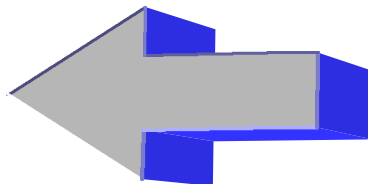


CONJUNTO DE INSTRUCCIONES DEL MICROCONTROLADOR 8051



ACALL dirección 11 (llamada absoluta)
--

ACALL llama incondicionalmente a una subrutina localizada en la dirección indicada. Durante esta instrucción se realizan los siguientes eventos: El contenido del PC se incrementa dos veces y apunta la dirección de la siguiente instrucción; el apuntador de apilamiento (Stack Pointer) se incrementa una vez, introduciendo el byte bajo del PC; incrementa nuevamente el SP para introducir el byte alto del PC; por último, el PC es cargado con el contenido de la **DIRECCIÓN DESTINO**. La ejecución de las instrucciones de la subrutina comienza en ésta dirección, hasta que encuentre la instrucción RET, la cual restablece el PC que habrá sido almacenado en el SP, continuando nuevamente con el programa inicial.

La **DIRECCIÓN DESTINO** es obtenida por concatenación sucesiva de los cinco bits de más alto orden del PC incrementado; más los 3 bits de más alto orden y el byte bajo del código de operación (OPCODE). Colocados respectivamente en ese orden a partir del byte alto, para formar la nueva dirección del PC. En otras palabras, las operaciones que se llevan a cabo son las siguientes:

OPERACIÓN: ACALL

(PC)	←	(PC)+2
(SP)	←	(SP)+1
((SP))	←	(PC ₇₋₀)
(SP)	←	(SP)+1
((SP))	←	(PC ₁₅₋₈)

El PC final es creado por:

(PC ₁₅₋₁₁)	←	(PC ₁₅₋₁₁ INCREMENTADO)
(PC ₁₀₋₈)	←	(OP CODE ₁₅₋₁₃) # de la pág.
(PC ₇₋₀)	←	(OP CODE ₇₋₀) direc. en la pág.

A continuación se presenta un ejemplo detallado de ésta instrucción.

EJEMPLO:

```

                                ORG 00H
0000 5100      ACALL LAZO1 ; 0101 0001 0000 0000
0002 E8      LAZO2 : MOV A, R0      pág.2      Direc. pág.
0003 C6              XCH A, @R0
0004 E9              MOV A, R1
0005 22              RET
0006 02 0200      LJMP 0200H

                                ORG 0200H
0200 F8 LAZO1: MOV R0, A
0201 22              RET
0202 1102      ACALL LAZO2; 000 1 0001 0000 0010
0204              END              pág.0      Direc. Pág.

```

En el ejemplo anterior se muestra en la dirección 0202 el llamado a la subrutina LAZO2, con el código de operación 1102, por lo tanto, la dirección a la cual debe de saltar se forma de la siguiente manera:

$$\begin{array}{rcl}
 \text{PC}_{\text{INC}} = 0204 & & \text{OPCODE} = 1102 \\
 \underline{0000\ 0\ 010\ 0000\ 0100\ \text{B}} & & \underline{000\ 1\ 0001\ 0000\ 0010\ \text{B}} \\
 \text{A} & & \text{B} \qquad \qquad \text{C} \\
 \text{PC}_{\text{FINAL}} = \text{A} - \text{B} - \text{C} \\
 \text{PC}_{\text{FINAL}} = 0000\ 0\ 000\ 0000\ 0010\ \text{B} = 0002\text{H}
 \end{array}$$

La dirección con la cual se carga el PC es 0002. Cabe aclarar que los bits 8 al 12 del OPCODE permanecen siempre constantes (10001), en cualquier llamado del tipo ACALL.

Como puede observarse también, el valor del nibble de más alto orden del PC INC y el PC FINAL es el mismo, ello nos limita la longitud del salto.

El destino debe por lo tanto estar dentro del mismo block de 2K de memoria del programa donde se encuentra la instrucción ACALL.

ADD suma la variable (SRC-BYTE) indicada y el Acumulador, dejando el resultado en el Acumulador. Las banderas de acarreo y acarreo auxiliar son establecidas, si hay un acarreo hacia afuera del bit 7 o del bit 3, de otro modo son limpiadas.

**ADD A,<SRC –
BYTE>**

Cuando se suman enteros sin signo, la bandera de acarreo indica que ocurrirá un sobreflujo.

OV se establece si hay un acarreo del bit 6 al 7 pero sin existir acarreo del bit 7 hacia afuera, o un acarreo hacia afuera del bit 7 pero no del bit 6 al 7; de otro modo OV es limpiada. Cuando se suman enteros signados, el OV indica un número negativo producido como la suma de dos operandos positivos, o uno positivo producido por la suma de dos operandos negativos.

Cuatro modos de fuentes de operandos dirigidos están permitidos: registro, directo, registro indirecto, o inmediato.

EJEMPLO:

El Acumulador contiene 0C3H (11000011B) y el registro 0 contiene 0AAH (10101010B). La instrucción, ADD A,R0, dejaría 6DH (01101101B) en el acumulador con la bandera del acarreo auxiliar limpiada y las banderas de acarreo y sobreflujo establecidas.

ADD A,Rn

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0010	1rrr
------	------

OPERACIÓN: **ADD** $(A) \leftarrow (A) + (Rn)$ **ADD A, @Ri**

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0010	111i
------	------

OPERACIÓN: **ADD** $(A) \leftarrow (A) + ((Ri))$ **EJEMPLO:**

RAM INTERNA

A=21	R0=10	0000
0000 28	ADD A,R0	0010
0001 26	ADD A,@R0	

10
12

En la primera instrucción se suman el contenido de A más el contenido de R0, es decir : $21 + 10 = 31$, el resultado es almacenado en el acumulador, ahora $A=31$.

En la segunda instrucción se suman el contenido del acumulador y el contenido de la localidad de memoria (dirección) apuntada por el registro R0, que en este caso es la 10, es decir: $31 + 12 = 43$, el resultado se almacena en el acumulador. Al término de las instrucciones el estado del programa sería $A=43$, $C=0$, $AC=0$, $OV=0$.

ADD A, directo

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 0 1 0	0 1 0 1	dirección directa
---------	---------	-------------------

OPERACIÓN: **ADD**

(A) ← (A)+(directo) **ADD A, #dato**

ADD A, #dato

BYTES: 2CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 0 1 0	0 1 0 0	dato inmediato
---------	---------	----------------

OPERACIÓN: **ADD**

(A) ← (A) + #dato

PSW7	PSW6	PSW2	BANDERAS QUE SE ACTIVAN
CY	CY	OV	

EJEMPLO:

	ORG 00H	RAM INTERNA
0000 7421	MOV A,#21H	R0 R1-----R7
0002 7810	MOV R0,#10H; A←21H	0000 10
0004 2500	ADD A, 00H ; A←31H	0009
0006 751012	MOV 10H,#12H	0010 12
0009 2412	ADD A,#12 ; A←43H	0018
000B	END	

ADDC simultáneamente suma el byte variable indicado, la bandera de acarreo y el contenido del Acumulador, dejando el resultado en el Acumulador. Las banderas de acarreo-auxiliar y la de acarreo están establecidas, respectivamente, si hay acarreo hacia el exterior del bit 3 ó del bit 7. De otra forma ellas están limpias.

Cuando se suman enteros signados negativamente la bandera de acarreo, indica que ocurrió un sobreflujo. OV se establece, si hay un acarreo del bit 6 al bit 7 pero no del bit 7

**ADDC A, <src-byte>
sumar con acarreo
(carry).**

hacia afuera, o del bit 7 hacia afuera pero no del bit 6 al bit 7. de otro modo OV está limpio. Cuando se suman enteros signados, el OV indica un número negativo producido como la suma de dos operandos positivos ó un número positivo producido por la suma de dos operandos negativos. Cuatro modos de direccionamiento de operandos fuente están permitidos: registro directo, registro indirecto ó inmediato.

EJEMPLO:

El Acumulador contiene 0C3H (11000011B) y el registro 0 contiene 0AAH (10101010B) con la bandera de acarreo fija. La instrucción, **ADDC A, R0**, dejará 6EH (01101110B) en el acumulador con AC limpiado y las banderas de acarreo y OV establecidas.

ADDC A, Rn

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0011	1rrr
------	------

OPERACIÓN: **ADDC**

(A) ← (A)+(C)+(Rn)

EJEMPLO:

A = C3	R0=AA C=1	C3 = 11000011
		AA= 10101010
		CT= <u> 1 </u>
		1 01101110

ADDC A,R0	⇒	A ← 6E	CY=1
		AC ← 0	OV=1

ADDC A, @Ri

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0011	011i
------	------

OPERACIÓN: **ADDC**

$$(A) \leftarrow (A)+(C)+((Ri))$$

ADDC A, directo

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 0 1 1	0 1 0 1	dirección directa
---------	---------	-------------------

OPERACIÓN: **ADDC**

$$(A) \leftarrow (A)+(C)+(\text{directo})$$

ADDC A, #dato

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 0 1 1	0 1 0 1	dato inmediato
---------	---------	----------------

OPERACIÓN: **ADDC**

$$(A) \leftarrow (A)+(C)+(\#dato)$$

AJMP transfiere la ejecución del programa a la dirección formada por la concatenación de los 5 bits de más alto orden del PC incrementado, más los bits del 7 al 5 del código de operación y el segundo byte que forma ésta instrucción AJMP. El destino debe por lo tanto estar dentro del mismo block de 2K de memoria del programa donde se encuentra la instrucción **AJMP**.

