Conjunto de Instrucciones del 8086

Tabla 1.1 Instrucción MOV[a]

		Ejemplo codificado			
Mnemónico general Op-code Operando	**Código Objeto**	**Mnemónico**	**Segmento de memoria**	**Operación simbólica**	**Descripción**
MOV dest,fuente	8B C3	MOV AX,BX	Dentro del CPU	AX←BX	Registro a registro.
	8A E3	MOV AH,BL	Dentro del CPU	AH←BL	Registro a registro.
	A1 00 10	MOV AX,MEMWDS[b]	Datos	AL←[1000H], AH←[1001H]	Memoria a registro.
	A0 02 10	MOV AL,MEMBDS[c]	Datos	AL←[1002H]	Memoria a registro.
	89 1E 00 10	MOV MEMWDS,BX[b]	Datos	[1000H]←BL, [1001H]←BH	Registro a memoria.
	88 1E 02 10	MOV MEMBDS,BL[c]	Datos	[1002H]←BL	Registro a memoria.
	C7 06 00 10 34 12	MOV MEMWDS,1234H[b]	Datos	[1000H]←34H, [1001H]←12H	Dato inmediato a memoria.
	C6 06 02 10 34	MOV MEMBDE,34H[c]	Datos	[1002H]←34H	Dato inmediato a memoria.
	B0 10	MOV AL,10H	Código	AL←10H	Dato inmediato a registro.
	B8 00 10	MOV AX,1000H	Código	Al←00H, AH←10H	Dato inmediato a registro.
	8E D8	MOV DS,AX	Dentro del CPU	DS←AX	Registro de propósito general a registro de segmento.
	8C C2	MOV DX,ES	Dentro del CPU	DX←ES	Registro de segmento a registro de propósito general.
	8E 06 00 10	MOV ES,MEMWDS[b]	Datos	ES←[1001H:1000H]	Memoria a registro de segmento.
	8C 0E 00 10	MOV MEMWDS,CS[b]	Datos	[1001H:1000H]←CS	Registro de segmento a memoria.

[a]La localidad de memoria se puede especificar utilizando cualquiera de los modos de direccionamiento mostrados en la tabla 1.1. En esta tabla se utiliza modo de direccionamiento directo para todos los ejemplos de acceso a memoria.
[b]MEMWDS se asume que apunta a la palabra que empieza en la localidad 1000H en el segmento de datos.
[c]MEMBDS se asume que apunta al byte de la localidad 1002H en el segmento de datos.

Tabla 1.2. Instrucciones de Transferencia de datos especiales[a]

		Ejemplo codificado			
Mnemónico general Op-code Operando	**Código Objeto**	**Mnemónico**	**Segmento de memoria**	**Operación simbólica**	**Descripción**
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	Intercambia los contenidos del operando fuente byte o word con el operando destino, ninguna bandera es afectada.
LAHF	9F	LAHF	Dentro del CPU	AH ← Banderas	Copia el byte menos significativo de las banderas en AH.
SAHF	9E	SAHF	Dentro del CPU	Banderas ← AH	Copia AH en el byte menos significativo de las banderas.
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	Introduce (lee) un byte o palabra de un puerto de E/S directo 0-255. Introduce (lee) un byte o palabra de un puerto de E/S indirecto 0-65535; la dirección del puerto está en DX; ninguna bandera es afectada.
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	Proporciona (escribe) un byte o palabra a puertos directos de E/S 0-255. Proporciona (escribe) un byte o palabra a puertos de E/S indirectos 0-65535; la dirección del puerto está en DX: ninguna de las banderas se afecta.
LEA dest,fuente[c]	8D 1E 00 10	LEA BX,MEMBDS[c]	Datos	BL←00; BH ←10H	La dirección efectiva del operando fuente se transfiere al operando destino, ninguna bandera se afecta.
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]	Transfiere una variable apuntador de 32 bits del operando fuente en memoria al registro destino y al registro DS o ES. Ninguna bandera se afecta.
XLAT	D7	XLAT	Datos	AL←[BX+AL];	Remplaza el byte en AL con un byte de la tabla (de 256 bytes) empezando en [BX]; usa AL como un desplazamiento dentro de esta tabla; ninguna bandera es afectada.

