ANEXO PARA CAPITULO 3

LISTADO DE INSTRUCCIONES

A continuación se ilustra el repertorio de instrucciones que poseen los microcontroladores JL3/JK3/JK1. Este set de instrucciones esta explicado en forma muy detallada y posee una gran cantidad de ventajas sobre otras familias de microcontroladores como por ejemplo la División y la Multiplicación, Verificación de operaciones como mayor que, menor que, etc.

Para comprender mejor el repertorio de instrucciones, se explicara instrucción por instrucción ilustrando su sintaxis, la función que realiza, las variantes que puede presentar con los diferentes operandos y finalmente el número de ciclos que consume a la hora de su ejecución.

Para comenzar a analizar todo el repertorio hay que tener en cuenta la función que realizan, es por ello que se consideró pertinente clasificar las instrucciones según la función que realizan, por lo tanto se puede decir que las instrucciones se clasifican en:

- **Instrucciones para manejo de datos**
- **Instrucciones que manejan bits**
- **Instrucciones para control de programas**
- **Instrucciones aritméticas y lógicas**
- **Instrucciones de operandos especiales**

Instrucciones para manejo de datos

CLR : Borrar

Descripción: Esta instrucción adjudica cero como valor al Acumulador (**A**) o al registro índice (**X**) o a una dirección correspondiente a un registro de propósito general específico. La operación realizada es la siguiente:

A = 00h
ó
X = 00h
ó
(M) = 00h

M = Registro o posición de Memoria

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
CLR OPR	OPR = Dirección de Memoria o Registro	3
CLRA	A = Acumulador	1
CLRX	X = Registro Índice	1
CLRH		1
CLR OPR,X		3
CLR,X	SP = Stack Pointer	2

Ejemplos:

CLRA ; A = 0 ; Acumulador = 0

CLRX ; X = 0 ; Registro índice X = 0

CLR $80 ; Valor almacenado en la dirección 80h = 0

LDA : Carga el Acumulador desde la memoria

Descripción: Esta instrucción carga en el Acumulador (**A**) un valor almacenado a manera de una constante o previamente establecido en un registro de propósito general. La operación realizada es la siguiente:

A = (M)

M = Constante o valor almacenado en un Registro o posición de Memoria

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
LDA #OPR	#OPR = Constante.	2
LDA OPR	OPR = Dirección de memoria o registro	3
LDA OPR,X		3
LDA ,X	X = Registro Índice	2
LDA OPR,SP	SP = Stack Pointer o Apuntador de Pila	4

Ejemplos:

```
LDA #10T        ; A = 10 ; Acumulador = 10

MOV #10T,$80    ; Almacena el valor 10 en la dirección 80h
LDA $80         ; A = 10 ; Carga en A el valor existente en la
                  dirección 80h

MOV #10T,$81    ; Almacena el valor 10 en la dirección 81h
LDX  #1         ; X=1
LDA  $80,X      ; A = 10, Carga en A el valor existente en la
                  dirección 81h ( 80h + X )
```

LDHX : Carga el registro de índice H:X con la memoria

Descripción: Esta instrucción carga en el Registro índice (**H:X**) un valor almacenado a manera de una constante o previamente establecido en un registro de propósito general. Hay que tener en cuenta que el registro **H:X** es un registro de 16 bits. Cuando se indica que se desea almacenar en **H:X** un dato almacenado previamente en una posición de memoria y como los registros son de 8 Bits, en la dirección que se indica en la instrucción (M) se encuentran los 8 bits de más peso y en la dirección siguiente (M+1) se encuentran los 8 bits de menos peso. La operación realizada es la siguiente:

H:X = (M:M+1)

M = Constante o valor almacenado en un Registro o posición de Memoria

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS

LDHX #OPR	#OPR = Constante.	3
LDHX OPR	OPR = Dirección de memoria o registro	4

Ejemplos:

LDHX #$2875 ; H:X = 2875h (10357 d)

MOV #$28,$80 ; Almacena el valor 28h en la dirección 80h
MOV #$75,$81 ; Almacena el valor 75h en la dirección 81h
LDHX $80 ; H:X = 2875h (10357 d)

LDX : Carga el registro de índice X con la memoria

Descripción: Esta instrucción carga en el Acumulador (**A**) un valor almacenado a manera de una constante o previamente establecido en un registro de propósito general. La operación realizada es la siguiente:

X = (M)

M = Constante o valor almacenado en un Registro o posición de Memoria

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
LDX #OPR	#OPR = Constante.	2
LDX OPR	OPR = Dirección de memoria o registro	3
LDX OPR,X		3
LDX ,X	X = Registro Índice	2
LDX OPR,SP	SP = Stack Pointer o Apuntador de Pila	4

Ejemplos:

LDX #10T ; X = 10 ; Registro índice = 10

MOV #10T,$80 ; Almacena el valor 10 en la dirección 80h
LDX $80 ; X = 10 ; Carga en X el valor existente en la dirección 80h

MOV #10T,$81 ; Almacena el valor 10 en la dirección 81h
LDX #1 ; X=1
LDX $80,X ; X = 10, Carga en X el valor existente en la dirección 81h (80h + X)

MOV : Mover

Descripción: Esta instrucción permite mover un contenido de un lugar a otro, puede ser de un registro a otro o una constante a un registro. La operación realizada es la siguiente:

$$(M)_{DESTINO} = (M)_{FUENTE}$$

M = Constante o valor almacenado en un Registro o posición de Memoria

Sintaxis: MOV FUENTE, DESTINO

INSTRUCCIÓN	Aclaración	No. CICLOS
MOV OPR,OPR	Registro a Registro	5
MOV OPR,X+		4
MOV #OPR,OPR	Constante a registro	4
MOV X+,OPR		4

Ejemplos:

MOV #10T,$80 ; Almacena el valor 10 en la dirección 80h

MOV #10T,$80 ; Almacena el valor 10 en la dirección 80h
MOV $80,$81 ; Almacena el valor de la dirección 80h en la dirección 81h, (Dirección 81h = 10)

NSA : Intercambia los nibbles del Acumulador

Descripción: Esta instrucción realiza un cambio en los nibbles presentes en el registro Acumulador (**A**). Los nibbles son los 4 bits de más peso o los 4 bits de menos peso. El intercambio de nibbles consiste en intercambiar entre sí el contenido de los nibbles presentes en el Acumulador, es decir, que los 4 bits de más peso pasan a ser ahora los 4 bits de menos peso y viceversa. La operación realizada es la siguiente:

Inicialmente **A = A[7:4] : A[3:0]**

Al ejecutar la instrucción **A = A[0:3] : A[7:4]**

A[7:4] = 4 Bits de más peso
A[0:3] = 4 Bits de menos peso

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
NSA	Intercambiar Nibles de A (A=(A[3:0]:A[7:4]))	3

Ejemplos:

```
LDA #$F8        ; A = F8h
NSA             ; A = 8Fh

LDA #$75        ; A = 75h
NSA             ; A = 57h
```

PSHA : Pone el Acumulador en la pila

Descripción: Esta instrucción carga el Acumulador (**A**) en la dirección almacenada en el puntero de Pila (Stack) y el puntero de pila es decrementado para apuntar a la dirección siguiente disponible. La operación realizada es la siguiente:

(SP) = A
SP = SP – 0001h

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
PSHA	Insertar A en el Stack	2

Ejemplos:

```
LDA #10T        ; A = 10; Acumulador = 10
PSHA            ; Guarda el valor del Acumulador en la
                  dirección disponible del Snack, es decir,
                  guarda el 10 en el Stack o Pila
```

PSHH : Pone la parte alta del registro H:X (H) en la pila

Descripción: Esta instrucción carga la parte alta del registro índice (**H:X**) en la dirección almacenada en el puntero de Pila (Stack) y el puntero de pila es decrementado para apuntar a la dirección siguiente disponible. La operación realizada es la siguiente:

(SP) = H

SP = SP – 0001h

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
PSHH	Insertar H en el Stack	2

Ejemplos:

```
LDHX #$2875   ; H:X = 2875h; H=28h, X=75h
PSHH          ; Guarda el valor correspondiente a la parte
              alta del registro H:X en la dirección disponible
              del Stack, es decir, guarda el 28h en el Stack o
              Pila
```

PSHX : Pone la parte baja del registro H:X (X) en la pila

Descripción: Esta instrucción carga la parte baja del registro índice (**H:X**) en la dirección almacenada en el puntero de Pila (Stack) y el puntero de pila es decrementado para apuntar a la dirección siguiente disponible. La operación realizada es la siguiente:

(SP) = X
SP = SP – 0001h

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
PSHX	Insertar X en el Stack	2

Ejemplos:

```
LDHX #$2875   ; H:X = 2875h; ; H=28h, X=75h
PSHX          ; Guarda el valor correspondiente a la parte
              baja del registro H:X en la dirección disponible
              del Stack, es decir, guarda el 75h en el Stack o
              Pila
```

PULA : Saca el Acumulador de la pila

Descripción: Esta instrucción carga en el Acumulador (**A**) con el valor almacenado en la última dirección apuntada por la Pila (Stack) y el puntero de pila es incrementado para apuntar a la dirección siguiente. La operación realizada es la siguiente:

A = (SP)
SP = SP + 0001h

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
PULA	Sacar A del Stack	2

Ejemplos:

```
LDA #10T      ; A = 10; Acumulador = 10
PSHA          ; Guarda el valor del Acumulador en la
              dirección disponible del Snack, es decir,
              guarda el 10 en el Stack o Pila
LDA #20T      ; A = 20; Acumulador = 20
PULA          ;Saca el última valor almacenado en la pila y lo
              guarda en el Acumulador, para el caso, A=10
```

PULH : Saca la parte alta del registro H:X (H) de la pila

Descripción: Esta instrucción carga la parte alta del registro índice (**H:X**) con el valor almacenado en la última dirección apuntada por la Pila (Stack) y el puntero de pila es incrementado para apuntar a la dirección siguiente. La operación realizada es la siguiente:

H = (SP)
SP = SP + 0001h

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
PULH	Sacar H del Stack	2

Ejemplos:

```
LDHX #$2875   ; H:X = 2875h; H=28h, X=75h
PSHH          ; Guarda el valor correspondiente a la parte
              alta del registro H:X en la dirección disponible
              del Stack, es decir, guarda el 28h en el Stack o
              Pila
LDHX #$3520   ; H:X = 3520h; H=35h, X=20h
PULH          ;Saca el última valor almacenado en la pila y lo
              guarda en la parte alta del registro índice
              (H:X), para el caso, H=28h
```

PULX : Saca la parte baja del registro H:X (X) de la pila

Descripción: Esta instrucción carga la parte baja del registro índice (**H:X**) con el valor almacenado en la última dirección apuntada por la Pila (Stack) y el puntero de pila es incrementado para apuntar a la dirección siguiente. La operación realizada es la siguiente:

X = (SP)
SP = SP + 0001h

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
PULX	Sacar X del Snack	2

Ejemplos:

```
LDHX #$2875   ; H:X = 2875h; H=28h, X=75h
PSHX          ; Guarda el valor correspondiente a la parte
              baja del registro H:X en la dirección disponible
              del Stack, es decir, guarda el 75h en el Stack o
              Pila
LDHX #$3520   ; H:X = 3520h; H=35h, X=20h
PULX          ;Saca el última valor almacenado en la pila y lo
              guarda en la parte baja del registro índice
              (H:X), para el caso, X=75h
```

STA : Guarda el Acumulador en la memoria

Descripción: Esta instrucción carga el Acumulador (**A**) en un registro de propósito general. La operación realizada es la siguiente:

(M) = A

M = Registro o posición de Memoria

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
STA OPR	OPR = Dirección de memoria o registro	3
STA OPR,X		4
STA ,X	X = Registro Índice	2
STA OPR,SP	SP = Stack Pointer o Apuntador de Pila	4

Ejemplos:

```
LDA #10T      ; A = 10; Acumulador = 10
```

```
STA $80      ; Guarda en la dirección 80h el valor del
             Acumulador; Dirección 80h = 10

LDA #10T     ; A = 10 ; Acumulador = 10
LDX  #1      ; X=1
STA  $80,X   ; Carga en la dirección 81h ( 80h + X ), el valor
             presente en el Acumulador. Dirección 81h=10
```

STHX : Guarda el registro índice

Descripción: Esta instrucción carga el valor presente en el Registro índice (**H:X**) en un registro de propósito general. Hay que tener en cuenta que el registro **H:X** es un registro de 16 bits. Cuando se indica que se desea almacenar **H:X** en una posición de memoria y como los registros son de 8 Bits, en la dirección que se indica en la instrucción (M) se encuentran los 8 bits de más peso y en la dirección siguiente (M+1) se encuentran los 8 bits de menos peso. La operación realizada es la siguiente:

$$(M:M+1) = H:X$$

M = Constante o valor almacenado en un Registro o posición de Memoria

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
STHX OPR	Asignar H:X en el registro OPR	4

