

Compiladores — Trabalho Laboratorial

Pedro Vasconcelos, DCC/FCUP

Setembro 2022

Descrição geral

Prentende-se que implementem um compilador básico para um subconjunto *Pascal-0* da linguagem *Pascal*, uma linguagem imperativa fortemente tipada definida por Niklaus Wirth nos anos 70. O compilador deverá ler código fonte Pascal-9 e gerar código *assembly* MIPS.

Além deste documento deve ainda consultar a referência da linguagem Pascal-0 ([pascal0-reference.pdf](#)) que contém descrições detalhadas da sintaxe e semântica (incluído regras de gramática).

Objetivos principais (80%)

- Tipos básicos (`integer`, `boolean`) e constantes (números inteiros, `true`, `false`)
- Expressões aritméticas: `+`, `-`, `*`, `div`, `mod`
- Declarações de variáveis e atribuições simples: `var := expr`
- Operadores de comparação: `=`, `<>`, `<`, `<=`, `>`, `>=`
- Execução condicional: `if-then` e `if-then-else`
- Comandos compostos: `begin ... end`
- Ciclos `while`
- Definição de procedimentos e funções com argumentos e retorno de valores
- Funções para entrada e saída de inteiros: `readint()`, `writeint()`

Objetivos extra (20%)

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

- Declaração e uso de constantes
- Operadores lógicos `not`, `and` e `or` (com avaliação *short-circuit*)
- Ciclos `for`
- Controlo de fluxo em ciclos usando `break`
- O tipo `string`, constantes e uma função para impressão: `writestr()`
- *Arrays* de inteiros e booleanos
- Verificação de erros de tipos

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., ficheiros Pascal-0 de exemplo para testar as diferentes fases dos compilador (análise léxica, sintática e geração de código).

Fases

Análise lexical e sintática (Apresentação na semana de 7 novembro). Nesta fase o compilador deve apenas ler o código de um programa Pascal-0 e imprimir a AST (se estiver sintaticamente correto) ou terminar com erro.

Geração de código (Apresentação na semana de 12 dezembro.) O compilador deve aceitar programas válidos e gerar código *assembly* usando código intermédio de 3 endereços. Para testar pode usar algum dos simuladores de MIPS (ver as referências).

Referências

- 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/>
- Simulador de MIPS: <http://courses.missouristate.edu/kenvollmar/mars/>

Pedro Vasconcelos, 2022.