Microprocesadores 1121060

Tema 3.

Modos de Direccionamiento.

1. Modos de direccionamiento
2. Formato de Instrucción
3. Ejemplo de Aplicaciones con todos los modos de direccionamiento

Tema 3. Modos de Direccionamiento

- **Direccionamiento Inmediato.**
- Transfiere un byte o palabra de datos inmediato hacia el operando destino. Este modo es usado para inicializar registros o localidades de memoria y para operara sobre ellos con valores constantes de datos.
- Ej: MOV AX,0ABCDH
  MOV BL,12H
- Las instrucciones que usan el modo de direccionamiento inmediato obtienen el dato como parte de la instrucción.
  B8 00 10 MOV AX,1000H
- Este modo no opera con registros de segmento, por lo que no se puede cargar un registro de segmento de manera inmediata.

Tema 3. Modos de Direccionamiento

- **Direccionamiento Directo.**
- Transfiere un byte o palabra contenido en una localidad de memoria en el segmento DS a un registro de 8 o 16 bits. La localidad de memoria puede ser el operando fuente o destino.
- Ej: **MOV AL,[1234H]**
- En el modo de direccionamiento directo, la dirección de memoria se proporciona directamente como parte de la instrucción (Puede ser a través de etiquetas, en las cuales el programador no necesita conocer la dirección numérica)
- **MOV AH, MEMBDS**
- 8A 00 10 **MOV AH,[MEMBDS]** AH←[1000H]

Tema 3. Modos de Direccionamiento

Direccionamiento Base mas Indice.

- Transfiere un byte o palabra entre un registro y una localidad de memoria direccionada por la suma de un registro base más un registro índice.
- Este modo es usado para el acceso a tablas.
- Ej. MOV AX,[BX+DI]
- En muchos casos, el registro base retiene la dirección de inicio de un arreglo de memoria, y el registro índice retiene la posición relativa de un dato en un arreglo.

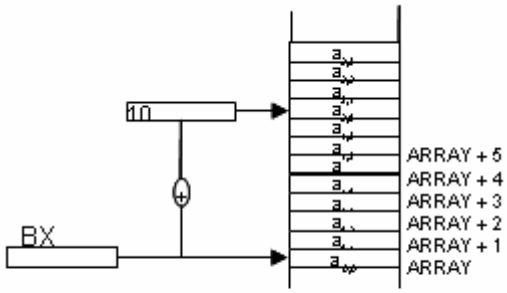
```
8B 00   MOV AX,[BX+SI]          AH←[BX+SI+1], AL←[BX+SI];
FF 21   JMP [BX+DI]             IP←[BX+DI+1:BX+DI];
FE 02   INC BYTE PTR[BP+SI]     [BP+SI]←[BP+SI]+1  ;
FF 0B   DEC WORD PTR[BP+DI]     [BP+DI+1:BP+DI]←[BP+DI+1:BP+DI]-1
```

Tema 3. Modos de Direccionamiento

Direccionamiento Relativo a Registro

- Transfiere un byte o palabra entre un registro y una localidad de memoria direccionada por **un registro** base o un registro índice **más un desplazamiento**.
- Si la localidad de memoria se direcciona por la **suma de un registro base y un desplazamiento**, también se conoce como **direccionamiento basado**.
- Ej. MOV AX,[BX+10H]
- Si la localidad de memoria se direcciona por la **suma de un registro índice y un desplazamiento**, también se conoce como **direccionamiento indexado.**
- Ej. MOV AX,[SI+500H]

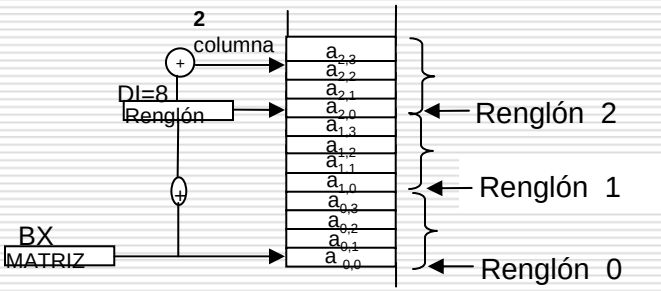
  - MOV AX,[BX+4]
  - MOV AX,ARRAY [SI]
  - MOV LIST[BP],CL
  - MOV ARRAY[DI],AL



MOV AL,[BX+10]

Tema 3. Modos de Direccionamiento

Este tipo de direccionamiento es usado comúnmente para direccionar arreglos de datos en memoria de dos dimensiones (matrices).



