




Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

6^ο Εξάμηνο

Συστήματα Μικροϋπολογιστών

5^η ΟΜΑΔΑ ΑΣΚΗΣΕΩΝ


Χατζή Ήβη

Κώδικας μακροεντολών: αρχείο *macros.asm*

```
READ MACRO
    MOV AH,8
    INT 21H
ENDM

READPRINT MACRO
    MOV AH,1
    INT 21H
ENDM

PRINT MACRO CHAR
    PUSH AX
    PUSH DX
    MOV DL,CHAR
    MOV AH,2
    INT 21H
    POP DX
    POP AX
ENDM

PRINT_STR MACRO MSG
    PUSH AX
    PUSH DX
    MOV DX, OFFSET MSG
    MOV AH,9
    INT 21H
    POP DX
    POP AX
ENDM

NEWLINE MACRO
    PUSH BX
    MOV BL,13
    PRINT BL
    MOV BL,10
    PRINT BL
    POP BX
ENDM

TABS MACRO          ;space
    PUSH AX
    PUSH DX
    MOV DL,9
    MOV AH,2
    INT 21H
    POP DX
    POP AX
ENDM
```

```

PRINT_NUM MACRO CHAR
    PUSH AX
    PUSH DX
    MOV DL, CHAR
    ADD DL, 30H
    MOV AH, 2
    INT 21H
    POP DX
    POP AX
ENDM

EXIT MACRO
    MOV AX, 4C00H
    INT 21H
ENDM

```

1^η ΑΣΚΗΣΗ:

Περιγραφή κώδικα:

Αποθηκεύουμε τους αριθμούς 128,127,126,...,2,1 με τη σειρά αυτή σε διαδοχικές θέσεις της μνήμης, (8-bit), ξεκινώντας από τη θέση TABLE.

α. Για την εύρεση των περιττών διαιρούμε διαδοχικά κάθε αριθμό με το δύο και αν το υπόλοιπο είναι διάφορο του μηδενός αθροίζουμε και αυξάνουμε τον μετρητή που δηλώνει το τρέχον πλήθος περιττών. Στη συνέχεια διαιρούμε το άθροισμα των περιττών με το πλήθος τους.

β. Για να βρούμε το μέγιστο και το ελάχιστο μεταξύ αυτών συγκρίνουμε σειριακά.

Χρησιμοποιούνται μακροεντολές από το αρχείο macros.asm και οι ρουτίνες 16BIN_DEC για την εκτύπωση του αριθμού σε δεκαδική μορφή, PRINT_NUM8_HEX και PRINT_HEX για την μετατροπή και εκτύπωση σε δεκαεξαδική μορφή. (Οι ρουτίνες αυτές υλοποιήθηκαν βάσει των αντίστοιχων διαφανειών του αρχείου mP11_80x86_programs.pdf.)

Έξοδος:

 emulator screen (80x25 chars)



