

Practica 6.- “Modos de direccionamiento simple”.

2.2.1 Modos de direccionamiento simples.

OBJETIVO DEL SUBTEMA. *Modificar el programa que se diseñó en el punto 2.1.4. El programa actualizado además, deberá generar el código máquina de los direccionamientos simples.*

Direccionamientos simples. (Inherente, Inmediato, Directo y Extendido)

Direccionamiento inherente (INH)

Las instrucciones de este tipo no tienen una dirección efectiva porque no guardan datos en una dirección de memoria, solo usan la memoria para obtener el código maquina.

Existen 2 tipos de direccionamientos inherentes, los implícitos que no requieren operádos:

CONLOC cop **<etiqueta>** **INST** ; datos en la instrucción

COP es el código de operación de la instrucción que lo documenta el fabricante el conjunto de instrucciones de las hojas de datos del microprocesador o Microcontrolador (en este caso archivo CPU12RG.pdf).

Y CONLOC es la dirección de memoria que le corresponde a la instrucción.

Y los inherentes por registro, que si llevan operando pero hacen referencia únicamente a registros internos, en otras palabras no ocupan de la memoria:

CONLOC cop eb <etiqueta> **INST** r1,r2 ; por registro

En donde r1 y r2 significan registro 1 y 2 (respectivamente); un ejemplo de esto podría ser la siguiente instrucción en lenguaje ensamblador:

tfr **A,B**

En la cual se indica que se copie lo que contiene el acumulador A en el acumulador B.

Otro ejemplo podría ser el siguiente:

exg **X,S**

El cual dice que se cambie lo que contenga el registro X en el registro S.

Para esta práctica solo se consideraran los que no llevan operandos, además que las instrucciones que en el manual estén marcadas como de otros direccionamientos pero no tengan operandos se podrán considerar como de este modo de direccionamiento.

Direccinamiento inmediato (IMM)

Se usa cuando se especifica que se quiere asignar específicamente un valor, el cual puede ser tanto de 8 bits como de 16 bits. La EA se encuentra inmediatamente después

del código de operación.

CONLOC	cop ii	<etiqueta>	INST	#opr8i
CONLOC	cop jj kk	<etiqueta>	INST	#opr16i

Con $0 \leq \text{opr8i} \leq 255$ para el de 8 bits, y $0 \leq \text{opr16i} \leq 65535$ para el de 16 bits. Donde ii es igual a opr8i en formato hexadecimal en 1 byte (completando con cero a su izquierda de ser necesario), y jj kk es opr16i en formato hexadecimal completado a 2 bytes.

Por ejemplo, para el siguiente fragmento de programa que se encarga de sumar el número 30 hexadecimal con el 20 también en hexadecimal, el código maquina se calcularía de la siguiente manera:

<i>0000 87</i>	clra	;INH, no existe EA
<i>0000 86 30</i>	ldaa #\$30	;IMM, EA = 0001H
<i>0002 8B 20</i>	adda #\$20	;IMM, EA= 0003H
<i>0004 CE 00 00</i>	ldx #\$00	;IMM, EA= 0005H

Direccionamiento directo (DIR)

El operando indica la dirección de memoria de donde se leerá el dato, pero la dirección debe estar en la pagina cero (entre 0000H y 00FFH). La EA = 00dd.

CONLOC	COP dd	<etiqueta>	INST opr8a
---------------	---------------	-------------------------	-------------------

Donde $0 \leq \text{opr8a} \leq 255$. Con dd=opr8a en formato hexadecimal de 1 byte. Suponiendo que deseamos conocer el código maquina del siguiente extracto de programa que se encarga de sumar 2 números, tenemos:

<i>0000 87</i>	clra	; EA no existe
<i>0001 96 80</i>	ldaa \$80	; EA= 0080H
<i>0003 9B 81</i>	adda \$81	; EA=0081H
<i>0005 5A 82</i>	staa \$82	; EA=0082H

