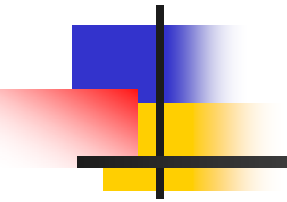


Operating Systems
(Introduction)

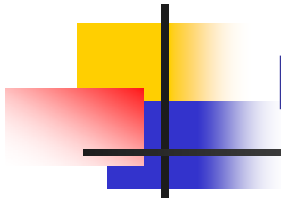
Chapter 1

These lecture materials are modified from the lecture notes written by A. Silberschatz, P. Galvin and G. Gagne.

Spring, 2020

Notice

- 본 사이트에서 수업 자료로 이용되는 저작물은 저작권법 제25조 수업목적저작물 이용 보상금제도의 의거, 한국복제전송저작권협회와 약정을 체결하고 적법하게 이용하고 있습니다. 약정범위를 초과하는 사용은 저작권법에 저촉될 수 있으므로 수업자료의 대중 공개·공유 및 수업 목적외의 사용을 금지합니다.



References

- Main textbook
  - A. Silberschatz et al. “Operating system concepts,” tenth edition
- References
  - R. Arpaci-Dusseau et al. “Operating systems: three easy pieces”
    - http://pages.cs.wisc.edu/~remzi/OSTEP/

Outline

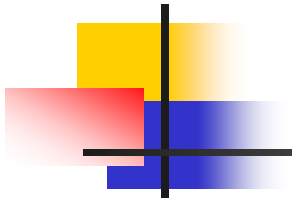
- Operating systems definitions
- Computer systems organization
- Computer system architecture
- Operating systems structure
- Operating systems operation
- What to learn?

Operating systems definitions

- Definition
  - No universally accepted definition
- But, by looking at the operating system's role in the overall computer system, we may define an operating system as follows:

What is an operating system ?

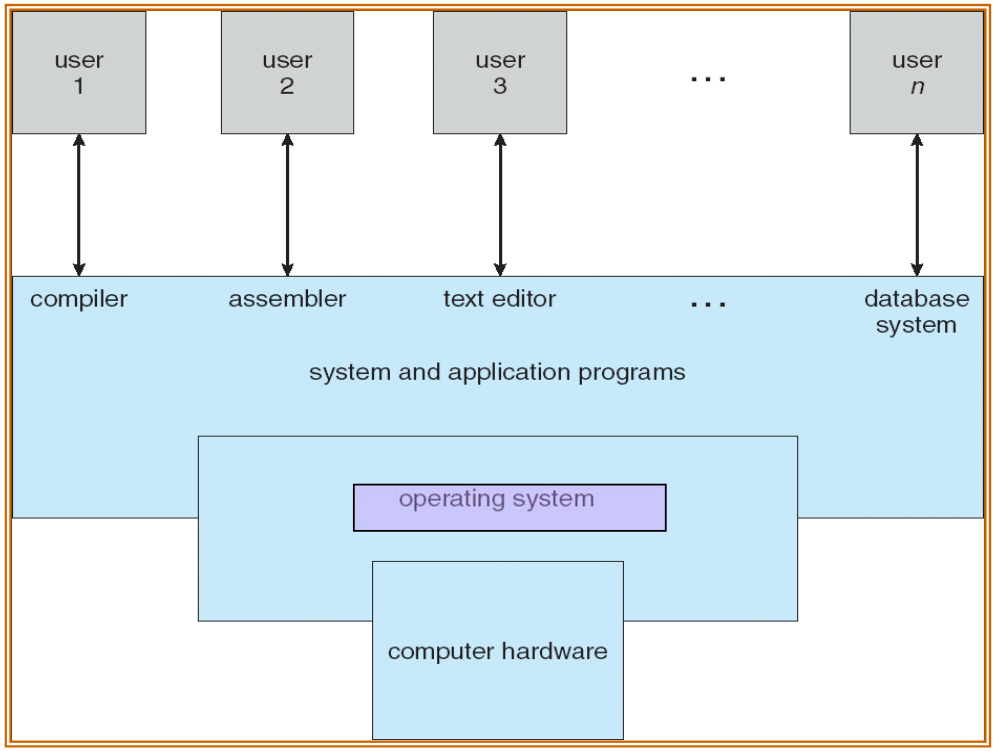
An operating system is a program
1. that manages the computer hardware
2. that controls the execution of programs



- The general roles of the OS
  - Hardware management
    - Access to I/O devices
    - Access to files
    - Accounting
    - Error detection
  - Program execution
    - Scheduling
    - Error reporting

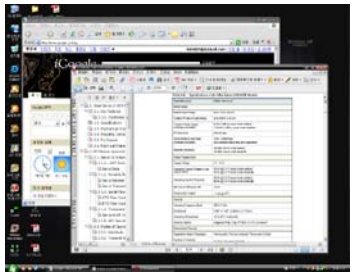
Where is it ?

Between application programs and the computer hardware.