Ejemplo:

```
LDHX #$2875  ; H:X = 2875h (10357 d)
STHX  $80    ; Almacena el valor 28h en la dirección 80h
             ; Almacena el valor 75h en la dirección 81h
```

STX : Guarda el registro índice X en la memoria

Descripción: Esta instrucción carga el registro índice (**X**) en un registro de propósito general. La operación realizada es la siguiente:

(M) = X

M = Registro o posición de Memoria

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
STX OPR	OPR = Dirección de memoria o registro	3
STX OPR,X		4
STX ,X	X = Registro Índice	2
STX OPR,SP	SP = Stack Pointer o Apuntador de Pila	4

Ejemplos:

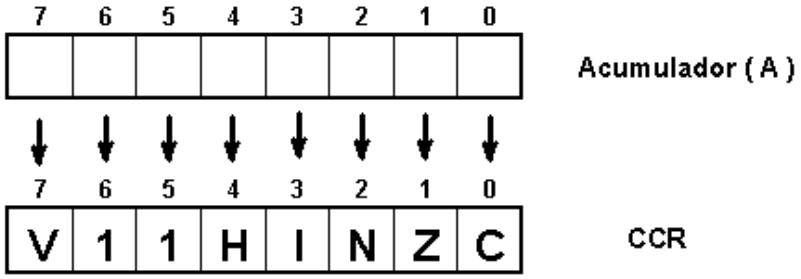
```
LDX #10T      ; X = 10; Registro índice = 10
STX $80       ; Guarda en la dirección 80h el valor del
              registro índice (X); Dirección 80h = 10

LDX  #1       ; X=1
STA  $80,X    ; Carga en la dirección 81h ( 80h + X ), el valor
              presente en el registro índice. Dirección 81h=1
```

TAP : Transfiere el Acumulador (A) al Registro de código de condición (CCR)

Descripción: Esta instrucción carga el registro de código de condición (**CCR**) con el valor almacenado en el Acumulador (A). El registro **CCR** es aquel en el que se encuentran las banderas de Acarreo (**C**), Cero (**Z**), Negativo (**N**), etc. Para los que han programado microcontroladores PIC es el equivalente al registro **STATUS.** La operación realizada es la siguiente:

CCR = A



Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS

TAP	Transferir A al CCR , CCR = A	2

Ejemplos:

```
LDA %01100000      ; A = %01100000 (96 d)
TAP                ; Guarda en el registro CCR, el valor presente
                   en el Acumulador (A); CCR = %01100000
```

TAX: Trasfiere el Acumulador (A) al registro de índice (X)

Descripción: Esta instrucción carga el registro índice (**X**) con el valor almacenado en el Acumulador (**A**). La operación realizada es la siguiente:

X = A

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
TAX	Transferir A a X , X = A	1

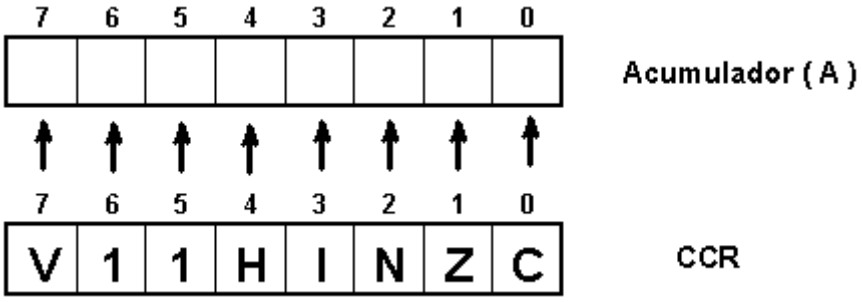
Ejemplos:

```
LDA #25T       ; A = 25
TAX            ; X = 25
```

TPA : Transfiere el registro de código de condición (CCR) al Acumulador (A)

Descripción: Esta instrucción carga el Acumulador (A) con el valor almacenado en el registro de código de condición (**CCR**). El registro **CCR** es aquel en el que se encuentran las banderas de Acarreo (**C**), Cero (**Z**), Negativo (**N**), etc. Para los que han programado microcontroladores PIC es el equivalente al registro **STATUS.** La operación realizada es la siguiente:

A = CCR



Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
TPA	Transferir CCR a A ; A = CCR	1

Ejemplos:

```
LDA %01100000     ; A = %01100000 (96 d)
TAP               ; Guarda en el registro CCR, el valor
                    presente en el Acumulador (A);
                    CCR=%01100000
LDA %11110000     ; A = %11110000 (240 d)
TPA               ; Guarda en el Acumulador (A), el valor
                    presente en el registro (CCR);
                    A=%01100000
```

TSX : Transfiere el puntero de pila al registro de índice (H:X)

Descripción: Esta instrucción carga el registro índice (H:X) con el último valor almacenado en la pila (**X**) incrementado en 1. La operación realizada es la siguiente:

$$H:X = (SP) + 1$$

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
TSX	Transferir SP a H:X , H:X = SP + 1	2

Ejemplos:

```
LDA #$35      ; A = 35h
PSHA          ; SP = 35h, Guarda el valor de A en la Pila
TSX           ; H:X = 0036h = 35h + 1
```

TXA : Transfiere el registro de índice (X) al Acumulador (A)

Descripción: Esta instrucción carga el Acumulador (**A**) con el valor almacenado en el registro índice (**X**). La operación realizada es la siguiente:

A = X

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
TXA	Transferir X a A , A = X	1

Ejemplos:

```
LDX #35T        ; X = 35
TXA             ; A = 35
```

TXS: Transfiere el registro de índice (H:X) al puntero de pila

Descripción: Esta instrucción carga en la Pila el registro índice (**H:X**) decrementado en 1. La operación realizada es la siguiente:

(SP) = H:X - 1

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
TXS	Transferir H:X a SP , SP = H:X - 1	2

Ejemplos:

```
LDHX #$2875   ; H:X = 2875h
TXS           ; Guarda en la pila el valor presente en el
              registro H:X-1; (SP) = 2875h – 1 = 2874h
```

INSTRUCCIONES QUE MANEJAN BITS

BCLR n : Pone en cero un bit en la memoria

Descripción: Esta instrucción pone en "0" el Bit n del registro (M). Para los que han programado microcontroladores PIC es la instrucción equivalente a **BCF.** La operación realizada es la siguiente:

(M)n = 0

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
BCLR n,OPR	Borrar el bit n del registro OPR	4

Ejemplos:

```
MOV #%11110000,$80    ; Dirección 80h = %11110000
BCLR 7,$80            ; Dirección 80h = %01110000

MOV #%11110000,$80    ; Dirección 80h = %11110000
BCLR 6,$80            ; Dirección 80h = %10110000

MOV #%11110000,$80    ; Dirección 80h = %11110000
BCLR 5,$80            ; Dirección 80h = %11010000
```

BSET n: Poner en 1 el bit n

Descripción: Esta instrucción pone en "1" el Bit n del registro (M). Para los que han programado microcontroladores PIC es la instrucción equivalente a **BSF**. La operación realizada es la siguiente:

(M)n = 1

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
BSET n,OPR	Poner en 1 el bit n del registro OPR	4

Ejemplos:

```
MOV #%11110000,$80    ; Dirección 80h = %11110000
BSET 0,$80            ; Dirección 80h = %11110001

MOV #%11110000,$80    ; Dirección 80h = %11110000
BSET 1,$80            ; Dirección 80h = %11110010

MOV #%11110000,$80    ; Dirección 80h = %11110000
BSET 2,$80            ; Dirección 80h = %11110100
```

BIT : Bit de prueba de la memoria con el Acumulador

Descripción: Efectúa una comparación lógica AND entre dos cantidades; la primera cantidad corresponde al Acumulador (Registro A), la segunda cantidad puede ser una constante o un dato

almacenado previamente en un registro y la respuesta no afecta a ninguna de los dos registros. La operación realizada es la siguiente:

Comparación = A & (M)

M = Constante o Valor almacenado en una posición de Memoria

Los únicos Bits afectados son Z y N así:

N : tomará valor 1 si al efectuar la operación el Bit de más peso es "1", de lo contrario tomará valor "0"
Z : tomará valor "1" si al efectuar la operación el resultado es 00h, de lo contrario tomará valor "0"

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
BIT #OPR	#OPR = Constante.	2
BIT OPR	OPR = Dirección de memoria o registro	3
BIT OPR,X		3
BIT ,X	X = Registro Índice	2
BIT OPR,SP	SP = Stack Pointer o Apuntador de Pila	4

Ejemplos:

LDA #%10101010 ; A = 10101010 (170 d), Acumulador=170
BIT #%11110000 ; Comparación = A & (11110000)

Entonces Comparación = 10100000 (160 d) ; Z=0, N=1

MOV #$F0,$80 ; Almacena F0h (240 d) en la dirección 80h
LDA #$0F ; A = 0Fh (15 d)
BIT $80 ; Comparación = A & (Valor en dirección 80h)

Entonces Comparación = 00h (0 d) ; Z=1, N=0

MOV #$F0,$82 ; Almacena F0 en la dirección 82h
LDX #2 ; X=2
LDA #$0F ; A = 0Fh (15 d)
AND $80,X ; Comparación = A & (Valor en dirección 82h)

Entonces Comparación = 00h (0 d) ; Z=1, N=0

CLC : Pone en cero el bit de acarreo

Descripción: Esta instrucción pone en "0" el Bit de Acarreo presente en el registro (CCR). Esta instrucción puede ser utilizada para

preparar el Bit de Acarreo antes de utilizar una instrucción de desplazamiento o rotación que involucre este Bit. La operación realizada es la siguiente:

Bit C = 0

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
CLC	Borrar el Bit de Carry	1

Ejemplo:

CLC ; Carry = 0

CLI : Pone en cero el bit de mascara de la interrupción

Descripción: Esta instrucción pone en "0" el Bit de máscara de Interrupción presente en el registro (CCR). Cuando este Bit está en "0", el microcontrolador habilita el servicio de interrupciones. La operación realizada es la siguiente:

Bit I = 0

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
CLI	Borrar el Bit de Interrupción	2

Ejemplo:

CLI ; Máscara de Interrupción (I) = 0

SEC : Pone en 1 el bit de acarreo

Descripción: Esta instrucción pone en "1" el Bit de Acarreo presente en el registro (CCR). Esta instrucción puede ser utilizada para preparar el Bit de Acarreo antes de utilizar una instrucción de desplazamiento o rotación que involucre este Bit. La operación realizada es la siguiente:

Bit C = 1

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
SEC	Colocar el bit de Carry en 1	1

Ejemplo:

SEC ; Carry = 1

SEI : Pone en 1 el bit de la mascara de interrupcion

Descripción: Esta instrucción pone en "1" el Bit de máscara de Interrupción presente en el registro (CCR). Cuando este Bit está en "1", el microcontrolador deshabilita el servicio de interrupciones. La operación realizada es la siguiente:

Bit I = 1

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
SEI	Colocar el bit de Interrupción en 1 I = 1	2

Ejemplo:

SEI ; Máscara de Interrupción (I) = 1

INSTRUCCIONES PARA CONTROL DE PROGRAMAS

BCC : Saltar si el bit de acarreo esta en 0

Descripción: Esta instrucción verifica el estado del Bit de Acarreo presente en el registro (CCR). Si al verificar el estado del Bit está en "0", el microcontrolador salta a la etiqueta deseada por el usuario. La operación realizada es la siguiente:

PC = PC + 0002 + Dirección de Etiqueta, Si C = 0

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
BCC Etiqueta	Saltar a la etiqueta si el Bit de Carry es 0	3

Ejemplo:

LDA #20T ; A = 20, Acumulador = 20
SUB #10T ; A = 20 - 10

BCC SALTO1 Como para el caso A (20d) es mayor que la segunda cantidad (10d), el resultado de la diferencia es positivo y el Bit de Acarreo (C) toma como valor "0"; por ello el programa salta a la etiqueta "SALTO1", por de lo contrario, no se realiza el salto y se continúa en la línea siguiente.

BCS : Saltar si el bit de acarreo está en 1

Descripción: Esta instrucción realiza el proceso contrario a la instrucción anterior, verifica el estado del Bit de Acarreo presente en el registro (CCR). Si al verificar el estado del Bit está en "1", el microcontrolador salta a la etiqueta deseada por el usuario. La operación realizada es la siguiente:

PC = PC + 0002 + Dirección de Etiqueta, Si C = 1

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
BCS Etiqueta	Saltar a la etiqueta si el Bit de Carry es 1	3

Ejemplo:

LDA #10T **;** A = 10, Acumulador = 10
SUB #20T **; A** = 10 - 20
BCS SALTO1 Como para el caso A (10d) es menor que la segunda cantidad (20d), el resultado de la diferencia es negativa y el Bit de Acarreo (C) toma como valor "1"; por ello el programa salta a la etiqueta "SALTO1", por de lo contrario, no se realiza el salto y se continúa en la línea siguiente.

