Chapter 1: Interrupts

What is an OS? Program that controls the execution of application programs. Interface between applications & hardware.	Function of an operating system: Resource management & allocation, Controls execution of user programs, Providing services to system users	Components of an OS: CPU, RAM, I/O modules, System Bus	What is an instruction? Command to perform a specific task.
Components involved: Program counter, instruction register, accumulator, memory address register, memory buffer register	Instruction (fetch/execute) cycle: PC points to instruction in memory -> CPU fetch instruction that is being pointed -> Instruction is loaded into IR -> PC incremented to next instruction -> CPU interprets & execute the instruction.	# possible instructions = # Unique instructions with opcode bits = 2^(opcode bits)	Maximum directly addressabsle memory capacity = # Unique memory addresses = 2^(Address bits)
Size of data bus to use = number of bits in instruction format	What is an interrupt: Mechanism where the normal sequencing of instructions of CPU is interrupted to address a different task.	Polling: Wait for I/O Completion. CPU constantly ask I/O if it is done.	Types of interrupts: Program interrupt (Execution of illegal instruction, illegal access of memory space), Timer interrupt (Perform specific tasks on a regular basis), I/O interrupts (Signal completion of an I/O operation), Hardware failure (low battery, power failure, memory error)
Fetch execute instruction cycle with interrupt: CPU -> Fetch next instruction -> Execute instruction -> (No interrupt, back to CPU) -> Check for interrupt -> Initiate interrupt handler -> CPU	Fetch & execute stage is atomic and cannot be interrupted, Interrupt is served after the end of an execution stage	Multiple interrupts: Two approaches, Disable interrupts while interrupt is being processed, use a priority scheme	Memory hierachy: (Smaller/Faster) Register -> Cache -> RAM -> Secondary storage -> Tertiary storage (Larger/Slower)
Registers, cache are part of bios	L1 cache: Data cache & instruction cache, separation is to prevent overcrowding of either caches	L2 cache: Larger cache that stores both data & instructions	L3 cache: Shared by all cores, store data & instructions to be shared among all cores

Chapter 1: Interrupts (cont)

I/O techniques: Programmed I/O (CPU polling and then initiates data transfer between memory & I/O device when device is ready), Interrupt driven I/O (When I/O device is ready to transfer data, device sends an interrupt to CPU), Direct Memory Access (DMA) (Allows I/O devices to directly read & write to memory without continuous CPU involvement, improving data transfer efficiency)	Multiprocessor systems: Two or more processes working together in a single machine, all processors share computer memory, I/O devices, work in parallel to enhance performance & reliability.	Types of Multiprocessor systems: Symmetric multiprocessing (No one is in charge, processes are of similar capability), Asymmetric multiprocessing (One processor is the master, controlling the system & distributing tasks to other processors, the slaves.	Multicore systems: Chip multiprocessor, each core consists of all components of a CPU, OS allows for parallelism in multiprocessing environment, shares memory & I/O devices

Chapter 2: Multiprogramming, time sharing

What is a kernel: One program running at all times on the computer. Responsibilities: Device management, Scheduling, System calls & APIs, Protection & Fault tolerance, security	Kernel types: MicroKernel (Only the most essential services, high security but low performance), MonolithicKernel (Has everything, very efficient but not secure), HybridKernel (Mix of both approaches, essential services in kernel space, most other services + drivers in user space)	Modes of operation: User mode (Applications you run are limited in what they can do), Kernel mode (Full access to hardware & can execute any CPU instruction)	Uniprogramming: One program runs until completion before the next program starts, no 2 programs run at same time)
Multiprogramming: Multiple programs run concurrently to optimize CPU util, memory should be large for > 2 programs & allow context switch among all of them.	CPU util (%): sum(duration * CPU time : process) / total duration (uni), max duration (multi)	Memory util - multi (%): sum[(# Jobs left at t) * duration / memory capacity at t = 5m, 10m, 15m]/max duration	Elapsed time: sum of all duration for uni, max duration for muti

Chapter 2: Multiprogramming, time sharing (cont)

System throug-hput: # Jobs completed by a certain time.	Mean(avg) turnaround time: sum(t for each process to complete)/number of processes	Time sharing system: multiple interactive jobs, shared processor time, simultaneous access. (min user response time w time sharing, max cpu util w/o time sharing)	Time slicing techniques: system clock generates time interrupts at a rate of t, at each interrupt, OS takes back control from current user program, saves state and assign processor to another user program (ISR), state of current user program is saved to disk and state of next user program is loaded to MM for execution.
Android OS is a Linux based OS system,	Android Application framework:Ac-tivity Manager: An activity represents a single screen with a user Interface. Responsible for starting, stopping, and resuming activities. Window Manager: Surface manager that manages frame buffering and low-level drawing. Manage top-level window's look and behavior. Package Manager: Installs and removes applic-ations. Telephony Manager: Allows interaction with phone, SMS, and MMS services.	Content Providers: Manage data that need to be shared between applications such as contacts, calendar info, which are stored in SQL database. Resource Manager: Manages non-code resources, such as strings, graphics, layout files. View System: Provides the user interface (UI) that displays information and responds to user actions, Lists, grids, text boxes, buttons, etc. Location Manager: Allows developers to tap into location-based services, whether by GPS, cell tower IDs, or local Wi-Fi databases. Notification Manager: Manages events, such as arriving messages and appointments. XMPP: Provides standa-rdized messaging functions between applications	Android Activity: UI screen, Android Run time (ART): VM for android, bytecode to machine code.

