PUCRS - Escola Politécnica

Disciplina: Sistemas Operacionais - 2021/1 - Trabalho Prático

Prof. Fernando Luís Dotti

Integrantes: Matheus Homrich, Nathan Schostack, Pedro Soares Pinto Zart, Thiago Villa

Implementação

Para compilar e rodar a Máquina Virtual basta baixar o arquivo .zip, iniciar o arquivo Sistema.java pelo ambiente de desenvolvimento de sua preferência e abrir um terminal. Com ambos abertos, deve-se localizar o método "test1" no código e alterar a mesma conforme o programa desejado:

```
// ------------------------------------------- teste do sistema ,  veja classe de programas
public void test1(){
    Aux aux = new Aux();
    Word[] p = new Programas().p4BubbleSort;
    aux.carga(p, vm.m);
    vm.cpu.setContext(0);
    System.out.println("---------------------------------- programa carregado ");
    aux.dump(vm.m, 0, 33);
    System.out.println("---------------------------------- após execucao ");
    vm.cpu.run();
    aux.dump(vm.m, 0, 33);
}
```

No exemplo acima, o teste será feito com o programa número 4, denominado "p4BubbleSort". Para testar outro programa, é só alterar o que está circulado em vermelho e compilar e executar novamente a VM.

Programas

Todos os programas foram feitos e estão funcionando.

Programa 2: um programa que lê um valor de uma determinada posição (carregada no início), se o número for menor que zero coloca -1 no início da posição de memória para saída; se for maior que zero este é o número de valores da sequência de Fibonacci a serem escritos em sequência a partir de uma posição de memória.

```
public Word[] p2FibonacciJMP = new Word[] {

        new Word(Opcode.LDI, 0, -1, 8),

        new Word(Opcode.STD,0,-1,25),

        new Word(Opcode.LDD, 1, -1, 25),

        new Word(Opcode.LDI, 7, -1, 8),

        new Word(Opcode.JMPIG, 7, 1, -1),

        new Word(Opcode.LDI, 3, -1, -1),

        new Word(Opcode.STD,3,-1,26),

        new Word(Opcode.STOP, -1, -1, -1), // 7
```

```
        new Word(Opcode.LDI, 0, -1, 0),
        new Word(Opcode.LDI, 1, -1, 1),
        new Word(Opcode.LDI, 2, -1, 27),
        new Word(Opcode.LDD, 3, -1, 25),
        new Word(Opcode.LDI, 7, -1, 7),
        new Word(Opcode.STX, 2, 0, -1),
        new Word(Opcode.ADDI,2,-1,1),
        new Word(Opcode.SUBI,3,-1,1),
        new Word(Opcode.JMPIE, 7, 3, -1),
        new Word(Opcode.STX, 2, 1, -1),
        new Word(Opcode.ADDI,2,-1,1),
        new Word(Opcode.SUBI,3,-1,1),
        new Word(Opcode.JMPIE, 7, 3, -1),
        new Word(Opcode.ADD,0,1,-1),
        new Word(Opcode.ADD,1,0,-1),
        new Word(Opcode.JMP, -1, -1, 13)
    };
```

