ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΜ&ΜΥ
Εργαστήριο Μικροϋπολογιστών

3η Εργαστηριακή Άσκηση
Ακ. έτος 2011-2012

Ομάδα C07:

Ελένη Ευαγγελάτου Α.Μ.: 03108050
Γρηγόρης Λύρας Α.Μ.: 03109687
Βασιλεία Φραγκιαδάκη Α.Μ.: 03108026

28 Δεκεμβρίου 2011

Άσκηση (i)

Σε αυτή την άσκηση ζητείται να διαβάζεται από το πληκτρολόγιο ένας δυαδικός αριθμός των δέκα bits και στη συνέχεια να εκτυπώνεται ο αντίστοιχος δεκαδικός. Γι' αυτό το σκοπό, διαβάζουμε τον δυαδικό αριθμό ανά ψηφίο και τον αποθηκεύουμε στον διπλό καταχωρητή DX, με ένα loop που εκτελείται δέκα φορές και κάθε φορά ολισθαίνει κατά μία θέση προς τα αριστερά τον DX προσθέτοντας ένα μόνο εάν το ψηφίο που διάβασε είναι 1. Στη συνέχεια ο δυαδικός αριθμός μετατρέπεται στο δεκαδικό σύστημα με διαίρεση αρχικά με 1000 για την εύρεση των χιλιάδων, με διαίρεση του υπολοίπου από την προηγούμενη διαίρεση με 100 για την εύρεση των εκατοντάδων και διαίρεση του υπολοίπου με 10 για την εύρεση των δεκάδων. Το υπόλοιπο είναι οι μονάδες. Κάθε φορά εκτυπώνεται ο αντίστοιχος αριθμός (με τον αντίστοιχο χαρακτήρα ASCII).
Κυρίως κώδικας:

```
INCLUDE MACROS.TXT
;============DATA SEGMENT===============
DATA SEGMENT
    MESSAGE1 DB "GIVE A 10-BINARY NUMBER: $"
    MESSAGE2 DB "DECIMAL: $"
    PKEY DB "PRESS ANY KEY TO START OR Q IN ORDER TO EXIT. $"
    NEW_LINE DB 0AH, 0DH, '$'    ;OI ASCII KWDIKOI GIA ALLAGH GRAMMHS
ENDS
;============STACK SEGMENT==============
STACK SEGMENT
    DW 128 DUP(?)
ENDS

;============CODE SEGMENT===============
CODE SEGMENT
    ASSUME CS:CODE, SS:STACK, DS:DATA, ES:DATA


MAIN PROC FAR

; SET SEGMENT REGISTERS:
    MOV AX, DATA
    MOV DS, AX
    MOV ES, AX

START:
    PRINT_STRING MESSAGE1       ;PRINT_STRING ME XRHSH TOY MACRO
    CALL BIN_KEYBOARD           ;O XRHSTHS DINEI TON BINARY
    PUSH DX
    PRINT_STRING NEW_LINE       ;ALLAGH GRAMMHS
    PRINT_STRING MESSAGE2
    POP DX
    CALL DEC_CONVERTION         ;TON METATREPW SE DEKADIKO
    PRINT_STRING NEW_LINE       ;ALLAGH GRAMMHS
    PRINT_STRING PKEY           ;ODHGIES PROS TON XRHSTH GIA TO TI NA PATHSEI
    READ                        ;DIABAZEI AUTO POU EDWSE O XRHSTHS
    CMP AL, 'Q'                 ;AN PATH8HKE TO Q
    JE QUIT                     ; TELOS PROGRAMMATOS
    CMP AL, 'q'                 ;AN PATH8HKE TO q
    JE QUIT                     ; TELOS PROGRAMMATOS
    PRINT_STRING NEW_LINE       ;ALLAGH GRAMMHS
    JMP START
QUIT:
    MOV AL,0H
    EXIT                        ;APO TO MACRO


MAIN ENDP
;========DIABASMA TOY BINARY APO TO KEYBOARD============
BIN_KEYBOARD  PROC NEAR
    MOV DX, 0
    MOV CX, 10   ;O CX EINAI DEFAULT COUNTER GIA LOOPS. 8ELW 10 NOUMERA NA DIABASW

IGNORE:
    READ         ;DIABAZEI XARAKTHRA APO PLHKTROLOGIO XWRIS NA TO TUPWSEI
    CMP AL, 'Q'  ;BLEPW AN EINAI Q
    JE QUIT      ;AN EINAI TOTE KANOUME EXIT
    CMP AL, 'q'  ;BLEPW AN EINAI Q
    JE QUIT      ;AN EINAI TOTE KANOUME EXIT
```

```
    SHL DX,1
    CMP AL, '0'  ;ALLIWS,BLEPW AN EINAI 0
    JE ZERO
    CMP AL, '1'  ;ALLIWS,BLEPW AN EINAI 1
    JE ONE
    JMP IGNORE
ONE:
    INC DX   ;GIA NA DIABASW 10 ARI8MOUS...
ZERO:
    LOOP IGNORE
ADDR2:
    RET

BIN_KEYBOARD ENDP

;=========METATROPH K PRINT SE DEKADIKO==================
DEC_CONVERTION PROC NEAR
    MOV AX,DX
    MOV DX,0
    ;O DX::AX EINAI O DEFAULT DIAIRETEOS
    MOV BX, 1000
    DIV BX       ; DIAIRW ME 1000
    PRINT_NUM AL ; TO AL EXEI TO PHLIKO DHLADH THN XILIADA
    MOV AX,DX    ; DIAIRETHS 8A GINEI TO PROHGOUMENO UPOLOIPO
    MOV DX,0
    MOV BX, 100  ; DIAIRW ME 100
    DIV BL
    PRINT_NUM AL ; TO AL 8A EXEI TO PHLIKO POU 8A NAI OI EKATONTADES
    MOV AL,AH    ; DIAIRETHS 8A GINEI TO PROHGOUMENO UPOLOIPO
    MOV AH,0
    MOV BX, 10   ; DIAIRW ME 10
    DIV BL
    PRINT_NUM AL ; TO PHLIKO EDW EXEI TIS DEKADES
    PRINT_NUM AH ; TO UPOLOIPO EDW EXEI TIS MONADES

    RET

DEC_CONVERTION ENDP
;============================================================
CODE ENDS

END MAIN
```

