# IF2230
# Introduction to Operating Systems

Intro OS Sem 2 – 2022-2023

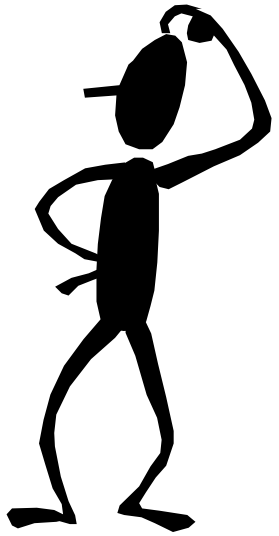
Achmad Imam Kistijantoro (imam@staff.stei.itb.ac.id)

Judhi Santoso (judhi@staff.stei.itb.ac.id)

Robithoh Annur (robithoh@stafff.stei.itb.ac.id)

# What is an operating system?

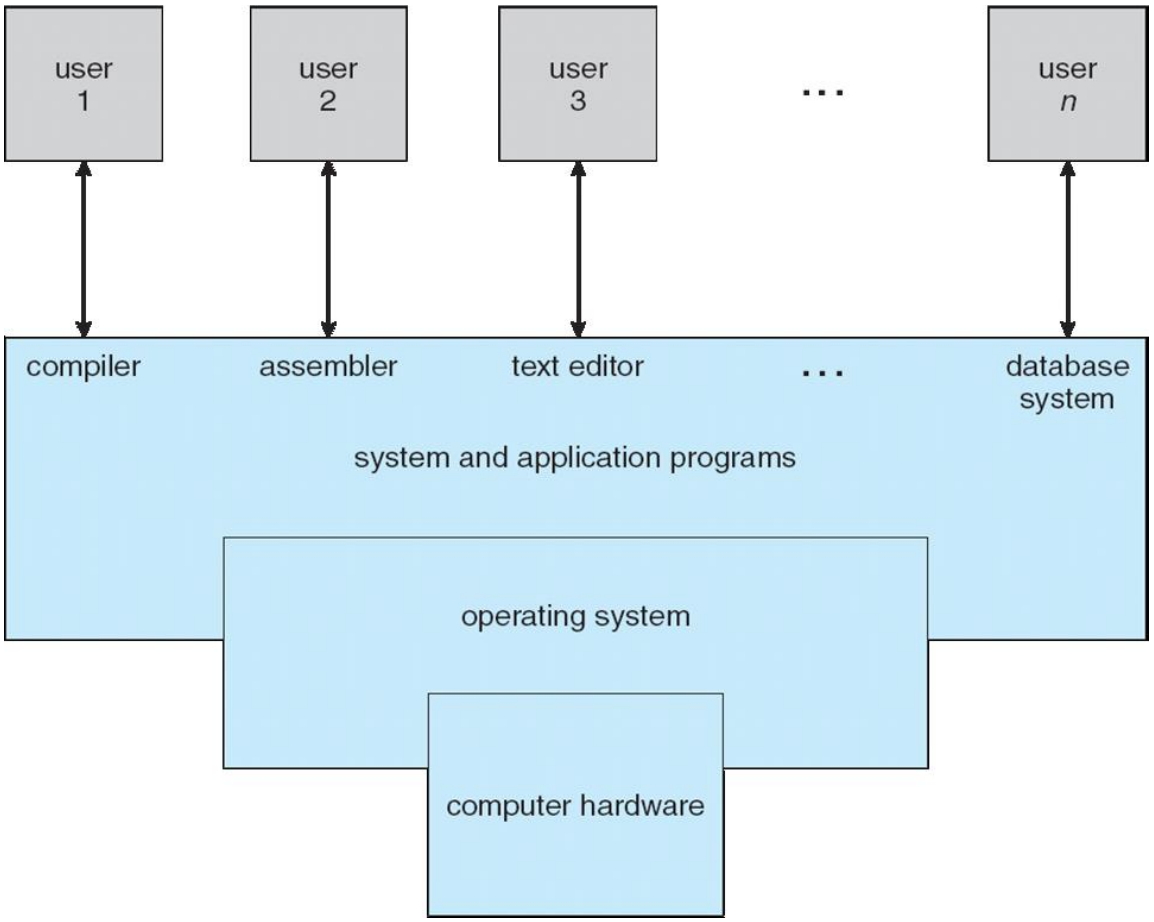
- What is an *Operating System*?
  - The software interface between hardware and its users
- Operating systems:
  - Execute user and system programs
  - Manage and coordinate computer hardware
  - Serve as resource allocators
  - Are typically interrupt-driven



# Computer System Structure

- Computer system can be divided into four components:
  - Hardware – provides basic computing resources
    - CPU, memory, I/O devices
  - Operating system
    - Controls and coordinates use of hardware among various applications and users
  - Application programs – define the ways in which the system resources are used to solve the computing problems of the users
    - Word processors, compilers, web browsers, database systems, video games
  - Users
    - People, machines, other computers

# Four Components of a Computer System



# What Operating Systems Do

- Depends on the point of view
- Users want convenience, **ease of use**
  - Don't care about **resource utilization**
- But shared computer such as **mainframe** or **minicomputer** must keep all users happy
- Users of dedicate systems such as **workstations** have dedicated resources but frequently use shared resources from **servers**
- Handheld computers are resource poor, optimized for usability and battery life
- Some computers have little or no user interface, such as embedded computers in devices and automobiles

# Operating System Definition

- OS is a **resource allocator**
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use
- OS is a **control program**
  - Controls execution of programs to prevent errors and improper use of the computer

# Operating System Definition (Cont.)

- No universally accepted definition
- "Everything a vendor ships when you order an operating system" is good approximation
  - But varies wildly
- "The one program running at all times on the computer" is the **kernel**. Everything else is either a system program (ships with the operating system) or an application program.

# System software

- What is system software?
  - Computer programs that directly *control* the operations of the computer and its devices
- Operating systems:
  - Coordinate and orchestrate all activities of the hardware devices in a computer
  - Provide both a *Graphical User Interface* (*GUI*) and a *Command-Line Interface* (*CLI*) for its users

# Operating system design goals

- From a user's perspective:
  - easy to use
  - easy to learn
  - reliable
  - safe
  - fast
  - etc.

