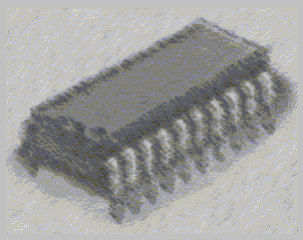
Tema 4. Lenguaje máquina y lenguaje ensamblador



Estructura de Computadores

I. T. Informática de Gestión / Sistemas

Curso 2008-2009

Índice

- Introducción
- Juego de instrucciones
- Estructura de un programa ensamblador del i80x86/88
- Modos de direccionamiento
- Ejemplo de hardware real: i80x86/88
  - Segmentación de memoria en i80x86/88
  - Modos de direccionamiento en el i80x86/88
- Formato de instrucciones:
  - Ejemplos de instrucciones
  - Formato de instrucciones en i80x86/88

Introducción (I)

Instrucción:

- Operación expresada mediante la codificación binaria de cadenas de 1's y 0's. Se le denomina lenguaje máquina
- El lenguaje máquina es distinto para cada computador. Excepto cuando existe compatibilidad entre familias

Repertorio de instrucciones o juego de instrucciones:

- Conjunto de órdenes que puede ejecutar un computador

Lenguaje ensamblador:

- Juego de instrucciones expresado con mnemónicos

Introducción (y II)

Programa:

- Conjunto ordenado de instrucciones que resuelve una tarea
- **Secuencia básica de ejecución de una instrucción:**
  - Lectura de memoria de la instrucción
  - Interpretación de la instrucción (por la unidad de control)
  - Ejecución de la instrucción (bajo las señales generadas por la unidad de control)
  - Actualización del contador de programa
- Las instrucciones se pueden clasificar según:
  - **El juego de instrucciones:** operaciones posibles y determinación de la siguiente instrucción a ejecutar
  - **El modo de direccionamiento:** ubicación de operandos
  - **Formato de las instrucciones:** codificación en binario

Juego de instrucciones (I)

El juego de instrucciones debe ser:

- Capaz de realizar una tarea computable en tiempo finito
- Eficaz (alta velocidad de cálculo)

Tipos de instrucciones:

- Instrucciones de transferencia
- Instrucciones de bifurcación
- Instrucciones aritméticas y lógicas
- Instrucciones de comparación y de bit
- Instrucciones de desplazamiento
- Instrucciones de entrada/salida
- Instrucciones de control

Juego de instrucciones (II)
Instrucciones de transferencia

- Copian en el operando destino la información del operando fuente sin modificar éste
- No modifican el estado de los flags
- Generalmente transfieren palabras pero pueden mover fracciones de ellas o bloques enteros

Las más frecuentes son (i8088/80x86):

- MOV transfiere el operando fuente al destino
- PUSH transfiere el operando fuente a la pila
- SP ← SP - 2
- [SP] ← Operando fuente
- POP transfiere el último dato de la pila al operando destino
- Operando destino ← [SP]
- SP ← SP + 2

Juego de instrucciones (III)
Instrucciones aritméticas y lógicas

Instrucciones aritméticas:

- **ADD**: suma sin acarreo
- **ADC**: suma con acarreo
- **SUB**: resta sin acarreo
- **SBB**: resta con acarreo
- **MUL**: multiplicación sin signo
- **IMUL**: multiplicación con signo
- **DIV**: división sin signo
- **IDIV**: división con signo
- **INC**: incrementar
- **DEC**: decrementar
- **NEG**: cambia de signo dejando el operando en C2

Instrucciones lógicas:

- AND, NOT, OR, XOR

Juego de instrucciones (IV)
Instrucciones de bifurcación (I)

- Modifican la secuencia normal de ejecución de un programa
- Actúan sobre el contador de programa (PC), controlan la secuencia de ejecución de un programa. Son un caso especial de transferencia, donde el operando destino es el PC

Clasificación:

- **Saltos:**
  - **Incondicionales**: JMP etiqueta (IP ← etiqueta)
  - **Condicionales**: J{condición} etiqueta
    Si condición, IP ← etiqueta. Si no, IP ← sig.Instrucción
- **Llamadas a subrutinas:** (saltos con retorno)
  - **Procedimientos**: CALL
  - **Interrupciones**: INT
    - Software
    - BIOS:
      - S.O.
      - Hardware
- **Saltos incondicionales**: siempre se produce el salto
- **Saltos condicionales**: se realiza el salto si se da la condición sobre los flags

