CSE 350: Advanced Data Structures and Indexes (Spring 2026)

Lecture 1: Introduction & Course Logistics;

Storage Hierarchy and External Memory Model

1/22/2026

University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

Lecture: Norton 213, T&H 3:30 pm – 4:50 pm.
Recitation: Norton 213, H 5:00 pm – 5:50 pm.
In-person attendance required.
Find more on course website & Piazza:
https://cse.buffalo.edu/~zzhao35/teaching/cse350_spring26
https://piazza.com/buffalo/spring2026/cse350/home

Today's agenda

- Introduction
  - DataBase Management Systems (DBMS)
  - Storage hierarchy and data storage systems
  - External memory model
- Course logistics
- Recitation: Repository setup; C++ programming and POSIX I/O Interface

What is a Database?

- Database is
  - a collection of interrelated data
  - often organized in a certain structure for convenient and efficient access

- Databases are found almost everywhere, sometimes unnoticed
  - Business: sales, accounting, human resource, IT support, …
  - Financial industry: banking, credit card, investment platform
  - University: student records, course registration, LMS (e.g., UB Learns), ...
  - Some less obvious examples of databases
    - Software package and configuration DB (e.g., windows registry)
    - Your photo library (e.g., Google Photos)
    - Your personal finance records
    - …

Why using a DataBase Management System?

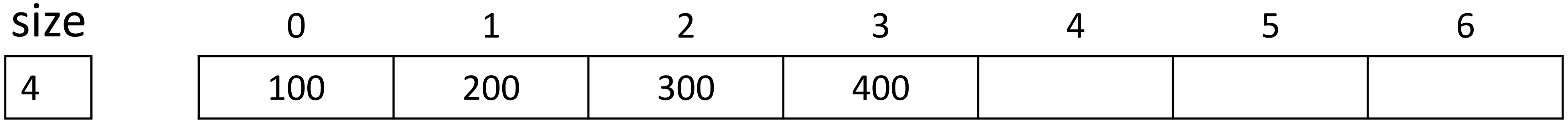
- DataBase Management System (DBMS) is a software system for convenient and efficient data access over databases,

  which provides:
  - Data abstraction
    - Flexible data manipulation and query interfaces
    - Scalable data storage over storage hierarchy
    - Efficient query and transaction processing
  - Integrity checks
  - Concurrency control and atomicity
  - Fault tolerance
  - Security and privacy
  - …

Data structures

- Data storage (layout) + operations (algorithm)

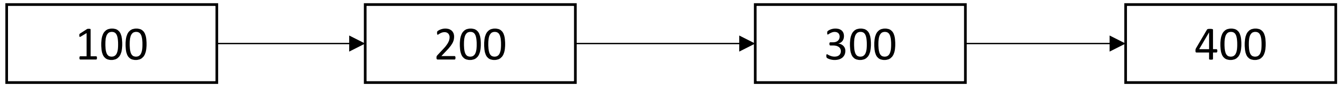
- Example: in-memory data structures
  - 4 integers: 100, 200, 300, 400
  - Need to support lookup (whether the value exists)



