Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Συστήματα Μικροϋπολογιστών

Σειρά Ασκήσεων 5	Θανάσουλας Γρηγόριος | ΑΜ: 03114131 Μόνου Σταματίνα | ΑΜ: 03114077

Άσκηση 1

```
; multi-segment executable file template.

print macro char
    push dx
    mov dl, char
    mov ah, 2
    int 21h
    pop dx
endm

; Πλήθος αριθμών στη μνήμη
N EQU 8

data segment
    ; add your data here!
    ADDRN dw N dup(?)
ends

stack segment
ends

code segment

start:
; set segment registers:
    mov ax, data
    mov ds, ax
    mov es, ax

    ; add your code here

    ; Φόρτωση μνήμης με σσυνεχόμενους αριθμούς
    mov cl, N
    cld                     ; df = 0
    mov di, OFFSET ADDRN
```

```
    mov ax, 2
write_again:
    stosw
    inc ax                  ; Φόρτωση επόμενου δεδομένου
    loop write_again


    ; Άθροισμα αριθμών σε ψευδο-32 bit καταχωρητή
    ; =>>>> 32 bit: dX:bX

    mov cl, N
    cld                     ; df = 0
    mov si, OFFSET ADDRN
    mov dx, 0               ; Αρχικοποίηση καταχωητών αθροίσματος
    mov bx, 0
load_again:
    lodsw                   ; Φόρτωση στον καταχωρητή
    add bx, ax
    jnc no_overflow         ; Αν δεν έχουμε υπερχείληση
    inc dx                  ; dx <- dx + 1
no_overflow:
    loop load_again

    ; Διαίρεση dl:dh:ah:al/Διαρέτης = Πηλίκο στον dh:dl

    mov ax, bx              ; Μετακίνηση διαρέτέου
    mov bx, N               ; Διαιρέτης
    div bx                  ; Εκτέλεση διαίρεσης

    mov dl, ah
    call print_hex_full
    mov dl, al
    call print_hex_full


    mov cl, N
    cld                     ; df = 0
    mov si, OFFSET ADDRN
    mov dx, 0FFFFh           ; dx = local min
    mov bx, 0000H           ; bx = local max
load_again_2:
    lodsw                   ; Φόρτωση στον καταχωρητή
    ror ax, 1                  ; Μεταφορά στον καταχωρητή
    jc load_again_2         ; Αν Cy = 1 τότε περιττός, διάβασε τον
επόμενο
    rol ax, 1
local_min_calc:
    cmp ax, dx              ; current < local min ?
    ja local_max_calc       ; local min <-- current
    mov dx, ax
```

```
local_max_calc:
    cmp ax, bx              ; current > local max ?
    jb next
    mov bx, ax
next:
    loop load_again_2

     ; Παίρνει σε δυαδική μορφή ένα ψηφίο και εκτυπώνει τη δεκαεξαδική
του μορφή

    print " "
    print "m"
    print ":"
    push dx
    mov dl, dh
    call print_hex_full
    pop dx
    call print_hex_full

    print " "
    print "M"
    print ":"
    mov dl, bh               ; Εκτύπωση 8MSB
    call print_hex_full
    mov dl, bl               ; Εκτύπωση 8LSB
    call print_hex_full


    mov ax, 4c00h ; exit to operating system.
    int 21h
ends
  print_hex proc near
    cmp dl, 9
    jg addr1
    add dl, 30h
    jmp addr2
addr1:
    add dl, 37h
addr2:
    print dl
    ret
print_hex endp




print_hex_full  proc near
     push dx
     push ax
     push bx
```

```
     push dx        ; Σώσιμο του αποτελέσματος

     sar dx, 1
     sar dx, 1
     sar dx, 1
     sar dx, 1
     and dl, 0fh
     call print_hex

     pop dx
     and dl, 0fh    ; Μάσκα dl
     call print_hex
     pop bx
     pop ax
     pop dx
     ret
print_hex_full    endp

end start ; set entry point and stop the assembler.
```

Άσκηση 2

```
; multi-segment executable file template.