AJMP dirección 11
(salto absoluto)

EJEMPLO :

La etiqueta "SALTDR" está en la dirección del programa 0123H. La instrucción, **AJMP SALTDR** está en la localización 0345H y cargará el PC con 0123H.

BYTES: 2 CICLOS: 2

CÓDIGO DE OPERACIÓN:

a10 a9 a8 0	0 0 0 1	a7	a6	a5	a4	a3	a1	a1
-------------	---------	----	----	----	----	----	----	----

OPERACIÓN: **AJMP**

$$(PC) \leftarrow (PC)+2$$

El PC final es creado por:

$$\begin{aligned} (PC_{15-11}) &\leftarrow (PC_{15-11} \text{ INCREMENTADO}) \\ (PC_{10-8}) &\leftarrow (OP \text{ CODE }_{15-13}) \# \text{ de la pág.} \\ (PC_{7-0}) &\leftarrow (OP \text{ CODE }_{7-0}) \text{ direc. en la pág.} \end{aligned}$$

EJEMPLO :

```

                                ORG 00H
0000 C100    AJMP ETA ;ETA =0600H = 0000 0 110 0000 0000B
0600 7450    ETA: MOV A,#50
END

```

ANL <dest.>,<fuente>

ANL Ejecuta la operación lógica AND, bit a bit entre las variables indicadas y guarda los resultados en la variable destino. Las banderas no son afectadas.

Esta operación lógica permite 6 combinaciones de direccionamientos. Cuando el destino es el acumulador, los direccionamientos pueden ser por registro, directo, registro-indirecto, ó inmediato. Cuando el destino es una dirección, la fuente puede ser el acumulador o el dato inmediato.

NOTA :

Cuando ésta instrucción es usada para modificar un puerto de salida, el valor usado como dato del puerto será leído del latch del puerto NO DE LA PATA DEL MISMO.

EJEMPLO :

Si el acumulador contiene 0C3H (11000011B) y el registro contiene 55H (01010101B) luego la instrucción, **ANL A,R0** dejará 41H (01000001B) en el acumulador. Cuando el destino es un byte direccionado directamente, ésta instrucción limpiará combinaciones de bits en cualquier localidad RAM o registro.

El byte máscara que determina el patrón de bits que serán limpiados puede ser una constante contenida en la instrucción o un valor calculado en el acumulador. La instrucción, **ANL P1,#01110011B**, limpiará los bits 7, 3, y 2 del latch del puerto de salida 1.

ANL A,Rn

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 1 0 1	1 r r r
---------	---------

OPERACIÓN: ANL

$$(A) \leftarrow (A) \wedge (R_n)$$
ANL A,Directo

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 1 0 1	0 1 0 1	dirección directa
---------	---------	-------------------

OPERACIÓN: ANL

$$(A) \leftarrow (A) \wedge (\text{directo})$$
ANL A,@Ri

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 1 0 1	0 1 1 i
---------	---------

OPERACIÓN: ANL

$$(A) \leftarrow (A) \wedge ((R_i))$$
ANL A,#dato

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 1 0 1	0 1 0 0	dato inmediato
---------	---------	----------------

OPERACIÓN: ANL

$$(A) \leftarrow (A) \wedge \#dato$$
ANL directo,A

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 1 0 1	0 0 1 0
---------	---------

dirección directa

OPERACIÓN: **ANL**

$$(\text{directo}) \leftarrow (\text{directo}) \wedge (A)$$
ANL directo,#dato

BYTES: 3 CICLOS: 2

CÓDIGO DE OPERACIÓN:

0 1 0 1	0 0 1 1
---------	---------

dirección directa	dato inmediato
-------------------	----------------

OPERACIÓN: **ANL**

$$(\text{directo}) \leftarrow (\text{directo}) \wedge \# \text{dato}$$
EJEMPLO :

A=C3 R0=55

ANL A,R0 ; 11000011 = A
 01010101 = R0
 01000001 = A x R0 = 41

ANL P1,#01110011B ; si P1 es puerto de lectura

P1 = 11111111

P1 = 01110011

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

ANL C, bit AND lógico para bits variables
--

Si el valor Booleano del bit fuente es un 0 lógico, limpia la bandera de acarreo; de otro modo deja la bandera de acarreo en su estado corriente. Un slash ("/") precediendo al operando en el lenguaje de ensamblador indica que el complemento lógico del bit direccionado es usado como el valor fuente, pero el bit fuente por si mismo no es afectado. Ninguna otra bandera es afectada.

Sólo el direccionamiento directo es permitido por el operando fuente.

EJEMPLO :

Se coloca la bandera de acarreo si y solo si , P1.0=1, ACC.7=1 y OV=0:

MOV C,P1.0; C ← P1.0 ; establece el CARRY
 ANL C,ACC.7; C = C x ACC.7 ; Y-lóg. del bit ACC.7 con el CARRY
 ANL C,/OV; C = C x OV ; Y - lógico con el inverso de la bandera OV C se establece en los 3 casos.

ANL C,bit

BYTES: 2 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 0 0 0	0 0 1 0
---------	---------

bit direccionado

OPERACIÓN: **ANL**

(C) ← (C)^(bit)

ANL C,bit

BYTES: 2 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 0 1 1	0 0 0 0
---------	---------

bit direccionado

OPERACIÓN: **ANL**

(C) ← (C)^(bit)

CJNE. Compara las magnitudes del primero y segundo operando, y brinca si sus valores no son iguales. El salto es calculado por la suma algebraica del valor actual del PC y el desplazamiento relativo (operando REL). La bandera de acarreo es establecida si el valor entero sin signo de la <palabra destino> es menor que el valor entero sin signo de la <palabra fuente>; por otro lado el acarreamiento es limpiado. Ningún operando es afectado.

CJNE
 <dest>,<fuente>
Compara y brinca
si no es igual

Los primeros dos operandos permiten cuatro combinaciones de modos de direccionamiento; el acumulador puede ser comparado con cualquier palabra directamente direccionada ó un dato inmediato, y cualquier localidad de RAM indirecta ó registro trabajando que puede ser comparado con una constante inmediata.

EJEMPLO :

El acumulador contiene 34H. El registro 7 contiene 56H. La primera instrucción secuencia,

CJNE R7,#60H,NO_IGUAL**;R7=60H****NO_IGUAL: JC RIG_BAJO****;si R7<60H****;si R7>60H**

establece la bandera de acarreo y brinca a la instrucción con la etiqueta NO_IGUAL. Probando la bandera de acarreo esa instrucción determina si R7 es mayor o menor que 60H.

Si el dato que ha sido presentado al puerto 1 es también 34H entonces la instrucción;

ESPERA: CJNE A,P1,ESPERA

limpia la bandera de acarreo y continua con la siguiente instrucción, debido a que el acumulador es igual al dato leído del puerto P1. Si es cualquier otro valor el programa deberá de girar en este punto hasta que el puerto 1 tenga el dato 34H.

NOTA :

El dato es tomado directamente de las terminales del puerto.

CJNE A, DIRECTO,REL

BYTES: 3 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1	0	1	1	0	1	0	1
---	---	---	---	---	---	---	---

dirección directa	dirección relativa
-------------------	--------------------

OPERACIÓN: (PC) ← (PC) + 3

Si (A) <> (directo)

Entonces:

(PC) ← (PC) + salto relativo

Si (A) < (directo)

Entonces:

(C) ← 1

De lo contrario:

(C) ← 0

CJNE A,#DATO,REL

BYTES: 3 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 0 1 1	0 1 0 1	dato inmediato	dirección relativa
---------	---------	----------------	--------------------

OPERACIÓN: (PC) \leftarrow (PC) + 3Si (A) \neq DATO

Entonces:

(PC) \leftarrow (PC) + salto relativo

Si (A) < dato

Entonces:

(C) \leftarrow 1

Además:

(C) \leftarrow 0**CJNE Rn #DATO, REL**

BYTES: 3 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 0 1 1	1 r r r	dato inmediato	dirección relativa
---------	---------	----------------	--------------------

OPERACIÓN: **CJNE**(PC) \leftarrow (PC) + 3Si (Rn) \neq dato

Entonces:

(PC) \leftarrow (PC) + salto relativo

Si (Rn) < dato

Entonces:

(C) \leftarrow 1

De lo contrario:

(C) ← 0

CJNE @Ri,#DATO,REL

BYTES: 3 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 0 1 1	0 1 1 i
---------	---------

dato inmediato	dirección relativa
----------------	--------------------

OPERACIÓN: (PC) ← (PC) + 3

Si ((Ri)) <> dato

Entonces:

(PC) ← (PC) + salto relativo

Si ((Ri)) < dato

Entonces:

(C) ← 1

De lo contrario:

(C) ← 0

<p>CLR A Limpia el acumulador</p>
--

CLR A. El acumulador es limpiado (todos los bits se colocan en cero). Las banderas no son afectadas.

EJEMPLO :

El acumulador contiene 5CH (01011100B). La instrucción, **CLR A** dejará el acumulador colocado a 00H (00000000B).

