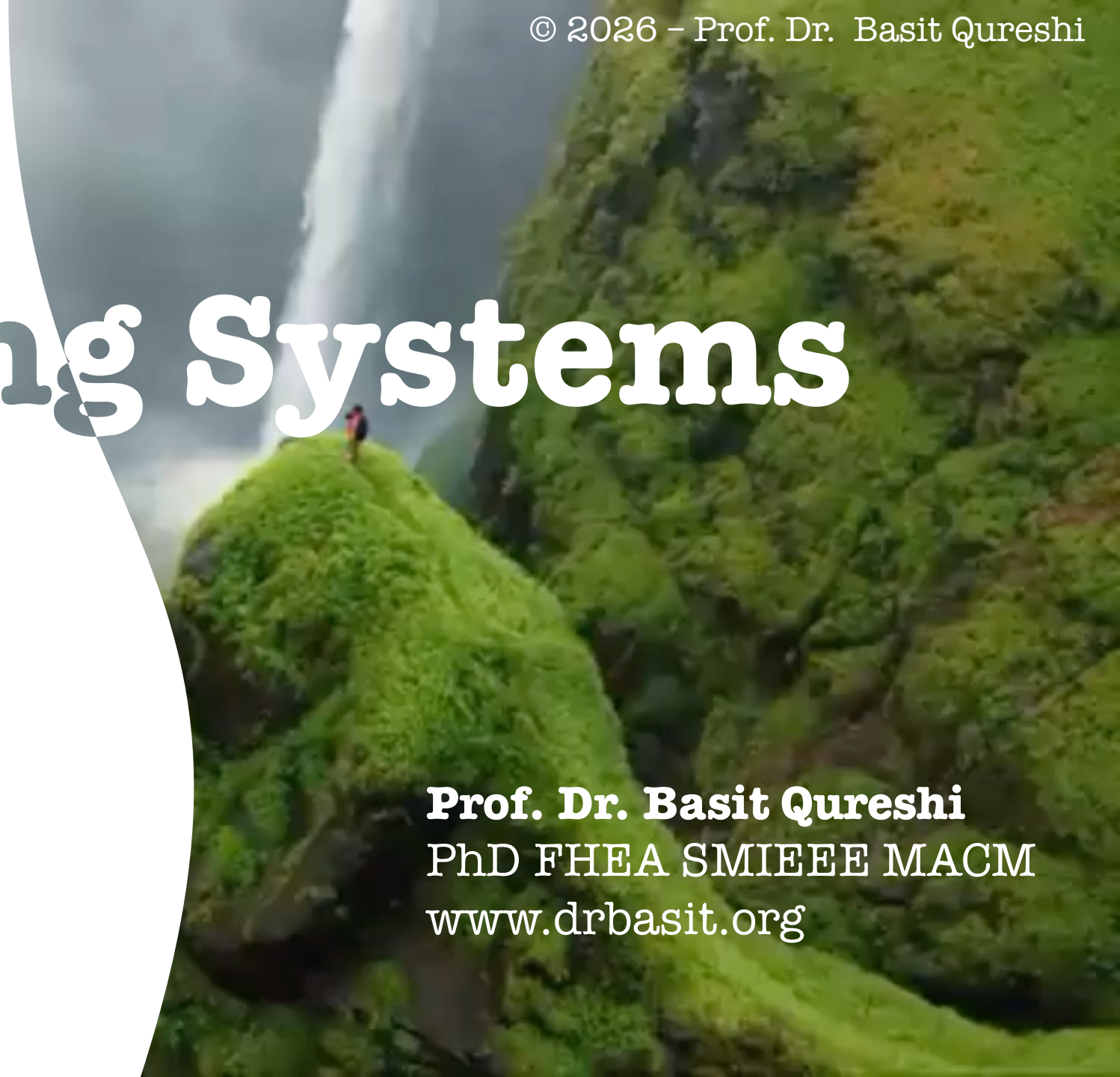
CS330 Operating Systems



BASICS

Prof. Dr. Basit Qureshi
PhD FHEA SMIEEE MACM
www.drbasit.org

جامعة الأمير سلطان
PRINCE SULTAN UNIVERSITY

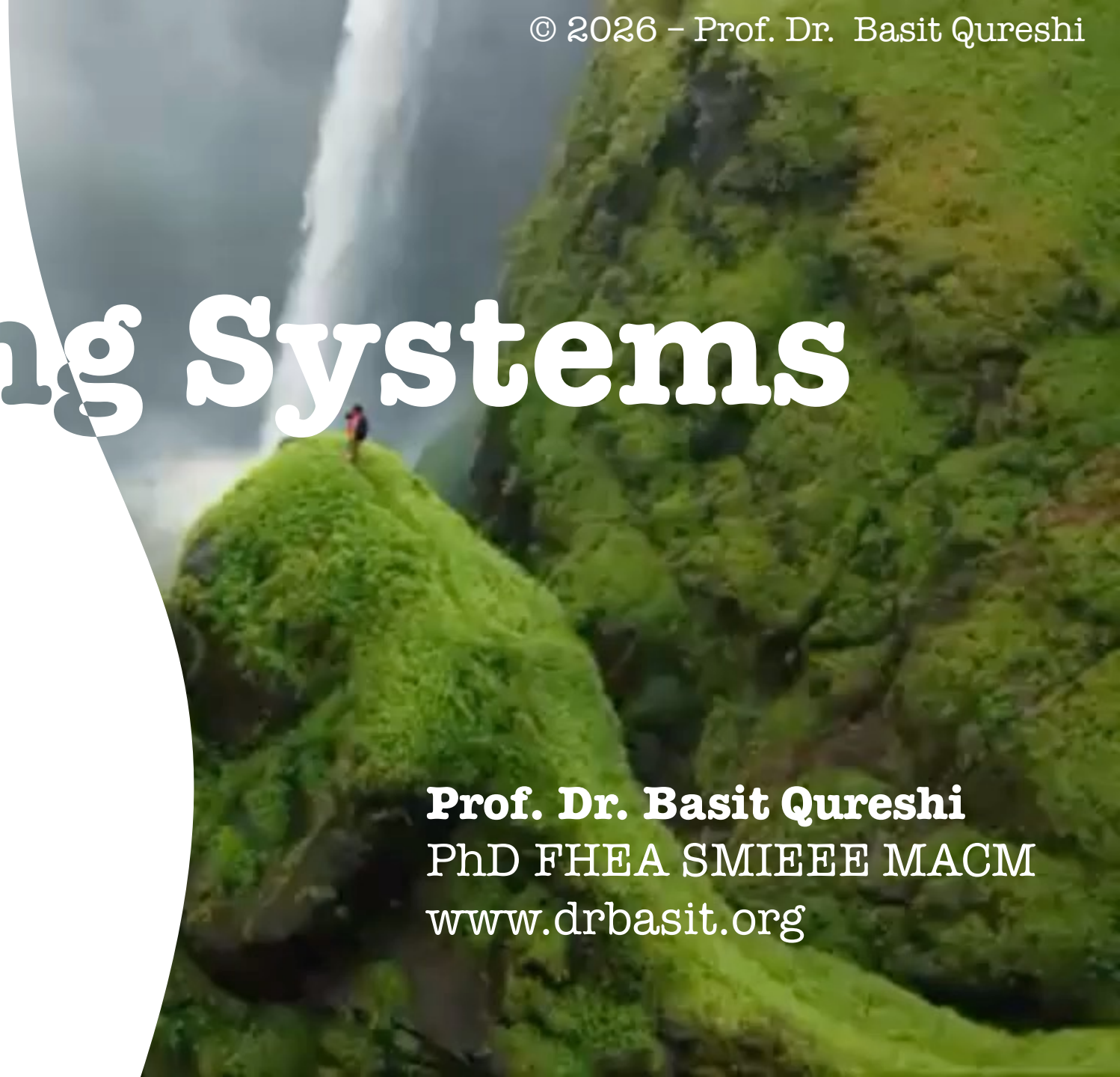
TOPICS

- Computer System Organization
- Interrupts
- OS structures and operations

Credit:
These notes are extracted from the following resources:

- Silberschatz, Galvin and Gagne, *Operating System Concepts*, 10 ed, Wiley, 2018
- Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau, *Operating Systems: Three Easy Pieces*, CreateSpace, 2018
- Robert Love, *Linux Kernel Development*, Addison-Wesley, 2010
- William Shotts, *The Linux command line*, No Starch Press, 2026

CS330 Operating Systems



COMPUTER SYSTEM ORGANIZATION

Prof. Dr. Basit Qureshi
PhD FHEA SMIEEE MACM
www.drbasit.org

COMPUTER SYSTEM ORGANIZATION

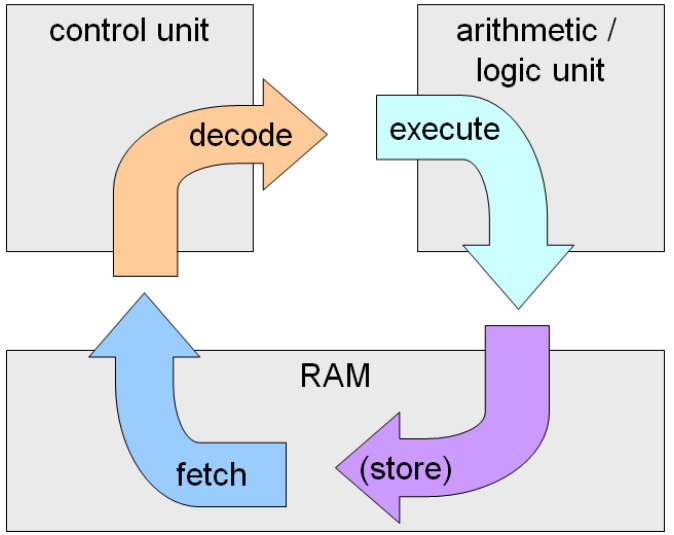
CPU Execution cycle:

The CPU execution cycle—often called the **fetch-decode-execute (FDE) cycle**—is the continuous loop a computer processor uses to run program instructions. It has three main steps:

Fetching the instruction from memory

Decoding what it means, and

Executing the action



This cycle repeats billions of times per second.

https://dev.to/farhadrahimiklie/inside-the-cpu-a-complete-guide-to-the-instruction-execution-cycle-and-how-data-is-retrieved-from-4j2c

COMPUTER SYSTEM ORGANIZATION

CPU components:

Registers:

