LABORATORIO DE SISTEMAS DIGITALES CON MICROPROCESADORES

Modos de Direccionamiento

1

Modos de Direccionamiento

1. ¿Qué son los Modos de direccionamiento?
2. Formato de Instrucción
3. Ejemplo de Aplicaciones con todos los modos de direccionamiento

2

¿Qué son los Modos de Direccionamiento?

- Los ***Modos de Direccionamiento*** son las diversas formas con las que se puede indicar a un *µP* dónde encontrar o depositar un dato, en una instrucción. Identifica los operandos de la operación a realizar, fuente y destino, de los datos sobre los que se operará.
- Los operandos se pueden especificar por: un registro de la CPU, una localidad de memoria, un puerto de Entrada/Salida o un dato inmediato.
- Existen dos grupos principales de Modos de Direccionamiento, según los operandos se encuentren en registros o en memoria serán: modos de *Registro* o modos de *Memoria*.
- La dirección física de memoria o registro donde se encuentra realmente el operando se llama *Dirección Efectiva*, y se representa < *ea* >. Según el modo de direccionamiento utilizado, la CPU tendrá que realizar unos cálculos distintos hasta obtener el valor de dicha dirección efectiva.
- Las instrucciones pueden contener hasta dos operandos que se denominan *Operando Fuente*, el del lado derecho, y *Operando Destino*, el del lado izquierdo. En estos casos, cada operando tendrá su propio modo de direccionamiento.

3

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 operar sobre ellos con valores constantes de datos.
- Ejemplo: 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**.

4

Direccionamiento por Registro

- Transfiere un byte o palabra desde un registro fuente hasta un registro destino.

- Ej. MOV AX,CX
- INC BX
- El operando no requiere ninguna referencia de memoria.

Nota: Los dos operandos no pueden ser registros de segmento.

- El registro de segmento de código (CS) nunca puede utilizarse como destino.
- No se permite el acceso entre registros de segmento ni de distintos tamaños.

5

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]

6

Direccionamiento Indirecto

- El modo de direccionamiento directo se usa para acceder localidades de memoria de manera no frecuente. Sin embargo cuando una localidad de memoria debe ser leída o escrita varias veces dentro de un programa, la búsqueda repetida de la dirección lógica hace este modo ineficiente. El modo de **direccionamiento indirecto** resuelve este problema almacenando esta dirección de memoria en un registro base (BX, BP)o un registro índice (SI o DI)

- MOV [DI],BH
- MOV [BP],DL

8B 04	MOV AX, [SI]	AL←[SI]; AH←[SI+1]
FF 25	JMP [DI]	IP←[DI+1:DI]
FE 46 00	INC BYTE PTR[BP]	[BP]←[BP] + 1
FF 0F	DEC WORD PTR[BX]b	[BX+1:BX]←[BX+1:BX]-1

7

Direccionamiento Base más Índice

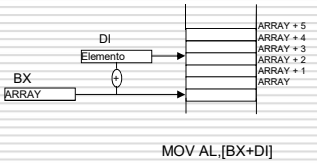
- Transfiere un byte o palabra entre un registro y una localidad de memoria direccionada por la suma de un registro base más un registro índice.
- Este modo es usado para el acceso a tablas.
- Ej. MOV AX,[BX+DI]
- En muchos casos, el registro base retiene la dirección de inicio de un arreglo de memoria,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

8

Ejemplo Direccionamiento Base más Índice

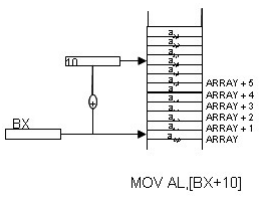
- Ej. Si se desea direccionar los elementos en un arreglo de datos localizados en el segmento de datos en la localidad ARRAY. Se requiere cargar BX con la dirección ARRAY y DI con el número del elemento del arreglo que se desea acceder.
- MOV BX, OFFSET ARRAY
- MOV DI,3
- MOV AL,[BX+DI]



9

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



10

Direccionamiento Relativo Base más Índice (1)

- Transfiere un byte o palabra entre un registro y la localidad de memoria direccionada por un registro base más un registro índice más un desplazamiento.

