5η ΣΕΙΡΑ ΑΣΚΗΣΕΩΝ ΣΥΣΤΗΜΑΤΑ ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΩΝ

Αϊβαλής Θεόδωρος 03117099
Σαλιαράκης Παύλος 03117135

ΣΗΜΜΥ 12/7/2020

ΑΣΚΗΣΗ 1:

Θέλουμε να γραφούν τρεις ρουτίνες PRINT_DEC, PRINT_OCT και PRINT_BIN που να δέχονται μέσω του DL έναν 8-bit αριθμό και να τον τυπώνουν στην οθόνη ενός προσωπικού υπολογιστή σε δεκαδική, οκταδική και δυαδική μορφή αντίστοιχα. Στη συνέχεια γράφεται πρόγραμμα που διαβάζει από το πληκτρολόγιο έναν αριθμό δύο ψηφίων σε δεκαεξαδική μορφή χρησιμοποιώντας την ρουτίνα HEX_KEYB και μόλις συμπληρωθούν 2 έγκυρα ψηφία να τυπώνει τον αριθμό σε δεκαεξαδική, δεκαδική, οκταδική και δυαδική μορφή με ένα χαρακτήρα '=' μεταξύ τους. Στη συνέχεια αναμένει νέο διψήφιο δεκαεξαδικό αριθμό. Το πρόγραμμα τερματίζεται σε οποιοδήποτε σημείο εισαγωγής χαρακτήρα με τον 'Q'.

```
INCLUDE MACROS.ASM
DATA SEGMENT
   ;edw mpainoun ta dedomena
   MSG1 DB 'DEC:$'
   EQUAL DB '=$'
   MSG2 DB 'BIN:$'
   MSG3 DB 'OCT:$'

ENDS

STACK SEGMENT
   DW 128 DUP(0)
ENDS

CODE SEGMENT
   ASSUME CS:CODE,DS:DATA

MAIN PROC FAR
   ;arxikopoihsh twn segment registers
   MOV AX,DATA
   MOV DS,AX
   MOV ES,AX

START:          ;edw arxizei o kwdikas
  CALL HEX_KEYB
  CMP AL,'Q'      ;elegxos an patithike to Q gia eksodo
  JE QUIT
  MOV DH,AL
  CALL HEX_KEYB
  CMP AL,'Q'
  JE QUIT
  MOV DL,AL
  MOV CL,4
  ROL DH,CL
  ADD DL,DH
  NEW_LINE
  PRINT_STRING MSG1
  CALL PRINT_DEC
  PRINT_STRING EQUAL
  PRINT_STRING MSG2
  CALL PRINT_BIN
```

```
  PRINT_STRING EQUAL
  PRINT_STRING MSG3
  CALL PRINT_OCT



 QUIT:
   EXIT
   MAIN ENDP


;tmhma routinwn


PRINT_DEC PROC NEAR
  PUSH DX
  MOV AH,0
  MOV AL,DL ;AX = 00000000@@@@@@@@ (ektipwnetai arithmos twn 8 bit)
  MOV DL,10
  MOV CX,1 ;metritis dekadwn
LOOP_10:
  DIV DL ;diairoume ton arithmo me to 10
  PUSH AX ;apothikeuoume tis monades
  CMP AL,0 ;an to piliko einai iso me 0  tote exoume spasei ton arithmo se dekadika psifia
  JE PRINT_DIGITS_10
  INC CX ;auksanoume ton arithmo twn dekadwn
  MOV AH,0
  JMP LOOP_10 ;an to piliko den einai 0 ksanadiairoume
PRINT_DIGITS_10:
  POP BX ;kanoume pop dekadiko psifio pros ektipwsi
  MOV BL,BH
  MOV BH,0 ;DX = 00000000@@@@@@@@ (o ASCII tou aarithmou pou tha ektipwthei)
  ADD BX,30H
  PRINT BL ;ektpwsi
  LOOP PRINT_DIGITS_10 ;epanalipsi gia ola ta psifia
  POP DX
  RET
ENDP PRINT_DEC



PRINT_BIN PROC NEAR
   PUSH DX
   PUSH BX
   PUSH CX
   MOV CX,8
BIN_START:
   SHL DL,1
   MOV BL,0
   ADC BL,30H ;kane shift kathe psifio kai an einai  0 ektipwse '0' (ascii code 30) alliws '1' (31)
   PRINT BL
   LOOP BIN_START
   POP CX
   POP BX
   POP DX
   RET
PRINT_BIN ENDP
```

```
PRINT_OCT PROC NEAR ;xwrizoume ton arithmo se traides psifiwn kai ektipwnoume ksekinontas
apota deksia pros ta aristera
   PUSH DX
   PUSH CX
   PUSH BX
   MOV BL,DL ;3 lsb psifia
   AND BL,7
   ADD BL,30H
   PUSH BX
         ;PRINT BL
   MOV BX,DX
   MOV CX,3
   SHR BX,CL
   AND BL,7 ;kane shift pros ta deksia 3 fores gia ta epomena 3 psifia
   ADD BL,30H
   PUSH BX
   MOV BX,DX ;kane shift pros ta deksia 6 fores gia t aepomena 3 psifia
   MOV CX,6
   SHR BX,CL
   AND BL,7
   ADD BL,30H
   PRINT BL
   POP BX
   PRINT BL
   POP BX
   PRINT BL
   POP BX
   POP CX
   POP DX
   RET
 PRINT_OCT ENDP


HEX_KEYB PROC NEAR
   PUSH DX
IGNORE:
   READ
   CMP AL,'Q'
   JE ADDR2
   CMP AL,30H
   JL IGNORE
   CMP AL,39H
   JG ADDR1
   PUSH AX
   POP AX
   SUB AL,30H
   JMP ADDR2
ADDR1:
   CMP AL,'A'
   JL IGNORE
   CMP AL,'F'
   JG IGNORE
   PUSH AX
   POP AX
   SUB AL,37H
```

```
ADDR2:
   POP DX
   RET
HEX_KEYB ENDP

CODE ENDS
 END MAIN
```

