Σχολή Ηλεκτρολόγων Μηχανικών
&
Μηχανικών Υπολογιστών

Εργαστήριο Μικροϋπολογιστών

Τρίτη Εργαστηριακή Άσκηση

Λαμπράκος Χρήστος, Α.Μ: 03112062

Μανδηλαράς Νικηφόρος, Α.Μ: 03112012

Σπαθαράκης Δημήτριος, Α.Μ: 03113523

Έβδομο Εξάμηνο

Παραδοτέα: 22/11/2015

Άσκηση i (Μετατροπή δυαδικού σε δεκαδικό αριθμό)

Η λογική μας για την συγκεκριμένη άσκηση ήταν να διατηρούμε ένα βοηθητικό γινόμενο που κάθε στιγμή περιείχε την κατάλληλη δύναμη του δύο (από 2 ^ 7 έως 2 ^ 0). Ξεκινώντας από το MSB της εισόδου ανάλογα αν το ψηφίο ήταν «1» προσθέταμε την τιμή του βοηθητικού γινομένου στο τελικό μας αποτέλεσμα αλλιώς όχι. Τέλος ενημερώναμε το βοηθητικό γινόμενο διαιρώντας το περιεχόμενο του κατά δύο και είμαστε έτοιμοι να συνεχίσουμε με το επόμενο ψηφίο της εισόδου. Μ αυτό τον τρόπο περάσαμε από δυαδικό σε δεκαεξαδικό αριθμό. Για την μετατροπή δεξαεξαδικού σε δεκαδικό χρησιμοποιήσαμε των αλγόριθμο που έχουμε δει και στις προηγούμενες σειρές ασκήσεων και τυπώσαμε κατά σειρά στην οθόνη εκατοντάδες, δεκάδες, μονάδες καταλήγοντας έτσι στο τελικό αποτέλεσμα. Στα διαδικαστικά το πρόγραμμα δέχεται ως είσοδο μόνο μηδέν ή ένα αλλιώς ξαναζητά είσοδο, είναι συνεχούς λειτουργίας και τερματίζει μόνο με το "Q".

Testone

```
org 100h

.DATA
   MSG1 DB 'GIVE AN 8-BIT BINARY NUMBER: $'
   MSG2 DB 'DECIMAL: $'
   linefeed db 13, 10, "$"

.CODE
INIT:
   MOV DX,OFFSET MSG1  ;typwnw mhnuma sthn othonh
  MOV AH,09H
  INT 21H
  MOV CL,0                ; deikths
  MOV BL,128             ;voithitiko ginomeno
  MOV BH,0                ; arithmos/athroisma
  MOV CH,2

READ:
 MOV AH,00H
 INT 16H
 CMP AL,'Q'
 JE RETURN           ; an Q diakoptw
 CMP AL,30H          ;an oxi 0 tsekarw an einai 1
 JE ELE

UNO:
 CMP AL,31H
 JA READ              ;an oxi 1 janazhtaw eisodo
 ADD BH,BL  ;prosthetw to voithiko ginomeno sto athroisma

ELE:
 MOV DL,AL
 MOV AH,02H
 INT 21H
 MOV AH,0
 MOV AL,BL
 DIV CH         ;diairw me 2 to voithiko ginomeno
 MOV BL,AL
 INC CL                ; auxanw ton deikth
 CMP CL,8         ;elegxw an diavasa 8bit
 JB READ

PRINT:
 MOV AH, 09                  ;allazw grammh
 MOV DX, offset linefeed
 INT 21h           ;an nai typwnw mhnyma kai apotelesma
 MOV DX,OFFSET MSG2
```

```
 MOV AH,09H
 INT 21H
  CMP BH,200          ; compare with 200
 JB SM200
 SUB BH,200          ; AL-=200
 MOV AL,02H          ; ekatontades = 2
 JMP DECS
SM200:               ; <200
 CMP BH,64H          ; compare with 100
 JB SM100
 SUB BH,64H          ; AL-= 100
 MOV AL,01H          ; ekatontades = 1
 JMP DECS

SM100:
 MOV AL,00H          ; ekatontades = 0

DECS:
 MOV DL,AL
 ADD DL,30H
 MOV AH,02H
 INT 21H             ; restore from b
 MOV AL,00H          ; dekades = 0

LOOP0:
 CMP BH,0AH          ; compare with 10
 JB SM10
 INC AL               ;dekades ++
 SUB BH,0AH; AL -= 10
 JMP LOOP0

SM10:                   ;<10
 MOV DL,AL
 ADD DL,30H
 MOV AH,02H
 INT 21H
 MOV DL,BH
 ADD DL,30H
 MOV AH,02H
 INT 21H
 MOV AH, 09
 MOV DX, offset linefeed
 INT 21h

 JMP INIT          ; synexomenh leitoyrgia

RETURN:
  RET
  END
```