Τα macros που χρησιμοποιήσαμε:

```
;This macro change registers AH,AL
READ MACRO
    MOV AH,1
    INT 21H
ENDM

;This macro changes registers AH,DL
PRINT MACRO CHAR
    PUSH AX
    PUSH DX
    MOV DL,CHAR
    MOV AH,02H
    INT 21H
    POP DX
    POP AX
ENDM

;This macro change registers AH,DX
PRINT_STRING MACRO STRING
        PUSH AX
        PUSH DX
    MOV DX,OFFSET STRING ;Assume that string is a variable or constant, NOT an address
    MOV AH,09H
    INT 21H
    POP DX
    POP AX
ENDM

PRINT_NUM MACRO CHAR
    PUSH DX
    PUSH AX
```

```
    MOV DL, CHAR
    ADD DL, 30H
    MOV AH, 2
    INT 21H
    POP AX
    POP DX
ENDM

PAUSE MACRO
    PUSH AX
    PUSH DX
    LEA DX,PKEY         ;<=>MOV DX, OFFSET PKEY;GIVES THE OFFSET OF PKEY TO DX
    MOV AH,9
    INT 21H             ;OUTPUT STRING AT DS:DX
    MOV AH,8            ;WAIT FOR PRESSING OF A KEY
    INT 21H         ;WITHOUT ECHO->8
    PRINT 0AH
    PRINT 0DH
    POP DX
    POP AX
ENDM

EXIT MACRO
    MOV AH,4CH
    INT 21H
ENDM
```

Άσκηση (ii)

