

## Module 2: Computer-System Structures

- Computer System Operation
- I/O Structure
- Storage Structure
- Storage Hierarchy
- Hardware Protection
- General System Architecture

Operating System Concepts

2.1

Silberschatz and Galvin 1999

---

---

---

---

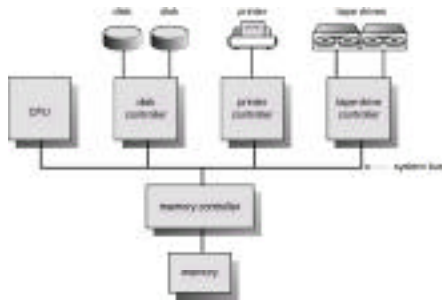
---

---

---

---

## Computer-System Architecture



Operating System Concepts

2.2

Silberschatz and Galvin 1999

---

---

---

---

---

---

---

---

## Computer-System Operation

- I/O devices and the CPU can execute concurrently.
- Each device controller is in charge of a particular device type.
- Each device controller has a local buffer.
- CPU moves data from/to main memory to/from local buffers
- I/O is from the device to local buffer of controller.
- Device controller informs CPU that it has finished its operation by causing an *interrupt*.

Operating System Concepts

2.3

Silberschatz and Galvin 1999

---

---

---

---

---

---

---

---

## Common Functions of Interrupts

- Interrupt transfers control to the interrupt service routine generally, through the *interrupt vector*, which contains the addresses of all the service routines.
- Interrupt architecture must save the address of the interrupted instruction.
- Incoming interrupts are *disabled* while another interrupt is being processed to prevent a *lost interrupt*.
- A *trap* is a software-generated interrupt caused either by an error or a user request.
- An operating system is *interrupt driven*.

Operating System Concepts

2.4

Silberschatz and Galvin 1999

---

---

---

---

---

---

---

---

## Interrupt Handling

- The operating system preserves the state of the CPU by storing registers and the program counter.
- Determines which type of interrupt has occurred:
  - *polling*
  - vectored interrupt system
- Separate segments of code determine what action should be taken for each type of interrupt

Operating System Concepts

2.5

Silberschatz and Galvin 1999

---

---

---

---

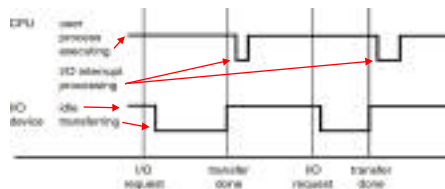
---

---

---

---

## Interrupt Time Line For a Single Process Doing Output



Operating System Concepts

2.6

Silberschatz and Galvin 1999

---

---

---

---

---

---

---

---

### I/O Structure

- **Synchronous** - After I/O starts, control returns to user program only upon I/O completion.
  - wait instruction idles the CPU until the next interrupt
  - wait loop (contention for memory access).
  - At most one I/O request is outstanding at a time, no simultaneous I/O processing (for a single "thread").
- **Asynchronous** - After I/O starts, control returns to user program without waiting for I/O completion.
  - *System call* - request to the operating system to allow user to wait for I/O completion.
  - *Device-status table* contains entry for each I/O device indicating its type, address, and state.
  - Operating system indexes into I/O device table to determine device status and to modify table entry to include interrupt.

Operating System Concepts 2.7 Siberschatz and Galvin 1999

---

---

---

---

---

---

---

---

### Two I/O methods

Synchronous

Asynchronous

Operating System Concepts 2.8 Siberschatz and Galvin 1999

---

---

---

---

---

---

---

---

### Device-Status Table

Device: disk reader 1 STATUS: OK	→	request for file pointer address: 38040 length: 1202	→	
Device: file pointer 1 STATUS: busy				
Device: disk unit 1 STATUS: OK				
Device: disk unit 2 STATUS: OK				
Device: disk unit 3 STATUS: busy		request for disk unit 3		
...				
		file read operation read address: x3045 length: 2000		request for disk unit 2
				file write operation write address: 00405 length: 100

Operating System Concepts 2.9 Siberschatz and Galvin 1999

---

---

---

---

---

---

---

---

### Direct Memory Access (DMA) 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.

Operating System Concepts      2.10      Silberschatz and Galvin 1999

---

---

---

---

---

---

---

---

### Storage Structure

- Main memory – only large storage media that the CPU can access directly.
- Secondary storage – extension of main memory that provides large nonvolatile storage capacity.
- Magnetic disks – rigid metal or glass platters covered with magnetic recording material
  - Disk surface is logically divided into *tracks*, which are subdivided into *sectors*.
  - The *disk controller* determines the logical interaction between the device and the computer.

