ESPECIFICACION DE LOS MODOS DE DIRECCIONAMIENTO

DIRECCIONAMIENTO INHERENTE:

Se usa cuando toda la información requerida para la instrucción esta en el operador.

INCA
INCB
DECA
ROT Instrucción de rotación o corrimiento.
ABA [A]+[B] --> A

DIRECCIONAMIENTO DIRECTO:

Se usa cuando el dato a procesar reside en una localidad de memoria, la cual se especifica primero para acceder a dicho dato.
Requiere dos bytes para su ejecución (1 BYTE [operador] + 1 BYTE [operando])

DIRECCION	CODIGO DE OPERACIÓN	ETIQUETA	NEMOTÈCNICO	OPERANDO
20	**9630**		**LDAA**	**$30**
	96 es el código de operación + 30 que es la dirección del dato que se desea acceder			Observe el símbolo $ para darle sentido de dirección a 30, además una dirección en hexadecimal

DIRECCIONAMIENTO INMEDIATO:

Se usa cuando el operando de la instrucción contiene el dato que se desea procesar (ocupa 2 Bytes). 1 Byte para el operador (instrucción) y 1 Byte para el operador el cual contiene el dato que se desea acceder.

DIRECCION	CODIGO DE OPERACIÓN	ETIQUETA	NEMOTÈCNICO	OPERANDO
55	C6FF		LDAB	#$FF
	C6 es el código de operación + FF que es el dato, en este caso el operando es un dato			Observe como se usa el símbolo # para dar sentido de dato a $FF, y no darle sentido de dirección.

SUPONGA:

Dirección	dato
$50	15
$51	55
$52	6ª

Sumar 15 y 55 y que el resultado se coloque en la localidad $52.

LDAA $50 cargamos el acumulador A de manera directa.
ADDA $51 sumamos de manera directa el acumulador A.
STAA $52 almacenamos el contenido del acumulador A en la localidad 52.

Si el contenido de la localidad de memoria $55 es 2F y el contenido del acumulador B es 15.

$0054	
$0055	**2F**
$0056	
$0057	
$0058	

ACUM B **15**

¿Cuál es el contenido del acumulador B después de que se efectúan las dos siguientes instrucciones.

ADDB #$55
ADDB $55

	1	5
+	5	5
	6	**A**

	6	A		0	1	1	0	1	0	1	0
+	2	F		0	0	1	0	1	1	1	1
	9	**9**		**1**	**0**	**0**	**1**	**1**	**0**	**0**	**1**

Por lo tanto el contenido del acumulador B es 99.

DIRECCIONAMIENTO INDEXADO:

Como su nombre lo indica, el direccionamiento indexado hace uso de los registros (8 o 16 bits) según sea el microcontrolador que se hace.

Las instrucciones requieren 2 bytes para su operación, 1 byte para el operador (instrucción) y 1 byte para el operando el cual en este caso contiene el número que se ha de sumar al registro índice para obtener la dirección de referencia. Cabe hacer notar que el registro índice X maneja solo índices de memoria.

```
LDX     #$0123
ADDA    $A0,X

LDX     #0123
LDAA    $00,X       ;carga al acumulador A con el contenido de la 0123+A0
```

