

Practica 8.- “Direccionamientos Indexados”.

2.2.2.1- Modos de direccionamiento indizados simples.

OBJETIVO DEL SUBTEMA. Modificar el programa que se diseñó en el punto 2.2.2. El programa actualizado además, deberá generar el código máquina de los direccionamientos indexados simples (5, 9, 16 bits).

INTRODUCCION

Direccionamiento Indexado.

En este modo de direccionamiento, la dirección del operando también se calcula sumando un registro de la CPU al campo de operando, este registro es un registro específico para este uso llamado registro índice. En los ordenadores con organización de registros generales, el registro índice puede ser cualquiera de los registros de la CPU. En los ordenadores en que el contador de programa es considerado como un registro de uso general el modo relativo es un caso particular del direccionamiento indexado. A la cantidad que hay que sumar al registro índice para conseguir la dirección del operando también se le llama desplazamiento u offset. Este modo de direccionamiento es especialmente útil para el direccionamiento de vectores y matrices en bucles ya que, si se quieren direccionar elementos consecutivos del vector o matriz, basta mantener en el desplazamiento la dirección del primer elemento e ir incrementando el registro índice. También sirve para acceder de forma relativa a elementos de vectores cercanos a uno dado, para ello, se carga la dirección del elemento de referencia en el registro índice y después se accede mediante direccionamiento indexado, con el desplazamiento adecuado, al anterior, al siguiente, etc., esto mismo también es aplicable a pilas, en que, en ocasiones, hay que acceder a datos cercanos, por encima o por debajo, al dato señalado por algún apuntador. Una consecuencia de todo esto es una modalidad de direccionamiento indexado de que disponen algunos ordenadores, denominada autoindexación, que hace que el registro índice sea incrementado o decrementado en el tamaño del operando antes o después de acceder al mismo. Los ordenadores que poseen autoindexación incorporan los modos de direccionamiento descritos en los dos apartados siguientes. En algunos ordenadores existen variantes del direccionamiento indexado en que se obtiene la dirección del operando sumando el contenido de varios registros con el desplazamiento, esto puede servir para especificar el comienzo de un vector mediante un desplazamiento respecto a un registro y el elemento del vector mediante un registro índice. En este direccionamiento la dirección del dato de la instrucción se obtiene sumando el primer operando al contenido de un registro índice que puede ser X, Y, SP o PC especificado en el segundo operando. $EA = r_i + \text{desplazamiento}$, en donde r_i es uno de los registros índice.

Direccionamiento indexado con desplazamiento constante

En este direccionamiento el desplazamiento (primer operando) es un número constante de 5, 9 o 16 bits.

CONLOC COP xb <etiqueta> INST oprx5,xysp

CONLOC COP xb ff <etiqueta> INST oprx9,xysp

CONLOC COP xb ee ff <etiqueta> INST oprx16,xysp

Donde $-16 \leq \text{oprx5} \leq 15$, $-256 \leq \text{oprx9} \leq 255$ y $0 \leq \text{oprx16} \leq 65535$;

el post byte xb se obtiene de la siguiente manera para el modo de 5 bits:

xb = rr0nnnnn como número binario.

No debemos de confundir esta rr con la obtenida para el modo de direccionamiento relativo,

aquí la rr es un numero binario de dos bits que se refiere al registro índice que se señala en el operando y que puede ser:

x **00**
y **01**
sp **10**
pc **11**

Y nnnnn se refiere a oprx5, completados a 5 bits. Una vez que se tienen todos los datos y se sustituyen en la formula se toman los 4 bits más significativos y se transforman a su equivalente en formato hexadecimal, lo mismo se hace con los 4 bits menos significativos, los 2 dígitos hexadecimales se juntan y ellos representan el post byte xb.

Para los modos indizados de 9 y 16 bits el post byte xb se obtiene con la formula:

xb = 111rr0zs binario

Z es 0 para el modo de 9 bits y 1 para el modo de 16 bits; s es 0 para el modo de 16 bits y para el modo de 9 bits es igual al bit de signo de la representación binaria de oprx9.

Para el modo indizado de 9 bits, ff se refiere a la representación en 1 byte hexadecimales de los 8 bits menos significativos de oprx9; y para el modo indizado de 16 bits ee es la representación en 1 byte de los 8 bits más significativos de oprx16 y ff la representación en hexadecimal de los 8 bits menos significativos del mismo oprx16.

