Disciplina: Linguagens de Programação
Docentes: Teresa Gonçalves e Nuno Miranda

Carregamento de Programas CIMS

Realizado por:

- Alexandre Rodrigues 35449
- Pedro Oliveira 35480

12 de abril de 2018

Índice

Introdução

Foi-nos pedido para implementar uma máquina abstracta CIMS numa linguagem à nossa escolha (Java, Python, C).

O objetivo desta primeira parte será ler um programa CIMS na sua entrada padrão e carregá-la na memória de instruções na máquina.

A estrutura de dados que achámos mais apropriada para esta tarefa foi uma ArrayList. Caso tivéssemos escolhido um Array teríamos de o inicializar com um tamanho. Podia dar-se o caso de precisarmos de mais espaço, o que seria inconveniente visto que o nosso tamanho seria fixo. Ao usarmos uma ArrayList não estamos dependentes de um tamanho.

Implementação

A implementação foi feita da seguinte forma, criámos uma classe abstrata para as instruções que apenas contém a Label, visto que é o único elemento que todas têm em comum.

De seguida criámos uma classe para cada tipo de instrução (Saida, Salto, AcessoVars, etc.), que, dependendo da instrução, o construtor receberá mais argumentos para além da Label, ou não.

Posteriormente alterámos o ficheiro CIMS.cup de modo a criar os objetos para cada uma das nossas classes.

A nossa ArrayList foi criada no ficheiro CIMS.java.

Ficheiros

- Instrucoes: criação da classe instruções que só disponibiliza a Label;

- Aritmetica: criação da função geral para as instruções aitméticas;

- Salto: criação da função geral para as instruções de salto;

- ManipulacaoInts: crição da função geral para as instruções para manipulação de inteiros;

- AcessoVars: criação da função geral para as instruções de acesso a variáveis;

- AcessoArgs: criação da função geral para as instruções de acesso a argumentos;

- ChamadaFunc: criação da função geral para as instruções de chamada de funções;

- Saida: criação da função geral para as instruções de saída;

- Label: especificação da função etiqueta;

- CIMS.lex: ficheiro de análise lexical, devolve uma string de tokens;

- Cims.cup: ficheiro de análise semântica, guarda as instruções e argumentos em memória;

- CIMS.java: ficheiro onde criamos e inicializamos a ArrayList a usar;

- Main: criação da máquina abstrata e print das instruções em memória.

Observações

Não incluímos ClassPath nem para o java_cup nem para o java_lex no nosso makefile, pois temo-los na mesma diretoria juntamente com os restantes ficheiros.

Para a execução do make usamos o comando “make”, no entanto temos de ter o ficheiro que queremos usar na mesma diretoria que o ficheiro make.

Para mostrar a memória usamos o comando “java Main < exemplo.cims”, onde exemplo é o nome do ficheiro usado no make.

Conclusão

Inicialmente tivémos dificuldades em perceber como realizar a implementação do trabalho, ultrapassado este obstáculo, tornou-se um trabalho relativamente acessível.