Primeira Prova de Construção de Compiladores
Primeiro Semestre de 2008, DC-UFSCar
Prof. José de Oliveira Guimarães
Quinta 08h-12h

Todas as questões da prova utilizam a gramática da linguagem do trabalho 2:

```
Program ::= [ VarDec { VarDec } ] CompositeCommand
CompositeCommand ::= "begin" Command  { Command } "end"
Command ::=  PrintCommand | AssignmentCommand
           | WhileCommand | ForCommand | IfCommand
           | CompositeCommand
VarDec ::= Type Id { "," Id  } ";"
Type ::= "integer" | "boolean"
AssignmentCommand ::= Id "=" OrExpr ";"
PrintCommand ::=  "print" OrExpr ";"
WhileCommand ::= "while"  OrExpr "do"  Command
ForCommand ::= "for" Id "=" OrExpr "to" OrExpr "do" Command
IfCommand ::= "if"  OrExpr "then"  Command
      [ "else" Command ]
```

As produções para expressões, OrExpr e outras, não serão necessárias para fazer esta prova. Para fazer as questões abaixo, é necessário saber que:

a) a expressão do if deve ser booleana;
b) a variável de um comando for pode ser integer ou boolean, mas as duas outras expressões devem ser do mesmo tipo que a variável. Isto é, em
   for k = e1 to e2 do cmd:
   k, e1 e e2 devem ser do mesmo tipo (isto é diferente do trab. 2).

Assuma que já existam todos os métodos de análise sintática que não são exigidos nesta prova. Por exemplo, assuma que exista um método

    Expr orExpr()

que faz a análise sintática de uma expressão e retorne o objeto correspondente. Da mesma forma, existem métodos CommandList compositeCommand() e Variable varDec().

1. (7.0) Baseado nesta gramática, faça:
   a) as classes da ASA Command, ForCommand e IfCommand. Coloque as variáveis de instância e heranças (se houver). Não é necessário nenhum método;
   b) os métodos command, forCommand e ifCommand do analisador sintático com todas as conferências semânticas. Naturalmente, construa a ASA durante esta análise.

Os métodos do item b) estão na classe Compiler que possui um método nextToken responsável pela análise léxica. O token corrente é colocado na variável de instância token de Compiler, do tipo int. As constantes da análise léxica estão na classe Symbol (IDENT para Id, VAR, FOR, etc). Há uma variável de instância symbolTable na classe Compiler do tipo Hashtable. Lembre-se de que esta classe possui métodos Object get(Object key) e Object put(Object key, Object value). O método error de Compiler deve ser utilizado para emitir mensagens de erro (que não precisam ser significativas --- chame error sem parâmetros). Se o token corrente for Symbol.IDENT, a variável de instância stringValue de Compiler guarda o identificador (uma string). Admita que exista uma classe abstrata Expr superclasse de todas as classes de expressões. Esta classe possui um método Type getType() que retorna o tipo da expressão. A classe Type é superclasse das classes IntegerType e BooleanType. Há um único objeto de cada uma delas e este objeto representa o tipo correspondente através de toda a compilação. Cada um destes objetos pode ser manipulado fazendo-se "Type.integerType" e "Type.booleanType", onde integerType e booleanType são variáveis estáticas da classe Type. Qualquer outra dúvida a respeito de métodos/classes do compilador que você precise poderá ser esclarecida pelo professor.

2. (3.0) Faça o método genASM da classe da ASA ForCommand. A geração de código deve ser para o seguinte assembler.

Assembler	significado
mov Ri, Rj	Ri = Rj
mov Ri, N	Ri = N
push Ri	empilha Ri
pop Ri	desempilha Ri
add Ri, N add Ri, Rj	Ri = Ri + N Ri = Ri + Rj
sub Ri, N	Ri = Ri – N
sub Ri, Rj	Ri = Ri – Rj
mult Ri, N	Ri = Ri*N
mult Ri, Rj	Ri = Ri*Rj
goto L	goto para o label L
goto< Ri, Rj, L	se Ri < Rj, goto L
goto= Ri, Rj, L	se Ri == Rj, goto L
goto> Ri, Rj, L	se Ri > Rj, goto L

Assuma que existam dez registradores, R0, R1, R2, ... R9 e que N seja qualquer número natural menor do que 32768 (exclusive). Os métodos genC das subclasses da classe Expr geram código que deixam o resultado da expressão no topo da pilha. As expressões do for devem ser avaliadas uma única vez antes do início do comando, na ordem em que foram declaradas. Assuma que você pode utilizar R0 e R1 para qualquer a variável do for e qualquer outro valor temporário. Mas os label´s do assembler não podem se reutilizados ...

3. (2.0) Faça a representação gráfica da ASA do seguinte comando:
   for i = 0 to n do s = s + i;
   Utilize as convenções mostradas no quadro.