Instruction Code Summary

H / L	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP	JBC bit, rel	JB bit, rel	JNB bit, rel	JC rel	JNC rel	JZ rel	JNZ rel	SJMP rel	MOV DPTR, # datos16	ORL C, /bit	ANL C, /bit	PUSH dir	POP dir	MOVX A, @DPTR	MOVX @DPTR, A
1	AJMP (P0)	ACALL (P0)	AJMP (P1)	ACALL (P1)	AJMP (P2)	ACALL (P2)	AJMP (P3)	ACALL (P3)	AJMP (P4)	ACALL (P4)	AJMP (P5)	ACALL (P5)	AJMP (P6)	ACALL (P6)	AJMP (P7)	ACALL (P7)
2	LJMP dir16	LCALL dir16	RET	RETI	ORL dir, A	ANL dir, A	XRL dir, A	ORL C, bit	ANL C, bit	MOV bit, C	MOV C, bit	CPL bit	CLR bit	SETB bit	MOVX A, @R0	MOVX @R0, A
3	RR A	RRC A	RL A	RLC A	ORL dir, # datos	ANL dir, # datos	XRL dir, # datos	JMP @A+DPTR	MOVC A, @A+PC	MOVC A, @A+DPTR	INC DPTR	CPL C	CLR C	SETB C	MOVX A, @R1	MOVX @R1, A
4	INC A	DEC A	ADD A, # datos	ADDC A, # datos	ORL A, # datos	ANL A, # datos	XRL A, # datos	MOV A, # datos	DIV AB	SUBB A, # datos	MUL AB	CJNE A, # datos, rel	SWAP A	DA A	CLR A	CPL A
5	INC dir	DEC dir	ADD A, dir	ADDC A, dir	ORL A, dir	ANL A, dir	XRL A, dir	MOV dir, # datos	MOV dir, dir	SUBB A, dir		CJNE A, dir, rel	XCH A, dir	DJNZ dir, rel	MOV A, dir	MOV dir, A
6	INC @R0	DEC @R0	ADD A, @R0	ADDC A, @R0	ORL A, @R0	ANL A, @R0	XRL A, @R0	MOV @R0, # datos	MOV dir, @R0	SUBB A, @R0	MOV @R0, dir	CJNE @R0, # datos, rel	XCH A, @R0	XCHD A, @R0	MOV A, @R0	MOV @ R0, A
7	INC @R1	DEC @R1	ADD A, @R1	ADDC A, @R1	ORL A, @R1	ANL A, @R1	XRL A, @R1	MOV @R1, # datos	MOV dir, @R1	SUBB A, @R1	MOV @R1, dir	CJNE @R1, # datos, rel	XCH A, @R1	XCHD A, @R1	MOV A, @R1	MOV @ R1, A
8	INC R0	DEC R0	ADD A, R0	ADDC A, R0	ORL A, R0	ANL A, R0	XRL A, R0	MOV R0, # datos	MOV dir, R0	SUBB A, R0	MOV R0, dir	CJNE R0, # datos, rel	XCH A, R0	DJNZ R0, rel	MOV A, R0	MOV R0, A
9	INC R1	DEC R1	ADD A, R1	ADDC A, R1	ORL A, R1	ANL A, R1	XRL A, R1	MOV R1, # datos	MOV dir, R1	SUBB A, R1	MOV R1, dir	CJNE R1, # datos, rel	XCH A, R1	DJNZ R1, rel	MOV A, R1	MOV R1, A
A	INC R2	DEC R2	ADD A, R2	ADDC A, R2	ORL A, R2	ANL A, R2	XRL A, R2	MOV R2, # datos	MOV dir, R2	SUBB A, R2	MOV R2, dir	CJNE R2, # datos, rel	XCH A, R2	DJNZ R2, rel	MOV A, R2	MOV R2, A
B	INC R3	DEC R3	ADD A, R3	ADDC A, R3	ORL A, R3	ANL A, R3	XRL A, R3	MOV R3, # datos	MOV dir, R3	SUBB A, R3	MOV R3, dir	CJNE R3, # datos, rel	XCH A, R3	DJNZ R3, rel	MOV A, R3	MOV R3, A
C	INC R4	DEC R4	ADD A, R4	ADDC A, R4	ORL A, R4	ANL A, R4	XRL A, R4	MOV R4, # datos	MOV dir, R4	SUBB A, R4	MOV R4, dir	CJNE R4, # datos, rel	XCH A, R4	DJNZ R4, rel	MOV A, R4	MOV R4, A
D	INC R5	DEC R5	ADD A, R5	ADDC A, R5	ORL A, R5	ANL A, R5	XRL A, R5	MOV R5, # datos	MOV dir, R5	SUBB A, R5	MOV R5, dir	CJNE R5, # datos, rel	XCH A, R5	DJNZ R5, rel	MOV A, R5	MOV R5, A
E	INC R6	DEC R6	ADD A, R6	ADDC A, R6	ORL A, R6	ANL A, R6	XRL A, R6	MOV R6, # datos	MOV dir, R6	SUBB A, R6	MOV R6, dir	CJNE R6, # datos, rel	XCH A, R6	DJNZ R6, rel	MOV A, R6	MOV R6, A
F	INC R7	DEC R7	ADD A, R7	ADDC A, R7	ORL A, R7	ANL A, R7	XRL A, R7	MOV R7, # datos	MOV dir, R7	SUBB A, R7	MOV R7, dir	CJNE R7, # datos, rel	XCH A, R7	DJNZ R7, rel	MOV A, R7	MOV R7, A

2Bytes	3Bytes
2Ciclos	4Ciclos

FIGURA B–1
Mapa de códigos de operación

C

Definiciones de las instrucciones[1]

LEYENDA

Símbolo	Interpretación
←	es reemplazado por . . .
()	el contenido de . . .
(( ))	los datos apuntados por . . .
rrr	uno de ocho registros; 000 = R0, 001 = R1, etc.
dddddddd	bits de datos
aaaaaaaa	bits de dirección
bbbbbbbb	dirección de un bit
i	direccionamiento indirecto utilizando R0 (i = 0) o R1 (i = 1)
eeeeeeee	dirección relativa de 8 bits

ACALL addrll

Función: Llamada absoluta

Descripción: ACALL llama incondicionalmente a una subrutina que se ubica en la dirección indicada. La instrucción incrementa dos veces al contador de programa (PC) para obtener la dirección de la instrucción que le sigue, y entonces almacena el resultado de 16 bits en la pila (con el byte inferior primero) e incrementa dos veces al apuntador de pila. La dirección de destino se obtiene al concatenar satisfactoriamente los cinco bits de orden superior del PC ya incrementado, los bits 7 a 5 del código de operación, y el segundo byte de la instrucción.

[1] Adaptadas del documento Controladores de 8 bits (270645) incrustados, Santa Clara, CA: Intel Corporation, 1991, con permiso de Intel Corporation.

La subrutina llamada debe, por lo tanto, iniciar dentro del mismo bloque de 2K de la memoria de programa donde inicia el primer byte de la instrucción que sigue de la instrucción ACALL. Ninguna bandera se ve afectada.

Ejemplo: Al principio, SP equivale a 07H. La etiqueta "SUBRTN" es una ubicación en memoria del programa 0345H. Después de ejecutar la instrucción,

```
ACALL        SUBRTN
```

en la ubicación 0123H, el SP contiene el valor 09H, las ubicaciones en memoria RAM interna 08H y 09H contienen los valores 25H y 01H, respectivamente, y el PC contiene el valor 0345H.

Bytes: 2

Ciclos: 2

Codificación: aaa10001 aaaaaaaa

Nota: aaa = Al0 ← A8 y aaaaaaaa = A7 − A0 de la dirección de destino.

Operación: $(PC) \cdot (PC) + 2$

$(SP) \leftarrow (SP) + 1$

$((SP)) \leftarrow (PC7 - PC0)$

$(SP) \leftarrow (SP) + 1$

$((SP)) \leftarrow (PC15 - PC8)$

$(PC10 = PC0) \leftarrow$ dirección de página

ADD A,<byte origen>

Función: Suma

Descripción: ADD hace una suma entre la variable byte indicada y el acumulador, y deja el resultado en el acumulador. Las banderas de acarreo y de acarreo auxiliar se activan, respectivamente, si hay un acarreo del bit 7 o del bit 3, y se borran en el caso contrario. Cuando se hace una suma de enteros sin signo, la bandera de acarreo indica si ocurrió un desbordamiento.

OV se activa si ocurre un acarreo del bit 6 pero no del bit 7, o un acarreo del bit 7 pero no del bit 6; en el caso contrario, OV se borra. Cuando se hace una suma de enteros con signo, OV indica que se produjo un número negativo resultante de la suma de sus dos operandos positivos, o que se produjo una suma positiva de dos operandos negativos.

Se permiten cuatro modos de direccionamiento para el operando de origen: por registros, directo, indirecto por registros, o inmediato.

Ejemplo: El acumulador contiene el valor 0C3H (000011B) y el registro 0 contiene 0AAH (10101010B). La instrucción,

```
ACC         A,R0
```

deja el valor 6DH (01101110B) en el acumulador con la bandera AC en cero, y tanto la bandera de acarreo como la OV se establecen a 1.

