Regarding performance:

- We can have a variable representing all the important ball info:

b_{15} b_{14} b_{13} b_{12} ... b_4 b_3 b_2 b_1 b_0

b_0 - can be used to represent information in the x-axis (b_0 = Left = 0, b_0 = Right = 1)

b_1 - can be used to represent information in the y-axis (b_1 = Down = 0, b_1 = Up = 1)

Then we can define masks for comparison:

Direction ↑ = 00...0**0** = 00...00 = 0
Direction ↗ = 00...0**1** = 00...01 = 1
Direction ↙ = 00...**1**0 = 00...10 = 2
Direction ↘ = 00...**11** = 00...11 = 3

Example:

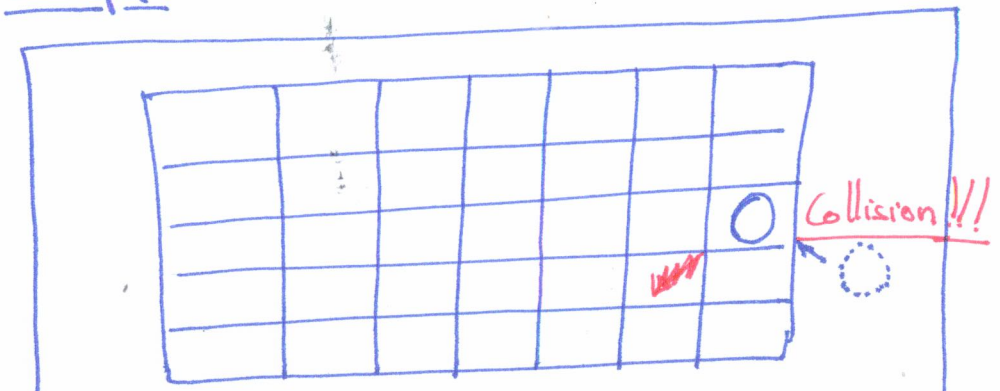
Determine BallDirection:

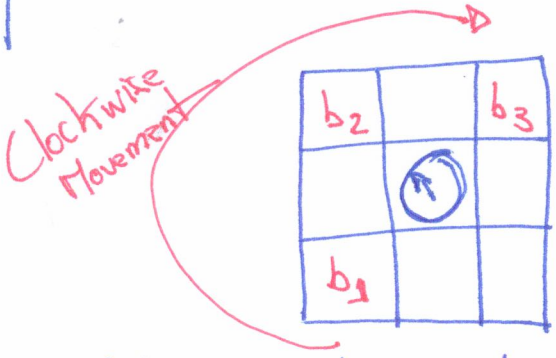
```
AND BallInfo, Direction↑
JMP.NZ BallDirectionIs↑
AND BallInfo, Direction↗
JMP. NZ BallDirectionIs↗
AND BallInfo, Direction↙
JMP. NZ BallDirectionIs↙
AND BallInfo, Direction↘
JMP. NZ BallDirectionIs↘
```

Regarding wall collisions:

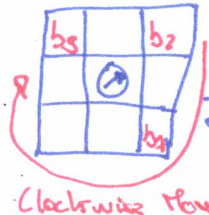
- Lets assume a simplified collision method (nothing too fancy)

- Example: Clockwise collision method





If b_2 is empty go to b_2 return;
If b_1 is empty go to b_1 return;
If b_3 is empty go to b_3 return;



If b_2 is empty go to b_2 return;
If b_1 is empty go to b_1 return;
If b_3 is empty go to b_3 return;