Κώδικας για τη 1^η άσκηση: αρχείο ex1.asm

```
INCLUDE macros.asm

DATA_SEG SEGMENT
    TABLE DB 128 DUP(?)
    TWO DB DUP(2)
DATA_SEG ENDS

CODE_SEG SEGMENT
    ASSUME CS:CODE_SEG, DS:DATA_SEG

MAIN PROC FAR
    MOV AX, DATA_SEG
    MOV DS, AX

    MOV DI, 0
    MOV CX, 128
FILL_TABLE:
    MOV TABLE[DI], CL
    INC DI
    LOOP FILL_TABLE

    MOV DH, 0
    MOV AX, 0
    MOV BX, 0        ;counter of odds
    MOV DI, 0        ;table iterator
    MOV CX, 128      ;counter

ADD_ODDS:
    PUSH AX
    MOV AH, 0
    MOV AL, TABLE[DI]
    DIV TWO          ;divide by 2
    CMP AH, 0        ;even
    POP AX
    JE IS_EVEN
    MOV DL, TABLE[DI]
    ADD AX, DX        ;if odd add
    INC BX            ;and increase counter

IS_EVEN:
    INC DI            ;next number
    LOOP ADD_ODDS     ;repeat check until CX=0

AVERAGE:
    MOV DX, 0
    DIV BX            ;sum/num
    CALL 16BIN_DEC

    NEWLINE

    MOV AL, TABLE[0]
    MOV BL, TABLE[127]
    MOV DI, 0
```

```

    MOV CX, 128

CHECKMAX:
    CMP AL, TABLE[DI]
    JC NEWMAX
    JMP CHECKMIN

NEWMAX:
    MOV AL, TABLE[DI]

CHECKMIN:
    CMP TABLE[DI], BL
    JC NEWMIN
    JMP NEXT

NEWMIN:
    MOV BL, TABLE[DI]

NEXT:
    INC DI
    LOOP CHECKMAX

    CALL PRINT_NUM8_HEX
    PRINT ' '
    MOV AL, BL
    CALL PRINT_NUM8_HEX

    EXIT

MAIN ENDP

16BIN_DEC PROC NEAR
    MOV CX, 0
ADDR1:
    MOV DX, 0
    MOV BX, 10
    DIV BX
    PUSH DX
    INC CX
    CMP AX, 0
    JNE ADDR1
ADDR2:
    POP DX
    PRINT_NUM DL
    LOOP ADDR2
    RET
16BIN_DEC ENDP

PRINT_NUM8_HEX PROC NEAR
    MOV DL, AL
    AND DL, 0F0H
    MOV CL, 4
    ROR DL, CL
    CMP DL, 0
    JE SKIP0

```

```
        CALL PRINT_HEX
SKIP0:
        MOV DL,AL
        AND DL,0FH
        CALL PRINT_HEX
        RET
PRINT_NUM8_HEX ENDP

PRINT_HEX PROC NEAR
        CMP DL,9
        JG CHARACTER
        ADD DL,30H ;48
        JMP SHOW
CHARACTER:
        ADD DL,37H ;55
SHOW:
        PRINT DL
        RET
PRINT_HEX ENDP

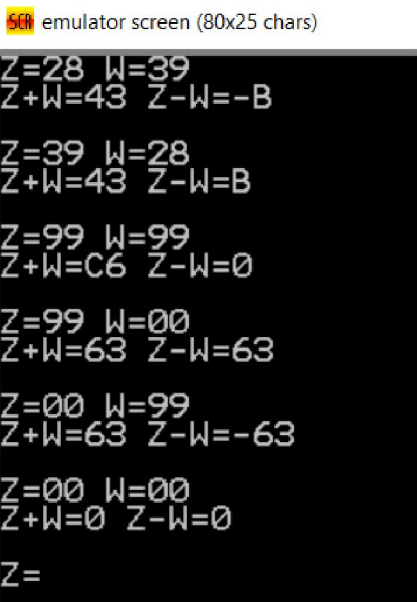
CODE_SEG ENDS
END MAI
```

2^η ΑΣΚΗΣΗ

Περιγραφή κώδικα:

Το πρόγραμμα αρχικά διαβάζει και τυπώνει 2 διψήφιους δεκαδικούς αριθμούς Z, W χρησιμοποιώντας τη ρουτίνα DEC_KEYB για καθένα από τα 2 ψηφία τους. Πολλαπλασιάζουμε το πρώτο ψηφίο με 10 και προσθέτουμε το δεύτερο για να προκύψει ο κάθε αριθμός. Αποθηκεύουμε το Z στον καταχωρητή BL και τον W στον καταχωρητή CL, τους αθροίζουμε και τυπώνουμε το άθροισμά τους. Για την αφαίρεση ελέγχουμε πρώτα αν ο W είναι μεγαλύτερος από τον Z, και στην περίπτωση αυτή κάνουμε την αφαίρεση W-Z και τυπώνουμε ένα '-' πριν τη διαφορά τους. Το τύπωμα ψηφίων γίνεται με την υπορουτίνα PRINT_HEX, ενώ το τύπωμα των διψήφιων δεκαεξαδικών γίνεται με την υπορουτίνα PRINT_HEX2, η οποία καλεί μία ή δύο φορές την PRINT_HEX ελέγχοντας αν το πρώτο ψηφίο είναι μηδέν. Χρησιμοποιούμε επίσης μακροεντολές από το αρχείο macros.asm.

