** Write an assembly language program to divide two numbers.

```
include 'emu8086.inc'
.MODEL SMALL
.STACK 100H
.DATA

   NUM1 DW ?            ; First number
   NUM2 DW ?            ; Second number
   RESULT DW ?          ; Result


.CODE

START:
   MOV AX, @DATA
   MOV DS, AX

   ; Read first single-digit number
   print 'Enter NUM1: '
   MOV AH, 1            ; Set up for reading a character
   INT 21H              ; Read character
   SUB AL, '0'          ; Convert ASCII to integer
   MOV AH, 0            ; Clear the upper byte
   MOV NUM1, AX         ; Store the result in NUM1

   ; new line

   MOV AH, 2         ; DOS function to display a character
   MOV DL, 0Dh       ; Carriage return (ASCII 13)
   INT 21H           ; Call DOS interrupt
   MOV DL, 0Ah       ; Line feed (ASCII 10)
   INT 21H

   ; Read second single-digit number
   print 'Enter NUM2: '
   MOV AH, 1            ; Set up for reading a character
   INT 21H              ; Read character
   SUB AL, '0'          ; Convert ASCII to integer
   MOV AH, 0            ; Clear the upper byte
   MOV NUM2, AX         ; Store the result in NUN2



   MOV AX, NUM1         ; Load NUM1 into AX
   MOV DX, 0            ; Clear DX for 16-bit
   ADD AX, NUM2         ; Load NUN2 into BX

   MOV RESULT, AX       ; Store  RESULT


   ; Exit program
   MOV AH, 4CH
   INT 21H

END START
```

** Write an assembly language program to SUB two numbers.

```
include 'emu8086.inc'
.MODEL SMALL
.STACK 100H
.DATA

   NUM1 DW ?           ; First number
   NUM2 DW ?           ; Second number
   RESULT DW ?         ; Result


.CODE

START:
   MOV AX, @DATA
   MOV DS, AX

   ; Read first single-digit number
   print 'Enter NUM1: '
   MOV AH, 1           ; Set up for reading a character
   INT 21H             ; Read character
   SUB AL, '0'         ; Convert ASCII to integer
   MOV AH, 0           ; Clear the upper byte
   MOV NUM1, AX        ; Store the result in NUM1

   ; new line

   MOV AH, 2         ; DOS function to display a character
   MOV DL, 0Dh       ; Carriage return (ASCII 13)
   INT 21H           ; Call DOS interrupt
   MOV DL, 0Ah       ; Line feed (ASCII 10)
   INT 21H

   ; Read second single-digit number
   print 'Enter NUM2: '
   MOV AH, 1           ; Set up for reading a character
   INT 21H             ; Read character
   SUB AL, '0'         ; Convert ASCII to integer
   MOV AH, 0           ; Clear the upper byte
   MOV NUM2, AX        ; Store the result in NUN2



   MOV AX, NUM1        ; Load NUM1 into AX
   MOV DX, 0           ; Clear DX for 16-bit
   SUB AX, NUM2        ; Load NUN2 into BX

   MOV RESULT, AX      ; Store  RESULT


   ; Exit program
   MOV AH, 4CH
   INT 21H

END START
```

**** Write an assembly language program to MUL two numbers.**

```
include 'emu8086.inc'
.MODEL SMALL
.STACK 100H
.DATA
   NUM1 DW ?            ; First number
   NUM2 DW ?            ; Second number
   RESULT DW ?          ; Result
.CODE
START:

   MOV AX, @DATA
   MOV DS, AX
                     ; Read first single-digit number
   print 'Enter NUM1: '
   MOV AH, 1           ; Set up for reading a character
   INT 21H             ; Read character
   SUB AL, '0'         ; Convert ASCII to integer
   MOV AH, 0           ; Clear the upper byte
   MOV NUM1, AX         ; Store the result in NUM1
                         ; new line
   MOV AH, 2          ; DOS function to display a character
   MOV DL, 0Dh        ; Carriage return (ASCII 13)
   INT 21H           ; Call DOS interrupt
   MOV DL, 0Ah        ; Line feed (ASCII 10)
   INT 21H
                                         ; Read second single-digit number
   print 'Enter NUM2: '
   MOV AH, 1           ; Set up for reading a character
   INT 21H             ; Read character
   SUB AL, '0'         ; Convert ASCII to integer
   MOV AH, 0           ; Clear the upper byte
   MOV NUM2, AX         ; Store the result in NUN2
   MOV BX, NUM2        ; Load NUM2 into BX
   MOV AX, NUM1        ; Load NUM1 into AX
   MUL BX          ; Load NUN2 into BX

   MOV RESULT, AX       ; Store  RESULT

   ; Exit program
   MOV AH, 4CH
   INT 21H

END START
```

