1 Introduction

Computer systems have two major components:

- *hardware*–electronic,mechanical, optical devices
- *software*–programs.

Without support software, a computer is of little use. With its *software*, however, a computer can store, manipulate, and retrieve information, and can engage in many other activities. Software can be grouped into the following categories:

- *systems software* (operating system & utilities)
- *applications software* (user programs)

Summary,

- Hardware provides basic computing resources (CPU, memory, I/O devices).
- Operating system controls and coordinates the use of the hardware among the various application programs for the various users.
- Application programs define the ways in which the system resources are used to solve the computing problems of the users (compilers, database systems, video games, business programs).
- Users (people, machines, other computers).

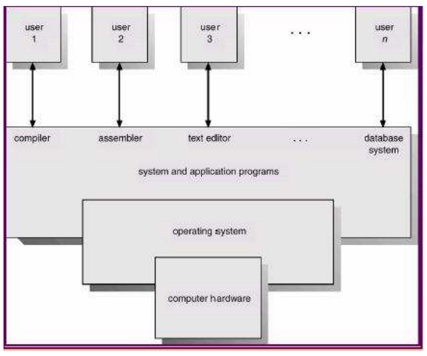


Figure 1: Abstract view

1.1 What is Operating System?

A program that acts as an intermediary between a user of a computer and the computer hardware.

- An operating system (OS) is everything in system that isn't an application or hardware
- An OS provides orderly and controlled allocation and use (i.e., *sharing, optimization of resource utilization*) of the resources (Processor, Memory, I/O devices) by the users (jobs) that compete for them.
- Support programs (typically called daemons) running in the machine that handle higher level services such as mail transport (networking), off-line file system checking (system robustness), web serving (server work), etc.
- Protection and Security
- Provides an abstraction layer over the concrete hardware. Use the computer hardware in an efficient manner (converting hardware into useful form;) "hide" the complexity of the underlying hardware and give the *user* a better view (an abstraction) of the computer for applications by providing:
  - Standard library
    - allow applications to reuse common facilities
    - make different devices look the same
    - provide higher-level abstractions
    - What are the right abstractions *Challenge.*
  - Resource manager, Resource - "Something valuable" e.g. CPU, memory (RAM), I/O devices (disk). Each program gets **time** with the resource and each program gets **space** on the resource
    - Multiple users/applications can share, why share: (1) devices are expensive, and (2) there is need to share data as well as communicate
    - Protect applications from one another
    - Provide fair and efficient access to resources
    - OS cannot please all the people all the time, but it should please most of the people most of the time, so: What mechanisms? What policies? (e.g.,. which user/process should get priority for printing on a common shared printer?); *Challenges*

1.1.1 Functionalities in OS:

Desired functionalities of OS depend on outside factors like users' & application's "Expectations" and "Technology changes" in Computer Architecture (hardware).
OS must adapt:

- Change abstractions provided to users
- Change algorithms to change these abstractions
- Change low-level implementation to deal with hardware

The current operating systems are driven by such evolutions.

1.2 History of operating Systems

The earliest computers, developed in the 1940s, were programmed in *machine language* and they used front panel switches for input. The programmer was also the operator interacting with the computer directly from the system console (control panel).

- programmers needed to sign-up in advance to use the computer one at a time
- executing a single program (often called a *job*) required substantial time to setup the computer.
- First generation 1945 - 1955, vacuum tubes, plug boards
- Second generation 1955 - 1965, transistors, batch systems
- Third generation 1965 - 1980, ICs and multiprogramming
- Fourth generation 1980 - present, personal computers
- Next generation ??, personal digital assistants (PDA), information appliances

Two distinct phases in history: Expensive computers, Cheap computers

1.2.1 Mainframe Systems

First commercial systems: Enormous, expensive and slow, I/O: Punch cards and line printers.

