Συστήματα Μικροϋπολογιστών

6ο Εξάμηνο

5η Ομάδα Ασκήσεων

Χαρδούβελης Γεώργιος-Ορέστης Κυριάκου Αθηνά

el15100 el17405

6ο Εξάμηνο 6ο Εξάμηνο

**Άσκηση 1:**

Ο κώδικας για του προγράμματος που υλοποιεί το ερώτημα φαίνεται παρακάτω και βρίσκεται στο αρχείο **ex1.asm** . Δημιουργήθηκαν οι ρουτίνες PRINT\_DEC, PRINT\_OCT και PRINT\_BIN.

org 100h

LABS: MOV BX,3

MOV [BX],'$'

DEC BX

MOV [BX],'='

MOV DX,1

DEC BX

DEC BX

MOV AX,0

CALL HEX\_KEYB ;saves the first hex digit in AL (calls it for the first time>

MOV AH,AL

CMP AH,10

JB LABSA

JMP LABSB

LABSA: ADD AH,48

JMP LABSC

LABSB: ADD AH,55

LABSC: MOV [BX],AH ;puts it to the address of BX

INC BX ;increases BX by 1

MOV CL,16

MUL CL

MOV CH,AL

CALL HEX\_KEYB ;saves the second hex digit in AL

MOV AH,AL

CMP AH,10

JB LABSD

JMP LABSE

LABSD: ADD AH,48

JMP LABSF

LABSE: ADD AH,55

LABSF:

MOV [BX],AH ;puts it to the address of BX

INC BX ;increases BX by 1

ADD AL,CH

;the number is now in AL

LAB5: ;print the number in hex format

;since the number is 8-bits, at most 2 hex bits will be needed.

;number is currently in AL

MOV DX,0

MOV DL,AL ;moving the number to DL

MOV CH,DL

MOV DX,0

MOV AH,09H

INT 21H

MOV DL,CH

CALL PRINT\_DEC ;calling the three printing routines, firstly PRINT\_DEC, then PRINT\_OCT, then PRINT\_BIN

MOV BX,1

MOV [BX],'$'

DEC BX

MOV [BX],'='

MOV CH,CL

MOV DX,0

MOV AH,09H

INT 21H

MOV DL,CH

CALL PRINT\_OCT

MOV BX,1

MOV [BX],'$'

DEC BX

MOV [BX],'='

MOV CH,CL

MOV DX,0

MOV AH,09H

INT 21H

MOV DL,CH

CALL PRINT\_BIN

MOV BX,0

MOV [BX],10

INC BX

MOV [BX],'$'

MOV DX,0

MOV AH,09H

INT 21H

JMP LABS

LAB6: HLT

HEX\_KEYB: ;waits to get 'Q' or a hex digit from the keyboard. returns the value of this hex digit in AL.

LAB3: MOV AH,08H

INT 21H

CMP AL,'Q' ;checking if Q was input

JNE LAB3C ;if it is Q, quit the program

JMP LAB6

LAB3C: ;check if the input was valid, this means, if it belongs in

[48,57], [65,70] or [97,102].

CMP AL,48

JB LAB3

CMP AL,57

JBE LAB3N

CMP AL,65

JB LAB3

CMP AL,70

JBE LAB3U

CMP AL,97

JB LAB3

CMP AL,102

JBE LAB3L

JMP LAB3

LAB3N:

SUB AL,48

RET

LAB3U:

SUB AL,55

RET

LAB3L:

SUB AL,87

RET

PRINT\_DEC: ;since the number is 8-bits, it will have at most 3

decimal digits. so we will need at most 3

addresses to save the number

MOV BX,3

MOV [BX],'$'

MOV AL,DL

MOV CL,DL

MOV DL,3

LAB0: DEC BX ;puts the digits in order in the addresses to print them

DEC DL

MOV CH,10

MOV AH,0

DIV CH

ADD AH,48

MOV [BX],AH

CMP AL,0

JA LAB0

MOV AH,09H

INT 21H

MOV DL,CL

RET

PRINT\_OCT: ;255 in decimal (the maximum 8-bit value) is

377 in oct, which means that at most 3 digits will be needed.

MOV BX,3

MOV [BX],'$'

MOV AL,DL

MOV CL,DL

MOV DL,3

LAB1: DEC BX ;puts the digits in order in the addresses to print

them

DEC DL

MOV CH,8

MOV AH,0

DIV CH

ADD AH,48

MOV [BX],AH

CMP AL,0

JA LAB1

MOV AH,09H

INT 21H

MOV DL,CL

RET

PRINT\_BIN:

MOV BX,8 ;255 in decimal has 8 binary digits. so '$' will be

stored at the 8th place.