MOV AL,[BX+DI+2]

Selecciona elemento $a_{2,2}$

MOV BX, OFFSET MATRIZ

MOV DI,8

MOV AL,[BX+DI+2]

$$\begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \end{pmatrix}$$

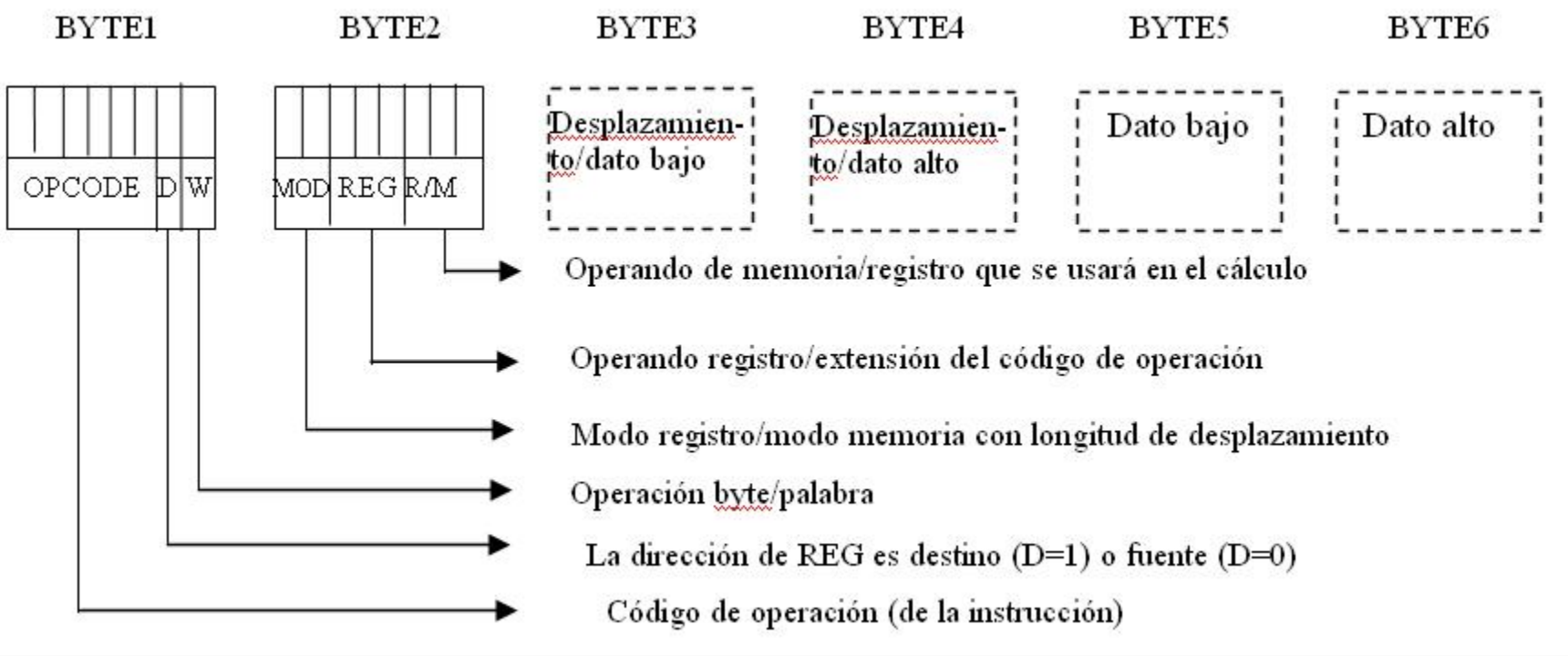
Tema 3. Modos de Direccionamiento

		Instrucciones de Cadena (string)			
Mnemónico general Op-code Operando	**Código Objeto**	**Mnemónico**	**Segmento de memoria**	**Operación simbólica**	**Descripción**
STOSB	AA	STOSB	Extra	ES:[DI]←AL Si DF=0, DI←DI+1 Si DF=1, DI←DI-1	Transfiere un byte o palabra del registro AL o AX a la cadena direccionada por DI en el segmento extra; Si DF=0, incrementa DI, de lo contrario decrementa DI; las banderas no son afectadas.
STOSW	AB	STOSW	Extra	ES:[DI] ← AL ES:[DI+1] ←AH Si DF=0, DI←DI+2 Si DF=1, DI←DI-2	
LODSB	AC	LODSB	Datos	AL← DS:[SI] Si DF=0,SI←SI+1 Si DF=1, SI←SI-1	Transfiere un byte o palabra de la cadena direccionada por SI en el segmento de datos al registro AL o AX; Si DF=0, incrementa SI, de lo contrario decrementa SI; las banderas no son afectadas.
LODSW	AD	LODSW	Datos	AL←DS:[SI] AH←DS:[SI+1] Si DF=0, SI←SI+2 Si DF=1, SI←SI-2	

