Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Disciplina Software Básico (DCC008) - 2012_01

Trabalho Prático 01

1 Descrição Geral

Esse trabalho envolve a implementação de um simulador para uma máquina básica que será utilizada ao longo do curso para implementarmos diversos conceitos que iremos estudar.

Atenção todos trabalhos da disciplina dependerão deste simulador, então implemente-o com cuidado pois esta máquina virtal pode impactar na nota de todos trabalhos.

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 01 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 `tp1_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 `tp1_seulogin.tar.gz`;
  - Estrutura de diretórios da pasta `tp1_seulogin`: Arquivo `Makefile`, arquivo `documentacao.pdf`, pasta `src` contendo o código fonte e pasta `test` contendo os programas de teste.

3 Especificação da 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 máquina tem duas formas de endereçamento: (1) direto e relativo ao `PC` para desvios e chamadas e (2) absoluto para dados;
- As instruções são codificadas em um inteiro. Com exceção das instruções `RET`, `HALT` e `NOT`, todas instruções possuem um operando (o qual será referenciado como `M`), este M é uma posição de memória, especificada conforme o modo de endereçamento existente, M também é codificado em inteiros. As instruções `RET`, `HALT` e `NOT` não possuem operandos. O conjunto de instruções da máquina está detalhado na Tabela 1.

4 Descrição da Tarefa

Sua tarefa é implementar um programa interpretador que simule a máquina virtual MV, ou seja, que executa programas em linguagem de máquina conforme definido acima.

O trabalho pode ser visto como duas partes: a primeira é o interpretador da máquina propriamente dito, que contém uma representação da memória da máquina, e o interpretador do programa na memória. Os registradores `PC` e `SP` devem estar inicializados para execução, o primeiro com a posição inicial do programa e o segundo com uma posição qualquer da memória. Observe que a pilha cresce decrementando o `SP`, e decresce incrementando o `SP`.

A segunda parte é o carregador do programa, que consiste em ler de um arquivo um programa em linguagem de máquina que contem as instruções.

Essas duas "peças" são fundamentais para a continuidade dos trabalhos práticos da disciplina. Os outros trabalhos dependerão delas.

Código	Símbolo	Significado	Ação
01	LOAD	Carrega AC	AC ← Memória[M]
02	STORE	Armazena AC	Memória[M] ← AC
03	PUSH	Empilha valor	SP ← SP - 1; Memória[SP] ← Memória[M]
04	POP	Desempilha valor	Memória[M] ← Memória[SP]; SP ← SP + 1;
05	JMP	Desvio incondicional	PC ← PC + M
06	JPG	Desvia se maior que 0	Se AC $>$ 0, PC ← PC + M
07	JPGE	Desvia se maior ou igual a 0	Se AC $\geq$ 0, PC ← PC + M
08	JPL	Desvia se menor que 0	Se AC $<$ 0, PC ← PC + M
09	JPLE	Desvia se menor ou igual a 0	Se AC $\leq$ 0, PC ← PC + M
10	JPE	Desvia se igual a 0	Se AC $=$ 0, PC ← PC + M
11	JPNE	Desvia se diferente de 0	Se AC $\neq$ 0, PC ← PC + M
12	XOR	XOR (bit a bit)	AC ← AC **xor** Memória[M]
13	AND	AND (bit a bit)	AC ← AC **and** Memória[M]
14	OR	OR (bit a bit)	AC ← AC **or** Memória[M]
15	NOT	NOT (bit a bit)	AC ← **not** AC
16	ADD	Soma operando em AC	AC ← AC + Memória[M]
17	SUB	Subtrai operando de AC	AC ← AC - Memória[M]
18	READ	Lê valor para a memória	Memória[M] ← “Valor lido”
19	WRITE	Escreve conteúdo da memória	“Imprime” Memória[M]
20	CALL	Chamada de subrotina	SP ← SP - 1; Memória[SP] ← PC; PC ← PC + M
21	RET	Retorno de subrotina	PC ← Memória[SP]; SP ← SP + 1
22	HALT	Parada	

Tabela 1: Conjunto de instruções da máquina virtual.

5 Formato da Entrada de Dados

O programa a ser interpretado pela MV deverá ser escrito em um arquivo texto sem formatação, formado por um inteiro por linha, sendo esse valor uma instrução da máquina virtual ou um operando (M) da instrução da linha anterior. Exemplo:

```
18
17
```

- Na primeira linha temos a operação READ e o PC=1;
- Na segunda linha o operando M da instrução READ e o PC=2.

Observação: O problema acima deve ser lido da seguinte forma: (1) leia um valor da entrada padrão (e.g. scanf); (2) armazene o valor lido na posição M (que neste caso é igual a 17), onde PC=3 (isto porque a cada posição (linha) que o programa lê o PC deve ser incrementado), e M=17 que é o valor informado no programa.
Programa teste inicial:

```
18
17
18
18
01
17
17
18
06
04
19
18
05
02
19
17
22
00
00
```

Para um teste inicial do seu simulador, utilize o programa em linguagem de máquina acima, carregando-o a partir do endereço 0, e o apontador da pilha inicialmente para a posição 1000.

Atenção: Tal programa não testa todas as instruções então outros programas devem ser obrigatoriamente implementados com o objetivo de cubrir todas instruções. A qualidade dos experimentos implementados será avaliada na correção.

6 Formato da Saída de Dados

Duas opções de saída devem estar disponíveis para utilização:

1. Simples: Só imprime na tela o resultado do programa ou mensagens importantes. Ou seja, quando executar a instrução `READ`, deve ser exibida uma mensagem pedindo para informar um valor inteiro e quando executar a instrução `WRITE`, será feita a impressão na tela do valor conforme descrito na especificação de tal instrução.

2. Modo *verbose*: Imprime o passo a passo da execução, exibindo a cada instrução o valor de `PC`, `SP`, `AC`, e a instrução que está sendo executada. Essas informações são importantes para o acompanhamento do fluxo de execução e detecção de erros.

7 Formato de Chamada da MV

- Valor inicial do `PC`: informado como **primeiro argumento** na chamada da MV.
- Valor inicial do `SP`: informado como **segundo argumento** na chamada da MV.
- Posição da memória a partir da qual o programa será carregado: informado como **terceiro argumento** na chamada da MV.
- Modo de saída de dados [s|v]: informado como **quarto argumento** na chamada da MV.
- Nome do arquivo contendo o programa a ser executado pela máquina virtual: informado como **quinto argumento** na chamada da MV.

Exemplo:

```
./mv 0 500 10 v teste1
```

A chamada acima tem a seguinte semântica: executar a máquina virtual, inicializando `PC` com o valor 0, `SP` com valor 500, carregar a partir do endereço 10 o programa escrito no arquivo teste1 e em seguida interpretar suas instruções. Além disso, foi informado à MV para exibir os dados de execução a cada passo do programa.

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

Possivelemente, os testes de funcionamento da MV 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 da MV, não podendo afetar a interface de entrada e saída.