ESI TWN DYO ARI8MWN
75      MOV AH,0
76      POP DX
77      POP CX
78      POP BX
79  RET
80  NUMBER ENDP
81
82  SKIP_NUMBER:
83  DEFINE_NUMBER ENDM
84
85  DEFINE_PRINT_HEX MACRO
86  ; TYPWNEI TON ANTISTOIXO HEX ARI8MO POU VRISKETAI STON AH
87      LOCAL ADDR,ADR2
88      LOCAL SKIP_PRINT_HEX
89      JMP SKIP_PRINT_HEX
90
91      PRINT_HEX PROC NEAR
92      PUSH BX
93      PUSH CX
94      PUSH DX
95      CMP AH,9        ; AN O ARI8MOS EINAI METAKSY 0 K 9 PROS8ETW 30H
96      JG ADDR
97      ADD AH,30h
98      JMP ADR2
99
100 ADDR:
101     ADD AH, 37h     ; DIAFORETIKA PROS8ETW 37H ('A' = 41H)
102 ADR2:
103     PRINT AH
104     POP DX
105     POP CX
106     POP BX
107     RET
108 PRINT_HEX ENDP
109
110 SKIP_PRINT_HEX:
111 DEFINE_PRINT_HEX ENDM
112
113 DEFINE_PRINT_DEC MACRO
114     LOCAL ADDR1,ADDR2
115     LOCAL SKIP_PRINT_DEC
116     JMP SKIP_PRINT_DEC
117
118 PRINT_DEC PROC NEAR
119 ; TYPWNEI TON ANTISTOIXO DEKADIKO POU VRISKETAI STON AX
120     PUSH DX
121     PUSH CX
122     PUSH BX
123     MOV CX,0
```

```
124
125 ADDR1:
126     MOV DX,0
127     MOV BX,10
128     DIV BX
129     PUSH DX
130     INC CX
131     CMP AX,0
132     JNZ ADDR1
133
134 ADDR2:
135     POP DX
136     PRINT_NUM DX
137     LOOP ADDR2
138     POP BX
139     POP CX
140     POP DX
141     RET
142 PRINT_DEC ENDP
143 SKIP_PRINT_DEC:
144 DEFINE_PRINT_DEC ENDM
145
146
147 DEFINE_READ_HEX MACRO
148 ; DIAVAZEI TOUS DYO ARI8MOUS KAI THN PRAXH(+ H -)
149     LOCAL IGNORE,CONTIN,PROSTHE,CONT2,PSIFIO_1,PSIFIA_2,PSIFIA_3,PSIFIA_4,SYNEX,IGNORE1
150     LOCAL CONT3,CONT4,PSIFIO1_2,PSIFIA2_2,PSIFIA3_2,PSIFIA4_2,TELOS
151     LOCAL SKIP_READ_HEX
152     JMP SKIP_READ_HEX
153
154 READ_HEX PROC NEAR
155     MOV BX,0
156     MOV CX,0
157     MOV DX,0
158
159 IGNORE:
160     READ            ; DIAVAZEI MEXRI 4 HEX PSIFIA, STAMATAEI OTAN DIAVASEI + H -
161     CMP AL,2BH      ; ELEGXOS GIA +
162     JE PROSTHE
163     CMP AL,2DH      ; ELEGXOS GIA '-'
164     JE CONT2        ; 'q'/'Q' TERMATISMOS
165     CMP AL,'Q'
166     JE QUIT
167     CMP AL,'q'
168     JE QUIT
169     CMP BX,4        ; AGNOEI PERA TWN 4 PSIFIWN
170     JE IGNORE
171     CMP AL,30H
172     JL IGNORE       ; AGNOEI PERA TWN HEX PSIFIWN
173     CMP AL,39H
174     JLE CONTIN
175     CMP AL,'A'
176     JL IGNORE
177     CMP AL,'F'
178     JLE CONTIN
179     CMP AL,'a'
180     JL IGNORE
181     CMP AL,'f'
182     JLE CONTIN
183     JMP IGNORE
184
185 CONTIN:
186     MOV AH,0
187     PUSH AX
188     INC BL          ; METRHTHS PSIFIWN
189     JMP IGNORE
190
191 PROSTHE:
192     MOV PROSTH,1    ; SAN FLAG GIA PROS8ESH
193 CONT2:
194     CMP BX,00       ; FTIAXNEI TON ARI8MO ANALOGA ME TON ARI8MO TWN SPIFIWN
195     JZ QUIT
196     CMP BL,1
197     JE PSIFIO_1
198     CMP BL,2
```