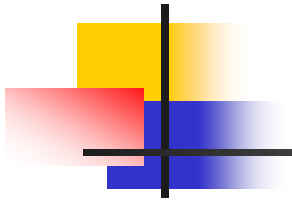
- Objectives of the operating systems;
  - 1. Execute user programs and make it easier to solve user problems.
  - 2. Make the computer system convenient to use.
  - 3. Use the computer hardware in an efficient manner

DOS

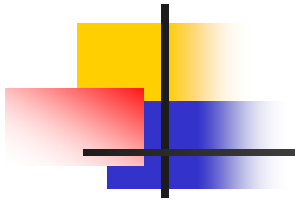


windows

iPAD



- We can say that;
  - 1. OS is a **resource allocator**
    - Manages all the resources
      - CPU time, memory space, file-storage space, I/O devices
    - Decides how to allocate the resources to conflicting requests for efficient and fair resource use
  - 2. OS is a **control program**
    - Controls the execution of programs to prevent errors and improper use of the computer systems



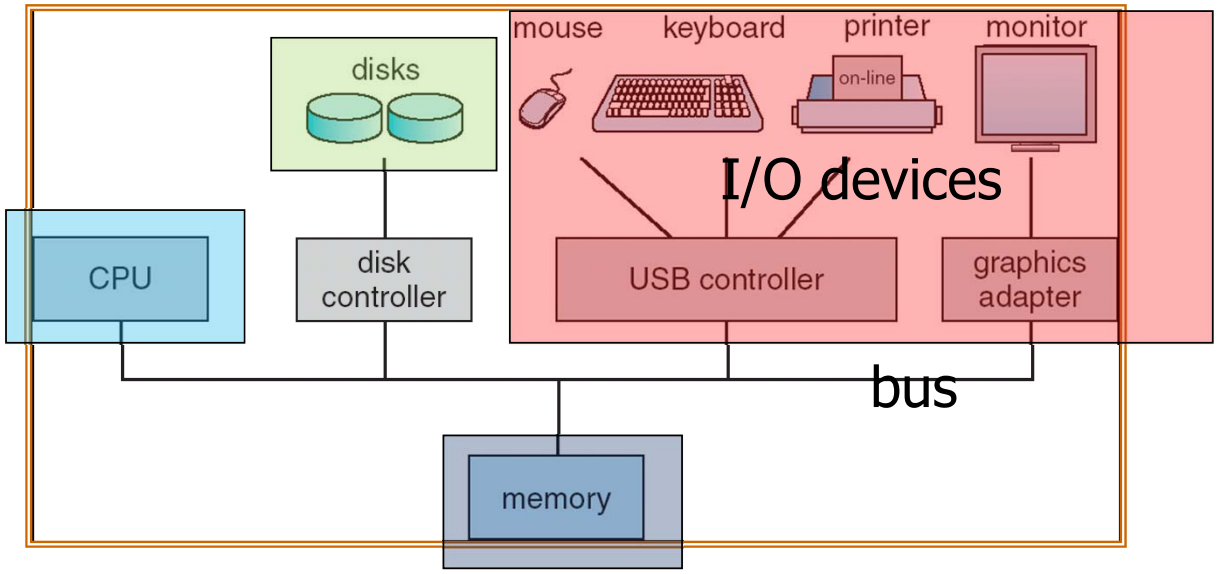
- Another definition;
  - “The one program running at all times on the computer” is the **kernel** or the operating system
    - Everything else is either a system program or an application program

Computer systems organization

OS is a resource manager

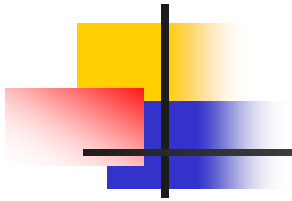
- Computer systems
  - Personal computers
  - Large-scale systems
    - Servers
  - Hand-held systems
  - …

Resources

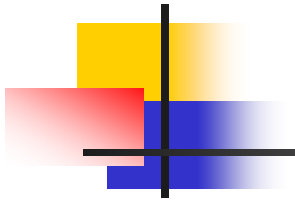


- One or more CPUs and device controllers are connected through common bus providing access to shared memory
- Concurrent execution of CPUs and devices competing for memory cycles

- Four basic principles;
  - 1. Computer system I/O operation
  - 2. I/O structure
  - 3. Interrupt
  - 4. Storage structure
- Additional features;
  - Multi-processor systems