ADD A,Rn

Bytes: 1
Ciclos: 1
Codificación: 00101rrr
Operación: (A) ← (A) + (Rn)

ADD A,directa

Bytes: 2
Ciclos: 1
Codificación: 00100101 aaaaaaaa
Operación: (A) ← (A) + (directa)

ADD A,@Ri

Bytes: 1
Ciclos: 1
Codificación: 0010011i
Operación: (A) ← (A) + ((Ri))

ADD A,#datos

Bytes: 2
Ciclos: 1
Codificación: 00100100 dddddddd
Operación: (A) ← (A) + #datos

ADDC A,<byte origen>

Función: Suma con acarreo

Descripción: ADDC suma en forma simultánea la variable byte indicada, la bandera de acarreo, y el contenido del acumulador, y deja el resultado en el acumulador. Las banderas de acarreo y de acarreo auxiliar se activan, respectivamente, si hay un acarreo del bit 7 o del bit 3, y se borran en el caso contrario. Cuando se hace una suma de enteros sin signo, la bandera de acarreo indica que ocurrió un desbordamiento.

OV se activa si hay un acarreo del bit 6 pero no del bit 7, o un acarreo del bit 7 pero no del bit 6; en cualquier otro caso, OV se borra. Cuando se hace una suma de enteros con signo, OV indica que se produjo un número negativo como la suma de dos operandos positivos o que se produjo una suma positiva con dos operandos negativos.

Se permiten cuatro modos de direccionamiento a los operandos de origen: por registros, directo, indirecto por registros, o inmediato.

Ejemplo: El acumulador contiene el valor 0C3H (11000011B) y el registro 0 contiene el valor 0AAH (10101010B) con la bandera de acarreo activada. La instrucción,

```
ADDC A,R0
```

deja el valor 6EH (01101110B) en el acumulador con AC en cero, y activa tanto la bandera de acarreo como la OV con el valor de 1.

ADDC A,Rn

Bytes: 1
Ciclos: 1
Codificación: 00110rrr
Operación: (A) ← (A) + (C) + (Rn)

ADDC A,directa

Bytes: 2
Ciclos: 1
Codificación: 00110101 aaaaaaaa
Operación: (A) ← (A) + (C) + (directa)

ADDC A,@Ri

Bytes: 1
Ciclos: 1
Codificación: 0011011i
Operación: (A) ← (A) + (C) + (Ri)

ADDC A,#datos

Bytes: 2
Ciclos: 1
Codificación: 00110100 dddddddd
Operación: (A) ← (A) + (C) + #datos

AJMP dir11

Función: Salto absoluto

Descripción: AJMP transfiere la ejecución del programa a la dirección indicada, la cual se forma al tiempo de ejecución concatenando los cinco bits de orden superior del PC (*después* de incrementar dos veces al PC), los bits 7 a 5 del código de operación, y el segundo byte de la instrucción. El destino debe encontrarse, por lo tanto, dentro del mismo bloque de 2K de la memoria de programa que el primer byte de la instrucción que sigue de la instrucción AJMP.

Ejemplo: La etiqueta "SALTADIR" está en la ubicación en memoria de programa 0123H. La instrucción,

```
AJMP SALTADIR
```

está en la ubicación 0345H y carga al PC con el valor 0123H.

Bytes: 2

Ciclos: 2

Codificación: aaa00001 aaaaaaaa

Nota: aaa = A10−A8 y aaaaaaaa = A7−A0 de la dirección de destino.

Operación: (PC) ← (PC) + 2

(PC10−PC0) ← dirección de página

ANL <byte destino>,<byte origen>

Función: AND lógico para variables tipo byte

Descripción: ANL realiza una operación de AND lógico orientada a bits entre las variables indicadas, y almacena el resultado en la variable de destino. Ninguna bandera se ve afectada.

Los dos operandos permiten seis combinaciones para el modo de direccionamiento. Cuando el destino es el acumulador, el origen puede utilizar un direccionamiento por registros, directo, indirecto por registros, o inmediato; cuando el destino es una dirección directa, el origen puede ser el acumulador o datos inmediatos.

Nota: Cuando se utiliza esta instrucción para modificar un puerto de salida, el valor utilizado como el dato original del puerto se lee del latch de datos de salida, *no* se lee de las terminales de entrada.

Ejemplo: Si el acumulador contiene el valor 0C3H (11000011B) y el registro 0 contiene el valor 55H (010101011B), entonces la instrucción,

```
ANL A,R0
```

deja el valor 41H (01000001B) en el acumulador.

Cuando el destino es un byte direccionado directamente, la instrucción borra las combinaciones de bits en cualquier ubicación en RAM o registro en hardware. El byte de máscara determina si el patrón de los bits a borrar es una constante contenida en la instrucción o si es el valor calculado en el acumulador al tiempo de ejecución. La instrucción,

```
ANL P1,#01110011B
```

borra los bits 7, 3 y 2 del puerto 1 de salida.

ANL A,Rn

Bytes: 1
Ciclos: 1
Codificación: 01011rrr
Operación: (A) ← (A) AND (Rn)

ANL A,directa

Bytes: 2
Ciclos: 1
Codificación: 01010101 aaaaaaaa
Operación: (A) ← (A) AND (directa)

ANL A,@Ri

Bytes: 1
Ciclos: 1
Codificación: 0101011i
Operación: (A) ← (A) + (C) + ((Ri))

ANL A,#datos

Bytes: 2
Ciclos: 1
Codificación: 01010100 dddddddd
Operación: (A) ← (A) AND #datos

ANL directa,A

Bytes: 2

Ciclos: 1
Codificación: 01010010 aaaaaaaa
Operación: (directa) ← (directa) AND (A)

ANL directa,#datos

Bytes: 3
Ciclos: 2
Codificación: 01010011 aaaaaaaa dddddddd
Operación: (directa) ← (directa) AND #datos

ANL C,<bit origen>

Función: AND lógico para variables tipo bit

Descripción: Si el valor booleano del bit de origen es un 0 lógico, entonces borra la bandera de acarreo; de lo contrario, deja la bandera de acarreo en su estado actual. Un signo de división (/) incluido antes del operando en el programa en lenguaje ensamblador indica que se debe utilizar el complemento lógico del bit direccionado como el valor de origen, *pero no afecta al propio bit de origen*. Ninguna otra bandera se ve afectada.

Sólo se permite el direccionamiento directo para el operando origen.

Ejemplo: Activa la bandera de acarreo si, y sólo si, P1.0 = 1, ACC.7 = 1, y OV = 0:

```
MOV C,P1.0      ;CARGA C CON EL ESTADO DE LA
                 TERMINAL DE ENTRADA
ANL C,ACC.7     ;EFECTÚA UN AND LÓGICO DEL
                 ACARREO CON EL BIT 7 DEL ACC
ANL C,/OV       ;EFECTÚA UN AND LÓGICO CON EL
                 INVERSO DE LA BANDERA OV
```

ANL C,bit

Bytes: 2
Ciclos: 2
Codificación: 10000010 bbbbbbbb
Operación: (C) ← (C) AND (bit)

ANL C,/bit

Bytes: 2
Ciclos: 2
Codificación: 10110000 bbbbbbbb
Operación: (C) ← (C) AND NOT(bit)

CALL (Consulte ACALL o LCALL)

CJNE <byte destino>,<byte origen>,rel

Función: Compara y salta si no es igual

Descripción: CJNE compara las magnitudes de los primeros dos operandos y realiza una bifurcación si sus valores no son iguales. El destino de la bifurcación se calcula mediante la suma de desplazamiento relativo con signo en el último byte de la instrucción con el PC, después de incrementar el PC al inicio de la siguiente instrucción. La bandera de acarreo se activa si el valor entero sin signo de <byte destino> es menor que el valor entero sin signo de <byte origen>; de lo contrario, la bandera de acarreo se borra. Ningún operando se ve afectado.

Los primeros dos operandos permiten cuatro combinaciones de modo de direccionamiento: el acumulador puede compararse con cualquier byte direccionado directamente o con datos inmediatos, y cualquier ubicación indirecta en RAM o registro habilitado puede compararse con una constante inmediata.

Ejemplo: El acumulador contiene el valor 34H. El registro 7 contiene el valor 56H. La primera instrucción en la secuencia

```
                CJNER7,#60H,NO_ES_IGUAL
;               . . .  . . .            ;R7 = 60H
NO_ES_IGUAL:    JC REG_BAJO             ;IF R7 < 60H
;               . . .  . . .            ;R7 > 60H
REG_BAJO        . . .  . . .            ;R7 < 60H
```

activa la bandera de acarreo y se bifurca hacia la instrucción en la etiqueta NO_ES_IGUAL. Al probar la bandera de acarreo, esta instrucción determina si R7 es mayor o menor que 60H.

Si los datos presentados en el puerto 1 también equivalen a 34H, entonces la instrucción,

```
ESPERA:         CJNE A,P1,ESPERA
```

borra la bandera de acarreo y continúa con la siguiente instrucción, ya que el acumulador no es igual al dato leído del puerto 1. (Si algún otro valor se envía como entrada a P1, el programa entra en un ciclo hasta que el dato en P1 cambia su valor a 34H.)

