****

**Σχολή Ηλεκτρολόγων Μηχανικών**

**&**

**Μηχανικών Υπολογιστών**

**Εργαστήριο Μικροϋπολογιστών**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**ΔΕΥΤΕΡΗ ΑΣΚΗΣΗ 8086**

**Λαμπράκος Χρήστος, Α.Μ: 03112062**

**Μανδηλαράς Νικηφόρος, Α.Μ: 03112012**

**Σπαθαράκης Δημήτριος, Α.Μ: 03113523**

**Ομάδα C05**

**ΈβδομοΕξάμηνο**

**Παραδοτέα: 29/11/2015**

**ΤΕΤΑΡΤΗ ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ**

**1. Αριθμομηχανή**

Το πρόγραμμα μας χωρίζεται σε 3 βασικές συναρτήσεις. Μία που διαβάζει μέχρι 3 μονοψήφια δεκαεξαδικά ψηφία και τους αποθηκεύει σε Ascii code στον καταχωρητή AL (συνάρτηση Read hex). Μία που παίρνει ένα 16bit αριθμό και τον τυπώνει σε δεκαεξαδική μορφή (συνάρτηση PRINT HEX). Και τέλος μία που παίρνει πάλι έναν 16bit αριθμό και τον τυπώνει σε δεκαδική μορφή. Αυτές καλούνται από την main η οποία το μόνο επιπλέον που κάνει είναι να ελέγχει αν δόθηκαν λιγότεροι από τρεις αριθμοί μαζί με "+" ή "-". Αν το σύμβολο δεν είναι σωστό να ξαναζητάει σωστό τελεστή, πράγμα που κάνει και στην περίπτωση που διαβάζει τρεις ακριβώς αριθμούς. Έπειτα καλεί άλλη μία φορά την Read Hex για τον δεύτερο αριθμό και μετά περιμένει αντίστοιχα με πριν το σύμβολο "=". Με το που το δεχτεί τυπώνει πρώτα σε δεκαεξαδική μορφή και μετά σε δεκαδική καλώντας τις αντίστοιχες συναρτήσεις. Σε οποιαδήποτε στιγμή δοθεί το πλήκτρο "Q" το πρόγραμμα τερματίζει

Main:

|  |
| --- |
| EXTRN RD\_HX:FAR  EXTRN PRT\_HX:FAR  EXTRN PRT\_DEC:FAR  D\_SEG SEGMENT  NUM1 DW 0  BUFF DB 2 DUP(?)  NUM2 DW 0  OPER DB 0  RESUL DW 0  D\_SEG ENDS  ;================================================================  C\_SEG SEGMENT  ASSUME CS:C\_SEG,DS:D\_SEG  MAIN PROC FAR  GIVE1: CALL RD\_HX  MOV NUM1,DX ;save number  CMP AL,2BH ;digit last given?  JGE SYM1 ;symbol, then  GVSM1: MOV AH,07H ;read symbol(no echo)  INT 21H  SYM1: CMP AL,2BH  JE GIVE2  CMP AL,2DH ;accept only '+' or '-' here  JNE GVSM1  GIVE2: MOV OPER,AL ;save operator  MOV DL,AL  MOV AH,02H  INT 21H  CALL RD\_HX  MOV NUM2,DX  GVSM2: CMP AL,3DH ;accept only '=' here  JE RATE  MOV AH,07H  INT 21H  JMP GVSM2  RATE: MOV DL,AL  MOV AH,02H  INT 21H  MOV DX,NUM1  CMP OPER,2BH  JNE MINUS  ADD DX,NUM2 ;result is ready  JMP GOOD  MINUS: SUB DX,NUM2  GOOD: MOV RESUL,DX  CALL PRT\_HX  MOV DL,3DH ;print '='  MOV AH,02H  INT 21H  MOV DX,RESUL  CALL PRT\_DEC  MOV AH,4CH  INT 21H  MAIN ENDP  ;================================================================  C\_SEG ENDS  END MAIN  PRINT DEC  LIB SEGMENT 'CODE'  ASSUME CS:LIB  PUBLIC PRT\_DEC  ;=================================================================  ;Prints 16-bit number saved in DX, in decimal form. Also works  ;with negative numbers. Does NOT print a new line.  ;=================================================================  PRT\_DEC PROC FAR  PUSH BX  PUSH CX  MOV BP,DX  ROL DX,1 ;check sign  JNC POSI  ROR DX,1 ;fix  NOT DX  ADD DX,1 ;two's complement  MOV BX,DX  MOV DL,2DH ;print neg sign  MOV AH,02H  INT 21H  MOV DX,BX  FXD: MOV CX,00 ;CL -> decades. CH ->hundreds.  MOV BX,00  CADES: CMP DX,0AH ;BL -> units. BH -> thousands  JL THOU  SUB DX,0AH  INC CL  CMP CL,0AH  JL CADES  SUB CL,0AH  INC CH  JMP CADES  THOU: MOV BL,DL  MOV BH,00  LOOPB: CMP CH,0AH  JL NOTH  SUB CH,0AH  INC BH  JMP LOOPB  NOTH: CMP BH,0  JE NOO  MOV DL,BH  ADD DL,30H  MOV AH,02H  INT 21H  NOO: MOV DL,CH  ADD DL,30H  MOV AH,02H  INT 21H  MOV DL,CL  ADD DL,30H  MOV AH,02H  INT 21H  MOV DL,BL  ADD DL,30H  MOV AH,02H  INT 21H  MOV DX,BP  POP CX  POP BX  RET  POSI: ROR DX,1  JMP FXD  PRT\_DEC ENDP  LIB ENDS  END  ;================================================================  PRINT HEX  LIB SEGMENT 'CODE'  ASSUME CS:LIB  PUBLIC PRT\_HX  ;=================================================================  ;Prints 16-bit number saved in DX, in hexadeximal form. Also works  ;with negative numbers. Does NOT print a new line.  ;=================================================================  PRT\_HX PROC FAR  PUSH CX  PUSH BX  ROL DX,1 ;check sign  JNC POS  ROR DX,1 ;fix  NOT DX  ADD DX,1 ;two's complement  MOV BX,DX ;save number  MOV DL,2DH ;print neg sign  MOV AH,02H  INT 21H  BLOOPA: MOV CL,04H ;loop counter  LOOPA: MOV DX,BX ;retrieve number  ROL DX,1 ;ROL DX,4  ROL DX,1  ROL DX,1  ROL DX,1  MOV BX,DX ;save rotated number  AND DX,000FH ;examine 4 bits  CMP DL,09H  JG BGG  CMP CL,4 ;it's not beautiful  JL CEED ;to print zero  CMP DL,00H ;as the first number  JE NXT  CEED: ADD DL,30H ;get ASCII code  BCK: MOV AH,02H ;and print  INT 21H  NXT: DEC CL ;update counter  JNZ LOOPA  MOV DX,BX ;retrieve number  POP BX  POP CX  RET  BGG: ADD DL,37H  JMP BCK  POS: ROR DX,1 ;fix number shifted because of  MOV BX,DX ;sign check  JMP BLOOPA  PRT\_HX ENDP  LIB ENDS  END  ;================================================================  READ HEX  D\_SEG SEGMENT  BUFF DB 2 DUP(?)  D\_SEG ENDS  LIB SEGMENT 'CODE'  ASSUME CS:LIB, DS:D\_SEG  PUBLIC RD\_HX  ;=================================================================  ;Reads HEX number of up to 3 digits. Calculates actual numeric  ;value, saved at DX. Finishes when a symbol is given, or when Q  ;is pressed. Symbol ASCII code saved at AL.  ;=================================================================  RD\_HX PROC FAR  PUSH BX  PUSH CX  MOV CX,0 ;digit counter  INPT: CMP CX,03H  JGE GOUT  MOV AH,07H ;read (no echo)  INT 21H  CMP AL,2BH ;'+' ?  JE SYM  CMP AL,2DH ;'-' ?  JE SYM  CMP AL,30H ;number?  JL INPT ;invalid is only possibility  CMP AL,39H  JLE NUM  CMP AL,3DH ;'=' ?  JE SYM  CHK3: CMP AL,41H ;letter?  JL INPT  CMP AL,46H  JLE LETT  CMP AL,51H ;'Q' ?  JNE INPT ;not valid input. again  KILL: MOV AH,4CH ;'Q' given -> stop program  INT 21H  LETT: INC CX  MOV DL,AL  MOV AH,02H  INT 21H  SUB AL,37H ;get actual numeric value  JMP FNSH  NUM: MOV DL,AL  MOV AH,02H  INT 21H  INC CX  SUB AL,30H ;get actual numeric value  FNSH: MOV AH,0 ;zero-pad  MOV BX,AX ;add last digit to result  CMP CX,01H ;if this was first digit, result  JG FCK ;is ready  LEA BP,BUFF  MOV [BP],AL ;update buffer  MOV DX,BX  JMP INPT  FCK: CMP CX,02H  JE GNEWS  BDNEWS: LEA BP,BUFF  MOV AL,[BP]  MOV SI,10H  MUL SI  ADD BX,AX  INC BP  MOV AL,[BP]  MOV SI,100H  MUL SI  ADD BX,AX  MOV DX,BX  JMP GOUT ;no more digits to give!  GNEWS: LEA BP,BUFF ;shift BUFF to the right  MOV DL,[BP]  MOV [BP],AL  INC BP  MOV [BP],DL  MOV AL,DL  MOV SI,10H  MUL SI  ADD BX,AX  MOV DX,BX  JMP INPT ;can press one more  SYM: CMP CL,00 ;has there been at least one  JE INPT ;digit?  GOUT: POP CX  POP BX  RET  RD\_HX ENDP  LIB ENDS  END  ;================================================================ |

