ΣΥΣΤΗΜΑΤΑ ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΩΝ 5η ΟΜΑΔΑ ΑΣΚΗΣΕΩΝ

ΦΟΙΤΗΤΕΣ

Αθανασίου Ιωάννης / Α.Μ.:03117041 /6ο Εξάμηνο

Καραβαγγέλης Αθανάσιος / Α.Μ.:03117022 / 6ο Εξάμηνο

ΑΣΚΗΣΗ 1η

- Ο κώδικας που φτιάξαμε με τις μακροεντολές, στο αρχείο "MACROS1.asm":

```
READ MACRO        ; AL <- ascii code
    MOV AH, 8
    INT 21H
ENDM

PRINT MACRO CHAR
     MOV DL, CHAR
     MOV AH, 2
     INT 21H
ENDM

PRINT_STR MACRO STRING
   MOV DX, OFFSET STRING
   MOV AH,9
   INT 21H
ENDM

EXIT MACRO
    MOV AX, 4C00H
    INT 21H
ENDM

NEW_LINE MACRO
   PUSH DX
   PUSH AX ; save them because we will use them

   MOV DX,13  ; DL <- 0DH (newline)
   MOV AH,2   ; \
   INT 21H    ; /  print char macro

   MOV DX,10   ; DL <- 0AH (go to the beggining of the line)
   MOV AH,2    ; \
   INT 21H     ; /  print char macro

   POP AX
   POP DX  ; pop from the stack
ENDM
```