ΑΣΚΗΣΗ 2 :

Θέλουμε πρόγραμμα που να αποθηκεύει τους αριθμούς 254, 253, …,1, 0, 255 με τη σειρά αυτή, σε διαδοχικές θέσεις της μνήμης αρχίζοντας από την θέση TABLE. Στη συνέχεια το πρόγραμμα να συμπληρωθεί με τους εξής δύο (2) υπολογισμούς και να τυπώνει τα αποτελέσματα σε 2 γραμμές στην οθόνη:

α. Το ακέραιο μέρος του μέσου όρου των άρτιων (128) από τα 256 δεδομένα σε δεκαεξαδική μορφή.

β. Το μέγιστο και τον ελάχιστο σε μέγεθος από το παραπάνω σύνολο δεδομένων. Τα δύο αυτά αποτελέσματα θα τυπωθούν με ένα κενό μεταξύ τους σε δεκαεξαδική μορφή.

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

 N EQU 255


 data segment
 ; vale edw ta dedomena
    TABLE dw N dup(?)
 ends

 stack segment
 ends

 code segment

 start:
 ; arxikopoihsh twn segment registers:
    mov ax, data
    mov ds, ax
    mov es, ax

 ; arxh tou kwdika
    mov cx, N
    dec cx
    cld ; df = 0
    mov di, OFFSET TABLE ; vazw ston kataxwrith tin thesi tou TABLE
    mov dx, 0 ; arxikopoihsh
    mov bx, 0
    mov ax, 254
    mov dl, 128
    mov dh, 2

 write_again:
```

```
    div dh ; diairesh me to 2
    cmp ah, 1 ; elegxos upoloipou
    je here ;gia peritto kanoume jump

    mov ax, cx ; gia artio grafoume ston sisoreuth tin timh tou
    add bx, ax
 here:
    mov ax, cx ; vazw thn iparxousa timh ston ax
    stosw ; apothikeush kai sunexeia st epomeno iteration
    loop write_again

