Justine Andrie Abella, BSCS 4  
  
**1. Computer System Operation**

**Definition & Overview**

A modern computer system comprises a CPU, memory, and peripheral device controllers, all interconnected via a system bus. The CPU and I/O devices can execute concurrently, mediated by interrupts and data transfer mechanisms. [Purdue Engineering+2UIC Computer Science+2](https://engineering.purdue.edu/~ebertd/469/notes/EE469-ch2.pdf?utm_source=chatgpt.com)

Key points:

* Each **device controller** manages a specific I/O device and has a local buffer and registers. The CPU communicates with device controllers via the system bus to issue commands or transfer data. [faculty.berea.edu+1](https://faculty.berea.edu/faculty/pearcej/CSC325/ch2.htm?utm_source=chatgpt.com)
* I/O operations are decoupled from CPU execution: while an I/O is in progress, the CPU can perform other computations. [Purdue Engineering+2users.soe.ucsc.edu+2](https://engineering.purdue.edu/~ebertd/469/notes/EE469-ch2.pdf?utm_source=chatgpt.com)
* The system is typically **interrupt-driven**: when a device completes an I/O operation (or needs attention), it signals an interrupt to the CPU, which causes the CPU to pause what it’s doing and handle the interrupt. [faculty.berea.edu+2Purdue Engineering+2](https://faculty.berea.edu/faculty/pearcej/CSC325/ch2.htm?utm_source=chatgpt.com)
* The CPU must save its current state (registers, program counter) before switching to handle an interrupt; after servicing, it restores the state and continues. [Purdue Engineering+2faculty.berea.edu+2](https://engineering.purdue.edu/~ebertd/469/notes/EE469-ch2.pdf?utm_source=chatgpt.com)

Additionally, the system often supports **multiprogramming**, time-sharing, and virtualization, all of which require managing multiple processes, context switching, and ensuring safe isolation between them (which is supported by protection mechanisms discussed later).

**2. Interrupts**

**What Is an Interrupt?**

An **interrupt** is a mechanism by which a hardware device or software condition can request attention by the CPU, causing the CPU to suspend its current execution and jump to an interrupt handler (also called an interrupt service routine or ISR). [users.soe.ucsc.edu+2Purdue Engineering+2](https://users.soe.ucsc.edu/~sbrandt/courses/Spring01/111/slides/mod2.1.pdf?utm_source=chatgpt.com)

Interrupts can be:

* **Hardware interrupts** (asynchronous): generated by external events, e.g. I/O devices signaling completion, timers, etc. [Purdue Engineering+2UIC Computer Science+2](https://engineering.purdue.edu/~ebertd/469/notes/EE469-ch2.pdf?utm_source=chatgpt.com)
* **Software interrupts** (synchronous), also called **traps** or **exceptions**: generated by the CPU itself (e.g. division by zero, illegal instruction, system call) to request OS intervention. [faculty.berea.edu+2users.soe.ucsc.edu+2](https://faculty.berea.edu/faculty/pearcej/CSC325/ch2.htm?utm_source=chatgpt.com)

**Interrupt Handling Steps**

1. **Interrupt detection**: the interrupt line is asserted; the CPU hardware detects it.
2. **State saving**: before switching context, the CPU saves its current registers, program counter, and status.
3. **Switch to kernel/monitor mode**: the CPU switches from user mode to privileged (kernel) mode so it can safely execute OS code.
4. **Identify interrupt source**: using polling (asking devices) or a vectored interrupt system (device supplies an interrupt vector). [faculty.berea.edu+2Purdue Engineering+2](https://faculty.berea.edu/faculty/pearcej/CSC325/ch2.htm?utm_source=chatgpt.com)
5. **Invoke the appropriate handler**: the OS executes the corresponding interrupt service routine.
6. **Restore state & resume**: after completion, the CPU restores its saved state and resumes the interrupted program.

Other considerations:

* **Nested interrupts / priority**: the system may support prioritizing interrupts such that higher priority interrupts can preempt lower ones. [UIC Computer Science+1](https://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/13_IOSystems.html?utm_source=chatgpt.com)
* **Maskable vs non-maskable interrupts**: some interrupts (e.g. for critical hardware failure) can’t be disabled. [UIC Computer Science](https://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/13_IOSystems.html?utm_source=chatgpt.com)
* **Disable/enable interrupts during handling**: during handling of one interrupt, the system may disable further interrupts (or lower priority ones) to avoid losing or corrupting state. [Purdue Engineering+2users.soe.ucsc.edu+2](https://engineering.purdue.edu/~ebertd/469/notes/EE469-ch2.pdf?utm_source=chatgpt.com)

