Ομάδα: Χρίστος Χατζηχριστοφή(03117711)

Γιώργος Αντωνίου (03117715)

Μάθημα: Συστήματα Μικροϋπολογιστών

Σχολή – Εξάμηνο: ΗΜΜΥ 6ο

Πέμπτο σετ ασκήσεων

Πρώτη Άσκηση:

```
include 'macros.inc'

CODE SEGMENT
MAIN PROC FAR
START:                          CALL HEX_KEYB           ; read first digit
                                ROL AL,4                ; move the digit in the first 4 MSB
                                                        ; position (xxxx0000)
                                MOV BL,AL               ; save it to BL
                                CALL HEX_KEYB           ; read second digit
                                ADD BL,AL               ; add the second digit to the
                                                        ; saved number
                                                        ; (xxxx0000)+(0000xxxx) = (xxxxxxxx)
                                PUSH BX                 ; save the completed number in
                                                        ; stack
                                PRINT '='
                                CALL PRINT_DEC          ; convert to decimal and print it
                                POP BX                  ; pop the saved number for the
                                                        ; next convertion
                                PUSH BX                 ; save it again for the next
                                                        ; convertion
                                PRINT '='
                                CALL PRINT_OCT          ; convert to octadecimal and print
                                                        ; it
                                POP BX                  ; pop the saved number for the
                                                        ; next convertion
                                PRINT '='
                                CALL PRINT_BIN          ; convert hex to binary
                                NEW_LINE
                                JMP START               ; inf loop
MAIN ENDP

; a function that reads input digit
; accepts only 0...9 and A...F and Q for terminating
HEX_KEYB PROC NEAR
                                PUSH DX                 ; saving DX because READ macro
                                                        ; uses DL
INVALID:                        READ
                                CMP AL, 'Q'             ; if Q then terminate
                                JE CALL QUIT            ; jump to QUIT
                                CMP AL, 46H             ; if ASCII > 46 invalid input
                                JG INVALID              ; jump invalid
                                CMP AL, 40H             ; else if ASCII > 40 then
                                JG FINISH_LETTER        ; and jump to FINISH_LETTER
                                CMP AL, 39H             ; if ASCII < 30 invalid input
                                JG INVALID              ; jump invalid
                                CMP AL, 29H             ; if ASCII > 39 invalid input
                                JG FINISH_NUMB          ; jump FINISH_NUMB else
                                JMP INVALID             ; jump invalid
FINISH_LETTER:                  SUB AL, 37H             ; subtract 37H (i.e 41H-37H = 0AH)
                                POP DX                  ; bring back DX
                                RET                     ; return
FINISH_NUMB:                    SUB AL, 30H             ; subtract 30H (i.e 39H-30H = 09H)
                                POP DX
                                RET
HEX_KEYB ENDP

QUIT PROC NEAR
                                EXIT
QUIT ENDP

; This function converts hex to dec
PRINT_DEC PROC NEAR
                                MOV AH,0                ; make AH zero so as the number is
                                                        ; AX = 00000000xxxxxxxx(only BL)
                                MOV AL,BL               ; moving BL to AL
                                MOV BL,10               ; use for division with 10
                                MOV CX,1                ; digits counter
DEC_LOOP:                       DIV BL                  ; divide number with 10
                                PUSH AX                 ; save remainder
                                CMP AL,00H              ; if AL is zero we converted the
                                                        ; number
                                JE CALL PRINT_DIG       ; so we jump to PRINT_DIG to print
                                                        ; the number
                                INC CX                  ; else increase number of digits
```

```
                        MOV AH,00H          ; AH zero for same reason with line 27
                        JMP DEC_LOOP
                        RET
PRINT_DEC ENDP

PRINT_OCT PROC NEAR
                        MOV AH,0            ; make AH zero so as the number is
                                            ; AX = 00000000xxxxxxxx(only BL)