- Single operator/programmer/user runs and debugs interactively:
  - Standard library with no resource coordination
  - Monitor that is always resident
    - initial control in monitor
    - control transfers to job
    - when job completes control transfers back to monitor
- Inefficient use of hardware: poor *throughput* and poor *utilization*
- *Performance metrics:*
  Throughput: like amount of useful work done per hour
  Utilization: keeping all devices busy
- Mainframe systems started to appear after world war 2.
- They initially executed one program at a time and were known as batch systems.

1.2.2 Batch and Multiprogrammed Systems

Group if jobs submitted to machine together, *Batch.* A job was originally presented to the machine (and its human operator) in the form of a set of cards – these cards held information according to how "punched" out of the cardboard. The operator grouped all of the jobs into various batches with similar characteristics before running them (all the quick jobs might run, then the slower ones, etc.).

- Operator collects job, orders efficiently, runs one at a time
- Amortize setup costs over many jobs
- Keep machine busy while programmer thinks
- User must wait for results until batch collected and submitted

Result: Improved system throughput and utilization, but lost interactivity. Since the I/O used slow mechanical devices, the CPU was often idle waiting on a card to be read or some result to be printed, etc. One way to minimize this inefficiency was to have a number of jobs available and to switch to running another one to avoid idleness.

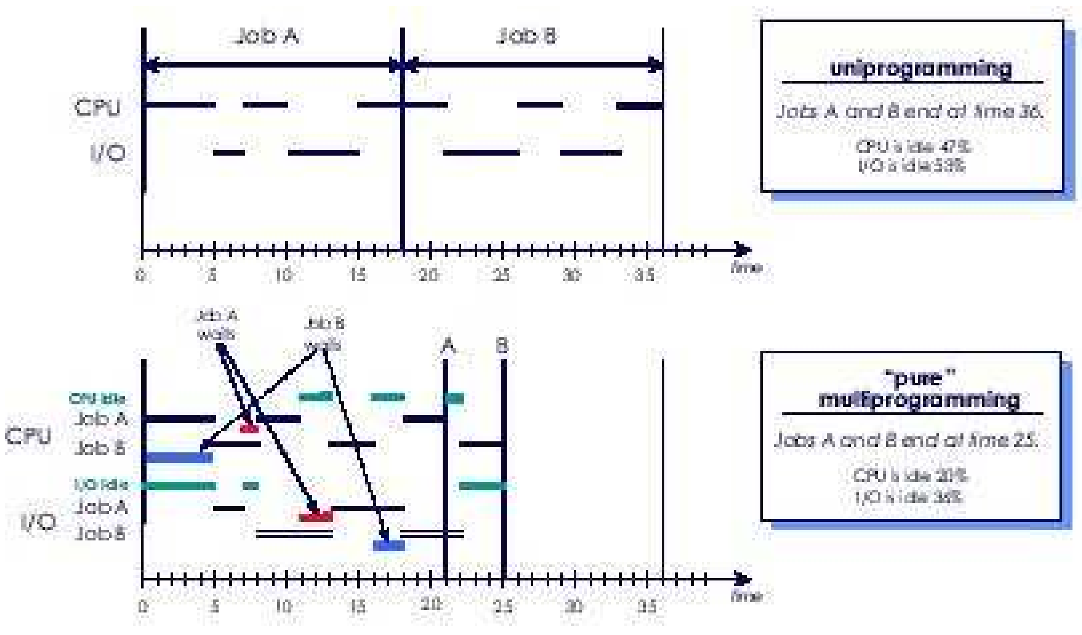


Figure 2: Job Interleaving

- Mechanical I/O devices much slower than CPU
- Overlap I/O with execution by providing pool of ready jobs
- New OS Functionality evolved: Buffering, Direct Memory Access (DMA), interrupt handling

Result: Improves throughput and utilization.
In multiprogrammed systems, a number of programs were resident in memory and the CPU could choose which one to run. One way to choose is to just keep executing the current program until an I/O delay is pending – instead of just waiting, the CPU would move onto the next program ready to be run.

- Keep multiple jobs resident in memory
- OS chooses which job to run
- When job waits for I/O switch to another resident job

Result: Job scheduling policies, Memory management and protection, improves throughput and utilization, still not interactive.

