Sistemas Operacionais

Vitor Hugo Garcez, João Pedro Martins, Alberto Rocha

1. Implementação:

Para compilar e executar o projeto basta instalar um pacote JDK em sua máquina, e por meio de uma IDE (de sua preferência) executar o programa.

2. Programas:

Os programas foram feitos dentro da classe **Programas** e são executados através da classe **Computer** utilizando o método ***main***. Esse método chama métodos separados dentro da própria classe, para que as saídas no terminal tenham informações importantes sobre suas execuções, por exemplo: quais instruções foram alocadas na memória, valores inseridos nelas e a quantidade de espaços na memória que mostraremos de uma vez só.

2.1 Bubble Sort:

Para executa-lo basta remover as barras duplas na linha 17 da classe computer.

```
-------------------------------- programa carregado
0:  [ LDI, 0, -1, 15  ]
1:  [ STD, 0, -1, 60  ]
2:  [ LDI, 0, -1, 7  ]
3:  [ STD, 0, -1, 61  ]
4:  [ LDI, 0, -1, 3  ]
5:  [ STD, 0, -1, 62  ]
6:  [ LDI, 0, -1, 9  ]
7:  [ STD, 0, -1, 63  ]
8:  [ LDI, 0, -1, 1  ]
9:  [ STD, 0, -1, 64  ]
10:  [ LDI, 0, -1, 60  ]
11:  [ STD, 0, -1, 50  ]
12:  [ STD, 0, -1, 51  ]
13:  [ STD, 0, -1, 52  ]
14:  [ LDI, 0, -1, 61  ]
15:  [ STD, 0, -1, 53  ]
16:  [ LDI, 0, -1, 65  ]
17:  [ STD, 0, -1, 54  ]
18:  [ LDD, 0, -1, 51  ]
19:  [ LDD, 1, -1, 52  ]
20:  [ LDD, 2, -1, 53  ]
21:  [ LDX, 3, 1, -1  ]
22:  [ LDX, 4, 2, -1  ]
23:  [ SUB, 3, 4, -1  ]
24:  [ LDI, 4, -1, 31  ]
25:  [ JMPIL, 4, 3, -1  ]
26:  [ LDX, 3, 1, -1  ]
27:  [ LDX, 4, 2, -1  ]
28:  [ SWAP, 3, 4, -1  ]
29:  [ STX, 1, 3, -1  ]
30:  [ STX, 2, 4, -1  ]
31:  [ ADDI, 1, -1, 1  ]
32:  [ ADDI, 2, -1, 1  ]
33:  [ LDD, 3, -1, 54  ]
34:  [ SUB, 3, 2, -1  ]
35:  [ LDI, 4, -1, 21  ]
37:  [ ADDI, 0, -1, 1  ]
38:  [ LDD, 3, -1, 54  ]
39:  [ SUB, 3, 0, -1  ]
40:  [ LDI, 4, -1, 19  ]
41:  [ JMPIG, 4, 3, -1  ]
42:  [ STOP, -1, -1, -1  ]
43:  [ NULL, 0, 0, 0  ]
44:  [ NULL, 0, 0, 0  ]
45:  [ NULL, 0, 0, 0  ]
46:  [ NULL, 0, 0, 0  ]
47:  [ NULL, 0, 0, 0  ]
48:  [ NULL, 0, 0, 0  ]
49:  [ NULL, 0, 0, 0  ]
50:  [ NULL, 0, 0, 0  ]
51:  [ NULL, 0, 0, 0  ]
52:  [ NULL, 0, 0, 0  ]
53:  [ NULL, 0, 0, 0  ]
54:  [ NULL, 0, 0, 0  ]
55:  [ NULL, 0, 0, 0  ]
56:  [ NULL, 0, 0, 0  ]
57:  [ NULL, 0, 0, 0  ]
58:  [ NULL, 0, 0, 0  ]
59:  [ NULL, 0, 0, 0  ]
60:  [ NULL, 0, 0, 0  ]
61:  [ NULL, 0, 0, 0  ]
62:  [ NULL, 0, 0, 0  ]
63:  [ NULL, 0, 0, 0  ]
64:  [ NULL, 0, 0, 0  ]
```

Fig 1: programa carregado

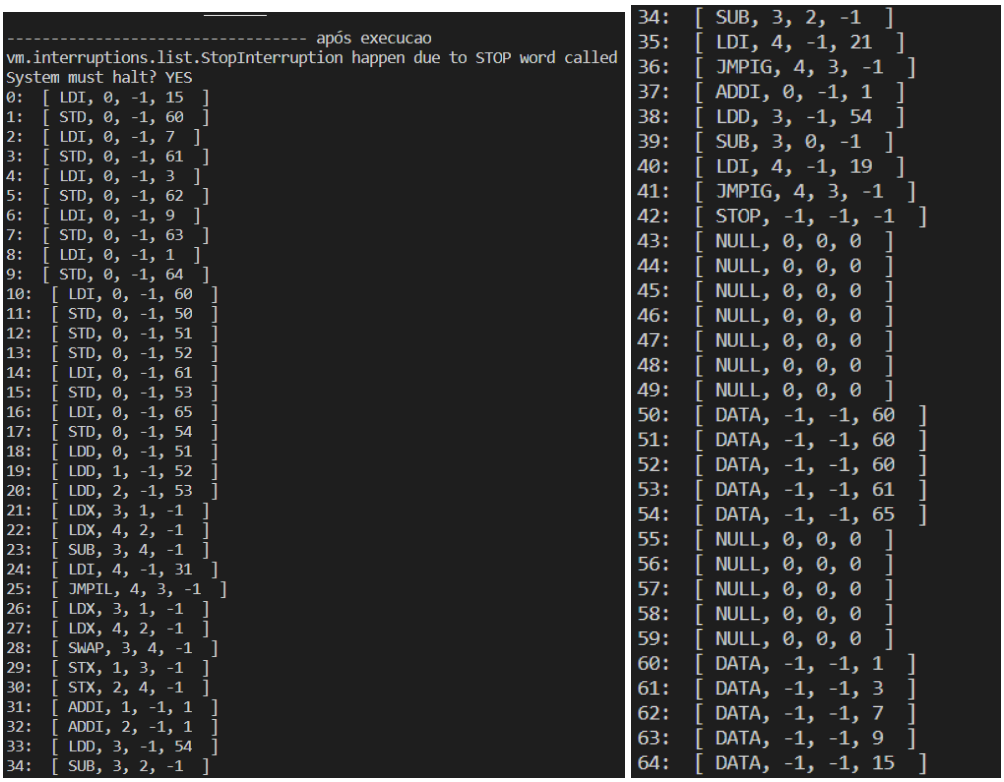

Figura 2: Programa pós execução

2.2 Fatorial

Para executa-lo basta remover as barras duplas na linha 16 da classe computer.

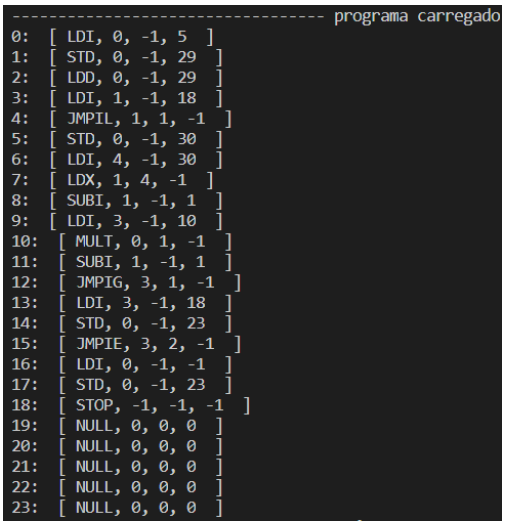

Figura 3: Programa Carregado Fatorial.

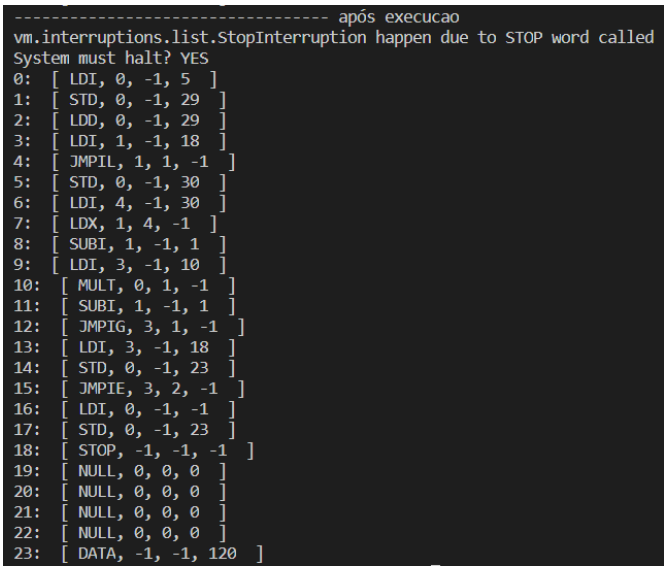

Figura 4: Fatorial Pós execução

2.3 Fibonacci

Para executa-lo basta remover as barras duplas na linha 15 da classe computer.

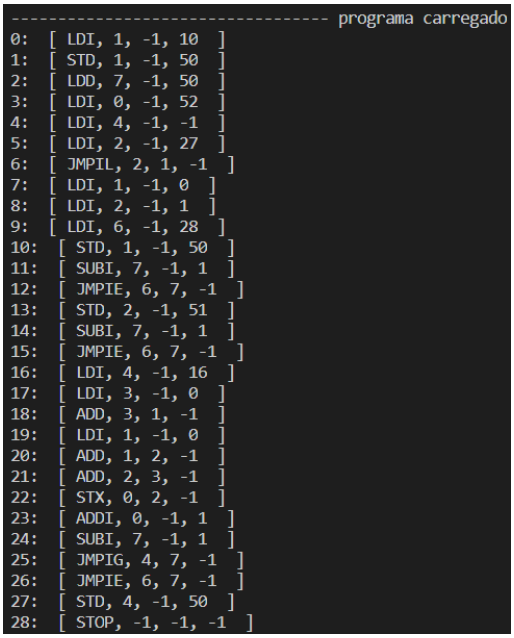

Figura 5: Fibonacci Carregado em memoria

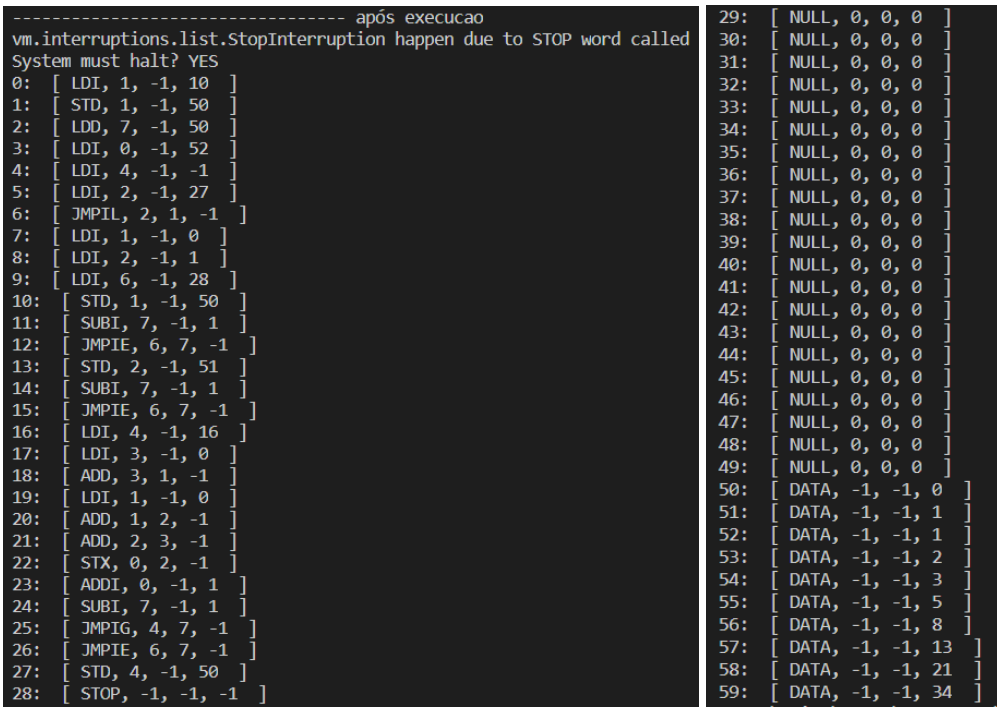

Figura 6: Fibonacci pós execução

2.4 teste de chamada de sistema

Para executa-lo basta remover as barras duplas na linha 18 da classe computer.

