AULAS TEÓRICO-PRÁTICAS DE COMPILADORES

2º semestre de 2002/2003

FICHA Nº 10 (3x3 horas)

Esta ficha apresenta o enunciado do último trabalho prático que será realizado com o acompanhamento proporcionado por 3 aulas práticas e com esforços extra aulas.

1 Exercício

Pretende-se implementar um interpretador ou um compilador para uma linguagem de programação muito simples, designada por **ualg**. Nesta linguagem todas as variáveis e constantes são do tipo inteiro com sinal (representado em 32 bits). A linguagem aceita expressões aritméticas e operações relacionais, construções condicionais do tipo `if` e `if-else` (sem encadeamentos de construções condicionais), e construções `while` (sem encadeamentos de construções `while`). Os programas na linguagem **ualg** são constituídos por uma única função (a função tem de retornar sempre uma variável e pode ter ou não argumentos) e não existem chamadas a funções. A linguagem **ualg** não deve diferenciar minúsculas de maiúsculas.

São apresentados na Figura 1 exemplos e algumas restrições da linguagem (a gramática da linguagem encontra-se representada em EBNF na secção 2) . A Figura 2 apresenta exemplos de programas em **ualg**.

Compilador: Para executar o compilador deve ser feito: `java ualg [-o] programa.ualg`. A opção **–o** deve ser incorporada para os alunos que consigam optimizar a geração do código:

1. Utilização do menor número possível de espaço em memória para armazenar variáveis (atribuir o maior número possível de variáveis a registos do microprocessador);

Interpretador: Para executar o interpretador deve ser feito: `java ualg [-p|-b] programa.ualg`. As opções `-p` e `-b` devem ser incorporadas para permitir a interpretação linha-a-linha (opção `-p`) e a interpretação até encontrar uma instrução `read` (opção `-b`).

Tipo	Construções, Operações, e entrada/saída	Exemplo	Restrições
Operações aritméticas e lógicas	*, /, +, -, <<, >>, &, |	`a=2*b;`	Operações RHS com apenas uma operação
Relacionais (Apenas utilizados como teste nas construções if, if-else, while)	==, !=, >, <, >=, <=	`If(a >= b) { … }`	Apenas utilizados nas construções if e while
Construções condicionais	If, if-else	`If(a == b) { C=2; }`	Não é permitido encadeamento de construções if
Construções cíclicas	while	`While(a != 2) { A = a-1; }`	Não é permitido encadeamento de construções while
Atribuição	=	`A=2;` `A=b;`	
Imprimir a string e o valor da	print	`Print("a", a);`	

variável no ecrã (com mudança de linha)			
Ler inteiro do ecrã	read	A=read;	

Figura 1. Operações e construções suportadas pela linguagem.

Programas na linguagem **ualg**

```
Count(word) {
   cont=0;
   n = 0;
   While(n < 32) {
      teste = word & 1;
      If(teste == 1) {
          cont = cont +1;
      }
       word = word >> 1;
       n = n + 1;
   }
   return cont;
}
```

Conta nº de bits a "1" do conteúdo da variável passada como argumento.

```
sqrt(vsqn) {
    vsq=vsqn;
    asq=0;
    a=0;
    tvsq=0;
    I=0;
    While(I < 6) {
      nasq1 = asq + a;
      nasq2 = nasq1 << 2;
       nasq = nasq2 | 1;
       sa = a << 1;
       tvsq1 = tvsq << 2;
       Vsq1 =  vsq >> 10;
       Vsq2 = vsq1 & 3;
        tvsq = tvsq1 | vsq2;
        vsq = vsq << 2;
        if(nasq <= tvsq) {
            a = sa | 1;
            asq = nasq;
        } else {
            a = sa ;
            asq = asq << 2 ;
        }
        I = i+1;
    }
    return a;
}
```

Calcula a raiz quadrada de um número inteiro. Retorna o valor inteiro da divisão.

Versão com instruções de entrada/saída:

```
Max() {
  A=Read;
  B=Read;
  Max=a;
  If(max < b) {
    Max = b;
  }
  Print("max: ", max);
  Return max;
}
```

Versão sem instruções de entrada/saída:

```
Max(a, b) {
  Max=a;
  If(max < b) {
    Max = b;
  }
  Return max;
}
```

Dois programas para determinar o valor máximo de duas variáveis

Figura 2.Exemplo de um procedimento na linguagem ualg.

Pretende-se que implemente um dos seguintes programas:

1. Interpretador para esta linguagem;
2. Compilador para esta linguagem que gere um procedimento em código MIPS (utilização do simulador **spim [4]** para validar os resultados) [5];

A nota final terá em conta os pesos dados a cada uma das tarefas a realizar. A contribuição de cada tarefa na nota global é apresentada na Figura 3.

A data de entrega do trabalho será definida durante as próximas semanas. Será realizada uma apresentação do trabalho efectuado por cada grupo. No inicio da apresentação deverá ser entregue um pequeno relatório a descrever as opções tidas durante o desenvolvimento e as restrições do compilador. O relatório deve incluir em anexo o ficheiro de entrada do JavaCC e do JJTree e a documentação do programa que deve ser gerada automaticamente pelo `javadoc`.

