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

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 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 "`tp2_seuNOME.tar.gz`", disponível para download no Moodle:
- **ATENÇÃO:** trabalhos que não seguem esse padrão serão penalizados.

3 Especificação do Montador

- Deverá ser implementado um montador de 2 passos.
- 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`, ..., `R7`) ou uma posição de memória no programa (identificada pelo label).
  - Os registradores de `R0` a `R7` são respectivamente 0, 1, ..., 7 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	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

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 a mediana deles. Exemplo:

$$\textbf{valores} = \{1, 77, 25, 59, 20, 41, 61\}\ \textbf{mediana} = 41$$

- **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:

$$\textbf{valor} = \{8\}\ \textbf{resultado} = 13$$

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

$$\textbf{valores} = \{5, 20\} : \textbf{resultado} = 100$$

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

$$\textbf{valores} = \{107, 20\}\ \textbf{quociente} = 5\ \textbf{resto} = 7$$

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.

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, imprimes 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
JUMP 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 de Khattab. 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 endreço, 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
                                     binário teste1.mv, com saída simples
./montador teste1.amv teste1.mv s  - traduz o programa teste1.amv no arquivo
                                     binário teste1.mv, com saída simples
./montador teste1.amv teste1.mv v  - traduz o programa teste1.amv no arquivo
                                     binário 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 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.

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.