1Write an assembly language program for performing 16-bit multiplication.**

```
.MODEL SMALL
.STACK 100H
.DATA

    NUM1   DW 1234H      ; First 16-bit number
    NUM2   DW 5678H      ; Second 16-bit number

    RESULT_HIGH DW 0     ; Upper 16 bits of the result
     RESULT_LOW DW 0      ; Lower 16 bits of the result

.CODE
MAIN PROC
    ; Initialize the Data Segment
    MOV AX, @DATA        ; Load the address of the data segment into AX
    MOV DS, AX          ; Initialize DS register

    ; Load the numbers into registers
    MOV AX, NUM1         ; Load NUM1 into AX
    MOV BX, NUM2         ; Load NUM2 into BX

    ; Perform multiplication
    MUL BX              ; Multiply AX by BX; result is in DX:AX

    ; Store the result
    MOV RESULT_LOW, AX    ; Store lower 16 bits of the result in RESULT_LOW
    MOV RESULT_HIGH, DX    ; Store upper 16 bits of the result in RESULT_HIGH

    ; Exit the program
    MOV AX, 4C00H       ; DOS function to terminate the program
    INT 21H             ; Call DOS interrupt

MAIN ENDP
END MAIN
```

2** Write an assembly language program to DIV two numbers.

```
include 'emu8086.inc'
.MODEL SMALL
.STACK 100H
.DATA
   NUM1 DW ?           ; First number
   NUM2 DW ?           ; Second number
   RESULT DW ?         ; Result of division
   REMENDER DW ?       ; Remainder after division
.CODE
START:
   MOV AX, @DATA
   MOV DS, AX

   ; Read first single-digit number
      print 'Enter NUM1: '
   MOV AH, 1           ; Set up for reading a character
   INT 21H             ; Read character
   SUB AL, '0'         ; Convert ASCII to integer
   MOV AH, 0           ; Clear the upper byte
   MOV NUM1, AX        ; Store the result in NUM

   MOV AH, 2           ; DOS function to display a character
   MOV DL, 0Dh         ; Carriage return (ASCII 13)
   INT 21H             ; Call DOS interrupt
   MOV DL, 0Ah         ; Line feed (ASCII 10)
   INT 21H             ; Call DOS interrupt

   ; Read second single-digit number
    print 'Enter NUM1: '
   MOV AH, 1           ; Set up for reading a character
   INT 21H             ; Read character
   SUB AL, '0'         ; Convert ASCII to integer
   MOV AH, 0           ; Clear the upper byte
   MOV NUM2, AX                ; Store the result in NUN

   ; Perform division: AX / NUN -> Quotient in AL, Remainder in AH
   MOV AX, NUM1                ; Load NUM into AX
   MOV DX, 0                   ; Clear DX for 16-bit division
   MOV BX, NUM2                ; Load NUN into BX (divisor)

   DIV BX                       ; Divide AX by BX (NUM / NUN)
   MOV RESULT, AX               ; Store quotient in RESULT
   MOV REMENDER, DX             ; Store remainder in REMENDER

                                   ; Exit program
   MOV AH, 4CH
   INT 21H

END START
```

3** Write an assembly language program to find square of a number.

```
include 'emu8086.inc'
.MODEL SMALL
.STACK 100H
.DATA

   NUM1 DW ?            ; First number
   RESULT DW ?          ; Result


.CODE

START:
   MOV AX, @DATA
   MOV DS, AX

  print 'Enter a number: '
   MOV AH, 1            ; Set up for reading a character
   INT 21H              ; Read character
   SUB AL, '0'          ; Convert ASCII to integer
   MOV AH, 0            ; Clear the upper byte
   MOV NUM1, AX         ; Store the result in NUM1


   MOV BX, NUM1         ; Load NUM2 into BX
   MUL BX

   MOV RESULT, AX       ; Store  RESULT

   ; Exit program
   MOV AH, 4CH
   INT 21H

END START
```

