**Εργαστήριο Μικροϋπολογιστών**

**3η Άσκηση**

Ομάδα: Δ12

Βακαλόπουλος Θεόδωρος, ΑΜ: 03114013

Μαυρομμάτης Ιάσων, ΑΜ: 03114771

Νικητοπούλου Δήμητρα, ΑΜ: 03114954

**Άσκηση 1**

Ο κώδικας της άσκησης φαίνεται παρακάτω:

include macros.txt

data\_seg segment

msg1 db ".",'$'

msg2 db 0ah,0dh,'$' ;new line

msg3 db "Decimal:",'$'

data\_seg ends

stack segment

dw 50 DUP(?)

stack ends

code segment

assume cs:code,ss:stack,ds:data\_seg,es:data\_seg

start:

mov ax,data\_seg

mov ds,ax

mov es,ax

begin:

mov ch,00h ;ch is a flag:if given D12, ch equals 3

call hex\_keyb

cmp al,0dh

jne p1

add ch,01h ;if 1st digit is D increase flag

p1: mov cl,4

rol al,cl ;1st digit is MSD so we shift it 4 steps left

mov bl,al

call hex\_keyb

print\_str msg1 ;print digital point

mov cl,al

cmp al,01h

jne p2

add ch,01h ;if 2nd digit is 1 increase flag

p2: add bl,al ;add the 2 digits to get the 2-digit hex number

call hex\_keyb

mov cl,al ;keep copy of decimal point

cmp al,02h

jne p3

add ch,01h ;if 3rd digit is 2 increase flag

p3: cmp ch,03h

je ending ;if flag is 3 we have to stop the program

print\_str msg2 ;new line

print\_str msg3

call print\_dec ;print bl

print\_str msg1

mov bl,cl ;ah : arithmos sto [0,16]

call printdp

print\_str msg2

jmp begin

ending:

exit

print\_dec proc near

push bx

push dx

hund:

mov bh,00h

mov ax,bx

mov dl,100 ;divide with 100 to find hundreds

div dl ;al holds div, ah holds mod

cmp al,00h

je deca ;if hundreds=0 dont print it

add al,'0'

print al ;print hundreds

deca: mov bl,ah ;take mod of previous division

mov ax,bx

mov dl,10 ;divide with 10 the remaining to find tens

div dl ;al holds div, ah holds mod

add al,'0'

print al ;print tens

mon: mov bl,ah ;take mod of previous division

add bl,'0'

print bl ;print units(the remainder)

pop dx

pop bx

ret

print\_dec endp

hex\_keyb proc near

ignore:

read

cmp al,30h ;10 digits have codes from 30 to 39

jl ignore

cmp al,39h

jle valid1

cmp al,41h ;hex lowercase letters have codes from 41 to 46

jl ignore

cmp al,46h

jle valid2

cmp al,61h ;hex uppercase letters have codes from 61 to 66

jl ignore

cmp al,66h

jle valid3

jmp ignore

valid1:

print al

sub al,30h ;leave ascii code

jmp leave

valid2:

print al

add al,0ah

sub al,41h ;leave ascii code

jmp leave

valid3:

print al

add al,0ah

sub al,61h ;leave ascii code

leave:

ret

hex\_keyb endp

printdp proc near

push ax

push cx ;to map [0,16] at [0,9] multiply with 10 and divide by 16

mov ch,16 ;map [0,16] -> [0,9]

mov ah,00

mov al,bl ;bl : arithmos sto [0,16]

mov cl,10

mul cl ;multiply hex decimal point with 10

div ch ;divide by 16->al holds current decimal point

add al,'0' ;go to ascii and print

print al

mov bh,03h ;counter so that it prints 4 decimal digits(we've checked that's the maximum number)

loop1:

mov al,ah ;take remainder print div->2nd decimal digit

mul cl

div ch ;x10/16 to find the i-th decimal digit

add al,'0' ;go to ascii

print al

sub bh,01h ;execute loop 3 times

jnz loop1

pop cx

pop ax

ret

printdp endp

code ends

end START

Σχόλια:

* Η μετατροπή του ακεραίου μέρους του αριθμού σε δεκαδική μορφή γίνεται στη ρουτίνα print\_dec με τη μέθοδο των επαναλαμβανόμενων διαιρέσεων
* Όσον αφορά στο κλασματικό μέρος του αριθμού η μετατροπή γίνεται στη ρουτίνα printdp η οποία επιτυγχάνει ακρίβεια 4 δεκαδικών ψηφίων όπως ζητείται στην εκφώνηση.