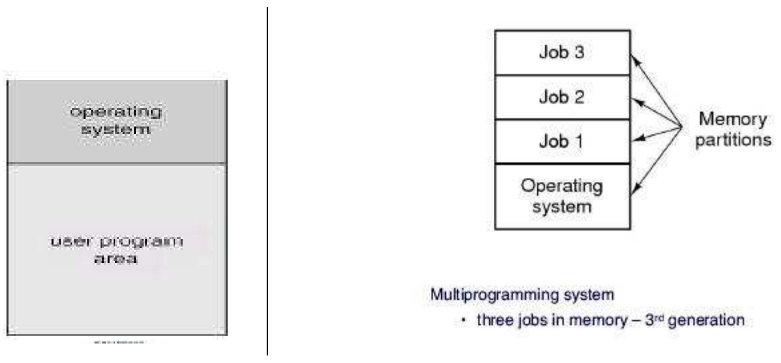


Figure 3: Memory Layout; Simple Batch, Multi Programming

1.2.3 Time Sharing

While multiprogrammed systems used resources more efficiently i.e. minimized CPU idle time, a user could not interact with a program. By having the CPU switch between jobs at relatively short intervals, we can obtain an interactive system.That is, a system in which a number of users are sharing the CPU (or other critical resource) with a timing interval small enough not to be noticed e.g. no more than 1 second. We say that a time-sharing system uses CPU scheduling and multiprogramming to provide each user with a small portion of a time-shared computer.

- Switch between jobs so frequently that get appearance of dedicated machines for each user/process.
- New OS Functionality: More complex job scheduling, memory management, concurrency control and synchronization
- Users easily submit jobs and get immediate feedbacks

OS Features Needed for Multiprogramming:

- I/O routines supplied by system
- Memory management system must allocate memory to several jobs
- CPU scheduling system must choose among several jobs ready to run
- Allocation of devices

Time-Sharing Systems Interactive Computing:

- The CPU multiplexed among several jobs in memory and on disk (CPU allocated only to jobs in memory). The CPU switches to the next job that can be run whenever the current job enters a wait state or after the current job has used a standard unit of time.When viewed over a relatively long time frame, we obtain the appearance that the CPU is simultaneously running multiple programs.
- Job swapped in and out of memory to disk. If the time-sharing computer does not have enough semiconductor memory installed to hold all of the desired programs, then a backing store must be used to temporarily hold the contents relating to some programs when other programs are present in semiconductor memory. In effect, we are now "memory sharing" between competing users (programs). This idea leads to a mechanism called virtual memory.
- On-line communication between user and system provided; when OS finishes execution of a command, it awaits next "control statement" from user.

The sensible sharing of resources such as CPU time and memory must be handled by the operating system, which is just another program running on the computer. For this control program to always be in control, we require that it never be blocked from running. The operating system, which might in fact be organized like a small number of cooperating programs, will lock itself into memory and then control CPU allocation priority in order that it never be blocked from running.

1.2.4 Personnel Computers

Single-user, dedicated.

- I/O devices keyboards, mice, screens, printers
- User convenience and responsiveness
- Can adopt technology developed for larger system
- Previously thought,
  - individuals have sole use of computer, do not need advanced CPU utilization, protection features
  - still true? See next list of operating systems...

- May run several different types of OS (Windows, Mac OS X, UNIX, Linux)
- Operating systems such as FreeBSD, NetBSD, Mac OS-X and Linux offer multitasking and virtual memory on PC hardware.
- The *BSD world has influenced the world of computing through networking advances, virtual file storage systems, dynamically self-optimizing resource allocation schemes, etc.
- While all *BSD systems have a family history derived from the original non-networking UNIX, Linux is mostly a "work-alike" re-implementation and can be traced back to MINIX which was developed by Tanenbaum for operating system teaching.

1.2.5 Parallel Processing Systems

Traditional multiprocessor system (share a common bus, clock, and memory), tightly-coupled; multiprocessing. The desire for increased throughput has led to system designs in which multiple streams of processing occurs in parallel.