- Ο κώδικας που φτιάξαμε για το κυρίως πρόγραμμα, στο αρχείο "1.asm":

```
INCLUDE MACROS1.ASM
   .8086
   .MODEL SMALL
   .STACK 256
```

```
.DATA

.CODE
  ASSUME CS:CODE, DS:DATA

MAIN PROC FAR

        MOV AX, DATA
        MOV DS,AX

START:

    CALL HEX_KEYB       ;read 1ST digit
    CMP AL,'Q'
    JE FINISH
    ROL AL,4                        ; MOVE IT TO 4 LSB
    MOV BL,AL             ; MOVE IT TO 4 LSB

    CALL HEX_KEYB       ; read 2ND digit
    CMP AL,'Q'
    JE FINISH
    ADD BL,AL                       ; SUM IN BL

    NEW_LINE

    PUSH BX                         ; PRINT HEX
    CALL PRINT_HEX
    PRINT '='       ; =
    POP BX

    PUSH BX
    CALL PRINT_DEC    ; PRINT DEC
    PRINT '='
    POP BX                          ; =

    PUSH BX
    CALL PRINT_OCT    ; PRINT OCT
    PRINT '='
    POP BX                          ; =

    CALL PRINT_BIN    ; PRINT BIN

    JMP START

FINISH:
   EXIT

MAIN ENDP

; -------------------------------------------------

HEX_KEYB PROC NEAR
   PUSH DX
```

```
ARXH1:
   READ
   CMP AL,30H       ; if <= 30 then skip it
   JL ARXH1
   CMP AL,39H
   JG SYMBOL            ; INPUT >= 9 GO TO SYMBOL
   SUB AL,30H           ; HERE IT IS A NUMBER SO sub 30 FOR THE ascii
   JMP FINAL
SYMBOL:
   CMP AL,'Q'                   ; if Q, SAVE IT AND WE WILL QUIT AFTER
   JE FINAL
   CMP AL,'A'
   JL ARXH1         ; BAD SYMBOL
   CMP AL,'F'
   JG ARXH1         ; BAD SYMBOL
   SUB AL,37H        ; TURN TO ASCII
FINAL:
   POP DX
   RET
ENDP HEX_KEYB

; -------------------------------------------------

PRINT_DEC PROC NEAR
   MOV AH,0         ; AH<- 0
   MOV AL,BL                    ; AL <- the 2 digits
   MOV BL,10        ; BL <- 10, the divisor
   MOV CX,1                     ; decs counter

LOOP1:
   DIV BL                       ; AX<-AX/BL, AL<-phliko, AH<-ypoloipo
   PUSH AX       ; save decs
   CMP AL,0              ; if decs=0 -> WE HAVE PUSHED ALL THE DIGITS
   JE DEC_DIGITS            ;
   INC CX                       ; decs counter ++  FOR THE LOOP LATER
   MOV AH,0      ;
   JMP LOOP1             ; loop again

DEC_DIGITS:
   POP DX                       ; digit TO PRINT at DH (we pushed AX, with the ypoloipo at AH)
   MOV DL,DH     ; DL <- digit
   MOV DH,0
   ADD DX,30H     ; to get ascii code if the digit
   MOV AH,2
   INT 21H                      ; the 2 print steps
   LOOP DEC_DIGITS       ; repeat for each digit  (CX TIMES)
   RET
ENDP PRINT_DEC

; -------------------------------------------------


PRINT_OCT PROC NEAR     ; same process but we divide with 8 instead of 10
   MOV AH,0
```

```
    MOV AL,BL
    MOV BL,8
    MOV CX,1

LOOP2:
    DIV BL
    PUSH AX
    CMP AL,0
    JE OCT_DIGITS
    INC CX
    MOV AH,0
    JMP LOOP2

OCT_DIGITS:
    POP DX
    MOV DL,DH
    MOV DH,0
    ADD DX,30H
    MOV AH,2
    INT 21H
    LOOP OCT_DIGITS
    RET
ENDP PRINT_OCT

; -------------------------------------------------------

PRINT_BIN PROC NEAR     ; same process but we divide with 2 instead of 8
    MOV AH,0
    MOV AL,BL
    MOV BL,2
    MOV CX,1

LOOP3:
    DIV BL
    PUSH AX
    CMP AL,0
    JE BIN_DIGITS
    INC CX
    MOV AH,0
    JMP LOOP3

BIN_DIGITS:
    MOV DH, AL
    PUSH DX
    POP DX
DIGITS2:
    POP DX
    MOV DL,DH
    MOV DH,0
    ADD DX,30H
    MOV AH,2
    INT 21H
    LOOP DIGITS2
    RET
```

```
ENDP PRINT_BIN

; -------------------------------------------------------------

PRINT_HEX PROC NEAR    ; same process but we divide with 16 instead of 2
   MOV AH,0
   MOV AL,BL
   MOV BL,16
   MOV CX,1

LOOP4:
   DIV BL
   PUSH AX
   CMP AL,0
   JE HEX_DIGITS
   INC CX
   MOV AH,0
   JMP LOOP4

HEX_DIGITS:
   POP DX
   MOV DL,DH
   MOV DH,0
   CMP DL,10      ; difference to make the ascii code
   JL  USUAL
   ADD DX,37H
   JMP AFTER_ASCII
USUAL:
   ADD DX,30H     ; number < 10 so the digit is 0,1,...,9 and we need to add 41 to get the ascii code
AFTER_ASCII:
   MOV AH,2
   INT 21H
   LOOP HEX_DIGITS
   RET
ENDP PRINT_HEX
```

ΑΣΚΗΣΗ 2η

- Ο κώδικας που φτιάξαμε με τις μακροεντολές για τις ασκήσεις 2,3,4 , στο αρχείο "MACRO_INST.asm":

```
READ MACRO
    MOV AH, 8
    INT 21H
ENDM

PRINT MACRO CHAR
     MOV DL, CHAR
     MOV AH, 2
     INT 21H
ENDM

PRINT_STR MACRO STRING
```

```
   MOV DX, OFFSET STRING
   MOV AH,9
   INT 21H
ENDM

EXIT MACRO
    MOV AX, 4C00H
    INT 21H
ENDM

NEW_LINE MACRO
   PUSH DX
   PUSH AX
   MOV DX,13
   MOV AH,2
   INT 21H
   MOV DX,10
   MOV AH,2
   INT 21H
   POP AX
   POP DX
ENDM

PRINT_DEC MACRO
   ADD DL, 30H
   MOV AH,2
   INT 21H
ENDM

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

Ο κώδικας μας σε assembly 8086 για την άσκηση 2 και στο αρχείο "2.asm":