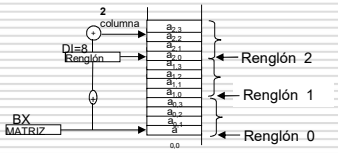
- Ej. MOV AX, [BX+SI+100H]
- MOV AX, ARRAY[BX+DI]
- MOV LIST[BP+DI],CL

11

Direccionamiento Relativo Base más Índice (2)

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

MOV BX, OFFSET MATRIZ
MOV DI,8
MOV AL,[BX+DI+2]



MOV AL,[BX+DI+2]

Selecciona elemento $a_{2,2}$

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

12

Direccionamiento de string

- En computación un string (cadena) es una secuencia de bytes o palabras almacenadas en memoria. Una tabla de datos es un ejemplo de string. Debido a su importancia el 8086 tiene algunas instrucciones diseñadas específicamente para manejar strings (cadenas) de caracteres.
- Estas instrucciones tienen un modo de direccionamiento especial y usan a DS:SI para apuntar al string fuente y a ES:DI para apuntar al string destino.
- MOVSB mueve el byte del dato fuente a la localidad destino. SI y DI se incrementan o decrementan automáticamente dependiendo del valor de la bandera D.

13

Instrucciones de cadena

		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	

14

Tabla de Modos de Direccionamiento

Direccionamiento	Cod. Ob.	Mnemónico	Segmento	Operación simbólica
Inmediato	B8 00 10	MOV AX,1000H	Código	AH←10H; AL ←00
Registro	8B D1	MOV DX,CX	Dentro del CPU	DX←CX
Directo	8A 00 10	MOV AH,[MEMBDS]	Datos	AH←[1000H]
Indirecto a registro	8B 04 FF 25 FE 46 00 FF 0F	MOV AX, [SI] JMP [DI] INC BYTE PTR[BP] DEC WORD PTR[BX]	Datos Datos Stack Datos	AL←[SI]; AH←[SI+1] IP←[DI+1:DI] [BP]←[BP] + 1 [BX+1:BX]←[BX+1:BX]-1
Indexado	8B 44 06 FF 65 06	MOV AX,[SI+6] JMP [DI+6]	Datos Datos	AL←[SI+6]; AH←[SI+7] IP ← [DI+7:DI+6]
Basado	8B 46 02 FF 67 02	MOV AX,[BP+2] JMP [BX+2]c	Stack Datos	AL←[BP+2]; AH←[BP+3] IP←[BX+3:BX+2]
Base mas índice	8B 00 FF 21 FE 02 FF 0B	MOV AX,[BX+SI] JMP [BX+DI] INC BYTE PTR[BP+SI] DEC WORD PTR[BP+DI]	Datos Datos Stack Stack	AL←[BX+SI]; AH←[BX+SI+1] IP←[BX+DI+1:BX+DI] [BP+SI]←[BP+SI]+1 [BP+DI+1:BP+DI]←[BP+DI+1:BP+DI]-1
Relativo base mas índice	8B 40 05 FF 61 05 FE 42 05 FF 4B 05	MOV AX,[BX+SI+5] JMP [BX+DI+5] INC BYTE PTR[BP+SI+5] DEC WORD PTR[BP+DI+5]	Datos Datos Stack Stack	AL←[BX+SI+5]; AH←[BX+SI+6] IP←[BX+DI+6:BX+DI+5] [BP+SI+5]←[BP+SI+5]+1 [BP+DI+6:BP+DI+5]←[BP+DI+6:BP+DI+5]-1
String	A4	MOVSB	Extra, Datos	ES:[DI]←DS:[SI] Si DF=0, entonces SI←SI+1; DI←DI+1 Si DF=1, entonces SI←SI-1; DI←DI-1

15

Codificación en Lenguaje Máquina

- Para convertir un programa en lenguaje ensamblador a código máquina, se debe convertir cada instrucción en lenguaje ensamblador a su instrucción equivalente en código máquina.
- El código máquina especifica qué 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.

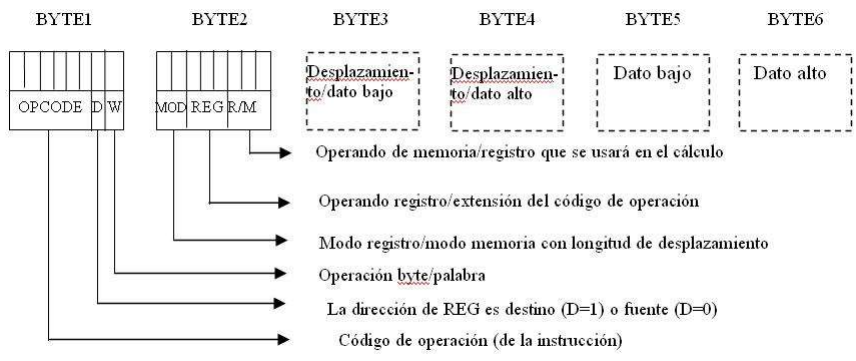
16

Codificación en Lenguaje Máquina

- El código máquina de las instrucciones del 8086 varía de 1 a 6 bytes. Las instrucciones de un byte generalmente especifican operaciones simples con un registro o un bit de bandera.
- El código máquina para las instrucciones puede obtenerse del siguiente formato general.

