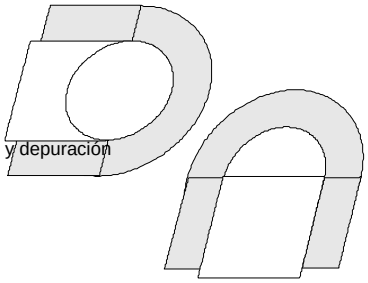
Tema 1: Fundamentos del ensamblador

- Estructura del 8086
- Tipos de instrucciones
- Instrucciones de transferencia
- Instrucciones aritméticas
- Estructura de un programa en ensamblador
- Pasos para la creación de un programa ejecutable
- Mi primer programa: edición, ensamblado, enlazado y depuración
- Directivas o pseudoinstrucciones
- El registro de estado
- Comandos del CodeView



Bibliografía básica

- Lenguaje ensamblador de los 80x86
  Jon Beltrán Heredia.
  Ed. Anaya Multimedia.

- 8088-8086/8087 programación ensamblador en entorno MS-DOS
  Miguel Angel Roselló.
  Ed. Anaya Multimedia.

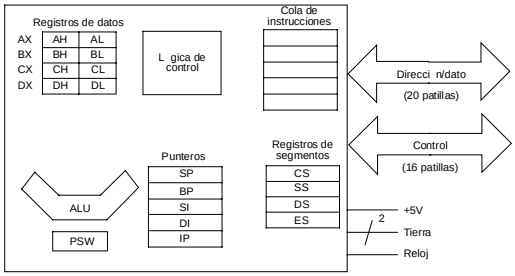
- Arquitectura, programación y diseño de sistemas basados en microprocesadores (8086/80186/80286)
  Yu-Cheng Lu, Glen A. Gibson.
  Ed. Anaya Multimedia.

- Lenguajes ensambladores
  R. Martínez Tomás.
  Ed. Paraninfo.

Estructura del i8086

El microprocesador 8086 tiene catorce registros de 16 bits. Dichos registros son:

- Registros de datos
- Registros de segmento
- Registros punteros de la pila
- Registros índices
- Registro puntero de instrucciones
- Registro de flags de estado



Banco de registros

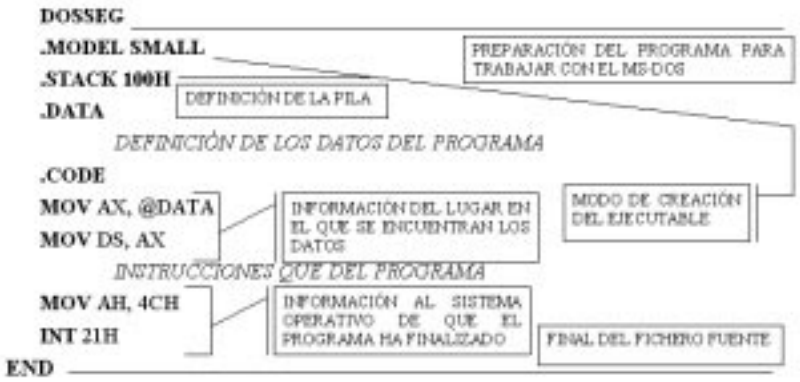
Registros de datos:

- AX (AH, AL)
- BX (BH, BL)
- CX (CH, CL)
- DX (DH, DL)

Punteros:

- SP - Puntero de pila
- BP - Puntero base de pila
- SI - Registro índice
- DI - Registro índice
- IP - Contador de programa

Estructura de un programa en ensamblador



Pasos para crear un fichero ejecutable

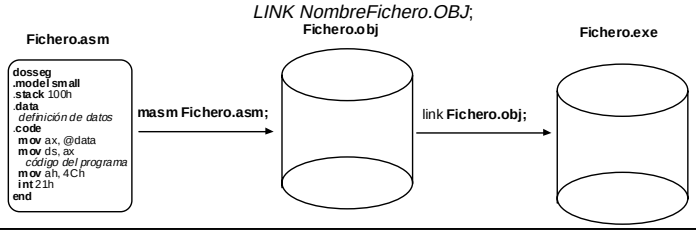
Programación:

- Los programas deben escribirse en un editor de texto ASCII
- El nombre del fichero debe tener como extensión ASM
- Para ensamblar el fichero se debe teclear en la línea de órdenes:

MASM *NombreFichero.ASM*;

- Si no se producen errores, se debe enlazar tecleando en la línea de órdenes:

LINK NombreFichero.OBJ;