CJNE A,directa,rel

Bytes: 3

Ciclos: 2

Codificación: 10110101 aaaaaaaa eeeeeeee

Operación: $(PC) \leftarrow (PC) + 3$

IF (A) <>(directa)
THEN
(PC) ← (PC) + dirección relativa
IF (A) <(directa)
THEN
(C) ← 1
ELSE
(C) ← 0

CJNE A,#datos,rel

Bytes: 3
Ciclos: 2
Codificación: 10110100 dddddddd eeeeeeee
Operación: (PC) ← (PC) + 3
IF(A) <>datos
THEN
(PC) ← (PC) + dirección relativa
IF(A)<datos
THEN
(C) ← 1
ELSE
(C) ← 0

CJNE Rn,#datos,rel

Bytes: 3
Ciclos: 2
Codificación: 10111rrr dddddddd eeeeeeee
Operación: (PC) ← (PC) + 3
IF (Rn) <>datos
THEN
(PC) ← (PC) + dirección relativa
IF (Rn) <datos
THEN
(C) ← 1
ELSE
(C) ← 0

CJNE @Ri,#datos,rel

Bytes: 3
Ciclos: 2
Codificación: 1011011i eeeeeeee
Operación: (PC) ← (PC) + 3
IF (Ri) <>datos
THEN
(PC) ← (PC) + dirección relativa
IF ((Ri)) <datos
THEN
(C) ← 1
ELSE
(C) ← −0

CLR A

Función: Borra acumulador
Descripción: El acumulador se borra (todos los bits se igualan a 0). Ninguna bandera se ve afectada.
Ejemplo: El acumulador contiene el valor 5CH (01011100B). La instrucción,

```
CLR         A
```

deja el acumulador igualado a 00H (00000000B).
Bytes: 1
Ciclos: 1
Codificación: 11100100
Operación: (A) ← 0

CLR bit

Función: Borra bit
Descripción: El bit indicado se borra (se reinicializa a 0). Ninguna otra bandera se ve afectada. CLR puede operar en la bandera de acarreo o en cualquier bit directamente direccionable.
Ejemplo: El puerto 1 contiene el valor 5DH (01011101B). La instrucción,

```
CLR P1.2
```

deja al puerto con el valor 59H (01011001B).

CLR C

Bytes: 1
Ciclos: 1
Codificación: 11000011
Operación: (C) ← 0

CLR bit

Bytes: 2
Ciclos: 1
Codificación: 11000010 bbbbbbbb
Operación: (bit) ← 0

CPL A

Función: Complementa al acumulador

Descripción: Cada bit del acumulador se complementa en forma lógica (complementos a 1). Los bits que antes contenían el valor 1 se cambian a 0, y viceversa. Ninguna bandera se ve afectada.

Ejemplo: El acumulador contiene el valor 5CH (01011100B). La instrucción

```
CPL     A
```

deja al acumulador con el valor 0A3H (10100011B).

Bytes: 1
Ciclos: 1
Codificación: 11110100
Operación: (A) ← NOT(A)

CPL bit

Función: Complementa bit

Descripción: La variable bit especificada se complementa. Un bit que era 1 se cambia a 0, y viceversa. Ninguna otra bandera se ve afectada. CLR puede operar sobre la bandera de acarreo o sobre cualquier bit directamente direccionable.

Nota: Cuando esta instrucción se utiliza para modificar una terminal de salida, el valor utilizado como el dato original proviene del latch de datos de salida, *no* de la terminal de entrada.

Ejemplo: El puerto 1 contiene el valor 5BH (01011011B). Las instrucciones,

```
CPL     P1.1
CPL     P1.2
```

dejan al puerto con el valor 5BH (01011011B).

CPL C

Bytes: 1
Ciclos: 1
Codificación: 10110011
Operación: (C) ← NOT(C)

CPL bit

Bytes: 2
Ciclos: 1
Codificación: 1010010 bbbbbbbb
Operación: (bit) ← NOT(bit)

DA A

Función: Ajuste decimal del acumulador para la suma

Descripción: DA A ajusta el valor de 8 bits en el acumulador resultante de una suma previa de dos variables (cada una en formato de BCD empaquetado), y produce dos dígitos de 4 bits. Se puede utilizar cualquier instrucción ADD o ADDC para realizar la suma.

Si los bits 3 a 0 del acumulador son mayores a 9 (xxxx1010-xxxx1111), o si la bandera AC es 1, se agrega un 6 al acumulador para producir el dígito BCD adecuado en el nibble de orden inferior. Esta suma interna activa la bandera de acarreo si un acarreo del campo de 4 bits de orden inferior se propagó a lo largo de todos los bits de orden superior, pero no borra la bandera de acarreo en el caso contrario.

Si la bandera de acarreo ahora está activada, o si los cuatro bits de orden superior ahora exceden a 9 (1010xxxx-1111xxxx), estos bits de orden superior se incrementan por 6, con lo cual se produce el dígito BCD adecuado en los bits de orden superior, pero no se borra el acarreo. Por lo tanto, la bandera de acarreo indica si la suma de las dos variables BCD originales es mayor a 99, ello permite obtener una suma decimal precisa. La bandera OV no resulta afectada.

Todo lo descrito líneas arriba ocurre durante un ciclo de instrucción. Esencialmente, esta instrucción realiza la conversión decimal al

agregar 00H, 06H, 60H, o 66H al acumulador, dependiendo de las condiciones iniciales en el acumulador y en la PSW.

Nota: DA A *no puede* simplemente convertir un número en hexadecimal que haya en el acumulador a notación BCD, y la instrucción DA A tampoco se aplica a la resta decimal.

Ejemplo: El acumulador contiene el valor 56H (01010110B), el cual representa los dígitos en el BCD empaquetado del número decimal 56. El registro 3 contiene el valor 67H (01100111B), el cual representa los dígitos en el BCD empaquetado del número decimal 67. La bandera de acarreo está activada. Las instrucciones,

```
ADDC A,R3
DA A
```

realizan primero una suma binaria de complementos a 2 estándar, lo que resulta en el valor OBEH (10111110B) en el acumulador. Las banderas de acarreo y de acarreo auxiliar se borran.

Después, la instrucción de ajuste decimal altera el acumulador al valor 24H (00100100B), indicando los dígitos en el BCD empaquetado del número decimal 24, los dos dígitos de orden inferior de la suma decimal de 56, 67, y el acarreo. La bandera de acarreo se activa mediante la instrucción de ajuste decimal, lo cual indica que ocurrió un desbordamiento decimal. La verdadera suma de 56, 67 y 1 es 124.

Las variables en BCD pueden incrementarse o disminuir agregando 01H o 99H. Si el acumulador contiene inicialmente 30H (lo que representa los dígitos 30 en decimal), entonces las instrucciones,

```
ADD     A,#99H
DA      A
```

dejan activada la bandera de acarreo y el valor 29H en el acumulador, ya que $30 + 99 = 129$. El byte de orden inferior de la suma puede interpretarse con el significado de $30 - 1 = 29$.

Bytes: 1

Ciclos: 1

Codificación: 11010100

Operación: (Suponga que en el acumulador los contenidos están en BCD.)

IF[[(A3 − A0) >9]AND [(AC = 1)]]

THEN (A3 − A0) ← (A3 − A0) + 6

AND

IF [[A7 − A4) >9]AND [(C) = 1]]

THEN (A7−A4) ← (A7 − A4) + 6

DEC BYTE

Función: Disminución

Descripción: La variable indicada se disminuye en 1. Un valor original de 00H provoca un desbordamiento negativo a 0FFH. Ninguna bandera se ve afectada. Se permiten cuatro modos de direccionamiento de operando: por acumulador, por registros, directo, o indirecto por registros.

Nota: Cuando esta instrucción se utiliza para modificar un puerto de salida, el valor utilizado como el dato del puerto original es del latch de datos de salida, *no* es de las terminales de entrada.

Ejemplo: El registro 0 contiene el valor 7FH (01111111B). Las ubicaciones en RAM interna 7EH y 7FH contienen los valores 00H y 40H, respectivamente. Las instrucciones,

```
DEC    @R0
DEC    R0
DEC    @R0
```

dejan al registro 0 con el valor 7EH y a las ubicaciones en RAM interna 7EH y 7FH con el valor 0FFH y 3FH.

DEC A

Bytes: 1

Ciclos: 1

Codificación: 00010100

Operación: $(A) \leftarrow (A) - 1$

DEC Rn

Bytes: 1

Ciclos: 1

Codificación: 00011rrr

Operación: $(Rn) \leftarrow (Rn) - 1$

DEC directa

Bytes: 2

Ciclos: 1

Codificación: 00010101 aaaaaaaa

Operación: $(\text{directa}) \leftarrow (\text{directa}) - 1$

DEC @Ri

Bytes: 1

Ciclos: 1

Codificación: 0001011i

Operación: $((Ri)) \leftarrow ((Ri)) - 1$

DIV AB

Función: División

Descripción: DIV AB divide un entero de 8 bits sin signo que haya en el acumulador entre un entero de 8 bits sin signo que esté en el registro B. El acumulador recibe la parte entera del cociente; el registro B recibe el resto del entero. Las banderas de acarreo y OV se borran.

Excepción: Si al principio B contenía el valor 00H, los valores regresados en el acumulador y en el registro B están indefinidos, y la bandera de desbordamiento se activa. La bandera de acarreo se borra en cualquier caso.

Ejemplo: El acumulador contiene el valor 251 (0FBH o 11111011B), y B contiene el valor 18 (12H o 00010010B). La instrucción,

```
DIV    AB
```

deja el valor 13 en el acumulador (0DH o 00001101B), y el valor 17 (11H o 00010000B) en el registro B, ya que $251 = 13 \times 18 + 17$. Las banderas de acarreo y la OV se borran.

