ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

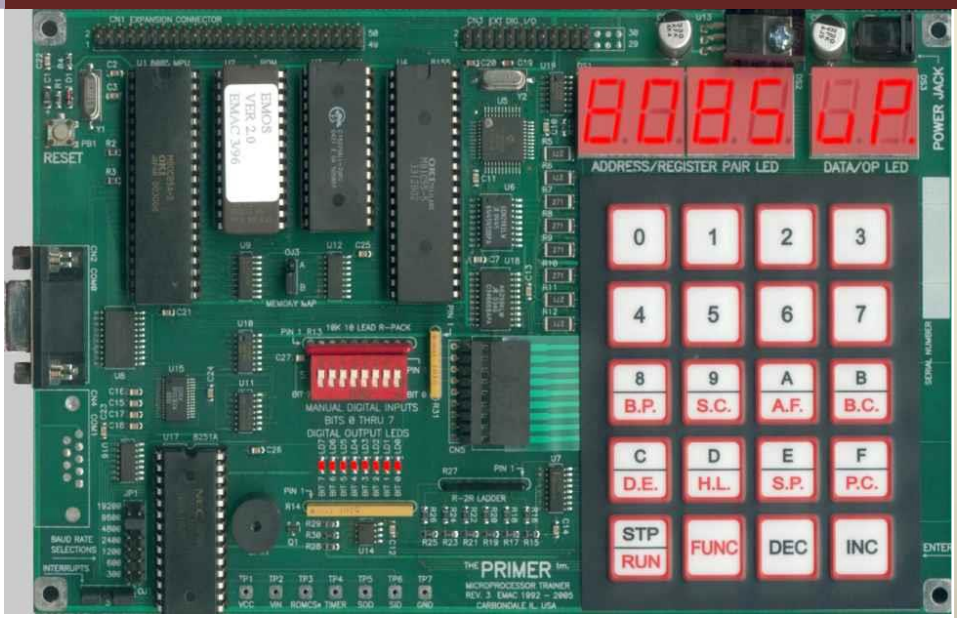
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ

&

ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Εργαστήριο Μικροϋπολογιστών

4η Σειρά Ασκήσεων



7ο Εξάμηνο

ΡΟΗ Υ

Αθανασίου Νικόλαος

ΑΜ 03112074

Βαβουλιώτης Γεώργιος

ΑΜ 03112083

Γιαννούλας Βασίλειος

ΑΜ 03112117

Θέμα 1ο

Το παρακάτω πρόγραμμα υλοποιεί μια αριθμομηχανή με δυνατότητες πρόσθεσης και αφαίρεσης δεκαεξαδικών αριθμών το πολύ 3 ψηφίων. Αρχικά αναμένεται η εισαγωγή του 1ου δεκαεξαδικού αριθμού ο οποίος ακολουθείται από το σύμβολο '+' ή '-' για λειτουργία πρόσθεσης και αφαίρεσης αντίστοιχα. Σημειώνεται ότι το πρόγραμμα λειτουργεί ακόμα και αν ο αριθμός έχει λιγότερα από 3 ψηφία καθώς θα αναγνωριστεί το σύμβολο '+' ή '-' ενώ αν δοθεί 3ψήφιος αριθμός, αγνοεί οποιοδήποτε σύμβολο δοθεί στην είσοδο εκτός από τα '+', '-'. Στη συνέχεια δίνεται ο 2ος δεκαεξαδικός αριθμός και ακολουθείται η ίδια διαδικασία, αναμένοντας αυτή τη φορά για το σύμβολο '='. Αφού δοθεί και το τελευταίο σύμβολο, οι δύο δεκαεξαδικοί αριθμοί και το σύμβολο της πράξης έχουν αποθηκευτεί σε 3 θέσεις μνήμης. Ανάλογα με το σύμβολο της πράξης εκτελείται η αντίστοιχη πράξη. Τέλος, το πρόγραμμα εκτυπώνει το αποτέλεσμα σε δεκαεξαδική και σε δεκαδική μορφή. Η διαδικασία είναι συνεχόμενη και τερματίζεται μόλις πατηθεί από το χρήστη το πλήκτρο 'Q'.
Για την υλοποίηση του προγράμματος, έγινε χρήση μίας βιβλιοθήκης με βοηθητικές συναρτήσεις για ανάγνωση και εκτύπωση χαρακτήρων μέσω του DOS και συναρτήσεις απεικόνισης ενός αριθμού σε δεκαεξαδική και δεκαδική μορφή.
Παραθέτουμε παρακάτω την βιβλιοθήκη 'lib.inc και την macros.txt που περιεχει τις μακροεντολές που θα χρησιμοποιήσουμε στο πρόγραμμα .

