UNIVERSIDAD AUTONOMA METROPOLITANA
Casa abierta al tiempo Azcapotzalco

DEPARTAMENTO DE ELECTRONICA

Microprocesadores

1121060

Tema 4

Conjunto de Instrucciones.

Microprocesadores 1121060

Tema 4.

Conjunto de Instrucciones.

1. Lenguaje Ensamblador
2. Instrucciones de Transferencia de datos
3. Operaciones Aritmético Lógicas
4. Instrucciones de Salto
5. Instrucciones de Manejo de Pila
6. Interrupciones
7. Ensamblado de Instrucciones

Tema 4. Conjunto de Instrucciones

Instrucción de Transferencia MOV

Mnemónico	Código Objeto	Mnemónico	Segmento de memoria	Operación simbólica
MOV dest,fuente	8B C3	MOV AX,BX	Dentro del CPU	AX←BX
	8A E3	MOV AH,BL	Dentro del CPU	AH←BL
	A1 00 10	MOV AX,MEMWDS	Datos	AL←[1000H], AH←[1001H]
	A0 02 10	MOV AL,MEMBDS	Datos	AL←[1002H]
	89 1E 00 10	MOV MEMWDS,BX	Datos	[1000H]←BL, [1001H]←BH
	88 1E 02 10	MOV MEMBDS,BL	Datos	[1002H]←BL
	C7 06 00 10 34 12	MOV MEMWDS,1234H	Datos	[1000H]←34H, [1001H]←12H
	C6 06 02 10 34	MOV MEMBDE,34H	Datos	[1002H]←34H
	B0 10	MOV AL,10H	Código	AL←10H
	B8 00 10	MOV AX,1000H	Código	Al←00H, AH←10H
	8E D8	MOV DS,AX	Dentro del CPU	DS←AX
	8C C2	MOV DX,ES	Dentro del CPU	DX←ES
	8E 06 00 10	MOV ES,MEMWDS	Datos	ES←[1001H:1000H]
	8C 0E 00 10	MOV MEMWDS,CS	Datos	[1001H:1000H]←CS

Tema 4. Conjunto de Instrucciones

Instrucciones de Transferencia

Mnemónico	Código Objeto	Mnemónico	Segmento de memoria	Operación simbólica
XCHG dest,fuente	93 86 C7 87 14	XCHG AX,BX XCHG AL,BH XCHG [SI],DX	Dentro del CPU Dentro del CPU Datos	AX ⇄ BX AL ⇄ BH [SI] ⇄ DL; [SI+1] ⇄ DH
LAHF	9F	LAHF	Dentro del CPU	AH ← Banderas
SAHF	9E	SAHF	Dentro del CPU	Banderas ← AH
IN acumulador,puerto	E4 26 E5 26 EC ED	IN AL,26H IN AX,26H IN AL,DX IN AX,DX	[b] [b] [b] [b]	AL← puerto 26H AL←puerto 26H, AH←puerto 27H AL←puerto DX AL← puerto DX; AH←puerto DX+1
OUT puerto,acumulador	E6 26 E7 26 EE EF	OUT 26H,AL OUT 26H,AX OUT DX,AL OUT DX,AX	[b] [b] [b] [b]	puerto 26H←AL puerto 26H←AL; puerto 27H←AH puerto DX←AL puerto DX←AL; puerto DX+1 ←AH
LEA dest,fuente[c]	8D 1E 00 10	LEA BX,MEMBDS	Datos	BL←00; BH ←10H
LDS dest,fuente[d] LES dest,fuente[d]	C5 1C C4 1C	LDS BX,DWORD PTR[SI] LES BX,DWORD PTR[SI]	Datos Datos	BL ←[SI]; BH←[SI+1]; DS←[SI+3:SI+2] BL ←[SI]; BH←[SI+1]; ES←[SI+3:SI+2]
XLAT	D7	XLAT	Datos	AL←[BX+AL];

Tema 4. Conjunto de Instrucciones

Instrucciones de Cadena (String)

Mnemónico	Código Objeto	Mnemónico	Segmento	Operación simbólica
STOSB	AA	STOSB	Extra	ES:[DI]←AL Si DF=0, DI←DI+1 Si DF=1, DI←DI-1
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
LODSW	AD	LODSW	Datos	AL←DS:[SI]; AH←DS:[SI+1] Si DF=0, SI←SI+2 Si DF=1, SI←SI-2
MOVSB	A4	MOVSB	Datos, Extra	ES:[DI] ← DS:[SI] Si DF=0,DI←DI+1, SI←SI+1 Si DF=1, DI←DI-1, SI←SI-1
MOVSW	A5	MOVSW	Datos, Extra	ES:[DI+1 :DI] ← DS:[SI+1: SI] Si DF=0,DI←DI+2, SI←SI+2 Si DF=1, DI←DI-2, SI←SI-2