[a]Los operandos pueden utilizar cualquier modo de direccionamiento mostrado en la tabla 1.1. En esta tabla se usa direccionamiento indirecto solo como ejemplo.
[b]Las instrucciones IN y OUT no involucran los registros de segmento.
[c]MEMBDS se asume que apunta al byte de la localidad 1000H en el segmento de datos.
[d]El destino debe ser un registro del CPU de 16 bits.

Tabla 1.3.1 Instrucciones de cadena (string)

		Ejemplo codificado			
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	

Tabla 1.3.2 Instrucciones de cadena (string) Continuación.

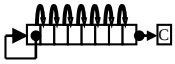
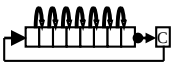
		Ejemplo codificado			
Mnemónico general Op-code Operando	**Código Objeto**	**Mnemónico**	**Segmento de memoria**	**Operación simbólica**	**Descripción**
MOVSB	A4	MOVSB	Datos, Extra	ES:[DI] ← DS:[SI] Si DF=0,DI←DI+1, SI←SI+1 Si DF=1, DI←DI-1, SI←SI-1	Transfiere un byte o palabra de una cadena direccionada por SI en el segmento de datos a una cadena direccionada por DI en el segmento extra; si DF=0, incrementa SI y DI, de lo contrario decrementa SI y DI; las banderas no son afectadas.
MOVSW	A5	MOVSW	Datos, Extra	ES:[DI] ← DS:[SI] ES:[DI+1]←DS:[SI+1] Si DF=0,DI←DI+2, SI←SI+2 Si DF=1, DI←DI-2, SI←SI-2	
SCASB	AE	SCASB	Extra	AL – ES:[DI]; actualiza banderas Si DF=0, DI← DI + 1 Si DF=1, DI← DI - 1	Substrae el byte o palabra de la cadena direccionado por DI en el segmento extra de AL o AX; Si DF=0, incrementa DI, de lo contrario decrementa DI, las banderas son actualizadas para reflejar la relación entre los dos operandos.
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	Substrae el byte o palabra del elemento cadena direccionado por DI en el segmento extra del byte o palabra del elemento cadena fuente direccionado por SI en el segmento de datos; Si DF=0, incrementa DI y SI; las banderas son actualizadas para reflejar la relación de los dos operandos.
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	

Tabla 1.4 Prefijo REP

		Ejemplo codificado			
Mnemónico general Op-code Operando	**Código Objeto**	**Mnemónico**	**Segmento de memoria**	**Operación simbólica**	**Descripción**
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 – 1 Repite hasta que CX=0 MOVSW; CX←CX-1 Repite hasta que CX=0	La instrucción de cadena seguida del prefijo REP se repite hasta que CX se decrementa hasta 0.
REPE/REPZ[a]	F3 AE F3 AF F3 A6 F3 A7	REPZ SCASB REPZ SCASW REPZ CMPSB REPZ CMPSW	Extra Extra Extra, datos Extra, datos	SCASB; CX ← CX – 1 Repite si ZF = 1 y CX ≠ 0 Similar a la anterior excepto SCASW Similar a la anterior excepto CMPSB Similar a la anterior excepto CMPSW	Repite la operación de cadena si la búsqueda (SCAS) o comparación (CMPS) es igual (ZF=1) y CX ≠ 0; el decremento de CX no afecta a las banderas.
REPNE/REPNZ[a]	F2 AE F2 AF F2 A6 F2 A7	REPNE SCASB REPNE SCASW REPNE CMPSB REPNE CMPSW	Extra Extra Extra, datos Extra, datos	SCASB; CX ← CX – 1 Repite si ZF = 0 y CX ≠ 0 Similar a la anterior excepto SCASW Similar a la anterior excepto CMPSB Similar a la anterior excepto CMPSW	Repite la operación de cadena si la búsqueda (SCAS) o comparación (CMPS) no es igual (ZF=0) y CX ≠ 0; el decremento de CX no afecta a las banderas.

[a]Cualquiera de los mnemónicos puede utilizarse.