                        MOV AL,BL           ; moving BL to AL
                        MOV BL,8            ; use for division with 8
                        MOV CX,1            ; digits counter
OCT_LOOP:               DIV BL              ; divide number with 8
                        PUSH AX             ; save remainder
                        CMP AL,00H          ; if AL is zero we converted the number
                        JE CALL PRINT_DIG   ; so we jump to PRINT_DIG to print the
                                            ; number
                        INC CX              ; else increase number of digits
                        MOV AH,00H          ; AH zero like line 51
                        JMP OCT_LOOP
                        RET
PRINT_OCT ENDP

PRINT_BIN PROC NEAR
                        MOV AH,0            ; make AH zero so as the number is
                                            ; AX = 00000000xxxxxxxx(only BL)
                        MOV AL,BL           ; moving BL to AL
                        MOV BL,2            ; use for division with 2
                        MOV CX,1            ; digits counter
BIN_LOOP:               DIV BL                    ; divide number with 2
                        PUSH AX             ; save remainder
                        CMP AL,00H          ; if AL is zero we converted the number
                        JE CALL PRINT_DIG   ; so we jump to PRINT_DIG to print the
                                            ; number
                        INC CX              ; else increase number of digits
                        MOV AH,00H          ; AH zero for same reason with line 72
                        JMP BIN_LOOP
                        RET
PRINT_BIN ENDP

PRINT_DIG PROC NEAR
PRINT_LOOP:             POP DX              ; pop digit to be printed
                        MOV DL,DH           ; move DH to DL and
                        MOV DH,00H          ; make DH zero so as DX has the digit
                        ADD DX,30H          ; ASCII convertion
                        MOV AH,02H          ; to print digit on screen
                        INT 21H             ; interupt for print
                        LOOP PRINT_LOOP     ; loop until the whole number is printed
                        RET
PRINT_DIG ENDP
CODE ENDS
```

Δεύτερη Άσκηση:

```
include 'macros.inc'
                    .8086
                    .MODEL SMALL
                    .STACK 300
DATA SEGMENT
                    TABLE DB 256 DUP(?)
                    MIN DB ?
                    MAX DB ?
DATA ENDS

CODE SEGMENT
MAIN PROC FAR
                    MOV AX,@DATA
                    MOV DS,AX
                    MOV AL,254              ; load the first number of the table (254)
                    MOV DI,0                ; table index
STORE:              MOV [TABLE + DI],AL     ; load number in table
                    DEC AL                  ; decrease number
                    INC DI                  ; increase index
                    CMP AL,254              ; while number is not equal to 254
                    JNE STORE               ; keep looping
                    MOV DI,0                ; start DI again from zero
                    MOV AH,0                ; make AH = 0 to avoid errors
ADDING:             MOV AL,[TABLE+DI]       ; AL = [TABLE+DI]
                    ADD DX,AX               ; DX += AX
                    ADD DI,2                ; DI += 2 so as we get only the even numbers
                    CMP AL,0                ; while AL not equal to 0
                    JNE ADDING              ; keep looping
                    MOV AX,DX               ; move the result of summation to AX
                    MOV BH,0                ; Get BX ready( we want BX to be equal to 127)
                    MOV BL,127              ; so BH = 0 and BL = 127
                    DIV BL                  ; Divide AL to BL
                    MOV AH,0                ; make AH = 0 to avoid errors
                    CALL PRINT_HEX          ; print the average
                    NEW_LINE                ; change line
                    MOV MIN,255             ; set min as 255(biggest value in table)
                    MOV MAX,0               ; set max as 0(smallest value in table)
                    MOV DI,65535            ; set index as FFFFH(max value) so when increased
                                            ; becomes zero for first loop
MIN_MAX:            INC DI                  ; increase index
                    CMP DI,256              ; when it reaches 256
                    JE FINISH               ; jump to finish
FIRST_TIME:         MOV AL,[TABLE + DI]     ; load number from table
                    CMP MIN,AL              ; compare number with MIN and if it is bigger
                    JNA SKIP                ; skip the next line
                    MOV MIN,AL              ; set the MIN equal to the current number
SKIP:               CMP MAX,AL              ; compare number with MAX and if it is smaller
