PRACTICA 3: “Búsqueda de instrucciones”

2.1.2 Instrucciones y el conjunto de instrucciones.

OBJETIVO DEL SUBTEMA.

Modificar el programa que se diseñó en el punto 2.1.1. El programa actualizado, además, deberá de realizar la búsqueda de cada instrucción identificada por cada línea y determinar cuales modos de direccionamiento le corresponde a cada una de ellas de acuerdo a la tabla de códigos de operación.

INTRODUCCIÓN

El conjunto de instrucciones que puede entender o ejecutar un Microprocesador o microcontrolador es exclusivo de ese dispositivo, ya que es el fabricante el que se encarga de definir sus instrucciones de acuerdo al diseño del hardware para poder implementar su circuito, así el conjunto de instrucciones en lenguaje ensamblador no sera compatible con ningún otro micro.

Es el mismo fabricante el encargado de especificar en sus hojas de datos el conjunto de instrucciones de sus dispositivos. De tal forma que un programa diseñado en lenguaje ensamblador para un microprpcesador especifico, debera revisarse cuando se desee ejecutar el mismo programa en otro dispositivo y lo mas probable es que el programa deberá cambiarse.

Para programar en ensamblador es necesario conocer las instrucciones con las que podemos realizar nuestras tareas. Estas instrucciones se llaman mnemónicos y son definidas por el fabricante, para conocerlas debemos consultar el conjunto de instrucciones en el manual del microcontrolador 68HC12 de freescale.

Cada arquitectura de computadoras tiene su propio lenguaje de máquina, y en consecuencia su propio lenguaje ensamblador. Los ordenadores difieren en el tipo y número de operaciones que soportan; también pueden tener diferente cantidad de registros, y distinta representación de los tipos de datos en memoria. Aunque la mayoría de las computadoras son capaces de cumplir esencialmente las mismas funciones, la forma en que lo hacen difiere y los respectivos lenguajes ensamblador reflejan tal diferencia.

Pueden existir múltiples conjuntos de mnemónicos o sintaxis de lenguaje ensamblador para un mismo conjunto de instrucciones, instanciados típicamente en diferentes programas ensamblador. En estos casos, la alternativa más popular es la provista por los fabricantes, y usada en los manuales del programa.

El código máquina, o lenguaje de máquina, está formado por instrucciones sencillas, que -dependiendo de la estructura del procesador- pueden especificar:

- Registros específicos para operaciones aritméticas, direccionamiento o control de funciones.
- Posiciones de memoria específicas (*offset*).
- Modos de direccionamiento usados para interpretar operandos.

Las operaciones más complejas se realizan combinando estas instrucciones sencillas, que pueden ser ejecutadas secuencialmente o mediante instrucciones de control de flujo.
Las operaciones disponibles en la mayoría de los conjuntos de instrucciones incluye:

- mover
  - llenar un registro con un valor constante
  - mover datos de una posición de memoria a un registro o viceversa
  - escribir y leer datos de dispositivos
- computar
  - sumar, restar, multiplicar o dividir los valores de dos registros, colocando el resultado en uno de ellos o en otro registro
  - realizar operaciones binarias, incluyendo operaciones lógicas (AND/OR/XOR/NOT)
  - comparar valores entre registros (mayor, menor, igual)
- afectar el flujo del programa
  - saltar a otra posición en el programa y ejecutar instrucciones allí
  - saltar si se cumplen ciertas condiciones (IF)
  - saltar a otra posición, pero guardar el punto de salida para retornar (CALL, llamada a subrutinas)

Algunas computadoras incluyen instrucciones complejas dentro de sus capacidades. Una sola instrucción compleja hace lo mismo que en otras computadoras puede requerir una larga serie de instrucciones, por ejemplo:

- salvar varios registros en la pila de una sola vez
- mover grandes bloques de memoria
- operaciones aritméticas complejas o de punto flotante (seno, coseno, raíz cuadrada)