Codificación en Lenguaje Máquina Formato de una Instrucción

- Para convertir un programa en lenguaje ensamblador a código máquina, debemos convertir cada instrucción en lenguaje ensamblador a su instrucción equivalente en código máquina.
- El código máquina especifica que operación se realizará, qué operando u operandos serán usados (registros, localidad de memoria, dato inmediato), el tamaño del dato (byte o palabra).
- Toda la información se encuentra codificada en los bits del código máquina de la instrucción.

Codificación en Lenguaje Máquina
Formato de una Instrucción



Codificación en Lenguaje Máquina
Formato de una Instrucción

- El bit Z se utiliza con el prefijo de repetición en instrucciones de cadena cuando se requiere comparar con la bandera de cero (Z). El bit Z=1 indica que la instrucción se repite mientras la bandera ZF=1, el bit Z=0 indica que la instrucción se repite mientras la bandera ZF=0.
- En algunas instrucciones que utilizan dato inmediato se requiere indicar el valor del campo “S”, el cual se emplea de la siguiente manera:
- Si los bits SW=01 entonces se requiere especificar los 16 bits del dato inmediato que se utilizará como operando.
- Si SW=11 entonces el byte de dato inmediato es extendido en signo para formar el operando de 16 bits.

Ejemplos de formatos de Instrucciones del 8086

intel

Table 2. Instruction Set Summary

Mnemonic and Description	Instruction Code			
DATA TRANSFER				
MOV – Move:	76543210	76543210	76543210	76543210
Register/Memory to/from Register	100010dw	mod reg r/m		
Immediate to Register/Memory	1100011w	mod000r/m	data	data if w = 1
Immediate to Register	1011wreg	data	data if w = 1	
Memory to Accumulator	1010000w	addr-low	addr-high	
Accumulator to Memory	1010001w	addr-low	addr-high	
Register/Memory to Segment Register	10001110	mod0regr/m		
Segment Register to Register/Memory	10001100	mod0regr/m		
PUSH – Push:				
Register/Memory	11111111	mod110r/m		
Register	01010reg			
Segment Register	000reg110			
POP – Pop:				
Register/Memory	10001111	mod000r/m		
Register	01011reg			
Segment Register	000reg111			
XCHG – Exchange:				
Register/Memory with Register	1000011w	mod regr/m		
Register with Accumulator	10010reg			
IN – Input from:				
Fixed Port	1110010w	port		
Variable Port	1110110w			
OUT – Output to:				
Fixed Port	1110011w	port		
Variable Port	1110111w			
XLAT – Translate Byte to AL	11010111			
LEA – Load EA to Register	10001101	mod regr/m		
LDS – Load Pointer to DS	11000101	mod regr/m		
LES – Load Pointer to ES	11000100	mod regr/m		
LAHF – Load AH with Flags	10011111			
SAHF – Store AH into Flags	10011110			
PUSHF – Push Flags	10011100			
POPF – Pop Flags	10011101			

Ejemplos de formatos de Instrucciones del 8086

Table 2. Instruction Set Summary (Continued)