```
INCLUDE MACRO_INST.ASM
   .8086
   .MODEL SMALL
   .STACK 256

DATA_SEG SEGMENT
   TABLE DB 256 DUP(?)
   MIN db ?
   MAX db ?
DATA_SEG ENDS

CODE_SEG SEGMENT
   ASSUME CS:CODE_SEG, DS:DATA_SEG

MAIN PROC FAR
   MOV AX,DATA_SEG
   MOV DS,AX
```

```
    MOV AL,254                  ;first number to be stored -> 254
    MOV DI,0                    ;index = 0

TABLE_SV:
    MOV [TABLE + DI],AL         ;save number in TABLE
    DEC AL                      ;AL<- AL - 1
    INC DI                      ;index++
    CMP AL,0                    ;if we reached 0 then exit loop
    JNE TABLE_SV                ;else continue

    MOV [TABLE + DI],255        ;255 will be stored last place after 0

    MOV DI,0                    ;initialize index
    MOV AH,0                    ;initialize AH
    MOV DX,0                    ;initialize DX

MEAN:
    MOV AL,[TABLE + DI]         ;load even number
    ADD DX,AX                   ;the sum which will then be divided for the mean value
    ADD DI,2                    ;we only want even numbers so we add 2
    CMP DI,254
    JL MEAN                     ;if di > 254 exit loop else continue adding

    ADD AL,[TABLE + DI]
    ADD DX,AX                   ;add to sum the last place of the table
    MOV AX,DX                   ;move sum to accumulator AX
    MOV BH,0
    MOV BL,128
    DIV BL                      ;divide sum with 128-> the number of even numbers

    MOV AH,0
                                ;print mean value in hex form
    CALL PRINT_NUMBER
    NEW_LINE                    ;print new line

    MOV DI,0                    ;initialize DI->0
    MOV MIN,0xFF                ;initialize MIN-> 255
    MOV MAX,0                   ;initialize MAX-> 0

MIN_AND_MAX:
    MOV AL,[TABLE + DI]         ;load number of TABLE[DI] to AL
    CMP MIN,AL                  ;compare with MIN
    JB IT
    MOV MIN,AL                  ;if AL < MIN then MIN = AL
IT:
    CMP MAX,AL                  ;compare with MAX
    JA IT2
    MOV MAX,AL                  ;if AL > MAX then MAX = AL
IT2:
    INC DI                      ;increase index
    CMP DI,256
    JNE MIN_AND_MAX             ;if we have reached 256 - 1 exit

    MOV AH,0
```

```
   MOV AL,MIN
   CALL PRINT_NUMBER          ;print min in hex form
   PRINT ' '                  ;print ' '
   MOV AH,0
   MOV AL,MAX
   CALL PRINT_NUMBER          ;print max in hex form
   NEW_LINE                   ;print new line


ENDP
EXIT

PRINT_NUMBER PROC NEAR        ;routine for printing number in hex

   MOV BL,16
   MOV CX,1                   ;sixteens count
LOOP1:
   DIV BL                     ;divide number with 16
   PUSH AX                    ;save units
   CMP AL,0                   ;continue until we have found the digits
   JE PRINT_HEX
   INC CX                     ;increment number of sixteens
   MOV AH,0
   JMP LOOP1
PRINT_HEX:
   POP DX
   MOV DL,DH
   MOV DH,0
   CMP DL,09H
   JLE DO
   ADD DX,37H                 ;ASCII offset for A B C D E F
   JMP DO1