BEQ : Saltar si es igual

Descripción: Realiza un salto a una etiqueta o línea definida por el usuario si al efectuar una comparación entre dos cantidades son iguales. Esta instrucción prueba el estado del Bit Z presente en el registro CCR y si su estado es "1", realiza el salto a la línea establecida. La operación realizada es la siguiente:

PC = PC +0002 + Dirección , Si Z=1

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
BEQ ETIQ	Saltar a la etiqueta si es igual , Z=1	3

Ejemplo:

```
LDA   #40T      ; A = 40, Acumulador = 40
CMP   #40T      ; Compara A con 40
BEQ   Salto1    ; Si son iguales salta a la etiqueta "Salto1", de
                  lo contrario sigue en la siguiente línea, como
                  para el caso son iguales se produce el salto.
```

BGE : Saltar si es mayor que o igual a (Operandos con signo)

Descripción: Realiza un salto a una etiqueta o línea definida por el usuario si al efectuar una comparación entre dos cantidades da como resultado que la primera cantidad es mayor o igual que la segunda cantidad. Hay que tener en cuenta que esta instrucción tiene en cuenta los signos presentes en ambas cantidades. La operación realizada es la siguiente:

PC = PC +0002 + Dirección , Si N ⊕ V=0

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
BGE ETQ	Saltar si es Mayor o Igual	3

Ejemplo:

```
LDA   #50T      ; A = 50, Acumulador = 50
CMP   #40T      ; Compara A con 40
BGE   Salto1    ; Si el Acumulador es mayor o igual que la
                  segunda cantidad salta a la etiqueta "Salto1",
                  de lo contrario sigue en la siguiente línea,
                  como para el caso e mayor se produce el salto.
```

BGT : Saltar si es mayor que (Operandos con signo)

Descripción: Realiza un salto a una etiqueta o línea definida por el usuario si al efectuar una comparación entre dos cantidades da como

resultado que la primera cantidad es mayor que la segunda cantidad. Esta instrucción realiza la misma función que BHI, con la diferencia de que se tiene en cuenta a la hora de la verificación el estado del Bit Z o del resultado de la operación de **N ⊕ V** correspondiente a los Bits N y V del registro CCR, si Z o **N ⊕ V** es Cero, se produce el salto. La operación realizada es la siguiente:

PC = PC +0002 + Dirección , Si Z o N ⊕ V=0

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
BGT ETQ	Saltar si es Mayor que (Operandos con Signo)	3

Ejemplo:

```
LDA   #90T      ; A = 90, Acumulador = 90
CMP   #40T      ; Compara A con 40
BGT   Salto1    ; Si el Acumulador es mayor que la segunda
                  cantidad salta a la etiqueta "Salto1", de lo
                  contrario sigue en la siguiente línea, como
                  para el caso e mayor se produce el salto.
```

BHCC : Saltar si el bit de carry medio es cero

Descripción: Esta instrucción verifica el estado del Bit de Acarreo medio (**H**) presente en el registro (**CCR**). Si al verificar el estado del Bit está en "0", el microcontrolador salta a la etiqueta deseada por el usuario. La operación realizada es la siguiente:

PC = PC + 0002 + Dirección de Etiqueta, Si H = 0

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
BHCC Etiqueta	Saltar a la etiqueta si el Bit de Carry Medio es 0	3

Ejemplo:

```
LDA  #%10001000      ; A = %10001000
```

ADD #%01010000 **; A** = A + %0101**0**000
BHCC SALTO1 ;Observese que al efectuar la suma de estas dos cantidades cuando se suman los Bits 3 (bits sombreados) de ambas cantidades se produce un acarreo, este acarreo es el que se conoce como Acarreo Medio (H), como para el caso este acarreo tiene como valor "0",se produce un salto a la etiqueta "SALTO1".

BHCS : Saltar si el bit de carry medio es 1

Descripción: Esta instrucción verifica el estado del Bit de Acarreo medio (**H**) presente en el registro (**CCR**). Si al verificar el estado del Bit está en "1", el microcontrolador salta a la etiqueta deseada por el usuario. La operación realizada es la siguiente:

PC = PC + 0002 + Dirección de Etiqueta, Si H = 1

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
BHCC Etiqueta	Saltar a la etiqueta si el Bit de Carry Medio es 0	3

Ejemplo:

LDA #%10001000 **;** A = %1000**1**000
ADD #%01011000 **; A** = A + %0101**1**000
BHCC SALTO1 ;Observese que al efectuar la suma de estas dos cantidades cuando se suman los Bits 3 (bits sombreados) de ambas cantidades se produce un acarreo, este acarreo es el que se conoce como Acarreo Medio (H), como para el caso este acarreo tiene como valor "1",se produce un salto a la etiqueta "SALTO1".

BHI : Saltar si es mayor

Descripción: Realiza un salto a una etiqueta o línea definida por el usuario si al efectuar una comparación entre dos cantidades da como resultado que la primera cantidad es mayor que la segunda cantidad. Esta instrucción realiza la verificación de acuerdo al estado de los Bits

Z o C del registro CCR, si cualquiera de estos dos bits es Cero, se produce el salto. La operación realizada es la siguiente:

PC = PC +0002 + Dirección , Si C o Z=0

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
BHI ETQ	Saltar si es Mayor	3

Ejemplo:

```
LDA   #50T      ; A = 50, Acumulador = 50
CMP   #40T      ; Compara A con 40
BHI   Salto1    ; Si el Acumulador es mayor que la segunda
                  cantidad salta a la etiqueta "Salto1", de lo
                  contrario sigue en la siguiente línea, como
                  para el caso e mayor se produce el salto.
```

BHS : Saltar si es mayor o igual

Descripción: Realiza un salto a una etiqueta o línea definida por el usuario si al efectuar una comparación entre dos cantidades da como resultado que la primera cantidad es mayor o igual que la segunda cantidad. Esta instrucción realiza la misma operación que BGE pero basándose no en el estado de los Bits N y V sino en el estado del Acarreo (C). La operación realizada es la siguiente:

PC = PC +0002 + Dirección , Si C=0

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
BHS ETQ	Saltar si es Mayor o Igual	3

Ejemplo:

```
LDA   #50T      ; A = 50, Acumulador = 50
```

CMP #40T ; Compara A con 40
BHS Salto1 ; Si el Acumulador es mayor o igual que la segunda cantidad salta a la etiqueta "Salto1", de lo contrario sigue en la siguiente línea, como para el caso e mayor se produce el salto.

BIH : Saltar si el pin IRQ está en 1

Descripción: Esta instrucción verifica el estado del Pin IRQ. Si al verificar el estado del Bit está en "1", el microcontrolador salta a la etiqueta deseada por el usuario. La operación realizada es la siguiente:

PC = PC + 0002 + Dirección de Etiqueta, Si IRQ = 1

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
BIH Etiqueta	Saltar a la etiqueta si el Pin IRQ es 1	3

Ejemplo:

BIH SALTO1 ; Como IRQ es un pin de entrada, esta instrucción es muy util a la hora de verificar su estado; si en el momento de ejecución su estado es "1", el microcontrolador salta a la etiqueta "SALTO1"

BIL : Saltar si el pin IRQ está en 0

Descripción: Esta instrucción verifica el estado del Pin IRQ. Si al verificar el estado del Bit está en "0", el microcontrolador salta a la etiqueta deseada por el usuario. La operación realizada es la siguiente:

PC = PC + 0002 + Dirección de Etiqueta, Si IRQ = 0

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
BIL Etiqueta	Saltar a la etiqueta si el Pin IRQ es 0	3

Ejemplo:

BIL SALTO1 ; Como IRQ es un pin de entrada, esta instrucción es muy util a la hora de verificar su estado; si en el momento de ejecución su estado es "0", el microcontrolador salta a la etiqueta "SALTO1"

BLE : Saltar si es menor que o igual a (Operándos con signo)

Descripción: Realiza un salto a una etiqueta o línea definida por el usuario si al efectuar una comparación entre dos cantidades da como resultado que la primera cantidad es menor o igual que la segunda cantidad. Esta instrucción realiza la misma función que BLS, con la diferencia de que se tiene en cuenta a la hora de la verificación el estado del Bit Z o del resultado de la operación de **N** ⊕ **V** correspondiente a los Bits N y V del registro CCR, si Z o **N** ⊕ **V** es "1", se produce el salto. La operación realizada es la siguiente:

PC = PC +0002 + Dirección , Si Z o N ⊕ V=1

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
BLE ETQ	Saltar si es Mayor que (Operandos con Signo)	3

Ejemplo:

```
LDA   #40T      ; A = 40, Acumulador = 40
CMP   #90T      ; Compara A con 90
BLE   Salto1    ; Si el Acumulador es menor que la segunda
                  cantidad salta a la etiqueta "Salto1", de lo
                  contrario sigue en la siguiente línea, como
                  para el caso e menor se produce el salto.
```

BLO : Saltar si es menor

Descripción: Realiza un salto a una etiqueta o línea definida por el usuario si al efectuar una comparación entre dos cantidades da como resultado que la primera cantidad es menor que la segunda cantidad. Esta instrucción realiza la verificación de acuerdo al estado del Bit C del registro CCR, si éste bit es "1", se produce el salto. La operación realizada es la siguiente:

PC = PC +0002 + Dirección , Si C =1

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
BLO ETQ	Saltar si es Menor	3

Ejemplo:

```
LDA   #40T      ; A = 40, Acumulador = 40
CMP   #50T      ; Compara A con 50
BLO   Salto1    ; Si el Acumulador es menor que la segunda
                  cantidad salta a la etiqueta "Salto1", de lo
                  contrario sigue en la siguiente línea, como
                  para el caso e menor se produce el salto.
```

BLS :Saltar si es menor o igual

Descripción: Realiza un salto a una etiqueta o línea definida por el usuario si al efectuar una comparación entre dos cantidades da como resultado que la primera cantidad es menor o igual que la segunda cantidad. Esta instrucción realiza la verificación de acuerdo al estado de los Bits Z o C del registro CCR, si cualquiera de estos dos bits es "1", se produce el salto. La operación realizada es la siguiente:

PC = PC +0002 + Dirección , Si C o Z=1

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
BLS ETQ	Saltar si es Menor o igual	3

Ejemplo:

LDA #40T **; A** = 40, **Acumulador** = 40
CMP #50T **; Compara A con 50**
BLS Salto1 **; Si el Acumulador es menor o igual que la segunda cantidad salta a la etiqueta "Salto1", de lo contrario sigue en la siguiente línea, como para el caso e menor se produce el salto.**

BLT : Saltar si es menor que (Operándos con signo)

Descripción: Realiza un salto a una etiqueta o línea definida por el usuario si al efectuar una comparación entre dos cantidades da como resultado que la primera cantidad es menor que la segunda cantidad. Esta instrucción realiza la misma función que BLO, con la diferencia de que se tiene en cuenta a la hora de la verificación el resultado de la operación de **N** $\oplus$ **V** correspondiente a los Bits N y V del registro CCR, si **N** $\oplus$ **V** es "1", se produce el salto. La operación realizada es la siguiente:

PC = PC +0002 + Dirección , Si N $\oplus$ **V=1**

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
BLT ETQ	Saltar si es Menor que (Operandos con Signo)	3

Ejemplo:

LDA #70T **; A** = 70, **Acumulador** = 70
CMP #80T **; Compara A con 80**
BLT Salto1 **; Si el Acumulador es menor que la segunda cantidad salta a la etiqueta "Salto1", de lo contrario sigue en la siguiente línea, como para el caso e menor se produce el salto.**

BMC : Saltar si la bandera de interrupción es 0

Descripción: Esta instrucción verifica el estado del Bit de interrupción (**I**) presente en el registro (**CCR**). Si al verificar el estado

del Bit está en "**0**", el microcontrolador salta a la etiqueta deseada por el usuario. La operación realizada es la siguiente:

PC = PC + 0002 + Dirección de Etiqueta, Si I = 0

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
BMC Etiqueta	Saltar a la etiqueta si el Bit I del CCR es 0	3

Ejemplo:

BMS SALTO1 ; Si en el momento de ejecución su estado es "0", el microcontrolador salta a la etiqueta "SALTO1"

BMI : Saltar si es Negativo

Descripción: Esta instrucción salta a una etiqueta en particular siempre y cuando después de realizar una operación el resultado sea Negativo, reflejandose según el estado del Bit de Negativo (N) presente en el registro (CCR). Si al verificar el estado del Bit está en "1", el microcontrolador salta a la etiqueta deseada por el usuario. La operación realizada es la siguiente:

PC = PC + 0002 + Dirección de Etiqueta, Si N = 1

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
BMI ETIQ	Saltar si el resultado de una operación es Negativo	3

Ejemplo:

LDA #10T **;** A = 10, Acumulador = 10
SUB #20T **; A** = 10 - 20
BMI SALTO1 Como para el caso A (10d) es menor que la segunda cantidad (20d), el resultado de la diferencia es Negativo y el Bit de Negativo (N) toma como valor "1"; por ello el programa salta a la etiqueta "SALTO1", por de lo contrario, no se realiza el salto y se continúa en la línea siguiente.