Στην άσκηση αυτή ζητείται τυπώνοντας τα κατάλληλα μηνύματα να διαβάζουμε ένα δεκαδικό αριθμό τεσσάρων ψηφίων, να τον εκτυπώνουμε και αν πατηθεί <enter> να τον μετατρέπουμε στον αντίστοιχο δεκαεξαδικό τον οποίο και να εκτυπώνουμε . Για το σκοπό αυτό, διαβάζουμε τον αριθμό ανά ψηφίο και τον αποθηκεύουμε τελικά στον καταχωρητή DP, με ένα loop που εκτελείται τέσσερις φορές (μία για κάθε ψηφίο), όπου δημιουργούμε τον αριθμό πολλαπλασιάζοντας κάθε φορά το προηγούμενο αποτέλεσμα επί 10 και προσθέτοντας το τρέχον ψηφίο. Στη συνέχεια, αφού πατηθεί <enter> τυπώνεται ο αντίστοιχος δεκαεξαδικός, τον οποίο υπολογίζουμε απομονώνοντας κάθε φορά στον καταχωρητή BL τα εκάστοτε τέσσερα ψηφία τα οποία αφού μετατρέψουμε στον αντίστοιχο δεκαεξαδικό χαρακτήρα τον εκτυπώνουμε στην οθόνη. Ξεκινάμε από τα τέσσερα MSB και προχωράμε προς τα τέσσερα LSB του BP. Με το χαρακτήρα "Q" ή "q" το πρόγραμμα τερματίζεται.
Κυρίως κώδικας:

```
INCLUDE MACROS.TXT

DATA SEGMENT
    ; ADD YOUR DATA HERE!
    PKEY DB "INSERT 4 DECIMAL NUMS AND THEN <ENTER>...$"
    MESSAGE1 DB "GIVE FOUR NUMBERS: $"
    MESSAGE2 DB "HEX = $"
    NEW_LINE DB 0AH, 0DH, '$' ;OI ASCII KWDIKOI GIA ALLAGH GRAMMHS
ENDS

STACK SEGMENT
    DW   128  DUP(?)
ENDS

CODE SEGMENT

MAIN PROC FAR
; SET SEGMENT REGISTERS:
    MOV AX, DATA
    MOV DS, AX
    MOV ES, AX

START:

    PRINT_STRING MESSAGE1
    CALL DEC_KEYBOARD   ; KATEU8EIAN ME TO READ TA BAZEIS STON BP A8ROIZONTAS
    PRINT_STRING NEW_LINE
BCK:
    READ
    CMP AL,0DH    ;koita gia enter
    JE CNT
    CMP AL,'Q'
    JE QUIT
```

```
    CMP AL,'q'
    JE QUIT
    JMP BCK

CNT:
    PRINT_STRING MESSAGE2
    ;TUPWSE TA HEX TOU 16BITOU BP
    CALL DIGITS_TO_HEXS
    PRINT_STRING NEW_LINE
    JMP START


QUIT:
    MOV AL,0H
    EXIT

MAIN ENDP


;==========PROCEDURES=============
DEC_KEYBOARD PROC NEAR
    MOV DX, 0
    MOV CX, 4 ;GIATI 8A DEXTW 4 ARI8MOUS

IGNORE:
    READ
    CMP AL,'Q'
    JE QUIT
    CMP AL,'q'
    JE QUIT
    CMP AL, '0'
    JL IGNORE
    CMP AL, '9'
    JG IGNORE
    SUB AL, 30H
    MOV BL,AL   ;APO8UKEUSE TO TREXON STON BL
    MOV BH,0
    MOV AX,DX   ; FORTWSE TO PROHGOUMENO A8ROISMA APO DX
    MOV DX,10   ;BALE STO DL 10
    MUL DX      ;AX=AX*10
    ADD AX,BX   ;+BL
    MOV DX,AX   ;KAI BALTO PALI STO DX
    LOOP IGNORE
    MOV BP,DX
    RET

DEC_KEYBOARD ENDP



;======MAKE 16 BITS TO HEX=======

DIGITS_TO_HEXS PROC NEAR
     ; KSEKINAME APO TA MSB KAI TYPWNOYME HEX ANA 4DES
      MOV BX, BP
      MOV BL, BH ;APOMONWNW TA 4 MSB
      SHR BL, 4  ;OLIS8HSE TA STIS 4 LEAST SIGNIF 8ESEIS
      CALL PRINT_HEX
      MOV BX, BP
      MOV BL,BH
      AND BL, 0FH
      CALL PRINT_HEX
      MOV BX, BP
      AND BL, 0F0H
      SHR BL, 4
      CALL PRINT_HEX
      MOV BX, BP
      AND BL, 0FH
      CALL PRINT_HEX


      RET
DIGITS_TO_HEXS ENDP

PRINT_HEX PROC NEAR
```

```
    CMP BL,9 ;AN O ARI8MOS EINAI METAKSU 0 K 9 PROS8ETW 30H
    JG ADDR1
    ADD BL, 30H
    JMP ADDR2

ADDR1:
    ADD BL, 37H ;DIAFORETIKA PROS8ETW 37H ('A' = 41H)
ADDR2:
    PRINT BL
    RET

PRINT_HEX ENDP
;======END OF MAKE 16 BITS TO HEX========


END MAIN
```