17

Codificación en Lenguaje Máquina Formato de una Instrucción



18

Codificación en Lenguaje Máquina
Formato de una Instrucción

El primer byte contiene la siguiente información:

- El campo opcode (código de operación) especifica la operación que será realizada.
- El bit D indica la dirección del campo REG; D=0 si REG es fuente, D=1 si REG es destino.
- El bit W indica el tamaño de la operación; W=0 tamaño byte, W=1 tamaño palabra.
- El bit V se utiliza en instrucciones de rotación o desplazamiento para indicar el número de cuenta; v=0 para cuenta=1 y v=1 para usar cuenta en CL.
- X indica que no importa el valor del bit

19

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.

20

Codificación en Lenguaje Máquina
Formato de una Instrucción

\ MOD R/M	00	01	10	11	
000	[BX+SI]	[BX+SI+D8]	[BX+SI+D16]	AL	AX
001	[BX+DI]	[BX+DI+D8]	[BX+DI+D16]	CL	CX
010	[BP+SI]	[BP+SI+D8]	[BP+SI+D16]	DL	DX
011	[BP+DI]	[BP+DI+D8]	[BP+DI+D16]	BL	BX
100	[SI]	[SI+D8]	[SI+D16]	AH	SP
101	[DI]	[DI+D8]	[DI+D16]	CH	BP
110	[Num16 bits]	[BP+D8]	[BP+D16]	DH	SI
111	[BX]	[BX+D8]	[BX+D16]	BH	DI

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

21

- *Ejemplos de formatos de Instrucciones del 8086*

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	mod0reg r/m		
Segment Register to Register/Memory	10001100	mod0reg r/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 reg r/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 reg r/m		
LDS – Load Pointer to DS	11000101	mod reg r/m		
LES – Load Pointer to ES	11000100	mod reg r/m		
LAHF – Load AH with Flags	10011111			
SAHF – Store AH into Flags	10011110			
PUSHF – Push Flags	10011100			
POPF – Pop Flags	10011101			

22

- *Ejemplos de formatos de Instrucciones del 8086*

Table 2. Instruction Set Summary (Continued)

Mnemonic and Description	Instruction Code			
ARITHMETIC	76543210	76543210	76543210	76543210
ADD = Add:				
Reg./Memory with Register to Either	000000dw	mod reg r/m		
Immediate to Register/Memory	100000sw	mod000r/m	data	data if s:w = 01
Immediate to Accumulator	0000010w	data	data if w = 1	
ADC = Add with Carry:				
Reg./Memory with Register to Either	000100dw	mod reg r/m		
Immediate to Register/Memory	100000sw	mod010r/m	data	data if s:w = 01
Immediate to Accumulator	0001010w	data	data if w = 1	
INC = Increment:				
Register/Memory	1111111w	mod000r/m		
Register	01000reg			
AAA = ASCII Adjust for Add	00110111			
BAA = Decimal Adjust for Add	00100111			
SUB = Subtract:				
Reg./Memory and Register to Either	001010dw	mod reg r/m		
Immediate from Register/Memory	100000sw	mod101r/m	data	data if s w = 01
Immediate from Accumulator	0010110w	data	data if w = 1	
SSB = Subtract with Borrow				
Reg./Memory and Register to Either	000110dw	mod reg r/m		
Immediate from Register/Memory	100000sw	mod011r/m	data	data if s w = 01
Immediate from Accumulator	0001111w	data	data if w = 1	
DEC = Decrement:				
Register/memory	1111111w	mod001r/m		
Register	01001reg			
NEG = Change sign	1111011w	mod011r/m		
CMP = Compare:				
Register/Memory and Register	001110dw	mod reg r/m		
Immediate with Register/Memory	100000sw	mod111r/m	data	data if s w = 01
Immediate with Accumulator	0011110w	data	data if w = 1	
AAS = ASCII Adjust for Subtract	00111111			
DAS = Decimal Adjust for Subtract	00101111			
MUL = Multiply (Unsigned)	1111011w	mod100r/m		
IMUL = Integer Multiply (Signed)	1111011w	mod101r/m		
AAM = ASCII Adjust for Multiply	11010100	00001010		
DIV = Divide (Unsigned)	1111011w	mod110r/m		
IDIV = Integer Divide (Signed)	1111011w	mod111r/m		
AAD = ASCII Adjust for Divide	11010101	00001010		
CBW = Convert Byte to Word	10011000			
CWD = Convert Word to Double Word	10011001			