- System goals:
  - reliability
  - flexibility
  - extensibility
  - speed(y)
  - efficiency
  - maintainability
  - etc.

# Operating system services (i)

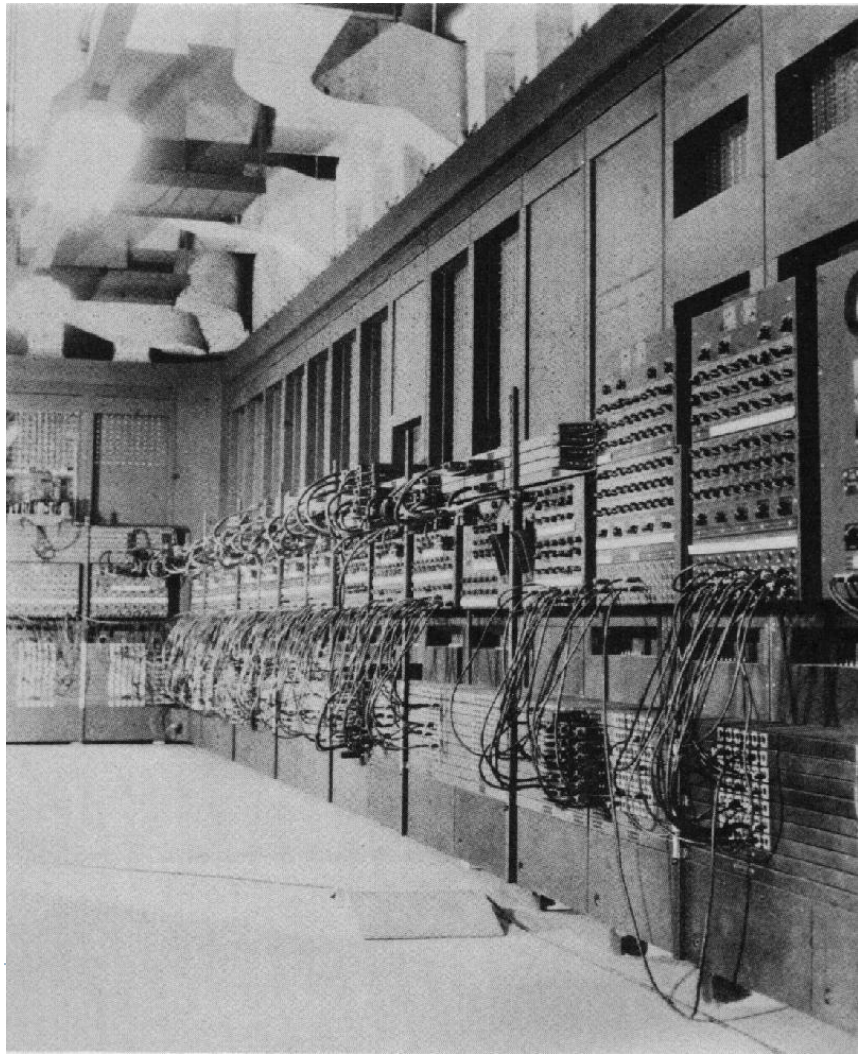
- An operating system provides *services*:
  - Program execution
    - Load programs into memory, run/suspend/halt programs, handle/display errors
  - I/O operations
    - Seamlessly interact with I/O devices, including disks, networks connection, etc.
  - Filesystem manipulation
    - Read/write/traverse filesystem directories, read/write files, enforce permissions, search for files

# Operating system services (ii)

- Other operating system *services*:
  - Inter-Process Communications (IPC)
    - Processes exchange information via shared memory, message passing, sockets, pipes, files, etc.
    - Often spans multiple computers and networks
  - Error detection and recovery
    - Detect errors in CPU, memory, I/O devices, processes, network connections, etc.
    - Recover from errors gracefully, ensuring correct and consistent operations

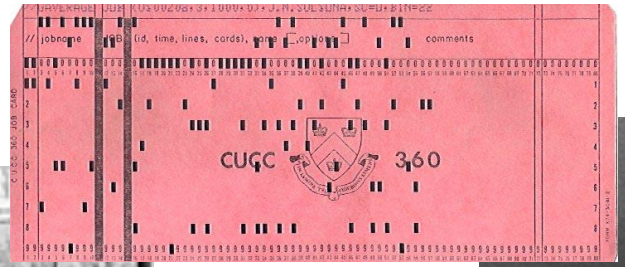
# History of operating systems (i)

- In the beginning...
  ...the 1940s



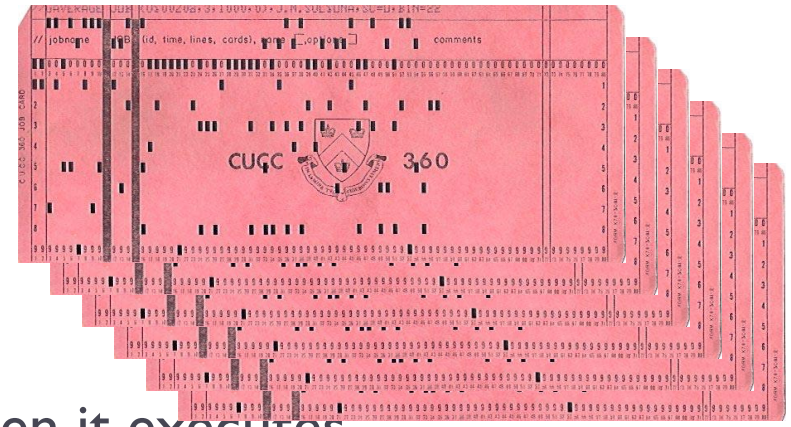
# History of operating systems (ii)