X | 0123 |

	0	1	2	3
+			A	0
	0	**1**	**C**	**3**

	$0122	
Pc -->	$0123	
	$0124	
	$0125	

Pc -->	$01C3	**Dato1**

Indexed (5-bit offset)	**INST** *oprx5,xysp*	IDX	5-bit signed constant offset from X, Y, SP, or PC
Indexed (pre-decrement)	**INST** *oprx3,–xys*	IDX	Auto pre-decrement x, y, or sp by 1 ~ 8
Indexed (pre-increment)	**INST** *oprx3,+xys*	IDX	Auto pre-increment x, y, or sp by 1 ~ 8
Indexed (post-decrement)	**INST** *oprx3,xys–*	IDX	Auto post-decrement x, y, or sp by 1 ~ 8
Indexed (post-increment)	**INST** *oprx3,xys+*	IDX	Auto post-increment x, y, or sp by 1 ~ 8
Indexed (accumulator offset)	**INST** *abd,xysp*	IDX	Indexed with 8-bit (A or B) or 16-bit (D) accumulator offset from X, Y, SP, or PC
Indexed (9-bit offset)	**INST** *oprx9,xysp*	IDX1	9-bit signed constant offset from X, Y, SP, or PC (lower 8 bits of offset in one extension byte)
Indexed (16-bit offset)	**INST** *oprx16,xysp*	IDX2	16-bit constant offset from X, Y, SP, or PC (16-bit offset in two extension bytes)
Indexed-Indirect (16-bit offset)	**INST [***oprx16,xysp***]**	[IDX2]	Pointer to operand is found at... 16-bit constant offset from X, Y, SP, or PC (16-bit offset in two extension bytes)
Indexed-Indirect (D accumulator offset)	**INST [D,***xysp***]**	[D,IDX]	Pointer to operand is found at... X, Y, SP, or PC plus the value in D

El hc12 usa modos indexados de versiones redefinidas del m68hc11 que reducen la ejecución de tiempo y elimina medida del código por usar el registro índice Y. En la mayoría de los casos. Las mejoras del tiempo de ejecución son debidas a un número reducido de ciclos para todas las instrucciones indexadas y la velocidad de más rápida del sistema de reloj.

El esquema de direccionamiento indexado usa un postbyte plus cero, uno, o dos bytes de extensión después del código de operación de la instrucción. El postbyte y las extensiones hacen las siguientes tareas:

1. Especificar cual registro índice es usado
2. Determinar si un valor es usado en un acumulador como un offset.
3. Habilitación automática de un pre- o pos-increment o un pre- o pos-decrement.
4. Especificar medida de un incremento o decremento.
5. Especificación del uso de offset’s signados de 5-, 9- o 16- bits.

Esta aproximación elimina las diferencias entre el uso de los registros X y Y dramáticamente mejorando las capacidades del direccionamiento indexado.

Las mayores ventajas del esquema del direccionamiento indexado son:

1. El apuntador de pila puede ser usado como un registro índice en todas las operaciones indexadas.
2. El contador de programa puede ser usado como un registro índice en todo, pero en modos de autoincrementa o autodecrementa.
3. Acumuladores A,B, o D pueden ser usados como acumulador de offset's.
4. Pre- or post increment o pre- o post decrement por -8 a +8.
5. Una elección de offset's de 5-, 9- o 16-bits signados constantes.
6. Uso de dos nuevos modos indexado-indirecto. – modo indexado-indirecto con 16 bit de offset. – modo indexado indirecto con offset de acumulador D.

La siguiente tabla es un resumen de las capacidades del direccionamiento indexado y una descripción de la codificacion del postbyte. El postbyte es localizado como **xb** en la descripción de las instrucciones. Descripciones detalladas de las variaciones del modo del direccionamiento indexado son mostradas seguidas a la tabla.

Table 3-2. Summary of Indexed Operations

Postbyte Code (xb)	Source Code Syntax	Comments rr; 00 = X, 01 = Y, 10 = SP, 11 = PC
rr0nnnnn	,r n,r –n,r	5-bit constant offset n = –16 to +15 r can specify X, Y, SP, or PC
111rr0zs	n,r –n,r	Constant offset (9- or 16-bit signed) z- 0 = 9-bit with sign in LSB of postbyte(s) –256 ≤ n ≤ 255 1 = 16-bit –32,768 ≤ n ≤ 65,535 if z = s = 1, 16-bit offset indexed-indirect (see below) r can specify X, Y, SP, or PC
111rr011	[n,r]	16-bit offset indexed-indirect rr can specify X, Y, SP, or PC –32,768 ≤ n ≤ 65,535
rr1pnnnn	n,–r n,+r n,r– n,r+	Auto predecrement, preincrement, postdecrement, or postincrement; p = pre-(0) or post-(1), n = –8 to –1, +1 to +8 r can specify X, Y, or SP (PC not a valid choice) +8 = 0111 ... +1 = 0000 –1 = 1111 ... –8 = 1000
111rr1aa	A,r B,r D,r	Accumulator offset (unsigned 8-bit or 16-bit) aa-00 = A 01 = B 10 = D (16-bit) 11 = see accumulator D offset indexed-indirect r can specify X, Y, SP, or PC
111rr111	[D,r]	Accumulator D offset indexed-indirect r can specify X, Y, SP, or PC

All indexed addressing modes use a 16-bit CPU register and additional information to create an effective address. In most cases the effective address specifies the memory location affected by the operation. In some variations of indexed addressing, the effective address specifies the location of a value that points to the memory location affected by the operation.

