Software Performance Engineering

Recitation 1.2

Sophia Sun
Tuesday, Sep 9, 2025

Course Logistics

Homework 2 is released!

- You have a checkoff part and the actual homework part
- Checkoff is due: tonight 10 pm
- Homework 2 is due: Monday September 15

- Project 1 beta: September 23, Tuesday
- Project 1 final: October 2, Thursday

- Reach tier 25 for Project 1 beta to get **at least B grade**!

Gradescope

- For write-ups, when you make your submission, please link each question to the page that you answer is on.
- Eg: for question 1, your answer is on page 1 and 2. For question 2, your answer is on page 2 and 3.
- It is ok to link multiple pages to one question!

- This makes it easier for TA to clearly see your submission for every question and don’t miss anything!

Gradescope

- Make sure you follow the handout **carefully** and submit **all the required materials**.

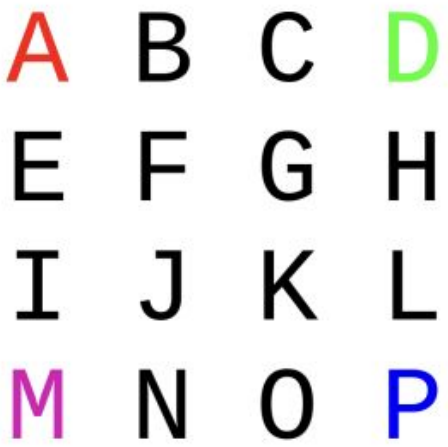
- Write-up 12: Compile again with make clean; make LOCAL=1 UBSAN=1. What do you see when you run ./is_power_of_two? Paste the error that the sanitizer throws. In addition, explain where and why this error happened. Fix the error you found and explain your fix. Finally, compile and run the program again and verify that the output is correct. Paste your program output.

Coding warm up & QA time

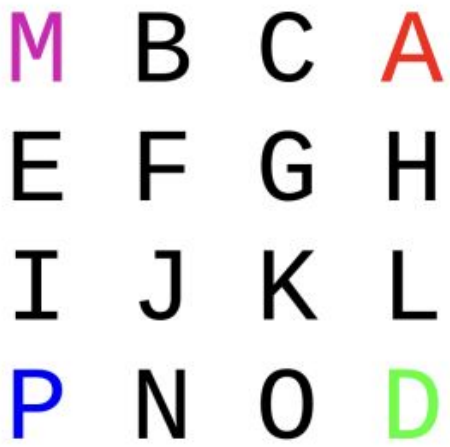
Project 1: continue

```
29  void rotate_bit_matrix(uint8_t *img, const bits_t N) {
30    // Get the number of bytes per row in `img`
31    const uint32_t row_size = bits_to_bytes(N);
32
33    uint32_t w, h, quadrant;
34    for (h = 0; h < N / 2; h++) {
35      for (w = 0; w < N / 2; w++) {
36        uint32_t i = w, j = h;
37        uint8_t tmp_bit = get_bit(img, row_size, i, j);
38
39        // Move a bit from one quadrant to the next and do this
40        // for all 4 quadrants of the `img`
41        for (quadrant = 0; quadrant < 4; quadrant++) {
42          uint32_t next_i = N - j - 1, next_j = i;
43          uint8_t save_bit = tmp_bit;
44
45          tmp_bit = get_bit(img, row_size, next_i, next_j);
46          set_bit(img, row_size, next_i, next_j, save_bit);
47
48          // Update the `i` and `j` with the next quadrant's values and
49          // the `next_i` and `next_j` will get the new destination values
50          i = next_i;
51          j = next_j;
52        }
53      }
54    }
55
56    return;
57  }
```