ΒΙΒΛΙΟΘΗΚΗ lib.inc

```
DEFINE_CREATE_HEX MACRO

JMP skip_proc_create_hex

;DIAVAZW TON ARITHMO POU DINETAI MESW TOY AX KAI TON PROSTHETW KATALLHLA STON TELIKO BX
CREATE_HEX PROC NEAR
        MOV AH,0
        XCHG AX,BX      ;ANTALLAGH PERIEXOMENWN TWN AX,BX
        MOV DX,16D      ;KANW 1 SHIFT LEFT SE HEX ARITHMO
        MUL DX
        ADD BX,AX       ;PROSTHETW AUTON P DIAVASA KAI EXW ETOIMO TON HEX STON BX
        RET
CREATE_HEX ENDP

skip_proc_create_hex:
DEFINE_CREATE_HEX ENDM


DEFINE_PRINT_DEC MACRO
LOCAL ADR2,ADR3

JMP skip_proc_print_dec

PRINT_DEC PROC NEAR
        MOV CX,0        ;COUNTER OF DIGITS
        MOV AX,DX       ;AX=ARITHMOS PROS EMFANISI
ADR2:
        MOV DX,0
        MOV BX,10
```

```
        DIV BX      ;DIAIRW ME TO 10 KAI A=ADIV10 KAI D=AMOD10
        PUSH DX     ;APOTHIKEUW TO MOD STO STACK
        INC CX      ;DIGITS++
        CMP AX,0    ;AN TELEIWSE O ARITHMOS TOTE FUGE
        JNZ ADR2    ;ALLIWS SUNEXISE NA SPAS STA PSHFIA
ADR3:
        POP DX      ;KANW POP OSES FORES LEEI O CX
        ADD DX,30H  ;KANTON ASCII
        PRINT DL    ;KAI TYPWSE TON
        LOOP ADR3
        RET
PRINT_DEC ENDP

skip_proc_print_dec:
DEFINE_PRINT_DEC ENDM



DEFINE_READ_HEX MACRO
LOCAL IGNORE,ITSNUM,ITSLETTER,ADRR1,ADRR2

JMP skip_proc_read_hex

;ROUTINA POU DIABASEI ENA HEX PSHFIO KAI TO EPISTREFEI WS DYADIKO MESW TOY AL

READ_HEX PROC NEAR
        PUSH DX
IGNORE:
        CMP AL,30H  ;AN EINAI <0 TOTE IGNORE
        JL ADRR1
        CMP AL,46H  ;AN EINAI >F TOTE IGNORE
        JG ADRR1
        CMP AL,41H  ;AN EINAI APO A EWS F TOTE EINAI LETTER
        JGE ITSLETTER
        CMP AL,39H  ;AN EINAI APO 0 EWS 9 TOTE EINAI NUM
        JLE ITSNUM
        JMP ADRR1   ;ALLIWS IGNORE
ITSNUM:
        PUSH AX
        PRINT AL    ;PRINT TON NUM
        POP AX
        SUB AL,30H  ;DIORTHWSH TOU NUM SE DUADIKI MORFH
        CALL CREATE_HEX ;KANTO PSHFIO HEX
        JMP ADRR2
ITSLETTER:
        PUSH AX
        PRINT AL    ;PRINT TO LETTER
        POP AX
        SUB AL,37H  ;DIORTHWSH TOU LETTER SE DUADUKI MORFH
        CALL CREATE_HEX ;KANTO PSHFIO HEX
        JMP ADRR2
```

```
ADRR1:
        INC CX        ;!!KANTO +1 XEIROKINHTA GT DEN PIANETAI GIA PSHFIO POU DIAVASA
ADRR2:
        POP DX
        RET
READ_HEX ENDP


skip_proc_read_hex:
DEFINE_READ_HEX ENDM




DEFINE_PRINT_HEX1 MACRO
LOCAL ADRR4,ADRR5

JMP skip_proc_print_hex1

PRINT_HEX1 PROC NEAR  ; H PRINT_HEX1 EKTYPWNEI TA 4LSB TOU DL WS HEX PSHFIO
        PUSH DX
        CMP DL,9          ;AN >9 TOTE EINAI LETTER
        JG ADDR4
        ADD DL,30H        ;ALLIWS EINAI DIGIT
        JMP ADDR5
ADDR4:
        ADD DL,37H
ADDR5:
        PRINT DL
        POP DX
        RET
PRINT_HEX1 ENDP

skip_proc_print_hex1:
DEFINE_PRINT_HEX1 ENDM



DEFINE_PRINT_HEX MACRO

JMP skip_proc_print_hex

PRINT_HEX PROC NEAR ; H ROYTINA PRINT_HEX EKTYPWNEI WS HEX TO ORISMA POU THS
DINETAI MESW TOY DL
        PUSH AX
        PUSH CX
        MOV AL,DL      ;A EXEI TON ARITHMO
        AND DL,0F0H    ;KRATAW TA 4 MSB
        MOV CL,4
        RCR DL,CL      ;4 SHIFTS DEKSIA
        CALL PRINT_HEX1 ;TYPWSE TA
```

```
        MOV DL,AL
        AND DL,0FH     ;TWRA KRATAW TA 4 LSB
        CALL PRINT_HEX1 ;KAI TA TYPWNW
        POP CX
        POP AX
        RET
PRINT_HEX ENDP

skip_proc_print_hex:
DEFINE_PRINT_HEX ENDM


DEFINE_PRINT32 MACRO


JMP skip_proc_print32

PRINT32 PROC NEAR
  PUSH DX
  XCHG DL,DH
  CALL PRINT_HEX
  XCHG DL,DH
  CALL PRINT_HEX
  POP DX
  RET
PRINT32 ENDP

skip_proc_print32:
DEFINE_PRINT32 ENDM
```

