Proposta de projeto 2 - compilador de C--

1° Quadrimestre de 2021

1 Introdução

Este projeto é uma introdução a Compiladores. Para entender corretamente o que deve ser feito deve-se entendera a dinâmica de um compilador real, tudo isso será aprofundado na disciplina Compiladores em algum momento no futuro.

2 Base de Compiladores

Um compilador é qualquer código que seja capaz de traduzir uma linguagem-fonte em uma linguagem-objeto. Quando tanto a linguagem-fonte quanto a linguagem-objeto são de baixo nível, o compilador recebe o nome especial de montador (como o que você utilizou na última parte dos laboratórios) e quanto ambas são de alto nível recebe o nome especial de tradutor. A utilidade básica de um compilador (esperamos que vocês já saibam disso) é transformar um código de entrada (ASM por exemplo) em um código que possa ser interpretado pela máquina (MVN por exemplo), porém existem outras motivações para se escrever um montador, como para se aplicar a técnica chamada "bootstrap", que serve para obter compiladores sem passar pela dificuldade de codificar em linguagens de baixo nível.

A estrutura macroscópica de um compilador, no geral, tem 3 componentes principais:

1. Analisador léxico: este componente é responsável por verificar se todas as palavras escritas no código recebido são coerentes e não infringem as regras da linguagem-fonte e registrar identificadores e variáveis. é representado por uma máquina de estados finitos.

2. Analisador sintático: este componente é responsável por verificar se as sequências de palavras no código recebido são coerentes e não infringem

as regras da linguagem-fonte. é representado por uma máquina de estados finitos.

3. Analisador semântico: este componente é responsável por traduzir as funções da linguagem-fonte para a linguagem-objeto e escrever o arquivo de saída com o código final.

Um compilador também pode gerar mensagens de erro e aviso e escrever um arquivo de log para registrar informações importantes da compilação (como os erros e avisos). Para entender melhor todos estes processos vale ler os capítulos 1 e 5 do livro do prof. João José Neto "Introdução à Compilação".

3 A linguagem C--

Esta liguagem que vamos utilizar aqui é uma simplificação da linguagem C (assumimos que vocês tem familiaridade com ela) com apenas algumas funcionalidades simples. O C-- suporta dois tipos de variável, int (2 bytes) e char (1 byte). Ele permite atribuir à variáveis valores, resultados de expressões ou outras variáveis do mesmo tipo. Caso a variável seja tipo char, as operações são realizadas com seus valores em ASCII e depois convertidas em char novamente. Atribuição de valores é feita com o símbolo = .Cada linha de código deve terminar com ; e comentários de linha inteira podem ser feitos utilizando a notação. O C-- ainda permite a definição de rótulos à linhas da mesma forma que fazemos em ASM, "rótulo:" seguido do código ao qual se deseja atribuir aquele rótulo (os rótulos são utilizados para retorno e avanço dentro do código).

As palavras reservadas do C-- são:

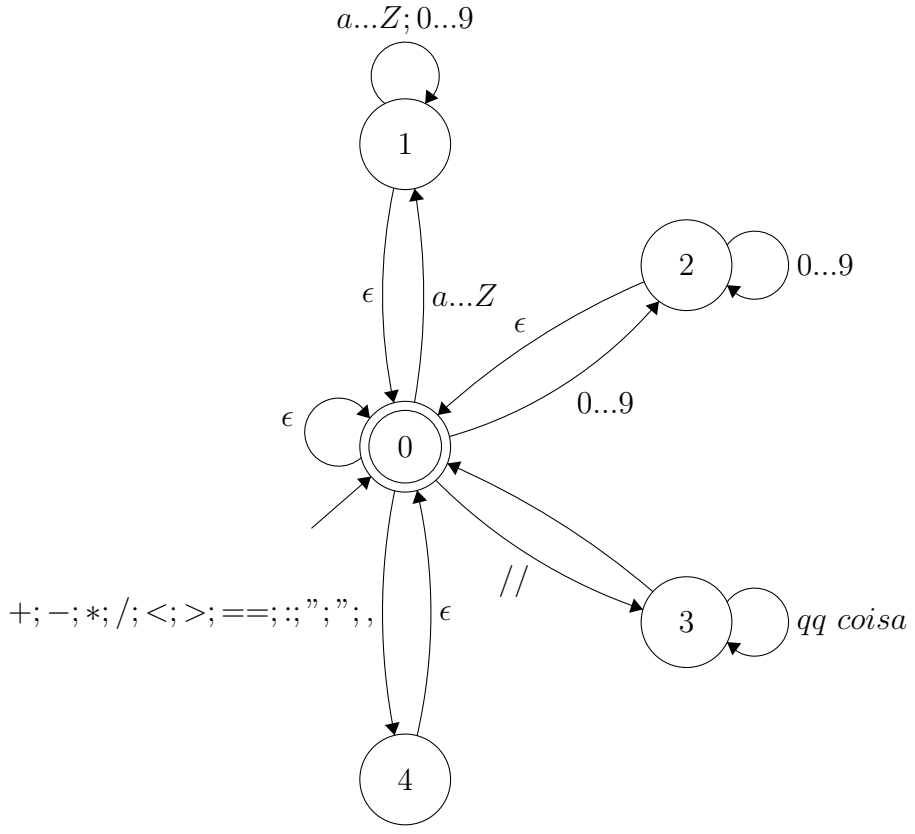
- int: representa o tipo inteiro de 2 bytes.
- char: representa o tipo caracter de 1 byte.
- if: representa execução condicional.
- goto: representa um pulo para outra linha definida por um rótulo.
- scan: representa comando de leitura do teclado.
- print: representa comando de escrita na tela.