He aquí un ejemplo:

```
0000          ORG $4300
4300 CE 08 00      Idx  #$0800
4303 69 00      LIMPIA  clr  0,x
4305 08          inx
4306 8E 09 00      cpx  #$0900
4309 25 F8      blo  LIMPIA
430B 3D          rts
430C          END
```

Direccionamiento indexado con desplazamiento del acumulador

En este direccionamiento el desplazamiento es uno de los registros acumuladores (A, B o D).
EA = ri + acumulador.

CONLOC COP xb <etiqueta> INST abd,xysp

Donde el post byte xb se calcula con la siguiente fórmula:

xb = 111rr1aa también en binario.

El acumulador es el que nos va a dar aa, ya que si este es el acumulador A, aa será igual a 00, si por otra parte es el acumulador B será 01 y por ultimo si se trata del acumulador D, aa será 10.

Veamos pues un ejemplo, el cual convierte un número hexadecimal a su representación en BCD:

```
0000          ORG $0E43
0E43 CE 0E 4E      Idx  #TABLA
0E46 96 00      OTROldaa $00
0E48 84 0F      anda #$0F
0E4A A6 E4      ldaa A,X
0E4C 20 F8      bra  OTRO
0E4E 00 01 02 03 TABLA  DB  %0000, %0001, $02, $03
0E52 04 05 06 07      DB  $04, $05, $06, $07
0E56 08 09 10 11      DB  $08, $09, $10, $11
```

```

0E5A 12 13 14 15      DB   $12, $13, $14, $15
0E5E                  END

```

Indexado con pre/post incremento/decremento

En este direccionamiento a diferencia de los demás indexados la dirección efectiva se obtiene solo del contenido del registro índice pero este valor se calcula después de incrementar o decrementar el contenido del registro índice, si se trata de pre incremento o decremento respectivamente. Si se trata de post incremento o decremento, primero se calcula la dirección efectiva y luego se incrementa o decrementa el registro índice. El valor en que va a ser incrementado o decrementado el registro índice se especifica en el primer operando y puede ser un valor entre 1 y 8.

CONLOC COP xb <etiqueta> INST oprx3,+xys ; pre-incremento

CONLOC COP xb <etiqueta> INST oprx3,-xys ; pre-decremento

CONLOC COP xb <etiqueta> INST oprx3,xys+ ; post-incremento

CONLOC COP xb <etiqueta> INST oprx3,xys- ; post-decremento

Esos son los formatos para el pre incremento, pre decremento, post incremento y post decremento respectivamente, donde $1 \leq \text{opr}x3 \leq 8$. Donde el post byte xb se obtiene con la formula:

xb = rr1pnnnn binario como todos los xb

Como ya hemos visto, rr depende del registro índice al cual se haga referencia en el mismo operando; p es 0 para pre y 1 para post, y por ultimo nnnn representa $\text{opr}x3 - 1$, si se trata de un incremento o $-\text{opr}x3$ para decremento en un formato de 4 bits.

Esto se aprecia mejor en el siguiente ejemplo:

```

0000          ORG   $E00
0E00 CE 08 00      Idx   #$0800
0E03 69 30  LIMPIA  clr   1,x+
0E05 8E 09 00      cpx   #$0900
0E08 25 F9        blo   LIMPIA
0E0A 3D           rts
0E0B           END

```

1.4.6 Direccionamiento Indirecto

Direccionamiento Indirecto por Registro

Este direccionamiento utiliza el contenido de registros internos como una dirección de memoria. La dirección efectiva es igual al contenido del registro índice, este registro trabaja como apuntador, en el HC12 se puede obtener con el direccionamiento indexado de desplazamiento constante de 5 bits haciendo el desplazamiento igual a 0.

Direccionamiento Indirecto por Memoria.

En este otro direccionamiento indirecto es una localidad de memoria la que contendrá el valor de la dirección efectiva, dentro de la memoria se encuentra el apuntador a otra dirección de memoria, en el caso del HC12 existe combinado con el direccionamiento indexado.

