Εργαστήριο Μικροϋπολογιστών

3η Εργαστηριακή Άσκηση

Σημείωση

Για την ευκολότερη υλοποίηση των προγραμμάτων, χρησιμοποιήθηκαν δύο βοηθητικές βιβλιοθήκες **macros.asm** και **helpers.asm** τις οποίες παραθέτουμε παρακάτω:

macros.asm

```
; Print a single character.
PRINT MACRO CHAR
       PUSH DX
       PUSH AX
       MOV DL,CHAR
       MOV AH,2
       INT 21H
       POP AX
       POP DX
ENDM

; Print a string.
PRINT_STR MACRO MESSAGE
       PUSH DX
       PUSH AX
       MOV DX, OFFSET MESSAGE
       MOV AH,9
       INT 21H
       POP AX
       POP DX
ENDM
```

```
; Read and print input character.
READ MACRO
       MOV AH,1
       INT 21H
ENDM

; Read input character without printing.
READ_B MACRO
       MOV AH,8
       INT 21H
ENDM

; Exit to OS.
EXIT MACRO
       MOV AX,4C00H
       INT 21H
ENDM

; Isolate #(POS) nibble of AX in AL.
SINGLE MACRO POS
       SHR AX,POS
       SHR AX,POS
       SHR AX,POS
       SHR AX,POS
       AND AX,0FH
ENDM
```

helpers.asm

```
; Read a hexadecimal digit in AL. Ignore everything but 'Q'.
READHEX PROC NEAR
AGAINREADHEX:
       READ_B
       CMP AL,'Q'
       JE ENDREADHEX
       CMP AL,'0'
       JL AGAINREADHEX
       CMP AL,'9'
       JG READHEXLETTER
       SUB AL,'0'
       JMP ENDREADHEX
READHEXLETTER:
       CMP AL,'A'
       JL AGAINREADHEX
       CMP AL,'F'
       JG AGAINREADHEX
       SUB AL,55
ENDREADHEX:
       RET
READHEX ENDP
```

```
; A converter procedure. It executes (AX)/(BX), stores the result in CL and the
; remainder in AX.
CONVERT PROC NEAR
CONVAGAIN:
       CMP AX,BX
       JC CONVDONE
       SUB AX,BX
       INC CX
       JMP CONVAGAIN
CONVDONE:
       RET
CONVERT ENDP

; Convert HEX in AX to DEC. Accurate for up to 16-bit HEX numbers.
HEXTODEC PROC NEAR
       PUSH BX
       PUSH CX
       MOV CX,0

       MOV BX,1000
       CALL CONVERT
       SHL CX,4

       MOV BX,100
       CALL CONVERT
       SHL CX,4

       MOV BX,10
       CALL CONVERT
       SHL CX,4

       ADD CX,AX
       MOV AX,CX
       POP CX
       POP BX
       RET
HEXTODEC ENDP

; Convert DEC in AX to HEX. Accurate for up to 16-bit DEC numbers.
DECTOHEX PROC NEAR
       PUSH BX
       PUSH CX
       MOV CL,4
       MOV BX,0
DECTOHEXAGAIN:
       PUSH AX
       PUSH CX
       MOV CL,12
       SHR AX,CL
       POP CX
       AND AX,000FH
       PUSH CX
```

```
       ADD BX,BX                ; (BX) = 2*N
       MOV CX,BX                ; (CX) = 2*N
       ADD BX,BX                ; (BX) = 4*N
       ADD BX,BX                ; (BX) = 8*N
       ADD BX,CX                ; (BX) = 10*N
       POP CX
       ADD BX,AX
       POP AX
       PUSH CX
       MOV CL,4
       SHL AX,CL
       POP CX
       DEC CL
       JNZ DECTOHEXAGAIN

       MOV AX,BX
       POP CX
       POP BX
       RET
DECTOHEX ENDP

; A procedure that prints a single HEX digit.
