1 Introduction to File Structures

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:

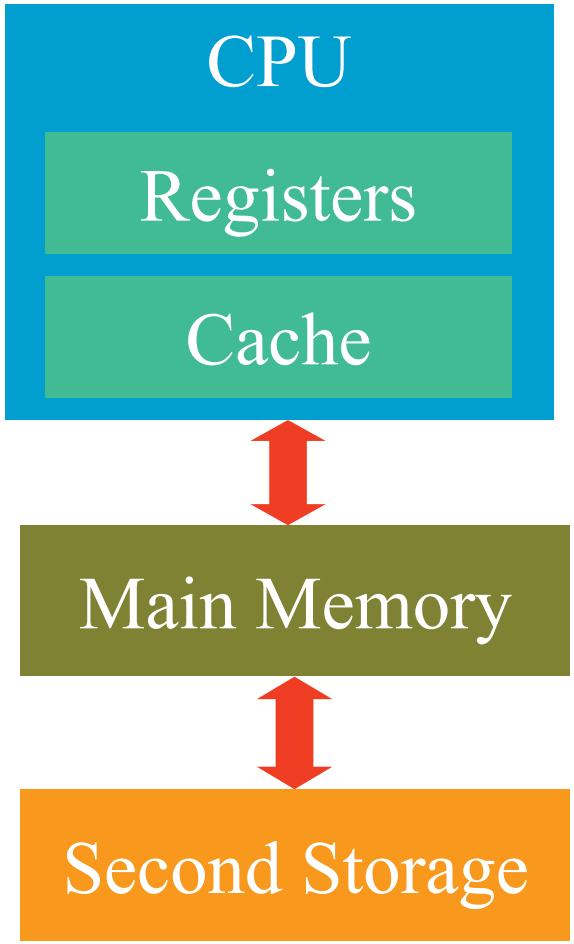
  Representation of Data

  +

  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

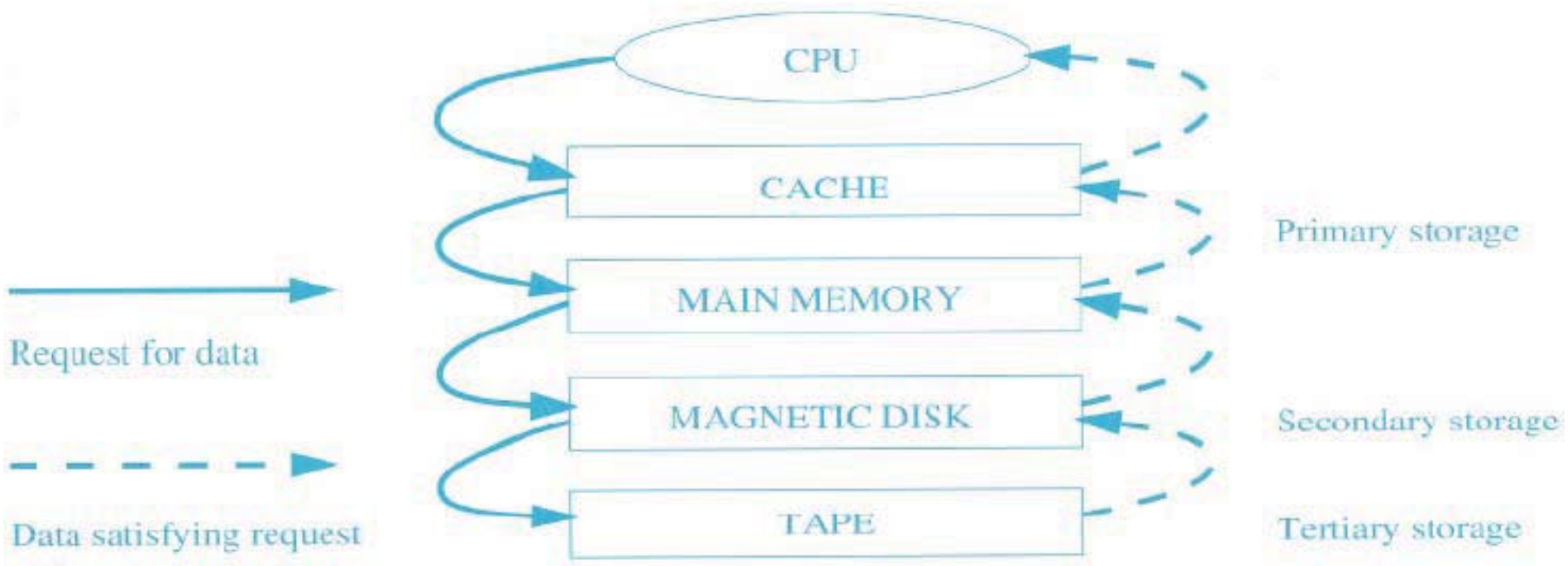


Differences

- 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.