Tarefa	Percentagem na nota
Analisador sintáctico sem geração da árvore sintáctica (inclui especificação da gramática no JavaCC [1][2])	20%
Criação da árvore sintáctica com o JJTree [3]	20%
Geração de código/interpretação para/de sequências de instruções com expressões aritméticas	10%
Geração de código/interpretação para/de estruturas condicionais (if, if-else)	10%
Geração de código/interpretação para/de estruturas cíclicas (while)	10%
Optimizações ao nível do código *assembly* gerado (opção do compilador **–o**) ou tipos de interpretação (opções **–b** e **–p**)	10%
Código, apresentação do trabalho, etc.	20%
Melhor compilador e melhor interpretador (o melhor compilador poderá vir a ser utilizado na disciplina de Arquitectura de Computadores)	**Bónus:** 10%

Figura 3. Percentagem na nota do trabalho de cada tarefa.

2 Gramática da Linguagem

De seguida é apresentada a gramática da linguagem (esta gramática foi escrita de forma a ser o mais intuitiva possível):

IDENTIFIER = [a-z][a-z0-9]*

LITERAL = [0-9]+

STRING = “\““ [a-zA-Z0-9”:”” “]+ “\““

// os operadores têm o mesmo significado e precedência do que na linguagem Java

RELA_OP = “>“ | “<“ | “==“ | “!=“ | “>=” | “<=”

// os operadores têm o mesmo significado e precedência do que na linguagem Java

OTHER_OP = “*“ | “/“ | “&“ | “|“ | “<<” | “>>”

ADD_SUB = “+“ | “-“

LPAR = “(“

RPAR = “)”

VIRG = “,”

PVIRG = “;”

LCHAVETA = “{“

RCHAVETA = “}“

ASSIGN = “=“

WHILE = “while”

IF = “if”

ELSE = “else”

RETURN = “return”

PRINT=”print”

READ=”read”

Start → IDENTIFIER LPAR [Varlist] RPAR LCHAVETA Stmtlst RETURN IDENTIFIER PVIRG RCHAVETA

Varlist → IDENTIFIER {VIRG IDENTIFIER}

Stmtlst → {Stmt}

Stmt → Expr1 | Expr3 | Expr4 | Print

Expr1 → IDENTIFIER ASSIGN Rhs PVIRG

Rhs → Term ((ADD_SUB | OTHER_OP) Term)?

Term → IDENTIFIER | ((ADD_SUB)? LITERAL) | READ

Expr3 → WHILE Exprtest LCHAVETA Stmtsimple1 RCHAVETA

Expr4 → IF Exprtest LCHAVETA Stmtsimple2 RCHAVETA

Expr4 → IF Exprtest LCHAVETA Stmtsimple2 RCHAVETA ELSE LCHAVETA Stmtsimple2 RCHAVETA

Print → PRINT LPAR STRING VIRG IDENTIFIER RPAR PVIRG

Exprtest → LPAR IDENTIFIER RELA_OP (IDENTIFIER | ((ADD_SUB)? LITERAL)) RPAR

Stmtsimple1 → {Expr1 | Expr4 | Print}

Stmtsimple2 → {Expr1 | Expr3 | Print}

3 Dicas

A linguagem **ualg** não deve diferenciar minúsculas de maiúsculas. Para isso devem atribuir o valor verdadeiro à opção IGNORE_CASE do JavaCC:

```
options {
   IGNORE_CASE=true;
}
```

Lembrem-se que cada símbolo não-terminal da gramática pode ser um procedimento no ficheiro descritivo da gramática.

Durante a fase de validação do analisador gramatical podem utilizar uma makefile que execute o compilador com vários programas teste (programas em **ualg** que permitam verificar se

determinadas construções da linguagem estão a ser aceites pelo analisador gramatical). Isto permite-vos automatizar o teste da gramática.

4 Referências

1. JavaCC: http://www.experimentalstuff.com/Technologies/JavaCC/index.html

2. Oliver Enseling, "Build your own languages with JavaCC", Copyright © 2003 JavaWorld.com, an IDG company, http://www.javaworld.com/javaworld/jw-12-2000/jw-1229-cooltools_p.html (cópia local em pdf: http://w3.ualg.pt/~jmcardo/ensino/PS2003/AulaTP7/jw-1229-cooltools.pdf)

3. **[FICHA Nº 9]** Documento de introdução ao JJTree incluído na distribuição da ferramenta: http://w3.ualg.pt/~jmcardo/ensino/PS2003/jjtree.intro

4. Spim: http://www.cs.wisc.edu/~larus/spim.html

5. Rever as primeiras 3 aulas teóricas:

http://w3.ualg.pt/~jmcardo/ensino/PS2003/AulaT1/AulaT1.pdf

http://w3.ualg.pt/~jmcardo/ensino/PS2003/AulaT2/AulaT2.pdf

http://w3.ualg.pt/~jmcardo/ensino/PS2003/AulaT3/AulaT3.pdf