new_line macro
    print 0ah      ; ÍÝá ãñáììÞ
    print 0dh      ; Carriage return
endm

print_str macro string
    mov dx, offset string
    mov ah, 9
    int 21h
endm

print macro char
    mov dl, char
    mov ah, 2
    int 21h
endm


data segment
    ; add your data here!
    Z_MSG   db  "Z=",'$'
```

```
    W_MSG   db  "W=",'$'
    ADD_MSG   db  "Z+W=",'$'
    SUB_MSG   db  "Z-W=",'$'
    ERR       db "WRONG INPUT! TRY AGAIN!",'$'
ends

stack segment
    dw   128  dup(0)
ends

code segment
start:
; set segment registers:
    mov ax, data
    mov ds, ax
    mov es, ax

    ; add your code here

main proc far

    mov cl, 00h;
loop:
    call read_num
    cmp al, 0dh     ; Check if enter was hit
    je validation
    mov ah, 00h
    push ax
    inc cl          ; cl++
    jmp loop

validation:         ; Checks if 4 valid numbers were enetered previously
    new_line
    cmp cl, 04h
    jz  print_results


print_error:
    new_line
    print_str ERR
    new_line
    jmp start

print_results:
     print_str W_MSG
     pop bx         ;2ï øçößï ôïõ W
     pop cx         ;1o øçößï ôïõ w
     mov dl, cl
     call print_bin
     mov dl, bl
```

```
call print_bin
print " "
mov ch, 0ah    ; Ðïëëáðëáóéáóìüò ãéá äåêÜäò
mov al, cl

mul ch         ; al = 10 * 1o øçößï ôïõ w
add al, bl     ; al = áîßá ôïõ w óå äõáäéêÞ ìïñöÞ



pop bx         ; ÁíÜêôçóç 2ïõ øçößïõ z
pop cx         ; ¢íÜêôçóç 1ïõ øçößïõ z
push ax         ; ÓþóéìÏ ôïõ w


print_str Z_MSG
mov dl, cl
call print_bin
mov dl, bl
call print_bin
mov ch, 0ah    ; Ðïëëáðëáóéáóìüò ãéá äåêÜäò
mov al, cl
mul ch         ; al = 10 * 1o øçößï ôïõ z
add al, bl     ; al = z



push ax
new_line
print_str ADD_MSG
;O z âñßóêåôáé óôï al åíþ ï w óôïí bl
pop ax
pop bx


mov dl, al
add dl, bl     ; dl = al + bl

push ax
push bx
call print_hex_full
print " "
print_str SUB_MSG

pop bx
pop ax

mov dl, al
sub dl, bl    ; dl = al - bl
jns print_abs ; Åêôýðùóç èåôéêÞò äéáöïñÜò
```

```
     neg dl
     push dx
     print "-"
     pop dx
print_abs:
     call print_hex_full
     new_line

jmp start

main endp



; Reads a valid number character and stores its value in register AL
; On enter hit returns it's ASCI value 0dh
read_num proc near

    ; Reads key asci code in al
    mov ah, 01h
    int 21h

    cmp al, 0dh     ; Check if enter key
    jne check_num
    ret

check_num:          ; Check if number character was entered (30h ~ 39h)
    cmp al, 30h
    jl read_num

    cmp al, 39h
    jg read_num

    ; Extract its decimal value
    sub al, 30h

    ret
read_num endp


; Prints the ascii character
print_asci proc near
    mov ax, 4c00h ; exit to operating system.