- *Tightly coupled system* processors share memory and a clock; communication usually takes place through the shared memory. For a system with **n** processors that is to run **n** or more separate programs, the speedup may approach **n**. It will not reach **n** because there will be some contention for access to shared elements such as the memory system.
- Advantages of parallelism:
  - higher throughput and better fault tolerance
  - Economical (?)
  - Increased availability ($\neq$ reliability)
- *Symmetric multiprocessing (SMP)*, a symmetric multi-processor system shares the execution of the operating system amongst all of the processors – it is usually multi-threaded and contains no block structures. The CPUs are equal i.e. we say that they are all peers.
  - Each processor runs identical copy of OS
  - Many processes at once without performance loss
  - Most modern operating systems support SMP

- *Asymmetric multiprocessing*, an asymmetric multi-processor system contains a single CPU called the master that carefully controls access to single threaded sections of the kernel – this processor controls the activities of the slave processors. Asymmetric MP systems are less efficient.
  - Each processor is assigned specific task; master processor schedules, allocates work to slave processors.
  - Mostly for specialized high-end computation
- What can we say about an n-processor system that has m < n application programs to run? Unless some of the application programs can support multiple threadsÂ of simultaneous execution, then the speedup may only approach m (since n - m processors are idle).
- As programmers, you will learn about thread models (and support libraries) that allow you to develop multi-threaded applications. Important applications of multi-threading include database servers, and users of databases will be aware that aspects of simultaneous access can require special care.
- In specialized application areas such as high speed digital signal processing (DSP) e.g. radar processing, digital mobile base station processing, etc., hybrid parallel systems may be assembled with individual CPUs having significant private memory plus a connection onto a common shared memory.
- While these tightly coupled systems require specialized hardware support in order that the CPUs can share the common memory system, another approach is to use a network to join together more conventional systems into what is termed a distributed system.

1.2.6 Distributed Systems

Multicomputers (do not share memory and clock); loosely-coupled.

- Networked computers
- Require networking infrastructure
- Local area networks (LAN) or Wide area networks (WAN)
- May be either client-server or peer-to-peer systems

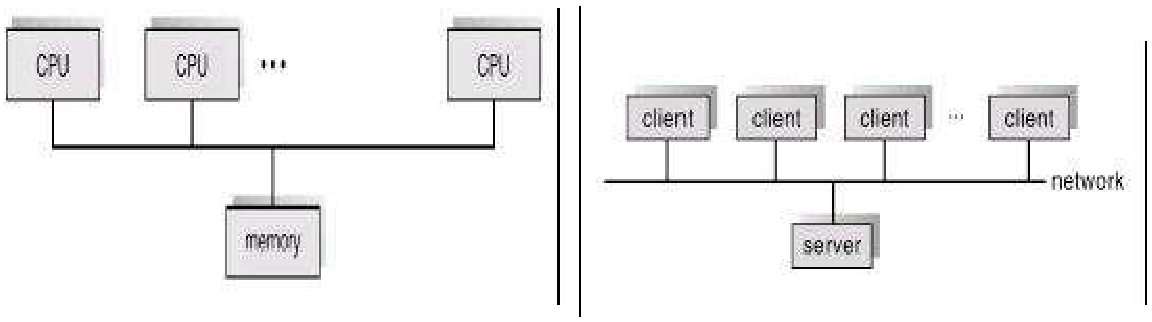


Figure 4: SMP architecture, Client-Server

  - Client and server roles may vary e.g. X terminal is a windows *server*; runs on machine you think of as *client*
  - Client-Server Systems are a common form of distributed system in which the client system and server system are not similar.
  - An example of a client-server system is the file server on campus. Here, a central server provides file access to authenticated users at client machines.
  - Peer-to-Peer Systems are another form of distributed system in which the participating computer systems are similar.

- Students wishing to experiment with programming at this parallel level can investigate the PVM (Parallel Virtual Machine) and MPI (Message Passing Interface) libraries that are widely available.

