# Trabalho Prático 2 – Compiladores 1

Departamento De Ciência da Computação – UFMG – 2021/1

Daniel Souza de Campos – 2018054664

Letícia da Silva Macedo Alves – 2018054443

## Sumário



## 1 Introdução

O objetivo desse trabalho é desenvolver um Editor de Ligação (Linker) como continuação do Trabalho Prático 1 no qual foi desenvolvido apenas um Montador (Assembler).

A função do Editor de Ligação é conseguir gerar um arquivo executável a partir de vários arquivos diferentes que foram gerados pelo Montador. Desse modo, ele deverá alocar o espaço de memória total necessário pelo arquivo executável a partir desses vários arquivos e resolver possíveis referências a Labels definidos em outros arquivos (Problema da Referência Externa).

## 2 Ambiente de Desenvolvimento

O trabalho foi desenvolvido em máquinas com sistema operacional Windows, na linguagem C++11 com o compilador g++. Usamos também como ferramenta de desenvolvimento o Visual Studio Code.

Além disso, foi utilizado o GitHub como sistema de versionamento de código e o trabalho está disponível em https://github.com/Pendulun/Linker.

# 3 Desenvolvimento do Linker

No Trabalho Prático 1, o Montador ficou encarregado de gerar o arquivo final, com extensão *mv,* a ser executado pela máquina virtual a partir de um único arquivo *amv* de entrada com instruções Assembly. Naquele trabalho, assumimos que todos os Labels referenciados no arquivo foram definidos nesse mesmo arquivo. Além disso, não era necessário ter um Label de nome “main” para ser identificado como ponto de partida. Dessa forma, o arquivo executável gerado estava pronto para ser enviado para a máquina virtual.

Nesse trabalho, entretanto, o Montador deve ser capaz de receber vários arquivos com extensão *amv* de uma vez e deve gerar um arquivo intermediário, com extensão *txt*, para cada arquivo de entrada. Esses novos arquivos de saída serão a entrada do Linker.

Além disso, não deve ser obrigatório que os Labels referenciados em um arquivo *amv* estejam definidos nesse mesmo arquivo e, em algum arquivo de entrada, deve estar definido um Label “main” para ser o ponto de partida do programa. Portanto, algumas alterações no Montador foram realizadas.

## 3.1 Alterações no Montador do TP1

Como dito antes, deve-se gerar um arquivo *txt* para cada arquivo de entrada. Além disso, devemos tratar a situação na qual um Label foi referenciado, mas não está presente na Tabela de Símbolos daquele arquivo e pensar em todas as informações necessárias para o funcionamento correto do Linker.

Dessa forma, os arquivos gerados pelo Montador têm a seguinte estrutura:

1. Tamanho total ocupado pelas instruções do arquivo
2. Linha em Branco
3. Identificador: “#INSTRUCOES”
4. Instruções do programa em inteiros
5. Linha em Branco
6. Identificador: “#TABELA”
7. Tabela de Símbolos do arquivo sendo que cada linha contém um par de chave-valor sendo esses separados por um “:”.

Durante o desenvolvimento do trabalho, notamos que ainda seria possível realizar a tradução das instruções Assembly em dois passos normalmente, mas, para tratar o Problema da Referência Externa, no passo 4, apenas imprimimos o nome do Label referenciado precedido pelo identificador “#”.

Como exemplo, imagine que um Label de nome “Label1” foi referenciado, mas nunca definido no mesmo arquivo. Dessa forma, ao invés de imprimirmos

juntamente com as instruções no passo 4, o endereço relativo para o seu acesso, imprimimos “#Label1”.

É importante notar que **assumimos que não existem Labels formados apenas de caracteres numéricos**.

Executar o Montador uma vez para cada arquivo *mv*, e não passar todos os arquivos de uma vez, não deverá influenciar no funcionamento do Linker.

## 3.2 Passos do Linker

A entrada do Linker será um conjunto de arquivos *txt* que foram gerados pelo Montador. **Assumimos que todos os arquivos necessários para gerar o executável final serão passados de uma vez como entrada para o Linker**.

A saída do Linker será um arquivo de nome *Executavel.mv* e ele funciona em dois passos.

### 3.2.1 Leitura da Tabela de Símbolos e Tamanho dos Arquivos

O primeiro passo do Linker é ler cada arquivo passado como argumento e anotar o seu tamanho e a sua Tabela de Símbolos.

Ao anotar o tamanho de cada arquivo, teremos o tamanho total necessário pelo arquivo executável. Além disso, saber o tamanho de cada arquivo ajudará a calcular o endereço absoluto inicial na memória onde as instruções do arquivo estarão.