Tabla 1.5. Instrucciones de Corrimiento y Rotación

		Ejemplo codificado[a]			
Mnemónico general Op-code Operando	**Código Objeto**	**Mnemónico**	**Segmento de memoria**	**Operación simbólica**	**Descripción**
SAL/SHL[b] dest,contador	D1 E0 D3 E0	SAL AX,1 SAL AX,CL	Dentro del CPU Dentro del CPU	A←shl A, A_1←0 	Un operando byte o palabra se recorre a la izquierda una o CL veces; AF es indefinida, todas las demás banderas son actualizadas; para corrimientos de un solo bit, OF OF es activada si cambia el signo del operando.
SAR dest, contador	D0 F8 D2 F8	SAR AL,1 SAR AL,CL	Dentro del CPU Dentro del CPU	A←shr A, (S)←A(S)[e] 	
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		Un operando byte o palabra se rota a la izquierda o a la derecha una o CL veces; solo se afectan CF y OF; para rotaciones de un solo bit, OF se activa si el signo del operando cambia.
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[c] ROR MEMBDS,CL[d]	Datos Datos		

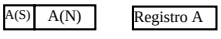
[a]Cualquiera de los modos de direccionamiento puede utilizarse.
[b]Cualquier forma del mnemónico puede utilizarse.
[c]MEMWDS se asume que apunta a la palabra que empieza en la localidad 1000H en el segmento de datos.
[d]MEMBDS se asume que apunta al byte de la localidad 1004H en el segmento de datos.
[e]Figuras para los corrimientos aritméticos A representa un registro o localidad de memoria. 

Tabla 1.6 Instrucciones Lógicas

		Ejemplo codificado			
Mnemónico general Op-code Operando	**Código Objeto**	**Mnemónico[a]**	**Segmento de memoria**	**Operación simbólica**	**Descripción**
NOT dest	F7 D3 F6 14	NOT BX NOT BYTE PTR[SI]	Dentro del CPU Datos	BX← $\overline{BX}$ [SI]← $\overline{[SI]}$	Complementa todos los bits del operando byte o palabra; no se afecta ninguna bandera.
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	Realiza la función AND bit a bit en tamaño palabra o byte con los operandos fuente y destino. Almacena el resultado en el operando destino; AF es indefinida, las demás banderas se actualizan[b].
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	Realiza la función OR bit a bit en tamaño palabra o byte con los operandos fuente y destino, almacena el resultado en el operando destino; AF es indefinida, las demás banderas se actualizan[b].
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	Realiza la función OR exclusiva bit a bit en tamaño palabra o byte con los operandos fuente y destino, almacena el resultado en el operando destino; AF es indefinida, las demás banderas se actualizan[b].
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	Realiza la función AND bit a bit en tamaño palabra o byte con los operandos fuente y destino, los operandos permanecen sin cambiar (no guarda resultado); AF es indefinida, las demás banderas se actualizan[b].

[a]Cualquier modo de direccionamiento puede utilizarse.
[b] CF y OF son puestas a cero.

Tabla 1.7 Instrucciones de Adición y Substracción

		Ejemplo codificado			
Mnemónico general Op-code Operando	**Código Objeto**	**Mnemónico[a]**	**Segmento de memoria**	**Operación simbólica**	**Descripción**

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
					Reemplaza el byte o palabra destino con la suma de los operandos fuente y destino; actualiza todas las banderas.

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	Reemplaza el byte o palabra destino con la suma de los operandos fuente y destino más el acarreo; actualiza todas las banderas.

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
					Reemplaza el byte o palabra destino con la diferencia entre el operando fuente y destino; actualiza todas las banderas.

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[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	Reemplaza el byte o palabra destino con la diferencia entre el operando fuente y destino menos el acarreo; actualiza todas las banderas.

Tabla 1.8 Instrucciones de Adición y Substracción (Continuación)

		Ejemplo codificado			
Mnemónico general Op-code Operando	**Código Objeto**	**Mnemónico[a]**	**Segmento de memoria**	**Operación simbólica**	**Descripción**
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	Agrega uno al operando destino byte o palabra; almacena el resultado en el operando destino; todas las banderas excepto CF se actualizan.
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	Resta uno del operando destino byte o palabra; almacena el resultado en el operando destino; todas las banderas excepto CF se actualizan.
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]	Genera el complemento a 2 del operando destino byte o palabra; actualiza todas las banderas (CF=1 excepto cuando el operando es cero)

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
					Substrae el operando fuente byte o palabra del operando destino; los operandos permanecen sin cambios; las banderas se actualizan para reflejar la relación de los operandos.