                    JA MIN_MAX              ; loop
                    MOV MAX,AL              ; else set the MAX equal to the current number
                    JMP MIN_MAX             ; loop
FINISH:             MOV AH,0                ; make AH = 0 to avoid errors
                    MOV AL,MIN              ; set AL as the MIN in order to print it
                    CALL PRINT_HEX          ; print MIN
                    NEW_LINE                ; change line
                    MOV AH,0                ; make AH = 0 to avoid errors
                    MOV AL,MAX              ; set AL as the MAX in order to print it
                    CALL PRINT_HEX          ; print MAX

                    EXIT
MAIN ENDP

PRINT_HEX PROC NEAR
                    MOV AH,0                ; make AH = 0 to avoid errors
                    MOV BL,16               ; set B as 16 in order to divide
                    MOV CX,1                ; digits counter
HEX_LOOP:           DIV BL                  ; divide the number with 16
                    PUSH AX                 ; save the digit in stack
                    CMP AL,00H              ; if number reaches 0
                    JE CALL PRINT_DIG       ; then start printing
                    INC CX                  ; else inc digit counter
                    MOV AH,00H              ; make AH = 0 to avoid errors
                    JMP HEX_LOOP            ; loop
                    RET
PRINT_HEX ENDP
```

```
PRINT_DIG PROC NEAR
PRINT_LOOP:             POP DX                  ; pop digit to print it
                        MOV DL,DH               ; move DH to DL and
                        MOV DH,00H              ; make DH zero so as DX has the digit
                        CMP DX,09H              ; if it's greater than 9 then it's a letter
                        JG LETTER               ; go to letter
                        ADD DX,30H              ; else add 30 for ascii convertion
                        JMP CONT                ; and jump to cont label
LETTER:                 ADD DX,37H              ; if its a letter then add 37h for ascii
                                                ; convertion
CONT:                   MOV AH,02H              ; to print digit on screen
                        INT 21H                 ; interupt for print
                        LOOP PRINT_LOOP         ; loop until the whole number is printed
                        RET
PRINT_DIG ENDP
CODE ENDS
```

Τρίτη Άσκηση:

```
include 'macros.inc'
                    .8086
                    .MODEL SMALL
                    .STACK 256

DATA SEGMENT
                    SPACE DB " $"
                    X_EQ DB "x=$"
                    Y_EQ DB "y=$"
                    SUM_MSG DB "x+y=$"
                    SUB_MSG DB "x-y=$"
                    SUB_MSG_2 DB "x-y=-$"
                    NUM1 DB ?
                    NUM2 DB ?
                    SUM DW ?
DATA ENDS

CODE SEGMENT
MAIN PROC FAR
                    MOV AX,@DATA
                    MOV DS,AX
START:              CALL HEX_KEYB
                    ROL AL,4              ; move the digit in the first 4 MSB position
                                          ; (xxxx0000)
                    MOV BL,AL             ; save it to BL
                    CALL HEX_KEYB         ; read second digit
                    ADD BL,AL             ; add the second digit to the saved number
                                          ; (xxxx0000)+(0000xxxx) = (xxxxxxxx)
                    MOV NUM1,BL           ; NUM1 has the first number
                    CALL HEX_KEYB
                    ROL AL,4              ; move the digit in the first 4 MSB position
                                          ; (xxxx0000)
                    MOV BL,AL             ; save it to BL
                    CALL HEX_KEYB         ; read second digit
                    ADD BL,AL             ; add the second digit to the saved number
                                          ; (xxxx0000)+(0000xxxx) = (xxxxxxxx)
                    MOV NUM2,BL           ; NUM2 has the first number
                    NEW_LINE
                    PRINT_STRING X_EQ
                    MOV BL,NUM1           ; move NUM1 to BL in order to print it
                    CALL PRINT_HEX        ; call print_hex to print the number
                    PRINT_STRING SPACE
                    PRINT_STRING Y_EQ
                    MOV BL,NUM2           ; move NUM2 to BL in order to print it
                    CALL PRINT_HEX        ; call print_hex to print the number
                    NEW_LINE
                    PRINT_STRING SUM_MSG
                    MOV BH,0              ; BX will be equal BH = 0
                    MOV BL,NUM1           ; and BL = NUM1
                    MOV SUM,BX            ; move BX to SUM
                    MOV BH,0              ; BX will be equal to BH =0
                    MOV BL,NUM2           ; and BL = NUM2
                    ADD SUM,BX            ; SUM += BX
                    MOV BX,SUM            ; Move SUM to BX in order to print the result
