Εργαστήριο Μικροϋπολογιστών

4η Εργαστηριακή άσκηση

Ομάδα Γ04

Συνεργάτες:
- **Σκούλος Ραφαήλ Α.Μ: 03112404**
- **Αναστάσης Σταθόπουλος Α.Μ: 03112140**
- **Τζίνης Ευθύμιος Α.Μ: 03112007**

Άσκηση 1: **Υλοποίηση Αριθμομηχανής**

Σε αυτή την άσκηση καλούμαστε να υλοποιήσουμε μια αριθμομηχανή με δυνατότητες προσθεσης και αφαίρεσης δεκαεξαδικών αριθμών το πολύ 3 ψηφίων. Στη συνέχεια θα εμφανίζει πρώτα το δεκαεξαδικό αποτέλεσμα και έπειτα το αντίστοιχο δεκαδικό.

Παρακάτω φαίνεται η υλοποίηση του προγράμματος:

```
include 'lib4i.inc'

data segment
   num1 DW 0
   num2 DW 0
   NEWLINE DB 0AH,0DH,'$'
data ends

code_seg segment
   assume cs:code, ds:data, es:data

main proc far
start:
   mov ax,data
   mov ds,ax
   mov es,ax

ReadNum1:
   mov num1,0
   mov num2,0
   mov bl,0
   mov cx,3
   mov ah,0            ;read 3-digit number

ignore:
   READ
```

```
    cmp al,'Q'          ;if Q terminate
    je finish
    cmp al,30h          ;if less than 0 ignore it
    jl ignore
    cmp al,39h
    jg checkCaps        ;if greater than 9 check if it is between A-F
    PRINT al
    sub al,30h          ;subtract 30h to get the correct number
save:
    mov ah,0
    sal num1,4
    add num1,ax         ;multiply *16
    loop ignore
   jmp getOp
save3:
    mov ah,0
    add AL,0AH
    sal num1,4
    add num1,ax         ;multiply *16
    loop ignore
    jmp getOp

checkCaps:
    cmp al,'A'          ;if less than A ignore it
    jl ignore
    cmp al,'F'          ;if greater than F check if it is between a-f
```

```
    jg checkSmall
    PRINT al
    sub al,'A'          ;subtract 31h to get the correct number
    jmp save3

checkSmall:
    cmp al,'a'
    jl ignore           ;if less than a ignore it
    cmp al,'f'
    jg ignore           ;if greater than f ignore it
    PRINT al
    sub al,'a'          ;subtract 51h to get the right number
    jmp save3

getOp:
    READ
    cmp al,'Q'          ;if Q then terminate
    je finish
    cmp al,'+'
    je addition
    cmp al,'-'
    je subtraction
    jmp getOp

addition:
    PRINT al
```

```
    mov bl,al           ;in bl we keep the operation
    jmp ReadNum2

subtraction:
    PRINT al
    mov bl,al

ReadNum2:
    mov cx,3            ;read the second 3-digit number
ignore2:
    READ
    cmp al,'Q'          ;if Q terminate
    je finish
    cmp al,30h          ;if less than 0 ignore it
    jl ignore2
    cmp al,39h
    jg checkCaps2       ;if greater than 9 check if it is between A-F
    PRINT al
    sub al,30h          ;subtract 30h to get the correct number
save2:
    MOV AH,0
    sal num2,4
    add num2,ax         ;multiply *16
    loop ignore
    jmp wait1
```

```
save4:
   mov ah,0
   add AL,0AH
   sal num2,4
   add num2,ax          ;multiply *16
   loop ignore2
   jmp wait1

checkCaps2:
   cmp al,'A'           ;if less than A ignore it
   jl ignore2
   cmp al,'F'           ;if greater than F check if it is between a-f