[a]MEMWDS se asume que apunta a la palabra que inicia en la localidad 1000H en el segmento de datos.
[b] Operandos Inmediatos no se permiten.
[c] MEMBDS se sume que apunta al byte en la localidad 1004H en el segmento de datos.

Tabla 1.9 Instrucciones de Multiplicación y División.

		Ejemplo codificado			
Mnemónico general Op-code Operando[a]	**Código Objeto**	**Mnemónico[a]**	**Segmento de memoria**	**Operación simbólica**	**Descripción**
MUL fuente	F6 E3 F7 E1 F6 27 F7 26 00 10	MUL BL MUL CX MUL BYTE PTR[BX] MUL MEMWDS[b]	Dentro del CPU Dentro del CPU Datos Datos	AX ← AL * BL DX:AX←AX * CX AX← AL * [BX] DX:AX←AX* [1001H:1000H]	Multiplicación sin signo de un operando fuente byte o palabra y el acumulador; el producto se almacena en AX; si el resultado no puede almacenarse en un solo byte (en operaciones de 8 bits) o en una simple palabra (en operaciones de 16 bits) CF y OF se ponen a uno, de los contrario se ponen a cero; las demás banderas son indefinidas.
IMUL fuente	F6 EB F7 E9 F6 2F F7 2E 00 10	IMUL BL IMUL CX IMUL BYTE PTR[BX] IMUL MEMWDS[b]	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)	Similar a MUL excepto que se utilizan números con signo; el rango del operando fuente es de –128 a +127 para multiplicaciones de byte y de –32768 a +32767 para multiplicaciones en palabra; CF y OF se ponen a uno si el resultado no puede representado en el registro de bajo orden; de lo contrario, se ponen en cero, el signo se extiende en el registro más significativo; las demás banderas no se afectan.
DIV	F6 F3 F7 F1 F6 37 F7 36 00 10	DIV BL DIV CX DIV BYTE PTR[BX] DIV MEMWDS[b]	Dentro del CPU Dentro del CPU Datos Datos	AX ← AX / BL DX:AX←DXAX / CX AX← AX / [BX] DX:AX←DXAX / [1001H:1000H]	División sin signo del acumulador (para divisores de 8 bits) o acumulador y DX (para divisores de 16 bits); para divisores de 8 bits el cociente se deja en AL y el residuo en AH; para divisores de 16 bits el cociente queda en AX y el residuo en DX; si el cociente excede la capacidad de su registro destino (AL o AX) una interrupción de tipo 0 se genera; todas las banderas están indefinidas.
IDIV	F6 FB F7 F9 F6 3F F7 3E 00 10	IDIV BL IDIV CX IDIV BYTE PTR[BX] IDIV MEMWDS[b]	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)	Similar a DIV excepto que se realiza la división con signo; el rango del operando fuente es de –128 a +127 para divisiones en 8 bits y de –32768 a +32767 en divisiones de 16 bits.

[a]Los operandos inmediatos no son permitidas.
[b] MEMWDS se asume que apunta a la palabra empezando en la localidad 1000H en el segmento de datos.

Tabla 1.10 Instrucciones de Ajuste Aritmético.