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1 1 1 0	0 1 0 0
---------	---------

OPERACIÓN: **CLR**

(A) ← 0

<p>CLR bit limpia el bit.</p>
--

CLR. El bit indicado es limpiado (se convierte a cero). Ninguna otra bandera es afectada. CLR puede operar sobre una bandera de acarreo o cualquier bit directamente direccionable.

EJEMPLO :

El puerto 1 ha sido previamente escrito con 5DH (01011101B). La instrucción,

CLR P1.2

dejará el puerto colocado en 59H (01011001B El).

CLR C

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1 1 0 0	0 0 1 1
---------	---------

OPERACIÓN: **CLR**

(C) \leftarrow 0

CLR bit

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1 1 0 0	0 0 1 0	dirección del bit
---------	---------	-------------------

OPERACIÓN: **CLR**

(bit) \leftarrow 0

CPL A. Cada bit del acumulador está lógicamente complementado (complemento a unos). Los bits que previamente contienen un 1 son cambiados a 0 y viceversa. Las banderas no son afectadas.

CPL A complemento del acumulador.
--

EJEMPLO:

El acumulador contiene 5CH (01011100B). La instrucción, **CPL A** dejará el acumulador colocado a 0A3H (10100011B).

CPL A

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

OPERACIÓN: CPL

$$(A) \leftarrow \neg(A)$$

Complemento del acumulador

CPL BIT
Complemento del bit

CPL BIT. La variable bit especificada es complementada. Un bit el cual ha estado en 1 es cambiado a 0, y viceversa. Ninguna otra bandera es afectada. CPL puede operar sobre la bandera de acarreo (carry) o cualquier bit direccionable.

NOTA :

Cuando ésta instrucción es utilizada para modificar una terminal de salida, el valor usado como dato original será leído del latch de salida (cerrojo), no de la entrada de la terminal del puerto 1;

EJEMPLO :

El puerto 1 ha sido previamente escrito con 5DH (01011101B).
La instrucción secuencia,

CPL P1.1

CPL P1.2

dejará el puerto colocado a 5BH (01011011B).

CPL C

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1 0 1 1	0 0 1 1
---------	---------

OPERACIÓN: **CPL**

(C) ← /(C)

CPL bit

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1	0	1	1	0	0	1	0
---	---	---	---	---	---	---	---

dirección del bit

OPERACIÓN:

(bit) ← /(bit)

Complemento del acarreo

DA A, Ajusta el valor de los 8 bits del acumulador, resultado de una suma anterior de dos variables (cada una en formato BCD), produciendo dos dígitos de cuatro bits. Cualquier instrucción ADD o ADDC puede haber sido usada para ejecutar la suma.

Si en el acumulador los bits de 3-0 son, más grandes que 9 (XXXX1010-XXXX1111), o si la bandera AC es 1, 6 es sumado al acumulador produciendo el dígito en BCD del nibble de bajo orden. Esta suma interna establecerá la bandera de acarreo si un acarreo hacia el exterior del campo de los cuatro bits de alto orden, del o contrario, limpiará la bandera de acarreo.

Si la bandera de acarreo es ahora establecida ó si los cuatro bits de alto orden ahora excedieron de nueve (1010XXXX-1111XXXX), estos bits de alto orden son incrementados por seis, produciendo el dígito BCD en el nibble de alto orden. De igual forma, se establecería la bandera de acarreo si hubiera un acarreo hacia el exterior de los bits de alto orden, de lo contrario la bandera será limpiada. Así la bandera de acarreo indica si la suma de las dos variables BCD, originales es más grande que 100, permitiendo sumar múltiples decimales con precisión. OV no es afectado.

Todo esto ocurre durante el ciclo de una instrucción. Esencialmente, esta instrucción ejecuta la conversión decimal por adición de 00H, 06H, 60H, ó 66H al acumulador, dependiendo del acumulador inicial y las condiciones del PSW.

NOTA:

DA A no puede simplemente covertir un número hexadecimal en el acumulador a notación BCD; no se aplica DA A a substracción decimal.

EJEMPLO:

DA A
Ajuste decimal del
acumulador por
adición

El acumulador contiene el valor 56H (1010110B) representando el paquete de dígitos BCD del número decimal 56. El registro 3 contiene el valor 67H (01100111B) representando el paquete de dígitos del número decimal 67. La bandera de acarreo está establecida. La secuencia de instrucciones,

ADDC A,R3

DA A

ejecutará una suma estándar en complemento a dos binario, resultando el valor 0BEH (10111110) en el acumulador. Las baderas de acarreo y acarreo auxiliar serán limpiadas.

La instrucción de ajuste decimal alterará entonces el acumulador al valor 24H (00100100B), indicando el paquete de dígitos del número decimal 24, los dos dígitos de bajo orden de la suma decimal de 56, 67, y el acarreo. La bandera de acarreo será establecida por la instrucción de ajuste decimal, indicando que ocurrió un sobreflujo. La verdadera suma 56, 67, y 1 es 124.

Las variables BCD pueden ser incrementadas o decrementadas por adición de 01H ó 99H respectivamente. Si el acumulador inicialmente contiene 30H (representando los dígitos del 30 decimal), entonces la secuencia de las instrucciones,

ADD A,#99H

DA A

dejarán el acarreo establecido y 29H en el acumulador, puesto que $30 + 99 = 129$. El byte de bajo orden de la suma puede ser interpretado como $30 - 1 = 29$. En este caso es convencional si se incluye o no el valor del acarreo como parte del resultado.

BYTES:1 CICLOS: 1

CODIGO DE OPERACIÓN:

1 1 0 1	0 1 0 0
---------	---------

OPERACIÓN: **DA**

Los contenidos del acumulador están contenidos en BCD

SI $[(A_{3-0}) > 9] \vee [(AC) = 1]$
Entonces $(A_{3-0}) \leftarrow (A_{3-0}) + 6$

SI $[(A_{7-4}) > 9] \vee [(C) = 1]$
Entonces $(A_{7-4}) \leftarrow (A_{7-4}) + 6$

DEC. La variable indicada es decrementada en 1. Cuando el valor original del BYTE es 00H al decrementarse pasará a 0FFH y existirá un sobreflujo. Ninguna otra bandera es afectada. Cuatro operandos de modos de direccionamiento son permitidos: acumulador, registro, directo ó registro indirecto.

NOTA:

Cuando ésta instrucción es usada para modificar un puerto de salida, el valor usado como el dato original del puerto será leído del latch de salida, no de la terminal de entrada.

EJEMPLO:

El registro 0 contiene 7FH (01111111B). Las localidades 7EH y 7FH de la RAM interna contienen 00H y 40H, respectivamente. La secuencia de instrucciones,

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

Dejará el registro 0 colocado a 7EH y la localidad de la RAM interna 7EH y 7FH colocada a 0FFH y 3FH. Se indicará además que hubo un sobreflujo, OV = 1.

DEC A

BYTES:1 CICLOS:1

CÓDIGO DE OPERACIÓN:

0 0 0 1	0 1 0 0
---------	---------

OPERACIÓN: **DEC**
(A) $\leftarrow (A) - 1$

DEC Rn

BYTES:1 CICLOS: 1

CODIGO DE OPERACIÓN:

0 0 0 1	1 r r r
---------	---------

OPERACIÓN: **DEC**

(Rn) \leftarrow (Rn) - 1

DEC directo

BYTES:2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 0 0 1	0 1 0 1	dirección directa
---------	---------	-------------------

OPERACIÓN: **DEC** $(\text{directo}) \leftarrow (\text{directo}) - 1$ **DEC @Ri**

BYTES:1 CICLOS:1

CODIGO DE OPERACIÓN:

0 0 0 1	0 1 1 1
---------	---------

OPERACIÓN:

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

DIV AB Divide

DIV AB divide el entero no signado del acumulador entre el entero no signado del registro B. El acumulador recibe la parte entera del cociente; el registro B el residuo. El carry y la bandera OV serán limpiadas.

EXCEPCIÓN:

Si B contiene originalmente 00H, los valores del acumulador y el registro B será indefinido, la bandera de sobreflujo será establecida. La bandera de acarreo de limpiada en cualquier caso.

EJEMPLO:

El acumulador contiene 251 (0FBH ó 11111011B) y B contiene 18 (12H ó 00010010B). La instrucción,

DIV AB

Dejará 13 en el acumulador (0DH ó 00001101B) y el valor 17 (11h ó 00010001B) en B, puesto que $251 = (13 \times 18) + 17$. Carry y OV serán limpiados.

BYTES: 1 CICLOS: 4

CODIGO DE OPERACIÓN:

1 0 0 0	0 1 0 0
---------	---------

OPERACIÓN: **DIV**

$$\begin{array}{ccc} (A)_{15-8} & & \\ & \leftarrow & (A) / (B) \\ (B)_{7-0} & & \end{array}$$

DJNZ decrementa en uno la localidad indicada, si el byte no es igual a cero, salta a la dirección formada por la suma algebraica del PC incrementado y el desplazamiento relativo <REL>, de otra manera continúa con la siguiente instrucción.

**DJNZ <byte>,<rel>
decrementa y brinca
si no es cero**

Un valor original de 00H pasará a 0FFh. Las banderas no son afectadas. La localidad decrementada puede ser un registro o un byte direccionado directamente.

NOTA:

Cuando la instrucción es usada para modificar un puerto de salida, el valor usado como dato original del puerto será leído del latch de salida, no de las terminales de entrada.