```
    JE PSIFIA_2
    CMP BL,3
    JE PSIFIA_3
    CMP BL,4
    JE PSIFIA_4

PSIFIO_1:
    POP AX
    MOV AH,30H      ; AH => '0'
    CALL NUMBER
    JMP SYNEX       ; 1 PSIFIO ->AX

PSIFIA_2:
    POP AX
    MOV CL,AL       ; ENDIAMESOS KATAXWRHTHS GIA TO LSB BIT(OPWS DIVAZOYME AP' TH STOIVA)
    POP AX          ; MSB BIT
    MOV AH,AL
    MOV AL,CL
    CALL NUMBER
    JMP SYNEX       ; 2 PSIFIA -> AX

PSIFIA_3:
    POP AX          ; 3o-2o (2 LSB BITS)
    MOV BL,AL
    POP AX
    MOV AH,AL
    MOV AL,BL
    CALL NUMBER
    MOV DX,AX

    POP AX
    MOV AH,30H      ; 1o PSIFIO
    CALL NUMBER
    MOV AH,AL       ; POL/ZOYME EPI 16^2= 2^8 TO MSB BIT <=> PHGAINEI STON MSB KATAXWRHTH
    MOV AL,DL
    JMP SYNEX       ; 3 PSIFIA ->AX

PSIFIA_4:
    POP AX
    MOV CL,AL
    POP AX
    MOV AH,AL       ; 2o LSB BIT P ANASYROYME AP' TH STOIVA
    MOV AL,CL
    CALL NUMBER
    MOV DX,AX       ; 4o-3o PSIFIO STON DX

    POP AX
    MOV CL,AL
    POP AX          ; 2o-1o BIT
    MOV AH,AL
    MOV AL,CL
    CALL NUMBER
    MOV AH,AL
    MOV AL,DL       ; SAN POL/SMOS EPI 256 TA 2 MSB PSIFIA
                    ; 4 PSIFIA ->AX

SYNEX:              ; 2os ARI8MOS
    PUSH AX         ; APO8IKEYSH 1ou ARI8MOY STH STOIVA

    MOV BX,0        ; MHDENISMOS METRHTH PSIFIWN
IGNORE1:
    READ
    CMP AL,0DH      ; ELEGXOS GIA <ENTER>
    JE CONT4
    CMP AL,'Q'
    JE QUIT
    CMP AL,'q'
    JE QUIT
    CMP BX,4        ; AGNOEI PERA TWN 4 PSIFIWN
    JE IGNORE1      ; KAI TWN HEX PSIFIWN
    CMP AL,30H
    JL IGNORE1
    CMP AL,39H
    JLE CONT3
    CMP AL,'A'
```

```
    JL IGNORE1
    CMP AL,'F'
    JLE CONT3
    CMP AL,'a'
    JL IGNORE1
    CMP AL,'f'
    JLE CONT3
    JMP IGNORE1

CONT3:
    PUSH AX
    MOV AH,0
    INC BL
    JMP IGNORE1

CONT4:
; IDIA DIADIKASIA GIA NA "FTIAXOYME" TO 2o ARI8MO
    CMP BL,00H
    JE QUIT
    CMP BL,1
    JE PSIFIO1_2
    CMP BL,2
    JE PSIFIA2_2
    CMP BL,3
    JE PSIFIA3_2
    CMP BL,4
    JGE PSIFIA4_2


PSIFIO1_2:
    POP AX
    MOV AH,30H
    CALL NUMBER
    JMP TELOS       ; 1 PSIFIO ->AX

PSIFIA2_2:
    POP AX
    MOV CL,AL       ; ENDIAMESOS KATAXWRHTHS GIA TO LSB PSIFIO
    POP AX
    MOV AH,AL
    MOV AL,CL
    CALL NUMBER
    JMP TELOS       ; 2 PSIFIA ->AX