Bytes: 1

Ciclos: 4

Codificación: 10000100

Operación: $(A) \leftarrow$ COCIENTE DE (A)/(B)

$(B) \leftarrow$ RESTO DE (A)/(B)

DJNZ<byte>,<dirección relativa>

Función: Disminuye y salta si no es igual a cero

Descripción: DJNZ disminuye la ubicación indicada por el primer operando y se bifurca a la dirección indicada por el segundo operando si el valor resultante no es igual a 0. Un valor original de 00H causa un desbordamiento negativo al valor 0FFH. Ninguna bandera se ve afectada. El destino de la bifurcación se calcula mediante una suma del valor de desplazamiento relativo con signo que haya en el último byte de la instrucción para el PC, después de incrementar el PC al primer byte de la siguiente instrucción.

La ubicación disminuida puede ser un registro o un byte directamente direccionado.

Nota: Cuando esta instrucción se utiliza para modificar un puerto de salida, el valor utilizado como el dato del puerto original se lee del latch de datos de salida, *no* de las terminales de entrada.

Ejemplo: Las ubicaciones en memoria RAM interna 40H, 50H, y 60H contienen los valores 01H, 70H, y 15H, respectivamente. Las instrucciones,

```
DJNZ 40H,ETIQUETA1
DJNZ 50H,ETIQUETA2
DJNZ 60H,ETIQUETA3
```

provocan un salto a la instrucción en ETIQUETA2 con los valores 00H, 6FH, y 15H en las 3 ubicaciones en RAM. El primer salto *no se realiza*, debido a que el resultado fue 0.

Esta instrucción proporciona una manera simple para ejecutar un ciclo de programa cierto número de veces, o para agregar un retraso moderado (de 2 a 512 ciclos de máquina) con una sola instrucción. Las instrucciones,

```
          MOV   R2,#8
DISPARA:  CPL   P1.7
          DJNZ  R2,DISPARA
```

disparan a P1.7 ocho veces, lo que genera cuatro pulsos de salida en el bit 7 del puerto 1. Cada pulso tiene duración de tres ciclos de máquina, dos para la instrucción DJNZ y uno más para alterar la terminal.

DJNZ Rn,rel

Bytes: 2
Ciclos: 2
Codificación: 11011rrr eeeeeeee
Operación: $(PC) \leftarrow (PC) + 2$
$(Rn) \leftarrow (Rn) - 1$
IF $(Rn)0$
THEN
$(PC) \leftarrow (PC) + byte\_2$

DJNZ directa,rel

Bytes: 3
Ciclos: 2

Codificación: 11010101 aaaaaaaa eeeeeeee

Operación: (PC) ← (PC) + 2
(directa) ← (directa) − 1
IF (directa) <>0
THEN
(PC) ← (PC) + byte_2

INC <byte>

Función: Incremento

Descripción: INC incrementa la variable indicada por 1. Un valor original de 0FFH causa un desbordamiento a 00H. Ninguna bandera se ve afectada. Se permiten tres modos de direccionamiento: por registros, directo, o indirecto por registros.

Nota: Cuando esta instrucción se utiliza para modificar un puerto de salida, el valor utilizado como el dato del puerto original es del latch de datos de salida, *no* de las terminales de entrada.

Ejemplo: El registro 0 contiene el valor 7EH (01111110B). Las ubicaciones en memoria RAM interna 7EH y 7FH contienen los valores 0FFH y 40H, respectivamente. Las instrucciones,

```
INC  @R0
INC  R0
INC  @R0
```

dejan al registro 0 con el valor de 7FH y a las ubicaciones en memoria RAM interna 7EH y 7FH con los valores de 00H y 41H, respectivamente.

INC A

Bytes: 1

Ciclos: 1

Codificación: 00000100

Operación: (A) ← (A) + 1

INC Rn

Bytes: 1

Ciclos: 1

Codificación: 00001rrr

Operación: (Rn) ← (Rn) + 1

INC directa

Bytes: 2
Ciclos: 1
Codificación: 00000101 aaaaaaaa
Operación: (directa) ← (directa) + 1

INC @Ri

Bytes: 1
Ciclos: 1
Codificación: 0000011i
Operación: ((Ri)) ← ((Ri)) + 1

INC DPTR

Función: Incrementa el apuntador de datos

Descripción: Incrementa el apuntador de datos de 16 bits en 1. Se realiza un incremento de 16 bits (módulo 2^{16}); un desbordamiento del byte de orden inferior del apuntador de datos (DPL) del valor 0FFH a 00H incrementa el byte de orden superior (DPH). Ninguna bandera se ve afectada.

Este es el único registro de 16 bits que puede incrementarse.

Ejemplo: Los registros DPH y DPL contienen los valores 12H y 0FEH, respectivamente. Las instrucciones,

```
INC DPTR
INC DPTR
INC DPTR
```

cambian DPH y DPL a 13H y 01H.

Bytes: 1
Ciclos: 2
Codificación: 10100011
Operación: (DPTR) ← (DPTR) + 1

JB bit, rel

Función: Salta si el bit está activado

Descripción: Si el bit indicado es igual a 1, la instrucción salta a la dirección indicada; de lo contrario, procede con la siguiente instrucción. El destino de la bifurcación se calcula mediante una suma del desplazamiento relativo con signo en el tercer byte de la instrucción con el PC, después de

incrementar el PC al primer byte de la siguiente instrucción. *El bit verificado no se modifica*. Ninguna bandera se ve afectada.

Ejemplo: El dato presente en el puerto 1 de entrada es 11001010B. El acumulador contiene el valor 56H (01010110B). Las instrucciones,

```
JB P1.2,ETIQUETA1
JB ACC.2,ETIQUETA2
```

provocan que la ejecución del programa se bifurque hacia la instrucción en ETIQUETA2.

Bytes: 3

Ciclos: 2

Codificación: 0100000 bbbbbbbb eeeeeeee

Operación: $(PC) \leftarrow (PC) + 3$

IF $(bit) + 1$

THEN

$(PC) \leftarrow (PC) + byte\_2$

JBC bit,rel

Función: Salta si el bit está activado y después borra el bit

Descripción: Si el bit indicado es igual a 1, la instrucción borra el bit y salta a la dirección indicada; de lo contrario, procede con la siguiente instrucción. *El bit no se borra si ya es igual a 0*. El destino de la bifurcación se calcula mediante una suma del desplazamiento relativo con signo en el tercer byte de la instrucción con el PC, después de incrementar el PC al primer byte de la siguiente instrucción. Ninguna bandera se ve afectada.

Nota: Cuando esta instrucción se utiliza para modificar un puerto de salida, el valor utilizado como el dato del puerto original se lee del latch de datos de salida, *no* de las terminales de entrada.

Ejemplo: El acumulador contiene el valor 56H (01010110B). Las instrucciones,

```
JBC ACC.3,ETIQUETA1
JBC ACC.2,ETIQUETA2
```

provocan que la ejecución del programa continúe en la instrucción identificada por ETIQUETA 2, y el acumulador se modifica al valor 52H (01010010B).

Bytes: 3

Ciclos: 2

Codificación: 00010000 bbbbbbbb eeeeeeee

Operación: (PC) ← (PC) + 3
IF(bit) = 1
THEN
(bit) ← 0
(PC) ← (PC) + byte_2

JC rel

Función: Salta si la bandera de acarreo está activada

Descripción: Si la bandera de acarreo está activada, la instrucción salta a la dirección indicada; de lo contrario, procede con la siguiente instrucción. El destino de la bifurcación se calcula mediante una suma del desplazamiento relativo con signo en el segundo byte de la instrucción con el PC, después de incrementar dos veces al PC. Ninguna bandera se ve afectada.

Ejemplo: La bandera de acarreo está en cero. Las instrucciones,

```
JC  ETIQUETA1
CPL C
JC  ETIQUETA2
```

activan la bandera de acarreo, y causan que la ejecución del programa continúe en la instrucción identificada por ETIQUETA2.

Bytes: 2

Ciclos: 2

Codificación: 01000000 eeeeeeee

Operación: (PC) ← (PC) + 2
IF (C) = 1
THEN
(PC) + (PC) + byte_2

JMP <destino> (Consulte SJMP, AJMP o LJMP)

JMP@A+DPTR

Función: Salto indirecto

Descripción: Suma el contenido sin signo de 8 bits del acumulador con el apuntador de 16 bits, y carga el resultado de la suma al contador de programa. Este resultado es la dirección para las subsiguientes búsquedas de instrucciones. Se realiza una suma de 16 bits (módulo 2^{16}): un acarreo de los ocho bits de orden inferior se propaga a lo largo de los bits de orden superior. No se alteran ni el acumulador ni el apuntador de datos. Ninguna bandera se ve afectada.

Ejemplo: El acumulador contiene un número par de 0 a 6. Las siguientes instrucciones generan una bifurcación a una de cuatro instrucciones AJMP en una tabla de saltos empezando en TBL_SALTO:

```
                MOV  DPTR,#TBL_SALTO
                JMP  @A + DPTR
TBL_SALTO:      AJMP ETIQUETA0
                AJMP ETIQUETA1
                AJMP ETIQUETA2
                AJMP ETIQUETA3
```

Si el acumulador es igual a 04H cuando inicia esta secuencia, la ejecución del programa se bifurca a ETIQUETA2. Recuerde que AJMP es una instrucción de dos bytes, así que la instrucción de salto empieza en una dirección, ignora a otra, continúa en la siguiente, y así sucesivamente.

Bytes: 1

Ciclos: 2

Codificación: 01110011

Operación: (PC) ← (PC) + (A) + (DPTR)

JNB bit,rel

Función: Salta si el bit no está activado

Descripción: Si el bit indicado es igual a 0, se genera una bifurcación a la dirección indicada; de lo contrario, procede con la siguiente instrucción. El destino de la bifurcación se calcula mediante una suma del desplazamiento relativo con signo en el tercer byte de la instrucción con el PC, después de incrementar el PC al primer byte de la siguiente instrucción. *El bit verificado no se modifica*. Ninguna bandera se ve afectada.

