UNIVERSIDAD NACIONAL DE INGENIERIA(UNI)

AREA DE CONOCIMIENTO DE LA TECNOLOGIA Y LA COMUNICACIÓN
INGENIERIA EN COMPUTACION
ARQUITECTURA DE MAQUINAS COMPUTADORAS
MANEJO DE EMU8086

Elaborado por:

Br.Jesly Isayana Moraga Carnet: 2022-0485U	Br. Mayorga Salinas Juridia del Rosario. Carnet: 2022-0365U	Br.Jamie Sicely Rodriguez Sanchez Carnet: 2021-0772I

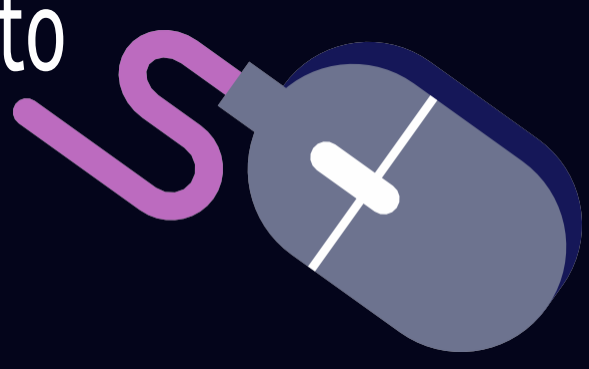
GRUPO: 4S1-COM-S
FECHA ENTREGA: 28/03/2025

U

Ensamblador de la familia ix86

El ensamblador x86 es un lenguaje de bajo nivel diseñado para los procesadores de la arquitectura Intel x86. Este lenguaje permite escribir instrucciones que interactúan directamente con el hardware del procesador. Se usa principalmente en sistemas embebidos, desarrollo de controladores y optimización de código en software de alto rendimiento.

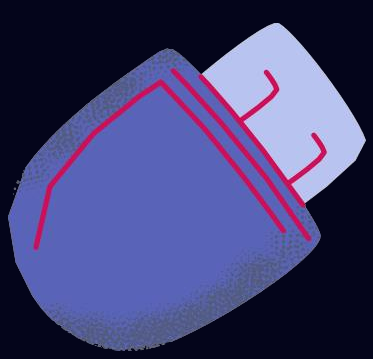
Declaracion de segmento



En ensamblador x86, la memoria se organiza en segmentos para facilitar la administración de datos y código. Los principales segmentos son:

- **.text: Código ejecutable.**
- **.data: Datos inicializados.**
- **.bss: Datos no inicializados.**
  stack: Manejo de llamadas y variables locales.

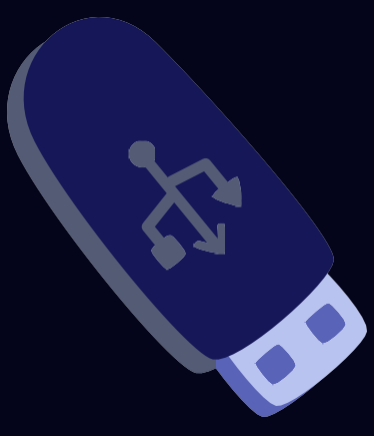
Modos de direccionamiento



Definen cómo acceder a los datos en memoria.
Algunos modos son:
Inmediato: MOV AX, 5
Directo: MOV AX, [1000H]
Indirecto: MOV AX, [BX]
Indexado: MOV AX, [SI + 10H]

Estructuras de control de flujo

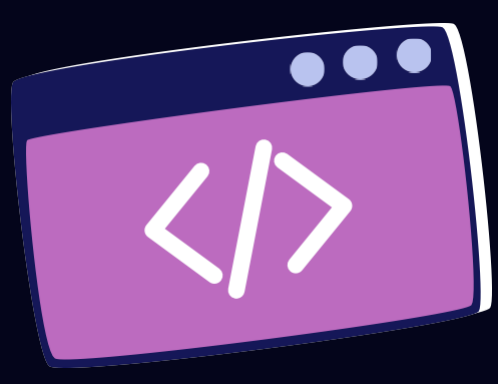
Permiten alterar la ejecución del programa:
Condicionales: CMP AX, BX seguido de JE, JNE, JG, JL, etc.
Bucles: LOOP etiqueta, JMP etiqueta.
Llamadas a procedimientos: CALL y RET.