**Άσκηση 2**

;\*\*\*\*\*\* Define Macros \*\*\*\*\*\*\*

READ MACRO ;Read character from keyboard

MOV AH, 8 ;The ASCII code of the character is

INT 21H ;stored in register AL

ENDM

PRINT MACRO CHAR ;Print character CHAR

MOV DL, CHAR

MOV AH, 2

INT 21H

ENDM

PRINT\_STR MACRO STRING ;Print a sequence of characters

MOV DX, OFFSET STRING

MOV AH, 9

INT 21H

ENDM

EXIT MACRO ;Exit program

MOV AX, 4C00H

INT 21H

ENDM

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

DATA\_SEG SEGMENT

MSG1 DB 0AH, 0DH, 'GIVE 2 DECIMAL DIGITS:$'

MSG2 DB 0AH, 0DH, 'OCTAL= $'

DATA\_SEG ENDS

CODE\_SEG SEGMENT

ASSUME CS:CODE\_SEG, DS:DATA\_SEG

MAIN PROC FAR

MOV AX, DATA\_SEG

MOV DS, AX

START:

PRINT\_STR MSG1

MOV CL, 0 ;Counter of the decimal digits inserted

LOOP1:

CALL DEC\_KEYB ;Read one decimal digit

CMP AL, 'Q' ;If 'Q' was pressed exit program

JE QUIT

CMP AL, 0DH ;If '/n' was pressed check counter

JE CHECK

INC CL ;Otherwise increase counter

MOV BH, BL ;Store the last two decimal digits in

MOV BL, AL ;registers BH and BL respectively

JMP LOOP1 ;Continue read process

CHECK:

CMP CL, 02H ;Check if at least 2 decimal digits have

JGE NEXT ;already been inserted

JMP LOOP1 ;If yes go on, otherwise continue reading

NEXT:

MOV AL, BH ;Create hex number from decimal

ROL BH, 3 ;BH = 8 \* first\_digit

ADD BH, AL ;Add two more times BH to the result so

ADD BH, AL ;that BH = 10 \* first\_digit

ADD BH, BL ;BH = 10 \* first\_digit + second\_digit

PRINT\_STR MSG2

MOV CL, BH ;Copy BH to CL

AND BH, 00C0H ;Isolate 2 MSBs of BH

ROL BH, 2 ;Rotate them to become 2 LSBs

ADD BH, 30H ;Create ASCII code of the number

PRINT BH ;Display MSB octal digit

MOV BH, CL ;Restore initial value of BH

AND BH, 38H ;Isolate bytes 4-6

ROR BH, 3 ;Rotate them to become LSBs

ADD BH, 30H ;Create ASCII code of the number

PRINT BH ;Display middle octal digit

AND CL, 07H ;Isolate 3 LSBs

ADD CL, 30H ;Create ASCII code of the number

PRINT CL ;Display LSB octal digit

JMP START ;Loop forever

QUIT:

EXIT

MAIN ENDP

DEC\_KEYB PROC NEAR

PUSH DX

IGNORE:

READ

CMP AL, 'Q'

JE RETURN

CMP AL, 0DH

JE RETURN

CMP AL, 30H

JL IGNORE

CMP AL, 39H

JG IGNORE

PUSH AX

PRINT AL

POP AX

SUB AL, 30H

RETURN:

POP DX

RET

DEC\_KEYB ENDP

CODE\_SEG ENDS

END MAIN

Σχόλια:

* Το πρόγραμμα δέχεται δεκαδικά ψηφία μέσω της ρουτίνας DEC\_KEYB και αφού εξασφαλίσει ότι έχουν δοθεί τουλάχιστον 2 λαμβάνει υπόψιν τα 2 τελευταία για τη μετατροπή.
* Η μετατροπή του δεκαδικού αριθμού σε οκταδικό γίνεται αφού πρώτα μετατραπεί ο δεκαδικός σε δεκαεξαδικό. Στη συνέχεια η μετατροπή σε οκταδικό γίνεται ουσιαστικά θεωρώντας ανα τρία τα bit στην αναπαράσταση του αριθμού ως ένα οκταδικό ψηφίο.