Juego de instrucciones (V)
Instrucciones de bifurcación (II)

- Las condiciones más frecuentes admitidas por el 80x86/8088 son:

[N] { Z – Cero; E - Igual; C - Acarreo; S - Signo; O - Overflow (Desbordamiento) } [N] { G - Mayor que; L - Menor que; A - Superior*; B - Inferior* } [E]

{ P - Paridad; PE - Paridad par; PO - Paridad impar; CXZ - CX=0 }

- *Se refieren a operandos sin signo

Juego de instrucciones (VI)
Instrucciones de bifurcación (III)

- **Bucles**: operación (decremento de contador) + salto condicional sobre la operación
- LOOP Etiq realiza un bucle:
  CX ← CX -1;
  Si CX ≠ 0 entonces IP← Etiq, si no IP ← siguiente instrucción

Ejemplo:

```
MOV CX, 4
Bucle:
    INC BX
    ADD BX, CX
    LOOP Bucle
```

Juego de instrucciones (VII)
Instrucciones de bifurcación (IV) CALL

- **Llamadas a subrutinas**: salvan la posición de retorno
  Las instrucciones de salto a la subrutina y de regreso al programa principal van emparejadas

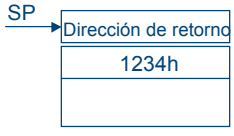
- **LLAMADA A UNA SUBRUTINA**
  CALL Etiq: salto con retorno a una subrutina
  SP← SP-2; [SP]←IP; IP← Etiq

1) Guardar en la pila la dirección de la siguiente instrucción a ejecutar
2) IP ←Dirección de la subrutina

Juego de instrucciones (VIII)
Instrucciones de bifurcación (V) RET

- **RETORNO DE UNA SUBRUTINA**
- **RET**: retorno a la secuencia principal
  IP ←[SP]; SP ← SP+2

1) Extraer de la pila la dirección de la siguiente instrucción a ejecutar
2) IP ← Dirección de retorno de la subrutina



Juego de instrucciones (IX)
Instrucciones de bifurcación (VI) Interrupciones

- Las interrupciones implican una ruptura en la secuencia del programa saltando al código que da ese servicio y cuando se ha terminado, se vuelve a la ejecución del programa en curso
- Pueden ser:
  - **Interrupciones hardware**: son generadas por los circuitos asociados al microprocesador en respuesta a algún evento como pulsar una tecla del teclado
  - **Interrupciones software**: son generadas por un programa para llamar a ciertas subrutinas almacenadas en memoria ROM o RAM. Es posible cambiarlas y crear otras nuevas.
- Los pasos para llamar a una interrupción son: identificar la interrupción necesaria, pasar los parámetros a la subrutina, llamar a la interrupción
- Las interrupciones salvaguardan los flags y los registros que emplean

Juego de instrucciones (X)
Instrucciones de bifurcación (VII) INT

- **LLAMADA A UNA INTERRUPCIÓN**
- **INT**: llamada a una rutina de interrupción (CALL + flags a la pila)

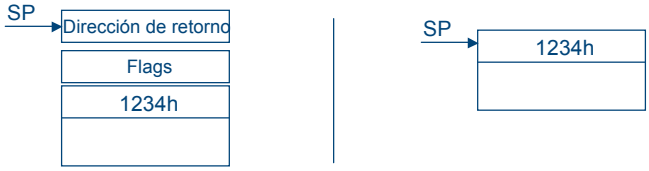
  1) Guardar en la pila los flags
  2) Guardar en la pila la dirección de la siguiente instrucción a ejecutar
  3) IP ← Dirección de la interrupción

Juego de instrucciones (XI)
Instrucciones de bifurcación (y VIII) IRET

- **REGRESO DE UNA INTERRUPCIÓN**

- **IRET**: retorno de la rutina de interrupción (RET + devuelve flags)

1) Guardar en la pila la dirección de la siguiente instrucción a ejecutar
2) IP ← Dirección de la subrutina
3) Flags ← Flags anteriores a la llamada a la interupción



Juego de instrucciones (XII)
Instrucciones de comparación y de bit

Instrucción de comparación:

- No almacenan el resultado, sólo modifican los flags
- **CMP**: compara números (resta pero no guarda el resultado solamente modifica los flags de estado)

Instrucciones de bit:

- Tienen la función de modificar un solo bit o leer su estado. Son típicas en la configuración de los flags
- **TEST**: comparación lógica a nivel de bits
- **CLI**: clear flag de interrupción
- **STI**: set flag de interrupción
- **CLC**: clear flag de acarreo
- **STC**: set flag de acarreo (realiza la operación lógica AND pero no guarda el resultado únicamente modifica los flags)

Juego de instrucciones (XIII)
Instrucciones de desplazamiento

- Todas las instrucciones de desplazamiento y rotaciones tienen en común:
  - El último valor desplazado se copia en el flag de acarreo
  - Si el número de desplazamientos es mayor que uno, se debe colocar el valor en el registro CL

Instrucciones de desplazamiento:

- **SAR**: desplazamiento aritmético a la derecha
- **SAL**: desplazamiento aritmético a la izquierda
- **SHR**: desplazamiento lógico a la derecha
- **SHL**: desplazamiento lógico a la izquierda
- **ROR**: rotación a la derecha
- **ROL**: rotación a la izquierda
- **RCR**: rotación a la derecha a través del acarreo
- **RCL**: rotación a la izquierda a través del acarreo

Juego de instrucciones (y XIV)
Instrucciones de E/S y de control

Instrucciones de entrada/salida

- **IN**: Transfiere información desde un puerto de entrada a un registro
- **OUT**: Escribe información en un puerto de salida desde un registro

Instrucciones de control

- **WAIT**: hace esperar al procesador
- **HLT**: detiene el procesador
- **NOP**: no operación

Estructura de un programa en ensamblador (I)

Estructura de un programa en ensamblador:

- **dosseg** ; prepara los segmentos para trabajar con DOS
- **.model small** ; define el modo del ejecutable
- **.stack** 100h ; define el tamaño de la pila
- **.data** ; zona de definición de los datos

  definición de datos
- **.code**
- **mov** ax, @data ; inicialización de los datos en
- **mov** ds, ax ; el segmento de datos

  código del programa
- **mov** AH, 4Ch ; terminación del programa y
- **int** 21h ; devolución del control a DOS
- **end** ; fin de programa

Estructura de un programa en ensamblador (II)
Suma dos números Num1 y Num2

```
dosseg          ; prepara los segmentos para trabajar con DOS
.model small    ; define el modo del ejecutable
.stack 100h     ; define la pila
.data           ; zona de definición de los datos
Num1 DB 20h
Num2 DB 33h
Res  DB ?
.code
mov AX, @data   ; inicialización de los datos en
mov DS, AX      ; el segmento de datos
mov AL, Num1
add AL, Num2
mov Res, AL
mov AH, 4Ch     ; terminación del programa y
int 21h  ; devolución del control al DOS
end      ; fin de programa
```

Estructura de un programa en ensamblador (III)
Escritura del texto Hola Mundo

```
dosseg              ; prepara los segmentos para trabajar con DOS
.model small        ; define el modo del ejecutable
.stack 100h         ; define la pila
.data               ; zona de definición de los datos
    Texto DB 'Hola mundo$'
.code
mov AX, @data       ; inicialización de los datos en
mov DS, AX          ; el segmento de datos
mov AH, 9
lea DX, Texto
int 21h
mov Ah, 4Ch         ; terminación del programa y
int 21h             ; devolución del control a DOS
end                 ; fin de programa
```


Modos de direccionamiento (I)

- El modo de direccionamiento determina la ubicación de un operando:
  - En la propia instrucción
  - En un registro
  - En memoria principal

Modos de direccionamiento		µP 8086/88	Ejemplos
Inmediato		Inmediato	MOV AX, 15H
Directo	De registro	A registro	MOV AX, BX
	De memoria	(No existe)	
	De página base	Directo	MOV CX, ETIQUETA
Relativo	Al contador de programa	Solamente para saltos	
	A un registro base	Relativo a base	MOV BX+ARTÍCULO, AL
	A un registro índice	Mediante índice	MOV DL, VECTOR[SI]
		Mediante índice y base	MOV AH, [BH][SI]+ARRAY
	A pila	A pila	PUSH BX
Indirecto		(No existe)	
Implícito		Algunas instrucciones	

Modos de direccionamiento (II)

- El operando se encuentra en la propia instrucción
- Ejemplo: MOV CX, 0010h (Su código máquina es B9 10 00 h)
- La manera de expresar el dato inmediato depende del formato de la máquina
  - Big endian
  - Little endian