   jg checkSmall2
   PRINT al
   sub al,'A'           ;subtract 31h to get the correct number
   jmp save4

checkSmall2:
   cmp al,'a'
   jl ignore2           ;if less than a ignore it
   cmp al,'f'
   jg ignore2           ;if greater than f ignore it
   PRINT al
   sub al,'a'           ;subtract 51h to get the right number
   jmp save4
```

```
wait1:
    READ
    cmp al,'Q'
    je finish
    cmp al,'='          ;wait until i read '='
    jne wait1
    PRINT al            ;print '='

    cmp bl,'-'
    je subtr
    mov dx,num2         ;if operand is '+' then add them

    add num1,dx
    PRINT_HEX num1
    PRINT '='
    PRINT_DEC num1
    PRINT_STR NEWLINE
    jmp start

subtr:
    mov dx,num2
    cmp num1,dx         ;if second operand is greater then we have negative result
    jl negative
    sub num1,dx
    PRINT_HEX num1
    PRINT '='
```

```
    PRINT_DEC num1
    PRINT_STR NEWLINE
    jmp start

negative:
    mov dx,num1
    sub num2,dx
    PRINT '-'
    PRINT_HEX num2
    PRINT '='
    PRINT '-'
    PRINT_DEC num2
    PRINT_STR NEWLINE
    jmp start

finish:
code_seg ends
END MAIN
```

Η βιβλιοθήκη **‘lib4i.inc’** είναι:

```
READ MACRO
  MOV AH,08H
  INT 21H
ENDM

PRINT MACRO CHAR                ;PRINT A CHAR USING ITS ASCII VALUE
  PUSH AX
  PUSH DX
  MOV DL,CHAR
  MOV AH,02H
  INT 21H
  POP DX
  POP AX
ENDM

PRINT_STR MACRO STRING                                  ;PRINT STRING - WE USE IT TO
PRINT NEWLINE
  PUSH AX
  PUSH DX
  MOV DX, OFFSET STRING
  MOV AH, 09H
  INT 21H
  POP DX
  POP AX
ENDM
```

```
PRINT_HEX MACRO HEX

        LOCAL
THOUS,THOUS_OK,NEXT1,NEXT2,NEXT3,NUMBERS,NUMBERS1,NUMBERS2,NUMBERS3,
FINAL,DEC_OK,HUN_OK,CDEC,CHUNDR


        PUSH AX                 ;KEEP NECESSARY REGS AND VARS IN STACK
        PUSH DX
        PUSH HEX


  MOV AH,02H
        MOV DX,0
        CMP HEX,1000H               ;I OMIT THE FIRST '
        JL CHUNDR
THOUS:                              ;COUNT THE NUMBER OF 1000Hs
        CMP HEX,1000H
        JL THOUS_OK
        INC DX
        SUB HEX,1000H
        JMP THOUS
THOUS_OK:                                   ;DONE WITH COUNTING
        CMP DL,0AH
        JL NUMBERS
        ADD DL,37H                      ;PRINT A-F HERE
        INT 21H
        MOV DX,0
        JMP NEXT1
NUMBERS:                                ;PRINT THE NUMBERS HERE
        ADD DL,30H
        INT 21H
        MOV DX,0
        JMP NEXT1
```

```
CHUNDR:                                          ;I OMIT THE SECOND '0'
        CMP HEX,100H
        JL CDEC
NEXT1:                              ;COUNT THE NUMBER OF 100Hs
        CMP HEX,100H
   JL HUN_OK
        INC DX
        SUB HEX,100H
        JMP NEXT1
HUN_OK:                                  ;DONE WITH COUNTING
        CMP DL,0AH
        JL NUMBERS1
        ADD DL,37H
        INT 21H              ;PRINT DIGITS AND LETTERS SEPERATELY
        MOV DX,0                    ;LIKE ABOVE
        JMP NEXT2