First step: block-wise rotation

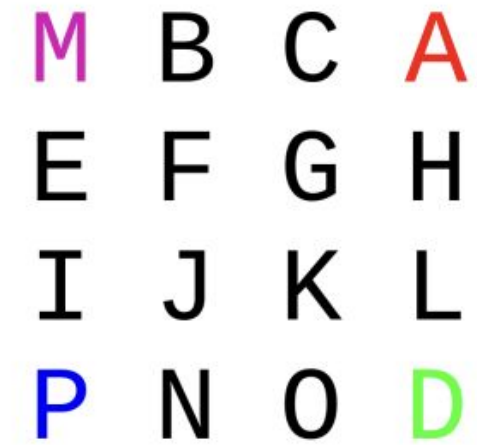


(a) Original image

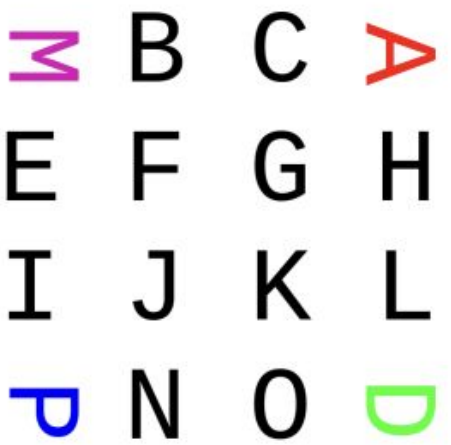


(b) Image after translating blocks

Next step: in-block rotation

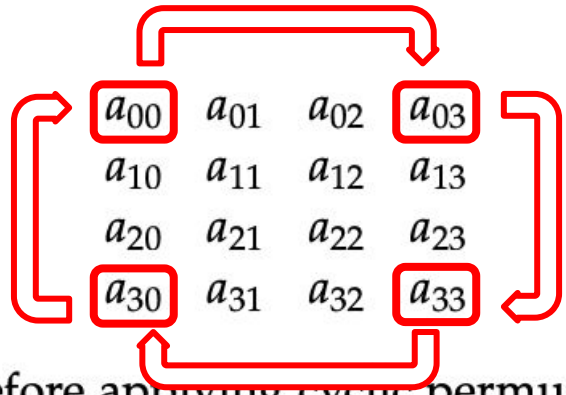


(b) Image after translating blocks



(c) Image after rotating blocks

Next step: in-block rotation



(a) before applying cyclic permutation

$$\begin{matrix} a_{30} & a_{01} & a_{02} & a_{00} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{33} & a_{31} & a_{32} & a_{03} \end{matrix}$$

(b) after applying cyclic permutation

One possible approach: row-column-row

- An algorithm for rotating block of bits
- Basic idea:
  - rotate row r left by r + 1
  - rotate column c down by c + 1
  - rotate row r left by r

Row-column-row algorithm

- An algorithm for rotating block of bits
- Basic idea:
  - rotate row r left by r + 1
  - rotate column c down by c + 1
  - rotate row r left by r

	0	1	2	3
0	A	B	C	D
1	E	F	G	H
2	I	J	K	L
3	M	N	O	P

Row-column-row algorithm

- An algorithm for rotating block of bits
- Basic idea:
  - **rotate row r left by r + 1**
  - rotate column c down by c + 1
  - rotate row r left by r

	0	1	2	3
0	A	B	C	D
1	E	F	G	H
2	I	J	K	L
3	M	N	O	P

Row-column-row algorithm

- An algorithm for rotating block of bits
- Basic idea:
  - **rotate row r left by r + 1**
  - rotate column c down by c + 1
  - rotate row r left by r

	0	1	2	3
0	B	C	D	A
1	E	F	G	H
2	I	J	K	L
3	M	N	O	P

Row-column-row algorithm

- An algorithm for rotating block of bits
- Basic idea:
  - **rotate row r left by r + 1**
  - rotate column c down by c + 1
  - rotate row r left by r

	0	1	2	3
0	B	C	D	A
1	E	F	G	H
2	I	J	K	L
3	M	N	O	P

Row-column-row algorithm

- An algorithm for rotating block of bits
- Basic idea:
  - **rotate row r left by r + 1**
  - rotate column c down by c + 1
  - rotate row r left by r

	0	1	2	3
0	B	C	D	A
1	G	H	E	F
2	I	J	K	L
3	M	N	O	P

Row-column-row algorithm

- An algorithm for rotating block of bits
- Basic idea:
  - **rotate row r left by r + 1**
  - rotate column c down by c + 1
  - rotate row r left by r

	0	1	2	3
0	B	C	D	A
1	G	H	E	F
2	I	J	K	L
3	M	N	O	P

Row-column-row algorithm

- An algorithm for rotating block of bits
- Basic idea:
  - **rotate row r left by r + 1**
  - rotate column c down by c + 1
  - rotate row r left by r

