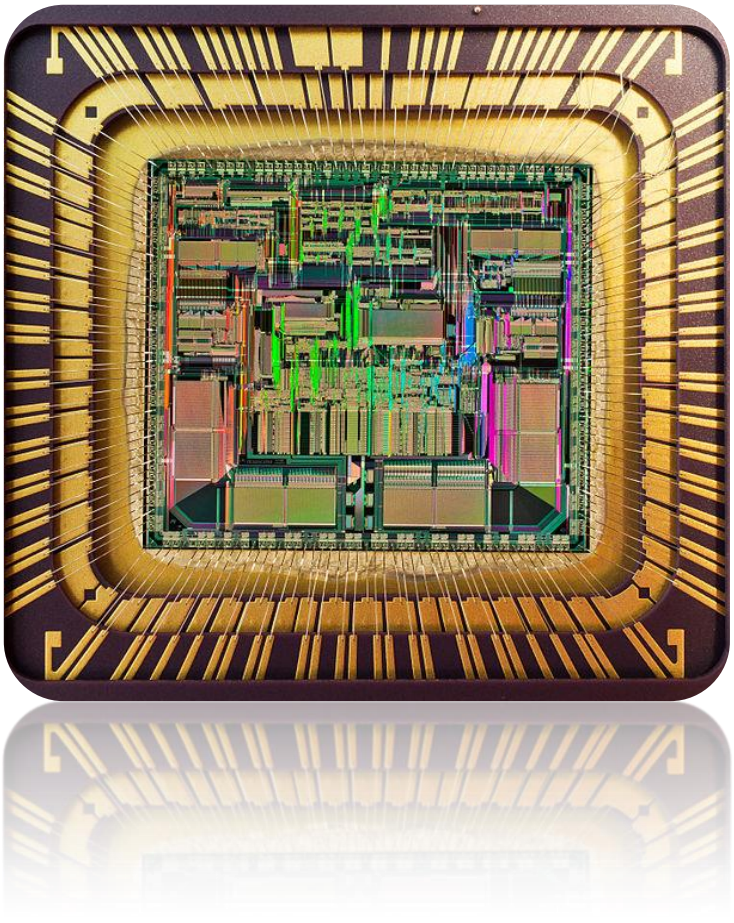
ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝΣΥΣΤΗΜΑΤΑ ΜΙΚΡΟΥΠΟΛΟΓΙΣΤΩΝ

5η Ομάδα Ασκήσεων

Βαβουλιώτης Γεώργιος
ΑΜ : 03112083
6ο Εξάμηνο

Γιαννόπουλος Αναστάσης
Α.Μ. : 03112176
6ο Εξάμηνο



Εφόσον είχαμε το δικαίωμα της επιλογής διαλέξαμε να υλοποιήσουμε τις άσκησεις 2,3,4 και 5 κάνοντας ουσιαστικά μια παραπάνω άσκηση από τις ζητούμενες. Οι υλοποιήσεις φαίνονται παρακάτω :