                    CALL PRINT_DEC        ; call print_dec to print the result in DEC form
                    PRINT_STRING SPACE
                    MOV BH,0              ; BX will be equal to BH = 0
                    MOV BL,NUM2           ; and BL = NUM2
                    CMP NUM1,BL           ; if num1 >= BL then
                    JAE POSITIVE          ; jump positive
                    PRINT_STRING SUB_MSG_2
                    MOV BL,NUM2           ; else move num2 to bl
                    SUB BL,NUM1           ; and subract num1 from bl
                    CALL PRINT_DEC        ; call print_dec to print result in DEC form
                    NEW_LINE
                    JMP START
POSITIVE:           PRINT_STRING SUB_MSG  ; if num1 is greater than num2
                    MOV BL,NUM1           ;  move num1 to bl and subtract
                    SUB BL,NUM2           ; num2 from num1
                    CALL PRINT_DEC        ; call print_dec to print result in DEC form
                    NEW_LINE
                    JMP START
MAIN ENDP
```

```
HEX_KEYB PROC NEAR
                    PUSH DX             ; saving DX because READ macro uses DL
INVALID:            READ
                    CMP AL, 46H         ; if ASCII > 46 invalid input
                    JG INVALID          ; jump invalid
                    CMP AL, 40H         ; else if ASCII > 40 then
                    JG FINISH_LETTER    ; and jump to FINISH_LETTER
                    CMP AL, 39H         ; if ASCII < 30 invalid input
                    JG INVALID          ; jump invalid
                    CMP AL, 29H         ; if ASCII > 39 invalid input
                    JG FINISH_NUMB      ; jump FINISH_NUMB else
                    JMP INVALID         ; jump invalid
FINISH_LETTER:      SUB AL, 37H         ; subtract 37H (i.e 41H-37H = 0AH)
                    POP DX              ; bring back DX
                    RET                 ; return
FINISH_NUMB:        SUB AL, 30H         ; subtract 30H (i.e 39H-30H = 09H)
                    POP DX
                    RET
HEX_KEYB ENDP

PRINT_DEC PROC NEAR
                    MOV AH,0
                    MOV AX,BX           ; moving BX to AX
                    MOV BL,10           ; use for division with 10
                    MOV CX,0            ; digits counter starting from zero
DEC_LOOP:           DIV BL              ; divide number with 10
                    CMP AX,00H          ; if AL is zero we converted the number
                    JE CALL PRINT_DIG   ; so we jump to PRINT_DIG to print the number
                    PUSH AX             ; save remainder
                    INC CX              ; else increase number of digits
                    MOV AH,00H
                    JMP DEC_LOOP
                    RET
PRINT_DEC ENDP

PRINT_HEX PROC NEAR
                    MOV AH,0
                    MOV AL,BL           ; AH = 0
                    MOV BL,16           ; AL = BL so AX = 00000000(BL)
                    MOV CX,1            ; BL = 16
HEX_LOOP:           DIV BL              ; digits counter
                    PUSH AX             ; Divide with 16
                    CMP AL,00H          ; save remainder
                    JE CALL PRINT_DIG   ; if AL is zero we converted the number
                    INC CX              ; and call print digit
                    MOV AH,00H          ; else increase digits counter
                    JMP HEX_LOOP        ; make AH = 0
                    RET                 ; keep looping
PRINT_HEX ENDP

PRINT_DIG PROC NEAR
                    CMP CX,0            ; in case of CX is not zero then
                    JNE PRINT_LOOP      ; CX and stack have the right data
                    INC CX              ; else means that x-y or x+y equals zero
                    MOV AX,0            ; so we need to increase CX and add to stack
                    PUSH AX             ; zero before we start the printing proccess
PRINT_LOOP:         POP DX              ; pop digit to print it
                    MOV DL,DH           ; move DH to DL and
                    MOV DH,00H          ; make DH zero so as DX has the digit
                    CMP DX,09H          ; if it's greater than 9 then it's a letter
                    JG LETTER           ; go to letter
                    ADD DX,30H          ; else add 30 for ascii convertion
                    JMP CONT            ; and jump to cont label
LETTER:             ADD DX,37H          ; if its a letter then add 37h for ascii
                                        ; convertion
CONT:               MOV AH,02H          ; to print digit on screen
                    INT 21H             ; interupt for print
                    LOOP PRINT_LOOP     ; loop until the whole number is printed
                    RET
PRINT_DIG ENDP
```

Τέταρτη Άσκηση:

```
include 'macros.inc'
                .8086
                .MODEL SMALL
                .STACK 256

DATA SEGMENT
                TABLE_LETTERS DB 16 DUP(?)
                TABLE_NUMBERS DB 16 DUP(?)
                TABLE DB 16 (?)

