Tema 4:
Lenguaje máquina - Lenguaje ensamblador

- Introducción
- Juego 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
- Estructura de un programa ensamblador del µP 80x86/88
- Modos de direccionamiento:
  - Direccionamiento inmediato
  - Direccionamiento directo
  - Direccionamiento relativo
  - Direccionamiento indirecto
  - Direccionamiento implícito
- Ejemplo de hardware real: µP 80x86/88
  - Segmentación de memoria en µP 80x86/88
  - Modos de direccionamiento en el µP 80x86/88
- Formato de instrucciones:
  - Ejemplos de instrucciones
  - Formato de instrucciones en µP 80x86/88

Espacio reservado para notas del alumno

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
- Estructura de Computadores (Capítulo 2)
  José M. Angulo
  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

Espacio reservado para notas del alumno

Introducción (I)

Instrucción:

- Operación expresada mediante la codificación binaria de **cadenas de 1's y 0's** ⇨ **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

Espacio reservado para notas del alumno

Introducción (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)

Espacio reservado para notas del alumno

Introducción (III)

- 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

Espacio reservado para notas del alumno

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

Espacio reservado para notas del alumno

Juego de instrucciones (II)
Instrucciones de transferencia de datos

- 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 (8088/80x86):
  - **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

Espacio reservado para notas del alumno

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**

Espacio reservado para notas del alumno

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
        - Sistema operativo
        - Hardware (E/S)
- Saltos incondicionales: siempre se produce el salto
- Saltos condicionales: se realiza el salto si se da la condición sobre los *flags*

Espacio reservado para notas del alumno

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] [E]
- **G** - Mayor que
- **L** - Menor que
- **A** - Superior*
- **B** - Inferior*

P - Paridad
PE - Paridad par
PO - Paridad impar

CXZ - CX=0

*Se refieren a operandos sin signo

Espacio reservado para notas del alumno

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

Espacio reservado para notas del alumno

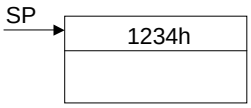
Juego de instrucciones (VII)
Instrucciones de bifurcación (IV)
Subrutinas (I)

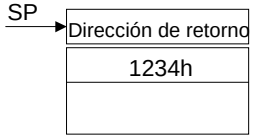
Llamadas a subrutinas: salvan la posición de retorno
Las instrucciones de salto a la subrutina y de regreso al programa principal van emparejadas

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

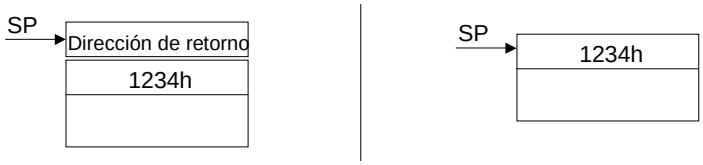
Espacio reservado para notas del alumno

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

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

Espacio reservado para notas del alumno

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

- 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.

Espacio reservado para notas del alumno

Juego de instrucciones (X)
Instrucciones de bifurcación (VII)
Interrupciones (II)

- Los pasos para llamar a una interrupción son:
  - Identificar la interrupción necesaria
  - Pasar los parámetros a la subrutina
  - Llamar a la interrupción
- Las interrupciones implican una ruptura en la secuencia del programa
- Se salta al código que da ese servicio y cuando se ha terminado, se vuelve a la ejecución del programa en curso
- Las interrupciones salvaguardan los *flags* y los registros que emplean

```
MOV AH, 4Ch      ; Indica al sistema operativo de la
INT 21h          ; terminación del programa en curso
```

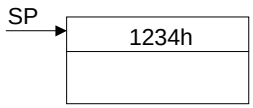

Espacio reservado para notas del alumno

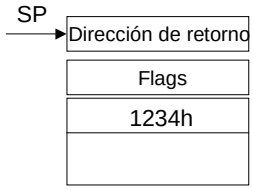
Juego de instrucciones (XI)
Instrucciones de bifurcación (VIII)
Interrupciones (III)

