Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Disciplina Software Básico (DCC008) - 2015/02

Trabalho Prático 1 - Emulador

1 Descrição Geral

O presente trabalho propõe a implementação de um emulador para uma máquina virtual básica que será utilizada ao longo do curso. Ele será utilizado para implementarmos diversos conceitos apresentados durante o curso.

Atenção: todos trabalhos da disciplina dependerão deste emulador, então faça-o com bastante cuidado, uma vez que esta máquina virtual pode impactar na nota de todos os demais trabalhos.

2 Informações Importantes

- O trabalho deve ser feito **individualmente**, podendo ser discutido entre os colegas, mas o código fonte não poderá ser compartilhado.
- 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, porém entregas em atrasos serão penalizadas. A penalização pelo atraso será geométrica com o mesmo conforme a fórmula a seguir:

$$Desconto(\%) = \frac{2^{d-1}}{0,32}$$

  A fórmula indica o percentual 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 em 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 tais decisões. O documento não precisa ser necessariamente 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 é necessário 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 feita via **Moodle**, na tarefa criada especificamente para tal. A entrega deve ser realizada estritamente no seguinte formato:
  - Pasta contendo os arquivos do trabalho. A pasta deve ser nomeada como `tp1_seulogin`, onde `seu_login` refere-se ao seu login no DCC. Esta pasta deve ser compactada no formato `.tar.gz`. Assim, o pacote final a ser enviado será o arquivo `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.
- **ATENÇÃO:** trabalhos entregues fora do padrão serão penalizados.

3 Especificação da Máquina Virtual

A máquina virtual (MV) a ser simulada foi projetada exclusivamente para a disciplina. Seguem as especificações:

- A menor unidade endereçável na 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 16 registradores de propósito geral;
- Os registradores de propósito específico são:
  - PC (contador de programas): contém o endereço da próxima instrução a ser executada;
  - SP (apontador da pilha): aponta para o elemento no topo da pilha;
  - PSW (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 15;

- A única forma de endereçamento existente na máquina é direta, relativa 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 15).

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

As operações marcadas com * atualizam o valor do PSW.

Cód	Símbolo	Operandos	Significado	Ação
01	LOAD	R M	Carrega Registrador	$Reg[R] \leftarrow Mem[M + PC]$
02	STORE	R M	Armazena Registrador	$Mem[M + PC] \leftarrow Reg[R]$
03	READ	R	Lê valor para registrador	$Reg[R] \leftarrow$ "valor lido"
04	WRITE	R	Escreve conteúdo do registrador	"Imprime" $Reg[R]$
05	COPY	R1 R2	Copia registrador	$Reg[R1] \leftarrow Reg[R2]$ *
06	NEG	R1	Inverte o sinal do registrador	$Reg[R1] \leftarrow -Reg[R1]$ *
07	SUB	R1 R2	Subtrai dois registradores	$Reg[R1] \leftarrow Reg[R1] - Reg[R2]$ *
08	ADD	R1 R2	Soma dois registradores	$Reg[R1] \leftarrow Reg[R1] + Reg[R2]$ *
09	AND	R1 R2	AND (bit a bit) de dois registradores	$Reg[R1] \leftarrow Reg[R1]$ AND $Reg[R2]$ *
10	OR	R1 R2	OR (bit a bit) de dois registradores	$Reg[R1] \leftarrow Reg[R1]$ OR $Reg[R2]$ *
11	XOR	R1 R2	XOR (bit a bit) de dois registradores	$Reg[R1] \leftarrow Reg[R1]$ XOR $Reg[R2]$ *
12	NOT	R1	NOT (bit a bit) de um registrador	$Reg[R1] \leftarrow$ NOT $Reg[R1]$ *
13	JMP	M	Desvio incondicional	$PC \leftarrow PC + M$
14	JZ	M	Desvia se zero	Se $PSW[zero], PC \leftarrow PC + M$
15	JNZ	M	Desvia se não zero	Se $!PSW[zero], PC \leftarrow PC + M$
16	JN	M	Desvia se negativo	Se $PSW[negativo], PC \leftarrow PC + M$
17	JNN	M	Desvia se não negativo	Se $!PSW[negativo], PC \leftarrow PC + M$
18	PUSH	R	Empilha valor do registrador	$SP \leftarrow SP - 1$ $Mem[SP] \leftarrow Reg[R]$
19	POP	R	Desempilha valor no registrador	$Reg[R] \leftarrow Mem[SP]$ $SP \leftarrow SP + 1$
20	CALL	M	Chamada de subrotina	$SP \leftarrow SP - 1$ $Mem[SP] \leftarrow PC$ $PC \leftarrow PC + M$
21	RET		Retorno de subrotina	$PC \leftarrow Mem[SP]$ $SP \leftarrow SP + 1$
22	HALT		Parada	

Tabela 1: Instruções da Máquina Virtual

4 Descrição da Tarefa

Sua tarefa é implementar um programa interpretador que emule a Máquina de Khattab (MK), 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 contém 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 15).

Exemplo de executável:

```
3 - 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
1 - 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
8 - 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
4 - 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
22 - 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`, `SP`, dos bits de `PSW`, 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 SP: informado como **segundo argumento** na chamada da MV.
- Posiçao da memória a partir da qual o programa sera carregado: informado como **terceiro argumento** na chamada da MV.
- Nome do arquivo contendo o programa a ser executado pela máquina virtual: informado como **quarto argumento** na chamada da MV.
- Modo de saída de dados [s|v]: informado como **quinto argumento** na chamada da MV. Parâmetro opcional, caso não seja informado o modo de saída deve ser o *simples*.

Exemplos:

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

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

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.