Παράδειγμα εξόδου:

 emulator screen (80x25 chars)

```
Z=28 W=39
Z+W=43 Z-W=-B

Z=39 W=28
Z+W=43 Z-W=B

Z=99 W=99
Z+W=C6 Z-W=0

Z=99 W=00
Z+W=63 Z-W=63

Z=00 W=99
Z+W=63 Z-W=-63

Z=00 W=00
Z+W=0 Z-W=0

Z=
```

Κώδικας για τη 2^η άσκηση: αρχείο ex2.asm

```
INCLUDE macros.asm

DATA SEGMENT
    MZ DB 'Z=$'
    MW DB "W=$"
    MA DB "Z+W=$"
    MS DB "Z-W=$"
    MM DB "Z-W=-$"
DATA ENDS

CODE SEGMENT
    ASSUME CS:CODE, DS:DATA

    MAIN PROC FAR
        MOV AX,DATA
        MOV DS,AX

    START:
        PRINT_STR MZ
        CALL DEC_KEYB ;get 10s digit of Z
        MOV BL,10
        MUL BL ;multiply by 10
        MOV BL,AL ;store MSB in BL
        CALL DEC_KEYB ;get 1s digit of Z
        ADD BL,AL ;BL stores Z

        PRINT ' '
        PRINT_STR MW
        CALL DEC_KEYB ;get 10s digit of W
        MOV CL,10
        MUL CL ;multiply by 10
        MOV CL,AL ;store MSB in CL
        CALL DEC_KEYB
        ADD CL,AL

        NEWLINE

        PRINT_STR MA
        MOV AL,BL
        ADD AL,CL ;AL=Z+W
        CALL PRINT_HEX2

        PRINT ' '
        PRINT_STR MS
        MOV AL,BL
        CMP AL,CL
        JB NEGSUB ;compare Z,W
        MOV AL,BL ;Z>=W
        SUB AL,CL ;AL=Z-W
        JMP PRINTSUB

    NEGSUB: ;Z<W
        PRINT '-'
```



```
    MOV AL,CL
    SUB AL,BL      ;AL=W-Z
```

```
PRINTSUB:
    CALL PRINT_HEX2
```

```
    NEWLINE
    NEWLINE
    JMP START
```

```
EXIT
MAIN ENDP
```

```
DEC_KEYB PROC NEAR      ;from class slides
IGNORE:
    READ
    CMP AL,30H
    JB IGNORE
    CMP AL,39H
    JA IGNORE
    PRINT AL
    SUB AL,30H
    RET
DEC_KEYB ENDP
```

```
PRINT_HEX PROC NEAR    ;from class slides
    CMP DL,9
    JG ADDR1
    ADD DL,30H
    JMP ADDR2
ADDR1:
    ADD DL,37H
ADDR2:
    PRINT DL
    RET
PRINT_HEX ENDP
```

```
PRINT_HEX2 PROC NEAR   ;to print 2 digit hex
    MOV DL,AL
    AND DL,0F0H        ;isolate MSB
    PUSH CX
    MOV CL,4
    ROR DL,CL
    POP CX
    CMP DL,0
    JE SKIP            ;don't print MSB if 0
    CALL PRINT_HEX     ;print MSB
SKIP:
    MOV DL,AL
    AND DL,0FH         ;isolate LSB
    CALL PRINT_HEX     ;print LSB
```

```

    RET
PRINT_HEX2 ENDP

CODE ENDS
END MAIN

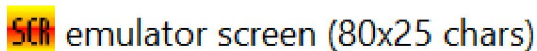
```

