UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

TRABALHO PRÁTICO – SOFTWARE BÁSICO – 2019/1º

Este documento descreve o trabalho prático que será usado para treinar e fixar os conceitos aprendidos em sala de aula na disciplina Software Básica. O trabalho consiste de duas etapas: um emulador de uma máquina virtual básica, proposta para facilitar a implementação; e um montador para essa máquina, que permita a usuários escreverem programas em assembly para essa máquina básica.

CONSIDERAÇÕES GERAIS

- O trabalho deverá ser implementado obrigatoriamente na linguagem C/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;
- A entrega do trabalho deverá ser realizada por meio do Moodle, na tarefa criada especificamente para tal;
- ATENÇÃO: trabalhos que não seguem esse padrão serão penalizados.

ESPECIFICAÇÃO DA MÁQUINA VIRTUAL

A máquina virtual (MV) a ser emulada foi 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 4 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 estado 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 3;
- A única forma de endereçamento existente na máquina é direto, relativo ao PC;

- As instruções READ e WRITE lêem 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 3).

O conjunto de instruções da Máquina Virtual está detalhado na tabela abaixo. As instruções marcadas com * atualizam o estado do PEP.

Cód.	**Símb.**	**Oper.**	**Definição**	**Ação**
0	HALT		Parada	
1	LOAD	R M	Carrega Registrador	Reg[R] ← Mem[M+PC]
2	STORE	R M	Armazena Registrador	Mem[M+PC] ← Reg[R]
3	READ	R	Lê valor para registrador	Reg[R] ← "valor lido"
4	WRITE	R	Escreve conteúdo do registrador	"Imprime" Reg[R]
5	COPY	R1 R2	Copia registrador	Reg[R1] ← Reg[R2] *
6	PUSH	R	Empilha valor do registrador	AP ← AP−1; Mem[AP] ← Reg[R]
7	POP	R	Desempilha valor no registrador	Reg[R] ← Mem[AP]; AP ← AP+1
8	JUMP	M	Desvio incondicional	PC←PC+M
9	JZ	M	Desvia se zero	Se PEP [zero], PC ← PC+M
10	JNZ	M	Desvia se não zero	Se !PEP [zero], PC ← PC+M
11	JN	M	Desvia se negativo	Se PEP [negativo], PC ← PC+M
12	JNN	M	Desvia se não negativo	Se !PEP [negativo], PC ← PC+M
13	CALL	M	Chamada de subrotina	AP ← AP – 1; Mem[AP] ← PC; PC←PC+M
14	RET		Retorno de subrotina	PC ← Mem[AP]; AP ← AP+1
15	AND	R1 R2	AND (bit a bit) de dois registradores	Reg[R1] ← Reg[R1] AND Reg[R2] *
16	OR	R1 R2	OR (bit a bit) de dois registradores	Reg[R1] ← Reg[R1] OR Reg[R2] *
17	NOT	R1	NOT (bit a bit) de um registrador	Reg[R1] ← NOT Reg[R1] *
18	XOR	R1 R2	XOR (bit a bit) de dois registradores	Reg[R1] ← Reg[R1] XOR Reg[R2] *
19	ADD	R1 R2	Soma dois registradores	Reg[R1] ← Reg[R1] + Reg[R2] *
20	SUB	R1 R2	Subtrai dois registradores	Reg[R1] ← Reg[R1] − Reg[R2] *
21	MUL	R1 R2	Multiplica dois registradores	Reg[R1] ← Reg[R1] × Reg[R2] *
22	DIV	R1 R2	Dividendo entre dois registradores	Reg[R1] ← dividendo(Reg[R1] ÷ Reg[R2]) *
23	MOD	R1 R2	Resto entre dois registradores	Reg[R1] ← resto(Reg[R1] ÷ Reg[R2]) *
24	CMP	R1 R2	Compara dois registradores	Reg[R1] – Reg[R2] *
25	TST	R1 R2	Testa dois registradores	Reg[R1] AND Reg[R2] *

O FORMATO DE ENTRADA DO EMULADOR (LOADER)

Seu emulador deve rodar em linha de comando, e receberá como entrada um arquivo executável para a Máquina Virtual com o seguinte formato:

- O arquivo é armazenado em modo texto, com uma série de inteiros representados em decimal.
- No arquivo executável, os inteiros correspondentes aos códigos de operação e operandos da máquina, serão precedidos por 4 inteiros adicionais, também codificados em decimal, formato texto, a dizer:
  - Tamanho do programa: em inteiros – posições de memória;

