Construção de Compiladores 1 - 2015.1 - Prof. Daniel Lucrédio
Aula 09 - Geração de código e otimização - roteiro

Demonstração 1
Ambiente de execução P-código

---

1. Abrir o NetBeans e mostrar o projeto da máquina P-código
2. Testar com diferentes exemplos

```
// 2 * a + (b - 3)
lda 0
rdi
lda 1
rdi
ldc 2
lod 0
mpi
lod 1
ldc 3
sbi
adi
wri
// Fatorial
lda 0
rdi
lod 0
ldc 0
grt
fjp L1
lda 1
ldc 1
sto
lab L2
lda 1
lod 1
lod 0
mpi
sto
lda 0
lod 0
ldc 1
sbi
sto
lod 0
ldc 0
equ
fjp L2
lod 1
wri
lab L1
stp
```

Demonstração 2
Gerando P-código a partir da linguagem Alguma

---

1. Criar um arquivo de teste no Desktop

```
:DECLARACOES
num:INTEIRO
potencia:INTEIRO
aux:INTEIRO
resultado:INTEIRO
:ALGORITMO
LER num
LER potencia
SE potencia = 0
ENTAO
   ATRIBUIR 1 A resultado
SENAO
   INICIO
      ATRIBUIR num A resultado
      ATRIBUIR 1 A aux
      ENQUANTO aux < potencia
         INICIO
            ATRIBUIR resultado * num A resultado
            ATRIBUIR aux + 1 A aux
         FIM
   FIM
FIMSE
IMPRIMIR resultado
```

2. Abrir o NetBeans, e criar um novo projeto Java “AlgumaParserComGeracaoDeCodigo”
3. Abrir o ANTLRWorks, e criar a seguinte gramática

```
grammar Alguma;

programa : ':' 'DECLARACOES' listaDeclaracoes ':' 'ALGORITMO' listaComandos;
listaDeclaracoes : (declaracao)+;
declaracao : VARIAVEL ':' TIPO_VAR;
expressaoAritmetica : termoAritmetico (OP_ARIT1 termoAritmetico)*;
termoAritmetico : fatorAritmetico (OP_ARIT2 fatorAritmetico)*;
fatorAritmetico : NUMINT | NUMREAL | VARIAVEL | '(' expressaoAritmetica ')';
expressaoRelacional : termoRelacional (OP_BOOL termoRelacional)*;
termoRelacional : expressaoAritmetica OP_REL expressaoAritmetica | '['
expressaoRelacional ']';
listaComandos : (comando)+;
comando : comandoAtribuicao | comandoEntrada | comandoSaida | comandoCondicao
| comandoRepeticao | subAlgoritmo;
comandoAtribuicao : 'ATRIBUIR' expressaoAritmetica 'A' VARIAVEL;
comandoEntrada : 'LER' VARIAVEL;
comandoSaida : 'IMPRIMIR' VARIAVEL;
comandoCondicao : 'SE' expressaoRelacional 'ENTAO' comando ('SENAO' comando)?
'FIMSE';
```

```
comandoRepeticao : 'ENQUANTO' expressaoRelacional comando;
subAlgoritmo : 'INICIO' listaComandos 'FIM';

TIPO_VAR : 'INTEIRO' | 'REAL';
NUMINT : ('0'..'9')+;
NUMREAL : ('0'..'9')+ ('.' ('0'..'9')+)?;
OP_ARIT1 : '+' | '-';
OP_ARIT2 : '*' | '/';
OP_REL : '>' | '>=' | '<' | '<=' | '<>' | '=';
OP_BOOL      : 'E' | 'OU';
VARIAVEL : ('a'..'z'|'A'..'Z') ('a'..'z'|'A'..'Z'|'0'..'9')*;
COMENTARIO : '%' ~('\n'|'\r')* '\r'? '\n' {skip();};
WS : ( ' ' |'\t' | '\r' | '\n') {skip();};
```

4. Testar o analisador com o seguinte código, mostrar a árvore sendo gerada

```
        ANTLRInputStream input = new ANTLRInputStream(new
FileInputStream(<teste>));
        AlgumaLexer lexer = new AlgumaLexer(input);
        CommonTokenStream tokens = new CommonTokenStream(lexer);
        AlgumaParser parser = new AlgumaParser(tokens);
        AlgumaParser.ProgramaContext result = parser.programa();
        System.out.println(result.toStringTree());
```

5. No NetBeans, criar as classes para análise semântica

```
public class EntradaTabelaDeSimbolos {
    public String nome;
    public int valor;
}

public class TabelaDeSimbolos {
    private Map<String, EntradaTabelaDeSimbolos> tabelaDeSimbolos;

    public TabelaDeSimbolos() {
        tabelaDeSimbolos = new HashMap<>();
    }

    public void inserir(String nome, int valor) {
        EntradaTabelaDeSimbolos etds = new EntradaTabelaDeSimbolos();
        etds.nome = nome;
        etds.valor = valor;
        tabelaDeSimbolos.put(nome, etds);
    }

    public EntradaTabelaDeSimbolos verificar(String nome) {
        if(!tabelaDeSimbolos.containsKey(nome))
            return null;
        else return tabelaDeSimbolos.get(nome);
    }
}
```