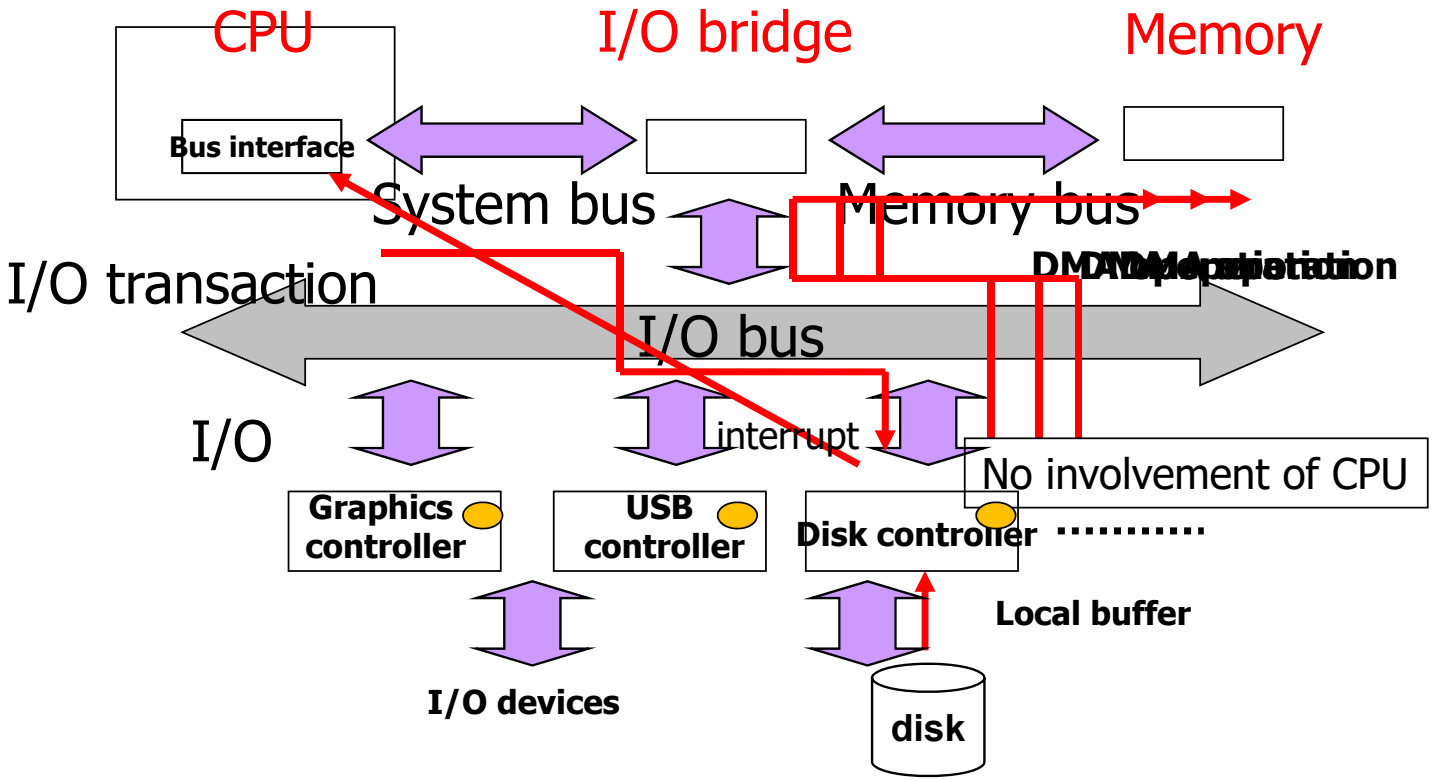


- Computer-system I/O operation
  - Each device controller is in charge of a particular device type.
    - Each device controller has a local buffer.
  - I/O: data is moved from/to main memory to/from local buffers
  - I/O devices and the CPU can execute concurrently.
    - Why? DMA (Direct Memory Access)
  - Device controller informs CPU that it has finished its operation by causing an *interrupt*.

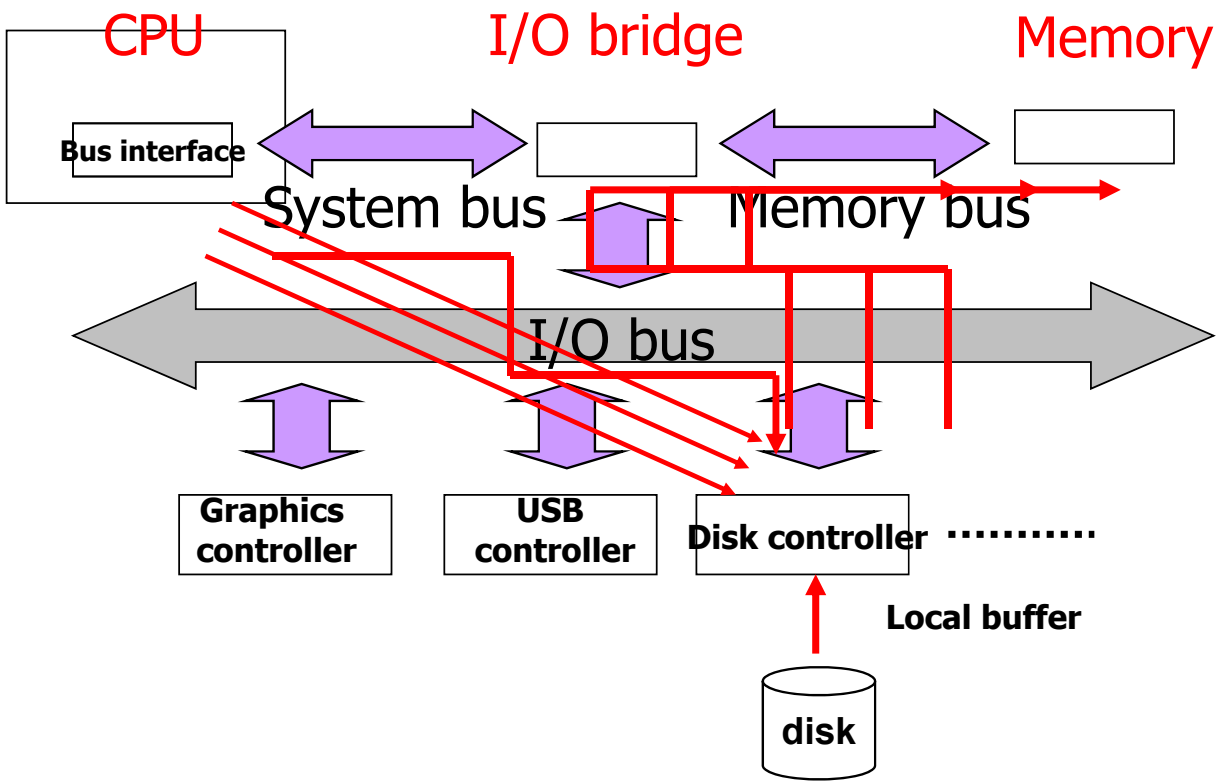


- I/O structure
  - I/O transactions is performed through buses
    - A bus is a collection of parallel wires that carry address, data, and control signals.
    - A system bus that connects the CPU and the I/O bridge
    - A memory bus that connects the I/O bridge and the memory
    - I/O buses are typically shared by multiple devices.
  - DMA (Direct Memory Access) is typically used
    - The process whereby a device performs a read or a write bus transaction on its own without CPU intervention
    - So, with DMA, the CPU would initiate the transfer, do other operations while the transfer is in progress

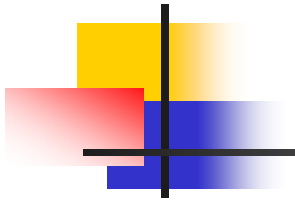
DMA operation



What if the system does not support DMA operation ?