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

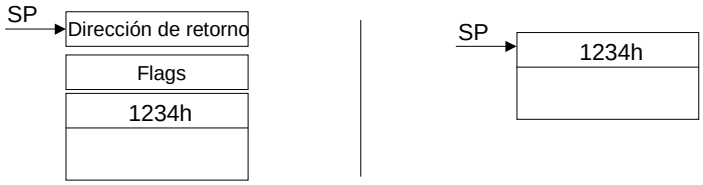
Espacio reservado para notas del alumno

Juego de instrucciones (XII)
Instrucciones de bifurcación (IX)
Interrupciones (IV)

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

Espacio reservado para notas del alumno

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

- **Instrucciones de comparación**
  - No almacenan el resultado, sólo modifican los *flags*
  - **CMP**: compara números (resta)
  - **TEST**: comparación lógica a nivel de bits (AND)
- **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

Espacio reservado para notas del alumno

Juego de instrucciones (XIV)
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

Espacio reservado para notas del alumno

Juego de instrucciones (XV)
Instrucciones de entrada/salida
Instrucciones 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

Espacio reservado para notas del alumno

Estructura de un programa 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
```


Espacio reservado para notas del alumno

Estructura de un programa ensamblador (II)

```
Suma dos números Num1 y Num2 y deja el resultado en Res:
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
```


Espacio reservado para notas del alumno

Estructura de un programa 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
```


Espacio reservado para notas del alumno

Modos de direccionamiento (I)

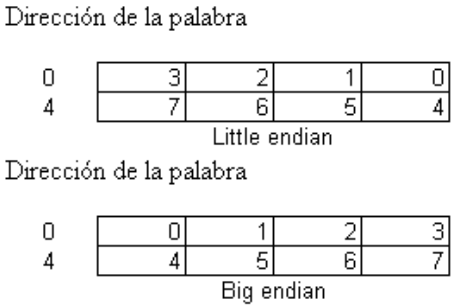
- El modo de direccionamiento determina la ubicación de un operando:
  - En la propia instrucción
  - En un registro
  - En memoria principal

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	

Espacio reservado para notas del alumno

Modos de direccionamiento (II)
Direccionamiento inmediato

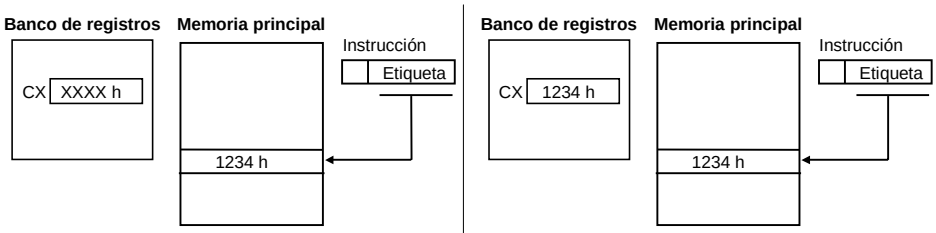
- El operando se encuentra en la propia instrucción
- Ejemplo: **MOV CX, 0010h** (Su código máquina es B9 10 00 h)
- La manera de expresar el dato inmediato depende del formato de la máquina
  - Big endian
  - Little endian



Espacio reservado para notas del alumno

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:
  - **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** (p. ej. 80x86/88). Ej. Mov CX, Etiqueta



Antes de ejecutar la instrucción

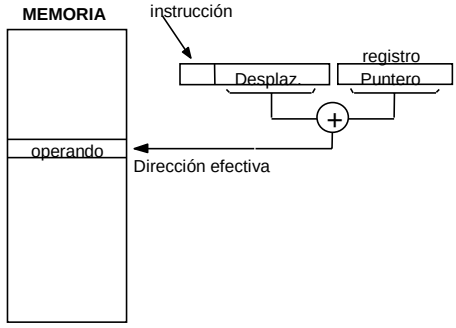
Después de ejecutar la instrucción

Espacio reservado para notas del alumno

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]**



Espacio reservado para notas del alumno

Modos de direccionamiento (V)
Direccionamiento relativo (II)