**Άσκηση 3**

include macros.txt

data\_seg segment

msg1 db 0ah,0dh,'$' ;new line

msg2 db "input: ",0ah,0dh,'$'

msg3 db "changed\_input: ",0ah,0dh,'$'

msg4 db "2 greatest numbers: ",0ah,0dh,'$'

msg5 db " '=' was given, end of program",0ah,0dh,'$'

msg6 db " ", 0ah,0dh,'$' ;keno

data\_seg ends

stack segment

dw 50 DUP(?)

stack ends

code segment

assume cs:code,ss:stack,ds:data\_seg,es:data\_seg

start:

mov ax,data\_seg

mov ds,ax

mov es,ax

begin:

mov si,1840h ;1840h ,memory for numbers

mov [si],00h

mov si,1820h ;1820h, memory for capitals

mov [si],00h

mov si,1800h ;1800h, memory for lower-case

mov [si],00h

mov si,1860h ;1860h, memory for max

mov [si],20h

mov si,1861h ;1861h, memory for second biggest

mov [si],20h

print\_str msg2 ;input:

call input

print\_str msg1 ;print new line

cmp bh,01h ;if = is given, stop execution

je ending

print\_str msg3 ;changed input:

mov ch,00h

mov ah,00h ;counter of appearances of max

mov di,1840h

mov cl,[di] ;initialise counter of loops with content of[1840h]

cmp cl,00h ;if no number was given print space and go on

je print\_keno

print\_Num: ;increase pointer of the array "counter" times and print every number

add di,01h

push cx

mov cx,di ;keep copy of address

mov al,[di]

print al ;print current number

mov di,1861h ;if (second max<current & second max!=max) second max=current

mov bl,[di] ;bl contains second max

cmp bl,al

jg move\_on2

mov si,1860h

mov bl,[si] ;bl contains max

cmp al,bl

je move\_on3

mov [di],al ;if condition change second max in memory

jmp move\_on2

move\_on3:

add ah,01h ;if current=max increase counter of appearances of max

move\_on2:

mov di,cx ;di takes its former value for next loop

pop cx

loop print\_Num

print\_keno:

print " "

mov ch,00h

mov di,1820h

mov cl,[di] ;initialise counter

cmp cl,00h ;if no capitals were given print space and go on

je print\_keno2

print\_upper:;increase pointer of the array "counter" times and print every char

add di,01h

mov al,[di]

print al ;take current char and print it

loop print\_upper

print\_keno2:

print " "

mov ch,00h

mov di,1800h

mov cl,[di];initialise counter

cmp cl,00h ;if no capitals were given print space and go on

je print\_biggest

print\_lower:;increase pointer of the array "counter" times and print every char

add di,01h

mov al,[di]

print al ;take current char and print it

loop print\_lower

print\_biggest:

print\_str msg1 ;print new line

print\_str msg4 ;2 greatest numbers:

mov di,1840h

mov cl,[di]

cmp cl,00h ;if no number was given, print nothing and start over

je return

cmp cl,01h

jne p2

add di,01h ;if one number was given, print it and start over

mov al,[di]

print al

jmp return

p2: ;if at least 2 numbers were given

mov cl,02h ;initialise counter of numbers printed

cmp ah,01h

je printbig

mov si,1860h ;if maximum appears at least 2 times

mov al,[si] ;print it two times and start over

print al

print al

jmp return

printbig:;if 2 different numbers have to be printed(1st and 2nd max)

add di,01h

mov al,[di];traverse the array and print max and second max with

mov si,1860h ;without changing their sequence

mov bh,[si]