Macro.txt

```
EXIT    MACRO                   ;macro exits back to DOS
        MOV AX,4C00H
        INT 21H
ENDM

PRINT   MACRO CHAR              ;macro to print a char
        PUSH DX
        PUSH AX
        MOV DL,CHAR             ;o CHAR einai ston DX
        MOV AH,2
        INT 21H
        POP AX
        POP DX
ENDM

READ    MACRO                   ;macro to read a char from keyboard
        MOV AH,8
        INT 21H                 ;o,ti diavasa paei sto AL
ENDM
```

```
PRINT_STRING MACRO STRING          ;macro that prints a whole string
        PUSH DX
        PUSH AX
        LEA DX,STRING              ;to STRING einai ston DX
        MOV AH,09H
        INT 21H
        POP AX
        POP DX
ENDM

ERROR MACRO:
        PRINT_STRING NEWLINE
        PRINT_STRING MSG2
ENDM
```

Assembly code 8086

```
INCLUDE 'macros.txt'
INCLUDE 'lib.inc'

DATA_SEG SEGMENT
        X1 DW ?            ;STO X1 THA APOTHIKEYW TON 1st OP
        X2 DW ?            ;STO X2 THA APOTHUKEYW TON 2nd OP
        X3 DB ?           ;STO X3 THA APOTHIKEYW TO PROSHMO
        NEWLINE DB 0AH,0DH,'$'
DATA_SEG ENDS

STACK_SEG SEGMENT STACK
        DB 50 DUP(?)               ;50 bytes for stack
STACK_SEG ENDS


CODE_SEG SEGMENT BYTE
        ASSUME CS:CODE_SEG,DS:DATA_SEG,SS:STACK_SEG

DEFINE_CREATE_HEX
DEFINE_PRINT_DEC
DEFINE_READ_HEX
DEFINE_PRINT_HEX1
DEFINE_PRINT_HEX
DEFINE_PRINT32


MAIN PROC FAR
        MOV AX,DATA_SEG
        MOV DS,AX                          ;DS <- DATA_SEG
        MOV AX,STACK_SEG
        MOV SS,AX                          ;SS <- STACK_SEG
        JMP START

TERMINATE:
```

