UNIT IV · LESSON 01

Computer System & OS Overview

computer organisation · interrupts · I/O · OS functions · types · dual mode · system calls · OS structures · booting

30
questions in Unit IV
OS · Networks · Cloud

COMPONENTS COMPUTER SYSTEM

- CPU, main memory, I/O devices connected by a **system bus** (data, address, control lines)
- CPU registers: **PC** (next instruction), **IR**, MAR, MBR, **SP**, PSW (flags), general-purpose
- **Instruction cycle**: fetch → decode → execute (→ interrupt check)
- Von Neumann: program and data share one memory; Harvard: separate memories
- **Memory hierarchy**: registers → cache → RAM → SSD/disk → tape (faster & costlier going up)
- **Locality**: temporal (reuse soon) and spatial (nearby addresses) → caching works
- Effective access time = $h \cdot t_c + (1 - h)(t_c + t_m)$ (look-through)

3 METHODS I/O TECHNIQUES (CONTD.)

- **Spooling** (printer), **buffering** (single, double, circular), **caching**

EVENTS INTERRUPTS

- Signal causing the CPU to **suspend** the current program and run an **ISR** (interrupt service routine)
- **Hardware** interrupts (I/O complete, timer) vs **software** (trap / exception: divide by 0, system call)
- **Maskable** vs **non-maskable** (NMI: power failure)
- Steps: finish instruction → save PC & PSW → find ISR via **interrupt vector** → run ISR → restore state
- Multiple interrupts: **disable** during ISR (sequential) or **priority** (nested)

3 METHODS I/O TECHNIQUES

Technique	CPU involvement	Use
Programmed I/O	CPU busy-waits (polling)	simple, slow devices
Interrupt-driven I/O	CPU interrupted per word/char	keyboards
DMA	CPU only at start & end of block	disk, network, bulk data

- DMA controller steals bus cycles (**cycle stealing**) or uses burst mode

WHAT AN OS DOES — OS FUNCTIONS

- **Resource manager** and **extended machine** (convenient interface)

Function	Examples
Process management	create, schedule, synchronise, terminate
Memory management	allocate, protect, virtual memory
File management	files, directories, access control
I/O / device management	drivers, buffering, spooling
Secondary storage	free space, disk scheduling
Protection & security	authentication, access rights
User interface	CLI (shell), GUI, batch
Networking, accounting, error detection	

KERNEL VS USER — DUAL MODE & PROTECTION

- **Mode bit**: kernel (0) / user (1); **privileged instructions** only in kernel mode
- Privileged: I/O instructions, set timer, disable interrupts, modify page tables, halt
- Non-privileged: read clock, add, compare, generate a trap
- **Timer** interrupt stops a process from holding the CPU forever
- Memory protection: **base and limit registers**

EVOLUTION — TYPES OF OS

Type	Key idea
Serial / simple batch	jobs grouped; no interaction
Multiprogrammed batch	several jobs in memory, CPU switches on I/O → better utilisation
Time-sharing (multitasking)	short time slices → interactive response
Real-time	**deadlines**: hard (missile, pacemaker) / soft (video streaming)
Distributed	many networked computers appear as one
Network OS	share files/printers over a network
Multiprocessor (parallel)	SMP (symmetric) / AMP (asymmetric)
Embedded / mobile	limited resources (Android, VxWorks)
Clustered	nodes share storage; high availability

- Multiprogramming goal: **CPU utilisation**; time-sharing goal: **response time**

API TO KERNEL — SYSTEM CALLS

- Interface between process and OS; called through a **trap** (software interrupt), mode switches to kernel
- Parameters passed in **registers**, a **block/table** in memory, or the **stack**

API TO KERNEL SYSTEM CALLS (CONTD.)

Category	UNIX examples	Windows
Process control	fork, exec, exit, wait	CreateProcess, ExitProcess
File management	open, read, write, close	CreateFile, ReadFile
Device management	ioctl, read, write	SetConsoleMode
Information	getpid, alarm, sleep	GetCurrentProcess
Communication	pipe, shmget, mmap	CreatePipe
Protection	chmod, umask, chown	SetFileSecurity

- Programs use **APIs** (POSIX, Win32) that wrap system calls

START-UP BOOTING

- Power on → **BIOS/UEFI** firmware in ROM → **POST** → bootloader (MBR/GPT, GRUB) → load kernel → init/systemd → login
- **Bootstrap** program initialises registers, memory and devices
- MBR = first sector (**512 bytes**) of the disk

DESIGN OS STRUCTURES

Structure	Idea	Example
Simple / monolithic	whole OS in one kernel space program	MS-DOS, early UNIX, Linux (modular monolithic)
Layered	layer n uses only layer n−1; easy to debug, slower	THE (Dijkstra)
Microkernel	minimal kernel (IPC, scheduling, basic memory); services in user space	Mach, QNX, MINIX
Modular	loadable kernel modules	Linux, Solaris
Hybrid	mix	Windows NT, macOS (XNU)
Exokernel	kernel only multiplexes hardware	research

- Microkernel: more **reliable & extensible**, but **message-passing overhead**
- **Virtual machines**: hypervisor type 1 (bare metal) / type 2 (hosted)

TERMS KERNEL VS SHELL

- **Kernel**: core, always resident in memory
- **Shell**: command interpreter (bash, cmd)
- **Utilities**: system programs (compilers, editors, file managers)
- Context switch = save state of one process, load another (pure overhead)

Q → A RAPID RECALL

Program that manages hardware	operating system
Mode for privileged instructions	kernel (supervisor) mode
Transfers a block without CPU per word	DMA
Interrupt generated by software / error	trap (exception)
Goal of multiprogramming	maximise CPU utilisation
Goal of time sharing	minimise response time
OS with strict deadlines	hard real-time OS
fork() belongs to	process-control system calls
Structure with minimal kernel	microkernel
First program run at power-on	BIOS/UEFI bootstrap (in ROM)
Size of MBR	512 bytes

TRAP EXAM TRAPS

- Reading the clock is not privileged; **setting** the timer is
- A system call is **not** an ordinary procedure call — it switches mode via a trap
- Multiprogramming ≠ multiprocessing (one CPU vs many CPUs)
- Microkernel is slower due to IPC, not faster
- DMA still interrupts the CPU — once per **block**