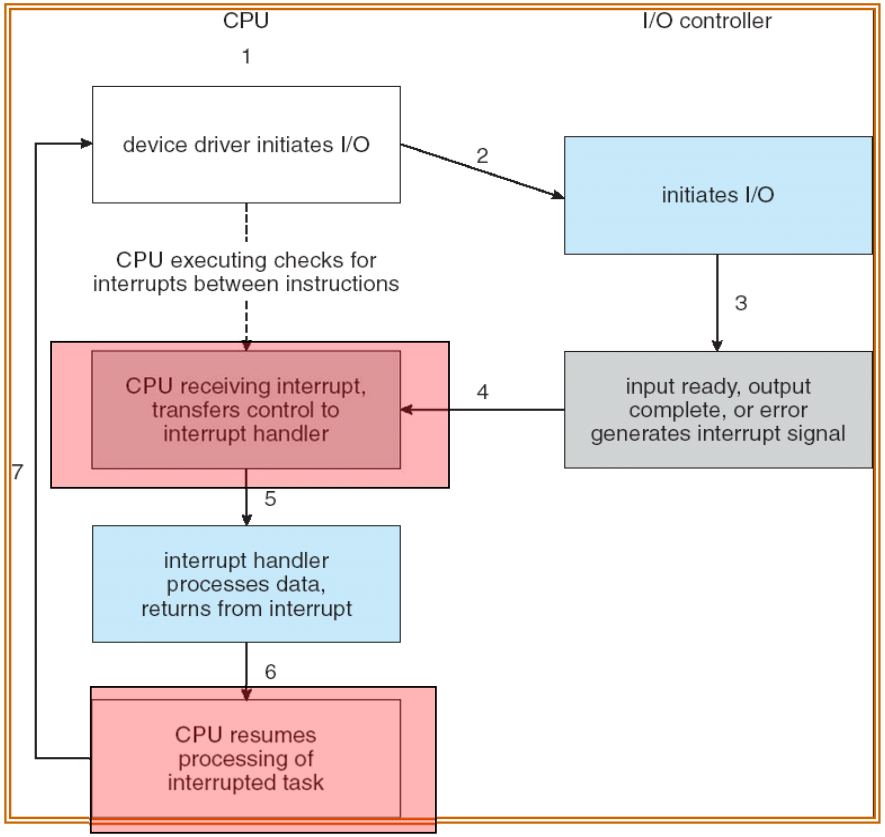


I/O devices and the CPU cannot execute concurrently.

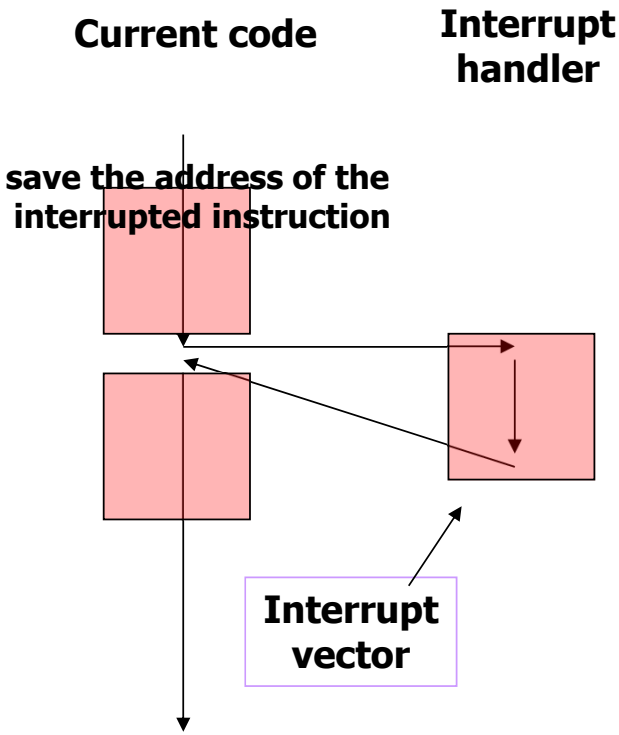


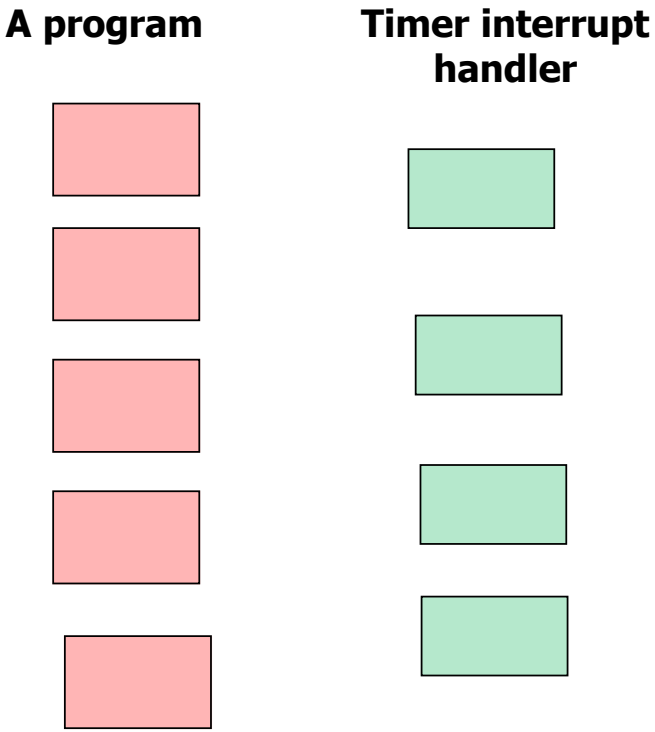
- Interrupt
  - Interrupt transfers control to the interrupt service routine generally, through the *interrupt vector*, which contains the addresses of all the service routines.
  - Interrupt architecture must save the address of the interrupted instruction.
  - An operating system is *interrupt* driven.
    - e.g.) timer interrupt