    mov ax, 0 ; den exoume valei to 0 kai to 255
    stosw ;
    mov ax,255
    stosw


    mov ax, bx ; vazoume ton sisoreuth ston ax
    div dl ; kai diairoume me to plhthos twn artiwn

    mov dl, ah ; kratame to phliko
    call print_hex_full

    mov cx, N
    inc cx
    cld ; df = 0
    mov di, OFFSET TABLE ; epanarxikopoioume gia na vroume min kai max
    mov dx, 0FFFFh ; dx = local min
    mov bx, 0000H ; bx = local max
load_again:
    lodsw

minimum:
    cmp ax, dx ; elegxos gia to an einai mikrotero

    ja maximum
    mov dx, ax ; local min <- current

maximum:
    cmp ax, bx ; current > local max ?
    jb next
    mov bx, ax

next:
    loop load_again ; sti sinexeia ftiaxnoume to output
    push dx
    print " "
    print "m"
    print ":"
    pop dx
    mov ax, dx
    mov dl, ah
    call print_hex_full
    mov dl, al
    call print_hex_full
    push dx
    print " "
```

```
    print "M"
    print ":"
    pop dx
    mov dl, bh ;
    call print_hex_full
    mov dl, bl ;
    call print_hex_full

    mov ax, 4c00h ; eksodos sto leitourgiko sistima
    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

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

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

 end start
```

ΑΣΚΗΣΗ 3:

Σε ένα προσωπικό υπολογιστή, που βασίζεται στον μΕ 80x86, γράφουμε πρόγραμμα Assembly με τις παρακάτω προδιαγραφές:

1. Να δέχεται δυο (2) διψήφιους δεκαεξαδικούς αριθμούς: x και y από το πληκτρολόγιο (0-F), τους οποίους να τυπώνει στην οθόνη, για παράδειγμα, ως εξής:
`x=2F y=3A`

2. Στην επόμενη γραμμή της οθόνης (με την προϋπόθεση ότι συμπληρώθηκαν τέσσερα έγκυρα HEX ψηφία), να υπολογίζει και να τυπώνει το άθροισμα και τη διαφορά των 2 αριθμών σε δεκαδική μορφή, όπως φαίνεται παρακάτω:
`x+y=105 x-y=-11`

```
INCLUDE MACROS.ASM
 DATA SEGMENT

 ; edv mpainoun ta dedomena
 X_MSG DB "X=",'$'
 Y_MSG DB "Y=",'$'
 ADD_MSG DB "X+Y=",'$'
 SUB_MSG DB "X-Y=",'$'
 ERR DB "WRONG INPUT! TRY AGAIN!",'$'

 ENDS


 STACK SEGMENT

 DW 128 DUP(0)

 ENDS

 CODE SEGMENT

 START:
 ; segment registers:
 MOV AX, DATA
 MOV DS, AX
 MOV ES, AX

 ; apo edv arxizei o kvdikas


 MAIN PROC FAR

 MOV CL, 00H
 LOOP:
 CALL READ_NUM
 CMP AL, 0DH ; elegxv an paththike ENTER
 JE VALIDATION ; an nai pav sto validation
 MOV AH, 00H
 PUSH AX
 INC CL ; auksanv to CL++
 JMP LOOP
```

```
VALIDATION: ; elegxv an 4 egkyra noumera exoun paththei

NEW_LINE
CMP CL, 04H
JZ PRINT_RESULTS


PRINT_ERROR:
NEW_LINE
PRINT_STRING ERR ;typvnv to mhnyma
NEW_LINE
JMP START        ;epanalambanv

;kathe 16adikos mpainei sto stack: first msb then lsb
PRINT_RESULTS:
PRINT_STRING X_MSG
POP AX ;lsb of y
POP DX ;msb of y
POP BX ;lsb of x
POP CX ;msb of x
PUSH DX ;msb of y
PUSH AX ;lsb of y
MOV DL, CL
CALL PRINT_HEX
MOV DL, BL
CALL PRINT_HEX
PRINT " "
MOV CH, 16
MOV AL, CL ;shift left 4 theseis
MUL CH
OR AL, BL ;AL == x
POP BX ;lsb of y
POP CX ;msb of y
PUSH AX ;x mono sto stack