DO:                           ;ASCII offset for single digit
   ADD DX,30H
DO1:
   MOV AH,2
   INT 21H
   LOOP PRINT_HEX
   RET
ENDP PRINT_NUMBER
```

ΑΣΚΗΣΗ 3η

Ο κώδικας μας σε assembly 8086 για την άσκηση 3 και στο αρχείο 3.asm:

```
INCLUDE MACRO_INST.ASM
   .8086
   .MODEL SMALL
   .STACK 256
   .DATA

   STRING DB "EXERCISE 3:" , '$'
   .CODE
```

```
ASSUME CS:CODE


MAIN PROC FAR
    MOV AX,@DATA
    MOV DS,AX
    LEA DX,STRING           ;load the address of the string in DX and output the string loaded in DX
    MOV AH,09H
    INT 21H
    NEW_LINE

    PRINT "X"               ;print X=
    PRINT "="               ;read first digit of first number
    CALL HEX_KEYB
    MOV DL,AL               ;4 lsb of DL contain the first digit
    MOV BL,BH
    CALL HEX_KEYB           ;read second digit of first number
    MOV DH,AL               ;4 lsb of DH contain the second digit

    PUSH DX
    PRINT BL
    PRINT BH
    POP DX

    ROL DL,4                ;4 msb of DL now have the first digit
    ADD DL,DH               ;DL has the first number after addition of DH to it

    PUSH DX

    PRINT ' '
    PRINT 'Y'               ;print Y= with a space before
    PRINT '='
    CALL HEX_KEYB           ;read first digit of second number
    MOV DL,AL               ;4 lsb of DL contain the first digit
    MOV BL,BH
    CALL HEX_KEYB           ;read second digit of seocnd number
    MOV DH,AL               ;4 lsb of DH contain the second digit

    PUSH DX
    PRINT BL
    PRINT BH
    POP DX

    MOV BL,DL
    MOV BH,DH
    ROL BL,4                ;4 msb of DL have the first digit
    ADD BL,BH               ;DL has the first number after addition of DH to it

    POP DX


    NEW_LINE                ;print new line

    PUSH BX
    PUSH DX                 ;print X+Y=
    PRINT 'X'
    PRINT '+'
    PRINT 'Y'
    PRINT '='
```

```
    POP DX

    PUSH DX                 ;adding the 2 numbers
    AND DH,0x00
    AND BH,0x00
    ADD DX,BX               ;store result of addition in DX
    PUSH AX
    MOV AX, DX              ;move result to AX
    CALL PRINT_DEC          ;print them in decimal form
    POP AX
    POP DX
    POP BX


    PUSH DX                 ;print X-Y=
    PRINT ' '
    PRINT 'X'
    PRINT '-'
    PRINT 'Y'
    PRINT '='
    POP DX


    PUSH BX
    PUSH DX
    AND DH,0x00
    CMP DL,BL               ;if the substraction returns a negative
    JAE GOOD_ENOUGH
    PUSH DX                 ;print a '-' and perform the opposite substraction
    PRINT '-'
    POP DX
    SUB BL,DL               ;opposite substraction done like this
    MOV DL,BL               ;store result in DL
    JMP FINAL
  GOOD_ENOUGH:              ;if the substraction would be positive then just do it
    SUB DL,BL
  FINAL:
    MOV AX,DX               ;print result in decimal form after storing it in AX
    CALL PRINT_DEC
    POP DX
    POP BX

  RET
MAIN ENDP


HEX_KEYB PROC NEAR          ;same as the one of other exercises
    PUSH DX
  DO:
    READ
    CMP AL,30H
    JL DO
    CMP AL,39H
    JG FLAG1
    MOV BH,AL
    SUB AL,30H
    JMP FLAG2
  FLAG1:
    CMP AL,'A'
```

