


Arquitectura de Computadoras



Curso 2020
Clase 1
Instrucciones - MDD

Prof. Jorge M. Runco

Curso 2020

1



Temas de clase

- Registros
- Instrucciones
- Tipos de instrucciones
- Modos de direccionamiento
- Programas

Organización de registros

- Registros visibles al usuario: son utilizados por el programador (AX, BX,... BP, SP..)
- Registros de control y estado: son utilizados por la UC para controlar la operación de la CPU: no son visibles por el programador (MAR, MBR, IP.....)

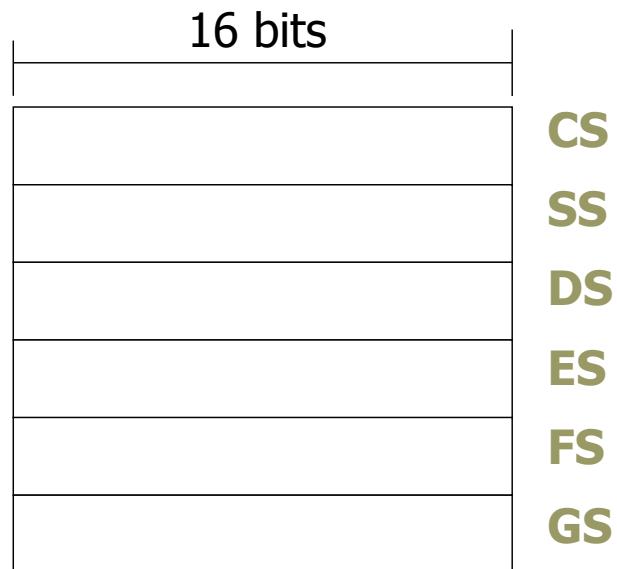
Organización de registros CPU PII Intel (principales)(1)

16 bits	8 bits	8 bits	
	AH AX AL		EAX
	BH BX BL		EBX
	CH CX CL		ECX
	DH DX DL		EDX

De uso general

Organización de registros

CPU PII Intel (principales)(2)



Segmentos

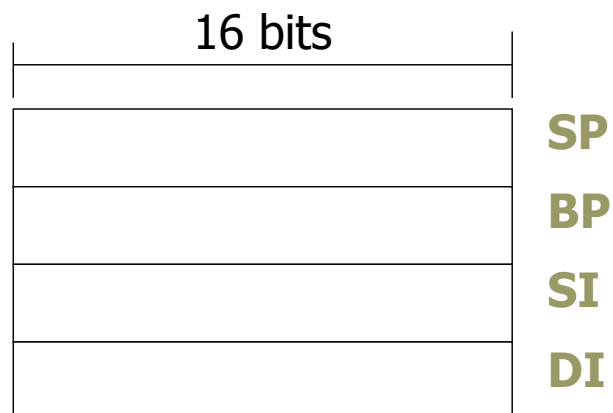
Prof. Jorge Runco

Curso 2020

5

Organización de registros

CPU PII Intel (principales)(3)



Punteros de pila y registros índices

Prof. Jorge Runco

Curso 2020

6

Organización de registros

CPU PII Intel (principales)(4)

EIP

EFLAGS

PC y banderas

Organización de registros

CPU PII Intel (principales)(5)

- AX : acumulador, es el principal en las operaciones aritméticas
- BX : puntero base (dir de memoria)
- CX : contador, interviene en instrucciones de ciclo
- DX : datos, participa en multiplicación y división

Organización de registros

CPU PII Intel (principales)(6)

- SI y DI : apuntadores que utilizan las instrucciones que recorren arreglos o tablas
- BP y SP : también son apuntadores a memoria, pero a una zona especial: pila ó stack
- E : reg de 32 bits

Instrucciones - Intel

- Tienen la forma :

instrucción destino,fuente

- destino y fuente son 2 operandos, donde c/u de ellos está especificado por alguno de los mdd vistos, el otro operando es un registro de la CPU

Instrucciones - Intel (2)

❖ Llamando :

- mem = especificación de una dirección de memoria
- reg = registro de la CPU
- inm = dato inmediato



Las instrucciones
tienen la forma

Instrucciones - Intel (3)

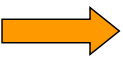
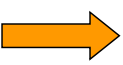


- Instrucción mem, reg
- Instrucción reg , mem
- Instrucción reg , reg
- Instrucción reg , inm
- Instrucción mem, inm

Instrucciones - Intel (4)

- El nombre destino y fuente proviene del hecho que si hay un movimiento de datos, es desde la derecha (fuente) hacia la izquierda (destino).
- En una suma hay 2 operandos y el resultado se almacena en el lugar del operando izquierdo (destino).