3^η ΑΣΚΗΣΗ:

Περιγραφή κώδικα:

Υλοποιούμε τις ρουτίνες PRINT_DEC, PRINT_OCT και PRINT_BIN, οι οποίες δέχονται μέσω του BX έναν 12-bit δεκαεξαδικό αριθμό και τον εμφανίζουν σε δεκαδική, οκταδική και δυαδική μορφή. Η εισαγωγή του αριθμού γίνεται καλώντας 3 φορές τη ρουτίνα HEX_KEYB που διαβάζει ένα δεκαεξαδικό ψηφίο και ενώνοντας τα ψηφία ώστε να σχηματιστεί ο 12-bit αριθμός που αποθηκεύεται στον BL. Σε κάθε εισαγωγή ελέγχουμε αν δόθηκε ο χαρακτήρας 'T', οπότε τερματίζεται η λειτουργία του προγράμματος. Τα ψηφία του αριθμού εξάγονται διαιρώντας διαδοχικά με το 10 για δεκαδικό, 8 για οκταδικό, 2 για δυαδικό, μέχρι το πηλίκο των διαιρέσεων να μηδενιστεί και αποθηκεύονται προσωρινά στη στοίβα με σειρά από το LSB στο MSB. Στη συνέχεια καλούνται διαδοχικά οι ρουτίνες PRINT_DEC, PRINT_OCT και PRINT_BIN που εμφανίζουν στην οθόνη τον αριθμό σε δεκαδική, οκταδική και δυαδική μορφή αντίστοιχα.

Παράδειγμα εξόδου:

 emulator screen (80x25 chars)

```

FFF=4095=7777=111111111111
F30=3888=7460=111100110000
100=256=400=1000000000
0A5=165=245=10100101

```

Κώδικας για την 3^η άσκηση: αρχείο ex3.asm

```

INCLUDE macros.asm

CODE_SEG SEGMENT
    ASSUME CS:CODE_SEG

    MAIN PROC FAR

        START:
            CALL HEX_KEYB
            CMP AL, 'T'
            JE QUIT
            MOV BL, AL
            ROL BL, 4
            CALL HEX_KEYB
            CMP AL, 'T'
            JE QUIT
            OR BL, AL

```

```

    ROL BX, 4
    CALL HEX_KEYB
    CMP AL, 'T'
    JE QUIT
    OR BL, AL

    PRINT '='
    CALL PRINT_DEC
    PRINT '='
    CALL PRINT_OCT
    PRINT '='
    CALL PRINT_BIN
    NEWLINE

    JMP START

QUIT:
    EXIT
MAIN ENDP

HEX_KEYB PROC NEAR
    IGNORE:
        READ
        CMP AL, 'T'
        JE RETURN
        CMP AL, 48      ;48, ('O'=30H)
        JL IGNORE      ;<0
        CMP AL, 57      ;57, ('9'=39H)
        JG ISLETTER     ;>9
        PRINT AL
        SUB AL, 48      ;ASCII code
        JMP RETURN
    ISLETTER:
        CMP AL, 'A'
        JL IGNORE
        CMP AL, 'F'
        JG IGNORE
        PRINT AL
        SUB AL, 55      ;(55)ASCII to number
    RETURN:
        RET
HEX_KEYB ENDP

PRINT_DEC PROC NEAR
    MOV DX, BX
    PUSH BX
    MOV CX, 0
    MOV AX, DX
    GET_DEC:
        MOV DX, 0
        MOV BX, 10
        DIV BX           ;DIVIDE BY 10 AND STORE REMAINDER IN STAC
        PUSH DX          ;REMAINDER IS SMALLEST DIGIT YET (SMALL FIRST,BIG LAST)

