**¿como se hacen las llamadas a funciones como parte de una expresión?**

Si son llamadas a funciones no void, significa que su valor de retorno ya esta programado para ser utilizado cuádruplos mas adelante (el cual se encuentra indexado con el mismo nombre que la función llamada), sin embargo, llamar a una función hace que la función se realice y por lo tanto inicialice o asigne un valor a esa variable de retorno que ya esta programada para ser utilizada.

Al llamar a una función, se realiza un cambio de contexto en cuanto se encuentra el cuádruplo “era”, esto significa que se guarda la memoria temporal actual en un stack y se crea memoria temporal nueva para la función llamada, también más adelante como parte de ese cambio de contexto (pero hecho por la acción “gosub”) se guarda en un stack en que función estaba antes de haber hecho el cambio de contexto, y también se guarda en otro stack en que iP (apuntador a cuádruplo) estaba antes de hacer el cambio de contexto. Hecho el cambio de contexto, se procede con los siguientes cuádruplos, que son la asignación a los parámetros de una función, lo cual sucede de la siguiente manera: los parámetros de una función ya se les fue asignada desde compilación una dirección, y los cuádruplos esperan que haya algo en esas direcciones cuando estos parámetros vayan a ser utilizados en algún cuádruplo mas adelante, por lo que lo único que hay que hacer es mover los valores que tenemos en los cuádruplos con acción “parameter” a la dirección virtual en donde se espera que guardemos los valores de los parámetros de la función para que estos puedan ser utilizados en los cuádruplos de mas adelante donde se opera con ellos, y es así como dejamos inicializados los parámetros. Luego de inicializar los parámetros de la función, lo siguiente que falta es ejecutar la función llamada al haya el “gosub”), lo cual ocurre simplemente cambiando el IP actual al IP de inicio de la función (obviamente como se menciono antes, guardando en un stack el IP donde nos quedamos y el nombre de la función en la que estábamos, es en esta acción donde ocurre es aparte del cambio de contexto). Finalmente, al realizar la función, en funciones no void nos topamos con una acción “return”, la cual hace que se guarde en memoria global, en aquella dirección indexada con el mismo nombre que la función activa (he aquí por que guardar en un stack las funciones que van quedando pendientes de resolver es util, pues nos permite recordar a quien debíamos darle return), el valor que se encuentra almacenado en uno de los operando del cuádruplo return y que esta presto para ser guardado como variable global, representando así el valor de retorno de la función. De esta manera, dicha dirección indexada con el nombre de la función queda asignada o inicializada, y por lo tanto, lista para ser utilizada por los siguientes cuádruplos, como se tenía programado hacer. Sin embargo, como se hizo para agregar esa variable global, retorno de la función llamada, a los cuádruplos sin si quiera tener un valor? la respuesta es que en compilación, al encontrarse con el fin de la llamada, no solo se agrega el cuádruplo “gosub”, sino que se agrega como operando a la lista de operandos a la variable esperada (ósea, el valor de retorno de la función, la cual se llama igual que el nombre de la función llamada), y eso hace que el compilador siga corriendo y use ese operando agregado (el valor de retorno de la función llamada) en los cuádruplos de otras operaciones

**¿como funciona la declaración de los arreglos?**

Declarar un arreglo es sencillo pues comienzas con el valor máximo de indexamiento, que es una constante la cual puede ser leída desde compilación; para hacer esto lo único que hay que hacer es: asignarle a la variable su dirección base (al igual que se hace con el resto de las variables al ser declaradas, que se les asigna la ultima dirección virtual disponible), pero a diferencia del resto, se hace un desplazamiento en la disponibilidad de memorias virtuales no en 1, sino en k, siendo k el tamaño del arreglo (es decir, en vez de asumir que la siguiente dirección virtual esta disponible, la siguiente dirección virtual disponible estará k direcciones más adelante). esto quiere decir que al asignarle memoria a una arreglo, se le proporciona su dirección base, y se dejan reservadas k direcciones para uso privado del arreglo