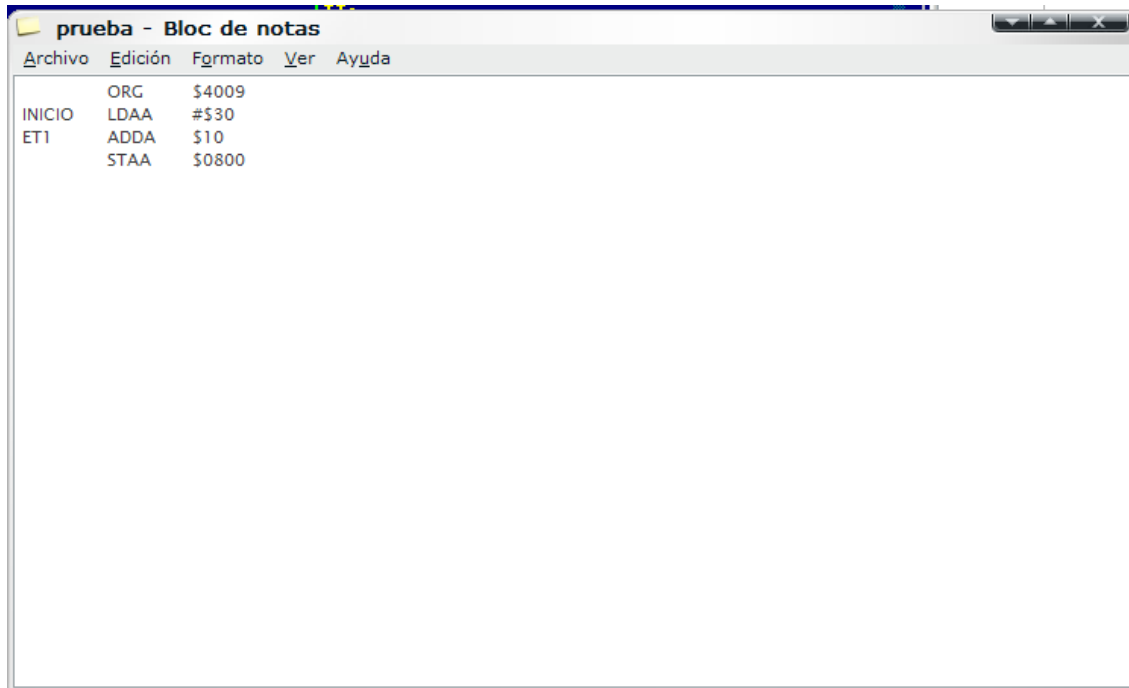
Direccionamiento extendido (EXT)

Igual que en el direccionamiento directo el operando especifica la dirección de memoria, con la diferencia de que se puede especificar cualquier dirección de memoria. La EA=hhl.

<i>0000 79 00 10</i>	clr	\$10
<i>0001 B6 08 00</i>	ldaa	\$800
<i>0003 BB 08 01</i>	adda	\$801
<i>0005 7A 08 20</i>	staa	\$820

Desarrollo:

En esta practica se tendrán que calcular los códigos maquinas de los direccionamientos simples (directo, inmediato, extendido) estos se generaran como parte del segundo paso del ensamblador donde partiremos del archivo Ist que teníamos previamente en donde los códigos maquina estaban representados por "ii", "jj kk", "dd", "hh ll", lo que tendrá que hacer el programa será cambiar estas literales por los operandos de la instrucción.



```
prueba - Bloc de notas
Archivo  Edición  Formato  Ver  Ayuda
ORG      $4009
INICIO   LDAA    #$30
ET1      ADDA    $10
         STAA    $0800
```

Comenzaremos por abrir el archivo en donde se encuentra nuestro código fuente con las instrucciones que deseamos ejecutar.



```
TC.EXE
dame el archivo que deseas abrir:
prueba.asm_
```

El programa deberá de validar las instrucciones y colocar el código maquina correspondiente para esto utilizaremos el archivo lst anterior, lo abrimos y leemos la primera cadena del archivo, también tendremos que leer por segunda vez el archivo en donde se encuentra el código fuente para extraer el valor de los operandos

```

TC.EXE
dame el archivo que deseas abrir:
prueba.asm
etiqueta:NULL
instruccion:ORG
operando [0]:$4009
línea original: ORG      $4009
ORG      DIRECTIVA      *0

numero de bytes:0
CONLOC:0
*****
conloc= 4009etiqueta:INICIO
instruccion:LDA#
operando [0]:#$30
línea original:INICIO   LDA#   #$30
LDA#      IMM8      86 ii   *2

numero de bytes:2
CONLOC:4009
*****
etiqueta:ET1
instruccion:ADDA
operando [0]:$10
línea original:ET1      ADDA   $10
ADDA      DIR      9B dd   *2

numero de bytes:2
CONLOC:400b
*****
etiqueta:NULL
instruccion:STAA
operando [0]:$0800
línea original:STAA     $0800
STAA      EXT      7A hh ll   *3

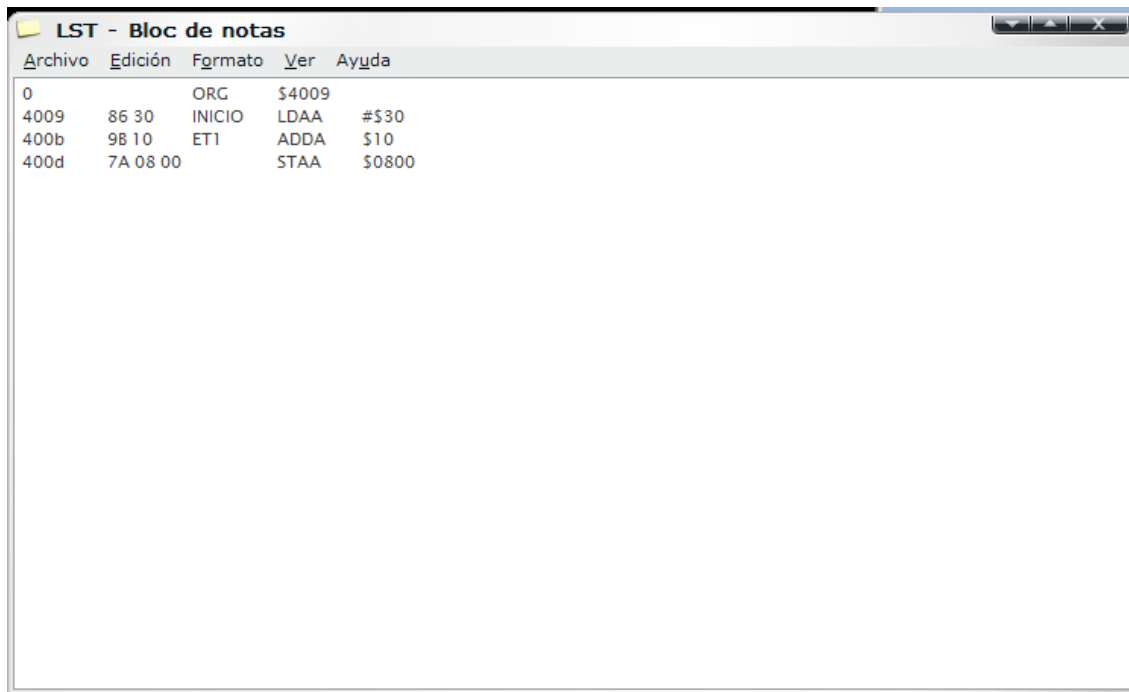
numero de bytes:3
CONLOC:400d
*****
5
2
INMEDIATO4009  86 30  INICIO  LDA#   #$30
4
DIRECTO400b   9B 10  ET1    ADDA   $10
5
EXTENDIDO400d 7A 08 00          STAA   $0800
hecho por HESH

