2018/02
Software Básico
Universidade Federal de Minas Gerais
Ciência da Computação
André Gomes Heringer

Documentação Trabalho Prático 1 - Emulador Software Básico

Introdução

Durante as aulas de software básico foram discutidas as diferentes maneiras de realizar operações definidas em uma camadas mais abstrata de software para uma camada mais próxima da linguagem de máquina, chamadas de interpretação (quando um programa lê instrução por instrução da camada abstrata e executar cada comando na camada mais baixa) e tradução (quando um novo programa é gerado, na linguagem mais baixa, a partir das instruções da camada mais abstrata).

Foi pedido que os alunos implementação um programa que emulasse e interpretasse uma Máquina de Khattab. Definada da seguinte maneira:

- 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).

As instruções da Máquina de Khattab são descritas pela tabela a seguir:

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

Implementação

Para a implementação do emulador foi criadas as seguintes estruturas de dados:

- Um vetor de 1000 posições para servir como memória
- Um vetor de 8 posições que representam os registradores de propósito geral

- Um inteiro que representa PC
- Um inteiro que representa o início da pilha, SP
- Um vetor de char com duas posições que representa o PEP

O problema pode ser dividido em basicamente duas partes, na primeira delas é preciso carregar o programa na memória de maneira adequada, a partir de um arquivo de entrada. Posteriormente é preciso ler o programa da memória e executar os comandos da Máquina de Khattab de maneira como foi especificado acima na tabela.

Foram implementados dois módulos separadas para realizar cada uma das funções descritas acima. O módulo *"loader.c"* para carregar o programa na memória e tratar possíveis erros na entrada. Este módulo tem como interface uma única função:

- int input_parser: Retorna 0 se bem sucedido e 1 caso contrário. Recebe como parâmetros o local de inicio da memória, o início da pilha o arquivo de entrada e o vetor da memória.

O segundo módulo chamado *"emu.c"* contém a lógica do emulador propriamente dito organizado em uma função chamada "*execution_loop*" que serve como loop principal do programa, este módulo também contém duas funções para facilitar a execução (execute_command e update_pep).

- int execution_loop: Retorna 0 caso a execução não tenha erros e 1 caso contrário, e recebe um argumento que define um modo verboso.
- int execute_command: Executa uma instrução da Maquina de Khattab, recebe o código da instrução, retorna 1 depois de executar a instrução com sucesso e 0 quando executa a instrução de HALT especificamente.
- void update_pep: Função simples que atualiza o PEP de acordo com o retorno da última instrução. Recebe o resultado a última instrução.

Nesta implementação o interpretador assume que após executar uma instrução a mesma irá redirecionar o valor de PC para o início da próxima instrução dessa maneira no caso de uma instrução JUMP o interpretador realizará as seguintes operações:

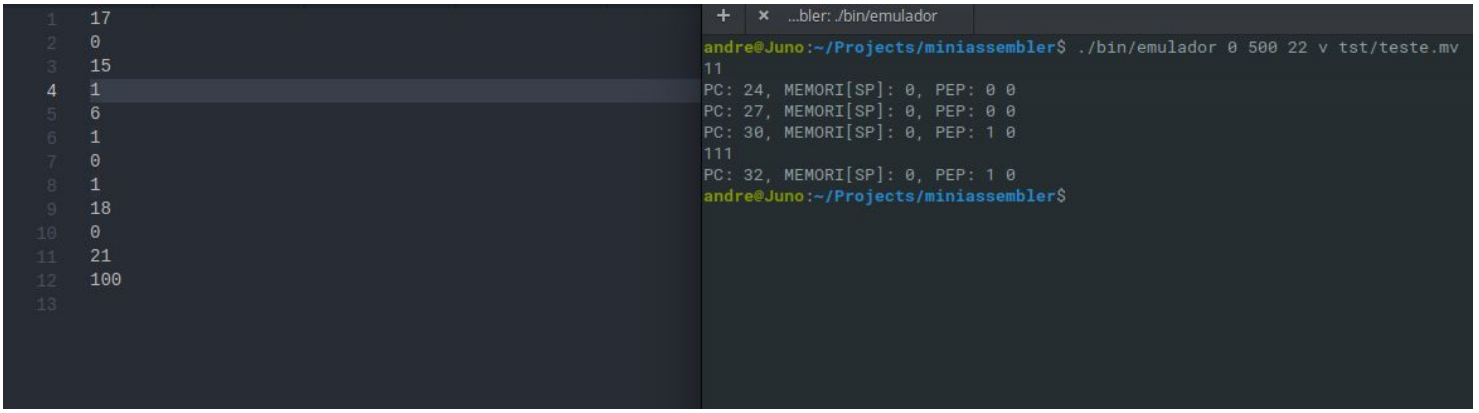
1. Incrementa o PC
2. Lê o valor do parâmetro M e soma o valor de M à PC