ΑΣΚΗΣΗ 2

Ο κώδικας σε assembly 8086 για την δεύτερη άσκηση φαίνεται παρακάτω:

```
name "exer2"
        PRINT macro message
        push ax             ;print a message on screen
        push dx
        mov dx, offset message
        mov ah, 9
        int 21h
        pop dx
        pop ax
        endm


begin:  mov dl,0
        PRINT firstmessage


        mov ah, 1           ; read 1st number
        int 21h

        cmp al,49           ;input have to be 4 digits
        jc wronginput1      ;read 4 times from keyboard
        cmp al,58           ;check if input is a digit
        jnc  wronginput1    ;if it's not change dl to 1 to show
                            ;that there is an error
        jmp correctinput1   ;is there are no errors dl stays at 0

wronginput1:
        mov dl,1

correctinput1:
        sub al,30h
        mov bh,10
        mul bh
        mov bh,al


        mov ah, 1
        int 21h

        cmp al,48
        jc wronginput2
        cmp al,58
        jnc  wronginput2
        jmp correctinput2
```

```
wronginput2:
        mov dl,1

correctinput2:
        sub al,30h
        add bh,al

        PRINT newline
        PRINT secondmessage

        mov ah, 1 ;read 2nd number
        int 21h

        cmp al,49
        jc wronginput3
        cmp al,58
        jnc  wronginput3
        jmp correctinput3

wronginput3:
        mov dl,1

correctinput3:
        sub al,30h
        mov bl,10
        mul bl
        mov bl,al

        mov ah, 1
        int 21h

        cmp al,48
        jc wronginput4
        cmp al,58
        jnc  wronginput4
        jmp correctinput4

wronginput4:
        mov dl,1

correctinput4:
        sub al,30h
        add bl,al

        PRINT newline
                            ;first number on bh
        PRINT x             ;second number on bl

        cmp dl,1            ;print the numbers we read
        je wronginput5

        mov al,bh
        call printdec
        jmp correctinput5
```

```
wronginput5:
        PRINT dash

correctinput5:
        PRINT y

        cmp dl,1
        je wronginput6
        mov al,bl
        call printdec
        jmp correctinput6

wronginput6:
        PRINT dash

correctinput6:

enternot:                   ;wait till enter
        mov ah,00h
        int 16h
        cmp al,0dh
        jne enternot

        cmp dl,1            ;check if input was correct
        je error            ;else display error message
                            ;and start again
        PRINT newline

        push bx
        mov bl,bh
        mov bh,0
        mov cx,bx
        pop bx
        push bx
        mov bh,0
        add cx,bx
        mov ax,cx
        pop bx

        PRINT xaddy

        mov ax,cx
        call printhex   ;Sum print


        PRINT xsuby

        push bx
        mov bl,bh
        mov bh,0
        mov cx,bx
        pop bx
        mov bh,0
        sub cx,bx
```

```
        mov ax,cx
        test ax,ax          ;AND for ax,ax
        jns posit           ;don't save the result.
        push ax             ;check the difference
        mov al, '-'
        mov ah,0eh
        int 10h
        pop ax
        neg ax

posit:  cmp ax,0
        je zero             ;if difference is 0
        call printhex       ;print it
        jmp zero_not

zero:
        mov al,'0'          ;if difference is zero => print '0'
        mov ah,0eh
        int 10h

zero_not:
        PRINT newline
        jmp begin
        hlt


error:  PRINT errormsg      ;error message and start again
        jmp begin

        firstmessage db "enter first number:  $"
        secondmessage db "enter second number:  $"
        newline db 0Dh,0Ah, "$"
        x db "x=$"
        y db "  y=$"
        xaddy db "x+y=$"
        xsuby db "  x-y=$"
        errormsg db 0Dh,0Ah,"WRONG INPUT",0Dh,0Ah,"$"
        dash db "-$"


        printhex proc
        push dx         ;show result on screen
        push cx
        mov dx,0

        cmp ax,0h       ;the result is hexadecimal
        je end1
        mov cx,10h
        div cx
        push dx
        call printhex
        pop dx
        mov ax,dx
        cmp ax,0ah
```

```
        jnc notnumber
        add ax,48
        jmp number

notnumber:
        add ax,55

number:
        mov ah,0eh
        int 10h

end1:   pop cx
        pop dx
        ret

        printdec proc
        push dx          ;show result on screen
        push cx
        push ax
        mov ah,0         ;the result is decimal
        mov dx,0
        cmp al,0h
        je procdec1
        mov cx,10
        div cx
        push dx
        call printdec
        pop dx
        mov al,dl
        cmp al,0ah
        jnc notdecnumber
        add al,48
        jmp decnumber

notdecnumber:
        add al,55

decnumber:
        mov ah,0eh
        int 10h

procdec1:
        pop ax
        pop cx
        pop dx
        ret

        endp
```