EJEMPLO:

Las localidades de la RAM interna 40H, 50H y 60H contienen el valor 01H, 70H y 15H, respectivamente. La secuencia de instrucciones,

```
DJNZ 40H,LABEL_1
DJNZ 50H,LABEL_2
DJNZ 60H,LABEL_3
```

Causará un salto a la instrucción con la etiqueta LABEL_2 con los valores 00H, 6FH y 15H, en las tres localidades de la RAM. El primer salto no se efectúa porque el resultado es cero.

La instrucción provee una forma simple de ejecutar el lazo de un programa un número dado de veces, o provocar un tiempo de retraso moderado (de 3 a 512 ciclos de máquina) con una sola instrucción. La secuencia de instrucciones,

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

Alternará el P1.7 ocho veces, causando cuatro pulsos de salida que aparecerán en el bit 7 del puerto de salida 1. Cada pulso durará tres ciclos de máquina; dos por DJNZ y uno por alterar la terminal.

DJNZ Rn,rel

BYTES: 2 CICLOS: 2

CÓDIGO DE OPERACIÓN:

0 0 0 0	0 1 0 1	dirección directa
---------	---------	-------------------

OPERACIÓN: **DJNZ** $(PC) \leftarrow (PC) + 2$ $(Rn) \leftarrow (Rn) - 1$ Si $(Rn) > 0$ ó $(Rn) < 0$

Entonces:

 $(PC) \leftarrow (PC) + rel$ **DJNZ directo,rel**

BYTES: 3 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 1 0 1	0 1 0 1	Dirección directa	Dirección relativa
---------	---------	-------------------	--------------------

OPERACIÓN: **DJNZ** $(PC) \leftarrow (PC) + 2$ $(directo) \leftarrow (directo) - 1$ Si $(directo) > 0$ ó $(directo) < 0$

Entonces:

 $(PC) \leftarrow (PC) + rel$

INC<byte> incremento.
--

INC incrementa el byte en 1. Si éste se encuentra inicialmente en 0FFH pasará a 00H. Las banderas no son afectadas. Cuatro modos de direccionamiento son permitidos: Acumulador, registro, directo, ó registro-indirecto.

NOTA :

Cuando esta instrucción es usada para modificar un puerto de salida, el valor usado como el dato original del puerto será leído del latch de salida, no de la terminal de entrada.

EJEMPLO :

El registro 0 contiene 7EH(01111110B). Las localidades de la RAM interna 7EH y 7FH contienen 0FFH y 40H, respectivamente.

La instrucción secuencia,

INC @R0
INC R0
INC @R0

dejará el registro 0 colocado a 7FH y las localidades de la RAM interna 7EH y 7FH conteniendo (respectivamente) 00H y 41H.

INC A

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 0 0 0	0 1 0 0
---------	---------

OPERACIÓN: **INC**

(A) \leftarrow (A) + 1

NOTA:

En esta instrucción de INC la bandera de paridad se ve afectada.

INC Rn

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 0 0 0	0 r r r
---------	---------

OPERACIÓN: **INC**

(Rn) \leftarrow (Rn) + 1

INC directo

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 0 0 0	0 1 0 1	dirección directa
---------	---------	-------------------

OPERACIÓN: **INC**

$$(\text{directo}) \leftarrow (\text{directo}) + 1$$
INC @Ri

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 0 0 0	0 1 1 i
---------	---------

OPERACIÓN: **INC**

$$((Ri)) \leftarrow ((Ri)) + 1$$

INC DPTR
incrementa el apuntador de
datos.

INC DPTR. Incrementa el valor del apuntador de datos en 1. El incremento en un registro de 16 bits es optimizado; un sobreflujo del byte de bajo orden del apuntador de datos (D L) de 0FFH a 00H incrementará el byte de alto orden (D H). Las banderas no son afectadas.

Este es el único registro de 16 bits que puede ser incrementado.

EJEMPLO :

Los registros DPH y DPL contienen 12H y 0FEH, respectivamente. La secuencia de instrucciones,

INC DPTR
INC DPTR
INC DPTR

cambiará DPH y DPL a 13H y 01H respectivamente.

INC

BYTES: 1 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 0 1 0	0 0 1 1
---------	---------

OPERACIÓN: **INC**

$$(DPTR) \leftarrow (DPTR) + 1$$

JB. Si el bit indicado es uno, salta a la dirección formada por la suma algebraica del PC incrementado y del byte de desplazamiento relativo, REL; de otra manera procede con la siguiente instrucción. El bit marcado no se modifica. Las banderas no son afectadas.

EJEMPLO :

El dato presente en el puerto de entrada 1 es CA (11001010B). El acumulador contiene 56 (01010110B). La secuencia de instrucciones,

JB 1,ETIQUETA1
JB ACC.2,ETIQUETA2

causará que el programa en ejecución salte a la instrucción marcada con ETIQUETA2.

JB

BYTES: 3 CICLOS: 2

CÓDIGO DE OPERACIÓN:

0 0 1 0	0 0 0 0	dirección del bit	dirección relativa
---------	---------	-------------------	--------------------

OPERACIÓN: **JB**

Si (bit) = 1
 Entonces:

$$(C) \leftarrow (C) + 3$$

$$(C) \leftarrow (C) + \text{rel}$$

JBC. Si el bit marcado es uno, lo limpia y salta a la dirección formada por la suma algebraica del C incrementado y del byte de desplazamiento relativo, REL; de otra manera procede con la siguiente instrucción. Las banderas no son afectadas.

JBC bit,rel
Si el bit esta establecido
lo limpia y salta.

NOTA : Cuando esta instrucción es usada para examinar una terminal de salida, el valor usado como dato original será leído del latch de salida, no de la terminal de entrada.

EJEMPLO :

El acumulador contiene 56H (01010110B). La secuencia de instrucciones,

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

causará que la ejecución del programa continúe en la instrucción identificada por ETIQUETA2, con el acumulador modificado a 52H (01010010B).

JBC

BYTES: 3 CICLOS: 2

CÓDIGO DE OPERACIÓN:

0 0 0 1	0 0 0 0
---------	---------

dirección del bit	dirección relativa
-------------------	--------------------

OPERACIÓN: **JBC**

Si (bit) = 1
Entonces:

$$(PC) \leftarrow (C) + 3$$

$$(bit) \leftarrow 0$$

$$(PC) \leftarrow (PC) + rel$$

JC. Si la bandera de acarreo está establecida, salta a la dirección formada por la suma algebraica del C incrementado y del byte de desplazamiento relativo, REL; de otra manera procede con la siguiente instrucción. Las banderas no son afectadas.

EJEMPLO:

La bandera de acarreo es limpiada. La instrucción secuencia,

```

JC    ETIQUETA 1
CPLC  C
JC    ETIQUETA 2

```

colocará el carry y causará la ejecución del programa para continuar con la instrucción identificada por la etiqueta LABEL 2.

JC

BYTES: 2 CICLOS: 2

CÓDIGO DE OPERACIÓN:

0 1 0 0	0 0 0 0
---------	---------

dirección relativa

OPERACIÓN: **JC**

Entonces:

$$(C) \leftarrow (C) + 2$$

$$(C) \leftarrow (C) + rel$$

JMP @A+DPTR salto indirecto.
--

JUMP @A+DPTR. Suma los ocho bits no signados contenidos en el acumulador con los 16 bits del apuntador de datos y carga la suma resultante al contador del programa. Esta será la nueva dirección para las siguientes búsquedas de las instrucciones.

La suma de los 16 bits es optimizada. El carry de salida de los ocho bits de bajo orden se propaga a los bits de alto orden. Ni el acumulador ni el dato del apuntador son alterados. Las banderas no son afectadas.

EJEMPLO :

Un número par de 0 a 6 está en el acumulador. La siguiente secuencia de instrucciones saltará a una de cuatro instrucciones AJMP en un salto empezando por **JMP_TBL**:

```

MOV      DPTR,#JMP_TBL
JMP      @A+DPTR
JMP_TBL: AJMP ETIQT0
          AJMP ETIQT1
          AJMP ETIQT2
          AJMP ETIQT3

```

Si el acumulador es igual a 04H cuando empieza ésta secuencia, la ejecución de la instrucción **JMP @A+DPTR**, saltará a la etiqueta ETIQT2. Recuerda que AJMP es una instrucción del byte 2, así la instrucción empieza en cada dirección.

BYTES: 1 CICLOS: 2

CÓDIGO DE OPERACIÓN:

0 1 1 1	0 0 1 1
---------	---------

OPERACIÓN: **JMP**

(PC) ← (A) + (DPTR)

JNB. Si el bit indicado es un cero, salta a la dirección formada por la suma algebraica del PC incrementado y del byte de desplazamiento relativo, REL; de otra manera procede con la siguiente instrucción.

JNB bit,rel
salta si el bit no es
colocado

El bit examinado no es modificado. Las banderas no son afectadas.

EJEMPLO:

El dato presente en el puerto de entrada 1 es 11001010B. El acumulador contiene 56H (01010110B). La instrucción secuencia,

```

JNB P1.3,ETIQT1
JNB ACC.3,ETIQT2

```

causará la ejecución del programa para continuar con la instrucción o etiqueta ETIQT2.

