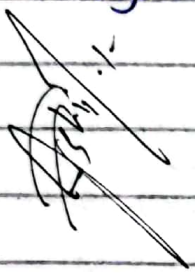
Chapter no 1:- (1.1, 1.2, 1.4, 1.11)

Q:- What operating Systems do?

- Operating System → manages computers hardware.
- System divides into →
  - Hardware (cpu, memory)
  - OS (intermediary blw hardware and application programs)
  - Application Programs (software programs)
  - User



- Embeded Systems → with minimal or no direct user interface. e.g., Tv, Washing machine

- Resource manager → OS can be seen as resource manager.
- Control program → OS can be control program.

- Moore's law → number of transistors on an integrated circuit would double every ~18 months has held true.
  - ↓ co-founder of intel.

- Components of OS →
  - Kernal → (Linux Kernal) (Window NT Kernal) core part of OS that runs at all the time, managing system resources.
  - System daemons → After Kernal loads, system daemons starts. On Linux system, system process manages these. ↓ handles background tasks
  - System Programs → associated with OS but not the part of the Kernal.
  - Application program → Programs not directly associated with OS operation but run on the system.

- Sophistication → In 1988, US department of justice case against microsoft for building too much functionality with its OS.
  - ↓ more Knowledge
  - ↓ Web Browser was integral part

Input / output operations :- Transfer of data b/w computers and external devices or storage system.

ISR (Interrupt Service routine) → • special function in computer. • automatically runs when sth important needs immediate attention (Keyboard inputs)

- Mobile OS → includes Kernel + middleware frameworks, ...

by hardware or software.

Computer - System organization

- Interrupts → signals that indicate an event needs immediate attention
  - managing asynchronous events in
  - program initiates I/O operation → device driver loads register in device controller → device controller performs I/O operations
    - Data transfer to local buffer (ongoing)
    - No data transfer complete? ← Continue I/O operation
    - Yes → device controller generates an interrupt → CPU receives interrupt → CPU stops current execution → Jumps to ISR
      - ISR processes the interrupt.
      - update device driver with I/O operation results
      - device driver updates program
      - CPU resumes previous execution.

- Interrupt handling (continued on next page) → • Interrupt request → When an interrupt occurs, CPU stops its current task and transfer control to ISR.
  - ISR → After ISR executes, CPU resumes its previous task.

- Interrupt Management → • Table of pointers → Table of pointers (Interrupt Vector table) that holds addresses of various ISR. Interrupt number is used to index this table to locate appropriate ISR.
  - (Interrupt Vector table) ← also called
  - for managing interrupts efficiently

- Saving & restoring state → CPU saves current state before servicing the interrupt, and restore it afterward to ensure the previous computation smoothly.

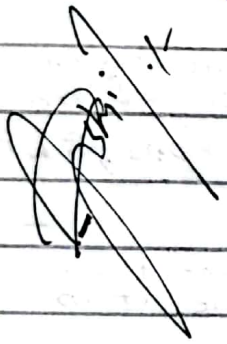
- Location of ISR addresses → Table of pointers is located in low memory.

- Interrupt Cycle → process of raising, catching and servicing an interrupt.

- Interrupt request line. → • CPU has this.
  • Wire that CPU senses after executing every instruction.

- Interrupt Vector → CPU detects interrupt request → reads interrupt number → uses it as a index into interrupt vector.
  ↓
  Table of addresses/ used to find ISR

- Interrupt handling → • Deferring interrupt handling → • finish a task without interuption So, system can wait to handle interuption until important job is done.
  • Efficient dispatching → send to right handler.
  • Multilevel Interupts → System should tell which interupts are more urgent than others and handles important firstly.



- CPU & Interupt Controller → Manages interupt handling
  CPU has 2 Interupt request lines.
  - Non-maskable Interrupt (NMI) → Reserved for critical events like unrecoverable memory errors.
  - Maskable Interrupt → Can be turned off by CPU & used by device controllers for general requests.

- Vector interupt mechanism → Instead of searching through everything to find out what caused interrupt, Computer uses shortcut (address) that tells exactly where to go.

- Interupt priorities → priorities basis.

Retain data even when power is removed.

← EEPOM → (Electrically erasable Programmable Read-only memory)

DDR → Double data rate / Type of DRam.

DRAM → Dynamic RAM.

