**Universidad de Guadalajara**

**Centro Universitario de Ciencias Exactas e Ingenierías**

**Taller de programación de sistemas**

**Reporte # 3**

**Romero Gastelu, María Elena**

**NRC: 02316**

**ALUMNO:**

GÓMEZ TOVAR EDGAR IVÁN

CODIGO: 303526879

20-10-2008

**Descripción de las funciones y variables utilizadas**

* **Función:** Id\_operando ()

Esta función es accesada directamente al recibir un código de operación valido y se encarga de hacer todas las validaciones necesarias a los operandos, desde la de las distintas bases numéricas, hasta la de las reglas de los distintos modos de direccionamiento. Al intentar validar un operando, si este tiene valores numéricos se llama a la función extraer () para verificar el valor del mismo (hay que recordar que al ser archivos de texto no pueden utilizarse directamente los valores ya que son caracteres) y de esta forma conocer el rango del modo de direccionamiento. Una vez que se ha validado correctamente un operando se llama a la función busqueda\_tabop ().

La función cuenta con las siguientes variables:

* **Enteras:** incorrecto, i, aux, aux2, aux3, senal, senal\_auto, signo, coma.

Las variables incorrecto, aux, aux2, aux3, senal, senal\_auto, signo y coma actúan como banderas. La variable i es un contador.

* **Carácter:** letra.

Letra guarda caracteres para imprimirlos en pantalla cuando ocurra un error.

* **Función:** extraer ()

Esta función es llamada dentro de la función id\_operando () si se necesita conocer algún valor numérico dentro de los operandos, y solo es llamada si el operando ha sido validado en sus demás características anteriormente por la función ya mencionada, por lo que esta función no realiza ningún tipo de validación ya que presupone que el operando que recibe esta correcto en su sintaxis.

Ya que se están trabajando con archivos de texto no pueden utilizarse directamente los valores de la variable operando (si es que el operando representa un valor, como por ejemplo en un modo de direccionamiento directo), por lo que la función toma este valor en forma de cadena, y en cualquier base (hexadecimal, octal, binaria, etc.) y lo transforma a su representación numérica en decimal valiéndose de los códigos ASCII de los distintos dígitos; por ejemplo si tenemos la cadena $15F guardada en forma de operando, obtenemos su representación numérica restándole primero 55 al valor ASCII de la letra F y esto multiplicándolo por la base numérica en la posición del digito (en este caso 16 a la 0), lo mismo aplicaría para los demás dígitos pero en vez de restarle un 55 se les restaría un 48 y todo se sumaría para obtener el total que es devuelto al lugar donde se llamo a la función.

La función cuenta con las siguientes variables:

* **Enteras:** longitud, base, subindice, listo, i, resultado, signo.
* **Carácter:** operando\_aux[50].

Además modifique la siguiente función que había usado con anterioridad en la segunda práctica:

* **Función:** busqueda\_tabop ()

Esta función, al igual que en la práctica pasada, se encarga de hacer una búsqueda en el TABOP.txt para verificar si el código de operación que se está analizando se encuentra en el ya mencionado TABOP.txt, el cambio hecho a esta función es que ahora ya no muestra en pantalla los distintos valores que acompañaban a cada código de operación que concordaba con el que se tenía previamente, ahora únicamente se encarga de ir comparando cada vez el código de operación almacenado en el TABOP.txt con el que uno está utilizando y si se encuentra concordancia se analiza el modo de direccionamiento asociado a ese código de operación especifico y si es igual al modo de direccionamiento que dedujo la función id\_operando () entonces verifica también la cantidad de bytes totales del operando para imprimir la información en pantalla. Para estas operaciones utiliza la variable modos, que es modificada en la función id\_operando () al encontrarse validado correctamente un operando, estos valores pueden ir del 1 al 17 y cada uno representa un modo de direccionamiento distinto, que como mencione anteriormente, al compararse con los valores del TABOP.txt y ser idénticos se imprime en pantalla así como su cantidad de bytes. Si por ejemplo, la función id\_operando () analiza un operando y deduce que se trata de un modo de direccionamiento extendido (por dar un ejemplo) manda a llamar a esta función previamente asignándole un 5 a la variable modos, si esta función al buscar en el TABOP.txt encuentra que el código de operación si se encuentra y que además acepta un modo de direccionamiento extendido entonces imprime en pantalla que tal operando es de modo extendido y que conlleva tantos bytes, si por el contrario la función busca el código de operación en el TABOP.txt pero se encuentra con que este no acepta el modo extendido se imprime en pantalla un error acerca de que ese código de operación no admite ese modo de direccionamiento especifico.