Mnemonic and Description	Instruction Code			
LOGIC	76543210	76543210	76543210	76543210
NOT = Invert	1111011 w	mod010 r/m		
SHL/SAL = Shift Logical/Arithmetic Left	110100 v w	mod100 r/m		
SHR = Shift Logical Right	110100 v w	mod101 r/m		
SAR = Shift Arithmetic Right	110100 v w	mod111 r/m		
ROL = Rotate Left	110100 v w	mod000 r/m		
ROR = Rotate Right	110100 v w	mod001 r/m		
RCL = Rotate Through Carry Flag Left	110100 v w	mod010 r/m		
RCR = Rotate Through Carry Right	110100 v w	mod011 r/m		
AND = And:				
Reg./Memory and Register to Either	001000 d w	mod reg r/m		
Immediate to Register/Memory	1000000 w	mod100 r/m	data	data if w = 1
Immediate to Accumulator	0010010 w	data	data if w = 1	
TEST = And Function to Flags, No Result:				
Register/Memory and Register	1000010 w	mod reg r/m		
Immediate Data and Register/Memory	1111011 w	mod000 r/m	data	data if w = 1
Immediate Data and Accumulator	1010100 w	data	data if w = 1	
OR = Or:				
Reg./Memory and Register to Either	000010 d w	mod reg r/m		
Immediate to Register/Memory	1000000 w	mod001 r/m	data	data if w = 1
Immediate to Accumulator	0000110 w	data	data if w = 1	
XOR = Exclusive or:				
Reg./Memory and Register to Either	001100 d w	mod reg r/m		
Immediate to Register/Memory	1000000 w	mod110 r/m	data	data if w = 1
Immediate to Accumulator	0011010 w	data	data if w = 1	
STRING MANIPULATION				
REP = Repeat	1111001 z			
MOVS = Move Byte/Word	1010010 w			
CMPS = Compare Byte/Word	1010011 w			
SCAS = Scan Byte/Word	1010111 w			
LODS = Load Byte/Wd to AL/AX	1010110 w			
STOS = Stor Byte/Wd from AL/A	1010101 w			
CONTROL TRANSFER				
CALL = Call:				
Direct within Segment	11101000	disp-low	disp-high	
Indirect within Segment	11111111	mod010 r/m		
Direct Intersegment	10011010	offset-low	offset-high	
		seg-low	seg-high	
Indirect Intersegment	11111111	mod011 r/m		

Ejemplos de formatos de Instrucciones del 8086

intel

Table 2. Instruction Set Summary (Continued)

Mnemonic and Description	Instruction Code	
	76543210	76543210
PROCESSOR CONTROL		
CLC – Clear Carry	11111000	
CMC – Complement Carry	11110101	
STC – Set Carry	11111001	
CLD – Clear Direction	11111100	
STD – Set Direction	11111101	
CLI – Clear Interrupt	11111010	
STI – Set Interrupt	11111011	
HLT – Halt	11110100	
WAIT – Wait	10011011	
ESC – Escape (to External Device)	11011xxx	modxxxr/m
LOCK – Bus Lock Prefix	11110000	

NOTES:
AL = 8-bit accumulator
AX = 16-bit accumulator
CX = Count register
DS = Data segment
ES = Extra segment
Above/below refers to unsigned value
Greater = more positive;
Less = less positive (more negative) signed values
if d = 1 then "to" reg; if d = 0 then "from" reg
if w = 1 then word instruction; if w = 0 then byte instruction
if mod = 11 then r/m is treated as a REG field
if mod = 00 then DISP = 0*, disp-low and disp-high are absent
if mod = 01 then DISP = disp-low sign-extended to 16 bits, disp-high is absent
if mod = 10 then DISP = disp-high; disp-low
if r/m = 000 then EA = (BX) + (SI) + DISP
if r/m = 001 then EA = (BX) + (DI) + DISP
if r/m = 010 then EA = (BP) + (SI) + DISP
if r/m = 011 then EA = (BP) + (DI) + DISP
if r/m = 100 then EA = (SI) + DISP
if r/m = 101 then EA = (DI) + DISP
if r/m = 110 then EA = (BP) + DISP*
if r/m = 111 then EA = (BX) + DISP
DISP follows 2nd byte of instruction (before data if required)
*except if mod = 00 and r/m = 110 then EA = disp-high; disp-low.
if s w = 01 then 16 bits of immediate data form the operand
if s w = 11 then an immediate data byte is sign extended to form the 16-bit operand
if v = 0 then "count" = 1; if v = 1 then "count" in (CL)
x = don't care
z is used for string primitives for comparison with ZF FLAG

SEGMENT OVERRIDE PREFIX

0 0 1 reg 1 1 0

REG is assigned according to the following table:

16-Bit (w = 1)	8-Bit (w = 0)	Segment
000 AX	000 AL	00 ES
001 CX	001 CL	01 CS
010 DX	010 DL	10 SS
011 BX	011 BL	11 DS
100 SP	100 AH	
101 BP	101 CH	
110 SI	110 DH	
111 DI	111 BH	

Instructions which reference the flag register file as a 16-bit object use the symbol FLAGS to represent the file:
FLAGS = X:X:X:X:(OF):(DF):(IF):(TF):(SF):(ZF):X:(AF):X:(PF):X:(CF)