1.2.7 Clustered Systems

- 2 or more systems share resources
- Provides high availability
- *Asymmetric clustering*: one server runs application, rest stand by.
- *Symmetric clustering*: all N run application

1.2.8 Real–Time

Deadline (time critical) requirements. A real-time system is required to produce a result within a non-negotiable time period.

- Common uses:

    - control device in dedicated application, e.g., control scientific experiment, medical imaging, industrial control, space shuttle control systems, anti-lock automotive brake systems, banking systems, etc.
    - some display systems

- Real-Time systems may be either;
    - *hard* (must react in time), the real-time system absolutely must complete critical tasks within a guaranteed time.
        * Secondary storage limited or absent, data stored in short term memory, or read-only memory (ROM)
        * Conflicts with time-sharing systems, not supported by general-purpose operating systems.
    - *soft* real-time (deal with failure to react in time), the real-time system can satisfy its performance criteria by running any critical task at a higher priority (of CPU access).
        * Limited utility in industrial control of robotics
        * Useful in applications (multimedia, virtual reality) requiring advanced operating-system features.
        * In some instances, off-the-shelf operating systems such as Linux or *BSD may be modified to support soft real-time operation. An alternative is for the Linux or *BSD operating system to be run as a task within some other (less conventional) real-time operating system.
        * An example of soft real-time service is a multi-media server delivering audio or video – if it fails, no loss of life (other than social life) occurs.

1.2.9 Embedded, Smart-Card and Handheld

- Embedded systems are the most common. They typically run real-time operating systems with custom I/O designed for specific tasks.
- For example, a microwave oven contains a microprocessor chip with built in peripherals such as timers and I/O lines so that cooking may be controlled and keypads and LCD modules handled.
- Personal Digital Assistants (PDAs)
- Cellular telephones, Cameras? ...

- Smart-card and digital mobile telephones also run custom real-time operating systems. At least two “standards” exists – one is JAVA based. The computation load from handling encryption means that the designer has an interesting problem given limited resources of electrical power, memory, and CPU capacity.
- Hand-held systems must also deal with limited resources although their screens have recently become more substantial.
- Issues:
  - Limited memory
  - Variety of interconnect standards
  - Slow processors
  - Small screens
- Their current evolution may be towards a form of “cut back” PC. Sounds like a PC in 1985...

1.3 Computer Hardware Review

Understanding operating systems requires some basic understanding of computer systems.

1.3.1 Processor

- Processor
  - Fetches instructions from memory, decodes and executes them
  - Set of instructions is processor specific
  - Instructions include:
    - load value from memory into register
    - combine operands from registers or memory
    - branch
  - All CPU’s have registers to store
    - key variables and temporary results
    - information related to control program execution
- Processor Registers

  - Data and address registers
    - Hold operands of most native machine instructions
    - Enable programmer to minimize main-memory references by optimizing register use
    - user-visible
  - Control and status registers
    - Used by processor to control operating of the processor
    - Used by operating-system routines to control the execution of programs
    - Sometimes not accessible by user (architecture dependent)

- User–Visible Registers
  - May be referenced by machine language instructions
  - Available to all programs - application programs and system programs
  - Types of registers
    - Data
    - Address
      - Index
      - Segment pointer
      - Stack pointer
    - Many architectures do not distinguish different types

- Control and Status Registers
  - Program Counter (PC), Contains the address of an instruction to be fetched
  - Instruction Register (IR), Contains the instruction most recently fetched
  - Processor Status Word (PSW)
    - condition codes
    - interrupt enable/disable
    - supervisor/user mode
  - Condition Codes or Flags
    - Bits set by the processor hardware as a result of operations

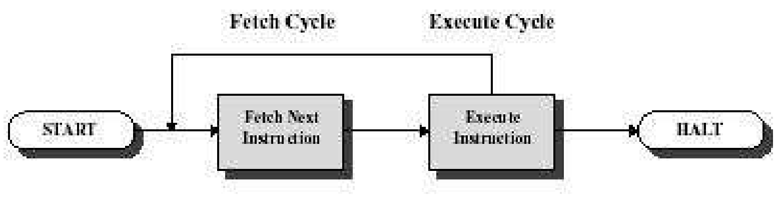


Figure 5: Fetch and Execute

    * Can be accessed by a program but not altered
    * Examples: positive/negative result, zero, overflow