ΑΣΚΗΣΗ 3

Ο κώδικας σε assembly 8086 για την τρίτη άσκηση φαίνεται παρακάτω:

```
name "exer3"
        PRINT macro message
        push ax             ;prints a message on screen
        push dx
        mov dx, offset message
        mov ah, 9
        int 21h
        pop dx
        pop ax
        endm

read:   PRINT input

readagain:                  ;all routines are called here
        call readdec
        cmp al,0            ;if first digit is zero
        je readagain        ;if is xero we have to read it again
        mov cl,al
        call readdec
        mov dl,10
        mov bl,al
        mov al,cl
        mul dl
        add bl,al

        PRINT newline
        call printdec
        PRINT equal
        call printhex
        PRINT equal
        call printoct
        PRINT equal
        call printbin
        PRINT newline
        jmp read:
        hlt


        newline db 0Dh,0Ah, "$"
        equal db " = $"
        input db "Insert a valid input: $"


        erase proc      ;erase the last printed char
        push ax
        mov ah,0eh
        mov al,8
        int 10h
        mov al,32
        int 10h
```

```
        mov al,8
        int 10h
        pop ax
        ret

        printdec proc
        push dx         ;shows a number on screen
        push cx
        push bx
        push ax    ;the result is decimal  because the base is 10

        mov bh,0
        mov dx,0
        cmp bl,0h
        je  enddec
        mov cx,10
        mov ax,bx
        div cx
        mov bx,ax
        push dx
        call printdec
        pop dx
        mov bl,dl
        cmp bl,0ah
        jnc notdecnumber
        add bl,48
        jmp decnumber

notdecnumber:
        add bl,55

decnumber:
        mov al,bl
        mov ah,0eh
        int 10h

enddec:
        pop ax
        pop bx
        pop cx
        pop dx
        ret


        printhex proc
        push dx         ;shows a number on screen
        push cx
        push bx
        push ax         ;the result is hexadecimal(base is 16)
        mov bh,0
        mov dx,0
        cmp bl,0h
        je  endhex
        mov cx,10h
        mov ax,bx
        div cx
        mov bx,ax
```

```
        push dx
        call printhex
        pop dx
        mov bl,dl
        cmp bl,0ah
        jnc nothexnumber
        add bl,48
        jmp hexnumber

nothexnumber:
        add bl,55

hexnumber:
        mov al,bl
        mov ah,0eh
        int 10h

endhex:
        pop ax
        pop bx
        pop cx
        pop dx
        ret

        printoct proc
        push dx          ;shows a number on screen
        push cx
        push bx
        push ax          ;the result is octal because the base is 8
        mov bh,0
        mov dx,0
        cmp bl,0h
        je endoct
        mov cx,8
        mov ax,bx
        div cx
        mov bx,ax
        push dx
        call printoct
        pop dx
        mov bl,dl
        cmp bl,0ah
        jnc notoctnumber
        add bl,48
        jmp octnumber

notoctnumber:
        add bl,55

octnumber:
        mov al,bl
        mov ah,0eh
        int 10h
```

```
endoct: pop ax
        pop bx
        pop cx
        pop dx
        ret

        printbin proc
        push dx          ;shows a number on screen
        push cx
        push bx
        push ax          ;the result is binary because the base is2
        mov bh,0
        mov dx,0
        cmp bl,0h
        je  end_bin
        mov cx,2
        mov ax,bx
        div cx
        mov bx,ax
        push dx
        call printbin
        pop dx
        mov bl,dl
        cmp bl,0ah
        jnc notbinnumber
        add bl,48
        jmp binnumber

notbinnumber:
        add bl,55

binnumber:
        mov al,bl
        mov ah,0eh
        int 10h

end_bin:
        pop ax
        pop bx
        pop cx
        pop dx
        ret

        readdec proc     ;read a valid decimal

readagain1:              ;if input is not valid we read again
                         ;until we find a valid input
        mov ah, 1
        int 21h

        cmp al,48
        jc wronginput
        cmp al,58


        jnc  wronginput
        jmp correctinput
```

```
wronginput:
        call erase
        jmp readagain1

correctinput:
        sub al,30h
        ret

        endp
```

ΑΣΚΗΣΗ 4

Ο κώδικας σε assembly 8086 για την τέταρτη άσκηση φαίνεται παρακάτω:

```
name "exer4"
        PRINT macro message
        push ax             ;print a message on screen
        push dx
        mov dx, offset message
        mov ah, 9
        int 21h
        pop dx
        pop ax
        endm

begin:  mov cl,20
        mov dl,20
        mov bx,0800h

read:                       ;read all the valid inputs, store on
                            ;memory
        call readchar
        mov [bx],al
        inc bx
        loop read

        PRINT newline
        mov cl,dl
        mov bx,0800h

write:                      ;print the numbers from the memory
        mov al,[bx]         ;change letters before we print them
        inc bl
        cmp al,97
        jc nmbr
        cmp al,123
        jnc nmbr
        sub al,32

nmbr:   mov ah,0eh
        int 10h
        loop write

        PRINT newline
        jmp begin
```

```
end1:   hlt

        newline db 0Dh,0Ah, "$"

        erase_char proc     ;erase the last printed char
        push ax
        mov ah,0eh
        mov al,8
        int 10h
        mov al,32
        int 10h
        mov al,8
        int 10h
        pop ax
        ret

        readchar proc        ;read one char and store it at al if
                             ;it's valid
readagain:
        mov ah, 1            ;if it's not valid read again
        int 21h
        cmp al,61            ;if input is =   terminate program
        je end1
        cmp al,13            ;if input is ENTER =>  stop reading
        jne enternot         ;make cl=1 and store at dl the number
                             ;of chars that we read
        mov dl,20
        sub dl,cl
        mov cl,1
        cmp dl,0             ;check if enter pressed
        jne norestart
        PRINT newline
        jmp begin

norestart:
        jmp correctinpt

enternot:
        cmp al,48
        jc wronginpt
        cmp al,58
        jnc wrongnew
        jmp correctinpt


wrongnew:
        cmp al,97
        jc wronginpt
        cmp al,123

        jnc wronginpt
        jmp correctinpt

wronginpt:
        call erase_char     ;erase invalid char
        jmp readagain
```

```
correctinpt:
        ret

        endp
```