Dirección de la palabra

0	3	2	1	0
4	7	6	5	4

Little endian

Dirección de la palabra

0	0	1	2	3
4	4	5	6	7

Big endian

Modos de direccionamiento (III)
Direccionamiento directo

- Es directo cuando la instrucción contiene el lugar donde se encuentra el operando
- Según el lugar donde se encuentra el operando puede ser:
  - Direccionamiento directo a registro
  - Direccionamiento directo a memoria
    - Dirección completa (p. ej. Z80 sobre 64 Kb con 16 bits)
    - Dirección sobre una página del mapa de memoria; también se conoce como direccionamiento de página base

Ejemplos:
- **MOV** AX, BX
- **MOV** CX, Etiqueta

Modos de direccionamiento (IV)
Direccionamiento relativo (I)

- La instrucción indica el desplazamiento del operando con respecto a un puntero
- La dirección efectiva es calculada por la unidad de control sumando, o restando, el desplazamiento al puntero de referencia que suele estar en un registro
- Dependiendo del puntero se tienen diferentes modos de direccionamiento

Ejemplos:

- **MOV** AL, [BX]
- **ADD** CH, Numero[SI]
- **MOV** BL, [SP+4]

Modos de direccionamiento (V)
Direccionamiento relativo (y II)

Dirección efectiva = Reg. Referencia + desplazamiento

Modo de direccionamiento	Registro de referencia	Cálculo dirección efectiva
Relativo a contador de programa	Contador de programa (CP)	DE = CP + desplazamiento
Relativo a registro base	Un registro base (R B)	DE = RB + desplazamiento
Relativo a registro índ ice	Un registro índice (R I)	DE = RI + desplazamiento
Relativo a pila	Registro de pila (SP)	DE = SP + desplazamiento

Ventajas	Inconvenientes
▪ Las instrucciones son más compactas ▪ El código puede cambiar de lugar en el mapa de memoria con cambiar el valor del puntero ▪ Gran facilidad de manejo de estructuras de datos	▪ Se requiere una operación adicional para el cálculo de la dirección del operando

Modos de direccionamiento (VI)
Direccionamiento indirecto

- La posición indicada por la instrucción no es el operando sino la dirección de memoria en la que se encuentra, por lo que se necesita un acceso adicional a memoria
- La dirección de memoria se puede dar mediante:
  - Direccionamiento directo a memoria
  - Direccionamiento relativo
- Es posible que se realicen múltiples niveles de indirección
- Su utilidad más común es la de acceso a diversas informaciones mediante tablas de punteros

Ejemplos:

- **MOV** AL, [[100]]
- **MOV** CL, [[B + 1234h]]

Modos de direccionamiento (VII)
Direccionamiento implícito

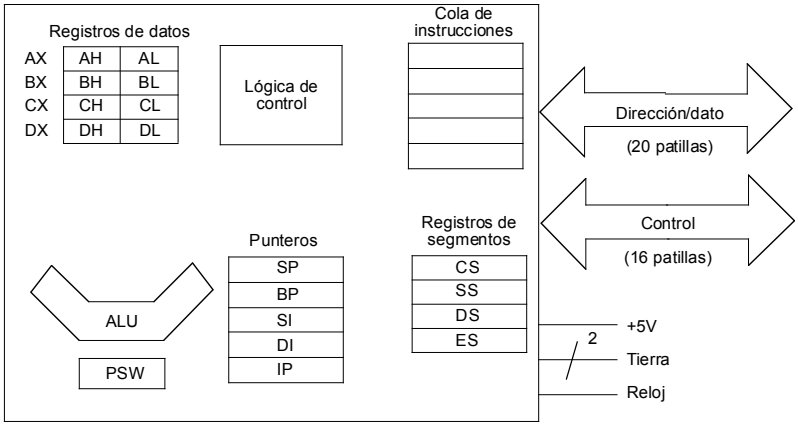
- En la instrucción no se indica explícitamente el lugar donde se encuentra el operando
- Requiere que el programador conozca con que operandos se está trabajando

Ejemplos:

- **MUL** BX → DX,AX = AX x BX
  - donde AX y DX son operandos implícitos

- **RET**
  - realiza las siguientes operaciones:
    - IP ← [SP]
    - SP ← SP +2

Ejemplo de hardware real: µP 80x86/88 (I)



Ejemplo de hardware real: µP 80x86/88 (II)
Segmentación de memoria en µP 80x86/88 (I)