Ejemplo: Los datos presentes en el puerto 1 de entrada son 110010108. El acumulador contiene el valor 56H (01010110B). Las instrucciones,

```
JNB P1.3,ETIQUETA1
JNB ACC.3,ETIQUETA2
```

provocan que la ejecución del programa continúe en la instrucción en ETIQUETA2.

Bytes: 3

Ciclos: 2

Codificación: 00110000 bbbbbbbb eeeeeeee

Operación: (PC) ← (PC) + 3

IF (bit) = 0

THEN

(PC) ← (PC) + byte_2

JNC rel

Función: Salta si la bandera de acarreo no está activada

Descripción: Si la bandera de acarreo es igual a 0, genera una bifurcación a la dirección indicada; de lo contrario, procede con la siguiente instrucción. El destino de la bifurcación se calcula mediante una suma del desplazamiento relativo con signo en el segundo byte de la instrucción con el PC, después de incrementar dos veces al PC para que apunte hacia la siguiente instrucción. La bandera de acarreo no se modifica.

Ejemplo: La bandera de acarreo está activada. Las instrucciones,

```
JNC ETIQUETA1
CPL C
JNC ETIQUETA2
```

borran la bandera de acarreo y causan que la ejecución del programa continúe en la instrucción identificada mediante ETIQUETA2.

Bytes: 2

Ciclos: 2

Codificación: 01010000 eeeeeeee

Operación: $(PC) \leftarrow (PC) + 2$

IF $(C) = 0$

THEN

$(PC) \leftarrow (PC) + byte\_2$

JNZ rel

Función: Salta si el acumulador no es igual a cero

Descripción: Si cualquier bit del acumulador es igual a 1, genera una bifurcación a la dirección indicada; de lo contrario procede con la siguiente instrución. El destino de la bifurcación se calcula mediante la suma del desplazamiento relativo con signo en el segundo byte de la instrucción con el PC, después de incrementar dos veces al PC. El acumulador no se modifica. Ninguna bandera se ve afectada.

Ejemplo: El acumulador originalmente contiene el valor 00H. Las instrucciones,

```
JNZ   ETIQUETA1
INC   A
JNZ   ETIQUETA2
```

igualan el acumulador a 01H y continúan en ETIQUETA2.

Bytes: 2

Ciclos: 2

Codificación: 01110000 eeeeeeee

Operación: (PC) ← (PC) + 2
IF (A) <>0
THEN
(PC) ← (PC) + byte_2

JZ rel

Función: Salta si el acumulador es igual a cero

Descripción: Si todos los bits del acumulador son iguales a 0, genera una bifurcación a la dirección indicada; de lo contrario procede con la siguiente instrucción. El destino de la bifurcación se calcula mediante una suma del desplazamiento relativo con signo en el segundo byte de la instrucción con el PC, después de incrementar dos veces al PC. El acumulador no se modifica. Ninguna bandera se ve afectada.

Ejemplo: Al principio, el acumulador contiene el valor 01H. Las instrucciones,

```
JZ  ETIQUETA1
DEC A
JZ  ETIQUETA2
```

cambian al acumulador al valor 00H y causan que la ejecución del programa continúe en la instrucción identificada mediante ETIQUETA2.

Bytes: 2

Ciclos: 2

Codificación: 01100000 eeeeeeee

Operación: (PC) ← (PC) + 2
IF (A) = 0
THEN
(PC) ← (PC) + byte_2

LCALL dir16

Función: Llamada larga a la subrutina

Descripción: LCALL llama a una subrutina que se ubica en la dirección indicada. La instrucción agrega el valor 3 al contador de programa para generar la dirección de la siguiente instrucción y entonces almacena el resultado de 16 bits en la pila (byte inferior primero), e incrementa el apuntador de pila por 2. Los bytes de orden superior e inferior del PC se cargan, respectivamente, con los bytes segundo y tercero de la instrucción LCALL. La ejecución del programa continúa con la instrucción en dicha dirección. La subrutina puede, por lo tanto, iniciar en cualquier parte de los 64K-byte de espacio de direcciones de memoria de programa. Ninguna bandera se ve afectada.

Ejemplo: Al principio, el apuntador de pila es igual a 07H. La etiqueta "SUBRTN" se asigna a la ubicación en memoria de programa 1234H. Después de ejecutar la instrucción,

```
LCALL     SUBRTN
```

en la ubicación 0123H, el apuntador de pila contiene el valor 09H, las ubicaciones en memoria RAM interna 08H y 09H contienen los valores 26H y 01H, respectivamente, y el PC contiene el valor 1234H.

Bytes: 3

Ciclos: 2

Codificación: 00010010 aaaaaaaa aaaaaaaa

Nota: El segundo byte contiene los bits de dirección 15 a 8, y el tercer byte contiene los bits de dirección 7 a 0.

Operación: $(PC) \leftarrow (PC) + 3$

$(SP) \leftarrow (SP) + 1$

$(SP) \leftarrow (PC7-PC0)$

$(SP) \leftarrow (SP) + 1$

$((SP)) \leftarrow (PC15-PC8)$

$(PC) \leftarrow dir15-dir0$

LJMP dir16

Función: Salto largo

Descripción: LJMP carga los bytes de orden superior e inferior del PC (respectivamente) con los byte segundo y tercero de la instrucción, y con esto provoca una bifurcación incondicional hacia la dirección indicada. El destino, por lo tanto, puede estar en cualquier parte de los 64K del espacio de direcciones de memoria de programa. Ninguna bandera se ve afectada.

Ejemplo: La etiqueta "DIRSALTO" se asigna a la instrucción en la ubicación 1234H de la memoria de programa. La instrucción,

```
LJMP   DIRSALTO
```

en la ubicación 0123H carga el contador de programa con el valor 1234H.

Bytes: 3

Ciclos: 2

Codificación: 00010010 aaaaaaaa aaaaaaaa

Nota: El segundo byte contiene los bits de dirección 15 a 8, y el tercer byte contiene los bits de dirección 7 a 0.

Operación: $(PC) \leftarrow dir15-dir0$

MOV<byte destino>,<byte origen>

Función: Transfiere una variable de byte

Descripción: La variable de byte indicada por el segundo operando se copia a la ubicación especificada por el primer operando. El byte de origen no se ve afectado. Ningún registro ni bandera se ven afectados.

Esta es definitivamente la operación más flexible. Se permiten quince combinaciones de modo de direccionamiento para el origen y el destino.

Ejemplo: La ubicación en RAM interna 30H contiene el valor 40H. El valor de la ubicación 40H en RAM es 10H. El dato presente en el puerto 1 de entrada es 11001010B (0CAH). Las instrucciones,

```
MOV R0,#30H      ;R0 ← 30H
MOV A,@R0        ;A ← 40H
MOV R1,A;        ;R1 ← 40H
MOV B,@R1        ;B ← 10H
MOV @R1,P1       ;RAM (40H)
                 ; ← 0CAH
MOV P2,P1        ;P2 ← 0CAH
```

dejan los valores 30H en el registro 0, 40H tanto en el acumulador como en el registro 1, 10H en el registro B, y 0CAH (11001010B) en la ubicación 40H en RAM y en el puerto 2 de salida.

MOV A,Rn

Bytes: 1

Ciclos: 1

Codificación: 11101rrr

Operación: (A) ← (Rn)

MOV A,directa

Bytes: 2

Ciclos: 1

Codificación: 11100101 aaaaaaaa

Operación: (A) ← (directa)

Nota: MOV A,ACC no es una instrucción válida.

MOV A,@Ri

Bytes: 1

Ciclos: 1

Codificación: 1110011i
Operación: (A) ← ((Ri))

MOV A,#datos

Bytes: 2
Ciclos: 1
Codificación: 01110100 dddddddd
Operación: (A) ← #datos

MOV Rn,A

Bytes: 1
Ciclos: 1
Codificación: 11111rrr
Operación: (Rn) ← (A)

MOV Rn,directa

Bytes: 2
Ciclos: 2
Codificación: 10101rrr
Operación: (Rn) ← (directa)

MOV Rn,#datos

Bytes: 2
Ciclos: 1
Codificación: 01111rrr dddddddd
Operación: (Rn) ← #datos

MOV directa,A

Bytes: 2
Ciclos: 1
Codificación: 11110101 aaaaaaaa
Operación: (directa) ← A

MOV directa,Rn

Bytes:	2
Ciclos:	2
Codificación:	10001rrr aaaaaaaa
Operación:	(directa) ← (Rn)

MOV directa,directa

Bytes:	3
Ciclos:	2
Codificación:	10000101 aaaaaaaa aaaaaaaa
	Nota: El segundo byte contiene la dirección de origen; el tercer byte contiene la dirección destino.
Operación:	(directa) ← (directa)

MOV directa,@Ri

Bytes:	2
Ciclos:	2
Codificación:	1000011i aaaaaaaa
Operación:	(directa) ← ((Ri))

MOV directa,#datos

Bytes:	3
Ciclos:	2
Codificación:	01110101 aaaaaaaa dddddddd
Operación:	(directa) ← #datos

MOV @Ri,A

Bytes:	1
Ciclos:	1
Codificación:	1111011i
Operación:	((Ri)) ← A

MOV @Ri,directa

Bytes: 2
Ciclos: 2
Codificación: 1010011i aaaaaaaa
Operación: ((Ri)) ← (directa)

MOV @Ri,#datos

Bytes: 2
Ciclos: 1
Codificación: 0111011i dddddddd
Operación: ((Ri)) ← #datos

MOV <bit destino>,<bit origen>

Función: Transfiere variable de bit

Descripción: La variable booleana indicada por el segundo operando se copia a la ubicación especificada por el primer operando. Uno de los operandos debe ser la bandera de acarreo; el otro puede ser cualquier bit directamente direccionable. Ningún otro registro o bandera se ven afectados.