- Automation in the 1950s with punch cards

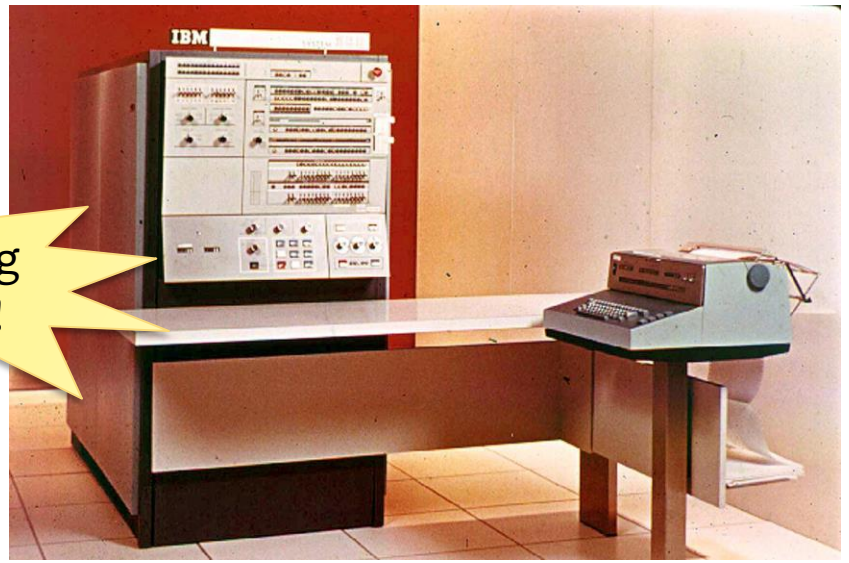

# Batch jobs

- A *job* is a unit of work submitted by a user to the operating system
- Jobs typically consist of:
  - a program either in a source language or in "executable" binary form
  - input data used by the program when it executes




# History of operating systems (iii)

- IBM 360 introduced (in 1964)

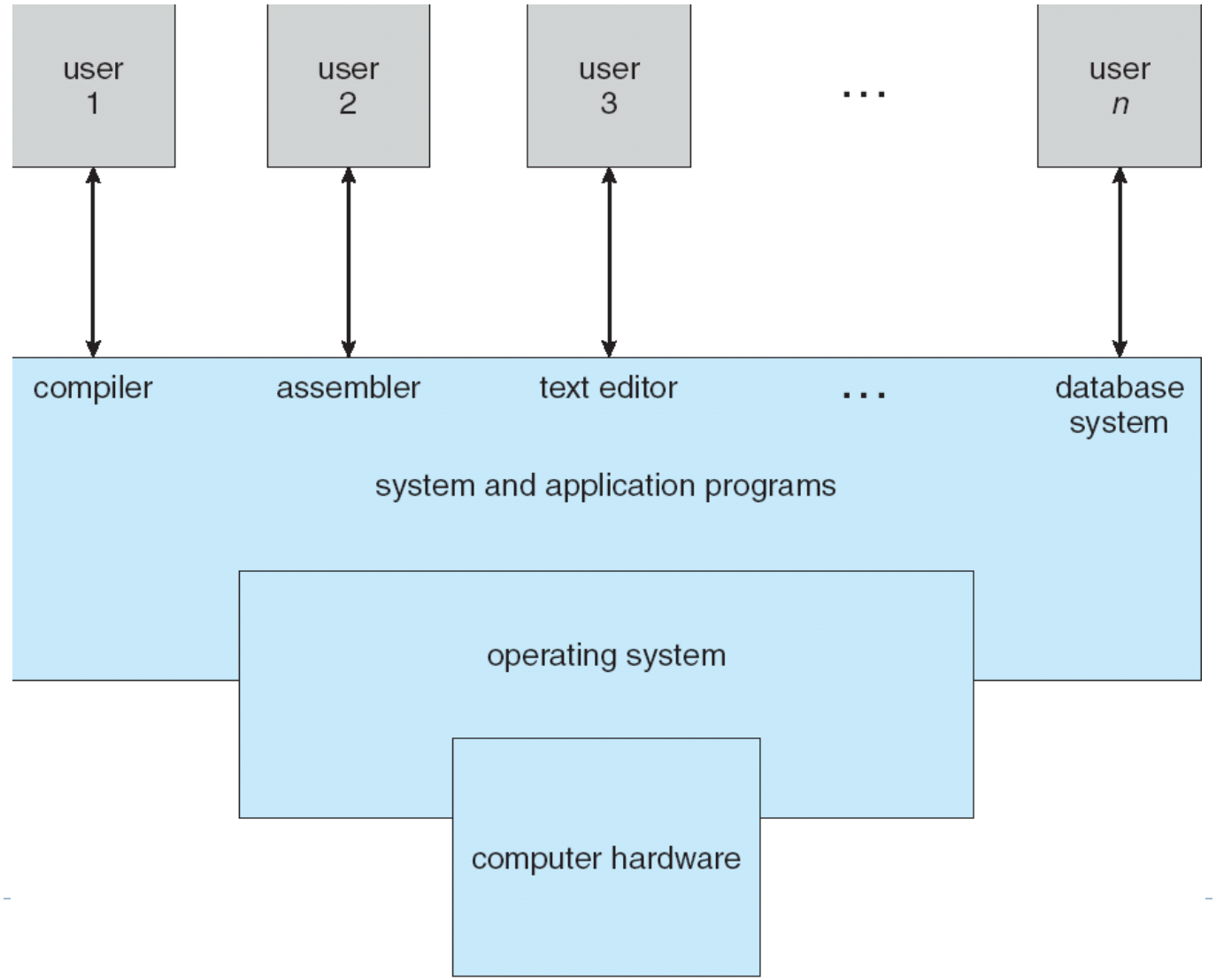



a computing revolution!