**2. Τερματικό**

Στη δεύτερη άσκηση προσομοιώσαμε, μέσω dosbox, λειτουργία τερματικού. Ανοίξαμε 2 instances του dosbox (το καθένα με διαφορετική ρύθμισηστο configurationfile ):

• Στο αρχείο dosbox.conf του πρώτου instance, θα πρέπει να μπει η επιλογή **serial1 nullmodem**

• Στο αντίστοιχο αρχείο του δεύτερου instance, θα γραφτεί **serial1nullmodem server:localhost**

Στη συνέχεια, σε κάθε instance τρέξαμε το εκτελέσιμο αρχείο που παράγει ο παρακάτω κώδικας σε assembly 8086 .

Κάθε terminal χωρίστηκεσε 2 μέρη ( Server-receiver , Terminal-sender) και γράφοντας στο terminal λαμβάναμε τους αντίστοιχους χαρακτήρες στο server. Για την υλοποίηση του παραπάνω χρησιμοποιήσαμε τις δοθείσες ρουτίνες του RS232, επίσης φτιάξαμε μία βιβλιοθήκη που ουσιαστικά περιλάμβανε το κυρίως πρόγραμμα μας. Το source code συμπληρώνεται με κάποια macros που χρησιμοποιήθηκαν.

|  |
| --- |
| MAIN  INCLUDE MACROS.TXT INCLUDE EXTRA\_MACROS.TXT INCLUDE RS232\_ROUTINES.INC INCLUDE TERM\_LIB.INC  org 100h .data  PKEY DB "Press any key...$"  NEW\_LINE DB 0AH,0DH,"$  LOC\_MSG DB "LOCAL$" REM\_MSG DB "REMOTE$" SEPERATOR DB 80 DUP(0C4H),"$" ECHO\_MSG DB "With(1) or Without(0) ECHO? $"  BAUD\_RATE\_MSG DB "Give Baud ate:(1)300,(2)600,(3)1200,(4)2400,(5)4800,(6)9600:$" LOCAL\_LIN DB 0 LOCAL\_COL DB 0 REMOTE\_LIN DB 12 REMOTE\_COL DB 0 WHERE\_2\_WRITE DB 0  ECHO\_FLG DB 0 B\_R\_CHOICE DB 0 .code MAIN PROC FAR  ;=-=-=-=-==-=-=-=-=-=-=-CODE-=-=-=-=-=-=-=-=-=-=-==-=-=-=-=-=-=-=  START: CALL INPUT\_CHOOSE MOV AL,B\_R\_CHOICE ;sthing 0000 0xxx  CALL OPEN\_RS232  CALL PRINT\_START\_SCRN CALL MAIN\_LOOP  EXODOS:  SCROLL\_UP\_WIN 0 0 24 80 0 ;to clear screen LOCATE 0 0 0 ;to locate at the begining EXIT  MAIN ENDP  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  DEFINE\_OPEN\_RS232 DEFINE\_RXCH\_RS232 DEFINE\_TXCH\_RS232 DEFINE\_INPUT\_CHOOSE DEFINE\_PRINT\_START\_SCRN DEFINE\_MAIN\_LOOP  RS232 Routines  DEFINE\_OPEN\_RS232 MACRO  LOCAL START,SKIP\_OPEN\_RS232  JMP SKIP\_OPEN\_RS232  ;This routine initializes RS232 standard communication  ;Messes with AX,DX,DI  OPEN\_RS232 PROC NEAR  JMP START  BAUD\_RATE\_DIVISOR LABEL WORD ;divisor=115200/baud\_rate, same declaration as  DW 1047 ;110 baud rate (OFFSET BAUD\_RATE\_DIVISOR)+0 BR=000 ;BAUD\_RATE\_DIVISOR DW 1047,768,384,192,96,48,24,12  DW 768 ;150 baud rate (OFFSET BAUD\_RATE\_DIVISOR)+2 BR=001  DW 384 ;300 baud rate (OFFSET BAUD\_RATE\_DIVISOR)+4 BR=010  DW 192 ;600 baud rate (OFFSET BAUD\_RATE\_DIVISOR)+6 BR=011  DW 96 ;1200 baud rate (OFFSET BAUD\_RATE\_DIVISOR)+8 BR=100  DW 48 ;2400 baud rate (OFFSET BAUD\_RATE\_DIVISOR)+10 BR=101  DW 24 ;4800 baud rate (OFFSET BAUD\_RATE\_DIVISOR)+12 BR=110  DW 12 ;9600 baud rate (OFFSET BAUD\_RATE\_DIVISOR)+14 BR=111 "+14->LSByte, +15->MSByte"  START:  STI ;Set interrupt flag != CLI; Clear Interrupt Flag (?)  ; Initial Values of RS232  MOV AH,AL ;AH<-AL parameters:BR2|BR1|BR0|EVEN\_OR\_ODD\_PARITY|PARITY\_ON|NUM\_STOP\_BIT|WORD\_LENGTH\_1|WORD\_LENGTH\_0  MOV DX,3FBH ;Line Control REGISTER address  MOV AL,80H ;AL<-1000 0000 : DLAB=1  OUT DX,AL ;send to register  MOV DL,AH ;DL<- Parameters  ROL DL,4  AND DX,0EH ;DH<-00H, DL<-0000 BR2|BR1|BR0|0 --->offset=0,2,4,6,8,10,12,14  MOV DI,OFFSET BAUD\_RATE\_DIVISOR  ADD DI,DX ;DI<-memory address of correct divisor  MOV DX,3F9H ;MSByte of Baudrate divisor REGISTER adddress (DLAB=1)  MOV AL,CS:[DI]+1;CS:[DI]+1 -> MSByte of divisor  OUT DX,AL ;send to register  MOV DX,3F8H ;LSByte of Baudrate divisor (DLAB=1)  MOV AL,CS:[DI] ;CS:[DI] -> LSByte of divisor  OUT DX,AL ;send to register  MOV DX,3FBH ;Line Control REGISTER address  MOV AL,AH ;AL<-parameters  AND AL,1FH ;AL<-0(DLAB)|0(SOUT not deactivated)|0(normal parity bit)|EVEN\_OR\_ODD\_PARITY|PARITY\_ON|NUM\_STOP\_BIT|WORD\_LENGTH\_1|WORD\_LENGTH\_0  OUT DX,AL ;send to register  MOV DX,3F9H ;Interrupt Enable REGISTER address  MOV AL,0 ;disabled interrupts 0 Rx data int. enable  ;1 Tx holding reg. empty int.  ;2 Rx status int. enable (ie Parity, Framing, overrun and BREAK enable).  ;3 Modem signal change int. enable.  OUT DX,AL  RET  OPEN\_RS232 ENDP  SKIP\_OPEN\_RS232:  DEFINE\_OPEN\_RS232 ENDM  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  DEFINE\_RXCH\_RS232 MACRO  LOCAL END\_RXCH\_RS232  LOCAL SKIP\_RXCH\_RS232  JMP SKIP\_RXCH\_RS232  ;This routine READS a char from serial port  ;Messes with AL,DX  RXCH\_RS232 PROC NEAR  MOV DX,3FDH ;Line Status REGISTER Address  IN AL,DX ;Input Status of Line (to check if there is something to read)  AND AL,1 ;AL (AND) 00000001 ->IF NonZero => DR=1 => something has come  JZ END\_RXCH\_RS232 ;AL<-0(NUL) means there is nothing to Read (!\*Hope we don't receive NUL char from serial port\*!)  MOV DX,3F8H ;Data Read/Write REGISTER address.  IN AL,DX ;READ IT!  END\_RXCH\_RS232:  RET  RXCH\_RS232 ENDP  SKIP\_RXCH\_RS232:  DEFINE\_RXCH\_RS232 ENDM  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  DEFINE\_TXCH\_RS232 MACRO  LOCAL SKIP\_TXCH\_RS232  LOCAL TXCH\_RS232\_2  JMP SKIP\_TXCH\_RS232  ;This routine SENDS a char to serial port  ;Messes with AL(there is the CHAR\_2\_SEND),DX  TXCH\_RS232 PROC NEAR  PUSH AX  MOV DX,3FDH ;Line Status Register Address  TXCH\_RS232\_2:  IN AL,DX ;Input Status of Line (to check if TRANSMITTER REGISTER is clear to send)  TEST AL,20H ;AL (AND) 0010 0000 ->IF NonZero => THRE=1 => Transmitter Holding Register is empty, we can send  JZ TXCH\_RS232\_2 ;Loop from proc\_begin, until Transmitter Register is empty!  MOV DX,3F8H ;Data Read/Write REGISTER address.  