Tipos de instrucciones

Tipo de instrucción	Efecto
Instrucciones de transferencia	Mueven información entre registros, registros y posiciones de memoria, o entre registros y puertos de entrada/salida
Instrucciones aritméticas	Realizan operaciones aritméticas: sumas, restas, etc.
Instrucciones de manejo de bits	Realizan operaciones de desplazamiento, rotación y lógicas sobre registros o posiciones de memoria
Instrucciones de transferencia de control	Sirven para controlar la ejecución de las instrucciones del programa
Instrucciones de entrada salida	Mueven información entre registros y puertos de entrada/salida
Instrucciones de manejo de cadenas	Realizan operaciones sobre cadena de bytes o palabras
Instrucciones de interrupción	Provocan que el microprocesador realice un servicio que se le solicita diferente a las instrucciones que está ejecutando

Instrucciones de transferencia de datos

- **Nombre:** **MOV**
- **Formato:** **MOV destino, origen**
- **Descripción:**

  Transfiere un byte o una palabra desde el operando origen al operando destino
- **Ejemplos:**

```
MOV CX, 112h          ; CX = 112h
MOV ES, AX            ; ES = AX
MOV AL, 12h           ; AL = 12h
MOV PAL_MEM, BX       ; PAL_MEM = BX
```

Instrucciones aritméticas (I)

- **Nombre:** **ADD**
- **Formato:** **ADD destino, origen**
- **Descripción:**

  Suma los dos operandos y el resultado lo deja en el operando destino. Los operandos deben ser del mismo tipo
- **Ejemplos:**

```
ADD CL, BL      ; CL = CL + BL
ADD AL, 12h     ; AL = AL + 12h
ADD CX, DX      ; CX = CX + DX
```

- **Nombre:** **ADC**
- **Formato:** **ADC destino, origen**
- **Descripción:**

  Suma los dos operandos más el posible acarreo de la operación anterior. El resultado se almacena en el operando destino. Además los operandos deben ser del mismo tipo
- **Ejemplos:**

```
ADC CL, BL      ; CL = CL + BL + CF
ADC AL, 12h     ; AL = AL + 12h + CF
ADC CX, DX      ; CX = CX + DX + CF
```


Instrucciones aritméticas (II)

- **Nombre:** **SUB**
- **Formato:** **SUB destino, origen**
- **Descripción:**

  Resta el operando origen del operando destino. El resultado se almacena en el operando destino y además, ambos operandos deben ser del mismo tipo
- **Ejemplos:**

```
SUB CL, BL      ; CL = CL - BL
SUB AL, 12h     ; AL = AL - 12h
SUB CX, DX      ; CX = CX - DX
```

- **Nombre:** **SBB**
- **Formato:** **SBB destino, origen**
- **Descripción:**

  Resta el operando origen del operando destino. Resta uno si el flag de acarreo está activo. Los operandos deben ser del mismo tipo. El resultado se almacena en el operando destino
- **Ejemplo:**

```
SBB CX, DX      ; CX = CX - DX - CF
```

Instrucciones aritméticas (III)

- **Nombre:** **MUL**
- **Formato:** **MUL origen**
- **Descripción:**

  Multiplica, sin considerar el signo el acumulador (AL o AX) por el operando origen. Si el operando origen es de tipo byte el resultado se almacena en AX. Si es de tipo palabra se almacena en DX (palabra superior) y AX (palabra inferior)
- **Ejemplo:**

```
                ; AX = 1234h
                ; BX = 1000h
MUL BX          ; DX = 0123h, AX = 4000h
```

- **Nombre:** **IMUL**
- **Formato:** **IMUL origen**
- **Descripción:**

  Multiplica, considerando el signo, el acumulador AL o AX por el operando origen. Si el operando fuente es un byte se almacena el resultado en AX. Si se trata de una palabra, se almacena en DX (palabra superior) AX (palabra inferior)
- **Ejemplos:**

```
                ; AL = FEh = -2
                ; BL = 12h = 18
IMUL BL         ; AX = FFDCh = -36
```


Instrucciones aritméticas (IV)

- **Nombre:** **DIV**
- **Formato:** **DIV origen**
- **Descripción:**

  Divide, sin considerar el signo, el acumulador AL o AX y su extensión (AH o DX) por el operando origen. El resultado se almacena en AL o AX, según el operando sea de un byte o de una palabra. El resto se almacena en la extensión del acumulador AH o DX
- **Ejemplos:**

```
                ; AX = 0013h = 19
                ; BL = 02h = 2
DIV BL          ; AH = 1, AL = 9
```

- **Nombre:** **IDIV**
- **Formato:** **IDIV origen**
- **Descripción:**

  Divide, considerando el signo, el acumulador AL o AX y su extensión (AH o DX) por el operando origen. El resultado se almacena en AL o AX, según el operando sea de un byte o de una palabra. El resto se almacena en la extensión del acumulador AH o DX
- **Ejemplos:**

```
                ; AX = FFEDh = -19
                ; BL = 02h = 2
IDIV BL         ; AH = 1, AL = F7h = -9
```