		Ejemplo codificado			
Mnemónico general Op-code Operando	**Código Objeto**	**Mnemó-nico**	**Segmento de memoria**	**Operación simbólica**	**Descripción**
DAA	27	DAA	Dentro del CPU	Si AL ∧ 0F > 9 o AF = 1, entonces AL←AL+ 6; AF ← 1 Si A > 9F o CF = 1, entonces AL ← AL + 60H; CF ← 1	Ajusta AL a un par de dígitos decimales empaquetados válidos seguidos de la adición de dos operandos decimales válidos empaquetados o desempaquetados; todas las banderas excepto OF (indefinida) se afectan.
DAS	2F	DAS	Dentro del CPU	Si AL ∧ 0F > 9 o AF = 1, entonces AL←AL-6; AF ← 1 Si A > 9F o CF = 1, entonces AL ← AL - 60H; CF ← 1	Ajusta AL a un par de dígitos decimales empaquetados válidos seguidos de la resta de dos operandos decimales válidos empaquetados o desempaquetados; todas las banderas excepto OF (indefinida) se afectan.
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	Ajusta AL a un dígito decimal desempaquetado seguido de la adición de dos operandos decimales válidos desempaquetados; los 4 bits de mayor orden de AL serán cero y AH se incrementa en uno; todas las banderas excepto AF y CF son indefinidas.
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	Ajusta AL a un único dígito decimal desempaquetado seguido de la substracción de dos operandos decimales desempaquetados válidos; los 4 bits de mayor orden de AL serán cero y AH es decrementado en uno; todas las banderas excepto AF y CF son indefinidas.
AAM	D4 0A	AAM	Dentro del CPU	AH ← AL / 0AH AL ← Residuo	Seguido de la multiplicación de dos operandos decimales válidos desempaquetados, AAM convierte el resultado en AL en dos dígitos decimales válidos desempaquetados en AH y AL; todas las banderas excepto PF, SF y ZF son indefinidas.
AAD	D5 0A	AAD	Dentro del CPU	AL ← (AH * 0AH) + AL AH ← 0	Antes de dividir AX entre un operando decimal de un dígito desempaquetado, AAD convierte los dos números decimales desempaquetados en AX a un número binario en AL y cero en AH; el cociente producido por la división siguiente será un número decimal válido desempaquetado en AL y el cociente en AH; todas las banderas excepto PF, SF y ZF son indefinidas.
CBW	98	CBW	Dentro del CPU	Si AL < 80H, entonces AH ← 0 Si AL > 7F, entonces AH ← FFH	Antes de dividir AX por un operando byte, CBW extiende el signo del dividendo en tamaño byte contenido en AL hacia AH, es decir convierte el byte en AL en una palabra válida con signo en AX; no se afecta ninguna bandera.
CWD	99	CWD	Dentro del CPU	Si AX < 8000H, entonces DX← 0 Si AX > 7FFFFH, entonces DX ← FFFFH	Similar a CBW pero extiende el signo del dividendo contenido en AX en una doble palabra contenida en DX: AX; no afecta banderas.

Tabla 1.11 Instrucciones JUMP

		Ejemplo codificado			
Mnemónico general Op-code Operando	**Código Objeto**	**Mnemónico**	**Segmento de memoria**	**Operación simbólica**	**Descripción**
JMP near etiqueta	E9 -- -- [a] FF 26 00 10 FF 27 FF E0	JMP MEMN[b] JMP [MEMWDS][c] JMP [BX] JMP AX	Código Datos Datos Dentro del CPU	IP← MEMN IP←[MEMWDS+1:MEMWDS] IP←[BX+1:BX] IP←AX	Transfiere el control a la localidad de la etiqueta (cercana) dentro del segmento; el modo de direccionamiento debe ser directo, indirecto a memoria, o registro indirecto.
JMP SHORT etiqueta	EB --[d]	JMP SHORT MEMS[e]	Código	IP←MEMS	Transfiere el control a la localidad de la etiqueta (corta); no se permiten modos indirectos con esta instrucción.
JMP far etiqueta	EA 03 00 D3 9E FF 2E 05 10 FF 2F	JMP FAR PTR MEMF[f] JMP [MEMWWDS][g] 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]	Transfiere el control a la localidad de la etiqueta fuera del segmento.
Jcond[h] short etiqueta	73 --[d]	JNC MEMS	Código	Si CF=0, entonces IP←MEMS	Transfiere el control a la dirección de la etiqueta (corta) si la condición es verdadera; todos los saltos condicionales requieren etiquetas cortas.
JCXZ short etiqueta	E3 --[d]	JCXZ MEMS	Código	Si CX=0, entonces IP←MEMS	Transfiere el control a la dirección de la etiqueta corta si CX=0.