Άσκηση ii

Η υλοποίηση μας, περιμένει να δοθούν από το πληκτρολόγιο τουλάχιστον τέσσερις δεκαδικοί αριθμοί τους οποίους και αποθηκεύει στον καταχωρητή BX. Κατά τη διάρκεια της εισόδου δε δέχεται κανέναν άλλο χαρακτήρα και περιμένει το πάτημα του [Enter] αφού έχουν δοθεί τέσσερις αριθμοί για να προβεί στον υπολογισμό του δεκαεξαδικού αριθμού. Στη συνέχεια πολλαπλασιάζει το περιεχόμενο του καταχωρητή BX με 10 και απομονώνει 4 MSB ψηφία τα οποία και τυπώνει στην έξοδο. Το πρόγραμμα ζητάει επαναληπτικά είσοδο από τον χρήστη και τερματίζει όταν δοθεί ο αριθμός της ομάδας μας.

Testwo

```
data segment
msg1 db 0AH,0DH,"GIVE DECIMAL DIGITS:$"
msg2 db 0AH,0DH,"HEX=$"
ends
stack segment
dw 128 dup(0)
ends

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

start1:

ignore:

mov CX, 0h
mov dx,0

PRINT_STR msg1

tag1:
mov ax,10           ; polaplasise me 10
mul dx              ; ton mexri tora arithmo
mov dx,ax
mov BX, DX             ; Save DX at BX.

loopi:

READ

;cmp AL, 0Dh     ; 0Dh = 13d = chr(\n)
;    je CONT   ; Loop until [ENTER] is pressed.

cmp al,'0'
jl loopi
cmp al,'9'
jg LOOPI
cmp AL, 0Dh     ; 0Dh = 13d = chr(\n)
je CONT   ; Loop until [ENTER] is pressed.

;CMP CX, 4h
;Jl PAK
```

```
push ax
PRINT al
POP AX

mov AH, 0       ; Zero AX's high byte.
sub AX, 48      ; 48 = chr(0).
add DX, AX      ; Add AX to DX.
mov BX, DX      ; Save DX at BX.

cmp cx , 3h
jl pak

ENTERLOOP:
    READ            ; Awaits for [ENTER]
    cmp AL, 0Dh     ; 0Dh = 13d = chr(\n)
    je cont   ; Loop until [ENTER] is pressed.

pak:
ADD CX , 1h

JMP TAG1

cont:
 ;MOV AX,BX
;MOV BL,10
;DIV BL
;MOV BX,AX

MOV AL,00H
mov CX, 4       ; Set loop counter (4 digits).
PRINT_STR msg2

tupoma: ;fere t 4 prota bits se 8esh gia print

    rol BX, 1       ; Right-shift BX by one
nibble
    rol BX, 1       ; ...
    rol BX, 1       ; ...
    rol BX, 1       ; ...
    mov DX, BX      ; Save shifted number to DX.
        and DX, 000Fh   ; Mask 4 lowest bits of
DX.
    call PRINT_HEX

;49232  MAS KANEI C05 POU EINAI H OMADA MAS------

CMP AL,10H
JZ PAKTELOS

CMP DL, 'C'
JNZ PAK2
CMP AL,00H
JZ PAKADD

PAK2:
CMP DL, '0'
JNZ PAK3
CMP AL , 01H
JZ PAKADD

PAK3:
CMP DL , '5'
JNZ PAKTELOS
```

```
CMP DL, 02H
JZ  PAKTELOS:

 JMP TELOS
PAKTELOS:
MOV AL,10H
JMP LOOPA

PAKADD: ADD AL,1H

LOOPA:
loop tupoma
jmp start1
;-------------------------------------------------
PRINT_HEX proc near
    cmp DL, 9       ; DL <= 9?
    jle _ADD10      ; yes: jump to appropriate
fixing code.
    add DL, 37H     ; no : Prepare DL by adding
chr(A) - 10d = 37h
    jmp _HEX_OUT    ; ... and go to output stage
_ADD10:
    add DL, 30h     ; Prepare DL by adding chr(0)
