

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.

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 pelo professor em sala de aula.
- 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 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 única forma de endereçamento existente na máquina é direto, relativo ao PC.
- As instruções são codificadas em um inteiro. A maioria delas possui um operando, que é uma posição de memória (M), especificada conforme o modo de endereçamento existente, também codificados em inteiros. As exceções são as instruções RET e HALT, que não possuem operandos. O conjunto de instruções da máquina está detalhado na Tabela 1.

Código	Símbolo	Significado	Ação
01	LOAD	Carrega AC	$AC \leftarrow \text{Memória}[PC+M]$
02	STORE	Armazena AC	$\text{Memória}[PC+M] \leftarrow AC$
03	JMP	Desvio incondicional	$PC \leftarrow PC + M$
04	JPG	Desvia se maior que 0	Se $AC > 0$, $PC \leftarrow PC + M$
05	JPE	Desvia se igual a 0	Se $AC = 0$, $PC \leftarrow PC + M$
06	JPL	Desvia se menor que 0	Se $AC < 0$, $PC \leftarrow PC + M$
07	JPNE	Desvia se diferente de 0	Se $AC \neq 0$, $PC \leftarrow PC + M$
08	PUSH	Empilha valor	$SP \leftarrow SP - 1$; $\text{Memória}[SP] \leftarrow \text{Memória}[PC+M]$
09	POP	Desempilha valor	$\text{Memória}[PC+M] \leftarrow \text{Memória}[SP]$; $SP \leftarrow SP + 1$;
10	READ	Lê valor para a memória	$\text{Memória}[PC+M] \leftarrow \text{“Valor lido”}$
11	WRITE	Escreve conteúdo da memória	“Imprime” $\text{Memória}[PC+M]$
12	CALL	Chamada de subrotina	$SP \leftarrow SP - 1$; $\text{Memória}[SP] \leftarrow PC$; $PC \leftarrow PC + M$
13	RET	Retorno de subrotina	$PC \leftarrow \text{Memória}[SP]$; $SP \leftarrow SP + 1$
14	ADD	Soma operando em AC	$AC \leftarrow AC + \text{Memória}[PC+M]$
15	SUB	Subtrai operando de AC	$AC \leftarrow AC - \text{Memória}[PC+M]$
16	XOR	XOR lógico	$AC \leftarrow AC \text{ xor } \text{Memória}[PC+M]$
17	AND	AND lógico	$AC \leftarrow AC \text{ and } \text{Memória}[PC+M]$
18	OR	OR lógico	$AC \leftarrow AC \text{ or } \text{Memória}[PC+M]$
19	HALT	Parada	

Tabela 1: Conjunto de instruções da máquina virtual.

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 num formato livre (“executável”, a ser definido por cada aluno, em sua implementação, e devidamente documentado) que contenha as instruções propriamente ditas, além dos dados para a inicialização da máquina de forma que o programa funcione.

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

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:

```
10
15
10
14
01
11
15
10
04
04
11
06
03
02
11
01
19
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 e não deve ser utilizado como único teste da MV.

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 testel
```

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 `testel` 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, 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

Possivelmente, 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.