PRINT_HEX PROC NEAR
       PUSH AX
       CMP AL,10
       JC DECD
       ADD AL,55
       PRINT AL
       JMP PRHXRET
DECD:
       ADD AL,48
       PRINT AL
PRHXRET:
       POP AX
       RET
PRINT_HEX ENDP

; Print 4 digits contained in AX. Ignore leading zeros.
PRINT_RESULT_HEX PROC NEAR
       PUSH BX
       PUSH CX
       PUSH AX

       MOV CL,4            ; Set CL to 4, for efficient ROR.
       MOV CH,4            ; Digits to be printed, default is 4.

       MOV BX,AX           ; Store the input in BX.
       ROR BX,CL           ; Move first digit to the 4 LSB bits in BH.
       MOV AH,0            ; Initialize non-zero flag to 0.
LOOPPRH:
       MOV AL,BH
       AND AL,0FH          ; Isolate current digit.
```

```
       CMP AH,0           ; Check if a non-zero digit has been printed.
       JNZ TYPEANYRH      ; If so, print unconditionally.
       CMP AL,0           ; Else. check if current is zero.
       JZ NEXTDRH         ; If so, proceed without printing.
       MOV AH,1           ; Else, set AH flag.
TYPEANYRH:
       CALL PRINT_HEX     ; If we got here, we print the hex digit contained in AL.
NEXTDRH:
       ROL BX,CL          ; Shift BX to get the next digit.
       DEC CH             ; Decrease digit counter,
       JNZ LOOPPRH        ; and repeat until 4 digits are printed.

       CMP AH,0           ; If the non-zero flag is never set,
       JNZ P4RETRH
       PRINT '0'          ; print zero.
P4RETRH:
       POP AX
       POP CX
       POP BX
       RET
PRINT_RESULT_HEX ENDP

; Print 3 OCT digits from AX.
PRINT_RESULT_OCT PROC NEAR
       PUSH BX
       PUSH CX
       PUSH AX

       MOV CL,3           ; Set CL for efficient shifting.
       MOV CH,3           ; Store the number of digits to be printed.

       MOV BX,AX          ; Store the input in BX.
       ROL BX,2           ; Move the first digit in the 4 LSB bits of BH.
       MOV AH,0           ; Initialize non-zero flag to 0.
LOOPPRO:
       MOV AL,BH
       AND AL,07H         ; Isolate current digit.
       CMP AH,0           ; Check if a non-zero digit has been printed.
       JNZ TYPEANYRO      ; If so, print unconditionally.
       CMP AL,0           ; Else. check if current is zero.
       JZ NEXTDRO         ; If so, proceed without printing.
       MOV AH,1           ; Else, set AH flag.
TYPEANYRO:
       CALL PRINT_HEX     ; If we got here, we print the hex digit contained in AL.
NEXTDRO:
       ROL BX,CL          ; Shift BX to get the next digit.
       DEC CH             ; Decrease digit counter,
       JNZ LOOPPRO        ; and repeat until 3 digits are printed.

       CMP AH,0           ; If the non-zero flag is never set,
       JNZ P4RETRO
       PRINT '0'          ; print zero.
```

```
P4RETRO:
       POP AX
       POP CX
       POP BX
       RET
PRINT_RESULT_OCT ENDP
```

Άσκηση 1

```
       INCLUDE macros.asm              ; Some basic helper macros.

STACK_SEG SEGMENT STACK
       DW 50 DUP(?)
STACK_SEG ENDS

DATA_SEG SEGMENT
       input_msg DB "GIVE 3 HEX DIGITS: $"
       output_msg DB "Decimal: $"
       quit_msg DB "QUIT$"
       NEW_LINE DB 0AH,0DH,'$'
DATA_SEG ENDS

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

INCLUDE helpers.asm             ; Some helper procedures.