BYTES: 3 CICLOS: 2

CÓDIGO DE OPERACIÓN:

0 0 1 1	0 0 0 0	bit de dirección	dirección relativa
---------	---------	------------------	--------------------

OPERACIÓN: **JNB**

(PC) \leftarrow (PC) + 3

Si (bit) = 0

Entonces (PC) \leftarrow (PC) + rel

JNC. Si el bit de acarreo es un cero, salta a la dirección formada por la suma algebraica del PC incrementado y del byte de desplazamiento relativo, REL; de otra manera procede con la siguiente instrucción. La bandera de acarreo no es modificada.

EJEMPLO :

La bandera de acarreo está establecida. La secuencia de instrucciones:

JNC ETIQT1
CPL C
JNC ETIQT2

limpiará el acarreo y continuará la ejecución del programa en la instrucción identificada con la etiqueta ETIQT2

BYTES: 2 CICLOS: 2

CÓDIGO DE OPERACIÓN:

0 1 0 1	0 0 0 0	dirección relativa
---------	---------	--------------------

OPERACIÓN: **JNC**

(PC) \leftarrow (PC) + 2

Si (C) = 0

Entonces (PC) \leftarrow (PC) + rel

JNZ rel salta si el acumulador no es cero.

JNZ. Si cualquier bit del acumulador es un 1, salta a la dirección formada por la suma algebraica del PC incrementado y del byte de desplazamiento relativo, REL; de otra manera procede con la siguiente instrucción. La bandera de acarreo no es modificada. de otra manera procede con la siguiente instrucción.

EJEMPLO :

El acumulador originalmente contiene 00H. La secuencia de instrucciones,

```

JNZ ETIQUETA1
INC A
JNZ ETIQUETA2

```

colocará el acumulador a 01H y saltará a la instrucción identificada con ETIQUETA2.

BYTES: 2 CICLOS: 2

CÓDIGO DE OPERACIÓN:

0 1 1 1	0 0 0 0	dirección relativa
---------	---------	--------------------

OPERACIÓN: **JNZ**

(PC) \leftarrow (PC) + 2

Si (A) \neq 0

Entonces (PC) \leftarrow (PC) + rel

JZ. Si todos los bits del acumulador son cero, salta a la dirección formada por la suma algebraica del PC incrementado y del byte de desplazamiento relativo, REL; de otra manera procede con la siguiente instrucción. El acumulador no se modifica. Las banderas no son afectadas.

JZ rel salta si el acumulador es cero.

EJEMPLO :

El acumulador originalmente contiene 01H. La secuencia de instrucciones,

```

JZ ETIQUETA 1
DEC A
JZ ETIQUETA 2

```

cambiará el acumulador a 00H y causará que la ejecución del programa continúe en la instrucción identificada por ETIQUETA 2.

BYTES: 2 CICLOS: 2

CÓDIGO DE OPERACIÓN:

0 1 1 0	0 0 0 0	dirección relativa
---------	---------	--------------------

OPERACIÓN: **JZ**

(PC) ← (PC) + 2
 Si (A) = 0
 Entonces (PC) ← (PC) + rel

**LCALL dirección 16
 llamada larga.**

LCALL llama una subrutina que puede empezar en cualquier parte de la memoria de programa (64K bytes). La instrucción suma tres al contador del programa para que apunte a la dirección de la siguiente instrucción, luego incrementa el SP introduciendo el byte bajo del PC e incrementa nuevamente el SP para introducir el byte alto del PC. El PC se carga con el segundo y tercer byte de la instrucción **LCALL**. La ejecución de las instrucciones de la subrutina comienza en ésta dirección, hasta que encuentre la instrucción **RET**, la cual restablece el PC que había sido almacenado en el SP, continuando nuevamente con el programa inicial. Las banderas no son afectadas.

EJEMPLO :

Inicialmente el apuntador de apilamiento es igual a 07H. La etiqueta "**SBRTN**" es asignada a la memoria del programa en la localidad 1234H. Después de la ejecución la instrucción, **LCALL SUBRTN** se localiza en 0123H, el apuntador de apilamiento contendrá 09H, las localidades de la RAM interna 08H y 09H contendrá 26H y 01H, y la PC contendrá 1235H.

BYTES: 3

CICLOS: 2

0 0 0 1	0 0 1 0
---------	---------

direc. 15-direc. 8	direc. 7-direc. 0
--------------------	-------------------

OPERACIÓN: **LCALL**

(PC) ← (PC) + 3
 (SP) ← (SP) + 1
 ((SP)) ← (PC₇₋₀)
 (SP) ← (SP) + 1
 ((SP)) ← (PC₁₅₋₈)
 (PC) ← direc.15-0

**LJMP dirección 16
 salto largo.**

LJMP causa un salto incondicional a cualquier parte en el espacio de memoria del programa (64K bytes). El PC se carga con los 2 últimos bytes de la instrucción y salta para continuar con la ejecución del programa a partir de esa dirección. Las banderas no son afectadas.

EJEMPLO :

La etiqueta "**SALTO**" es asignada a la instrucción del programa de memoria en la localidad 1145H.

0156 021145 LJMP SALTO

1145 7827 SALTO: MOV R0,#27

La instrucción LJMP se localiza en 0156H, cargará el contador del programa con 1145H y saltará a dicha dirección continuando la ejecución del programa en ese punto.

BYTES: 3 CICLOS: 2

CÓDIGO DE OPERACIÓN:

0 0 0 0	0 0 1 0	direc. bits 15-8	direc. bits 7-0
---------	---------	------------------	-----------------

OPERACIÓN: **LJMP**

(PC) ← direc.15-0

MOV. La variable indicada por el segundo operando "Byte Fuente" es copiada en la localidad especificada por el primer operando "Byte destino". La palabra fuente no es afectada. Ningún otro registro o bandera es afectado.

Esta es la operación más extensa y flexible con que cuenta el microcontrolador. Quince combinaciones de modos de direccionamiento de fuente y destino son permitidos.

EJEMPLO :

La localidad de la RAM interna 30H contiene el dato 40H. El valor de la RAM interna 40H es 10H. El dato presente en el puerto de entrada 1 es 11001010B (0CAH).

```

00          ORG 00H
0000 7830   MOV R0,#30H ;      R0 ← 30H
0002 E6     MOV A,@R0 ;      A ← 40H
0003 F9     MOV R1,A ;      R1 ← 40H
0004 87F0   MOV B,@R1 ;      B ← 10H
0006 A790   MOV @R1,P1 ;      RAM(40H) ← 0CAH
0008 8590A0 MOV P2,P1 ;      P2 #0CAH
0000        END

```

deja el valor 30H en el registro 0, 40H en ambos el acumulador y el registro 1, 10H en el registro B, y 0CAH (11001010B) ambos en la localidad 40H de la RAM y sobre el puerto 2 de salida.

MOV A,Rn

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1 1 1 0	1 r r r
---------	---------

OPERACIÓN: **MOV**

(A) ← (Rn)

MOV A,directo

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1 1 1 0	0 1 0 1
---------	---------

dirección directa

OPERACIÓN: **MOV**

(A) ← (directo)

MOV A,ACC no es una instrucción válida.

MOV A,@Ri

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1 1 1 0	0 1 1 i
---------	---------

OPERACIÓN: **MOV**

(A) ← ((Ri))

MOV A,#dato

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 1 1 1	0 1 0 0
---------	---------

dato inmediato

OPERACIÓN: **MOV**

(A) ← #dato

MOV Rn,A

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1 1 1 1	1 r r r
---------	---------

OPERACIÓN: **MOV**

(Rn) ← (A)

MOV Rn,directo

BYTES: 2 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 0 1 0	1 r r r
---------	---------

dirección directa

OPERACIÓN: **MOV**

(Rn) ← (directo)

MOV Rn,#dato

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 1 1 1	1 r r r
---------	---------

dato inmediato

OPERACIÓN: **MOV**

(Rn) ← #dato

MOV directo,A

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1 1 1 1	0 1 0 1
---------	---------

dirección directa

OPERACIÓN: **MOV**

(directo) ← (A)

MOV directo,Rn

BYTES: 2 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 0 0 0	1 r r r	dirección directa
---------	---------	-------------------

OPERACIÓN: **MOV**

(directo) ← (Rn)

MOV directo,directo

BYTES: 3 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 0 0 0	0 1 0 1	direc. directa(dest.)	direc.directa(fuen.)
---------	---------	-----------------------	----------------------

OPERACIÓN: **MOV**

(directo) ← (directo)

MOV directo,@Ri

BYTES: 2 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 0 0 0	0 1 1 i	dirección directa
---------	---------	-------------------

OPERACIÓN: **MOV**

(directo) ← ((Ri))

MOV directo,#dato

BYTES: 3 CICLOS: 2

CÓDIGO DE OPERACIÓN:

0 1 1 1	0 1 0 1	dirección directa	dato inmediato
---------	---------	-------------------	----------------

OPERACIÓN: **MOV**

(directo) ← #dato

MOV @Ri,A

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1 1 1 1	0 1 1 i
---------	---------

OPERACIÓN: **MOV**

((Ri)) ← (A)

MOV @Ri,directo

BYTES: 2 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 0 1 0	0 1 1 i	dirección directa
---------	---------	-------------------

OPERACIÓN: **MOV**

((Ri)) ← (directo)

MOV @Ri,#dato

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 1 1 1	0 1 1 i	dato inmediato
---------	---------	----------------

OPERACIÓN: **MOV**

((Ri)) ← #dato

MOV. El bit indicado por el segundo operando es copiado en la localidad especificada por el primer operando. Uno de los operandos debe ser la bandera de acarreo; el otro puede ser cualquier bit direccionable directamente. Ningún otro registro o bandera es afectado.