[a]Estos dos bytes son el offset (desplazamiento) en complemento a 2 entre la localidad de memoria cercana MEMN y la instrucción que sigue inmediatamente a la instrucción JMP. Por ejemplo, si el salto es de 12 localidades hacia delante, el offset es de 00 0CH. Similarmente, 12 localidades hacia atrás requieren un offset de FFF4H. El offset queda limitado a 32767 localidades hacia delante (7FFFH) y –32768 localidades hacia atrás (8000H).

[b]MEMN se asume que apunta a la localidad de memoria cercana (i.e., dentro del segmento).

[c]MEMWDS se asume que apunta a la palabra que inicia en la localidad 1000H en el segmento de datos. Los corchetes son opcionales ya que MEMWDS define una palabra, no la localidad a la que se va a brincar.

[d]Este byte es el offset en complemento a 2 entre la localidad de memoria corta MEMS y la instrucción siguiente inmediata a la instrucción de salto SHORT. El offset se limita a 127 localidades hacia delante (7FH) y –128 localidades hacia atrás (80H).

[e] MEMS se asume que apunta a la localidad de memoria corta (i.e., +127 o –128 localidades de la dirección de la instrucción inmediata siguiente a la instrucción JMP SHORT). El operador SHORT es opcional si MEMS es conocido por el ensamblador.

[f] MEMF se asume que apunta a la localidad de memoria lejana 9ED3H:0003H.

[g] MEMWWDS se asume que apunta a la doble palabra que inicia en la localidad 1005H en el segmento de datos. Los corchetes son opcionales ya que MEMWWDS define una doble palabra, no la localidad a la que saltará.

[h] Ver tabla 1.13 para la lista de condiciones permitidas.

Tabla 1.12 Condiciones de saltos.

Saltos Condicionales para números sin signo		
Mnemónico de la Instrucción de Salto	**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 de la Instrucción de Salto	**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 ⊕ OF)=1
JLE / JNG	Salta si es menor o igual o salta si no es mayor	((SF ⊕ OF) + ZF) = 1
Saltos Condicionales para números con signo y sin signo		
Mnemónico de la Instrucción de Salto	**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

Tabla 1.13 Instrucciones de LOOP.

		Ejemplo codificado			
Mnemónico general Op-code Operando[a]	**Código Objeto**	**Mnemónico[a]**	**Segmento de memoria**	**Operación simbólica**	**Descripción**
LOOP etiqueta corta	E2 --[a]	LOOP MEMS[b]	Código	CX ← CX –1 Si CX ≠ 0, entonces IP ← MEMS	Decrementa CX y transfiere el control a la dirección de la etiqueta corta si CX ≠ 0.
LOOPE /[c] etiqueta corta LOOPZ	E1 --[a]	LOOPZ MEMS[b]	Código	CX ← CX –1 Si (CX≠0)^(ZF=1), entonces IP ← MEMS	Decrementa CX y transfiere el control a la dirección de la etiqueta corta si CX ≠ 0 y la última instrucción afecta la bandera de cero (Z=1).
LOOPNE /[c] etiqueta corta LOOPNZ	E0 --[a]	LOOPNZ MEMS[b]	Código	CX ← CX –1 Si (CX≠0)^(ZF=0), entonces IP ← MEMS	Decrementa CX y transfiere el control a la dirección de la etiqueta corta si CX ≠ 0 y la última instrucción afecta el resultado de la bandera de cero en ZF=0.

[a]Este byte es desplazamiento en complemento a 2 entre la localidad de memoria corta MEMS y la instrucción inmediata siguiente a la instrucción de LOOP. El desplazamiento se limita a un máximo de 127 localidades hacia delante (7FH) y 128 localidades hacia atrás (80H).
[b] MEMS se asume que apunta a la localidad de memoria corta (i.e., +127 o 128 localidades de la dirección de la instrucción inmediata seguida de la instrucción LOOP).
[c]Cualquiera de los dos mnemónicos puede ser utilizado.