Programa 3: dado um inteiro em alguma posição de memória, se for negativo armazena -1 na saída; se for positivo responde o fatorial do número na saída.

```
public Word[] p3Fatorial = new Word[] {
        new Word(Opcode.LDI, 0, -1, 5),
              new Word(Opcode.STD,0,-1,30),
        new Word(Opcode.LDD, 1, -1, 30),
        new Word(Opcode.LDI, 7, -1, 11),
        new Word(Opcode.LDI, 6, -1, 18),
        new Word(Opcode.LDI, 5, -1, 21),
              new Word(Opcode.JMPIG, 7, 1, -1),
        new Word(Opcode.JMPIE, 6, 1, -1),
        new Word(Opcode.LDI, 3, -1, -1),
        new Word(Opcode.STD,3,-1,31),
        new Word(Opcode.STOP, -1, -1, -1), // 10
```

```
        new Word(Opcode.LDI, 4, -1, 16),
        new Word(Opcode.LDD, 0, -1, 30),
        new Word(Opcode.LDD, 1, -1, 30),
        new Word(Opcode.SUBI, 1, -1, 1),
        new Word(Opcode.JMPIE, 5, 1, -1), // 15

        new Word(Opcode.MULT,0,1,-1),
        new Word(Opcode.SUBI, 1, -1, 1),
        new Word(Opcode.JMPIG, 4, 1, -1),
        new Word(Opcode.STD,0,-1,31),
        new Word(Opcode.STOP, -1, -1, -1),

        new Word(Opcode.LDI, 0, -1, 1),
        new Word(Opcode.STD,0,-1,31),
        new Word(Opcode.STOP, -1, -1, -1),

        new Word(Opcode.LDI, 0, -1, 1),
        new Word(Opcode.STD,0,-1,31),
        new Word(Opcode.STOP, -1, -1, -1)

    };
```

Programa 4: para um N definido (5 por exemplo) o programa ordena um vetor de N números em alguma posição de memória; ordena usando bubble sort loop ate que não swap nada passando pelos N valores faz swap de vizinhos se da esquerda maior que da direita.

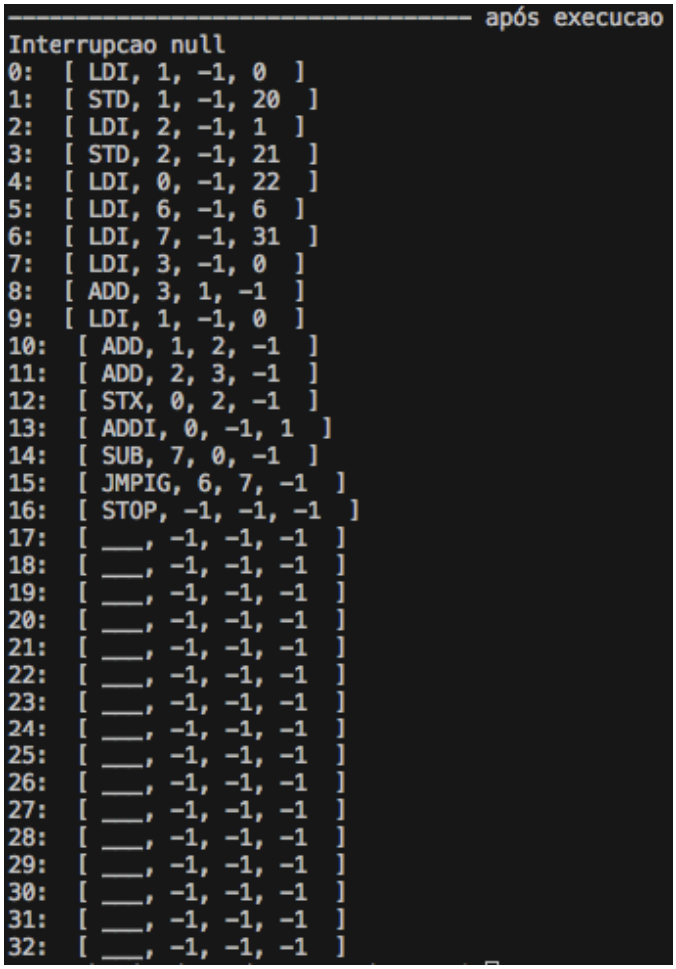
```
public Word[] p4BubbleSort = new Word[] {
        new Word(Opcode.LDI, 0, -1, 12),  //carregando valor na memoria
        new Word(Opcode.STD, 0, -1, 40),
```

```
new Word(Opcode.LDI, 0, -1, 20),
new Word(Opcode.STD, 0, -1, 41),

new Word(Opcode.LDI, 0, -1, 12),
new Word(Opcode.STD, 0, -1, 42),

new Word(Opcode.LDI, 0, -1, 1),
new Word(Opcode.STD, 0, -1, 43),

new Word(Opcode.LDI, 0, -1, 29),
new Word(Opcode.STD, 0, -1, 44),

new Word(Opcode.LDI, 0, -1, -12),
new Word(Opcode.STD, 0, -1, 45),

new Word(Opcode.LDI, 0, -1, 0),
new Word(Opcode.STD, 0, -1, 46),// valores carregados

new Word(Opcode.LDI, 3, -1, 6),
new Word(Opcode.LDI, 4, -1, 6),
new Word(Opcode.LDI, 5, -1, 20),
new Word(Opcode.LDI, 6, -1, 33),
new Word(Opcode.LDI, 7, -1, 38),
new Word(Opcode.LDI, 0, -1, 40),

new Word(Opcode.JMPIE, 6, 3, -1), //inicio loop

new Word(Opcode.SUBI, 3, -1, 1),
```