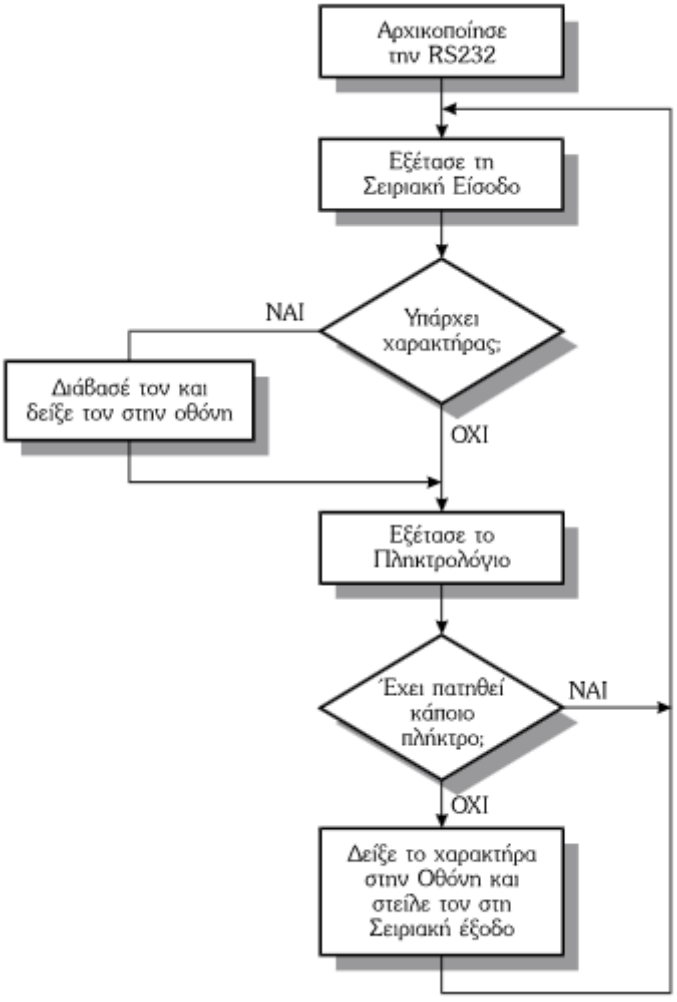
```
        EXIT

PUSH_SYMBOL:
        PRINT AL            ;PRINT TO SYMBOL
        MOV X3,AL             ;X3=SYMBOL
        JMP SECOND            ;PHGAINE NA DIAVASEIS TON 2nd OP

START:
        MOV CX,3            ;KATHE ARITHMOS EXEI EWS 3 DIGITS
        MOV BX,0
L1:
        READ
        CMP AL,'Q'          ;AN PATHSW Q TOTE TERMINATE
        JE TERMINATE
        CMP AL,2BH              ;CODE '+'
        JE PUSH_SYMBOL
        CMP AL,2DH              ;CODE '-'
        JE PUSH_SYMBOL
        CALL READ_HEX       ;DIAVASE ENA HEX DIGIT
        LOOP L1             ;LOOPARE EWS 3 FORES

IGN:                ;AN PATHSW PANW APO 3 DIGITS TOTE AGNOHSE
        READ
        CMP AL,'Q'
        JE TERMINATE
        CMP AL,2BH              ; '+'
        JE PUSH_SYMBOL
        CMP AL,2DH              ; '-'
        JE PUSH_SYMBOL
        JMP IGN

SECOND:             ;DIAVASMA 2nd OP
        MOV X1,BX           ;X1 = 1st OP (HEX)
        MOV CX,3            ;OMOIWS GIA TON 2nd OP
        MOV BX,0
L2:
        READ
        CMP AL,'Q'
        JE TERMINATE
        CMP AL,'='
        JE FINISH
        CALL READ_HEX
        LOOP L2

IGN2:
        READ
        CMP AL,'Q'
        JE TERMINATE
        CMP AL,'='
        JE FINISH
        JMP IGN2
```

```
FINISH:
        MOV X2,BX               ;X2 = 2nd OP(HEX)
        CMP X3,'+'
        JE ADDNUM
        CMP X3,'-'
        JE SUBNUM
        EXIT

ADDNUM:
  PRINT '='
  MOV DX,X1
  ADD DX,X2             ;DX = X1+X2
  CALL PRINT32          ;PRINT TO SUM(HEX)
  PRINT '='
  CALL PRINT_DEC        ;PRINT TO SUM(DEC)
  PRINT_STRING NEWLINE
  JMP START             ;SUNEXHS LEITOURGEIA

SUBNUM:
  PRINT '='
  MOV DX,X1
  CMP DX,X2
  JL NEGATIVE           ;AN X1<X2 TOTE VGALE ENA '-' KAI KANE X2-X1
  SUB DX,X2             ;DX=X1-X2
  CALL PRINT32          ;PRINT SE HEX
  PRINT '='
  CALL PRINT_DEC        ;PRINT SE DEC
  PRINT_STRING NEWLINE
  JMP START

NEGATIVE:
        PRINT '-'
        MOV DX,X2
        SUB DX,X1
        CALL PRINT32
        PRINT '='
        PRINT '-'
        CALL PRINT_DEC
        PRINT_STRING NEWLINE
        JMP START
MAIN ENDP

CODE_SEG ENDS
        END MAIN
```