```

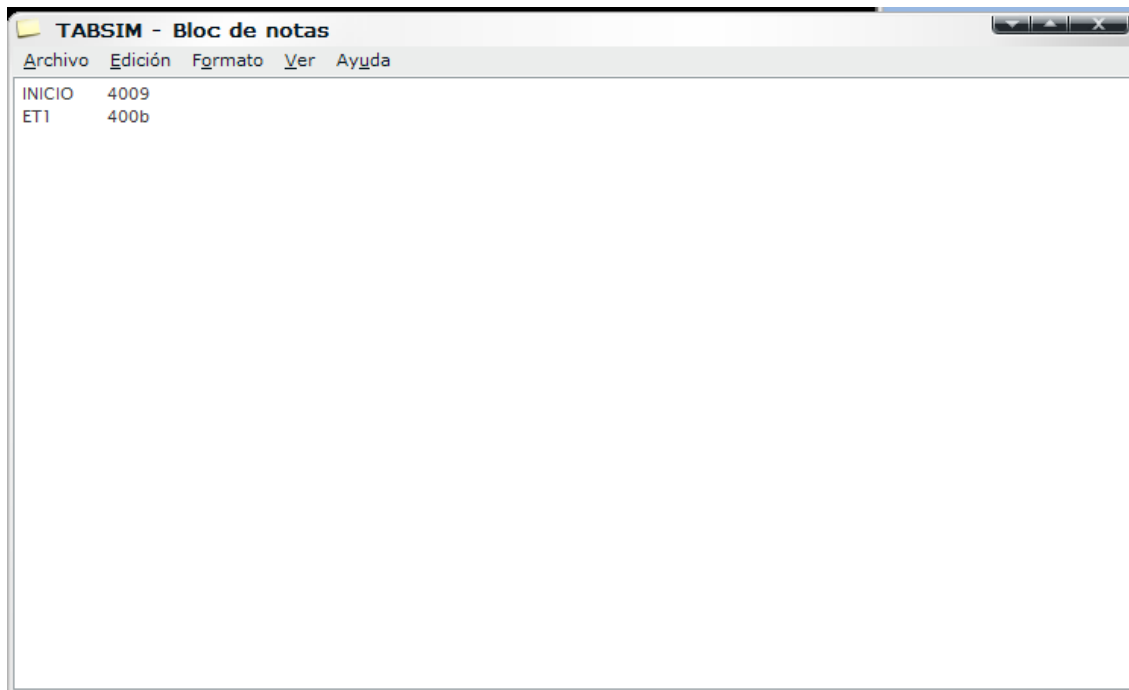
Una vez que tengamos los operandos tendremos que validar a que modo de direccionamiento pertenece y el tamaño del operando para a si poder intercambiar las literales por el valor del operando de dicha instrucción.

Una vez que hayamos sobrescrito en la instrucción el valor del operando abriremos un archivo temporal en el cual iremos escribiendo las instrucciones ya completas con su código maquina, las iremos escribiendo una por una en el temporal al mismo tiempo en el que se inserta su código maquina en el mismo orden, al encontrar el final del archivo cerraremos el temporal y el lst intermedio.

Después solo borraremos el archivo lst que teníamos sin los códigos maquina y renombraremos el temporal en el cual fuimos escribiendo las líneas con las instrucciones y el código maquina completo.



El nuevo archivo lst tendrá que tener el mismo formato que el anterior pero con los códigos maquina correspondientes a cada instrucción.



No olvidaremos que en tabsim tendrán que estar todas las etiquetas que se hayan encontrado en el programa puesto que después podrían ser útiles.