NUMBERS1:
        ADD DL,30H
        INT 21H
        MOV DX,0
        JMP NEXT2
CDEC:                                    ;I OMMIT THE THID '0'
        CMP HEX,10H
        JL NEXT3
NEXT2:                              ;COUNT THE NUMBER OF 10Hs
        CMP HEX,10H
        JL DEC_OK
        INC DX
        SUB HEX,10H
```

```
        JMP NEXT2
DEC_OK:
        CMP DL,0AH
        JL NUMBERS2
        ADD DL,37H
        INT 21H
        MOV DX,0
        JMP NEXT3
NUMBERS2:
        ADD DL,30H
        INT 21H
        MOV DX,0

NEXT3:                                          ;PRINT THE 10Hs
        CMP HEX,09H
        JL NUMBERS3
        ADD HEX,37H
        JMP FINAL
NUMBERS3:
        ADD HEX,30H
FINAL:
        MOV DX,HEX
        INT 21H

        POP HEX
        POP DX
        POP AX
ENDM
```

```
PRINT_DEC MACRO NUM
        LOCAL
THOUS,THOUS_OK,HUNDR,HUNDR_OK,DEC,DEC_OK,CHUNDR,CDEC,FINAL

        PUSH AX                 ;KEEP NECESSARY REGS AND VARS IN STACK
        PUSH DX
        PUSH NUM

        MOV AH,02H
        MOV DX,00H

        CMP NUM,03E8H           ;COMPARE WITH 1000 DEC
        JL CHUNDR               ;IF LESS THAN 1000, OMIT THE FIRST '0'
THOUS:                          ;COUNT THE NUMBER OF THOUSANDS
        CMP NUM,03E8H
        JL THOUS_OK
        INC DL
        SUB NUM,03E8H
        JMP THOUS
THOUS_OK:                       ;DONE WITH COUNTING THE THOUSANDS
        ADD DL,30H              ;NOW PRINT
        INT 21H
        MOV DL,00H
        JMP HUNDR
CHUNDR:
        CMP NUM,0064H           ;OMIT THE SECOND '0'
        JL CDEC
HUNDR:                          ;COUNT HUNDREDS
        CMP NUM,0064H
```

```
        JL HUNDR_OK
        INC DL
        SUB NUM,0064H
        JMP HUNDR
HUNDR_OK:
        ADD DL,30H
        INT 21H
        MOV DL,00H
        JMP DEC
CDEC:
        CMP NUM,0AH
        JL FINAL
DEC:                                    ;COUNT DECADES
        CMP NUM,000AH
        JL DEC_OK
        INC DL
        SUB NUM,000AH
        JMP DEC
DEC_OK:
        ADD DL,30H
        INT 21H
FINAL:
        MOV DX,NUM
        ADD DL,30H
        INT 21H

        POP NUM
        POP DX
        POP AX
ENDM
```

Άσκηση 2 **Μετετροπή PC σε Τερματικό**

Σκοπός της άσκησης αυτής είναι η δημιουργία ενός περιβάλλοντος επικοινωνίας για 2 PC όπου στη μισή οθόνη θα μπορούμε να πληκτρολογούμε και να στέλνουμε μηνύματα, ενώ στην άλλη μισή να λαμβάνουμε μηνύματα. Για να τρέξουμε τα δυο τερματικά στο ίδιο PC χρησιμοποιήσαμε DosBox.

Στη συνέχεια φαίνεται ο κώδικας της υλοποίησής μας:

```
READ MACRO
   MOV AH,8
   INT 21H
ENDM
PRINT_STR MACRO STRING
   MOV DX,OFFSET STRING
   MOV AH,9
   INT 21H
ENDM
PRINT MACRO CHAR
   MOV DL,CHAR
   MOV AH,2H
   INT 21H
ENDM

CLEAR MACRO
   MOV CX,0H     ;CX SHOWS THE UPPER LEFT SIDE
   MOV DH,24     ;LINE DOWN RIGHT SIDE
   MOV DL,80      ;COLUMN RIGHT SIDE
   MOV BH,4FH    ;COLORS (BLACK AND WHITE IS JUST 43)
```

```