Τα macros που χρησιμοποιήσαμε:

```
;This macro change registers AH,AL
READ MACRO
    MOV AH,1
    INT 21H
ENDM

;This macro changes registers AH,DL
PRINT MACRO CHAR
    PUSH AX
    PUSH DX
    MOV DL,CHAR
    MOV AH,02H
    INT 21H
    POP DX
    POP AX
ENDM

;This macro change registers AH,DX
PRINT_STRING MACRO STRING
        PUSH AX
        PUSH DX
    MOV DX,OFFSET STRING ;Assume that string is a variable or constant, NOT an address
    MOV AH,09H
    INT 21H
    POP DX
    POP AX
ENDM

PRINT_NUM MACRO CHAR
    MOV DL, CHAR
    ADD DL, 30H
    MOV AH, 2
    INT 21H
ENDM

PAUSE MACRO
    PUSH AX
    PUSH DX
    LEA DX,PKEY        ;<=>MOV DX, OFFSET PKEY;GIVES THE OFFSET OF PKEY TO DX
    MOV AH,9
    INT 21H            ;OUTPUT STRING AT DS:DX
    MOV AH,8           ;WAIT FOR PRESSING OF A KEY
    INT 21H        ;WITHOUT ECHO->8
    PRINT 0AH
    PRINT 0DH
    POP DX
    POP AX
ENDM

EXIT MACRO
    MOV AH,4CH
    INT 21H
ENDM

GETHON MACRO R
    CALL GETHEX
    MOV R,AX
```

```
    CALL GETHEX
    SHL R,4
    OR R,AX
    CALL GETHEX
    SHL R,4
    OR R,AX
    CALL GETHEX
    SHL R,4
    OR R,AX
ENDM
```

Άσκηση (iii)

Εδώ ζητείται να διαβάζουμε το πολύ 20 χαρακτήρες από το πληκτρολόγιο και μετα το πατημα του <enter>, να τους εκτυπώνουμε στην έξοδο ομαδοποιημένους κατά αριθμούς, μικρά και κεφαλαία γράμματα αγνοώντας τα κενά. Έτσι, δεσμεύουμε χώρο 21 byte για κάθε μία από τις τρεις "ομάδες", που αρχικοποιούμε στο χαρακτήρα τερματισμού string "$" (το 21ο byte χρησιμεύει για να έχουμε χαρακτήρα τερματισμού στην περίπτωση που διαβάσουμε 20 χαρακτήρες ίδιας ομάδας). Στη συνέχεια διαβάζουμε τους χαρακτήρες αγνοώντας τους μη επιθυμητούς, μέσα σε ένα loop 20 επαναλήψεων, το οποίο σταματάει νωρίτερα σε περίπτωση που δεχτεί <enter>, και κάθε έναν τον αποθηκεύουμε στον αντίστοιχο πίνακα, αυξάνοντας κάθε φορά μία μεταβλητή που χρησιμεύει σαν δείκτης σ' αυτό το χώρο. Στη συνέχεια εκτυπώνουμε κάθε πίνακα διαδοχικά χρησιμοποιώντας τη μακροεντολή print_string. Το πρόγραμμα τερματίζεται αν δοθεί ο χαρακτήρας "/".
Κυρίως κώδικας:

```
INCLUDE MACROS.TXT

STACK_SEG SEGMENT STACK
    DW 128 DUP(?)
ENDS


DATA_SEG SEGMENT
    MSG DB "GIMME <=20 CHARS END PRESS RETURN '/' TO QUIT",0AH,0DH,"$"
    MSG2 DB " => $"
    SPACE DB " "
    LINE DB 0AH,0DH,"$"
    NUMS DB 21 DUP("$")
    NCNT DW 0
    LOWC DB 21 DUP("$")
    LCNT DW 0
    UPRC DB 21 DUP("$")
    UCNT DW 0


ENDS

CODE_SEG SEGMENT
    ASSUME CS:CODE_SEG,SS:STACK_SEG,DS:DATA_SEG,ES:DATA_SEG

MAIN PROC FAR
;FOR SEGMENT REGISTERS
    MOV AX,DATA_SEG
    MOV DS,AX
    MOV ES,AX

START:
    PRINT_STRING MSG
    MOV DX,0
    MOV BX,0
    CALL GET_INPUT
CNT:
    PRINT_STRING MSG2
    PRINT_STRING NUMS
    PRINT SPACE
    PRINT_STRING LOWC
    PRINT SPACE
    PRINT_STRING UPRC
    PRINT_STRING LINE
    JMP START

EX:
    MOV AL,0H
    EXIT
```

```
MAIN ENDP


GET_INPUT PROC NEAR
    MOV DX,0
    MOV CX,20
READL:
    READ
    CMP AL,0DH
    JE CNT
    CMP AL,'/'
    JE EX
    CMP AL,30H ;0
    JL READL
    CMP AL,40H ;9+1
    JL NUMBERS
    CMP AL,41H ;A
    JL READL
    CMP AL,5BH ;Z+1
    JL ULETTER
    CMP AL,61H ;a
    JL  READL
    CMP AL,7BH ;z+1
    JL LLETTER
    JMP READL
NUMBERS:
    MOV BX,OFFSET NUMS
    ADD BX,NCNT
    MOV [BX] ,AL
    INC NCNT
    LOOP READL
    RET
LLETTER:
    MOV BX,OFFSET LOWC
    ADD BX,LCNT
    MOV [BX] ,AL
    INC LCNT
    LOOP READL
    RET
ULETTER:
    MOV BX,OFFSET UPRC
    ADD BX,UCNT
    MOV [BX] ,AL
    INC UCNT
    LOOP READL
    RET
GET_INPUT ENDP


CODE_SEG ENDS

END MAIN
```