Instrucciones aritméticas (V)

- **Nombre:** **INC**
- **Formato:** **INC destino**
- **Descripción:**

  Suma una unidad al operando destino. El operando puede ser de tipo byte o palabra
- **Ejemplos:**

```
                ; AX = 1234h
INC AX          ; AX = 1235h
INC AH          ; AH = 13h
```

- **Nombre:** **DEC**
- **Formato:** **DEC destino**
- **Descripción:**

  Resta una unidad al operando destino. El operando puede ser de tipo byte o palabra
- **Ejemplos:**

```
                ; AX = 1234h
DEC AX          ; AX = 1233h
DEC AH          ; AH = 11h
```

- **Nombre:** **NEG**
- **Formato:** **NEG destino**
- **Descripción:**

  Cambia de signo mediante el complemento a 2 del operando destino. Deja el resultado en el operando destino. El operando puede ser de tipo byte o palabra
- **Ejemplos:**

```
NEG AL  ; Si AL = F2h antes de la operación, después AL = 0Eh
```


Mi primer programa en ensamblador (I)
Edición del fichero primero.asm

```
Tittle      Primero.asm

Comment # Programa que suma dos números enteros de 8 bits sin signo situados en memoria y almacena el resultado en memoria #

dosseg

.model small

.stack 100h

.data

    Sumando_1       db 12h
    Sumando_2       db 34h
    Suma            db ?

.code

    mov ax, @data       ; segmento de datos en AX
    mov ds, ax          ; lo pone en DS
    mov al, Sumando_1; Primer ioerando al
                        ; acumulador
    add al, Sumando_2; lo suma con el segundo
                        ; operando
    movl Suma, al       ; el resultado lo almacena
                        ; en memoria
    mov ah, 4Ch
    int 21h

end
```

Mi primer programa en ensamblador (II)
Ensamblado del fichero primero.asm

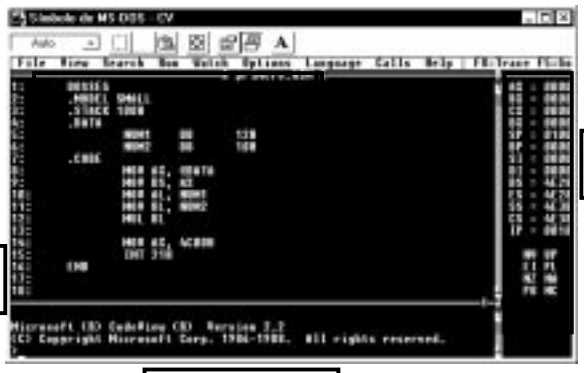
Mi primer programa en ensamblador (III)
Enlazado del fichero primero.obj

Mi primer programa en ensamblador (IV)
Ejecución paso a paso con el Code View (I)

Depuración:

- Si los programas no funcionan correctamente o se desea observar su funcionamiento se pueden ejecutar paso a paso
- El programa que permite realizar esta operación es el CODEVIEW
- Sintaxis CV *NombreFichero.EXE*
- Con F2 se activa o desactiva la ventana de registros
- Con F8 se ejecuta paso a paso y comprueba las subrutinas también
- Con F10 se ejecuta paso a paso y se salta la comprobación de las subrutinas

Mi primer programa en ensamblador (V)
Ejecución paso a paso con el Code View (II)



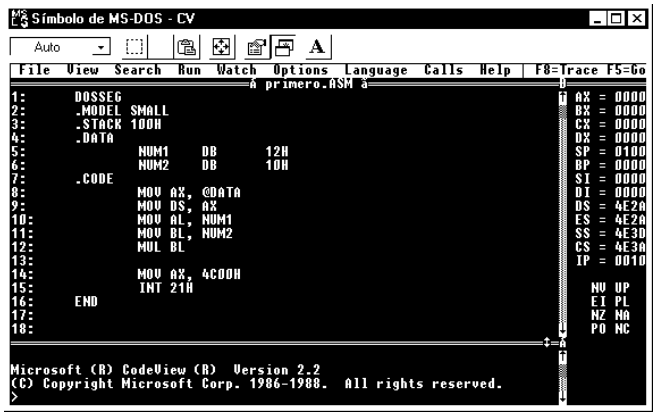
Registros del i8086

Zona de código

Zona de órdenes

Mi primer programa en ensamblador (VI)
Ejecución paso a paso con el Code View (III)

Comienzo del programa



Mi primer programa en ensamblador (VII)
Ejecución paso a paso con el Code View (IV)

Programa finalizado

Directivas o pseudoinstrucciones (I)

Directivas:

- No son como las sentencias “ejecutables” de los lenguajes de alto nivel, sino que sirven para reservar espacio de almacenamiento, asignar nombres a constantes, formar estructuras de datos, etc.
- No son órdenes destinadas al microprocesador, sino al programa ensamblador.
- Se pueden dividir en cuatro grupos funcionales:
  - Directivas de datos
  - Directivas condicionales
  - Directivas de listado
  - Directivas de macros

Directivas o pseudoinstrucciones (II)

Definición de datos

- **DB** (DEFINE BYTE)
  - **Formato:** [nombre_variable] DB expresión
  - **Descripción:** reserva memoria para una variable de tipo byte (8 bits) y los posteriores. “nombre_variable” es opcional y es el nombre asignado al primer byte.
  - **Operandos:** “expresión” es el valor inicial de la variable y puede ser:
    - Una constante positiva o negativa o expresión de ellas (-128 ≤ expresión ≤ 127 con signo).
    - Un signo “?” que indica indefinición de valor.
    - Una cadena de caracteres delimitada por comillas simples o dobles.
    - n1 DUP (n2) que indica repetición n1 veces de la expresión n2.
  - **Ejemplos:**

```
valores   DB 30, -15, 20
          DB 12*3
cadena    DB “Hola mundo”
```

Directivas o pseudoinstrucciones (III)

Definición de datos

- **DW** (DEFINE WORD)
  - **Formato:** [nombre_variable] DW expresión
  - **Descripción:** reserva memoria para una variable de tipo palabra (16 bits) inicializando o no esa palabra y las posteriores. “nombre_variable” es opcional y es el nombre asignado a la primera palabra.
  - **Operandos:** “expresión” es el valor inicial de la variable y puede ser:
    - Una constante positiva o negativa o expresión de ellas.
    - Un signo “?” que indica indefinición de valor.
    - El desplazamiento de una variable
    - n1 DUP (n2) que indica repetición n1 veces de la expresión n2.
  - **Ejemplos:** valores DW 300, -150, 2000
    DW 120*3

Directivas o pseudoinstrucciones (IV)

Definición de datos

- **PTR** (POINTER)
  - **Formato:** [tamaño dato] PTR
  - **Descripción:** es empleada para aquellas transferencias de datos en las que no se puede determinar el tamaño de los mismos.
  - **Operandos:** “tamaño dato” es el tamaño del dato al que se referencia:
    - BYTE El tamaño es de tipo byte
    - WORD El tamaño es de tipo palabra
  - **Ejemplos:** [INC B]
    INC WORD PTR [BX]
    INC BYTE PTR [BX]

Directivas o pseudoinstrucciones (V)

Control del ensamblador

- **DOSSEG** Prepara los segmentos para trabajar con DOS
- **.MODEL tipo** El tipo puede ser Tiny, Small, Medium, Compact, Large y Huge
  El argumento se denomina “modelo de memoria” y es empleado por el ensamblador para generar los nombres de los segmentos
- **.CODE** Abre el segmento de código
- **. DATA** Abre el segmento de datos
- **.STACK tamaño** Para fijar el tamaño de la pila
- **END etiqueta** Cierra el segmento de programa. En la etiqueta se especifica el punto en el que debe comenzar la ejecución del programa

De formato del listado

- **TITLE** Especifica el texto que aparecerá en cada página de listado
- **COMMENT delimitador texto delimitador** Permite incluir en el código fuente comentarios de varias líneas entre dos caracteres delimitadores, que no podrán usarse en ese texto

El registro de estado

- Al ejecutar las instrucciones el i8086 debe conocer el resultado de dicha ejecución. Para ello cuenta con el registro de flags o indicadores

Registro de estado en el i8086

Representación de los flags en el programa Code View

Flag de estado	Activado (1)	No activado (0)
Acarreo	CY	NC
Paridad	PE	PO
Acarreo Auxiliar	AC	NA
Cero	ZR	NZ
Signo	NG	PL
Interrupción	EI	DI
Dirección	DN	UP
Desbordamiento	OV	NV

Comandos del Code View (I)

Modificación de posiciones de memoria

- **E [tipo dato] [dirección] [valor hexadecimal]**

Algunos de los tipos de datos permitidos:

- **B** Byte (8-bit hexadecimal unsigned integer)
- **W** Word (16-bit unsigned hexadecimal)

Ejemplos:

EB Sumando_1 30

EB Suma 1A

Comandos del Code View (II)

Visualización de posiciones de memoria

- **Menú Watch, submenú Add Watch**
- **W [tipo dato] [dirección]**
- **W [tipo dato] [dirección] [L [longitud_tabla]]**

Algunos de los tipos de datos permitidos:

- **B** Byte (8-bit hexadecimal unsigned integer)
- **W** Word (16-bit unsigned hexadecimal)

Ejemplos:

WB Sumando_1

WB Tabla L 6