**Descripción acerca de la validación de los modos de direccionamiento y las bases numéricas**

Al llamarse a la función id\_operando () lo primero que se hace es asignarle un 0 a la variable modos así como también a la variable incorrecto, luego se compara el valor del operando con null, si son iguales se le asigna un 1 a la variable modos y se manda a llamar a la función busqueda\_tabop () que al corroborar que el operando existe en el TABOP y que ese operando tiene el modo de direccionamiento inherente imprime en pantalla que se trata de un modo de direccionamiento inherente y la cantidad de bytes que se indiquen.

Si no es null hay varias posibilidades, puede que el operando en su primera posición sea “#”, “$”, “@”, “%”, “-“, “[“, 0… 9, a… z, A… Z, o inclusive una coma; dependiendo del valor que tenga esa primera posición será el tratamiento que se le dé al operando.

Si operando en su primera posición es # entonces existe la posibilidad de que se trate de un modo de direccionamiento inmediato (si su validación es correcta); en este caso se verifica su siguiente posición, si es un fin de cadena entonces se imprime en pantalla que ese es un formato invalido para los modos de direccionamiento inmediato y se le asigna un 1 a la variable incorrecto, sino es fin de cadena y es un “$” entonces se entra en un ciclo while que se estará ejecutando hasta que se encuentre el fin de cadena, dentro del ciclo si se encuentra un carácter diferente de 0… 9, a… f, A… F se le asigna un true a la variable incorrecto y se imprime en pantalla que se encontró un digito invalido para un numero de base 16, si por el contrario el segundo carácter del operando es un “@” se hace la misma rutina pero ahora verificando que todos los caracteres del operando sean 0… 7, lo mismo será con un “%” (validando solo 1 y 0) y también si el primer digito de la variable operando es 0… 9. Al terminar el ciclo correspondiente se verifica el valor de la variable incorrecto, si es un false (lo que indicaría que todos los dígitos de la variable operando pertenecen a su base numérica, por lo que es correcto, sin importar la base numérica) se llama a la función extraer () y si el valor que devuelve está entre 0 y 255 se le asigna un 2 a la variable modos (lo que indica que se trata de un modo de direccionamiento inmediato de 8 bits) y se manda a llamar a la función busqueda\_tabop (), si está entre 256 y 65535 se le asigna un 3 a la variable modos (direccionamiento inmediato de 16 bits) y se manda a llamar a la función busqueda\_tabop(), si el valor devuelto por extraer () no cae dentro de ninguno de esos rangos se imprime en pantalla eso mismo; sino, si incorrecto tiene un true solo sale de la función id\_operando ().

Si operando en su primera posición tiene un “$” existe la posibilidad de que se trate de un modo de direccionamiento directo o extendido (si su validación es correcta); entonces se analiza el siguiente carácter del operando y si es un fin de cadena se imprime en pantalla que ese es un formato invalido para los números de base 16 y se le asigna un 1 a la variable incorrecto, sino no es un fin de cadena se entra en un ciclo while que se estará ejecutando hasta que se encuentre el fin de cadena, dentro del ciclo si se encuentra un carácter diferente de 0… 9, a… f, A… F se le asigna un true a la variable incorrecto y se imprime en pantalla que se encontró un digito inválido para un número de base 16. Al terminar el ciclo correspondiente se verifica el valor de la variable incorrecto, si es un false se llama a la función extraer () y si el valor que devuelve está entre 0 y 255 se le asigna un 4 a la variable modos (lo que indica que se trata de un modo de direccionamiento directo) y se manda a llamar a la función busqueda\_tabop (), si está entre 256 y 65535 se le asigna un 5 a la variable modos (direccionamiento extendido) y se manda a llamar a la función busqueda\_tabop(), si el valor devuelto por extraer () no cae dentro de ninguno de esos rangos se imprime en pantalla que se salió de rango; sino, si incorrecto tiene un true solo sale de la función id\_operando ().