Indexed addressing mode instructions use a postbyte to specify index registers (X and Y), stack pointer (SP), or program counter (PC) as the base index register and to further classify the way the effective address is formed. A special group of instructions cause this calculated effective address to be loaded into an index register for further calculations:

- Load stack pointer with effective address (LEAS)
- Load X with effective address (LEAX)
- Load Y with effective address (LEAY)

DIRECCIONAMIENTO INDEXADO – OFFSET 5 BIT CONSTANT

Este modo de direccionamiento indexado usa un offset signado de 5 bits el cual es incluido en la instrucción postbyte. Este corto offset es sumado a la base del registro índice (X, Y, SP, or PC) para formar la dirección efectiva de la localización de memoria que será afectada por la instrucción. Esto da un rango de -16 through +15 de el valor en la base del registro indice. Aunque otro modo de direccionamiento indexado permite offset's de 9 o 16 bits, aquellos modos requieren también de bytes de extensión adicional en la instrucción para esta información extra.

Ejemplo:

```
LDAA    0, X
STAB    -8, Y
```

Para este ejemplo asuma que X tiene un valor de $1000, y Y tiene un valor de $2000, antes de la ejecución.
El modo de offset de 5 bits no cambia el valor en el registro índice, así X aún será $1000 y Y aun seguirà siendo $2000 después de la ejecución de estas instrucciones.

En el primer ejemplo A será cargado con el valor de la dirección $1000.
En el segundo ejemplo, el valor del acumulador B será almacenado $1FF8 (que es $2000-$8).

DIRECCIONAMIENTO INDEXADO – OFFSET 9 BIT CONSTANT

Este modo de direccionamiento indexado usa un offset signado de 9 bits el cual es sumado a la base del registro indice (X,Y,SP o PC) para formar la dirección efectiva de la localización de memoria afectada por a instrucción. Esto da un rango de -256 a +255 del valor en la base del registro indice. El bit mas significativo (bit de signo) del offset es incluido en la instrucción postbyte y los 8 bits son entregados como un byte extensión después de la instrucción postbyte en el flujo de la instrucción.

Ejemplo:

```
LDAA    $FF, X  ;cargarà A con el valor de la dirección $10FF.
LDAB    -20, Y  ;cargarà B con el valor de la dirección $1FEC.
```

Asuma x=$1000 y=$2000 antes de la ejecución de estas instrucciones. Estas instrucciones no alteran los registros después de la ejecución.

DIRECCIONAMIENTO INDEXADO – OFFSET 16 BIT CONSTANT

Este modo de direccionamiento indexado usa un offset signado de 16 bits el cual es sumado a la base del registro indice (X,Y,SP o PC) para formar la dirección efectiva de la localizaciòn de memoria afectada por a instrucción. Esto permite acceder a alguna direcciòn en el espacio de dirección de 64 Kbyte. Puesto que el bus de direcciones y el offset son ambos de 16 bits no importa si el valor del offset es considerado a ser signado o un valor no signado. ($FFFF podria ser pensado como +65,535 o como -1).

DIRECCIONAMIENTO INDIRECTO-INDEXADO – OFFSET 16 BIT CONSTANT

Este modo de direccionamiento indexado suma un offset 16-bit instruction-supplied.

SANTIAGO CRUZ CARLOS

DIRECCIONAMIENTO RELATIVO:

Se utiliza para las instrucciones de ramificación o de salto.

RELATIVO	**INCONDICIONAL**	Ocupan 2 bytes 1 byte código de operación 1 byte un desplazamiento
	CONDICIONAL	El desplazamiento se trata como un número de 8 bits signado en complemento a 2 que se suma al program counter para obtener la dirección de salto.

