Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Disciplina Software Básico (DCC008) - 2011_01

Trabalho Prático 04

1 Descrição Geral

O objetivo deste trabalho é projetar e implementar um **Editor de Ligação** para os programas feitos na linguagem interpretada pela máquina virtual construída nos trabalhos anteriores. A tarefa do editor de ligação ou *linker* é rearranjar o código do programa, incorporando a ele todas as partes referenciadas no código original, resultando em um código executável pelo processador. Essa tarefa pode ser realizada também pelos chamados carregadores.

2 Informações Importantes

- O trabalho deve ser feito **individualmente**, podendo ser discutido entre os colegas, mas código fonte não poderá ser trocado.
- A data de entrega será divulgada no **Moodle** por meio da criação do recurso para a entrega do trabalho.
- **Política de Atrasos:** A entrega de cada trabalho prático deve ser realizada até a data estipulada na tarefa correspondente do Moodle. As tarefas foram programadas para aceitar submissões atrasadas, mas estas serão penalizadas. A penalização pelo atraso será geométrica com o mesmo conforme a fórmula abaixo:

$$Desconto(\%) = \frac{2^{d-1}}{0,32}$$

  Essa fórmula dá a porcentagem de desconto para d dias de atraso.

  ATENÇÃO: Note que depois de 6 dias o trabalho é avaliado em 0 pontos. Com base na política acima, é altamente recomendável que se esforcem para entregar o TP dentro do prazo para que não tenhamos problemas futuros.
- O trabalho deverá ser implementado **obrigatoriamente na linguagem C**.
- Deverá ser entregue o código fonte com os arquivos de dados necessários para a execução e arquivos Makefile que permitam a compilação do programa nas máquinas UNIX do departamento.
- Além disso, deverá ser entregue uma pequena documentação contendo todas as decisões de projeto que forem tomadas durante a implementação, sobre aspectos não contemplados na especificação, assim como uma justificativa para essas decisões. Esse documento não precisa ser extenso (entre 3 e 5 páginas).

- A ênfase do trabalho está no funcionamento do sistema e não em aspectos de programação ou interface com o usuário. Assim, não deverá haver tratamento de erros, ou seja, pode-se considerar que o programa tratado esteja correto. As operações de entrada e saída podem ser implementadas da forma mais simples possível.

- Todas as dúvidas referentes ao Trabalho Prático 04 serão esclarecidas por meio do fórum, devidamente nomeado, criado no ambiente **Moodle** da disciplina.

- A entrega do trabalho deverá ser realizada por meio do **Moodle**, na tarefa criada especificamente para tal. O que deve ser entregue e em que formato:

  - Pasta contendo os arquivos do trabalho. A pasta deve ser nomeada como `tp4_seulogin`, onde `seulogin` refere-se ao seu login no DCC. Esta pasta deve ser compactada no formato `.tar.gz`. Assim, o pacote final a ser enviado é `tp4_seulogin.tar.gz`
  - Estrutura de diretórios da pasta `tp4_seulogin`:
    * Arquivo `documentacao.pdf`
    * Arquivo `Makefile_montador` contendo as regras de compilação para o montador modificado
    * Pasta `src_montador` contendo o código fonte do montador modificado
    * Arquivo `Makefile_ligador` contendo as regras de compilação para o editor de ligação
    * Pasta `src_ligador` contendo o código fonte do editor de ligação
    * Pasta `test` contendo os programas de teste

3 Características da Máquina Virtual

Como os trabalhos práticos da disciplina são dependentes das etapas anteriores, convém relembrar algumas informações sobre a máquina virtual:

- A menor unidade endereçável nessa máquina é um inteiro.

- Os tipos de dados tratados pela máquina também são somente inteiros.

- A máquina possui uma memória de não menos que 1000 posições e 3 registradores: o `PC` (contador de programas), o `AC` (acumulador) e o `SP` (apontador da pilha).

- A única forma de endereçamento existente na máquina é direto, relativo ao `PC`.

- As instruções são codificadas em um inteiro. A maioria delas possui um operando, que é uma posição de memória (`M`), especificada conforme o modo de endereçamento existente, também codificados em inteiros. As exceções são as instruções `RET` e `HALT`, que não possuem operandos. Relembrando, o conjunto de instruções da máquina está detalhado na Tabela 1.

