

Trabalho Prático 2 - Montador

1 Descrição Geral

Este trabalho envolve a implementação de um montador para a máquina básica sendo utilizada ao longo do curso.

2 Informações Importantes

- O trabalho deve ser feito **individualmente**, podendo ser discutido entre os colegas, mas o código fonte não poderá ser compartilhado.
- 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, porém entregas em atrasos serão penalizadas. A penalização pelo atraso será geométrica com o mesmo conforme a fórmula seguir:

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

A fórmula indica o percentual 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 em 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 tais decisões. O documento não precisa ser necessariamente 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 é necessário 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 feita via **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 `tp2_seulogin.tar.gz` disponível para download no Moodle.
- **ATENÇÃO:** trabalhos entregues fora do padrão serão penalizados.

3 Especificação do Montador

- Deverá ser implementado um montador de 2 passos, conforme descrito no livro.
- Deverão ser acrescentadas duas pseudo-instruções: **WORD**, usada para “alocar” uma posição de dados na memória; e **END** que indica o final do programa para o montador.
 - **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**, que é um inteiro.
 - **END** → Indica o final do programa para o montador.
- A linguagem simbólica é bastante simples, e cada linha terá o seguinte formato:


```
[<label>:] <operador> <operando1> <operando2> [; comentário]
```

Ou seja:

- Se houver algum label, ele será definido no início da linha e deverá terminar com :
- A presença do operador é obrigatória.
- Os operandos dependem da instrução, algumas instruções não possuem operando, enquanto outras tem 1 operando e outras 2 operandos.
- Podem haver comentários no fim da linha, sendo que devem começar por ;
- Os operandos podem ser um registrador (**R0**, **R1**, ..., **R15**) ou uma posição de memória no programa (identificada pelo label).
- Os registradores de **R0** a **R15** são respectivamente 0, 1, ..., 15 em linguagem de máquina.
- A posição de memória dos desvios e instruções **LOAD/STORE** é relativa ao PC após a leitura do operando.
- O conjunto de instruções é o mesmo da máquina anterior, conforme a Tabela 1.

- Espaços em branco extras entre o label, operador, operandos e comentário são permitidos, não devendo afetar o funcionamento do montador.
- Podem existir linhas vazias e linhas com apenas comentário, que devem ser ignoradas pelo montador.

| Cód | Símbolo | Operandos | Significado | Ação |
|-----|---------|-----------|---------------------------------------|---|
| 01 | LOAD | R M | Carrega Registrador | $Reg[R] \leftarrow Mem[M + PC]$ |
| 02 | STORE | R M | Armazena Registrador | $Mem[M + PC] \leftarrow Reg[R]$ |
| 03 | READ | R | Lê valor para registrador | $Reg[R] \leftarrow$ “valor lido” |
| 04 | WRITE | R | Escreve conteúdo do registrador | “Imprime” $Reg[R]$ |
| 05 | COPY | R1 R2 | Copia registrador | $Reg[R1] \leftarrow Reg[R2] *$ |
| 06 | NEG | R1 | Inverte o sinal do registrador | $Reg[R1] \leftarrow -Reg[R1] *$ |
| 07 | SUB | R1 R2 | Subtrai dois registradores | $Reg[R1] \leftarrow Reg[R1] - Reg[R2] *$ |
| 08 | ADD | R1 R2 | Soma dois registradores | $Reg[R1] \leftarrow Reg[R1] + Reg[R2] *$ |
| 09 | AND | R1 R2 | AND (bit a bit) de dois registradores | $Reg[R1] \leftarrow Reg[R1] \text{ AND } Reg[R2] *$ |
| 10 | OR | R1 R2 | OR (bit a bit) de dois registradores | $Reg[R1] \leftarrow Reg[R1] \text{ OR } Reg[R2] *$ |
| 11 | XOR | R1 R2 | XOR (bit a bit) de dois registradores | $Reg[R1] \leftarrow Reg[R1] \text{ XOR } Reg[R2] *$ |
| 12 | NOT | R1 | NOT (bit a bit) de um registrador | $Reg[R1] \leftarrow \text{NOT } Reg[R1] *$ |
| 13 | JMP | M | Desvio incondicional | $PC \leftarrow PC + M$ |
| 14 | JZ | M | Desvia se zero | Se $PSW[zero]$, $PC \leftarrow PC + M$ |
| 15 | JNZ | M | Desvia se não zero | Se $!PSW[zero]$, $PC \leftarrow PC + M$ |
| 16 | JN | M | Desvia se negativo | Se $PSW[negativo]$, $PC \leftarrow PC + M$ |
| 17 | JNN | M | Desvia se não negativo | Se $!PSW[negativo]$, $PC \leftarrow PC + M$ |
| 18 | PUSH | R | Empilha valor do registrador | $SP \leftarrow SP - 1$ $Mem[SP] \leftarrow Reg[R]$ |
| 19 | POP | R | Desempilha valor no registrador | $Reg[R] \leftarrow Mem[SP]$ $SP \leftarrow SP + 1$ |
| 20 | CALL | M | Chamada de subrotina | $SP \leftarrow SP - 1$ $Mem[SP] \leftarrow PC$ $PC \leftarrow PC + M$ |
| 21 | RET | | Retorno de subrotina | $PC \leftarrow Mem[SP]$ $SP \leftarrow SP + 1$ |
| 22 | HALT | | Parada | |