Definition

- **File Structures** is the Organization of Data in Secondary Storage Device in such a way that minimize the access time and the storage space.
- A **File Structure** is a combination of ***representations*** for data in files and of ***operations*** for accessing the data.
- A **File Structure** allows applications to ***read***, ***write*** and ***modify*** data. It might also support ***finding*** the data that matches some search criteria or ***reading through*** the data in some particular order.

Why Study File Structure Design?
I. Data Storage

- Computer Data can be stored in three kinds of locations:
  - Primary Storage ==> Memory [Computer Memory]
  - Secondary Storage [Online Disk/ Tape/ CDRom that can be accessed by the computer]
  - Tertiary Storage ==> Archival Data [Offline Disk/Tape/ CDRom not directly available to the computer.]

Our Focus

II. Memory versus Secondary Storage

- Secondary storage such as disks can pack thousands of megabytes in a small physical location.
- Computer Memory (RAM) is limited.
- However, relative to Memory, access to secondary storage is extremely slow [E.g., getting information from slow RAM takes 120. 10^{-9} seconds (= 120 nanoseconds) while getting information from Disk takes 30. 10^{-3} seconds (= 30 milliseconds)]

III. How Can Secondary Storage Access Time be Improved?

By improving the File Structure.

Since the details of the representation of the data and the implementation of the operations determine the efficiency of the file structure for particular applications, improving these details can help improve secondary storage access time.

Overview of File Structure Design
I. General Goals

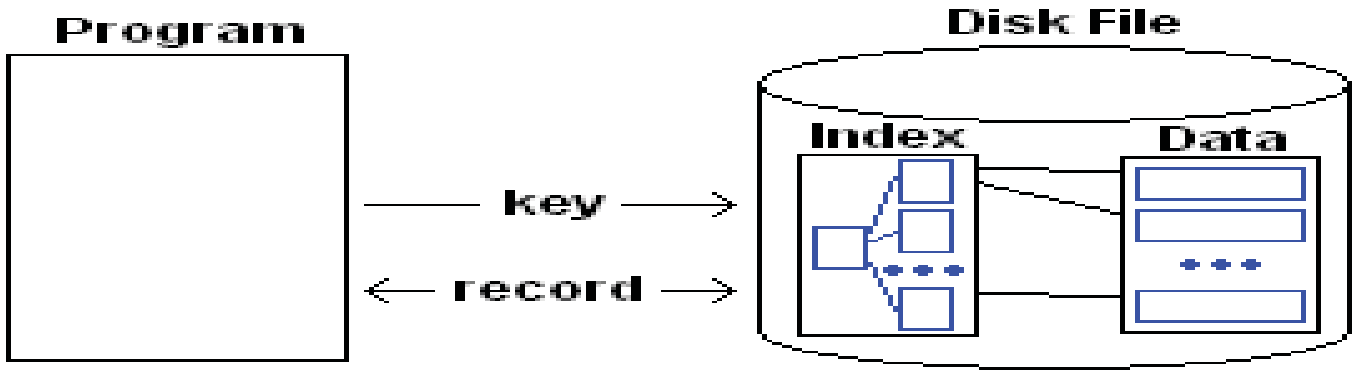
- Get the information we need with one access to the disk.
- If that's not possible, then get the information with as few accesses as possible.
- Group information so that we are likely to get everything we need with only one trip to the disk.

II. Fixed versus Dynamic Files

- It is relatively easy to come up with file structure designs that meet the general goals when the files never change.
- When files grow or shrink when information is added and deleted, it is much more difficult.

II. The emergence of Disks and Indexes

- As files grew very large, unaided sequential access was not a good solution.
- Disks allowed for ***direct*** access.
- Indexes made it possible to keep a list of ***keys*** and ***pointers*** in a small file that could be searched very quickly.
- With the key and pointer, the user had direct access to the large, primary file.



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