   MOV AX,700H   ;AH=07 FOR THE INTERRUPT AND AL=00 ALL THE LINES
   INT 10H
ENDM

GO_THERE_CX MACRO COL          ;CX SHOW THE LINE WE WANT TO MOVE
   PUSH AX                     ;SAVE REGS
   PUSH BX
   PUSH DX
   MOV AH,02H                   ;USE INTERRUPT 10/02
   MOV DH,CL                   ;DH<---CL HAS THE LINE
   MOV DL,COL                   ;DL<---THE SELECTED COLUMN
   MOV BH,0H                   ;WE WANT ONLY THE FIRST PAGE
   INT 10H
   POP DX
   POP BX                      ;PUSH BACK
   POP AX
ENDM
PRINT_THERE MACRO CHAR
   PUSH AX
   MOV AL,CHAR                 ;WE USE INTERRUPT 10/0E
   MOV AH,0EH                ;MOVES THE CURSOR ONE POSITION AS WELL
   INT 10H
   POP AX
ENDM
GO_THERE MACRO LINE COL  ;WE JUST GO TO THE SELECTED LINE AND COL
   PUSH AX
   PUSH BX
   PUSH DX
```

```
   MOV AH,02H                    ;INT 10/02
   MOV BP,OFFSET LINE
   MOV DH,DS:[BP]                ;DH HAS THE LINE NUMBER
   MOV BP,OFFSET COL
   MOV DL,DS:[BP]                ;DL HAS THE COL NUMBER
   MOV BH,0H                     ;CURRENT PAGE
   INT 10H
   POP DX
   POP BX
   POP AX
ENDM

INCREASE_LINE MACRO LINE COLUMN NEXT_POS
   MOV BP,OFFSET COLUMN              ;COL_CNT++
   MOV DS:[BP],NEXT_POS    ;0 OR 41 DEPENDS ON THE LEFT OR RIGHT SIDE OF
TERMINAL
   MOV BP,OFFSET LINE      ;LINE_CNT++
   INC DS:[BP]
ENDM

SCROLL_UP MACRO LINES ULL ULC LRL LRC
   PUSH AX
   PUSH CX        ;WE USE INT 10/06
   PUSH DX
   MOV AL,LINES    ;AL=LINES
   MOV CH,ULL     ;CH= LINE OF UP LEFT BORDER
   MOV CL,ULC     ;CL=COLUMN OF UP LEFT
   MOV DH,LRL     ;DH,DL RESPECTIVELY
   MOV DL,LRC
```

```
   MOV AH,06H
   INT 10H
   POP DX
   POP CX
   POP AX
ENDM

NON_BLOCKING_READ MACRO
   MOV AH,06H          ;WE USE INT 21/06
   MOV DL,0FFH         ;IF WE READ A CHAR THEN AL HAS THE ASCII CODE
   INT 21H              ;ZF BECOMES 0 WHEN READ
ENDM



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

MAIN PROC FAR
   ECHO     DW 0
   ECHO_LINE DW 0    ;ECHO_LINE AND ECHO_COL ARE THE CNTS FOR THE
RIGHT WINDOW OF THE TERMINAL(PORT INPUT)
   ECHO_COL  DW 0
   LINE_CTR  DW 0    ;THESE CNTS ARE FOR THE LEFT WINDOW OF THE
TERMINAL(KEYB INPUT)
   COL_CTR   DW 41   ;41 is the middle column
   ECHOMSG DB "KALWS TON, MHPWS THES ECHO PASSA MOU? [Y/N] ? $"
   BAUDMSG DB "GIMME-GIMME BAUD RATE. PRESS  <1> FOR 300, <2> FOR 600,
<3> FOR 1200, <4> FOR 2400, <5> FOR 4800, <6> FOR 9600 $"
   NEW_LINE DB 0AH, 0DH, '$'
```

```
START:
   CLEAR                    ;CLEAR THE SCREEN AND SET THE BACKGROUND
COLOR
   PRINT_STR ECHOMSG
   CALL READ_ECHO           ;WE UPDATE ECHO VARIABLE
   PRINT_STR NEW_LINE
   PRINT_STR BAUDMSG
   CALL READ_BAUDRATE       ;AL HAS THE VALUE OF SELECTED BAUD
RATE (1BIT/SEC=1BAUD)
   CALL OPEN_RS232
   CLEAR
   CALL SCREEN_SPLIT        ; PRINT THE LINE IN THE MIDDLE
   GO_THERE ECHO_LINE ECHO_COL

READ_FROM_PORT:
   CALL RXCH_RS232          ;READ ONE CHAR FROM THE PORT COM
