install emu8086

Sample Input/Output Code:

```
.model small      ; Use small memory model (1 code + 1 data segment)
.stack 100h       ; Reserve 256 bytes for stack
.code             ; Start of code segment

main proc
   ; --- Input ---
   mov ah, 1      ; DOS function: read character, → prepare for “read char”
   int 21h        ; Call DOS → character goes into AL
   mov bl, al     ; Save input character in BL

   ; --- Output ---
   mov ah, 2      ; DOS function: print character, → prepare for “print char”
   mov dl, bl     ; Put saved character into DL (DOS expects it here)
   int 21h        ; Call DOS → prints the character, → prints DL

   ; --- Exit ---
exit:
   mov ah, 4Ch    ; DOS function: exit program
   int 21h        ; Return control to DOS
main endp
end main
```

Adding New Line in the Code:

```
mov ah, 2
mov dl, 10 ; Ascii value of new line is 10, moves the cursor down one line
int 21h
mov dl, 13 ; value of Carriage Return (CR) is 10, moves the cursor to the beginning of the line
int 21h
```

Variable declaration in assembly:

```
.model small
.stack 100h
.data
a db 5
b db ?
.code

main proc
    mov ax, @data
    mov ds, ax

    mov ah, 1
    int 21h        ; input b
    mov b, al

    mov ah, 2
    mov dl, 10 ;
    int 21h
    mov dl, 13 ;
    int 21h

    mov ah, 2
    mov dl, a
    add dl, 48     ; convert number to ASCII
    int 21h

    mov ah, 2
    mov dl, b
    int 21h

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

String printing in assembly Language:

```
.model small
.stack 100h
.data
a db 'Bangladesh is my country $'
.code

main proc
    mov ax, @data  ;loads that segment address into AX.
    mov ds, ax  ; AX is by default register, copies the address into DS (Data Segment register).

    mov ah, 9
    lea dx, a ; load effective address of variable 'a'
    int 21h

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

Input output using user:

Enter a number : 5
The number is : 5

```
; Program: Display and input a number, then show it back
; Model: Small (code + data < 64 KB)
; Assembler: MASM / TASM
; Author: Mujahidul Islam

.model small          ; Defines small memory model
.stack 100h           ; Reserves 256 bytes for the stack

.data
a db 'Enter a number: $'    ; Message 1 for input prompt
b db 'The number is : $'    ; Message 2 for output display

.code
```

```
main proc
    ; Initialize Data Segment
    mov ax, @data        ; Load address of data segment
    mov ds, ax           ; Move it into DS register

    ; Display "Enter a number: "
    mov ah, 9            ; Function 9: Display string
    lea dx, a            ; Load effective address of message 'a'
    int 21h              ; DOS interrupt call

    ; Print a new line (Carriage Return + Line Feed)
    mov ah, 2
    mov dl, 10           ; Line Feed (move to next line)
    int 21h
    mov dl, 13           ; Carriage Return (go to start of line)
    int 21h

    ; Input a single character (digit)
    mov ah, 1            ; Function 1: Read a character from keyboard
    int 21h              ; AL = input character (ASCII)
    mov bl, al           ; Store the input in BL register

    ; Print new line again
    mov ah, 2
    mov dl, 10
    int 21h
    mov dl, 13
    int 21h

    ; Display "The number is : "
    mov ah, 9            ; Function 9: Display string
    lea dx, b            ; Load address of message 'b'
    int 21h

    ; Display the entered number (stored in BL)
    mov ah, 2            ; Function 2: Display single character
    mov dl, bl           ; Move input character to DL for display
    int 21h              ; Show it on screen

exit:
```

```
    ; Exit to DOS
    mov ah, 4Ch          ; Function 4Ch: Exit program
    int 21h

main endp
end main
```