EJEMPLO :

<p>MOV <bit destino> <bit fuente> movimiento de bit</p>

La bandera de acarreo es originalmente establecida a 1 lógico. El dato presente en el puerto 3 de entrada es C5 (11000101B). El dato previamente escrito al puerto 1 de salida es 35H (00110101B). Las instrucciones,

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

dejarán el acarreo limpiado y cambiará el puerto 1 a 39H (00111001B).

MOV C,bit

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1 0 1 0	0 0 1 0
---------	---------

dirección del bit

OPERACIÓN: **MOV**

(C) ← (bit)

MOV bit,C

BYTES: 2 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 0 0 1	0 0 1 0
---------	---------

dirección del bit

OPERACIÓN: **MOV**

(bit) ← (C)

MOV DPTR,#DATO 16 carga al apuntador de datos con una constante de 16 bits.
--

MOV DPTR.El apuntador de datos es cargado con un dato de 16 bits contenidos en el segundo y tercer bytes de la instrucción, que corresponden al byte de alto orden (DPH) y al byte de bajo orden (DPL) del DPTR respectivamente. Las banderas no son afectadas. Esta es la única instrucción que mueve un dato de 16 bits.

EJEMPLO :

La instrucción, **MOV DPTR,#35A4H** cargará el valor 35A4H en el apuntador de datos : DPH almacenará 35H y DPL almacenará A4H.

BYTES: 3 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 0 0 1	0 0 0 0
---------	---------

dato inmediato 15-8	dato inmediato 7-0
---------------------	--------------------

OPERACIÓN: **MOV**

DPH ← #dato₁₅₋₈
 DPL ← #dato₇₋₀

La instrucción **MOVC** carga el acumulador con un byte código, ó constante del programa de memoria. La dirección del byte escogido es la suma de los 8 bits del contenido original del acumulador y los 16 bits del registro base, el cual puede ser el apuntador de datos (DPTR) o el contador del programa (PC). En el último caso, el PC es incrementado a la dirección de la siguiente instrucción antes de ser sumado con el acumulador. En cualquiera de los casos el registro base no es alterado. La suma de los 16 bits es optimizada. Un acarreo de salida del byte de bajo orden se propaga al byte de alto orden. Las banderas no son afectadas.

EJEMPLO :

Un valor entre 0 y 3 está en el acumulador. La siguiente instrucción trasladará el valor en el acumulador a uno de cuatro valores definidos por el DB (byte definido) directivo.

```
REL_PC:    INC    A
           MOVC  A,@A + PC
           RET
           DB    66H
           DB    77H
           DB    88H
           DB    99H
```

Si la subrutina es llamada con el acumulador igual a 01H, retornará con 77H en el acumulador. La instrucción INC A es necesitada antes de la instrucción MOVC para saltar por encima de la instrucción RET a las constantes de la tabla.

MOVC A,@A + DPTR

BYTES: 1 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 0 0 1	0 0 1 1
---------	---------

OPERACIÓN: **MOVC**

(A) ← ((A) + (DPTR))

MOVC A,@A + PC

BYTES: 1 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 0 0 0	0 0 1 1
---------	---------

OPERACIÓN: **MOVC**

$$\begin{array}{lll} (\text{PC}) & \leftarrow & (\text{PC}) + 1 \\ (\text{A}) & \leftarrow & ((\text{A}) + (\text{PC})) \end{array}$$

La instrucción **MOVX** transfiere el dato entre el acumulador y un byte de la memoria externa. Existen dos tipos de instrucciones; las que proveen un direccionamiento indirecto de 8 bits y las que lo hacen con 16 bits para acceder un dato de la RAM externa.

En el primer caso, el contenido de R0 y R1 proveen los 8 bits de la parte baja de la dirección dada por el puerto P0. Ocho bits son suficientes para una expansión externa I/O decodificada o para un arreglo RAM relativamente pequeño. Para algunos arreglos grandes, cualquier puerto de salida o terminal puede ser usado para direccionar bits de salida de alto orden. Estas terminales deberán ser controladas por instrucciones de salida que precedan la instrucción MOVX.

ATENCIÓN:

En el direccionamiento indirecto de 8 bits, en el momento de acceder el Dato, el contenido del P2 que se encuentra en el Espacio de Funciones Especiales será enviado sin excepción a las terminales de salida del puerto.

Debe tenerse especial cuidado cuando se accese una memoria de 256 bytes, de utilizar las terminales restantes del puerto de manera correcta, o bien, cuando se utilice una memoria mayor y se desee acceder por páginas, por medio de los registros R0 o R1, asegurarse que el P2 señale el número de página adecuado.

En el segundo caso, el apuntador de datos DPTR, genera una dirección de 16 bits. Por el puerto P2 saldrá el byte de alto orden (el contenido de DPH) de la dirección y por el puerto P0 el byte de bajo orden (DPL) el cual se multiplexa con el dato. En el registro de funciones especiales el P2 retiene su contenido previo mientras el buffer de salida del P2 está emitiendo el contenido de DPH. Esta forma es más rápida y más eficiente cuando accesa arreglos de datos muy grandes (hasta 64K bytes), dado que no necesita instrucciones adicionales en el P2 para acceder el dato.

Es posible en algunas situaciones mezclar los dos tipos de MOVX. Un arreglo de RAM grande con sus líneas de dirección de alto orden manejadas por P2 puede ser direccionado por vía del apuntador de datos, o guardando primeramente en el puerto P2 la dirección de alto orden, seguida por una instrucción MOVX usando R0 o R1.

EJEMPLO :

Una RAM externa de 256 bytes, usando un multiplexor de líneas dato/dirección (v.gr. el 74HC373), es conectada al puerto 0 del 8051. El puerto 3 provee líneas de control por la RAM externa. El puerto 1 y 2 son usados por I/O normal. El registro 0 y 1 contienen 12H y 34H.

La localidad 34H de la RAM externa contiene el valor 56H. La secuencia de instrucciones,

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

copia el valor 56H en el acumulador y la localidad 12H de la RAM externa.

MOVX A,@Ri

BYTES: 1 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 1 1 0	0 0 1 i
---------	---------

OPERACIÓN: **MOVX**

(A) ← ((Ri))

MOVX A,@DPTR

BYTES: 1 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 1 1 0	0 0 0 0
---------	---------

OPERACIÓN: **MOVX**

(A) ← ((DPTR))

MOVX @Ri,A

BYTES: 1 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 1 1 1	0 0 1 i
---------	---------

OPERACIÓN: **MOVX**

((Ri)) ← (A)

MOVX @DPTR,A

BYTES: 1 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 1 1 1	0 0 0 0
---------	---------

OPERACIÓN: **MOVX**

(DPTR) ← (A)

MUL AB multiplica los ocho bits del acumulador y los del registro B. El byte de bajo orden del producto de 16 bits, se almacena en el acumulador, y el byte de alto orden en B. Si el producto es más grande que 255 (0FFH) la bandera de sobreflujo es establecida, de lo contrario es limpiada.

EJEMPLO :

Originalmente el acumulador contiene el valor 80 (50H). El registro B contiene el valor 160 (0A0H). La instrucción, **MUL AB** dará el producto 12,800 (3200H), así B es cambiado a 32H (00110010B) y el acumulador es limpiado. La bandera de sobreflujo es colocada, el acarreo es limpiado.

BYTES: 1 CICLOS: 4

CÓDIGO DE OPERACIÓN:

1 0 1 0	0 1 0 0
---------	---------

OPERACIÓN: **MUL**

(A)₇₋₀ ← (A) X (B)

(B)₁₅₋₈

<p>NOP No operación</p>

NOP. La ejecución continúa con la instrucción siguiente. No se afecta, ni el contador del programa, ni registros o banderas.

EJEMPLO :

Se desea producir un pulso bajo en la terminal de salida del bit 7 del puerto 2, exactamente durante los últimos 5 ciclos. Una secuencia SETB/CLR generaría un pulso de un ciclo, de tal manera, que se necesitarían 4 ciclos adicionales. Esto puede realizarse de la siguiente manera (se asume que no existen interrupciones):

CLR P2.7
NOP
NOP

NOP
NOP
SETB P2.7

NOP

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 0 0 0	0 0 0 0
---------	---------

OPERACIÓN: **NOP**

(PC) ← (PC) + 1

ORL ejecuta la operación OR lógica entre las variables indicadas, guardando el resultado en el byte destino. Las banderas no son afectadas.

Los dos operandos permiten 6 combinaciones de modos de direccionamiento. Cuando el destino es el acumulador, la fuente puede provenir de los siguientes direccionamientos; por registro, directo, registro indirecto, o direccionamiento inmediato; cuando el destino es una dirección directa, la fuente puede ser el acumulador o dato inmediato.

NOTA :

Cuando ésta instrucción es usada para modificar un puerto de salida, el valor usado como dato original, será leído del latch de salida del puerto y no de sus terminales.

EJEMPLO :

Si el acumulador contiene 0C3H (11000011B) y el R0 contiene 55H (01010101B) entonces la instrucción, **ORL A,R0** dejará el acumulador conteniendo el valor 0D7H (11010111B).

