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

Trabalho Prático 02

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 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 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 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 02 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 `tp2_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 `tp2_seulogin.tar.gz`
  - Estrutura de diretórios da pasta `tp2_seulogin`: Arquivo `Makefile`, arquivo `documentacao.pdf`, pasta `src` contendo o código fonte e pasta `test` contendo os programas de teste.

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`
  - `END` → Indica o final do programa para o montador
- A linguagem simbólica é bastante simples, e cada linha terá o seguinte formato:

  `[<label>:]` `<operador>` `<operando>` `[; comentário]`

  Ou seja:
  - Se houver algum label, ele será definido no início da linha e deverá terminar com `:`
  - A presença do operador é obrigatória
  - O operando só não aparecerá nas operações `RET` e `HALT` e na pseudo-instrução `END`
  - Podem haver comentários no fim da linha, sendo que devem começar por `;`
- O conjunto de instruções é o mesmo da máquina anterior, conforme Tabela 1.

4 Descrição da Tarefa

Os alunos deverão implementar o montador como especificado acima. Além disso, deverão ser criados dois programas de testes, a saber:

- **Fatorial:** Programa que lê um números imprime o resultado do fatorial deste:

$$4! = 24$$

- **Máximo divisor comum (MDC):** Programa que lê dois números e imprime o MDC destes:

$$mdc(24, 16) = 4$$

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 $\neq$ 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.

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.

Exemplo:

```
READ A
READ B
LOAD A
STORE C
SUB B
JPG L
LOAD B
STORE C
L: WRITE A
WRITE B
WRITE C
HALT
A: WORD 0
B: WORD 0
C: WORD 0
END
```

Para um teste inicial do seu montador, utilize o programa acima.
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 texto cujo formato corresponde ao formato de entrada da máquina virtual. Detalhes sobre o formato utilizado pela máquina virtual podem ser obtidos na especificação do Trabalho Prático 01.

2. Modo *verbose*: além da criação do arquivo acima, deve imprimir a tabela de símbolos da tela, ao final da execução.

7 Formato de Chamada da Montador

- Modo de saída de dados [s|v]: informado como **primeiro argumento** na chamada do Montador.

- Nome do arquivo contendo o programa a ser traduzido pelo montador: informado como **segundo argumento** na chamada do montador.

- Nome do arquivo de saída: informado como **terceiro argumento** na chamada do montador.

Exemplo:

```
./montador s teste1 saida1
```

A chamada acima tem a seguinte semântica: executar o montador para traduzir o programa contido no arquivo `teste1` para o formato aceito pela máquina virtual e gravar o resultado no arquivo `saida1`. Além disso, foi informado ao montador para utilizar o modo de saída de dados simples.

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, 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.

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.