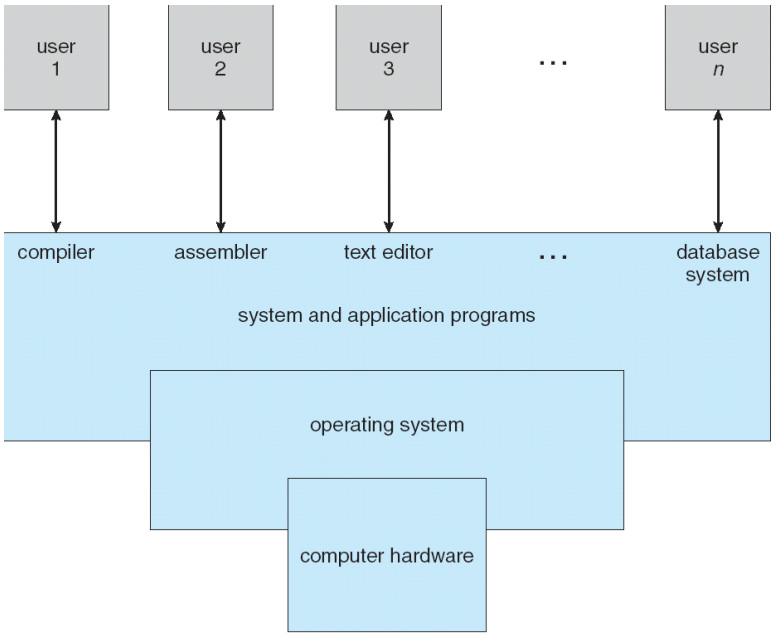

what's so revolutionary?

# Multiprogramming (i)



# Multiprogramming (ii)

- In multiprogramming, several jobs reside in memory simultaneously
  - CPU use is shared and managed by the operating system

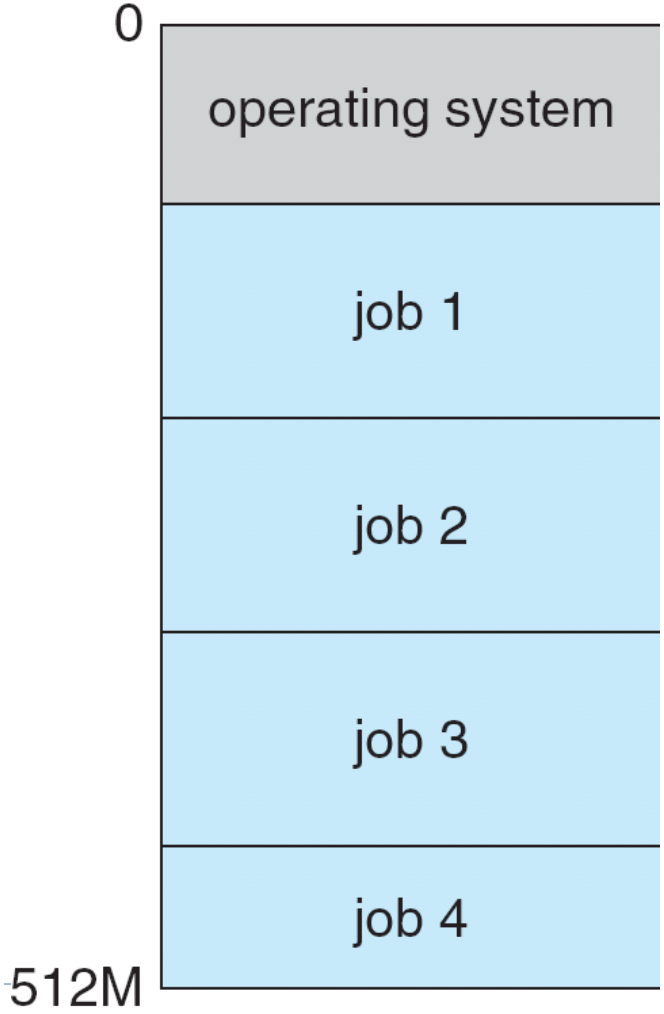



# Multiprogramming (iii)

- Multiprogramming provides efficient use of the computer (CPU) and its resources (I/O)
  - One user cannot keep the CPU and I/O devices busy at all times
  - Multiprogramming organizes jobs such that the CPU always has exactly one to execute

# Multiprogramming (iv)

- Computer is often idle – why?
  - CPU and hardware *significantly* faster than I/O
  - When a user or process is blocked waiting for I/O, the operating system switches to another job
  - A subset of jobs is stored in memory, awaiting CPU or I/O

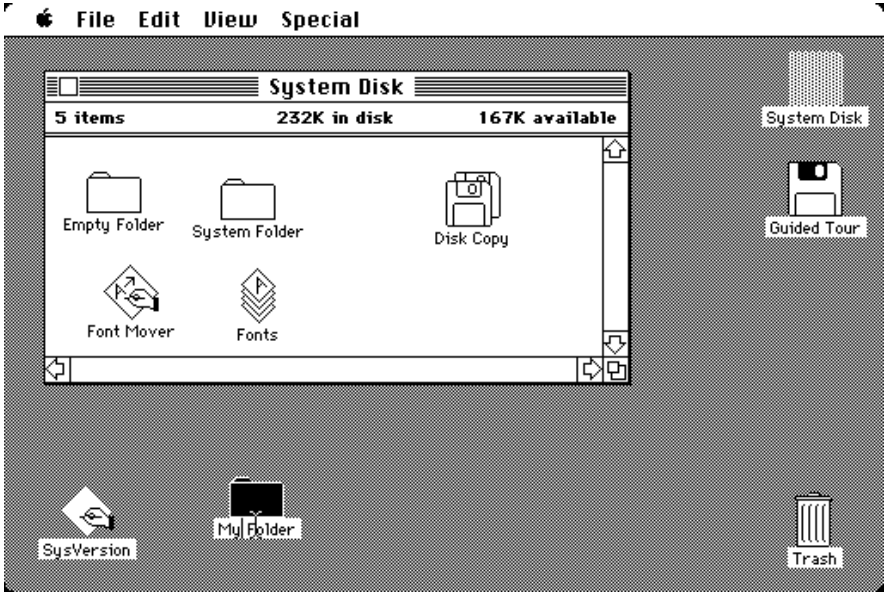



# Timesharing and multitasking

- To ensure fairness, use a *timesharing scheme* in which the CPU cycles through all jobs
  - Each job is given a fixed amount of CPU time
  - Switching from one running job (or process) to another is called a *context switch*
  - A process may relinquish its time if blocked on an I/O request

# History of operating systems (iv)

- Text CRTs (1970s) to an early Mac (1984)


# History of operating systems (v)

- Personal computer revolution (1970s/80s)