- El 80x86/88 se puede direccionar 1MB con 20 líneas de dirección pero sus registros internos tan solo son de 16 bits
- Solución → segmentación de memoria
- Las direcciones se generan combinando una base y un desplazamiento, cada uno de 16 bits:

  Dirección física = base x 10h + desplazamiento
- Cada base genera una página o segmento de 64 Kb con funciones especificas:

Base Registro de segmento	**Función**
CS	Contiene el código ejecutable
SS	Se reserva para la pila (*stack*)
DS	Contiene los datos
ES	Segmento extra de datos

Ejemplo de hardware real: µP 80x86/88 (III)
Segmentación de memoria en µP 80x86/88 (II)

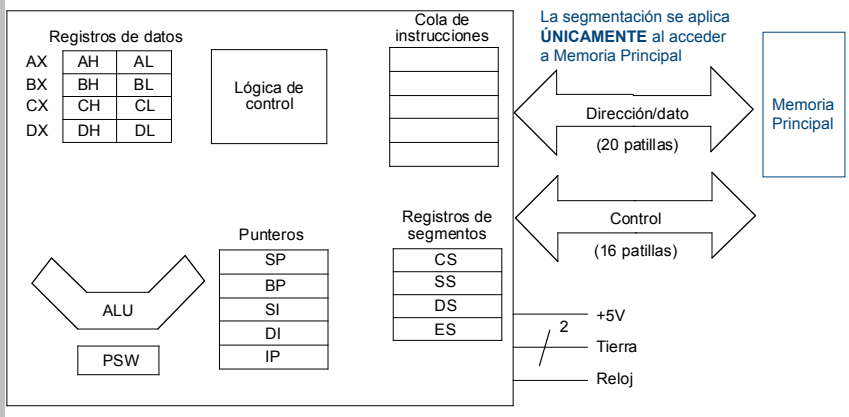
Banco de registros:

- **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

Registros de segmentos:

- SS - Segmento de pila
- DS - Segmento de datos
- ES - Segmento extra de datos
- CS - Segmento de código

Ejemplo de hardware real: µP 80x86/88 (IV)
Segmentación de memoria en µP 80x86/88 (III)



Ejemplo de hardware real: μP 80x86/88 (V)
Segmentación de memoria en μP 80x86/88 (IV)

Modo de direccionamiento	Ejemplo	Cálculo dirección efectiva
Directo	MOV CL, Etiqueta	DF = DS x 10h + Etiqueta
Relativo a base	MOV AH, [BX]+ Elemento	DF = DS x 10h + BX + Elemento
Mediante índice	MOV DL, Elemento[SI]	DF = DS x 10h + SI + Elemento
Mediante índice y base	ADD CH, Elemento[BX][SI]	DF = DS x 10h + BX + SI + Elemento
La segmentación solamente se emplea cuando uno de los operandos se encuentra en memoria En el caso de que se trate de la dirección de la siguiente instrucción a ejecutar se empleará CS y no DS Si lo que se busca es un operando en la pila el registro de segmento a emplear será el SS		

Ejemplo de hardware real: μP 80x86/88 (VI)
Segmentación de memoria en μP 80x86/88 (V)

B. Registros
CS = 0100h
DS = 0200h
SS = 0300h
BX = 1234h

B. Registros
SI = 0010h
IP = 0025h
SP = 0200h

Memoria
Num= 1500h

EJEMPLOS

- **Dirección física de memoria de la instrucción a ejecutar:**
  DF = CS x 10h + IP = 0100h x 10h + 0025h = 01025h
- **Dirección física de memoria del operando fuente MOV AL, Num[SI]**
  DF = DS x 10h + Num + SI = 0200h x 10h + 1500 h + 0010h = 03510h
- **Dirección física en la que se encuentra la cabecera de la pila**
  DF = SS x 10h + SP = 0300h x 10h + 0200h = 03200h

Formato de las instrucciones (I)

- Es la representación en binario de cada una de las instrucciones
- Cada instrucción “contiene” explicitamente o implicitamente toda la información que necesita para ejecutarse:
  - Código de operación, indica a la UC el tipo de operación, aritmética, lógica, de transferencia, salto, etc.
  - El valor o la posición donde se hallan los operandos
  - El lugar donde se tiene que depositar el resultado
  - Dirección de la siguiente instrucción a ejecutar

Cod. operación	Operandos	Resultado	Dir. sig. instrucc.