   CMP AL,0                 ;IF THERE IS NOTHING THERE
   JE CHECK_KEYB            ;CHECK KEYBOARD
   MOV BL,AL                ;IF WE HAVE ACHAR IN BL THEN PRINT IT

   CMP BL,8
; === COMPARE WITH BACKSPACE ===
   JZ GOT_BACKSPACE_HANDLE
   CMP BL,0DH               ;IF ITS ENTER THEN INCREASE THE LINE CNT
   JE INC_LINE_RIGHT
   GO_THERE LINE_CTR COL_CTR    ; MOVE CURSOR TO THE APPROPRIATE
LINE & COL
   PRINT_THERE BL
   MOV BP,OFFSET COL_CTR        ; AND COL_CTR++
```

```
   MOV BL,DS:[BP]
   INC BL                    ; IF IT REACHES AT THE END OF THE LINE
   CMP BL,80                 ; IT SHOULD BE RESET TO 41
   JNE NOINC_LINE_RIGHT

;SAME AS BELOW
INC_LINE_RIGHT:
   INCREASE_LINE LINE_CTR COL_CTR 41      ; INCREASE LINE AND SAVE NEW LINE/COLUMN
   GO_THERE LINE_CTR COL_CTR          ; AND MOVE THE CURSOR TO THE NEXT POSITION
   MOV BP,OFFSET LINE_CTR
   MOV BL,DS:[BP]          ;BL<-LINE_CTR
   CMP BL,24               ;IF WE GET OVER THE LINES OF TERMINAL
   JNE CHECK_KEYB          ;WE SCROLL UP
   SUB BL,1
   MOV DS:[BP],BL          ;LINE_CTR--
   SCROLL_UP 1 0 41 24 79     ;SCROLL UP THE RIGHT WINDOW APPROPRIATE
   GO_THERE LINE_CTR COL_CTR
   JMP CHECK_KEYB

NOINC_LINE_RIGHT:
   MOV DS:[BP],BL             ; STORE THE COLUMN (NO LINE INCREASE NO SCROLL)


CHECK_KEYB:
   NON_BLOCKING_READ          ;NO BLOCK READ MACRO
   JZ READ_FROM_PORT       ;IF ZF==1 THEN WE DID NOT PRESSED ANY KEY
   MOV BL,AL
```

```
    CMP BL,27           ;WE CAN TERMINATE WITH ESC ANY TIME WE WANT TO
    JE QUIT


    CMP BL,8
; === COMPARE WITH BACKSPACE ===
    JZ BACKSPACE_HANDLE


    CALL TXCH_RS232
    MOV BP,OFFSET ECHO     ;ECHO IS 1 IF WE SELECTED YES
    CMP DS:[BP],1
    JNE READ_FROM_PORT        ;IF ECHO==0 THEN LOOP AGAIN
    CMP BL,0DH            ;IF WE HAVE PRESSED ENTER
    JE  INC_LINE_LEFT
    GO_THERE ECHO_LINE ECHO_COL ;ELSE PRINT THE CHAR THERE
    PRINT_THERE BL
    MOV BP,OFFSET ECHO_COL
    MOV BL,DS:[BP]          ;BL HAS THE COLUMN CTR
    INC BL
    CMP BL,39               ;39 IS THE END OF THE LINE
    JNE NOINC_LINE_LEFT



;SAME AS ABOVE
INC_LINE_LEFT:          ;SAME PROCEDURE AS BEFORE BUT FOR THE OTHER
PART OF THE TERMINAL
    INCREASE_LINE ECHO_LINE ECHO_COL 0
    GO_THERE ECHO_LINE ECHO_COL
    MOV BP,OFFSET ECHO_LINE
    MOV BL,DS:[BP]
```

```
   CMP BL,24
   JNE READ_FROM_PORT
   SUB BL,1
   MOV DS:[BP],BL
   SCROLL_UP 1 0 0 24 38
   GO_THERE ECHO_LINE ECHO_COL
   JMP READ_FROM_PORT

NOINC_LINE_LEFT:
   MOV DS:[BP],BL          ;BL HAS THE COL CTR
   JMP READ_FROM_PORT

;=========BACKSPACE
HANDLER==============================================
BACKSPACE_HANDLE:
   CALL TXCH_RS232        ;AL HAS THE BACKSPACE AND WE SEND IT TO THE
OTHER TERMINAL
   MOV BP,OFFSET ECHO     ;ECHO IS 1 IF WE SELECTED YES
   CMP DS:[BP],1
   JNE READ_FROM_PORT        ;IF ECHO==0 THEN LOOP AGAIN
   MOV BP,OFFSET ECHO_COL
   MOV BL,DS:[BP]          ;BL HAS THE COLUMN CTR