Formato de una sentencia en ensamblador

Una instrucción en ensamblador sigue esta estructura:
[etiqueta:] instrucción [operandos] [;comentario]
Ejemplo:
INICIO: MOV AX, 100H ; Carga 100H en AX

Palabras reservadas



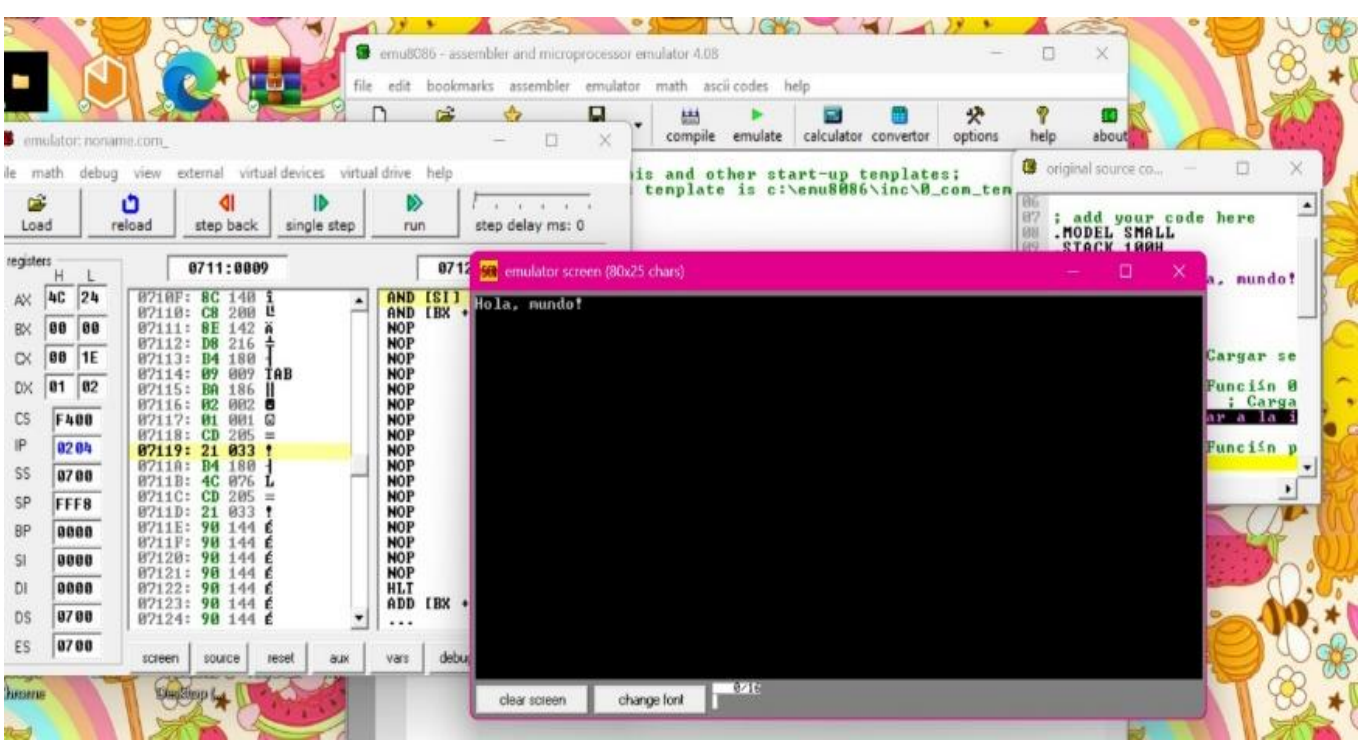
Son términos del lenguaje ensamblador con un significado especial, como:
Instrucciones: MOV, ADD, SUB, JMP, CALL
Registros: AX, BX, CX, DX, SI, DI, SP, BP
Directivas: DB, DW, EQU, SEGMENT, ENDS, ASSUME

Ejercicios Básicos ejecutar códigos de ensamblador para arquitecturas con el programa EMU8086 - MICROPROCESSOR EMULATOR.

- **Mostrar un mensaje en pantalla: Escribe un programa que muestre un mensaje en pantalla utilizando interrupciones para manejar la salida. (hola mundo).**

interrupción INT 21H del DOS con la función 09H para imprimir la cadena en pantalla.

