**Συστήματα Μικρουπολογιστών - 5ή Σειρά Ασκήσεων**

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Ακαδημαϊκό έτος : 2018 – 2019 Εξάμηνο : 6ό

Μέλη ομάδας : Βόσινας Κωνσταντίνος ΑΜ : 03116435

Ανδριόπουλος Κωνσταντίνος ΑΜ : 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