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

Trabalho Prático 2 – Ligador

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

Introdução

O trabalho prático 2 da disciplina de Compiladores tem como objetivo a criação de um ligador para uma máquina previamente montada, a fim de treinar e fixar os conceitos aprendidos em sala de aula na disciplina relacionados ao processo de ligação 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 uma lógica parecida com a do montador, em dois passos com geração de código intermediário.

O primeiro passo foi adequar o montador para imprimir a label no código final caso sua declaração não fosse encontrada no mesmo código. Além disso, após a linha com as instruções traduzidas, é impressa a tabela de símbolos do programa. O formato de impressão contém um símbolo por linha, com seu respectivo ILC em sequência.

Sendo assim, a principal ideia utilizada foi fazer a primeira passagem reescrevendo as instruções traduzidas pelo montador. Porém, antes disso, é feita uma passagem para descobrir algumas informações importantes, como o número total de instruções do programa final, e qual programa é o principal (contém a label *main*). Para essa passagem, foi criada uma variável responsável por guardar o ILC total de todos os programas e, com isso, o tamanho de cada programa é adicionado ao ILC. Além disso, é conferido se o programa que estamos inspecionando atualmente contém a label main. Caso sim, o nome do arquivo desse programa é armazenado de forma que seja possível saber qual é o programa principal.

Após isso, passamos pelo programa reescrevendo as instruções em um arquivo intermediário. As labels contidas na tabela de símbolos de cada programa são armazenadas em uma tabela de símbolos global no ligador. Porém, para manter as referências corretas, foi criado um contador dinâmico que armazena o ILC inicial do programa atual. Com isso, o ILC da label no programa original é acrescido desse contador dinâmico e, só então, é escrito na tabela de símbolos global. Dessa forma, podemos ter uma referência de todas as labels que foram declaradas por todos os programas a serem ligados. Para essa escrita, no entanto, optamos por colocar o código do main ao final. Dessa forma, podemos evitar que o código final esteja em uma ordem onde, após a execução do main, ele execute o código de outras partes sem que elas sejam devidamente chamadas.

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 leitura do código intermediário. Ao início do processo, uma nova variável ILC é criada e é acrescida de 1 para cada instrução lida. No entanto, ao encontrar um valor que não é um número inteiro, 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 na tabela de símbolos 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, o label é escrito novamente. 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*. 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 ordem seria adotada para a ligação dos códigos passados para o programa. 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.