ΑΣΚΗΣΗ 5

Ο κώδικας σε assembly 8086 για την πέμπτη άσκηση φαίνεται παρακάτω :

```
name "exer5"
        PRINT macro message
        push ax             ;print a message on screen
        push dx
        mov dx, offset message
        mov ah, 9
        int 21h
        pop dx
        pop ax
        endm

        PRINT startornot

user_decision:
        mov ah, 1            ;get (y) to start or (n) to terminate
        int 21h
        cmp al,78            ;this is N
        je end1
        cmp al,110           ;this is n
        je end1
        cmp al,89            ;this is Y
        je start1
        cmp al,121           ;this is y
        je start1
        call erasechar
        jmp user_decision


start1: PRINT newline


restart:
        mov ch,0
        mov dx,0
        PRINT input
        call readhex        ;3 hexadecimal digits=> call read_hex
                            ;3 times
        mov bl,al
        call readhex
        mov bh,al
        call readhex
        mov cl,al

        mov dh,bl           ;place all digits from bl,bh
```

```
        mov ax,10h          ;cl to dx as hexadecimal
        mul bh
        add dx,ax
        add dx,cx

        cmp dx,0BB8h
        jnc greaterthan3
        mov ax,5            ;input x , output,  y=5/3*x from 0 to
                            ;3 volts
        mul dx
        mov bx,3
        div bx
        jmp printres

greaterthan3:
        mov ax,5            ;input x , output,  y=5*x-1000 from 3
                            ;to 4 volts
        mul dx
        sub ax,10000

printres:
        cmp ax,270Fh        ;temperature 999.9d
        jnc error
        PRINT newline       ;integral part is printed
        mov bx,10
        mov dx,0
        div bx
        cmp ax,0
        je zero
        call printdex
        jmp decml

zero:   mov ah,0eh
        mov al,'0'
        int 10h

decml:  mov ah,0eh
        mov al,'.'
        int 10h


        mov ax,dx           ;print decimal part
        cmp ax,0
        je zeronew
        call printdex
        jmp restartnew

zeronew:
        mov ah,0eh
        mov al,'0'
        int 10h

restartnew:
        PRINT newline      ;restart
        jmp restart

end1:   hlt
```

```
error:  PRINT newline
        PRINT errormsg
        jmp restart

        errormsg db "ERROR"
        newline db 0Dh,0Ah, "$"
        startornot db "START (Y, N):",0Dh,0Ah, "$"
        input db "insert input: $"

        erasechar proc    ;erase the last character pressed
        push ax
        mov ah,0eh
        mov al,8
        int 10h
        mov al,32
        int 10h
        mov al,8
        int 10h
        pop ax
        ret

        readhex proc       ;read a valid hexadecimal digit

readagain:                 ;if is not valid we read it again
        mov ah, 1
        int 21h
        cmp al,'n'
        je end1
        cmp al,'N'
        je end1
        cmp al,48
        jc wronginpt
        cmp al,58
        jnc  wrongnew
        jmp correctinpt


wronginpt:
        call erasechar     ;erase the invalid character
        jmp readagain

wrongnewt:
        cmp al,65
        jc wronginpt

        cmp al,103
        jnc wronginpt
        cmp al,71
        jnc wrongnew
        sub al,7
        jmp correctinpt

wrongnew:
        cmp al,97
        jc wronginpt
        sub al,39
```

```
correctinpt:
        sub al,30h          ;in al have the number pressed in
                            ;hexadecimal
        ret

        printdex proc
        push dx             ;show result on screen
        push cx
        mov dx,0
        cmp ax,0h           ;the result is decimal because the
                            ;base is 10
        je end2
        mov cx,10
        div cx
        push dx
        call printdex
        pop dx
        mov ax,dx
        cmp ax,0ah
        jnc nonumber
        add ax,48
        jmp yesnumber

nonumber:
        add ax,55

yesnumber:
        mov ah,0eh
        int 10h

end2:   pop cx
        pop dx
        ret

        endp
```