= 30h
_HEX_OUT:
    PRINT DL        ; Print char to screen.
    ret             ; Return to caller.
PRINT_HEX endp
PRINT macro CHAR
 push AX ; Save AX on stack.
 push DX ; Save DX on stack.
 mov DL, CHAR ; Place char byte in DL
 mov AH, 2 ; Load DOS operation.
 int 21H ; Call DOS.
 pop DX ; Restore DX.
 pop AX
endm

PRINT_STR macro STRING
     push AX ; Save AX on stack.
     push DX ; Save DX on stack
     lea DX, STRING ; Load address of string @
DX
     mov AH, 9 ; Load DOS operation
     int 21H ; Call DOS.
     pop DX ; Restore DX.
     pop AX ; Restore AX.
endm

READ MACRO
    MOV AH,08H
    INT 21H
ENDM
 EXIT MACRO
    MOV AH,4CH
    INT 21H
ENDM

telos: exit
ends
end start ; set entry point and stop the
assembler.
```

Άσκηση iii

Το πρόγραμμα μας χωρίζεται σε 3 συναρτήσεις και είναι συνεχούς λειτουργίας. Η READ είναι η πρώτη που καλείται προκειμένου να πραγματοποιήσει το διάβασμα της εισόδου. Κάθε φορά ελέγχει το είδος του χαρακτήρα που δόθηκε και τον τοποθετεί στον κατάλληλο πίνακα (SMALL για μικρούς, NUMS για αριθμούς και CAPS για κεφαλαίους). Όταν τοποθετούμε ένα χαρακτήρα φροντίζουμε στους δύο πρώτους πίνακες να τοποθετούμε στην αμέσως επόμενη θέση ένα χαρακτήρα κενού ενώ στον τελευταίο πίνακα ένα χαρακτήρα αλλαγής γραμμής. Οι χαρακτήρες κενού αγνοούνται, καθώς και όσοι επιπλέον των 14 χαρακτήρων δοθούν. Με το χαρακτήρα « = » το πρόγραμμά μας τελειώνει. Η δεύτερη συνάρτηση PRINT αναλαμβάνει το τύπωμα στην οθόνη. Ξεκινάει και τυπώνει έναν έναν τους χαρακτήρες στον πίνακα SMALL μέχρι να βρει τον χαρακτήρα κενού τον οποίο τυπώνει επίσης. Το ίδιο ακριβώς συμβαίνει και με τον πίνακα NUMS ενώ στον πίνακα CAPS στην τελευταία θέση του πίνακα βρίσκεται η αλλαγή γραμμής. Αν σε κάποιο πίνακα δεν έχει δοθεί στοιχείο τότε δεν τυπώνουμε τον τερματικό χαρακτήρα και περνάμε στον επόμενο πίνακα. Γι αυτό το σκοπό χρησιμοποιούμε τον καταχωρητή CL ως σημαία. Τέλος η συνάρτηση MAXEN αποφαίνεται για τους δύο μεγαλύτερους εκ των αριθμών. Τοποθετεί αρχικά στον μεγαλύτερο τον πρώτο του πίνακα και ως δεύτερο μεγαλύτερο το κενό. Στην περίπτωση που δε δοθεί κάποιος αριθμός απλά δε θα τυπωθεί τίποτα γιατί τότε στην πρώτη θέση του πίνακα θα βρίσκεται κενός χαρακτήρας όπως αναφέραμε και προηγουμένως. Σε κάθε άλλη περίπτωση διασχίζει τον πίνακα και συγκρίνει κάθε αριθμό με το NMAX. Αν είναι μεγαλύτερο τότε αυτός ο NMAX τοποθετείται στον OMAX και ο νέος αριθμός στον NMAX. Σε αντίθετη περίπτωση γίνεται και η σύγκριση με τον OMAX και ανανεώνεται η τιμή αν προκύψει μεγαλύτερος αλλιώς παραμένουν ως έχουν. Σε κάθε περίπτωση στο τέλος τυπώνουμε τα περιεχόμενα και των δύο μεταβλητών, και κατόπιν η συνάρτηση επιστρέφει.