- Interrupt-driven I/O

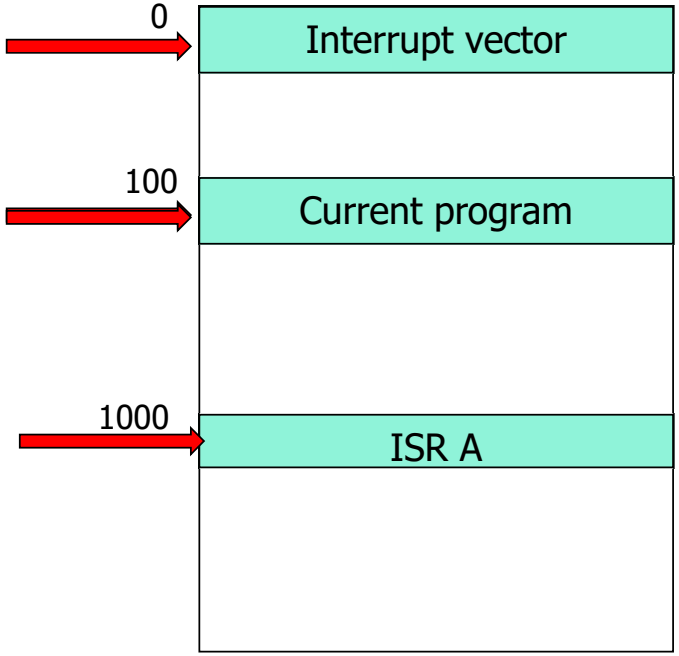


<Example>


Current code
Interrupt handler
save the address of the interrupted instruction
Interrupt vector

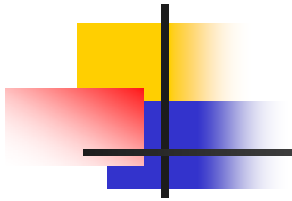

A program
Timer interrupt handler

- For your understanding;
  - "Program counter" register 
    - Indicates the next instruction to be executed



Finds the location of ISR A

Interrupt service routine for interrupt A



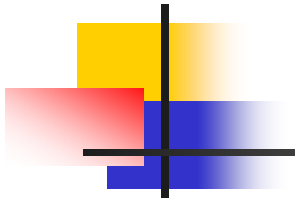
- Difference between interrupt and trap
  - Interrupt

asynchronous

    - Caused by events external to the processor
      - Indicated by setting the processor's interrupt pin
      - e.g. hitting ctl-c at the keyboard
  - Trap

synchronous

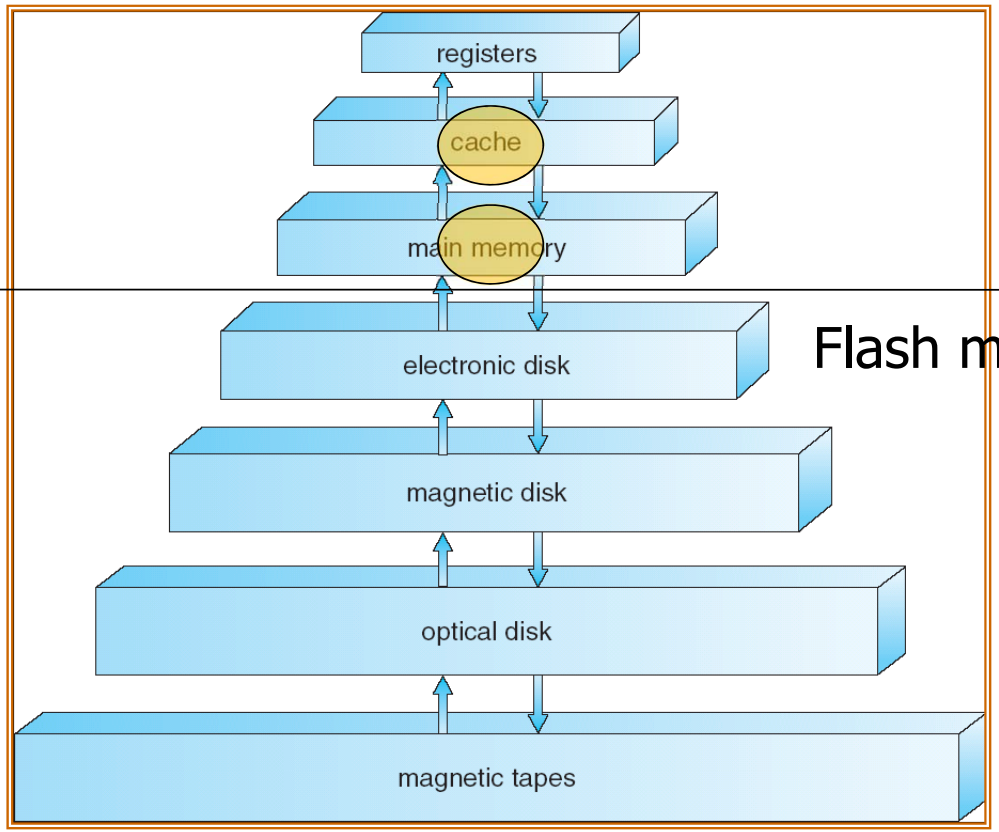
    - Caused by events that occur as a result of executing an instruction
    - Software interrupt
      - e.g.
        - System call (Refer to chapter 2)
        - Segmentation fault



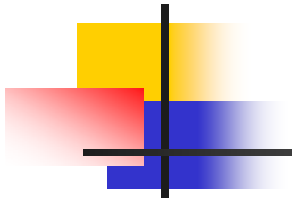
- Storage structure
  - Main memory – only large storage media that the CPU can access directly. (DRAM)
  - Secondary storage – extension of main memory that provides large nonvolatile storage capacity.
    - Magnetic disks – rigid metal or glass platters covered with magnetic recording material
      - Disk surface is logically divided into *tracks*, which are subdivided into *sectors*.
      - The *disk controller* determines the logical interaction between the device and the computer.