- **PC (Program Counter)** – holds the address of the next instruction.
- **IR (Instruction Register)** – holds the instruction being decoded/executed.
- **MAR (Memory Address Register)** – holds memory addresses.
- **MDR (Memory Data Register)** – holds data being transferred to/from memory.
- **General Registers** – AX, BX, RAX, RBX, etc., depending on architecture.

ALU (Arithmetic Logic Unit):

- Performs math (add, subtract, multiply) and logical operations (AND, OR, XOR).

Control Unit

- Directs the entire execution cycle, orchestrating fetch/decode/execute.

Cache

- Very fast memory layers (L1, L2, L3) that speed up instructions and data access.

RAM (Main Memory)

- Where instructions and program data live while a program runs

COMPUTER SYSTEM ORGANIZATION

Instruction cycle:

Fetch: Fetching means retrieving the next instruction from memory.

1. **PC → MAR**
   The Program Counter contains the address of the next instruction. That address is copied into the Memory Address Register.
2. **CPU requests memory read from RAM (or cache)**
   The address is placed on the address bus.
3. **Memory returns the instruction → MDR**
   Data from memory goes into the Memory Data Register.
4. **MDR → IR**
   The fetched instruction is placed in the **Instruction Register**.
5. **PC increments**
   PC = PC + size_of_instruction (typically +1, +4, etc.).

At this point, the CPU has loaded the instruction and is ready to decode.