Tabla 1.14.1 Instrucciones de llamada a subrutina y retornos de subrutina

		Ejemplo codificado			
Mnemónico general Op-code Operando	**Código Objeto**	**Mnemónico**	**Segmento de memoria**	**Operación simbólica**	**Descripción**
CALL etiqueta cercana	E8 -- --[a]	CALL MEMN[b]	Código	SP← SP –2; [SP+1:SP]← IP; IP← MEMN	IP se introduce en el tope de la pila y el control se transfiere dentro del segmento a la dirección de la etiqueta cercana.
	FF 16 00 10	CALL [MEMWDS][c]	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	
CALL etiqueta lejana	9A 00 10 D3 09	CALL FAR PTR MEMF[d]	Código	SP← SP –2; [SP+1:SP]← CS; CS←09D3H; SP← SP –2; [SP+1:SP]← IP; IP← 1000H	Se introduce en el tope de la pila CS e IP y el control es transferido a un nuevo segmento a la dirección de la etiqueta lejana.
	FF 1E 00 10	CALL [MEMWWDS][e]	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]	

Tabla 1.14.2 Instrucciones de llamada a subrutina y retornos de subrutina (Continuación)

		Ejemplo codificado			
Mnemónico general Op-code Operando	**Código Objeto**	**Mnemónico**	**Segmento de memoria**	**Operación simbólica**	**Descripción**
RET n (cercano)[f]	C3 C2 08 00	RET RET 8	Pila Pila	IP ← [SP+1:SP]; SP← SP +2; IP←[SP+1:SP]; SP← SP + 2 + 8	La palabra en el tope de la pila es extraído y puesto en el IP transfiriendo el control a esta nueva dirección; RET se usa normalmente para regresar el control a la instrucción siguiente de un llamado de subrutina cercano; si se incluyó el valor (n) opcional al RET, éste se suma al SP.
RET n (lejano)[f]	CB CA 08 00	RET RET 8	Pila Pila	IP←[SP+1:SP]; SP← SP + 2; CS ←[SP+1:SP]; SP←SP+2 IP ← [SP+1:SP]; SP←SP + 2; CS←[SP+1:SP]; SP←SP + 2 + 8	Similar a la de arriba excepto que la doble palabra en el tope del stack es extraída y puesta en el IP y CS, transfiriendo el control a esta nueva dirección lejana.

[a]Estos dos bytes son el offset (desplazamiento) en complemento a 2 entre la localidad de memoria cercana MEMN y la instrucción que sigue inmediatamente a la instrucción CALL. Por ejemplo, si el CALL es de 12 localidades hacia delante, el offset es de 00 0CH. Similarmente, 12 localidades hacia atrás requieren un offset de FFF4H. El offset queda limitado a 32767 localidades hacia delante (7FFFH) y –32768 localidades hacia atrás (8000H).
[b]MEMN se asume que apunta a la localidad de memoria cercana (i.e., dentro del segmento).
[c]MEMWDS se asume que apunta a la palabra que inicia en la localidad 1000H en el segmento de datos. Los corchetes son opcionales ya que MEMWDS define una palabra no una localidad a la que va a saltar.
[d]FAR PTR indica que MEMF está localizada en un segmento de código diferente. En este caso se asume que MEMF apunta a la localidad 09D3H:1000H.
[e]MEMWWDS se asume que apunta a la doble palabra empezando en la localidad 1000H en el segmento de datos. Los corchetes son opcionales ya que MEMWWDS define una doble palabra, no una localidad a la que va a saltar.
[f]El mismo mnemónico es usado para un retorno cercano o lejano. El ensamblador generará el código apropiado en base a la definición del enunciado del procedimiento (near o far) cercano o lejano.

Tabla 1.15 Instrucciones PUSH Y POP

		Ejemplo codificado			
Mnemónico general Op-code Operando	**Código Objeto**	**Mnemónico**	**Segmento de memoria**	**Operación simbólica**	**Descripción**
PUSH fuente[a]	51 1E FF 75 02	PUSH CX PUSH DS PUSH[DI+2]	Pila Pila Pila, datos	SP←SP-2; [SP+1]←CH; [SP]←CL SP←SP-2; [SP+1:SP]←DS SP←SP-2; [SP+1]←[DI+3]; [SP]←[DI+2]	Decrementa el SP en 2 y transfiere la palabra del operando fuente al tope de l a pila (stack) ahora apuntado por el nuevo valor de SP.
POP dest[a]	59 1F 8F 45 02	POP CX POP DS POP[DI+2]	Pila Pila Datos, pila	CL←[SP]; CH←[SP+1]; SP←SP+2 DS←[SP+1:SP]; SP←SP+2 [DI+3]←[SP+1]; [DI+2]←[SP]; SP←SP+2	Transfiere la palabra del tope de la pila (stack) apuntado por SP al operando destino e incrementa al SP en 2.
PUSHF	9C	PUSHF	Pila	SP←SP-2; [SP+1:SP]← Banderas	Introduce los 16 bits del registro de banderas a la pila.
POPF	9D	POPF	Pila	Banderas←[SP+1:SP]; SP←SP+2	Extrae la palabra que está en el tope de la pila al registro de banderas de 16 bits.