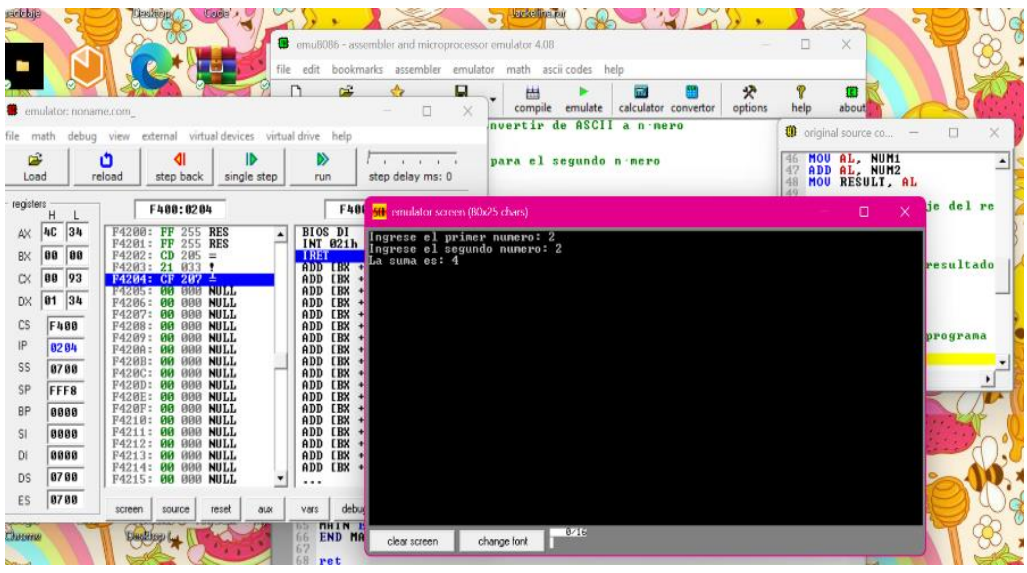
```
 .MODEL SMALL
.STACK 100H
.DATA
    MENSAJE DB 'Hola, mundo!$'  ; Mensaje a mostrar, el "$" indica fin de cadena para INT 21H

.CODE
MAIN PROC
    MOV AX, @DATA
    MOV DS, AX   ; Cargar segmento de datos

    MOV AH, 09H  ; Función 09H de INT 21H (mostrar cadena)
    LEA DX, MENSAJE  ; Cargar la dirección del mensaje
    INT 21H  ; Llamar a la interrupción para imprimir

    MOV AH, 4CH  ; Función para salir del programa
    INT 21H

MAIN ENDP
END MAIN
```



- **Suma de dos números: Escribe un programa que pida dos números al usuario, los sume y luego imprima el resultado en pantalla.**

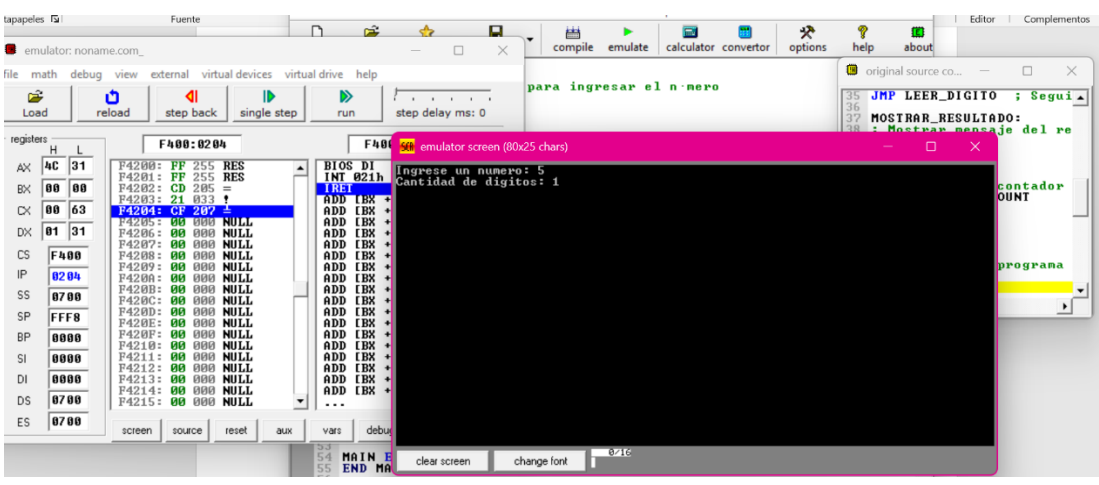
```
.MODEL SMALL
.STACK 100H
.DATA
    MSG1 DB 'Ingrese el primer numero: $'
    MSG2 DB 0DH, 0AH, 'Ingrese el segundo numero: $'
    MSG3 DB 0DH, 0AH, 'La suma es: $'
    NUM1 DB ?
    NUM2 DB ?
    RESULT DB ?

.CODE
MAIN PROC
    MOV AX, @DATA
    MOV DS, AX

    ; Mostrar mensaje para el primer número
    MOV AH, 09H
    LEA DX, MSG1
    INT 21H

    ; Leer el primer número
    MOV AH, 01H
    INT 21H
    SUB AL, '0'   ; Convertir de ASCII a número
    MOV NUM1, AL

    ; Mostrar mensaje para el segundo número
    MOV AH, 09H
    LEA DX, MSG2
    INT 21H

    ; Leer el segundo número
    MOV AH, 01H
    INT 21H
    SUB AL, '0'   ; Convertir de ASCII a número
    MOV NUM2, AL

    ; Sumar los dos números
    MOV AL, NUM1
    ADD AL, NUM2
    MOV RESULT, AL

    ; Mostrar mensaje del resultado
    MOV AH, 09H
    LEA DX, MSG3
    INT 21H

    ; Convertir el resultado a ASCII y mostrarlo
    ADD RESULT, '0'
    MOV DL, RESULT
    MOV AH, 02H
    INT 21H
```

```
; Finalizar el programa
MOV AH, 4CH
INT 21H
MAIN ENDP
  END MAIN
```



