# Συστήματα Μικρουπολογιστών - 5ή Σειρά Ασκήσεων

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Ακαδημαϊκό έτος : 2018 – 2019 Εξάμηνο : 6ο
Μέλη ομάδας : Βόσινας Κωνσταντίνος AM : 03116435
Ανδριόπουλος Κωνσταντίνος AM : 03116023

## ***Άσκηση 1η***

```
DATA_SEG DATA SEGMENT
    TABLE 128 DUP(?)
    SPACE DB " "
DATA_SEG ENDS

CODE_SEG CODE SEGMENT
    ASSUME CS:CODE,DS:DATA
MAIN PROC FAR
    MOV AX,DATA_SEG
    MOV DS,AX
    MOV CX, 128 ;First store all numbers
    LEA BX,TABLE    ;Set counter and fetch address of table
LOOP1:
    MOV [BX],CX
    INC BX      ;Store numbers and increment BX
    LOOP LOOP1
    MOV CX, 128
    LEA SI,TABLE
    MOV AL,129  ;AL contains min, initially set at 129
    MOV AH,0    ;AH contains max, initially set at 0
    MOV DX,0    ;D contains sum to be printed
LOOP2:
    MOV BX,[SI] ;Fetch a number
    INC SI
    TEST BL,01  ;Check if number is even
    JNZ EVEN
    ADD DL,BL   ;Add to sum if it is odd

EVEN:   CMP AL,BL   ;Check if num<min
        JGE  SKIP1  ;If not, skip
    MOV AL,BL   ;Else change AL
SKIP1:
    CMP BL,AH   ;Similarily for max
        JGE  SKIP2
    MOV AH,BL

SKIP2:  LOOP LOOP2  ;Loop for all numbers
    SAR DX,6    ;Shift right six times, DX = DX/
    PRINT_HEX   ;Print sum
    PRINT_STR SPACE
    MOV DL, AL  ;Print min
    PRINT_HEX
    PRINT_STR SPACE
    MOV DL,AH   ;Print max

    PRINT_HEX
MAIN ENDP
```

```
PRINT_HEX PROC NEAR ;Based on process from the book
    CMP DL,9
    JLE ADDR3
    ADD DL,37H
    JMP ADDR4
ADDR3:
    ADD DL,30H
ADDR4
    PRINT DL
    RET
PRINT_HEX ENDP

CODE_SEG ENDS
    END MAIN
```

## ***Άσκηση 2η***

```
DATA SEGMENT
    MSG1 DB "Z=$"
    MSG2 DB " W=$"
    MSG3 DB 0AH,ODH,"Z+W=$"
    MSG4 DB "  Z-W=$"
ENDS

CODE SEGMENT
    ASSUME CS:CODE,DS:DATA

MAIN PROC FAR
    MOV AX,DATA_SEG
    MOV DS,AX
ADR0:
    PRINT_STR MSG1
    CALL HEX_KEYB   ;Read first number, first digit
    MOV BL,16D
    MUL BL
    MOV BL,AL       ;Multiply by 16, so it is MSB, store in BL
    CALL HEX_KEYB   ;Read second digit LSB
    ADD BL,AL       ;DL now contains full hex value of Z

ADR1:
    PRINT_STR MSG2
    CALL HEX_KEYB   ;Read MSB of second number
    MOV BH, 16D
    MUL BH
    MOV BH,AL       ;Same as before, now W in BH
    CALL HEX_KEYB
    ADD BH,AL
ADR2:
    MOV AL,BL       ;First, add result
    ADD AL,BH
    MOV CX,4        ;LOOP 4 times, each print a hex number
    PRINT_STR   MSG3
ADR3:
    ROL AX,4        ;Setting 4 MSB's to print
    MOV DL,AL
    AND DL,0FH      ;Mask first 4 bits, print routine uses D reg
    PUSH AX         ;Save A
    CALL PRINT_HEX
    POP AX
    LOOP ADR3
ADR4:
    MOV AL,BL
    SUB AL,BH       ;Then, calculate Z-W (BL-BH)
    PRINT_STR MSG4
    MOV CX,4
ADR5:
    ROL AX,4        ;Same as before
    MOV DL,AL
    AND DL,0FH
    PUSH AX
    CALL PRINT_HEX
    POP AX
    LOOP ADR3
    JMP ADR0        ;Start over