Si operando en su primera posición tiene un “@” existe la posibilidad de que se trate de un modo de direccionamiento directo o extendido; entonces se analiza el siguiente carácter del operando y si es un fin de cadena se imprime en pantalla que ese es un formato invalido para los números de base 8 y se le asigna un 1 a la variable incorrecto, sino no es un fin de cadena se entra en un ciclo while que se estará ejecutando hasta que se encuentre el fin de cadena, dentro del ciclo si se encuentra un carácter diferente de 0… 7 se le asigna un true a la variable incorrecto y se imprime en pantalla que se encontró un digito inválido para un número de base 8. Al terminar el ciclo se verifica el valor de la variable incorrecto, si es un false se llama a la función extraer () y si el valor que devuelve está entre 0 y 255 se le asigna un 4 a la variable modos (lo que indica que se trata de un modo de direccionamiento directo) y se manda a llamar a la función busqueda\_tabop (), si está entre 256 y 65535 se le asigna un 5 a la variable modos (direccionamiento extendido) y se manda a llamar a la función busqueda\_tabop(), si el valor devuelto por extraer () no cae dentro de ninguno de esos rangos se imprime en pantalla que se salió de rango; sino, si incorrecto tiene un true solo sale de la función id\_operando ().

Si operando en su primera posición tiene un “%” se trata de un modo directo o extendido; entonces se analiza el siguiente carácter del operando y si es un fin de cadena se imprime en pantalla que ese es un formato invalido para los números de base 2 y se le asigna un 1 a la variable incorrecto, sino no es un fin de cadena se entra en un ciclo while que se estará ejecutando hasta que se encuentre el fin de cadena, dentro del ciclo si se encuentra un carácter diferente de 1 y 0 se le asigna un true a la variable incorrecto y se imprime en pantalla que se encontró un digito inválido para un número de base 2. Al terminar el ciclo se verifica el valor de la variable incorrecto, si tiene un true se indica en pantalla que la base numérica 16 no admite ese carácter, si es un false se llama a la función extraer () y si el valor que devuelve está entre 0 y 255 se le asigna un 4 a la variable modos (lo que indica que se trata de un modo de direccionamiento directo) y se manda a llamar a la función busqueda\_tabop (), si está entre 256 y 65535 se le asigna un 5 a la variable modos (direccionamiento extendido) y se manda a llamar a la función busqueda\_tabop(), si el valor devuelto por extraer () no cae dentro de ninguno de esos rangos se imprime en pantalla que se salió de rango; sino, si incorrecto tiene un true solo sale de la función id\_operando ().