Ejemplo: La bandera de acarreo está activada originalmente. El dato presente en el puerto 2 de entrada es 11000101B. El dato previamente enviado al puerto 1 de salida es el valor 35H (00110101B). Las instrucciones,

```
MOV P1.3,C
MOV C,P3.3
MOV P1.2,C
```

dejan limpio al acarreo y cambian el puerto 1 al valor 39H (00111001B).

MOV C,bit

Bytes: 2
Ciclos: 1
Codificación: 10100010 bbbbbbbb
Operación: (C) ← (bit)

MOV bit,C

Bytes: 2
Ciclos: 2

Codificación: 10010010 dirección de bit
Operación: (bit) ← (C)

MOV DPTR,#datos16

Función: Carga el apuntador de datos con una constante de 16 bits

Descripción: El apuntador de datos se carga con la constante de 16 bits indicada. La constante de 16 bits se ubica en los bytes segundo y tercero de la instrucción. El segundo byte (DPH) es el byte de orden superior, mientras que el tercer byte (DPL) contiene el byte de orden inferior. Ninguna bandera se ve afectada.

Ejemplo: La instrucción,

```
MOV DPTR,#1234H
```

carga el valor 1234H al apuntador de datos: DPH contiene el valor 12H y DPL contiene el valor 34H.

Bytes: 3

Ciclos: 2

Codificación: 10010000 dddddddd dddddddd

Nota: El segundo byte contiene los bits de datos inmediatos 15 a 8, y el byte 3 contiene los bits 7 a 0.

Operación: (DPTR) ← #datos16

MOVC A,@A+<registro base>

Función: Transfiere un byte de código o un byte constante

Descripción: Las instrucciones MOVC cargan al acumulador con un byte de código o un byte constante de la memoria de programa. La dirección del byte obtenido es la suma del contenido original de 8 bits sin signo en el acumulador y del contenido de un registro base de 16 bits, el cual puede ser el apuntador de datos o el PC. En el segundo caso, el PC se incrementa a la dirección de la siguiente instrucción antes de agregarlo al acumulador; en caso contrario, el registro base no se altera. Se realiza una suma de 16 bits para que el acarreo de los 8 bits de orden inferior pueda propagarse a lo largo de los bits de orden superior. Ninguna bandera se ve afectada.

Ejemplo: El acumulador contiene un valor ubicado entre 0 y 3. La siguiente subrutina convierte el valor en el acumulador a 1 de 4 valores definidos por la directiva DB (definición de byte).

```
REL_PC:         INC A
                MOVC A,@A + PC
                RET
                DB 66H
                DB 77H
```

```
DB 88H
DB 99H
```

Si la subrutina se llama con el acumulador igual a 01H, regresa con el valor 77H en el acumulador. Se necesita la instrucción INC A antes de la instrucción MOVC para "sobrepasar" la instrucción RET por encima de la tabla. Si varios bytes de código separan la instrucción MOV de la tabla, el número correspondiente debe agregarse al acumulador.

MOVC A,@A+DPTR

Bytes: 1
Ciclos: 2
Codificación: 10010011
Operación: $(A) \leftarrow ((A) + (DPTR))$

MOVC A,@A+PC

Bytes: 1
Ciclos: 2
Codificación: 10000011
Operación: $(PC) \leftarrow (PC) + 1$
$(A) \leftarrow ((A) + (PC))$

MOVX <byte destino>,<byte origen>

Función: Transferencia externa

Descripción: Las instrucciones MOVX transfieren datos entre el acumulador y un byte de memoria externa para datos; es por esto que se agrega una "X" al término MOV. Existen dos tipos de instrucciones, y la única diferencia entre ellas es si proporcionan una dirección indirecta de 8 o de 16 bits a la memoria RAM externa para datos.

En el primer tipo de instrucción, el contenido de R0 o R1 en el banco de registros actual proporciona una dirección de 8 bits multiplexada con los datos en P0. Ocho bits son suficientes para implementar la decodificación de expansión de E/S externa o para un arreglo relativamente pequeño en memoria RAM. Podemos utilizar cualquiera de las terminales de puerto de salida en arreglos un poco más grandes para enviar los bits de dirección de orden superior como salida. Estas terminales estarían controladas por una instrucción de salida procediendo a la instrucción MOVX.

En el segundo tipo de instrucción MOVX, el apuntador de datos genera una dirección de 16 bits. P2 envía como salida los ocho bits de dirección de orden superior (el contenido de DPH), mientras que P0 multiplexa los

ocho bits de orden inferior (DPL) con los datos. El registro con funciones especiales P2 retiene su contenido anterior, mientras que los búferes de salida de P2 emiten el contenido de DPH. Esta forma resulta más rápida y eficiente cuando accedemos a arreglos muy grandes (de hasta 64K bytes), ya que no necesitamos de ninguna instrucción adicional adicional para configurar los puertos de salida.

Es posible que en ciertas situaciones se puedan mezclar los dos tipos de la instrucción MOVX. Un arreglo en RAM grande, en donde P2 controla las líneas de orden superior de su dirección, puede direccionarse mediante el apuntador de datos o mediante código para enviar los bits de dirección de orden superior a P2 como salida, seguido de una instrucción MOVX utilizando R0 o R1.

Ejemplo: Una memoria RAM externa de 256 bytes utilizando líneas de dirección y de datos multiplexadas (por ejemplo, un 8155 RAM/E/S/TEMPORIZADOR de Intel) está conectada al puerto 0 del 8051. El puerto 3 proporciona las líneas de control para la RAM externa. Los puertos 1 y 2 se utilizan para E/S normal. Los registros 0 y 1 contienen los valores 12H y 34H. La ubicación 34H en la RAM externa contiene el valor 56H. La secuencia de instrucciones,

```
MOVX A,@R1
MOVX @R0,A
```

copia el valor 56H tanto al acumulador como a la ubicación 12H en memoria RAM externa.

MOVX A,@Ri

Bytes: 1
Ciclos: 2
Codificación: 1110001i
Operación: (A) ← ((Ri))

MOVX A,@DPTR

Bytes: 1
Ciclos: 2
Codificación: 11100000
Operación: (A) ← ((DPTR))

MOVX @Ri,A

Bytes: 1
Ciclos: 2
Codificación: 11110011
Operación: ((Ri)) ← (A)

MOVX @DPTR,A

Bytes: 1
Ciclos: 2
Codificación: 11110000
Operación: (DPTR) ← (A)

MUL AB

Función: Multiplicación

Descripción: MUL AB multiplica los enteros sin signo de 8 bits en el acumulador y en el registro B. El byte de orden inferior del producto de 16 bits queda en el acumulador y el byte de orden superior queda en el registro B. Si el producto es mayor a 255 (0FFH), la bandera de desbordamiento se activa; ésta se borra en el caso contrario. La bandera de acarreo siempre se borra.

Ejemplo: Al principio, el acumulador contiene el valor 80 (50H). El registro B contiene el valor 160 (0A0H). La instrucción,

```
MUL AB
```

resulta en un producto de 12,800 (3200H), así que B cambia a 32H (00110010B) y el acumulador se borra. La bandera de desbordamiento se activa, y el acarreo se borra.

Bytes: 1
Ciclos: 4
Codificación: 10100100
Operación: (B) ← BYTE SUPERIOR DE (A) × (B)
(A) ← BYTE INFERIOR DE (A) × (B)

NOP

Función: No hay operación

Descripción: La ejecución continúa en la siguiente instrucción. Ningún registro o bandera, a excepción del PC, se ven afectados.

Ejemplo: En el bit 7 del puerto 2, deseamos producir un pulso de salida a nivel bajo que dure exactamente cinco ciclos. Una secuencia simple de SETB/CLR genera un pulso de un ciclo, así que debemos insertar cuatro ciclos adicionales. Esto puede llevarse a cabo (si suponemos que las interrupciones no están habilitadas) con las instrucciones,

```
CLR P2.7
NOP
```

```
NOP
NOP
NOP
SETB P2.7
```

Bytes: 1

Ciclos: 1

Codificación: 00000000

Operación: (PC) ← (PC) + 1

ORL <byte destino>,<byte origen>

Función: OR lógico para variables de byte

Descripción: ORL realiza una operación de OR lógico orientada a bits entre las variables indicadas y almacena el resultado en el byte de destino. Ninguna bandera se ve afectada.

Los dos operandos permiten seis combinaciones de modo de direccionamiento. Cuando el destino es el acumulador, el origen puede utilizar un direccionamiento por registros, directo, indirecto por registros, o inmediato; cuando el destino es una dirección directa, el origen puede ser el acumulador o datos inmediatos.

Nota: Cuando esta instrucción se utiliza para modificar un puerto de salida, el valor utilizado como el dato del puerto original se lee del latch de datos de salida, *no* de las terminales de entrada.