MAIN ENDP
```

```
HEX_KEYB PROC NEAR  ;modified, source from book page 378
    PUSH DX
IGNORE:
    READ
    CMP AL,30H
    JL IGNORE
    CMP AL,39H
    JG ADDR1
    PUSH AX
    PRINT AL
    POP AX
    SUB AL,30H
    JMP ADDR2
ADDR1:
    CMP AL,'A'
    JL IGNORE
    CMP AL,'F'
    JG IGNORE
    PUSH AX
    PRINT AL
    POP AX
    SUB AL,37H
ADDR2:
    POP DX
    RET
HEX_KEYB ENDP


PRINT_HEX PROC NEAR
    CMP DL,9
    JLE ADDR3
    ADD DL,37H
    JMP ADDR4
ADDR3:
    ADD DL,30H
ADDR4
    PRINT DL
    RET
PRINT_HEX ENDP

CODE_SEG ENDS
    END MAIN
```

## ***Άσκηση 3η***

```
DATA SEGMENT
    MSG1 DB "Enter first digit=$"
    MSG2 DB 0AH,ODH,"Enter second digit=$"
    EQUALS DB " = $"
    SPACE DB 0AH,0DH
ENDS

CODE SEGMENT
    ASSUME CS:CODE,DS:DATA

MAIN PROC FAR
    MOV AX,DATA_SEG
    MOV DS,AX
ADR0:
    PRINT_STR MSG1
    CALL HEX_KEYB   ;Read first number, first digit
    CMP AL,'T'
    JE QUIT
    MOV BL,16D
    MUL BL
    MOV BL,AL       ;Multiply by 16, so it is MSB, store in BL
    CALL HEX_KEYB   ;Read second digit LSB
    ADD BL,AL       ;DL now contains full hex value of Z
PRINT_DIGITS:
    PRINT_HEX
    PRINT_STR EQUALS
    PRINT_DEC
    PRINT_STR EQUALS
    PRINT_OCT
    PRINT_STR EQUALS
    PRINT_STR SPACE
    JMP ADR0
QUIT:
    EXIT

MAIN ENDP



HEX_KEYB PROC NEAR  ;modified, can also read 'T' source from book page 378
    PUSH DX
IGNORE:
    READ
    CMP AL,'T'
    JE ADDR2
    CMP AL,30H
    JL IGNORE
    CMP AL,39H
    JG ADDR1
    PUSH AX
    PRINT AL
    POP AX
    SUB AL,30H
    JMP ADDR2
ADDR1:
    CMP AL,'A'
    JL IGNORE
    CMP AL,'F'
    JG IGNORE
    PUSH AX
    PRINT AL
    POP AX
    SUB AL,37H
ADDR2:
    POP DX
    RET
HEX_KEYB ENDP


PRINT_HEX PROC NEAR ;Same as book
    CMP DL,9
    JLE ADDR3
    ADD DL,37H
    JMP ADDR4
ADDR3:
    ADD DL,30H
ADDR4
    PRINT DL
    RET
PRINT_HEX ENDP



PRINT_BIN PROC NEAR
    PUSH DX     ;Save registers used
    PUSH CX
    PUSH AX
    MOV AX,DX
    MOV CX,8    ;Loop 8 times
LB1:
                ;We want the digits to be printed MSB->LSB
    ROL DL,1    ;Shift left once to get MSB to LSB's position
    MOV AL,DL
    AND DL,01H  ;Isolate first digit
    PRINT_HEX   ;Print it
    MOV DL,AL
    LOOP LB1
    POP AX
    POP CX
    POP DX
    RET
PRINT_BIN ENDP


PRINT_OCT PROC NEAR
    PUSH DX     ;Save registers
    PUSH CX
    PUSH AX
    MOV AX,DX
    ;There are 8 digits, we want them printed in groups of 3

    ROL DL,2    ;First get the 2 MSB's to LSB to be printed
    AND DL,03H  ;Keep only first two bits
    PRINT_HEX

    MOV DL,AL   ;Now get the next 3 bits to be printed
    ROL DL,3
    MOV AL,DL
    AND DL,07H  ;Keep first 3 bits
    PRINT_HEX

    MOV DL,AL   ;Same as before, final 3 bits
    ROL DL,3
    AND DL,07H
    PRINT_HEX
    POP AX
    POP CX
    POP DX
    RET
PRINT_OCT ENDP


PRINT_DEC PROC NEAR ;From the book, page 381