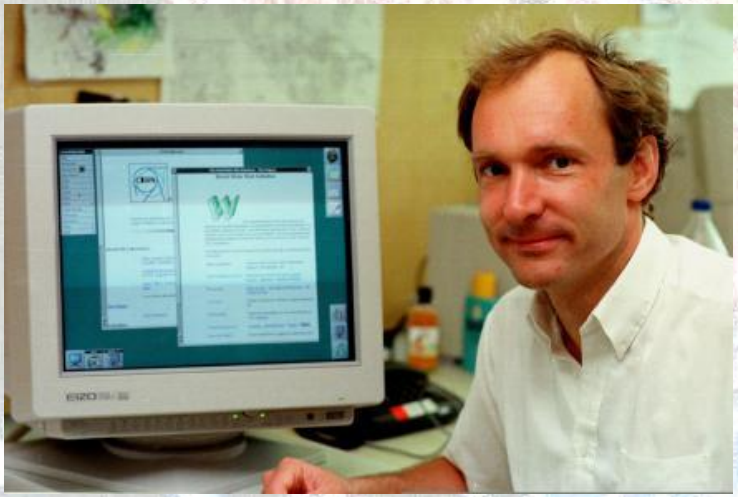
# History of operating systems (vi)

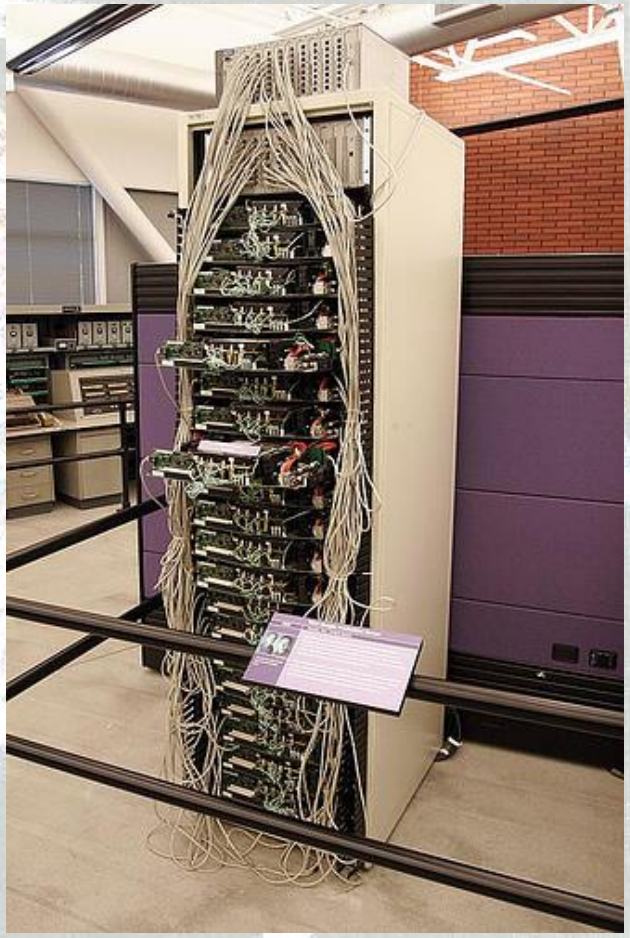
- The war begins...

sell your stock....

# History of operating systems (vii)

- World Wide Web and Internet revolution (1990s/2000s)

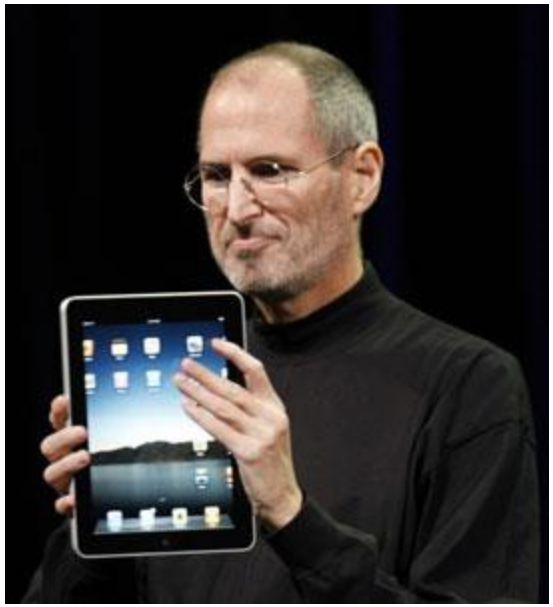




# History of operating systems (viii)

- Mobile revolution (2010s)