Tema 4. Conjunto de Instrucciones

Instrucciones de Cadena (String)

Mnemónico	Código Objeto	Mnemónico	Segmento	Operación simbólica
SCASB	AE	SCASB	Extra	AL – ES:[DI]; actualiza banderas Si DF=0, DI← DI + 1 Si DF=1, DI← DI - 1
SCASW	AF	SCASW	Extra	AX – ES:[DI+1:DI]; actualiza banderas Si DF=0, DI← DI + 2 Si DF=1, DI← DI - 2
CMPSB	A6	CMPSB	Extra, Datos	DS:[SI] - ES:[DI]; actualiza banderas Si DF=0 DI←DI+1, SI←SI+1 Si DF=1 DI←DI-1, SI←SI-1
CMPSW	A7	CMPSW	Extra, Datos	DS:[SI+1:SI] - ES:[DI+1:DI]; actualiza banderas Si DF=0 DI←DI+2, SI←SI+2 Si DF=1 DI←DI-2, SI←SI-2
Prefijo REP				
REP	F3 AA F3 AB F3 A4 F3 A5	REP STOSB REP STOSW REP MOVSB REP MOVSW	Extra Extra Extra, datos Extra, datos	STOSB; CX← CX – 1 Repite hasta que CX=0 STOSW; CX← CX – 1 Repite hasta que CX=0 MOVSB; CX← CX – 1Repite hasta que CX=0 MOVSW; CX←CX-1Repite hasta que CX=0

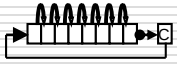
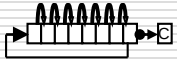
Tema 4. Conjunto de Instrucciones
Instrucciones de Cadena (String)

Mnemónico	Código Objeto	Mnemónico	Segmento	Operación simbólica
REP	F3 AA	REP STOSB	Extra	STOSB; CX← CX – 1 Repite hasta que CX=0
	F3 AB	REP STOSW	Extra	STOSW; CX← CX – 1 Repite hasta que CX=0
	F3 A4	REP MOVSB	Extra, datos	MOVSB; CX← CX – 1Repite hasta que CX=0
	F3 A5	REP MOVSW	Extra, datos	MOVSW; CX←CX-1Repite hasta que CX=0
REPE/REPZ	F3 AE	REPZ SCASB	Extra	SCASB; CX ← CX – 1 Repite si ZF = 1 y CX ≠ 0
	F3 AF	REPZ SCASW	Extra	Similar a la anterior excepto SCASW
	F3 A6	REPZ CMPSB	Extra, datos	Similar a la anterior excepto CMPSB
	F3 A7	REPZ CMPSW	Extra, datos	Similar a la anterior excepto CMPSW
REPNE/REPNZ	F2 AE	REPNE SCASB	Extra	SCASB; CX ← CX – 1 Repite si ZF = 0 y CX ≠ 0
	F2 AF	REPNE SCASW	Extra	Similar a la anterior excepto SCASW
	F2 A6	REPNE CMPSB	Extra, datos	Similar a la anterior excepto CMPSB
	F2 A7	REPNE CMPSW	Extra, datos	Similar a la anterior excepto CMPSW

Tema 4. Conjunto de Instrucciones

Instrucciones de Rotación y Corrimiento

A←shl A, {S}←0\(S)←

Mnemónico	Código Objeto	Mnemónico	Segmento de memoria	Operación simbólica
SAL/SHL[b] dest,contador	D1 E0 D3 E0	SAL AX,1 SAL AX,CL	Dentro del CPU Dentro del CPU	
SAR dest, contador	D0 F8 D2 F8	SAR AL,1 SAR AL,CL	Dentro del CPU Dentro del CPU	
SHR dest, contador	D1 2C D2 2C	SHR WORD PTR[SI],1 SHR BYTE PTR[SI],CL	Datos Datos	
RCL dest, contador	D1 D3 D3 D3	RCL BX,1 RCL BX,CL	Dentro del CPU Dentro del CPU	
RCR dest, contador	D0 DB D2 DB	RCR BL,1 RCR BL,CL	Dentro del CPU Dentro del CPU	
ROL dest, contador	D1 04 D2 04	ROL WORD PTR[SI],1 ROL BYTE PTR[SI],CL	Datos Datos	
ROR dest, contador	D1 0E 00 10 D2 0E 04 10	ROR MEMWDS,1 ROR MEMBDS,CL	Datos Datos	

