UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ARTES, CIÊNCIAS E HUMANIDADES

EXERCÍCIO DE PROGRAMAÇÃO
ORGANIZAÇÃO DE COMPUTADORES DIGITAIS
RELATÓRIO

André Fillipe Cavicchiolli **NUSP:** 9878002
Bryan Munekata **NUSP:** 9911444
Rodrigo Rossi dos Santos **NUSP:** 9844828
Tiago de Luna Farias **NUSP:** 9875503

SUMÁRIO

CÓDIGO

M.java

Essa classe foi criada para utilizarmos como lista ligada no desenvolvimento da classe principal.

ocd_ep2.java

VARIÁVEIS

AX, BX, CX, DX, PC - variáveis do assembly onde os valores e as operações serão armazenadas.

MÉTODOS

decimalParaBinario

Como o próprio nome do método sugere, ela transforma o parâmetro dado em decimal para binário.

binarioParaDecimal

Como o próprio nome do método sugere, ela transforma o parâmetro dado em binário para decimal.

compilaCode

Esse método recebe como parâmetro um arquivo e vai lendo o código assembly contido nele transformando cada comando em um binário. Esse binário, posteriormente, será utilizado para rodar o código dentro da memória que desenhamos.

inc

Esse método implementa a operação de incremento do Assembly.

add

Esse método implementa a operação de adição do Assembly.

sub

Esse método implementa a operação de subtração do Assembly.

mul

Esse método implementa a operação de multiplicação do Assembly.

div

Esse método implementa a operação de divisão do Assembly.

cmp

Esse método implementa a operação de comparação do Assembly.

je

Esse método implementa a operação de *jump equal* do Assembly.

jne

Esse método implementa a operação de *jump not equal* do Assembly.

jg

Esse método implementa a operação de *jump greater* do Assembly.

jge

Esse método implementa a operação de *jump greater or equal* do Assembly.

jl

Esse método implementa a operação de *jump less* do Assembly.

jle

Esse método implementa a operação de *jump less or equal* do Assembly.

leCodigo

Esse método lê o código transformado em binário por um método anterior (compilaCode).

busca

Esse método implementa o ciclo de busca em nosso compilador.

estado

Esse método mostra o estado dos registradores após cada ciclo.

TESTES FEITOS

Ao longo da criação do EP fomos fazendo testes utilizando códigos Assembly para verificar se nossas funções estavam dando o resultado desejado. Abaixo seguem os testes que fizemos:

Teste 1

```
MOV AX,100
MOV BX,CX
ADD AX,10
INC AX
```

Teste 2

```
ADD AX,10
```

Teste 3

```
JE 100
```

Teste 4

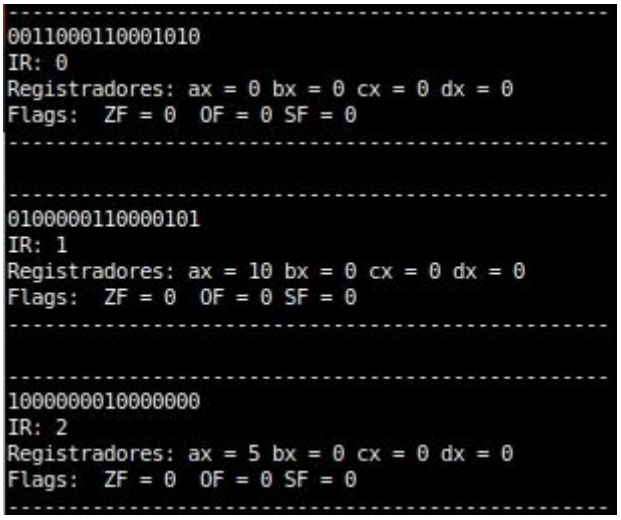
```
ADD AX,10
SUB AX,5
```

Teste 5

```
ADD AX,10
SUB AX,5
```

JE 0

Print do teste 5



MODO DE USO

Para utilizar o código, recomendamos os seguintes passos:

1. Abra o .java no prompt
2. Insira o arquivo *.txt* com as instruções Assembly que será utilizado

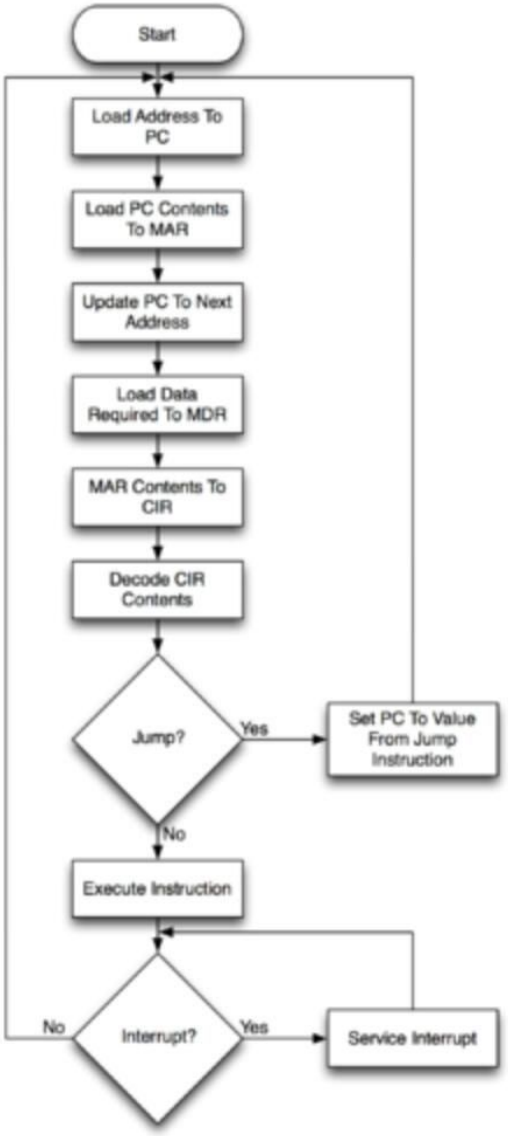
Obs. 1: a classe construtor (M) é importante para que o código funcione, portanto utilize-a juntamente com a classe principal.

Obs. 2: deve-se tomar o cuidado de não inserir números que não resultem em binários maiores do que 7 bits.

PROBLEMAS RESOLVIDOS

Como resolvemos os maiores problemas:

1. Nosso primeiro problema a ser resolvido era como transformar o código que seria dado através de um arquivo assembly em binários que seriam utilizados para acessar a memória desenhada no EP. O problema foi resolvido seguindo a sugestão do professor em sala de aula: transformamos cada um dos comandos assembly em binários utilizando um switch case. Para isso, utilizamos duas funções: uma que transforma os comandos em binário e outra que lê os códigos criados pela primeira função.
2. Tivemos que discutir como alterar os registradores na memória e como diferenças de semântica na escrita poderiam ser tratadas.
3. Para a implementação do ciclo de instrução tivemos que buscar um diagrama (imagem abaixo) que explicasse a forma que ele deveria ser implementado. A interpretação do diagrama foi fundamental para montarmos o código e saber o caminho que o ciclo tomava.



A diagram of the instruction cycle.

Que problema não resolvemos

O único problema que não conseguimos resolver foi a quantidade de bits que o teste pode conter. Em nosso código o int de registradores é de 7 bits, consequentemente não se deve utilizar números grandes nos testes feitos. O tratamento que não demos é para números que contenham mais de 7 bits.

REFERÊNCIAS

http://www.guj.com.br/t/transforma-decimal-em-binario/47061

http://www.ppgia.pucpr.br/~santin/cc/2009/6/