  - Endereço de carregamento: posição na memória onde o programa deverá ser inicialmente carregado;
  - Valor inicial da pilha: inicialização do registrador AP;
  - Entry Point do programa: posição de memória onde a execução do programa deve começar – inicialização do registrador PC;
- O arquivo executável tem um cabeçalho de identificação que contem a seguinte linha de texto sozinha. Se essa linha não for encontrada no arquivo, o loader deve acusar um erro de formato não-executavel.
  - "MV-EXE↵" (fim de linha)

Como exemplo, note o arquivo abaixo:

```
MV-EXE
12 100 999 100
3 0 1 1 6 19 0 1 4 0 0 100
```

Esse arquivo contém um programa que lê um valor da entrada, e imprime o valor recebido + 100 na saída. Use esse arquivo como teste de seu loader e simulador. Note, no entanto, que esse pequeno programa não testa todas as instruções, portanto, novos testes precisam ser realizados para garantir o funcionamento do programa.

SAÍDA DO EMULADOR

O emulador deverá possuir pelo menos dois modos de operação definidos por linha de comando (opção -v), a dizer:

- Modo simples: a saída do emulador é somente aquilo que o programa que está rodando mandar imprimir, ou, possivelmente, alguma mensagem de erro do emulador;
- Modo verbose: nesse modo, o emulador deverá imprimir, a cada instrução, a operação que está sendo executada, acrescido de um dump dos 7 registradores (3+4) presentes na arquitetura. Esse modo deverá ser usado para depuração e testes do trabalho prático.

ESPECIFICAÇÃO DO MONTADOR

Ainda como parte do trabalho, um montador de dois passos deve ser implementado para a Máquina Virtual. Abaixo, segue a especificação desse montador:

- A linguagem simbólica é bastante simples, e cada linha terá o seguinte formato:
  [<label>:] <operador> <operando1> <operando2> [; comentário]

  Ou seja:
  - Se houver algum label, ele será definido no início da linha e deverá terminar com ":";
  - A presença do operador é obrigatória;
  - Os operandos dependem da instrução, algumas instruções não possuem operando, enquanto outras tem 1 operando e outras 2 operandos;
  - Podem haver comentários no fim da linha, sendo que devem começar por ";";
  - Os operandos podem ser um registrador (R0, R1, R2 ou R3) ou uma posição de memória no programa (identificada pelo label);

  - Os registradores de R0 a R3 são respectivamente 0, 1, 2 ou 3 em linguagem de máquina;
  - A posição de memória dos desvios e instruções LOAD/STORE é relativa ao PC após a leitura do operando.

- O conjunto de instruções é o mesmo descrito acima;
- Deverão ser acrescentadas duas pseudo-instruções: WORD, e END, com descrições abaixo:
  - WORD I → Reserva a posição de memória e a inicializa com o valor I (inteiro);
  - END → Indica o final do programa para o montador.
- Espaços em branco extras entre o label, operador, operandos e comentário são permitidos, não devendo afetar o funcionamento do montador;
- Podem existir linhas vazias, inclusive linhas apenas com comentários, que devem ser ignoradas pelo montador.

Como teste do montador, utilize o programa abaixo:

```
READ R0
READ R1
STORE R0 A
SUB R0 R1
LOAD R0 A
JN MB
COPY R2 R0
JUMP L
MB: COPY R2 R1
L: WRITE R0
WRITE R1
WRITE R2
HALT
A: WORD 0
END
```

O montador deverá receber como linha de comando o nome do arquivo texto contendo o programa em assembly e deverá gerar na saída padrão o arquivo executável conforme descrito acima, de entrada para o loader do emulador.

SOBRE A DOCUMENTAÇÃO ANEXA

Juntamente com o trabalho, cada aluno deverá submeter um documento com informações relevantes sobre seu trabalho, a dizer:

- Deve conter as decisões importantes tomadas nas definições do projeto que por ventura não estejam contempladas na especificação do trabalho
- Deve conter informação de como executar cada um dos programas. É importante que os formatos de entrada e de saída especificados sejam cumpridos em virtude da automatização na correção dos trabalhos.
- Não é importante incluir listagem de código fonte nessa descrição

- Deve conter elementos que comprovem que o programa foi testado com alguma profundidade.

PROGRAMAS DE TESTE A SEREM DESENVOLVIDOS EM ASSEMBLY

Como teste do trabalho, implemente os seguintes programas em assembly da Máquina Virtual:

1. Mediana: programa que lê 5 números e imprime a mediana deles:
   Ex. Valores = [1, 77, 25, 59, 20] --- Mediana = 25

2. Fibonacci: programa que lê um número N da entrada e imprime o N-ésimo número de Fibonacci:
   Ex. Valor = [8] --- resultado = 13