TRABALHO PRÁTICO 4:
Ligador

Pedro Lopes Miranda Junior

[1]Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG)

plmj@dcc.ufmg.br

Resumo. *Este trabalho constitui na implementação de um ligador para a máquina virtual criada anteriormente.*

1. Introdução

Ao longo do curso foram estudados diversos conceitos sobre liguagem de máquina. Perante isso, foram dados diversos trabalhos que testaram alguns dos conceitos abordados.

Após a implementação de uma máquina virtual, de um montador e de um expansor de macros, é necessário um editor de ligação que faça a ligação entre os vários módulos de um programa em linguagem assembly e transforme-os em um único programa em linguagem de máquina.

O ligador lê o assembly dos arquivos passados por parâmetro e gera um código,em linguagem de máquina - para a máquina virtual -, com as instruções criadas anteriormente.

Esse passo fica antes da execução por parte da máquina virtual e após a transcrição para código em linguagem de máquina por parte do montador. O ligador gera uma tabela de símbolos, tabela esta identica aos passos anteriores.

2. Solução Proposta

Para solucionar o problema, foi criada uma estrutura de dados que chama linker (Lincador,Ligador) que contém a tabela de símbolos.

Já a tabela de símbolos guarda o nome e o valor apontado pela label.

2.1. Algoritmos

Foram implementadas algumas funções que auxiliam no funcionamento do montador, portanto todas as funções operam sobre o tipo linker.

2.1.1. Funções

1: Funções do ligador

initialize; *Inicializa o ligador*
join; *Une as tabelas de símbolos do arquivo principal com os demais módulos*
search; *Pesquisa por um label retornando o valor da linha do mesmo*
verify; *Verifica a existencia de um símbolo*
execute; *Executa o ligador transcrevendo a saída*

3. Implementação

Na implementação do problema proposto foram tomadas várias decisões, dentre elas criar um tipo de dados para o ligador dividindo o código em funções, de modo que na função principal fique o menor conteúdo possível, ajudando no encapsulamento do código e em futuras manutenções e melhorias.

Foi decidido criar um tipo de dados para o liagdor pois esse tipo de disposição facilita a adição futura de demais estruturas. Além disso o acesso ao TAD é feito de maneira melhor estruturada e o encapsulamento é melhor feito.

Além disso foi necessário fazer uma modificação no montador. Nessa modificação foram acresentadas duas novas estruturas ao TAD mounter (deslocamento e tamanho), além de imprimir a tabela de símbolos no arquivo gerado pelo montador. Isso é necessário pois o ligador precisa dessas informações para executar seu passo.

3.1. Código

O código foi dividido em arquivos *.c* e *.h* que estão listados abaixo

3.1.1. Arquivos .c

- **linker.c:** Contém a função principal do ligador;
- **linkerdata.c:** Contém as funções do tipo limker;

3.1.2. Arquivos .h

- **linkerdata.h:** Contém as definições das funções do tipo linker

3.2. Compilação

O programas deve ser compilado através de um makefile, chamando *ligador* ou através do compilador GCC chamando:

```
gcc -Wall linker.c linkerdata.c -o ligador
```

3.3. Execução

Para a execução do programa deverão ser recebidos,impreterívelmente, o nome dos módulos a serem ligados, o nome do arquivo principal e o arquivo de saída.

O comando para execução do programa é da forma:

```
./ligador <módulos> -m <main> -o <saída>
```

3.3.1. Formato da entrada

O arquivo de entrada citado deverá ser um programa em linguagem de máquina, onde cada linha conterá uma instrução, como abaixo, além de tabela de símbolos no final separado por **###** como abaixo:

```
01
5
01
4
81
-7
90
0
0
###
9 2
A 7
```

3.3.2. Formato da saída

O programa imprimirá no arquivo de saída o código em linguagem de máquina como se segue:

```
01
5
01
4
81
-7
90
0
0
```

4. Avaliação Experimental

O programa foi executado e testado numa máquina rodando o sistema baseado em debian Linux Mint. A máquina em questão tem uma memória de 8GB e um processador Core I7 de 2.2GHz.

Foram feitos vários testes de execução do programa,onde todas as funções testadas funcionaram de maneira rápida e sem erro. Abaixo apresentamos imagens da execução do programa e sua saída em modo verbose para um dos testes criados.

```
~/Projetos/sb/ligador/tp4_plmj]$ make ligador
gcc -o ./src_ligador/linkerdata.o -c ./src_ligador/linkerdata.c -Wall
gcc -o ./src_ligador/main.o -c ./src_ligador/linker.c -Wall
gcc -o ligador ./src_ligador/main.o ./src_ligador/linkerdata.o
```

Figura 1. Comando Make

```
~/Projetos/sb/ligador/tp4_plmj]$ ./ligador test/calculo.maq -m test/main.maq -o test/saida.maq
```

Figura 2. Comando de execução do programa

```
~/Projetos/sb/ligador/tp4_plmj]$ cat test/saida.maq
01
5
01
4
81
-7
90
0
0
```

Figura 3. saída usando verbose

5. Conclusão

O trabalho correu sem grandes problemas, sendo a parte mais difícil a criação dos progrmas de testes, pois montar alguns daqueles códigos em asembly é extremamente difícil.

O programa atendeu a diversos valores de entrada e creio que a solução proposta atenda ao especificado