PSIFIA3_2:
    POP AX          ; 3o-2o
    MOV BL,AL
    POP AX
    MOV AH,AL
    MOV AL,BL
    CALL NUMBER
    MOV DX,AX

    POP AX
    MOV AH,30H      ; 1o PSIFIO
    CALL NUMBER
    MOV AH,AL
    MOV AL,DL       ; POL/ZOYME EPI 256 TO 1o PSIFIO
    JMP TELOS       ; 3 PSIFIA ->AX

PSIFIA4_2:
    POP AX
    MOV CL,AL
    POP AX
    MOV AH,AL
    MOV AL,CL
    CALL NUMBER
    MOV DX,AX       ; 4o-3o PSIFIO STON DX

    POP AX
    MOV CL,AL
    POP AX
    MOV AH,AL       ; 1o-2o
    MOV AL,CL
    CALL NUMBER
```

```
    MOV AH,AL
    MOV AL,DL       ; SAN POL/SMOS EPI 256 TA 2 MSB PSIFIA
                    ; 4 PSIFIA ->AX

TELOS:
    MOV DX,AX       ; 2os ARI8MOS ->DX
    POP AX          ; 1os ARI8MOS ->AX
RET
READ_HEX ENDP
SKIP_READ_HEX:
DEFINE_READ_HEX ENDM
```

Κυρίως κώδικας εκτελέσιμου:

```
INCLUDE LIB.INC         ; PERILAMVANOYME TH VILVIO8HKH
INCLUDE MACROS.TXT      ; KAI TIS MAKROENTOLES

DATA SEGMENT
    ; ADD YOUR DATA HERE!
    PROSTH DB 0         ; FLAG GIA PROS8ESI // FLAG GIA PLHN STHN EKTYPWSH THS AFAIRESHS
    MESSAGE DB 3DH,"$"
    GRAMMI DB 0ah,0dh,"$"
    PLHN DB "-$"
ENDS

STACK SEGMENT
    DW   128  DUP(0)
ENDS

CODE SEGMENT
START:
; SET SEGMENT REGISTERS:
    MOV AX, DATA
    MOV DS, AX
    MOV ES, AX

    MOV PROSTH,0        ; ARXIKOPOIHSH FLAG
    CALL READ_HEX       ; 1os ->AX, 2os ->DX

    PUSH AX
    PUSH DX
    PUSH BX
    PRINT_STRING GRAMMI ; ALLAGI GRAMMHS KAI '='
    MOV CL,3DH
    PRINT CL
    POP BX
    POP DX
    POP AX

    CMP PROSTH,1        ; ELEGXOS FLAG GIA PRAXH
    JE PROSTHESI
    JMP AFAIRESH

PROSTHESI:
    ADD AX,DX
    JNC EKTYPWSH
    PUSH AX
    PUSH DX
    PUSH BX
    MOV CL,31H          ; YPERXEILISH STHN PROS8ESI -> 50 PSIFIO = 1
    ;MOV PROSTH,02      ; FLAG GIA KRATOUMENO STHN PROS8ESH
    PRINT CL
    POP BX
    POP DX
    POP AX

EKTYPWSH:               ; EKTYPWSH APOTELESMATOS APO PROS8ESI H AFAIRESH
    PUSH AX             ; APO8IKEUSH ARXIKOU ARI8MOU
    CMP AH,0            ; MHN EKTYPWSEIS TA 2 PRWTA PSIFIA AN EINAI 0
    JE EPOMENO
    PUSH AX             ; ALLIWS KALESE THN ASCII KAI EKTYPWSE TA 2 PSIFIA
    MOV AL,AH           ; ARI8MOS STON AL
    CALL ASCII
    PRINT AH
    PRINT AL
    POP AX
```

```
EPOMENO:
    CALL ASCII          ; TO IDIO GIA TON ARI8MO STON AL
    PRINT AH
    PRINT AL

    PUSH AX
    PUSH DX
    PUSH CX
    MOV BL,3DH          ; '='
    PRINT BL
    POP CX
    POP DX
    POP AX

    POP AX              ; EXAGOUME TON ARXIKO ARI8MO PROS EKTYPWSH

    CMP PROSTH,03       ; ELEGXOS FLAG GIA EKTPYSWSH PLHN
    JNE ADDRESS
    PUSH AX
    PUSH DX
    PRINT_STRING PLHN
    POP DX
    POP AX