**3. Direct Memory Access (DMA) Structure**

**Motivation & Concept**

In simple interrupt-driven I/O, the CPU may need to be interrupted for *each* word (or byte) transferred. This imposes high overhead when dealing with high-speed devices. DMA (Direct Memory Access) is a hardware mechanism that allows a device controller to transfer a block of data directly between its local buffer and main memory without continuous CPU involvement. [faculty.berea.edu+2Purdue Engineering+2](https://faculty.berea.edu/faculty/pearcej/CSC325/ch2.htm?utm_source=chatgpt.com)

With DMA:

* The CPU sets up the transfer by programming the DMA controller with the source, destination, and block size.
* The DMA controller takes over the data transfer, while the CPU proceeds with other tasks.
* Only two interrupts occur: one at the start (optional) and one upon completion of the block transfer. [faculty.berea.edu+2Purdue Engineering+2](https://faculty.berea.edu/faculty/pearcej/CSC325/ch2.htm?utm_source=chatgpt.com)

This reduces CPU overhead and improves throughput. [Purdue Engineering+1](https://engineering.purdue.edu/~ebertd/469/notes/EE469-ch2.pdf?utm_source=chatgpt.com)

**DMA Structure & Variants**

* A dedicated **DMA controller** is used (sometimes integrated into chipset).
* **Third-party DMA**: uses a separate DMA controller that interacts with CPU and memory. [Wikipedia](https://en.wikipedia.org/wiki/Direct_memory_access?utm_source=chatgpt.com)
* Modern systems also employ **IOMMU (Input–Output Memory Management Unit)** to map device-visible addresses to physical memory, protecting against errant or malicious device memory accesses. [Wikipedia](https://en.wikipedia.org/wiki/Input%E2%80%93output_memory_management_unit?utm_source=chatgpt.com)

In sum, DMA allows efficient block transfers, freeing the CPU from low-level transfer duties.

**4. Storage Structure**

**Hierarchy of Storage**

Computer storage is organized in a multi-level hierarchy, trading off speed, cost, and volatility. Typical levels include:

1. **Registers** (fastest, smallest, volatile)
2. **CPU cache** (L1, L2, possibly L3)
3. **Main memory (RAM)**
4. **Secondary storage** (non-volatile, e.g. SSD, HDD)
5. **Tertiary / archival storage** (magnetic tapes, optical media) [users.soe.ucsc.edu+3UIC Computer Science+3Purdue Engineering+3](https://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/1_Introduction.html?utm_source=chatgpt.com)

Each successive level is larger in capacity, slower in access, and cheaper per bit. [UIC Computer Science+1](https://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/1_Introduction.html?utm_source=chatgpt.com)

**Main Memory vs Secondary Storage**

* **Main memory** is the only storage directly accessible by the CPU during execution; instructions and data reside in memory. [UIC Computer Science+2faculty.berea.edu+2](https://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/1_Introduction.html?utm_source=chatgpt.com)
* **Secondary storage** extends main memory by providing large, nonvolatile (persistent) capacity (e.g. magnetic disks, solid-state drives). [faculty.berea.edu+1](https://faculty.berea.edu/faculty/pearcej/CSC325/ch2.htm?utm_source=chatgpt.com)
* Disk surfaces are divided into **tracks** and **sectors**, and the disk controller handles translating logical requests into physical operations. [Purdue Engineering+1](https://engineering.purdue.edu/~ebertd/469/notes/EE469-ch2.pdf?utm_source=chatgpt.com)

**Caching & Buffering**

* **Caching**: frequently accessed data is kept in faster, smaller levels (e.g. cache, main memory) to reduce average access time. [UIC Computer Science+2Purdue Engineering+2](https://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/1_Introduction.html?utm_source=chatgpt.com)
* **Buffering / Spooling**: temporary storage (buffers) is used to accommodate speed mismatches between I/O devices and memory or to organize I/O more efficiently. [Purdue Engineering+1](https://engineering.purdue.edu/~ebertd/469/notes/EE469-ch2.pdf?utm_source=chatgpt.com)

**5. Hardware Protection**

Hardware (or architectural) protection mechanisms are necessary to enforce separation and safe resource usage among processes and the OS. The key categories of hardware protection are:

1. **CPU protection**
2. **Memory protection**
3. **I/O protection**

These are often enabled via a dual-mode (or multi-mode) operation in hardware. [Purdue Engineering+3faculty.berea.edu+3users.soe.ucsc.edu+3](https://faculty.berea.edu/faculty/pearcej/CSC325/ch2.htm?utm_source=chatgpt.com)

**Dual-Mode Operation**

* The hardware provides two (or more) modes: **user mode** and **kernel (monitor / supervisor) mode**.
* A **mode bit** in the CPU indicates which mode is active. [faculty.berea.edu+2users.soe.ucsc.edu+2](https://faculty.berea.edu/faculty/pearcej/CSC325/ch2.htm?utm_source=chatgpt.com)
* Some instructions are **privileged** and can only be executed in kernel mode (e.g. I/O instructions, enabling/disabling interrupts, setting memory limits).
* When an interrupt or trap occurs, the hardware automatically switches to kernel mode so that the OS can handle the event safely. [faculty.berea.edu+2users.soe.ucsc.edu+2](https://faculty.berea.edu/faculty/pearcej/CSC325/ch2.htm?utm_source=chatgpt.com)

Thus dual-mode operation is the foundation that allows hardware protection in CPU, memory, and I/O.

