Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Disciplina Software Básico (DCC008) - 2018/2

Trabalho Prático 4 - Ligador

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 especificada através de uma tarefa no Moodle;
- **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 exclusivamente o código fonte com os arquivos de dados necessários para a execução e um arquivo Makefile que permita a compilação do programa nas máquinas UNIX do departamento;

- Além disso, deverá ser entregue uma pequena documentação contendo todas as decisões de projeto que foram 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 no programa de entrada;
- Todas as dúvidas referentes ao Trabalho Prático 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. As instruções de submissão, alguns arquivos de teste, e o esqueleto da organização dos arquivos estão presentes no arquivo "`tp4_seulogin.tar.gz`", disponível para download no Moodle:
- **ATENÇÃO:** trabalhos que não seguem esse padrão serão penalizados.

3 Especificação da Máquina

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, 3 registradores de propósito específico e 8 registradores de propósito geral;
- Os registradores de propósito específico são:
  - PC (contador de programas): contém o endereço da próxima instrução a ser executada;
  - AP (apontador da pilha): aponta para o elemento no topo da pilha;
  - PEP (palavra de status do processador): consiste em 2 bits que armazenam o estado da última operação lógico/aritmética realizada na máquina, sendo um dos bits para indicar que a última operação resultou em zero, e outro bit para indicar que a última operação resultou num resultado negativo;
- Os registradores de propósito geral são indexados por um valor que varia de 0 a 7;
- A única forma de endereçamento existente na máquina é direto, relativo ao PC;
- As instruções `READ` e `WRITE` leem e escrevem um inteiro na saída padrão do emulador;
- As instruções são codificadas em um inteiro, podendo ter dois, um ou nenhum operando, que é o caso das instruções `RET` e `HALT`.

- Os operandos podem ser uma posição de memória (M, codificado como inteiro) ou um registrador (R, codificado como um inteiro entre 0 e 7).
- O conjunto de instruções é o mesmo da máquina anterior, conforme a Tabela 1. As operações marcadas com * atualizam o valor do PEP

Cód	Símbolo	Operandos	Significado	Ação
01	ADD	R1 R2	Soma dois registradores	$Reg[R1] \leftarrow Reg[R1] + Reg[R2]$ *
02	SUB	R1 R2	Subtrai dois registradores	$Reg[R1] \leftarrow Reg[R1] - Reg[R2]$ *
03	AND	R1 R2	AND (bit a bit) de dois registradores	$Reg[R1] \leftarrow Reg[R1]$ AND $Reg[R2]$ *
04	OR	R1 R2	OR (bit a bit) de dois registradores	$Reg[R1] \leftarrow Reg[R1]$ OR $Reg[R2]$ *
05	XOR	R1 R2	XOR (bit a bit) de dois registradores	$Reg[R1] \leftarrow Reg[R1]$ XOR $Reg[R2]$ *
06	NOT	R1	NOT (bit a bit) de um registrador	$Reg[R1] \leftarrow$ NOT $Reg[R1]$ *
07	JUMP	M	Desvio incondicional	$PC \leftarrow PC + M$
08	JZ	M	Desvia se zero	Se $PEP[zero], PC \leftarrow PC + M$
09	JNZ	M	Desvia se não zero	Se $!PEP[zero], PC \leftarrow PC + M$
10	JN	M	Desvia se negativo	Se $PEP[negativo], PC \leftarrow PC + M$
11	JNN	M	Desvia se não negativo	Se $!PEP[negativo], PC \leftarrow PC + M$
12	PUSH	R	Empilha valor do registrador	$AP \leftarrow AP - 1$ $Mem[AP] \leftarrow Reg[R]$
13	POP	R	Desempilha valor no registrador	$Reg[R] \leftarrow Mem[AP]$ $AP \leftarrow AP + 1$
14	CALL	M	Chamada de subrotina	$AP \leftarrow AP - 1$ $Mem[AP] \leftarrow PC$ $PC \leftarrow PC + M$
15	LOAD	R M	Carrega Registrador	$Reg[R] \leftarrow Mem[M + PC]$
16	STORE	R M	Armazena Registrador	$Mem[M + PC] \leftarrow Reg[R]$
17	READ	R	Lê valor para registrador	$Reg[R] \leftarrow$ "valor lido"
18	WRITE	R	Escreve conteúdo do registrador	"Imprime" $Reg[R]$
19	COPY	R1 R2	Copia registrador	$Reg[R1] \leftarrow Reg[R2]$ *
20	RET		Retorno de subrotina	$PC \leftarrow Mem[AP]$ $AP \leftarrow AP + 1$
21	HALT		Parada	

Tabela 1: Instruções da Máquina de Khattab

Além das instruções acima, existem também as pseudo-instruções `WORD` e `END`:

- `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: 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 adição, subtração, multiplicação, divisão inteira (imprimindo o quociente e o resto) e exponenciação. Onde:
  - b é a operação a ser realizada: b=1: adição; 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;
  - Cada operação deve ser implementada em um módulo diferente.
- **Número Primo:** Programa que lê um número n e imprime o primeiro número primo maior que n.

A implementação desses programas de teste serão avaliados, portanto não devem ser compartilhados entre os colegas. Por outro lado, outros programas de teste adicionais podem ser compartilhados. Lembre-se de colocar todos os testes (inclusive os obrigatórios) no diretório "`tst/`" e de citá-los em sua documentação.

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.amv`:

```
        READ R0
        READ R1
        STORE R0 A
        STORE R1 B
        LOAD R0 ZERO
        LOAD R1 ZERO
        CALL CALC
        HALT
   A:   WORD 0
   B:   WORD 0
ZERO:   WORD 0
        END
```

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

```
CALC:   LOAD R5 A
        LOAD R6 B
        SUB R5 R6
        JN MB
  MA:   LOAD R7 A
        JUMP OK
  MB:   LOAD R7 B
  OK:   WRITE R5
        WRITE R6
        WRITE R7
        RET
        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.

A saída do *Editor de Ligação* deve ser submetida a mv para garantir que o programa foi traduzido corretamente.

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.mmv modulo2.mmv modulo3.mmv -m main.mmv -o programa.mv
```

A chamada acima tem a seguinte semântica: executar o editor de ligação para relocar e ligar os módulos escritos nos arquivos (`modulo1.amv`, `modulo2.amv`, `modulo3.amv` e `main.amv`), sendo que o arquivo `amv.o` contém o programa principal. A saída deve ser escrita no arquivo `programa.mv`. 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 (e.g. imagens da telas de execução). Os arquivos relativos a testes devem ser enviados no pacote do trabalho, conforme descrito na Seção 2. A documentação deve conter referências a esses arquivos, explicação do que eles fazem e dos resultados obtidos.

10 Considerações Finais

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.

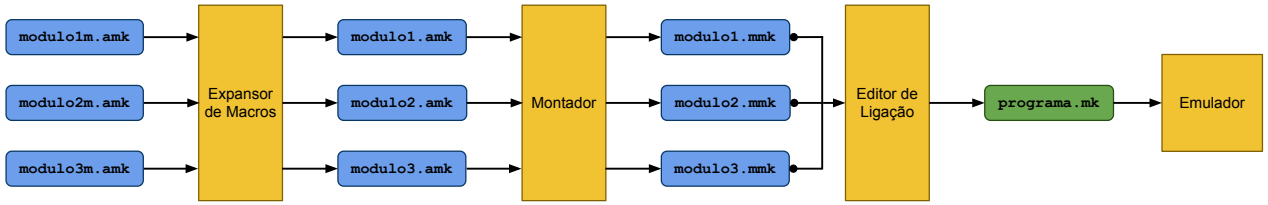

Figura 1: Fluxo de operação das ferramentas implementadas nos trabalhos práticos.

As extensões dos arquivos são apenas convenções para facilitar a identificação dos tipos de arquivos. No caso desse exemplo, as extensões são:

- `.amv`: Arquivo com linguagem assembly da Máquina de Khattab.
- `.mmv`: Arquivo intermediário que o novo montador deve gerar e ser lido pelo editor de ligação. Esse formato deve ser especificado pelo aluno e devidamente documentado.
- `.mv`: Arquivo binário executável da da Máquina de Khattab que será executado pelo emulador.