Formato de las instrucciones (II)

Características generales:

- Las instrucciones se "encajan" en alguno de los formatos disponibles
- Los formatos son sistemáticos (campos de longitud y posición fijas)
- El primero de los campos es el código de operación
- Siempre que se pueda, se supone información implícita para acortar:
- Siguiente instrucción en la siguiente posición de memoria, salvo bifurcaciones
- En vez de usar la operación pura se asignan diferentes códigos de operación para diferentes modos de direccionamiento
- La ubicación del resultado coincide con el operando destino

Cod. operación	Operandos

Formato de las instrucciones (III)
Ejemplos de instrucciones

- **Z80**

Tipo de instrucción	L. Máquina	L. Ensamblador	Operación
Transferencia	323353	LD (5333), A	M(5333) ▪ A
	Transfiere el contenido del registro A a la posición de memoria 5333 h		
Multiplicación	-	-	-
	No existe equivalente		

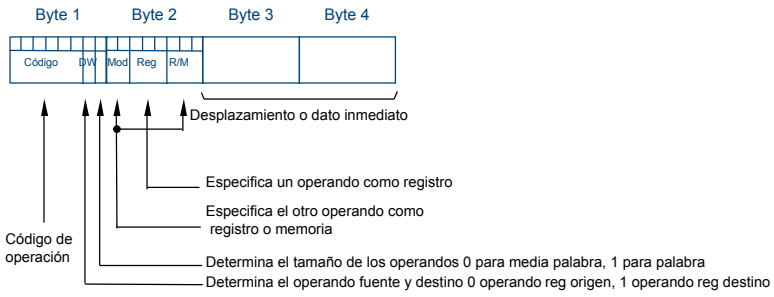
- **i80x86/8088**

Tipo de instrucción	L. Máquina	L. Ensamblador	Operación
Transferencia	A33353	MOV [5333], AX	M(5333) ← AX
	Transfiere el contenido del registro AX (acumulador) a la posición de memoria 5333 h		
Multiplicación	F7E3	MUL BX	DX,AX ← AX x BX
	Multiplica el contenido de los registros AX y BX, y deja el resultado en AX y DX (32 bits)		

Formato de las instrucciones (IV)
Formato de instrucciones μP i80x86/8088 (I)

- Cuenta con múltiples formatos cuyo código máquina va desde un byte hasta seis bytes

Formato de las instrucciones registro-registro y registro-memoria:



Formato de las instrucciones (V)
Formato de instrucciones μP i80x86/8088 (II)

- **El primer byte contiene:**
  - Código de operación
  - El bit de dirección de registro (D):
    - Si D = 1 tengo que REG = operando destino
    - Si D = 0 tengo que REG = operando fuente
  - El bit de tamaño del dato (W): especifica si la operación será realizada sobre datos de media palabra o de una palabra:
    - Si W = 0 los datos son de 8 bits (ó 16 bits)
    - Si W = 1 los datos son de 16 bits (ó 32 bits)

Formato de las instrucciones (VI)
Formato de instrucciones μP i80x86/8088 (III)

- **El segundo byte contiene los operandos** (uno de ellos es un registro):
- **REG** se usa para identificar un registro

REG	W=0	W=1
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

- **MOD** indica el modo de direccionamiento

MOD	Función:
00	Modo memoria sin desplazamiento
01	Modo memoria con desplazamiento de media palabra
10	Modo memoria con desplazamiento de una palabra
11	Modo registro

Formato de las instrucciones (VII)
Formato de instrucciones μP i80x86/8088 (IV)

- **El segundo byte contiene los operandos** (*continuación*):
- **R/M** se usa para identificar un registro o una posición de memoria y depende del valor del campo MOD
- D8 es un desplazamiento de tamaño media palabra (8 ó 16 bits) y D16 es un desplazamiento de tamaño palabra (16 bits ó 32 bits)

MOD = 11			CALCULO DE LA DIRECCION EFECTIVA			
R/M	W=0	W=1	R/M	MOD = 00	MOD = 01	MOD = 10
000	AL	AX	000	[BX]+[SI]	[BX]+[SI]+D8	[BX]+[SI]+D16
001	CL	CX	001	[BX]+[DI]	[BX]+[DI]+D8	[BX]+[DI]+D16
010	DL	DX	010	[BP]+[SI]	[BP]+[SI]+D8	[BP]+[SI]+D16
011	BL	BX	011	[BP]+[DI]	[BP]+[DI]+D8	[BP]+[DI]+D16
100	AH	SP	100	[SI]	[SI]+D8	[SI]+D16
101	CH	BP	101	[DI]	[DI]+D8	[DI]+D16
110	DH	SI	110	direccion directa	[BP]+D8	[BP]+D16
111	BH	DI	111	[BX]	[BX]+D8	[BX]+D16

