TRABALHO PRÁTICO 4: Editor de Ligação

Felipe Moraes Gomes - felipemoraes@dcc.ufmg.br

1 Introdução e Definição do Trabalho

Este trabalho descreve a implementação e operação de editor de ligação para uma linguagem assembly hipotética, baseada no conjunto de instruções do RISC. O editor recebe como entrada múltiplos arquivos, em um formato intermediário, contendo somente o que ele precisa para efetuar a ligação.

O restante deste documento está organizado da seguinte forma: A 2ª seção trata da implementação do montador e de sua organização no código; A 3ª seção resume o formato de execução, entrada e saída do programa; A 4ª seção contém os testes realizados; A 5ª seção conclui o trabalho. Após isso, é colocado um apêndice contendo uma listagem dos arquivos do projeto.

2 Implementação e Organização

O editor funciona em 3 etapas. Primeiro, ele abre cada um dos arquivos objetos gerados pelo montador, e os carrega para a memória, a começar pelo main. Ao fazer isso, ele aloca o espaço que cada arquivo ocupará no executável gerado, gerando os offsets. Em seguida, a linkagem é feita, onde as referências irresolvidas são buscadas nos outros arquivos. Não havendo êxito na busca, o editor emite um erro. Finalmente, os arquivos fontes são realocados para o arquivo final, já com as dependêncidas resolvidas, e o executável é gerado.

2.1 Dados e Variáveis

- *Object*: Estrutura contendo o início e o fim do objeto; a tabela de referências conhecidas; a tabela de todas as referências; o tamanho do programa e o programa antes da alocação e linkagem.

2.2 Métodos e Funções

  - *void leituraObjetos(nome do arquivo de entrada, lista de estruturas objeto)*: Lê os dados dos módulos e monta cada estrutura objeto.

– *void alocacao(lista de estruturas objeto, inteiro com o número de objetos)*: Aloca o espaço que cada arquivo ocupará no executável gerado, gerando os offsets.
– *void realocacao(lista de estruturas objeto, inteiro com o número de objetos)*: As referências irresolvidas são buscadas nos outros arquivos.

2.3 Fluxo de Execução

O programa inicialmente lê os parâmetros passados a ele, verificando seu formato. Caso correto, ele cria um vetor da estrutura objeto associado a cada arquivo, e seta os offsets correspondetens. Após isso, as referências são resolvidas, e as instruções de cada objeto são inseridos em série no arquivo de saída (a começar pelo main, segundo então a ordem de entrada). Não havendo falhas neste processo, o main retorna 0.

3 Controle & IO

3.1 Execução e Compilação do Expansor

O programa pode ser compilado através do g++, pelo utilitário *make*, usando o makefile providenciado. Uma vez compilado, chamadas devem seguir o formato:

./ligador n[FonteN.mmk] [-m main.mmk] [-o Executável.mk]*

Onde se especifica uma quantidade variável de arquivos fonte FonteN.mmk, um único arquivo principal main.amk, prefixado por -m, e um arquivo de saída Executável.mk, prefixado por -o. Os termos podem aparecer em qualquer ordem.

3.2 Formato dos Arquivos de Entrada e Saída

Cada linha dos programas de entrada do ligador deve seguir o formato:

– Número de referências conhecidas daquele módulo

- N referências conhecidas e seus ILCs
- Número de referências totais daquele módulo
- M número de referências totais daquele módulo e onde foram referênciadas naquele módulo.
- X inteiros que representam o programa total, onde o inteiro 32767 representa uma referência desconhecida naquele módulo.

Onde *label* e *comentário* são opcionais, porém devendo ser devidamente delimitados (o sufixo ":" para o label e o prefixo ";" para o comentário). O tipo e quantidade dos operandos é dependente da instrução.

4 Testes

Vários programas foram testados, de forma a obter cobertura total do código. Os testes podem ser executados na MV tornando $PC = EndInicial = 0$ e $SP = 1000$. Os testes fizeram forte uso do expansor de macros, que promove grandes alterações sintáticas, visando aproximar uma linguagem mais alta.

- *Calculadora (calc.amk):* Implementa uma calculadora simples, que recebe 3 parâmetros: operando A, operando B e o número que representa a operação. Faz-se (1) $A+B$, (2) $A-B$, (3) $A*B$, (4) A/B e $A\%B$, (5) A^B.
- *Primo (primo.amk):* Lê um valor do teclado e imprime o primeiro número primo maior que ele. Possui dependências com o módulo div.

5 Conclusão

O ligador é eficaz em lidar com programas escritos neste conjunto de instruções, permitindo recompilação parcial e programas multi-modulares.