ADDRESS:
    CALL PRINT_DEC      ; EKTYPWSH ANTISTOIXOU DEKADIKOU ARI8MOU
    PRINT_STRING GRAMMI ; ALLAGI GRAMMHS

    JMP START

AFAIRESH:
    CMP AX,DX           ; UNSIGNED SYGKRISH ARI8MWN
    JB MIKROTEROS       ; AX < DX
    SUB AX,DX           ; AX >= DX
    JMP EKTYPWSH

MIKROTEROS:
   MOV PROSTH,03        ; FLAG = 03 GIA TYPWSH PLHN
   PUSH AX
   PUSH DX
   PRINT_STRING PLHN
   POP DX
   POP AX
   SUB DX,AX
   MOV AX,DX
   JMP EKTYPWSH

QUIT:
    MOV AX, 4C00H       ; EXIT TO OPERATING SYSTEM.
    INT 21H

ENDS

DEFINE_READ_HEX
DEFINE_PRINT_HEX
DEFINE_PRINT_DEC
DEFINE_ASCII
DEFINE_NUMBER

END START
```

Τα macros που χρησιμοποιήσαμε:

```
READ MACRO
     MOV AH,1
     INT 21H
ENDM

PRINT_NUM MACRO CHAR
    PUSH DX
    PUSH AX
    MOV DX,CHAR
    ADD DX,30H
    MOV AH,2
    INT 21H
    POP AX
```

```
    POP DX

ENDM

PRINT MACRO CHAR
    PUSH AX
    PUSH DX
    MOV DL,CHAR
    MOV AH,02H
    INT 21H
    POP DX
    POP AX
ENDM

PRINT_STRING MACRO STRING
    PUSH AX
    PUSH DX
    MOV DX,OFFSET STRING  ;ASSUME THAT STRING IS A VARIABLE OR CONSTANT, not AN ADDRESS
    MOV AH,09H
    INT 21H
    POP DX
    POP AX
Endm
```

Άσκηση (ii)

Στο μέρος αυτό πραγματοποιήσαμε την μετατροπή του PC σε τερματικό. Γι'αυτόν τον σκοπό δεχόμαστε χαρακτήρες από την θύρα RS232 και στέλνουμε τους χαρακτήρες που πληκτρολογούνται στην ίδια θύρα. Ουσιαστικά έχουμε επικοινωνία μεταξύ δύο τερματικών, όπου ό,τι γράφουμε στην πλευρά αποστολής του ενός το λαμβάνουμε στην πλευρά λήψης του άλλου. Η procedure RXCH_RS232 λαμβάνει έναν χαρακτήρα από την είσοδο (αν έχει σταλεί) και τον καταχωρεί στον AL. Αν δεν έχουμε νέο χαρακτήρα αποθηκεύει στον AL το 0. Αντίστοιχα η procedure TXCH_RS232 στέλνει τον χαρακτήρα που θέλουμε στην εν λόγω θύρα, από τον ίδιο καταχωρητή (AL). Αν δεν έχουμε κάποιον χαρακτήρα να στείλουμε τότε βάζουμε στον AL το 0.

Όσον αφορά τον χωρισμό της οθόνης, έγινε κάθετα. Θεωρήσαμε τις "συντεταγμένες" του μέσου της οθόνης και πάντα στην ίδια στήλη τυπώνουμε τον χαρακτήρα "|" (με ascii κωδικό τον 179) μέσα σε ένα loop. Αναφορικά με την χρήση του πληκτρολογίου, ο χρήστης πραγματοποιεί έξοδο από το περιβάλλον πατώντας το πλήκτρο escape. Επίσης αν πατήσει 'Y' έχουμε λειτουργία echo, δηλαδή όσα γράφει τυπώνονται και στην οθόνη, αλλιώς με 'N' δεν έχουμε echo. Αυτό περιέχεται στην ρουτίνα read_echo. Ακόμη με τα πλήκτρα '1' έως '6' διαλέγει τον ρυθμό επικοινωνίας (baud rate) για τις τιμές 300, 600, 1200, 2400, 4800, και 9600 αντίστοιχα, στην ρουτίνα read_br. Τέλος, παραθέτουμε το αρχείο με τις μακροεντολές μας. Οι μακροεντολές move_there και move_there_cx χρησιμοποιήθηκαν για τη μετάβαση του κέρσορα στις επιλεγμένες συντεταγμένες της οθόνης τόσο για την σωστή τοποθέτηση του χαρακτήρα "|" και όσο και για την εκτύπωση στα δύο "παράθυρα" του τερματικού, καθώς και η μακροεντολή scroll προφανώς για το scroll.

Κυρίως κώδικας:

```
INCLUDE MACROS.TXT

CODE SEGMENT
    ASSUME CS:CODE,DS:CODE,ES:CODE
    ORG 100H

MAIN PROC FAR
    ECHO      DW 0
    ECHO_LINE DW 0
    ECHO_COL  DW 0
    MSG_LINE  DW 0
    MSG_COL   DW 41
    MESSAGE1 DB "FOR ECHO FUNCTION PRESS <Y> . OTHERWISE PRESS <N> ? $"
    MESSAGE2 DB "PLEASE GIVE BAUD RATE. PRESS  <1> FOR 300, <2> FOR 600, <3> FOR 1200, <4> FOR 2400, <5> FOR 4800, <6> FOR9600 $
    NEW_LINE DB 0AH, 0DH, '$'


START:
    CLEAR                          ; CLEAR THE SCREEN
    PRINT_STRING MESSAGE1
    CALL READ_ECHO
    PRINT_STRING NEW_LINE
    PRINT_STRING MESSAGE2
```

```
    CALL READ_BR ;BAUT RATE
    CALL OPEN_RS232
    CLEAR                           ; CLEAR THE SCREEN
    CALL SCREEN_SPLIT               ; PRINT THE LINE IN THE MIDDLE
    MOVE_THERE ECHO_LINE ECHO_COL   ; MOVE TO ECHO LINE COLUMN

READ_RS232:
    CALL RXCH_RS232                 ; READ COM
    CMP AL,0                        ; IF THERE IS NOTHING THERE
    JE NOCHAR                       ; CHECK KEYBOARD
    MOV BL,AL                       ; MOVE IT TO BL FOR PRINTING
; === PRINT THE MESSAGE ===
    CMP BL,0DH
    JE INCMLINE
; === IF GIVEN ENTER SHOULD INC LINE ===
    MOVE_THERE MSG_LINE MSG_COL     ; MOVE CURSOR TO APPROPRIATE LINE COL
    PRINT_THERE BL                  ; OK PRINT IT
    MOV BP,OFFSET MSG_COL           ; AND INCREASE
    MOV BL,DS:[BP]                  ; THE COLUMN
    INC BL                          ; IF IT REACHES 80
    CMP BL,80                       ; IT SHOULD BE RESET TO 41
    JNE NOINCMLINE
INCMLINE:
    INCREASE_LINE MSG_LINE MSG_COL 41       ; INCREASE LINE AND SAVE NEW LINE/COLUMN
    MOVE_THERE MSG_LINE MSG_COL             ; AND MOVE THE CURSOR THERE
; === THIS SHOULD SCROLL UP WHEN NEEDED ===
    MOV BP,OFFSET MSG_LINE
    MOV BL,DS:[BP]
    CMP BL,22
; === COMPARE WITH 22 [LINES IN DOSBOX] ===
    JNE NOCHAR
    SUB BL,1                                ;
    MOV DS:[BP],BL
    SCROLL_UP 1 0 41 23 79
    JMP NOCHAR                              ; IF YOU DON'T NEED TO SCROLL
                                            ; THEN CHECK THE KEYBOARD
NOINCMLINE:
    MOV DS:[BP],BL                  ; STORE THE COLUMN (NO LINE INCREASE NO SCROLL)


