Magic Bitboards:

Goal: Generate the possible moves for sliding pieces by removing paths blocked by blockers.

Find the mask for which squares would effect the rooks movement

A screenshot of a game

Description automatically generated

The mask would be the bottom left. Squares at the end of the path aren’t considered here as if a

This mask is simply the mask of squares where if a piece was on these squares, itd limit the rook’s movement. It is the potential\_blockers\_mask.

We can then & this mask with board to get the occupancy bitboard. (The actual mask of blocking pieces). This is referred to as the occupancy bitboard.

Then multiply by a magic number to bring the list of bits in the blockers mask to the front of the mask, such that it uses the first n bits, where n is the number of blockers. This reduces the size of the 64 bit bb to say, a 5 bit bb.

This magic number should be unique depending on the piece and square. For example, a rook on a1 would have a different magic number than a bishop on a1, and a rook on a2.

Then, these bits would represent the index of a precomputed lookup table of attack bitboards for that piece on that square. This final bitboard is the bitboard that shows the squares the rook on a1 can move to, considering the other pieces on the board.

Uninterrupted attack mask (standard lookup from the ROOK\_ATTACKS etc)

Relevant blocking squares mask (if another piece would on this, itd block the rook from getting past)

Actual squares of blocking pieces (occupancy bitboard) by & with board mask

Multiply by magic number to access lookup table

Use this index to get the attack mask for the rook on a1, considering the occupancy bitboard.

Maybe use kindergarten instead