    int 21h
ends

; Ðáßñíåé óå äõáäéêÞ ìïñöÞ Ýíá øçößï êáé åêôõðíåé ôç äåêáåîáäéêÞ ôïõ
ìïñöÞ
print_hex proc near
```

```
    cmp dl, 9
    jg addr1
    add dl, 30h
    jmp addr2
addr1:
    add dl, 37h
addr2:
    print dl
    ret
print_hex endp

print_bin proc near
    cmp dl, 9
    jg b_addr1
    add dl, 30h
    jmp b_addr2
b_addr1:
    add dl, 37h
b_addr2:
    print dl
    ret
print_bin endp


print_dec proc near
    add dl, 30h
    print dl
    ret
print_dec endp


print_hex_full  proc near
     push ax
     push bx

     push dx         ; Óþóéìï ôïõ áðïôåëÝóìáôïò

     sar dx, 1
     sar dx, 1
     sar dx, 1
     sar dx, 1
     and dl, 0fh
     call print_hex

     pop dx
     and dl, 0fh    ; ÌÜóêá dl
     call print_hex
     pop bx
     pop ax
     ret
```

```
print_hex_full    endp

end start ; set entry point and stop the assembler.
```

Άσκηση 3

```
; multi-segment executable file template.

print macro char
    push dx
    push ax

    mov dl, char
    mov ah, 2
    int 21h

    pop ax
    pop dx
endm

new_line macro
    print 0ah      ;
    print 0dh      ; Carriage return
endm

data segment
    ; add your data here!
ends

stack segment
ends

code segment
start:
; set segment registers:
    mov ax, data
    mov ds, ax
    mov es, ax

    ; add your code here
```

```
    ; Διάβασμα πρώτου ψηφίου
    call hex_key
    ; Έλεγχος για τερματισμός προγράμματος
    ;mov al, 0fh;
    cmp al, 'T'
    je stop
    mov cl, 4
    mov bl, al
    shl bl, cl  ; Μετακίνηση 4 θέσεις δεξιά



    ; Δίάβασμα 2ου ψηφίου
    call hex_key
    cmp al, 'T'
    je stop
    add bl, al  ; Προσθήκη 4 τελευταίων ψηφίων

    print "H"
    print "="
    call print_dec
    print "D"
    print "="
    call print_oct
    print "o"
    print "="
    call print_bin
    print "b"
    new_line

    jmp start

; Δέχεται στον καταχωρρητή dl την αξία του ψηφίου που θα τυπωθεί
print_digit proc near
    cmp dl, 9
    jg addr1
    add dl, 30h
    jmp addr2
addr1:
    add dl, 37h
addr2:
    print dl
    ret
print_digit endp


PRINT_DEC proc near
     push dx
     push cx
     push ax
```

```
    mov ah, 00h
    mov al, bl
    mov cl, 100    ; Απομόνωση εκατοντάδων
    div cl         ; Διαίρεση με το 100
    mov dl, al     ; Εκτύπωση πηλίκο
    call print_digit

    mov cl, 10     ; Απομόνωση δεκάδων
    mov al, ah     ; Μετακίνηση πηλίκου ως νέου αριθμού
    mov ah, 0
    div cl
    mov dl, al
    call print_digit

    mov dl, ah     ; Εκτύπωση μονάδων
    call print_digit

    pop ax
    pop cx
    pop dx
    ret

PRINT_DEC endp

PRINT_OCT proc near
    push dx
    push cx

    mov cl, 6
    mov dl, bl     ; Εκτύπωση 1ου ψηφίου
    sar dl, cl
    and dl, 03h
    call print_digit


    mov cl, 3
    mov dl, bl     ; Εκτύπωση 2ου ψηφίου
    sar dl, cl
    and dl, 07h
    call print_digit

    mov dl, bl     ; Εκτύπωση 3ου ψηφίου
    and dl, 07h
    call print_digit

    pop cx
    pop dx
    ret
```

```
PRINT_OCT endp

PRINT_BIN proc near
    push bx
    push cx

    mov cx, 8      ; Loop 8 φορές
again:
    rol bl, 1
    jc print1
    mov dl, 00h    ; Εκτύπωση 0
    call print_digit
    LOOP again
    pop cx
    pop bx
    ret
print1:
    mov dl, 01h     ; Εκτύπωση 1
    call print_digit
    LOOP again
    pop cx
    pop bx
    ret
PRINT_BIN endp

hex_key proc near
ignore:
    ; Read
    mov ah, 1
    int 21h

    ; Check if exit key pressed
    cmp al, 'T'
    je exit

    cmp al, 30h
    jl ignore

    cmp al, 39h
    jg addr_1

    ; Extract number
    sub al, 30h
    jmp exit

addr_1:
    cmp al, 'A'
    jl ignore
    cmp al, 'F'
    jg ignore
```

```
    sub al, 37h ; Extract number
exit:
    ret
hex_key endp


stop:
    mov ax, 4c00h ; exit to operating system.
    int 21h
ends
end start ; set entry point and stop the assembler.