Si el operando en su primera posición tiene un carácter del 0… 9 o un signo de menos entonces existe la posibilidad de que se trate de un modo de direccionamiento directo, extendido, alguno de los indizados o alguno de los de auto pre/post incremento/decremento; al empezar se le asigna false a las variables signo, coma, aux, aux2 y aux3, si el operando en su siguiente posición fue un signo de menos se le asigna un true a la variable signo y en caso contrario se le asigna un false, después se entra en un ciclo while que se estará ejecutando hasta que se encuentre el fin de cadena o hasta que la variable coma tenga un true, dentro del ciclo si se encuentra un carácter diferente de 0… 9 y la coma se le asigna un true a la variable incorrecto y se imprime en pantalla que se encontró un digito que no corresponde a los números de base 10 y si se encuentra una coma se le asigna un true a la variable coma. Al terminar el ciclo se hacen varias validaciones, si operando en su posición 0 fue un signo de menos y después de este viene una coma se le asigna un true a la variable aux, si operando en su primera posición es un signo de menos y después esta el fin de cadena se le asigna un true a la variable aux2, si hay una coma y después de esta viene el fin de cadena se le asigna un true a aux3, luego de estas validaciones si aux tiene un true se imprime en pantalla que no se introdujo ningún digito antes de la coma, si aux2 tiene un true se imprime que no puede estar el signo de menos solo, si aux3 tiene un true se imprime que no se introdujo ningún registro; después de esto, si las variables incorrecto, aux, aux2 y aux3 tienen false (lo que indica que no hubo ningún error sintáctico) se pueden tomar 2 caminos, si la variable coma es un false (lo que indicaría que no se encontró la coma por lo que el operando estaría totalmente validado) se llama a la función extraer () y si el valor que devuelve está entre 0 y 255 se le asigna un 4 a la variable modos (lo que indica que se trata de un modo de direccionamiento directo) y se manda a llamar a la función busqueda\_tabop (), si está entre 256 y 65535 se le asigna un 5 a la variable modos (direccionamiento extendido) y se manda a llamar a la función busqueda\_tabop (), si el valor devuelto por extraer () no cae dentro de ninguno de esos rangos se imprime en pantalla que se salió de rango. Si por el contrario la variable coma tiene un true (lo que significa que si hubo una coma) se le asigna un true a la variable incorrecto y un false a la variable senal\_auto al igual que a la variable senal, se verifican los valores que le siguen a la coma, si son x, X, y ó Y si enseguida de estos esta el fin de cadena se le asigna un false a la variable incorrecto y un true a la variable senal, si en vez de estar el fin de cadena está el signo de más se le asigna 13 a la variable modos (post incremento), false a incorrecto y true a senal\_auto, si en vez de estar el fin de cadena está el signo de menos se le asigna 11 a la variable modos (post decremento), false a incorrecto y true a senal\_auto, lo mismo seria si después de la coma se encuentra s ó S y después p ó P, ó si después de la coma esta p ó P y después c ó C y luego el signo menos o el más. Si después de la coma esta un signo de menos y la validación del registro posterior es correcta se le asigna 10 a la variable modos (pre decremento), false a incorrecto y true a senal\_auto y si después de la coma esta el signo de más y la validación del registro posterior es correcta se le asigna 12 a la variable modos (pre incremento), false a incorrecto y true a senal\_auto. En este punto se evalúa el valor de la variable incorrecto y si es true (lo que indica que no se entro en ninguna de las anteriores condiciones, por lo que el registro es inválido) se imprime en pantalla un error, pero si es true se vislumbran otras 2 posibilidades:

- Si senal es true (lo que indicaría un modo indizado de 5, 9 ó 16 bits) se llama a la función extraer, si esta devuelve un valor de entre -16 y 15 se le asigna un 6 a la variable modos (modo indizado de 5 bits) y se manda a llamar a la función busqueda\_tabop (), si el valor que devolvió extraer () está entre -256 a -17 o entre 16 a 255 se le asigna un 7 a la variable modos (modo indizado de 9 bits) y se manda a llamar a la función busqueda\_tabop (),si el valor que devolvió extraer () está entre 256 y 65535 se le asigna un 8 a la variable modos (modo indizado de 16 bits) y se manda a llamar a la función busqueda\_tabop (), por ultimo si el valor devuelto fue cualquier otro se indica en pantalla un error sobre los rangos del numero.

-Ahora, si por el contrario la variable senal\_auto fue true se llama a la función extraer, si esta devuelve un valor de entre 1 y 8 simplemente se manda a llamar a la función busqueda\_tabop () (ya no es necesario asignarle un valor a la variable modos por qué se hizo anteriormente en las comparaciones), si el valor devuelto por extraer () está fuera del rango se imprime un mensaje de error en pantalla