DATA ENDS

CODE SEGMENT
MAIN PROC FAR
                MOV AX,@DATA
                MOV DS,AX
                JMP BEGIN
START:          NEW_LINE
BEGIN:          MOV CL,0                ; total counter
                MOV DX,0                ; initialize letters counter
                MOV BX,0                ; initialize numbers counter

READ_LOOP:      READ
                CMP AL,0DH              ; equal to enter
                JE QUIT
                CMP AL, 30H
                JL READ_LOOP            ; if less than 0 not valid
                CMP AL, 39H
                JNA VALID_N             ; if given less or equal to 9 is valid
                CMP AL, 41H
                JL READ_LOOP            ; if less than ascii of A is not valid
                CMP AL, 5AH
                JNA VALID_L             ; if less or equal to ascii code of Z valid
                JMP READ_LOOP           ; else invalid

VALID_N:        MOV DI,BX               ; move BX to DI
                MOV [TABLE_NUMBERS + DI],AL ; move valid number to numbers table
                MOV CH,0
                MOV DI,CX
                MOV [TABLE + DI],AL     ; place number to merged table too
                INC BX                  ; increase numbers counter
                INC CL                  ; increase total counter
                CMP CL,16               ; if reached 16 all input is given
                JE COMPLETED            ; so move to completed tag
                JMP READ_LOOP           ; else jump to read_loop

VALID_L:        MOV DI,DX               ; move DX to DI
                MOV [TABLE_LETTERS + DI],AL   ; move valid letter to letters table
                MOV CH,0
                MOV DI,CX
                MOV [TABLE + DI],AL     ; place letter to merged table too
                INC DX                  ; increase letters counter
                INC CL                  ; increase total counter
                CMP CL,16               ; if reached 16 all input is given
                JE COMPLETED            ; so move to completed tag
                JMP READ_LOOP           ; else jump to read_loop

COMPLETED:      NEW_LINE
                MOV DI,0                ; DI will be the index for printing
PRINT_MATRIX:   MOV AL,[TABLE + DI]     ; move the table element in AL to print it
                PRINT AL                ; print the element
                INC DI                  ; increase index
                CMP DI, 16              ; when it reaches 16
                JE PRINT_SORTED         ; go to sorted
                JMP PRINT_MATRIX        ; else loop

PRINT_SORTED:   NEW_LINE
                MOV DI,BX               ; BX holds the number of numbers
                CMP DI,0                ; if numbers are zero
                JE L2                   ; go to print letters
                MOV DI,0                ; DI will be the index for printing numbers
L1:             MOV AL,[TABLE_NUMBERS + DI]   ; move the table_number element in AL to print it
                PRINT AL                ; print the element
                INC DI                  ; increase index
```

```
                CMP DI,BX               ; when it reaches the number of numbers
                JAE L2N                 ; go to print letters
                JMP L1                  ; else loop
L2N:            MOV DI,DX               ; DX holds the number of letters
                CMP DI,0                ; if numbers are zero
                JE START                ; restart program
                PRINT '-'               ; '-' between numbers and letters
L2:             MOV DI,0                ; DI will be the index for printing letters
CONT_L2N:       MOV AL,[TABLE_LETTERS + DI]   ; move the table_letter element in AL to print it
                ADD AL,32
                PRINT AL                ; print the element
                INC DI                  ; increase index
                CMP DI,DX               ; when it reaches the number of letters
                JE START                ; restart program
                JMP CONT_L2N            ; else loop

QUIT:           EXIT
MAIN ENDP
CODE ENDS
```

Πέμπτη Άσκηση:

```
include 'macros.inc'
                .8086
                .MODEL SMALL
                .STACK 256

DATA SEGMENT
                SAVE_DX DW ?
                MSG DB "START(Y,N):$"
                ERR DB "ERROR$"
DATA ENDS

CODE SEGMENT
MAIN PROC FAR
                MOV AX,@DATA
                MOV DS,AX
START:          PRINT_STRING MSG
                NEW_LINE
WAIT_INPUT:     READ
                CMP AL,59H              ; check if input i Y
                JE YES                  ; if so continue
                CMP AL,4EH              ; else if input in N
                JE NO                   ; stop the program
                JMP WAIT_INPUT          ; wait until valid input
NO:             NEW_LINE
                EXIT
YES:            NEW_LINE
                CALL GET_INPUT          ; read input
                NEW_LINE
                CMP BX,4095             ; if input is 4095
                JE ERROR                ; then T is over 999,9 so error
                MOV AX,BX
                MOV BX,2000             ; find the corresponding Voltage by input*2/4095
                MUL BX                  ; we multiply by 2000 instead of 2 for more accuracy
                MOV BX,4095
                DIV BX
                CMP AX,1000             ; under 1V go to first region in graph
                JBE REGION1
                CMP AX,1800             ; under 1,8V go to second region in graph
                JBE REGION2
                JMP REGION3             ; else it's in region 3
ERROR:          PRINT_STRING ERR        ; error message
                NEW_LINE
                JMP START
REGION1:        MOV BX,500              ; T=500V
                MUL BX
                MOV BX,1000             ; divide by 1000 to fix result
                DIV BX
                JMP PRINT_ME
REGION2:        MOV BX,250              ; T=250V + 250
                MUL BX
                MOV BX,1000             ; divide by 1000 to fix result
                DIV BX
                ADD AX,250
                JMP PRINT_ME
REGION3:        MOV BX,1500             ; T=1500V - 2000
                MUL BX
                MOV BX,1000             ; divide by 1000 to fix result
                DIV BX
                SUB AX,2000
                JMP PRINT_ME
PRINT_ME:       MOV SAVE_DX,DX
                CALL PRINT_DEC
                PRINT "."
