Projeto do Compilador

Objetivo

O objetivo deste projeto é adquirir experiência prática na implementação de linguagens de programação através da construção de um compilador funcional para uma simples linguagem procedimental, subconjunto inspirado na linguagem C, que inclui somente variáveis inteiras, array de inteiros, funções, condição, e repetição.

Avaliação

- **10%** Analisador Léxico
- **40%** Analisador Sintático/Semântico
- **50%** Gerador de Código

A Linguagem

A descrição da linguagem pode ser encontrada aqui.

Notas sobre Implementação

O compilador deve gerar código correto (não necessariamente otimizado) para o processador MIPS (32 bits). Detalhes da arquitetura MIPS aqui e do conjunto de instruções aqui.

Requisitos

- **Deve** ser implementado em C/C++
- **Deve** rodar no ambiente Linux/Unix
- *Pode* opcionalmente usar ferramentas automáticas (Lex/Flex & Yacc/Bison)

O programa do compilador deve receber dois parâmetros/argumentos (argv). O primeiro argumento será o arquivo do código fonte como entrada (e.g., sort.c) e o segundo argumento será o arquivo de saída (e.g., sort.asm) onde será gravado o código em assembly MIPS; por exemplo:

```
$ ./compilador sort.c sort.asm
```

Ferramentas
MATA61 Compiladores

Para executar e avaliar os resultados gerados pelos programas da linguagem alvo (MIPS assembly) iremos usar o simulador SPIM (detalhes no manual aqui). O SPIM (MIPS32 Simulator) pode ser baixado aqui. Se você usa Ubuntu Linux, pode instalar o SPIM pelo comando "**sudo apt-get install spim**".

Para saber mais sobre geradores automáticos de analisadores léxico e sintático, consulte os seguintes ponteiros:

- The Lex & Yacc Page
- Flex in a Nutshell
- Introduction to Bison
- Building Abstract Syntax Trees

Consulte aqui caso tenha dúvidas sobre como alocar memória em MIPS para variáveis (globais e locais) e arrays dinamicamente (Heap).

Entrada/Saída

A linguagem deve suportar as seguintes funções de entrada e saída:

- `int input(void) {...}` - não recebe nenhum parâmetro e retorna um valor inteiro do dispositivo de entrada padrão
- `void println(int x) {...}` - exibe o valor inteiro passado como parâmetro na saída padrão, seguido de uma quebra de linha (\n).

Tais funções não são definidas na gramática da linguagem, e sim como parte do ambiente de execução da linguagem. Para implementá-las, você deve usar *system calls* específicas ("print_int" e "read_int") disponíveis no SPIM (veja detalhes aqui).

Parte 1: Analisador Léxico

Parte 2: Analisador Sintático/Semântico

Parte 3: Gerador de Código