**Universidad de Guadalajara**

**Centro Universitario de Ciencias Exactas e Ingenierías**

**Taller de programación de sistemas**

**Reporte # 6**

Romero Gastelu, María Elena

**NRC**: 02316

**Alumno:**

Gómez Tovar Edgar Iván

**Código:**

303526879

10-11-2008

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

* **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].

**Descripción de los algoritmos utilizados para obtener los post bytes xb de cada uno de los modos indizados**

Al comenzar se entra en un ciclo que se estará repitiendo mientras que no se encuentre el fin de cadena, en este ciclo se extraerá el registro al que hace referencia el operando (x, y, sp, pc); si este registro es “x” entonces se copea la cadena “00” en el arreglo rr, si por el contrario fuera “y” se copea la cadena “01” en el arreglo mencionado, “10” para “sp” y por ultimo 11 para “pc”. Una vez hecho esto se extrae el valor numérico que antecede a la coma del operando, y se almacena en una variable entera temporal; en este punto se tienen todos los elementos para aplicar las distintas formulas para obtener lo diferentes códigos maquina, por lo cual se verifica que tipo de operando es (y que en previas practicas se había determinado), por lo que aquí hay varios caminos a recorrer:

1. Si se trata de un modo indizado de 5 bits tenemos que deducir el post byte xb con la siguiente formula rr0nnnnn, como ya calculamos rr solo queda calcular nnnnn para lo cual hacemos uso del valor decimal que también calculamos anteriormente, por lo que ese valor se lo mandamos a la función dec\_bin indicándole que de ser necesario complete hasta 5 bits (para formar nnnnn); ahora concatenamos la cadena rr, “0” y nnnnn y el resultado (una cadena binaria de 8 bits) se lo mandamos a la función divide, esta función nos devolverá el valor en 2 bytes hexadecimales de la cadena binaria por lo que solo resta concatenar el código maquina calculado por el TABOP.txt para el codop, con la cadena devuelta por la función divide e imprimir en pantalla.
2. Si se trata de un modo indizado de 9 bits tenemos que deducir el post byte xb con la siguiente formula 111rr0zs, como ya calculamos rr solo queda calcular s (bit de signo); haciendo uso del valor decimal que calculamos anteriormente podemos verificar si este mismo es negativo, en cuyo caso su representación binaria tendría el bit s con un 1 y viceversa. Concatenamos la cadena “111”, rr, “00” (z es igual a 0 en los indizados de 9 bits) y s (recordemos que si el numero calculado anteriormente fue negativo este bit será 1, y viceversa) y el resultado (una cadena binaria de 8 bits) se lo mandamos a la función divide, esta función nos devolverá el valor en 2 bytes hexadecimales de la cadena binaria por lo que concatenamos el código maquina calculado por el TABOP.txt para el codop, con la cadena devuelta por la función divide y le agregamos la representación hexadecimal en 2 bytes del numero anteriormente calculado (esto será el post byte ff) e imprimimos en pantalla.
3. Si se trata de un modo indizado de 16 bits tenemos que deducir el post byte xb con la misma fórmula que usamos en el modo anterior pero ahora con el bit z en 1 y el bit s en 0, como ya calculamos rr solo queda concatenar la cadena “111”, rr, “010” y el resultado (una cadena binaria de 8 bits) se lo mandamos a la función divide, esta función nos devolverá el valor en 2 bytes hexadecimales de la cadena binaria por lo que concatenamos el código maquina calculado por el TABOP.txt para el codop, con la cadena devuelta por la función divide y le agregamos la representación hexadecimal en 4 bytes del numero anteriormente calculado (estos serán los post bytes ee y ff) e imprimimos en pantalla.
4. Si se trata de algún modo de pre/post incremento/decremento tenemos que deducir el post byte xb con la fórmula rr1pnnnn, como ya calculamos rr queda calcular el bit p y nnnn, para el primero solo verificamos si es un modo de pre incremento/decremento lo que lo dejaría en 0 y viceversa y para deducir nnnn tomamos el valor decimal obtenido anteriormente y se lo mandamos a la función dec\_bin indicándole que de ser necesario complete a 4 bits, después concatenamos la cadena rr, “1” y nnnn, el resultado (una cadena binaria de 8 bits) se lo mandamos a la función divide, esta función nos devolverá el valor en 2 bytes hexadecimales de la cadena binaria por lo que concatenamos el código maquina calculado por el TABOP.txt para el codop, con la cadena devuelta por la función divide e imprimimos en pantalla.
5. Por último si se trata del modo indizado de acumulador tenemos que deducir el post byte xb con la fórmula 111rr1aa, como ya calculamos rr queda calcular aa, esto lo obtendremos revisando la primer posición del operando (a=00, b=01, d=10), después concatenamos “111”, rr, “1” y aa, el resultado (una cadena binaria de 8 bits) se lo mandamos a la función divide, esta función nos devolverá el valor en 2 bytes hexadecimales de la cadena binaria por lo que concatenamos el código maquina calculado por el TABOP.txt para el codop, con la cadena devuelta por la función divide e imprimimos en pantalla.