Direccionamiento indexado indirecto

Existen 2 tipos de direccionamiento indexado indirecto, el de 16 bits:

-Direccionamiento indexado indirecto de 16 bits de desplazamiento constante

Este modo de direccionamiento indexado añade a los 16 bits de la instrucción de desplazamiento a la base del registro índice para formar la dirección de la localidad de

memoria que contiene un puntero a la memoria afectada por la instrucción. La instrucción misma no es al puntero a la dirección de memoria la que actúa, mejor dicho a la localidad de memoria del puntero a la dirección a actuar.

CONLOC COP xb ee ff <etiqueta> INST [opr16,xysp]

Donde xb se puede obtener con esta fórmula:

xb = 111rr011 binario

Y en la cual rr se refiere al registro índice; y ee ff es opr16 representado en 2 bytes.

Y también existe el direccionamiento indexado indirecto del acumulador D:

Este direccionamiento indexado añade el valor en el acumulador D al valor en la base del registro índice para formar la dirección de memoria que contiene el puntero de la localidad de memoria afectada por la instrucción.

<etiqueta> INST [D,xysp]

Y para el cual se calcula su código máquina así:

COP xb [D,xysp]

Donde xb viene de la siguiente fórmula:

111rr111

En la cual (y como hemos visto anteriormente) el post byte xb viene dado por el registro índice.

Al igual que en las anteriores, debemos tener el archivo .lst para sustituir el código máquina de los direccionamientos indexados en sus diferentes formatos.

El código máquina de los indexados puede ser:

Xb
Xb ff
Xb ee ff

Para obtener el código máquina hay una serie de pasos para obtener el xb. Primero debemos de analizar los formatos del xb para cada variante de los indexados:

					XB
IDX	5bits				rr0nnnnn
IDX	3bits	pre/post	inc/dec		rr1pnnnn
IDX		acumulador			111rr1aa
IDX1	9bits				111rr00s
IDX2	16bits				111rr01s
[IDX2]		indirecto			111rr011
[D,IDX]					111rr111
rr					
x = 00		si el operando es positivo	s=0	A=00	post p =1
y = 01		si el operando es negativo	s=1	B=01	pre p =0
sp=10				D=10	
pc=11					

Luego hay que sustituir las rr de acuerdo al registro que se utilice, además de las aa, s, p y n's cuando sea necesario.

DESARROLLO

Primero utilice unas banderas en la parte del código en que se valida el tipo de IDX, para

saber cuál será el xb que utilizaré, además de hacer validaciones para los s, p, aa y n's en la parte del código que sea necesario, ya que no todos los IDX los utilizan.

Para los "aa" les asigne su valor en decimal, no en binario.

A=0

B=1

D=2.

Para los registros x, y, sp y pc, también les asigne su valor en decimal:

X=0

Y=1

Sp=2

Pc=3

Después hice un algoritmo que haga el cálculo del xb.

Para esto debemos checar que tipo de indexado es y asignarle su valor pero también en decimal:

IDX	5 bits.....	xb=0
IDX	3 bits pre/post inc/dec.....	xb=32
IDX	acumulador.....	xb=228
IDX1	9bits.....	xb=224
IDX2	16bits.....	xb=226
[IDX2]	indirecto.....	xb=227
[D,IDX]	xb=231

Después solo utilizamos corrimientos y compuertas or para calcular xb.

Para el IDX de 5 bits:

nnnn=opr5 & 31

- 1.-Hacemos un corrimiento de 6 posiciones a rr.....rr=rr<<6.
- 2.-Hacemos una compuerta or entre xb y rr.....xb=xb|rr.
- 3.-Hacemos una compuerta or entre xb y nnnn.....xb=xb|nnnn.

Para IDX de 3 bits:

Para incremento (+xys ó xys+)

nnnn=opr3-1

para decremento (-xys ó xys-)

nnnn=opr3*(-1)&15

- 1.-Hacemos un corrimiento a rr de 6 posiciones.....rr=rr<<6.
- 2.-Hacemos una compuerta or entre xb y rr.....xb=xb|rr.
- 3.-Hacemos un corrimiento a p de 4 posiciones.....p=p<<4.
- 4.-Hacemos la compuerta or entre xb y p.....xb=xb|p.
- 5.-Terminamos con una compuerta or entre xb y nnnn.....xb=xb|nnnn.

Para IDX de acumulador:

1. Corrimiento a rr de 3 posiciones.....rr=rr<<3.
2. Compuerta or entre xb y rr.....xb=xb|rr.
3. Compuerta or entre xb y aa.....xb=xb|aa.

