Estilo, Jan Coleen S. 2020-05789 ST-7L
Exercise 12

1) to determine if it works in all cases given its implementation constraints (e.g., hardcoded amount of execution time for an instruction type) --obviously, you'd have to set it up and run it,

Next, I tested the limitations that I have seen in the program. I tried to test if the program will work if the vector of the ReservationStatus does not match with the Reservation Status Numbers.

```
// NUMBER OF RESERVATION STATIONS
const int Num_ADD_RS = 1; // it should be 4
const int Num_MULT_RS = 2;
const int Num_DIV_RS = 3;
```

```
ReservationStation
        ADD1(AddOp, OperandInit),
        ADD2(AddOp, OperandInit),
        ADD3(AddOp, OperandInit),
        ADD4(AddOp, OperandInit);
ReservationStation
        MULT1(MultOp, OperandInit),
        MULT2(MultOp, OperandInit);
ReservationStation
        DIV1(DivOp, OperandInit),
        DIV2(DivOp, OperandInit),
        DIV3(DivOp, OperandInit);
// Pack reservation stations into vector
vector<ReservationStation> ResStation = {ADD1,
                                         ADD2,
                                         ADD3,
                                         ADD4,
                                         MULT1,
                                         MULT2,
                                         DIV1,
                                         DIV2,
                                         DIV3};
```

What happens is that even though we initialized 6 amount of reservation stations, the program still creates 9 reservation stations. Here is a screenshot of our initial table.

```
INITIAL VALUES:
Instruction #: 0  Operation: 0  1 <- 2 op 3
Instruction #: 1  Operation: 0  1 <- 1 op 5
Instruction #: 2  Operation: 1  6 <- 9 op 8
Instruction #: 3  Operation: 2  9 <- 4 op 10
Instruction #: 4  Operation: 3  11 <- 12 op 6
Instruction #: 5  Operation: 2  8 <- 1 op 5
Instruction #: 6  Operation: 2  7 <- 2 op 3
RS #: 0  Busy: 0  op: 0  Vj: 0  Vk: 0  Qj: 1002  Qk: 1002
RS #: 1  Busy: 0  op: 0  Vj: 0  Vk: 0  Qj: 1002  Qk: 1002
RS #: 2  Busy: 0  op: 0  Vj: 0  Vk: 0  Qj: 1002  Qk: 1002
RS #: 3  Busy: 0  op: 0  Vj: 0  Vk: 0  Qj: 1002  Qk: 1002
RS #: 4  Busy: 0  op: 2  Vj: 0  Vk: 0  Qj: 1002  Qk: 1002
RS #: 5  Busy: 0  op: 2  Vj: 0  Vk: 0  Qj: 1002  Qk: 1002
RS #: 6  Busy: 0  op: 3  Vj: 0  Vk: 0  Qj: 1002  Qk: 1002
RS #: 7  Busy: 0  op: 3  Vj: 0  Vk: 0  Qj: 1002  Qk: 1002
RS #: 8  Busy: 0  op: 3  Vj: 0  Vk: 0  Qj: 1002  Qk: 1002
Register Content:
5000 1 2 3 4 5 6 7 8 9 10 11 12
Register Status:
1000 1000 1000 1000 1000 1000 1000 1000 1000 1000 1000 1000 1000
```

However, when the functional loop starts, the other reservation statuses for add are not being used. As seen in Clock Cycle 6 and 7, Instruction 1 was only issued after the writeback of Instruction 0.

```
Register Content:
5000 5 2 3 4 5 6 7 8 9 10 11 12
Inst      Issue     Execute   WB        SystemClock
                                                6

0         1         2-5         6
1         0         0-0         0
2         0         0-0         0
3         0         0-0         0
4         0         0-0         0
5         0         0-0         0
6         0         0-0         0


Register Content:
5000 5 2 3 4 5 6 7 8 9 10 11 12
Inst      Issue     Execute   WB        SystemClock
                                                7

0         1         2-5         6
1         7         0-0         0
2         0         0-0         0
3         0         0-0         0
4         0         0-0         0
5         0         0-0         0
6         0         0-0         0
```

Here is the list of instructions that were used.

```
Instruction
        //(rd,rs,rt,opcode)
        I0(1,2,3,AddOp),
        I1(1,1,5,AddOp), //
        I2(6,9,8,SubOp),
        I3(9,4,10,MultOp),
        I4(11,12,6,DivOp),
        I5(8,1,5,MultOp),
        I6(7,2,3,MultOp);
```

2) if the behavior is the one we expected based on the discussions regarding Tomasulo's algorithm.

I first tried the data hazards such as RAW, WAW, and WAR to test if the program runs in all test cases. All of these data dependencies are working correctly in the program.

```
Instruction
        //(rd,rs,rt,opcode)
        I0(1,2,3,AddOp),
        I1(1,1,5,AddOp), // WAW and RAW
        I2(6,9,8,SubOp),
        I3(9,4,10,MultOp), // WAR
        I4(11,12,6,DivOp),
        I5(8,1,5,MultOp),
        I6(7,2,3,MultOp);
```

```
Register Content:
5000 1 2 3 4 5 1 7 8 9 10 11 12
Inst      Issue     Execute   WB        SystemClock
                                                10

0         1         2-5       6
1         2         7-10      0
2         3         4-7       8
3         4         5-0       0
4         5         9-0       0
5         6         0-0       0
6         0         0-0       0
```

```
Register Content:
5000 10 2 3 4 5 1 7 8 40 10 11 12
Inst      Issue     Execute   WB        SystemClock
                                                20


0         1         2-5         6
1         2         7-10        11
2         3         4-7         8
3         4         5-16        17
4         5         9-0         0
5         6         12-0         0
6         18        19-0         0
```

```
Register Content:
5000 10 2 3 4 5 1 7 50 40 10 11 12
Inst      Issue     Execute   WB        SystemClock
                                                30


0         1         2-5         6
1         2         7-10        11
2         3         4-7         8
3         4         5-16        17
4         5         9-0         0
5         6         12-23        24
6         18        19-30        0
```

```
Register Content:
5000 10 2 3 4 5 1 6 50 40 10 11 12
Inst      Issue     Execute   WB        SystemClock
                                                40


0         1         2-5         6
1         2         7-10        11
2         3         4-7         8
3         4         5-16        17
4         5         9-0         0
5         6         12-23        24
6         18        19-30        31
```

```
Register Content:
5000 10 2 3 4 5 1 6 50 40 10 12 12
Inst      Issue     Execute   WB        SystemClock
                                                47


0         1         2-5         6
1         2         7-10        11
2         3         4-7         8
3         4         5-16        17
4         5         9-46        47
5         6         12-23        24
6         18        19-30        31
```