BMS : Saltar si la bandera de interrupción es 1

Descripción: Esta instrucción verifica el estado del Bit de interrupción (**I**) presente en el registro (**CCR**). Si al verificar el estado del Bit está en "1", el microcontrolador salta a la etiqueta deseada por el usuario. La operación realizada es la siguiente:

PC = PC + 0002 + Dirección de Etiqueta, Si I = 1

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
BMS Etiqueta	Saltar a la etiqueta si el Bit I del CCR es 1	3

Ejemplo:

BMS SALTO1 ; Si en el momento de ejecución su estado es "1", el microcontrolador salta a la etiqueta "SALTO1"

BNE : Saltar si no es igual

Descripción: Realiza un salto a una etiqueta o línea definida por el usuario si al efectuar una comparación entre dos cantidades no son iguales. Esta instrucción prueba el estado del Bit Z presente en el registro CCR y si su estado es "0", realiza el salto a la línea establecida. La operación realizada es la siguiente:

PC = PC +0002 + Dirección , Si Z=0

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
BNE ETIQ	Saltar a la etiqueta si no es igual	3

Ejemplo:

LDA #40T ; A = 40, **Acumulador** = 40
CMP #30T ; Compara A con 30
BNE Salto1 ; Si no son iguales salta a la etiqueta "Salto1", de lo contrario sigue en la siguiente línea,

como para el caso no son iguales se produce el salto.

BPL : Saltar si es positivo

Descripción: Esta instrucción salta a una etiqueta en particular siempre y cuando después de realizar una operación el resultado sea Positivo, reflejandose según el estado del Bit de Negativo (N) presente en el registro (CCR). Si al verificar el estado del Bit está en "0", el microcontrolador salta a la etiqueta deseada por el usuario. La operación realizada es la siguiente:

PC = PC + 0002 + Dirección de Etiqueta, Si N = 0

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
BPL ETIQ	Saltar si el resultado de una operación es Positivo	3

Ejemplo:

LDA #20T **;** A = 20, Acumulador = 20
SUB #10T **; A** = 20 - 10
BPL SALTO1 Como para el caso A (20d) es mayor que la segunda cantidad (10d), el resultado de la diferencia es positivo y el Bit de Negativo (N) toma como valor "0"; por ello el programa salta a la etiqueta "SALTO1", por de lo contrario, no se realiza el salto y se continúa en la línea siguiente.

BRA : Salto incondicional

Descripción: Esta instrucción permite saltar a una etiqueta o línea en especial. La diferencia fundamental con la instrucción **BSR,** es que a través de ésta instrucción se pueden realizar saltos a subrutinas sin posibilidad de retorno. Como la gran mayoría de los lectores han trabajado con microcontroladores PIC, esta instrucción es la equivalente a **GOTO**. La operación realizada es la siguiente:

PC = PC + 0002 + Dirección ; Carga el PC (contador de Programa), con la dirección de inicio de la subrutina o etiqueta solicitada

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
BRA ETIQ	Saltar a la Etiqueta siempre	3

Ejemplo:

En el siguiente ejemplo se explica un programa que enciende un LED y luego lo apaga, para encenderlo se llama a la rutina **"LED_ON"** y para apagarlo se llama la rutina **"LED_OFF".**

```
RUTINA    BRA    LED_ON    ; Llama rutina para encender LED
SALTO1    BRA    LED_OFF   ; Llama rutina para apagar LED
SALTO2    BRA    RUTINA    ; Salta a etiqueta RUTINA
```

```
; Rutina para Encender un LED
LED_ON    BSET   LED,PORTB          ;Encender Led
          BRA    SALTO1             ;Retorno
```

```
; Rutina para Apagar un LED
LED_OFF   BCLR   LED,PORTB          ;Apagar Led
          BRA    SALTO1             ;Retorno
```

BRCLR n: Saltar si el bit n es 0

Descripción: Esta instrucción efectua un salto en particular después de verificar el estado del Bit n del registro de memoria (M). Si al verificar el estado del Bit está en "0", el microcontrolador salta a la etiqueta deseada por el usuario. La operación realizada es la siguiente:

PC = PC + 0003 + Dirección de Etiqueta, Si Mn = 0

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
BRCLR N,OPR,ETIQ	Saltar a la etiqueta si el bit N del registro OPR esta en 0	5

Ejemplo:

MOV #%11110000,$80 ; Dirección 80h = %11110000

BRCLR 0,$80,Salto1 ; Si el Bit 0 del registro correspondiente a la dirección 80h está en "0", se produce un salto a la etiqueta "Salto1" ; y como es el caso, salta!.

BRN : Nunca saltar

Descripción: Esta instrucción nunca permite saltar a una etiqueta o línea en especial. Puede ser considerada como la ejecución de 2 NOP con una duración de 3 ciclos para su ejecución; es utill durante el programa de depuración para negar el efecto de otra instrucción de salto sin perturbar el Byte de desplazamiento. La operación realizada es la siguiente:

PC = PC + 0002

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
BRN ETIQ	Nunca Saltar a la Etiqueta	3

Ejemplo:

LDA #10T **;** A = 10, Acumulador = 10
CMP #20T **;** Compara 10 con 20
BRN SALTO1 **;** Independiente de la comparación realizada, Nunca saltará a la etiqueta "SALTO1", y continuará en la líne asiguiente

BRSET n: Saltar si el bit n es 1

Descripción: Esta instrucción efectua un salto en particular después de verificar el estado del Bit n del registro de memoria (M). Si al verificar el estado del Bit está en "1", el microcontrolador salta a la etiqueta deseada por el usuario. La operación realizada es la siguiente:

PC = PC + 0003 + Dirección de Etiqueta, Si Mn = 1

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
BRSET	Saltar a la etiqueta si el bit N del registro OPR esta	5

N,OPR,ETIQ	en 1	

Ejemplo:

```
MOV    #%11110000,$80 ; Dirección 80h = %11110000
BRCLR 7,$80,Salto1  ; Si el Bit 7 del registro correspondiente a
                      la dirección 80h está en "1", se produce
                      un salto a la etiqueta "Salto1"  ; y como
                      es el caso, salta!.
```

BSR : Saltar a subrutina

Descripción: Esta instrucción permite saltar a una Subrutina. Como la gran mayoría de los lectores han trabajado con microcontroladores PIC, esta instrucción es la equivalente a CALL. La operación realizada es la siguiente:

```
PC = PC + 0002        ; Avanza el PC para ir a la Dirección
SP = SP + 0001h       ; Guarda parte baja del PCL en la Pila
SP = SP + 0001h       ; Guarda parte alta del PCL en la Pila
PC = PC + Dirección   ; Carga el PC (contador de Programa),
                        con la dirección de inicio de la subrutina
                        solicitada
```

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
BSR ETIQ	Saltar a Subrutina	4

Ejemplo:

En el siguiente ejemplo se explica un programa que enciende un LED y luego lo apaga, para encenderlo se llama a la rutina **"LED_ON"** y para apagarlo se llama la rutina **"LED_OFF".**

```
RUTINA    BSR    LED_ON    ; Llama rutina para encender LED
          BSR    LED_OFF   ; Llama rutina para apagar LED
          BRA    RUTINA    ; Salta a etiqueta RUTINA
```

```
; Rutina para Encender un LED
LED_ON    BSET   LED,PORTB          ;Encender Led
                RTS                         ; Retorno de Subrutina
```

```
; Rutina para Apagar un LED
LED_OFF   BCLR   LED,PORTB          ;Apagar Led
          RTS                       ;Retorno de Subrutina
```

CBEQ : Compara y salta si es igual

Descripción: Realiza una comparación entre dos cantidades; La primera cantidad corresponde al valor que se encuentra almacenado en el Acumulador (**A**) o el registro índice (**X**), la segunda cantidad puede ser una constante o un dato almacenado previamente en un registro; si al efectuar la comparación las dos cantidades resultan iguales se produce un salto a una línea o etiqueta en especial definida por el usuario . La operación realizada es la siguiente:

Comparación = A – (M)
ó
Comparación = X – (M)

PC = PC +0003 + Dirección
Si el resultado de la comparación es 0 SALTA

M = Constante o Valor almacenado en un registro o posición de Memoria

Sintaxis:

INSTRUCCIÓN	Aclaración	No. Ciclos
CBEQ OPR,ETIQ		5
CBEQA #OPR,ETIQ	Comparar el valor de A con el valor #OPR o el dato almacenado en OPR y saltar si son iguales a la	4
CBEQX #OPR,ETIQ	etiqueta	4
CBEQ OPR,X+,ETIQ	X+ = Registro H:X	5
CBEQ X+,ETIQ		4
CBEQ OPR,SP,ETIQ		5

Ejemplos:

LDA #40T ; A = 40, **Acumulador** = 40
CBEQA #30T,Salto1 ; Compara 40 con 30 si son iguales salta a la etiqueta "Salto1", de lo contrario sigue en la siguiente línea

```
LDA     #40T        ; A = 40, Acumulador = 40
MOV     #40T,$80    ;Dirección 80h = 40
CBEQ    $80,Salto1  ; Compara el valor presente en el
                    Acumulador con el valor presente en la
                    dirección 80h si son iguales salta a la
                    etiqueta "Salto1", de lo contrario sigue
                    en la siguiente línea, como para el caso
                    son iguales SALTA!
```

DBNZ : Decrementa y salta si no es cero

Descripción: Esta función como su nombre lo indica, decrementa en 1 el valor presente en una variable, ya sea el Acumulador (A), el registro índice (X) o una posición de memoria (M), con la singularidad de que además de efectuar el decremento de la variable realiza una comparación del resultado obtenido con CERO; Si al efectuar la comparación del resultado con Cero no son iguales efectua un salto a una etiqueta o línea en especial definida por el usuario. La operación realizada es la siguiente:

A = A – 1
ó
X = X – 1
ó
(M) = (M) – 1

Compara resultado con 0 y si no es igual SALTA

M = Valor almacenado en un registro o posición de Memoria

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
DBNZ OPR,ETIQ		5
DBNZA ETIQ		3
DBNZX ETIQ	Decrementar y saltar si no es Cero	3
DBNZ OPR,X,ETIQ		5
DBNZ X,ETIQ		4
DBNZ OPR,SP,ETIQ		5

Ejemplos:

LDA #40T ; A = 40,
DBNZA SALTO1 ; A = A – 1 = 39, compara A con 0 y como no son iguales salta a la etiqueta "SALTO1"

MOV #40T,$80 ; Dirección 80h = 40,
DBNZ $80, SALTO1 ; Dirección 80h=Dirección 80h – 1 = 39, compara el valor almacenado en la Dirección 80h con 0 y como no son iguales salta a la etiqueta "SALTO1"

JMP : Saltar

Descripción: Esta instrucción permite saltar a una etiqueta o línea en especial. La diferencia fundamental con la instrucción **BRA,** es que a través de ésta instrucción se pueden realizar saltos más largos a subrutinas sin posibilidad de retorno. Como la gran mayoría de los lectores han trabajado con microcontroladores PIC, esta instrucción es la equivalente a **GOTO**. La operación realizada es la siguiente:

PC = PC + Dirección ; Carga el PC (contador de Programa), con la dirección de inicio de la subrutina o etiqueta solicitada

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
JMP OPR	Saltar a la dirección OPR	2
JMP OPR,X		3
JMP ,X		2

Ejemplo:

En el siguiente ejemplo se explica un programa que enciende un LED y luego lo apaga, para encenderlo se llama a la rutina **"LED_ON"** y para apagarlo se llama la rutina **"LED_OFF".**

```
RUTINA   JMP   LED_ON    ; Llama rutina para encender LED
SALTO1   JMP   LED_OFF   ; Llama rutina para apagar LED
SALTO2   BRA   RUTINA    ; Salta a etiqueta RUTINA
```

```
; Rutina para Encender un LED
```

```
LED_ON      BSET    LED,PORTB           ;Encender Led
                    JMP     SALTO1              ;Retorno
```

```
; Rutina para Apagar un LED
LED_OFF     BCLR    LED,PORTB           ;Apagar Led
            JMP     SALTO2              ;Retorno
```

INSTRUCCIONES ARITMÉTICAS Y LÓGICAS

ADC : Suma con Acarreo

Descripción: Efectúa la suma de dos cantidades incluyendo en la suma el valor presente en el Bit de Carry; la primera cantidad corresponde al Acumulador (Registro A), la segunda cantidad puede ser una constante o un dato almacenado previamente en un registro y la respuesta queda almacenada en el registro Acumulador. La operación realizada es la siguiente:

A=A+(M)+C

M = Constante o Valor almacenado en una posición de Memoria

C = Carry o Acarreo

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
ADC #OPR	#OPR = Constante.	2
ADC OPR	OPR = Dirección de memoria o registro	3
ADC OPR,X		3
ADC ,X	X = Registro Índice	2
ADC OPR,SP	SP = Stack Pointer o Apuntador de Pila	4

Ejemplos:

LDA #10T **;** A = 10 , Carga el valor de 10 en el Acumulador
ADC #20T **;** A=A+20+C , Suma A con 20 y con el Carry
Si C =1 entonces A = 10 + 20 + 1 = 31
Si C =0 entonces A = 10 + 20 + 0 = 30

MOV #10T,$80 ; Almacena el valor 10 en la dirección 80h
LDA #20T **;** A = 20
ADC $80 **;** A=A+(Valor almacenado en dirección 80h)+C
Si C =1 entonces A = 20 + 10 + 1 = 31

Si C =0 entonces A = 20 + 10 + 0 = 30

MOV #10T,$81 ; Almacena el valor 10 en la dirección 81h
LDX #1 ; X=1
LDA #20T ; A = 20
ADC $80,X ; A=A+(Valor almacenado en dirección 80h+X)+C
Si C =1 entonces A = 20 + 10 + 1 = 31
Si C =0 entonces A = 20 + 10 + 0 = 30

ADD : Suma sin Acarreo

Descripción: Efectúa la suma de dos cantidades sin incluir en la suma el valor presente en el Bit de Carry; la primera cantidad corresponde al Acumulador (Registro A), la segunda cantidad puede ser una constante o un dato almacenado previamente en un registro y la respuesta queda almacenada en el registro Acumulador. La operación realizada es la siguiente:

A=A+(M)

M = Constante o Valor almacenado en un registro o posición de Memoria

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
ADD #OPR	#OPR = Constante, por ejemplo #10T ; 10 Decimal	2
ADD OPR	OPR = Dirección de memoria o registro	3
ADD OPR,X		3
ADD ,X	X = Registro Índice	2
ADD OPR,SP	SP = Stack Pointer o Apuntador de Pila	4

Ejemplos:

LDA #10T ; A = 10 , Carga el valor de 10 en el Acumulador
ADD #20T ; A=A+20 , Suma A con 20

Entonces A = 10 + 20 = 30

MOV #$0A,$80 ; Almacena el valor 0Ah (10d) en la dirección 80h
LDA #20T ; A = 20
ADD $80 ; A=A+(Valor almacenado en dirección 80h)

Entonces A = 20 + 10 = 30

MOV #10T,$81 ; Almacena el valor 10 en la dirección 81h
LDX #1 ; X=1
LDA #20T ; A = 20
ADD $80,X ; A=A+(Valor almacenado en dirección 80h+X)

Entonces A = 20 + 10 = 30

AND: Función lógica AND

Descripción: Efectúa la función lógica AND entre dos cantidades; la primera cantidad corresponde al Acumulador (Registro A), la segunda cantidad puede ser una constante o un dato almacenado previamente en un registro y la respuesta queda almacenada en el registro Acumulador. La operación realizada es la siguiente:

A = A & (M)

M = Constante o Valor almacenado en un registro o posición de Memoria

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
AND #OPR	#OPR = Constante	2
AND OPR	OPR = Dirección de memoria o registro	3
AND OPR,X		3
AND ,X	X = Registro Índice	2
AND OPR,SP	SP = Stack Pointer o Apuntador de Pila	4

Ejemplos:

LDA #%10101010 ; A = 10101010 (170 d), Acumulador=170
AND #%11110000 ; A = A & (11110000)

Entonces A = 10100000 (160 d)
MOV #$F0,$80 ; Almacena F0h (240 d) en la dirección 80h
LDA #$0F ; A = 0Fh (15 d)
AND $80 ; A = A & (Valor almacenado en dirección 80h)

Entonces A = 00h (0 d)

MOV #$F0,$82 ; Almacena F0 en la dirección 82h

LDX #2 **;** X=2
LDA #$0F **;** A = 0Fh (15 d)
AND $80,X **;** A = A & (Valor almacenado en dirección 82h)

Entonces A = 00h (0 d)

AIS : Suma el valor inmediato al puntero de pila con signo

Descripción: Efectúa la suma entre dos cantidades; la primera cantidad corresponde al valor inmediato al puntero de pila (SP Stack Pointer), la segunda cantidad debe ser una constante y la respuesta queda almacenada en el puntero de pila como valor inmediato. La operación realizada es la siguiente:

SP = SP + (Constante)

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
AIS #OPR	SP=SP+DATO	2

Ejemplo:

LDA #10T ; A = 10, Acumulador = 10
PSHA **;** Inserta A en el Stack o Pila, SP = 10
AIS #20T ; SP = SP + 20 = 30
PULA **;** Saca el último valor del Stack y lo guarda en A

Entonces A = 30

AIX : Suma el valor inmediato al registro de índice con signo

Descripción: Efectúa la suma entre dos cantidades; la primera cantidad corresponde al valor inmediato al registro índice H:X, considerando que al efectuar ésta suma se tiene en cuenta el signo de las cantidades a sumar; la segunda cantidad debe ser una constante y la respuesta queda almacenada en el registro índice H:X como valor inmediato. La operación realizada es la siguiente:

H:X = H:X + (Constante)

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
AIX #OPR	H:X = H:X + DATO	2

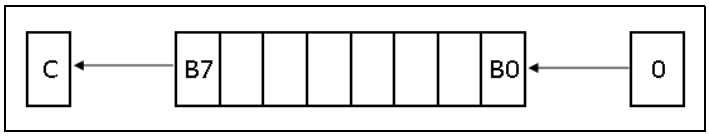
Ejemplo:

LDHX #10T ; H:X = 10, **Registro índice H:X** = 10
AIX #20T ; H:X = H:X + 20 = 30

Entonces H:X = 30

ASL : Desplazamiento Aritmético a la Izquierda

Descripción: Efectúa un desplazamiento aritmético a la izquierda. Este tipo de desplazamiento consiste en rotar a la izquierda todo el registro una posición, quedando el Bit que se encontraba en la posición de más peso (Bit 7) en el Bit de Carry y el Bit menos peso (Bit 0) será llenado con un 0, tal como se ilustra a continuación.



Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
ASL OPR	OPR = Dirección de Memoria o Registro	4
ASLA	A = Acumulador	1
ASLX	X = Registro Índice	1
ASL OPR,X		4
ASL,X		3
ASL OPR,SP	SP = Stack Pointer	5

Ejemplo:

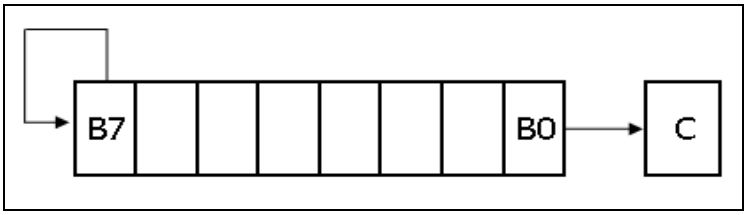
LDA #%10101010 ; A = 10101010 (170 d), Acumulador=170
ASLA ; A = 01010100 ; Carry = 1

Entonces A = 01010100 (84 d)

Obsérvese que en el ejemplo, el valor inicial cargado en el Acumulador fue el 10101010 binario y al realizar la rotación a la izquierda el "1" que se encontraba en el Bit 7, paso al Bit de Carry, todos los bits se trasladaron una posición a la izquierda y el Bit 0 o Bit de menos peso tomo el valor 0 por corresponder a un desplazamiento aritmético.

ASR : Desplazamiento aritmético a la derecha

Descripción: Efectúa un desplazamiento aritmético a la derecha. Este tipo de desplazamiento consiste en rotar a la derecha todo el registro una posición, quedando el Bit que se encontraba en la posición de menos peso (Bit 0) en el Bit de Carry y el Bit más peso (Bit 7) será llenado de manera consecutiva con el mismo valor que tenía inicialmente, tal como se ilustra a continuación.



Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
ASR OPR	OPR = Dirección de Memoria o Registro	4
ASRA	A = Acumulador	1
ASRX	X = Registro Índice	1
ASR OPR,X		4
ASR OPR,SP	SP = Stack Pointer	5

Ejemplo:

LDA #%10101010 ; A = 10101010 (170 d), Acumulador=170
ASRA ; A = 11010101 **; Carry = 0**

Entonces A = 11010101 (213 d)

Obsérvese que en el ejemplo, el valor inicial cargado en el Acumulador fue el 10101010 binario y al realizar la rotación a la redecha el "0" que se encontraba en el Bit 0, paso al Bit de Carry, todos los bits se trasladaron una posición a la derecha y el Bit 7 o Bit de más peso tomo nuevamente el valor que tenía antes de la rotación.

CMP : Compara el Acumulador con la memoria

Descripción: Realiza una comparación entre dos cantidades; La primera cantidad corresponde al valor que se encuentra almacenado en el Acumulador (A), la segunda cantidad puede ser una constante o un dato almacenado previamente en un registro y la respuesta queda disposición del usuario, según lo que desee saber al respecto de ésta operación, por ejemplo, si una cantidad es mayor, menor o igual que la otra, entre otras preguntas que se pueden formular. La operación realizada es la siguiente:

Comparación = A – (M)

M = Constante o Valor almacenado en un registro o posición de Memoria

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
CMP #OPR	#OPR = Constante	2
CMP OPR	OPR = Dirección de Memoria o Registro	3
CMP OPR,X	X = Registro Índice	3
CMP ,X		2
CMP OPR,SP	SP = Stack Pointer	4

Ejemplos:

```
LDA  #10T          ; A = 10,  Acumulador = 10
CMP #20T       ; Compara 10 con 20
```

Entonces Comparación = 10 – 20, ahora se pregunta lo que desea conocer el programador con respecto a la comparación de éstas dos cantidades, por ejemplo, si se quiere que el programa salte a una etiqueta llamada "Menor" cuando la cantidad que está en el Acumulador (A) sea menor que la cantidad con la que se realizo la comparación. Debajo de la línea de comparación se coloca la condición de salto según el resultado tal como se ilustra a continuación:

```
LDA  #10T          ; A = 10,  Acumulador = 10
```

CMP #20T **;** Compara 10 con 20
BLO Menor Como para el caso A (10d) es menor que la cantidad con la que se comparo (20d), el programa salta a la etiqueta "Menor", de lo contrario, no se realiza el salto y se continúa en la línea siguiente.

MOV #$0A,$80 **;** Almacena 0Ah (10 d) en la dirección 80h
LDA #$0F **;** A = 0Fh (15 d)
CMP $80 Compara el contenido de la dirección 80h (0Ah) con el valor almacenado en el Acumulador (0Fh)
BHI Mayor Como para el caso el Acumulador A (15d) es mayor que la cantidad con la que se comparo (10d), el programa salta a la etiqueta "Mayor", de lo contrario, no se realiza el salto y se continúa en la línea siguiente.

COM : Complemento a uno

Descripción: Reemplaza el contenido del Acumulador (A), el registro índice (X), o el registro de memoria (M) con su complemento a uno. Este complemento consiste en reemplazar cada Bit perteneciente al registro correspondiente por su respectivo complemento, es decir, que todos los bits que se encontraban en el registro con valor "1" tendrán como nuevo valor "0" y que todos los bits que se encontraban en el registro con valor "0" tendrán como nuevo valor "1". La operación correspondiente a tal complemento es la siguiente:

A = FFh – A ; A = 255 – A
ó
(M) = FFh - (M) ; (M) = 255 - (M)
ó
X = FFh – X ; X = 255 - X

M = Valor almacenado en un registro o posición de Memoria

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
COM OPR	OPR = Dirección de Memoria o Registro	4
COMA	A = Acumulador	1
COMX	X = Registro Índice	1
COM OPR,X		4
COM ,X		3
COM OPR,SP	SP = Stack Pointer	5

Ejemplos:

LDA #%11110000 **;** A = 11110000 (240 d)
COMA **; A = 255 – 240 = 15 = 0Fh**

Entonces A = 0Fh = 00001111

Obsérvese en el ejemplo que el valor inicial del Acumulador era 11110000 ó 240d y al efectuar la operación de complemento a uno, el resultado fue 00001111 ó 15d lo cual ilustra que todos los bits que tenían valor "1" tomaron como nuevo valor "0" y viceversa.

MOV #170,$80 **;** Almacena el valor 170d en la dirección 80h
COM $80 Realiza el complemento a uno del valor existente en la dirección 80h

Entonces, el nuevo valor almacenado en la dirección 80h de la memoria será:

Nuevo Valor = 255 – 170 = 85

Si se analiza un poco, se puede observar que 170d es 10101010 y 85d es 01010101, lo cual concuerda con lo mencionado en el ejemplo anterior.