POP AX ;Retrieve AL<-CHAR\_2\_SEND  OUT DX,AL ;Send it to Transmitter Register(=Data Read/Write Register)  RET  TXCH\_RS232 ENDP  SKIP\_TXCH\_RS232:  DEFINE\_TXCH\_RS232 ENDM  TERM LIB  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*PROJECT\*\*\*\*4-2\*\*\*LIBRARY\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ;\* This library defines three procedures \*  ;\* \*  ;\* 1.INPUT\_CHOOSE initializes ECHO CHOICE and BAUD RATE \*  ;\* 2.PRINT\_START\_SCRN prints the main screen \*  ;\* 3.MAIN\_LOOP main loop procedure of our program \*  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  DEFINE\_INPUT\_CHOOSE MACRO  LOCAL ECHO\_ERR,BAUD\_RATE\_ERR  LOCAL SKIP\_INPUT\_CHOOSE  JMP SKIP\_INPUT\_CHOOSE  INPUT\_CHOOSE PROC NEAR  SCROLL\_UP\_WIN 0 0 24 80 0  LOCATE 0 0 0  PRINT\_STRING ECHO\_MSG  ECHO\_ERR:  READ  CMP AL,30H  JB ECHO\_ERR  CMP AL,31H  JA ECHO\_ERR  PRINT AL  SUB AL,30H  MOV ECHO\_FLG,AL  PRINT\_STRING NEW\_LINE  PRINT\_STRING BAUD\_RATE\_MSG  BAUD\_RATE\_ERR:  READ  CMP AL,31H  JB BAUD\_RATE\_ERR  CMP AL,36H  JA BAUD\_RATE\_ERR  PRINT AL  SUB AL,2FH ;example(gave '1'):31h=29h=2h->010->baud rate 300  SHL AL,5 ;AL<-xxx0 0000  AND AL,0E0H  ADD AL,3 ;AL<-xxx0 0011 (xxx||EVEN\_OR\_ODD\_PARITY|PARITY\_ON|NUM\_STOP\_BIT|WORD\_LENGTH\_1|WORD\_LENGTH\_0)  MOV B\_R\_CHOICE,AL  PRINT\_STRING NEW\_LINE  PRINT\_STRING PKEY  READ  SCROLL\_UP\_WIN 0 0 3 80 0  RET  INPUT\_CHOOSE ENDP  SKIP\_INPUT\_CHOOSE:  DEFINE\_INPUT\_CHOOSE ENDM  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  DEFINE\_PRINT\_START\_SCRN MACRO  LOCAL SKIP\_PRINT\_START\_SCRN  JMP SKIP\_PRINT\_START\_SCRN  PRINT\_START\_SCRN PROC NEAR  LOCATE 0 0 00H  PRINT\_STRING LOC\_MSG  MOV LOCAL\_LIN,1  LOCATE 11 0 00H  PRINT\_STRING SEPERATOR  LOCATE 12 0 0  PRINT\_STRING REM\_MSG  MOV REMOTE\_LIN,13  RET  PRINT\_START\_SCRN ENDP  SKIP\_PRINT\_START\_SCRN:  DEFINE\_PRINT\_START\_SCRN ENDM  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  DEFINE\_MAIN\_LOOP MACRO  LOCAL FULL\_REM\_WIN,KEY\_RECEIVED  LOCAL FULL\_REM\_WIN\_2,GO\_PRINT\_RECEIVED  LOCAL SEND\_CHECK,FULL\_LOC\_WIN  LOCAL KEY\_PUSHED,FULL\_LOC\_WIN\_2  LOCAL GO\_PRINT,GO\_ON\_SEND  LOCAL SKIP\_MAIN\_LOOP  JMP SKIP\_MAIN\_LOOP  MAIN\_LOOP PROC NEAR  CALL RXCH\_RS232 ;AL<-0 (NUL) means there is nothing to Read  CMP AL,0 ;else AL<-char received  JE SEND\_CHECK  ;[section=CHAR RECEIVED]  CMP AL,0DH ;check if ENTER received  JNE KEY\_RECEIVED ;if not ENTER jump to KEY\_PUSHED  CMP REMOTE\_LIN,22 ;Lines can be printed-limit  JE FULL\_REM\_WIN  ADD REMOTE\_LIN,1  MOV REMOTE\_COL,0  JMP SEND\_CHECK  FULL\_REM\_WIN:  SCROLL\_UP\_WIN 13 0 22 79 1  MOV REMOTE\_COL,0  JMP SEND\_CHECK  KEY\_RECEIVED:  CMP REMOTE\_COL,80 ;0-79 column have been written (80 chars)  JNE GO\_PRINT\_RECEIVED  CMP REMOTE\_LIN,10 ;Lines can be printed-limit  JE FULL\_REM\_WIN\_2  ADD REMOTE\_LIN,1  MOV REMOTE\_COL,0  JMP GO\_PRINT\_RECEIVED  FULL\_REM\_WIN\_2:  SCROLL\_UP\_WIN 13 0 22 79 1  MOV REMOTE\_COL,0  GO\_PRINT\_RECEIVED:  LOCATE REMOTE\_LIN REMOTE\_COL 0  PRINT AL  ADD REMOTE\_COL,1  ;[\section]  SEND\_CHECK:  READ\_NW ;if ZF=0 there was something to read (in AL)  JZ MAIN\_LOOP ;if ZF=1 loop!  