Tema 4. Conjunto de Instrucciones
Instrucciones Lógicas

Mnemónico [BX]←	**Código Objeto**	**Mnemónico[a]**	**Segmento de memoria**	**Operación simbólica**
NOT dest	F7 D3 F6 14	NOT BX NOT BYTE PTR[SI]	Dentro del CPU Datos	BX←BX' [SI] ← [SI]'
AND dest, fuente	23 CA 22 3C 25 00 80	AND CX, DX AND BH,BYTE PTR [SI] AND AX, 8000H	Dentro del CPU Datos Código	CX ← CX ^ DX BH ← BH ^ [SI] AX ← AX ^ 8000H
OR dest, fuente	0B CA 0A 3C 0D 00 80	OR CX,DX OR BH, BYTE PTR[SI] OR AX,8000H	Dentro del CPU Datos Código	CX ← CX + DX BH ← BH + [SI] AX ← AX + 8000H
XOR dest, fuente	33 CA 32 3C 35 00 80	XOR CX,DX XOR BH, BYTE PTR[SI] XOR AX,8000H	Dentro del CPU Datos Código	CX ← CX ⊕ DX BH ← BH ⊕ [SI] AX ← AX ⊕ 8000H
TEST dest, fuente	85 D1 84 3C A9 00 80	TEST CX,DX TEST BH, BYTE PTR[SI] TEST AX,8000H	Dentro del CPU Datos Código	CX ^ DX; actualiza banderas BH ^ [SI]; actualiza banderas AX ^ 8000H; actualiza banderas

Tema 4. Conjunto de Instrucciones
Instrucciones Aritméticas

Mnemónico	Código Objeto	Mnemónico[a]	Segmento de memoria	Operación simbólica

ADD dest, fuente	03 F2
	00 2F
	81 C7 00 80
	81 06 00 10 00 80
		ADD SI, DX
		ADD BYTE PTR[BX],CH
		ADD DI,8000H
		ADD MEMWDS,8000H[a]
			Dentro del CPU
			Datos
			Dentro del CPU
			Datos
				SI ← SI + DX
				[BX]←[BX] + CH
				DI← DI + 8000H
				[1001H:1000H]← [1001H:1000H]+8000H

ADC dest, fuente	13 F2 10 2F 81 D7 00 80 81 16 00 10 00 80	ADC SI, DX ADC BYTE PTR [BX],CH ADC DI, 8000H ADC MEMWDS,8000H[a]	Dentro del CPU Datos Dentro del CPU Datos	SI ← SI + DX + CF [BX]←[BX]+CH + CF DI← DI + 8000H + CF [1001H:1000H]← [1001H:1000H]+8000H+CF

SUB dest, fuente	2B F2
	28 2F
	81 EF 00 80
	81 2E 00 10 00 80
		SUB SI, DX
		SUB BYTE PTR[BX],CH
		SUB DI,8000H
		SUB MEMWDS,8000H[a]
			Dentro del CPU
			Datos
			Dentro del CPU
			Datos
				SI ← SI - DX
				[BX]←[BX] - CH
				DI← DI - 8000H
				[1001H:1000H]← [1001H:1000H]-8000H

SBB dest, fuente	1B F2 18 2F 81 DF 00 80 81 1E 00 10 00 80	SBB SI, DX SBB BYTE PTR [BX],CH SBB DI, 8000H SBB MEMWDS,8000H	Dentro del CPU Datos Dentro del CPU Datos	SI ← SI - DX - CF [BX]←[BX]- CH - CF DI← DI - 8000H - CF [1001H:1000H]← [1001H:1000H]-8000H-CF

Tema 4. Conjunto de Instrucciones

Instrucciones Aritméticas

		Ejemplo codificado		
Mnemónico	**Código Objeto**	**Mnemónico[a]**	**Segmento de memoria**	**Operación simbólica**
INC dest[b]	FE C3 FF 05 FE 06 04 10	INC BL INC WORD PTR[DI] INC MEMBDS[c]	Dentro del CPU Datos Datos	BL←BL+1 [DI+1:DI]←[DI+1:DI]+1 [1004H]← [1004H]+1
DEC dest[b]	FE CB FF 0D FE 0E 04 10	DEC BL DEC WORD PTR[DI] DEC MEMBDS[c]	Dentro del CPU Datos Datos	BL←BL-1 [DI+1:DI]←[DI+1:DI]-1 [1004H]← [1004H]-1
NEG dest[b]	F6 DB F7 1D F6 1E 04 10	NEG BL NEG WORD PTR[DI] NEG MEMBDS[c]	Dentro del CPU Datos Datos	BL←0 - BL [DI+1:DI]← 0 - [DI+1:DI] [1004H]← 0 - [1004H]

CMP dest,
fuente
	3A C4
	39 0D
	81 3E 00 10 00
	80
	81 FF 00 80
		CMP AL,AH
		CMP [DI],CX
		CMP MEMWDS,800H[a]
		CMP DI,8000H
			Dentro del CPU
			Datos
			Datos
			Dentro del CPU
				AL – AH; actualiza banderas
				[DI+1:DI]-CX; actualiza banderas
				[1001H:1000H]-8000H;actualiza
				banderas
				DI-8000H;actualiza banderas