6. Adicionar as ações semânticas para geração de código

```
grammar Alguma;

@members {
TabelaDeSimbolos ts = new TabelaDeSimbolos();
int endereco = 0;
int label = 0;
}

programa returns [ String pcod ] :
    ':' 'DECLARACOES' listaDeclaracoes ':' 'ALGORITMO' listaComandos
    { $pcod = $listaComandos.pcod + "stp\n"; }
    ;

listaDeclaracoes : (declaracao)+;

declaracao :
    VARIAVEL ':' TIPO_VAR
    { if(ts.verificar($VARIAVEL.getText()) != null) {
            throw new RuntimeException("Erro semântico: variável " +
                $VARIAVEL.getText() + " declarada mais do que uma vez!");
      } else {
         ts.inserir($VARIAVEL.getText(), endereco++);
      }
    }
    ;

expressaoAritmetica returns [ String pcod ]:
    termo1=termoAritmetico { $pcod = $termo1.pcod; }
    (OP_ARIT1 termo2=termoAritmetico
        { $pcod += $termo2.pcod;
          if($OP_ARIT1.getText().equals("+")) $pcod += "adi\n";
          else if($OP_ARIT1.getText().equals("-")) $pcod += "sbi\n";
        }
    )*
    ;

termoAritmetico returns [ String pcod ]:
    fator1=fatorAritmetico { $pcod = $fator1.pcod; }
    (OP_ARIT2 fator2=fatorAritmetico
        { $pcod += $fator2.pcod;
          if($OP_ARIT2.getText().equals("*")) $pcod += "mpi\n";
          else if($OP_ARIT2.getText().equals("/")) $pcod += "dvi\n";
        }
    )*
    ;

fatorAritmetico returns [ String pcod ]:
      NUMINT { $pcod = "ldc "+$NUMINT.getText() + "\n"; }
```

```
    | NUMREAL { $pcod = "ldc "+$NUMREAL.getText() + "\n"; }
    | VARIAVEL { if(ts.verificar($VARIAVEL.getText()) == null)
                    throw new RuntimeException("Erro semântico: variável "+
                        $VARIAVEL.getText() + " não foi declarada!");
                 else {
                    int endereco = ts.verificar($VARIAVEL.getText()).valor;
                    $pcod = "lod "+endereco + "\n";
                 }
               }
    | '(' expressaoAritmetica ')' { $pcod = $expressaoAritmetica.pcod; }
    ;

expressaoRelacional returns [ String pcod ]:
    termo1=termoRelacional { $pcod = $termo1.pcod; }
    (OP_BOOL termo2=termoRelacional
     { $pcod += $termo2.pcod;
       if($OP_BOOL.getText().equals("E")) $pcod += "and\n";
       else if($OP_BOOL.getText().equals("OU")) $pcod += "or\n";
     }
    )*
    ;

termoRelacional returns [ String pcod ]:
      exp1=expressaoAritmetica OP_REL exp2=expressaoAritmetica
      { $pcod = $exp1.pcod + $exp2.pcod;
        if($OP_REL.getText().equals(">")) $pcod += "grt\n";
        else if($OP_REL.getText().equals(">=")) $pcod += "gte\n";
        else if($OP_REL.getText().equals("<")) $pcod += "let\n";
        else if($OP_REL.getText().equals("<=")) $pcod += "lte\n";
        else if($OP_REL.getText().equals("<>")) $pcod += "neq\n";
        else if($OP_REL.getText().equals("=")) $pcod += "equ\n";
      }
    | '[' expressaoRelacional ']' { $pcod = $expressaoRelacional.pcod; }
    ;

listaComandos returns [ String pcod ]:
    { $pcod = ""; }
    (comando { $pcod  += $comando.pcod; }
    )+
    ;

comando returns [ String pcod ]:
      comandoAtribuicao { $pcod = $comandoAtribuicao.pcod; }
    | comandoEntrada    { $pcod = $comandoEntrada.pcod; }
    | comandoSaida      { $pcod = $comandoSaida.pcod; }
    | comandoCondicao   { $pcod = $comandoCondicao.pcod; }
    | comandoRepeticao  { $pcod = $comandoRepeticao.pcod; }
    | subAlgoritmo      { $pcod = $subAlgoritmo.pcod; }
    ;

comandoAtribuicao returns [ String pcod ]:
```

```
    'ATRIBUIR' expressaoAritmetica 'A' VARIAVEL
    { if(ts.verificar($VARIAVEL.getText()) == null) {
        throw new RuntimeException("Erro semântico: variável " +
                $VARIAVEL.getText() + " não foi declarada!");
      } else {
        int endereco = ts.verificar($VARIAVEL.getText()).valor;
        $pcod = "lda " + endereco + "\n" +
                $expressaoAritmetica.pcod +
                "sto\n";
      }
    }
    ;

comandoEntrada returns [ String pcod ]:
    'LER' VARIAVEL
    { if(ts.verificar($VARIAVEL.getText()) == null) {
        throw new RuntimeException("Erro semântico: variável " +
                $VARIAVEL.getText() + " não foi declarada!");
      } else {
        int endereco = ts.verificar($VARIAVEL.getText()).valor;
        $pcod = "lda " + endereco + "\n" +
            "rdi\n";
      }
    }
    ;

comandoSaida returns [ String pcod ]:
    'IMPRIMIR' VARIAVEL
    { if(ts.verificar($VARIAVEL.getText()) == null) {
        throw new RuntimeException("Erro semântico: variável " +
            $VARIAVEL.getText() + " não foi declarada!");
      } else {
        int endereco = ts.verificar($VARIAVEL.getText()).valor;
        $pcod = "lod " + endereco + "\n" +
                "wri\n";
      }
    }
    ;

comandoCondicao returns [ String pcod ]:
    { boolean temSenao = false; }
    'SE' exp=expressaoRelacional 'ENTAO' cmd1=comando ('SENAO' cmd2=comando {
temSenao = true; })? 'FIMSE'
    { int label1 = label++;
      $pcod = $exp.pcod;
      $pcod += "fjp L"+label1+"\n";
      $pcod += $cmd1.pcod;
      if(temSenao) {
        int label2 = label++;
        $pcod += "ujp L"+label2+"\n";
        $pcod += "lab L"+label1+"\n";
```

```
        $pcod += $cmd2.pcod;
        $pcod += "lab L"+label2+"\n";
      } else {
        $pcod += "lab L"+label1+"\n";
      }
    }
    ;

comandoRepeticao returns [ String pcod ]:
    'ENQUANTO' exp=expressaoRelacional comando
    { int label1 = label++;
      int label2 = label++;
      $pcod = "lab L"+label1+"\n";
      $pcod += $exp.pcod;
      $pcod += "fjp L"+label2+"\n";
      $pcod += $comando.pcod;
      $pcod += "ujp L"+label1+"\n";
      $pcod += "lab L"+label2+"\n";
    }
    ;

subAlgoritmo returns [ String pcod ]:
    'INICIO' listaComandos 'FIM'
    { $pcod = $listaComandos.pcod; }
    ;

TIPO_VAR : 'INTEIRO' | 'REAL';
NUMINT : ('0'..'9')+;
NUMREAL : ('0'..'9')+ ('.' ('0'..'9')+)?;
OP_ARIT1 : '+' | '-';
OP_ARIT2 : '*' | '/';
OP_REL : '>' | '>=' | '<' | '<=' | '<>' | '=';
OP_BOOL      : 'E' | 'OU';
VARIAVEL : ('a'..'z'|'A'..'Z') ('a'..'z'|'A'..'Z'|'0'..'9')*;
COMENTARIO : '%' ~('\n'|'\r')* '\r'? '\n' {skip();};
WS : ( ' ' |'\t' | '\r' | '\n') {skip();};
```