- **Contar el número de dígitos en un número: Escribe un programa que cuente cuántos dígitos tiene un número ingresado por el usuario.**

```
.MODEL SMALL
.STACK 100H
.DATA
    MSG1 DB 'Ingrese un numero: $'
    MSG2 DB 0DH, 0AH, 'Cantidad de digitos: $'
    DIGIT_COUNT DB 0   ; Contador de dígitos

.CODE
MAIN PROC
    MOV AX, @DATA
    MOV DS, AX

    ; Mostrar mensaje para ingresar el número
    MOV AH, 09H
    LEA DX, MSG1
    INT 21H

    ; Leer los dígitos del número
    MOV DIGIT_COUNT, 0  ; Inicializar contador

LEER_DIGITO:
    MOV AH, 01H  ; Leer un carácter del teclado
    INT 21H
    CMP AL, 0DH  ; Comparar con ENTER (fin de entrada)
    JE MOSTRAR_RESULTADO
    INC DIGIT_COUNT  ; Incrementar contador
    JMP LEER_DIGITO  ; Seguir leyendo

MOSTRAR_RESULTADO:
    ; Mostrar mensaje del resultado
    MOV AH, 09H
    LEA DX, MSG2
    INT 21H

    ; Convertir el contador a ASCII y mostrarlo
    MOV AL, DIGIT_COUNT
    ADD AL, '0'
    MOV DL, AL
    MOV AH, 02H
    INT 21H

    ; Finalizar el programa
    MOV AH, 4CH
    INT 21H

MAIN ENDP
END MAIN
```



- **Invertir un número: Escribe un programa que invierta un número de 16 bits ingresado por el usuario.**

```
ORG 100h

MOV AH, 09h
LEA DX, MSG_INPUT
INT 21h

CALL READ_NUM

MOV AH, 09h
LEA DX, MSG_OUTPUT
INT 21h

CALL REVERSE_NUM

CALL PRINT_NUM

MOV AH, 4Ch
INT 21h

READ_NUM PROC
    MOV BX, 0

READ_LOOP:
    MOV AH, 01h
    INT 21h

    CMP AL, 13
    JE FIN_READ

    SUB AL, 30h
    SUB AL, 30h
    MOV AH, 0
    MOV DX, BX
    MOV CX, 10
    MUL CX
    ADD AX, DX
    MOV BX, AX
    JMP READ_LOOP

FIN_READ:
    MOV AX, BX
    RET
READ_NUM ENDP

REVERSE_NUM PROC
    MOV CX, 0
    MOV DX, 0
    MOV BX, 10

REVERSE_LOOP:
    CMP AX, 0
    JE FIN_REVERSE

    MOV DX, 0
    DIV BX

    PUSH DX
    INC CX
    JMP REVERSE_LOOP

FIN_REVERSE:
    RET
REVERSE_NUM ENDP

PRINT_NUM PROC
    MOV AH, 02h
PRINT_LOOP:
    CMP CX, 0
    JE FIN_PRINT

    POP DX
    ADD DL, 30h
    INT 21h
    LOOP PRINT_LOOP

FIN_PRINT:
    RET
PRINT_NUM ENDP

MSG_INPUT DB 'Ingrese el numero a invertir: $'
MSG_OUTPUT DB 13,10, 'Numero invertido: $'

END
```