READSOME PROC NEAR
AGAINREADSOME:
       READ_B
       CMP AL,'0'
       JL AGAINREADSOME
       CMP AL,'9'
       JG READSOMELETTER
       SUB AL,'0'
       JMP ENDREADSOME
READSOMELETTER:
       CMP AL,'A'
       JL AGAINREADSOME
       CMP AL,'F'
       JG AGAINREADSOME
       SUB AL,55
ENDREADSOME:
       RET
READSOME ENDP

PRINT_FRACTIONAL PROC NEAR; A procedure that produces the desired output.
       PUSH AX                   ; We store AX,DX
       PUSH DX
       MOV AX,DX                 ; DX contains the number we just read, but HEXTODEC
```

```
       SHR AX,4                ; requires it is stored in AX. We must also isolate
       CALL HEXTODEC           ; H1H0.
       MOV DX,AX               ; Backup the converted number in DX.

       MOV CL,3                ; (CL) = (digits to be printed)
LOPO1:
       CMP CL,0
       JE FIN1
       DEC CL
       SINGLE CL               ; A helper macro that singles out the digit at the
       CALL PRINT_HEX          ; position defined by CL.
       MOV AX,DX               ; Restore full number and repeat!
       JMP LOPO1
FIN1:
       PRINT '.'               ; Print the dot.

       POP DX                  ; Restore DX to contain the original HEX number.
       MOV BX,DX               ; Move it to BX...
       AND BX,0FH              ; and trim H1H0.
       MOV AX,625              ; 625 = 1/16*10000
       MUL BX                  ; (DX AX) = 625*(H-1)

       CALL HEXTODEC           ; Convert the number to decimal.
       MOV DX,AX               ; Back it up in DX.
       MOV CL,4                ; Now print 4 digits.
LOPO2:
       CMP CL,0
       JE FIN2
       DEC CL
       SINGLE CL
       CALL PRINT_HEX
       MOV AX,DX
       JMP LOPO2
FIN2:
       PRINT_STR NEW_LINE      ; Print a new line,
       POP AX                  ; restore AX...
       RET                     ; and return.

PRINT_FRACTIONAL ENDP

MAIN PROC FAR
       MOV AX,DATA_SEG
       MOV DS,AX
       MOV ES,AX

START:
       PRINT_STR input_msg
       MOV BL,3                ; Read 3 hex digits.
       MOV CL,4                ; Preload CL with 4, to shift bits efficiently.

       MOV DX,0
MORE:
```

```
       CMP BL,1                ; Once we read two HEX digits, we print the dot.
       JNE SKIP
       PRINT '.'
SKIP:
       CALL READSOME           ; Helper procedure that reads a HEX from keyboard,
       CALL PRINT_HEX          ; ignoring any irrelevant input. We then print it.
       SHL DX,CL               ; Shift the accumulated number to position
       AND AX,0FH              ; Clean up the HEX digits in (AH)
       ADD DX,AX               ; And add the resulted number to the accumulator DX.
       DEC BL                  ; One less digit to read! :)
       JNZ MORE

       PRINT_STR NEW_LINE

       CMP DX,0E12H            ; If our group name (E12) is entered, we quit.
       JZ QUIT

       PRINT_STR output_msg
       CALL PRINT_FRACTIONAL   ; A procedure that expects the desired number in DX
                               ; and prints in proper format.
       JMP START

QUIT:
       PRINT_STR quit_msg      ; Print an exit message...
       EXIT                    ; and return control to the OS.
MAIN ENDP

CODE_SEG ENDS
       END MAIN
```

Άσκηση 2

```
       INCLUDE macros.asm      ; Some basic helper macros.

STACK_SEG SEGMENT STACK
       DW 50 DUP(?)