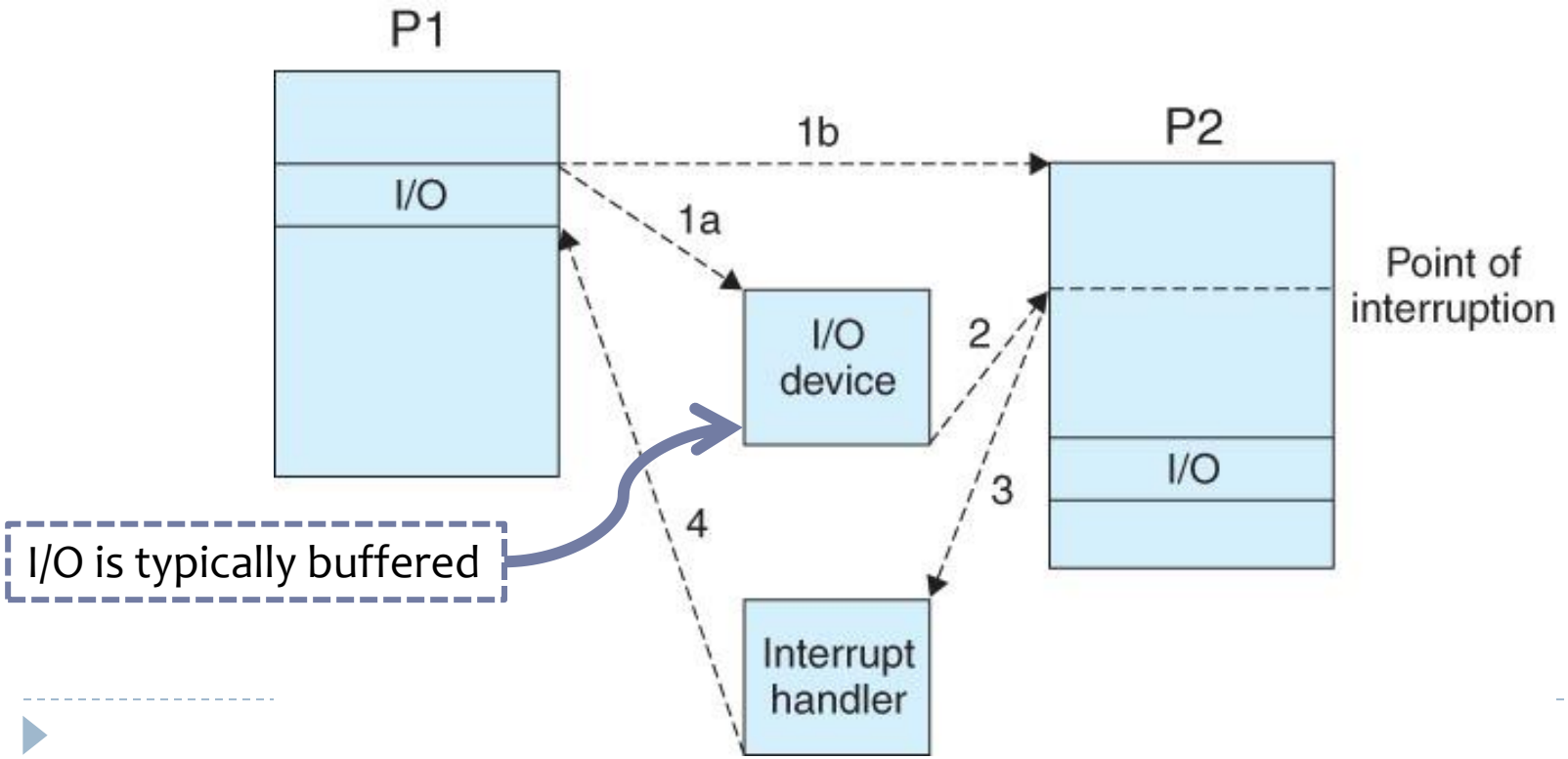


# We interrupt this program…

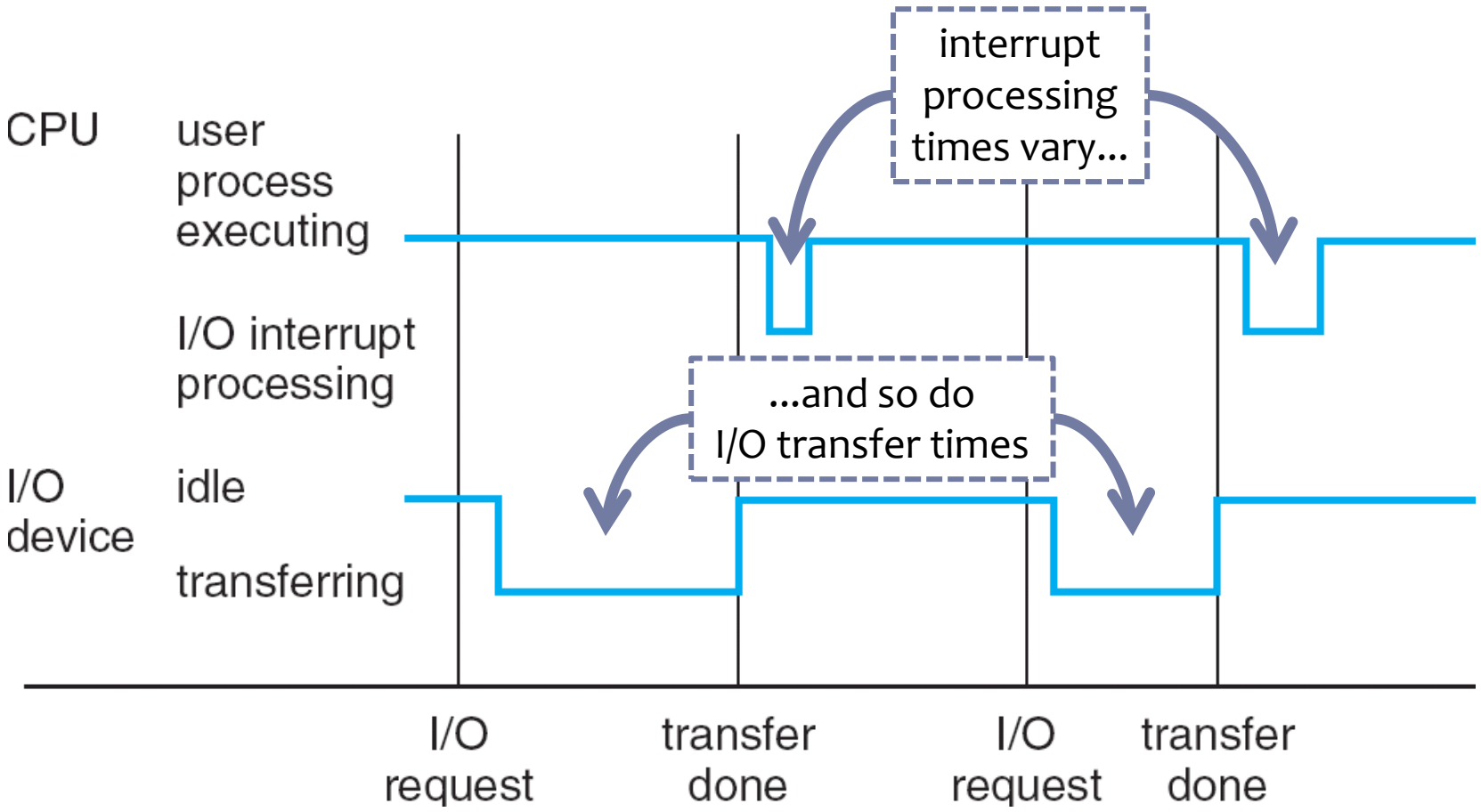
- Software instructions are executed by a Central Processing Unit (CPU)
  - An external hardware event triggers an interrupt by signaling the CPU
    - e.g. mouse movement, keyboard event
  - Software triggers an interrupt by executing a system call
    - e.g. disk read, graphics output, printer output