	BRA	**SIGUE**	($0276)
SIGUE			

Tenemos 1 byte para operando, 2^8=256
¿Cuantos números podemos representar con 256 combinaciones, incluyendo el cero?
256/2=128
127 positivos + 1 cero
128 negativos

Los números negativos se obtienen sacando el complemento a 2 de los positivos[1].
Ejemplo: saque el negativo de 2

02 en Hexadecimal			**0**				**2**	
	0	0	0	0	0	0	1	0
Complemento a uno	1	1	1	1	1	1	0	1
+								1
Complemento a dos	1	1	1	1	1	1	1	0
-02 en hexadecimal			**F**				**E**	

decimal									hexadecimal
0	**0**	0	0	0	0	0	0	0	$00
1	**0**	0	0	0	0	0	0	1	$01
*									
*									
+127	**0**	1	1	1	1	1	1	1	$7F
-128	**1**	0	0	0	0	0	0	0	$80
*									
*									
-2	**1**	1	1	1	1	1	1	0	$FE
-1	**1**	1	1	1	1	1	1	1	$FF

[1] Los bit en negritas son los bits de signo respectivos.

SANTIAGO CRUZ CARLOS

Para mayor comprensión realice el siguiente ejercicio:

	DIR	COD OP		NEMOTECNICO	OPERANDO
PC -->	212B	202C		BRA	$2169
PC+1 -->	213C				
PC +2-->	213D				
		OFFSET			
	2169				

¿Puede saltar a la $2169?

```
   2  1  6  9   OJO: para hacer la resta, 9 le pide una unidad a 6, se convierte en A
-  2  1  3  D   D para llegar a A, deben haber C unidades hexadecimales. :D
   0  0  2  C   El 6 se convierte en 5
```

Ahora analice, vea que el salto de +127, es decir, $7F es mayor que $2C, o $2C<$7F, recuerde que el salto es signado, por lo que no viola la restricción y si puede saltar a la $2169

	DIR	COD OP		NEMOTECNICO	OPERANDO
	$212E				
	213ª		OFFSET		
PC -->	213B	20F1		BRA	$212E
PC+1 -->	213C				
PC +2-->	213D				

```
   2  1  3  D   OJO: para hacer la resta, D le pide una unidad a 3, se convierte en F
-  2  1  2  E   E para llegar a D, debe haber F unidades hexadecimales. :D
   0  0  0  F   El 3 se convierte en 2
```

Luego a 0F debemos sacarle el complemento a 2

```
                    0        F
                 0  0  0  0  1  1  1  1
Complemento a uno  1  1  1  1  0  0  0  0
                +                       1
Complemento a dos  1  1  1  1  0  0  0  1
                          F        1
```

Ahora analice, vea que el salto de +128, es decir, $80 es mayor que $F1, o $F1<$80, recuerde que el salto es signado, por lo que no viola la restricción y si puede saltar a la $212E

DIRECCIONAMIENTO RELATIVO CONDICIONADO

Se efectúa solo cuando un código de condición particular o combinación de códigos particulares son establecidos (SET) o eliminados (CLEAR) por lo tanto el resultado de las instrucciones que preceden a una instrucción condicionada determina si la derivación o salto se efectúa o no se efectúa. Dichos resultados se ven en el **Registro de Condición de Estados** (BANDERAS).

Este registro proporciona información acerca de la última operación efectuada en la ALU

	7	6	5	4	3	2	1	0
CCR	**S**	**X**	**H**	**I**	**N**	**Z**	**V**	**C**

BANDERA "Z"

El bit Z es **'1'** siempre que el resultado de una instrucción sea **'0'** y se manifiesta en un registro o en una localidad de memoria.

CPMA #$20 ; (A) – ($20), compara el contenido del acumulador A con el contenido de la dirección $20,

tome en cuenta: que al hacer la comparación en ningún momento se modifica el contenido del acumulador A.