```
    JL DO
    CMP AL,'F'
    JG DO
    MOV BH,AL                ;store hex representation of input
    SUB AL,37H
  FLAG2:
    POP DX
    RET
HEX_KEYB ENDP


PRINT_HEX PROC NEAR          ;print the hexadecimal number
    PUSH DX

    MOV CX,DX
    AND DX,0xF000
    ROL DH,4
    CMP DH,0x09
    JA LETTER_0
    ADD DH,30H
    JMP NEXT0
  LETTER_0:
    ADD DH,37H
    NEXT0:
    PRINT DH

    MOV DX,CX
    AND DX,0x0F00
    CMP DH,0x09
    JA LETTER_1
    ADD DH,30H
    JMP NEXT1
  LETTER_1:
    ADD DH,37H
  NEXT1:
    PRINT DH

    MOV DX,CX
    AND DX,0x00F0
    ROL DL,4
    CMP DL,0x09
    JA LETTER_2
    ADD DL,30H
    JMP NEXT2
  LETTER_2:
    ADD DL,37H
    NEXT2:
    PRINT DL

    MOV DX,CX
    AND DX,0x000F
    CMP DL,0x09
    JA LETTER_3
    ADD DL,30H
    JMP NEXT3
  LETTER_3:
    ADD DL,37H
  NEXT3:
    PRINT DL
```

```
      POP DX
   RET
PRINT_HEX ENDP

PRINT_DEC PROC NEAR
   MOV BL,10
   MOV CX,1                    ;decades counter
LOOP_10:
   DIV BL                      ;divide number by 10
   PUSH AX                     ;save decades
   CMP AL,0                    ;if there are no more decades then we have reached single digits
   JE PRINT_DIGITS_            ;the whole number into dec digits
   INC CX                      ;increase number of decades
   MOV AH,0
   JMP LOOP_10                 ;if we have not reached single digits I have to divide again so loop again
PRINT_DIGITS_:
   POP DX                      ;pop dec digit to be printed
   MOV DL,DH
   MOV DH,0                    ;DX = 00000000xxxxxxxx (ASCII of number to be printed)
   ADD DX,30H                  ;make ASCII code
   MOV AH,2
   INT 21H                     ;print
   LOOP PRINT_DIGITS_
   RET
ENDP PRINT_DEC


   END MAIN
```

ΑΣΚΗΣΗ 4η

Ο κώδικας μας σε assembly 8086 για την άσκηση 4 και στο αρχείο “4.asm”:

```
INCLUDE MACRO_INST.ASM
   .8086
   .MODEL SMALL
   .STACK 256

   .DATA
   TABLE DB 16 DUP(?)


   .CODE
   ASSUME DS:DATA

MAIN PROC FAR
   MOV AX,DATA
   MOV DS,AX

START:
   MOV DI,0                    ;initialize DI to 0
   MOV CX,0                    ; same for CX
```

```
READING:
   READ_IN
   CMP AL,13                ;compare AL with ASCII of enter
   JE END_PROG                      ;if enter is pressed end.
   CMP AL,'0'               ;numbers > 0 -> accept
   JL READING               ;else read again
   CMP AL,'9'               ;accept numbers < 9
   JNA ACCEPTED
   CMP AL,'A'               ;accepts letters between A
   JL READING
   CMP AL,'Z'               ;and Z
   JG READING               ;if not read again else accept

ACCEPTED:
   MOV [TABLE + DI],AL      ;insert in table if terms are fullfilled
   INC DI                   ;DI ++
   INC CL                   ;CL++
   CMP CL,16                ;if we reach 16 characters then print them
   JZ PRINT_OUT
   JMP READING

PRINT_OUT:
   MOV DI,0                 ;DI<-0
PRINT_LOOP:
   MOV AL,[TABLE + DI]      ;print chars until we have printed all of them.
   PRINT AL
   INC DI
   INC CH
   CMP CH,15
   JG YAPRINT               ;then go to PRINT2
   JMP PRINT_LOOP