   CMP BL,0              ;START OF LINE
   JNZ DELETE_CHAR        ;IF WE ARE NOT IN THE BEGINNING DELETE A CHAR
   JMP PREV_LINE      ;ELSE GO TO THE PREVIOUS LINE

DELETE_CHAR:
   SUB BL,1              ;MOVE ONE BACK THE POINTER
   MOV DS:[BP],BL
```

```
   GO_THERE ECHO_LINE ECHO_COL ;ELSE PRINT THE CHAR THERE
   PRINT_THERE ' '         ;PRINT A SPACE IN THERE
   GO_THERE ECHO_LINE ECHO_COL
   JMP READ_FROM_PORT

PREV_LINE:
   MOV BP,OFFSET ECHO_LINE
   MOV BL,DS:[BP]
   CMP BL,0                ;IF WE ARE AT THE FIRST LINE DELETE ELSE GO BACK
   JZ  READ_FROM_PORT
   ;ELSE LINE-- AND COLUMN=38
   SUB BL,1
   MOV DS:[BP],BL  ;LINE--
   MOV BP,OFFSET ECHO_COL
   MOV BL,39               ;BL IS IN THE LAST COL POSITION
   MOV DS:[BP],BL          ;SAVE IT AND DELETE THE PREVIOUS CHAR
   JMP DELETE_CHAR         ;BP SHOW AT THE COL PTR
;=================================================================
====

;=======IF WE GET BACKSPACE FROM THE OTHER TERMINAL IS PRETTY
MUCH THE SAME=======
GOT_BACKSPACE_HANDLE:
   MOV BP,OFFSET COL_CTR
   MOV BL,DS:[BP]          ;BL HAS THE COLUMN CTR
   CMP BL,41               ;START OF LINE
   JNZ GOT_DELETE_CHAR        ;IF WE ARE NOT IN THE BEGINNING DELETE A
CHAR
   JMP GOT_PREV_LINE       ;ELSE GO TO THE PREVIOUS LINE
```

```
GOT_DELETE_CHAR:
   SUB BL,1              ;MOVE ONE BACK THE POINTER
   MOV DS:[BP],BL
   GO_THERE LINE_CTR COL_CTR ;ELSE PRINT THE CHAR THERE
   PRINT_THERE ' '       ;PRINT A SPACE IN THERE
   GO_THERE LINE_CTR COL_CTR
   JMP CHECK_KEYB        ;NOW WE CHECK KEYBOARD

GOT_PREV_LINE:
   MOV BP,OFFSET LINE_CTR
   MOV BL,DS:[BP]
   CMP BL,0              ;IF WE ARE AT THE FIRST LINE DELETE ELSE GO BACK
   JZ  CHECK_KEYB
   ;ELSE LINE-- AND COLUMN=79
   SUB BL,1
   MOV DS:[BP],BL  ;LINE--
   MOV BP,OFFSET COL_CTR
   MOV BL,80             ;BL IS IN THE LAST COL POSITION
   MOV DS:[BP],BL        ;SAVE IT AND DELETE THE PREVIOUS CHAR
   JMP GOT_DELETE_CHAR   ;BP SHOW AT THE COL PTR
;=================================================================
======

QUIT:
   CLEAR      ;EXIT TO BIOS
   MOV AL,0H
   MOV AH,4CH
   INT 21H
MAIN ENDP
```

```
;=/=/=/=READ THE ECHO ANS OF THE USR=/=/=/=/
READ_ECHO PROC NEAR

ECHOLOOP:
   READ
   CMP AL, 'Y'   ;CHECK IF ITS YES OR NO AS AN ANSWER
   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       ;IF YOU PRESS ESC YOU EXIT THE PROGRAM
   JMP ECHOLOOP

ECHO_ON:
   MOV ECHO,1
   RET
ECHO_OFF:
   RET

READ_ECHO ENDP
```

```
;===========READ THE BAUT RATE==============
READ_BAUDRATE PROC NEAR

BRLOOP:
   READ
   CMP AL, '1'
   JE BAUD_300
   CMP AL, '2'
   JE BAUD_600
   CMP AL, '3'
   JE BAUD_1200
   CMP AL, '4'
   JE BAUD_2400        ;CHECK THE BAUT RATE USER PREFERS
   CMP AL, '5'
   JE BAUD_4800
   CMP AL, '6'
   JE BAUD_9600
   CMP AL, 27
   JE QUIT               ;YOU CAN EXIT THE PROGRAM IF YOU PRESS ESC
   JMP BRLOOP

BAUD_300:              ;BITS 1,0 10-->7BITS 11->8BITS