- Instruction Fetch and Execute
  - Program counter (PC) holds address of the instruction to be fetched next
  - The processor fetches the instruction from memory
  - Program counter is incremented after each fetch
  - Overlapped on modern architectures (pipelining)
- Instruction Register
  - Fetched instruction is placed in the instruction register
  - Types of instructions
    * Processor-memory, transfer data between processor and memory
    * Processor-I/O, data transferred to or from a peripheral device
    * Data processing, arithmetic or logic operation on data
    * Control, alter sequence of execution

1.3.2 Main Memory

- Referred to as real memory or primary memory
- volatile, because its contents are lost when the power is removed
- Should be, fast, abundant, cheap, Unfortunately, that's not the reality..., Solution: combination of fast & expensive and slow & cheap memory

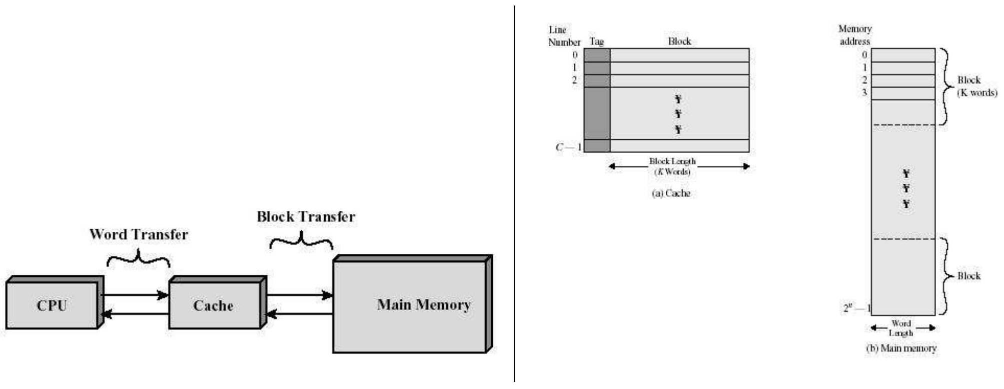


Figure 6: Cache Memory

- Program instructions and the data used by programs being executed must reside in high speed semiconductor memory called random access memory (RAM) in order to obtain high speed operation. We say random access because the CPU can access any byte of storage in any order.

Disk Cache

- A portion of main memory used as a buffer to temporarily to hold data for the disk
- Disk writes are clustered
- Some data written out may be referenced again. The data are retrieved rapidly from the software cache instead of slowly from disk
- Mostly transparent to operating system

Cache Memory

- Contain a small amount of very fast storage which holds a subset of the data held in the main memory
- Processor first checks cache
- If not found in cache, the block of memory containing the needed information is moved to the cache replacing some other data

Cache Design

- Cache size, small caches have a significant impact on performance

- Line size (block size), the unit of data exchanged between cache and main memory
- hit means the information was found in the cache
- larger line size $\Rightarrow$ higher hit rate
- until probability of using newly fetched data becomes less than the probability of reusing data that has been moved out of cache

Mapping function, determines which cache location the data will occupy. Replacement algorithm

- determines which line to replace
- Least-Recently-Used (LRU) algorithm

Write policy,

- When the memory write operation takes place
- Can occur every time line is updated (write-through policy)
- Can occur only when line is replaced (write-back policy)
  - Minimizes memory operations
  - Leaves memory in an obsolete state

1.3.3 I/O Modules and Structure

- secondary memory devices
- communications equipment
- terminals

CPU much faster than I/O devices

- waiting for I/O operation to finish is inefficient
- not feasible for mouse, keyboard
- I/O module sends an interrupt to CPU to signal completion
- Interrupts normal sequence of execution
- I/O requests can be handled synchronously or asynchronously.