Chapter 2: Multiprogramming, time sharing (cont)

Android system libraries: Surface Manager (display management) OpenGL (graphics engine) Media Framework (video/audio streaming) SQLite (relational database engine) Browser Engine Bionic LibC (system C library)

Differences between android OS & Linux OS: Power management (Power collapse, component level power management, Wakelocks), IPC (Inter process communication)

Chapter 3: Process description and control

What is a program: Set of instructions that the computer can execute. Can have multiple instances.

What is a process: Task that you do on your computer, an instance of a program. Components: Executable program, associated data, execution context.

Process image: current activity state reflected by CPU registers (PC, etc), Progam code, Data Sections (Global constants & variables), Stack, Heap

Process elements: Identifier, State, Priority, Program Counter, Memory pointers, Context data, I/O States info, Accounting info

Process control block (PCB): Data structure created & managed by OS, full of info about each process. New process created -> OS updates PCB with all process details, key tool that allows support for multiple processes.

Dispatcher: Small OS that follows a scheduling policy, handles context switching for CPU, spends a lot of time saving instructions, load new instructions into CPU from PCB. Trace: Detailed log of what a process does

Process Creation:App launch, OS start a process to do task, some process start in background w/o you directly interacting with them.Some processes are started by other processes. Parent process may wait for child to finish or continue concurrently. e.g Process A spawns other process & so forth forming a tree.

Linux system calls: Fork(): Create copy of current process, exec():Execute a program, wait():Wait for child process to finish and change state, kill(): Send a signal to terminate a process, pipe(): IPC

Chapter 3: Process description and control (cont)

Two state (Running/Not Running) process model: Process created and move to queue -> Process stay till CPU is ready -> Dispatcher move a process from not running to running state -> Process is temporarily stopped, move back to not running state due to an interrupt -> If not, Process finish execution & leave the system.	Five state process model: New, ready, running, blocked, exit	five State transitions: New process is admitted to ready queue -> dispatcher assign process to be run by CPU -> If time out goes back to ready state -> If event wait (I/O completion), process admitted to block queue, event occurs, remove from block queue & move to ready queue to await dispatcher -> If process completes task, exits	To handle different types of events, there can be multiple block queues.
How does OS decide which process run when & how long: Scheduling policy (Dispatcher choose from ready queue for CPU to run), Dispatcher, Clock (timeout of process)	Solutions for limited available memory: Virtual memory (Move part of a process from RAM to disk), Swapping (move some blocked processes entirely out to disk)	Suspended queue: Process that are moved to disk is placed in a suspend queue, OS can decide to bring suspended processes back to main memory.	Transition states: Same as five state processes but for processes in the block/ready queue, can be suspended and moved to disk (block/suspend & ready/suspend) if there are any memory constraints in RAM. Suspended processes can be moved back to the ready queue when there is enough memory to hold them or when an event they are waiting for occurs.
Information required by OS to control processes and manage resources: Memory tables, I/O tables, File tables, Process tables	Memory tables: RAM, Secondary memory (HDD/SDD), Virtual Memory. I/O Tables: Used by OS to manage I/O devices. File tables: Provide info about the existence of files location on secondary memory, current status and other attributes.	Process tables: Process location (Where a process and its data is located in memory), Process Attributes (# Attributes used by OS for control)	Process list structures: OS has a list of PCBs in each queue (running, ready, block) it use to keep track of all processes.

Chapter 3: Process description and control (cont)

Transitions: When a new process is created -> unique identifier assigned -> Memory space allocated for process image -> PCB is initialised -> set up linkage between parent and child processes -> Update all other data structures	Process/Context switching: Necessary for multitasking. Occurs from a clock interrupt, I/O interrupt, Traps (Errors generated with running process), Supervisor call (User process calls for I/O operation & is blocked)	Mode switching: User Mode (Where user applications run, Less privileged), Kernel Mode (Where the OS runs, Highly privileged)	Costs involved for mode switching: Save state of current process, switch cpu to kernel mode to execute system call, restore state of process once system call is completed and switch back to user mode.
Mode switch with process switch: Save context of processor -> Update PCB -> Move the PCB to the appropriate queue -> Select another process for execution -> Update the PCB of the process selected -> Update memory management data structures for address translation -> Restore the context of the processor to that which existed at the time the selected process was last switched out	Mode switch w/o process switch: Interrupt is pending -> Save the context (PC, processor registers, stack pointer) into the PCB of the current process -> Sets the program counter to the starting address of an interrupt handler program -> Switches from user mode to kernel mode	Traditional OS: All user processes rely on a single monolithic kernel	Process switching functions: OS function executes within user processes. Mode switch w/o process switch in a user program (fopen()), Process switching function takes place when a process switches its state.

process based OS: Processes are assigned different priorities to be scheduled for running, good for multi processor env