Θέμα 2ο

Στον παρόν θέμα ο σκοπός είναι να δημιουργήσουμε μια προσομοίωση τερματικού.Το πρόγραμμα τρέχει στο MASM με τις κατάλληλες εντολές και αρχικά εμφανίζει ένα παράθυρο στο οποίο διαχωρίζει το χώρο REMOTE και LOCAL με παύλες.Τρέχουμε ξανά το ίδιο object file μέσω MASM για να δημιουργήσουμε δεύτερο παράθυρο διαλόγου.Έπειτα εμφανίζονται και στα δύο στον χρήστη επιλογές για ECHO mode την οποία αν ενεργοποιήσει(1) αυτά που γράφει εμφανίζονται και στο δικό του LOCAL αλλίως μόνο στου άλλου το REMOTE και η επιλογή για BAUD rate το οποίο καθορίζει την ταχύτητα επικοινωνίας μεταξύ των δύο παραθύρων. Στις μεταβλητές linrem colrem linloc colloc κρατάμε τις θέσεις του κέρσορα στο REMOTE και στο LOCAL αντίστοιχα.

Ξεκινώντας αναφορικά με την υλοποίηση, χρησιμοποιούμε το πρότυπο σειριακής επικοινωνίας που χρησιμοποιεί τη θύρα RS232. Ο αλγόριθμος που ακολουθείται για την υλοποίηση του terminal φαίνεται στο ακόλουθο διάγραμμα ροής:



1. Για το διαχωρισμό της οθόνης, όπως και για το πιο σημείο της οθόνη πρέπει να γράψουμε κάθε φορά χρησιμοποιείται η κλήση του BIOS int 10h/02, η οποία μετακινεί τον κέρσορα στο επιθυμητό σημείο.
2. Η αρχικοποίηση της θύρας RS232 γίνεται με χρήση της συνάρτησης open_rs232, η λήψη χαρακτήρα από τη σειριακή θύρα μέσω της

rxch_RS232, ενώ η αποστολή χαρακτήρα μέσω της txch_RS232. Οι παραπάνω συναρτήσεις δίνονται στον εργαστηριακό οδηγό.

3. Εάν ληφθεί ή πληκτρολογηθεί (και επιλεχθεί echo on) το backspace τότε μεταφέρουμε τον κέρσορα μια θέση πίσω και στη συνέχεια εμφανίζουμε το χαρακτήρα space ώστε να «σβηστεί» ο αμέσως προηγούμενος χαρακτήρας.
4. Εάν ληφθεί ή πληκτρολογηθεί (και επιλεχθεί echo on) το enter τότε μεταφέρουμε τον κέρσορα στην αρχή της επόμενης γραμμής.
5. Εάν γεμίσει είτε ο χώρος που εμφανίζονται αυτά που πληκτρολογούνται είτε ο χώρος που εμφανίζονται τα ληφθέντα δεδομένα, τότε το αντίστοιχο κομμάτι μεταφέρεται κατά μία γραμμή πάνω μέσω της ρουτίνας του bios int 10h/06.
6. Το πρόγραμμα τερματίζει με ESC.