YAPRINT:
   NEW_LINE                 ;print new line
   MOV DI,0
   MOV CH,0
YAPRINT_LOOP:               ;we iterate over the table and when we find a number we print it
   MOV AL,[TABLE + DI]      ;when we reach 16 chars(AL = 16) we have printed all the numbers
   CMP AL,3AH               ;and then we print small letters
   JL PRINT_NUM
   JMP CONTINUING
PRINT_NUM:
   PRINT AL
CONTINUING:
   INC DI                   ;DI++
   INC CH                   ;CH++
   CMP CH,15                ;compare with 15
   JG PRINT_LETTERS
   JMP YAPRINT_LOOP

PRINT_LETTERS:
   PRINT '-'                ;print a dash first
   MOV DI,0
   MOV CH,0
```

```
PRINT_LETTERS_LOOP:
   MOV AL,[TABLE + DI]        ;we run the table again and now we print only the letters
   CMP AL,41H
   JGE PRINT_A_LET
   JMP CONTINUING2
PRINT_A_LET:
   ADD AL,32                  ;we convert caps to regular
   PRINT AL                   ; and print them
CONTINUING2:
   INC DI                     ;DI++
   INC CH                     ;CH++
   CMP CH,15                  ;if we reach the end of the table
   JG BEGINNING
   JMP PRINT_LETTERS_LOOP

BEGINNING:                    ;then we have finished and we change line and go to start
   NEW_LINE
   JMP START

END_PROG:                     ;program ends through here
   EXIT
   MAIN ENDP
```

ΑΣΚΗΣΗ 5η

- Ο κώδικας που φτιάξαμε με τις μακροεντολές, στο αρχείο "MACROS2.asm":

```
READ MACRO
    MOV AH, 8
    INT 21H
ENDM

PRINT MACRO CHAR
     MOV DL, CHAR
     MOV AH, 2
     INT 21H
ENDM

PRINT_STR MACRO STRING
   MOV DX, OFFSET STRING
   MOV AH,9
   INT 21H
ENDM

EXIT MACRO
    MOV AX, 4C00H
    INT 21H
ENDM
```

```
NEW_LINE MACRO
   PUSH DX
   PUSH AX
   MOV DX,13
   MOV AH,2
   INT 21H
   MOV DX,10
   MOV AH,2
   INT 21H
   POP AX
   POP DX
ENDM