    PUSH AX
    PUSH BX
    PUSH DX
    MOV CX, 0         ;Counter = 0
    MOV AX, DX
ADDR5:  MOV DX, 0
    MOV BX,10
    DIV BX            ;Divide by 10
    PUSH DX           ;Save remainder on stack
    INC CX            ;One more digit
    CMP AX, 0         ;If remainder==0 no more digits
    JNE ADDR5
ADDR6:
    POP DX
    ADD DX,30H        ;Find ascii and print
    PRINT DL
    LOOP ADDR6        ;Loop for all digits
    POP DX
    POP BX
    POP AX
    RET
PRINT_DEC ENDP


CODE_SEG ENDS
    END MAIN
```

## ***Άσκηση 4η***

```
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

DATA_SEG SEGMENT
    TABLE DB 20 DUP(20) ;ston pinaka TABLE
    NL DB 0AH,0DH,'$'   ;apothikeuontai oi 20
DATA_SEG ENDS           ;haraktires

CODE_SEG SEGMENT
    ASSUME CS:CODE_SEG, DS:DATA_SEG

MAIN PROC FAR
    MOV AX,DATA_SEG
    MOV DS,AX

START:
        MOV BX,0        ;metritis twn stoixeiwn
  GEMISMA_TABLE:        ;pou apothikeuontai ston
        CMP BX,20              ;TABLE
        JE EMFANISH_STOIXEIWN  ;an xeperasoun ta 20
        READ                   ;emfanizontai ta
        CMP AL,'='      ;apotelesmata stin othoni
        JE TELOS
        CMP AL,13
        JE IF_ENTER
        CMP AL,'0'        ;eleghos an o haraktiras
        JL GEMISMA_TABLE  ;einai metaxy 0-9 h a-z
        CMP AL,'9'
        JLE IF_0_TO_9
        CMP AL,'a'
        JL GEMISMA_TABLE
        CMP AL,'z'
        JLE IF_a_TO_z
        JMP GEMISMA_TABLE

  IF_ENTER:               ;an patithei ENTER kai yparhoun
        CMP BX,0          ;stoixeia ston pinaka,
        JE GEMISMA_TABLE        ;emfanizontai stin othoni
        JMP EMFANISH_STOIXEIWN

  IF_0_TO_9:
        PRINT AL
        MOV TABLE[BX],AL
        INC BX
        JMP GEMISMA_TABLE

  IF_a_TO_z:
        PRINT AL
        SUB AL,32
        MOV TABLE[BX],AL
        INC BX
        JMP GEMISMA_TABLE

  EMFANISH_STOIXEIWN:
        PRINT_STR NL    ;ta apotelesmata emfanizontai sthn
        MOV CX,BX       ;epomeni grammi
        MOV BX,0
        PRINT_ST:
            CMP BX,CX     ;eleghoume kathe stigmh poio
            JE TELOS_PRINT   ;stoixeio emfanizetai, molis
            MOV AL,TABLE[BX] ;o metritis ginei megalyteros
            PRINT AL     ;tou arithmou twn stoixeiwn tou
            INC BX       ;pinaka, metavainoume stin
            JMP PRINT_ST ;TELOS_PRINT

        TELOS_PRINT:
            PRINT_STR NL  ;allagh grammis kai anamonh gia
            MOV BX,0      ;plhktrologisi newn haraktirwn
            JMP GEMISMA_TABLE
  TELOS:

MAIN ENDP

CODE_SEG ENDS
    END MAIN
```

## ***Άσκηση 5η***

```
INCLUDE MACROS
DATA_SEG SEGMENT
    TEMP DW ? ;Input temperature
    MSG_1 DB "START (Y,N):",0AH,0DH,"$"
    MSG_2 DB 0AH,0DH,"DISPLAY: D",0AH,0DH,
    DB "QUIT: N",0AH,0DH,"$"
    MSG_3 DB "T = ","$"
    MSG_ER DB "ERROR",0AH,0DH,"$"
    UNITS DB 020H,0F8H,"C",0AH,0DH,"$"