STACK_SEG ENDS

DATA_SEG SEGMENT
       input_msg DB "GIVE 2 DECIMAL DIGITS: $"
       output_msg DB "OCTAL= $"
       quit_msg DB "QUIT$"
       NEW_LINE DB 0AH,0DH,'$'
DATA_SEG ENDS

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

INCLUDE helpers.asm            ; Some helper procedures.
```

```
READSOME PROC NEAR

AGAINREADSOME:                  ; A procedure to read decimal digits, 'Q' and ENTER.
       READ_B
       CMP AL,0DH
       JE ENDREADSOME
       CMP AL,'Q'
       JE ENDREADSOME
       CMP AL,'0'
       JL AGAINREADSOME
       CMP AL,'9'
       JG AGAINREADSOME
       SUB AL,'0'
ENDREADSOME:
       RET

READSOME ENDP

MAIN PROC FAR
       MOV AX,DATA_SEG
       MOV DS,AX
       MOV ES,AX

       MOV DX,0                 ; DX stores the decimal number represented by the
       MOV CL,4                 ; last two digits. We set CL for efficient SHR.

START:
       PRINT_STR input_msg
       MOV CH,0                 ; (CH) = digits read
READING:
       CALL READSOME            ; Read a single character.
       CMP AL,'Q'               ; If it's 'Q', we quit.
       JE QUIT
       CMP AL,0DH               ; If it's ENTER,
       JNE SKIP
       CMP CH,2                 ; ...check if more than 2 decimals have been read.
       JL READING               ; If not, ignore the ENTER.
       JMP PRINTOCT             ; Else, calculate the output and print it.
SKIP:
       CMP CH,2                 ; Store min(2,digits read) in CH.
       JE SKIPINC
       INC CH
SKIPINC:                        ; We reach this part if we have a decimal digit.
       CALL PRINT_HEX           ; We print it.
       SHL DX,CL                ; Then we adjust DX accordingly.
       ADD DX,AX
       AND DX,255

       JMP READING              ; Keep on reading digits.

PRINTOCT:                       ; Print the converted number.
       PRINT_STR NEW_LINE       ; Formatting...
```

```
       PRINT_STR output_msg
       MOV AX,DX                ; Move DX to AX.
       CALL DECTOHEX            ; Convert it to HEX (for personal ease).
       CALL PRINT_RESULT_OCT    ; Print it in Octal.
       PRINT_STR NEW_LINE
       JMP START                ; Return to start.

QUIT:
       PRINT_STR NEW_LINE       ; Print a new line,
       PRINT_STR quit_msg       ; an exit message...
       EXIT                     ; and return control to the OS.
MAIN ENDP

CODE_SEG ENDS
       END MAIN
```

Άσκηση 3

```
       INCLUDE macros.asm       ; Some basic helper macros.

STACK_SEG SEGMENT STACK
       DW 50 DUP(?)
STACK_SEG ENDS

DATA_SEG SEGMENT
       TABLE DB 14 DUP(?)       ; 14 byte table.
       quit_msg DB "QUIT$"
       NEW_LINE DB 0AH,0DH,'$'
DATA_SEG ENDS

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

;; Returns only when an allowed character is pressed.
;; Ignored the rest withouht printing.
READ_STD PROC NEAR
NOTREADY:
       READ_B
       CMP AL,'='               ; If '=' was pressed
       JE RSTFIN
       CMP AL,0DH               ; returns immediately.
       JE RSTFIN
       CMP AL,' '               ; If ' ' was pressed
       JNE NUMBER
       PRINT AL                 ; we print it
       JMP RSTFIN               ; and return.
NUMBER:
       CMP AL,'0'               ; Check if number.
       JL NOTREADY              ; if lower than'0', read next.
       CMP AL,'9'               ; If greater than '9',
       JG CALPH                 ; Check if capital letter.
```

```
       PRINT AL                  ; if not, number in [0,9], print,
       JMP RSTFIN                ; return.
CALPH:
       CMP AL,'A'                ; Same for 'A'-'Z'.
