15-213 Recitation 7

Introduction to Computer Systems

Fahim Dalvi

10 October, 2013

Today

- Cachelab
- Exam stuff

Cachelab

- Due: 21st, Monday after Eid
- Very different from the previous 3 homeworks
  - Requires you to code for "real"
- Warnings are treated as errors
  - For your own good
- Brush up on your C
  - http://www.qatar.cmu.edu/~amtibaa/15-123/schedule.html
  - Start from the "Intro to C Programming" lecture

Cachelab – Bird's Eye view

- Part (a)
  - Cache Simulator
  - A 200-300 line program
- Part (b)
  - Optimizing matrix transpose
  - All about performance!

Part (a) – Cache Simulator

- You will be implementing cache simulator
  - A cache simulator is NOT a cache!
  - Memory contents NOT stored!
  - Simply counts hits, misses, and evictions

Part (a) – Cache Simulator

- You will be implementing cache simulator
  - A cache simulator is NOT a cache!
  - Memory contents NOT stored!
  - Simply counts hits, misses, and evictions

Question: How will you use the block offsets?

Part (a) – Cache Simulator

- Your cache simulator need to work for different s, b, E, given at run time.
- Uses LRU replacement policy
- Will be tested on various “tracefiles”

Files - fscanf

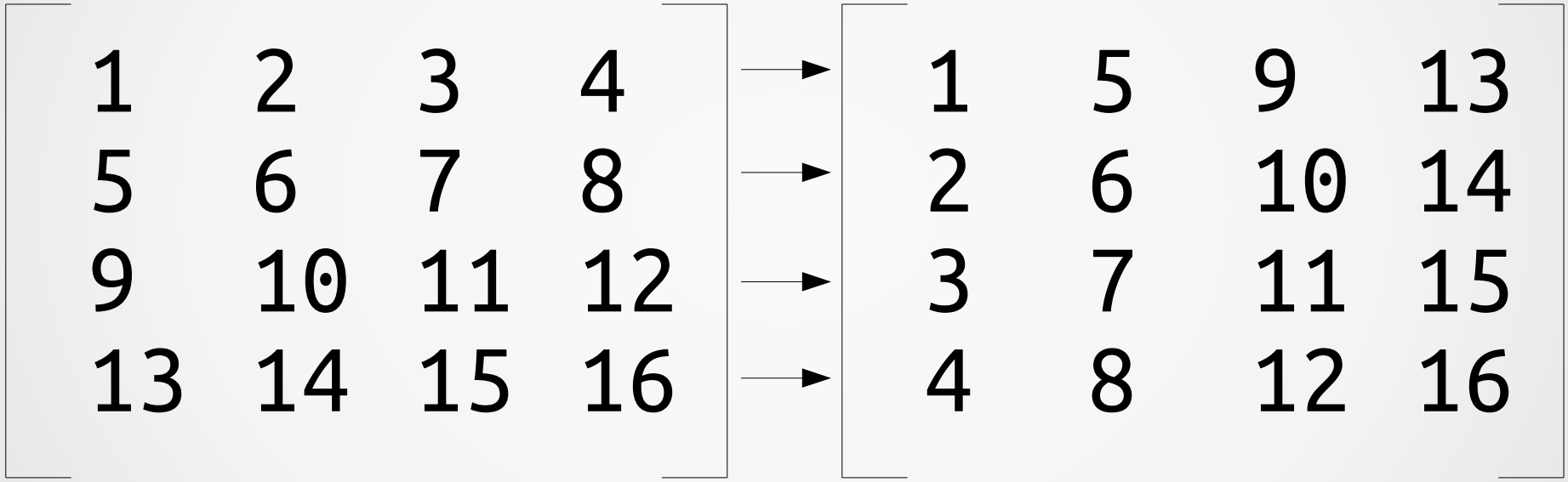
- Include `<stdio.h>`
- `File *my_fp=fopen(char * filename, char *mode)`
  - Mode = “r” → read, “w” → write, “w+” → append
  - Returns NULL if opening fails
- `fscanf(fp,char *format, pointers to vars...`
  - Same format as printf
  - Returns number of items scanned
- `fclose(fp)`
  - Don't forget to do this when done with the file

Options - getopt

- Makes your life infinitely (maybe not) easier
  - You don't HAVE to use it
- Your programs MUST use the same command line arguments as the reference programs or the autograder will not work
- Learn more about it:
  - `man 3 getopt`