Para IDX1 y IDX2:

1. Corrimiento a rr de 3 posiciones.....rr=rr<<3.
2. Compuerta or entre xb y rr.....xb=xb|rr.
3. Compuerta or entre xb y s.....xb=xb|s.

Para [IDX2] y [D,IDX]:

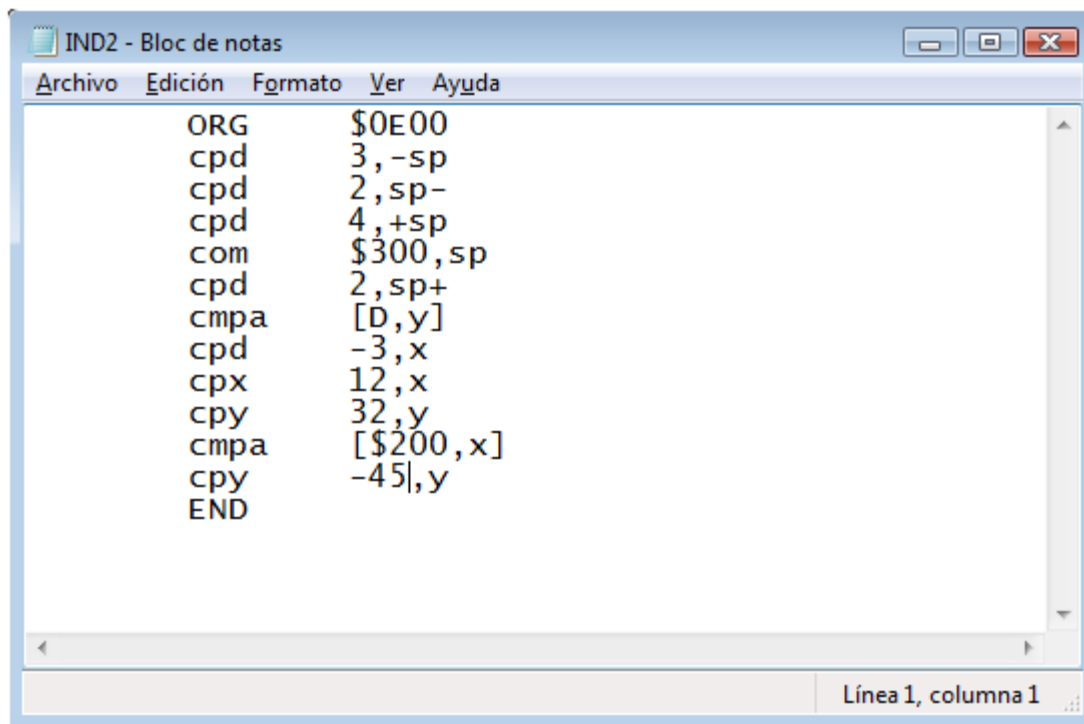
1. Corrimiento a rr de 3 posiciones.....rr=rr<<3.
2. Compuerta or entre xb y rr.....xb=xb|rr.

Después de obtener xb debemos sustituirlo en el código maquina.

Además, cuando tenemos xb ee y xb ee ff, debemos hacer lo mismo que en la practica 6; tomar el valor del operando, y sustituirlo en el código maquina. Si es necesario debemos rellenar con 0's los caracteres que hagan falta.

Más adelante se muestra un ejemplo donde la línea de instrucción "com \$300,sp" tiene el código maquina 61 xb ee ff, donde xb da como resultado f2 y ee ff lo obtenemos agregándole el valor del operando y en vista de que el operando tiene 3 dígitos y debemos sustituir 4 caracteres del código maquina, se le agrega un cero a la izquierda, dando como resultado 61 f2 03 00.

EJEMPLO DE ARCHIVO .ASM



```
IND2 - Bloc de notas
Archivo Edición Formato Ver Ayuda

ORG      $0E00
cpd      3,-sp
cpd      2,sp-
cpd      4,+sp
com      $300,sp
cpd      2,sp+
cmpa     [D,y]
cpd      -3,x
cpx      12,x
cpy      32,y
cmpa     [$200,x]
cpy      -45|,y
END

Línea 1, columna 1
```

EJEMPLO DE IND2.LST GENERADO POR EL IND2.ASM

