Εργαστήριο Μικροϋπολογιστών

4η εργαστηριακή άσκηση

Τμήμα: Β Ομάδα: 15

Συνεργάτες: Μαρουφίδης Ιωάννης (03113506),
Περράκης Γεώργιος (03113511) ,
Σοφιανίδης Γεώργιος (03113179)

1η Άσκηση

```
data segment

   msg1 db "Give a 9-bit 2's complement number: $"
   msg2 db "Decimal: $"
   newline db 0ah,0dh,'$'
   pt1 db ".00$"
   pt2 db ".25$"
   pt3 db ".50$"
   pt4 db ".75$"

ends

stack segment
   dw   128  dup(0)
ends

code segment
start:
   mov ax, data
   mov ds, ax
```

```
    mov es, ax

    ; our code starts here
new_number:
    mov dx, offset msg1          ; here we print starting message
    mov ah, 9
    int 21h
    mov bl, 0                    ; number of accepted digits to register bl
    mov dh, 0                    ; and binary number to register dx
    mov dl, 0
read_loop:
    mov ah,8                     ; input is stored to register al
    int 21h
    cmp al, '0'                  ; we check if that is a binary digit
    je is_binary_digit
    cmp al, '1'
    je is_binary_digit
    cmp al, 'B'                  ; here we check if user want to terminate the programme
    jne read_loop                ; by giving the string "B15"
found_B:
    mov ah,8                     ; if we find a "B" character we read again
    int 21h
    cmp al, 'B'                  ; if we find again we re-read
    je found_B
    cmp al, '0'                  ; if we find a "0" after our "B" it is considered as
    je is_binary_digit           ; a part of our number
    cmp al, '1'
    jne read_loop                ; if we find anything else except "1" we ignore it
    inc bl                       ; if we find a "1" we increase number of accepted digits
    cmp bl, 1
    sub al,'0'
    je first                     ; here we add the accepted digit to register dx
```

```
    cmp bl, 9                ; by addition and then rotating the register right
    je before_finish
    add dl, al
    rol dl, 1
    jmp later
first:
    add dh, al
later:
    add al, '0'
    push dx
    mov dl,al                ; here we print the accepted digit (1)
    mov ah,2
    int 21h
    pop dx
    mov ah,8                 ; here we read next digit after having read the
    int 21h                  ; string "B1" already
    cmp al, 'B'              ; if its "B" again we re-read
    je found_B
    cmp al, '5'              ; if its "5" we terminate the programme
    je end_programme
    cmp al, '0'              ; if its "0" or "1" we accept it
    je is_binary_digit:
    cmp al, '1'
    je is_binary_digit:
    jmp read_loop            ; if its anything else we ignore it
before_finish:
    add al,'0'
    add dl, al
    rol dl, 1
    push dx                  ; here we print 9th accepted digit
    mov dl,al
    mov ah,2
```

```
    int 21h
    pop dx
    jmp finish
is_binary_digit:                ; we convert character to a number
    sub al,'0'                  ; by substracting its ascii code
    inc bl
    cmp bl, 1
    je first_digit
    add dl, al                  ; we place current digit to register dl
    cmp bl, 9h
    je print
    rol dl, 1                   ; and we rotate it left
    jmp print
first_digit:
    add dh, al                  ; first digit is placed in register dh
print:
    add al, '0'
    cmp bl, 8
    jne just_the_digit
    push dx
    push ax
    mov ah,2                    ; here we print decimal point
    mov dl, 2eh
    int 21h
    pop ax
    pop dx
just_the_digit:                 ; here we print accepted digit
    push dx
    mov dl,al
    mov ah,2
    int 21h
    pop dx
```

```
    cmp bl, 9
    je finish
    jmp read_loop
finish:
    push dx
    mov dx, offset newline        ; here we change line
    mov ah, 9
    int 21h
    mov dx, offset msg2           ; we print next message
    mov ah, 9
    int 21h
    pop dx
    push dx
    cmp dh, 1h                    ; here we determine sign and print it
    je negative
    mov dl, 2bh                   ; print '+'
    mov ah,2
    int 21h
    pop dx
    jmp separate_number
negative:
    mov dl, 2dh                   ; or print '-'
    mov ah,2
    int 21h
    pop dx
    not dx                        ; here we have the absolute value of our number
    add dx, 1h                    ; (if its negative)
    and dx, 0000000111111111b
separate_number:
    push dx
    mov cl, 2h
    shr dx, cl
```

```
    mov bl, dl                  ; result to register bl without fraction
    pop dx
    and dl, 03h
    mov bh, dl                  ; and fraction to register bh
    mov ah, 0h                  ; we set up registers for the division
    mov al, bl
    mov cl, 64h
    div cl                      ; here we find number of hundreds
    mov ch, 0h
    cmp al, 0h
    je no_of_tens
    mov ch, 1                   ; if number was greater than 99 ch=1 (flag)
    add al, '0'
    mov dl, al
    push ax                     ; we print number of hundreds if
    mov ah, 2h                  ; that is greater than 0
    int 21h
    pop ax
no_of_tens:
    mov al, ah
    mov ah, 0h
    mov cl, 10d
    div cl
    cmp al, 0h
    je check_if_printed
print_ten:
    add al, '0'
    mov dl, al
    push ax                     ; we print number of tens if
    mov ah, 2h                  ; that is greater than 0
    int 21h
    pop ax
```

```
    jmp no_of_ones
check_if_printed:               ; if that is zero we check if we have printed hundreds
    cmp ch, 1
    jne no_of_ones              ; we didnt
    jmp print_ten               ; or we did so we go back to print
no_of_ones:
    add ah, '0'
    mov dl, ah
    mov ah, 2h
    int 21h
                                ; here we print fraction
    cmp bh, 00000000b
    je prt1
    cmp bh, 00000001b
    je prt2
    cmp bh, 00000010b
    je prt3
    mov dx, offset pt4          ; case .75
    mov ah, 9
    int 21h
    jmp ending
prt1:
    mov dx, offset pt1          ; case .00
    mov ah, 9
    int 21h
    jmp ending
prt2:
    mov dx, offset pt2          ; case .25
    mov ah, 9
    int 21h
    jmp ending
prt3:
```

```
    mov dx, offset pt3          ; case .50
    mov ah, 9
    int 21h
ending:
    mov dx, offset newline      ; then we change line
    mov ah, 9
    int 21h
    jmp new_number

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

end_programme:

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

Στην παραπάνω άσκηση μετατρέπουμε έναν 9-ψήφιο δυαδικό αριθμό συμπληρωμένο ως προς δύο στον αντίστοιχο δεκαδικό. Η λειτουργία του προγράμματος τερματίζεται με την εισαγωγή των συνεχόμενων ψηφίων ‘B15’ δηλαδή της ομάδας μας ενδιάμεσα στα 9 δυαδικά ψηφία. Οποιαδήποτε μη δυαδικά ψηφία δοθούν από το χρήστη αγνοούνται. Το πρόγραμμα μας μετατρέπει τον δυαδικό αριθμό σε δεκαδικό και τον τυπώνει διαχωρίζοντας τα δύο μέρη του με την υποδιαστολή.

2η Άσκηση