Ejemplo: Si el acumulador contiene el valor 0C3H (11000011B) y R0 contiene el valor 55H (01010101) entonces la instrucción,

```
ORL A,R0
```

deja el valor 0D7H (11010111B) en el acumulador. Cuando el destino es un byte directamente direccionable, la instrucción puede activar combinaciones de bits en cualquier ubicación en RAM o registro en hardware. El patrón de bits a activar está determinado por un byte de enmascarado que puede ser ya sea un valor de datos constante en la instrucción o una variable calculada en el acumulador al tiempo de ejecución. La instrucción,

```
ORL P1,00110010B
```

activa los bits 5, 4 y 1 del puerto 1 de salida.

ORL A,Rn

Bytes: 1

Ciclos: 1

Codificación: 01001rrr
Operación: (A) ← (A) OR (Rn)

ORL A,direct

Bytes: 2
Ciclos: 1
Codificación: 01000101 aaaaaaaa
Operación: (A) ← (A) OR (direct)

ORL A,@Ri

Bytes: 1
Ciclos: 1
Codificación: 0100011i
Operación: (A) ← (A) OR ((Ri))

ORL A,#datos

Bytes: 2
Ciclos: 1
Codificación: 01000100 dddddddd
Operación: (A) ← (A) OR #datos

ORL directa,A

Bytes: 2
Ciclos: 1
Codificación: 01000010 aaaaaaaa
Operación: (directa) ← (directa) OR (A)00

ORL directa,#datos

Bytes: 3
Ciclos: 2
Codificación: 01000011 aaaaaaaa dddddddd
Operación: (directa) ← (directa) OR #datos

ORL C,<bit origen>

Función: OR lógico para variables de bit

Descripción: Activa la bandera de acarreo si el valor booleano es un 1 lógico; en caso contrario, deja el acarreo en su estado actual. Un signo de división (/) precediendo al operando en el lenguaje ensamblador indica que se utiliza el complemento lógico del bit direccionado como el valor de origen, pero el bit de origen no se ve afectado. Ninguna otra bandera se ve afectada.

Ejemplo: Activa la bandera de acarreo si, y sólo si, P1.0 = 1, ACC.7 = 1, u OV = 0.

```
MOV C,P1.0    ;CARGA CY CON LA TERMINAL DE
              ENTRADA P1.0
ORL C,ACC.7   ;OR DE CY CON EL BIT 7 DEL ACC
ORL C,/OV     ;OR DE CY CON EL INVERSO DE OV
```

ORL C,bit

Bytes: 2

Ciclos: 2

Codificación: 01110010 bbbbbbbb

Operación: $(C) \leftarrow -(C)$ OR (bit)

ORL C,/bit

Bytes: 2

Ciclos: 2

Codificación: 10100000 bbbbbbbb

Operación: $(C) \leftarrow (C)$ OR NOT(bit)

POP directa

Función: Recuperación de la pila

Descripción: Se lee el contenido de la ubicación en RAM interna direccionada por el apuntador de pila, y el apuntador de pila se disminuye por 1. Después el valor leído se transfiere al byte directamente direccionado indicado. Ninguna bandera se ve afectada.

Ejemplo: El apuntador de pila originalmente contiene el valor 32H, y las ubicaciones en RAM interna 30H hasta la 32H contienen los valores 20H, 23H, y 01H, respectivamente. Las instrucciones,

```
POP     DPH
POP     DPL
```

dejan el apuntador de pila igual al valor 30H y el apuntador de datos en 0123H. En este punto, la instrucción,

```
POP     SP
```

deja el apuntador de pila establecido en 20H. Observe que, en este caso especial, el apuntador de pila se decrementa a 2FH antes de cargarse con el valor que se sacó (20H).

Bytes: 2

Ciclos: 2

Codificación: 11010000 aaaaaaaa

Operación: (directa) ← ((SP))

(SP) ← (SP) − 1

PUSH directa

Función: Meter datos en la pila

Descripción: El apuntador de pila se incrementa por 1. El contenido de la variable indicada se copia entonces en la ubicación de RAM interna direccionada por el apuntador de pila. En caso contrario no se afectan las banderas.

Ejemplo: Al entrar en una rutina de interrupción, el apuntador de pila contiene el valor 09H. El apuntador de datos mantiene el valor 0123H. Las instrucciones,

```
PUSH    DPL
PUSH    DPH
```

dejan el valor 0BH en el apuntador de pila y almacenan los valores 23H y 01H en las ubicaciones en RAM interna 0AH y 0BH, respectivamente.

Bytes: 2

Ciclos: 2

Codificación: 11000000 aaaaaaaa

Operación: (SP) ← (SP) + 1

((SP)) ← (directa)

RET

Función: Regresar de la subrutina

Descripción: RET recupera los bytes de orden superior e inferior del PC en forma sucesiva de la pila, y disminuye el apuntador de pila en 2. La ejecución del programa continúa en la dirección resultante, la cual es por lo general la instrucción que va inmediatamente después de una instrucción ACALL o LCALL. Ninguna bandera se ve afectada.

Ejemplo: Al principio, el apuntador de pila contiene el valor 0BH. Las ubicaciones en RAM interna 0AH y 0BH contienen los valores 23H y 01H, respectivamente. La instrucción,

```
RET
```

deja el valor 09H en el apuntador de pila. La ejecución del programa continúa en la ubicación 0123H

Bytes: 1

Ciclos: 2

Codificación: 00100010

Operación: $(PC1-PC8) \leftarrow ((SP))$

$(SP) \leftarrow (SP) - 1$

$(PC7-PC0) \leftarrow ((SP))$

$(SP) \leftarrow (SP) - 1$

RETI

Función: Regresar de una interrupción

Descripción: RETI recupera los bytes de orden superior e inferior del PC en forma sucesiva de la pila, y restablece la lógica de interrupciones para aceptar interrupciones adicionales al mismo nivel de prioridad que el que se acaba de procesar. El apuntador de pila disminuye en 2. Ningún otro registro se ve afectado; la PSW *no* se restablece automáticamente al estado que tenía antes de la interrupción. La ejecución del programa continúa en la dirección resultante, la cual es por lo general una instrucción que va inmediatamente después del punto en donde se detecta la petición de interrupción. Si una interrupción del mismo nivel (o de nivel más bajo) está pendiente cuando la instrucción RETI se ejecuta, entonces una instrucción se ejecuta antes de que la interrupción pendiente sea procesada.

Ejemplo: Al principio, el apuntador de pila contiene el valor 0BH. Se detecta una interrupción durante la instrucción que termina en la ubicación 0123H. Las ubicaciones en RAM interna 0AH y 0BH contienen los valores 23H y 01H, respectivamente. La instrucción,

```
RETI
```

deja el valor 09H en el apuntador de pila y regresa la ejecución del programa a la ubicación 0123H.

Bytes: 1

Ciclos: 2

Codificación: 00110010

Operación: $(PC15-PC8) \leftarrow ((SP))$

(SP) ← (SP) − 1
(PC7−PCO) ← ((SP))
(SP) ← (SP) − 1

RL A

Función: Desplaza el acumulador hacia la izquierda

Descripción: En el acumulador los ocho bits son desplazados un bit hacia la izquierda. El bit 7 se desplaza hacia la posición del bit 0. Ninguna otra bandera es afectada.

Ejemplo: El acumulador contiene el valor 0C5H (11000101B). La instrucción,

```
RL      A
```

deja el valor 8BH (10001011B) en el acumulador sin afectar al acarreo.

Bytes: 1

Ciclos: 1

Codificación: 00100011

Operación: (An + 1) ← (An), n = 0−6
(A0) ← (A7)

RLC A

Función: Desplaza el acumulador hacia la izquierda a través de la bandera de acarreo

Descripción: Los ocho bits en el acumulador y la bandera de acarreo se acarrean juntos un bit hacia la izquierda. El bit 7 se transfiere a la bandera de acarreo; el estado original de la bandera de acarreo se transfiere a la posición del bit 0. Ninguna otra bandera se ve afectada.

Ejemplo: El acumulador contiene el valor 0C5H (11000101B), y el acarreo es igual a 0. La instrucción,

```
RLC A
```

deja el valor 8BH (10001011B) en el acumulador con el acarreo activado.

Bytes: 1

Ciclos: 1

Codificación: 00110011

Operación: (An + 1) ← (An), n = 0−6

(A0) ← (C)
(C) ← (A7)

RR A

Función:	Desplaza el acumulador hacia la derecha
Descripción:	Los ocho bits en el acumulador se desplazan un bit hacia la derecha. El bit 0 se desplaza hacia la posición del bit 7. Ninguna bandera se ve afectada.
Ejemplo:	El acumulador contiene el valor 0C5H (11000101B) sin afectar el acarreo. La instrucción, `RR A` deja el valor 0E2H (11100010B) en el acumulador sin afectar al acarreo.
Bytes:	1
Ciclos:	1
Codificación:	00000011
Operación:	(An) ← (An + 1), n = 0 − 6 (A7) ← (A0)

RRC A

Función:	Desplaza el acumulador hacia la derecha a través de la bandera de acarreo
Descripción:	Los ocho bits en el acumulador y la bandera de acarreo se desplazan juntos un bit hacia la derecha. El bit 0 se transfiere a la bandera de acarreo; el valor original de la bandera de acarreo se transfiere a la posición del bit 7. Ninguna otra bandera se ve afectada.
Ejemplo:	El acumulador contiene el valor 0C5H (11000101B), y el acarreo es igual a 0. La instrucción, `RRC A` deja el valor 62H (01100010B) en el acumulador y activa el acarreo.
Bytes:	1
Ciclos:	1
Codificación:	00010011
Operación:	(An) ← (An + 1), n = 0 − 6 (A7) ← (C) (C) ← (A0)

SETB <bit>

Función: Activa un bit

Descripción: SETB activa el bit indicado al valor 1. SETB puede operar en la bandera de acarreo o en cualquier bit directamente direccionable. Ninguna otra bandera se ve afectada.