COMPUTER SYSTEM ORGANIZATION

Instruction cycle:

Decode: The CPU's Control Unit **interprets** what the instruction means.

Examples:

- If the instruction is **ADD R1, R2**, the control unit activates the **ALU**.
- If the instruction is **LOAD R1, [0x1000],** the CPU prepares for a **memory** read.
- If the instruction is a **branch**, the control unit checks **conditions** and prepares to modify the **PC**.

The CPU breaks the instruction into:

- **Opcode** (operation: add, move, jump)
- **Operands** (register numbers, memory addresses, constants)
- **Addressing modes** (direct, indirect, immediate, indexed)

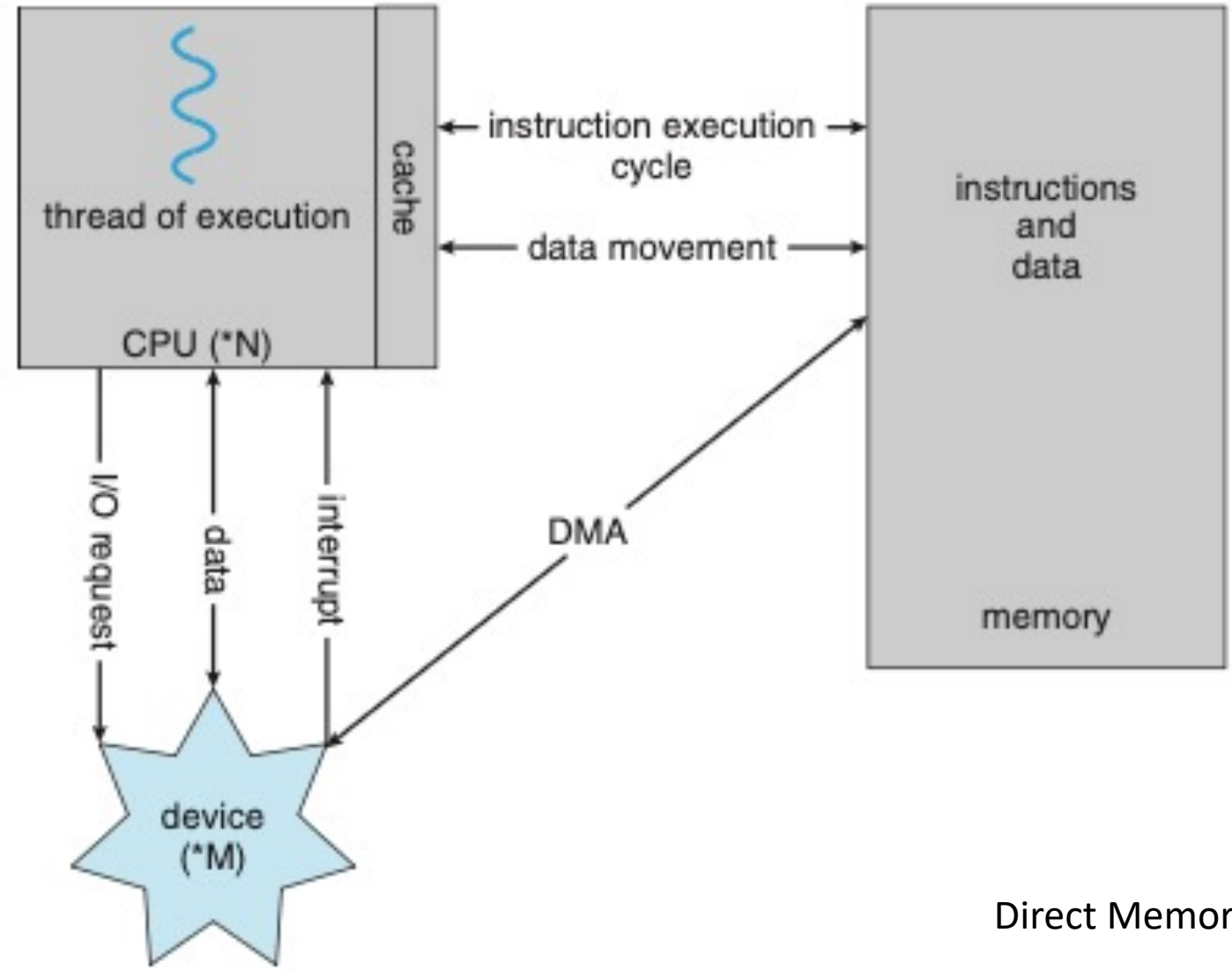
No data processing happens yet—only interpretation.

COMPUTER SYSTEM ORGANIZATION

Instruction cycle:

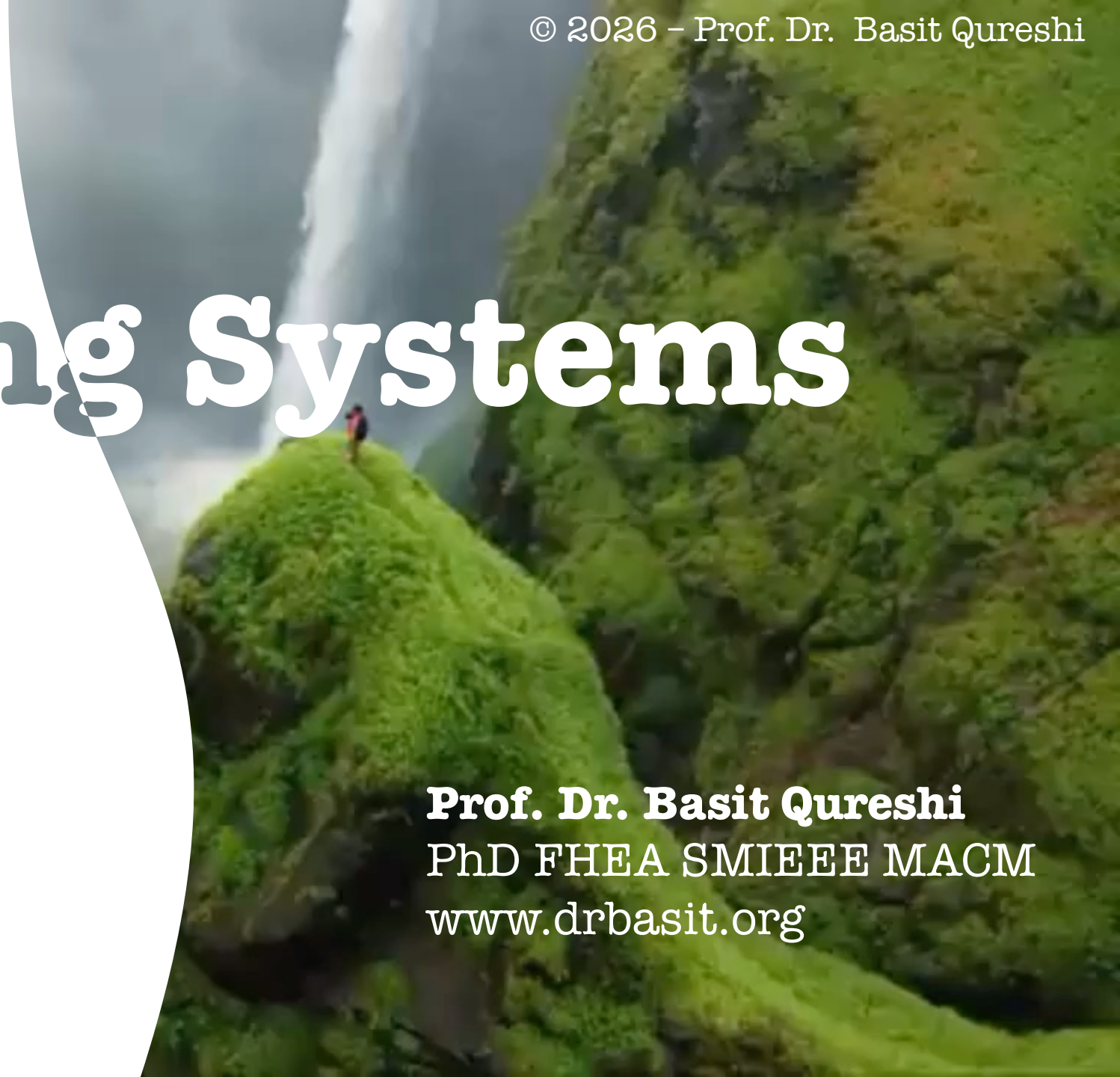
Execute: The CPU performs the action.

- The ALU performs the operation. Result is stored in a register (e.g., R1 = R1 + R2).
- Memory READ
  - CPU gets data from RAM. The effective memory address is calculated (base + offset).
  - The address is placed into MAR.
  - Data from RAM moves over the data bus into MDR -> Register
- Memory WRITE
  - CPU puts data into MDR.
  - CPU loads address into MAR.
  - CPU signals memory to write MDR to that RAM address.
- BRANCH
  - If the condition is true: PC is updated to new address
- I/O: Control signal activates IO controller



Direct Memory Access (DMA)

CS330 Operating Systems



INTERRUPTS

Prof. Dr. Basit Qureshi
PhD FHEA SMIEEE MACM
www.drbasit.org

INTERRUPTS

CPU Interrupt:

A CPU interrupt is a **signal** sent to the computer's processor that temporarily **stops** the current program. It forces the CPU to handle a **high-priority event** right away, like a *key press, mouse movement, or data arriving from a network*. Once finished, the CPU **goes back** to its original task

INTERRUPTS

Types of Interrupts

- **Hardware Interrupts**: Sent by physical devices (such as keyboards, hard drives, or timers) when they need the processor to read or send data
- **Software Interrupts (Exceptions)**: Triggered inside the processor by running programs, often to request a service from the operating system or because an error occurred (like dividing a number by zero)
- **Non-maskable interrupts** deal with critical emergencies (like a sudden hardware failure) and must be handled immediately

INTERRUPTS

Common functions of an Interrupt:

- Interrupts transfer control to a new program called *the interrupt service routine* (ISR) generally, through the ***interrupt vector,*** which contains the addresses of all the service routines (ISRs).
- Interrupt architecture must save the address of the interrupted instruction.
- Generally operating systems are ***interrupt driven-*** OS waits for an event.
- Events are signaled by the occurrence of an interrupt or a trap.
- A ***trap*** / ***Exception*** is a software-generated interrupt caused either by an error or a user request.

INTERRUPTS

Interrupt handling:

1. Save the State:

The operating system preserves the state of the CPU by storing registers and the program counter.

2. Find the handler:

The OS determines which type of interrupt has occurred by one of two methods:

- *Polling*- querying of all I/O devices, detect which requested services interrupted the service.
- *vectored interrupt system* (Table of addresses)

