Generación de código

Lenguaje SETU

Autor: Ítalo Garleni Rodríguez

Definición del lenguaje

Variables y tipos

Las variables del lenguaje SETU son de cuatro tipos: float, char, array y string. Las dos primeras se pueden considerar básicas, mientras que las dos siguientes son agrupaciones de las anteriores. Los array/string deben tener un tamaño fijo declarado, el cual no puede ser una variable, siempre debe ser un valor numérico. Para declarar variables, basta con informar al compilador de que la variable va a ser usada. Por ejemplo:

(a array[5])
(b string[3])
(c char)

Sin embargo, para declarar variables float no es necesario informar antes. Éstas se autogeneran cuando se usan en una expresión. Obviamente dicha variable no puede machacar su estado anterior, es decir, si tiene un tipo concreto no se puede cambiar. Por ejemplo:

(a = 5)
(b = 6)
(c = a + b)

Esto crearía 3 variables float sin necesidad de declaración previa.

Expresiones

Las expresiones aceptadas por este lenguaje son de tres tipos, aritméticas, relacionales y lógicas.

Aritméticas:

- Valor – valor → resta
- Valor * valor → producto
- Valor / valor → división

- Valor % valor → módulo
- Valor + valor → suma
- Valor1 : valor2 → despliegue en rango (de valor1 a valor2, incrementando 1, se usa para el FOR)
- Valor < valor → comparación menor
- Valor <= valor → comparación menor o igual
- Valor > valor → comparación mayor
- Valor >= valor → comparación mayor o igual
- Valor == valor → comparación de igualdad
- Valor != valor → comparación de desigualdad

Lógicas:

- ! valor → negación lógica
- Valor && valor → AND lógico
- Valor || valor → OR lógico

Estructuras de control

El tratamiento de booleanos no existe en este lenguaje. Por tanto, las estructuras de control aceptan tanto comparaciones de números como cálculos en sus condiciones de finalización. Dependiendo de si el resultado es cero o no, hará una cosa u otra.

La estructura FOR de este lenguaje se forma de la siguiente manera:

```
(a = 0)
((for i in 1:5)
        (a = a + i)
)
```

Este for recorre en “i” los valores del 1 al 5, sumando dicho numero a “a” en cada iteración.
La estructura WHILE se escribe así:

```
(a = 0)
((while (a<5))
        (a = a+1)
)
```

Para declarar una estructura con dos posibles salidas se utiliza el IF. Un ejemplo de uso es el siguiente:

```
(a = 0)
( (if (a > 0))
        (
        (a = a -3) //camino si es cierto
        )
        (
        (a = a +3) //camino si es falso
        )
)
```

Funciones

Las funciones de SETU, devuelven siempre un valor. Pueden aceptar varios valores de entrada. En todos los ficheros SETU debe existir una función principal “main”, que es donde comienza la ejecución de dicha aplicación. Todas las funciones declaran su variable de salida al principio, por lo que no es necesario realizar una sentencia “return” en este lenguaje. Es importante inicializar la variable de retorno para que no devuelva valores indeterminados. Un ejemplo de función es el siguiente:

```
((float x potencia (float a float b) ) //dos variables de entrada (a y b) , y como salida x
        (x = 0)
        ((for i in 1:b)
                (x = x + a)
        )
)
```

Detalles del compilador

Detalles generales

El código generado por el compilador se basa en almacenarlo todo en la pila. Toda variable, dato o detalle de ejecución se almacena y se administra en la pila. Dicha pila tiene su cabecera guardada en el registro R7, exceptuando algunos casos que se comentarán posteriormente. La máquina Q se ha modificado, y se suministra su objeto compilado en los archivos de la entrega con el nombre de “IQ”.

Normalmente, todo trabajo realizado almacena en pila lo que ha generado para que el que requiera ese resultado, lo encuentre en la cima de la pila. Siguiendo esta línea, toda entidad debe dejar la pila limpia y con el resultado generado encima de ella.