    UF_1 DB "Give a 3 digit hex number.",0AH,0DH,"$"
    BYE DB "BYE","$"
    NEW_LINE DB 0AH,0DH,"$"
DATA_SEG ENDS
CODE_SEG SEGMENT
ASSUME CS:CODE,DS:DATA,SS:STACK
;-----------------------------------------------------------------
MAIN PROC FAR
    MOV AX,DATA_SEG
    MOV DS,AX ;DS = base address of DATA SEGMENT
START:
    PRINT_STR MSG_1
    PRINT_STR MSG_2
    KEEP_WORKING:
    READ
    CMP AL,"N" ;If 'N' was pressed, exit
    JE EXIT
    CMP AL,"D" ;If 'C' was pressed, proceed
    JNE KEEP_WORKING
    CALL READ_HEX_3 ;Read the temperature from "port"
CONVERT:
    MOV AX,TEMP ;AX = X
    CMP AX,4095 ;If X > 4095 (TEMP > 999.9)
    JG ERROR ;print "ERROR"
    CMP AX,3000 ;If X > 3000 (TEMP > 500)
    JG OVER_500 ;jump to OVER_500
    MOV BX,5
    MUL BX ;Y = 5 * X
    MOV BX,3
    DIV BX ;Y = Y / 3
    JMP READY
OVER_500:
    SUB AX,3000 ;X = X - 3000
    MOV BX,4999
    MUL BX ;Y = 4999 * X
    MOV BX,1095
    DIV BX ;Y = Y / 1095
    ADD AX,5000 ;Y = Y + 5000
READY:
    MOV TEMP,AX ;Replace the old value of TEMP
    ;with the converted one
    PRINT_STR MSG_3 ;"T = "
    CALL PRINT_BCD ;Print the temperature in BCD
    PRINT_STR UNITS ;Print units and change line
    JMP KEEP_WORKING
ERROR:
    PRINT_STR MSG_ER
    JMP KEEP_WORKING
EXIT:
    PRINT_STR BYE
    MOV AX,4C00H
    INT 21H
```

```
MAIN ENDP
;-----------------------------------------------------------------
READ_HEX_3 PROC NEAR
PRINT_STR UF_1
      MOV CL,12
      MOV DX,0H
KEEP_READING:
      READ ;Read digit
      CMP AL,30H ;Make sure it is a number,
      JL KEEP_READING ;or a letter between A and F,
      CMP AL,3AH ;else keep reading
      JL NUMBER
      CMP AL,41H
      JL KEEP_READING
      CMP AL,46H
      JG KEEP_READING
      PUSH DX
      PRINT AL ;Print the letter
      POP DX
      SUB AL,37H
      JMP BOTTOM
NUMBER:
      PUSH DX
      PRINT AL ;Print the number
      POP DX
      SUB AL,30H
BOTTOM:
      MOV AH,0H ;AH = 0
      SUB CL,4 ;CL -= 4
      ROL AX,CL ;Rotate left for CL bits
      OR DX,AX
      CMP CL,0 ;If CL = 0 stop reading
      JNE KEEP_READING
      MOV TEMP,DX ;Store the number in memory
      PRINT_STR NEW_LINE ;new line
      RET
READ_HEX_3 ENDP
;-----------------------------------------------------------------
PRINT_BCD PROC NEAR
    MOV AX,TEMP ;AX = TEMP
    MOV CX,0 ;digit COUNTER
    MOV BX,0AH ;BX = 10
    DIVIDE:
    MOV DX,0 ;DX = 00H
    DIV BX ;AX/10 = AX
    PUSH DX ;Push remainder
    INC CX ;CX += 1
    CMP AX,0 ;If AX = 0 stop
    JNE DIVIDE
    NEXT_DIGIT:
    POP DX ;Pop digit
    ADD DX,30H ;Convert it to ASCII
    CMP CX,1
    JNE SKIP
    PUSH DX
    PRINT "."
    POP DX
    SKIP:
    PRINT DL ;Print it
    LOOP NEXT_DIGIT ;Loop until CX = 0
    RET
PRINT_BCD ENDP
;-----------------------------------------------------------------
CODE_SEG ENDS
```

```
END MAIN
```