PRINT_STRING Y_MSG ;typvnv to minima
MOV DL, CL
CALL PRINT_HEX
MOV DL, BL
CALL PRINT_HEX
MOV CH, 16
MOV AL, CL
MUL CH
OR AL, BL ;AL == y



PUSH AX
NEW_LINE
PRINT_STRING ADD_MSG  ;typvnv to minima

POP AX ;AL = y
POP BX ;BL = x
MOV DL, AL
ADD DL, BL ; DL = AL + BL

PUSH AX
PUSH BX
```

```
CALL PRINT_DEC
PRINT " "
PRINT_STRING SUB_MSG

POP AX ;AL = x
POP BX ;BL = y


MOV DL, AL
SUB DL, BL ; DL = AL - BL
JNS PRINT_ABS ;an einai arnhtikos typvnv ABS
NEG DL
PUSH DX
PRINT "-"
POP DX
PRINT_ABS:
CALL PRINT_DEC
NEW_LINE


QUIT:
EXIT
MAIN ENDP


; diabazei egkyro arithmo / xarakthra kai apothikeuv ston AL
; me ENTER gyrizei ton ASCI kvdiko
READ_NUM PROC NEAR

; diabazei key ASCI CODE sto AL
MOV AH, 01H
INT 21H

CMP AL, 0DH ; CHECK an pathse ENTER
JNE CHECK_NUM
RET

CHECK_NUM: ; elegxv an patise kati metaksi (30H ~ 39H)

CMP AL,30H
JL READ_NUM
CMP AL,39H
JG NOT_DIGIT
PUSH AX
POP AX
SUB AL,30H
JMP IS_DIGIT


NOT_DIGIT:

CMP AL,'A'
JL CHECK_NUM
CMP AL,'F'
JG CHECK_NUM
PUSH AX
POP AX
SUB AL,37H
```

```
IS_DIGIT:

RET
READ_NUM ENDP



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_DEC PROC NEAR
PUSH DX
MOV AH,0
MOV AL,DL ;AX = 00000000xxxxxxxx(ta 8 msb tha typvthoun)
MOV DL,10
MOV CX,1 ;metrhths decadvn

LOOP_10:
DIV DL ;diairv me 10
PUSH AX ;svzv monades
CMP AL,0
JE PRINT_DIGITS_10 ;an einai mhden typvnv me bash dekades
INC CX ;auksanv decades
MOV AH,0
JMP LOOP_10 ;epanalambanv an xreiazetai

PRINT_DIGITS_10:
POP BX ;pop dec digit pou tha ektypvthei
MOV BL,BH
MOV BH,0 ;DX = 00000000xxxxxxxx (ASCII tou arithmou pou tha typvthei)
ADD BX,30H ;briskv ASCII code
PRINT BL ;print
LOOP PRINT_DIGITS_10 ;Loop gia ola ta psifia
POP DX
RET
ENDP PRINT_DEC