CMP AL,1BH ;check if ESC  JE EXODOS  CMP ECHO\_FLG,1  JNE GO\_ON\_SEND  ;[section=ECHO ON]  CMP AL,0DH ;check if ENTER  JNE KEY\_PUSHED ;if not ENTER jump to KEY\_PUSHED  CMP LOCAL\_LIN,10 ;Lines can be printed-limit  JE FULL\_LOC\_WIN  ADD LOCAL\_LIN,1  MOV LOCAL\_COL,0  JMP GO\_ON\_SEND  FULL\_LOC\_WIN:  SCROLL\_UP\_WIN 1 0 10 79 1  MOV LOCAL\_COL,0  JMP GO\_ON\_SEND  KEY\_PUSHED:  CMP LOCAL\_COL,80 ;0-79 column have been written (80 chars)  JNE GO\_PRINT  CMP LOCAL\_LIN,10 ;Lines can be printed-limit  JE FULL\_LOC\_WIN\_2  ADD LOCAL\_LIN,1  MOV LOCAL\_COL,0  JMP GO\_PRINT  FULL\_LOC\_WIN\_2:  SCROLL\_UP\_WIN 1 0 10 79 1  MOV LOCAL\_COL,0  GO\_PRINT:  LOCATE LOCAL\_LIN LOCAL\_COL 0  PRINT AL  ADD LOCAL\_COL,1  ;[\section]  GO\_ON\_SEND:  CALL TXCH\_RS232  JMP MAIN\_LOOP  RET ;not necessary, because it's infinite loop(ends with jump to EXODOS)  MAIN\_LOOP ENDP  SKIP\_MAIN\_LOOP:  DEFINE\_MAIN\_LOOP ENDM  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  MACROS  ;This macro change registers AH,AL  READ MACRO  MOV AH,8  INT 21H  ENDM  ;This macro changes registers AH,DL  PRINT MACRO CHAR  PUSH AX  PUSH DX  MOV DL,CHAR  MOV AH,02H  INT 21H  POP DX  POP AX  ENDM  ;This macro change registers AH,DX  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  PRINT\_NUM MACRO CHAR  MOV DL, CHAR  ADD DL, 30H  MOV AH, 2  INT 21H  ENDM  PAUSE MACRO  PUSH AX  PUSH DX  LEA DX,PKEY ;<=>MOV DX, OFFSET PKEY;GIVES THE OFFSET OF PKEY TO DX  MOV AH,9  INT 21H ;OUTPUT STRING AT DS:DX  MOV AH,8 ;WAIT FOR PRESSING OF A KEY  INT 21H ;WITHOUT ECHO->8  PRINT 0AH  PRINT 0DH  POP DX  POP AX  ENDM  EXIT MACRO  MOV AH,4CH  INT 21H  ENDM  SCROLL\_UP\_WIN MACRO START\_LIN START\_COL END\_LIN END\_COL UP\_NUM  ;messes with AX,BH,CX,DX  PUSH AX  MOV AH,06H  MOV AL,UP\_NUM ;number of lines to scroll up|0->all lines  MOV CH,START\_LIN  MOV CL,START\_COL  MOV DH,END\_LIN  MOV DL,END\_COL  MOV BH,07H ;attribute:0000(black) bckgrnd clr, 0111(light grey)char clr  INT 10H  POP AX  ENDM  SCROLL\_DOWN\_WIN MACRO START\_LIN START\_COL END\_LIN END\_COL UP\_NUM  PUSH AX  MOV AH,07H  MOV AL,UP\_NUM ;number of lines to scroll up|0->all lines  MOV CH,START\_LIN  MOV CL,START\_COL  MOV DH,END\_LIN  MOV DL,END\_COL  MOV BH,07H ;attribute:0000(black) bckgrnd clr, 0111(light grey)char clr  INT 10H  POP AX  ENDM  READ\_NW MACRO  ;messes with AX,DL,returns in AL=char, if ZF=0(there was something to read)  ;reads without echo  MOV AH,06H  MOV DL,0FFH  INT 21H  ENDM  LOCATE MACRO LIN COL PAGE  ;messes with AH,DX,BH  MOV AH,02H  MOV DH,LIN  MOV DL,COL  MOV BH,PAGE  INT 10H  ENDM  PRINT\_BIOS MACRO CHAR  MOV AH,0AH ;funct code  MOV AL,CHAR  MOV BH,00H ;page num  MOV CX,1 ;times we print char  INT 10H  ENDM |