El nivel de lenguaje ensamblador tiene aspectos importantes de los niveles de microarquitectura, en los cuales se encuentra (ISA y sistema operativo) estos dos se utilizan para la traducción en lugar de la interpretación. Algunas características del lenguaje se describen a continuación Los programas que sirven para traducir algún programa para el usuario se llama traductores, el lenguaje en que esta escrito el programa original se llama lenguaje fuente, el lenguaje original que sea modificado se llama lenguaje objeto.
Se usa la traducción cuando se cuenta con un procesador (ya sea hardware o un interprete) para el lenguaje objeto pero no para el lenguaje fuente, Si la traducción se realiza correctamente, la ejecución del programa traducido dará exactamente los mismos resultados que habría dado la ejecución del programa fuente. Hay dos diferencias entre traducción e interpretación, en la traducción no se ejecuta directamente el programa original, en el lenguaje fuente se convierte en un programa equivalente llamado programa objeto o programa binario ejecutable y este funciona solo cuando se ha acabado la traducción.
El código máquina, un simple patrón de bits, es hecho legible reemplazando *valores crudos* por símbolos denominados mnemónicos. Se inventó para facilitar la tarea de los primeros programadores que hasta ese momento tenían que escribir directamente en código binario. Inicialmente el código de ceros y unos (el programa) debía introducirse en una tarjeta perforada. La posición ocupada por cada punto equivalía a un "1" o a un "0" según hubiera o no una perforación. Lo cual suponía una forma casi idéntica en la que hoy se escriben los datos binaros en soportes tales como los CDs y DVDs.
Mientras que una computadora reconoce la instrucción de máquina 8661H
10000110 01100001

para los programadores de microprocesadores HC12 es mucho más fácil reconocer dicha instrucción empleando lenguaje ensamblador :
ldaa #$61
Cada instrucción de la máquina se transforma en una única instrucción en código simbólico.
Pero además, para mejorar la legibilidad del programa, el código simbólico introduce instrucciones adicionales, que no corresponden a ninguna instrucción de la máquina y que proporcionan información. Se llaman "pseudoinstrucciones".
El código simbólico puede parecer de difícil acceso, pero es más fácil de recordar e interpretar que el binario o el hexadecimal.
Los lenguajes simbólicos no resuelven definitivamente el problema de cómo programar un ordenador de la manera más sencilla posible. Para utilizarlos, hay que conocer a fondo el microprocesador, los registros de trabajo de que dispone, la estructura de la memoria, y muchas cosas más.
Además, el lenguaje ensamblador está muy ligado al microprocesador para que sea posible escribir programas independientes de la máquina en que van a ejecutarse.
Este **código simbólico** no puede ser ejecutado directamente por un ordenador, por lo que es preciso traducirlo previamente. Pero la traducción es un proceso mecánico y repetitivo, que se presta a su realización por un programa de ordenador.
Los programas que traducen código simbólico al lenguaje de máquina se llaman ensambladores ("assembler", en inglés), porque son capaces de ensamblar el programa traducido a partir de varias piezas, procedimientos o subrutinas a código binario ("1" y "0") que entiende el procesador.

