Operating Systems

- software that manages a computers hardware
- also acts as the basis for application programs

① comp system

hardware   OS   Applications prog ⇇ user

go b/w

singular interaction

② comp system

hardware   software   data

- OS's job : ① resource allocator ← also hides HW complexity
  ② manage execution of user programs } efficiency

- OS can be defined as a program that runs at all times on the computer → called a kernel
  ↓
  can come w/ middleware too (helps application developement)

Operating System Operations → but HW

① Single Processor : 1 CPU core

② Multiprocessor Systems : 2/+ processors w/ a single CPU core or CPU w/ multiple cores

more power efficient

same main mem but diff cache and registers

- aka tightly coupled systems ⇒ communication through shared energy
- parallel processing
  - increased throughput
  - economical due to sharing of memory and I/O devices
  - increased reliability due to redundancy

① batch systems

② multiprogrammed & timesharing systems

③ embedded and cyber physical systems

or **Types of Computing Systems**

Operating System Operations → (SW)

"job" → I/O
→ CPU ops

① Simple batch systems

1st generation

- batching similar jobs : thereby reducing set up time
- automatic job sequencing : transfers control from one job to another
- simple memory layout : only one user job in memory at any point in time
- not very efficient : when job waits for I/O, CPU idles

② Multiprogrammed (Time-Sharing) Systems

one kind of

2nd gen. more advanced

- several jobs are kept in the main memory, CPU is multiplexed among them
- job is swapped from memory to hard disk ← needs functionality of VM
- supports multiple online users

requires OS features like

- memory management
- CPU scheduling
- I/O device scheduling

- basically multiple jobs
- CPU is utilized better

③ Desktop Systems → Mac, Linux, Windows.

latest technology

- dedicated to a single user
- several I/O devices ← hardware drive is important
- efficiency and user convience and responsiveness is the top goal
- fancy GUI → graphical user interface

4. Embedded and cyber-physical systems

- physical systems whose operations are monitored and controlled by a reliable computing and communication core

- resource constrained : low power, small memory, low bandwidth

- domain-specific OSes ← have to be very efficient
eg. android, tinyOS

5. Real-time systems

- used as control devices in a dedicated application
eg. industrial controls, automotive, avionics, medical devices

- well defined fixed time constraints
LynxOS, RTLinux

6. Handheld systems

- a subset of embedded and cyber-physical systems
eg. iOS, android.

- constraints :
  - less memory
  - slow processor speed
  - small display

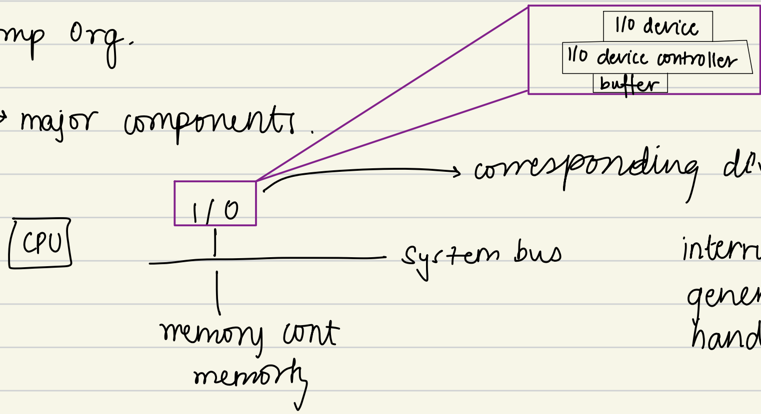
computer system operation | system hierarchy | hardware protection

Computer System Architecture

computer system Operations

1. Comp Org.

↳ major components.



corresponding device controller

interrupt
generated by HW
handled by CPU

CPU scheduling
memory control
I/O control
hard disk (file system)

} OS manager

imp pointers :

- I/O devices and CPU execute concurrently
- each device controller is in charge of a particular device type and has a local buffer and controls moving data b/w buffer and memory
- ↳ informs CPU that it has finished the operation by causing an interrupt

ALL OS'S ARE INTERRUPT DRIVEN ⇒ NO INTERRUPT, OS IDLES

Interrupts

code in OS, accesses OS data structures

- control $\xrightarrow{\text{interrupt}}$ interrupt service routine

  ↑

  interrupt vector ← table of interrupts and ISRs

- incoming interrupts are disabled while another interrupt is being processed to prevent any loss of interrupts