NOCHAR:
    READNB
; === NO CHAR THEN READ COM ===
    JZ READ_RS232
    MOV BL,AL
; === PRINT ECHO ===
    CMP BL,27
; === COMPARE WITH ESCAPE ===
    JE QUIT
    CMP BL,8
; === COMPARE WITH BACKSPACE ===
    JE READ_RS232
    CALL TXCH_RS232
; === SHOULD I ECHO R SHOULD I NOT? ===
    MOV BP,OFFSET ECHO
    CMP DS:[BP],1
    JNE READ_RS232
    CMP BL,0DH
    JE  INCELINE
; === IF GIVEN ENTER SHOULD INC LINE ===
    MOVE_THERE ECHO_LINE ECHO_COL
    PRINT_THERE BL
    MOV BP,OFFSET ECHO_COL
    MOV BL,DS:[BP]
    INC BL
    CMP BL,40
    JNE NOINCELINE
INCELINE:
    INCREASE_LINE ECHO_LINE ECHO_COL 0
    MOVE_THERE ECHO_LINE ECHO_COL
; === THIS SHOULD SCROLL UP WHEN NEEDED ===
    MOV BP,OFFSET ECHO_LINE
    MOV BL,DS:[BP]
    CMP BL,22
; === COMPARE WITH 22 [LINES IN DOSBOX] ===
```

```
    JNE READ_RS232
    SUB BL,1
    MOV DS:[BP],BL
    SCROLL_UP 1 0 0 23 39
    JMP READ_RS232

NOINCELINE:
    MOV DS:[BP],BL
    JMP READ_RS232

QUIT:
    CLEAR
    MOV AL,0H
    EXIT

MAIN ENDP


SCREEN_SPLIT PROC NEAR
    MOV CX, 24  ; 24 FORES NA TREXEI TO LOOP

LOOP1:
    MOVE_THERE_CX 40 ; PAEI STHN I GRAMMH KAI STHLH PANTA THN 40 KAI TUPWNEI'|'
    PRINT_THERE 179
    LOOP LOOP1
    MOVE_THERE_CX 40 ; PAEI STHN I GRAMMH KAI STHLH PANTA THN 40 KAI TUPWNEI'|'
    PRINT_THERE 179
    RET

SCREEN_SPLIT ENDP

READ_ECHO PROC NEAR

;PI8ANON NA XREIASTEI PUSH K POP TON AL
LOOP_E:
    READ         ;DIABAZEI XARAKTHRA APO TO PLHKTROLOGIO
    CMP AL, 'Y'  ;AN YES TOTE EXOUME ECHO
    JE ECHO_ON
    CMP AL, 'y'
    JE ECHO_ON
    CMP AL, 'N'
    JE ECHO_OFF
    CMP AL, 'n'
    JE ECHO_OFF
    CMP AL, 27
    JE QUIT       ;AN PATH8EI ESCAPE PHGAINEI STO QUIT
    JMP LOOP_E

ECHO_ON:
    MOV BP,OFFSET ECHO
    MOV DS:[BP],1
    RET
ECHO_OFF:
    RET

READ_ECHO ENDP

READ_BR PROC NEAR

    ;PI8ANON NA XREIASTEI PUSH K POP TON AL
LOOP_BR:
    READ         ;DIABAZEI XARAKTHRA APO TO PLHKTROLOGIO
    CMP AL, '1'  ;AN YES TOTE EXOUME ECHO
    JE BAUD_300
    CMP AL, '2'
    JE BAUD_600
    CMP AL, '3'
    JE BAUD_1200
    CMP AL, '4'
    JE BAUD_2400
    CMP AL, '5'
    JE BAUD_4800
    CMP AL, '6'
    JE BAUD_9600
    CMP AL, 27
```

```
    JE QUIT       ;AN PATH8EI ESCAPE PHGAINEI STO QUIT
    JMP LOOP_BR

;======DIABASMA GIA TO BAUD RATE APO PLHKTROLOGIO=======

BAUD_300:
    MOV AL,43H
    RET
BAUD_600:
    MOV AL,63H
    RET
BAUD_1200:
    MOV AL,83H
BAUD_2400:
    MOV AL,0A3H
    RET
BAUD_4800:
    MOV AL,0C3H
    RET
BAUD_9600:
    MOV AL,0E3H
    RET
READ_BR ENDP