Τα macros που χρησιμοποιήσαμε:

```
;This macro change registers AH,AL
READ MACRO
    MOV AH,1
    INT 21H
ENDM

;This macro changes registers AH,DL
PRINT MACRO CHAR
    PUSH AX
    PUSH DX
    MOV DL,CHAR
    MOV AH,02H
    INT 21H
    POP DX
    POP AX
ENDM

;This macro change registers AH,DX
PRINT_STRING MACRO STRING
        PUSH AX
        PUSH DX
```

```
    MOV DX,OFFSET STRING ;Assume that string is a variable or constant, NOT an address
    MOV AH,09H
    INT 21H
    POP DX
    POP AX
ENDM

PRINT_NUM MACRO CHAR
    MOV DL, CHAR
    ADD DL, 30H
    MOV AH, 2
    INT 21H
ENDM

PAUSE MACRO
    PUSH AX
    PUSH DX
    LEA DX,PKEY         ;<=>MOV DX, OFFSET PKEY;GIVES THE OFFSET OF PKEY TO DX
    MOV AH,9
    INT 21H             ;OUTPUT STRING AT DS:DX
    MOV AH,8            ;WAIT FOR PRESSING OF A KEY
    INT 21H         ;WITHOUT ECHO->8
    PRINT 0AH
    PRINT 0DH
    POP DX
    POP AX
ENDM

EXIT MACRO
    MOV AH,4CH
    INT 21H
ENDM
```

Άσκηση (iv)