Com a Tabela de Símbolos dos arquivos, montamos uma Super Tabela de Símbolos que contém todos os símbolos definidos ao longo de todos os arquivos e os seus endereços absolutos de definição.

Importante dizer que **assumimos que não existem dois Labels com o mesmo nome, todos os Labels referenciados foram definidos em algum momento e que existe um único Label chamado “main**”.

Como calculamos o endereço absoluto de todos os Labels presentes na Tabelas de Símbolos, teremos o endereço absoluto do Label “main” e esse será o *entry-point* do programa.

### 3.2.2 Leitura das instruções dos arquivos

O segundo passo do Linker é uma nova leitura em cada arquivo, mas, dessa vez, procuramos a seção das instruções.

Nessa parte, temos em posse a Super Tabela de Símbolos que contém todos os Labels definidos ao longo do programa e seus respectivos endereços

absolutos de definição em memória. Além disso, também temos o endereço absoluto de início na memória relativo a cada arquivo diferente. Dessa forma, caso esse endereço inicial para um arquivo “Arq1.txt” for 20, os endereços das instruções desse arquivo começarão na posição 20.

Basicamente, o que essa parte faz é ler sequencialmente as instruções de um arquivo e escrever uma cópia dessas instruções no arquivo executável na sua seção de instruções. Entretanto, ao identificar a presença de um “#” como primeiro caractere de uma instrução, sabemos que ali está uma referência a um Label externo como explicado na seção 3.1.

Para tratar esses casos de referência externa, fazemos o cálculo do endereço relativo com relação ao Label e o imprimimos no arquivo executável.

Exemplo: Sabe-se que a posição absoluta atual é 20. Identificamos que a próxima instrução é uma referência da seguinte maneira “#Label1”. Procuramos o endereço absoluto de “Label1” na Super Tabela de Símbolos e achamos “50”. Calculamos o endereço relativo “20 - 50 – 1 = -31” e o escrevemos no arquivo executável.

Dessa forma, realizamos duas passadas em cada arquivo de entrada no Linker e geramos o arquivo de saída executável.

### 3.2.3 O arquivo executável

O arquivo de saída do Linker se chama “Executavel.mv”. Ele tem a seguinte estrutura:

1. MV-EXE
2. Linha em Branco
3. Linha com quatro inteiros separados por espaço
    a. **Tamanho do programa**: Quantas posições de memória o programa vai ocupar. É o valor da soma dos tamanhos totais ocupado pelas instruções de cada arquivo *txt* gerado pelo Montador;
    b. **Endereço de carregamento**: Posição que definimos 0 por padrão;
    c. **Valor inicial da pilha**: **Mil posições de memória depois relativa à última posição de memória ocupada pelo programa**: Endereço de carregamento + Tamanho do Programa + 1000;
    d. **Entry Point do Programa**: Endereço absoluto de memória do Label “main”.
4. Linha em Branco;
5. Sequência de inteiros representando a tradução do programa completo.

## 4 Testes Realizados

Para testar se o Assembler e o Linker funcionam corretamente, utilizamos os mesmos testes presentes no TP1. Com todos esses testes passamos por toda a tabela de instruções e conseguimos abranger todos os requisitos do trabalho.

Como exemplo, dividimos o TestePirâmide em dois arquivos.

O TestePirâmide recebe como entrada um número inteiro positivo e imprime, de forma ascendente, de 0 até esse número (inclusive) e, depois, de forma descendente, do número de entrada menos um até 0 (inclusive).

Dividimos esse exemplo em dois arquivos diferentes, separando os Labels que ele utiliza e definindo um Label “main”. O arquivo executável gerado pelo Linker, após passar antes pelo Assembler, foi executado normalmente pelo emulador.

# 5 Conclusão

Esse trabalho tinha como objetivo desenvolver um Editor de Ligação que realizasse parte do trabalho do Montador do TP1 além de permitir que vários arquivos utilizem mutuamente os Labels definidos entre si.

Acreditamos que esse objetivo tenha sido alcançado uma vez que, utilizando de tudo o que assumimos, conseguimos alterar o Montador e desenvolver um Linker que realiza a alocação, ligação e relocação a partir de vários arquivos de entrada produzindo um único arquivo executável pela máquina virtual.

Para atestar a corretude do trabalho, realizamos os testes mencionados na seção 4.

Continuamos com um Montador em dois passos e o Linker também necessita de dois passos. Não conseguimos pensar em uma forma de desenvolver o trabalho na qual o Linker só precisaria de um passo para gerar o executável.

A maior dificuldade que tivemos foi definir quais seriam as informações geradas pelo Montador que seriam necessárias pelo Editor de Ligação e como elas seriam utilizadas de forma eficiente para o funcionamento total do Linker.