Tema 4. Conjunto de Instrucciones

Instrucciones Aritméticas

Mnemónico	Código Objeto	Mnemónico	Segmento de memoria	Operación simbólica
MUL fuente	F6 E3 F7 E1 F6 27 F7 26 00 10	MUL BL MUL CX MUL BYTE PTR[BX] MUL MEMWDS	Dentro del CPU Dentro del CPU Datos Datos	AX ← AL * BL DX:AX←AX * CX AX← AL * [BX] DX:AX←AX* [1001H:1000H]
IMUL fuente	F6 EB F7 E9 F6 2F F7 2E 00 10	IMUL BL IMUL CX IMUL BYTE PTR[BX] IMUL MEMWDS	Dentro del CPU Dentro del CPU Datos Datos	AX ← AL * BL (con signo) DX:AX←AX * CX (con signo) AX← AL * [BX] (con signo) DX:AX←AX* [1001H:1000H] (con signo)
DIV	F6 F3 F7 F1 F6 37 F7 36 00 10	DIV BL DIV CX DIV BYTE PTR[BX] DIV MEMWDS	Dentro del CPU Dentro del CPU Datos Datos	AX ← AX / BL DX:AX←DXAX / CX AX← AX / [BX] DX:AX←DXAX / [1001H:1000H]
IDIV	F6 FB F7 F9 F6 3F F7 3E 00 10	IDIV BL IDIV CX IDIV BYTE PTR[BX] IDIV MEMWDS	Dentro del CPU Dentro del CPU Datos Datos	AX ← AX / BL (con signo) DX:AX←DXAX / CX (con signo) AX← AX / [BX] (con signo) DX:AX←DXAX /[1001H:1000H] (con signo)

Tema 4. Conjunto de Instrucciones
Instrucciones de Ajuste

Mnemónico	Código Objeto	Mnemó-nico	Segmento de memoria	Operación simbólica
DAA	27	DAA	Dentro del CPU	Si AL ∧ 0F > 9 o AF = 1, entonces AL←AL+ 6; AF ← 1 Si AL > 9F o CF = 1, entonces AL ← AL + 60H; CF ← 1
DAS	2F	DAS	Dentro del CPU	Si AL ∧ 0F > 9 o AF = 1, entonces AL←AL-6; AF ← 1 Si AL > 9F o CF = 1, entonces AL ← AL - 60H; CF ← 1
AAA	37	AAA	Dentro del CPU	Si AL ∧ 0F > 9 o AF = 1, entonces AL←AL+ 6; AH ← AH + 1; AF← 1; CF ←AF; AL ← AL ∧ 0F
AAS	3F	AAS	Dentro del CPU	Si AL ∧ 0F > 9 o AF = 1, entonces AL←AL- 6; AH ← AH - 1; AF← 1; CF ←AF; AL ← AL ∧ 0F
AAM	D4 0A	AAM	Dentro del CPU	AH ← AL / 0AH AL ← Residuo
AAD	D5 0A	AAD	Dentro del CPU	AL ← (AH * 0AH) + AL AH ← 0
CBW	98	CBW	Dentro del CPU	Si AL < 80H, entonces AH ← 0 Si AL > 7F, entonces AH ← FFH
CWD	99	CWD	Dentro del CPU	Si AX < 8000H, entonces DX← 0 Si AX > 7FFFFH, entonces DX ← FFFFH

Tema 4. Conjunto de Instrucciones
Instrucciones de Ajuste.

- DAA (decimal adjust after addition)

```
mov al,35h
add al,48h;  AL=7DH
daa          ;  AL=83H
```

- DAS (decimal adjust after substraction)

```
mov bl,48h
mov al,85h   ;
sub al, bl    ;AL=3DH
das           ;AL=37H
```

Tema 4. Conjunto de Instrucciones
Instrucciones de Ajuste.

- **AAA(ASCII adjust after addition)**

```
mov ah,0
mov al,'8'    ; AX=0038H
add al,`2'    ;AX=006AH
aaa           ;AX=0100H
or ax,3030h ; AX=3130H='10'
```

- **AAS (ASCII adjust after substraction)** Ajusta el resultado binario obtenido de una instrucción SUB o SBB. Provoca que el resultado en AL sea consistente con una representación ascii. Por ejemplo, el siguiente fragmento de código resta el ascii 9 del ascii 8, en donde después de la instrucción SUB, AX es igual a 00FFh (-1) y la instrucción AAS convierte AX a FF09, el complemento a 10 de -1.

```
Datos segment
        val1 db '8'
        val2 db '9'
Datos ends
Codigo segment 'code'
            :
        mov ah,0
        mov al,val1      ;AX=0038H
        sub  al,val2     ;AX=00FFH
        aas              ;AX=FF09H
```

Tema 4. Conjunto de Instrucciones
Instrucciones de Ajuste.