BANDERA "C". El bit C es **'1'** en cualquiera de los siguientes casos:

1) Por suma cuando el resultado de la misma produce un acarreo de salida.

```
   1 0 1 1
+  1 0 0 0
C=1 0 0 1 1
```

2) Por instrucciones de substracción p comparación cuando el valor absoluto del substraendo es mayor que el valor absoluto del minuendo.
3) Por instrucción de corrimiento o rotación. SHIFT o ROTATE.

1	**0**	**1**	**1**	**0**	**1**	**0**	**1**		
Corrimiento									
0	**1**	**0**	**1**	**1**	**0**	**1**	**0**		C=**1**

4) Cuando se efectúa SEC (SET CARRY)

BANDERA "V"

El bit **"V"** es **'1'** cuando el resultado de una operación aritmética produce un sobreflujo o bajo flujo en complemento a 2.

Sobreflujo ocurre cuando el resultado de una operación aritmética produce un número más positivo que la capacidad de almacenamiento del registro.

Bajoflujo ocurre cuando el resultado de una operación aritmética produce un número más negativo que la capacidad de almacenamiento del registro, esto así mismo afecta el bit de signo.

N	V		
0	**0**	Resultado positivo	
0	**1**	Bajoflujo	Resultado aparentemente positivo
1	**0**	Resultado negativo	
1	**1**	Sobreflujo	Resultado aparentemente negativo

BANDERA "N"

El bit "N" es **'1'** que el resultado de una operación sea negativo (que el bit màs significativo sea **'1'**)

BANDERA "H"

El bit "H" es **'1'** cuando en una suma se produce un acarreo de salida del bit 3 hacia el bit 4 en un registro.

```
        C=1
0 0 0 0   1 1 0 1
0 0 0 0   1 0 1 0
          0 1 1 1
```

Note que es para saber cuando llevamos un **'1'** a la mitad.

BANDERA "I"

El bit **"I"** se hace **'1'** cuando ocurre una interrupción en el microprocesador o microcontrolador. Al ocurrir una interrupción el microprocesador termina la instrucción que esta en proceso y el control se deriva al efectuar una rutina de interrupción, que no es más que un programa alterno que se ejecuta para tener dicha interrupción y sus modos de operación se verán más adelante.

BANDERA "X"

Deshabilita la fuente de interrupción No enmascarable neg(XIRQ)

CCR	S	X=1	H	I	N	Z	V	C

Cuando se hace un reset en el mP, la bandera x se pone en **'1'**

Se puede poner por software a **'0'**, esta, ya no se puede volver a poner a **'1'** por software dentro del mismo programa hasta otro reset del mP.

BANDERA "S"

El bit **"S"** deshabilita la instrucción stop. (para el reloj, mantiene el estado del mC)

SALTO SIMPLE

Instrucción		Ec. Booleana	Prueba
BCS	Branch carry set	C=1	C=1
BCC	Branch carry clear	C=0	C=0
BEQ	Branch equal	Z=1	Registro = 0 / reg1=reg2
BNE	Branch no equal	Z=0	Registro ≠ 0 / reg1≠reg2
BMI		N=1	MSB=1
BPL		N=0	MSB=0
BUS		V=1	Resultado > reg [reg>+127]
BUC		V=0	-Res > -reg [-res > -128]

SALTO P/NUMEROS NO SIGNADOS

Instrucción	Ec. Booleana	Prueba
BLO	C=1	regm < regn
BHS	C=0	regm ≥ regn
BHI	C+Z=0	regm > regn
BLS	C+Z=1	Regm ≤ regn

Deducción:

C	Z	C+Z	
0	0	0 --> r>m	
0	1	1 --> r=m	≤
1	0	1 --> r<m	
1	1	1 --> por operación suma	

SALTO P/NUMEROS SIGNADOS

Instrucción	Ec. Booleana	Prueba
BGE	N ⊕ V = 0	regm ≥ regn
BLT	N ⊕ V = 1	regm < regn
BGT	Z +(N ⊕ V) = 0	regm > regn

BLE	Z +(N ⊕ V) = 1	Regm ≤ regn

DIRECCIONAMIENTO EXTENDIDO

Ocupa 3 bytes para su código de operación.