MOV [BX],'$'

MOV AL,DL

MOV CL,DL

MOV DL,8

LAB2: DEC BX ;puts the digits in order in the addresses to print

them

DEC DL

MOV CH,2

MOV AH,0

DIV CH

ADD AH,48

MOV [BX],AH

CMP AL,0

JA LAB2

MOV AH,09H

INT 21H

MOV DL,CL

RET

**Άσκηση 2:**

Στον κώδικα για την υλοποίηση της άσκησης, επιλέχθηκε στρογγυλοποίηση προς τα κάτω για τους δεκαδικούς αριθμούς. Δηλαδή, καθώς ο μέγιστος δεκαδικός αριθμός είναι ο 255 και ο ελάχιστος το 0 με μέσο όρο 127.5, βάσει της συγκεκριμένης επιλογής για στρογγυλοποίηση προς τα κάτω, χρησιμοποιούμε το 127. Επίσης, το πρόγραμμα αρχικά αποθηκεύει τους αριθμούς στον πίνακα με χρήση ενός βρόχου αποθήκευσης. Στη συνέχεια εκτελούνται σε έναν βρόχο αναζήτησης και οι δύο ζητούμενοι υπολογισμοί.

Ο κώδικας της συγκεκριμένης άσκησης, αποθηκευμένος στο αρχείο **ex2.asm** είναι:

org 100h

TABLE EQU 00

MOV AX,0

MOV BX,TABLE

MOV AL,254

LAB0: MOV [BX],AL

DEC AL

INC BX

CMP AL,0

JA LAB0

MOV [BX],0

INC BX

MOV [BX],255

;doing the calculations, in only one loop:

MOV BX,TABLE

;CH will have the highest number. (initial value = all zeros)

;CL will have the lowest number. (initial value = all ones)

;the current sum of the even numbers will be calculated in SI.

;DH will have the current number.

;DL will count how many numbers are checked.

;we know that all the even numbers are 128 - so there will be a division of the final content of SI with 128

;which means that the result (we don't care about the remainder) will be in AL.

;so AL will have the average.

MOV CH,0

MOV SI,0

MOV CL,255

MOV DL,255

MOV BX,TABLE

JMP LAB1

LAB5: DEC DL

LAB1:

XCHG BX,[4500]

MOV DH,[BX]

INC BX

CMP DH,CH

JNA LAB2

MOV CH,DH

LAB2: CMP DH,CL

JNB LAB3

MOV CL,DH

LAB3: XCHG BX,[4500]

MOV BL,DH

MOV AL,BL

RCR AL,1

JC LAB3B

ADD SI,BX

LAB3B: CMP DL,0

JNE LAB5

MOV AX,SI

MOV BL,128 ;we don't need BX anymore, so we fill BL with the number 128

MOV DX,0 ;<AH:AL> will be divided by 128.

DIV BL ;division

;the result now is in (AL> and the remainder in <AH>

;we now need to print the contents of CH (highest number), of CL (lowest number) and of (AH:AL) (average).

;we will use PRINT\_DEC for this, which is the same as in the exercise 5\_1.

;PRINT\_DEC influences registers AH,AL,BH,BL,CH and CL.

;we will use the stack to save contents of CX and AX.

MOV DL,AH

XCHG AX,[4502]

XCHG CX,[4504]

CALL PRINT\_DEC ;printing AH (the first part of the average)

XCHG AX,[4502]

MOV DL,AL

XCHG AX,[4502]

CALL PRINT\_DEC ;printing AL (the second part of the average)

MOV DL,10 ;new line

MOV AH,02H

INT 21H

XCHG CX,[4504]

MOV DL,CL ;printing the smallest number

XCHG CX,[4504]

CALL PRINT\_DEC

XCHG CX,[4504]

MOV DL,32 ;space

MOV AH,02H

INT 21H

MOV DL,CH ;printing the largest number

XCHG CX,[4504]

CALL PRINT\_DEC

HLT

PRINT\_DEC: ;since the number is 8-bits, it will have at most 3 decimal digits. so we will need at most 3 addresses to save the number

MOV BX,3

MOV [BX],'$'