AAM (ASCII adjust after multiplication)

```
Datos segment
    ascVal db 05h,06h
Datos ends
Codigo segment
        :
    mov bl,ascVal ;      primer operando
    mov al,ascVal+1 ; segundo operando
    mul bl              ;AX=001EH
    aam                 ;AX=0300H
    or ax,3030h ;AX=3330H='30'
```

AAD (ASCII adjust before division)

```
Datos segment
    cociente  db ?
    residuo   db ?
Datos ends
Codigo segment
        :
    mov ax,0307H  ;dividendo (BCD )
    aad           ;AX=0025H (AH*10+AL)
    mov bl,05     ;divisor
    div bl        ;AX=0207H
    mov cociente,al
    mov residuo,ah
```

Tema 4. Conjunto de Instrucciones
Instrucciones de Salto Incondicional

Mnemónico	Código Objeto	Mnemónico	Segmento	Operación simbólica
JMP near etiqueta	E9 -- -- FF 26 00 10 FF 27 FF E0	JMP MEMN JMP [MEMWDS] JMP [BX] JMP AX	Código Datos Datos Dentro del CPU	IP← MEMN IP←[MEMWDS+1:MEMWDS] IP←[BX+1:BX] IP←AX
JMP SHORT etiqueta	EB --	JMP SHORT MEMS	Código	IP←MEMS
JMP far etiqueta	EA 03 00 D3 9E FF 2E 05 10 FF 2F	JMP FAR PTR MEMF JMP [MEMWWDS] JMP DWORD PTR[BX]	Código Datos Datos	IP←0003H; CS←9ED3H IP←[1006H:1005H]; CS←[1008H:1007H] IP←[BX+1:BX]; CS←[BX+3:BX+2]
Jcond^{h} short etiqueta	73 --	JNC MEMS	Código	Si CF=0, entonces IP←MEMS
JCXZ short etiqueta	E3 --	JCXZ MEMS	Código	Si CX=0, entonces IP←MEMS

Tema 4. Conjunto de Instrucciones
Instrucciones de Salto Condicional

Saltos Condicionales para números con signo y sin signo		
Mnemónico	**Significado**	**Condición**
JE / JZ	Salta si es igual o salta si es cero	ZF = 1
JNE / JNZ	Salta si no es igual o salta si no es cero	ZF = 0
JC	Salta si hay acarreo	CF = 1
JNC	Salta si no hay acarreo	CF = 0
JS	Salta si el signo es negativo	SF = 1
JNS	Salta si el signo es positivo	SF = 0
JO	Salta si hay desbordamiento (sobreflujo)	OF = 1
JNO	Salta si no hay desbordamiento (sobreflujo)	OF = 0
JP / JPE	Salta si hay paridad o salta si la paridad es par	PF = 1
JNP / JPO	Salta si no hay paridad o salta si la paridad es impar	PF = 0

Tema 4. Conjunto de Instrucciones
Instrucciones de Salto Condicional

Saltos Condicionales para números sin signo		
Mnemónico	**Significado**	**Condición**
JA / JNBE	Salta si es mayor o salta si no es menor igual	CF = 0 y ZF = 0
JAE / JNB	Salta si es mayor o igual o salta si no es menor	CF = 0
JB / JNAE	Salta si es menor o salta si no es mayor igual	CF = 1
JBE / JNA	Salta si es menor o igual o salta si no es mayor	CF = 1 o Z = 1
Saltos Condicionales para números con signo		
Mnemónico	**Significado**	**Condición**
JG / JNLE	Salta se es mayor o salta si no es menor igual	ZF = 0 y SF = OF
JGE / JNL	Salta si es mayor o igual o salta si no es menor	SF = OF
JL / JNGE	Salta si es menor o salta si no es mayor igual	(SF $\oplus$ OF)=1
JLE / JNG	Salta si es menor o igual o salta si no es mayor	((SF $\oplus$ OF) + ZF) = 1

Tema 4. Conjunto de Instrucciones
Instrucción de Ciclo

Mnemónico	**Código Objeto**	**Mnemónico[a]**	**Segmento**	**Operación simbólica**
LOOP etiqueta corta	E2 --[a]	LOOP MEMS[b]	Código	CX ← CX –1 Si CX ≠ 0, entonces IP ← MEMS
LOOPE / LOOPZ etiqueta corta	E1 --[a]	LOOPZ MEMS[b]	Código	CX ← CX –1 Si (CX≠0)^(ZF=1), entonces IP ← MEMS
LOOPNE / LOOPNZ etiqueta corta	E0 --[a]	LOOPNZ MEMS[b]	Código	CX ← CX –1 Si (CX≠0)^(ZF=0), entonces IP ← MEMS