**¿como funciona el acceso a arreglos?**

Hacer esto es más complicado si no se consta de una constate para acceder directamente a la dirección del arreglo, pues si se trata de un indexamiento directo (una constante como indice), es fácil acceder a la dirección del arreglo solicitada, pero si se trata de un indexamiento indirecto (el index es una expresión), no queda más que movernos a ejecución para poder resolver la expresión y así conocer el valor de indexamiento. El proceso ocurre como sigue (para el indexamiento indirecto, porque el indexamiento directo es trivial y se puede resolver en compilación): como existe un conflicto entre que el compilador necesita seguir corriendo y operar con la dirección del arreglo ya como un operando disponible para ser usado por los siguientes cuádruplos, pero como no se puede disponer de tal operando completo pues no se puede conocer la dirección verdadera del arreglo sino hasta ejecución, lo que se hace es mandar como operando a la lista de operandos un operando con la información básica del arreglo y una “una migaja de pan secreta” (para acordarnos en ejecución que a ese operando, que es el arreglo al que se esta acezando, le falta su dirección virtual verdadera), y como migaja de pan usaremos lo siguiente: antes de guardar el operador antes mencionado, tuvimos que haber forzado la inserción de un cuádruplo que realice la suma entre la dirección base del arreglo y el resultado final de la expresión, lo cual generara un operando resultante temporal (el cual tiene su propia y única dirección virtual), y como es este operando en el cual se encontrará lo que buscamos (el index real del arreglo), sera a este operando al que usemos como migaja de pan, agregando al operando incompleto del arreglo además de su dirección base, una dirección falsa donde guardemos la dirección de aquel operando que tiene guardado su dirección verdadera. suena complicado, pero el orden de los eventos es realmente el siguiente: en compilación, me encuentro con que el index es una expresión, por lo que hago un cuádruplo que sume la dirección base con la expresión, lo cual guarda en el cuádruplo un temporal como resultado, y ademas lo retorna; en este temporal se encuentra la dirección verdadera del arreglo, por lo que al operando del arreglo le dejo guardado (en su atributo fakeAddress) QUIÉN tiene guardada el valor de su dirección de indexamiento real, es decir, le agrego al operando del arreglo la dirección en la cual esta su verdadera dirección de indexamiento, y esa dirección es la de la temporal resultante de sumar la dirección base del arreglo con el resultado de la expresión, y finalmente inserto un cuádruplo “arrayindex” cuya misión en ejecución es buscar aquel cuádruplo donde se encuentra el operando del arreglo al que le deje la migaja de pan, para asignarle a ese operando de arreglo su address real, para que así las operaciones con ese arreglo puedan hacerse correctamente. Dicho esto, lo que hace en ejecución la acción arrayindex es, recibir al operando en donde se encuentra el el valor real de indexamiento del arreglo, extraer dicho valor, y buscar en los cuádruplos de más adelante aquel operando donde deje la migaja de pan, pues es a ese al cual le tengo que cambiar su address por la address verdadera la cual es el valor mencionado extraído (el valor de indexamiento real) y que permitirá acceder correctamente a la casilla solicitada del arreglo. Esto es posible gracias a que es SOLO un operando al cual marque con una migaja de pan ESPECIFICA, independientemente de que haya mas operando con migajas de pan guardadas (mas arreglos pendientes de resolver en ejecución), en el momento de realizar una acción “arrayindex” es SOLO UN operando el que tiene la migaja de pan que estoy buscando; esto es, solo un operando tiene en su fakeaddress el mismo valor que el address del operando de indexamiento indirecto que estoy recibiendo en la acción “arrayindex”. Por ejemplo, digamos en "arrayindex" recibi el operando A, y sabemos que la direccion de A es 1515, entonces debo buscar AQUEL UNICO operando cuya fakeaddress sea 1515, y es a ese operando al que le voy a asignar (no en su fakeaddress, sino en su address real) el valor contenido en A, el cual es la dirección real del operando que tiene la migaja de pan.