1 Byte operador (instrucción)
2 Byte operandos, los cuales tienen en esta ocasión la dirección de referencia.

Ejemplos:

```
LDAB   $2F6
STAB   $2F46
LDAA   #3F4    ;ERROR, el acumulador A es de 8 bits, no puedes cargarle 16 bits
JMP    $2F6A
```

APUNTADOR DE PILA (stack pointer)

Es un registro dentro del mP que contiene la dirección de la anterior localidad disponible para el stack. (para el HC12)

PILA (stack)

Es una área de memoria que se usa para el almacenamiento de información importante o datos que son cargados frecuentemente.

Tipo de pila: LIFO - LAST INPUT FIRTS OUTPUT

Instrucciones que se realizan con el stack pointer:

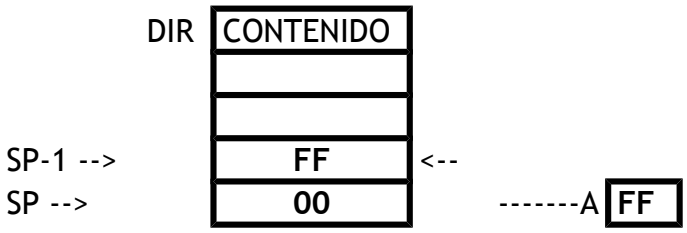
DES	decrementa sp
INS	incrementa el sp
LDS	carga el sp
TXS	transfiere el contenido de X al sp
TSX	transfiere el contenido del sp al X
STS	almacena el sp (directo, index, ext.)

INSTRUCCIONES PUSH - PULL

PUSH

Una instrucción **push** decrementa primero el stack pointer y después toma el contenido del acumulador y lo transfiere a la localidad que apunta el stack pointer en la pila del stack

Ejemplo: PUSHA

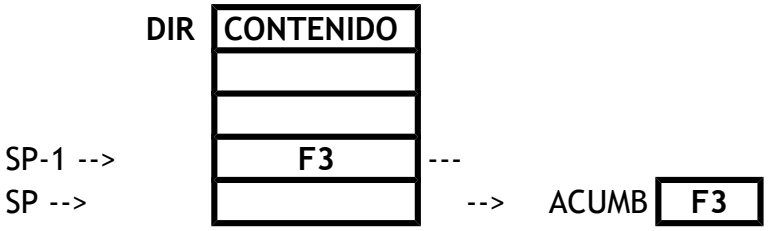


1º. Decrementa apuntador
2º. Mete dato D1

PULL

Una instrucción **pull** toma primero el contenido de la localidad que apunta el stack pointer en la pila del stack y después incrementa el stack pointer.