23

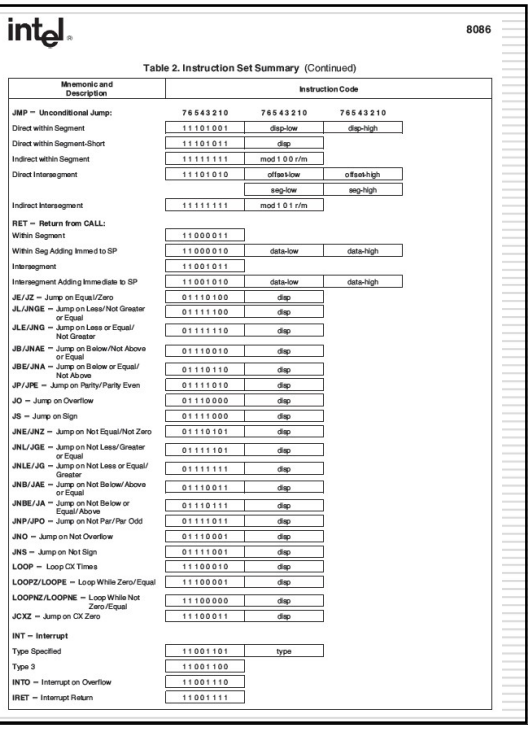
- *Ejemplos de formatos de Instrucciones del 8086*

Table 2. Instruction Set Summary (Continued)

Mnemonic and Description	Instruction Code			
LOGIC	76543210	76543210	76543210	76543210
NOT = Invert	1111011w	mod010r/m		
SHL/SAL = Shift Logical/Arithmetic Left	110100vw	mod100r/m		
SHR = Shift Logical Right	110100vw	mod101r/m		
SAR = Shift Arithmetic Right	110100vw	mod111r/m		
ROL = Rotate Left	110100vw	mod000r/m		
ROR = Rotate Right	110100vw	mod001r/m		
RCL = Rotate Through Carry Flag Left	110100vw	mod010r/m		
RCR = Rotate Through Carry Right	110100vw	mod011r/m		
AND = And:				
Reg./Memory and Register to Either	001000dw	mod reg r/m		
Immediate to Register/Memory	1000000w	mod100r/m	data	data if w = 1
Immediate to Accumulator	0010010w	data	data if w = 1	
TEST = And Function to Flags, No Result:				
Register/Memory and Register	1000010w	mod reg r/m		
Immediate Data and Register/Memory	1111011w	mod000r/m	data	data if w = 1
Immediate Data and Accumulator	1010100w	data	data if w = 1	
OR = Or:				
Reg./Memory and Register to Either	000010dw	mod reg r/m		
Immediate to Register/Memory	1000000w	mod001r/m	data	data if w = 1
Immediate to Accumulator	0000110w	data	data if w = 1	
XOR = Exclusive or:				
Reg./Memory and Register to Either	001100dw	mod reg r/m		
Immediate to Register/Memory	1000000w	mod110r/m	data	data if w = 1
Immediate to Accumulator	0011010w	data	data if w = 1	
STRING MANIPULATION				
REP = Repeat	1111001z			
MOVS = Move Byte/Word	1010010w			
CMPS = Compare Byte/Word	1010011w			
SCAS = Scan Byte/Word	1010111w			
LODS = Load Byte/Wd to AL/AX	1010110w			
STOS = Stor Byte/Wd from AL/A	1010101w			
CONTROL TRANSFER				
CALL = Call:				
Direct within Segment	11101000	disp-low	disp-high	
Indirect within Segment	11111111	mod010r/m		
Direct Intersegment	10011010	offset-low	offset-high	
		seg-low	seg-high	
Indirect Intersegment	11111111	mod011r/m		