Κυρίως κώδικας:

```
INCLUDE MACROS.TXT

STACK_SEG SEGMENT STACK
    DW 128 DUP(?)
ENDS


DATA_SEG SEGMENT
    FIRST DB "First number: $"
    SECOND DB "Second number: $"
    SPACE DB " "
    LINE DB 0AH,0DH,"$"


ENDS

CODE_SEG SEGMENT
    ASSUME CS:CODE_SEG,SS:STACK_SEG,DS:DATA_SEG,ES:DATA_SEG

MAIN PROC FAR
    MOV AX,DATA_SEG
    MOV DS,AX
    MOV ES,AX
    CALL GET_INPUT
    MOV AX,BX   ;AX=X0
    MOV DX,0    ;DX=0
    MUL SI      ;DX::AX = X0*Y0
    MOV BP,AX
    ; == 16 LSB in BP
    PUSH BP     ; now pushed
    PUSH CX     ; don't need X1 right now
    MOV CX,DX   ; so use it as buffer
    MOV AX,BX   ;AX=X0
    MOV DX,0    ;DX=0
    MUL DI      ;DX::AX = X0*Y1
    ADD AX,CX   ;AX+=previous DX
    JNC NOTOVF1
    INC DX      ;if carry increase DX
```

```
NOTOVF1:
    POP CX      ; will need X1
    MOV BX,DX   ; X0 is no longer needed
    PUSH BX     ; save DX
    MOV BX,AX   ; save AX
    MOV AX,CX   ; AX = X1
    MOV DX,0    ; DX = 0
    MUL SI      ; DX::AX = X1*Y0
    ADD AX,BX   ; AX+=previous AX
    JNC NOTOVF2 ; if carry increase DX
    INC DX
NOTOVF2:
    POP BX
    MOV BP,AX    ;2ND DIGIT
    PUSH BP
    MOV AX,CX
    MOV CX,DX  ; REALLY??
    MOV DX,0
    MUL DI
    ADD AX,BX
    JNC NOTOVF3
    INC DX
NOTOVF3:
    ADD AX,CX
    JNC NOTOVF4
    INC DX
NOTOVF4:
    MOV BP,AX
    PUSH BP
    MOV BP,DX
    ;now BP has the answer
    CALL DIGITS_TO_HEXS
    POP BP
    CALL DIGITS_TO_HEXS
    POP BP
    CALL DIGITS_TO_HEXS
    POP BP
    CALL DIGITS_TO_HEXS


    MOV AL,0H
    EXIT
MAIN ENDP

GET_INPUT PROC NEAR
    PRINT_STRING FIRST
    GETHON CX
    GETHON BX
    PRINT_STRING LINE
    PRINT_STRING SECOND
    GETHON DI
    GETHON SI
    PRINT_STRING LINE
    RET
GET_INPUT ENDP


GETHEX PROC NEAR
R:  READ
    MOV AH,0
    CMP AL,30H ;0
    JL R
    CMP AL,40H ;9+1
    JL NUM
    CMP AL,41H ;A
    JL R
    CMP AL,47H ;F+1
    JL CAPS
    CMP AL,61H ;a
    JL  R
    CMP AL,67H ;f+1
    JL  SMALL
    JMP R
NUM:
    SUB AL,30H
```

```
    RET
CAPS:
    SUB AL,37H
    RET
SMALL:
    SUB AL,57H
    RET

GETHEX ENDP


;======MAKE 16 BITS TO HEX=======

DIGITS_TO_HEXS PROC NEAR
     MOV BX, BP
     MOV BL, BH ;APOMONWNW TA 4 MSB
     SHR BL, 4  ;OLIS8HSE TA STIS 4 LEAST SIGNIF 8ESEIS
     CALL PRINT_HEX
     MOV BX, BP
     MOV BL,BH
     AND BL, 0FH
     CALL PRINT_HEX
     MOV BX, BP
     AND BL, 0F0H
     SHR BL, 4
     CALL PRINT_HEX
     MOV BX, BP
     AND BL, 0FH
     CALL PRINT_HEX


     RET
DIGITS_TO_HEXS ENDP

PRINT_HEX PROC NEAR
    CMP BL,9 ;AN O ARI8MOS EINAI METAKSU 0 K 9 PROS8ETW 30H
    JG ADDR1
    ADD BL, 30H
    JMP ADDR2

ADDR1:
    ADD BL, 37H ;DIAFORETIKA PROS8ETW 37H ('A' = 41H)
ADDR2:
    PRINT BL
    RET

PRINT_HEX ENDP
;======END OF MAKE 16 BITS TO HEX========









CODE_SEG ENDS

END MAIN
```

Τα macros που χρησιμοποιήσαμε:

```
;This macro change registers AH,AL
READ MACRO
    MOV AH,1
    INT 21H
ENDM

;This macro changes registers AH,DL
PRINT MACRO CHAR
    PUSH AX
    PUSH DX
    MOV DL,CHAR
    MOV AH,02H
    INT 21H
    POP DX
    POP AX
```

```
ENDM

;This macro change registers AH,DX
PRINT_STRING MACRO STRING
        PUSH AX
        PUSH DX
    MOV DX,OFFSET STRING ;Assume that string is a variable or constant, NOT an address
    MOV AH,09H
    INT 21H
    POP DX
    POP AX
ENDM

PRINT_NUM MACRO CHAR
    MOV DL, CHAR
    ADD DL, 30H
    MOV AH, 2
    INT 21H
ENDM

PAUSE MACRO
    PUSH AX
    PUSH DX
    LEA DX,PKEY        ;<=>MOV DX, OFFSET PKEY;GIVES THE OFFSET OF PKEY TO DX
    MOV AH,9
    INT 21H            ;OUTPUT STRING AT DS:DX
    MOV AH,8           ;WAIT FOR PRESSING OF A KEY
    INT 21H        ;WITHOUT ECHO->8
    PRINT 0AH
    PRINT 0DH
    POP DX
    POP AX
ENDM

EXIT MACRO
    MOV AH,4CH
    INT 21H
ENDM

GETHON MACRO R
    CALL GETHEX
    MOV R,AX
    CALL GETHEX
    SHL R,4
    OR R,AX
    CALL GETHEX
    SHL R,4
    OR R,AX
    CALL GETHEX
    SHL R,4
    OR R,AX
ENDM
```