       JL NOTREADY
       CMP AL,'Z'
       JG SALPH
       PRINT AL
       JMP RSTFIN
SALPH:
       CMP AL,'a'                ; Same for 'a'-'z'.
       JL NOTREADY
       CMP AL,'z'
       JG NOTREADY               ; Not allowed char, wait to read next.
       PRINT AL
RSTFIN:
       RET
READ_STD ENDP

;; Prints from TABLE the characters in BL-BH.
TYPE_IF PROC NEAR
       MOV SI,0                  ; Set destination counter.
REPET:
       CMP SI,CX                 ; Check if all CX input characters have been read.
       JZ FINTI                  ; If yes, return.
       MOV AL,[BP+SI]
       CMP AL,BL                 ; Check if char is
       JL NBCH                   ; lower than BL
       CMP AL,BH                 ; or greater than BH.
       JG NBCH                   ; If yes, skip printing.
       PRINT AL
NBCH:
       INC SI                    ; Increase counter...
       JMP REPET                 ; and repeat.
FINTI:
       RET
TYPE_IF ENDP

; Printing of two greater numbers (in order of precedence).
TWOBIG PROC NEAR
       MOV DL,-1                 ; Second greatest number in DL.
       MOV DH,-1                 ; Greatest in DH (not ASCII).
       MOV SI,0                  ; Reset destination counter.
       MOV AH,0                  ; Reset precedence counter (Values 0:DH - 1:DL).
REPAT:
       CMP SI,CX                 ; Check if all CX chars have been read.
       JZ FINAL                  ; if yes, go to final printing.
       MOV AL,[BP+SI]
       CMP AL,'0'                ; Check if number.
       JL NBGH
       CMP AL,'9'
       JG NBGH                   ; If not, move to next.
```

```
       SUB AL,48    ; If yes, transform it from ASCII to number (deASCIIfication).
       CMP AL,DH                ; Compare it with Greatest (DH).
       JL SECCH                 ; If not greater, compare it with Second Greatest
(DL).
       MOV DL,DH                ; If yes, move Greatest in Second Greatest.
       MOV DH,AL                ; And this in Greatest.
       MOV AH,0                 ; Also, set DL as prior (Newer = Bigger).
       JMP NBGH                 ; Move to next character.
SECCH:
       CMP AL,DL                ; Compare with Segond Greatest.
       JL NBGH                  ; If lower, move to next character,
       MOV DL,AL                ; If not, replace.
       MOV AH,1                 ; and set DH as prior (Newer = Smaller)
NBGH:
       INC SI                   ; Increase counter
       JMP REPAT                ; move to next character of table TABLE.
FINAL:
       CMP DH,0                 ; Check if any number was pressed (if not, DH=-1).
       JL ENDTWO                ; If not, print nothing.
       CMP AH,1                 ; Check if Second Greatest (small) is new (if yes,
       JE NEW_SMALL             ; it obviously exists)
       CMP DL,0                 ; Check if small exists.
       JL SKIP_SMALL            ; If not, only one number was pressed.
       ADD DL,48                ; If yes, it exists.
       PRINT DL
SKIP_SMALL:
       ADD DH,48                ; ASCIIfication and printing in precedence order.
       PRINT DH
       JMP ENDTWO
NEW_SMALL:
       ADD DH,48                ; ASCIIfication and printing in precedence order.
       PRINT DH
       ADD DL,48
       PRINT DL
ENDTWO:
       RET
TWOBIG ENDP


MAIN PROC FAR
       MOV AX,DATA_SEG
       MOV DS,AX
       MOV ES,AX
       MOV BP,OFFSET TABLE      ; Address of input TABLE is saved in base register
BP

; MAIN PROGRAM
; Reads as many as 14 latin characters, numbers or spaces
; and then prints them in groups, as requested.
; The two biggest numbers are printed in the last line.
; Terminates if '=' is pressed.
START:
```

```
; Input reading and saving on the TABLE.
