Software Performance Engineering


# Recitation 1.1

Sophia Sun

Tuesday, Sep 2, 2025

# Course Logistics

# Some due dates

The passed due dates:

- HW0 (bio sign up and course info quizle): Tuesday, Aug 26
- HW1 checkoff: Friday, Aug 29
- HW0 extension: Sunday, Aug 31
- HW1: Monday, Sep 1

Upcoming due dates:

- Weekly report for week 1: Wednesday, Sep 3
- Project 1 Beta: Tuesday, September 23
- Project 1 Final: Thursday, October 9

# Office Hour

- No fixed office hour
- reach out during recitation & on piazza
- If you need some in-person discussion time, schedule an office hour with me!
- My email: sunzeai@msu.edu
- My office: EB 3353

# Semester overview

- 6 homeworks
  - follow the handout instructions
  - submit your write-up
    - Gradescope and GitHub homework repo
  - some part will be checked off in class
- 3 projects (1 optional for extra credit)
  - project 1: invididual, project 2: assigned team
  - beta submission & beta write-up
  - anonymized code shared for everyone after beta
  - final submission & final write-up

# Weekly status report

- A short paragraph of your weekly summary
- Open on Fridays, close on Mondays, submit on Gradescope
- What you can talk about:
  - Your impression of the past week's lectures and recitations
  - Your feelings regarding your project partner or the course in general.
  - Feedback on interactions you had with the course staff.
  - Reflect on how effectively you feel you are working on class assignments and how much time you're devoting.
  - Identify the aspects of the class you found most engaging or frustrating.
  - Offer constructive criticism about how the class is being taught.
  - Report on any outside events that may have impacted your work on the course

# The late-day policy for assignments

- 3 late days total for each semester
- atomic, no subdivision (no 0.5 day)
- homework assignment and project write-up only, not for project beta and final assignment (the coding part)
- inform the TA **before the deadline**
  - make a private post on piazza with #late-days tag
- if exceeding the 3 late days: require a note from Student Support Service to accept your late work

# Let's start project 1!

# Summarize homework 1

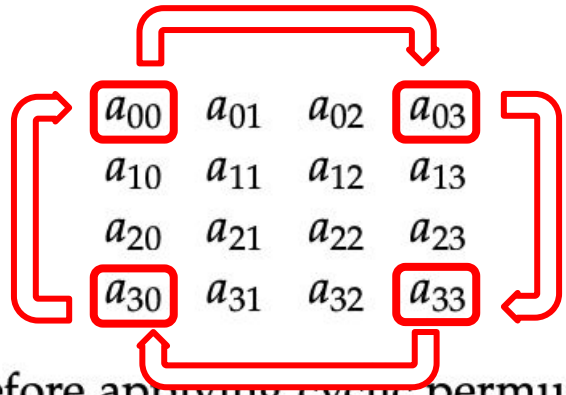
- set up and test out the virtual environment
- basic C programming practice
- useful tools:
  - debugging tool: GDB, DEBUG=1, tbassert
  - memory checker: address sanitizer, valgrind
  - code coverage: llvm-cov

# Project 1: Bit Hacks

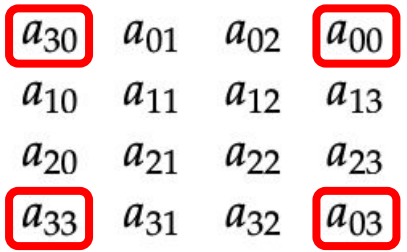

**Figure 1:** **(a)** The original 192-by-192 monochrome image. **(b)** The image after rotation.

```c
void rotate_bit_matrix(uint8_t *img, const bits_t N) {
  // Get the number of bytes per row in `img`
  const uint32_t row_size = bits_to_bytes(N);

  uint32_t w, h, quadrant;
  for (h = 0; h < N / 2; h++) {
    for (w = 0; w < N / 2; w++) {
      uint32_t i = w, j = h;
      uint8_t tmp_bit = get_bit(img, row_size, i, j);

      // Move a bit from one quadrant to the next and do this
      // for all 4 quadrants of the `img`
      for (quadrant = 0; quadrant < 4; quadrant++) {
        uint32_t next_i = N - j - 1, next_j = i;
        uint8_t save_bit = tmp_bit;

        tmp_bit = get_bit(img, row_size, next_i, next_j);
        set_bit(img, row_size, next_i, next_j, save_bit);

        // Update the `i` and `j` with the next quadrant's values and
        // the `next_i` and `next_j` will get the new destination values
        i = next_i;
        j = next_j;
      }
    }
  }

  return;
}
```

# Basic idea: “follow the cycles”




(a) before applying cyclic permutation




(b) after applying cyclic permutation

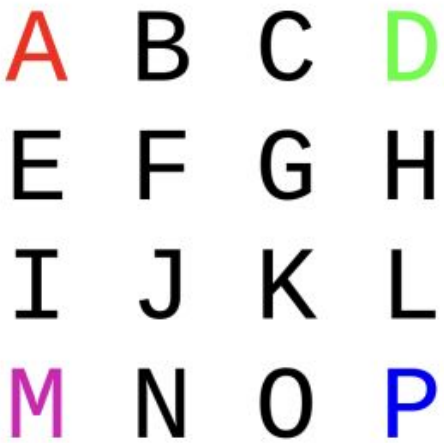
By applying the same rotation to each group of 4 bits, the program ends up rotating the entire matrix clock-wise.