CPHX : Compara el valor del registro H:X con la memoria

Descripción: Realiza una comparación entre dos cantidades; La primera cantidad corresponde al valor que se encuentra almacenado en el registro índice (H:X) considerando que el registro H:X es un registro de 16 Bits, la segunda cantidad puede ser una constante o un dato almacenado previamente en un registro y la respuesta queda disposición del usuario, según lo que desee saber al respecto de ésta operación, por ejemplo, si una cantidad es mayor, menor o igual que la otra, entre otras preguntas que se pueden formular. La operación realizada es la siguiente:

Comparación = H:X – (M)

M = Constante o Valor almacenado en un registro o posición de Memoria

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
CPHX #OPR	#OPR = Constante	3
CPHX OPR	OPR = Dirección de Memoria o Registro	4

Ejemplos:

LDHX #40T **; H:X** = 40, **Registro Índice** = 40
CPHX #30T **; Compara 40 con 30**

Entonces Comparación = 40 – 30, ahora se pregunta lo que desea conocer el programador con respecto a la comparación de éstas dos cantidades, por ejemplo, si se quiere que el programa salte a una etiqueta llamada "Mayor" cuando la cantidad que está en el Registro Índice (H:X) sea mayor que la cantidad con la que se realizo la comparación. Debajo de la línea de comparación se coloca la condición de salto según el resultado tal como se ilustra a continuación:

LDHX #40T **; H:X** = 40, **Registro Índice** = 40
CPHX #30T **; Compara 40 con 30**
BHI Mayor Como para el caso H:X (40d) es menor que la cantidad con la que se comparo (30d), el programa salta a la etiqueta "Menor", de lo contrario, no se realiza el salto y se continúa en la línea siguiente.

MOV #$0F,$80 **;** Almacena 0Fh (15 d) en la dirección 80h
LDHX #$0A **; H:X** = 0Ah (10 d)
CPHX $80 Compara el contenido de la dirección 80h (0Fh) con el valor almacenado en el Registro Índice (0Ah)
BLO Menor Como para el caso H:X (10d) es menor que la cantidad con la que se comparo (15d), el programa salta a la etiqueta "Menor", de lo contrario, no se realiza el salto y se continúa en la línea siguiente.

CPX : Compara el valor del registro X con la memoria

Descripción: Realiza una comparación entre dos cantidades; La primera cantidad corresponde al valor que se encuentra almacenado

en el registro índice (X) considerando que el registro X es un registro de 8 Bits a diferencia de la instrucción anterior que era de 16 Bits, la segunda cantidad puede ser una constante o un dato almacenado previamente en un registro y la respuesta queda disposición del usuario, según lo que desee saber al respecto de ésta operación, por ejemplo, si una cantidad es mayor, menor o igual que la otra, entre otras preguntas que se pueden formular. La operación realizada es la siguiente:

Comparación = X – (M)

M = Constante o Valor almacenado en un registro o posición de Memoria

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
CPX #OPR	#OPR = Constante	2
CPX OPR	OPR = Dirección de Memoria o Registro	3
CPX ,X	X = Registro Índice	3
CPX OPR,X		2
CPX OPR,SP	SP = Stack Pointer	4

Ejemplo:

LDX #40T ; X = 40, **Registro Índice** = 40
CPX #30T ; Compara 40 con 30

Se debe tener en cuenta que para esta instrucción el procedimiento de comparación es semejante a los mencionados anteriormente con las instrucciones **CMP y CPHX.**

DAA : Ajuste decimal del Acumulador

Descripción: Cuando se efectúa una suma de dos cantidades utilizando instrucciones como ADD y ADC, muchas veces es necesario

según el proceso que se desee realizar un ajuste a Decimal. Las cantidades con las que se realizo la operación están en formato BCD (Decimal Codificado en Binario).

Esta instrucción ajusta tanto el contenido del Acumulador (A) como el estado del Bit de Acarreo medio (CCR) después de realizada una suma.

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
DAA	Ajustar a decimal el registro A	2

Para entender mejor este tipo de ajuste, es necesario tener en cuenta la siguiente tabla, la cual es aplicada según los resultados obtenidos en el Acumulador después de realizar una suma en formato BCD**.**

1	2	3	4	5	6
Valor inicial del bit C	Valor de A[7:4]	Valor inicial del bit H	Valor de A[3:0]	Factor de corrección	Valor corregido del bit C
0	0-9	0	0-9	00	0
0	0-8	0	A-F	06	0
0	0-9	1	0-3	06	0
0	A-F	0	0-9	60	1
0	9-F	0	A-F	66	1
0	A-F	1	0-3	66	1
1	0-2	0	0-9	60	1
1	0-2	0	A-F	66	1
1	0-3	1	0-3	66	1

Observando la tabla anterior se puede establecer que:

- La columna 1 corresponde al valor inicial del Bit C o Acarreo, este acarreo proviene de operaciones anteriormente realizadas; puede tener como valor tanto “0” como “1”.
- La columna 3 Corresponde al Bit de Acarreo medio, el cual puede tomar valores de “0” y “1” según el resultado de una suma, teniendo en cuenta que este acarreo aparece entre los Bits 3 y 4 del registro Acumulador.
- La columna 6 corresponde al Acarreo final, el cual puede también tomar valores de “0” y “1”, este acarreo tomara valor “1” cuando en el momento de realizar el ajuste la suma sea mayor a 99

- Las columnas 2 y 4 corresponden a los cuatro Bits de mas peso y a los cuatro bits de menos peso del registro Acumulador respectivamente.
- La columna 5 obedece a unos valores que se suman al resultado obtenido según los valores presentes en los cuatro bits de mas y menos peso para hacer efectivo el ajuste a BCD.

Ejemplos:

```
LDA #$28      ; A = 28h  o A=28 (BCD)
ADD #$22      ; A=A+22h , Suma A con 22h o 22 (BCD)
```

Como	**28h**	**Pero**	**28**
	+ 22h		**+ 22**
	4Ah		**50**

entonces aplicamos la instrucción DAA después de la instrucción de suma asi:

```
LDA #$28      ; A = 28h  o A=28 (BCD)
ADD #$22      ; A=A+22h , Suma A con 22h o 22 (BCD)
DAA
```

Al efectuar la suma utilizando la instrucción ADD, en el Acumulador quedo almacenado el valor 4Ah (A=4Ah), pero al momento de ejecutar la instrucción DAA observamos que:

4Ah Se descompone de la siguiente manera:

4 : Bits de mas peso
A : Bits de menos peso

Según la tabla anterior, el valor de corrección seria el correspondiente al presente en la segunda fila. Esto obedece a que los cuatro bits de mas peso tienen un valor entre 0-8 y los cuatro bits de menos peso tienen un valor entre A-F. Aplicando el factor de corrección a BCD (06) se tiene que:

Sumando	**4Ah**	**lo que corresponde a**	**28**	
	+ 06h		**+ 22**	
	50h		**50**	**(BCD)**

Hay que tener en cuenta que este factor de corrección es aplicado automáticamente por la instrucción DAA según el resultado obtenido en el Acumulador al realizar la suma de dos cantidades. Lo cual indica

que al ejecutar la instrucción DAA en el Acumulador queda el valor corregido y ajustado a formato BCD tal como se ilustra a continuación

```
LDA #$28      ; A = 28h  o A = 28 (BCD)
ADC #$22      ; A = A + 22h ; A = 4Ah
DAA           ; A = 50h es decir A = 50 (BCD)
```

DEC : Decremento

Descripción: Esta función como su nombre lo indica, decrementa en 1 el valor presente en una variable, ya sea el Acumulador (A), el registro índice (X), o una posición de memoria (M). La operación realizada es la siguiente:

A = A – 1
ó
X = X – 1
ó
(M) = (M) – 1

M = Valor almacenado en un registro o posición de Memoria

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
DEC OPR	OPR = Dirección de Memoria o Registro	4
DECA	A = Acumulador	1
DECX	X = Registro Índice	1
DEC OPR,X		4
DEC ,X		3
DEC OPR,SP	SP = Stack Pointer	5

Ejemplos:

```
LDA   #40T      ; A = 40,
DECA            ; A = A – 1 = 39

LDX   #20T      ; X = 20,
DECX            ; X = X – 1 = 19
```

DIV : División

Descripción: Esta función realiza la división sin signo entre dos cantidades; la primera cantidad corresponde a un número de 16 bits, de los cuales los 8 bits de más peso se almacenan en el registro **H** o parte alta del registro índice **(H:X)** y los 8 bits de menos peso se registran en el Acumulador (Registro A), quedando el dividendo en el registro **(H:A)**, la segunda cantidad corresponde al divisor y es almacenado en el registro índice **(X)**, lo cual indica que el divisor tiene una longitud máxima de 8 bits. La respuesta a la operación de división queda almacenada en el registro Acumulador **(A)**, es decir, el cociente queda almacenado allí y finalmente el residuo queda almacenado en el registro **(H)** o parte alta del registro índice **(H:X)**. La operación realizada es la siguiente:

A = (H:A) / (X) ; (H) = Residuo

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
DIV	Dividir A=(H:A)/X (A = Cociente , H = Residuo)	7

Ejemplo:

Se desea realizar la división entre dos cantidades 10000 (Diez Mil) y 200 (Doscientos).

Al convertir **10000d** en hexadecimal **(2710h)** se puede observar que los 8 bits de más peso corresponden al valor **(27h)** y los 8 de menos peso corresponden al valor **(10h).** Este valor 10h se encuentra en el registro **(X)** y debe ser almacenado en el Acumulador **(A)**; finalmente en el registro **(X)** se debe almacenar el valor **200d** correspondiente al divisor. El proceso entonces es el siguiente:

```
LDHX #10000T      ; H:X = 10000, H = 27h y X = 10h
TXA               ; A = X , A = 10h  -> H:A = 2710h
LDX  #200T        ; X = 200
DIV               ; Dividir A = (H:A)/X
                    (A = Cociente, H = Residuo)
```

El resultado de la operación es: A = 50 y H = 0

EOR : Función lógica OR Exclusiva

Descripción: Efectúa la función lógica OR Exclusiva entre dos cantidades; la primera cantidad corresponde al Acumulador (Registro A), la segunda cantidad puede ser una constante o un dato almacenado previamente en un registro y la respuesta queda almacenada en el registro Acumulador. La operación realizada es la siguiente:

$$\mathbf{A = A \oplus (M)}$$

M = Constante o Valor almacenado en un registro o posición de Memoria

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
EOR #OPR	#OPR = Constante	2
EOR OPR	OPR = Dirección de memoria o registro	3
EOR OPR,X		3
EOR ,X	X = Registro Índice	2
EOR OPR,SP	SP = Stack Pointer o Apuntador de Pila	4

Nota:

Uno de los usos de esta función es a la hora de comparar dos cantidades, si al efectuar la función OR Exclusiva entre las dos cantidades el resultado es Cero, se puede decir que las dos cantidades son **IGUALES.**

Ejemplos:

LDA #%10101010 **;** A = 10101010 (170 d), Acumulador=170
EOR #%10101010 **;** A = A ⊕ (11110000) **= 00000000**

Entonces A = 00000000 (0 d)

MOV #$F0,$80 **;** Almacena F0h (240 d) en la dirección 80h
LDA #$0F **;** A = 0Fh (15 d)
EOR $80 **;** A = A ⊕ (Valor almacenado en dirección 80h)

Entonces A = FFh (255 d)

MOV #$F0,$82 ; Almacena F0 en la dirección 82h
LDX #2 ; X=2
LDA #$0F ; A = 0Fh (15 d)
AND $80,X ; A = A ⊕ (Valor almacenado en dirección 82h)

Entonces A = FFh (255 d)

INC : Incrementar

Descripción: Esta función como su nombre lo indica, incrementa en 1 el valor presente en una variable, ya sea el Acumulador (A), el registro índice (X), o una posición de memoria (M). La operación realizada es la siguiente:

A = A + 1
ó
X = X + 1
ó
(M) = (M) + 1

M = Valor almacenado en un registro o posición de Memoria

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
INC OPR	OPR = Dirección de Memoria o Registro	4
INCA	A = Acumulador	1
INCX	X = Registro Índice	1
INC OPR,X		4
INC ,X		3
INC OPR,SP	SP = Stack Pointer	5

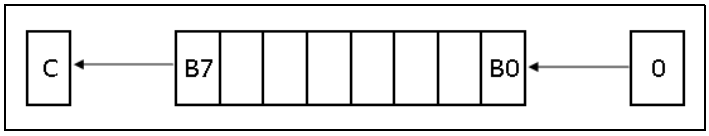
Ejemplos:

LDA #40T ; **A** = 40,
INCA ; **A = A + 1 = 41**

LDX #20T ; **X** = 20,
INCX ; **X = X + 1 = 21**

LSL : Desplazamiento lógico a la izquierda

Descripción: Efectúa un desplazamiento lógico a la izquierda, este tipo de desplazamiento es exactamente igual al desplazamiento aritmético a la izquierda. Consiste en rotar a la izquierda todo el registro una posición, quedando el Bit que se encontraba en la posición de más peso (Bit 7) en el Bit de Carry y el Bit menos peso (Bit 0) será llenado con un 0, tal como se ilustra a continuación.



Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
LSL OPR	OPR = Dirección de Memoria o Registro	4
LSLA	A = Acumulador	1
LSLX	X = Registro Índice	1
LSL OPR,X		4
LSL ,X		3
LSL OPR,SP	SP = Stack Pointer	5

Ejemplo:

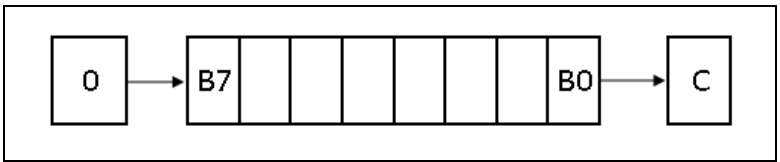
LDA #%10101010 ; A = 10101010 (170 d), Acumulador=170
LSLA ; A = **01010100 ; Carry = 1**

Entonces A = 01010100 (84 d)

Obsérvese que en el ejemplo, el valor inicial cargado en el Acumulador fue el 10101010 binario y al realizar la rotación a la izquierda el "1" que se encontraba en el Bit 7, paso al Bit de Carry, todos los bits se trasladaron una posición a la izquierda y el Bit 0 o Bit de menos peso tomo el valor 0 por corresponder a un desplazamiento lógico.

LSR : Desplazamiento lógico a la derecha

Descripción: Efectúa un desplazamiento lógico a la derecha, este tipo de desplazamiento si es diferente al desplazamiento aritmético a la derecha. Consiste en rotar a la derecha todo el registro una



posición, quedando el Bit que se encontraba en la posición de menos peso (Bit 0) en el Bit de Carry y el Bit más peso (Bit 7) será llenado con un 0, tal como se ilustra a continuación.

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
LSR OPR	OPR = Dirección de Memoria o Registro	4
LSRA	A = Acumulador	1
LSRX	X = Registro Índice	1
LSR OPR,X		4
LSR ,X		3
LSR OPR,SP	SP = Stack Pointer	5

Ejemplo:

LDA #%10101010 ; A = 10101010 (170 d), Acumulador=170
LSRA ; A = 01010101 ; Carry = 0
Entonces A = 01010101 (85 d)

Obsérvese que en el ejemplo, el valor inicial cargado en el Acumulador fue el 10101010 binario y al realizar la rotación a la derecha el "0" que se encontraba en el Bit 0, paso al Bit de Carry, todos los bits se trasladaron una posición a la derecha y el Bit 7 o Bit de más peso tomo el valor 0 por corresponder a un desplazamiento lógico.

MUL : Multiplicación sin signo

Descripción: Esta función realiza la multiplicación sin signo entre dos cantidades de 8 bits cada una; la primera cantidad (Multiplicando) corresponde al registro Acumulador **(A)**, la segunda cantidad (Multiplicador) corresponde al registro índice **(X)**. La respuesta a la operación de multiplicación corresponde a una cantidad de 16 bits, donde los 8 bits de más peso quedan almacenados en el

registro índice **(X)** y los 8 bits de menos peso quedan almacenados en el registro Acumulador **(A)**, es decir, el producto queda almacenado en el registro **(X:A)**. La operación realizada es la siguiente:

X:A = A * X

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
MUL	Multiplicación sin Signo (X:A = A * X)	5

Ejemplo:

Se desea realizar la multiplicación entre dos cantidades 50 (Cincuenta) y 200 (Doscientos).

```
LDA   #50          ; A = 50
LDX   #200T        ; X = 200
MUL                ; Multiplicar X:A = A * X = 10000
```

El resultado de la operación es: X:A = 10000 (Diez Mil)

Al convertir **10000d** en hexadecimal **(2710h)** se puede observar que los 8 bits de más peso corresponden al valor **(27h)** y los 8 de menos peso corresponden al valor **(10h).** El valor 27h se queda almacenado en el registro **(X)** y el valor 10h queda almacenado en el registro **(A).**

NEG : Complemento a dos

Descripción: Esta función como su nombre lo indica, realiza el complemento a 2 o negación de una cantidad, ya sea el Acumulador (A), el registro índice (X), o una posición de memoria (M). La operación realizada es la siguiente:

A = -A
ó

X = -X
ó
(M) = -(M)

M = Valor almacenado en un registro o posición de Memoria

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
NEG OPR	OPR = Dirección de Memoria o Registro	4
NEGA	A = Acumulador	1
NEGX	X = Registro Índice	1
NEG OPR,X		4
NEG ,X		3
NEG OPR,SP	SP = Stack Pointer	5

Ejemplos:

LDA #40T **; A** = 40,
NEGA **; A = -40**

LDX #20T **; X** = 20,
NEGX **; X = -20**

NOP : Sin operación

Descripción: Esta función de un solo Byte provoca que el contador de programa sea incrementado en 1. Ningún otro registro es afectado; es decir, es una instrucción que no realiza ninguna función en particular pero genera un retardo correspondiente a 1 Ciclo. Esta instrucción es muy utilizada para escribir rutinas de retardos basados en instrucciones.

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
NOP	No Operación	1

Ejemplo:

NOP ; Provoca un retardo de 1 Ciclo de Máquina

ORA : Función lógica OR

Descripción: Efectúa la función lógica OR entre dos cantidades; la primera cantidad corresponde al Acumulador (Registro A), la segunda cantidad puede ser una constante o un dato almacenado previamente en un registro y la respuesta queda almacenada en el registro Acumulador. La operación realizada es la siguiente:

$$A = A \mid (M)$$

M = Constante o Valor almacenado en un registro o posición de Memoria

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
ORA #OPR	#OPR = Constante	2
ORA OPR	OPR = Dirección de memoria o registro	3
ORA OPR,X		3
ORA ,X	X = Registro Índice	2
ORA OPR,SP	SP = Stack Pointer o Apuntador de Pila	4

Ejemplos:

LDA #%10101010 ; A = 10101010 (170 d), Acumulador=170
ORA #%10101010 ; A = A | (10101010)

Entonces A = 10101010 (170 d)

MOV #$F0,$80 ; Almacena F0h (240 d) en la dirección 80h
LDA #$0F ; A = 0Fh (15 d)
EOR $80 ; A = A | (Valor almacenado en dirección 80h)

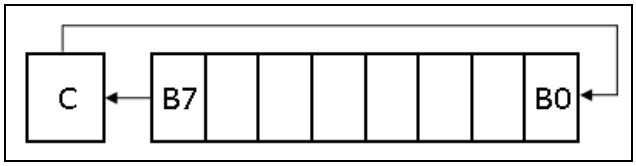
Entonces A = FFh (255 d)

MOV #$F0,$82 ; Almacena F0 (240 d) en la dirección 82h
LDX #2 ; X=2
LDA #$80 ; A = 80h (128 d)
AND $80,X ; A = A | (Valor almacenado en dirección 82h)

Entonces A = F0h (240 d)

ROL : Rotación a la izquierda por acarreo

Descripción: Efectúa la rotación en una posición de todo un registro a la izquierda con la diferencia de que la posición que queda en blanco es ocupada por el valor presente en el Bit de Acarreo; es decir, que utiliza el bit de Acarreo como puente para realizar la rotación de los bits pertenecientes a un registro. El Bit que se encontraba en la posición de más peso (Bit 7) pasa al Bit de Carry y el Bit menos peso (Bit 0) será ocupado con el valor que se encontraba en el Bit de Carry, tal como se ilustra a continuación.



Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
ROL OPR	OPR = Dirección de Memoria o Registro	4
ROLA	A = Acumulador	1
ROLX	X = Registro Índice	1
ROL OPR,X		4
ROL ,X		3
ROL OPR,SP	SP = Stack Pointer	5

Ejemplo:

LDA #%10101010 ; A = 10101010 (170 d),
ROLA
Si en el momento de efectuar la rotación **Carry = 1**

A = 01010101 ; Carry = 1

Pero, Si en el momento de efectuar la rotación **Carry = 0**

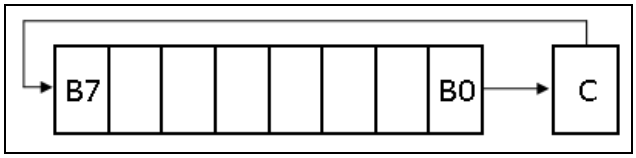
A = 01010100 ; Carry = 1

Obsérvese que en el ejemplo, el valor inicial cargado en el Acumulador fue el 10101010 binario y al realizar la rotación a la izquierda el "1" que se encontraba en el Bit 7, paso al Bit de Carry,

todos los bits se trasladaron una posición a la izquierda y el Bit 0 o Bit de menos peso tomo el valor que presentaba el Bit de Acarreo.

ROR : Rotación a la derecha por acarreo

Descripción: Efectúa la rotación en una posición de todo un registro a la derecha con la diferencia de que la posición que queda en blanco es ocupada por el valor presente en el Bit de Acarreo; es decir, que utiliza el bit de Acarreo como puente para realizar la rotación de los bits pertenecientes a un registro. El Bit que se encontraba en la posición de menos peso (Bit 0) pasa al Bit de Carry y el Bit más peso (Bit 7) será ocupado con el valor que se encontraba en el Bit de Carry, tal como se ilustra a continuación.



Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
ROR OPR	OPR = Dirección de Memoria o Registro	4
RORA	A = Acumulador	1
RORX	X = Registro Índice	1
ROR OPR,X		4
ROR ,X		3
ROR OPR,SP	SP = Stack Pointer	5

Ejemplo:

LDA #%10101010 ; A = 10101010 (170 d),
RORA

Si en el momento de efectuar la rotación **Carry = 1**

A = 11010101 ; Carry = 0

Pero, Si en el momento de efectuar la rotación **Carry = 0**

A = 01010101 ; Carry = 0

Obsérvese que en el ejemplo, el valor inicial cargado en el Acumulador fue el 10101010 binario y al realizar la rotación a la derecha el "0" que se encontraba en el Bit 0, paso al Bit de Carry, todos los bits se trasladaron una posición a la derecha y el Bit 7 o Bit de más peso tomo el valor que presentaba el Bit de Acarreo.

SBC : Resta con acarreo

Descripción: Efectúa la operación de substracción o resta de dos cantidades incluyendo en la substracción el valor presente en el Bit de Carry; la primera cantidad corresponde al Acumulador (Registro A), la segunda cantidad puede ser una constante o un dato almacenado previamente en un registro y la respuesta queda almacenada en el registro Acumulador. La operación realizada es la siguiente:

A = A-(M)-(C)

M = Constante o Valor almacenado en un registro o posición de Memoria

C = Bit de Acarreo

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
SBC #OPR	#OPR = Constante, por ejemplo #10T ; 10 Decimal	2
SBC OPR	OPR = Dirección de memoria o registro	3
SBC OPR,X		3
SBC ,X	X = Registro Índice	2
SBC OPR,SP	SP = Stack Pointer o Apuntador de Pila	4

Ejemplos:

LDA #20T **;** A = 20 , Carga el valor de 10 en el Acumulador
SUB #10T **;** A=A+10 , Resta A con 10

Si C =1 Entonces A = 20 - 10 - 1 = 9
Si C =0 Entonces A = 20 - 10 - 0= 10

MOV #200,$80 ; Almacena el valor 200 en la dirección 80h
LDA #20T ; A = 20
SUB $80 ; A=A-(Valor almacenado en dirección 80h)

Si C =1 Entonces A = 20 - 200 – 1 = -181
Si C =0 Entonces A = 20 - 200 – 0 = -180

MOV #50T,$81 ; Almacena el valor 50 en la dirección 81h
LDX #1 ; X=1
LDA #100T ; A = 100
SUB $80,X; A=A-(Valor almacenado en dirección 80h+X)

Si C =1 Entonces A = 100 - 50 - 1 = 49
Si C =0 Entonces A = 100 - 50 - 0 = 50

SUB : Resta

Descripción: Efectúa la operación de substracción o resta de dos cantidades sin incluir en la substracción el valor presente en el Bit de Carry; la primera cantidad corresponde al Acumulador (Registro A), la segunda cantidad puede ser una constante o un dato almacenado previamente en un registro y la respuesta queda almacenada en el registro Acumulador. Si el resultado de la substracción es negativo, el Bit de Acarreo tomará como valor "1", de lo contrario tomará como valor "0". La operación realizada es la siguiente:

A=A-(M)

M = Constante o Valor almacenado en un registro o posición de Memoria

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
SUB #OPR	#OPR = Constante, por ejemplo #10T ; 10 Decimal	2
SUB OPR	OPR = Dirección de memoria o registro	3
SUB OPR,X		3
SUB ,X	X = Registro Índice	2
SUB OPR,SP	SP = Stack Pointer o Apuntador de Pila	4