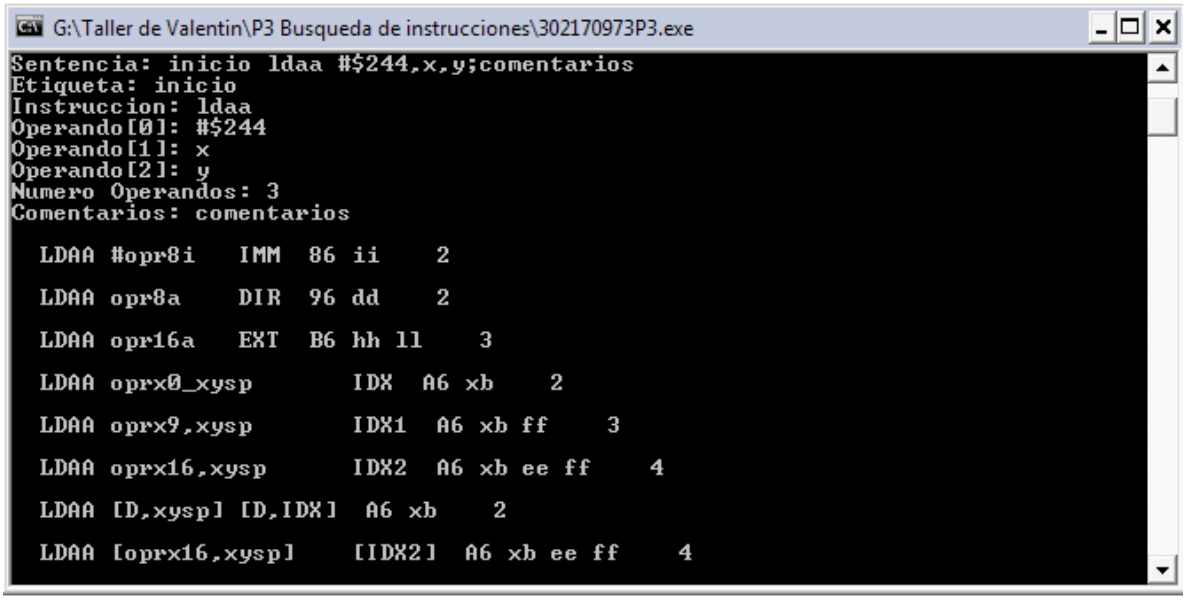
DESARROLLO

Conforme a la practica numero dos se desarrollo esta práctica en donde conforme a una instrucción se desplegaran en pantalla los tipos de formato para dicha instrucción.