- Storage structure

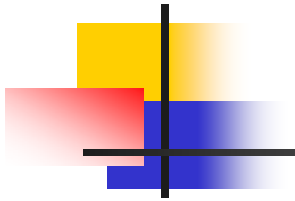
Speed
Cost
Volatility



Flash memory



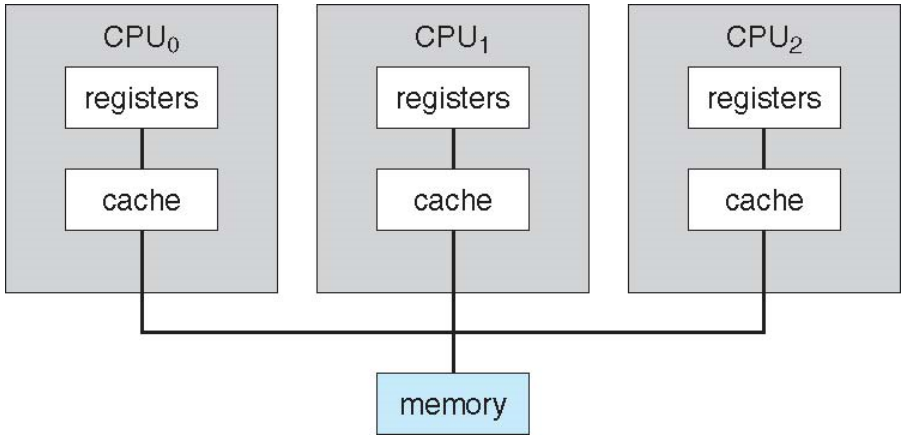
- Storage technology trends
  - Different storage technologies have different price and performance tradeoffs
  - The price and performance properties of different storage technologies are changing at dramatically different rates.
- Caching is an important principle of computer systems



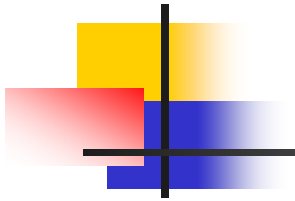
- How to access storage medium
  - Faster storage (cache) checked first to determine if information is there
    - If it is, information used directly from the cache (fast)
    - If not, data copied to cache and used there
- Cache (e.g. cache) is smaller than storage being cached (e.g. main memory)
  - Cache management is important
  - Considerations
    - Cache size and replacement policy

Computer systems architecture

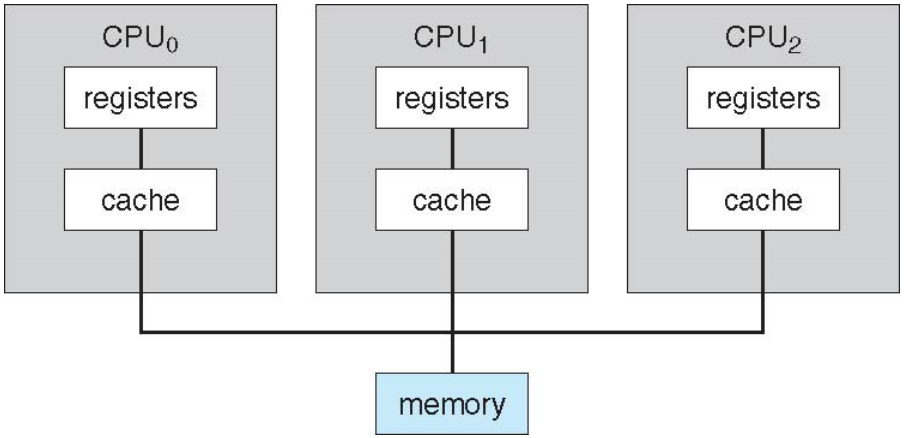
- Most systems use a single general-purpose processor (PDAs through mainframes)
  - Most systems have special-purpose processors as well
- Multiprocessors systems growing in use and importance
  - Also known as parallel systems, tightly-coupled systems
  - Advantages
    - 1. Increased throughput
    - 2. Economy of scale
    - 3. Increased reliability



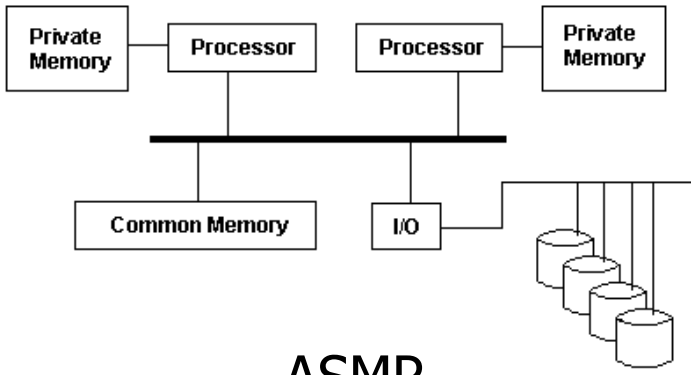
- 1. Increased throughput
  - We expect to get more work done in less time
  - The speedup ratio with *N* processors is not N; it is less than *N*
- 2. Economy of scale
  - Processors can share peripherals, storage and memory.
- 3. Increased reliability
  - The failure of one processor will not halt the system