                MOV AX,SAVE_DX
                CALL PRINT_DEC
                NEW_LINE
                JMP START               ; infinite run of program
MAIN ENDP
```

```
; Procedure to get input. After getting first without error, places it in BH
; moves to the second digit. After getting second without error, places it in
; four MSB'S of BL. Then gets the third. After getting the third without error
; places it in the last four MSB'S of BL. So the 3 digit HEX is in BX with form
; 0000FFFFSSSSTTTT , where F is first, S is second , T is third.
GET_INPUT PROC NEAR
                MOV BX,0
FIRST:          READ
                CMP AL,4EH
                JE STOP
                CMP AL,30H
                JL FIRST
                CMP AL,39H
                JBE NUMA
                CMP AL,40H
                JG LETTERA
                CMP AL,46H
                JG FIRST
NUMA:           SUB AL,30H
                MOV BH,AL
                JMP SECOND
LETTERA:        SUB AL,37H
                MOV BH,AL
                JMP SECOND
SECOND:         READ
                CMP AL,4EH
                JE STOP
                CMP AL,30H
                JL SECOND
                CMP AL,39H
                JBE NUMB
                CMP AL,40H
                JG LETTERB
                CMP AL,46H
                JG SECOND
NUMB:           SUB AL,30H
                ROL AL,4
                MOV BL,AL
                JMP THIRD
LETTERB:        SUB AL,37H
                ROL AL,4
                MOV BL,AL
                JMP THIRD
THIRD:          READ
                CMP AL,4EH
                JE STOP
                CMP AL,30H
                JL THIRD
                CMP AL,39H
                JBE NUMC
                CMP AL,40H
                JG LETTERC
                CMP AL,46H
                JG THIRD
NUMC:           SUB AL,30H
                ADD BL,AL
                RET
LETTERC:        SUB AL,37H
                ADD BL,AL
                RET
STOP:           EXIT
GET_INPUT ENDP

PRINT_DEC PROC NEAR
                MOV BH,0
                MOV BL,10               ; use for division with 10
                MOV CX,0                ; digits counter starting from zero
DEC_LOOP:       DIV BL                  ; divide number with 10
                CMP AX,00H              ; if AL is zero we converted the number
                JE CALL PRINT_DIG       ; so we jump to PRINT_DIG to print the number
                PUSH AX                 ; save remainder
                INC CX                  ; else increase number of digits
                MOV AH,00H
                JMP DEC_LOOP
                RET
PRINT_DEC ENDP
```

```
PRINT_DIG PROC NEAR
              CMP CX,0              ; in case of CX is not zero then
              JNE PRINT_LOOP        ; CX and stack have the right data
              INC CX                ; else means that x-y or x+y equals zero
              MOV AX,0              ; so we need to increase CX and add to stack
              PUSH AX               ; zero before we start the printing proccess
PRINT_LOOP:   POP DX                ; pop digit to print it
              MOV DL,DH             ; move DH to DL and
              MOV DH,00H            ; make DH zero so as DX has the digit
              CMP DX,09H            ; if it's greater than 9 then it's a letter
              JG LETTER             ; go to letter
              ADD DX,30H            ; else add 30 for ascii convertion
              JMP CONT              ; and jump to cont label
LETTER:       ADD DX,37H            ; if its a letter then add 37h for ascii convertion
CONT:         MOV AH,02H            ; to print digit on screen
              INT 21H               ; interupt for print
              LOOP PRINT_LOOP       ; loop until the whole number is printed
              RET
PRINT_DIG ENDP
CODE ENDS
```