- trap : CPU generated interrupt caused by a software error or request

  ↑

  basically software generated interrupt

- OS preserves state of CPU by storing reg & prog counter

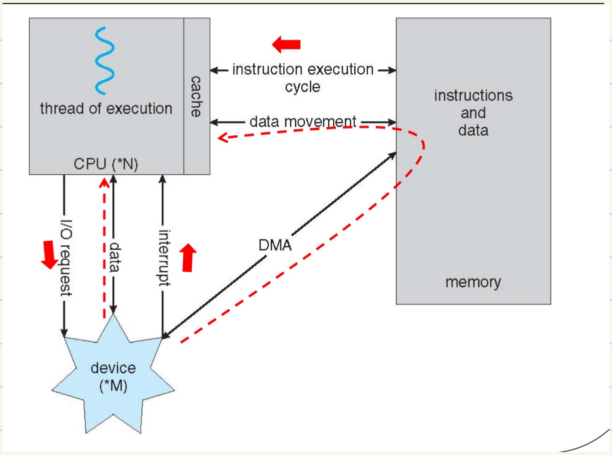
  context switch

Direct Memory Access

- used for high-speed I/O devices that are able to transmit info at close to memory speeds.
- OS sets up the memory blocks, counters, and buffers once

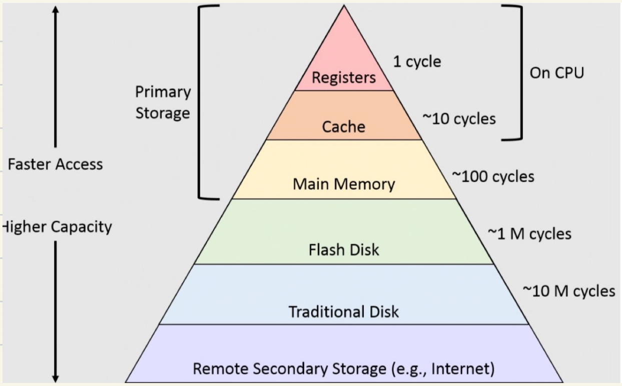
  ↓

  device controller can then transfer blocks of data from the buffer directly to the memory and only <u>1 interrupt is generated per block</u>.


thread of execution
cache
CPU (*N)
instruction execution cycle
data movement
instructions and data
memory
I/O request
data
interrupt
DMA
device (*M)

Storage Hierarchy

- registers → cache → main memory → secondary storage
- 80/20 rule : 80% memory access goes to 20% of data items



Hardware Protection

dual-mode operation

I/O protection

memory protection

Dual-Mode Operation

- user mode : execution of user processes

not same as root/admin

- kernel mode (supervisor / system / monitor mode) :
  execution of <u>operating system processes</u>
  eg. interrupt handling ↳ priviledged instructions
- mode bit : added to computer hardware (0 ← kernel, 1 ← user)
- default : user mode, on incoming interrupts / traps, hardware changes mode to kernel mode

I/O Protection

- user prog can issue illegal I/O operations and hence I/O must be protected.

∴ ALL I/O INSTRUCTIONS ARE PRIVILEDGED INSTRUCTIONS
⇒ kernel mode execution

as they must go through the OS to ensure correctness + legality

- CPU generates a trap for I/O ops that try to bypass the OS

Memory Protection

- required for interrupt vector and ISR ← cause if that is messed up, a lot of problems are caused.
- also, main memory is divided into kernel and user space
- done by defining memory area for each program and, adding 2 special registers that determine the range of the memory addresses that a program can access

base & limit registers

starting → range

and memory outside the range cannot be accessed.

OS decides these values, and so loading them into the CPU registers is a priviledged instruction. done only in kernel mode

Operating System Services

OS has layers :

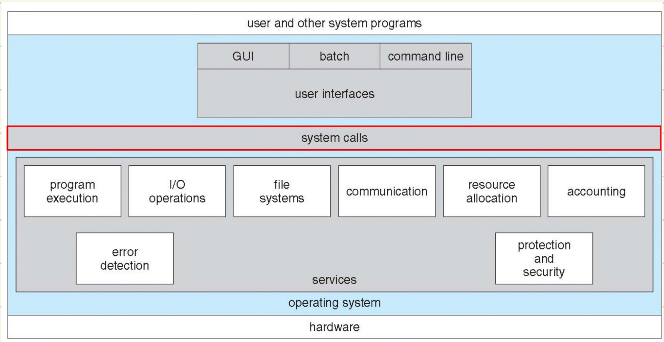
UI , system calls , services

accessed via API

called system calls

executed in kernel mode

interface b/w user prog & OS services



System calls are in assembly lang, C, C++

eg. fopen ()

switch from user to kernel , get file , switch back to user ← 2 mode switches

I/O :: priviledged