Os operadores aceitos no C-- são:

- +: representa a operação de soma.
- -: representa a operação de subtração.
- *: representa a operação de multiplicação.
- /: representa a operação de divisão inteira.
- <: representa o comparador menor.
- >: representa o comparador maior.
- ==: representa o comparador igual.

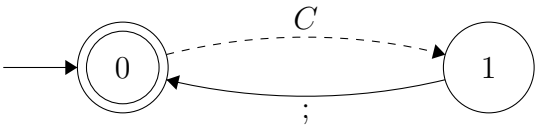
Para simplificar o trabalho, deixamos aqui as máquinas de estado finito que serão necessárias para implementação.

Para o analisador léxico utilize a seguinte máquina (ϵ significa qualquer forma de espaço em branco):

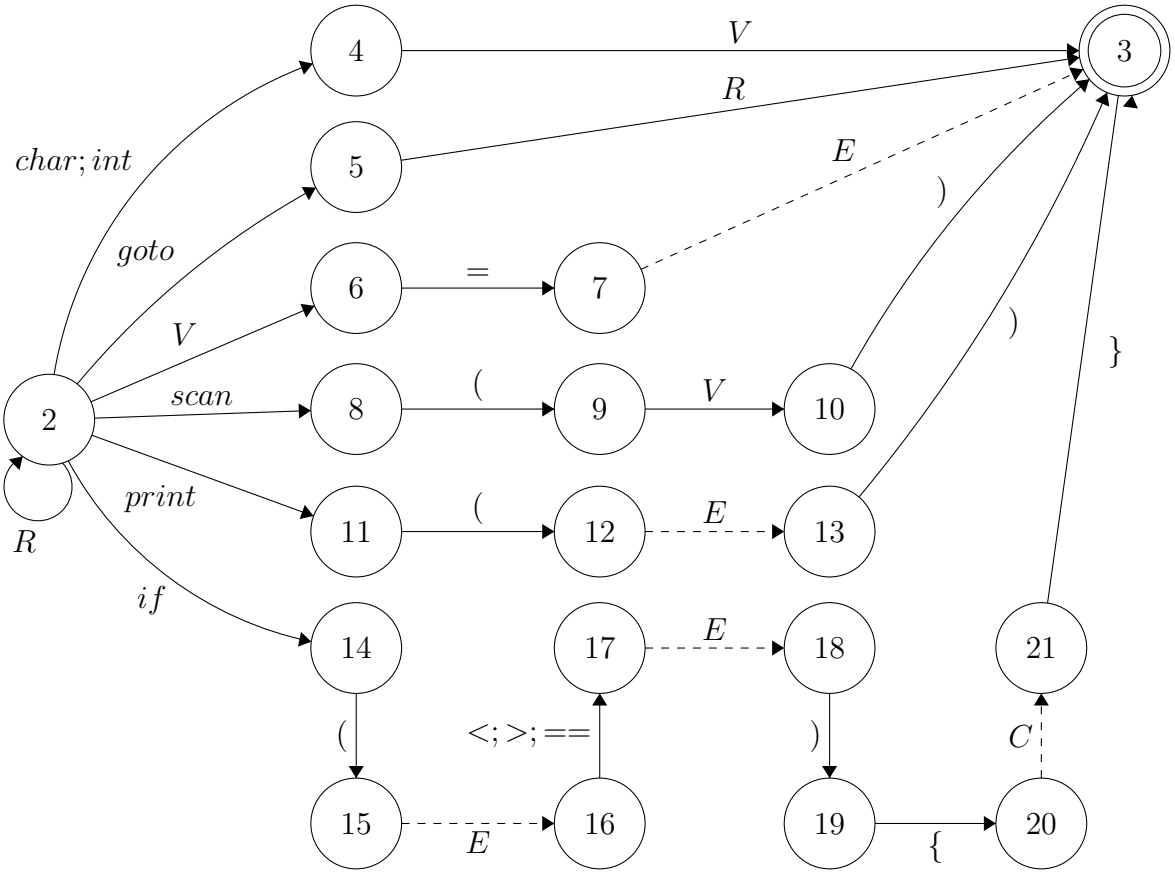


Para o analisador sintático utilize a seguinte máquina (C representa "Comando", R "rótulo", V "variável", E "expressão e N "número", os outros símbolos representam a si mesmos"):

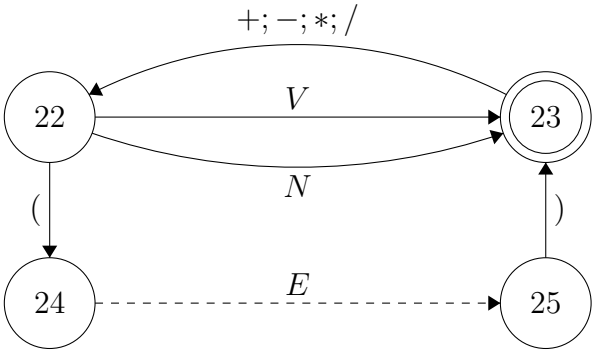
Máquina para um programa completo:



a Máquina para um comando:



Máquina para uma expressão:



4 O trabalho

O trabalho consiste em implementar um compilador de C--, isto é, criar um código que leia um arquivo em C-- e o traduza para ASM, seguindo as regras de codificação de C-- colocadas acima. O código a ser compilado será colocado no arquivo "code.cmm" e o código compilado em "code.asm" e importado da mesma forma que importamos arquivos durante toda a disciplina (definindo no arquivo "disp.lst"). Definir escopo de uso do compilador, mensagens de erro coerentes, eventuais limitações e funcionalidades não comentadas acima FAZEM parte do trabalho e pode ser feito (desde que justificado no relatório) em parceira com um grupo realizando a proposta 3.

