TRABALHO PRÁTICO 1:
Máquina Virtual

Pedro Lopes Miranda Junior

[1]Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG)

plmj@dcc.ufmg.br

Resumo. *Este trabalho constitui na implementação de uma máquina básica capaz de executar um código passado em linguagem de máquina.*

1. Introdução

Ao longo do curso serão estudados diversos conceitos sobre liguagem de máquina. Perante isso, serão dados diversos trabalhos que testarão alguns dos inúmeros conceitos abordados.

O primeiro trabalho tem como objetivo implementar o simulador para uma máquina virtual básica, capaz de decodificar algumas funções simples. Esse trabalho é de extrema importância pois a partir dele todos os demais trabalhos serão executados.

A máquina virtual é constituída de registradores de uso geral, uma memória de no mínimo 1024 espaços, e registradores de uso especial e todas essas estruturas são capazes de endereçar o tamanho de um inteiro (4 bytes).

Ao todo a máquina virtual pode decodificar 23 funções, divididas em acesso à memória, operações lógicas, aritmeticas e saltos, condicionais ou não).

2. Solução Proposta

Para solucionar o problema, foi criada uma estrutura de dados que chamada vm (Virtual Machine), que contem a memória, os registradores e outras ifnofrmações adicionais, como está descrita abaixo.

2.1. Estruturas de dados

2.1.1. VM

1: Estrutura da máquina virtual

mem[MEMSIZE],m,fmem; *Memória, endereços de memória disponíveis e endereço de memória final*
pc,sp,fp; *Registradores especificos: PC,SP e FP*
reg[4]; *Registradores de uso geral: reg[0]=RA, reg[1]=RB, reg[2]=RB e reg[3]=RD*
rflags[2]; *Registradores de flag: rflags[0]=zero,rflags[1]=negativo*
char mode; *Carácter que indica o modo de impressão do código*

2.2. Algoritmos

Foram implementadas algumas funções que auxiliam no funcionamento da máquina virtual, portanto todas as funções operam sobre o tipo VM.

2.2.1. Funções

2: Funções da máquina virtual
initialize; *Inicialisa a máquina virtual com os parâmetros passados*
readmem; *Faz a leitura do programa para a memória*
exec; *Executa a máquina virtual*
getinst; *Busca a próxima instrução a ser executada*
decode; *Decodifica e executa a instrução buscada*
incpc; *Incrementa o registrador PC*
verbose; *Imprime as informações quando no modo verbose*

3. Implementação

Na implementação do problema proposto foram tomadas várias decisões, dentre elas criar um tipo de dados para a máquiva virtual, dividir o código em funções de modo que na função principal fique o menor conteúdo possível ajudando no encapsulamento do código e em futuras manutenções e melhorias.

Foi decidido criar um tipo de dados para a máquina virtual pois esse tipo de disposição facilita a adição futura de demais estruturas na máquina virtual. Além disso o acesso ao TAD é feito de maneira melhor estruturada e o encapsulamento é melhor feito.

A divisão do código em funções ajuda no encapsulamento do TAD, e na melhor modularização do mesmo.

3.1. Código

O código foi dividido em arquivos *.c* e *.h* que estão listados abaixo

3.1.1. Arquivos .c

- **vm.c:** Contém a função principal da máquina virtual;
- **vmdata.c:** Contém as funções do tipo VM;

3.1.2. Arquivos .h

- **vmdata.h:** Contém as definições das funções do tipo VM

3.2. Compilação

O programas deve ser compilado através de um makefile, chamando *vm* ou através do compilador GCC chamando:

```
gcc -Wall vm.c vmdata.c -o vm
```

3.3. Execução

Para a execução do programa deverão ser recebidos,impreterívelmente, como parâmetros o valor inicial do PC, valor inicial - tamanho - do SP, posição de memória inicial, o modo de saída - S ou V - dos dados e o nome do arquivo com o código a ser executado.

O comando para execução do programa é da forma:

```
./vm 10 250 5 <modo> <arquivo>
```

3.3.1. Formato da entrada

O arquivo de entrada citado deverá ser um programa em linguagem de máquina, onde cada linha conterá uma instrução, como abaixo:

```
01
30
90
01
30
11
00
25
11
01
24
42
01
00
```

3.3.2. Formato da saída

O formato da saída depende da especificação do modo de execução do programa.
Em modo simples, o programa apenas imprimirá os comandos de impressão na saída padrão, já em modo verbose a forma de impressão será após cada instrução, tendo cada linha os valores de cada um dos parametro, como consta no exemplo abaixo:

```
INST: XX; PC: XX; SP: XX; FP: XX;
Flag.Zero: X; Flag.Neg: X; RA: XX;
RB: XX; RC: XX; RD: XX;
```

4. Avaliação Experimental

O programa foi executado e testado numa máquina rodando o sistema baseado em debian Linux Mint. A máquina em questão tem uma memória de 8GB e um processador Core I7 de 2.2GHz.

