

Alunos: João Brentano, João Granzinoli, Eduardo Soares, William de Lima

Implementação

Para o desenvolvimento de nossa máquina virtual, o grupo decidiu que seria mais vantajoso continuar com a implementação provida em Java previamente pelo professor, logo seu funcionamento continua muito próximo a tal e com isso, para testar as novas funcionalidades disponibilizamos alguns programas que serão demonstrados em Usabilidade.

Esta nova versão da máquina virtual conta com todas as funcionalidades requisitadas pelo professor, vide arquivos PDF disponibilizados no Moodle da disciplina.

Usabilidade

Para conseguir acessar e testar os programas disponíveis, deve-se abrir o arquivo Sistema.java e encontrar a função "main". Neste método pode-se perceber que todos os programas menos um estão comentados; para testar cada programa comente o atual e descomente o que deseja testar.

```
public static void main(String args[]) {  
    // PROGRAMA NORMAL FIBONACCI  
  
    // Sistema s = new Sistema();  
    // s.test1();  
  
    // PROGRAMA INTERRUPÇÃO ENDEREÇO INVÁLIDO  
  
    // Sistema s = new Sistema();  
    // s.errorMemory();  
  
    // PROGRAMA TESTE CHAMADA DE SISTEMA INPUT  
  
    // Sistema s = new Sistema();  
    // s.programTestTrapInput();  
  
    // PROGRAMA TESTE CHAMADA DE SISTEMA OUTPUT  
  
    // Sistema s = new Sistema();  
    // s.programTestTrapOutput();  
  
    // PROGRAMA INTERRUPÇÃO INSTRUÇÃO INVÁLIDA  
  
    // Sistema s = new Sistema();  
    // s.programTestInvalidInstruction();  
}
```

```

// PROGRAMA TESTE OVERFLOW

Sistema s = new Sistema();
s.programTestOverflow();
}

```

Para executar o teste, compile e execute o arquivo.

Para compilar use "javac Sistema.java"
 Para executar use "java Sistema"

Saídas

Em seguida será possível visualizar a saída de cada programa após a execução dentro da máquina virtual.

PROGRAMA TESTE OVERFLOW

```

----- programa carregado
0: [ LDI, 0, -1, 2147483647 ]
1: [ LDI, 1, -1, 1236 ]
2: [ ADD, 0, 1, -1 ]
3: [ STD, 0, -1, 8 ]
4: [ STD, 1, -1, 9 ]
5: [ STOP, 1, -1, 0 ]
6: [ DATA, 50, -1, 1 ]
7: [ ___, -1, -1, -1 ]
8: [ ___, -1, -1, -1 ]
9: [ ___, -1, -1, -1 ]
----- após execucao
A interruption has just happened! interruptOverflow
0: [ LDI, 0, -1, 2147483647 ]
1: [ LDI, 1, -1, 1236 ]
2: [ ADD, 0, 1, -1 ]
3: [ STD, 0, -1, 8 ]
4: [ STD, 1, -1, 9 ]
5: [ STOP, 1, -1, 0 ]
6: [ DATA, 50, -1, 1 ]
7: [ ___, -1, -1, -1 ]
8: [ ___, -1, -1, -1 ]
9: [ ___, -1, -1, -1 ]

```

PROGRAMA INTERRUPÇÃO INSTRUÇÃO INVÁLIDA

```
----- programa carregado
0:  [ ___, 8, -1, 1  ]
1:  [ LDI, 9, -1, 50 ]
2:  [ TRAP, -1, -1, -1 ]
3:  [ STOP, 1, -1, 0  ]
4:  [ DATA, 50, -1, 1 ]
5:  [ ___, -1, -1, -1 ]
6:  [ ___, -1, -1, -1 ]
7:  [ ___, -1, -1, -1 ]
8:  [ ___, -1, -1, -1 ]
9:  [ ___, -1, -1, -1 ]

----- após execucao
A interruption has just happened! interruptInvalidInstruction
0:  [ ___, 8, -1, 1  ]
1:  [ LDI, 9, -1, 50 ]
2:  [ TRAP, -1, -1, -1 ]
3:  [ STOP, 1, -1, 0  ]
4:  [ DATA, 50, -1, 1 ]
5:  [ ___, -1, -1, -1 ]
6:  [ ___, -1, -1, -1 ]
7:  [ ___, -1, -1, -1 ]
8:  [ ___, -1, -1, -1 ]
9:  [ ___, -1, -1, -1 ]
```