# Interrupt mechanism

- Interrupts are handled much like calling a function in a programming language

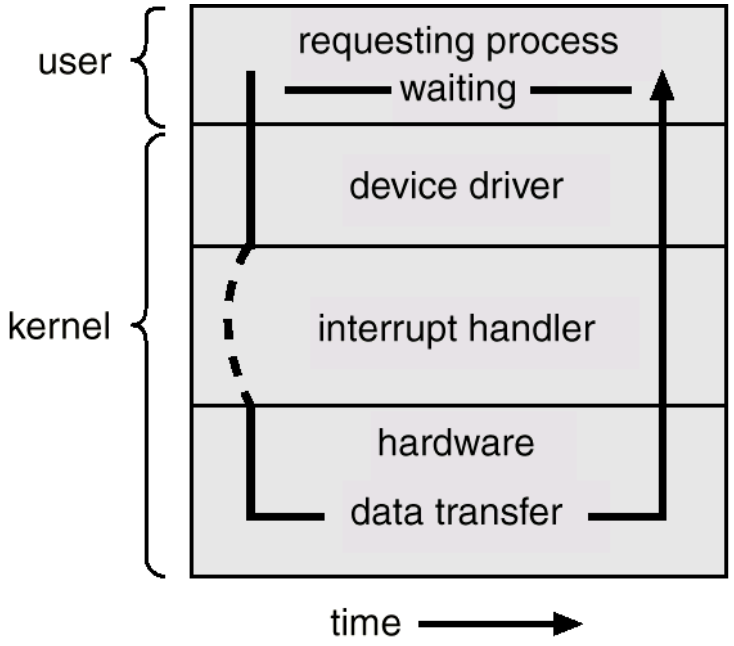


# Typical interrupt timeline

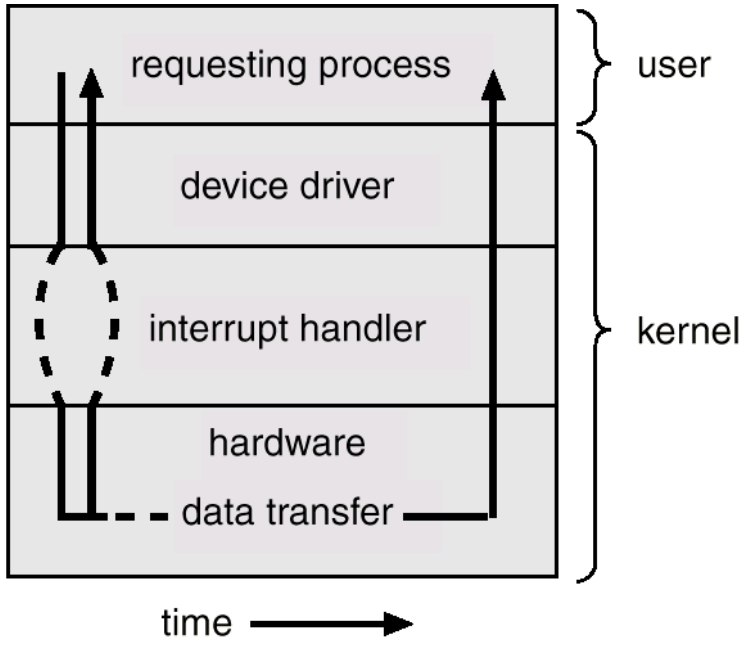


# Synchronous and Asynchronous I/O

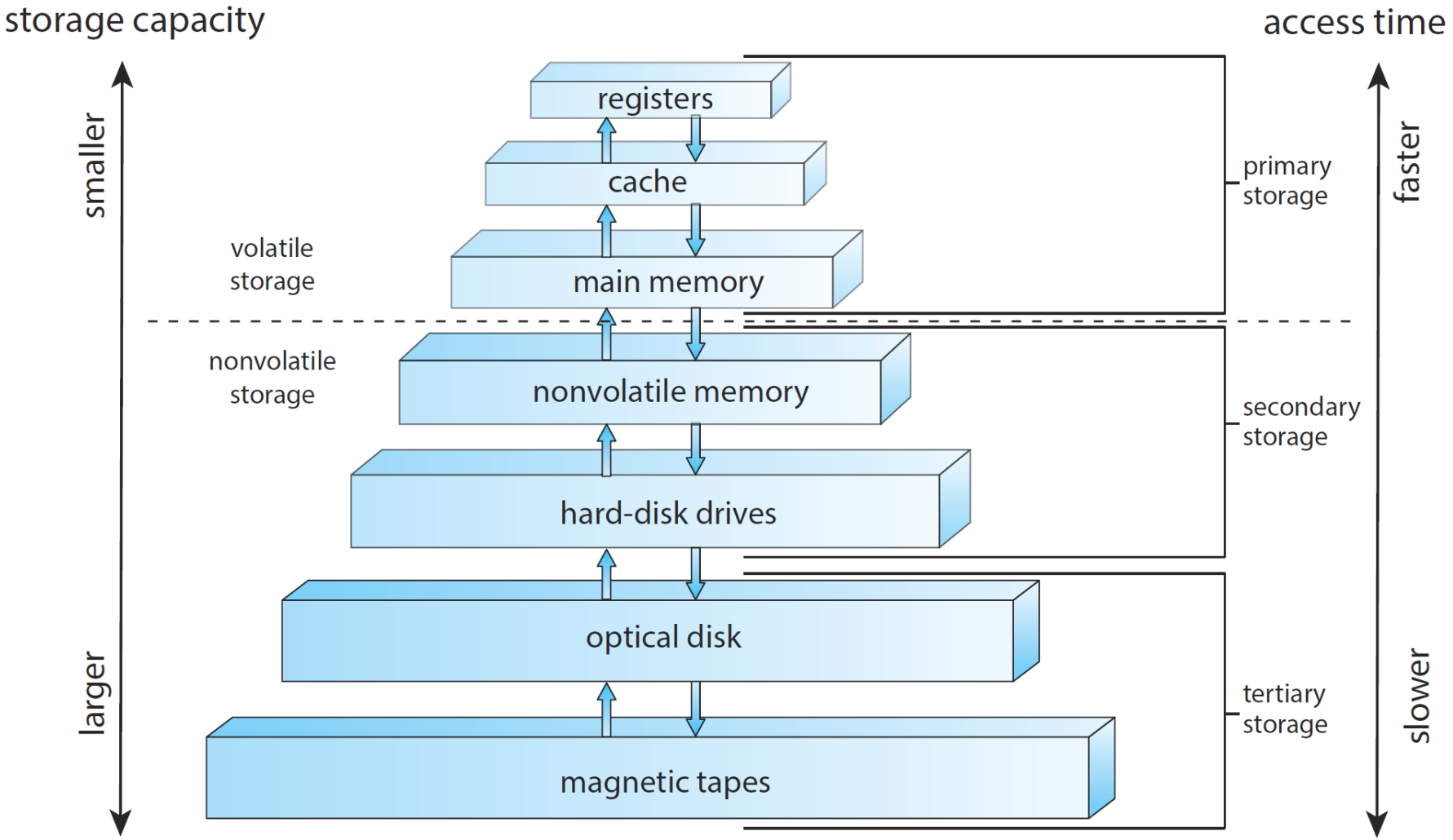
Synchronous




Asynchronous

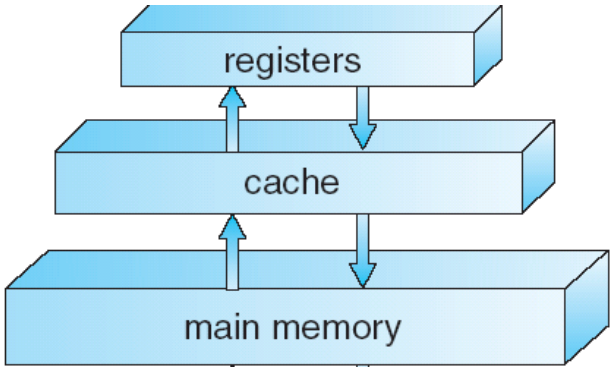


# Hierarchical storage architecture



# Caching (i)

- Caching is a technique in which data is temporarily stored in a smaller and faster memory component
  - Why implement caching in an operating




# Caching (ii)

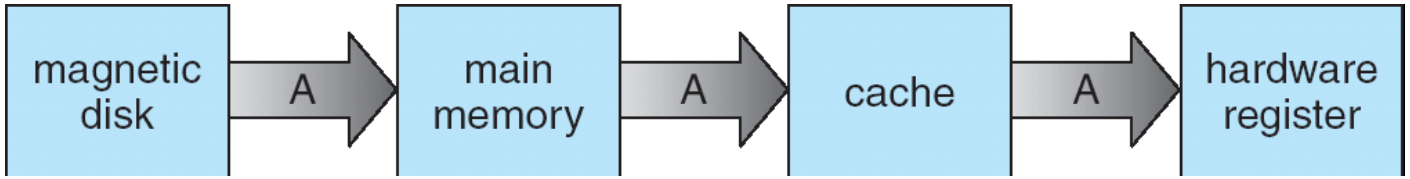
- A key goal in operating system design is achieving *fast and efficient performance*

| Level | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Name | registers | cache | main memory | disk storage |
| Typical size | < 1 KB | > 16 MB | > 16 GB | > 100 GB |
| Implementation technology | custom memory with multiple ports, CMOS | on-chip or off-chip CMOS SRAM | CMOS DRAM | magnetic disk |
