Compiladores — Trabalho Laborational

Pedro Vasconcelos, DCC/FCUP

Outubro 2020 (v1.0)

Descrição geral

Prentende-se que implementem um compilador básico para a linguagem *C0*, um dialeto da linguagem C desenvolvido na Universidade de Carnegie Mellon para ensino. O compilador deverá ler código fonte C0 e gerar código *assembly* MIPS.

Além deste documento deve ainda consultar a referência da linguagem C0 (c0-reference.pdf que contém descrições detalhadas da sintaxe e semântica (incluido regras de gramática).

Objetivos principais (80%)

- Tipos básicos (`int`, `bool`) e constantes (`true`, `false` e números inteiros)
- Expressões aritméticas: `+`, `-`, `*`, `/`, `%`
- Declarações de variáveis e atribuições simples: *var* `=` *expr*
- Operadores de comparação: `==`, `!=`, `<`, `<=`, `>`, `>=`
- Execução condicional: `if(`*expr*`)` *instr* / `if(`*expr*`)` *instr* `else` *instr*
- Blocos de instruções: `{` *instr* ... *instr* `}`
- Ciclos: `while(`*expr*`)` *instr*
- Definição de funções com argumentos e retorno de valores
- Funções para entrada e saída de inteiros: `scan_int()`, `print_int()`

Objetivos extra (20%)

Não é necessário implementar todos para ter cotação total. Será mais valorizada a qualidade do que a quantidade!

- Verificação de erros de tipos
- Operadores lógicos `!`, `&&` e `||` (com avaliação *short-circuit*)
- Ciclos `for`
- Controlo de fluxo em ciclos usando `break` e `continue`
- tipo `string`, constantes e uma função para impressão: `print_str()`
- *Arrays* de tipos básicos

Não necessita de implementar

Caraterísticas específicas do dialeto C0 que não existem na linguagem C: contratos, directivas `#use` e alocação dinâmica de memórica com *garbage collection*.

Programas de exemplo

```c
/* Calcular uma soma de quadrados. */
int main() {
  int s, n;
  s = 0;
  n = 1;
  while (n <= 10) {
    s = s + n*n;
    n = n + 1;
  }
  print_int(n);
}
```

```c
/* Testar se um inteiro positivo é primo. */
bool is_prime(int n) {
  int d;
  d = 2;
  if (n == 1)     // 1 não é primo
     return false;
  while (d <= n) {
     if (n%d == 0)
       return false;
     else
        d = d+1;
  }
  return true;
}

int main() {
  int n;
  n = scan_int();
  if (is_prime(n))
    print_str("prime");
  else
    print_str("not prime");
}
```

```c
/* Calcular factorial recursivamente */
int factorial(int n) {
   if (n == 0)
      return 1;
   return n * factorial(n-1);
}

int main() {
   print_int(factorial(read_int()));
}
```

Recomendações

- O trabalho deve ser realizado em grupos de dois estudantes
- Deve usar as técnicas estudadas nesta UC, nomeadamente decomposição em fases (análise lexical, sintática, semântica, geração de código intermédio e código máquina)
- Recomenda-se que utilize a linguagem Haskell e ferramentas *Alex* e *Happy* para geração de analisadores lexicais e sintáticos
- Pode usar outras linguagens programação desde que utilize técnicas equivalentes (exemplo: na linguagem C use *Flex* e *Bison*)
- Deve usar o *Github Classroom* para desenvolvimento e colaboração e como arquivo para as entregas do trabalho
- Decomponha o seu código em módulos lógicos seguindo a estrutura do compilador: um módulo `Lexer` para análise léxica, `Parser` para análise sintática, etc.
- Além do código do compilador deve ainda acrescentar ao repositório ficheiros de testes, i.e., ficheiro de exemplo de código fonte para testar as diferentes fases dos compilador (análise lexica, sintática e geração de código).
- Pense em casos de teste *positivos* e *negativos* — i.e. exemplos de código que deve ser aceite mas outros que deve ser rejeitados pelo compilador

Fases

Análise lexical e sintática (Apresentação na semana 16-21 novembro). Nesta fase o compilador deve apenas ler o código de um programa C0 e imprimir a AST (se estiver sintáticamente correto) ou terminar com erro.

Geração de código (Apresentação na última semana de 14-18 dezembro.) O compilador deve aceitar programas válidos e gerar código *assembly*. Para testar pode usar algum dos simuladores de MIPS (ver as referências).

Referências

- Página da linguagem C0: http://c0.typesafety.net/
- Descrição da linguagem C0: C0 Reference
- Documentação sobre MIPS: https://minnie.tuhs.org/CompArch/Resources/mips_quick_tutorial.html; ver também as aulas de Arquitetura de Computadores: https://www.dcc.fc.up.pt/~ricroc/aulas/1920/ac/
- Dois simuladores de MIPS: http://spimsimulator.sourceforge.net/; http://courses.missouristate.edu/kenvollmar/mars/

Pedro Vasconcelos, 2020.