24

- *Ejemplos de formatos de Instrucciones del 8086*



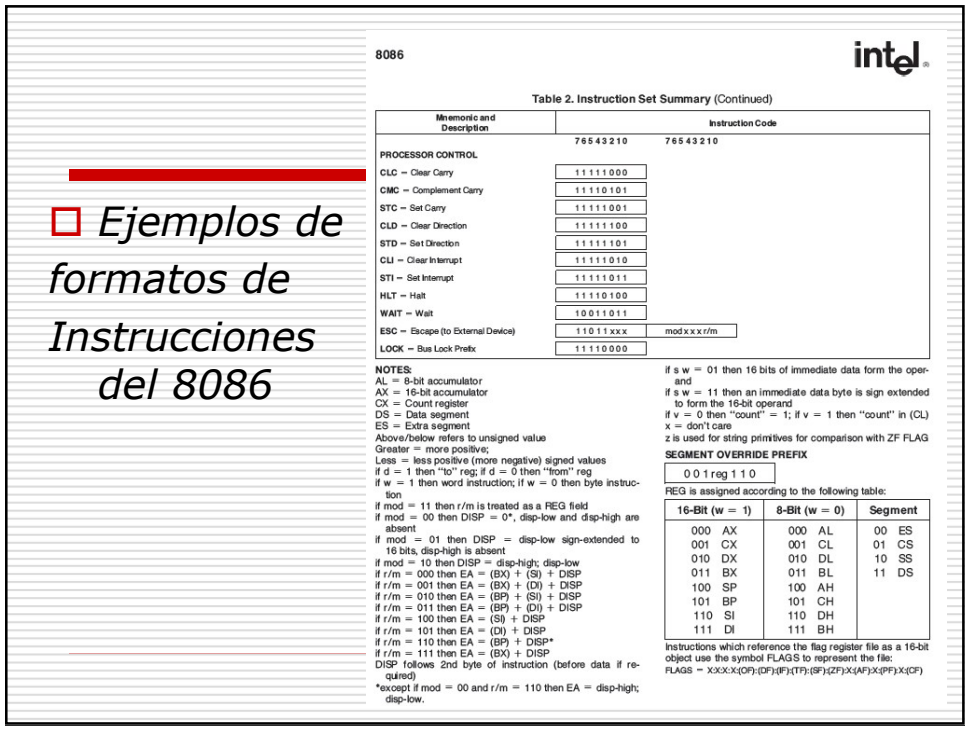
intel 8086

Table 2. Instruction Set Summary (Continued)

Mnemonic and Description	Instruction Code		
JMP – Unconditional Jump:	76543210	76543210	76543210
Direct within Segment	11101001	disp-low	disp-high
Direct within Segment-Short	11101011	disp	
Indirect within Segment	11111111	mod 100 r/m	
Direct Intersegment	11101010	offset-low	offset-high
		seg-low	seg-high
Indirect Intersegment	11111111	mod 101 r/m	
RET – Return from CALL:			
Within Segment	11000011		
Within Seg Adding Immed to SP	11000010	data-low	data-high
Intersegment	11001011		
Intersegment Adding Immediate to SP	11001010	data-low	data-high
JE/JZ – Jump on Equal/Zero	01110100	disp	
JL/JNGE – Jump on Less/Not Greater or Equal	01111100	disp	
JLE/JNG – Jump on Less or Equal/ Not Greater	01111110	disp	
JB/JNAE – Jump on Below/Not Above or Equal	01110010	disp	
JBE/JNA – Jump on Below or Equal/ Not Above	01110110	disp	
JP/JPE – Jump on Parity/Parity Even	01111010	disp	
JO – Jump on Overflow	01110000	disp	
JS – Jump on Sign	01111000	disp	
JNE/JNZ – Jump on Not Equal/Not Zero	01110101	disp	
JNL/JGE – Jump on Not Less/Greater or Equal	01111101	disp	
JNLE/JG – Jump on Not Less or Equal/ Greater	01111111	disp	
JNB/JAE – Jump on Not Below/Above or Equal	01110011	disp	
JNBE/JA – Jump on Not Below or Equal/Above	01110111	disp	
JNP/JPO – Jump on Not Par/Par Odd	01111011	disp	
JNO – Jump on Not Overflow	01110001	disp	
JNS – Jump on Not Sign	01111001	disp	
LOOP – Loop CX Times	11100010	disp	
LOOPZ/LOOPE – Loop While Zero/Equal	11100001	disp	
LOOPNZ/LOOPNE – Loop While Not Zero/Equal	11100000	disp	
JCXZ – Jump on CX Zero	11100011	disp	
INT – Interrupt			
Type Specified	11001101	type	
Type 3	11001100		
INTO – Interrupt on Overflow	11001110		
IRET – Interrupt Return	11001111		

25

- *Ejemplos de formatos de Instrucciones del 8086*



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	mod x x x r/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

001reg110

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)

26