El ámbito de una función o estructura de control viene definido por R6. En éste se guarda el puntero donde estaba la pila cuando comenzó el ámbito en el que se encuentra. En dicha posición P(R6) se halla el R6 del ámbito anterior.

El alineamiento de la pila se controla a la hora de guardar las variables. Siempre que se almacenen variables, debe ser en múltiplos de 4, por lo que las variables CHAR y STRING puede que ocupen un espacio extra a la hora de ser almacenadas (entre 1 y 3 octetos). De este modo nos aseguramos de que la pila siempre esté alineada. Además, R6 siempre está en una posición múltiplo de 4 para facilitar el alineamiento de la memoria.

Hay abierta una “ventana” a la posibilidad de añadir matrices a la generación de código, pero por cuestión de tiempo se ha dejado esa parte comentada.

statement

Los “statement” son las sentencias que se pueden realizar en todo ámbito. Es la línea de código básica a ejecutar en el código, por tanto se encuentra en las funciones y estructuras de control. Un statement se considera una asignación “assignment”, una declaración, un “print” o una estructura de control.

print

En este lenguaje sólo se pueden imprimir variables tipo float o tipo char. Para imprimir un vector es necesario que el programador realice un bucle. La máquina Q ha sido modificada para poder añadir esta función.

expression y value

Como se ha comentado previamente, dichas entidades realizan un cálculo o leen de variables para luego dejar el resultado encontrado en la cima de la pila. Existe un caso especial, la llamada a una función, que se comentará con más detalle luego.

assignment

Constan de dos partes. Primero, la declaración de la variable y búsqueda de su ubicación en la pila, y segundo la asignación de lo almacenado en pila por “expression”. Todo ello con previa comprobación de tipos en la que la asignación tiene que tener un significado lógico.

Los errores que puede dar es al asignar un resultado de una función, ya que en el momento de crear el código, no se sabe qué devuelve la función. Por tanto, se deja a manos del programador la buena realización de las llamadas a funciones. Tampoco se comprueba el acceso fuera de rango de un array/string, ya que no se puede saber la posición a acceder hasta el momento de ejecutar el programa.

function

Las funciones constan de un protocolo de llamada. Dicho protocolo es aplicado tanto en “expression” a la hora de llamar, como en function, a la hora de generar el código. La llamada debe colocar en pila los siguientes elementos:

-R6previo: para restaurar el ámbito anterior cuando se finalize. Además, se debe actualizar el nuevo R6.
-Dirección de retorno: aquí se haya el identificador L donde debe retornar cuando termine la función.

-Variables de entrada: por último, debe añadir todas las variables de entrada.

A la hora de retornar la función, basta con restaurar el R6 antiguo, colocar en la pila el resultado y retornar a donde se indicó en la entrada.

Por otra parte, no se puede llamar a una funcion dentro de los parámetros de entrada de una llamada a otra función (ej.: MultiplicarAB(Random(), 5)), ya que generaría conflictos. Además, : los tamaños de arrays y strings no se comprueban, por lo que está en las manos del programador realizar un buen control de ello.

Restricciones del compilador

En caso de que el compilador se desee modificar (el fichero sintactico.y), se debe considerar las siguientes restricciones para no generar conflictos:

1.- Al declarar un string o un array, éste tiene que ser de un tamaño definido por un número, no por una variable/expresion.

2.- Para añadir un ámbito, se debe realizar un tabla.addScope y un asignarR6(); o similar

3.-Cuando se realiza una tarea, se asume que los valores a usar están en la pila. Por tanto, el resultado generado se guarda en la cima para el que te ha llamado a tí.

4.- cuando se crea una variable generarVariableEnPila(), la pila sólo tiene que tener guardadas variables (nada de restos de ejecución) desde el R6 relativo

5.- No se puede usar R3 en medio de una expression, porque es para guardar posibles valores de un acceso a array/string.