Ejemplo: La bandera de acarreo está en cero. El puerto 1 de salida contiene el valor 34H (00110100B). Las instrucciones,

```
SETB C
SETB P1.0
```

dejan la bandera de acarreo igual a 1 y cambian la salida de datos en el puerto 1 a 35H (00110101B).

SETB C

Bytes: 1

Ciclos: 1

Codificación: 11010011

Operación: (C) ← 1

SETB bit

Bytes: 2

Ciclos: 1

Codificación: 11010010 bbbbbbbb

Operación: (bit) ← 1

SJMP rel

Función: Salto corto

Descripción: El control del programa se bifurca incondicionalmente a la dirección indicada. El destino de la bifurcación se calcula mediante una suma del desplazamiento con signo en el segundo byte de la instrucción con el PC, después de incrementar dos veces el PC. Por lo tanto, el rango de destinos permitido es desde los 128 bytes precediendo a esta instrucción hasta los 127 bytes que le siguen.

Ejemplo: La etiqueta “DIRREL” se asigna a una instrucción en la ubicación en memoria de programa 0123H. La instrucción,

```
SJMP RELADR
```

se ensambla a la ubicación 0100H. Después de que se ejecuta la instrucción, el PC contiene el valor 0123H.

(*Nota:* Bajo las condiciones antes descritas, la instrucción que sigue de la instrucción SJMP está en la ubicación 0102H. Por lo tanto, el byte de desplazamiento de la instrucción es el desplazamiento relativo (0123H – 0102H = 21H). Descrito de otra manera, una instrucción SJMP con un desplazamiento de 0FEH es un ciclo infinito de una sola instrucción.

Bytes: 2

Ciclos: 2

Codificación: 10000000 eeeeeeee

Operación: (PC) ← (PC) + 2

(PC) ← (PC) + byte_2

SUBB A,<byte origen>

Función: Suma con préstamo

Descripción: SUBB resta la variable indicada y la bandera de acarreo del acumulador, y deja el resultado en el acumulador. SUBB activa la bandera de acarreo (préstamo) si se requiere de un préstamo para el bit 7 y borra C en caso contrario. (Si C se activa *antes* de ejecutar una instrucción SUBB, esto indica que el préstamo es necesario para el paso anterior en una resta de precisión múltiple, así que el acarreo se resta del acumulador junto con el operando de origen). AC se activa si se necesita un préstamo para el bit 3 y se borra en caso contrario. OV se activa cuando se necesita un préstamo hacia el bit 6 pero no hacia el bit 7, o hacia el bit 7 pero no hacia el bit 6.

Cuando se restan enteros con signo, la bandera OV indica que se produjo un número negativo cuando un valor negativo se resta de un valor positivo, o que se produjo un número positivo cuando un número positivo se resta de un número negativo.

El operando de origen permite cuatro modos de direccionamiento: por registro, directo, indirecto por registro, o inmediato.

Ejemplo: El acumulador contiene el valor 0C9H (11001001B), el registro 2 contiene el valor 54H (01010100B), y la bandera de acarreo está activada. La instrucción,

```
SUBB A,R2
```

deja el valor 74H (01110100B) en el acumulador, y borra las banderas de acarreo y de AC pero deja activada la bandera OV.

Observe que 0C9H menos 54H es 75H. La diferencia entre este resultado y el anterior se debe a que la bandera de acarreo (préstamo) se activó antes de la operación. Si no se conoce el estado del acarreo antes de iniciar una resta de precisión simple o múltiple, el acarreo debe borrarse con una instrucción CLR C.

SUBB A,Rn

Bytes: 1
Ciclos: 1
Codificación: 10011rrr
Operación: (A) ← (A) − (C) − (Rn)

SUBB A,direct

Bytes: 2
Ciclos: 1
Codificación: 10010101 aaaaaaaa
Operación: (A) ← (A) − (C) − (directa)

SUBB A,@Ri

Bytes: 1
Ciclos: 1
Codificación: 1001011i
Operación: (A) ← (A) − (C) − ((Ri))

SUBB A,#datos

Bytes: 2
Ciclos: 1
Codificación: 10010100 dddddddd
Operación: (A) ← (A) − (C) − #datos

SWAP A

Función: Intercambia los nibbles dentro del acumulador

Descripción: SWAP A intercambia los nibbles de orden inferior y superior (campos de 4 bits) del acumulador (bits 3 a 0 y bits 7 a 4). La operación puede también considerarse como una instrucción de desplazamiento de 4 bits. Ninguna bandera se ve afectada.

Ejemplo: El acumulador contiene el valor 0C5H (11000101B). La instrucción,

```
SWAP A
```

deja el valor 5CH (01011100B) en el acumulador.

Bytes: 1
Ciclos: 1
Codificación: 11000100
Operación: (A3−A0) ↔ (A7−A4)

XCH A,<byte>

Función: Intercambia el acumulador con la variable de byte
Descripción: XCH carga el acumulador con el contenido de la variable indicada y al mismo tiempo escribe el contenido original del acumulador a dicha variable. El operando origen/destino puede utilizar el direccionamiento por registro, el directo o el indirecto por registro.
Ejemplo: R0 contiene la dirección 20H. El acumulador contiene el valor 3FH (00111111B). La ubicación en la RAM interna 20H contiene el valor 75H (01110101B). La instrucción,

```
XCH A,@R0
```

deja el valor 3FH (00111111B) en la ubicación en RAM 20H y el valor 75H (01110101B) en el acumulador.

XCH A,Rn

Bytes: 1
Ciclos: 1
Codificación: 11001rr
Operación: (A) ↔ (Rn)

XCH A,directa

Bytes: 2
Ciclos: 1
Codificación: 11000101 aaaaaaaa
Operación: (A) ↔ (directa)

XCH A,@Ri

Bytes: 1
Ciclos: 1

Codificación: 1100011i
Operación: $(A) \leftrightarrow ((Ri))$

XCHD A,@Ri

Función: Intercambia un dígito

Descripción: XCHD intercambia el nibble de orden inferior del acumulador (bits 0 a 3), el cual generalmente representa un dígito en hexadecimal o en BCD, con el nibble inferior de la ubicación en RAM interna direccionado indirectamente por el registro especificado. Los nibbles de orden superior (bits 7 a 4) de cada registro no se ven afectados. Ninguna bandera resulta afectada.

Ejemplo: R0 contiene la dirección 20H. El acumulador contiene el valor 36H (00110110B). La ubicación en RAM interna 20H contiene el valor 75H (01110101B). La instrucción,

```
XCHD A,@R0
```

deja el valor 76H (01110110B) en la ubicación en RAM 20H y el valor 35H (00110101B) en el acumulador.

Bytes: 1

Ciclos: 1

Codificación: 1101011i

Operación: $(A3-A0) \leftrightarrow ((Ri3-Ri0))$

XRL<byte destino>,<byte origen>

Función: OR exclusivo lógico para variables de byte

Descripción: XRL realiza una operación lógica de OR exclusivo orientada a bits entre las variables indicadas y almacena los resultados en el destino. Ninguna bandera se ve afectada.

Los dos operandos permiten seis combinaciones de modo de direccionamiento. Cuando el destino es el acumulador, el origen puede utilizar un direccionamiento por registro, directo, indirecto por registro, o inmediato; cuando el destino es una dirección directa, el origen puede ser el acumulador o datos inmediatos.

Nota: Cuando esta instrucción se utiliza para modificar un puerto de salida, el valor utilizado como el dato del puerto original se lee del latch de datos de salida, no de las terminales de entrada.

Ejemplo: Si el acumulador contiene el valor 0C3H (11000011B) y el registro 0 contiene el valor 0AAH (10101010B), entonces la instrucción,

```
XRL A,R0
```

deja el valor 69H (01101001B) en el acumulador.

Cuando el destino es un byte directamente direccionado, la instrucción puede complementar combinaciones de bits en cualquier ubicación en RAM o registro en hardware. El patrón de los bits a complementar se determina en el acumulador al tiempo de ejecución.
La instrucción,

```
XRL P1,#00110001B
```

complementa los bits 5, 4 y 0 del puerto de salida 1.

XRL A,Rn

Bytes: 1
Ciclos: 1
Codificación: 01101rrr
Operación: $(A) \leftarrow (A) \oplus (Rn)$

XRL A,direct

Bytes: 2
Ciclos: 1
Codificación: 01100101 aaaaaaaa
Operación: $(A) \leftarrow (A) \oplus (\text{directa})$

XRL A,@Ri

Bytes: 1
Ciclos: 1
Codificación: 0110011i
Operación: $(A) \leftarrow (A) \oplus ((Ri))$

XRL A,#data

Bytes: 2
Ciclos: 1
Codificación: 01100100 dddddddd
Operación: $(A) \leftarrow (A) \oplus$ #datos

XRL direct,A

Bytes:	2
Ciclos:	1
Codificación:	01100010 aaaaaaaa
Operación:	(directa) ← (directa) ⊕ (A)

XRL direct,#data

Bytes:	3
Ciclos:	2
Codificación:	01100011 aaaaaaaa dddddddd
Operación:	(directa) ← (directa) ⊕ #datos