      MOV DI,0                  ; Initialize DI, destination counter.
READING:
      CALL READ_STD             ; Reading one of the allowed characters.
      CMP AL,'='                ; If '=', quit.
      JE QUIT
      CMP AL,0DH                ; If Enter
      JE DISPLAY                ; move to Display.
      CMP DI,14                 ; If counter = 14
      JZ READING                ; then returns and doesn't save anything more
      MOV [BP+DI],AL            ; else, the characteris saved in it's slot
      INC DI                    ; and the counter is increaded by 1.
      JMP READING               ; Constant repeating until Enter (or '=') is
pressed.

; Display input in requested form.
DISPLAY:
      PRINT_STR NEW_LINE
      MOV CX,DI                 ; Save the current size of input in CX. (From DI)

; TYPE_IF prints from input, only the characters in the beadth BL-BH.
      MOV BL,'0'          ; The values 'a', 'z' are put in BL and BH respectively,
      MOV BH,'9'                ;      so that only numbers between 0 and 9 are
printed.
      CALL TYPE_IF              ; Through TYPE_IF.
      PRINT ' '                 ; Space between groups.
      MOV BL,'A'                ; Equally for capital letters.
      MOV BH,'Z'
      CALL TYPE_IF
      PRINT ' '
      MOV BL,'a'                ; Equally for small letters.
      MOV BH,'z'
      CALL TYPE_IF

      PRINT_STR NEW_LINE
      CALL TWOBIG               ; TWOBIG prints the two Greatest input numbers.

      PRINT_STR NEW_LINE
      JMP START                 ; constant repetition.

QUIT:
      PRINT_STR NEW_LINE
      PRINT_STR quit_msg
      EXIT
MAIN ENDP

CODE_SEG ENDS
      END MAIN
```

Άσκηση 4

```
       INCLUDE macros.asm        ; Some basic helper macros.

STACK_SEG SEGMENT STACK
       DW 50 DUP(?)
STACK_SEG ENDS

DATA_SEG SEGMENT
       input_msg DB "GIVE 2 DECIMAL DIGITS: $"
       output_msg DB "OCTAL= $"
       quit_msg DB "QUIT$"
       NEW_LINE DB 0AH,0DH,'$'
DATA_SEG ENDS

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

INCLUDE helpers.asm              ; Some helper procedures.

READSOME PROC NEAR

AGAINREADSOME:                   ; A procedure to read decimal digits, Q, +, -, =.
       READ_B
       CMP AL,'='
       JE ENDREADSOME
       CMP AL,'+'
       JE ENDREADSOME
       CMP AL,'-'
       JE ENDREADSOME
       CMP AL,'Q'
       JE ENDREADSOME
       CMP AL,'0'
       JL AGAINREADSOME
       CMP AL,'9'
       JG AGAINREADSOME
       SUB AL,'0'
ENDREADSOME:
       RET

READSOME ENDP

MAIN PROC FAR
       MOV AX,DATA_SEG
       MOV DS,AX
       MOV ES,AX

       MOV CL,4                  ; We set CL for efficient SHR.

START:
       MOV BX,0                  ; Store the 2 numbers in BX,DX.
       MOV DX,0
```

```
       MOV CH,0                ; (CH) = digits read

FIRST:
       CALL READSOME           ; Read a single character.
       CMP AL,'Q'              ; If it's 'Q', we quit.
       JE QUIT
       CMP AL,'='
       JE FIRST
       CMP AL,'+'              ; If it's '+', then...
       JNE CONT1
       CMP CH,0                ; check if at least one digit has been read.
       JG ADDTHEM              ; If so add it with the next number.
       JMP FIRST               ; Else ignore the input.
CONT1:
       CMP AL,'-'              ; Likewise for '-'.