Input 3 initials, display

```
; Program: Read and display three initials (characters)
; Model: Small memory model
; Assembler: MASM / TASM
; Author: Mujahidul Islam

.model small          ; Use small memory model (code and data < 64KB)
.stack 100h           ; Reserve 256 bytes for stack

.data
a db 'Enter three initials: $'   ; Message prompt for user input

.code
main proc
    ; ---- Initialize data segment ----
    mov ax, @data      ; Load address of data segment into AX
    mov ds, ax         ; Move it into DS register (to access variables)

    ; ---- Display prompt ----
    mov ah, 9          ; DOS function 9: Display string
    lea dx, a          ; Load address of message 'a' into DX
    int 21h            ; Call DOS interrupt to display

    ; ---- Input first initial ----
    mov ah, 1          ; DOS function 1: Read a single character from keyboard
    int 21h            ; AL = character typed by user
    mov bl, al         ; Store it in BL (first character)

    ; ---- Input second initial ----
    mov ah, 1
```

```
int 21h
mov bh, al        ; Store it in BH (second character)

; ---- Input third initial ----
mov ah, 1
int 21h
mov cl, al        ; Store it in CL (third character)

; ---- Print newline (CR + LF) ----
mov ah, 2
mov dl, 10        ; ASCII 10 = Line Feed
int 21h
mov dl, 13        ; ASCII 13 = Carriage Return
int 21h

; ---- Display first initial ----
mov ah, 2         ; DOS function 2: Display one character
mov dl, bl        ; Load DL with first character
int 21h

; ---- Print newline ----
mov ah, 2
mov dl, 10
int 21h
mov dl, 13
int 21h

; ---- Display second initial ----
mov ah, 2
mov dl, bh        ; Load DL with second character
int 21h

; ---- Print newline ----
mov ah, 2
mov dl, 10
int 21h
mov dl, 13
int 21h

; ---- Display third initial ----
```

```
    mov ah, 2
    mov dl, cl        ; Load DL with third character
    int 21h

exit:
    ; ---- Exit program ----
    mov ah, 4Ch       ; DOS function 4Ch: Exit program
    int 21h

main endp
end main
```

General Purpose Registers

16-bit Register	High 8-bit	Low 8-bit	Common Use
AX	AH	AL	Accumulator (used for arithmetic, I/O, etc.)
BX	BH	BL	Base register (used for addressing or storage)
CX	CH	CL	Count register (used in loops, shifts, etc.)
DX	DH	DL	Data register (used in I/O and arithmetic)

So — each of these 4 registers can store two separate bytes (high and low).

✅ Therefore:

You already used	You can also use
BL, BH, CL	AL, AH, CH, DL, DH

That's 8 total 8-bit registers that can store 1 byte (character) each:

AH, AL, BH, BL, CH, CL, DH, DL

Swap/Exchange value

```
; Program: Input two characters, swap them, and display swapped output
; Model: Small memory model
```

```
; Assembler: MASM / TASM
; Author: Mujahidul Islam

.model small        ; Use small memory model
.stack 100h         ; Reserve 256 bytes for stack

.code
main proc

    ; ---- Input first character ----
    mov ah, 1       ; DOS Function 1 → read one character from keyboard
    int 21h         ; AL = ASCII code of typed character
    mov bl, al      ; Store the character in BL register

    ; ---- Input second character ----
    mov ah, 1       ; Again call function 1 → read another character
    int 21h
    mov bh, al      ; Store the second character in BH register

    ; ---- Swap the two characters ----
    xchg bl, bh     ; Exchange the contents of BL and BH

    ; ---- Print a new line (CR + LF) ----
    mov ah, 2
    mov dl, 10      ; ASCII 10 → Line Feed (move to next line)
    int 21h
    mov dl, 13      ; ASCII 13 → Carriage Return (go to line start)
    int 21h

    ; ---- Display first character (after swap) ----
    mov ah, 2       ; DOS Function 2 → display one character
    mov dl, bl      ; Move character from BL → DL
    int 21h

    ; ---- Display second character (after swap) ----
    mov ah, 2
    mov dl, bh      ; Move character from BH → DL
    int 21h

exit:
```

```
    ; ---- Exit program ----
    mov ah, 4Ch       ; DOS Function 4Ch → exit to DOS
    int 21h

main endp
end main
```