Ejemplo: PULLB

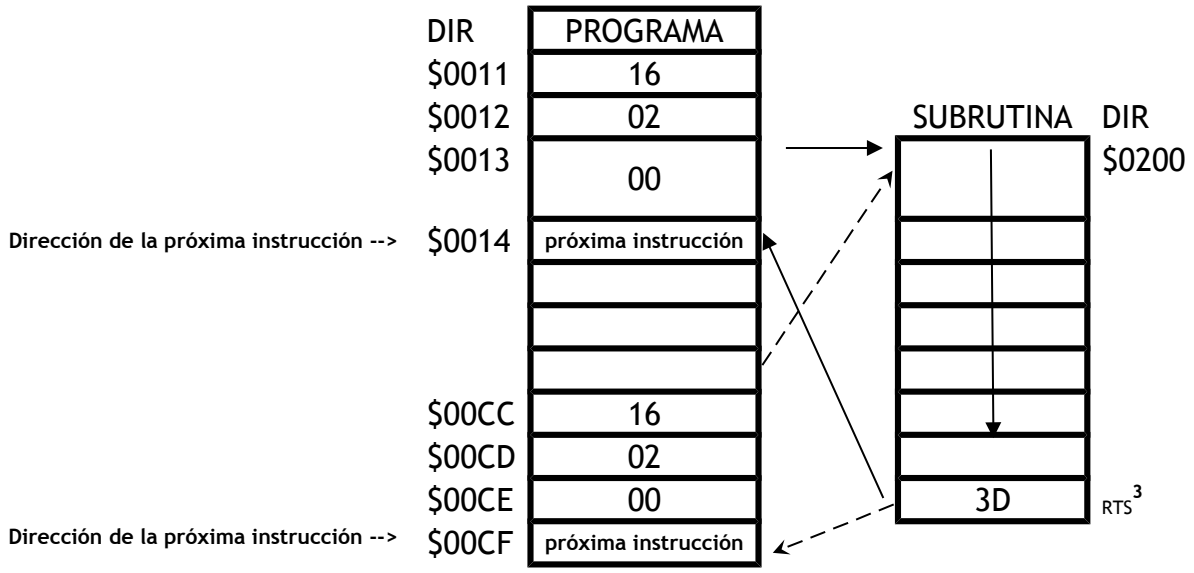


1º. Transfiere el dato
2º. Incrementa el sp

SUBRUTINAS

Es un programa pequeño que se usa más de una vez en el programa principal. Instrucción usada JSR[2]

Etiqueta JSR $0200

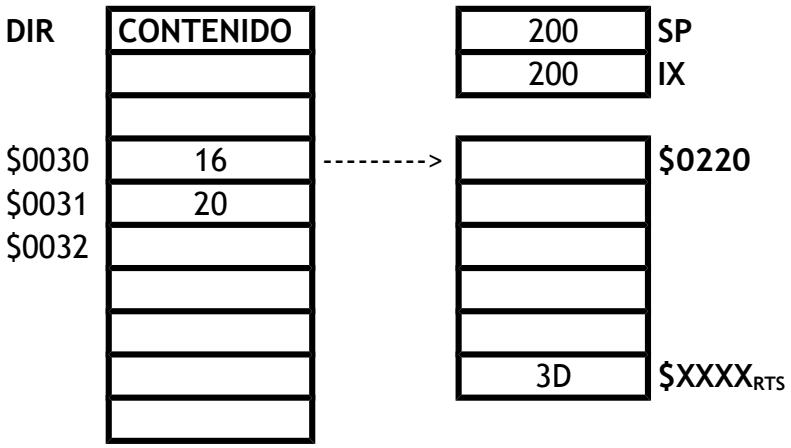


Verifique las flechas, indican la ida y regreso de la ejecución del programa.
¿Cómo sabe a donde regresar? Se guarda en la pila la dirección de la próxima instrucción.

Ejemplo, de línea de código:

Si el Sp y el Ix contiene ambos 200 y la instrucción **JSR $20,X** localizada en la dirección 30 se ejecuta.

