INSTITUTO MAUÁ DE TECNOLOGIA

MAUÁ

ECM253 – Linguagens Formais,Autômatos e Compiladores

Projeto 4

Analisador sintático com CUPS
Montador para o Marie

Prof. Marco Furlan

30 de outubro de 2019

1 O projeto

- No Moodlerooms da disciplina pode-se encontrar o arquivo `MarieComp.zip`. Nele há uma implementação de um montador que converte expressões aritméticas simples em código assembly para a arquitetura do Marie;
- A gramática de expressões utilizada foi a seguinte (BNF):

<program> ::= <expr_list>
<expr_list> ::= <expr_list><expr_part> | <expr_part>
<expr_part> ::= <expr>
<expr> ::= <expr>“+”<expr> | <expr>“-”<expr> | `number` | “(”<expr>“)” | `id` | `id`“=”<expr>

- No projeto, as expressões básicas reconhecidas no CUP geram uma parte da árvore **AST**. Quando esta expressão estiver pronta, a árvore poderá ser percorrida e gerar código.
- O esquema de percorrimento de árvore utiliza um **padrão de projeto** denominado **Reflexive Visitor** (<https://www.cse.wustl.edu/~cytron/cacweb/Tutorial/Visitor/>) e todo o código a ser gerado pode ser definido na classe `CodeGenerator` do projeto.

2 O que é para fazer?

- Primeiramente, executar o montador e testar com diversas expressões, executando o código Marie gerado em algum simulador, por exemplo, <https://marie.js.org>. O professor explicará o código em aula.
- Implantar um comando para exibir o resultado de expressões aritméticas, `print`. A gramática agora fica assim:

```
...
<expr> ::= <expr>“+”<expr> | <expr>“-”<expr> | number | “(”<expr>“)” | id | id“=”<expr> |
print<expr>
```

- Implantar alguma operação não existente no conjunto de instrução do Marie (de preferência alguma estudada em Arquitetura de Computadores, por exemplo, multiplicação) e então adicione-a à gramática e implemente-a no programa.

3 Sugestões

- O comando `print` pode ser encarado como um **operador unário**. Sua implementação consiste em alterar os arquivos `Scanner.jflex`, `Parser.cup` e `CodeGenerator.java`. Como o projeto atual considera apenas operadores binários, para criar um operador unário na análise sintática basta criar um operador cujo lado esquerdo é nulo e o lado direito é um caractere que representa impressão (por exemplo, “`@`”). Depois, sabendo que o operador é unário, alterar o arquivo `CodeGenerator.java` para gerar as instruções de impressão para o Marie.
- Para adicionar uma operação que não existe no Marie, por exemplo, multiplicação, adicionar o operador alterando convenientemente os arquivos `Scanner.jflex`, `Parser.cup` e depois alterar o arquivo `CodeGenerator.java` acrescentando as linhas que implemente tal operador.