```

Άσκηση 4

```
; multi-segment executable file template.

new_line macro
    print 0ah
    print 0dh      ; Carriage return
endm

print_str macro string
    mov dx, offset string
    mov ah, 9
    int 21h
endm

print macro char
    mov dl, char
    mov ah, 2
    int 21h
endm


data segment
    ; add your data here!
    input db 21 dup(?)
ends

stack segment

ends
```

```
code segment
start:
; set segment registers:
    mov ax, data
    mov ds, ax
    mov es, ax

    ; add your code here

    mov di, offset input
    cld             ; df = 0
    mov cx, 20      ; Μετρητής

again:
    call read_key   ; Ανάγνωση
    cmp al, '='     ; Πλήκτρο τερματισμού
    je stop
    cmp al, 0dh     ; Πλήκτρο enter
    je process
    stosb           ; Αποθήκευση στη μνήμη
    print al

    loop again

process:            ; Επεξεργασία δεδομένων
    new_line
    cld
    mov si, offset input
    mov cx, 20
again_1:
    lodsb
    call capitalize
    print al
    loop again_1



stop:
    mov ax, 4c00h   ; exit to operating system.
    int 21h
ends


capitalize proc near
    cmp al, 'a'
    jl cap_end
    cmp al, 'z'
    jg cap_end
    sub al, 32
cap_end:
```

```
    ret
capitalize endp

; Reads ascii codes of non capitalized letters and numbers
read_key proc near
ignore:
    ; Read
    mov ah, 8
    int 21h

    ; Check if exit key pressed
    cmp al, '='
    je exit

    ; Check if enter key was pressed
    cmp al, 0dh
    je exit

    cmp al, 30h
    jl ignore

    cmp al, 39h
    jg addr_1

    jmp exit

addr_1:
    cmp al, 'a'
    jl ignore
    cmp al, 'z'
    jg ignore
exit:
    ret
read_key endp


end start ; set entry point and stop the assembler.
```

Άσκηση 5

```
; multi-segment executable file template.


new_line macro
    print 0ah      ;
    print 0dh      ; Carriage return
```

```
endm

print_str macro string
    push dx
    push ax
    mov dx, offset string
    mov ah, 9
    int 21h
    pop dx
    pop ax
endm

data segment
    ; add your data here!
    START_MSG   db  "START(Y/N):",'$'
    ERR         db  "ERR",'$'
ends

print macro char
    push dx
    push ax
    mov dl, char
    mov ah, 2
    int 21h
    pop ax
    pop dx
endm


stack segment
ends

code segment
start:
; set segment registers:
    mov ax, data
    mov ds, ax
    mov es, ax

    ; add your code here
    print_str START_MSG
read_init:
    mov ah, 1
    int 21h
    cmp al, 'N'
    je stop
    cmp al, 'Y'
    jne read_init

    new_line
```

```
    ; Ανάγνωση 3 hex ψηφίων, τελικός αριθμός στο bx
    mov bx, 0000h
    call hex_key     ;Ανάγνωση 1ου ΄ψηφίου MSB
    cmp al, 'N'
    je stop
    mov ah, 0
    add bx, ax
    mov cl, 4
    shl bx, cl       ; Μετακίνηση


    call hex_key     ;Ανάγνωση 2ου ψηφίου MSB
    mov ah, 0
    cmp al, 'N'
    je stop
    add bx, ax
    mov cl, 4
    shl bx, cl       ; Μετακίνηση


    call hex_key     ;Ανάγνωση 3ου ψηφίου MSB
    mov ah, 0
    cmp al, 'N'
    je stop
    mov ah, 0
    add bx, ax

    ; Υπολογισμός τάσης απο τη γραφική
    ; V = 4/4095 * AD  === X.YY => V = 4*1000/4095 * AD = XYYY σε
δεκαδική μορφή

    mov ax, bx
    mov cx, 4000
    mul cx          ; ax = ax * 4000

    mov cx, 4095
    div cx          ; Τελικώς θα έχω ax = XXXXXY, δηλαδή την τάση με
ακρίβεια 2 δεκαδικών

    ; Η εξίσωση προσδιορισμού της θερμ. είναι θ = α * V - 1200, όπου α =
200 για V < 2.00 αλλιώς 800

    cmp ax, 2000
    jl case_1
    jmp case_2

case_1:
    mov cx, 2
    mov bx, 0
    jmp compute
```

```
case_2:
    mov cx, 8
    mov bx, 12000

compute:
    mul cx
    sub ax, bx

    cmp ax, 12000
    jg error


    ; Εκτύπωση αριθμού

     new_line
     mov cx, 10000    ; Απομόνωση xiliadων
     div cx          ; Διαίρεση με το 10000
     mov bx, ax      ; Εκτύπωση πηλίκο
     call print_digit

     mov ax, dx
     mov dx, 0
     mov cx, 1000    ; Απομόνωση εκατοντάδων
     div cx          ; Διαίρεση με το 1000
     mov bx, ax      ; Εκτύπωση πηλίκο
     call print_digit

     mov ax, dx
     mov dx, 0
     mov cx, 100     ; Απομόνωση δεκάδων
     div cx          ; Διαίρεση με το 100
     mov bx, ax      ; Εκτύπωση πηλίκο
     call print_digit

     mov ax, dx
     mov dx, 0
     mov cx, 10      ; Απομόνωση μονάδων
     div cx          ; Διαίρεση με το 100
     mov bx, ax      ; Εκτύπωση πηλίκο
     call print_digit

     print ","

     mov bx, dx      ; Εκτύπωση πηλίκο
     call print_digit
     print " "
     print "o"
     print "C"
     new_line
     new_line
```

```
    jmp start
error:
    new_line
    print_str ERR
     jmp start




stop:
    mov ax, 4c00h ; exit to operating system.
    int 21h
ends

hex_key proc near
ignore:
    ; Read
    mov ah, 1
    int 21h

    ; Check if exit key pressed
    cmp al, 'N'
    je exit

    cmp al, 30h
    jl ignore

    cmp al, 39h
    jg addr_1

    ; Extract number
    sub al, 30h
    jmp exit

addr_1:
    cmp al, 'A'
    jl ignore
    cmp al, 'F'
    jg ignore

    sub al, 37h ; Extract number
exit:
    ret
hex_key endp
```

```
; Δέχεται στον καταχωρρητή dl την αξία του ψηφίου που θα τυπωθεί
print_digit proc near
    cmp bl, 9
    jg addr1
    add bl, 30h
    jmp addr2
addr1:
    add bl, 37h
addr2:
    print bl
    ret
print_digit endp


PRINT_DEC proc near
     push dx
     push cx
     push ax


     mov ah, 00h
     mov al, bl
     mov cl, 100    ; Απομόνωση εκατοντάδων
     div cl         ; Διαίρεση με το 100
     mov dl, al     ; Εκτύπωση πηλίκο
     call print_digit

     mov cl, 10     ; Απομόνωση δεκάδων
     mov al, ah     ; Μετακίνηση πηλίκου ως νέου αριθμού
     mov ah, 0
     div cl
     mov dl, al
     call print_digit

     mov dl, ah     ; Εκτύπωση μονάδων
     call print_digit

     pop ax
     pop cx
     pop dx
     ret
PRINT_DEC endp

end start ; set entry point and stop the assembler.
```