```

```

        INC CX            ;MORE DIGITS
        CMP AX,0          ;PHLIKO 0 ARA DONE
        JNE GET_DEC       ;NOT DONE YET
SHOW_DEC:
        POP DX            ;GET DIGITS FROM STACK (BIG FIRST, SMALL LAST)
        ADD DX,30H
        PRINT DL
        LOOP SHOW_DEC     ;WE COUNT DIGITS TO PRINT IN CX          POP BX
        RET
PRINT_DEC ENDP

```

```

PRINT_OCT PROC NEAR

```

```

        PUSH BX
        MOV DX,BX
        MOV CX,0
        MOV AX,DX
GET_OCT:
        MOV DX,0
        MOV BX, 8
        DIV BX
        PUSH DX
        INC CX
        CMP AX,0
        JNE GET_OCT

```

```

SHOW_OCT:
        POP DX
        ADD DX,30H
        PRINT DL
        LOOP SHOW_OCT
        POP BX
        RET

```

```

PRINT_OCT ENDP

```

```

PRINT_BIN PROC NEAR

```

```

        MOV DX,BX
        MOV CX,0
        MOV AX,DX
GET_BIN:
        MOV DX,0
        MOV BX, 2
        DIV BX
        PUSH DX
        INC CX
        CMP AX,0
        JNE GET_BIN

```

```

SHOW_BIN:
        POP DX
        ADD DX,30H
        PRINT DL
        LOOP SHOW_BIN
        RET

```

```

PRINT_BIN ENDP

```

```

CODE_SEG ENDS
END MAIN


```

4^η ΑΣΚΗΣΗ

Περιγραφή κώδικα:

Το πρόγραμμα αρχικά διαβάζει, τυπώνει και αποθηκεύει στον πίνακα ARR χαρακτήρες a-z και 0-9, αγνοώντας όλους τους υπόλοιπους χαρακτήρες εκτός από το = και το ENTER. Αν διαβάσει =, το πρόγραμμα τερματίζει, ενώ αν διαβάσει ENTER ή αν γεμίσει ο πίνακας ARR (20 θέσεων) αλλάζει γραμμή. Μετά διατρέχει τον πίνακα ARR δύο φορές, όπου στην πρώτη τυπώνει μόνο τα γράμματα αφού τα μετατρέψει σε κεφαλαία, και στην δεύτερη τυπώνει μόνο τους αριθμούς. Χρησιμοποιεί τις μακροεντολές PRINT και READ από το αρχείο macros.asm.

Παράδειγμα εξόδου:

 emulator screen (80x35 chars)

```
a8x9s1fetd73a8kl  
AXSFETDAKL-891738  
  
qwertyuiopasdfghjklz  
QWERTYUIOPASDFGHJKLZ-  
  
01234567899876543210  
-01234567899876543210  
  
a  
A-  
  
3  
-3  
  
adc531  
ADC-531  
  
089klf  
KLF-089  
  
asf89jnf091msd324h5n  
ASFJNFLMSDHN-89093245
```