Dirección efectiva = Reg. Referencia + desplazamiento

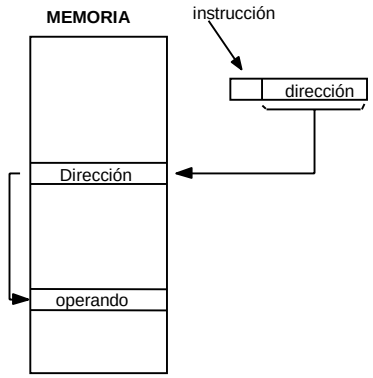
Modo de direccionamiento	Registro de referencia	Cálculo dirección efectiva
Relativo a contador de programa	Contador de programa (CP)	DF = CP + desplazamiento
Relativo a registro base	Un registro base (Rb)	DF = Rb + desplazamiento
Relativo a registro índice	Un registro índice (Ri)	DF = Ri + desplazamiento
Relativo a pila	Registro de pila (SP)	DF = 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

Espacio reservado para notas del alumno

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
- Es posible que se realicen múltiples niveles de indirección
- Su utilidad más común es la de acceso a diversas informaciones mediante tablas de punteros
- Ejemplo:
  - **MOV AX,[[100]]**



Espacio reservado para notas del alumno

Modos de direccionamiento (VII)
Direccionamiento implícito

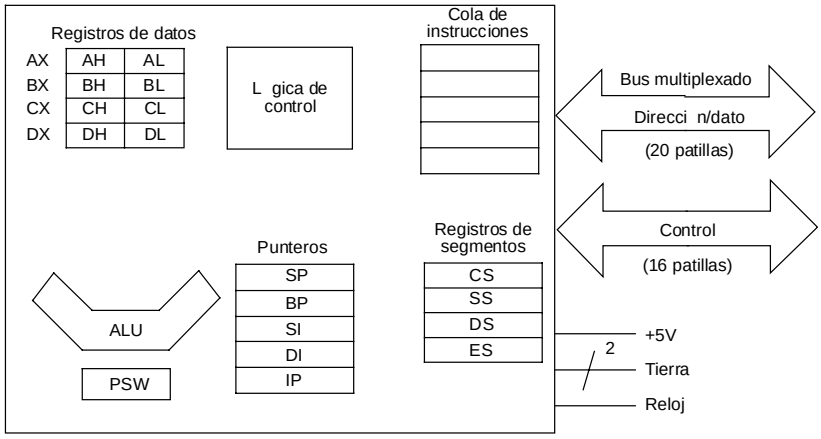
- En la instrucción no se indica explícitamente el lugar donde se encuentra el operando
- Requiere que el programador conozca con que operandos se está trabajando

- Ejemplos:
  - **MUL BX** ➪ AX x BX → DX,AX
    donde AX y DX son operandos implícitos

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

Espacio reservado para notas del alumno

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



Espacio reservado para notas del alumno

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:

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

Espacio reservado para notas del alumno

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

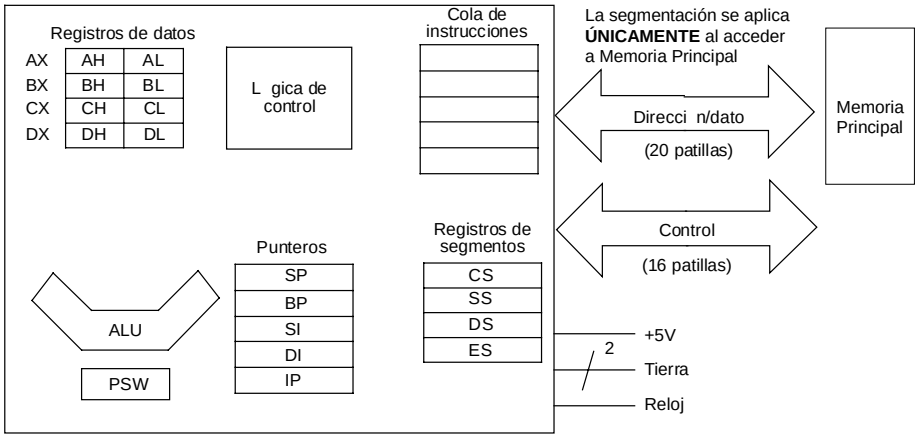
Banco de registros:

- 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
  - CS - Segmento de código
  - ES - Segmento extra de datos

Dirección Física = Segmento : Dirección Efectiva

Espacio reservado para notas del alumno

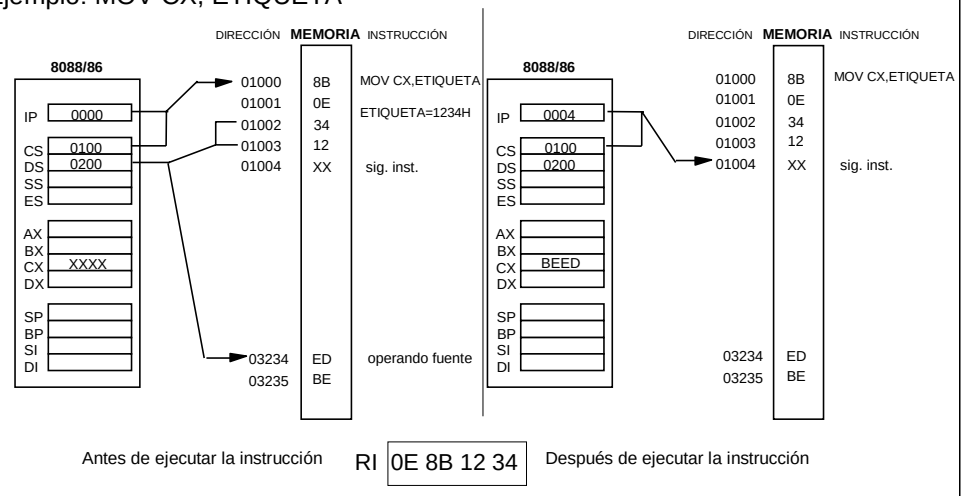
Ejemplo de hardware real: μP 80x86/88 (IV)
Segmentación de memoria en μP 80x86/88 (III)



Espacio reservado para notas del alumno

Ejemplo de hardware real: μP 80x86/88 (V)
Modos de direccionamiento en el μP 80x86/88 (I)
Direccionamiento directo en el μP 80x86/88

- Ejemplo: MOV CX, ETIQUETA

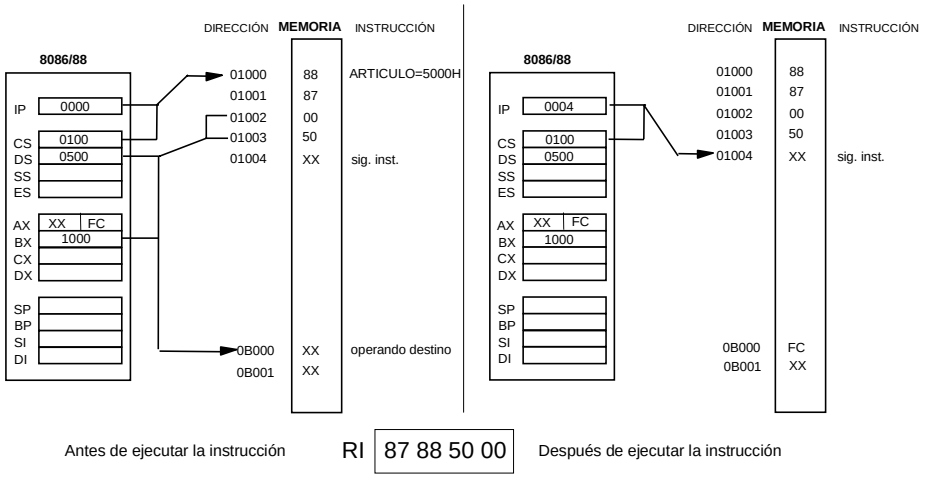


Antes de ejecutar la instrucción RI 0E 8B 12 34 Después de ejecutar la instrucción