Instrucciones-Intel 8086 – (MSX88)





Ejemplos:

- `ADD AX,BX`  `AX=AX+BX`
- `ADD AL,AH`  `AL=AL+AH`
- `MOV AL,CH`  `AL=CH`
- `SUB AX,BX`  `AX=AX - BX`

❖ Direcccionamiento por registro

Instrucciones-Intel 8086-(MSX88)(2)




Ejemplos:

- `ADD AX,35AFh`  `AX=AX+35AFh`
- `ADD AL,15`  `AL=AL+15`
- `MOV AL,3Eh`  `AL=3Eh`
- `SUB AX,1234h`  `AX=AX - 1234h`

❖ Direccionamiento Inmediato

Instrucciones - Intel 8086 (3)



Ejemplos:

- `ADD AX, [35AFh]` 
`AX = AX + contenido direcc. 35AFh y 35B0h`
- `ADD AL, DATO`  **(MSX88)**
`AL = AL + contenido variable DATO (8 bits)`
- `MOV CH, NUM1`  **(MSX88)**
`CH = contenido variable NUM1 (8 bits)`

❖ Direccionamiento Directo

Instrucciones-Intel 8086(MSX88)(4)



Ejemplos:

- `ADD AX, [BX]` 
AX = AX + dato almacenado en dirección contenida en BX y la que sigue
- `MOV [BX], AL` 
dato en la dirección contenida en BX = AL

❖ Direccionamiento Indirecto por registro

Instrucciones - Intel 8086 (5)



Ejemplos:

- `MOV CX, [BX+SI]` 
CX = dato almacenado en la direcc. BX+SI y la siguiente
- `MOV [BX+DI], AL` 
dato almacenado en la direcc. BX+DI = AL

❖ Direccionamiento base + índice

Instrucciones - Intel 8086 (6)



Ejemplos:

- `MOV AL, [BX+2]` 
AL=dato almacenado en dir BX+2
- `MOV [BX+2Ah], AX` 
dato almacenado en dir BX+2Ah y la que sigue = AX (16 bits)

❖ Direccionamiento Relativo por registro

Instrucciones - Intel 8086 (7)

Ejemplos:

- `MOV AL, [BX+SI+2]` 
AL = dato almacenado en la dir BX+SI+2
- `MOV [BX+DI+2Ah], AX` 
dato almacenado en la dir BX+DI+2Ah y la que sigue = AX (16 bits)

❖ Direccionamiento relativo base+índice

Orden de los bytes

1000 34 92 1 ORG 1000H
 2 NUM1 DW 9234H

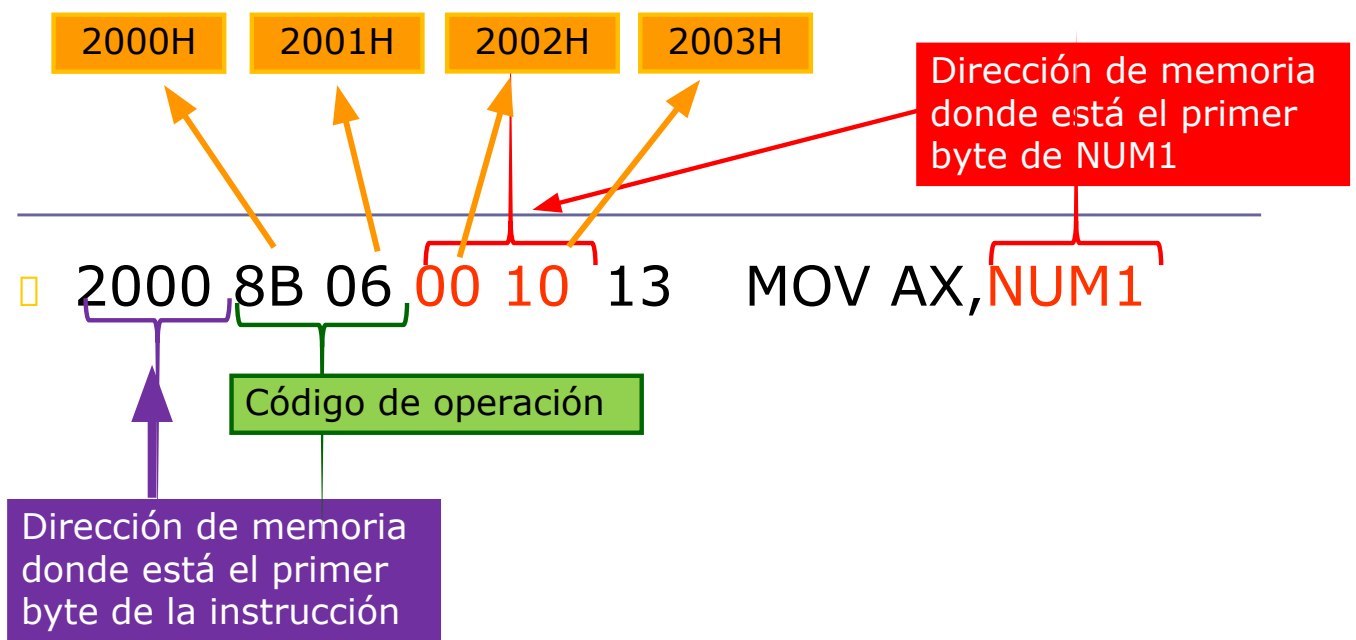
Dir 1001H

El número de 2 bytes es almacenado en sentido invertido, la parte alta del número en la dirección más alta

