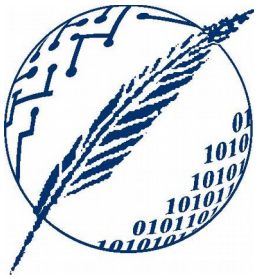
Compiladores e Intérpretes

Segundo Cuatrimestre de 2016

4° Etapa del Proyecto:

Compilador Completo De MiniJava + CeIVM



Matias Marzullo, 80902

Índice

Introducción

En el siguiente informe se detallara brevemente la información para la realización de la última etapa, analizador completo + CeIVM.

Principalmente se contaran como se desarrolló parte del compilador, de acuerdo a lo brindado por la cátedra, tanto explicación como check point.

Control de Declaraciones

En cuanto a las clases, brindan el offset de las variables y el tamaño del CIR.

En cuando a los métodos se crea una etiqueta por cada método.

Control de Sentencias

- Se reparó el error en el que se volvían a chequear los métodos heredados en las clases, para evitar conflictos de tipo. Es decir, solo se chequea si el método es redefinido.
- La sentencia "if" se controló que se utilicen bien los Branches, BF al principio y JUMP al final del then o else.
- La sentencia "while", también se controló que se usen bien los Branch, de la misma forma que el if. Pero con etiquetas FINWhile y PrincipioWhile.
- La asignación se tuvo en cuenta que primero se chequee la expresión y luego el lado izquiero. También se encontró una forma para saber cuando una expresión Primaria se encuentra en el lado izquierdo, utilizando una variable "esLadoIzq" en todos los heredados de Primario.
- Las expresiones binarias: una vez que se chequearon las dos expresiones, se genera la instrucción al token operador del nodo.
- Unarias: Luego de chequear la subexpresión, generan NEG para "-" o NOT para "!".
- Casting: DUP, luego se carga el IDClase del objeto LOADREF 1, se carga IDClase de la clase a castear: PUSH IDClase y se comparan. Se genera también un mensaje de error_casting
- Instanceof: IDClase del objeto se carga con LOADREF1, luego se carga IDClase a castear con PUSH IDClase y se comparan, generando EQ.

- Variable Primario: se pudo realizar tal como se propuso en el check point:

Si es el caso que hay un lado izquierdo y no tiene encadenados:

1. Si el id es un parámetro o variable local STORE n con su offset
2. Si el id es una variable de instancia del objeto: LOAD 3, SWAP, STOREREFn con su offset

En los otros casos:

1. Si es un parámetro o variable local: LOAD
2. Para Variable de instancia de la clase actual: LOAD 3,

- This: Para traducir una expresión this solo es necesario cargar el this en la pila dado que a this solo se lo puede utilizar del lado izquierdo de una asignación si tiene un encadenado, sino tiene encadenado tiene que ser un lado derecho.
- Llamada Estática: Para traducir la llamada a un método estático primero hay que reservar un lugar para el retorno si el tipo de retorno del método es distinto de Void, para eso usamos un RMEM 1.
- Llamada a Constructor: implica crear el nuevo CIR antes de llamar al constructor.