4. Write an assembly language program to find the largest number in an array of data.**

```
include 'emu8086.inc'
.model small
.stack 100h
.data
   array db 2,5,6,8,7,6,2,8,7,9
.code
main proc
   mov ax,@data
   mov ds,ax

   mov si,offset array
   mov cx,10  ;number of array element
   mov al,[si]

   loopx:
   cmp al,[si]
   jle update

   resume:
   inc si
   loop loopx

   print 'Largest number in array: '

   add al,48
   mov dl,al
   mov ah,02h
   int 21h
   jmp exit

   update:
   mov al,[si]
   jmp resume

   exit:
   main endp

end main
```

5. Write an assembly language program to find the smallest number in an array of data**

```
include 'emu8086.inc'
.model small
.stack 100h
.data
   array db 2,5,6,8,7,6,2,8,7,9
.code
main proc
   mov ax,@data
   mov ds,ax
   mov si,offset array
   mov cx,10  ;number of array element
   mov al,[si]

   loopx:
   cmp [si],al
   jle update

   resume:
   inc si
   loop loopx

   print 'Smallest number in array: '

   add al,48
   mov dl,al
   mov ah,02h
   int 21h
   jmp exit

   update:
   mov al,[si]
   jmp resume

   exit:
   main endp

end main
```

6. Write an assembly language program to convert a given binary to BCD.

**7. Write an assembly language program to sort the set numbers in ascending order.

```
include "emu8086.inc"
.model small
.stack 100h
.data
  numbers db 5, 2, 7, 1, 8, 3      ; Example set of numbers to sort
  numCount equ ($ - numbers)      ; Number of elements in the array
.code
main proc
  mov ax, @data
  mov ds, ax

  mov bx, 0                    ; Outer loop counter
  outer_loop:
    mov cx, numCount-1         ; Inner loop counter
    mov si, offset numbers     ; Point SI to the beginning of the array

  inner_loop:
    mov al, [si]             ; Load the current element
    mov dl, [si+1]             ; Load the next element

    cmp al, dl               ; Compare the current element with the next element
    jle skip_swap              ; If the current element is less than or equal to the next element, skip swapping

    mov [si], dl             ; Swap the elements
    mov [si+1], al

  skip_swap:
    inc si                   ; Move to the next element
    loop inner_loop            ; Continue inner loop until cx is zero

     inc bx                    ; Increment the outer loop counter
     cmp bx, numCount-1            ; Compare the outer loop counter with the number of elements - 1
      jl outer_loop                ; Jump back to the outer loop if bx is less than numCount - 1

   ; Display the sorted numbers
     mov si, offset numbers
     mov cx, numCount
  display_loop:
    mov dl, [si]
    add dl, 48               ; Convert to ASCII
    mov ah, 2                ; DOS display function
    int 21h

    print " "

    inc si
    loop display_loop
      mov ah, 4Ch                 ; Exit program
      int 21h
      main endp
```

```
end main
```

8. Write an assembly language program to move a block of data without overlap.

```
.MODEL SMALL
.STACK 100H
.DATA

   SOURCE DB 'Hello, World!', 0  ; Source data (null-terminated string)
   DESTINATION DB 20 DUP(0)      ; Destination buffer (20 bytes, initialized to 0)
   SOURCE_LENGTH EQU $ - SOURCE   ; Calculate the length of SOURCE (excludes the null terminator)

.CODE
MAIN PROC
   ; Initialize the Data Segment
   MOV AX, @DATA              ; Load the address of the data segment into AX
   MOV DS, AX                 ; Initialize DS register

   ; Load the address of the source and destination
   LEA SI, SOURCE             ; Load effective address of SOURCE into SI
   LEA DI, DESTINATION        ; Load effective address of DESTINATION into DI

   ; Move data from SOURCE to DESTINATION
   MOV CX, SOURCE_LENGTH      ; Load the length of SOURCE into CX
   CLD                        ; Clear the direction flag for forward copying

   REP MOVSB                  ; Repeat MOVSB CX times (move byte)

   ; Optionally: Terminate the program (can print DESTINATION)
   MOV AX, 4C00H              ; DOS function to terminate the program
   INT 21H                    ; Call DOS interrupt

MAIN ENDP
END MAIN
```