Si operando en su primera posición tiene un “[“ entonces existe la posibilidad de que se trate de un modo Indizado indirecto de 16 bits o de acumulador indirecto; primero se pone la variable incorrecto en true, si después del “[“ esta el fin de cadena se imprime en pantalla que es un formato invalido ese uso de corchetes y si después del signo “[“ esta una coma se imprime que no puede haber una instrucción vacía antes de la coma, si por el contrario después del “[“ hay una d o una D se revisa que después de esto exista una coma, luego un registro valido y al final el fin de cadena, si esto se cumple entonces se le asigna un false a la variable incorrecto y si esto se cumple se le asigna un 15 a la variable modos (modo indizado de acumulador indirecto). Si no, si después del “[“ esta algún digito 0… 9 o inclusive un signo de menos se le asigna un false a la variable coma, se entra en un ciclo while que se estará ejecutando hasta que se encuentre el fin de cadena y coma sea igual a false, dentro del ciclo si se encuentra un carácter diferente de 0… 9 y la coma se le asigna un true a la variable incorrecto y se imprime que ese digito es invalido para un numero decimal, también cuando se encuentre una coma se le asigna un true a la variable coma. Al terminar el ciclo se verifica el valor de la variable incorrecto y si es false y coma true (lo que indicaría que después del principio de corchete existen dígitos decimales validos y luego una coma) se verifica que hayan registros validos luego de la coma y después un fin de corchete y al final el fin de cadena en cuyo caso se le asigna un false a la variable incorrecto, si al terminar las validaciones incorrecto es true (lo que indica que no se cumplió con las validaciones) se indica que hay un nombre de registro invalido pero si es false se manda a llamar a extraer () y si devuelve un valor de entre 0 y 65535 entonces se le asigna un 9 a la variable modos (modo Indizado indirecto de 16 bits) y se manda a llamar a la función busqueda\_tabop () pero si el valor devuelto por extraer () no es válido se indica en pantalla el error de fuera de rango. En cualquier caso, si después de “[” existe “]” entonces se indica un error ya que no puede haber corchetes vacios.

Si operando en su posición 0 es a… z o A… Z entonces existe la posibilidad de que se trate de un modo extendido, indizado de acumulador o relativo de 8 y 16 bits; primero se le asigna un false a la variable coma y se entra en un ciclo while que se estará ejecutando hasta que se encuentre el fin de cadena y coma sea false, dentro del ciclo si se encuentra la coma se le asigna un true a la variable coma. Al terminar el ciclo se verifica el valor de la variable coma y si es true se le asigna un true a la variable incorrecto, se verifica que hayan registros validos luego de la coma y al después el fin de cadena en cuyo caso se le asigna un false a la variable incorrecto, si incorrecto es igual a false se le asignan 14 a la variable modos (modo indizado de acumulador) y se manda a llamar a busqueda\_tabop (), pero si es true se indica que el registro es incorrecto. Si no se detecto ninguna coma se le asigna un false a las variables aux, aux2 y senal, se entra en un ciclo while que se estará ejecutando hasta que se encuentre el fin de cadena, dentro del ciclo si se encuentra algún carácter que no sea a… z, A… Z, 0… 9 o un guion bajo se le asigna un true a aux2, si se detecta algún digito 0... 9 se le asigna un true a la variable senal y si el primer carácter del operando se repite se le asigna un true a la variable aux. Al terminar el ciclo si la variable aux es true se indica que la etiqueta tiene el carácter de inicio repetido, si aux2 es true se indica que hubo un carácter invalido para una etiqueta y si la etiqueta es mayor a 8 se indica que su longitud es incorrecta; si las variables aux y aux2 son false y la longitud del operando es menor o igual a 8 se hace una comparación, si senal es true se le asigna un 5 a la variable modos (modo extendido) y si senal es false, si el código de operación comienza con L o con l entonces se le asigna un 16 a modos (Relativo de 16 bits) y en caso contrario se le asigna un 17 a modos (Relativo de 8 bits) y se manda a llamar a la función busqueda\_tabop ().

Si operando en su posición 0 es una coma se le asigna un true a la variable incorrecto, si después tiene registros validos y después el fin de cadena se le asigna un false a incorrecto y si esto se cumple se le asigna un 6 a modos (modo Indizado de 5 bits), si no se indica que el formato es invalido así como si después de la coma esta el fin de cadena.

Cualquier otro carácter al comienzo de la variable operando es un error, en cuyo caso solo se regresa el control a donde se llamo a la función id\_operando ().