Espacio reservado para notas del alumno

Ejemplo de hardware real: µP 80x86/88 (VI)
Modos de direccionamiento en el µP 80x86/88 (II)
Direccionamiento relativo a base µP 80x86/88

- Ejemplo: MOV [BX]+ARTÍCULO, AL

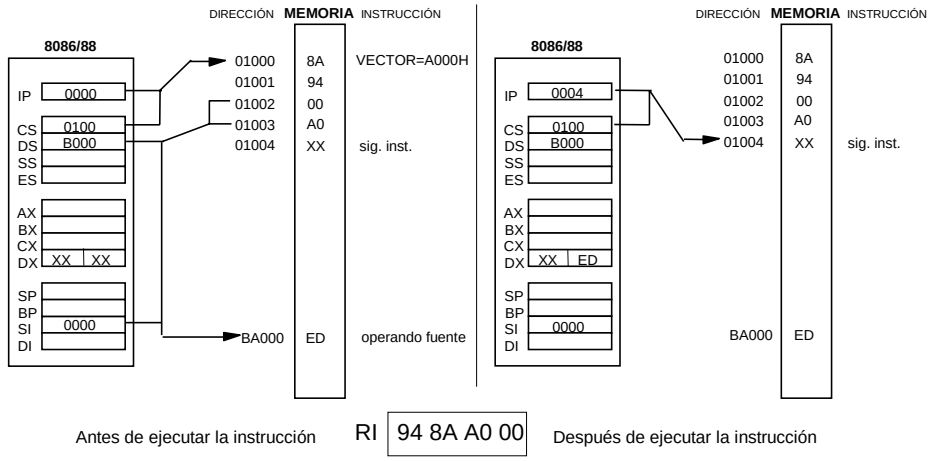


Antes de ejecutar la instrucción RI 87 88 50 00 Después de ejecutar la instrucción

Espacio reservado para notas del alumno

Ejemplo de hardware real: µP 80x86/88 (VII)
Modos de direccionamiento en el µP 80x86/88 (III)
Direccionamiento mediante índice en µP80x86/88

- Ejemplo: MOV DL, VECTOR[SI]

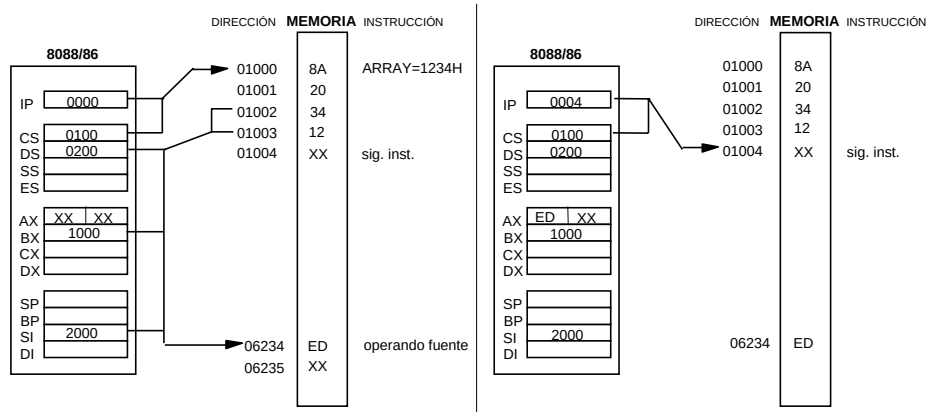


Antes de ejecutar la instrucción RI 94 8A A0 00 Después de ejecutar la instrucción

Espacio reservado para notas del alumno

Ejemplo de hardware real: μP 80x86/88 (VIII)
Modos de direccionamiento en el μP 80x86/88 (IV)
Direccionamiento mediante índice y base μP80x86/88