Desafio: Ponteiros são uma ferramenta muito útil em códigos. Um ponteiro é uma estrutura que armazena não o valor de uma variável, mas o endereço em que está guardada. Implemente supurte a ponteiros no seu compilador. Para referenciar o endereço de uma variável use o símbolo & e para referenciar o valor armazenado em uma variável apontada o símbolo * (cuidado: seu compilador não pode confundir esta notação com o operador de multiplicação).

O trabalho pode ser feito em duplas ou individualmente.

5 Perguntas

Seguem algumas perguntas a serem respondidas depois da codificação:

1. Os algoritmos propostos podem ser mais eficientes? Como? Se sim, por que a forma menos eficiente foi escolhida?

2. Os códigos escritos podem ser mais eficientes? Como? Se sim, por que a forma menos eficiente foi escolhida?

3. Qual foi a maior dificuldade em implementar o compilador?

4. O que teria que ser mudado no seu código para poder suportar definição de funções? E para suportar o uso de arrays (tanto de int como de char)?

6 Entrega

Dois ou mais arquivos devem ser entregues:

1. Compilador: um arquivo em ASM chamado "g--.asm" contendo o código do compilador de C--;

2. Relatório: um arquivo chamado "relatorio_NUSP1_NUSP2.pdf" (acho que não preciso explicar a que é NUSP1 e NUSP2) contendo uma descrição resumida do problema e dos conceitos e uma descrição detalhada das etapas de resolução, da estratégia utilizada em cada módulo e outras informações que julgarem úteis. O relatório pode conter imagens das execuções de teste;

3. Outros arquivos que julgar necessário.

7 Avaliação

- 4.0 - Funcionamento do código
- 3.0 - Relatório, sendo 2.5 para completude e 0.5 se for feito em LaTeX
- 1.0 - Clareza do código
- 1.0 - Bom uso das ferramentas disponíveis na MVN
- 1.0 - Implementação do desafio
- EXTRA: 0.5 - Direto na média, a ser atribuída se o trabalho for apresentado junto ao de outro grupo que tenha feito a proposta 3 e ambos completamente integrados