File Organization

Binnur Kurt
binnur.kurt@ieee.org

Istanbul Technical University
Computer Engineering Department

Version 0.1.9

About the Lecturer

- **BSc**

  İTÜ, Computer Engineering Department, 1995

- **MSc**

  İTÜ, Computer Engineering Department, 1997

- Areas of Interest
  - Digital Image and Video Analysis and Processing
  - Real-Time Computer Vision Systems
  - Multimedia: Indexing and Retrieval
  - Software Engineering
  - OO Analysis and Design

Welcome to the Course

- **Important Course Information**
  - Office Hours
    - 14:00-15:00 Tuesday
  - Course Web Page
    - http://www.cs.itu.edu.tr/~kurt/courses/blg341
  - E-mail
    - kurt@ce.itu.edu.tr

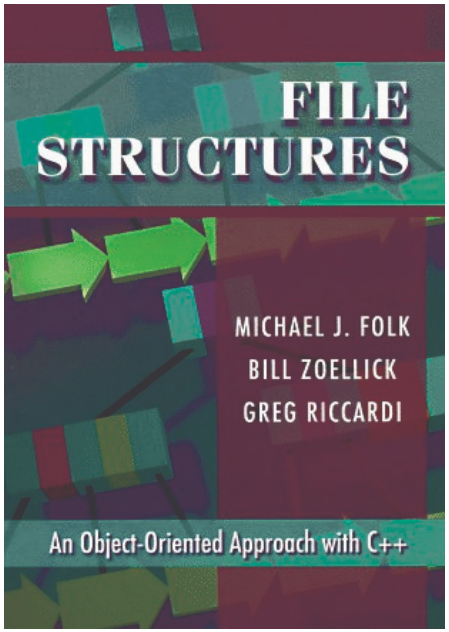
Grading Scheme

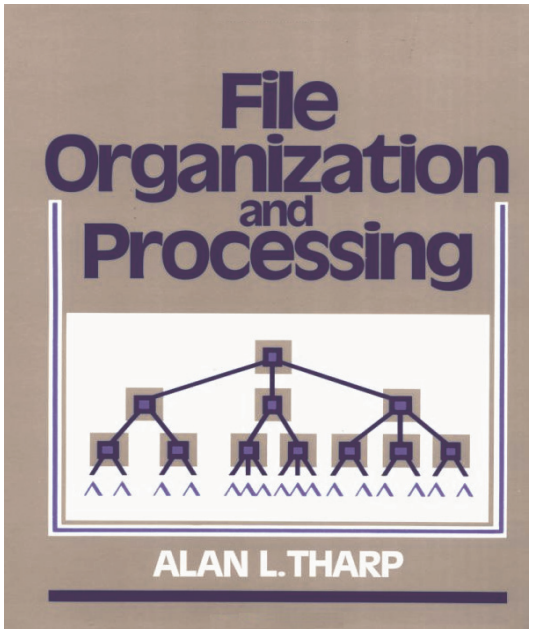
- 3 Projects (30%)
- A midterm exam (30%)
- A final exam (40%)
- You must follow the official Homework Guidelines (http://www.ce.itu.edu.tr/lisans/kilavuz.html).
- Academic dishonesty including but not limited to cheating, plagiarism, collaboration is unacceptable and subject to disciplinary actions. Any student found guilty will have grade F. Assignments are due in class on the due date. Late assignments will generally not be accepted. Any exception must be approved. Approved late assignments are subject to a grade penalty.