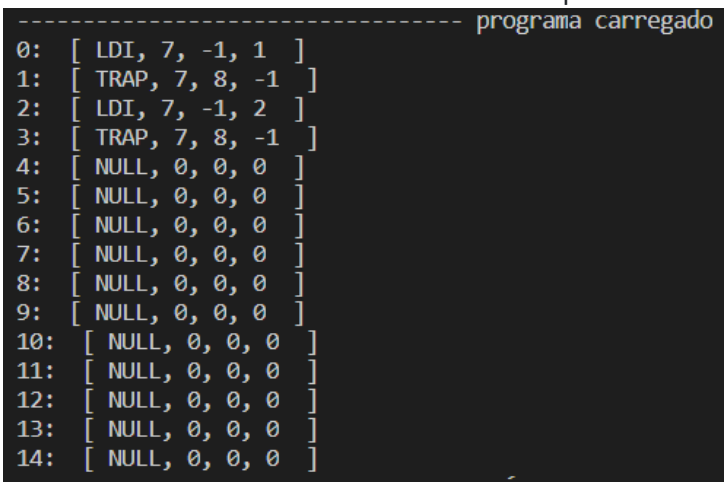

Figura 7: Teste de chamada de sistema carregado em memoria

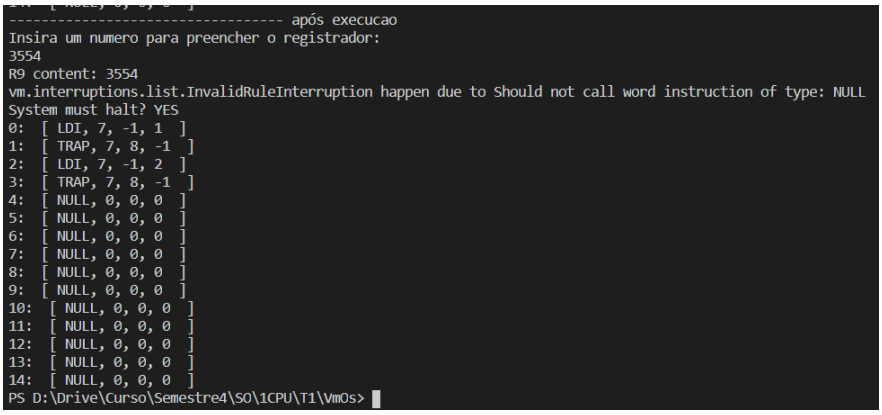

Figura 8: Teste de chamada de sistema pós execução

2.5 teste de interrupção de sistema

Para executa-lo basta remover as barras duplas na linha 19 da classe computer.

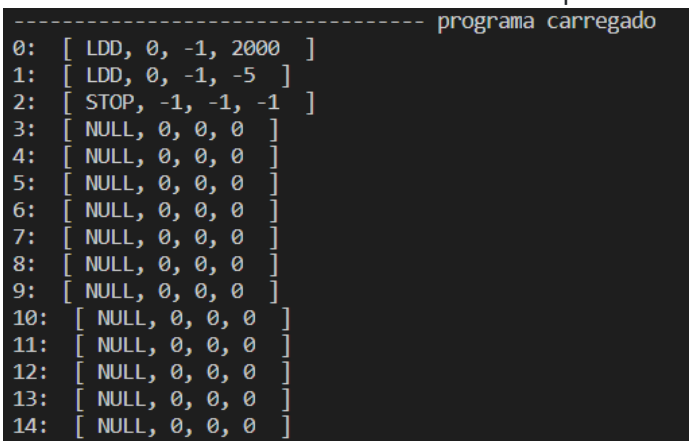

Figura 9: teste de Interrupção de sistema carregado em memoria

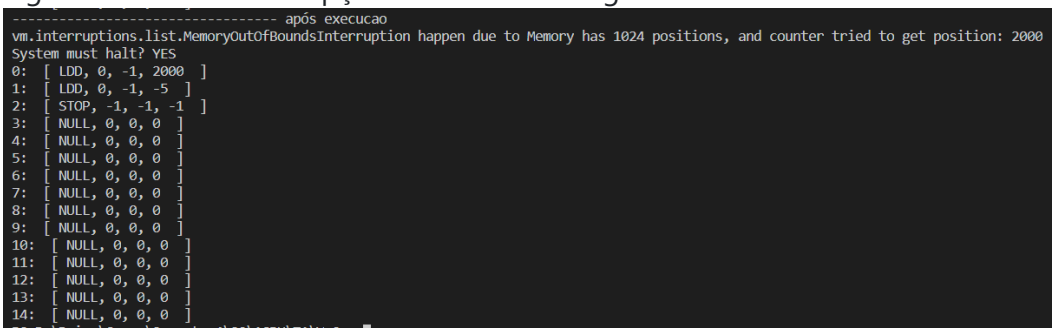

Figura 10: teste de interrupção de sistema pós execução

3. Considerações parciais

O trabalho foi feito utilizando a linguagem de programação Java, para fins de estudos estávamos discutindo o desenvolvimento usando a linguagem GO. Entretanto como as horas de trabalhos estavam puxando muito tempo dos

integrantes do grupo optamos por deixar o código base como estava, apenas remodelando-o para aplicar a engenharia de software dentro da VM.

O trabalho no geral, em certa parte, foi bem divertido. Conseguimos aprender o funcionamento do computador Viking que permite programarmos em Assembly.