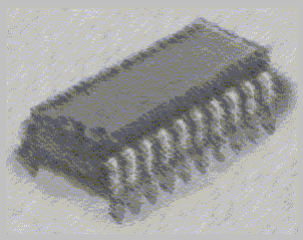
Tema 4. Fundamentos del ensamblador



Laboratorio de Estructura de Computadores

I. T. Informática de Gestión / Sistemas

Curso 2008-2009

Índice

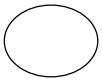
- Filosofía de programación
- Organigramas
- Estructura del 8086
- Estructura de un programa en ensamblador
- Pasos para la creación de un programa ejecutable
- Tipos de instrucciones
- Instrucciones de transferencia
- Instrucciones aritméticas
- Mi primer programa: edición, ensamblado, enlazado y depuración

Filosofía de la programación

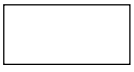
- Programar no debería ser un arte
- Programar de cabeza da muchos problemas
- Antes de hacer algo se debe saber qué se desea hacer
- Emplear ciertas técnicas para “ver” mejor el problema que se quiere programar
- La documentación es muy importante sobre todo después de que se ha hecho el programa
- Comentar los programas ayuda a mantenerlos

Organigramas (I)

- Una técnica muy sencilla consiste en hacer pequeños organigramas de lo que se quiere hacer con el programa
- El organigrama muestra gráficamente la estructura de nuestro programa
- Consta de los siguientes símbolos:

Comenzar / Terminar 

Decisión 

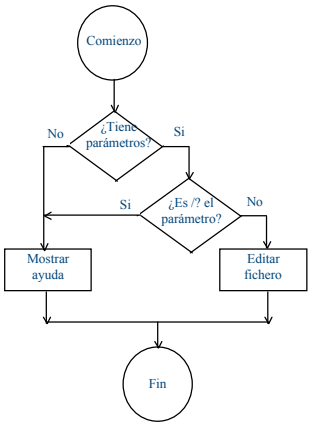
Secuencia de acciones 

Flujo de control

Organigramas (y II)

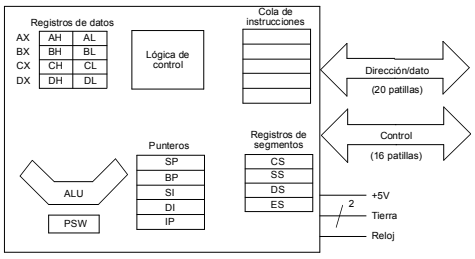
Ejemplo 2:

- Realizar un programa que si no se le pasa ningún parámetro o se le pasa el parámetro /? muestre un mensaje de ayuda indicando cómo funciona dicho programa. Y si no, se supone que se le pasa el nombre de un fichero entonces debe abrirlo y permitir que se introduzcan datos



Estructura del i8086 (I)

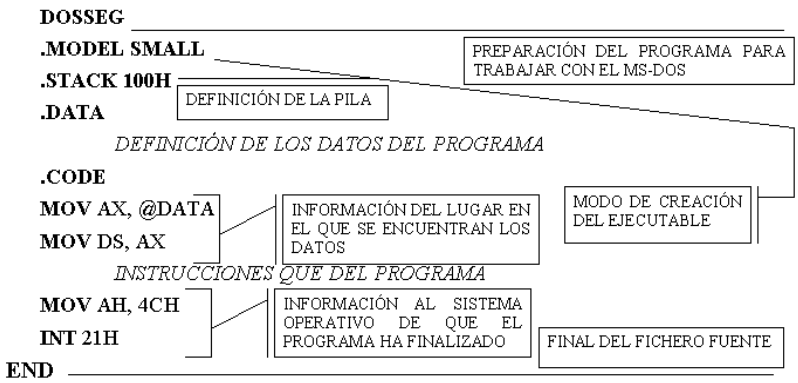
- El microprocesador 8086 tiene catorce registros de 16 bits. Dichos registros son:
  - Registros de datos
  - Registros de segmento
  - Registros punteros de la pila
  - Registros índices
  - Registro de instrucciones
  - Registro de flags de estado



Estructura del i8086 (y II)

- Registros de datos:
  - AX (AH, AL)
  - BX (BH, BL)
  - CX (CH, CL)
  - DX (DH, DL)
- Registros punteros:
  - SP - Puntero de pila
  - BP - Puntero base de pila
  - SI - Registro índice
  - DI - Registro índice
  - IP - Contador de programa