Κώδικας για την 4^η άσκηση: αρχείο ex4.asm

```
INCLUDE macros.asm

DATA SEGMENT
    ARR DB 20 DUP(?)
DATA ENDS

CODE SEGMENT
    ASSUME CS:CODE, DS:DATA

MAIN PROC FAR
    MOV AX, DATA
    MOV DS, AX

START:
    MOV CL, 0           ;counter
    MOV DI, 0
READCHAR:
    READ
    CMP AL, 61          ;if =
    JE FINISH           ;end program
    CMP AL, 13          ;if ENTER
    JE PREPRINT         ;prepare to print
    CMP AL, 48          ;if <0
    JB READCHAR         ;read next char
    CMP AL, 122         ;if >z
    JA READCHAR         ;read next char
    CMP AL, 57          ;if <=9
    JBE STORECHAR       ;jump to store char
    CMP AL, 97          ;if <a
    JB READCHAR         ;read next char
STORECHAR:
    PRINT AL            ;print char
    MOV ARR[DI], AL     ;store it in ARR
    INC DI              ;increment ARR pointer
    INC CL              ;increment counter
    CMP CL, 20          ;if counter<20
    JB READCHAR         ;read more

PREPRINT:
    NEWLINE
    CMP CL, 0           ;if ARR is empty
    JE READCHAR         ;read more
    MOV DI, 0           ;move pointer to start
    MOV BL, CL          ;store count in BL
    MOV BH, 0

LETTERS:
    MOV AL, ARR[DI]     ;get char
    CMP AL, 97          ;if <a skip this char
    JB SKIPLET
    CMP AL, 122         ;if >z skip this char
    JA SKIPLET
    SUB AL, 32          ;convert to uppercase
```

```

    PRINT AL                ;print char
SKIPLET:
    INC DI                  ;increment pointer
    LOOP LETTERS            ;loop using counter

    PRINT "-"
    MOV CX,BX                ;store count in CX
    MOV DI,0                 ;move pointer to start

NUMBERS:
    MOV AL,ARR[DI]           ;get char
    CMP AL,48                 ;if <0 skip this char
    JB SKIPNUM
    CMP AL,57                 ;if >9 skip this char
    JA SKIPNUM
    PRINT AL                 ;print char
SKIPNUM:
    INC DI                   ;increment pointer
    LOOP NUMBERS             ;loop using counter

    NEWLINE
    NEWLINE
    JMP START                ;repeat

FINISH:
    EXIT
MAIN ENDP
CODE ENDS
END MAIN

```

5^η ΑΣΚΗΣΗ:

Από τη χαρακτηριστική καμπύλη Τάση εισόδου ADC/Έξοδος ADC, προκύπτει ότι:

$$V = V_{εξ,AD} = \frac{4095}{4} V_{εισ,AD}$$

Επομένως, μέσω της χαρακτηριστικής καμπύλης προκύπτουν οι εξής εξισώσεις:

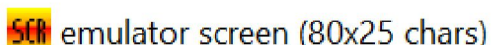
$$V = 2 \cdot \frac{4095}{4 \cdot 400} T \Rightarrow T = \frac{800}{4095}, \quad 0 \leq V \leq 2047$$

$$V = \frac{1200 - 400}{3 \cdot \frac{4095}{4} - 2 \cdot \frac{4095}{4}} T + 2 \cdot \frac{4095}{4} \Rightarrow T = \frac{3200}{4095} V - 1200, \quad 2048 \leq T \leq 3071$$

Αρχικά, εμφανίζεται στην οθόνη το μήνυμα “START(Y,N):”. Αν ο χρήστης δώσει ‘Y’ το πρόγραμμα θα λειτουργήσει, ενώ αν δώσει ‘N’ θα τερματίσει. Κατά τη εκκίνηση αναμένονται 3 έγκυρα HEX ψηφία, που αντιστοιχούν στην τάση εξόδου του ADC, ώστε να υπολογιστεί βάσει των παραπάνω τύπων η αντίστοιχη θερμοκρασία και να εμφανιστεί στην οθόνη σε δεκαδική μορφή. Το πρόγραμμα είναι συνεχούς λειτουργίας, αλλά αν οποιαδήποτε στιγμή δοθεί ο χαρακτήρας ‘N’, τερματίζεται. Για τιμές θερμοκρασίας μεγαλύτερες από 1200°C να εμφανίζεται το μήνυμα σφάλματος “ERROR”.