```
data segment
    table db 3 dup(?)
    msg1 db 'GIVE 3 HEX DIGITS: $'
    msg2 db 'DECIMAL= $'
    newln db 0dh,0ah,'$'
    c1000 db 0
    c100 db 0
```

```
    c10 db 0
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
strt:
    mov c1000, 0h
    mov c100, 0h
    mov c10, 0h
    lea dx,msg1             ; print starting message
    mov ah, 9h
    int 21h
    mov si, 0h
    mov bx, 0h              ;given hex number to register bx
loop_input:
    mov ah, 8h
    int 21h
    cmp al, 'U'             ;if character 'U' is given stop the programme
    je exit
    cmp al, 0dh             ;if enter is given we check number of accepted digits
    je check_enter
    cmp al, 30h
```

```
    jb loop_input
    cmp al, 39h             ;if a number is given it's considered as valid
    jbe valid
    cmp al, 41h             ;if a capital letter between 'A' and 'F' is
    jb loop_input           ;given it's considered as valid
    cmp al, 46h
    jbe valid
    jmp loop_input          ;if anything else is given we ignore it
check_enter:
    cmp si, 3h              ;if enter is pressed before 3 accepted digits
    je convert              ;we ignore it else we print the accepted digits
    jmp loop_input
valid:
    cmp si, 3h
    je loop_input
    mov table:[si], al      ;if we find a valid digit we keep it in
    sub table:[si], 30h     ;a table and we print it
    cmp table:[si], 9h
    jbe number
    sub table:[si], 7h
number:
    inc si
    mov dl, al
    mov ah, 2
    int 21h
    jmp loop_input
convert:
    lea dx, newln           ;we print a new line
    mov ah, 9h
    int 21h
    lea dx, msg2            ;and second message
    mov ah, 9h
```

```
    int 21h
make_hex_number:
    mov al, 1h
    mul table:[2]
    mov bx, ax
    mov al, 10h
    mul table:[1]
    add bx, ax
    mov ax, 100h
    mov dl, table:[0]
    mov dh, 0h
    mul dx
    add bx, ax              ;hex number to register bx
    mov ax, bx
    mov dx, 0h
    mov bx, 1000d           ;find number of 1000d
    div bx
    cmp al, 0h
    je less_than_1000
    mov c1000, 1h
    push dx
    add al, 30h
    mov dl, al
    mov ah, 2h
    int 21h
    mov dl, 2ch
    mov ah, 2h
    int 21h
    pop dx
less_than_1000:
    mov ax, dx              ;find number of 100d
    mov bl, 100d
```

```
    div bl
    cmp al, 0h
    jne pt
    cmp c1000, 1h
    jne less_than_100
pt:
    mov c100, 1h
    push ax
    add al,30h
    mov dl, al
    mov ah, 2h
    int 21h
    pop ax
less_than_100:          ;find number of 10d
    mov al, ah
    mov ah, 0h
    mov bl, 10d
    div bl
    cmp al, 0h
    jne pt1
    cmp c1000, 1h
    je pt1
    cmp c100, 1h
    jne less_than_10
pt1:
    push ax
    add al, 30h
    mov dl, al
    mov ah, 2h
    int 21h
    pop ax
less_than_10:           ;and last number of 1d
```

```
    add ah, 30h             ;and print them all
    mov dl, ah
    mov ah, 2h
    int 21h

    lea dx, newln           ;we print a new line
    mov ah, 9h
    int 21h
    jmp strt
exit:

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

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

Στην παραπάνω άσκηση, ένας τριψήφιος δεκαεξαδικός αριθμός μετατρέπετε στον αντίστοιχο δεκαδικό. Κατά σύμβαση, το πρόγραμμα μας δέχεται και τυπώνει μόνο αριθμούς ή κεφαλαίους χαρακτήρες ‘A’ - ‘F’. Οποιαδήποτε άλλα στοιχεία εισόδου αγνοούνται ως επίσης και ο χαρακτήρας ‘enter’ αν δοθεί πριν 3 δεκτά δεκαεξαδικά ψηφία. Με την εισαγωγή του συγκεκριμένου χαρακτήρα, έπειτα από 3 έγκυρα δεκαεξαδικά , τυπώνουμε τον αντίστοιχο δεκαδικό με την υποδιαστολή αν χρειαστεί για το διαχωρισμό των χιλιάδων. Η μεγαλύτερη τιμή εισόδου που μπορεί να δοθεί είναι FFF δηλαδή θα πρέπει να τυπώσουμε 4,095. Αρχικά, αποθηκεύουμε τον αριθμό σαν χαρακτήρες σε έναν πίνακα και στη συνέχεια τον μετατρέπουμε σε δεκαεξαδικό με αφαιρέσεις και πολλαπλασιασμούς με την αντίστοιχη δύναμη του 16. Το αποτέλεσμα φυλάγεται σε έναν διπλό καταχωρητή των 16 bit από τον οποίο με συνεχείς διαιρέσεις με δυνάμεις του 10 σχηματίζουμε τον αντίστοιχο δεκαδικό. Το πρόγραμμα μπορεί να τερματιστεί κατά τη διάρκεια της εισαγωγής χαρακτήρων με τον ειδικό χαρακτήρα ‘U’.

3η Άσκηση

```
data segment
  table db 16 dup(?)
  min1 db ?
  min2 db ?
  msg db 'Give up to 16 characters: $'
  newln db 0dh,0ah,'$'
data ends



code segment
   assume cs:code,ds:data

main proc far
    mov ax,data
    mov ds,ax

start:
    lea dx,msg            ;type the message for input
    mov ah,9
    int 21h

    mov si,0



loop_input:
    mov ah,8
    int 21h

    cmp al,'*'            ;if '*' is given we exit
    je exit
    cmp al,0dh            ;if enter is pressed we move to the print of group
```

```
    je next

    cmp al,20h          ;space is accepted
    je valid

    cmp al,30h          ;check if character is a number
    jl loop_input
    cmp al,39h
    jng valid

    cmp al,41h          ;check if it is a capital letter
    jl loop_input
    cmp al,5ah
    jng valid

    cmp al,61h          ;check if it is a lowercase letter
    jl loop_input
    cmp al,7ah
    jg loop_input

valid:
    cmp si,16           ;if we have more than 16 characters ignore the character
    jge loop_input

    mov dl,al           ;else print it
    mov ah,2
    int 21h

    mov table:[si],al   ;and save it in the table
    inc si
    jmp loop_input      ;continue untill we have 16 characters or enter/'*' is pressed
```

```
next:
   mov cx,si            ;*** cx=si in every call of the procedure print_group *** (for the number
of loops)
   lea dx,newln
   mov ah,9             ;change line
   int 21h

   push cx
   mov bl,41h           ;parameters of the procedure print_group that define the group
   mov bh,5ah           ;first group is capital letters
   call print_group
   pop cx

   mov dl,'-'           ;type '-'
   mov ah,2
   int 21h

   push cx
   mov bl,61h           ;second group is lowercase letters
   mov bh,7ah
   call print_group
   pop cx

   mov dl,'-'           ;type '-'
   mov ah,2
   int 21h

   push cx
   mov bl,30h           ;third group is numbers
   mov bh,39h
   mov min1, 3ah        ;we set min1 and min2 as 3ah
   mov min2, 3ah        ;so they are greater than '9'=39h
   call print_group
```

```
    pop cx

    lea dx,newln          ;change line
    mov ah,09h
    int 21h

    cmp min1, 3ah
    je start

    mov si, 0h            ;now that we have min1 and min2
print_if_found:           ;we search them in table to print
    mov al, table:[si]    ;them correctly (as they were given)
    cmp al, min1
    jne n1
    mov dl,al
    mov ah,2
    int 21h
    mov min1, 3ah
n1:
    cmp al, min2
    jne n2
    mov dl,al
    mov ah,2
    int 21h
    mov min2, 3ah
n2:
    inc si
loop print_if_found

    lea dx,newln          ;change line
    mov ah,09h
    int 21h
```

```
    jmp start              ;continue for next input

exit:
    mov ax, 4c00h
    int 21h

main endp

print_group proc near
    mov si,0
next_char:
    mov al,table:[si]
    cmp al,bl              ;bl and bh are defined in the main programm (limits)
    jl nxt                 ;if current character doesn't belong in the current group
    cmp al,bh              ;continue with the next one
    jg nxt
;--------------------------
    cmp bl, 30h            ;here we will find the 2 smallest numbers (if they exist)
    jne prt                ;if we dont check numbers, we ignore this part
    cmp al, min1
    ja p1
    push cx
    mov cl, min1
    mov min2, cl
    pop cx
    mov min1, al
    jmp prt
p1:
    cmp al, min2
    ja prt
    mov min2, al
```

```
;-------------------------
prt:
   mov dl,al            ;if it does
   mov ah,2             ;type it
   int 21h

nxt: inc si             ;move to the next character

loop next_char

ret

print_group endp



code ends

end main
```

Στην τελευταία άσκηση, δεχόμαστε ως είσοδο αριθμούς, κεφαλαίους και πεζούς χαρακτήρες, το πολύ ως 16, τους διαχωρίζουμε σε ομάδες με μία παύλα ενδιάμεσα και τους τυπώνουμε μαζί με τους δύο μικρότερους αριθμούς αν αυτοί υπάρχουν με τη σειρά εισαγωγής τους. Το πρόγραμμα τερματίζει με την εισαγωγή του ειδικού χαρακτήρα '*'. Οι υπόλοιποι χαρακτήρες αγνοούνται. Οι έγκυροι χαρακτήρες αποθηκεύονται σε έναν πίνακα 16 στοιχείων τον οποίο διατρέχουμε 3 φορές τυπώνοντας κάθε φορά την αντίστοιχη ομάδα χαρακτήρων. Για την εμφάνιση των δύο μικρότερων αριθμών χρησιμοποιούμε δύο μεταβλητές στις οποίες τους αποθηκεύουμε αν υπάρχουν και τέλος διατρέχουμε μία τελευταία φορά τον πίνακα και όταν τους συναντήσουμε, τους τυπώνουμε έτσι ώστε να τους έχουμε και με τη σειρά εισαγωγής τους.