size
4

0	1	2	3	4	5	6
100	200	300	400			

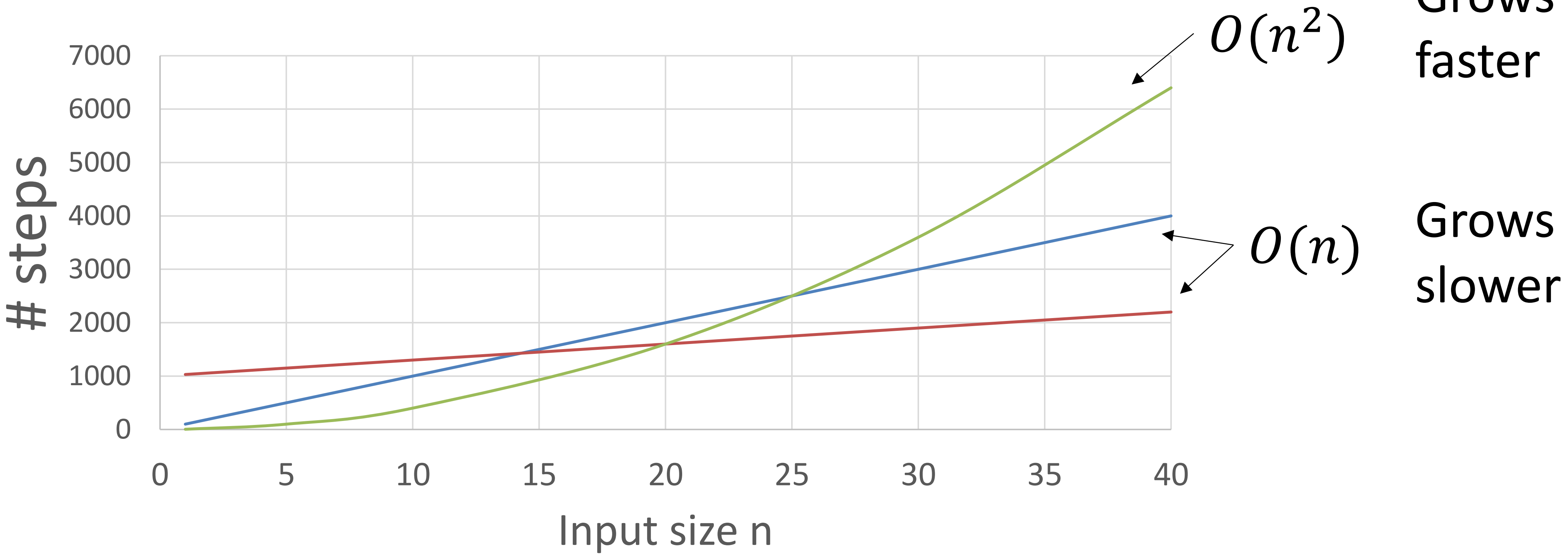
Array Space complexity: Time complexity (lookup):



Linked list Space complexity: Time complexity (lookup):

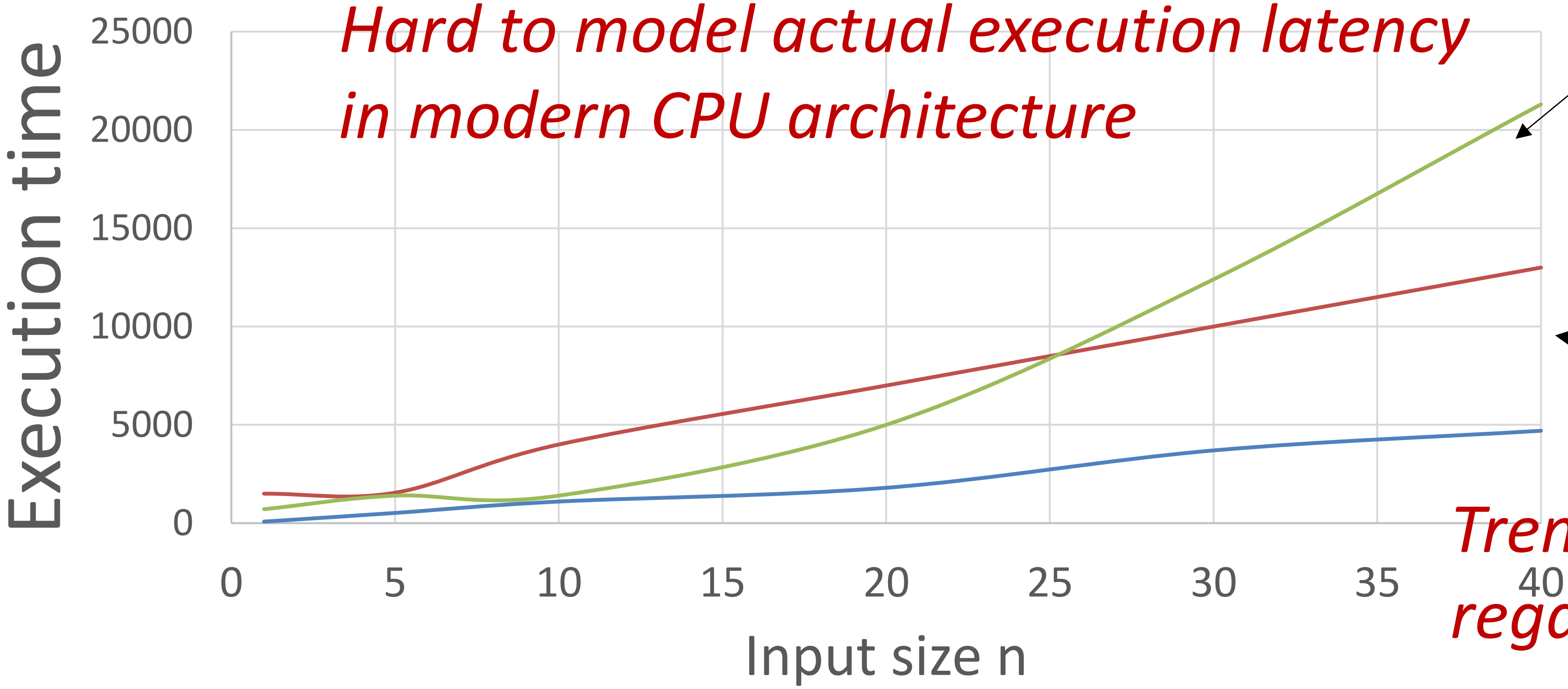
Asymptotic complexity analysis for in-memory