Tema 4. Conjunto de Instrucciones
Instrucciones de Manejo de Pila

Mnemónico general Op-code Operando	Código Objeto	Mnemónico	Segmento	Operación simbólica
CALL etiqueta cercana	E8 -- --	CALL MEMN	Código	SP← SP –2; [SP+1:SP]← IP; IP← MEMN
	FF 16 00 10	CALL [MEMWDS]	Datos	SP← SP –2; [SP+1:SP]← IP; IP←[1001H:1000H]
	FF15	CALL [DI]	Datos	SP← SP –2; [SP+1:SP]← IP; IP←[DI+1:DI]
	FF D7	CALL DI	Dentro del CPU	SP← SP –2; [SP+1:SP]← IP; IP←DI

Tema 4. Conjunto de Instrucciones
Instrucciones de Manejo de Pila

Mnemónico	Código Objeto	Mnemónico	Segmento	Operación simbólica
CALL etiqueta lejana	9A 00 10 D3 09	CALL FAR PTR MEMF	Código	SP← SP –2; [SP+1:SP]← CS; CS←09D3H; SP← SP –2; [SP+1:SP]← IP; IP← 1000H
	FF 1E 00 10	CALL [MEMWWDS]	Datos	Similar al anterior excepto: CS ←[1003H:1002H]; IP ← [1001H:1000H]
	FF 1D	CALL DWORD PTR[DI]	Datos	Similar al anterior excepto: CS←[DI+3:DI+2]; IP←[DI+1:DI]

Tema 4. Conjunto de Instrucciones
Instrucciones de Manejo de Pila

Mnemónico	Código	Mnemónico	Segmento	Operación simbólica
RET n (cercano)[f]	C3	RET	Pila	IP ← [SP+1:SP]; SP← SP +2;
	C2 08 00	RET 8	Pila	IP←[SP+1:SP]; SP← SP + 2 + 8
RET n (lejano)[f]	CB	RET	Pila	IP←[SP+1:SP]; SP← SP + 2; CS ←[SP+1:SP]; SP←SP+2
	CA 08 00	RET 8	Pila	IP ← [SP+1:SP]; SP←SP + 2; CS←[SP+1:SP]; SP←SP + 2 + 8

Tema 4. Conjunto de Instrucciones
Instrucciones de Manejo de Pila

Mnemónico general	Código Objeto	Mnemónico	Segmento	Operación simbólica
PUSH fuente	51	PUSH CX	Pila	SP←SP-2; [SP+1]←CH; [SP]←CL
	1E	PUSH DS	Pila	SP←SP-2; [SP+1:SP]←DS
	FF 75 02	PUSH[DI+2]	Pila, datos	SP←SP-2; [SP+1]←[DI+3]; [SP]←[DI+2]
POP dest	59	POP CX	Pila	CL←[SP]; CH←[SP+1]; SP←SP+2
	1F	POP DS	Pila	DS←[SP+1:SP]; SP←SP+2
	8F 45 02	POP[DI+2]	Datos, pila	[DI+3]←[SP+1]; [DI+2]←[SP]; SP←SP+2
PUSHF	9C	PUSHF	Pila	SP←SP-2; [SP+1:SP]← Banderas
POPF	9D	POPF	Pila	Banderas←[SP+1:SP]; SP←SP+2

Tema 4. Conjunto de Instrucciones
Instrucciones de Manejo de Pila

Mnemónico general	Código Objeto	Mnemónico	Segmento de memoria	Operación simbólica
INT tipo	CD 23	INT 23H	Pila e interrupción saltando a la tabla de vectores (00000 a 003FFH)	SP← SP –2; [SP+1:SP]← Banderas IF←0; TF←0; SP← SP –2; [SP+1:SP]← CS; CS←[0008FH:0008EH]; SP← SP –2; [SP+1:SP]← IP; IP←[0008DH:0008CH]
INTO	CE	INTO	Pila e interrupción saltando a la tabla de vectores (00000 a 003FFH)	Si OF=1, entonces SP← SP –2 [SP+1:SP]← Banderas; IF=0; TF=0; SP← SP –2; [SP+1:SP]← CS; CS←[00013H:00012H]; SP← SP –2; [SP+1:SP]←IP; IP ←[00011H:00010H]
IRET	CF	IRET	Pila	IP ←[SP+1:SP]; SP ← SP + 2; CS ←[SP+1:SP]; SP ← SP + 2; Banderas←[SP+1:SP]; SP ← SP+ 2

Tema 4. Conjunto de Instrucciones
Instrucciones de Control

Mnemónico general	Código objeto	Mnemónico	Segmento de memoria	Operación simbólica
STC	F9	STC	Dentro del CPU	CF ← 1
CLC	F8	CLC	Dentro del CPU	CF ← 0
CMC	F5	CMC	Dentro del CPU	CF ← $\overline{CF}$
STD	FD	STD	Dentro del CPU	DF ←1
CLD	FC	CLD	Dentro del CPU	DF ← 0
STI	FB	STI	Dentro del CPU	IF ← 1
CLI	FA	CLI	Dentro del CPU	IF ← 0
HLT	F4	HLT	Dentro del CPU	Ninguna
WAIT	9B	WAIT	Dentro del CPU	Ninguna
LOCK Instrucción	F0 A1 00 10	LOCK MOV AX,MEMWDS	Datos	Ninguna
NOP	90	NOP	Dentro del CPU	Ninguna
ESC número,fuente	DE 0E 00 10	ESC 31H,MEMWDS	Datos	Bus de datos ←[MEMWDS]