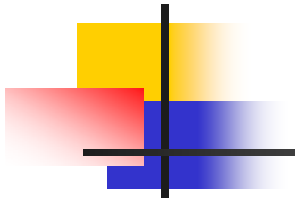
- Two types of multi-processor systems;
  - Asymmetric multiprocessing
    - One processor (master processor) that has access to the memory map as a whole, and other processors which act as slaves to the main or master processor
    - A master processor controls system; the other processors either look to the master for instruction
    - Usually, these slave processors will have their own memory which is not tied to the primary processors' memory
  - Symmetric multiprocessing
    - No master-slave relationship between processors
    - Each processor shares the memory and can interact each other directly


CPU0
registers
cache
CPU1
registers
cache
CPU2
registers
cache
memory

SMP

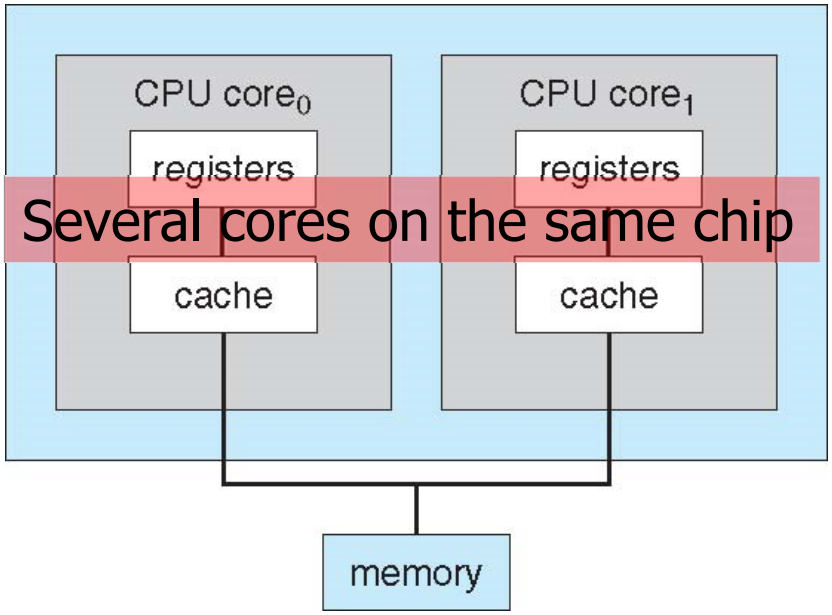

Private Memory
Processor
Processor
Private Memory
Common Memory
I/O

ASMP



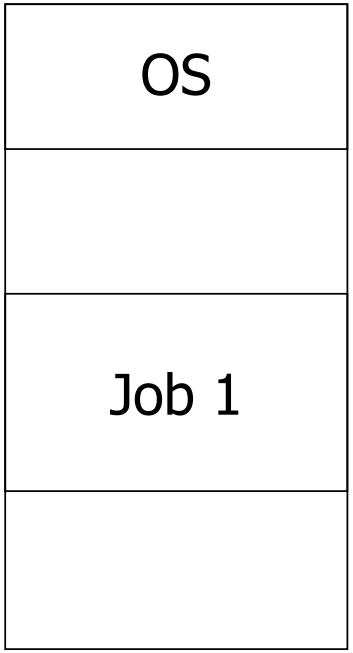
- Multiprocessor memory model
  - Uniform memory access (UMA)
    - Access to any RAM from any CPU takes the same amount of time
  - Non-uniform memory access (NUMA)
    - Memory access time depends on the memory location relative to a processor.

- Multi-core systems



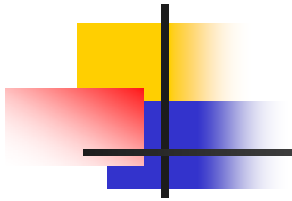
Operating systems structure

- Multiprogramming
  - Purpose
    - To keep CPU and I/O devices busy at all times
  - How ?
    - Multiprogramming organizes jobs (code and data) so CPU always has one to execute
    - A subset of total jobs in system is kept in memory
    - One job selected and run via **job scheduling**
    - When it has to wait (for I/O for example), OS switches to another job

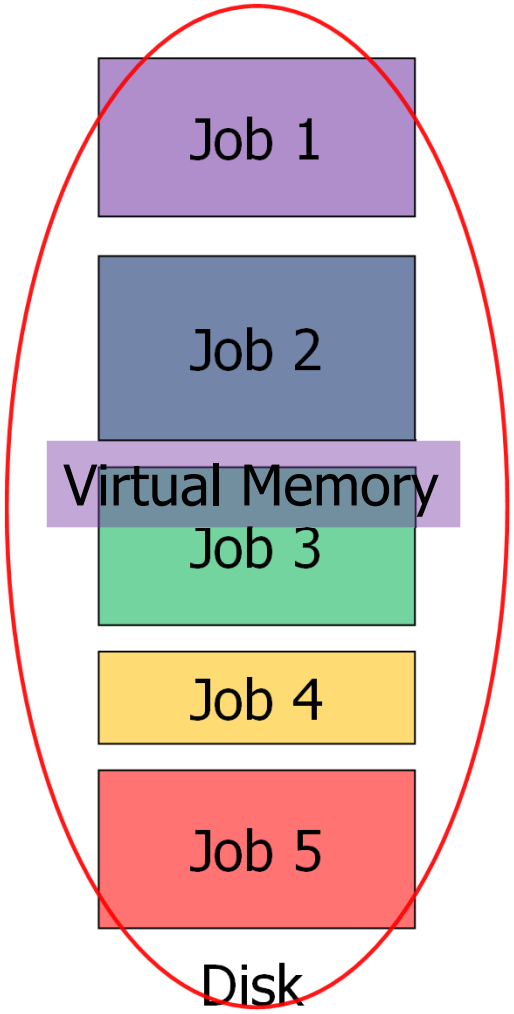


Single-programming
(MS-DOS)