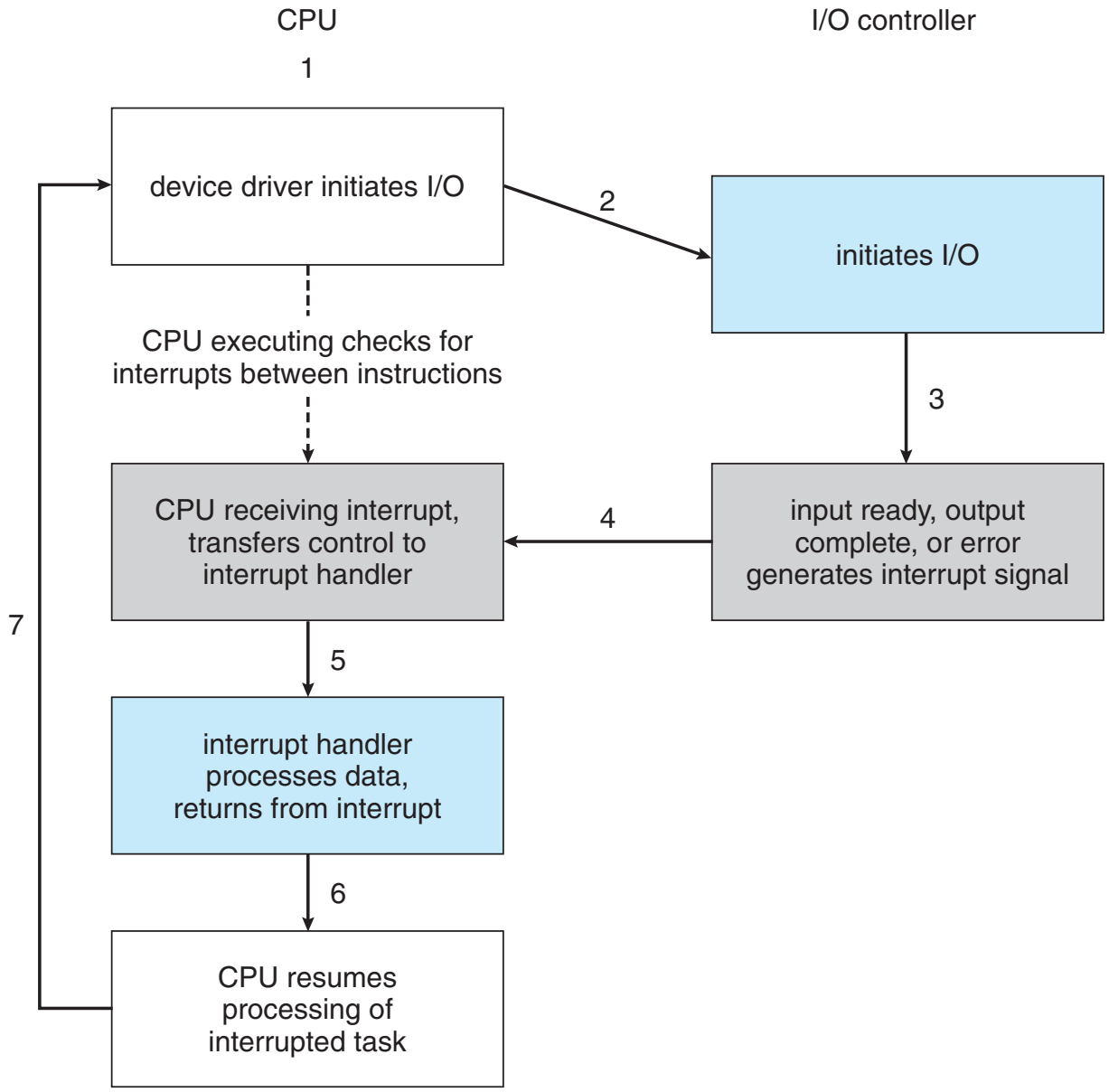
3. Execute Routine:

Separate segments of code determine what action should be taken for each type of interrupt. The OS executes the sequence of commands associated with the given interrupt.

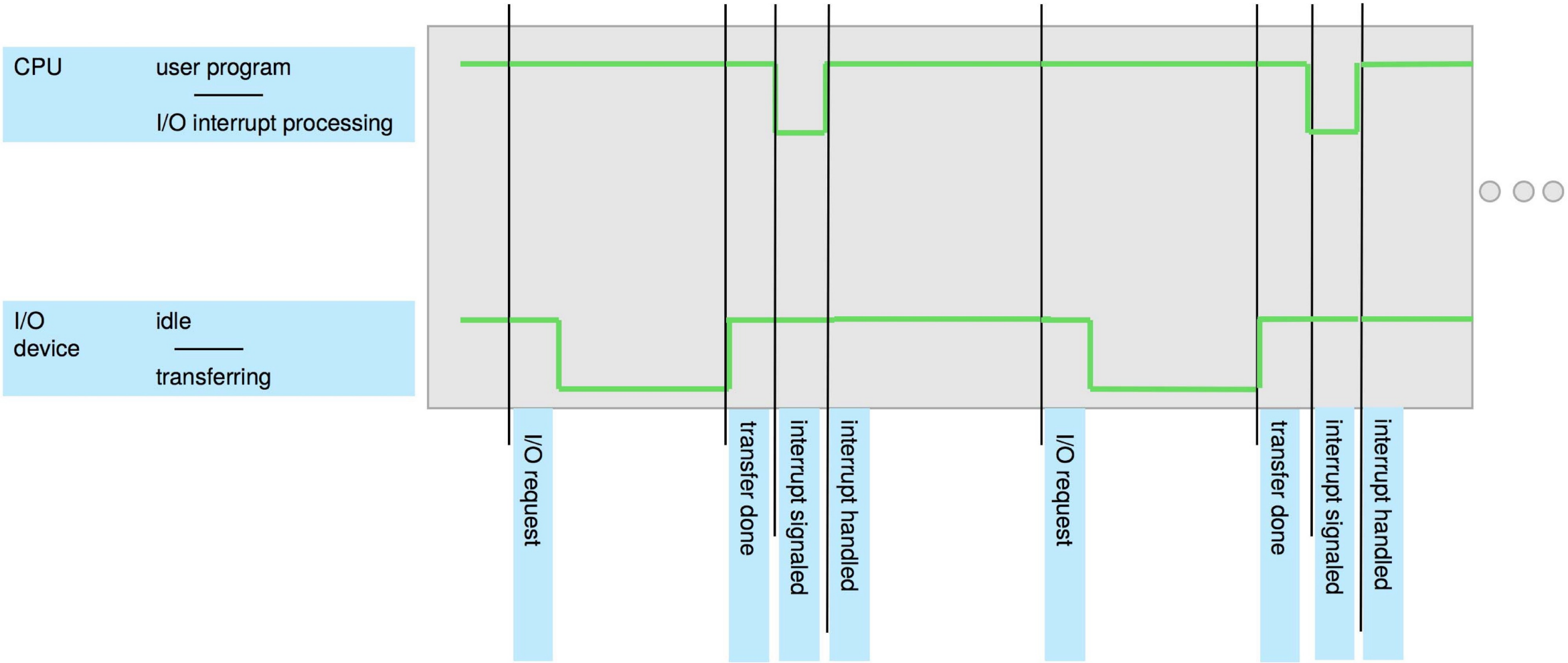
4. Resume:

The OS recovers the stored information from the original process and continues execution.