cmp al,bh ;if current=max print it

jne nope

print al

sub cl,01h ;if one is printed, renew the number of numbers that still need to be printed

jmp p1

nope:

mov si,1861h

mov bh,[si]

cmp al,bh ;if current =second max print it

jne p1

print al

sub cl,01h

p1:

jne printbig ;continue until 2 are printed

return:

print\_str msg1 ;print new line

jmp begin

ending:

print\_str msg5 ;print end of program

exit

input proc near

mov ch,00h

mov cl,14 ;14 chars maximum will be given

mov bh,00h ;flag, if '=' is given bh=1

ignore:

read

cmp al,0dh ;if enter, return

je leave

cmp al,3dh ;if '=', change flag and return

je stop

cmp al,20h ;if space, print space and continue

je valid

cmp al,30h

jl ignore

cmp al,39h

jle valid1 ;if number,store in memory,print and continue

cmp al,41h

jl ignore

cmp al,5ah

jle valid2 ;if capital,store in memory,print and continue

cmp al,61h

jl ignore

cmp al,7ah

jg ignore

valid3: ;if lower-case,store in memory,print and continue

mov di,1800h ;[1800h] is counter of lower-case letters given

mov bl,[di]

add bl,01h ;every time we increase it by one and store it again

mov [di],bl

print al ;print char

push bx

add bx,1800h ;find address 1800h+counter

mov di,bx

pop bx

mov [di],al ;store char in[1800h+counter]

loop ignore

jmp leave

valid2:

mov di,1820h ;[1820h] is counter of upper-case letters given

mov bl,[di]

add bl,01h ;every time we increase it by one and store it again

mov [di],bl

print al ;print char

push bx

add bx,1820h ;find address 1820h+counter

mov di,bx

pop bx

mov [di],al ;store char in[1820h+counter]

loop ignore

jmp leave

valid1:

mov di,1860h

mov bl,[di]

cmp bl,al ;if current>max, max=current

jg move\_on

mov [di],al

move\_on:

mov di,1840h ;[1840h] is counter of numbers given

mov bl,[di]

add bl,01h ;every time we increase it by one and store it again

mov [di],bl

print al ;print num

push bx

add bx,1840h ;find address 1840h+counter

mov di,bx

pop bx

mov [di],al ;store char in[1840h+counter]

loop ignore

jmp leave

valid:

print " " ;print space if given

loop ignore

jmp leave

stop:

mov bh,01h ;flag set

leave:

ret

input endp

code ends

end START

Σχόλια:

* Το διάβασμα των χαρακτήρων εισόδου (το πολύ 14) γίνεται στη ρουτίνα input η οποία κατατάσσει τους χαρακτήρες σε διαφορετικά προκαθορισμένα σημεία της μνήμης ανάλογα αν πρόκειται για μικρά, κεφαλαία γράμματα ή αριθμούς. Ειδικά στην περίπτωση των αριθμών η ρουτίνα υπολογίζει παράλληλα το μέγιστο μεταξύ όλων των αριθμών εισόδου.
* Στη συνέχεια το κυρίως πρόγραμμα τυπώνει ξεχωριστά τους μικρούς και κεφαλαίους χαρακτήρες καθώς και τους αριθμούς υπολογίζοντας παράλληλα το δεύτερο μεγαλύτερο αριθμό εισόδου.
* Τέλος τυπώνονται οι δύο μεγαλύτεροι αριθμοί εισόδου πραγματοποιώντας κατάλληλους ελέγχους αναφορικά με το πλήθος τους.