- Assumption: each memory access/computation step takes constant time
  - Scalability matters
    - What is the trend of increase in #steps as input size increases?
  - Ignores constant factors



Asymptotic complexity analysis for in-memory

- Assumption: each memory access/computation step takes constant time
  - Scalability matters
    - What is the trend of increase in #steps as input size increases?
  - Ignores constant factors – why?



Hard to model actual execution latency in modern CPU architecture

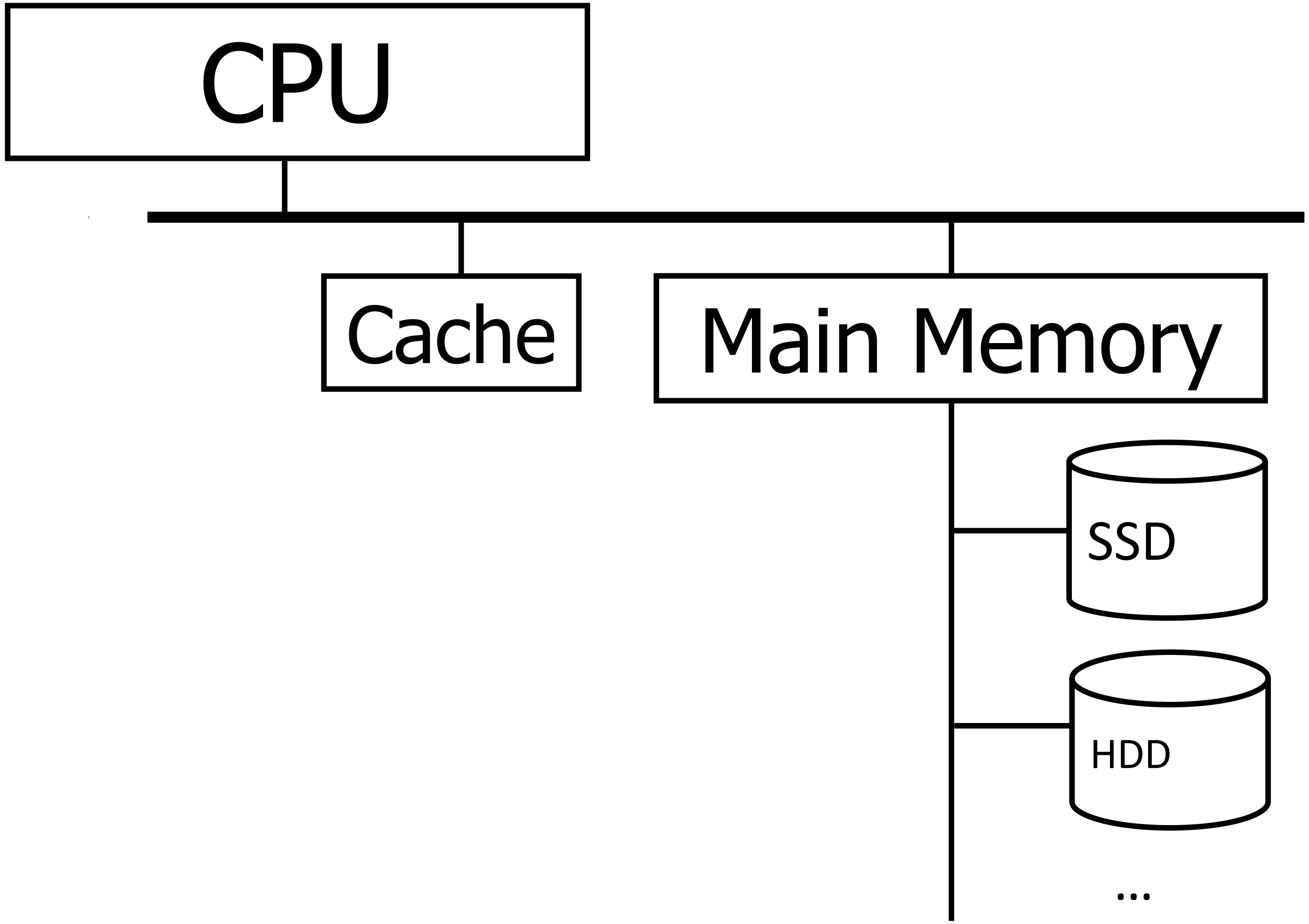
$O(n^2)$ Grows faster

$O(n)$ Grows slower

Trend of growth holds regardless of hardware

Typical (& oversimplified) computer architecture

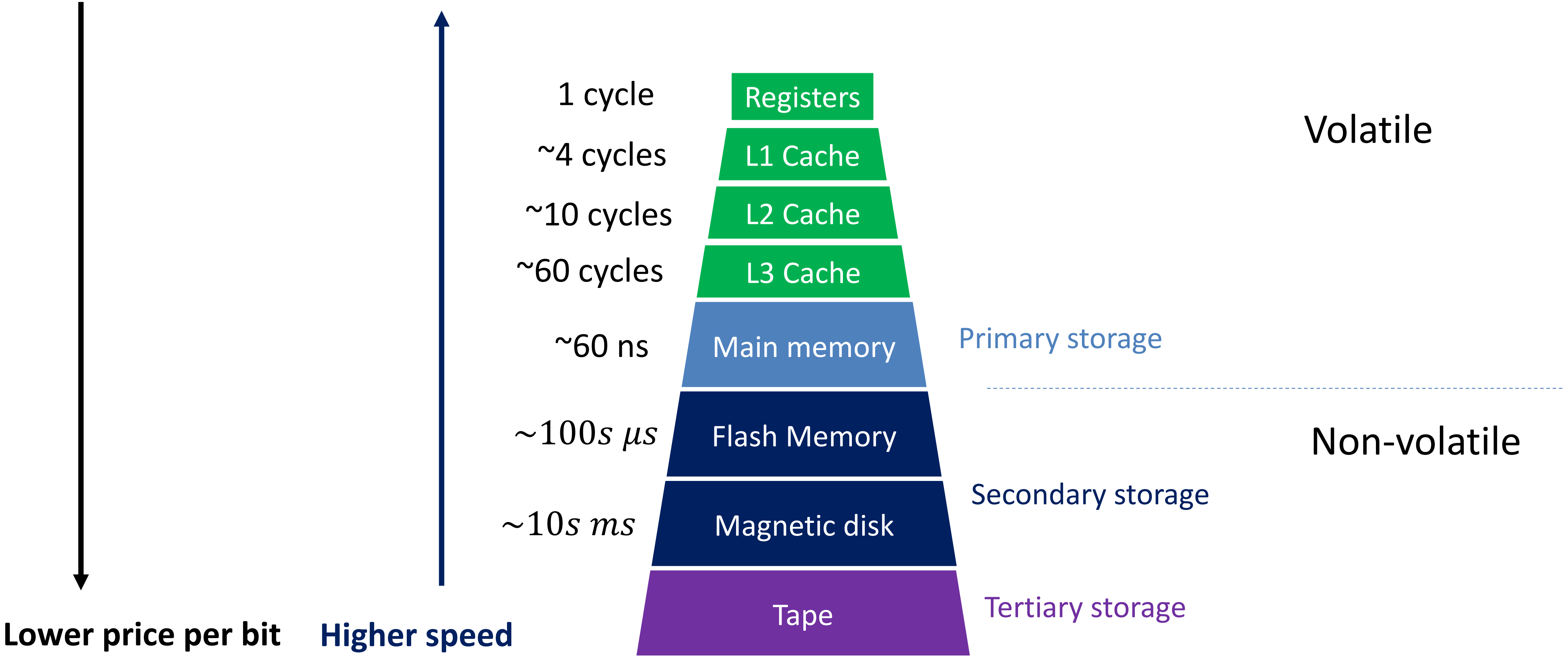
- A simplistic view of a computer



Typical Computer

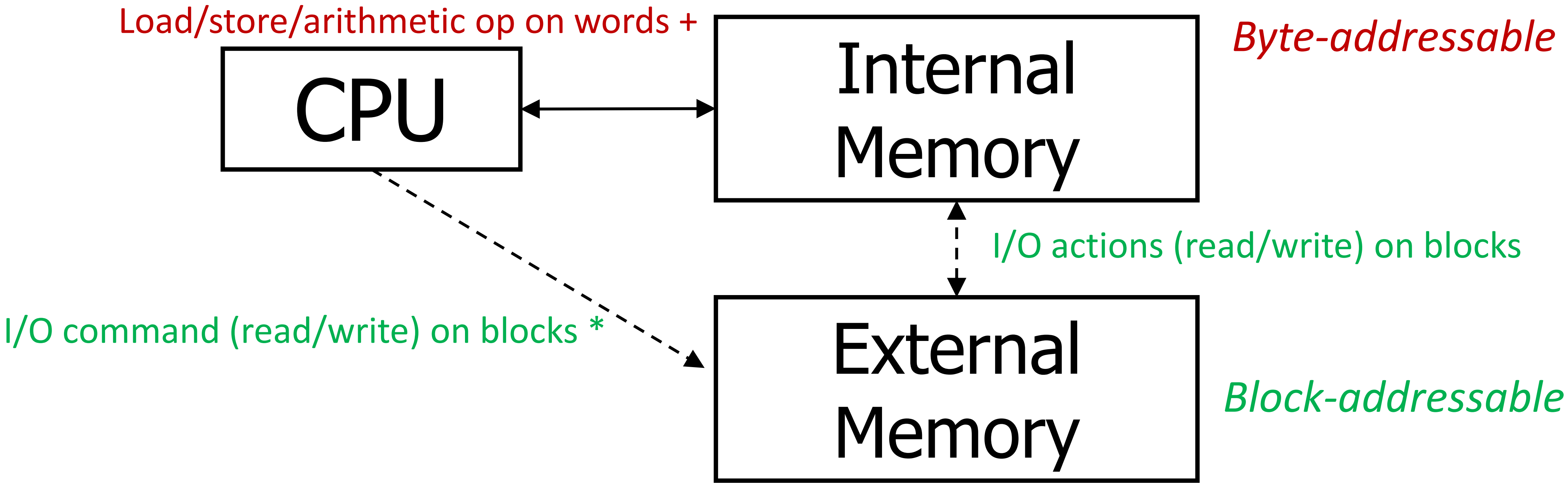
Secondary Storage

Storage Hierarchy



External Memory Model

- Two levels in storage hierarchy
  - I/O latency dominates computation/memory access latencies
  - Complexity analysis will focus on # of I/Os (i.e., # of blocks read/written)


Load/store/arithmetic op on words +
CPU
Internal Memory
Byte-addressable
I/O actions (read/write) on blocks
I/O command (read/write) on blocks *
External Memory
Block-addressable

* One block is a fixed number of consecutive bytes (e.g., 512 B, 4 KB), aligned to modulo = 0 boundaries.
+ A word is a unit of consecutive bytes for operations (e.g., a 4-byte integer).

