**Universidad de Guadalajara**

**Centro Universitario de Ciencias Exactas e Ingenierías**

**Taller de programación de sistemas**

**Reporte # 8**

Romero Gastelu, María Elena

**NRC**: 02316

**Alumno:**

Gómez Tovar Edgar Iván

**Código:**

303526879

24-11-2008

**Breve descripción de las funciones y variables utilizadas**

Básicamente en esta práctica utilice las mismas funciones auxiliares que en las anteriores, son las siguientes:

* **Función:** dec\_bin (int numero, int completar)

Esta función se encarga de convertir un número entero decimal a su equivalente binario en forma de cadena de caracteres y también rellena el número con una cantidad especificada de ceros a su izquierda (si es necesario). Recibe 2 valores enteros como argumentos, numero (la cantidad a convertir) y completar (la cantidad de bits que se requiere tenga la cadena y que podrían ser completados con ceros). Utiliza el método de las divisiones sucesivas en el cual se divide el numero por 2, si esta división es exacta (modulo igual a 0) agrega un 0 a la cadena final y viceversa. Además recibe como segundo argumento una cantidad, por ejemplo 8, con lo cual (y solo si es necesario) va a completar con ceros a la izquierda para cumplir el formato requerido (8 bits en este caso). Si el número es negativo, luego de calcular su representación positiva invierte los bits (complemento a la base) y le suma 1 (complemento a 2).

La función cuenta con las siguientes variables:

* **Enteras:** i, j, negativo.
* **Carácter:** binario [20].
* **Función:** dec\_hex (int numero, int completar)

Esta función se encarga de convertir un número entero decimal a su equivalente hexadecimal en forma de cadena de caracteres y también rellena el número con una cantidad especificada de ceros a su izquierda (si es necesario). Recibe 2 valores enteros como argumentos, numero (la cantidad a convertir) y completar (la cantidad de bytes que se requiere tenga la cadena y que podrían ser completados con ceros). Utiliza el mismo principio que la función anterior para convertir el número, pero ahora ajustándose a lo permitido por esta base numérica. Además recibe como segundo argumento una cantidad, por ejemplo 2, con lo cual (y solo si es necesario) va a completar con ceros a la izquierda para cumplir el formato requerido (2 bytes en este caso).

La función cuenta con las siguientes variables:

* **Enteras:** i, j, k, bandera, todavia.
* **Carácter:** aux\_num\_hex [20], num\_hex [20].
* **Función:** divide (char cadena[ ])

La función recibe una cadena de caracteres que representa un número binario de 8 bits, esta cadena es separada en 2 partes a las cuales a su vez se les calcula su valor numérico decimal; este valor decimal se manda como argumento a la función dec\_hex para que sea convertido en su representación numérica hexadecimal en forma de cadena de caracteres, estos 2 valores son luego concatenados para dejar una cadena con dígitos hexadecimales y con un formato de 2 bytes.

La función cuenta con las siguientes variables:

* **Enteras:** i, j.
* **Carácter:** parte1 [5], parte2 [5], xb [3].

Además, fuera de las funciones auxiliares, utilice:

* **Enteras:** et\_encontrada, desplazamiento.
* **Carácter:** CONTLOC\_T [5], etiqueta\_t [10].

Y la estructura que use en cada línea del archivo TABSIM.txt es la siguiente (necesaria para entender la descripción de la búsqueda en el mismo):

CONTLOC (ETIQUETA RELATIVA) CONTLOC ETIQUETA

O bien:

EQU (ETIQUETA ABSOLUTA) CONTLOC ETIQUETA

**Descripción de la forma de consulta de datos en el TABSIM.txt y cálculo del desplazamiento**

Al comenzar guardamos la posición del cursor en el archivo actual, cerramos el archivo y abrimos el TABSIM.txt, mandamos el cursor al comienzo del archivo y comienza la búsqueda de la etiqueta. Se le asigna un false a la variable et\_encontrada y se entra en un ciclo que se estará repitiendo mientras que no se llegue al final del archivo y la variable et\_encontrada tenga un false, dentro del ciclo primero se lee carácter por carácter hasta que se encuentra un tabulador (para que se recorra en la descripción de la línea actual del TABSIM.txt, por ejemplo “CONTLOC (ETIQUETA RELATIVA)”), ya que después del primer tabulador se encuentra el valor del CONTLOC comenzamos a guardar cada carácter leído en la variable CONTLOC\_T hasta que nos encontramos con un segundo tabulador, luego de encontrar este segundo tabulador se encuentra la etiqueta por lo que procedemos a guardar cada carácter leído en la variable etiqueta\_t; ahora comparamos etiqueta\_t con etiqueta y si son iguales et\_encontrada recibirá un true con lo que se saldrá del ciclo.

Al terminar la búsqueda en el TABSIM.txt si et\_encontrada tiene un false se indica en pantalla que tal etiqueta no se encontró en el archivo, se cierra el mismo, se abre el otro archivo y se manda el cursor hasta la posición que guardamos anteriormente. Si por el contrario et\_encontrada tiene un true tomamos el valor numérico decimal del operando (que será el valor que contiene CONTLOC\_T) y le restamos el valor del CONTLOC, el resultado se guarda en la variable desplazamiento; ahora verificamos el valor de dicha variable, si es mayor o igual a cero verificamos el modo de direccionamiento, si es un relativo de 8 bits verificamos el valor del desplazamiento y si excede a 127 imprimimos un error en pantalla, pero si es igual o menor a 127 le mandamos a la función dec\_hex el valor del desplazamiento y un 1, para que de ser necesario complete con un cero el valor en hexadecimal del desplazamiento, el valor obtenido por dicha función se concatena con el código maquina del codop y se imprime en pantalla. Si en vez de ser un relativo de 8 bits es de 16 bits verificamos el valor del desplazamiento y si excede a 32767 imprimimos un error en pantalla, pero si es igual o menor a 32767 le mandamos a la función dec\_hex el valor del desplazamiento y un 2, para que de ser necesario complete con ceros a la izquierda el valor en hexadecimal del desplazamiento, el valor obtenido por dicha función se concatena con el código maquina del codop y se imprime en pantalla.

Si cuando revisamos el valor del desplazamiento este fue menor a cero verificamos el modo de direccionamiento, si es un relativo de 8 bits verificamos el valor del desplazamiento y si es menor a -128 imprimimos un error en pantalla, pero si es igual o mayor a -128 le mandamos a la función dec\_bin el valor del desplazamiento y un 8, para que de ser necesario complete con ceros a la izquierda el valor binario (en 8 bits con complemento a 2) del desplazamiento, el valor obtenido por dicha función se manda a la función divide y el valor devuelto por dicha función se concatena con el código maquina del codop y se imprime en pantalla. Si en vez de ser un relativo de 8 bits es de 16 bits verificamos el valor del desplazamiento y si es menor a -32768 imprimimos un error en pantalla, pero si es igual o mayor a -32768 le mandamos a la función dec\_bin el valor del desplazamiento y un 16, para que de ser necesario complete con ceros a la izquierda el valor binario (en 16 bits con complemento a 2) del desplazamiento, el valor obtenido por dicha función se parte en 2 (ya que la función divide solo divide cadenas de longitud 8), cada mitad se manda por separado a la función divide y los valores obtenidos por dicha función se concatenan con el código maquina del codop y se imprime en pantalla.

Por último se cierra el archivo, se abre el otro y se manda el cursor hasta la posición que guardamos anteriormente.