**6. I/O Protection**

I/O protection ensures that user processes cannot directly manipulate I/O devices or bypass OS control over device operations. Key aspects:

* **I/O instructions (or operations)** are usually privileged; thus they can only be executed in kernel mode. This ensures that only the OS can directly control I/O devices. [faculty.berea.edu+2users.soe.ucsc.edu+2](https://faculty.berea.edu/faculty/pearcej/CSC325/ch2.htm?utm_source=chatgpt.com)
* Because user processes can’t issue I/O instructions, they must request I/O via **system calls** (traps) that the OS safely mediates. [faculty.berea.edu+1](https://faculty.berea.edu/faculty/pearcej/CSC325/ch2.htm?utm_source=chatgpt.com)
* Preventing one process from interfering with another’s I/O: a process should not be able to disrupt, snoop on, or take control of I/O operations belonging to another process. [GeeksforGeeks](https://www.geeksforgeeks.org/computer-organization-architecture/hardware-protection-and-type-of-hardware-protection/?utm_source=chatgpt.com)

In modern systems, additional hardware support (e.g. IOMMU) may also guard device memory access to prevent malicious or errant DMA from corrupting memory or I/O. [Wikipedia](https://en.wikipedia.org/wiki/Input%E2%80%93output_memory_management_unit?utm_source=chatgpt.com)

**7. Memory Protection**

Memory protection ensures that processes cannot access memory regions outside their allocation, nor corrupt the OS or other processes. Mechanisms include:

**Base & Limit Registers**

* A simple hardware scheme uses two registers:
  + **Base (or relocation) register**: holds the lowest physical address that the process may access.
  + **Limit register**: holds the size of the accessible region (i.e. maximum offset).
* Every memory reference is checked: physical\_address = base + virtual\_offset; if offset exceeds limit, a fault occurs. [faculty.berea.edu+1](https://faculty.berea.edu/faculty/pearcej/CSC325/ch2.htm?utm_source=chatgpt.com)
* The OS (kernel) can load and set these registers (privileged instruction). [faculty.berea.edu+1](https://faculty.berea.edu/faculty/pearcej/CSC325/ch2.htm?utm_source=chatgpt.com)

**Memory Management Units (MMU) & Virtual Memory**

* In real systems, protection is more sophisticated and involves an **MMU**, which maps virtual addresses to physical addresses, with access rights.
* The MMU can enforce:
  + Page-level permissions (read, write, execute)
  + Isolation between processes
  + Protection of kernel space and interrupt vectors
* Violations result in **page faults** or **memory access violations** (exceptions).

**Memory Protection Units (MPU)**

* In simpler or embedded systems, an **MPU** (Memory Protection Unit) is a less complex alternative to an MMU; it provides region-based protection with access attributes. [Wikipedia](https://en.wikipedia.org/wiki/Memory_protection_unit?utm_source=chatgpt.com)
* MPUs permit defining several memory regions with permissions (read/write/execute) and enforce access violation faults if a process tries to exceed bounds. [Wikipedia](https://en.wikipedia.org/wiki/Memory_protection_unit?utm_source=chatgpt.com)

Thus memory protection is crucial to maintain isolation, prevent accidental or malicious memory corruption, and guard the kernel.

**8. CPU Protection**

CPU protection ensures that no user process can monopolize the CPU or disable interrupts, thereby preventing the OS from regaining control. The main mechanism is:

**Timer & Preemption**

* The OS uses a **timer** (hardware timer) to generate periodic interrupts (e.g. every few milliseconds).
* Before giving control to a user process, the OS sets the timer.
* When the timer expires, a timer interrupt is raised, forcing the CPU to return control to the OS (i.e. **preemption**). [faculty.berea.edu+2users.soe.ucsc.edu+2](https://faculty.berea.edu/faculty/pearcej/CSC325/ch2.htm?utm_source=chatgpt.com)
* This mechanism ensures fairness (no process can hog the CPU indefinitely) and allows the OS to implement time-sharing.

**Privileged Instructions & Interrupt Masking**

* Certain instructions (e.g. to disable interrupts, change mode bit) are **privileged** and can only execute in kernel mode; user code cannot disable interrupts. This prevents a malicious or faulty user program from disabling the OS’s ability to regain control. [faculty.berea.edu+2users.soe.ucsc.edu+2](https://faculty.berea.edu/faculty/pearcej/CSC325/ch2.htm?utm_source=chatgpt.com)