Universidade Federal do Ceará
Centro de Ciências
Departamento de Computação
Disciplina de Construção de Compiladores I
Prof. Dr. Francisco Heron de Carvalho Junior

Relatório Final
-Construção de um compilador para linguagem MiniJava-

Equipe 9
Patrick Almeida
Helio Yuri
Victor Pontes
Paulo Henrique

Fortaleza, 25 de Junho de 2012

Introdução

O seguinte relatório visa destacar as dificuldades encontradas bem como as atividades que obtivemos sucesso ao longo da construção do compilador durante o período da disciplina.

Desenvolvimento

Etapa I: Analisador Léxico e Sintático

RESPONSÁVEIS: Paulo Henrique e Victor Pontes

A equipe utilizou um gerador automático JavaCC que gera um analisador léxico e um parser escritos na linguagem Java.

O gerador automático JavaCC recebe uma gramática, que está descrita no apêndice A do livro e a partir dela desenvolve-se todo o projeto. A gramática foi alterada com o objetivo de retirar as ambiguidades contidas em suas produções. Como orientado pelo Professor, a equipe utilizou o LOOKAHEAD. Os tokens gerados foram baseados nas regras da gramática.

A principal dificuldade dessa etapa foi a retirada das ambiguidades da gramática. Outra dificuldade que também podemos apontar foi a utilização do JavaCC, visto que nenhum membro da equipe tinha conhecimento acerca da ferramenta. O grupo teve dificuldade de conciliar horários, em alguns casos inviabilizando o encontro dos membros para discutir sobre o projeto.

Implementação feita por Paulo Henrique e Victor Pontes.

Etapa II: Árvore Sintática Abstrata, Verificação de Tipos e Tabela de Símbolos

RESPONSÁVEIS: Hélio Yuri, Patrick Almeida, Paulo Henrique Queiroz e Victor Pontes.

A equipe utilizou as classes fornecidas pelo livro texto da disciplina assim como o padrão de projeto visitor que auxiliaram na criação da tabela de símbolos e na checagem de tipos.

Ações semânticas foram inseridas no parser a fim de que a árvore sintática fosse criada e então executou-se no programa um teste através de uma função que exibia o conteúdo da árvore a fim de se certificar que todo o programa enviado havia sido armazendo na árvore sintática abstrata.

A principal dificuldade foi criar a tabela de símbolos, pois apesar de possuirmos um entendimento de como ela deveria ser implementada, nos testes realizados algumas classes, métodos e variáveis locais não eram armazenadas corretamente na estrutura de dados utilizada e a falta de reuniões periódicas por causa das dificuldades de se obter um horário em que todos estariam presentes prejudicou bastante o andamento do restante do projeto.

Implementação feita por Hélio Yuri (Tabela de Símbolos), Patrick Almeida e Hélio Yuri (Checagem de Tipos), Paulo Henrique e Victor Pontes (Árvore Sintática Abstrata).

Etapa III: Registro de Ativação e Código Intermediário

RESPONSÁVEIS: Patrick Almeida

Etapa para a construção da árvore de representação intermediária(IR). A facilidade que a árvore traz para o programa é que ela independe de linguagem fonte e da arquitetura de máquina alvo e além de fornecer essa facilidade, ela torna o processo de tradução para código de máquina menos complicado.

Para construir a árvore de representação intermediária devemos percorrer a árvore sintática abstrata usando o padrão Visitor, de modo que cada nó da árvore é um objeto Exp que possui uma expressão como atributo.

A maior dificuldade encontrada nessa etapa foi construir a árvore IR gerando os fragmentos utilizando o Frame. Também podemos apontar como dificuldade a geração código referente ao acesso das variáveis utilizadas no programa. Outro ponto negativo foi a reunião dos membros da equipe.

Implementação feita por Patrick Almeida (Representação Intermediária)

Etapa IV: Blocos Básicos, Traço e Seleção de Instruções

RESPONSÁVEIS: Victor Pontes e Paulo Henrique

Nesta etapa, a construção de blocos e traços é baseada na árvore canônica. Obtemos a árvore canônica a partir da árvore de representação intermediária(IR). Construímos a árvore canônica devido às incompatibilidades que podem existir entre a árvore de representação intermediária e a linguagem de máquina. Podemos apontar como dificuldade dessa etapa a seleção das instruções. Usamos a seleção de instrução para substituir cada ladrilho da árvore canônica por sua instrução de máquina correspondente utilizando o algoritmo MaximalMunch. Além disso, são determinados os temporários de valores intermediários calculados por expressões.

Implementação feita por Paulo Henrique (Blocos Básicos e Traço) e Victor Pontes (Seleção de Instruções).

Etapa V: Análise de Longevidade e Alocação de Registradores
Não implementada

Etapa VI: Integração dos componentes
Não implementada

Conclusão

A implementação do compilador foi bastante útil para sedimentar a fundamentação teórica que tivemos com o Professor ao longo da disciplina.
A prática através da programação, associada a uma boa discussão teórica em sala tornaram o processo de aprendizagem, de fato, vivenciado pelos alunos durante o semestre letivo.
No entanto o pouco tempo e a difícil integração da equipe no início do projeto foram os principais fatores que levaram a não realização completa do projeto.