MOV AL,DL

MOV CL,DL

MOV DL,3

LAB4: DEC BX

DEC DL

MOV CH,10

MOV AH,0

DIV CH

ADD AH,48

MOV [BX],AH

CMP AL,0

JA LAB4

MOV AH,09H

INT 21H

MOV DL,CL

RET

**Άσκηση 3:**

Το ζητούμενο πρόγραμμα είναι:

; You may customize this and other start-up templates;

; The location of this template is c:\emu8086\inc\0\_com\_template.txt

org 100h

MOV AH,08H ;asking for first digit

INT 21H

MOV CH,AL ;saving first digit in CH

INT 21H ;asking for second digit

MOV CL,AL ;saving second digit in CL

INT 21H ;asking for third digit (first of second number)

MOV DH, AL ;saving third digit in DH

INT 21H ;asking for fourth digit (second of second number)

MOV DL, AL ;saving fourth digit in DL

;we will firstly print the two numbers

MOV AH,09H

MOV BX,0 ;annuling BX

MOV [BX],'x'

INC BX

MOV [BX],'='

INC BX

MOV [BX],CH

INC BX

MOV [BX],CL

INC BX

MOV [BX],' '

INC BX

MOV [BX],'y'

INC BX

MOV [BX],'='

INC BX

MOV [BX],DH

INC BX

MOV [BX],DL

INC BX

MOV [BX],10 ;change line

INC BX

MOV [BX],'$'

CMP CH,57

JLE LAB0

CMP CH,70

JLE LAB1

SUB CH,87

JMP LAB2

LAB0: SUB CH,48

JMP LAB2

LAB1: SUB CH,55

LAB2: CMP CL,57

JLE LAB3

CMP CL,70

JLE LAB4

SUB CL,87

JMP LAB5

LAB3: SUB CL,48

JMP LAB5

LAB4: SUB CL,55

LAB5: CMP DH,57

JLE LAB6

CMP DH,70

JLE LAB7

SUB DH,87

JMP LAB8

LAB6: SUB DH,48

JMP LAB8

LAB7: SUB DH,55

LAB8: CMP DL,57

JLE LAB9

CMP DL,70

JLE LAB10

SUB DL,87

JMP LAB11

LAB9: SUB DL,48

JMP LAB11

LAB10: SUB DL,55

LAB11: XCHG DX,[4502]

MOV DX,0

MOV AH,09H

INT 21H ;printing x=[CH][CL], y=[DH][DL]

XCHG DX,[4502]

;we will now print the sum and the difference of the two numbers in decimal form.

;we firstly need to convert each digit to its respective value.

;we will now save the first number in CH and the second number in CL

MOV AH,16

MOV AL,CH

MUL AH

ADD AL,CL ;now AL has the first number

MOV CH,AL ;now CH has the first number

MOV AH,16

MOV AL,DH

MUL AH

ADD AL,DL ;now AL has the second number

MOV CL,AL ;now CL has the second number

;CH and CL have first and second number.

;BX will keep the addresses in which the digits will be stored at

;SI will have the sum

;DI will have the difference

;AX and DX will be used for divisions, retrieving the digits

MOV BX,0

MOV AX,0

MOV SI,0

MOV DI,0 ;initializing the registers

MOV [BX],'x'

INC BX

MOV [BX],'+'

INC BX

MOV [BX],'y'

INC BX

MOV [BX],'='

INC BX

MOV AL,CH

ADD SI,AX

MOV AL,CL

ADD SI,AX ;now SI has the sum

MOV AX,SI

MOV DL,10

MOV DH,0

LAB13: DIV DL

INC DH ;DH will count how many digits the number has.

MOV AH,0

CMP AX,0

JG LAB13 ;we are done with storing the sum in the addresses, but it is stored in the wrong digit order.

;to store it normally, we will reverse the digits in these addresses.

SUB DH,1

ADD BL,DH

ADD DH,2

MOV AX,SI

MOV DL,10

LAB13A: DIV DL

ADD AH,48

MOV [BX],AH

DEC BX

MOV AH,0

CMP AX,0

JG LAB13A

ADD BL,DH

MOV [BX],' '

INC BX

MOV [BX],'x'

INC BX

MOV [BX],'-'

INC BX

MOV [BX],'y'

INC BX

MOV [BX],'='

INC BX

CMP CH,CL

JB LAB14

JMP LAB15

LAB14: MOV [BX],'-'

INC BX

MOV AL,CH ;making CH bigger than CL

MOV CH,CL

MOV CL,AL

LAB15: MOV AH,0

MOV AL,CH

MOV DI,AX

MOV AL,CL

SUB DI,AX

MOV AX,DI

MOV DL,10

MOV DH,0

LAB16: DIV DL

INC DH ;DH will count how many digits the number has.

MOV AH,0

CMP AX,0

JG LAB16 ;we are done with storing the sum in the addresses, but it is stored in the wrong digit order.

;to store it normally, we will reverse the digits in these addresses.

SUB DH,1

ADD BL,DH

ADD DH,2

MOV AX,DI

MOV DL,10

LAB16A: DIV DL

ADD AH,48

MOV [BX],AH

DEC BX

MOV AH,0

CMP AX,0

JG LAB16A

ADD BL,DH

MOV [BX],'$'

MOV DX,0

MOV AH,09H

INT 21H

HLT

**Άσκηση 4**

Το πρόγραμμα assembly που ικανοποιεί τις προδιαγραφές που δίνονται από την εκφώνηση είναι το παρακάτω:

include 'C:\EMU8086\inc\emu8086.inc'

;Multi-segment executable file template

print\_str macro string

push dx

push ax

mov dx,offset string

mov ah,9

int 21h

pop ax

pop dx

endm

print macro char

push dx

push ax

mov dl,char

mov ah,2

int 21h

pop ax

pop dx

endm

;Add your data here

data segment

input db 17 dup(?)

new\_line db 0dh,0ah,'$'

err db "ERR",'$'

data ends

stack segment

stack ends

;Add your code here

code segment

initiliaze:

;Set segment registers

mov ax,data

mov ds,ax

mov es,ax

begin:

mov di,offset input

cld ;df = 0

mov cx,16 ;Counter, DONT change it so that loop instruction can be used

begin\_again:

call read\_key

cmp al,0dh ;ASCII code for 'Enter'

je end\_execution

stosb ;Store in memory

loop begin\_again

;Data processing

begin\_numbers:

print\_str new\_line

cld ;df = 0

mov si,offset input

mov cx,17

begin\_numbers\_again:

dec cx

cmp cx,0

je print\_pavla

lodsb

cmp al,48 ;AL>=48 <=> AL>=ASCII code for 0

jl begin\_numbers\_again ;No?Continue

cmp al,57 ;AL<=57 <=> AL<=ASCII code for 9

jg begin\_numbers\_again ;No?Continue

print al

jmp begin\_numbers\_again

print\_pavla:

print "-"

begin\_letters:

cld ;df = 0

mov si,offset input

mov cx,17

begin\_letters\_again:

dec cx

cmp cx,0

je print\_end

lodsb

cmp al,65 ;AL>=65 <=> AL>=ASCII code for 'A'

jl begin\_letters\_again ;No? Continue

cmp al,90 ;AL<=90 <=> AL<=ASCII code for 'Z'

jg begin\_letters\_again ;No?Continue

call decapitalize ;Decapitize first and then print

print al

jmp begin\_letters\_again

print\_end:

print\_str new\_line

print\_str new\_line

jmp initiliaze

error:

print\_str new\_line

print\_str err

end\_execution:

mov ax,4c00h ;Exit to operating system

int 21h

;This routine deletes the last character pressed

delete\_char proc near

push ax

mov ah,0eh ;This function displayes a character on the screen, advancing the cursor and scrolling

;the screen as necessary

mov al,8 ;ASCII code for backspace

int 10h

mov al,32 ;ASCII code for space

int 10h

mov al,8 ;ASCII code for backspace

int 10h

pop ax

ret

delete\_char endp

;This routine reads a valid digit (0-9) or capitalized letters (A-Z)

read\_key proc near

again\_read\_key:

;Read the digit

mov ah,1

int 21h

cmp al,0dh ;ASCII code for 'Enter'

je exit\_read\_key

cmp al,48 ;AL>=48 <=> AL>=ASCII code for 0

jl wrong\_read\_key ;No?Delete character and read again

cmp al,57 ;AL<=57 <=> AL<=ASCII code for 9

jg check\_letter\_read\_key ;No? Check if it is a letter form 'A' to 'Z'

jmp exit\_read\_key

check\_letter\_read\_key:

cmp al,65 ;AL>=65 <=> AL>=ASCII code for 'A'

jl wrong\_read\_key ;No?Delete character and read again

cmp al,90 ;AL<=90 <=> AL<=ASCII code for 'Z'

jg wrong\_read\_key ;No?Delete character and read again

jmp exit\_read\_key

wrong\_read\_key:

call delete\_char

jmp again\_read\_key

exit\_read\_key:

ret

read\_key endp

decapitalize proc near

add al,32 ;If it a capitalized letter, add ASCII code 32,

;to convert it in non - capitalized

exit\_decapitalize:

ret

decapitalize endp

code ends

end initiliaze ;Set entry point and stop the assembler

**Άσκηση 5**

Ζητείται πρόγραμμα assembly που θα γραφτεί σε υπολογιστή που βασίζεται στον μΕ80x86 και περιλαμβάνει σύστημα λήψης δεδομένων και θα πληρεί τις προδιαγραφές που φαίνονται στην εκφώνηση.

Το πρόγραμμα αυτό φαίνεται παρακάτω:

include 'C:\EMU8086\inc\emu8086.inc'

;Multi-segment executable file template

print\_str macro string

push dx

push ax

mov dx,offset string

mov ah,9

int 21h

pop ax

pop dx

endm

print macro char

push dx

push ax

mov dl,char

mov ah,2

int 21h

pop ax

pop dx

endm

;Add your data here

data segment

start\_msg db "START(Y/N):",'$'

input db "insert input: ",'$'

;The combination of the two moves the cursor to the beginning of the next row of the screen

;0ah=(10D), moves the cursor to the next row of the screen but maintaining the same column

;0dh=(13D) Carriage return, moves the cursor to the beginning of the current row

new\_line db 0dh,0ah,'$'

err db "ERR",'$'

data ends

stack segment

stack ends

;Add your code here

code segment

begin\_message:

;Set segment registers

mov ax,data

mov ds,ax

mov es,ax

print\_str start\_msg

begin\_decision:

mov ah,1 ;Get Y/y to begin the execution of the program

int 21h ;or N/n to stop the starting of a newexecution

;Below is described the management of each input hlt or continue

cmp al,78 ;ASCII code for N

je end\_execution

cmp al,110 ;ASCII code for n

je end\_execution

cmp al,89 ;ASCII code for Y

je start

cmp al,121 ;ASCII code for y

je start

call delete\_char ;If Y/y/N/y are not given, then erase the last character

;inserted (by calling the delete\_char proc) and start

;over the routine to read new int

jmp begin\_decision

start:

print\_str new\_line

;Read 3 HEX digits, they will be stored in bx reg

mov bx,0000h ;Position on the screen

;--------------------------1st MSB--------------------------

call read\_hex ;Read the 1st MSB digit

;(CHECK AGAIN after read\_hex)If N/n is pressed at any time, stop the execution

cmp al,78 ;ASCII code for N

je end\_execution

cmp al,110 ;ASCII code for n

je end\_execution

mov ah,0 ;Now ax = ah|al=0|(1st MSB)

add bx,ax ;bx = 000(1ST MSB)h

mov cl,4 ;Shifting the 1st MSB 4 bits left, to check the next MSB

shl bx,cl ;So bx = 00(1ST MSB)0h

;--------------------------2nd MSB--------------------------

call read\_hex ;Read the 2nd MSB digit

;(CHECK AGAIN after read\_hex)If N/n is pressed at any time, stop the execution

cmp al,78 ;ASCII code for N

je end\_execution

cmp al,110 ;ASCII code for n

je end\_execution

mov ah,0 ;Now ax = ah|al=0|(2nd MSB)

add bx,ax ;bx = 00(1ST MSB)(2nd MSB)h

mov cl,4 ;Shifting both MSBs 4 bits left, to check the last MSB

shl bx,cl ;So bx = 0(1ST MSB)(2nd MSB)0h

;--------------------------3rd MSB--------------------------

call read\_hex ;Read the 3rd MSB digit

;(CHECK AGAIN after read\_hex)If N/n is pressed at any time, stop the execution

cmp al,78 ;ASCII code for N

je end\_execution

cmp al,110 ;ASCII code for n

je end\_execution

mov ah,0 ;Now ax = ah|al=0|(3rd MSB)

add bx,ax ;bx = 0(1ST MSB)(2nd MSB)(3rd MSB)h

;-------------------------Volts--------------------------

;Calculate volts of the temperature given based on the graph

;V = (2/4095) \* AD == X.YYY ==> V = (2\*1000/4095)\*AD = XYYY

;So, volts are calculated with the precision of 3 decimals (multiplied by 1000)

mov ax,bx ;ax = 0(1ST MSB)(2nd MSB)(3rd MSB)h

mov cx,2000 ;cx = 7d0h

mul cx ;(ax' = ax \* 7d0h)h ==> (ax'= ax \* 2000)d

mov cx,4095 ;cx = 0fffh

div cx ;Finally I will get (ax'' = ax' / 0fffh)h, and more specifically

;V = (ax \* 2000)/4095

;Equation for the computation of the temperature: T = 500 \* a when V < 1.00

;T = 250 \* V + 250 when 1.00 < V < 1.80, T = 1500 \* a - 2000 when 1.80 < V < 2.00