Foram feitos vários testes de execução do programa,onde todas as funções testadas funcionaram de maneira rápida e sem erro. Abaixo apresentamos imagens da execução do programa e sua saída em modo verbose para um dos testes criados.

```
./vm 28 1000 122 v input/teste >> saida.txt
```

Figura 1. Comando de execução do programa

```
3296 INST: 60; PC: 16; SP: 1000; FP: 0; Flag.Zero: 0; Flag.Neg: 0; RA: 6636; RB: 7; RC: 4366488; RD: 1;
3297 INST: 41; PC: 19; SP: 1000; FP: 0; Flag.Zero: 0; Flag.Neg: 0; RA: 6636; RB: 7; RC: 4366488; RD: 1;
3298 INST: 51; PC: 21; SP: 1000; FP: 0; Flag.Zero: 0; Flag.Neg: 0; RA: 6636; RB: 7; RC: 4366488; RD: 1;
3299 INST: 22; PC: 24; SP: 1000; FP: 0; Flag.Zero: 0; Flag.Neg: 0; RA: 6636; RB: 6; RC: 4366488; RD: 1;
3300 INST: 21; PC: 27; SP: 1000; FP: 0; Flag.Zero: 0; Flag.Neg: 0; RA: 6636; RB: 6; RC: 4373124; RD: 1;
3301 INST: 60; PC: 16; SP: 1000; FP: 0; Flag.Zero: 0; Flag.Neg: 0; RA: 6636; RB: 6; RC: 4373124; RD: 1;
3302 INST: 41; PC: 19; SP: 1000; FP: 0; Flag.Zero: 0; Flag.Neg: 0; RA: 6636; RB: 6; RC: 4373124; RD: 1;
3303 INST: 51; PC: 21; SP: 1000; FP: 0; Flag.Zero: 0; Flag.Neg: 0; RA: 6636; RB: 6; RC: 4373124; RD: 1;
3304 INST: 22; PC: 24; SP: 1000; FP: 0; Flag.Zero: 0; Flag.Neg: 0; RA: 6636; RB: 5; RC: 4373124; RD: 1;
3305 INST: 21; PC: 27; SP: 1000; FP: 0; Flag.Zero: 0; Flag.Neg: 0; RA: 6636; RB: 5; RC: 4379760; RD: 1;
3306 INST: 60; PC: 16; SP: 1000; FP: 0; Flag.Zero: 0; Flag.Neg: 0; RA: 6636; RB: 5; RC: 4379760; RD: 1;
3307 INST: 41; PC: 19; SP: 1000; FP: 0; Flag.Zero: 0; Flag.Neg: 0; RA: 6636; RB: 5; RC: 4379760; RD: 1;
3308 INST: 51; PC: 21; SP: 1000; FP: 0; Flag.Zero: 0; Flag.Neg: 0; RA: 6636; RB: 5; RC: 4379760; RD: 1;
3309 INST: 22; PC: 24; SP: 1000; FP: 0; Flag.Zero: 0; Flag.Neg: 0; RA: 6636; RB: 4; RC: 4379760; RD: 1;
3310 INST: 21; PC: 27; SP: 1000; FP: 0; Flag.Zero: 0; Flag.Neg: 0; RA: 6636; RB: 4; RC: 4386396; RD: 1;
3311 INST: 60; PC: 16; SP: 1000; FP: 0; Flag.Zero: 0; Flag.Neg: 0; RA: 6636; RB: 4; RC: 4386396; RD: 1;
3312 INST: 41; PC: 19; SP: 1000; FP: 0; Flag.Zero: 0; Flag.Neg: 0; RA: 6636; RB: 4; RC: 4386396; RD: 1;
3313 INST: 51; PC: 21; SP: 1000; FP: 0; Flag.Zero: 0; Flag.Neg: 0; RA: 6636; RB: 4; RC: 4386396; RD: 1;
3314 INST: 22; PC: 24; SP: 1000; FP: 0; Flag.Zero: 0; Flag.Neg: 0; RA: 6636; RB: 3; RC: 4386396; RD: 1;
3315 INST: 21; PC: 27; SP: 1000; FP: 0; Flag.Zero: 0; Flag.Neg: 0; RA: 6636; RB: 3; RC: 4393032; RD: 1;
3316 INST: 60; PC: 16; SP: 1000; FP: 0; Flag.Zero: 0; Flag.Neg: 0; RA: 6636; RB: 3; RC: 4393032; RD: 1;