       JNE CONT2
       CMP CH,0
       JG SUBTHEM
       JMP FIRST
CONT2:
       AND AX,0FH              ; If we get there, a new decimal digit was entered.
       CALL PRINT_HEX          ; We print it.
       SHL DX,CL               ; Then we adjust DX accordingly.
       ADD DX,AX
       INC CH
       CMP CH,3                ; Check if more than 3 decimals have been read.
       JL FIRST                ; If not, allow reading more.

OPERATION:                     ; We have read 3 decimal digits and we are only
       CALL READSOME           ; expecting an operation symbol.
       CMP AL,'Q'
       JE QUIT
       CMP AL,'+'              ; Check if '+' was pressed...
       JE ADDTHEM              ; and jump accordingly.
       CMP AL,'-'              ; Likewise for '-'.
       JE SUBTHEM
       JMP OPERATION

ADDTHEM:
       PRINT AL                ; Print the operation symbol,
       MOV AX,0                ; set the (AX) flag to 0...
       JMP DONE                ; and move on.
SUBTHEM:
       PRINT AL
       MOV AX,1
DONE:
       PUSH AX                 ; Store the operation flag in the stack.

       MOV CH,0
SECOND:
       CALL READSOME           ; Read a single character.
       CMP AL,'Q'              ; If it's 'Q', we quit.
```

```
       JE QUIT
       CMP AL,'+'
       JE SECOND
       CMP AL,'-'
       JE SECOND
       CMP AL,'='              ; If '=' was pressed, then...
       JNE CONT3
       CMP CH,0                ; check if at least one digit has been read.
       JE SECOND               ; If not, ignore the input.
       JMP EQUALSENTERED       ; Else display the output.

CONT3:
       AND AX,0FH              ; If we get there, a new decimal digit was entered.
       CALL PRINT_HEX          ; We print it.
       SHL BX,CL               ; Then we adjust DX accordingly.
       ADD BX,AX
       INC CH
       CMP CH,3                ; ...check if more than 2 decimals have been read.
       JL SECOND               ; If not, allow to read more.

EQUALS:                        ; We have read 3 decimal digits and we are only
       CALL READSOME           ; expecting the '=' symbol to print the output.
       CMP AL,'Q'
       JE QUIT
       CMP AL,'='
       JNE EQUALS
EQUALSENTERED:                 ; Once '=' is properly entered,
       PRINT '='               ; print it.
       MOV AX,BX               ; Convert both numbers to HEX.
       CALL DECTOHEX
       MOV BX,AX
       MOV AX,DX
       CALL DECTOHEX
       MOV DX,AX

       MOV CH,0                ; Initialize the sign flag.
       POP AX                  ; Restore AX to check the operation flag.
       CMP AL,0
       JE SKIP
       NEG BX                  ; If '-' was pressed, we set (BX) = -(BX)
SKIP:
       ADD DX,BX               ; (DX) = (DX) + (BX)
       MOV AX,DX               ; Store the result in AX.
       CMP AX,0FFFFh           ; Check for its sign.
       JG SKIP2
       MOV CH,1                ; If it's negative, set the sign flag...
       PRINT '-'               ; and print '-'.
       NEG AX                  ; Negate the result, for proper printing.
SKIP2:
       CALL PRINT_RESULT_HEX   ; Print the result in HEX.

       PRINT '='
```

```
      CMP CH,1                ; Check if the sign flag is set.
      JNE SKIP3
      PRINT '-'               ; If so, print '-' again.
SKIP3:
      CALL HEXTODEC           ; Convert AX to decimal.
      CALL PRINT_RESULT_HEX   ; Print it.
      PRINT_STR NEW_LINE      ; Print a new line...

      JMP START               ; and start from scratch.

QUIT:
      PRINT_STR NEW_LINE      ; Print a new line,
      PRINT_STR quit_msg      ; an exit message...
      EXIT                    ; and return control to the OS.
MAIN ENDP

CODE_SEG ENDS
      END MAIN
```