

Trabalho Prático 1 - Emulador

1 Descrição Geral

Esse trabalho envolve a implementação de um emulador 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 emulador, então implemente-o com cuidado pois esta máquina virtual pode impactar na nota de todos os 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 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 “`tp1_seuNOME.tar.gz`”, disponível para download no Moodle;
- **ATENÇÃO:** trabalhos que não seguem esse padrão serão penalizados.

3 Especificação da Máquina de Khattab

A máquina a ser emulada é a **Máquina de Khattab**, projetada exclusivamente para a disciplina. Seguem as especificações:

- 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, 3 registradores de propósito específico e 8 registradores de propósito geral;
- Os registradores de propósito específico são:
 - PC (contador de programas): contém o endereço da próxima instrução a ser executada;
 - AP (apontador da pilha): aponta para o elemento no topo da pilha;
 - PEP (palavra de status do processador): consiste em 2 bits que armazenam o estado da última operação lógico/aritmética realizada na máquina, sendo um dos bits para indicar que a última operação resultou em zero, e outro bit para indicar que a última operação resultou num resultado negativo;
- Os registradores de propósito geral são indexados por um valor que varia de 0 a 7;
- A única forma de endereçamento existente na máquina é direto, relativo ao PC;
- As instruções **READ** e **WRITE** leem e escrevem um inteiro na saída padrão do emulador;
- As instruções são codificadas em um inteiro, podendo ter dois, um ou nenhum operando, que é o caso das instruções **RET** e **HALT**.

- Os operandos podem ser uma posição de memória (M, codificado como inteiro) ou um registrador (R, codificado como um inteiro entre 0 e 7).

O conjunto de instruções da Máquina de Khattab está detalhado na Tabela 1:

As operações marcadas com * atualizam o valor do PEP.

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	$Reg[R1] \leftarrow Reg[R1] \text{ AND } Reg[R2]$ *
04	OR	R1 R2	OR (bit a bit) de dois registradores	$Reg[R1] \leftarrow Reg[R1] \text{ OR } Reg[R2]$ *
05	XOR	R1 R2	XOR (bit a bit) de dois registradores	$Reg[R1] \leftarrow Reg[R1] \text{ XOR } Reg[R2]$ *
06	NOT	R1	NOT (bit a bit) de um registrador	$Reg[R1] \leftarrow \text{NOT } Reg[R1]$ *
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 $\neg PEP[zero]$, $PC \leftarrow PC + M$
10	JN	M	Desvia se negativo	Se $PEP[negativo]$, $PC \leftarrow PC + M$
11	JNN	M	Desvia se não negativo	Se $\neg PEP[negativo]$, $PC \leftarrow PC + M$
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	$Reg[R] \leftarrow \text{"valor lido"}$
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

Sua tarefa é implementar um programa interpretador que emule a Máquina de Khattab (MK), ou seja, que execute 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 AP 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 AP, e decresce incrementando o AP.

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.

5 Formato da Entrada de Dados

O programa a ser interpretado pela MV deverá ser escrito em um arquivo texto formado por um inteiro por linha, sendo esse valor uma instrução da máquina virtual ou um operando: uma posição de memória (M, codificado como inteiro) ou um registrador (R, codificado como um inteiro entre 0 e 7).

Exemplo de executável:

```
17 - Lemos 1 int (4 bytes), identifica instrução READ, PC = 1
0 - Lemos 1 int (4 bytes), ident. operando R de READ, PC = 2, instrução READ é executada
15 - Lemos 1 int (4 bytes), ident. instrução LOAD, PC = 3
1 - Lemos 1 int (4 bytes), ident. operando R de LOAD, PC = 4
6 - Lemos 1 int (4 bytes), ident. operando M de LOAD, PC = 5, instrução LOAD é executada
1 - Lemos 1 int (4 bytes), ident. instrução ADD, PC = 6
0 - Lemos 1 int (4 bytes), ident. operando R1 de ADD, PC = 7
1 - Lemos 1 int (4 bytes), ident. operando R2 de ADD, PC = 8, instrução ADD é executada
18 - Lemos 1 int (4 bytes), ident. instrução WRITE, PC = 9
0 - Lemos 1 int (4 bytes), ident. operando R de WRITE, PC = 10, instrução WRITE é executada
21 - Lemos 1 int (4 bytes), ident. instrução HALT, execução finalizada
100 - Valor armazenado na memória que não é executado, mas é lido por outra instrução
```

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.

O programa deve ser interpretado da seguinte forma: (1) leia um inteiro da entrada padrão (e.g. `scanf`) e armazene no registrador 0; (2) copie o conteúdo da posição de memória $PC + M$ (que neste caso, $M = 6$, $PC = 5$ e $PC + M = 11$) para o registrador 1; (2) Soma o valor do registrador 0 e do registrador 1 (no caso $R[1] = 100$) e armazene no registrador 0; (3) Imprime na entrada padrão (e.g. `printf`) o valor do registrador 0.

Observe que tal programa não testa todas as instruções, então outros programas devem ser obrigatoriamente implementados com o objetivo de cobrir todas instruções. A qualidade dos experimentos implementados será avaliada de alguma forma na correção.

6 Formato da Saída de Dados

Duas opções de saída devem estar disponíveis para utilização do emulador:

1. Simples: Imprime na tela **somente** o resultado do programa, que é o que o programa interpretado escrever na saída padrão, ou seja, quando executa a instrução `WRITE`.
2. Modo *verbose*: Imprime o passo a passo da execução, exibindo a cada instrução o valor atual de PC, AP, dos bits de PEP, dos registradores, 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 do Emulador

- O executável do emulador DEVE se chamar "emulador".
- Valor inicial do PC: informado como **primeiro argumento** na chamada da MV.
- Valor inicial do AP: 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:

```
./emulador 0 500 10 v teste.mv
```

A chamada acima tem a seguinte semântica: executar a máquina virtual, inicializando PC com o valor 0, AP com valor 500, carregar a partir do endereço 10 o programa escrito no arquivo teste.mv 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 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

POs 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.