INTERRUPTS


CPU
I/O controller
1
device driver initiates I/O
2
initiates I/O
CPU executing checks for interrupts between instructions
3
CPU receiving interrupt, transfers control to interrupt handler
4
input ready, output complete, or error generates interrupt signal
7
5
interrupt handler processes data, returns from interrupt
6
CPU resumes processing of interrupted task

INTERRUPTS

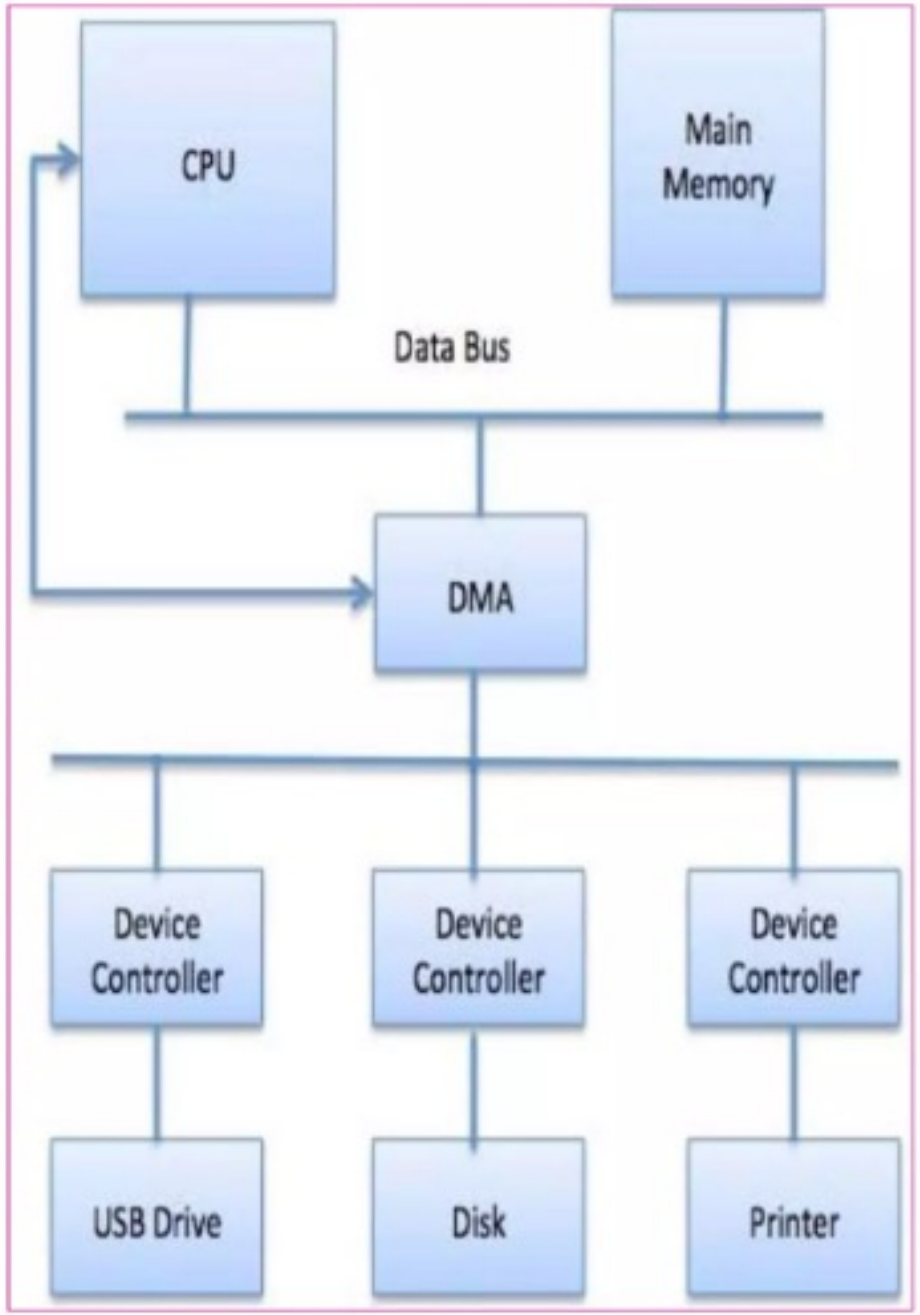

CPU
user program
I/O interrupt processing
I/O device
idle
transferring
I/O request
transfer done
interrupt signaled
interrupt handled
I/O request
transfer done
interrupt signaled
interrupt handled

DIRECT MEMORY ACCESS STRUCTURE

- Used for high-speed I/O devices able to transmit information at close to memory speeds
- Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention
- Only one interrupt is generated per block, rather than the one interrupt per byte

DIRECT MEMORY ACCESS (DMA) STRUCTURE

- Slow devices like keyboards will generate an interrupt to the main CPU after each byte is transferred. If a fast device such as a disk generated an interrupt for each byte, the operating system would spend most of its time handling these interrupts. So a typical computer uses ***direct memory access (DMA)*** hardware to reduce this overhead.
- Direct Memory Access (DMA) means CPU grants I/O module authority to read from or write to memory without involvement. DMA module itself controls exchange of data between main memory and the I/O device. CPU is only involved at the beginning and end of the transfer and interrupted only after entire block has been transferred.