Ejemplos:

LDA #20T **;** A = 20 , Carga el valor de 10 en el Acumulador
SUB #10T **;** A=A+10 , Resta A con 10

Entonces A = 20 - 10 = 10

MOV #200,$80 ; Almacena el valor 200 en la dirección 80h
LDA #20T **;** A = 20
SUB $80 **;** A=A-(Valor almacenado en dirección 80h)

Entonces A = 20 - 200 = -180

MOV #50T,$81 ; Almacena el valor 50 en la dirección 81h
LDX #1 **;** X=1
LDA #100T **;** A = 100
SUB $80,X; A=A-(Valor almacenado en dirección 80h+X)

Entonces A = 100 - 50 = 50

TST : Probar si la cantidad es negativa o cero

Descripción: Esta instrucción coloca en 1 los bits N y Z del registro bandera cuando al comparar una cantidad se verifica que es Negativa o Cero respectivamente, sin afectar el contenido de ningún registro. La operación realizada es la siguiente:

Comparación = A-00h
ó
Comparación = X-00h
ó
Comparación = (M)-00h

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
TST OPR	OPR = Dirección de Memoria o Registro	3
TSTA	A = Acumulador	1
TSTX	X = Registro Índice	1
TST OPR,X		3
TST ,X		2
TST OPR,SP	SP = Stack Pointer	4

Ejemplos:

LDA #10T **;** A = 10, Acumulador = 10
TSTA **;**Verifica si el contenido del Acumulador es Negativo o cero

Entonces Comparación = 10 – 00h, ahora se pregunta lo que desea conocer el programador con respecto a la comparación de éstas dos cantidades, por ejemplo, si se quiere que el programa salte a una etiqueta llamada "Cero" si la cantidad que está en el Acumulador (A) es Cero. Debajo de la línea de comparación se coloca la condición de salto según el resultado tal como se ilustra a continuación:

LDA #10T **;** A = 10, Acumulador = 10
TSTA **;**Verifica si el contenido del Acumulador
BEQ Cero Como para el caso A (10d) es diferente de Cero, el programa no salta a la etiqueta "Cero" continuando en la línea siguiente, de lo contrario, se realiza el salto.

MOV #$0A,$80 **;** Almacena 0Ah (10 d) en la dirección 80h
TST $80 Compara el contenido de la dirección 80h (0Ah)
BPL Positivo Como para el caso la cantidad con la que se encontraba en la dirección 80h es positiva, el programa salta a la etiqueta "Positivo", de lo contrario, no se realiza el salto y se continúa en la línea siguiente.

INSTRUCCIONES DE OPERANDOS ESPECIALES

RSP : Reset del puntero de pila

Descripción: Esta instrucción aplica Reset a la Pila en la parte alta de la Pila, es decir, Un registro de la pila tiene un tamaño de 16 Bits; los 8 Bits de más peso son colocados en 0. La operación realizada es la siguiente:

SP = 00FFh

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS

RTI : Retorno de la interrupción

Descripción: Esta instrucción permite retornar de una interrupción a la línea siguiente del programa principal donde se encontraba en el momento de haber sido llamado para atender una interrupción. Siempre, después de atender una interrupción y se desee continuar con la ejecución normal del programa se utiliza esta instrucción. Como la gran mayoría de los lectores han trabajado con microcontroladores PIC, esta instrucción es la equivalente a RETFIE. La operación realizada es la siguiente:

SP = SP + 0001h ; Restablece el CCR de la Pila
SP = SP + 0001h ; Restablece el Acumulador de la Pila
SP = SP + 0001h ; Restablece el X de la Pila
SP = SP + 0001h ; Restablece el PCH de la Pila
SP = SP + 0001h ; Restablece el PCL de la Pila

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
RTI	Retornar de una Interrupción	7

Ejemplo:

En el siguiente programa se puede observar que la interrupción por **TIMER** está establecida en la dirección **FFF2h**

```
TIMER    EQU  $FFF2
```

Normalmente el programa se mantendrá ejecutando el programa principal

```
INICIO      BSET   COPD,CONFIG1      ;Inhabilita el COPD
            MOV    #$00,PORTB        ;PTB7 = 0
            MOV    #$80,DDRB         ;Config.  el  PTB7
salidas
SALTO    BCLR   7,PORTB              ;PTB7 = 0
         BSR    RETARDO     ;Retardo de 1 Segundo
         BSET   7,PORTB              ;PTB7 = 1
         BSR    RETARDO     ;Retardo de 1 Segundo
```

```
            BRA     SALTO                   ;Ir a Salto


RETARDO  BCLR    LED,BANDERA          ;No ha transcurrido 1 Seg.
CONF_TIM MOV     #$36,TSC             ;B'00110110' al TSC
                 MOV     #$3D,TMODH          ;TMODH = $3D
                 MOV     #$09,TMODL          ;TMODL  = $09
                 MOV     #$46,TSC            ;B'01000110' al TSC
            CLI                          ;Habilita Interrupciones

ESPERA   BRSET   LED,BANDERA,FIN      ;Si transcurrio 1 Seg. FIN
            BRA     ESPERA           ;Si nó, ir a Espera
FIN         RTS                      ;Retorno de Subrutina
```

Cuando el TIMER se rebose, provocará interrupción por TIMER, saliendo del programa principal y dirigiéndose a la dirección **FFF2h,** para atender la interrupción.

```
            ORG     TIMER                ;Interrupción por
      TIMER
            DW      TIM
```

Atiende la interrupción y cuado ya se desea volver al programa principal para continuar con su ejecución, se ejecuta la instrucción RTI, dando por finalizado la atención a la interrupción.

```
TIM         PSHH                         ;Guarda el registro H
                 BSET    LED,BANDERA          ;Ya transcurrió 1
Segundo
FIN_INT     BCLR         7,TSC                ;Borra bandera de
interrup
                 PULH                         ;Sacar H del Stack
                 RTI
```

El código completo del programa es el siguiente y será explicado en detalle en el **capítulo 10.**

```
$ INCLUDE 'JL3REGS.INC'

FLASH       EQU  $ECE0
RESET       EQU  $FFFE
TIMER       EQU  $FFF2
RAM         EQU  $80
COPD        EQU  0
LED         EQU  7

            ORG     RAM
BANDERA     RMB     1               ;Registro Bandera de 1 Seg.
```

```
            ORG     FLASH
INICIO      BSET    COPD,CONFIG1        ;Inhabilita el COPD
            MOV     #$00,PORTB          ;PTB7 = 0
            MOV     #$80,DDRB           ;Config. el PTB7  salidas
SALTO       BCLR    7,PORTB             ;PTB7 = 0
             BSR    RETARDO       ;Retardo de 1 Segundo
            BSET    7,PORTB             ;PTB7 = 1
             BSR    RETARDO       ;Retardo de 1 Segundo
             BRA    SALTO               ;Ir a Salto

RETARDO     BCLR    LED,BANDERA         ;No ha transcurrido 1 Seg.
CONF_TIM    MOV     #$36,TSC            ;B'00110110' al TSC
            MOV     #$3D,TMODH          ;TMODH = $3D
            MOV     #$09,TMODL          ;TMODL  = $09
            MOV     #$46,TSC            ;B'01000110' al TSC
            CLI                         ;Habilita Interrupciones

ESPERA      BRSET   LED,BANDERA,FIN     ;Si transcurrio 1 Seg. FIN
            BRA     ESPERA              ;Si nó, ir a Espera
FIN         RTS                         ;Retorno de Subrutina

TIM          PSHH                       ;Guarda el registro H
            BSET    LED,BANDERA         ;Ya transcurrió 1 Segundo
FIN_INT     BCLR    7,TSC               ;Borra bandera de interrup
            PULH                        ;Sacar H del Stack
            RTI

            ORG     RESET               ;Cuando se energiza…
            DW      INICIO              ;Ir a INICIO

            ORG     TIMER               ;Interrupción por TIMER
            DW      TIM
```

RTS : Retorno de subrutina

Descripción: Esta instrucción permite retornar de una Subrutina a la línea siguiente del programa principal donde se encontraba en el momento de haber sido llamada por la instrucción **BSR**. Siempre, después de ejecutar una subrutina y se desee continuar con la ejecución del programa principal se utiliza esta instrucción. Como la gran mayoría de los lectores han trabajado con microcontroladores

PIC, esta instrucción es la equivalente a RETURN. La operación realizada es la siguiente:

SP = SP + 0001h ; Restablece el PCH de la Pila
SP = SP + 0001h ; Restablece el PCL de la Pila

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
RTS	Retornar de una Subrutina	4

Ejemplo:

En el siguiente ejemplo se explica un programa que enciende un LED y luego lo apaga, para encenderlo se llama a la rutina **"LED_ON"** y para apagarlo se llama la rutina **"LED_OFF".**

```
RUTINA    BSR    LED_ON    ; Llama rutina para encender LED
          BSR    LED_OFF   ; Llama rutina para apagar LED
          BRA    RUTINA    ; Salta a etiqueta RUTINA
```

```
; Rutina para Encender un LED
LED_ON    BSET   LED,PORTB          ;Encender Led
          RTS                       ;Retorno de Subrutina
```

```
; Rutina para Apagar un LED
LED_OFF   BCLR   LED,PORTB          ;Apagar Led
          RTS                       ;Retorno de Subrutina
```

JSR : Salto a subrutina

Descripción: Esta instrucción permite saltar a una Subrutina. La diferencia fundamental con la instrucción **BSR,** es que a través de ésta instrucción se pueden realizar saltos a subrutinas más alejadas, es decir, efectuar saltos más largos. Como la gran mayoría de los lectores han trabajado con microcontroladores PIC, esta instrucción es la equivalente a CALL. La operación realizada es la siguiente:

PC = PC + n ; n= 1,2 o 3 Según modo de Direccionamiento
SP = SP + 0001h ; Guarda parte baja del PCL en la Pila
SP = SP + 0001h ; Guarda parte alta del PCL en la Pila

PC = Dirección ; Carga el PC (contador de Programa), con la dirección de inicio de la subrutina solicitada

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
JSR OPR	OPR = Dirección inicial de Subrutina	4
JSR OPR,X		5
JSR ,X		4

Ejemplo:

En el siguiente ejemplo se explica un programa que enciende un LED y luego lo apaga, para encenderlo se llama a la rutina **"LED_ON"** y para apagarlo se llama la rutina **"LED_OFF".**

```
RUTINA    JSR    LED_ON    ; Llama rutina para encender LED
          JSR    LED_OFF   ; Llama rutina para apagar LED
          BRA    RUTINA    ; Salta a etiqueta RUTINA
```

```
; Rutina para Encender un LED
LED_ON    BSET   LED,PORTB          ;Encender Led
                 RTS                ;Retorno de Subrutina
```

```
; Rutina para Apagar un LED
LED_OFF   BCLR   LED,PORTB          ;Apagar Led
          RTS                       ;Retorno de Subrutina
```

STOP : Habilita la IRQ y Para el oscilador

Descripción: Esta instrucción permite reducir el consumo de potencia, eliminando toda disipación de potencia dinámica. Esto resulta por:

- Se coloca en 0 el prescalador del temporizador
- Deshabilita interrupción por TIMER

- Pone en 0 la bandera de interrupción por Temporizador
- Habilita interrupción Externa ($\overline{IRQ}$)
- Y deshabilita el oscilador

Cuando un $\overline{RESET}$ o la entrada de $\overline{IRQ}$ se pone en bajo, se habilita el oscilador, comienza un retardo de 1920 ciclos de reloj del procesador, permitiendo que el oscilador se estabilice, se saca el vector de petición de interrupción o el vector de Reset y se ejecuta la rutina de servicio, dependiendo del signo que fue ejecutado

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
STOP	Habilitar el pin IRQ, detener el Oscilador	1

SWI : Interrupción por software

Descripción: Esta instrucción permite generar una interrupción especial que ofrece grandes ventajas a la hora de desarrollar aplicaciones. La interrupción por software permite al programador interrumpir en un momento deseado la ejecución de una rutina. Siempre, después de atender esta interrupción y se desee continuar con la ejecución normal del programa se utiliza la instrucción **RTI**. La operación realizada es la siguiente:

```
PC = PC + 0001h ; Avanza el PC para devolver la dirección
SP = SP - 0001h  ; Guarda el PCL en la Pila
SP = SP - 0001h  ; Guarda el PCH en la Pila
SP = SP - 0001h  ; Guarda a X en la Pila
SP = SP - 0001h  ; Guarda el Acumulador en la Pila
SP = SP - 0001h  ; Guarda el CCR en la Pila
Bit I = 1        ; Bit de Interrupción en 1
PCH = FFCh       ; Vector dirección para interrupción
PCL = FFDh       ; por Software
```

Sintaxis:

INSTRUCCIÓN	Aclaración	No. CICLOS
SWI	Interrupción por Software	9