A execução do trabalho transcorreu sem maiores dificuldades, e os resultados obtidos correspondem ao esperado.

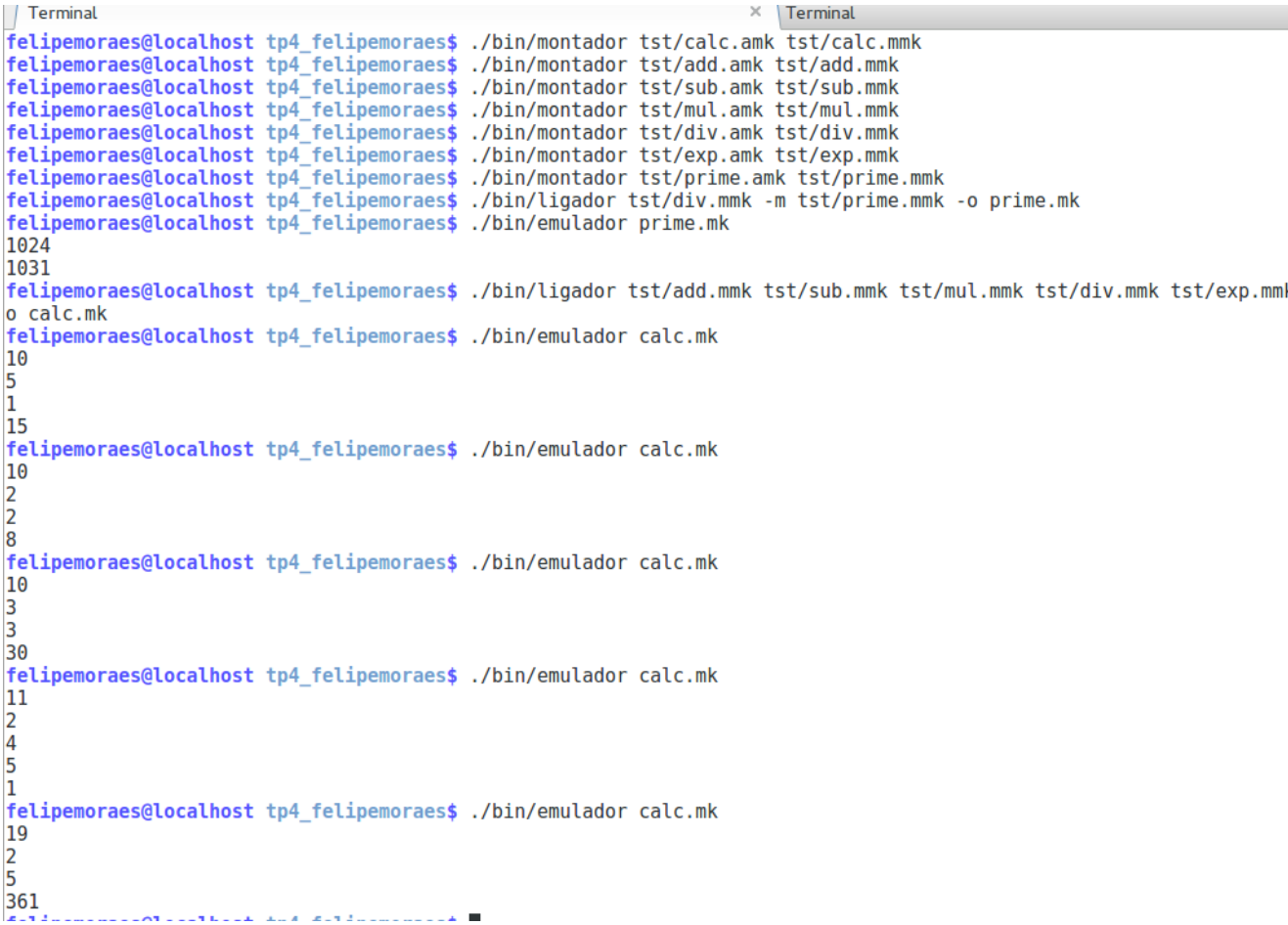

Figura 1: Compilação e execução dos testes

A Apêndice

A.1 Listagem de Arquivos

- Código Fonte:
  - *src/main.c:* Interpreta os parâmetros de entrada e controla o fluxo do programa.
  - *src/lista.c, src/lista.h:* Implementa TAD lista encadeada.
  - *src/linker.c, src/linker.h:* Implementa o ligador.
- Testes:
  - *tst/calc.amk*
  - *tst/add.amk*
  - *tst/sub.amk*
  - *tst/mul.amk*
  - *tst/div.amk*
  - *tst/exp.amk*
  - *tst/prime.amk*