Tabela 1: Instruções da Máquina Virtual

4 Descrição da Tarefa

Os alunos deverão implementar o montador como especificado acima para a máquina virtual implementada no TP1.

Além disso, deverão ser criados alguns programas de testes:

- **Mediana:** Programa que lê 7 números e imprime sua mediana. Exemplo:

valores = {1, 77, 25, 59, 20, 41, 61} **mediana** = 41

- **Fibonacci:** Programa que lê um número N e imprime o N-ésimo número da sequência de Fibonacci, considerando os dois primeiros números da sequência sendo 0 e 1. Exemplo:

valor = {8} **resultado** = 13

- **Divisão:** Programa que lê dois números, faz a divisão força bruta e imprime o quociente e o resto. Exemplo:

valores = {107, 20} **quociente** = 5 **resto** = 7

- **Exponenciação:** Programa que lê dois números (base, expoente), faz a exponenciação e imprime o resultado. Exemplo:

valores = {2, 3} **resultado** = 8

As implementações dos programas de teste serão avaliadas, portanto tais programas 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.

5 Formato da Entrada de Dados

O programa a ser traduzido pelo montador deverá ser escrito em um arquivo texto sem formatação, sendo que as instruções devem ser dispostas uma por linha do arquivo.

Para um teste inicial do seu montador, utilize o programa abaixo (o mesmo salvo em “**tst/teste1.amv**”). O programa pede dois inteiros, imprime os dois na ordem que foram informados e depois imprime o maior deles.

```

READ R0
READ R1
STORE R0 A
SUB R0 R1
LOAD R0 A
JN MB
COPY R2 R0
JMP L
MB: COPY R2 R1
L: WRITE R0
WRITE R1
WRITE R2
HALT
A: WORD 0
END

```

Atenção: Tal programa não testa todas as instruções e não deve ser utilizado como único teste do montador.

6 Formato da Saída de Dados

Duas opções de saída devem estar disponíveis para utilização:

1. *Simples*: gera um arquivo binário cujo formato corresponde ao formato de entrada do emulador da Máquina Virtual. Detalhes sobre o formato utilizado pela máquina virtual podem ser obtidos na especificação do Trabalho Prático 1.
2. Modo *verbose*: além da criação do arquivo acima, deve imprimir a tabela de símbolos, contendo o nome e o endereço, ao final da execução.

7 Formato de Chamada do Montador

- Nome do arquivo contendo o programa a ser traduzido pelo montador: informado como **primeiro argumento** na chamada do montador.
- Nome do arquivo de saída: informado com **segundo argumento** na chamada do montador.
- Modo de saída de dados [s|v]: informado como **terceiro argumento** na chamada do montador. Parâmetro opcional, caso não seja informado o modo de saída deve ser o *simples*.

Exemplos:

```
./montador teste1.amv teste1.mv - traduz o programa teste1.amv no arquivo
                                teste1.mv, com saída simples
./montador teste1.amv teste1.mv s - traduz o programa teste1.amv no arquivo
                                teste1.mv, com saída simples
./montador teste1.amv teste1.mv v - traduz o programa teste1.amv no arquivo
                                teste1.mv, com saída verbose
```

A saída do montador deve ser executada no emulador da MV para garantir que o programa foi traduzido corretamente.

8 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 das 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.

9 Considerações Finais

É obrigatório o cumprimento fiel de todas as especificações de interface descritas neste documento. As decisões de projeto devem fazer parte apenas da estrutura interna do montador, não podendo afetar a interface de entrada e saída.