  - In a synchronous system, a program makes the appropriate operating system call and, as the CPU is now executing operating system code, the original program's execution is halted i.e. it waits.
  - In an asynchronous system, a program makes its request via the operating system call and then its execution resumes. It will most likely not have had its request serviced yet!
  - The advantage of having an asynchronous mechanism available is that the programmer is free to organize other CPU activity while the I/O request is handled.

- Software that communicates with controller is called device driver
- Most drivers run in kernel mode
- To put new driver into kernel, system may have to
  - be relinked
  - be rebooted
  - dynamically load new driver
- we must have an event driven I/O system handling all of the pending I/O requests (maybe these are triggered when data arrives, or a peripheral device such as a CD drive indicates it is ready etc.).
- Requests that the operating system has not yet been able to service might mean that the program is currently "sleeping" or "waiting".

1.3.4 System bus

Communication among processors, memory, and I/O modules.

- A system bus would link the CPU and memory – this structure would involve a pathway along which data could travel (usually 32-bits side-by-side i.e. in bit-wise parallel), a pathway along which the address specifying a particular desired memory location could travel, and a few other lines which would tell the memory whether to store (write) or retrieve (read) data in an access.
- As any I/O device must pass data between the computer and the outside world, it will also be attached to the memory system and the CPU via the system bus.

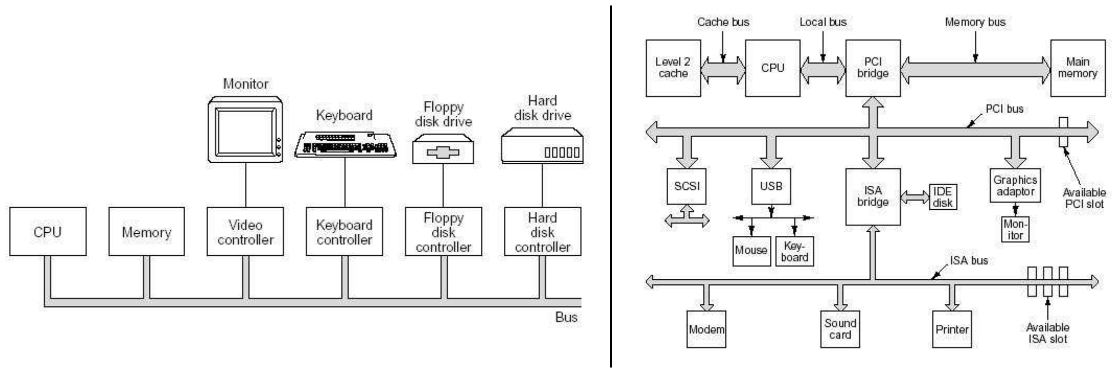
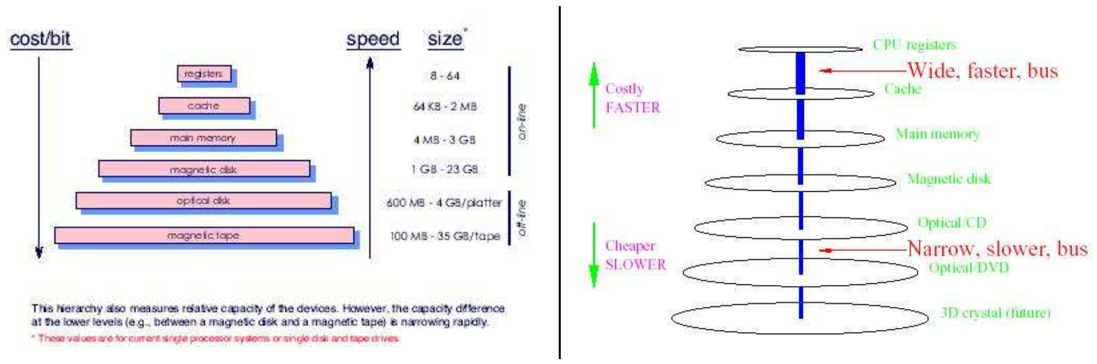


Figure 7: Top-level Components, Pentium System



This hierarchy also measures relative capacity of the devices. However, the capacity difference at the lower levels (e.g., between a magnetic disk and a magnetic tape) is narrowing rapidly.