[a]No se permiten operandos inmediatos.

Tabla 1.16 Instrucciones de Interrupción por software

		Ejemplo codificado			
Mnemónico general Op-code Operando	**Código Objeto**	**Mnemónico**	**Segmento de memoria**	**Operación simbólica**	**Descripción**
INT tipo	CD 23	INT 23H	Pila e interrupción saltando a la tabla de vectores (en 00000 a 003FFH)	SP← SP –2; [SP+1:SP]← Banderas IF←0; TF←0; SP← SP –2; [SP+1:SP]← CS; CS←[0008FH:0008EH];[a] SP← SP –2; [SP+1:SP]← IP; IP←[0008DH:0008CH][b]	Salva los registros de banderas, CS e IP en la pila y transfiere el control a la dirección lejana almacenada en la doble palabra empezando en la dirección absoluta proporcionada por el tipo* 4. Tipo es el número de interrupción.
INTO	CE	INTO	Pila e interrupción saltando a la tabla de vectores (en 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];[a] SP← SP –2; [SP+1:SP]←IP; IP ←[00011H:00010H][b]	Si una condición de sobreflujo (desbordamiento) existe (OF=1), se ejecuta un tipo de interrupción 4.
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	Transfiere el control de nuevo a la siguiente instrucción donde se invocó la interrupción, extrayendo el IP, CS y registro de banderas de la pila; IRET se utiliza normalmente para salir de cualquier procedimiento de interrupción.

[a]La dirección donde se encuentra el CS en la tabla de vectores está dada por [(tipo * 4) + 3: (tipo * 4) + 2].
[b]La dirección donde se encuentra el IP en la tabla de vectores está dada por [(tipo * 4) + 1: (tipo * 4)].

Tabla 1.17 Instrucciones de Control del Procesador.

		Ejemplo codificado			
Mnemónico general Op-code Operando	**Código Objeto**	**Mnemónico**	**Segmento de memoria**	**Operación simbólica**	**Descripción**
STC	F9	STC	Dentro del CPU	CF ← 1	Pone a uno la bandera de acarreo.
CLC	F8	CLC	Dentro del CPU	CF ← 0	Pone a cero la bandera de acarreo.
CMC	F5	CMC	Dentro del CPU	CF ← $\overline{CF}$	Complementa la bandera de acarreo.
STD	FD	STD	Dentro del CPU	DF ←1	Pone a uno la bandera de dirección (D) para auto-decremento en instrucciones de cadena (string).
CLD	FC	CLD	Dentro del CPU	DF ← 0	Pone a cero la bandera de dirección (D) para auto-incremento en instrucciones de cadena (string).
STI	FB	STI	Dentro del CPU	IF ← 1	Pone a uno la bandera de Interrupción (habilitando las interrupciones en la línea INTR).
CLI	FA	CLI	Dentro del CPU	IF ← 0	Pone a cero la bandera de Interrupción (deshabilitando las interrupciones en la línea INTR).
HLT	F4	HLT	Dentro del CPU	Ninguna	Estado Halt (detener, parar).
WAIT	9B	WAIT	Dentro del CPU	Ninguna	Pone en estado de espera (si $\overline{TEST}$=1) al procesador hasta que el coprocesador termina su ejecución.
LOCK Instrucción	F0 A1 00 10	LOCK MOV AX,MEMWDS[a]	Datos	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	90	NOP	Dentro del CPU	Ninguna	No operación.
ESC número,fuente	DE 0E 00 10	ESC 31H,MEMWDS[a]	Datos	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.

[a]MEMWDS se asume que apunta a la palabra que empieza en la localidad 1000H en el segmento de datos.