Código	Símbolo	Significado	Ação
01	LOAD	Carrega AC	AC ← Memória[PC+M]
02	STORE	Armazena AC	Memória[PC+M] ← AC
03	JMP	Desvio incondicional	PC ← PC + M
04	JPG	Desvia se maior que 0	Se AC > 0, PC ← PC + M
05	JPE	Desvia se igual a 0	Se AC = 0, PC ← PC + M
06	JPL	Desvia se menor que 0	Se AC < 0, PC ← PC + M
07	JPNE	Desvia se diferente de 0	Se AC ≠ 0, PC ← PC + M
08	PUSH	Empilha valor	SP ← SP - 1; Memória[SP] ← Memória[PC+M]
09	POP	Desempilha valor	Memória[PC+M] ← Memória[SP]; SP ← SP + 1;
10	READ	Lê valor para a memória	Memória[PC+M] ← "Valor lido"
11	WRITE	Escreve conteúdo da memória	"Imprime" Memória[PC+M]
12	CALL	Chamada de subrotina	SP ← SP - 1; Memória[SP] ← PC; PC ← PC + M
13	RET	Retorno de subrotina	PC ← Memória[SP]; SP ← SP + 1
14	ADD	Soma operando em AC	AC ← AC + Memória[PC+M]
15	SUB	Subtrai operando de AC	AC ← AC - Memória[PC+M]
16	XOR	XOR lógico	AC ← AC **xor** Memória[PC+M]
17	AND	AND lógico	AC ← AC **and** Memória[PC+M]
18	OR	OR lógico	AC ← AC **or** Memória[PC+M]
19	HALT	Parada	

Tabela 1: Conjunto de instruções da máquina virtual.

Além das instruções acima, existem também as pseudo-instruções:

- `WORD A` → Usada para alocar uma posição de memória cujo valor será `A`, ou seja, quando houver tal instrução, a posição de memória que está sendo referenciado pelo `PC` deverá receber o valor `A`.
- `END` → Indica o final do programa para o montador.
- `BEGINMACRO` → Indica o início da definição de uma macro.
- `ENDMACRO` → Indica o fim da definição de uma macro.

4 Descrição do Editor de Ligação

Os principais objetivos do Editor de Ligação a ser projetado e implementado neste trabalho são:

- Permitir relocação de programas: endereço de carga de programa deve ser definido somente após a tradução.
- Permitir tradução separada: os módulos de um programa são montados separadamente e depois combinados para formar um único programa objeto.

Algumas informações importantes sobre a implementação são:

- Deverá ser feita uma modificação no montador implementado no **Trabalho Prático 2** para que ele gere informações que serão utilizadas na relocação e ligação dos programas.
- Deverá ser implementado o editor de ligação que combine os diversos sub-programas que foram montados independentemente. Deve-se tomar cuidado para que no momento da ligação, o trecho de código correspondente à função `main`, ou seja, o ponto de início do programa, esteja no início do código objeto final, para que o objeto gerado possa ser executado com o valor inicial de `PC` correspondendo a zero. **Não** é necessário considerar situações em que o `PC` seja iniciado com valor diferente de zero.
- Para a implementação do editor de ligação, informação adicional precisa ser gerada pelo montador do trabalho prático 2. Além de não gerar erros no segundo passo da montagem, devido a símbolos desconhecidos, o arquivo gerado deve conter a tabela de símbolos do programa. O programa gerado pelo montador, portanto, não é necessariamente executável, mas um formato que servirá de entrada para o editor de ligação, que deve realizar as **3 tarefas conforme descritas no capítulo 7 do livro texto: alocação, ligação e relocação**, produzindo, assim, a partir de 1 ou mais arquivos gerados pelo montador, 1 programa executável único, no formato que possa ser carregado e executado na máquina virtual do trabalho prático 1.

5 Descrição da Tarefa

Os alunos deverão modificar o montador e implementar o editor de ligação definido acima. Além disso, deverão ser criados dois programas de testes, **obrigatoriamente** contendo mais de um módulo cada um, a saber:

- **Calculadora:** Programa que lê três números (*a, b, c*) e realiza as operações adicação, subtração, multiplicação, divisão inteira (imprimindo o quociente e o resto) e exponenciação. Onde:
  - *b* é a operação a ser realizada: *b*=1: adicação; *b*=2: subtração; *b*=3: multiplicação; *b*=4: divisão inteira (imprimindo o quociente e o resto); *b*=5:exponenciação;
  - *a* e *c* são os valores aos quais a operação será realizada.
- **Número Primo:** Programa que lê um número *n* e imprime o primeiro número primo maior que *n*.

6 Formato da Entrada de Dados

O formato de entrada do montador modificado é exatamente igual ao formato utilizado no montador implementado no trabalho prático 02. A entrada do editor de ligação deve seguir algum formato intermediário que combine código de máquina com as informações geradas pelo montador modificado.

Exemplo:
Programa principal contido no arquivo `main.asm`:

```
READ A
READ B
CALL CALC
HALT
A: WORD 0
B: WORD 0
END
```

Módulo de cálculo contido no arquivo `calculo.asm`:

```
CALC: LOAD A
STORE C
SUB B
JPG L
LOAD B
STORE C
L: WRITE A
WRITE B
WRITE C
RET
C: WORD 0
END
```

Para um teste inicial do seu editor de ligação, utilize o programa acima.
Atenção: Tal programa não testa todas as instruções e não deve ser utilizado como único teste do seu programa.

7 Formato da Saída de Dados

Esse formato deverá ser especificado por cada aluno como decisão de projeto. Basicamente, deverá ser considerado o fato de que o montador modificado deverá disponibilizar tanto o código traduzido quanto informações necessárias para o processo de ligação. A saída do editor de ligação é um programa objeto no formato aceito pela máquina virtual implementada no Trabalho Prático 01.

8 Formato de Chamada do Editor de Ligação

O editor de ligação receberá $n + 2$ argumentos quando da sua chamada:

- n nomes de arquivos de entrada: contendo os módulos objetos a serem ligados – informados como os n **primeiros argumentos** na chamada da aplicação.
- Nome do arquivo que contém o programa principal: informado por meio da opção `-m`.
- Nome do arquivo de saída do editor de ligação: informado por meio da opção `-o`.

Exemplo:

```
./ligador modulo1.o modulo2.o modulo3.o -m main.o -o programa.maq
```

A chamada acima tem a seguinte semântica: executar o editor de ligação para relocar e ligar os módulos escritos nos arquivos (`modulo1.o`, `modulo2.o`, `modulo3.o` e `main.o`), sendo que o arquivo `main.o` contém o programa principal. A saída deve ser escrita no arquivo `programa.maq`. Note que a única saída de dados do programa é o arquivo de saída. Não é necessário imprimir informações na tela.

9 Sobre a Documentação

- Deve conter as decisões de projeto.
- Deve conter as informações de como executar o programa. Obs.: é necessário cumprir os formatos definidos acima para a execução, mas tais informações devem estar presentes também na documentação.
- Não incluir o código fonte no arquivo de documentação.
- Deve conter elementos que comprovem que o programa foi testado, mas sem a necessidade de incluir os códigos dos programas de testes utilizados. Os arquivos relativos a testes devem ser enviados no pacote do trabalho, conforme descrito na Seção 2. A documentação deve conter apenas as referências a esses arquivos, o que eles fazem e os resultados obtidos.

10 Considerações Finais

Possivelmente, os testes de funcionamento do expansor serão automatizados, o que torna necessário o cumprimento fiel de todas as especificações de interface descritas neste documento. As decisões de projeto devem fazer parte apenas da estrutura interna do seu programa, não podendo afetar a interface de entrada e saída, com exceção da saída do montador modificado, que será baseada em uma decisão particular de projeto.

Após a implementação deste trabalho, tem-se um conjunto de ferramentas com as quais é possível escrever um programa em uma linguagem de alto nível e traduzi-lo para a linguagem que a máquina virtual reconhece, seguindo os passos abaixo:

1. Executar o **Expansor de Macros** em cada um dos módulos do programa.
2. Executar o **Montador** em cada um dos arquivos obtidos no passo anterior.

3. Executar o **Editor de Ligação**, obtendo-se o programa objeto único.

4. Executar a **Máquina Virtual**, tendo como entrada o programa objeto gerado na etapa anterior.

A Figura 1 mostra o fluxo esquemático dessas operações.

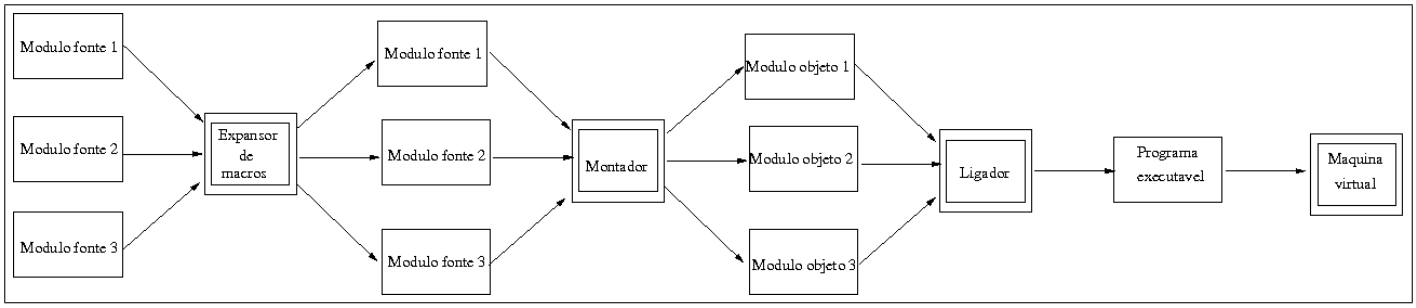

Figura 1: Fluxo de operação das ferramentas implementadas nos trabalhos práticos.