Con el direccionamiento directo, la instrucción puede colocar combinaciones de bits en cualquier localidad RAM interna o Registro. El patrón de bits que serán establecidos es determinado por un byte máscara, el cual puede ser también un dato o valor constante en la instrucción o una variable calculada en el acumulador. La instrucción, **ORL 1,#00110010B** colocará los bits 5, 4 y 1 del puerto de salida 1.

ORL A,Rn

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 1 0 0	1 r r r
---------	---------

OPERACIÓN: ORL

(A) ← (A) V (Rn)

ORL A,directo

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 1 0 0	0 1 0 1	dirección directa
---------	---------	-------------------

OPERACIÓN: **ORL**

(A) ← (A) V (directo)

ORL A,@Ri

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 1 0 0	0 1 1 i	dirección directa
---------	---------	-------------------

OPERACIÓN: **ORL**

(A) ← (A) V ((Ri))

ORL A,#dato

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 1 0 0	0 1 0 0	dato inmediato
---------	---------	----------------

OPERACIÓN: **ORL**

(A) ← (A) V #dato

ORL directo,A

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 1 0 0	0 0 1 0
---------	---------

dirección directa

OPERACIÓN: **ORL**

(directo) ← (directo) V (A)

ORL directo,#dato

BYTES: 3 CICLOS: 2

CÓDIGO DE OPERACIÓN:

0 1 0 0	0 0 1 1
---------	---------

dirección directa	dato inmediato
-------------------	----------------

OPERACIÓN: **ORL**

(directo) ← (directo) V #dato

ORL. Establece la bandera de acarreo si el valor Booleano es un 1 lógico; de otra manera no lo cambia. Un slash (o/o) precediendo el operando en el lenguaje ensamblador, indica que el complemento lógico del bit de direccionado es usado como el valor fuente, pero el bit original no es afectado. Las banderas no son afectadas.

EJEMPLO :

Coloca la bandera de acarreo sí y solo sí P1.0 = 1, ACC.7 = 1, ó OV = 0:

MOV C,P1.0 ;CARGA EL ACARREO CON LA TERMINAL DE ENTRADA P10**ORL C,ACC.7** ;O ACARREO CON EL ACC. BIT 7**ORL C,/OV** ;O ACARREO CON EL INVERSO DE OV.**ORL C,bit**

BYTES: 2 CICLOS: 2

CÓDIGO DE OPERACIÓN:

0 1 1 1	0 0 1 0
---------	---------

bit de dirección

OPERACIÓN: **ORL**

(C) ← (C) V (bit)

ORL C,/bit

BYTES: 2 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 0 1 0	0 0 0 0
---------	---------

bit de dirección

OPERACIÓN: **ORL**

(C) ← (C) V /(bit)

POP. El contenido de la localidad de la RAM interna direccionada por el apuntador de apilamiento es leído, y el apuntador de apilamiento es decrementado por 1. El valor leído es transferido al byte indicado por el direccionamiento directo. Las banderas no son afectadas.

EJEMPLO :

El apuntador de apilamiento originalmente contiene el valor 32H, y las localidades de RAM interna de la 30H a la 32H contienen los valores 20H, 23H y 01H, respectivamente. La instrucción secuencia,

POP DPH**POP DPL**

dejará el apuntador de apilamiento igual al valor 30H y el apuntador de datos se colocará a 0123H. En éste punto la instrucción, **POP SP** dejará el apuntador de apilamiento colocado a 20H. Nota que en éste caso especial el apuntador de apilamiento estaba decrementado a 2FH antes de ser cargado con el valor extraído (20H).

BYTES: 2 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 1 0 1	0 0 0 0
---------	---------

Dirección directa

OPERACIÓN: **POP**

(directo)---((SP))

(SP)---(SP) - 1

PUSH directo almacenamiento en la pila.
--

Push. El apuntador de apilamiento es incrementado por 1. El contenido de la variable indicada es luego almacenada en la localidad direccionada por el apuntador de apilamiento en la RAM interna. Las banderas no son afectadas.

EJEMPLO :

Al entrar en una rutina de interrupción el apuntador de apilamiento contiene 09H. El dato apuntador contiene el valor 0123H. La instrucción secuencia,

PUSH DPL
PUSH DPH

dejará el apuntador de apilamiento colocado a 0BH y guarda 23H y 01H en las localidades de la RAM interna 0AH y 0BH, respectivamente.

BYTES: 2 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 1 0 0	0 0 0 0	dirección directa
---------	---------	-------------------

OPERACIÓN: **PUSH**

(SP) \leftarrow (SP) + 1
((SP)) \leftarrow (directo)

RET extrae de la pila los bytes de bajo y alto orden del PC, decrementando dos veces el apuntador de apilamiento. Una vez que el PC es cargado con la nueva dirección, continúa con la ejecución del programa principal, en la instrucción siguiente a la instrucción que llamó a la subrutina (ACALL o LCALL). Las banderas no son afectadas.

EJEMPLO :

El apuntador de apilamiento originalmente contiene el valor 0BH. Las localidades de la RAM interna 0AH y 0BH contienen los valores 23H y 01H, respectivamente. La instrucción, **RET** dejará el apuntador de apilamiento con el valor 09H. La ejecución del programa continuará en la localidad 0123H.

BYTES: 1 CICLOS: 2

CÓDIGO DE OPERACIÓN:

0 0 1 0	0 0 1 0
---------	---------

OPERACIÓN: **RET**

(PC₁₅₋₈) \leftarrow ((SP))
(SP) \leftarrow (SP) - 1
(PC₇₋₀) \leftarrow ((SP))
(SP) \leftarrow (SP) - 1

RETI retorno de interrupción.

RETI extrae de la pila, los bytes de alto y bajo orden del PC y restablece la lógica de Interrupción que le permite aceptar interrupciones adicionales de la misma o menor prioridad. El apuntador de apilamiento es decrementado dos veces. Los otros registros no son afectados; el estado de pre-interrupción del PSW no es automáticamente resguardado. La ejecución del programa continúa a la dirección resultante, que generalmente es la instrucción siguiente al punto en el cual la interrupción fue detectada.

NOTA :

Una interrupción de más alta prioridad puede interrumpir el servicio de interrupción de una de baja prioridad. Si una interrupción de nivel bajo ó del mismo nivel de prioridad está pendiente cuando la instrucción RETI es ejecutada entonces una instrucción deberá ser ejecutada antes de que dicha interrupción pendiente sea procesada.

EJEMPLO :

El apuntador de apilamiento originalmente contiene el valor 0BH. Una interrupción fue detectada durante la instrucción finalizando en la localidad 0122H. La localidad de la RAM interna 0AH y 0BH contienen el valor 23H y 01H, respectivamente. La instrucción, **RETI** dejará el apuntador de apilamiento igual a 09H y regresará al programa ejecutando la localidad 0123H.

BYTES: 1 CICLOS: 2

CÓDIGO DE OPERACIÓN:

0 0 1 1	0 0 1 0
---------	---------

OPERACIÓN: **RETI**

(PC ₁₅₋₈)	←	((SP))
(SP)	←	(SP) - 1
(PC ₇₋₀)	←	((SP))
(SP)	←	(SP) - 1

RL. Los ocho bits en el acumulador son rotados un bit a la izquierda. El bit 7 es rotado a la posición del bit 0. Las banderas no son afectadas.

EJEMPLO :

El acumulador contiene el valor 0C5H (11000101B). La instrucción, **RLA** deja el acumulador conteniendo el valor 8BH (10001011B) con el acarreo no afectado.

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 0 1 0	0 0 1 1
---------	---------

OPERACIÓN: **RL**

$$\begin{array}{l} (An + 1) \\ (A0) \end{array} \leftarrow (An) \text{ } n = 0 - 6$$

RLC. Los ocho bits en el acumulador y la bandera de acarreo son rotados un bit a la izquierda. El bit 7 se mueve a la bandera de acarreo; el estado original de la bandera de acarreo se mueve a la posición del bit 0. Las banderas no son afectadas.

EJEMPLO :

El acumulador contiene el valor 0C5H (11000101B), y el acarreo es cero. La instrucción, **RLC A** deja el acumulador conteniendo el valor 8AH (10001010B) con el acarreo establecido.

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 0 1 1	0 0 1 1
---------	---------

OPERACIÓN: **RLC**

$$\begin{array}{l} (An + 1) \\ (A0) \\ (C) \end{array} \leftarrow \begin{array}{l} (An) \text{ } n = 0 - 6 \\ (C) \\ (A7) \end{array}$$

RR. Los ocho bits en el acumulador son rotados un bit a la derecha. El bit 0 es rotado a la posición del bit 7. Las banderas no son afectadas.

EJEMPLO :

El acumulador contiene el valor 0C5H (11000101B). La instrucción, **RR A** deja al acumulador conteniendo el valor 0E2H (11100010B) con el acarreo no afectado.

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 0 0 0	0 0 1 1
---------	---------

OPERACIÓN: **RR**

$$\begin{array}{l} (An) \\ (A7) \end{array} \leftarrow \begin{array}{l} (An + 1) \text{ } n = 0 - 6 \\ (A0) \end{array}$$

RRC A rota el acumulador a la derecha a través de la bandera de acarreo.

RRC. Los ocho bits en el acumulador y la bandera de acarreo son rotados un bit a la derecha. El bit 0 se mueve a la bandera de acarreo; el valor original de la bandera de acarreo se mueve a la posición del bit 7. Las otras banderas no son afectadas.