Testhree

```
D_SEG SEGMENT
INPUT DB 15 DUP(0)
CAPS  DB 15 DUP(?)
SMALL DB 15 DUP(?)
NUMSv DB 15 DUP(?)        ;here goes nothing
C1    DW 00               ;three counters
C2    DW 00
C3    DW 00
NUMS  DB 15 DUP(?)
NMAX  DB 00
OMAX  DB 00
D_SEG ENDS
;=================================================
C_SEG SEGMENT
      ASSUME CS:C_SEG,DS:D_SEG
MAIN PROC FAR
LOOPA:
      CALL READ
      CALL PRINT
      CALL MAXEN
      JMP LOOPA
MAIN ENDP
;=================================================
READ PROC NEAR     ;READ properly loads input at INPUT
      LEA BX,INPUT   ;array, also sets CAPS, SMALL and
      MOV CL,15           ;NUMS arrays
      MOV C1,0            ;initialize counters
      MOV C2,0
      MOV C3,0
LOOPB:
      MOV AH,07H          ;read next character (without
echo,
      INT 21H            ;just to be safe)
CHKSP:
      CMP AL,20H          ;was it space?
      JNZ CHKNUM          ;if not, check if it was a number
      JMP STRE            ;else store and proceed
CHKNUM:
```

```
      CMP AL,30H
      JGE UPPER
      CMP AL,0DH          ;only possible key is ENTER
      JZ DONE             ;if pressed, get out
      JMP LOOPB
UPPER:
      CMP AL,39H
      JG KEEP
      MOV DX,BX           ;found a number! save it!
      LEA BX,NUMS
      ADD BX,C1           ;align
      MOV [BX],AL
      MOV CH,20H
      INC BX
      MOV [BX],CH
      INC C1
      MOV BX,DX           ;retrieve pointer
      JMP STRE
KEEP:
      CMP AL,3DH          ;check for '='
      JZ KILL
      CMP AL,41H          ;check for caps...
      JL LOOPB
      CMP AL,5AH
      JG KEEP2
      MOV DX,BX           ;found a CAP! save it!
      LEA BX,CAPS
      ADD BX,C2           ;align
      MOV [BX],AL
      MOV CH,0AH
      INC BX
      MOV [BX],CH
      INC C2
      MOV BX,DX           ;retrieve pointer
      JMP STRE
KEEP2:
      CMP AL,61H          ;check for small...
      JL LOOPB
      CMP AL,7AH
      JG LOOPB            ;if nothing valid, wait for new
input
      MOV DX,BX           ;found a small! save it!
      LEA BX,SMALL
      ADD BX,C3           ;align
      MOV [BX],AL
      MOV CH,20H
      INC BX
      MOV [BX],CH
      INC C3
      MOV BX,DX           ;retrieve pointer
STRE:
      MOV [BX],AL         ;store char at input array
      DEC CL              ;update counter
      JZ DONE             ;also check if array is full
      INC BX              ;update pointer
      MOV AH,02H
      MOV DL,AL
      INT 21H             ;print VALID character
      JMP LOOPB
DONE:
      MOV AH,02H
      MOV DL,0AH          ;print a new line when finished
      INT 21H
      LEA BX,SMALL        ;just in case input does not contain
      MOV AL,20H          ;a certain kind of character (numbers