PROGRAMA TESTE CHAMADA DE SISTEMA OUTPUT

----- programa carregado

```
0: [ LDI, 8, -1, 2 ]
1: [ LDI, 9, -1, 8 ]
2: [ STD, 9, -1, 8 ]
3: [ TRAP, -1, -1, -1 ]
4: [ STOP, 1, -1, 0 ]
5: [ ___, -1, -1, -1 ]
6: [ ___, -1, -1, -1 ]
7: [ ___, -1, -1, -1 ]
8: [ ___, -1, -1, -1 ]
9: [ ___, -1, -1, -1 ]
```

----- após execucao

A system call has just happened! 2|8

Output:

```
[ DATA, -1, -1, 8 ]
```

A interruption has just happened! interruptStop

```
0: [ LDI, 8, -1, 2 ]
1: [ LDI, 9, -1, 8 ]
2: [ STD, 9, -1, 8 ]
3: [ TRAP, -1, -1, -1 ]
4: [ STOP, 1, -1, 0 ]
5: [ ___, -1, -1, -1 ]
6: [ ___, -1, -1, -1 ]
7: [ ___, -1, -1, -1 ]
8: [ DATA, -1, -1, 8 ]
9: [ ___, -1, -1, -1 ]
```

PROGRAMA CHAMADA DE SISTEMA INPUT

```
----- programa carregado
0: [ LDI, 8, -1, 1 ]
1: [ LDI, 9, -1, 8 ]
2: [ TRAP, -1, -1, -1 ]
3: [ STOP, 1, -1, 0 ]
4: [ ___, -1, -1, -1 ]
5: [ ___, -1, -1, -1 ]
6: [ ___, -1, -1, -1 ]
7: [ ___, -1, -1, -1 ]
8: [ ___, -1, -1, -1 ]
9: [ ___, -1, -1, -1 ]

----- após execucao
A system call has just happened! 1|8
Please input an integer:
2
Value stored in the position 8
Stored value:
[ DATA, -1, -1, 2 ]
A interruption has just happened! interruptStop
0: [ LDI, 8, -1, 1 ]
1: [ LDI, 9, -1, 8 ]
2: [ TRAP, -1, -1, -1 ]
3: [ STOP, 1, -1, 0 ]
4: [ ___, -1, -1, -1 ]
5: [ ___, -1, -1, -1 ]
6: [ ___, -1, -1, -1 ]
7: [ ___, -1, -1, -1 ]
8: [ DATA, -1, -1, 2 ]
9: [ ___, -1, -1, -1 ]
```

PROGRAMA INTERRUPÇÃO ENDEREÇO INVÁLIDO

```
4:  [ STD, 7, -1, 60  ]
5:  [ STOP, 1, -1, 0  ]
6:  [ LDI, 2, -1, 0  ]
7:  [ ADD, 2, 1, -1  ]
8:  [ LDI, 6, -1, 1  ]
9:  [ SUB, 1, 6, -1  ]
10: [ LDI, 7, -1, 8  ]
11: [ JMPIG, 7, 1, -1  ]
12: [ STD, 0, -1, 50  ]
13: [ STD, 1, -1, 51  ]
14: [ STD, 2, -1, 52  ]
15: [ STD, 3, -1, 53  ]
16: [ STD, 4, -1, 54  ]
17: [ STD, 5, -1, 55  ]
18: [ STD, 6, -1, 56  ]
19: [ STD, 7, -1, 57  ]
20: [ LDI, 1, -1, 59  ]
21: [ STD, 1, -1, 1024  ]
22: [ STOP, 1, -1, 0  ]
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  ]
A interruption has just happened! interruptInvalidAddress
```

PROGRAMA NORMAL FIBONACCI

```
----- programa carregado
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 ]
----- após execucao
A interruption has just happened! interruptStop
```

----- após execucao

A interruption has just happened! interruptStop

```
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: [ DATA, -1, -1, 0 ]
21: [ DATA, -1, -1, 1 ]
22: [ DATA, -1, -1, 1 ]
23: [ DATA, -1, -1, 2 ]
24: [ DATA, -1, -1, 3 ]
25: [ DATA, -1, -1, 5 ]
26: [ DATA, -1, -1, 8 ]
27: [ DATA, -1, -1, 13 ]
28: [ DATA, -1, -1, 21 ]
29: [ DATA, -1, -1, 34 ]
30: [ DATA, -1, -1, 55 ]
31: [ ___, -1, -1, -1 ]
32: [ ___, -1, -1, -1 ]
```