EJEMPLO :

El acumulador contiene el valor 0C5H (11000101B), el acarreo es cero. La instrucción, **RRC A** deja al acumulador conteniendo el valor 62 (01100010B) con el acarreo establecido.

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 0 0 1	0 0 1 1
---------	---------

OPERACIÓN: **RRC**

(An)	←	(An + 1) n = 0 - 6
(A7)	←	(C)
(C)	←	(A0)

SETB coloca el bit indicado en uno. SETB puede operar sobre la bandera de acarreo ó cualquier bit direccionable directamente. Las otras banderas no son afectadas.

EJEMPLO :

La bandera de acarreo es limpiada. El puerto 1 de salida ha sido escrito con el valor 34H (00110100B). Las instrucciones,

SETB C
SETB P1.0

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

SETB C

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1 1 0 1	0 0 1 1
---------	---------

OPERACIÓN: **SETB**

(C)	←	1
-----	---	---

SETB bit

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1 1 0 1	0 0 1 0	bit de dirección
---------	---------	------------------

OPERACIÓN: **SETB**

(bit) ← 1

SJMP. El programa de control salta incondicionalmente a la dirección indicada. El destino del salto es calculado por la suma del desplazamiento REL, más el contenido del Contador del Programa PC incrementado. El rango de destinaciones permitido es de 128 bytes hacia atrás y de 127 bytes hacia adelante.

EJEMPLO :

La etiqueta “RELAD” es asignado a una instrucción en la localidad del programa de memoria 0123H. La instrucción,

0100 8021 SJMP RELAD
102 ←EL PC APUNTA A ESTA LOCALIDAD

0123 7420 RELAD: MOV A,#20 ←NUEVA INSTRUCCIÓN A EJECUTAR

ensamblará en la localidad 0100H. Después la instrucción es ejecutada, el PC contendrá el valor 0123H.

NOTA :

Por debajo o por arriba de las condiciones la instrucción siguiente SJMP estará en 102H. Además, el byte de desplazamiento de la instrucción será el offset relativo (0123H-0102H) = 21H. De otra forma, un SJMP con un desplazamiento de 0FEH sería una instrucción de lazo infinito.

BYTES: 2 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 0 0 0	0 0 0 0	dirección relativa
---------	---------	--------------------

OPERACIÓN: **SJMP**

(PC) ← (PC) + 2

$$(PC) \leftarrow (PC) + rel$$

SUBB subtrae la variable indicada y la bandera de acarreo juntas del acumulador, dejando el resultado en el acumulador. **SUBB** establece la bandera de acarreo C, (borrow), si un préstamo es necesitado por el bit 7, por otro lado limpia C. AC es establecido si es necesitado un préstamo por el bit 3, de otra manera es limpiado. OV es establecido si es necesitado un préstamo por el bit 6, pero no por el bit 7, o por el bit 7, pero no por el bit 6.

Cuando se substraen enteros signados la señal de OV, indica un número negativo producido cuando un valor negativo es substraído de un valor positivo, o un resultado positivo cuando un número positivo es substraído de un número negativo.

El operando fuente permite cuatro modos de direccionamiento: Por registro, directo, registro-indirecto e inmediato.

EJEMPLO :

El acumulador contiene 0C9H (11001001B), el registro 2 contiene 54H (01010100B), y la bandera de acarreo es establecida. La instrucción, **SUBB A,R2** dejará el valor 74H (01110100B) en el acumulador, con la bandera de acarreo y AC limpiados pero OV establecido.

Note que 0C9H menos 54H es 75H. La diferencia entre éste y el resultado siguiente es debido a la bandera de acarreo (borrow) que ha sido establecida antes de la operación. Si el estado del acarreo no es conocido antes de empezar una substracción simple o de precisión múltiple, él debe de ser limpiado por una instrucción CLR C.

SUBB A,Rn

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1 0 0 1	1 r r r
---------	---------

OPERACIÓN: **SUBB**

$$(A) \leftarrow (A) - (C) - (Rn)$$

SUBB A,directo

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1 0 0 1	0 1 0 1	dirección directa
---------	---------	-------------------

OPERACIÓN: **SUBB**

$$(A) \leftarrow (A) - (C) - (\text{directo})$$

SUBB A,@Ri

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1 0 0 1	0 1 1 i
---------	---------

OPERACIÓN: **SUBB** $(A) \leftarrow (A) - (C) - ((Ri))$ **SUBB A,#dato**

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1 0 0 1	0 1 0 0	dato inmediato
---------	---------	----------------

OPERACIÓN: **SUBB** $(A) \leftarrow (A) - (C) - \#dato$

SWAP A intercambia los “NIBBLES” (campos de 4 bits) de alto y bajo orden del acumulador (bits 3-0 y bits 7-4). La operación puede también ser vista como una instrucción de rotación de 4 bits sin acarreo. Las banderas no son afectadas.

EJEMPLO:

El acumulador contiene el valor 0C5H (11000101B). La instrucción, **SWAP A** deja al acumulador conteniendo el valor 5CH (01011100B).

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1 1 0 0	0 1 0 0
---------	---------

OPERACIÓN: **SWAP** $(A_{3-0}) \rightarrow/\leftarrow (A_{7-4})$

XCH carga el acumulador con el contenido de la variable indicada, al mismo tiempo que escribe el contenido original del acumulador a la variable indicada. El operando fuente/destino puede usar direccionamiento por registro, directo, o registro indirecto.

SWAP A
intercambia la parte
alta y parte baja del
acumulador.

EJEMPLO:

R0 contiene la dirección 20H. El acumulador contiene el valor 3FH (00111111B). La localidad de la RAM interna 20H contiene el valor 75H (01110101B). La instrucción, **XCH A,@R0** dejará la localidad 20H de la RAM conteniendo los valores 3FH (00111111B) y 75H (01110101B) en el acumulador.

XCH A,Rn

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1 1 0 0	1 r r r
---------	---------

OPERACIÓN: **XCH**

(A) →/← (Rn)

XCH A,directo

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1 1 0 0	0 1 0 1
---------	---------

dirección directa

OPERACIÓN: **XCH**

(A) →/← (directo)

XCH A,@Ri

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1 1 0 0	0 1 1 i
---------	---------

OPERACIÓN: **XCH**

(A) →/← ((Ri))

XCHD intercambia el NIBBLE de bajo orden del acumulador (bits 3-0), con el NIBBLE bajo del dato de la RAM interna indirectamente direccionada por el registro especificado. Las banderas no son afectadas.

EJEMPLO:

R0 contiene la dirección 20H. El acumulador contiene el valor 36H (00110110B). La localidad 20H de la RAM interna contiene el valor 75H (01110101B). La instrucción, **XCHD A,@R0** dejará la localidad 20H de la RAM conteniendo el valor 76H (01110110B) y en el acumulador 35 (00110101B).

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1 1 0 1	0 1 1 i
---------	---------

OPERACIÓN: **XCHD**

$(A_{3-0}) \quad \rightarrow/\leftarrow \quad ((Ri_{3-0}))$

XRL ejecuta la operación lógica OR-Exclusivo entre las variable indicadas, guardando el resultado en el destino. Las banderas no son afectadas. Los dos operandos permiten 6 combinaciones de modos de direccionamiento. Cuando el destino es el acumulador, la fuente puede usar direccionamiento directo, por registro, por registro indirecto, e inmediato; cuando el destino es una dirección directa, la fuente puede ser el acumulador o un dato inmediato.

NOTA :

Cuando esta instrucción es usada para modificar un puerto de salida, el valor usado como dato original del puerto será leído del latch de salida, no de la terminal de entrada.

EJEMPLO:

Si el acumulador contiene 0C3H (11000011B) y el registro 0 contiene 0AAH (10101010B) entonces la instrucción, **XRL A,R0** dejará al acumulador conteniendo el valor 69H (01101001B).

Cuando el destino es un byte directamente direccionado, ésta instrucción puede complementar combinaciones de bits en cualquier localidad RAM interna o registro. El patrón de bits que será complementado es entonces determinado por un byte máscara, ya sea una constante contenida en la instrucción o una variable calculada en el acumulador en tiempo real .

La instrucción, **XRL P1,#00110001B** deberá complementar los bits 5, 4 y 0 del puerto de salida 1.

XRL A,Rn

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 1 1 0	1 r r r
---------	---------

OPERACIÓN: **XRL**

(A) ← (A) A (Rn)

XRL A,directo

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 1 1 0	0 1 0 1
---------	---------

dirección directa

OPERACIÓN: **XRL**

(A) ← (A) A (directo)

XRL A,@Ri

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 1 1 0	0 1 1 i
---------	---------

OPERACIÓN: **XRL**

(A) ← (A) A ((Ri))

XRL A,#DATA

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 1 1 0	0 1 0 0	dato inmediato
---------	---------	----------------

OPERACIÓN: **XRL**

(A) ← (A) A #dato

XRL directo,A

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 1 1 0	0 0 1 0	dirección directa
---------	---------	-------------------

OPERACIÓN: **XRL**

(directo) ← (directo) A (A)

XRL directo,#dato

BYTES: 3 CICLOS: 2

CÓDIGO DE OPERACIÓN:

0 1 1 0	0 0 1 1	dirección directa	dato inmediato
---------	---------	-------------------	----------------

OPERACIÓN: **XRL**

(directo) ← (directo) A #dato.