Formato de las instrucciones (VIII)
Formato de instrucciones μP i80x86/8088 (V)

- La instrucción **MOV BL, AL** "mueve el byte contenido en el registro fuente AL al registro destino BL"

Solución:

- En el primer byte los primeros 6 bits especifican la operación de mover y, por tanto, deben ser:
  - CODIGO DE OPERACION = 100010
- El bit D indica si el registro que señala el campo REG del segundo byte es el operando fuente o el destino. En este caso se codificará el registro BL en el campo REG del segundo byte; por tanto, D será igual a 1
- El bit W debe indicar una operación de tamaño byte. Por esta razón su valor será 0

Formato de las instrucciones (IX)
Formato de instrucciones μP i80x86/8088 (VI)

El resultado será el siguiente:

- **1er byte** = 1000 10102 = 8Ah
- En el segundo byte, REG indica el operando es BL.
- Su código correspondiente es:
  - REG = 011
- Como el segundo operando es también un registro
  - MOD = 11
- El campo R/M debe especificar que el registro es AL
  - R/M = 000
- Por tanto, el segundo byte completo es:
  - **2º byte** = 1101 10002 = D8h
- Y el código hexadecimal completo para la instrucción MOV BL, AL es:
  - 8A D8h

Formato de las instrucciones (X)
Formato de instrucciones μP i80x86/8088 (VII)

- Supongamos que se dispone de las siguientes variables, definidas en el segmento de datos:
  - Cadena **DB** 0, 0, 0, 0
  - Dato **DW** 0
- **Nota**: Cadena empieza en la posición de memoria 0h del DS y Dato en la posición 4h
- Se desea saber cuál es el código en de las siguientes instrucciones del 80x86/88:

MOV AL, BL:

- **Byte1:** C.O.: 1000 10 - D: 1 - W: 0
  Byte2: MOD:11 - REG:000 - R/M:011
- **Código en hexadecimal:** 8AC3 h

Cod.Op. D W	Mod Reg R/M
100010 1 0	11 000 011

Formato de las instrucciones (XI)
Formato de instrucciones μP i80x86/8088 (VIII)

MOV Dato, BX:

- **Byte1:** C.O.: 1000 10 - D: 0 - W: 1
  Byte2: MOD:00 - REG:011 - R/M:110
- **Código en hexadecimal:** 891E 0400h

Cod.Op. D W	Mod Reg R/M	$Dir.Dato_B$	$Dir.Dato_A$
100010 0 1	00 011 110	00000100	00000000

MOV BX, Dato:

- **Byte1:** C.O.: 1000 10 - D: 1 - W: 1
  Byte2: MOD:00 - REG:011 - R/M:110
- **Código en hexadecimal:** 8B1E 0400h

Cod.Op. D W	Mod Reg R/M	$Dir.Dato_B$	$Dir.Dato_A$
100010 1 1	00 011 110	00000100	00000000

Formato de las instrucciones (y XII)
Formato de instrucciones μP i80x86/8088 (y IX)

MOV CL, Cadena[SI]:

- **Byte1:** C.O.: 1000 10 - D: 1 - W: 0
  Byte2: MOD:10 - REG:001 - R/M:100
- **Código hexadecimal:** 8A8C 0000h

Cod.Op. D W	Mod Reg R/M	$D.Cadena_B$	$D.Cadena_A$
100010 1 0	10 001 100	00000000	00000000

Bibliografía

- Estructura y diseño de computadores (Capítulo 3)
  D. A. Patterson, J. L. Hennessy
  Ed. Reverté
- Fundamentos de los Computadores (Capítulos 6 y 13)
  Pedro de Miguel Anasagasti
  Ed. Paraninfo
- Arquitectura de Computadores (Capítulo 3)
  José A. de Frutos, Rafael Rico
  Ed. Universidad de Alcalá
- 8088-8086, 8087: Programación en Ensamblador en entorno MS-DOS
  Miguel A. Rodríguez-Roselló
  Ed. Anaya Multimedia