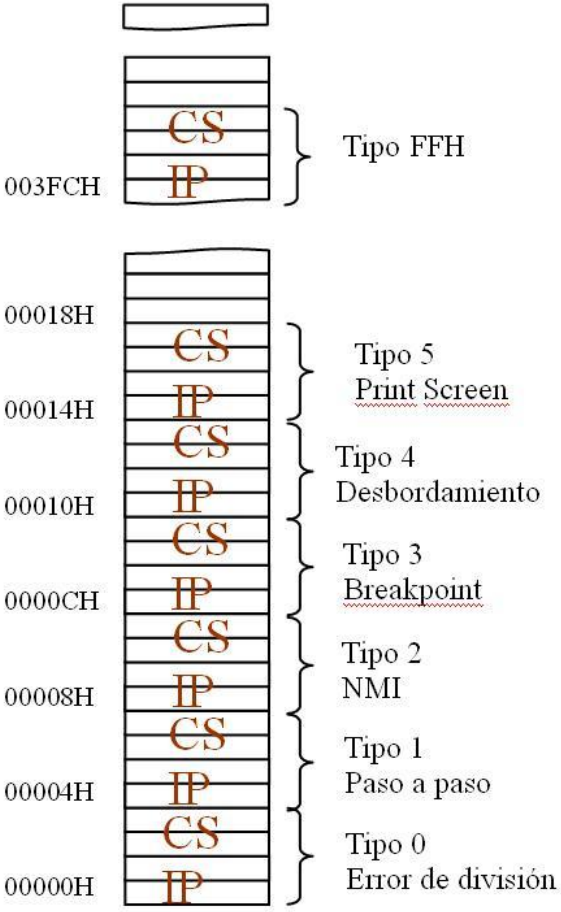
Tema 4. Conjunto de Instrucciones
Instrucciones de Control

Mnemónico general		**Segmento de memoria**	**Operación**
WAIT	WAIT	Ninguna	Pone en estado de espera ($\overline{TEST}$ =1) al procesador hasta que el coprocesador termina su ejecución.
LOCK Instrucción	LOCK MOV AX,MEMWDS	Ninguna	Bloquea el bus. Evita que el 8087 u otros coprocesadores cambien datos al mismo tiempo que el procesador; coloca la línea de salida $\overline{LOCK}$= 0. Lock es un prefijo de un byte que se utiliza para prevenir a los coprocesadores de acceder al bus hasta que Se complete la instrucción siguiente a lock.
NOP	NOP	Ninguna	No operación.
ESC número,fuente	ESC 31H,MEMWDS	Bus de datos ←[MEMWDS]	Coloca el contenido del operando fuente de memoria en el bus de datos y ejecuta un NOP. El primer operando identifica una instrucción escape particular para que sea Ejecuta por el coprocesador.

Tema 4. Conjunto de Instrucciones
Interrupciones

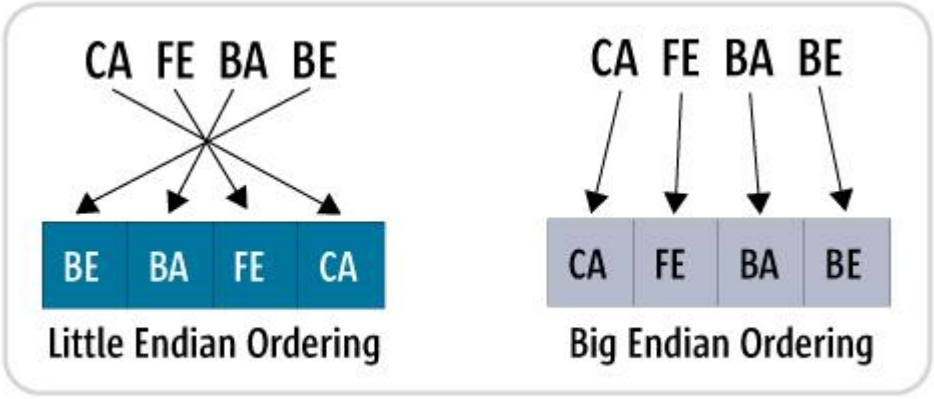
Tabla de Vectores de Interrupción

- La Tabla de vectores de interrupción se ubica en los primeros 1024 bytes de memoria (00000H- 003FFH) contiene 256 vectores de interrupción diferentes de 4 bytes.
- Un vector de interrupción contiene la dirección (segmento de código y desplazamiento IP) del procedimiento de interrupción.
  - Los dos bytes menos significativos contienen el IP y los dos bytes más significativos contienen el CS.