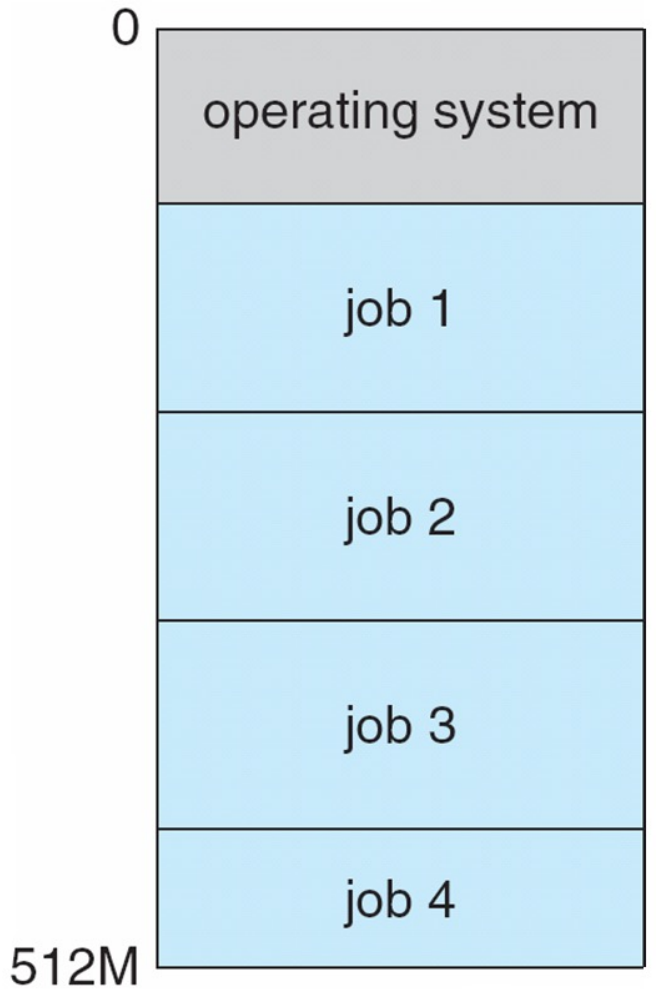
CS330 Operating Systems

OS STRUCTURES AND OPERATIONS

Prof. Dr. Basit Qureshi
PhD FHEA SMIEEE MACM
www.drbasit.org

OPERATING SYSTEM STRUCTURE

- Different OSs **vary** greatly internally but have **many common features**.
- First computer systems were **batch systems**. i.e. low CPU utilization.
  - **Multiprogramming** needed for efficiency
    - Single user cannot keep CPU and I/O devices busy at all times
    - Multiprogramming organizes jobs (code and data) so CPU always has one to execute
      - One job selected and run via **job scheduling**
      - When it has to **wait** (for I/O for example), OS switches to another job
  - **Timesharing (multitasking),** many users can share the computer. It is logical extension in which CPU switches jobs so frequently that **users can interact with each job** while it is running, creating **interactive** computing.

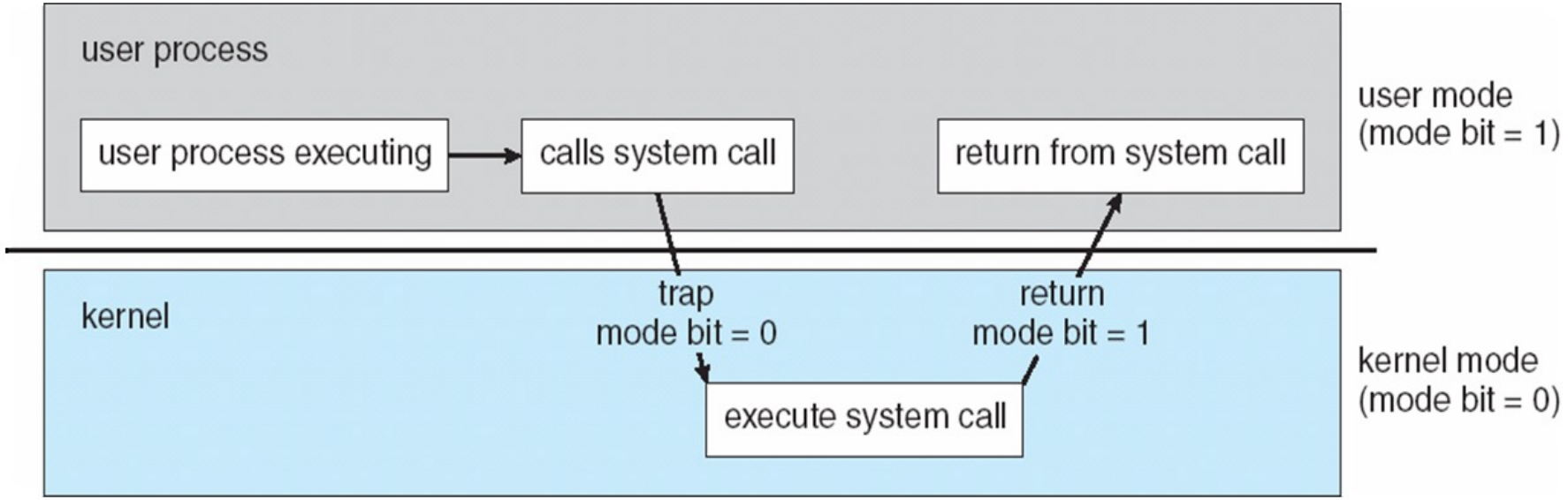


OPERATING-SYSTEM OPERATIONS

- **Dual-mode operation** allows OS to protect itself and other system components
  - **User mode –** computer system is executing on behalf of a user app
  - **Kernel mode** (also *kernel/system/admin/supervisor mode*) protects the OS when it is running – execution done on behalf of operating system.
  - **Mode bit** provided by hardware (kernel (0) or user (1))
    - Provides ability to distinguish when system is running user code or kernel code
    - Some instructions whose execution can harm the system are designated as **privileged**, only executable in kernel mode
    - **System call** changes mode to kernel, return from call resets it to user