	0	1	2	3
0	B	C	D	A
1	G	H	E	F
2	L	I	J	K
3	M	N	O	P

Row-column-row algorithm

- An algorithm for rotating block of bits
- Basic idea:
  - **rotate row r left by r + 1**
  - rotate column c down by c + 1
  - rotate row r left by r

	0	1	2	3
0	B	C	D	A
1	G	H	E	F
2	L	I	J	K
3	M	N	O	P

Row-column-row algorithm

- An algorithm for rotating block of bits
- Basic idea:
  - **rotate row r left by r + 1**
  - rotate column c down by c + 1
  - rotate row r left by r

	0	1	2	3
0	B	C	D	A
1	G	H	E	F
2	L	I	J	K
3	M	N	O	P

Rotate by 4 = no change

Row-column-row algorithm

- An algorithm for rotating block of bits
- Basic idea:
  - rotate row r left by r + 1
  - **rotate column c down by c + 1**
  - rotate row r left by r

	0	1	2	3
0	B	C	D	A
1	G	H	E	F
2	L	I	J	K
3	M	N	O	P

Row-column-row algorithm

- An algorithm for rotating block of bits
- Basic idea:
  - rotate row r left by r + 1
  - **rotate column c down by c + 1**
  - rotate row r left by r

	0	1	2	3
0	M	C	D	A
1	B	H	E	F
2	G	I	J	K
3	L	N	O	P

Row-column-row algorithm

- An algorithm for rotating block of bits
- Basic idea:
  - rotate row r left by r + 1
  - **rotate column c down by c + 1**
  - rotate row r left by r

	0	1	2	3
0	M	C	D	A
1	B	H	E	F
2	G	I	J	K
3	L	N	O	P

Row-column-row algorithm

- An algorithm for rotating block of bits
- Basic idea:
  - rotate row r left by r + 1
  - **rotate column c down by c + 1**
  - rotate row r left by r

	0	1	2	3
0	M	I	D	A
1	B	N	E	F
2	G	C	J	K
3	L	H	O	P

Row-column-row algorithm

- An algorithm for rotating block of bits
- Basic idea:
  - rotate row r left by r + 1
  - **rotate column c down by c + 1**
  - rotate row r left by r

	0	1	2	3
0	M	I	D	A
1	B	N	E	F
2	G	C	J	K
3	L	H	O	P

Row-column-row algorithm

- An algorithm for rotating block of bits
- Basic idea:
  - rotate row r left by r + 1
  - **rotate column c down by c + 1**
  - rotate row r left by r

	0	1	2	3
0	M	I	E	A
1	B	N	J	F
2	G	C	O	K
3	L	H	D	P

Row-column-row algorithm

- An algorithm for rotating block of bits
- Basic idea:
  - rotate row r left by r + 1
  - **rotate column c down by c + 1**
  - rotate row r left by r

	0	1	2	3
0	M	I	E	A
1	B	N	J	F
2	G	C	O	K
3	L	H	D	P

Row-column-row algorithm

- An algorithm for rotating block of bits
- Basic idea:
  - rotate row r left by r + 1
  - rotate column c down by c + 1
  - **rotate row r left by r**

	0	1	2	3
0	M	I	E	A
1	B	N	J	F
2	G	C	O	K
3	L	H	D	P

Row-column-row algorithm

- An algorithm for rotating block of bits
- Basic idea:
  - rotate row r left by r + 1
  - rotate column c down by c + 1
  - **rotate row r left by r**

	0	1	2	3
0	M	I	E	A
1	B	N	J	F
2	G	C	O	K
3	L	H	D	P

Row-column-row algorithm

- An algorithm for rotating block of bits
- Basic idea:
  - rotate row r left by r + 1
  - rotate column c down by c + 1
  - **rotate row r left by r**

	0	1	2	3
0	M	I	E	A
1	N	J	F	B
2	G	C	O	K
3	L	H	D	P

Row-column-row algorithm

- An algorithm for rotating block of bits
- Basic idea:
  - rotate row r left by r + 1
  - rotate column c down by c + 1
  - **rotate row r left by r**