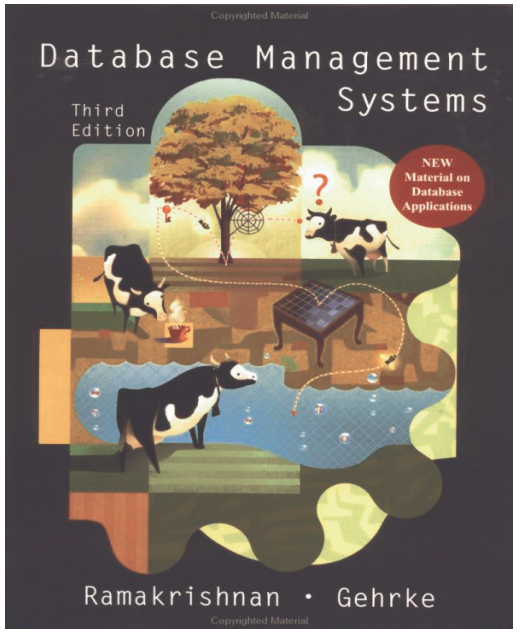
What we want to see in your programs

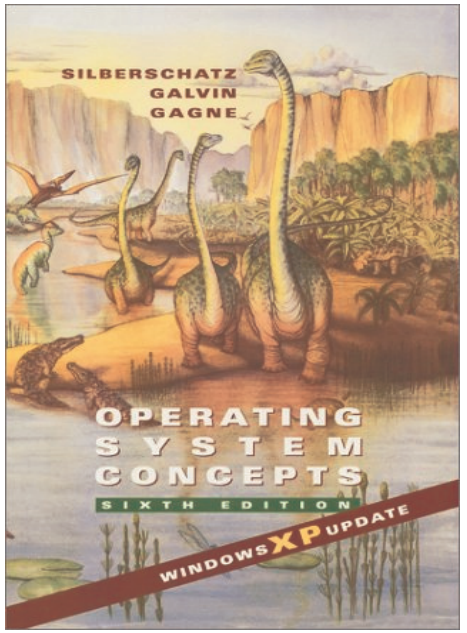
- All programs to be written in C/C++
- Self contained, well thought of, and well designed functions/classes
- Clean, well documented code, good programming style
- Modular design
- Do not write codes the way hackers do ☺

References









ITU Main Library

QA.76.73.C153.F65

QA.76.9.F5.T43

QA.76.76.O63.S55

This document is partially based on
http://www.site.uottawa.ca/~lucia/#Teaching

Tell me and I forget.
Show me and I remember.
Let me do and I understand.

—Chinese Proverb

Purpose of the Course

- Objective of Data Structures (BLG221) was to teach ways of efficiently organizing and manipulating data in *main memory*.
- In BLG341E, you will learn equivalent techniques for organization and manipulation of data in *secondary storage*.
- In the first part of the course, you will learn about "low level" aspects of file manipulation (basic file operations, secondary storage devices and system software).
- In the second part of the course, you will learn the most important high-level file structure tools (indexing, co-sequential processing, B trees, Hashing, etc).
- You will apply these concepts in the design of C programs for solving various file management problems

Course Outline

1. Introduction to file management.
2. Fundamental File Processing Operations.
3. Managing Files of Records: Sequential and direct access.
4. Secondary Storage, physical storage devices: disks, tapes and CD-ROM.
5. System software: I/O system, file system, buffering.
6. File compression: Huffman and Lempel-Ziv codes.
7. Reclaiming space in files: Internal sorting, binary searching, keysorting.
8. Introduction to Indexing.
9. Indexing

Course Outline

10. Cosequential processing and external sorting

11.Multilevel indexing and B trees

12.Indexed sequential files and B+ trees

13.Hashing

14.Extendible hashing

1 Introduction to File Management

Content

- Introduction to file structures
- History of file structure design

Introduction to File Organization

- Data processing from a computer science perspective:
  - Storage of data
  - Organization of data
  - Access to data
- This will be built on your knowledge of

Data Structures

Data Structures *vs.* File Structures

- Both involve:

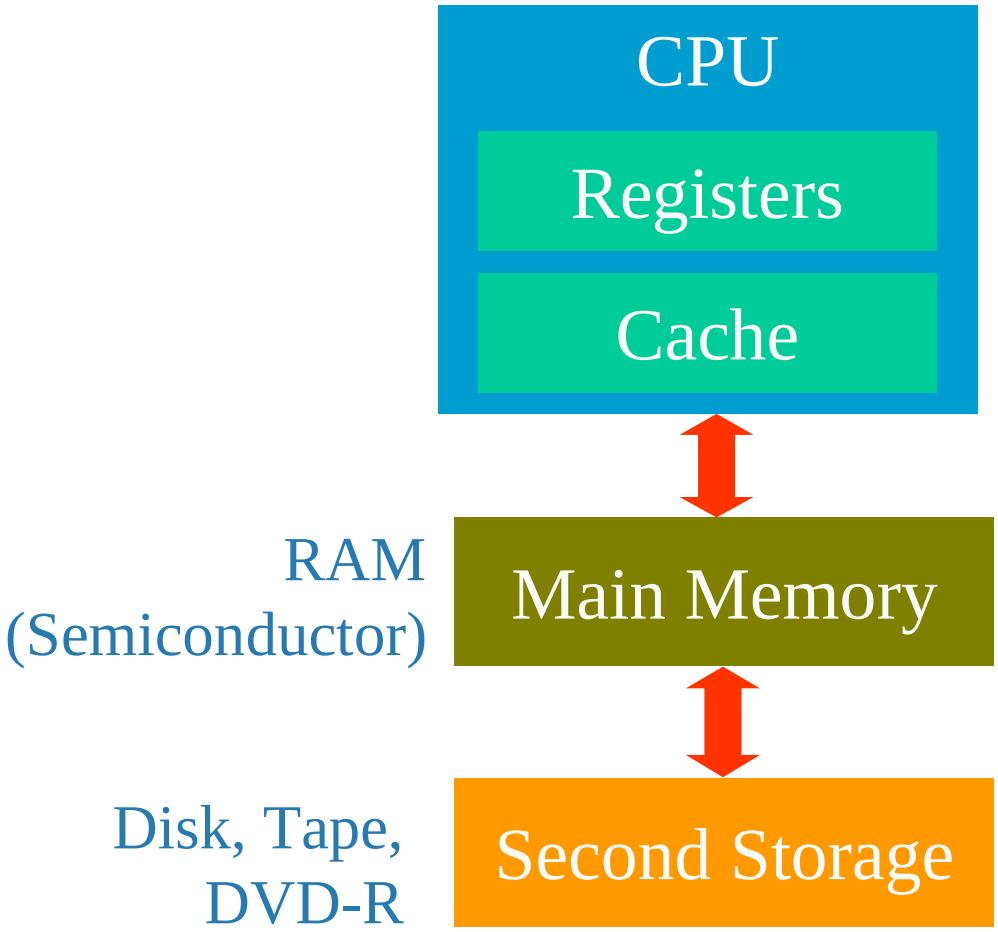
  Representation of Data

  +

  Operations for accessing data

- Difference:
  - Data Structures deal with data in **main memory**
  - File Structures deal with data in **secondary storage device** (File).