Add two numbers

```
; Program: Input two digits, add them, and display their sum
; Model: Small memory model
; Assembler: MASM / TASM
; Author: Mujahidul Islam

.model small          ; Use small memory model (code + data < 64KB)
.stack 100h           ; Reserve 256 bytes for stack

.code
main proc

    ; ---- Input first digit ----
    mov ah, 1         ; DOS function 1 → read a single character
    int 21h           ; AL = ASCII code of key pressed (e.g., '4' = 52)
    mov bl, al        ; Store the ASCII code in BL

    ; ---- Input second digit ----
    mov ah, 1
    int 21h
    mov bh, al        ; Store the ASCII code in BH

    ; ---- Add the two digits ----
    ; Note: Both are ASCII codes ('0' = 48)
    ; So we must first convert them to actual numeric values.

    sub bl, 48        ; Convert first ASCII digit to number (e.g., '4' → 4)
    sub bh, 48        ; Convert second ASCII digit to number (e.g., '3' → 3)
    add bl, bh        ; Add them together → BL = BL + BH (4 + 3 = 7)
```

```
    ; ---- Convert result back to ASCII for display ----
    add bl, 48          ; Convert number to ASCII (7 → '7')

    ; ---- Display the result ----
    mov ah, 2           ; DOS function 2 → display one character
    mov dl, bl          ; Move result character to DL
    int 21h             ; Display the sum

exit:
    ; ---- Exit program ----
    mov ah, 4Ch         ; DOS function 4Ch → terminate program
    int 21h

main endp
end main
```

Add three numbers

```
; Program: Input three digits, add them, and display sum (< 10)
; Model: Small memory model
; Assembler: MASM / TASM
; Author: Mujahidul Islam

.model small
.stack 100h
.code

main proc

    ; ---- Input first digit ----
    mov ah, 1           ; DOS function 1 → Read one character
    int 21h             ; AL = ASCII code of first digit
    mov bl, al          ; Store in BL

    ; ---- Input second digit ----
    mov ah, 1
    int 21h
    mov bh, al          ; Store in BH
```

```
    ; ---- Input third digit ----
    mov ah, 1
    int 21h
    mov cl, al          ; Store in CL

    ; ---- Convert ASCII to numeric values ----
    sub bl, 48          ; Convert '0'–'9' → 0–9
    sub bh, 48
    sub cl, 48

    ; ---- Add all three digits ----
    mov al, bl          ; AL = first digit
    add al, bh          ; AL = first + second
    add al, cl          ; AL = first + second + third

    ; ---- Convert result back to ASCII for display ----
    add al, 48          ; Convert 0–9 → '0'–'9'

    ; ---- Print result ----
    mov ah, 2           ; DOS function 2 → Display character in DL
    mov dl, al
    int 21h

exit:
    ; ---- Exit program ----
    mov ah, 4Ch         ; DOS function 4Ch → Exit
    int 21h

main endp
end main
```

Subtract two numbers

```
; Program: Input two digits and display their subtraction result (first - second)
; Model: Small memory model
; Assembler: MASM / TASM
; Author: Mujahidul Islam
```

```
.model small
.stack 100h
.code

main proc

    ; ---- Input first digit ----
    mov ah, 1           ; DOS Function 1: Read a character from keyboard
    int 21h             ; AL = ASCII code of input
    mov bl, al          ; Store in BL

    ; ---- Input second digit ----
    mov ah, 1
    int 21h
    mov bh, al          ; Store in BH

    ; ---- Convert ASCII to numeric values ----
    sub bl, 48          ; Convert '0'–'9' → 0–9
    sub bh, 48

    ; ---- Perform subtraction (first - second) ----
    mov al, bl          ; AL = first number
    sub al, bh          ; AL = AL - BH (first - second)

    ; ---- Convert result back to ASCII ----
    add al, 48          ; Convert 0–9 → '0'–'9'

    ; ---- Display the result ----
    mov ah, 2           ; DOS Function 2: Display character
    mov dl, al
    int 21h

exit:
    ; ---- Exit program ----
    mov ah, 4Ch         ; DOS Function 4Ch: Exit to DOS
    int 21h

main endp
end main
```