Στη συνέχεια ακολουθεί ο κώδικας της άσκησης, ο οποίος εκτελέστηκε σε δύο διαφορετικά instances του dosbox με παραμέτρους nullmodem και nullmodem server:localhost στον server1 των παραμέτρων του dosbox αντιστοιχως, για την προσομοιώση του terminal.

```
include 'emu8086.inc'

org 100h

.data

new_line DB 0Dh,0Ah,"$"

echo_mode DB "CHOOSE ECHO MODE: <1> FOR ECHO ON OR <0> FOR ECHO OFF$"


br DB 0Ah,0Dh,"CHOOSE BAUD RATE: <1> FOR 300 <2> FOR 600 <3> FOR 1200 <4> FOR 2400 <5> FOR 4800 OR <6> FOR 9600 $"


loc DB "LOCAL $"

rem DB "REMOTE $"

divline DB 08,80 DUP (0C4h),"$"

em DB ? ;echo mode

tmp DB ?

linloc DB 00h

colloc DB 0Bh

linrem DB 0Dh

colrem DB 0Bh
```

```
.code

main PROC FAR

PRINT_STR echo_mode
PRINT_STR new_line
ECHO_MODE1:
   MOV AH,08h
   int 21h
   CMP AL,30h
   JE Valid1
   CMP AL,31h
   JNE ECHO_MODE1

Valid1:
   mov em,AL
   PRINT_STR br
   PRINT_STR new_line

BAUD_RATE1:
   MOV AH,08h
   int 21h
   CMP AL,31h
   Jl BAUD_RATE1
   CMP AL,36h
   JLE  Valid2
   JMP BAUD_RATE1
```

```
Valid2:
   sub AL,30h
   and AL,03h
   call OPEN_RS232
   mov AH,00h
   mov AL,2
   int 10h
   mov AH,02h
   mov DH,00h
   PRINT_STR loc
   mov AH,02h
   mov DH,0Ch
   mov DL,01h
   mov BH,00h
   int 10h
   PRINT_STR divline

   mov AH,02h
   mov DH,0Dh
   mov DL,01h
   mov BH,00h
   int 10h
   PRINT_STR rem

RX:
   call RXCH_RS232
```

```
    cmp AL,0h
    je CHK
    mov AH,02h


    mov DH,linrem
    mov DL,colrem
    mov BH,00h
    int 10h
    cmp AL,0Dh
    je REM_ENTER
    cmp AL,08h

    jne NEXT4
    CALL BACKSP

NEXT4:
    mov ah,0Eh
    int 10h
    mov AH,03H
    mov BH,00H
    int 10h
    cmp DL,4Fh

    jne UPDRC

REM_ENTER:
```

```
    inc     DH
    cmp   DH,19h
    jne     NOSCROLL1
    mov   AH,06h

    mov   AL,01h
    mov   CH,0Dh
    mov   CL,0Bh
    mov   DH,18h
    mov   DL,4Fh
    mov   BH,07h
    int     10h
    mov   linrem,18h

    mov   colrem,0Bh

    jmp    CHK

UPDRC:

    cmp DL,0Bh


    jge NEXT5
    mov DL,0Bh

NEXT5:
```

```
    mov linrem,DH

    mov colrem,DL



    jmp CHK

NOSCROLL1:
    mov linrem,DH
    mov colrem,0Bh



CHK:

    mov AH,06h
    mov DL,0FFh



    int 21h
    jz RX

    cmp AL,1Bh

    je QUIT
    mov BL,em
    cmp BL,30h
    je TX
```

```
    mov tmp,AL

    mov AH,02h
    mov DH,linloc
    mov DL,colloc
    mov BH,00h
    int 10h
    cmp AL,0Dh

    je LOC_ENTER
    cmp AL,08h
    jne NEXT3
    CALL BACKSP

NEXT3:
    mov ah,0Eh
    int 10h
    mov AH,03H
    mov BH,00H
    int 10h
    cmp DL,4Fh
    ,

    jne UPDLC
LOC_ENTER:
    inc DH
```

```
    cmp DH,0Ch

    jne NOSCROLL2
    mov AH,06h

    mov AL,01h    ;SCROLL
    mov CH,00h
    mov CL,0Bh
    mov DH,0Bh
    mov DL,4Fh
    mov BH,07h
    int 10h
    mov linloc,0Bh
    mov colloc,0Bh

    jmp NEXT



UPDLC:
    cmp DL,0Bh

    jge NEXT2
    mov DL,0Bh
NEXT2: mov linloc,DH

    mov colloc,DL
```

```
   jmp NEXT

NOSCROLL2:
   mov linloc,DH
   mov colloc,0Bh


NEXT:
   mov AL,tmp


TX:
   call TXCH_RS232

   jmp RX


QUIT:
   MOV AX,4C00h ; direct exit to DOS
   INT 21H

main ENDP
```

```
OPEN_RS232 PROC NEAR
   JMP START
   BAUD_RATE LABEL WORD
   DW 1047
   DW 768
   DW 384
   DW 192
   DW 96
   DW 48
   DW 24
   DW 12
START:
   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

RXCH_RS232 PROC NEAR
   MOV DX,3FDh
   IN AL,DX
   TEST AL,1
   JZ NOTHING
   SUB DX,5
   IN AL,DX
   JMP EX2

NOTHING:
   MOV AL,0
EX2:
   RET
```

```
RXCH_RS232 ENDP



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

BACKSP PROC NEAR
   mov AH,02h

   dec DL
   int 10h
   CALL pthis
   DB ' ', 0
   ret

BACKSP ENDP
```

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



DEFINE_PTHIS
END
```