Compiladores
Proyecto de Curso

Generalidades

El proyecto a desarrollar durante este curso consiste en la implementación de un compilador que permita generar código ensamblador para un procesador MIPS R2000, a partir de programas escritos en lenguaje ADA 95.

Para correr los programas en MIPS se utilizará el simulador PCSPIM con capacidad de correr en un ambiente Windows.

Se utilizarán también los generadores JFlex y CUP para la implementación del compilador y el lenguaje a utilizar será Java.

El proyecto se desarrollará en grupos de dos estudiantes, estará dividido en dos fases, una en cada curso de la clase de Compiladores, éstas se describen a continuación:

Fase No.1

La primera fase del proyecto consiste en crear el analizador léxico y sintáctico. El proyecto a presentar debe considerar:

- Desarrollar una gramática que permita reconocer programas que corran en ADA 95 con las siguientes características:
    - Que puedan utilizar funciones y procedimientos con un número ilimitado de parámetros de tipo “in”, “out” o “in out”.
    - Debe reconocer los tipos Integer, Boolean, y Float
    - No deberá crear nuevos tipos necesariamente.
    - Debe incorporarse en la gramática las funciones para lectura y escritura de enteros, floats y strings. (Se utilizará Put y Get sin el prefijo asociado referente a la librería; es decir, las funciones estarán sobrecargadas)
    - Deben reconocer los bloques If, For (en su forma más sencilla), While y Loop
    - No se manejáran atributos de los tipos como ser ‘Last, ‘First, etc.

- Se permite la recursividad.

- Debe manejar procedimientos anidados

- Deberá manejarse la precedencia de operadores común, es decir + y – tienen precedencia más baja que * y /. De igual forma, el operador = tiene precedencia más baja que el resto de operadores relacionales.

- Debe ser capaz de reconocer y recuperarse de errores léxicos, errores en comentarios y errores sintácticos. Se deberá imprimir la línea y columna donde fueron encontrados cada uno de éstos y una descripción útil del mismo.

- El analizador sintáctico deberá crear un AST que será utilizado en las siguientes fases del proceso de compilación.

- Se entregarán todos los programas fuente; así como los archivos .class o .jar. El programa final deberá consistir en un solo programa que reciba como parámetros un archivo .adb. Deberán entregarse además, 3 programas fuente correctos y 3 con error.

- El programa deberá tener un cuerpo principal

Fase No.2

La fase No.2 consiste primordialmente en la generación del archivo .asm que correrá en el PCSPIM. Consideraciones:

- Deberán implementarse las funciones para lectura y escritura de variables, tanto para enteros como para float y string.

- Los parámetros a funciones podrán ser pasados de forma “in”, “out” o “in out”.

- Ningún identificador deberá ser declarado dos veces

- Chequeo del ámbito de los identificadores. No se permitirá utilizar una variable en el ámbito incorrecto. Si existen dos o más identificadores con el mismo nombre accesibles desde un ámbito especifico, la más cercana deberá ocultar las demás.

- Ningún identificador podrá ser usado sin ser declarado previamente

- Chequeo de tipos. No se permiten asignaciones ni comparaciones entre variables o constantes de diferentes tipos. También deberán chequearse que los parámetros enviados a las funciones sean el número y tipo correcto.

- Retorno de valores solamente en funciones

- Generación de código intermedio.

- Generación de código MIPS (No generará código para variables tipo float)

- Se entregarán todos los programas fuente; así como los archivos .class o .jar. El programa final deberá consistir en un solo programa que reciba como parámetros un archivo .adb y generará un archivo .asm a ser ejecutado en el simulador SPIM. Deberán entregarse además, 3 archivos fuente correctos y 3 con error; así como un reporte especificando las técnicas de programación utilizadas, los problemas encontrados y la solución encontrada a cada uno de ellos.

Convenciones Lexicográficas

- Comentario es toda secuencia de caracteres que se encuentra después de dos guiones (--) en una línea.

Puntos extras Fase 2

- Implementación de tipos definidos por el usuario

- Sobrecarga de funciones