lowercase to Uppercase

```
; Program: Convert a lowercase character to uppercase (guaranteed lowercase input)
; Model: Small memory model
; Assembler: MASM / TASM

.model small
.stack 100h
.code

main proc

    ; ---- Input a lowercase character ----
    mov ah, 1           ; DOS function 1: Read character
    int 21h             ; AL = ASCII code of input
    mov bl, al

    sub bl, 32          ; Convert lowercase to uppercase ('a'-'z' → 'A'-'Z')

    ; ---- Display the uppercase character ----
    mov ah, 2           ; DOS function 2: Display character
    mov dl, al
    int 21h

exit:
    ; ---- Exit program ----
    mov ah, 4Ch
    int 21h

main endp
end main
```

Uppercase to lowercase

```
; Program: Convert an uppercase character to lowercase
; Model: Small memory model
; Assembler: MASM / TASM

.model small
.stack 100h
.code
```

```
main proc

    ; ---- Input an uppercase character ----
    mov ah, 1           ; DOS function 1: Read character
    int 21h             ; AL = ASCII code of input
    add al, 32          ; Convert uppercase to lowercase ('A'-'Z' → 'a'-'z')

    ; ---- Display the lowercase character ----
    mov ah, 2           ; DOS function 2: Display character
    mov dl, al
    int 21h

exit:
    ; ---- Exit program ----
    mov ah, 4Ch
    int 21h

main endp
end main
```

Multiply two number with Static initialization

```
; Program: Multiply two numbers (statically initialized)
; Model: Small memory model
; Assembler: MASM / TASM

.model small
.stack 100h

.data
num1 db 6          ; First number (static)
num2 db 7          ; Second number (static)
result db ?        ; To store the result

.code
main proc

    ; ---- Load numbers ----
```

```
    mov al, num1     ; AL = first number
    mov bl, num2     ; BL = second number

    ; ---- Multiply ----
    mul bl           ; AL * BL → result in AX (since 8-bit multiply)
                     ; AL = low byte, AH = high byte

    ; ---- Store result (only low byte if result < 256) ----
    mov result, al   ; Save the multiplication result

    ; ---- Display result ----
    ; Convert number to ASCII for display (0–9 only)
    add al, 48       ; Convert 0–9 → '0'–'9'
    mov ah, 2        ; DOS function 2: Display character
    mov dl, al
    int 21h

exit:
    mov ah, 4Ch      ; DOS function 4Ch: Exit program
    int 21h

main endp
end main
```

Multiply two number with Dynamic initialization

```
; Program: Multiply two numbers entered by user (dynamic initialization)
; Model: Small memory model
; Assembler: MASM / TASM

.model small
.stack 100h
.code

main proc

    ; ---- Input first digit ----
    mov ah, 1          ; DOS function 1: Read character
    int 21h            ; AL = ASCII code of first digit
    sub al, 48         ; Convert ASCII to numeric (0–9)
```

```
    mov bl, al          ; Store first number in BL

    ; ---- Input second digit ----
    mov ah, 1
    int 21h
    sub al, 48          ; Convert ASCII to numeric (0–9)
    mov bh, al          ; Store second number in BH

    ; ---- Multiply ----
    mov al, bl          ; AL = first number
    mul bh              ; AL * BH → result in AX
                        ; AX = 16-bit result, AL = low byte, AH = high byte

    ; ---- Display result ----
    ; For simplicity, assume result < 10
    add al, 48          ; Convert 0–9 → ASCII
    mov ah, 2           ; DOS function 2: Display character
    mov dl, al
    int 21h

exit:
    mov ah, 4Ch         ; DOS function 4Ch: Exit
    int 21h

main endp
end main
```