Υλοποιήσαμε τις ρουτίνες HEX_KEYB για την εισαγωγή δεκαεξαδικού ψηφίου και PRINT_DEC για την μετατροπή και εμφάνιση του αντίστοιχου δεκαδικού αριθμού που αντιστοιχεί στο ακέραιο μέρος. Το ακέραιο μέρος αντιστοιχεί στο πηλίκο της διαίρεσης με το 4095, το οποίο βρίσκουμε μέσω της εντολής DIV. Για το δεκαδικό μέρος πολλαπλασιάζουμε το υπόλοιπο (που είναι αποθηκευμένο στον DX) με 10 και κρατάμε το πηλίκο της διαίρεσης με το 4095. Το κλασματικό μέρος είναι μονοψήφιο οπότε αρκεί να το αντιστοιχίσουμε σε δεκαδικό μέσω του κώδικα ASCII.

Παράδειγμα εξόδου:

 emulator screen (80x25 chars)

```
START<Y,N>:Y
3E5      194.7
E19      ERROR
DFF      ERROR
BFF      1199.8
124      57.0
700      350.0
701      350.2
```


Κώδικας για την 5^η άσκηση: αρχείο *ask5.asm*

```
INCLUDE macros.asm
DATA_SEG SEGMENT
    MSG1 DB "START(Y,N):$"
    MSG2 DB "ERROR$"
ENDS

CODE_SEG SEGMENT
    ASSUME CS:CODE_SEG, DS:DATA

MAIN PROC FAR
    MOV AX, DATA_SEG
    MOV DS, AX
    PRINT_STR MSG1
START:
    READ
    CMP AL, 'N'
    JE QUIT
    CMP AL, 'Y'
    JE ANS
    JMP START
ANS:
    PRINT AL
    NEWLINE
INPUT:
    MOV DX, 0
    CALL HEX_KEYB
    CMP AL, 'N'
    JE QUIT
    MOV DL, AL
    ROL DL, 4
    CALL HEX_KEYB
    CMP AL, 'N'
    JE QUIT
    OR DL, AL
    ROL DX, 4
    CALL HEX_KEYB
    CMP DL, 'N'
    JE QUIT
    OR DL, AL

    TABS
    MOV AX, DX
    CMP AX, 2047
    JBE FIRST_CASE
    CMP AX, 3071
    JBE SECOND_CASE
    PRINT_STR MSG2
    NEWLINE
    JMP NEXT

FIRST_CASE:
```

```

    MOV BX, 800
    MUL BX
    MOV BX, 4095
    DIV BX
    JMP SHOW

SECOND_CASE:
    MOV BX, 3200
    MUL BX
    MOV BX, 4095
    DIV BX
    SUB AX, 1200
SHOW:
    CALL PRINT_DEC

    MOV AX, DX
    MOV BX, 10
    MUL BX
    MOV BX, 4095
    DIV BX
    PRINT ', '
    ADD AL, 48
    PRINT AL
    NEWLINE
    JMP NEXT

QUIT:
    PRINT AL
    EXIT
MAIN ENDP

HEX_KEYB PROC NEAR
    IGNORE:
        READ
        CMP AL, 'N'
        JE RETURN
        CMP AL, 48      ;48, ('O'=30H)
        JL IGNORE      ;<0
        CMP AL, 57      ;57, ('9'=39H)
        JG ISLETTER     ;>9
        PRINT AL
        SUB AL, 48      ;ASCII code
        JMP RETURN
    ISLETTER:
        CMP AL, 'A'
        JL IGNORE
        CMP AL, 'F'
        JG IGNORE
        PRINT AL
        SUB AL, 55      ;(55)ASCII to number
    RETURN:
        RET
HEX_KEYB ENDP

PRINT_DEC PROC NEAR

```

```
        PUSH DX
        MOV CX, 0
GET_DEC:
        MOV DX, 0
        MOV BX, 10
        DIV BX
        PUSH DX
        INC CX
        CMP AX, 0
        JNE GET_DEC
SHOW_DEC:
        POP DX
        ADD DL, 30H
        PRINT DL
        LOOP SHOW_DEC
        POP DX
        RET
PRINT_DEC ENDP
```

```
CODE_SEG ENDS
END MAIN
```