6. Testar, usando o seguinte código

```
        AlgumaParser.ProgramaContext result = parser.programa();
        System.out.println(result.pcod);
```

7. Outros algoritmos para testar

```
:DECLARACOES
argumento:INTEIRO
fatorial:INTEIRO

:ALGORITMO
% Calcula o fatorial de um número inteiro
LER argumento
```

```
ATRIBUIR argumento A fatorial
SE argumento = 0 ENTAO ATRIBUIR 1 A fatorial
FIMSE
ENQUANTO argumento > 1
   INICIO
      ATRIBUIR fatorial * (argumento - 1) A fatorial
      ATRIBUIR argumento - 1 A argumento
   FIM
IMPRIMIR fatorial

:DECLARACOES
numero1:INTEIRO
numero2:INTEIRO
numero3:INTEIRO
aux:INTEIRO

:ALGORITMO
% Coloca 3 números em ordem crescente
LER numero1
LER numero2
LER numero3
SE numero1 > numero2 ENTAO
   INICIO
      ATRIBUIR numero2 A aux
      ATRIBUIR numero1 A numero2
      ATRIBUIR aux A numero1
   FIM
FIMSE
SE numero1 > numero3 ENTAO
   INICIO
      ATRIBUIR numero3 A aux
      ATRIBUIR numero1 A numero3
      ATRIBUIR aux A numero1
   FIM
FIMSE
SE numero2 > numero3 ENTAO
   INICIO
      ATRIBUIR numero3 A aux
      ATRIBUIR numero2 A numero3
      ATRIBUIR aux A numero2
   FIM
FIMSE
IMPRIMIR numero1
IMPRIMIR numero2
IMPRIMIR numero3
```