;==== INITIALIZE RS232 PORT ====
OPEN_RS232 PROC NEAR
    JMP BEGIN
BAUD_RATE LABEL WORD
    DW 1047
    DW 768
    DW 385
    DW 192
    DW 96
    DW 48
    DW 24
    DW 12
BEGIN:
    STI
    MOV AH,AL
    MOV DX,03FBH
    MOV AL,80H
    OUT DX,AL

    MOV DL,AH
    MOV CL,4
    ROL DL,CL
    AND DX,0EH
    MOV DI,OFFSET BAUD_RATE
    ADD DI,DX
    MOV DX,03F9H
    MOV AL,CS:[DI]+1
    OUT DX,AL

    MOV DX,03F8H
    MOV AL,CS:[DI]
    OUT DX,AL

    MOV DX,03FBH
    MOV AL,AH
    AND AL,01FH
    OUT DX,AL
    MOV DX,03F9H
    MOV AL,0H
    OUT DX,AL
    RET
OPEN_RS232 ENDP

;==== READ 1 CHAR FROM RS232 PORT ====
RXCH_RS232 PROC NEAR
    MOV DX,3FDH
    IN AL,DX

    TEST AL,1
    JZ NOTHING
```

```
    SUB DX,5
    IN AL,DX
    JMP EXIT2
NOTHING:
    MOV AL,0
EXIT2:
    RET

RXCH_RS232 ENDP

;==== WRITE 1 CHAR TO RS232 PORT ====
TXCH_RS232 PROC NEAR
    PUSH AX
    MOV DX,03FDH

TXCH_RS232_2:
    IN AL,DX
    TEST AL,020H
    JZ TXCH_RS232_2

    SUB DX,5
    POP AX
    OUT DX,AL
    RET
TXCH_RS232 ENDP

END MAIN
```

Τα macros που χρησιμοποιήσαμε:

```
;This macro change registers AH,AL
READ MACRO
    MOV AH,1
    INT 21H
ENDM

;Non blocking read
READNB MACRO
    MOV AH,06H
    MOV DL,0FFH
    INT 21H
ENDM

MOVE_THERE_CX MACRO COL
    PUSH AX
    PUSH BX
    PUSH DX
    MOV AH,02H
    MOV DH,CL
    MOV DL,COL
    MOV BH,0H
    INT 10H
    POP DX
    POP BX
    POP AX
ENDM

MOVE_THERE MACRO LINE COL
    PUSH AX
    PUSH BX
    PUSH DX
    MOV AH,02H
    MOV BP,OFFSET LINE
    MOV DH,DS:[BP]
    MOV BP,OFFSET COL
    MOV DL,DS:[BP]
    MOV BH,0H
    INT 10H
    POP DX
    POP BX
    POP AX
ENDM


INCREASE_LINE MACRO LINE COLUMN MINC
; === Zero Column ===
```

```
    MOV BP,OFFSET COLUMN
    MOV DS:[BP],MINC
; === Increase Line ===
    MOV BP,OFFSET LINE
    INC DS:[BP]

ENDM

SCROLL_UP MACRO LINES UPLR UPLC LORR LORC
    PUSH AX
    PUSH CX
    PUSH DX
    MOV AL,LINES
    MOV CH,UPLR
    MOV CL,UPLC
    MOV DH,LORR
    MOV DL,LORC
    MOV AH,06H
    INT 10H
    POP DX
    POP CX
    POP AX
ENDM

CLEAR MACRO
    XOR CX,CX
    MOV DH,24
    MOV DL,80
    MOV BH,7
    MOV AX,700H
    INT 10H
ENDM

; This will use INT 10/02H
; to move the cursor
; to the appropriate position
PRINT_THERE MACRO CHAR
    PUSH AX
    MOV AL,CHAR
    MOV AH,0EH
    INT 10H
    POP AX
ENDM

;This macro uses registers AH,DX
PRINT_STRING MACRO STRING
    MOV DX,OFFSET STRING ;Assume that string is a variable or constant, NOT an address
    MOV AH,09H
    INT 21H
ENDM

EXIT MACRO
    MOV AH,4CH
    INT 21H
ENDM
```