Tema 4. Conjunto de Instrucciones
Formato para almacenamiento de bytes

- El sistema *big-endian* adoptado por Motorola entre otros, consiste en representar los bytes en el orden "natural": así el valor hexadecimal CA FE BA BEH se codificaría en memoria en la secuencia {CA, FE, BA, BE}. En el sistema *little-endian* adoptado por Intel, entre otros, el mismo valor se codificaría como {BE, BA, FE, CA}, de manera que de este modo se hace más intuitivo el acceso a datos, porque se efectúa fácilmente de manera incremental de menos relevante a más relevante (siempre se opera con incrementos de contador en la memoria.



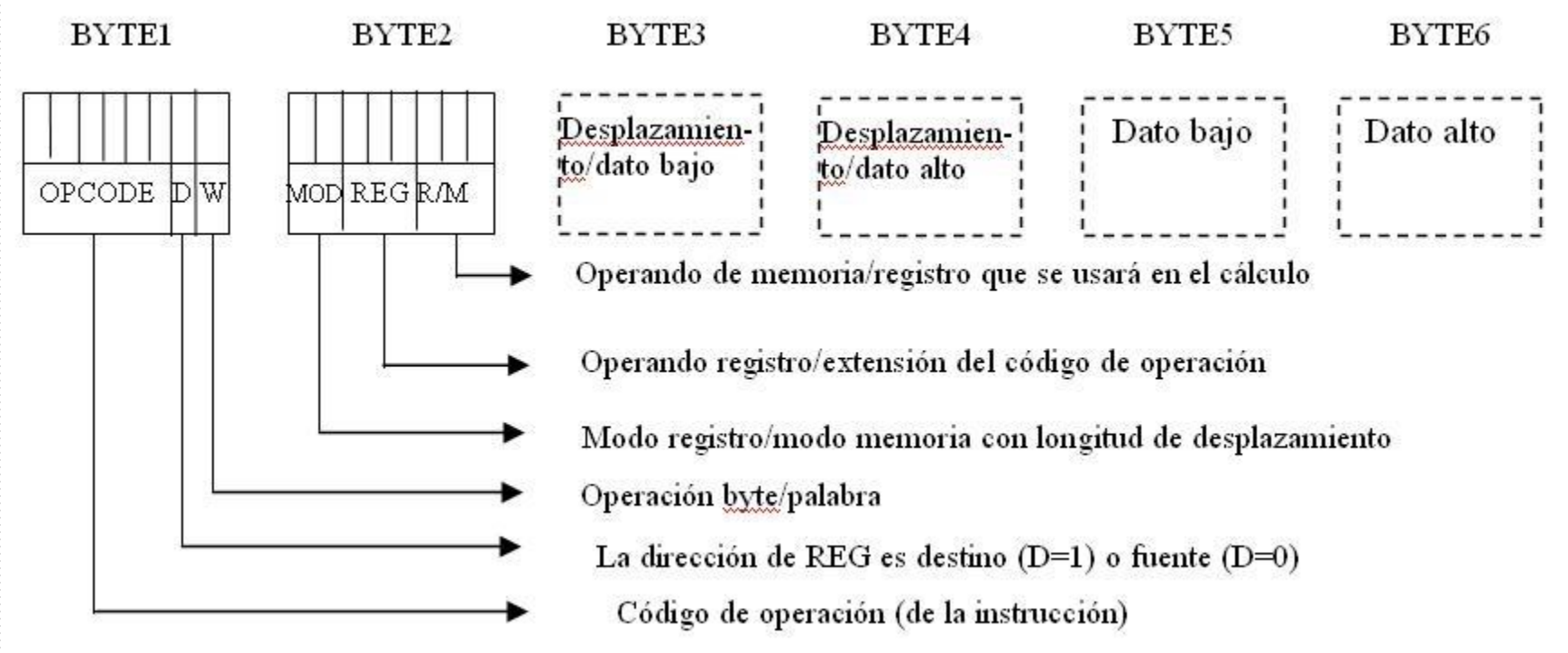
Codificación en Lenguaje Máquina
Formato de una Instrucción

- 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.

Tema 4. Conjunto de Instrucciones
Codificación en Lenguaje Máquina
Formato de una Instrucción

- El código máquina de las instrucciones del 8086 varia 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.

Tema 4. Conjunto de Instrucciones
Codificación en Lenguaje Máquina
Formato de una Instrucción



Tema 4. Conjunto de Instrucciones
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
- 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.

Tema 4. Conjunto de Instrucciones
Codificación en Lenguaje Máquina
Formato de una Instrucción

- 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.

Tema 4. Conjunto de Instrucciones
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

- *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	0001110w	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			

- *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			
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	0001110w	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			

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*

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		

Ejemplos de formatos de Instrucciones del 8086

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

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)