28. Desenvolver um programa monolítico, utilizando instruções rotuladas, sobre a máquina 2_REG, que implemente a função `B = (A * 2) + (A % 3)`. Apresentar a computação e a função computada para as entradas `3` e `4`.

```
R1: Se a_zero então vá_para Rx senão vá_para R2;
R2: Faça subtrair_a vá_para R3;
R3: Faça adicionar_b vá_para R4;
R4: Faça adicionar_b vá_para R5;
R5: Faça adicionar_b vá_para R6;
R6: Se a_zero então vá_para Rx senão vá_para R7;
R7: Faça subtrair_a vá_para R8;
R8: Faça adicionar_b vá_para R9;
R9: Faça adicionar_b vá_para R10;
R10: Faça adicionar_b vá_para R11;
R11: Se a_zero então vá_para Rx senão vá_para R12;
R12: Faça subtrair_a vá_para R1;
```

```
(R1, (3, 0))
(R2, (3, 0))
(R3, (2, 0))
(R4, (2, 1))
(R5, (2, 2))
(R6, (2, 3))
(R7, (2, 3))
(R8, (1, 3))
(R9, (1, 4))
(R10, (1, 5))
(R11, (1, 6))
(R12, (1, 6))
(R1, (0, 6))
(RX, (0, 6))
```

```
<PROG, 2_REG> : 3 -> 6
```

```
(R1, (4, 0))
(R2, (4, 0))
(R3, (3, 0))
(R4, (3, 1))
(R5, (3, 2))
(R6, (3, 3))
(R7, (3, 3))
(R8, (2, 3))
(R9, (2, 4))
(R10, (2, 5))
(R11, (2, 6))
(R12, (2, 6))
(R1, (1, 6))
(R2, (1, 6))
(R3, (0, 6))
(R4, (0, 7))
(R5, (0, 8))
(R6, (0, 9))
(RX, (0, 9))
```

```
<PROG, 2_REG> : 4 -> 9
```