* These values are for current single processor systems or single disk and tape drives.

Figure 8: Going down the hierarchy

1.3.5 Storage Structure and Hierarchy

- Decreasing cost per bit
- Increasing capacity
- Increasing access time
- Decreasing frequency of access of the memory by the processor
- Fully electronic memory systems are the fastest and most expensive, hence must be used in cost effective ways. This memory system is called main memory or primary memory.
- A source of cheaper-per-byte and non-volatile storage is provided by magnetic disk. However, the computer does not have direct random access to any byte at any time on the disk – the magnetic discs in the drive are rotating and magnetic heads move in and out in order to access any part of the surface area on the disc that holds data. This

means access usually involves a disc rotation delay and also a head positioning delay.

- Other common forms of non-volatile secondary storage include: optical CD drives (CD-R write-once or CD-RW read-write), recent flash memory chips in very small modules that can be inserted into laptop card interfaces or can be used for data logging.
- Stages such as the CPU registers and cache are typically located within the CPU chip so distances are very short and busses can be made very very wide (e.g. 128-bits), yielding very fast speeds.
- Future storage technology includes 3-dimensional crystal structures which allow optical access to a dense 3-dimensional storage facility.

1.3.6 Protection

- Single Tasking System
  - Only one program can perform at a time
  - Simple to implement, Only one process attattempting use resources
  - Few security risks
  - Poor utilization of the CPU and other resources
  - i.e., MS-DOS
- Multi Tasking System
  - Very complex
  - Serious security issues, how to protect one program from another sharing the same memory
  - Much higher utilization of system resources
  - i.e., Unix, Windows NT
- OS must protect itself from users -reserved memory only accessible by OS. The operating system is responsible for allocating access to memory space and CPU time and peripherals etc., and it will control dedicated hardware facilities to help it enforce whatever resource allocation policies are in force:

  - The memory controller, unremarked when it appeared in the basic computer ororganizationis under operating system control to detect and prevent ununauthorizedccess
  - A timer will also be under operating system control to manage CPU time allocation to programs competing for resources

- OS may protect users from another user. A fundamental requirement of multiple users of a shared computer system is that they do not interfere with each other. This gives rise to the need for separation of the programs in terms of their resource use and access;
  - If one program attempts to access main memory allocated to some other program, the access should be denied and an exception raised
  - If one program attempts to gain a larger proportion of the shared CPU time, this should be prevented

- One approach to implementing resource allocation is to have at least two modes of CPU operation, where one mode called the supervisory mode has its code kept in a reserved memory region, and to limit execution of special resource allocation instructions to only the program executing in the supervisory mode.

- Modes of operation
  - supervisor (protected, kernel) mode: *all* (basic and privileged) instructions available.
    - all hardware and memory available
    - mode the OS runs in
    - never let the user run in supervisory mode
  - user mode: a *subset* (basic only) of instructions.
    - limited set of hardware and memory available
    - mode all user programs run in
    - I/O protection, all I/O operations are privileged
    - Memory protection, base/limit registers (in early systems), memory management unit, MMU (in modern systems)
    - CPU control, timer (alarm clock), context switch
  - All I/O instructions are restricted to supervisory mode – so user programs can only access I/O by sending a request to the (controlling) operating system

  - All instructions controlling the memory management unit are restricted to supervisory mode – so user programs can only access the memory that the operating system has allocated
  - All instructions controlling the timer (or real-time clock) are restricted to supervisory mode – so user programs can only read the time of day, and can only have as much CPU time as the operating system allocates
  - All interrupt vector table entries, which are specific to each task or program that can run, must be configured (initially at least) by the (controlling) operating system

- One of the early advantages of UNIX operating systems was that a well defined set of system calls was developed to allow a programmer to request access to system resources.