| Access time (ns) | 0.25 – 0.5 | 0.5 – 25 | 80 – 250 | 5,000.000 |
| Bandwidth (MB/sec) | 20,000 – 100,000 | 5000 – 10,000 | 1000 – 5000 | 20 – 150 |
| Managed by | compiler | hardware | operating system | operating system |
| Backed by | cache | main memory | disk | CD or tape |

# Caching (iii)

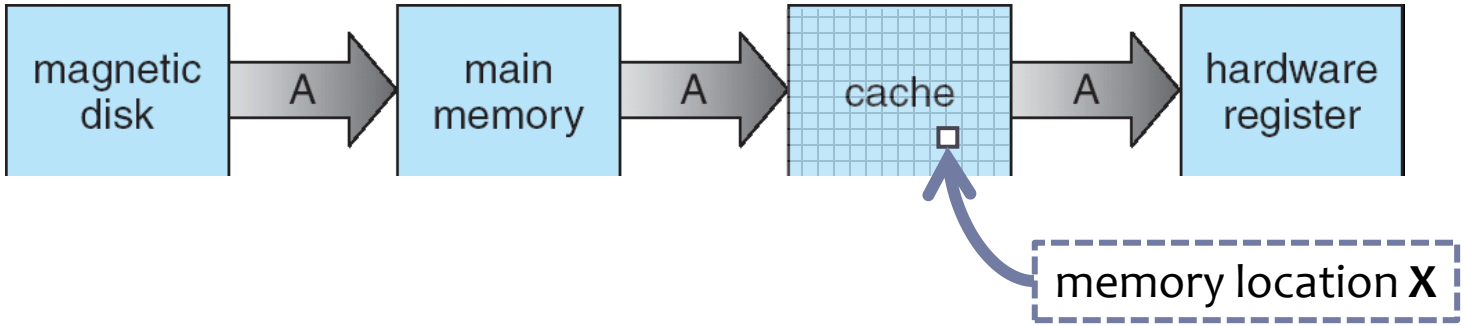
- What's the caching algorithm?
  - When the operating system attempts to read from memory, check to see if the requested data is already in the cache
  - If it is, data is read from the cache (*fast*!)
  - If not, data is copied from memory to the cache

(*maybe next time*...)




# Principle of locality

- When a running program reads from memory location **X**, the *principle of locality* predicts that the next memory location requested will be near **X**




- Store *pages* of data in a cache, where each page is typically the same size (e.g. 64KB)