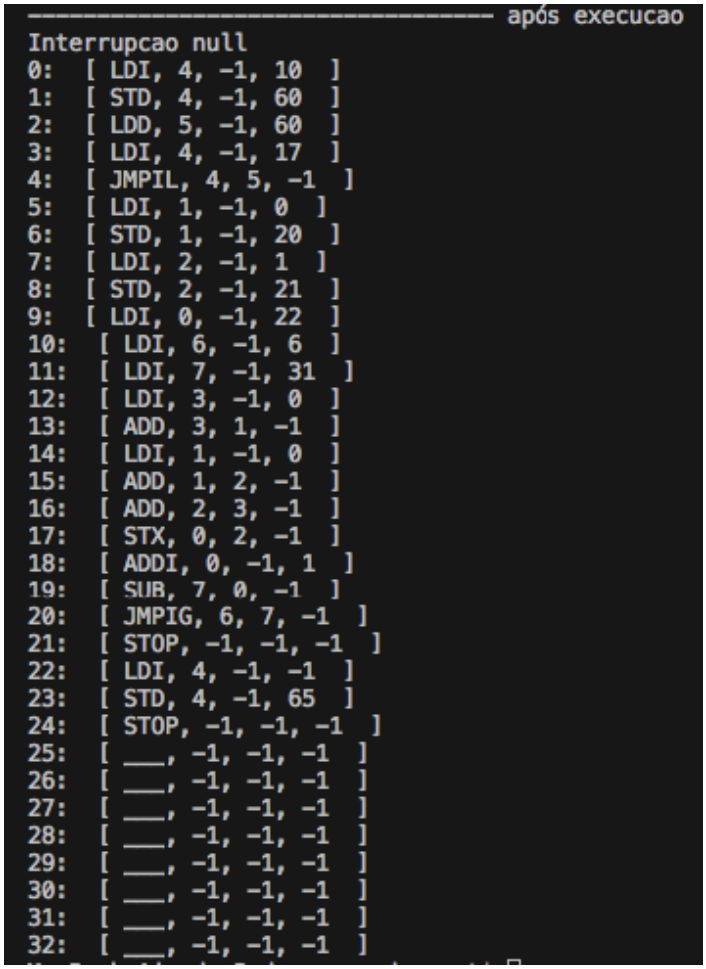
```
new Word(Opcode.LDX, 1, 0, -1),
new Word(Opcode.ADDI, 0, -1, 1),
new Word(Opcode.LDX, 2, 0, -1),
new Word(Opcode.SUB, 2, 1, -1),
new Word(Opcode.JMPIG, 5, 2, -1),

new Word(Opcode.LDX, 2, 0, -1),
new Word(Opcode.STX, 0, 1, -1),
new Word(Opcode.SUBI, 0, -1, 1),
new Word(Opcode.STX, 0, 2, -1),
new Word(Opcode.ADDI, 0, -1, 1),
new Word(Opcode.JMPI, 5, 0 , -1),

new Word(Opcode.JMPIE, 7, 4, -1),
new Word(Opcode.SUBI, 4, -1, 1),
new Word(Opcode.LDI, 0, -1, 40),
new Word(Opcode.LDI, 3, -1, 6),
new Word(Opcode.JMPIG, 5, 0, -1),//fim do loop

new Word(Opcode.STOP, -1, -1, -1)
```

Saídas

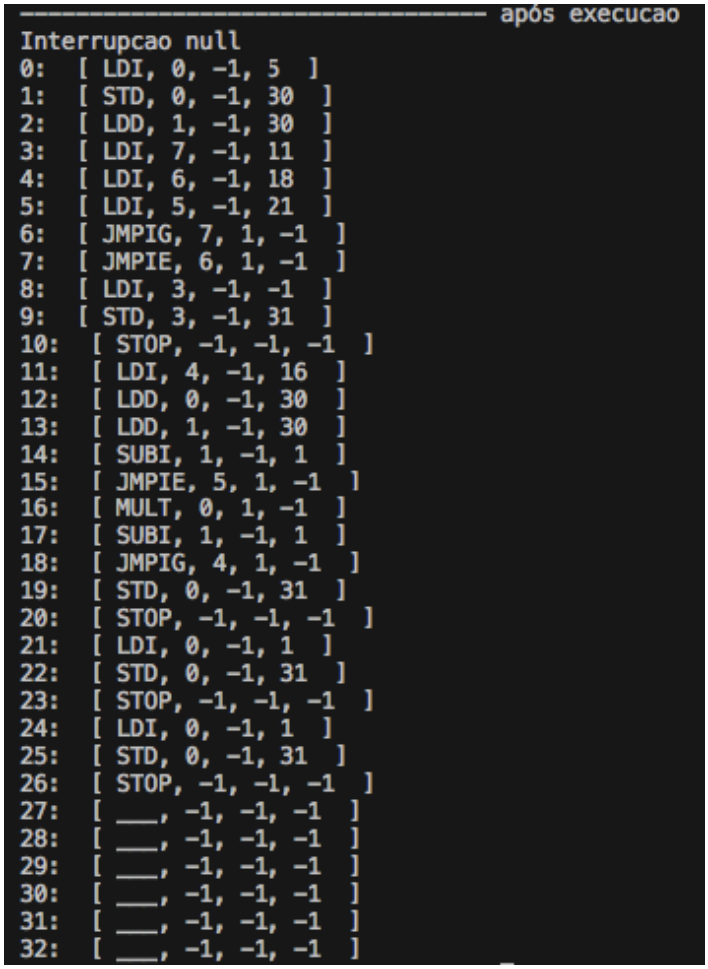
Programa Fibonacci:


---------- após execucao
Interrupcao null
0: [LDI, 1, -1, 0]
1: [STD, 1, -1, 20]
2: [LDI, 2, -1, 1]
3: [STD, 2, -1, 21]
4: [LDI, 0, -1, 22]
5: [LDI, 6, -1, 6]
6: [LDI, 7, -1, 31]
7: [LDI, 3, -1, 0]
8: [ADD, 3, 1, -1]
9: [LDI, 1, -1, 0]
10: [ADD, 1, 2, -1]
11: [ADD, 2, 3, -1]
12: [STX, 0, 2, -1]
13: [ADDI, 0, -1, 1]
14: [SUB, 7, 0, -1]
15: [JMPIG, 6, 7, -1]
16: [STOP, -1, -1, -1]
17: [___, -1, -1, -1]
18: [___, -1, -1, -1]
19: [___, -1, -1, -1]
20: [___, -1, -1, -1]
21: [___, -1, -1, -1]
22: [___, -1, -1, -1]
23: [___, -1, -1, -1]
24: [___, -1, -1, -1]
25: [___, -1, -1, -1]
26: [___, -1, -1, -1]
27: [___, -1, -1, -1]
28: [___, -1, -1, -1]
29: [___, -1, -1, -1]
30: [___, -1, -1, -1]
31: [___, -1, -1, -1]
32: [___, -1, -1, -1]

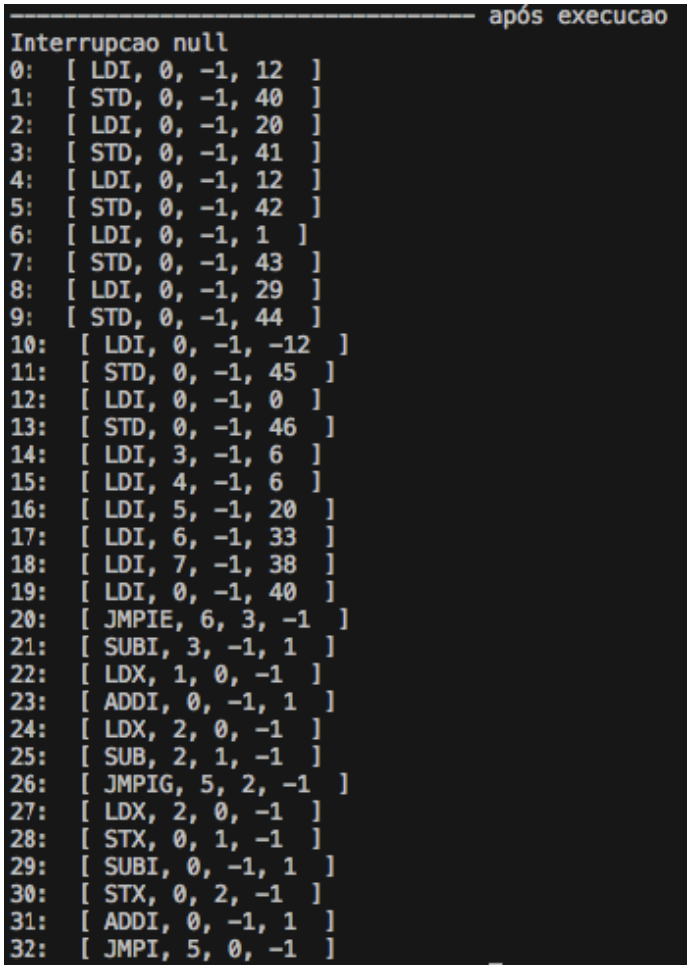
Programa 2:



Programa 3 Fatorial:



Programa 4 Bubble Sort


---------- após execucao
Interrupcao null
0: [LDI, 0, -1, 12]
1: [STD, 0, -1, 40]
2: [LDI, 0, -1, 20]
3: [STD, 0, -1, 41]
4: [LDI, 0, -1, 12]
5: [STD, 0, -1, 42]
6: [LDI, 0, -1, 1]
7: [STD, 0, -1, 43]
8: [LDI, 0, -1, 29]
9: [STD, 0, -1, 44]
10: [LDI, 0, -1, -12]
11: [STD, 0, -1, 45]
12: [LDI, 0, -1, 0]
13: [STD, 0, -1, 46]
14: [LDI, 3, -1, 6]
15: [LDI, 4, -1, 6]
16: [LDI, 5, -1, 20]
17: [LDI, 6, -1, 33]
18: [LDI, 7, -1, 38]
19: [LDI, 0, -1, 40]
20: [JMPIE, 6, 3, -1]
21: [SUBI, 3, -1, 1]
22: [LDX, 1, 0, -1]
23: [ADDI, 0, -1, 1]
24: [LDX, 2, 0, -1]
25: [SUB, 2, 1, -1]
26: [JMPIG, 5, 2, -1]
27: [LDX, 2, 0, -1]
28: [STX, 0, 1, -1]
29: [SUBI, 0, -1, 1]
30: [STX, 0, 2, -1]
31: [ADDI, 0, -1, 1]
32: [JMPI, 5, 0, -1]

};