❖ **Suma de un arreglo de números: Crea un arreglo con 10 números y suma todos sus valores.**

```
ORG 100h

MOV CX, 10
LEA SI, ARREGLO
MOV AX, 0

MOV AH, 09h
LEA DX, MENSAJE_ARREGLO
INT 21h


MOSTRAR_ARREGLO:
    MOV DL, [SI]
    ADD DL, 30h
    MOV AH, 02h
    INT 21h

    CMP CX, 1
    JE SALTAR_MAS
    MOV DL, '+'
    INT 21h

SALTAR_MAS:
    INC SI
    LOOP MOSTRAR_ARREGLO

MOV AH, 09h
LEA DX, MENSAJE_IGUAL
INT 21h

MOV CX, 10
LEA SI, ARREGLO

SUMA_LOOP:
    ADD AL, [SI]
    INC SI
    LOOP SUMA_LOOP

MOV AH, 0
MOV DX, AX
MOV CX, 0

DIVISION_DECIMAL:
    MOV BX, 10
    DIV BL
    PUSH DX
    INC CX
    CMP AX, 0
    JNE DIVISION_DECIMAL


IMPRIMIR_DIGITOS:
    POP DX
    ADD DL, 30h
    MOV AH, 02h
    INT 21h
    LOOP IMPRIMIR_DIGITOS

MOV AH, 4Ch
INT 21h

MENSAJE_ARREGLO DB 'Numeros: ', '$'
MENSAJE_IGUAL   DB ' = ', '$'
ARREGLO DB 1,2,3,4,5,6,7,8,9,10

END
```

Escribe un programa en ensamblador que multiplique dos números y muestre el resultado.

```
  .MODEL SMALL
.STACK 100H

.DATA
   MSG1 DB 'Ingrese el primer numero (0-9): $'  ; Solicitar primer número
   MSG2 DB 0DH, 0AH, 'Ingrese el segundo numero (0-9): $'  ; Solicitar segundo número
   MSG3 DB 0DH, 0AH, 'El resultado de la multiplicacion es: $'  ; Resultado

.CODE
MAIN PROC
   MOV AX, @DATA
   MOV DS, AX

   ; Mostrar mensaje para ingresar el primer número
   MOV AH, 09H
   LEA DX, MSG1
   INT 21H

   ; Leer el primer número
   MOV AH, 01H  ; Función para leer un carácter
   INT 21H
   SUB AL, '0'  ; Convertir de ASCII a número
   MOV BL, AL   ; Guardar el primer número en BL

   ; Mostrar mensaje para ingresar el segundo número
   MOV AH, 09H
   LEA DX, MSG2
   INT 21H

   ; Leer el segundo número
   MOV AH, 01H  ; Función para leer un carácter
   INT 21H
   SUB AL, '0'  ; Convertir de ASCII a número
   MOV BH, AL   ; Guardar el segundo número en BH

   ; Multiplicar los números (BL * BH)
   MOV AL, BL   ; Cargar el primer número en AL
   MUL BH       ; Multiplicar AL (primer número) por BH (segundo número)
   ; El resultado de la multiplicación estará en AX

   ; Mostrar el mensaje de resultado
   MOV AH, 09H
   LEA DX, MSG3
   INT 21H

   ; Convertir el primer dígito del resultado a ASCII y mostrarlo
   MOV DL, AL    ; El resultado de la multiplicación está en AL
   ADD DL, '0'   ; Convertir el número a carácter ASCII
   MOV AH, 02H   ; Función para imprimir un solo carácter
   INT 21H

   ; Verificar si el resultado tiene dos dígitos
   MOV DL, AH    ; El dígito más significativo está en AH
   CMP DL, 0     ; Comprobar si hay un dígito más significativo
   JE FINAL      ; Si no hay dígito más significativo, termina

   ; Si el resultado tiene dos dígitos, imprimir el segundo dígito
   ADD DL, '0'   ; Convertir el segundo dígito a carácter ASCII
   MOV AH, 02H   ; Función para imprimir un solo carácter
   INT 21H

FINAL:
   ; Finalizar el programa
   MOV AH, 4CH
   INT 21H

MAIN ENDP
END MAIN
```