Operating System Concepts      2.11      Silberschatz and Galvin 1999

---

---

---

---

---

---

---

---

### Moving-Head Disk Mechanism

Operating System Concepts      2.12      Silberschatz and Galvin 1999

---

---

---

---

---

---

---

---

### Storage Hierarchy

- Storage systems organized in hierarchy.
  - Speed
  - cost
  - volatility
- *Caching* – copying information into faster storage system; main memory can be viewed as a last *cache* for secondary storage.

Operating System Concepts      2.13      Silberschatz and Galvin 1999

---

---

---

---

---

---

---

---

### Storage-Device Hierarchy

Operating System Concepts      2.14      Silberschatz and Galvin 1999

---

---

---

---

---

---

---

---

### Hardware Protection

- Dual-Mode Operation
- I/O Protection
- Memory Protection
- CPU Protection

Operating System Concepts      2.15      Silberschatz and Galvin 1999

---

---

---

---

---

---

---

---

### Dual-Mode Operation

- Sharing system resources requires operating system to ensure that an incorrect program cannot cause other programs to execute incorrectly.
- Provide hardware support to differentiate between at least two modes of operations.
  1. *User mode* – execution done on behalf of a user.
  2. *Monitor mode* (also *supervisor mode* or *system mode*) – execution done on behalf of operating system.

Operating System Concepts      2.16      Silberschatz and Galvin 1999

---

---

---

---

---

---

---

---

### Dual-Mode Operation (Cont.)

- *Mode bit* added to computer hardware to indicate the current mode: monitor (0) or user (1).
- When an interrupt or fault occurs hardware switches to monitor mode.

```

    graph LR
      user((user)) -- "Interrupt/fault" --> monitor((monitor))
      monitor -- "set user mode" --> user
  
```

- *Privileged instructions* can be issued only in monitor mode.

Operating System Concepts      2.17      Silberschatz and Galvin 1999

---

---

---

---

---

---

---

---

### I/O Protection

- All I/O instructions are privileged instructions.
- Must ensure that a user program could never gain control of the computer in monitor mode (i.e., a user program that, as part of its execution, stores a new address in the interrupt vector).

Operating System Concepts      2.18      Silberschatz and Galvin 1999

---

---

---

---

---

---

---

---

### Memory Protection

- Must provide memory protection at least for the interrupt vector and the interrupt service routines.
- In order to have memory protection, add two registers that determine the range of legal addresses a program may access:
  - **base register** - holds the smallest legal physical memory address.
  - **Limit register** - contains the size of the range
- Memory outside the defined range is protected.

Operating System Concepts 2.19 Silberschatz and Galvin 1999

---

---

---

---

---

---

---

---

### A Base And A limit Register Define A Logical Address Space

The diagram illustrates a memory layout. On the left, a vertical stack of memory blocks is shown with addresses: 0, 250000, 380000, 420000, 580000, and 1324000. The blocks are labeled 'monitor', 'job 1', 'job 2', 'job 3', and 'job 4'. To the right, two registers are shown: 'base register' with value '300B40' and 'limit register' with value '12C800'. Arrows point from these registers to the boundaries of the 'job 2' block.

Operating System Concepts 2.20 Silberschatz and Galvin 1999

---

---

---

---

---

---

---

---

### Protection Hardware

The flowchart shows the hardware path for memory access. It starts with a 'CPU' providing an 'address'. This address goes through a 'base' register and a 'base + limit' register. Two decision diamonds check if the address is within the defined range. If 'no', it leads to 'fail to operating system monitor-addressing error'. If 'yes', it leads to 'memory'.

- When executing in monitor mode, the operating system has unrestricted access to both monitor and user's memory.
- The load instructions for the *base* and *limit* registers are privileged instructions.

Operating System Concepts 2.21 Silberschatz and Galvin 1999

---

---

---

---

---

---

---

---

### CPU Protection

- *Timer* – interrupts computer after specified period to ensure operating system maintains control.
  - Timer is decremented every clock tick.
  - When timer reaches the value 0, an interrupt occurs.
- Timer commonly used to implement time sharing.
- Time also used to compute the current time.
- Load-timer is a privileged instruction.

Operating System Concepts 2.22 Silberschatz and Galvin 1999

---

---

---

---

---

---

---

---

### General-System Architecture

- Given the I/O instructions are privileged, how does the user program perform I/O?
- System call – the method used by a process to request action by the operating system.
  - Usually takes the form of a trap to a specific location in the interrupt vector.
  - Control passes through the interrupt vector to a service routine in the OS, and the mode bit is set to monitor mode.
  - The monitor verifies that the parameters are correct and legal, executes the request, and returns control to the instruction following the system call.

Operating System Concepts 2.23 Silberschatz and Galvin 1999

---

---

---

---

---

---

---

---

### Use of A System Call to Perform I/O

Operating System Concepts 2.24 Silberschatz and Galvin 1999

---

---

---

---

---

---

---

---