Testes

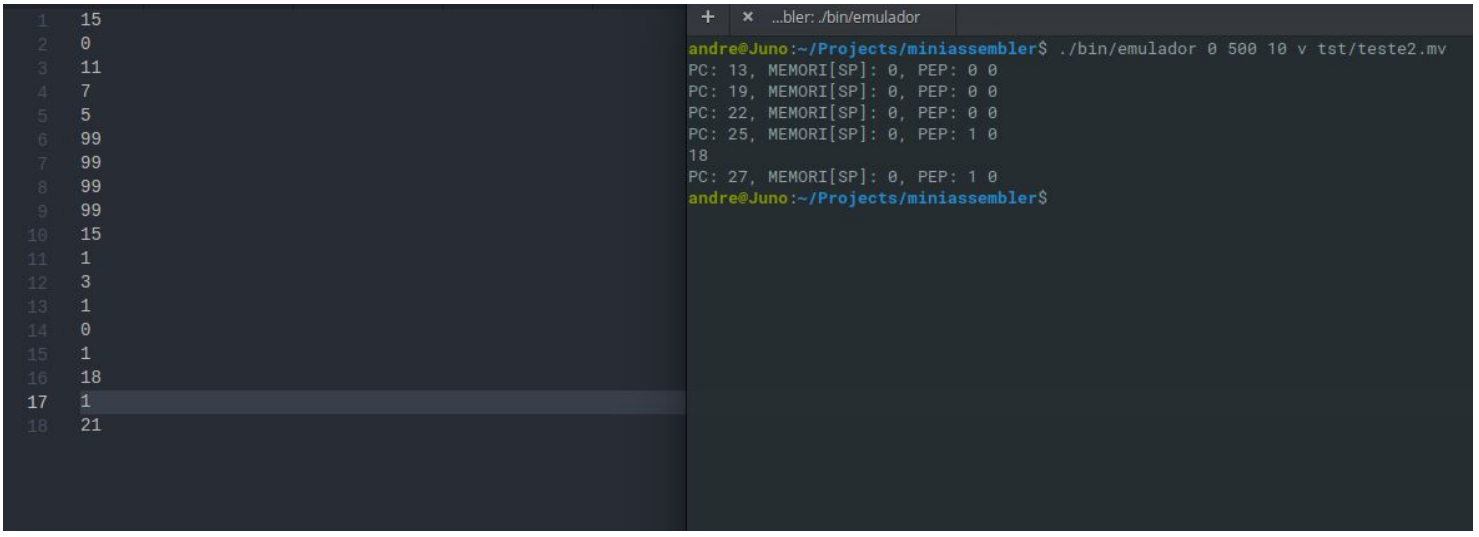
Para os testes realizados o programa foi compilado a partir do diretório base usando o comando *"make"* usando o compilador GCC 7.3.0 e os testes rodam em uma máquina com o sistema operacional Elementary OS 5 (Juno) baseado no Ubuntu 18.04 LTS.

Em cada um dos casos o arquivo de input é descrito a esquerda e o respectivo output gerado pelo programa a direita.

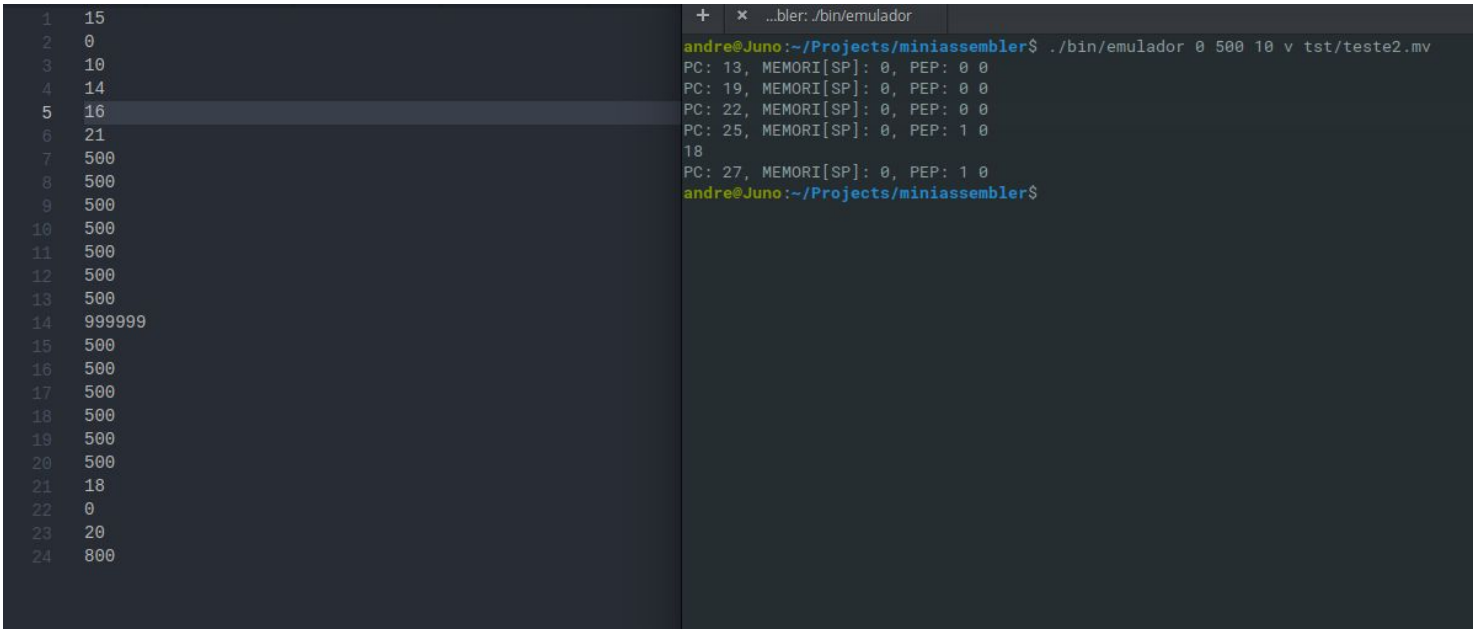
Caso de Teste 1:



Caso de Teste 2:



Caso de Teste 3:



Os testes descritos acima se encontram dentro do diretório “/tst” e podem ser reproduzidos usando os comandos que se encontram em cada um dos casos e seus respectivos parâmetros.