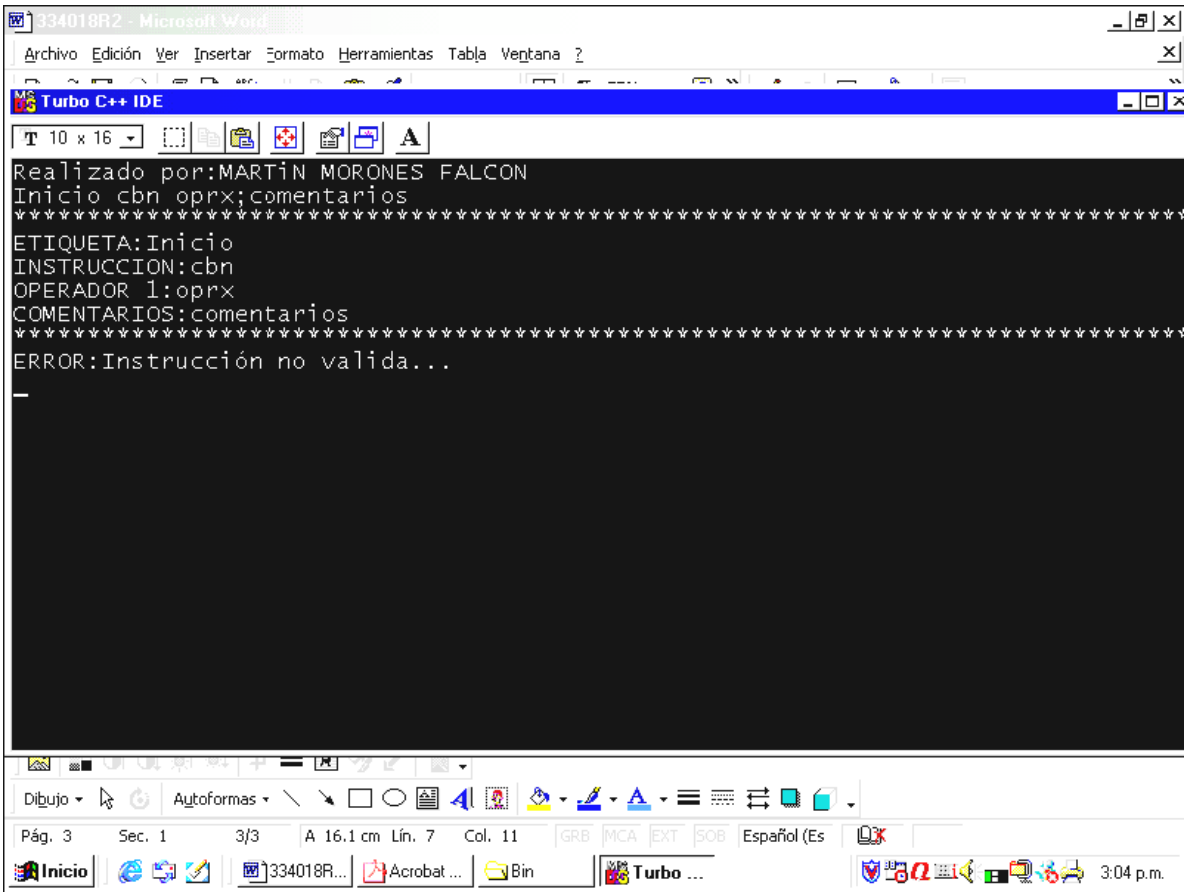
abc //indica q puede usarse A, B o CCR
abcdxys //indica q puede usarse A, B, CCR, D, X, Y o el SP
abd //indica q puede usarse A, B o D
abdxys //indica q puede usarse A, B, D, X, Y, o SP
dxys //indica q puede usarse D, X, Y o SP
msk8 //es un valor enmascarado de 8bits
opr8i //es un valor de 8 bits de direccionamiento inmediato
opr16i //es un valor de 16bits de direccionamiento inmediato
opr8a //es un valor de 8 bits usdo en direccionamiento directo
opr16a //es un valor de direccion de 16bits
oprx0_xysp // es usado en direccionamiento indexado
oprx3 //cualquier valor entero positivo de 1 a 8
oprx5 //cualquier valor entero de rango -16 a +15
oprx9 //cualquier valor entero positivo de -256 a 255
oprx16 //cualquier valor entero positivo de -32768 a 65535
rel8 //etiqueta de localidad de desplazamiento de -256 a 255 localidades
rel9 //etiqueta de localidad de desplazamiento de -512 a 511 localidades
rel16 //cualquier etiqueta de 64k de espacio de memoria

trapnum //cualquier entero en el rango de $30-$39 o $40-$FF
xys //indica q puede ser X o Y o SP
xysp //indica q puede ser X o Y o SP

Para esto se capturaron todas las instrucciones del hc12 en un archivo de texto llamado TABOP aquí están los formatos para cada instrucción de esta manera como solo se compara lo que tenemos en instrucción a TABOP, si se encuentra despliega sus formatos si no los encuentra busca si es una directiva de otro modo la instrucción se marca como no valida. En sí lo que vamos a hacer es desarrollar un programa en un lenguaje de programación de alto nivel. En el que nosotros introduzcamos una cadena donde escribamos una etiqueta, una instrucción, uno o más operandos y si se desea también comentarios. El programa nos va a desglosar un pequeño menú por así llamarlo donde nos va a mostrar los formatos que puede adoptar dicha instrucción y si no es válida se tiene que decir al usuario que es incorrecta.
Vamos a trabajar con 2 archivos ya que uno va a ser donde va a residir el programa que estamos haciendo y el otro que va a ser de texto en el que vamos a tener la tabla de instrucciones del microprocesador con el que estamos trabajando (TABOP). Aquí mostramos un ejemplo de lo que debe realizar el programa:



En esta pantalla se muestra el resultado cuando si existe la instrucción que se introdujo.



En esta otra pantalla se muestra el resultado de cuando la instrucción introducida no existe.