      ADD BX,C3           ;for example).
      MOV [BX],AL
      LEA BX,NUMS
      MOV AL,20H
      ADD BX,C1
      MOV [BX],AL
```

```
      LEA BX,CAPS
      MOV AL,0AH
      ADD BX,C2
      MOV [BX],AL
      RET
KILL:
      MOV AH,4CH          ;terminate program
      INT 21H
      RET
READ ENDP
;=============================================================
============
PRINT PROC NEAR
      MOV CL,00H          ;something like a flag
      LEA BX,SMALL
LOOPC:
      MOV DL,[BX]
      CMP DL,20H          ;we know array ends at 'SPACE'
      JZ NXT1
      INC CL              ;if at least one character has
      MOV AH,02H          ;been given, raise the flag
      INT 21H
      INC BX              ;point at next char
      JMP LOOPC
NXT1:
      CMP CL,00H          ;flag check (no point printing
      JZ B1               ;'SPACE' if no small char was
      MOV AH,02H          ;given)
      INT 21H
      MOV CL,00H
B1:
      LEA BX,NUMS         ;identical procedure for numbers..
LOOPv:
      MOV DL,[BX]
      CMP DL,20H
      JZ NXT2
      INC CL
      MOV AH,02H
      INT 21H
      INC BX
      JMP LOOPv
NXT2:
      CMP CL,00H
      JZ B2
      MOV AH,02H
      INT 21H
B2:
      LEA BX,CAPS         ;...and for the CAPS, except we
LOOPg:                   ;print a new line at array end
      MOV DL,[BX]
      CMP DL,0AH
      JZ COOL
      MOV AH,02H
      INT 21H
      INC BX
      JMP LOOPg
COOL:
      MOV DL,0AH
      MOV AH,02H
      INT 21H
      RET
PRINT ENDP
;=============================================================
MAXEN PROC NEAR
      LEA BX,NUMS         ;go at the numbers
      MOV DL,[BX]
      MOV NMAX,DL         ;define first number as max
      MOV OMAX,20H        ;the "old" max is nothing, for now
SCOUT:
      MOV DL,[BX]
      CMP DL,20H          ;finished?
```

```
      JZ LEAVES
      CMP DL,NMAX          ;if not, compare with new max
      JG NFOUND
      CMP DL,OMAX          ;...and with "old" max
      JG OFOUND
BACKIN:
      INC BX               ;point at next number
      JMP SCOUT
NFOUND:
      MOV DL,NMAX
      MOV OMAX,DL          ;give old max the new max's old value
      MOV DL,[BX]          ;update new max
      MOV NMAX,DL
      JMP BACKIN
OFOUND:
      MOV DL,[BX]
      CMP DL,NMAX        ;just in case there was only one number
      JZ BACKIN
      MOV DL,[BX]
      MOV OMAX,DL
      JMP BACKIN
LEAVES:
      MOV DL,NMAX          ;print the results
      MOV AH,02H
      INT 21H
      MOV DL,OMAX
      MOV AH,02H
      INT 21H
      MOV DL,0AH           ;also print a new line
      MOV AH,02H
      INT 21H
      RET
MAXEN ENDP
;=============================================================
C_SEG ENDS
END MAIN
```

ΤΕΛΟΣ ΕΡΓΑΣΙΑΣ