**Άσκηση 4**

include macros.txt

CODE\_SEG SEGMENT

ASSUME CS:CODE\_SEG, DS:DATA\_SEG

MAIN PROC FAR

START:

MOV SI, 1844H ;Store in address 1844H a flag

MOV [SI],00H ;flag=0 => 1st call of READ\_NUMBER routine

;flag=1 => 2nd call of READ\_NUMBER routine

CALL READ\_NUMBER ;Read first decimal number

CMP AL, "Q" ;Check if abort button was pressed

JE QUIT ;If yes end program

CONTINUE:

PUSH AX ;Save operator to stack

PRINT AL ;Print operator

MOV SI, 1840H ;Source address

MOV DI, 183DH ;Destnation address

MOV CX, 03H

COPY: ;Copy 3 decimal digits from memory

MOV AL, [SI] ;addresses 1840-1842H to addresses

MOV [DI], AL ;183D-183FH

INC SI

INC DI

LOOP COPY

MOV SI, 1844H ;Chnage flag in memory address 1844H

MOV [SI], 01H ;to prepare for second call of READ\_NUMBER routine

CALL READ\_NUMBER ;Read second decimal number

CMP AL, "Q" ;Check again if abort button was pressed

JE QUIT ;If yes end program

PRINT AL ;Print "=" which is stored in AL register

;after return from READ\_NUMBER routine

POP AX ;Restore operator in AL register

CMP AL, "+" ;Check which operator was pressed

JNE SUBTRACT

;ADD

MOV SI,183DH ;Start address of first decimal number

CALL CREATE\_HEX ;Convert first decimal number to hex

MOV BX, AX ;Save first hex number to BX register

MOV SI, 1840H ;Start address of second decimal number

CALL CREATE\_HEX ;Convert second decimal number to hex

ADD AX, BX ;Add the two hex numbers

JMP DISPLAY\_RESULT

SUBTRACT:

MOV SI,183DH

CALL CREATE\_HEX

MOV BX, AX

MOV SI, 1840H

CALL CREATE\_HEX

SUB BX, AX

MOV AX, BX ;Subtract the two hex numbers

DISPLAY\_RESULT:

ROL AX, 1 ;Check if result is positive or negative

JNC POSITIVE

PUSH AX

PRINT "-" ;If negative print "-" and take 2's

POP AX ;complement

ROR AX, 1

NOT AX

ADD AX, 0001H

MOV SI, 1845H ;Store in memory address 1845H a flag

MOV [SI], 01H ;Flag = 1 means negative number

JMP PRINT\_NUMBER

POSITIVE:

ROR AX, 1 ;Rotate rigth to restore the result

MOV SI, 1845H

MOV [SI], 00H ;Flag = 0 means positive number

PRINT\_NUMBER:

CALL PRINT\_HEX ;Print the result as a hex number

PUSH AX

PRINT "=" ;Print "="

POP AX

CALL CREATE\_DECIMAL ;Print the result as a decimal number

PRINT 0DH ;Newline

PRINT 0AH

JMP START ;Start again

QUIT:

EXIT

MAIN ENDP

;Routines

READ\_NUMBER PROC NEAR ;This procedure reads a 3-digit decimal

PUSH DX ;number and stores it in memory addresses

PUSH CX ;1840h-1842h

PUSH SI

MOV SI, 1840H ;Clean memory

MOV [SI] ,00H

INC SI

MOV [SI], 00H

INC SI

MOV [SI], 00H

MOV CL, 00H

IGNORE:

READ ;Read a character from keyboard

CMP AL, 'Q' ;If "Q" was pressed return

JE RETURN

CMP AL, "+" ;If "+" was pressed check

JE CHECK\_OPERATOR

CMP AL, "-" ;If "-" was pressed check

JE CHECK\_OPERATOR

CMP AL, "=" ;If "=" was pressed check

JE CHECK\_FOR\_EQUAL

CMP AL, 30H

JL IGNORE

CMP AL, 39H

JG IGNORE

CMP CL, 03H ;If already 3 digits have been pressed

JGE IGNORE ;ignore current digit

JMP NEXT ;If it is a valid digit continue

CHECK\_OPERATOR:

PUSH SI ;Save SI to stack

MOV SI,1844H ;Check flag if it is the first call

MOV DH, [SI] ;of the routine

POP SI

CMP DH, 00H

JNE IGNORE ;If no ignore character

CMP CL, 01H ;Else if at least one digit has already

JL IGNORE ;been pressed return

JMP RETURN

CHECK\_FOR\_EQUAL:

PUSH SI ;Save SI to stack

MOV SI, 1844H ;Check flag if it is the second call

MOV DH, [SI] ;of the routine

POP SI

CMP DH, 01H

JNE IGNORE ;If no ignore character "="

CMP CL, 01H ;Else if at least one digit has already

JL IGNORE ;been pressed return

JMP RETURN

NEXT: ;If we have a valid digit

PUSH AX

PRINT AL ;Print current digit

POP AX

SUB AL, 30H ;Convert from ASCII code to its value

MOV SI, 1841H ;Shift the values stored in memory

MOV DI, 1840H ;addresses 1840-1842H 1 position left

MOV DH, [SI]

MOV [DI], DH

INC DI

INC SI

MOV DH, [SI]

MOV [DI], DH

MOV SI, 1842H ;Save current digit in memory address

MOV [SI], AL ;AL

INC CL ;Increse counter for digits inserted

JMP IGNORE

RETURN:

POP SI

POP CX

POP DX

RET

READ\_NUMBER ENDP

CREATE\_HEX PROC NEAR ;This procedure converts a 3-digit

;decimal number which is stored in

PUSH BX ;memory starting from address 1840H

PUSH CX ;to a hex number which is contained

;in AX registe

MOV AL, [SI] ;AL = MSD

MOV AH, 00H

ROL AX, 6 ;AX = 64 \* MSD

MOV BL, [SI] ;BL = MSD

MOV BH, 00H

ROL BX, 5 ;BX = 32 \* MSD

MOV CL, [SI] ;CL = MSD

MOV CH, 00H

ROL CX, 2 ;CX = 4 \* MSD

ADD AX, BX

ADD AX, CX ;AX = 100 \* MSD

INC SI

MOV BH, [SI] ;BH = Second MSD

ROL BH, 3 ;BH = 8 \* Second MSD

MOV BL, [SI] ;BL = Second MSD

ROL BL, 1 ;BL = 2 \* Second MSD

ADD BL, BH

MOV BH, 00H ;BX = 10 \* Second MSD

INC SI

MOV CL, [SI]

MOV CH, 00H ;CX = LSD

ADD AX, BX

ADD AX, CX ;AX holds the hex number

POP CX

POP BX

RET

CREATE\_HEX ENDP

PRINT\_HEX PROC NEAR ;This procedure takes a 16bit hex

;number contained in BX register

MOV BH, AH ;and prints it

AND BH, 0FH ;Isolate MSD

JZ NEXT\_DIGIT1

CALL PRINT\_DIGIT ;If MSD != 0 print it

NEXT\_DIGIT1:

MOV BH, AL

AND BH, 00F0H ;Isolate second MSD

ROR BH, 4

JNZ PRINT\_DIG1 ;If second MSD != 0 print it

MOV CL, AH ;Else check if MSD was 0

AND CL, 0FH

CMP CL, 00H

JZ NEXT\_DIGIT2 ;If both are zero continue to next

PRINT\_DIG1:

CALL PRINT\_DIGIT ;Otherwise print it

NEXT\_DIGIT2:

MOV BH, AL ;Isolate LSD

AND BH, 0FH

CALL PRINT\_DIGIT ;Print LSD

RET

PRINT\_HEX ENDP

PRINT\_DIGIT PROC NEAR ;This procedure takes a digit

CMP BH, 09H ;contained in BH register and

JG ADDR1 ;prints it

ADD BH, 30H ;If digit is 0-9

JMP ADDR2

ADDR1:

ADD BH, 37H ;If digit is A-F

ADDR2:

PUSH AX

PRINT BH

POP AX

RET

PRINT\_DIGIT ENDP

CREATE\_DECIMAL PROC NEAR

;This procedure converts a hex

MOV SI, 1845H ;number stored in AX register

MOV CL, [SI] ;to decimal and prints it

CMP CL, 00H ;Check flag to check if number

JZ POS ;is positive or negative

PUSH AX

PRINT "-" ;If negative print "-"

POP AX

POS:

MOV BL, 00FFH ;Counter of thousands

THND:

INC BL ;Increase thousands

SUB AX, 03E8H ;AX <= AX - 1000(10)

CMP AX, 0000H

JGE THND ;If still positive loop

ADD AX, 03E8H ;Else add 1 thousand to become

;again positive and continue

MOV CL, 00FFH ;Counter of hundreds

HUND:

INC CL ;Increase hundreds

SUB AX, 0064H ;AX <= AX - 100(10)

CMP AX, 0000H

JGE HUND ;If still positive loop

ADD AX, 0064H ;Else add 1 hundred to become

;again positive and continue

MOV CH, 00FFH ;Counter of tens

DECA:

INC CH ;Increase tens

SUB AX, 000AH ;AX = AX - 10(10)

CMP AX, 0000H ;If still positive loop

JGE DECA ;Else add 1 ten to become

ADD AX, 000AH ;again positive and continue

CMP BL, 00H ;Check thousands

JZ NEXT\_DIGIT3 ;If thousands = 0 continue to next

;digit

MOV BH, BL ;Else print thousands

CALL PRINT\_DIGIT

NEXT\_DIGIT3:

MOV BH, CL ;Check hundreds

CMP CL, 00H ;If hundreds = 0 check thousands

JNE PRINT\_DIG2 ;If hundreds != 0 print them

CMP BL, 00H ;If both 0 continue to next digit

JZ NEXT\_DIGIT4

PRINT\_DIG2:

CALL PRINT\_DIGIT

NEXT\_DIGIT4:

MOV BH, CH ;Check tens

CMP CH, 00H ;If tens = 0 check thousands and hundreds

JNE PRINT\_DIG3 ;If all 0 continue to next digit

CMP CL, 00H ;Else print tens

JNE PRINT\_DIG3

CMP BL, 00H

JZ NEXT\_DIGIT5

PRINT\_DIG3:

CALL PRINT\_DIGIT

NEXT\_DIGIT5:

MOV BH, AL ;Print units

CALL PRINT\_DIGIT

RET

CREATE\_DECIMAL ENDP

CODE\_SEG ENDS

END MAIN

Σχόλια:

* Αρχικά το πρόγραμμα διαβάζει δύο δεκαδικούς αριθμούς το πολύ τριών ψηφίων καθώς και την πράξη που επιθυμούμε να υλοποιήσουμε (πρόσθεση ή αφαίρεση) με δύο διαδοχικές κλήσεις της ρουτίνας READ\_NUMBER.
* Στη συνέχεια τους μετατρέπει σε δεκαεξαδική μορφή μέσω της ρουτίνας CREATE\_HEX η οποία εκτελεί τη λειτουργία Hex = 100\*MSD + 10\*2ndMSD + LSD.
* Εκτελεί τη ζητούμενη λειτουργία στο δεκαεξαδικό σύστημα.
* Εκτυπώνει το αποτέλεσμα σε δεκαεξαδική μορφή λαμβάνοντας υπόψιν τόσο το πρόσημο του αποτελέσματος όσο και τον αριθμό των ψηφίων για την αναπαράστασή του. Συγκεκριμένα, εφόσον το αποτέλεσμα είναι αρνητικός αριθμός το πρόγραμμα υπολογίζει το συμπλήρωμα του ως προς δύο και τυπώνει το πρόσημο – πριν την εκτύπωση της απόλυτης τιμής του. Ακόμη εκτελούνται οι κατάλληλοι έλεγχοι ώστε να τυπώνονται ακριβώς τα σημαντικά ψηφία του αποτελέσματος χωρίς μηδενικά στην αρχή.
* Τέλος καλεί τη ρουτίνα CREATE\_DECIMAL η οποία μετατρέπει το αποτέλεσμα σε δεκαδική μορφή μέσω των επαναλαμβανόμενων διαιρέσεων με 1000, 100 και 10 αντίστοιχα και στη συνέχεια το εκτυπώνει με πρόσημο – αν είναι αρνητικός και λαμβάνοντας υπόψιν μόνο όσα ψηφία απαιτούνται χωρίς την εκτύπωση περιττών μηδενικών στην αρχή.