- Main Memory (Random access memory - RAM)
  - rewritable memory
  - implemented using DRAM.
- Bootstrap Program → • First program executed on poweron which loads operating systems
  - Stored in EEPOM.
- Volatile Storage ↓ Lose content when power is turned off: (RAM)

← Non-Volatile Storage ↓ Retains content when power is turned on (SSDs, HDDs, EEPOM)
- Instruction Execution cycle → • Fetches instructions from memory and stores it in instruction register.
  - decodes instructions & executes it.
  - results stored back in memory.
- Memory Hierarchy → • Registers (Smallest & fastest)
  - Main memory (Ram) (larger & slower than registers)
  - Secondary storage (SDD, HDD)
  - Tertiary storage (Backup systems (usb))
- Storage devices → • Mechanical Storage → HDDs (larger, less cost, slow)

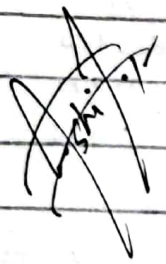
NVM ← Electrical storage → SSDs (smaller, faster, expensive)
- Cache memory → • Installed to improve performance by bridging speed gap b/w fast and slow storage components.

Virtual memory → Pretending more physical memory by temporarily moving data to hard drive.

↓ Beneficial for operation involving large amount of data, like disk I/O.

Phone ki storage converts to ram.

- I/O Structure. → Direct Memory access (DMA).



Method used to transfer data b/w device and memory without needing of the CPU.

Only one interrupt is generated per block to tell device driver that the task has completed

1.4 Operating System Operation

- Multiprogramming → • increase CPU utilization. • Keeps several processes in memory simultaneously.
- Multitasking → logical extension of multiprogramming
- CPU Scheduling → Process of determining which process or thread runs on the CPU, at any given time.

↓ maximizes CPU utilization.

- Physical memory → Actual hardware memory (RAM) installed in system.
- * Logical memory → • Address space presented to a process, abstracting actual physical memory.

Example:-

- EK program ky jis kay pass 4 GB memory hy. Ya logical memory.
- What if computer might have 2 GB actual physical Ram.

- Dual mode operations → • Kernal mode (device drivers) → • OS has full access to all hardware • System starts in this mode during boot-up
  - User mode → restricted mode where user application run.

• Mode bit in the hardware indicates current mode.

'0' for Kernal

'1' for user.

- GNU's Not Unix (GNU) → free software operating system.
  UNIX → OS.
  - System Calls → gateway for user programs to request services from os. etc
    certain instructions (I/O control)
  - Privileged instructions → can be controlled only in kernal mode. If user executes then hardware traps action and hands control to OS.
  - Multimode Operations → • More than 2 modes.
    • like an additional mode for Virtual machine manage (VMM)
    ↓
    more privileges than user but less than kernal.
  - Timer → • ensures that CPU is not monopolized by a user program.

1.11 Free and Open Source Operating Systems.

- provides access to source code, no cost use.
- provides access to source code but does not guarantee same licensing freedoms as free os.

- Richard Stallman's initiative (1984) → Developed a free, UNIX-compatible OS called GNU
  ↓
  freedom of use, not price.
- Four essential Freedoms →
  - Freely run program
  - study & change source code
  - give or sell copies
  - distribute copies.
- GNU Manifesto (1985) → • Published by Stallman.
  • Argues that all softwares should be free.

- Formation of free Software foundation → Established by Stallman.
  - Aims to encourage the use & development of free software.
- Copyleft → Licencing concept invented by Stallman.
  - Grant four essential freedoms.
- General Public License (GPL) →
  - Common copyleft licence under which software is released.
  - Requires that source code be distributed with binaries.
  - All copies must be released under same GPL licence.

- GNU/Linux →
  - Origin → In 1991, Linus Torvalds created UNIX-like kernal named as 'Linux' using GNU Tools. and invited contributors.
    - Linux was not free, in 1992, Linus released it under GPL, making it free and open source.
  - GNU/Linux refers to complete OS that combines Linux Kernel with GNU Tools.
- BSD UNIX (Berekely Software distribution) →
  - Free BSD → High performance + advance networking.
  - Net BSD → Simplicity
  - Open BSD → Security and code correctness
  - DragonFly BSD → HAMMER file system + optimization for multi-core processors.
  - Darwin → Core kernal of MacOS based on BSD UNIX and it open source.
- Solaris →
  - Commercial UNIX-based OS of Sun microsystems.
  - Was based on BSD UNIX but moved to AT&T's System V UNIX in 1991 as its base.
  - Oracle purched Sun.