- Ejemplo: MOV AH, [BX] [SI]+ARRAY



Antes de ejecutar la instrucción RI 20 8A 12 34 Después de ejecutar la instrucción

Espacio reservado para notas del alumno

Ejemplo de hardware real: µP 80x86/88 (IX)
Modos de direccionamiento en el µP 80x86/88

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		

B. Registros

```
CS = 0100h
DS = 0200h
SS = 0300h
BX = 1234h
SI = 0010h
IP = 0025h
SP = 0200h
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 = 03710h
- **Dirección física en la que se encuentra la cabecera de la pila**
  DF = SS x 10h + SP = 0300h x 10h + 0200h = 3200h

Espacio reservado para notas del alumno

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.

Espacio reservado para notas del alumno

Formato de 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

- Según esto, un formato típico cuenta con esta información:

Cod. operación	Operandos

Espacio reservado para notas del alumno

Formato de 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		

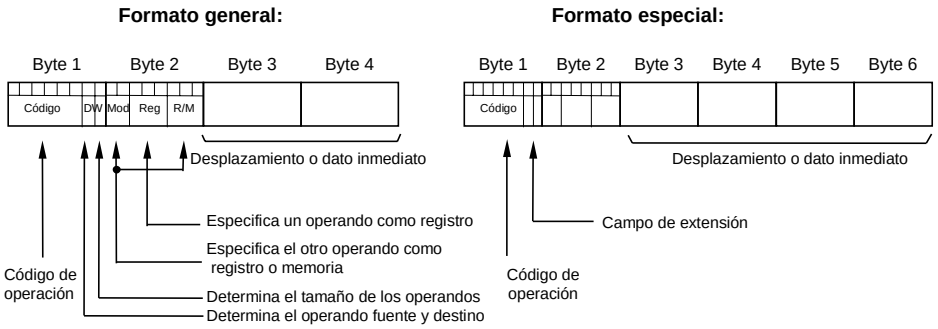
- **80x86/8088**

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)		

Espacio reservado para notas del alumno

Formato de instrucciones (IV)
Formato de instrucciones μP 80x86/8088 (I)

- Cuenta con dos formatos:



Espacio reservado para notas del alumno

Formato de instrucciones (V)
Formato de instrucciones μP 80x86/8088 (II)

Formato general:

- 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)

Espacio reservado para notas del alumno

Formato de instrucciones (VI)
Formato de instrucciones μP 80x86/8088 (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 según:

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

Espacio reservado para notas del alumno

Formato de instrucciones (VII)
Formato de instrucciones μP 80x86/8088 (IV)

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

- Donde 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)

Espacio reservado para notas del alumno

Formato de instrucciones (VIII)
Ejemplos de formatos μP 80x86/8088 (I)

- 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_2
  - 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

Espacio reservado para notas del alumno

Formato de instrucciones (IX)
Ejemplos de formatos μP 80x86/8088 (II)

- El resultado será el siguiente:
  - **1er byte = 1000 1010$_2$ = 8A$_{16}$**
- En el segundo byte, REG indica el operando es BL. Su código correspondiente es:
  - REG = 011
- Como el segundo operando es también un registro tengo que MOD debe valer 11. El campo R/M debe especificar que el registro es AL y su códificación es 000. Esto da:
  - MOD = 11 R/M = 000
- Por tanto, el segundo byte completo es:
  - **2º byte = 1101 1000$_2$ = D8$_{16}$**
- Y el código hexadecimal completo para la instrucción es:

  MOV BL,AL = 8A D8$_{16}$

Espacio reservado para notas del alumno

Formato de instrucciones (X)
Ejemplos de formatos μP 80x86/8088 (III)

- 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 lenguaje máquina 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

- **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

Espacio reservado para notas del alumno

Formato de instrucciones (XI)
Ejemplos de formatos μP 80x86/8088 (IV)

- **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

Espacio reservado para notas del alumno