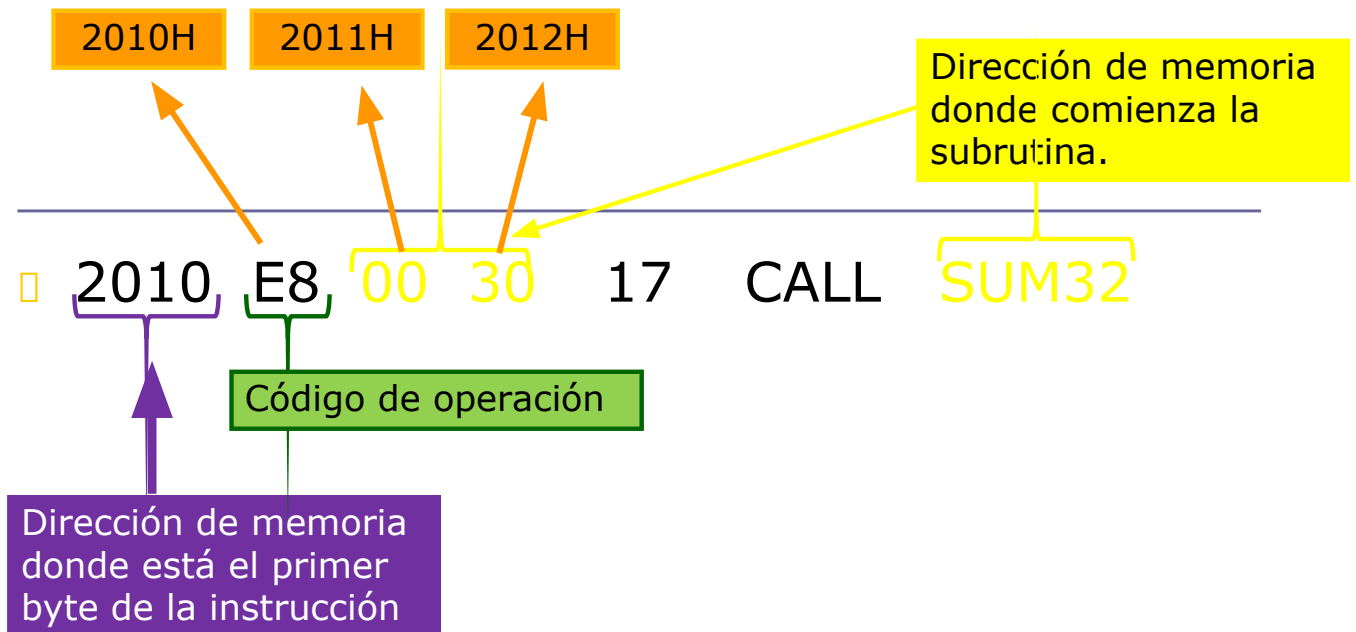
Orden de los bytes (2)

		12	ORG 2000H	
2000	8B 06 00 10	13	MOV AX, NUM1	
2004	8B 16 02 10	14	MOVDX, NUM1+2	
2008	8B 0E 04 10	15	MOV CX, NUM2	
200C	8B 1E 06 10	16	MOV BX, NUM2+2	
2010	E8 00 30	17	CALL SUM32	
2013	F4	18	HLT	
		19	END	

Este mecanismo se llama "little-endian"



La instrucción ocupa 4 bytes de memoria: 2000H, 2001H, 2002H y 2003H. Por eso la próxima instrucción comienza en 2004H. La variable NUM1 es reemplazada en la instrucción por 1000H, su dirección de comienzo (donde está almacenado el dato).



La instrucción ocupa 3 bytes de memoria: 2010H, 2011H y 2012H. Por eso la próxima instrucción comienza en 2013H. La etiqueta (nombre de la subrutina) SUM32 es reemplazada en la instrucción por 3000H, dirección de comienzo de la misma (dirección de la primera instrucción).

Problema

- Intel 80x86, Pentium y VAX son “little-endian”.
- IBM S/370, Motorola 680x0 (Mac), y la mayoría de los RISC son “big-endian”.

Incompatibilidad !!!