PRINT_HEX_FULL PROC NEAR
PUSH AX
PUSH BX
```

```
 PUSH DX

 SAR DX, 1
 SAR DX, 1
 SAR DX, 1
 SAR DX, 1
 AND DL, 0FH
 CALL PRINT_HEX

 POP DX
 AND DL, 0FH
 CALL PRINT_HEX
 POP BX
 POP AX
 RET

 PRINT_HEX_FULL ENDP

 END START
```

ΑΣΚΗΣΗ 4:

Σε ένα προσωπικό υπολογιστή, που βασίζεται στον μΕ 80x86, γράφουμε πρόγραμμα Assembly με τις παρακάτω προδιαγραφές:

1. Να αναμένει την πληκτρολόγηση 16 χαρακτήρων που να αποτελούνται από κεφαλαία αγγλικά A-Z και τους αριθμούς 0-9, τους οποίους να τυπώνει στην οθόνη, αγνοώντας κάθε άλλο χαρακτήρα.

2. Στη συνέχεια, με τη συμπλήρωση 16 έγκυρων χαρακτήρων στην επόμενη γραμμή το πρόγραμμα να τυπώνει το ίδιο κείμενο με πεζούς χαρακτήρες (a-z) ξεχωρίζοντας και τυπώνοντας τους αριθμούς 0-9 στην αρχή με μια παύλα '-' διατηρώντας όμως τη σειρά με την οποίαν δόθηκαν. Το πρόγραμμα είναι συνεχούς λειτουργίας και τερματίζεται σε οποιοδήποτε σημείο με τον χαρακτήρα ENTER.

```
INCLUDE MACROS.ASM
DATA SEGMENT   ;edv bazv to meros tou kvdika

    INPUT1 DB 16 DUP(?) ;gia noumera
    INPUT2 DB 16 DUP(?) ;gia arithmous
    NUM_COUNT DB 0      ;kataxvrhtes - metrhtes
    LET_COUNT DB 0
ENDS


STACK SEGMENT


ENDS


CODE SEGMENT

START:

;bazv tous segment registers
   MOV AX, DATA
   MOV DS, AX
   MOV ES, AX
```

```
;apo edv arxizei o kvdikas
   MOV BX,0
   CLD
   MOV CX,16 ;16 byte sunolika

   AGAIN:
   CALL READ_KEY
   CMP AL, 0DH ;elegxv an pathse enter
   JE STOP     ;an nai stamatav
   PRINT AL    ;allivs ektypvnv kai sunexizv
   LOOP AGAIN
   NEW_LINE

   NUMBER_PRINT:
   CLD
   MOV SI,OFFSET INPUT1
   MOV CH,0
   MOV CL,NUM_COUNT  ;CL->plhthos arithmvn
   TEST CL,0FFh     ;an den exei numbers
   JZ LETTER_PRINT  ;typvnv letters

   AGAIN_1:
   LODSB
   PRINT AL
   LOOP AGAIN_1
   PRINT '-'


   LETTER_PRINT:
   CLD
   MOV SI,OFFSET INPUT2
   MOV CH,0
   MOV CL,LET_COUNT ;CL-> arithmo grammatvn
   TEST CL,0FFh ;an den exei grammata stamatv
   JZ STOP

   AGAIN_2: ;an exei grammata
   LODSB
   ADD AL,32 ;32 einai h diafora metaksi kefalaiou kai mikrou grammatos
   PRINT AL
   LOOP AGAIN_2



   STOP:
   MOV AX, 4C00H ;eksodos apo to leitoyrgiko systhma
   INT 21H       ;kanv ektyopvsh
ENDS




   ; diabazv ASCII CODES tvn mikrvn grammatvn kai arithmvn
   READ_KEY PROC NEAR
   IGNORE:
   ; READ
   MOV AH, 8
```

```
   INT 21H

   ; elegxv an pathsa enter
   CMP AL, 0DH
   JE STOP           ;an nai stamatav

   ;elegxv an einai gramma h arithmos h kati lathos
   CMP AL, 30H
   JL IGNORE
   CMP AL, 39H
   JG ADDR_1
   MOV DL,NUM_COUNT
   MOV BX,OFFSET INPUT1 ;starting address of input1-numbers
   ADD BX,DX
   MOV [BX],AL
   INC DL
   MOV NUM_COUNT,DL ;auksanv kai svzv ton number counter
   JMP EXIT

   ADDR_1:
   CMP AL, 'A'      ;elegzv an einai kefalaio allivs to agnov
   JL IGNORE
   CMP AL, 'Z'
   JG IGNORE
   MOV DL,LET_COUNT
   MOV BX,OFFSET INPUT2 ;starting address of input2-letters
   ADD BX,DX
   MOV [BX],AL
   INC DL
   MOV LET_COUNT,DL ;auksanv kai svzv ton letter counter
   EXIT:
   RET
   READ_KEY ENDP