;DON'T Forget: Volts are multiplied by 1000

cmp ax,1000

jl case\_1

cmp ax,1800

jl case\_2

jmp case\_3

;In all 3 cases, do the computations so that ax reg will be formed as: ax = XXXXY,

;precision of 1 decimal (or multiplied by 10)

;In linear equation, a = 5 and b = 0

case\_1:

mov cx,5

mov bx,0

jmp compute\_2

;In linear equation, a = 25 and b = 2500

case\_2:

mov cx,25

mov bx,2500

jmp compute\_1

case\_3:

mov cx,15

mov bx,20000

jmp compute\_2

compute\_1:

mul cx

mov cx,10 ;It was the only occasion that temprateure was formed as T = ax = XXXXYY

;so we cut the 2nd decimal by dividing with 10

div cx

add ax,bx

jmp print\_temperature

compute\_2:

mul cx

sub ax,bx

cmp ax,9999

jg error

;Print the corresponding temperature

print\_temperature:

mov dx,ax

print\_str new\_line

mov dx,0

mov cx,10000 ;Isolate thousands

div cx ;ax' = ax / 10000

mov bx,ax ;Print the quotient

call print\_digit

mov ax,dx ;ax = {again ax}

mov dx,0

mov cx,1000 ;Isolate hundreads

div cx ;ax' = ax/1000

mov bx,ax ;Print the quotient

call print\_digit

mov ax,dx ;ax = {again ax}

mov dx,0

mov cx,100 ;Isolate tens

div cx ;ax' = ax/100

mov bx,ax ;Print the quotient

call print\_digit

mov ax,dx ;ax = {again ax}

mov dx,0

mov cx,10 ;Isolate units

div cx ;ax' = ax/10

mov bx,ax ;Print the quotient

call print\_digit

print ","

;Print the temperature with 1 decimcal precision

mov bx,dx

call print\_digit

print " "

print "o"

print "C"

print\_str new\_line

print\_str new\_line

jmp begin\_message

error:

print\_str new\_line

print\_str err

jmp begin\_message

end\_execution:

mov ax,4c00h ;Exit to operating system

int 21h

;This routine deletes the last character pressed

delete\_char proc near

push ax

mov ah,0eh ;This function displayes a character on the screen, advancing the cursor and scrolling

;the screen as necessary

mov al,8 ;ASCII code for backspace

int 10h

mov al,32 ;ASCII code for space

int 10h

mov al,8 ;ASCII code for backspace

int 10h

pop ax

ret

delete\_char endp

;This routine reads a valid hexademical digit

read\_hex proc near

again\_read\_hex:

;Read the digit

mov ah,1

int 21h

cmp al,78 ;ASCII code for N

je exit\_read\_hex

cmp al,110 ;ASCII code for n

je exit\_read\_hex

cmp al,48 ;AL>=48 <=> AL>=ASCII code for 0

jl wrong\_read\_hex ;No?Delete character and read again

cmp al,57 ;AL<=57 <=> AL<=ASCII code for 9

jg check\_letter\_read\_hex ;No? Check if it is a letter form 'A' to 'F'

;Extract hexademical number

sub al,30h

jmp exit\_read\_hex

check\_letter\_read\_hex:

cmp al,65 ;AL>=65 <=> AL>=ASCII code for 'A'

jl wrong\_read\_hex ;No?Delete character and read again

cmp al,70 ;AL<=70 <=> AL<=ASCII code for 'F'

jg wrong\_read\_hex ;No?Delete character and read again

;Extract hexademical number

sub al,37h

jmp exit\_read\_hex

wrong\_read\_hex:

call delete\_char

jmp again\_read\_hex

exit\_read\_hex:

ret

read\_hex endp

;Add the proper hexademical number that was substracted during the initial reading

print\_digit proc near

cmp bl,9 ;bl=>9, then add 39h, since it is a hexademical letter {A,B,C,D,E,F}

jg add

add bl, 30h ;else add just 30h, since it a hexademical number

jmp print\_digit\_end

add:

add bl,37h

print\_digit\_end:

print bl

ret

print\_digit endp

code ends

end begin\_message ;Set entry point and stop the assembler