	0	1	2	3
0	M	I	E	A
1	N	J	F	B
2	G	C	O	K
3	L	H	D	P

Row-column-row algorithm

- An algorithm for rotating block of bits
- Basic idea:
  - rotate row r left by r + 1
  - rotate column c down by c + 1
  - **rotate row r left by r**

	0	1	2	3
0	M	I	E	A
1	N	J	F	B
2	O	K	G	C
3	L	H	D	P

Row-column-row algorithm

- An algorithm for rotating block of bits
- Basic idea:
  - rotate row r left by r + 1
  - rotate column c down by c + 1
  - **rotate row r left by r**

	0	1	2	3
0	M	I	E	A
1	N	J	F	B
2	O	K	G	C
3	L	H	D	P

Row-column-row algorithm

- An algorithm for rotating block of bits
- Basic idea:
  - rotate row r left by r + 1
  - rotate column c down by c + 1
  - **rotate row r left by r**

	0	1	2	3
0	M	I	E	A
1	N	J	F	B
2	O	K	G	C
3	P	L	H	D

Row-column-row algorithm

- An algorithm for rotating block of bits
- Basic idea:
  - rotate row r left by r + 1
  - rotate column c down by c + 1
  - rotate row r left by r

	0	1	2	3
0	A	B	C	D
1	E	F	G	H
2	I	J	K	L
3	M	N	O	P

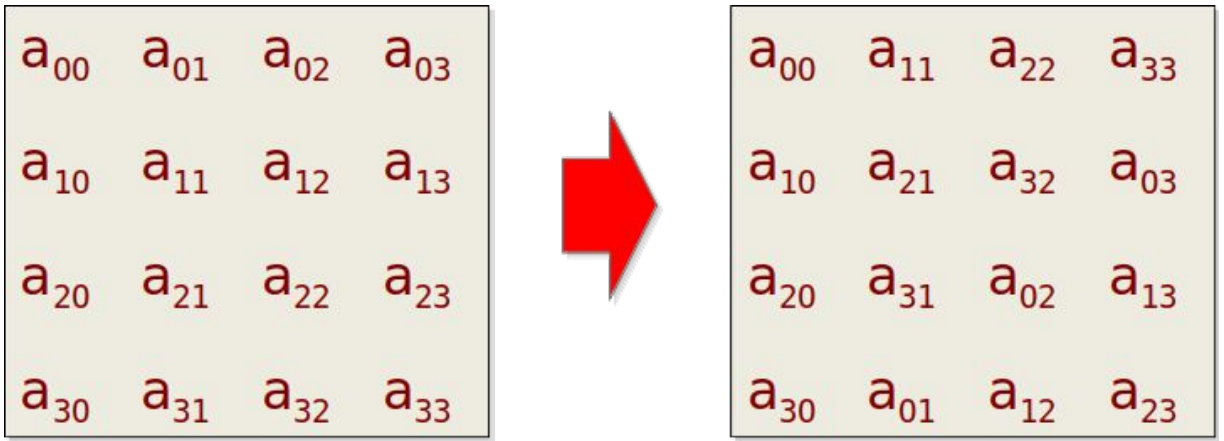
	0	1	2	3
0	M	I	E	A
1	N	J	F	B
2	O	K	G	C
3	P	L	H	D

How to implement row rotation?

Operator	Description
&	AND
|	OR
^	XOR (exclusive OR)
~	NOT (one's complement)
<<	shift left
>>	shift right

How to implement column rotation?

Input: $N \times N$ matrix of bits, stored in row-major order.
Goal: Circularly rotate ith column of bits up i rows.



In the example that follows, we have $N = 32$. Each row is stored in a 32-bit word, with column 0 in the most-significant bit.

Naive approach

```
const uint32_t N = 32;
const uint32_t mask = 1 << (N-1);
uint32_t A[N];
for (int i = 0; i < N; i++){
  uint32_t col = 0;
  // gather bits in column i
  for (int j = 0; j < N; j++)
    col = col | (((A[j] << i) & mask) >> j);
  // rotate bits in column i
  col = (col << i) | (col >> (N - i));
  // put column i back
  for (int j = 0; j < N; j++)
    A[j] = (A[j] & ~(mask >> i)) |
        (((col << j) >> i) & (mask >> i));
}
```

Coding time!

More hints

- Loop unrolling
- Function inlining
- Algebraic identities
- Tiling
- Prefetching
- ……