Part (b) – Optimizing Matrix Transpose

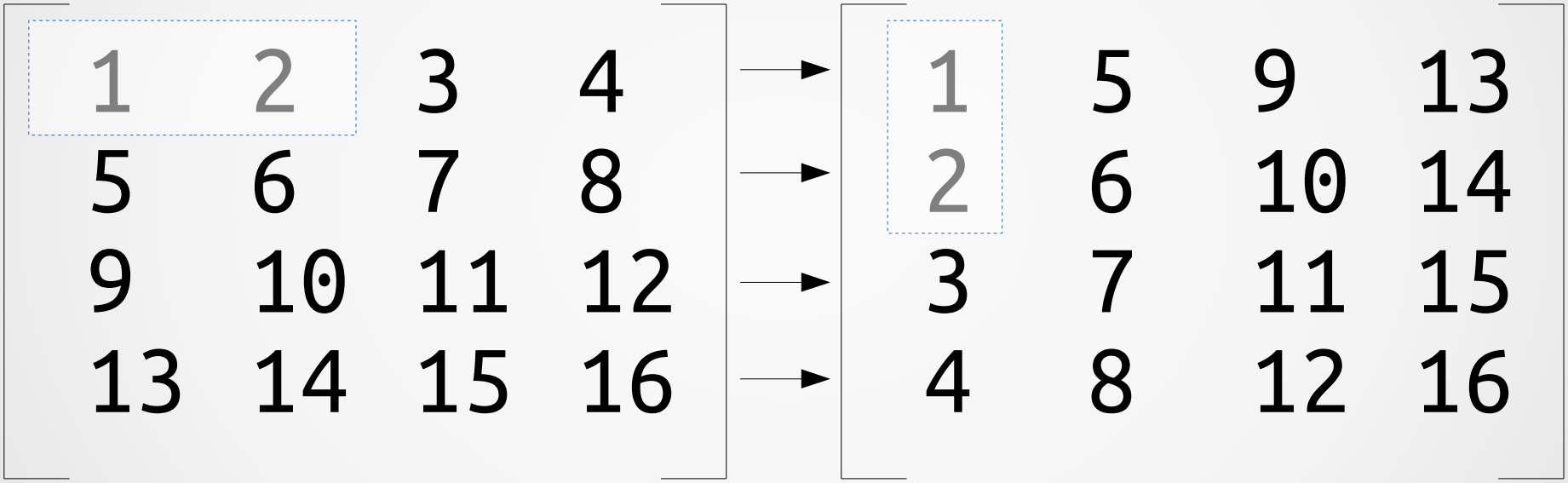
- Matrix transpose:



Matrix A Matrix B

Part (b) – Optimizing Matrix Transpose

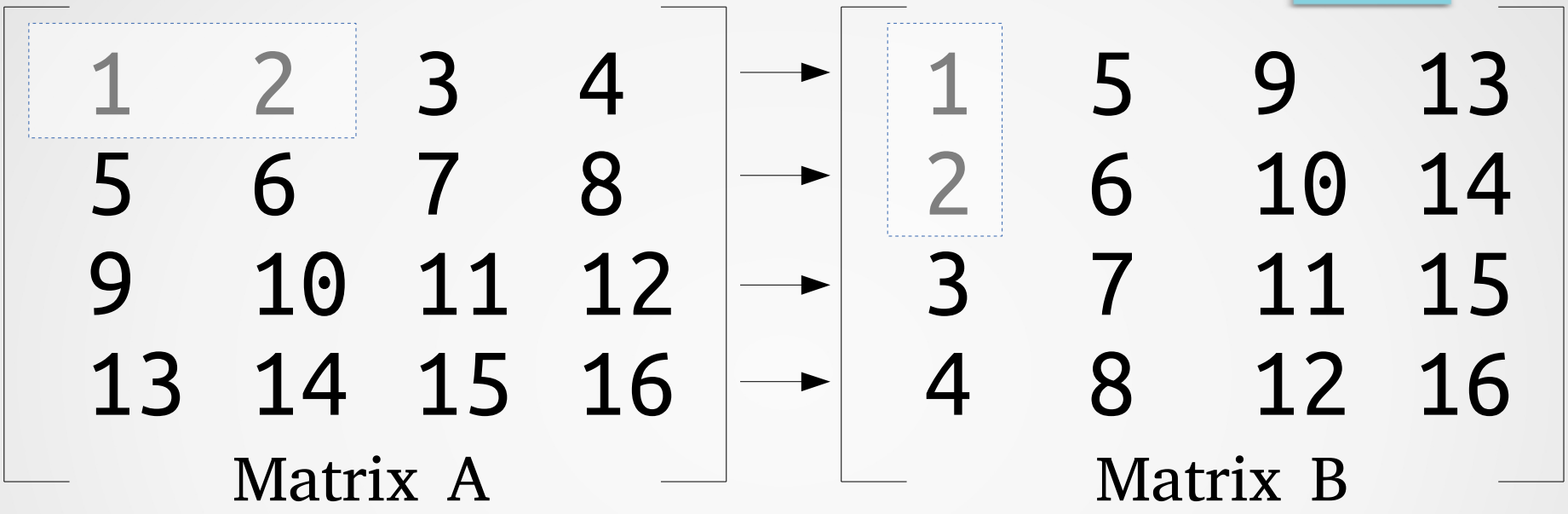
- Matrix transpose:



Matrix A Matrix B

Lets see what happens if the cache size is 8 bytes

Part (b) – Optimizing Matrix Transpose



- Access A[0][0] → cache miss
- Access B[0][0] → cache miss
- Access A[0][1] → cache hit
- Access B[1][0] → cache miss

- After we handle 1 & 2, Should we handle 3 & 4 first, or 5 & 6 first?

Part (b) – Optimizing Matrix Transpose

- What inspiration do you get from previous slide ?
  - Divide matrix into sub-matrices
  - This is called **blocking**
  - Size of sub-matrix depends on
    - cache block size, cache size, input matrix size
  - Try different sub-matrix sizes
- We hope you invent more tricks to reduce the number of misses !

Part (b) – Optimizing Matrix Transpose

- Cache:
  - Directly mapped → E=1
  - Block size is 32 bytes → b = 5
  - There are 32 sets → s = 5
  - Question: What is the size of the cache?
- Test Matrices:
  - 32 by 32, 64 by 64 and 61 by 67

Questions?

- Feel free to ask on Piazza!
- Eid Mubarak :D