Universidade Federal de Minas Gerais
Departamento de Ciência da Computação
Compiladores

# Trabalho Prático 2
# Ligador

Alunos: Jackson Nunes Silva
Júlia Fonseca de Sena

Professor: Renato Antônio Celso Ferreira

Julho
2021

# Conteúdo

# 1 Introdução

O objetivo do presente trabalho foi implementar um ligador para uma Máquina Virtual predefinida, complementado pelo montador de dois passos previamente implementado. A entrada é a saída de tal montador contendo a tabela de símbolos e o programa traduzido e a saída padrão é um arquivo executável no formato aceito pela máquina virtual. É uma forma de fixação e concretização de conceitos vistos em sala de aula de modo mais abstrato, observando o processo de montagem em prática.

# 2 Implementação

Ambos montador e ligador foram implementados na linguagem C e para rodá-los corretamente, deve-se utilizar inicialmente o comando *make*, que gera os executáveis *montador.exe* e *ligador.exe* na pasta **bin**. Portanto, os comandos de execução são:

⟨bin/montador arquivo_de_entrada⟩

⟨bin/ligador arquivo1_de_entrada arquivo2_de_entrada...⟩

## 2.1 Montador

Grande parte da implementação do montador de dois passos foi aproveitada do trabalho prático anterior. As únicas alterações feitas foram no formato da saída, imprimindo-se primeiro a tabela de símbolos, depois o tamanho do programa e a tradução feita pelo montador (para *labels* referenciados,isto é, fora suas declarações, é colocado seu nome ao invés da posição na memória e antes de constantes definidas pela pseudo-instrução **WORD**, é escrito um indicador homônimo). Esse programa recebe apenas um arquivo por vez.

## 2.2 Ligador

O ligador começa lendo as tabelas de símbolos de todos os arquivos de forma a juntá-las em uma única. Lê o *label* e checa se está na tabela do ligador. Se não estiver, checa qual seu endereço na tabela do arquivo (inteiro depois do *label* na entrada). Se for -1, é uma referência à outro arquivo, então é adicionado com -1. Caso for diferente de -1, coloca-se o endereço lido somado ao tamanho do programa (o qual também é recebido na saída do montador: soma-se o valor do programa do arquivo quando encontra-se a palavra "size:").

Se estiver na tabela de símbolos do ligador, só é alterado o endereço quando é diferente de -1 na tabela do arquivo e igual a -1 na do ligador. A lógica é a mesma, soma o endereço lido ao tamanho do programa. Ao encontrar a palavra "prog:"é o início da tradução do programa, então passa-se para os próximos arquivos. Por fim, imprime-se o necessário para o funcionamento da máquina virtual, sendo o endereço de carregamento e início da pilha fixados respectivamente em 1000 e 999, o tamanho do programa calculado com a soma do tamanho de cada arquivo, e o valor inicial de PC como o endereço do *label* **main** na tabela de símbolos construída.

No próximo passo, reabre-se todos os arquivos e pula para a linha após "prog:". Como só são colocados os *labels* quando são referenciados e não declarados, são somente utilizados em alguma instrução, logo a primeira palavra lida deve ser um inteiro. A única exceção disso é quando a primeira palavra é **WORD**, a qual é impressa para evitar confusões entre instruções e constantes. Se for **WORD**, imprime-se somente a constante seguinte à palavra.

Caso contrário, lê-se o inteiro, imprime-o, e identifica-se a que instrução corresponde. Lê-se os registradores e/ou memória necessários de acordo com a instrução, sendo a memória representado por um label cujo endereço na tabela de símbolos subtraído do endereço atual deve ser impresso (já que é relativo ao PC). Para os registradores, imprime-se o inteiro de 0 a 3 a que corresponde.

## 3 Testes

Para testar a codificação do montador foram implementados dois programas no Assembly da máquina virtual dada, que se encontram na pasta **tst**. Tais testes foram exatamente os sugeridos na especificação do trabalho prático 1, contudo, foram adaptados para serem implementados via múltiplos arquivos. Ou seja, um de retornar o n-ésimo número da seqûencia de Fibonacci de um $n$ dado na entrada (cujos nomes são *main_fibo.amv* e *func_fibo.amv*) e outro para retornar a mediana entre cinco números fornecidos (de nomes *main_med.amv*, *n2_med.amv*, *n3_med.amv*, *n4_med.amv*, *n5_med.amv*). Em suma, a lógica de cada um é a seguinte:

**Fibonacci:** Lê um inteiro da entrada e caso seja menor que 3, escreve 1 na saída. Caso contrário, inicia um *loop* subtraindo $n$ a cada iteração e, utilizando dois registradores e a pilha, realiza a soma de fibonacci. Os registradores R1 e R2 são iniciados com 0 e 1 inicialmente e o valor de R2 é empilhado, em seguida é feita a soma, armazenada no registrador R2. Assim, R1 desempilha o valor armazenado (correspondente ao antigo valor de R2) e o novo valor de R2 é empilhado (contendo a soma). Isso se repete até que $n$ seja 0 e quando isso ocorre, valor de R2 é escrito na saída. Nesse trabalho, foi dividido em dois arquivos, um com o *main* e o outro com a impressão e o *loop*

**Mediana:** Lê os cinco inteiros da entrada, colocando-os ordenadamente em cinco posições da memória. Primeiro compara o primeiro e o segundo número, colocando-os em ordem, e, após ler cada um dos outros, checa se os lidos anteriormente são maiores ou menores. Se for maior, o desloca para a posição de cima, se for menor, coloca o novo na posição acima dele e passa para leitura do próximo número. Por fim, imprime o número que se encontra na terceira posição, o qual é a mediana. Para testar a máquina, dividiu-se as comparações e leituras de novos números em 4 novos arquivos, além do *main*, cada arquivo correspondendo a um.

Além disso, foi utilizado o teste dos arquivos fornecidos pelo professor/monitor.

## 4 Conclusão

A implementação desse trabalho foi uma boa forma de concretização de como funcionaria a ligação de múltiplos arquivos, após sucesso no elaboração do montador no trabalho anterior. Não houveram grandes empecilhos com os conceitos solidificados em aula e com o complemento da lógica do montador de dois passos. Assim, a experiência obtida desse trabalho foi positiva.