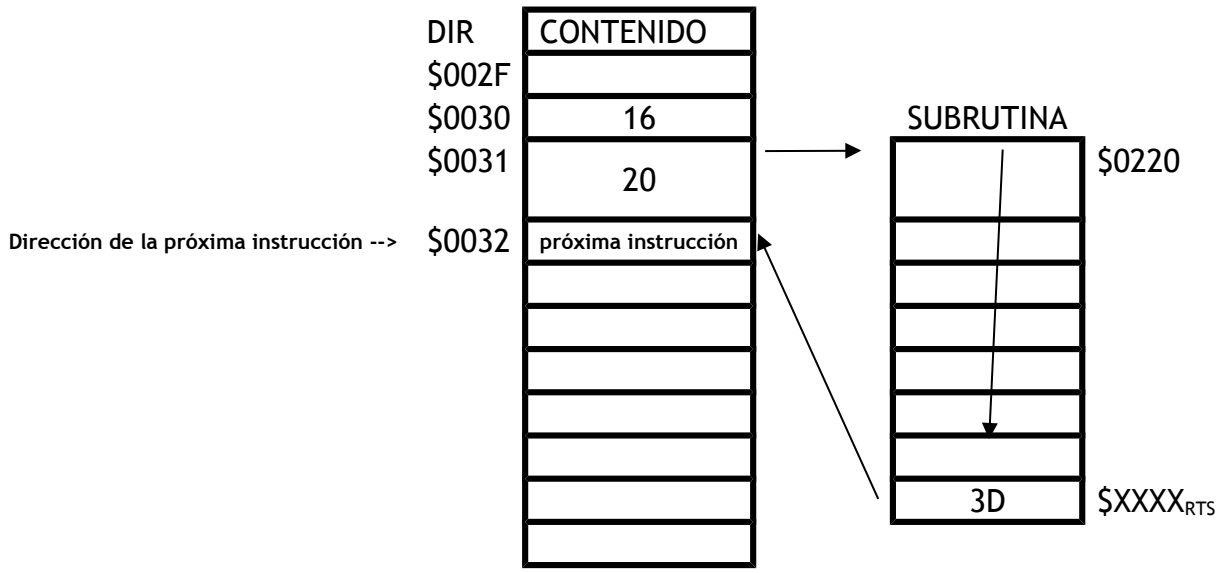
Etiqueta JSR $20, X



[2] El código de operación para la instrucción JSR es 16h (jump subroutine)
[3] El código de operación de la instrucción RTS es 3Dh (return subroutine)

SANTIAGO CRUZ CARLOS

Determinar la dirección de inicio de la subrutina.
Determinar el contenido final del sp y el contenido del stack.



	DIR	STACK
SP-2 -->	$01FE	**00**
SP-1 -->	$01FF	**32**
SP -->	$0200	**00**

RESPUESTA:

a) $0220 que es por la instrucción indexada
b) 01FE al finalizar de meter la instrucción apunta a 01FE
c) Lo que aparece en la pila (stack)

OJO: que lo único que se efectuó fue la instrucción JSR $20, X,

INSTRUCCIÓN MOVE

```
MOVEB                    ;mover un byte
MOVEW                    ;mover una palabra
```

Estas instrucciones tienen excepciones en la forma en que se direccional.

	ORIGEN	DESTINO
A)	IMM	EXT
B)	IMM	IDX
C)	EXT	EXT
D)	EXT	IDX
E)	IDX	EXT
F)	IDX	IDX

EJEMPLO A) IMM-EXT

```
MOVB  #$ii, $hhll      00 ≤ ii ≤ FF
                       0000 ≤ hhll ≤ FFFF

MOVB  #$2F, $0860
```

EJEMPLO B) IMM-IDX

```
MOVB  #$ii, $l, X      00 ≤ ii ≤ FF
                       -16 ≤ l ≤ 15

MOVB  #$2F, $F,X
MOVB  #$2F, -15,X
```

EJEMPLO C) EXT-EXT

```
MOVB   $hhll, $hhll    0000 ≤ hhll ≤ FFFF

MOVB  $0850, $0860
```

EJEMPLO D) EXT-IDX

```
MOVB   $hhll, $l, X    0000 ≤ hhll ≤ FFFF
                          -16 ≤ l ≤ 15
MOVB  $0850, $A, X
```

EJEMPLO E) IDX –EXT

MOVB $l, X, $hhll -16 ≤ l ≤ 15
0000 ≤ hhll ≤ FFFF

MOVB $0,X, $0860

EJEMPLO F) IDX-IDX

MOVB $l, X, $l, Y -16 ≤ l ≤ 15

MOVB $1,X+, $1,+Y saca valor, post-incrementa, pre-incrementa y mete valor

MODOS DE OPERACIÓN

El microprocesador puede operar en 8 diferentes modos. Cada modo tiene un diferente mapa de memoria y un bus de configuración externa. Después del reset, la mayoría de los recursos del sistema pueden ser mapeados a otras direcciones escribiendo en los registros de control apropiados.

MODOS DE OPERACIÓN

EJEMPLO: ¿QUÈ HACE LO SIGUIENTE?

```
REGRESA  MOVW  $2,X+, $2,Y+    ;mueve una palabra y actualiza apuntadores
         DBNE  B, REGRESA      ; decrementa y salta (repite) si no es igual a cero
```

COMUNICACIÓN PARALELA

Puertos A, B, E y K.

Características comunes:

- Bidireccionales
- Programables pin a pin como entrada o salida.
- Tienen un registro para habilitar o deshabilitar “pull-resistor”
- Registro para habilitar o deshabilitar salida en baja potencia.
- Registro para programar modos de operación.

A continuación se presenta un diagrama a bloques de lo que es un pin de un puerto en el HC12: