


Nicolas dos Santos Moraes

Para rodar os programas devemos ir a função main() e tirar o comentário da função que desejamos executar, lembre-se que roda apenas **uma função de cada vez** ou seja se quisermos rodar a função x as outras funções devem estar comentadas.

```
public static void main(String args[]) {  
    Sistema s = new Sistema();  
    //s.roda(progs.fibonacci10);  
    //s.roda(progs.progMinimo);  
    //s.roda(progs.fatorial);  
  
    //fase1  
    s.roda(progs.PA);  
    //s.roda(progs.PB);  
    //s.roda(progs.PC);  
  
    //fase2 testes interrupção  
    //s.roda(progs.interruptaoEndereco);  
    //s.roda(progs.interruptaoInstrucao);  
    //s.roda(progs.interruptaoOverflow);  
  
    //fase3 teste TRAP  
    //s.roda(progs.testeTRAP_IN);  
    //s.roda(progs.testeTRAP_OUT);  
    //s.roda(progs.fibonacciTRAP);  
    //s.roda(progs.fatorialTRAP);  
}
```



```
public static void main(String args[]) {  
    Sistema s = new Sistema();  
    //s.roda(progs.fibonacci10);  
    //s.roda(progs.progMinimo);  
    //s.roda(progs.fatorial);  
  
    //fase1  
    s.roda(progs.PA);  
    s.roda(progs.PB);  
    //s.roda(progs.PC);  
  
    //fase2 testes interrupção  
    //s.roda(progs.interruptaoEndereco);  
    //s.roda(progs.interruptaoInstrucao);  
    //s.roda(progs.interruptaoOverflow);  
  
    //fase3 teste TRAP  
    //s.roda(progs.testeTRAP_IN);  
    //s.roda(progs.testeTRAP_OUT);  
    //s.roda(progs.fibonacciTRAP);  
    //s.roda(progs.fatorialTRAP);  
}
```



Os programas feitos foram:

PA – Realiza a sequência de Fibonacci com o tamanho dado na primeira instrução do código, se o tamanho dado for negativo enviará -1 para a primeira posição da saída (POS 38).

PB – Calcula o fatorial do número dado na primeira instrução e dá o resultado na saída (POS 15), caso o número for negativo retorna -1.

PC – Faz o bubble-sort de 5 números que são armazenados na memória (4-3-5-1-2) e retorna suas posições organizados na memória (1-2-3-4-5).

interruptaoEndereco – Tenta acessar um endereço maior que o tamanho da memória (1025) para testar a interrupção de endereço.

interruptaoInstrucao – Tenta acessar um registro que não existe para testar a interrupção de instrução.

interruptaoOverflow – Tenta fazer o fatorial de 25, mas pelo resultado passar de 1 bilhão cai no número máximo e cai na interrupção de overflow.

testeTRAP_IN – Código simples que testa o uso de TRAP com o IN.

testeTRAP_OUT – Código simples que testa o uso de TRAP com o OUT.

fibonacciTRAP – É igual ao PA mas com uma TRAP no início do código que realiza a **leitura** para declarar qual será o tamanho da sequência.

fatorialTRAP – É igual ao PB mas com uma TRAP perto do fim do código que realiza a **escrita** do resultado do fatorial.

As telas de saídas esperada são:

Para PA:

```
33
r0: 52    r1: 144    r2: 233    r3: 89    r4: 33    r5: 52    r6: 22    r7: 0
[ STOP, -1, -1, -1 ]
Final de programa
```

Todos os programas que chegarem a instrução de STOP vão escrever a mensagem de final de programa.

```
26: [ LDI, 1, -1, 0 ]
27: [ ADD, 1, 2, -1 ]
28: [ ADD, 2, 3, -1 ]
29: [ STX, 0, 2, -1 ]
30: [ ADDI, 0, -1, 1 ]
31: [ SUB, 7, 0, -1 ]
32: [ JMPIG, 6, 7, -1 ]
33: [ STOP, -1, -1, -1 ]
34: [ DATA, -1, -1, -1 ]
35: [ DATA, -1, -1, -1 ]
36: [ DATA, -1, -1, -1 ]
37: [ DATA, -1, -1, -1 ]
38: [ DATA, -1, -1, 0 ]
39: [ DATA, -1, -1, 1 ]
40: [ DATA, -1, -1, 1 ]
41: [ DATA, -1, -1, 2 ]
42: [ DATA, -1, -1, 3 ]
43: [ DATA, -1, -1, 5 ]
44: [ DATA, -1, -1, 8 ]
45: [ DATA, -1, -1, 13 ]
46: [ DATA, -1, -1, 21 ]
47: [ DATA, -1, -1, 34 ]
48: [ DATA, -1, -1, 55 ]
49: [ DATA, -1, -1, 89 ]
50: [ DATA, -1, -1, 144 ]
51: [ DATA, -1, -1, 233 ]
52: [ DATA, -1, -1, -1 ]
```

A saída inicia na posição 38 e termina neste caso na posição 51 pois o tamanho dado para essa sequência é 14. Não foi colocada toda a saída pois era muito extenso.

PB:

```
0: [ LDI, 0, -1, 7 ]
1: [ STD, 0, -1, 50 ]
2: [ LDD, 0, -1, 50 ]
3: [ LDI, 1, -1, -1 ]
4: [ LDI, 2, -1, 13 ]
5: [ JMPIL, 2, 0, -1 ]
6: [ LDI, 1, -1, 1 ]
7: [ LDI, 6, -1, 1 ]
8: [ LDI, 7, -1, 13 ]
9: [ JMPIE, 7, 0, 0 ]
10: [ MULT, 1, 0, -1 ]
11: [ SUB, 0, 6, -1 ]
12: [ JMP, -1, -1, 9 ]
13: [ STD, 1, -1, 15 ]
14: [ STOP, -1, -1, -1 ]
15: [ DATA, -1, -1, 5040 ]
```

Realiza o fatorial de 7 e da o resultado na posição 15.

PC:

```
30: [ SUB, 2, 1, -1 ]
31: [ ADDI, 4, -1, 1 ]
32: [ SUBI, 6, -1, 1 ]
33: [ JMPILM, -1, 2, 99 ]
34: [ STX, 5, 1, -1 ]
35: [ SUBI, 4, -1, 1 ]
36: [ STX, 4, 0, -1 ]
37: [ ADDI, 4, -1, 1 ]
38: [ JMPIGM, -1, 6, 99 ]
39: [ ADDI, 5, -1, 1 ]
40: [ SUBI, 7, -1, 1 ]
41: [ LDI, 4, -1, 0 ]
42: [ ADD, 4, 5, -1 ]
43: [ ADDI, 4, -1, 1 ]
44: [ JMPIGM, -1, 7, 98 ]
45: [ STOP, -1, -1, -1 ]
46: [ DATA, -1, -1, 1 ]
47: [ DATA, -1, -1, 2 ]
48: [ DATA, -1, -1, 3 ]
49: [ DATA, -1, -1, 4 ]
50: [ DATA, -1, -1, 5 ]
51: [ DATA, -1, -1, -1 ]
52: [ DATA, -1, -1, -1 ]
53: [ DATA, -1, -1, -1 ]
```

Em PC vemos a ordem organizada de forma crescente na saída a partir da posição 46.

interrupcaoEndereco:

```
1
r0: 999    r1: 0    r2: 0    r3: 0    r4: 0    r5: 0    r6: 0    r7: 0
[ STD, 0, -1, 1025 ]
Endereco invalido
----- após execucao
0: [ LDI, 0, -1, 999 ]
1: [ STD, 0, -1, 1025 ]
2: [ STOP, -1, -1, -1 ]
```

Por ter tentado acessar uma posição de memória maior que o permitido retornou à interrupção.

interrupcaoIntrucao:

```
r0: 0    r1: 0    r2: 0    r3: 0    r4: 0    r5: 0    r6: 0    r7: 0
[ LDI, 10, -1, 999 ]
Instrucao invalida
----- após execucao
0: [ LDI, 10, -1, 999 ]
1: [ STOP, -1, -1, -1 ]
```

Por ter tentado acessar registrador maior que o permitido retornou à interrupção.