What dose this course cover?

- The design and analysis of data structures for external memory model
    - Simplified but widely applicable cost model
    - We’ll also cover some SQL/relational model/database design to motivate use cases

- Note, this course is not about
    - Database design, use and administration (CSE 460)
    - Programming/data structure/algorithm analysis/math… (CSE191/CSE 220/250/CSE331)
    - Upper layers in DBMS including query and transaction processing (CSE 462)

Why should I care about DBMS internals?

- > 90 billion dollar worth industry
  - Many more are directly or indirectly using DBMS products
- Many vendors and products:
  - Relational: MySQL, Oracle DB, Microsoft SQL Server, IBM Db2, PostgreSQL, SQLite…
  - Graph DB and Graph data processing: Neo4j, Virtuoso, GraphLab, Spark GraphX, …
  - Stream Processing: Apache Flink, Spark Streaming, Apache Storm, …
  - Semi-structured DB: MongoDB, CouchBase, DocumentDB, …
  - Distributed database: Google Spanner, Microsoft CosmosDB, …
  - …
- Used by many other research and application areas:
  - Artificial Intelligence/data mining/search engine/social media/fintech/…

Why should I care about DBMS internals?

- Huge demand in industry for those who can
  - query/manipulate data in database efficiently
  - fine-tune the imperfect DBMS/big data processing systems
  - work seamlessly with the data infrastructure team

- An actively researched area that
  - has strong real-life impacts and connection to the industry
  - has many related open engineering and research positions

- The goal of this course:
  - understanding the common problems and solutions in data management
  - gaining hands-on experience with building a complex software system
  - to be helpful in your future industrial/academic career

Logistics

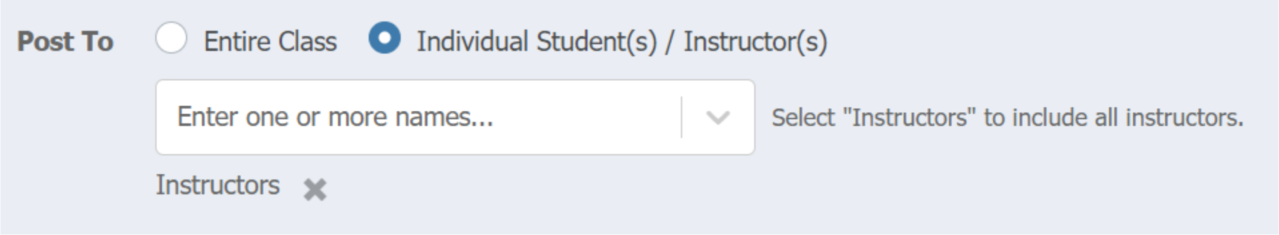
- Norton 213, T&H 3:30 pm – 4:50 pm (lecture), H 5:00 – 5:50 pm (recitation)
  - In-person attendance required for both lectures and recitations.
- Office hours:
  - No TA
  - Instructor (Zhuoyue Zhao)
    - No regular hours; message me for appointment
  - Rules:
    - First-come first-serve
    - Please come with concrete questions about course materials/projects/assignments/exams
      - Not intended for code debugging, troubleshooting of your dev environment, etc.
      - If you need help with troubleshooting of your dev environment, message me on Piazza
        - With screenshots or copy of error messages.

- Find more on course website: https://cse.buffalo.edu/~zzhao35/teaching/cse350_spring26

Logistics

- We mainly use Piazza for communication:
  - https://piazza.com/buffalo/spring2026/cse350/home
  - Please post messages on Piazza instead of sending emails

- When you have any private question/request for me:
  - please select “Instructors” in Post To



Post To ○ Entire Class ● Individual Student(s) / Instructor(s)

Enter one or more names... Select "Instructors" to include all instructors.

