Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Compiladores (DCC010/DCC053) – 2021/1º

Trabalho Prático 1 – Montador

Guilherme de Abreu Lima Buitrago Miranda - 2018054788
Victor Hugo Silva Moura - 2018054958

# Introdução

O trabalho prático 1 da disciplina de Compiladores tem como objetivo a criação de um montador para uma máquina previamente montada, a fim de treinar e fixar os conceitos aprendidos em sala de aula na disciplina relacionados ao processo de montagem de um programa.

Nas seções a seguir, encontram-se detalhes relativos à implementação do trabalho, testes e conclusão.

# Implementação

Para implementação deste trabalho, foi utilizada a lógica de tradução em dois passos com geração de código intermediário.

Sendo assim, a principal ideia utilizada foi fazer a primeira passagem pelo programa reescrevendo as instruções. Para essa passagem, foi criada uma variável responsável por guardar o ILC do programa e, com isso, cada instrução é decodificada e o tamanho dela é adicionado ao ILC. No caso de um registrador, ou de uma posição de memória (em uma instrução de memória), esses valores são escritos diretamente no código intermediário. Porém, ao chegar em alguma instrução marcada por um label, esse label é escrito em uma tabela, que foi armazenada em forma de hash, onde o nome do label é a key para o hash. Nessa posição, foi armazenado o ILC atual, que indica a posição inicial daquele label no programa. Além disso, ele não é escrito no código intermediário. Sendo assim, ao final da primeira passagem, temos o código intermediário, que é um cópia do programa original, sem labels e comentários, e uma tabela hash que contém o ILC de cada label encontrada no programa original.

Na segunda passagem, o código começa escrevendo o cabeçalho do arquivo final e em seguida escreve o tamanho do programa (indicado pelo ILC), endereço de carregamento, valor inicial da pilha (com base no endereço de carregamento e ILC) e ponto de início do programa. Após isso, inicia-se a tradução do código intermediário. Esse código é traduzido instrução por instrução, seguindo a tabela fornecida na especificação do trabalho. Além disso, a variável ILC é zerada no começo da leitura e é acrescida de 1 para cada instrução, registrador, inteiro (após a palavra WORD) ou posição de memória lidos. No entanto, ao encontrar uma instrução que não está na tabela de instruções, e nem na tabela de registradores, o montador inicialmente testa se a instrução encontrada foi a instrução WORD (essa instrução não foi colocada na tabela de instruções do programa). Caso não seja, sabemos que é uma posição de memória ou um inteiro após a instrução WORD. Sendo assim, inicialmente tenta-se fazer um parse desse número para inteiro. Caso o parse

funcione, o número é escrito no programa final. Caso contrário, sabemos que é uma posição de memória, indicada pelo label. Dessa forma, tentamos recuperar o label na tabela hash. Se a recuperação for bem sucedida (o label existir), é feito um cálculo da diferença do ILC atual para o ILC guardado no hash e essa diferença é escrita no programa final. Essa diferença indica o salto que o PC deveria dar para encontrar o dado contido no label. Se não for bem sucedida, nada é escrito. Ao finalizar a leitura de instruções, o montador imprime um caractere de quebra de linha e encerra sua execução.

## Testes

Para testar o programa, além do arquivo de teste ex.amv fornecido, foram desenvolvidos outros testes, realizando a soma, subtração, multiplicação e divisão de números. Executando tais testes “na mão” e comparando tais execuções com o código desenvolvido, ficou constatado que este estava em bom funcionamento.

Posteriormente, testamos um programa mais complexo, que testa um loop de 1 a N, conforme código contido dentro da pasta *tst*.

Por fim, desenvolveu-se um código de testes que testa a geração de uma sequência de Fibonacci, conforme sugerido no enunciado. Como esperado, o código também apresentou um comportamento adequado para o exemplo em questão e, portanto, deu-se a etapa de testes como concluída.

## Conclusão

Para além da prática com a linguagem C++, a implementação do trabalho foi bastante proveitosa para compreender efetivamente os conceitos vistos até o presente momento na disciplina.

O aprendizado se deu, sobretudo, na resolução de pequenos detalhes da implementação como decidir qual das versões do montador de dois passos seria montada. Além disso, foi bastante interessante a execução dos testes, e outras dúvidas dos colegas apresentadas no Microsoft Teams. Com isso, considera-se que o trabalho foi de grande proveito para o processo de aprendizagem.