TRANSITION FROM USER TO KERNEL MODE

- The Operating System must protect the CPU from being taken over by a user program (e.g. in an infinite loop) -> Timer
  - Set **interrupt** after specific period
  - Operating system **decrements counter**
  - When counter zero generate an interrupt
  - Set up before scheduling process to regain control or terminate program that exceeds allotted time



PROCESS MANAGEMENT

- **A process is a program in execution**. It is a unit of work within the system. Program is a *passive entity*, process is an ***active entity***.
- Process needs resources to accomplish its task
  - CPU, memory, I/O, files
  - Initialization data
- Process termination requires **reclaim** of any reusable resources
- Typically system has many processes, some user, some operating system running concurrently on one or more CPUs
  - Concurrency by multiplexing the CPUs among the processes / threads

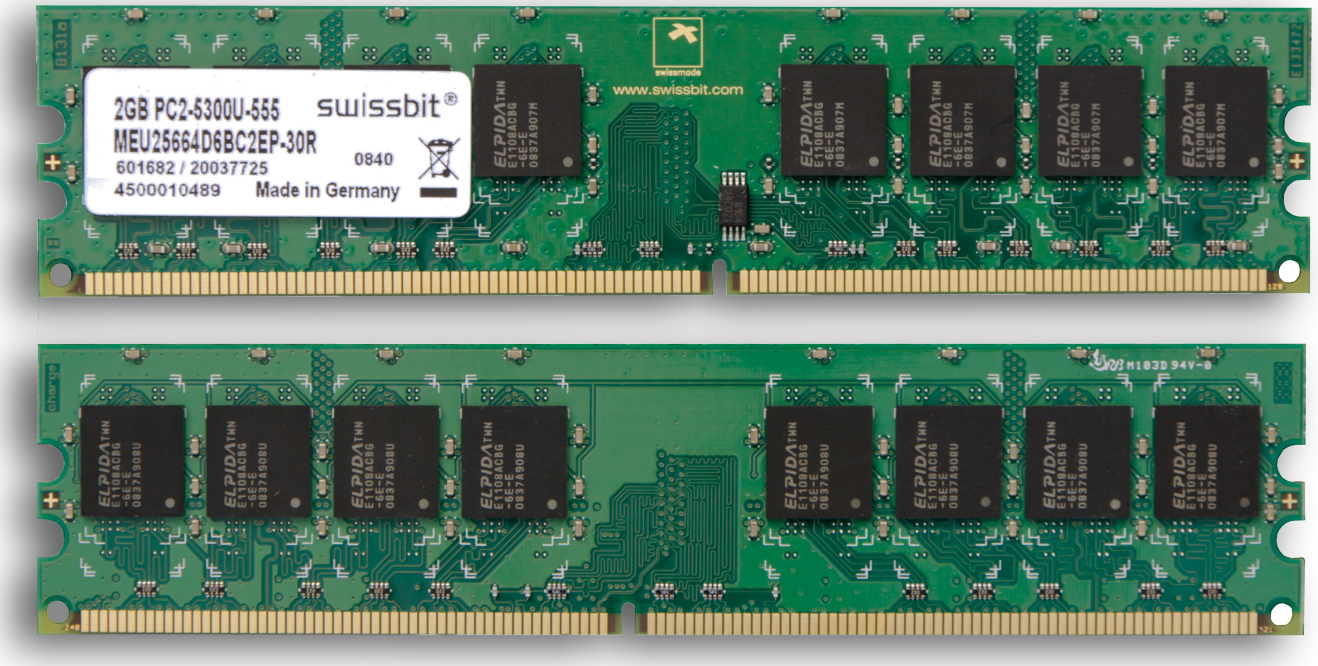
PROCESS MANAGEMENT ACTIVITIES

The operating system is responsible for the following activities in connection with process management:

- **Creating** and **deleting** both **user** and **system** processes
- **Suspending** and **resuming** processes
- Providing mechanisms for process **synchronization**
- Providing mechanisms for process **communication**
- Providing mechanisms for **deadlock handling**

MEMORY MANAGEMENT

- A program's instructions and data must be in the **main memory** for it to be executed by the CPU.
- To improve speed of execution and CPU utilization, many programs must reside in the **memory** *at the same time.*



MEMORY MANAGEMENT ACTIVITIES

Memory management activities

- Keeping track of which parts of memory are currently being used and by whom
- Deciding which processes (or parts) and data to move *into* and *out* of memory
- Allocating and deallocating memory space as needed

STORAGE MANAGEMENT

Users do not see bits and bytes. OS enables them to see data/programs as **files** and **folders**.

A **file** is a collection of related information e.g. program files, data files.

OS file management activities include

- Creating and deleting files and directories
- Primitives to manipulate files and dirs
- Mapping files onto secondary storage
- Backup files onto stable (non-volatile) storage media

STORAGE MANAGEMENT

- Mass-Storage Management
  - **Secondary storage** – hard disks
  - Need hard disks as main memory is too small and volatile.
  - Programs and data are loaded from the hard disk and results stored back on the hard disk.

- OS disk management activities
  - Free-space management
  - Storage allocation
  - Disk scheduling
- Some storage need not be fast
  - **Tertiary storage** includes tape drives, CDs, DVDs

PERFORMANCE OF VARIOUS LEVELS OF STORAGE

Level	1	2	3	4	5
Name	registers	cache	main memory	solid state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25 - 0.5	0.5 - 25	80 - 250	25,000 - 50,000	5,000,000
Bandwidth (MB/sec)	20,000 - 100,000	5,000 - 10,000	1,000 - 5,000	500	20 - 150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk or tape

- Movement between levels of storage hierarchy can be explicit or implicit

SUMMARY

- A OS manages all the **physical components** so that software programs *don't have to worry about* how the hardware works.
- The CPU execution cycle—often called the **fetch-decode-execute (FDE) cycle**—is the continuous loop a computer processor uses to run program instructions.
- A CPU interrupt is a **signal** sent to the computer's processor that temporarily **stops** the current program. It forces the CPU to handle a **high-priority event** right away.
- A OS **manages** hardware resources (processor, memory and storage).