END START
```

ΑΣΚΗΣΗ 5:

Σε ένα προσωπικό υπολογιστή, που βασίζεται στον μΕ 80x86 γράφουμε πρόγραμμα Assembly με τις παρακάτω προδιαγραφές:

Να παρακολουθεί και να απεικονίζει θερμοκρασίες από 0°C ως 1000°C στην οθόνη του PC, σε δεκαδική μορφή και με ακρίβεια ενός κλασματικού δεκαδικού ψηφίου. Η τάση που παρέχεται από τον αισθητήρα θερμοκρασίας έχει την χαρακτηριστική καμπύλη (Θερμοκρασία/ Τάση εξόδου) και ακολουθείται από ένα μετατροπέα από Αναλογική τιμή σε Ψηφιακή (ADC) των 12 bits . Το πρόγραμμα να αρχίζει με το μήνυμα "`START (Y, N):`" και ανάλογα με το χαρακτήρα που δίνεται να ξεκινάει ή να τερματίζεται. Μετά την εκκίνηση να αναμένει 3 HEX ψηφία και να είναι συνεχούς λειτουργίας. Επίσης και στη φάση της λειτουργίας να τερματίζεται αν δοθεί οποιαδήποτε στιγμή ο χαρακτήρας Ν. Για τιμές μεγαλύτερες από 999,9°C να εμφανίζεται το μήνυμα σφάλματος "`ERROR`".

```
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
; edv bazv ta dedomena!
START_MSG db "START(Y/N):",'$'
ERR db "ERROR",'$'
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:
; bazv tous kataxvrhtes:

mov ax, data
mov ds, ax
mov es, ax

; apo edv o kvdikas
print_str START_MSG
read_init:
mov ah, 1
int 21h
cmp al, 'N'
je stop
cmp al, 'Y'
jne read_init

new_line
mov bx, 0000h
call hex_key ; 1 MSB
cmp al, 'N'
je stop
mov ah, 0
add bx, ax
mov cl, 4
shl bx, cl ;
```

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


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

; edv vriskoyme thn timh ths eksodou tou A/D
; X = aY -> a = tan() = 2000/4096

mov ax, bx
mov cx, 2000
mul cx ; ax = ax * 2000


mov cx, 4095
div cx ; ax = XXXXXY

; epeita vriskv se poia apo tis treis periptwseis anhkei
cmp ax, 1000 ; if ax <= 1000
jle case_1
cmp ax, 1800 ; else if ax < 1800
jl case_2
jmp case_3 ; else

; X = aY + b

case_1: ; X = 5Y ->
mov cx, 5 ; a = 5
mov bx, 0 ; b = 0
jmp compute
case_2: ; X = 2.5 * Y + 2500 -> (kata prosseggish)
mov cx, 2 ; a = 2
mov bx, 3300 ; b = 3300 (> 0 so we add)
jmp compute_case_2
case_3: ; X = 15 * Y - 20.000
mov cx, 15 ; a = 15
mov bx, 20000 ; b = 20.000
compute:
mul cx
sub ax, bx
jmp cont
compute_case_2:
mul cx
add ax, bx
cont:
cmp ax, 10000
jg error
new_line
mov cx, 10000 ;
```

```
div cx ; 10000
mov bx, ax ;
call print_digit
; typvnv apotelesma kai pav pali sthn arxh
mov ax, dx
mov dx, 0
mov cx, 1000 ;
div cx ; 1000
mov bx, ax ;
call print_digit

mov ax, dx
mov dx, 0

mov cx, 100 ;
div cx ; 100
mov bx, ax ;
call print_digit

mov ax, dx
mov dx, 0
mov cx, 10 ;
div cx ; 100
mov bx, ax ;
call print_digit

print ","

mov bx, dx ;
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
```

```
; blepv an pathse enter
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



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 ;
div cl ; 100

mov dl, al ;
call print_digit

mov cl, 10 ;
mov al, ah ;
mov ah, 0
```

```
div cl
mov dl, al
call print_digit

mov dl, ah ;
call print_digit

pop ax
pop cx
pop dx
ret
PRINT_DEC endp

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