3317 INST: 41; PC: 19; SP: 1000; FP: 0; Flag.Zero: 0; Flag.Neg: 0; RA: 6636; RB: 3; RC: 4393032; RD: 1;
3318 INST: 51; PC: 21; SP: 1000; FP: 0; Flag.Zero: 0; Flag.Neg: 0; RA: 6636; RB: 3; RC: 4393032; RD: 1;
3319 INST: 22; PC: 24; SP: 1000; FP: 0; Flag.Zero: 0; Flag.Neg: 0; RA: 6636; RB: 2; RC: 4393032; RD: 1;
3320 INST: 21; PC: 27; SP: 1000; FP: 0; Flag.Zero: 0; Flag.Neg: 0; RA: 6636; RB: 2; RC: 4399668; RD: 1;
3321 INST: 60; PC: 16; SP: 1000; FP: 0; Flag.Zero: 0; Flag.Neg: 0; RA: 6636; RB: 2; RC: 4399668; RD: 1;
3322 INST: 41; PC: 19; SP: 1000; FP: 0; Flag.Zero: 0; Flag.Neg: 0; RA: 6636; RB: 2; RC: 4399668; RD: 1;
3323 INST: 51; PC: 21; SP: 1000; FP: 0; Flag.Zero: 0; Flag.Neg: 0; RA: 6636; RB: 2; RC: 4399668; RD: 1;
3324 INST: 22; PC: 24; SP: 1000; FP: 0; Flag.Zero: 0; Flag.Neg: 0; RA: 6636; RB: 1; RC: 4399668; RD: 1;
3325 INST: 21; PC: 27; SP: 1000; FP: 0; Flag.Zero: 0; Flag.Neg: 0; RA: 6636; RB: 1; RC: 4406304; RD: 1;
3326 INST: 60; PC: 16; SP: 1000; FP: 0; Flag.Zero: 0; Flag.Neg: 0; RA: 6636; RB: 1; RC: 4406304; RD: 1;
3327 INST: 41; PC: 19; SP: 1000; FP: 0; Flag.Zero: 0; Flag.Neg: 0; RA: 6636; RB: 1; RC: 4406304; RD: 1;
3328 INST: 51; PC: 21; SP: 1000; FP: 0; Flag.Zero: 0; Flag.Neg: 0; RA: 6636; RB: 1; RC: 4406304; RD: 1;
3329 INST: 22; PC: 24; SP: 1000; FP: 0; Flag.Zero: 0; Flag.Neg: 0; RA: 6636; RB: 0; RC: 4406304; RD: 1;
3330 INST: 21; PC: 27; SP: 1000; FP: 0; Flag.Zero: 0; Flag.Neg: 0; RA: 6636; RB: 0; RC: 4412940; RD: 1;
3331 INST: 60; PC: 16; SP: 1000; FP: 0; Flag.Zero: 0; Flag.Neg: 0; RA: 6636; RB: 0; RC: 4412940; RD: 1;
3332 INST: 41; PC: 19; SP: 1000; FP: 0; Flag.Zero: 1; Flag.Neg: 0; RA: 6636; RB: 0; RC: 4412940; RD: 1;
3333 INST: 51; PC: 29; SP: 1000; FP: 0; Flag.Zero: 1; Flag.Neg: 0; RA: 6636; RB: 0; RC: 4412940; RD: 1;
3334 INST: 12; PC: 32; SP: 1000; FP: 0; Flag.Zero: 1; Flag.Neg: 0; RA: 6636; RB: 0; RC: 4412940; RD: 1;
3335 4412940
3336 INST: 2; PC: 34; SP: 1000; FP: 0; Flag.Zero: 1; Flag.Neg: 0; RA: 6636; RB: 0; RC: 4412940; RD: 1;
3337 INST: 90; PC: 35; SP: 1000; FP: 0; Flag.Zero: 1; Flag.Neg: 0; RA: 6636; RB: 0; RC: 4412940; RD: 1;
```

Figura 2. Saída usando verbose

5. Conclusão

O trabalho correu sem grandes problemas, sendo a parte mais difícil o entendimento do funcionamento de algumas funções, pois qualquer coisa errada poderá influenciar totalmente os demais trabalhos da disciplina.

O programa atendeu a diversos valores de entrada e creio que a solução proposta atenda ao especificado