PRINT_DEC MACRO
   ADD DL, 30H
   MOV AH,2
   INT 21H
ENDM

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

- Ο κώδικας που φτιάξαμε για το κυρίως πρόγραμμα, στο αρχείο "5.asm":

```
 INCLUDE MACROS5.ASM
   .8086
   .MODEL SMALL
   .STACK 256

.DATA
   MSG1 DB 0AH,0DH,'START(Y,N):$'
   MSG2 DB 0AH,0DH,'ERROR$'

.CODE
   ASSUME CS : CODE_SEG, DS : DATA_SEG

MAIN PROC FAR
   MOV AX, DATA
   MOV DS, AX


START:
   PUSH DX
   PRINT_STR MSG1
   POP DX

WAIT1:

   CALL HEX_KEYB1                ;wait until you read Yor N , if you read N stop
   CMP AL, 'N'
```

```
    JE MY_EXIT
    CMP AL, 'Y'
    JNE WAIT1
    MOV AL ,0
    CALL HEX_KEYB2              ;1st digit
    CMP AL, 'N'
    JE MY_EXIT
    MOV DH,AL                   ; in DH

    CALL HEX_KEYB2              ;2nd digit
    CMP AL, 'N'
    JE MY_EXIT
    MOV DL,AL                   ; in DL

    CALL HEX_KEYB2              ;3nd digit
    CMP AL, 'N'
    JE MY_EXIT
    MOV BL,AL                   ; in BL

    ROL DL,4                    ; shift left for 4


    ADD DL,BL

    CMP DX ,3E8H                ; if number>999,9 , print error
    JL INPUT                    ; else go to INPUT

    PUSH DX
    PRINT_STR MSG2              ; print the error message
    NEW_LINE
    POP DX
    JMP START

INPUT:                          ; number in DX -> AX
    MOV AX,DX
    PUSH DX
    NEW_LINE                    ; BECAUSE NEWLINE USES DX
    POP DX
    CMP AX,500D                 ; 0<number<500 -> flag1
    JNA FLAG1

FLAG2:
    CMP AX,700D                 ; 700=<number<1000 -> FLAG3
    JNA FLAG4                   ; 500=<number<700 -> flag4

FLAG3:
    SUB AX , 700D               ; number -700
    MOV DX, 4095D
    MUL DX
    MOV CX, 300D
    DIV CX
    ADD AX , 36855D
    MOV CX ,0                   ; (1,8+0,2/300*(number -700))      *4095*10/2
```

```
DIGIT1:
   MOV DX, 0
   MOV BX, 10D
   DIV BX
   PUSH DX
   INC CX
   CMP AX, 0
   JNE DIGIT1
   DEC CX
   JMP MY_PRINT

FLAG1:
   MOV DX, 4095D
   MUL DX
   MOV CX, 100D
   DIV CX
   MOV CX, 0                    ; 1/500 *number *4095*10/2

DIGIT2:
   MOV DX, 0
   MOV BX, 10D
   DIV BX
   PUSH DX
   INC CX
   CMP AX, 0
   JNE DIGIT2
   DEC CX
   JMP MY_PRINT

FLAG4:
   SUB AX, 500D                 ; ( 1+0,8/200*(number -500)) *4095*10/2
   MOV DX, 4095D
   MUL DX
   MOV CX, 50D
   DIV CX
   ADD AX, 20475D
   MOV CX, 0
DIGIT3:
   MOV DX, 0
   MOV BX, 10D
   DIV BX
   PUSH DX
   INC CX
   CMP AX, 0
   JNE DIGIT3
   DEC CX
   JMP MY_PRINT

MY_PRINT:
   POP DX
   PUSH DX
   PRINT_DEC
   POP DX                       ;print the digits of result 1 by 1 except the last digit
   LOOP MY_PRINT
```

```
    PUSH DX                     ;print .
    PRINT '.'
    POP DX
    POP DX                      ;then print the last digit
    PUSH DX
    PRINT_DEC
    NEW_LINE
    POP DX
    JMP START


MY_EXIT:
    EXIT

   MAIN ENDP

HEX_KEYB1 PROC NEAR
    PUSH DX
ARXH1:
    READ
    CMP AL,30H
    JL ARXH1
    CMP AL,39H
    JG SYMBOL1          ;If input>=9, GO TO SYMBOL1
    SUB AL,30H          ; IF NUMBER, sub 30
    JMP END1
SYMBOL1:
    CMP AL,'Y'
    JE END1
        CMP AL,'N'
    JE END1
    CMP AL,'A'
    JL ARXH1
    CMP AL,'F'
    JG ARXH1
    SUB AL,37H          ; A<=input<=F sub 37
END1:
    POP DX
    RET
HEX_KEYB1 ENDP

HEX_KEYB2 PROC NEAR
    PUSH DX
ARXH2:
    READ
    CMP AL,30H
    JL ARXH2
    CMP AL,39H
    JG SYMBOL2          ;If input>=9, GO TO SYMBOL2
    SUB AL,30H          ; IF NUMBER, sub 30
    JMP END2
SYMBOL2:
    CMP AL,'N'
    JE END2
```

```
   CMP AL,'A'
   JL ARXH2
   CMP AL,'F'
   JG ARXH2
   SUB AL,37H          ; A<=input<=F sub 37
END2:
   POP DX
   RET
HEX_KEYB2 ENDP
```