Computer Architecture

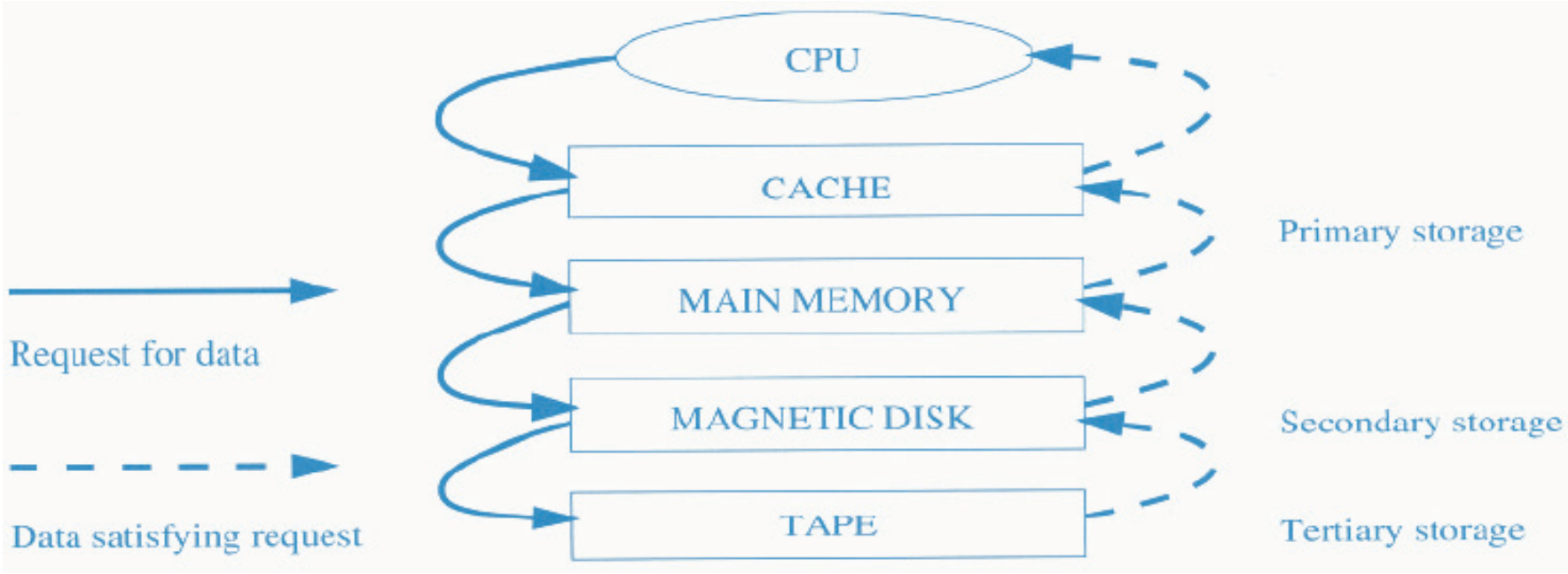


Differences

- Fast
- Small
- Expensive
- Volatile

- Slow
- Large
- Cheap
- Stable

Memory Hierarchy



Memory Hierarchy

- On systems with 32-bit addressing, only 2^{32} bytes can be directly referenced in main memory.
- The number of data objects may exceed this number!
- Data must be maintained across program executions. This requires storage devices that retain information when the computer is restarted.
  - We call such storage nonvolatile.
  - Primary storage is usually volatile, whereas secondary and tertiary storage are nonvolatile.

How Fast?

- Typical times for getting info
  - Main memory: ~120 nanoseconds = 120×10^{-9}
  - Magnetic Disks: ~30 milliseconds = 30×10^{-6}
- An analogy keeping same time proportion as above
  - Looking at the index of a book: 20 seconds

    versus
  - Going to the library: 58 days

Comparison

- Main Memory
  - Fast (since electronic)
  - Small (since expensive)
  - Volatile (information is lost when power failure occurs)
- Secondary Storage
  - Slow (since electronic and mechanical)
  - Large (since cheap)
  - Stable, persistent (information is preserved longer)

Goal of the Course

- Minimize number of trips to the disk in order to get desired information. Ideally get what we need in one disk access or get it with as few disk access as possible.
- Grouping related information so that we are likely to get everything we need with only one trip to the disk (e.g. name, address, phone number, account balance).

Locality of Reference in **Time** and **Space**

Good File Structure Design

- Fast access to great capacity
- Reduce the number of disk accesses
- By collecting data into buffers, blocks or buckets
- Manage growth by splitting these collections

History of File Structure Design

1. In the beginning… it was the tape
   - **Sequential access**
   - Access cost proportional to size of file

   [Analogy to sequential access to array data structure]
2. Disks became more common
   - **Direct access**

     [Analogy to access to position in array]
   - **Indexes** were invented
     - list of keys and points stored in small file
     - allows direct access to a large primary file

     Great if index fits into main memory.

     As file grows we have the same problem we had with a large primary file

History of File Structure Design

3. Tree structures emerged for main memory (1960`s)
   - **Binary search trees (BST`s)**
   - **Balanced**, self adjusting BST`s: e.g. AVL trees (1963)
4. A tree structure suitable for files was invented:

   B trees (1979) and **B+ trees**

   good for accessing millions of records with 3 or 4 disk accesses.
5. What about getting info with a single request?
   - Hashing Tables (Theory developed over 60's and 70's but still a research topic)

     good when files do not change too much in time.
   - Expandable, dynamic hashing (late 70's and 80's)

     one or two disk accesses even if file grows dramatically