

Compiladores I

Nome: Ivanir Paulo Cardoso Ignacchitti

2017093437

Nome: Gustavo Ribeiro Alves Rodrigues

2018130972

● Introdução

Nesse trabalho prático foi feita a implementação de um montador (assembler) de uma linguagem assembly simples.

Um montador(assembler) é um programa que traduz a linguagem assembly para a linguagem de máquina (binários). O programa recebe como entrada um arquivo de texto contendo o código fonte da linguagem a ser traduzida e tem como saída um arquivo contendo código em linguagem de máquina.

● O Problema

Como neste trabalho estamos trabalhando com uma versão simplificada de um montador, o arquivo de entrada consiste em uma linguagem assembly simples.

Cada instrução pode conter 4 campos sendo eles:

<label:> <operador> <operando> <;comentário>

Sendo o campo **<operador>** o único obrigatório.

Além disso, o campo operador pode ser uma instrução ou uma pseudo-instrução que foram definidas no escopo do trabalho.

● Implementação

Foi implementado um montador em 2 passos.

No primeiro passo, o arquivo de entrada é lido e a partir das instruções gerada uma tabela de símbolos que é preenchida com todas as instruções que possuem uma label.

No segundo passo, o arquivo de entrada é lido novamente, porém agora os operandos são alocados. Além disso, também é feita a contagem do número de instruções e é calculada a posição da memória das instruções que possuem uma label como operando. Por último é gerado um arquivo de saída no formato necessário para o teste.

Os campos das instruções devem ser separados por espaço como sugerido na seção anterior. Além disso, o campo comentário deve ser infixado com o caractere ; e o campo label deve ser pósfixado com o caractere : .

O montador foi implementado na linguagem **c++** com o auxílio da biblioteca **stl**.

O código fonte do montador é formado por uma classe principal chamada **Assembler**. Nesta classe foram implementados os métodos **firstpass** e **secondpass**, eles representam os passos de um montador two pass.

Outro método implementado por essa classe é o método **run**, que executa todo o montador gerando assim o arquivo de saída.

Falando em arquivo de saída, a classe **Assembler** possui o método **saveFile**, responsável por fazer a criação e a escrita da tradução no arquivo de saída.

Classes **MOT** e **POT** foram implementadas para representarem a tabela de instruções e de pseudo-instruções.

Uma outra decisão do projeto foi implementar uma classe chamada **Instruction**.

Essa classe é responsável por settar os campos das instruções antes de serem enviadas para os métodos das classes **MOT** e **POT**.

Duas classes foram implementadas para a tabela de símbolos. A primeira classe é a classe símbolo que possui 2 atributos, label que representa a label da tabela e a value, que representa a memória que é usada pela instrução + operando caso possua. Já a segunda classe é a própria tabela de símbolos que guarda cada símbolo quando necessário durante a primeira passada do algoritmo.

Por último a classe **FontCode** é responsável pela leitura do código fonte e as manipulações necessárias dos seus caracteres. Nela possuímos os metodos para ler uma linha e para separar uma linha pelos seus espaços.

● Extras

A pilha se move de acordo com as instruções que alteram a variável sp, que começa setada no top da pilha.

As posições de memória não podem ser negativas.

Por mais que a memória esteja implementada ela não aparenta estar 100% funcional e por isso parte dela está comentada.

- **Como Compilar e Testar**

No diretório principal do código, existe um Makefile pronto para ser usado.

- Abra o terminal no diretório principal do projeto.
- Execute o comando make.
- Um arquivo binário foi gerado com o nome *Assembler*
- Basta usar o comando ***./assembler input output***
- ***input:*** variável que representa o arquivo de entrada, ou seja, o código fonte.
- ***output:*** variável que representa o arquivo de saída.