Processamento de Linguagens (3º ano de Curso)

Trabalho Prático 2

Relatório de Desenvolvimento

Jorge Miguel Sol Ferreira (a64293)
Pedro José Freitas da Cunha (a67677)
José Pedro Brito Pereira (a67680)
Grupo 35

3 de Junho de 2015

Resumo

Este relatório documentará todos os passos tomados na realização do segundo trabalho prático da Unidade Curricular de Processamento de Linguagens.
Neste projecto é requerida a implementação de um compilador de uma Linguagem de Programação Imperativa Simples e posteriormente gerador de código assembly para uma máquina de stacks virtual.

Conteúdo

Capítulo 1

Introdução

Enquadramento Um **compilador** é uma peça de software que transforma o código fonte numa dada linguagem de alto nível em instruções que a máquina entenda (Código Máquina). As fazes da compilação incluem:
Análise léxica
Análise Sintática
Análise Semântica
Geração de Código

Conteúdo do documento Neste documento encontrar-se-ão as fases de resolução do problema especificado

Resultados – pontos a evidenciar O resultado do projecto a desenvolver será um gerador de códigoassembly para uma máquina de stacks virtual, partindo de uma Linguagem de Programação Imperativa Simples.

Estrutura do Relatório

No capítulo 2 iremos apresentar o caso de estudo em causa. No capítulo 3 iremos apresentar a estrutura de dados auxiliar à análise semântica e sua utilização no compilador a desenvolver, bem como fazer um esboço do que queremos que seja a nossa linguagem de programação (Gramática Independente do Contexto), para além de alguns exemplos de frases que tenham erros sintáticos, semânticos e frases correctas segundo a nossa especificação. No capítulo 4 iremos apresentar os passos utilizados para geração de código, se possível ou, em alternativa notificaçao de erro sintático. Finalmente, no capítulo 5 faremos uma apreciação crítica do trabalho realizado e trabalhos futuros. Em anexo iremos colocar o código desenvolvido que permitirá a geração de código máquina.

Capítulo 2

Análise e Especificação

2.1 Descrição informal do problema

Neste projecto é pretendido o desenho de uma Linguagem de Programação Imperativa Simples, para de seguida criar um compilador que gere pseudo-código Assembly de uma Máquina Virtual de Stacks

2.2 Especificação do Requisitos

Os requisitos para o compilador/linguagem a implementar são os seguintes:

- Permitir manusear variáveis do tipo inteiro(escalar ou array).
- Realizar as seguintes operações:
  - Atribuições de expressões a variáveis.
  - Ler do Standard Input.
  - Escrever para o Standard Output.
- Ciclos(for, while) e instruções Condicionais(if..else).
- Operações Aritméticas, Relacionais e Lógicas sobre inteiros.
- Indexação sobre arrays.
- As declarações de variáveis deverão ser no início do programa.
- Não deverá ser possível realizar redeclaraçoes nem utilizações sem declaraçao prévia.
- Se não existirem atribuições a uma variável, o valor da mesma deverá ser indefinido

Capítulo 3

Concepção/desenho da Resolução

3.1 Gramática

Ao desenvolver a gramática tentámos fazer com que a mesma ficasse o mais próximo possível do C, tendo pequenas diferenças com a gramática existente no C.
Abaixo encontra-se a gramática desenhada para a Linguagem:

```
Prog ->Decls Instrs

Decls ->InitVar
      | Decls InitVar

InitVar ->'int' Var Array ';'

Var ->id

Array ->
      | '[' num ']''

Instrs ->Instr
       | Instrs Instr

Inst ->If
     | While
     | For
     | Atr ';'
     | IO ';'
```

```
If ->'if' '(' Cond ')' '{' Instrs '}' Else

Else ->
     | 'else' '{' Instrs '}'

While  ->  'while' '(' Cond ')' '{' Instrs '}'

For ->'for' '(' Atr ';' Cond ';' Atr ')' '{' Instrs '}'

Atr ->Var Array '='Exp

IO ->'print' Out
   | 'input' Var

Out ->Exp
    | '\"' id '\"'

Exp ->Termo
    | Exp OpA Termo

Termo ->Fator
      | Termo OpM Fator

Fator ->Var Array
      | num
      | '(' Exp ')'
      | '!' Exp

Comp ->Exp
     | Exp OpConp Exp
```

3.2 Estruturas de Dados

Para realizarmos a análise semântica temos uma tabela de Hash para guardar todos os identificadores de variáveis e seus tipos de dados.
A estrutura de dados da tabela de Hash é a seguinte:

```
struct list{
           char *key;
           char *type;
           int init;
           int ind;
           int tamanho;
           struct list* next;};
```

```
struct table{
           int size;
           int elems;
           struct list **list;
};
```

Com a estrutura acima referida podemos então informações sobre uma variável (Identificador, Tipo, Estado de Inicialização, Índice do Registo na Máquina e Tamanho, que será 1 caso se trate de uma variável escalar ou então será o tamanho do vector).

3.3 Exemplos de Programas da Linguagem

Abaixo irão estar apresentados vários programas, em que os dois primeiros não cumprem os requisitos sintáticos (3.3.1) e semânticos (3.3.2) da nossa linguagem. Posteriormente iremos apresentar um programa que esteja sintática e semanticamente correcto (3.3.3).

3.3.1 Programas com erros sintáticos

3.3.2 Programas com erros semânticos

3.3.3 Programas correctos

Capítulo 4

Codificação e Testes

4.1 Alternativas, Decisões e Problemas de Implementação

4.2 Testes realizados e Resultados

Mostram-se a seguir alguns testes feitos (valores introduzidos) e os respectivos resultados obtidos:

Capítulo 5

Conclusão

Síntese do Documento.
Estado final do projecto; Análise crítica dos resultados.
Trabalho futuro.

Apêndice A

Código do Programa

```
aqui deve aparecer o código do programa,
tal como está formato no ficheiro-fonte "darius.java"
```