# Performance?

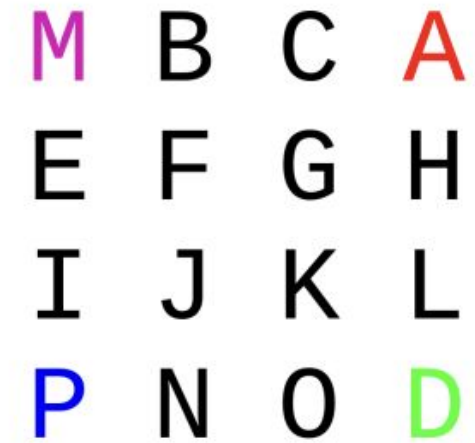
```
--- Execution log:

FYI: the max tier you can be graded on is 57.
Setting up test up to tier 25: Malloc 72192x72192 matrix...
Linear search from tier 0 to 8...
FAIL (timeout): Tier 0 :        Rotated 26624x26624     matrix in 1571 ms but the cutoff is 1000 ms
Blowing through this failure. Remaining blowthroughs: 1
PASS (ayy!):    Tier 1 :        Rotated 27712x27712     matrix in 850 ms
PASS (skrrt!):  Tier 2 :        Rotated 28864x28864     matrix in 948 ms
FAIL (timeout): Tier 3 :        Rotated 30080x30080     matrix in 1035 ms but the cutoff is 1000 ms
Blowing through this failure. Remaining blowthroughs: 0
FAIL (timeout): Tier 4 :        Rotated 31296x31296     matrix in 1167 ms but the cutoff is 1000 ms
Result: reached tier 2
```

# Some analogy

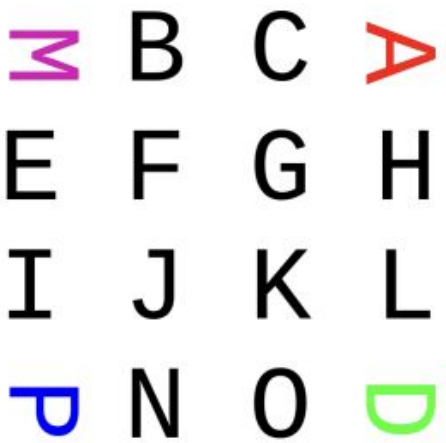
# First step: from bit to block




(a) Original image




(b) Image after translating blocks




(c) Image after rotating blocks

Rotate the blocks 4 in a group -> rotate inside each block

# Some notice in advance:

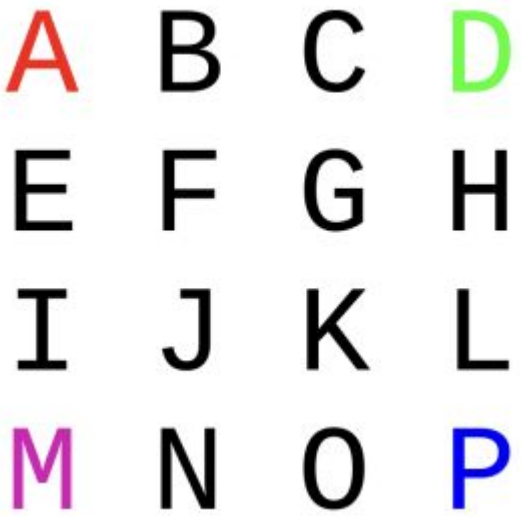
- No parallelism in project 1
- No built-in intrinsics for beta submission
  - if you don't know what built-in intrinsics are, we'll cover that in upcoming recitations
- Don't jump too fast, implement one step at a time
- Check your performance by `./rotate -t tiers`
  - I'd recommend using telerun as the performance is more consistent on telerun than testing locally
- Edit your `Makefile`:
  - `CC := clang-spe`
  - `ARCH := x86-64-v3`

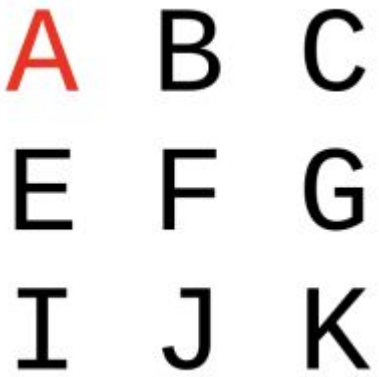
# Coding time!

# Endianess

- Little endian: bytes of words are stored in memory with least-significant bytes first
- 
- the actual word: 1A 2B 3C 4D 5E 6F 7A 8B
- store in memory: 8B 7A 6F 5E 4D 3C 2B 1A
- 
- If you encounter any bugs, think about the endianess

# Edge cases



# Coding time!