Instructors ✖

Logistics

- Important Dates:
  - Mid-term exam: 3/5/2026, Norton 213 (in class), 3:40 pm – 5:40 pm (120 minutes)
  - Final exam: 5/12/2026, Norton 213, 3:40 – 5:40 pm (120 minutes)
- Exam conflict policy:
  - If you have final exam conflicts as defined by the Office of the Registrar
    - please notify the instructor on Piazza by 2/4/2026
    - (we might not have enough seats if you do not notify us by that date)
    - you may still opt for the original final exam at any time with one-week prior notice

Grading

- Grading
  - Programming assignments: 50%
  - Quizzes: 20%
  - Mid-term exam: 15%
  - Final exam: 15%

- Grading policy:
  - No curving.

[0, 10)	[10, 20)	[20, 30)	[30, 40)	[40, 50)	[50, 60)	[60, 70)	[70, 80)	[80, 90)	[90, +∞)
F	D	C-	C	C+	B-	B	B+	A-	A

Exams and Written Assignments

- No written assignments
- Random In-class quizzes
  - Must hand-write answers and submit in class
  - Typically lasts 10 - 20 minutes
  - Please bring a pen and a calculator to every lecture and recitation

- Exams
  - Open-book exams
    - Only paper-copy of the course slides and lecture notes, the quizzes and solutions, the optional textbook, and your own lecture notes are allowed
    - No electronic devices except a calculator

Programming assignments

- Build an LSM tree in C++ 17 (4 projects)
- No teams
  - Must not collaborate on programming
  - see academic integrity policy for details
- Using generative AI is disallowed
  - No ChatGPT/Gemini/Claude/….
  - No Github Copilot (if you use an IDE, please make sure to disable it)
- Code must be kept in a private Github repository, even after this semester
  - You should not fork the repository or clone the repository outside the dev container

Academic Integrity Policy

- Academic integrity is critical to the learning process. It is your responsibility to understand and follow all the departmental and university academic integrity policies.
- Zero tolerance towards academic integrity violations, which includes but are not limited to
  - Sharing/copying code in programming assignments or
  - Plagiarizing write-ups
  - Cheating in exam
  - Making your code publicly available or available to any current or future students
  - Submitting code repository that does not belong to you
  - ***Use of generative AI in this class for any coursework***
- Any AI violation will result in an F grade and will be reported to the Office of Academic Integrity
  - Amnesty Policy: If you have concerns that you may have violated academic integrity on a particular assignment, and would like to withdraw the assignment, you may do so by sending your instructor an email **BEFORE THE VIOLATION IS DISCOVERED BY COURSE STAFF**.
    See PDF syllabus on the course website for details.

More on Academic Integrity Policy

- Examples of AI violation related to course project:
  - Viewing/committing/submitting code written by anyone else
    - ***including those generated or adapted from outputs from generative AI software (e.g., ChatGPT, Cursor, Github CoPilot, etc.)***
  - Viewing/copying/rephrasing answers found online or from a past or current student (including CSE 4/562)

- What is allowed and encouraged (on Piazza/in lecture/offline, publicly or privately)
  - Ask questions about lectures/projects/homework assignments
  - Preparation for mid-term and final exams
  - Seek clarification about projects/homework assignments
  - If you're unsure, please do ask.

Programming assignments

- Instructions for projects:

  Assignment pages contain very detailed instructions.
    - If something requires clarification, it’s most likely covered there.
  - Still have questions on coding or found bugs?
    - Feel free to post it on Piazza (though we may point you back to the instructions).
    - Your will get 1 extra credit towards your final grade for every validated bug or question that cannot be answered by the project instruction.

- Where to find project pages: https://cse.buffalo.edu/~zzhao35/teaching/cse350_spring26

Late policy for programming assignments

- You will always be graded on the latest submission
- Up to 5 grace days in total and up to 2 grace days for each project
  - only applicable to final submissions
  - no grace days allowed for intermediate checkpoints
  - no penalty if submission is within allowed grace days