Multiprogramming



- Some related concepts
  - If several jobs ready to run at the same time ⇨ CPU scheduling
  - If several jobs are ready to be brought into memory, and there is not enough room for them ⇨ Job scheduling
  - If processes don't fit in memory, swapping moves them in and out to run
  - Virtual memory allows execution of processes that are not completely in memory
  - Programs in execution ⇨ Process

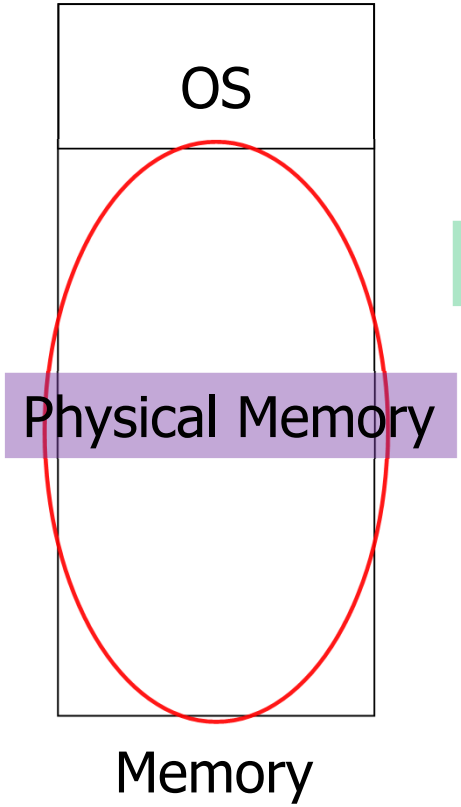


Which jobs can be brought into the memory ?

Job scheduling

Swapping

>

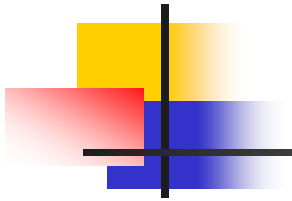


Which job can be executed by CPU ?

CPU scheduling

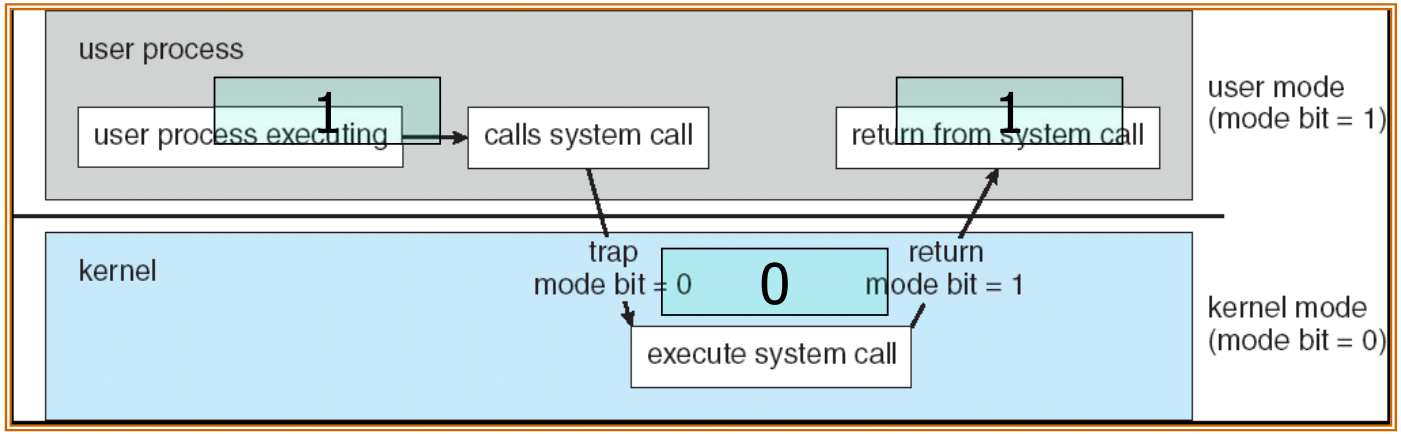
Operating system operation

- Modern OS is interrupt-driven
  - Interrupt driven by hardware
  - Software error or request creates **exception** or **trap**
    - Division by zero, request for operating system service



- Modern OS supports a dual-mode operation
  - User mode and kernel mode (or supervisor mode or privileged mode)
  - 1. Allows us to differentiate between user and kernel mode
    - A mode bit is added to the hardware to indicate the current mode
      - Kernel (0)
      - User (1)
  - 2. Provides us with means for protecting the operating system from errant users
    - Privileged instruction
      - Allowed to be executed only in kernel mode

Mode bit


user process
1
user process executing
calls system call
1
return from system call
user mode
(mode bit = 1)
kernel
trap
mode bit = 0
0
return
mode bit = 1
execute system call
kernel mode
(mode bit = 0)

- Privileged instructions
  - If an attempt is made to execute a privileged instruction in user mode, the hardware does not execute the instruction
  - Representative examples;
    - Instructions to switch to user mode
    - Instructions for I/O control
    - Instructions for interrupt