   MOV AL,43H          ;BIT 2==0 => TERM. DIGITS=1BIT
   RET                 ;BITS 4,3 NO PARITY
BAUD_600:        ;OTHER BITS 7,6,5 RESPECTIVELY TO THE BAUD RATE SELECTION
   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_BAUDRATE ENDP

;=======INITIALIZATION OF THE PORT===============
OPEN_RS232 PROC NEAR
 JMP RS232_ST
BAUD_RATE LABEL WORD  ;DIVISOR FOR BR
 DW 1047    ; 110 baud rate
 DW 768     ; 150 baud rate
 DW 384     ; 300 baud rate
 DW 192     ; 600 baud rate
 DW 96      ; 1200 baud rate
 DW 48      ; 2400 baud rate
 DW 24      ; 4800 baud rate
 DW 12      ; 9600 baud rate
RS232_ST: STI ; Set interrupt flag
 ;INITILIZE PORT
 MOV AH,AL      ; Save in it parameters in AH
```

```
MOV DX,3FBH    ;SHOW TO THE LINE CONTROLLER
MOV AL,80H
OUT DX,AL      ;DEFINE THE DIVISOR OF BR
MOV DL,AH      ;DL HAS NOW THE BR
MOV CL,4
ROL DL,CL
AND DX,0EH
MOV DI,OFFSET BAUD_RATE
ADD DI,DX          ;SHOW TO THE MSBS OF THE DIVISOR
MOV DX,3F9H
MOV AL,CS:[DI]+1   ;TAKE THEM
OUT DX,AL          ;-> TO THE MSB

MOV DX,3F8H        ;SHOW TO THE LSB OF THE DEVISOR
MOV AL,CS:[DI]
OUT DX,AL

MOV DX,3FBH        ;REGISTER OF LINE CONTROLLER
MOV AL,AH          ;PARAMETERS IN AL
AND AL,01FH        ;IGNORE 5,6,7 BITS OF BR SELECT
OUT DX,AL          ;WRITE TO THE LINE CONTROLLER THE PARAMETERS
MOV DX,3F9H        ;SHOW TO THE INTERRUPT ENABLE REGISTER
MOV AL,0H
OUT DX,AL          ;DISABLE INT
RET
OPEN_RS232 ENDP
```

```
;=======SPLIT THE TERMINALS======================

SCREEN_SPLIT PROC NEAR

   MOV CX, 24  ;THERE ARE 25 LINES IN THE TERMINAL

SPLIT_LOOP:

   GO_THERE_CX 39   ;CURSOR GOES TO MIDDLE COL AND TO THE LINE THAT CX
SHOWS

   PRINT_THERE 179  ; ASCII FOR '|'

   GO_THERE_CX 40   ;CURSOR GOES TO MIDDLE COL AND TO THE LINE THAT CX
SHOWS

   PRINT_THERE 179  ; ASCII FOR '|'

   LOOP SPLIT_LOOP

   GO_THERE_CX 39   ;CURSOR GOES TO MIDDLE COL AND TO THE LINE THAT CX
SHOWS

   PRINT_THERE 179  ; ASCII FOR '|'

   GO_THERE_CX 40

   PRINT_THERE 179

   RET

SCREEN_SPLIT ENDP


;==== READ 1 CHAR FROM RS232 PORT ================

RXCH_RS232 PROC NEAR   ;FROM THE LAB FILE

   MOV DX,3FDH       ;SHOW TO THE LINE CONTROL REG

   IN AL,DX          ;AL<-LINE STATUS

   TEST AL,1         ;CHAR?

   JZ NOTHING

   SUB DX,5          ;IF YES SHOW TO INPUT BUFFER

   IN AL,DX          ;TAKE THE CHAR IN AL

   JMP EXIT2

NOTHING:

   MOV AL,0          ;0 MEANS NO CHAR
```

```
EXIT2:
   RET

RXCH_RS232 ENDP

;============WRITE 1 CHARACTER TO THE PORT=========
TXCH_RS232 PROC NEAR
   PUSH AX      ;AL HAS THE CHARACTER THAT WE WANT TO SEND
   MOV DX,03FDH ;SHOW TO THE LINE CONTROLLER REG

TXCH_RS232_2:
   IN AL,DX       ;GET THE STATUS IN AL
   TEST AL,020H    ;IS BIT5 ZERO?
   JZ TXCH_RS232_2 ;IF IT IS THEN BUFFER IS NOT EMPTY

   SUB DX,5       ;ELSE SHOW TO THE EMPTY BUFFER
   POP AX
   OUT DX,AL      ;SEND HIM THE CHARACTER
   RET
TXCH_RS232 ENDP

CODE ENDS
END MAIN
```