9. Write an assembly language program to transfer of a string in forward direction.

```
.MODEL SMALL
.STACK 100H
.DATA

   SOURCE DB 'Hello, World!', 0  ; Source string (null-terminated)
   DESTINATION DB 20 DUP(0)      ; Destination buffer (20 bytes, initialized to 0)
   SOURCE_LENGTH EQU ($ - SOURCE - 1) ; Calculate length of SOURCE (excluding null terminator)

.CODE
MAIN PROC
   ; Initialize the Data Segment
   MOV AX, @DATA              ; Load the address of the data segment into AX
   MOV DS, AX                 ; Initialize DS register
   ; Load the address of the source and destination
   LEA SI, SOURCE              ; Load effective address of SOURCE into SI
   LEA DI, DESTINATION         ; Load effective address of DESTINATION into DI

   ; Move data from SOURCE to DESTINATION
   MOV CX, SOURCE_LENGTH       ; Load the length of SOURCE into CX
   CLD                         ; Clear the direction flag for forward copying

   REP MOVSB                   ; Repeat MOVSB CX times (move byte)

   ; Optionally: Terminate the program (can print DESTINATION)
   MOV AX, 4C00H               ; DOS function to terminate the program
   INT 21H                     ; Call DOS interrupt

MAIN ENDP
END MAIN
```

10 .Write an assembly language program to multiply two 16-bit binary numbers to give a 32-bit result.

```
.MODEL SMALL
.STACK 100H
.DATA
      NUM DW 1111b,1011b
      PRODUCT DW ?


.CODE
START:
   MOV AX,DATA
   MOV DS,AX

   LEA SI,NUM       ; SI pointed to the Multiplicand
   MOV AX,[SI]      ; Multiplicand has to be in AX register
   MOV BX,[SI+2]    ; SI+2 pointed to the Multiplier and move it to BX
   MUL BX
               ;Perform the multiplication
   MOV PRODUCT,AX   ;32 bit product stored in DX-AX registers
   MOV PRODUCT+2,DX

   MOV AH,4CH
   INT 21H

   CODE ENDS
END START
```

11. Write an assembly language program to find out the average two(max and low) temperatures.

```
include 'emu8086.inc'
.model small
.stack 100h
.data
max_t db 10
min_t db 6
.code
main proc
    mov ax,@data
    mov ds,ax

    mov al,max_t
    add al,min_t
    mov ah,00h
    mov bl,02h
    div bl ;


    mov bl,al

    print 'Average Temperature= '
    add bl,48
    mov dl,bl
    mov ah,02h
    int 21h

    exit:
    mov ah,4ch
    int 21h
    main endp
end main
```

12. Write an assembly language program to read a character from keyboard.

```
include 'emu8086.inc'
.model small
.stack 100h
.code

main proc

    ;input
    print 'Enter a character: '
    mov ah,1 ;inpt cmd
    int 21h
    mov bl,al
    ;New line
    mov ah,2
    mov dl,10
    int 21h
    mov dl,13
    int 21h

    ;output
    print 'Output: '
    mov ah,2   ;output command
    mov dl,bl
    int 21h


    exit:
    mov ah,4ch
    int 21h
    main endp
end main
```

13. Write an assembly language program for moving a string from one location to another in memory.

```
;Declaration Part
.MODEL SMALL
.DATA
STR1 DB 09H, "BANGLADESH",'$'     ;STR1 is the given string to be
transferred
STR2 DB ?                         ;STR2 is the location for the
transfer
ST1 DB 09H,"STRING1:$"            ;To display STR1:
ST2 DB 09H,"STRING2:$"            ;To display STR2:
LEN DB 0FH                        ;Length of the String is loaded Here

.CODE
MAIN PROC
    MOV AX,@DATA
    MOV DS,AX
    MOV ES,AX
    LEA SI,STR1                   ; Location of STR1 is loaded to SI
    LEA DI,STR2                   ; Location of STR2 is loaded to DI


    ;To display STR1:
    LEA DX,ST1
    MOV AH,09H
    INT 21H
    ;To display contents of STR1
    LEA DX,STR1
    MOV AH,09H
    INT 21H

    ;NEW LINE
    MOV DL,10
    MOV AH,02H
    INT 21H
    MOV DL,13
    MOV AH,02H
    INT 21H


    ;To display STR2:
    LEA DX,ST2
    MOV AH,09H
    INT 21H

    ;Transferring Part
    CLD                          ; Clear the contents of Direction
Flag
    MOV CH,00H                   ; Since CX should be 00xx
    MOV CL,LEN
    REP MOVSB                    ; Repeat the transfer untill CL=0

    ;To display the transferred contents of STR1 to STR2
    LEA DX,STR2
    MOV AH,09H
    INT 21H

    ;Program Termination
    MOV AH,4CH
    INT 21H
    MAIN ENDP

END MAIN
```