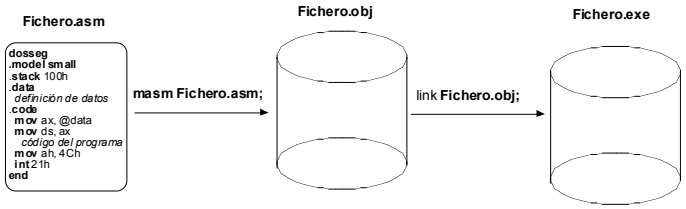
Estructura de un programa en ensamblador



Pasos para crear un programa 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 (I)

- **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 de transferencia de datos (II)

- **Nombre:** PUSH
- **Formato:** PUSH origen
- **Descripción:**
- Decrementa el puntero de pila (SP) en 2 y luego transfiere la palabra que se ha especificado en el operando origen a lo alto de la pila
- **Ejemplo:**
  - PUSH BX ; Pone BX en la pila

Instrucciones de transferencia de datos (y III)

- **Nombre:** POP
- **Formato:** POP destino
- **Descripción:**
- Transfiere un byte o una palabra desde la cima de la pila al operando destino y luego incrementa la pila en 2
- **Ejemplo:**
  - POP BX ; Pone en BX el contenido de la cima de la pila

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

Instrucciones aritméticas (II)

- **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 (III)

- **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

Instrucciones aritméticas (IV)

- **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 (V)

- **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

Instrucciones aritméticas (VI)

- **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 (VII)

- **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

Instrucciones aritméticas (VIII)

- **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 (IX)

- **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

Instrucciones aritméticas (X)

- **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

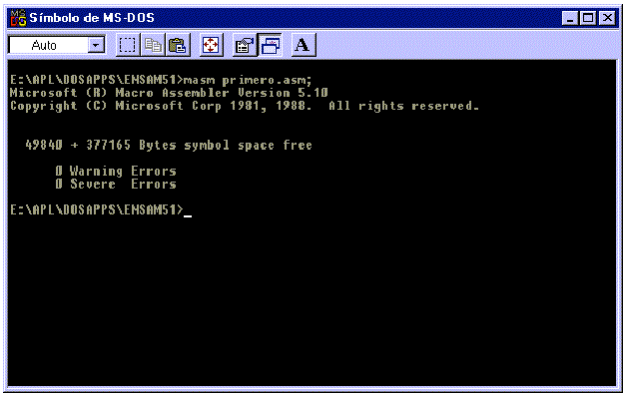
Instrucciones aritméticas (y XI)

- **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
- **Ejemplo:**
  - NEG AL ; Si AL = F2h antes de la operación, después AL = 0Eh

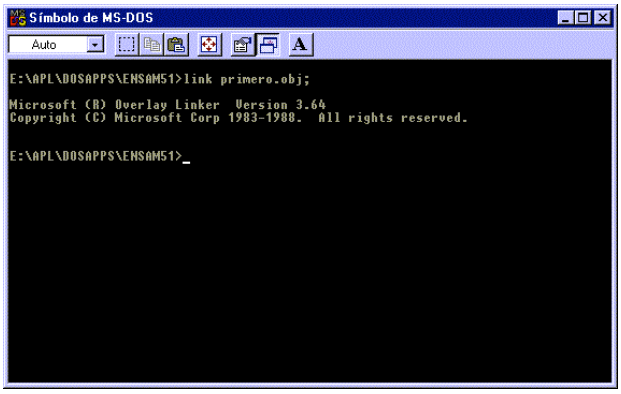
Mi primer programa en ensamblador (I)
Edición del programa primero.asm

```
dosseg
.model small
.stack 100h
.data
    num1        db 12h
    num2        db 10h
.code
    mov ax, @data
    mov ds, ax
    mov al, num1
    mov bl, num2
    mul bl
    mov ah, 4Ch
    int 21h
end
```


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



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

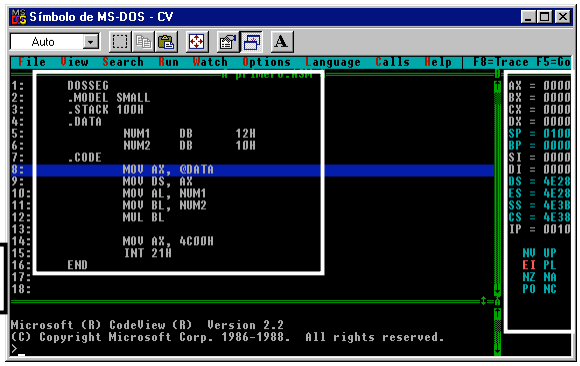


Mi primer programa en ensamblador (IV) Ejecución paso a paso con el Code View (I)

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 (y V)
Ejecución paso a paso con el Code View (y II)



Zona de código

Registros del i8086

Zona de órdenes

Bibliografía

- 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