interrupcaoOverflow:

```
10
r0: 19    r1: 127512000    r2: 13    r3: 0    r4: 0    r5: 0    r6: 1    r7: 13
[ MULT, 1, 0, -1 ]
Overflow
```

Por ter passado de 1 bilhão ele retorna o overflow.

testeTRAP_IN:

```

0      r0: 0      r1: 0      r2: 0      r3: 0      r4: 0      r5: 0      r6: 0      r7: 0
      [ LDI, 8, -1, 1 ]
1      r0: 0      r1: 0      r2: 0      r3: 0      r4: 0      r5: 0      r6: 0      r7: 0
      [ LDI, 9, -1, 4 ]
2      r0: 0      r1: 0      r2: 0      r3: 0      r4: 0      r5: 0      r6: 0      r7: 0
      [ TRAP, -1, -1, -1 ]
5
3      r0: 0      r1: 0      r2: 0      r3: 0      r4: 0      r5: 0      r6: 0      r7: 0
      [ STOP, -1, -1, -1 ]
Final de programa
----- após execucao
0: [ LDI, 8, -1, 1 ]
1: [ LDI, 9, -1, 4 ]
2: [ TRAP, -1, -1, -1 ]
3: [ STOP, -1, -1, -1 ]
4: [ DATA, -1, -1, 5 ]
```

testeTRAP_OUT:

```

4      r0: 999      r1: 0      r2: 0      r3: 0      r4: 0      r5: 0      r6: 0      r7: 0
      [ TRAP, -1, -1, -1 ]
out: 999
5      r0: 999      r1: 0      r2: 0      r3: 0      r4: 0      r5: 0      r6: 0      r7: 0
      [ STOP, -1, -1, -1 ]
Final de programa
----- após execucao
0: [ LDI, 0, -1, 999 ]
1: [ STD, 0, -1, 10 ]
2: [ LDI, 8, -1, 2 ]
3: [ LDI, 9, -1, 10 ]
4: [ TRAP, -1, -1, -1 ]
5: [ STOP, -1, -1, -1 ]
6: [ DATA, -1, -1, -1 ]
```

fibonacciTRAP:

```
0
    r0: 0      r1: 0      r2: 0      r3: 0      r4: 0      r5: 0      r6: 0      r7: 0
    [ LDI, 8, -1, 1 ]
1
    r0: 0      r1: 0      r2: 0      r3: 0      r4: 0      r5: 0      r6: 0      r7: 0
    [ LDI, 9, -1, 100 ]
2
    r0: 0      r1: 0      r2: 0      r3: 0      r4: 0      r5: 0      r6: 0      r7: 0
    [ TRAP, -1, -1, -1 ]
10
```

```
35: [ JMPIG, 6, 7, -1 ]
36: [ STOP, -1, -1, -1 ]
37: [ DATA, -1, -1, -1 ]
38: [ DATA, -1, -1, -1 ]
39: [ DATA, -1, -1, -1 ]
40: [ DATA, -1, -1, -1 ]
41: [ DATA, -1, -1, 0 ]
42: [ DATA, -1, -1, 1 ]
43: [ DATA, -1, -1, 1 ]
44: [ DATA, -1, -1, 2 ]
45: [ DATA, -1, -1, 3 ]
46: [ DATA, -1, -1, 5 ]
47: [ DATA, -1, -1, 8 ]
48: [ DATA, -1, -1, 13 ]
49: [ DATA, -1, -1, 21 ]
50: [ DATA, -1, -1, 34 ]
51: [ DATA, -1, -1, -1 ]
52: [ DATA, -1, -1, -1 ]
53: [ DATA, -1, -1, -1 ]
54: [ DATA, -1, -1, -1 ]
55: [ DATA, -1, -1, -1 ]
```

fatorialTRAP:

```
16
    r0: 0      r1: 5040      r2: 13      r3: 0      r4: 0      r5: 0      r6: 1      r7: 13
    [ TRAP, -1, -1, -1 ]
out: 5040
17
    r0: 0      r1: 5040      r2: 13      r3: 0      r4: 0      r5: 0      r6: 1      r7: 13
    [ STOP, -1, -1, -1 ]
Final de programa
```