

Module 2: Computer-System Structures

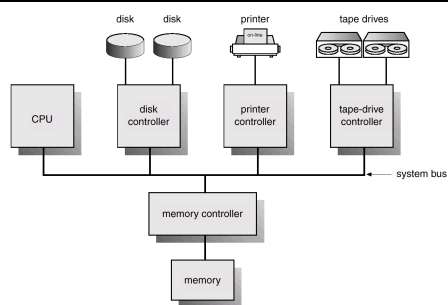
- Computer System Operation
- I/O Structure
- Storage Structure
- Storage Hierarchy
- Hardware Protection
- General System Architecture

Operating System Concepts

2.1

Laface 2012 –Silberschatz and Galvin©1999

Computer-System Architecture



Operating System Concepts

2.2

Laface 2012 –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

Laface 2012 –Silberschatz and Galvin©1999

Common Functions of Interrupts

- Interrupts 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*
- An operating system is *interrupt driven*

Operating System Concepts

2.4

Laface 2012 –Silberschatz and Galvin©1999

Trap

- A trap is a software-generated interrupt caused either by an error or a user request
- A trap is typically a **synchronous** event, for example
 - Division by zero
 - Memory access violation
- The program can restart from the interrupted operation, rather than from the following instruction

Operating System Concepts

2.5

Laface 2012 –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

2.6

Laface 2012 –Silberschatz and Galvin©1999

I/O Structure

- 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.
- After I/O starts, control returns to user program without waiting for I/O completion.

Operating System Concepts

2.7

Laface 2012 –Silberschatz and Galvin©1999

Two I/O methods

Synchronous

Diagram (a) illustrates synchronous I/O. A vertical timeline shows the interaction between user and kernel components. In the user space, a 'requesting process' sends a request to the 'device driver' in the kernel space. The process then enters a 'waiting' state. The 'device driver' initiates 'data transfer' with the 'hardware'. Once complete, the 'interrupt handler' in the kernel sends a signal back to the 'requesting process', which then resumes execution. The timeline is labeled 'time' at the bottom.

(a)

Asynchronous

Diagram (b) illustrates asynchronous I/O. Similar to (a), the 'requesting process' sends a request to the 'device driver', which initiates 'data transfer' with the 'hardware'. However, instead of waiting, the 'requesting process' continues its execution. The 'interrupt handler' in the kernel sends an interrupt signal to the process at a later point in time, indicated by a dashed line for the data transfer and a later return signal. The timeline is labeled 'time' at the bottom.

(b)

Operating System Concepts

2.8

Laface 2012 –Silberschatz and Galvin©1999

I/O interrupt management

The diagram shows the flow of an I/O interrupt. On the left, a process P1 sends an 'IO request' to the 'SVC' (Service) layer, which then passes it to the 'Kernel'. The 'Kernel' initiates the I/O operation. On the right, an 'interrupt' signal (represented by multiple green arrows) is sent from the hardware to the 'Kernel'. The 'Kernel' then sends a signal back to P1, indicating completion. A vertical timeline 't' is shown on the far left.

Operating System Concepts

2.9

Laface 2012 –Silberschatz 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

Laface 2012 –Silberschatz and Galvin©1999

Storage Structure

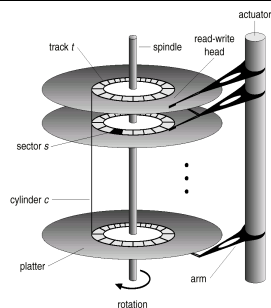
- Main memory – only large storage media that the CPU can access directly.
- Secondary storage – extension of main memory that provides large nonvolatile storage capacity.
- Magnetic disks – rigid metal or glass platters covered with magnetic recording material
 - Disk surface is logically divided into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

Operating System Concepts

2.11

Laface 2012 –Silberschatz and Galvin©1999

Moving-Head Disk Mechanism



Operating System Concepts

2.12

Laface 2012 –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 Concepts2.13Laface 2012 –Silberschatz and Galvin©1999

Storage-Device Hierarchy

```
graph TD; registers[registers] <--> cache[cache]; cache <--> main_memory[main memory]; main_memory <--> electronic_disk[electronic disk]; electronic_disk <--> magnetic_disk[magnetic disk]; magnetic_disk <--> optical_disk[optical disk]; optical_disk <--> magnetic_tapes[magnetic tapes];
```

Operating System Concepts2.14Laface 2012 –Silberschatz and Galvin©1999

Hardware Protection

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

Operating System Concepts2.15Laface 2012 –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. Kernel mode (also supervisor mode or system mode) – execution done on behalf of operating system.

Operating System Concepts
2.16
Laface 2012 –Silberschatz and Galvin©1999

Dual-Mode Operation (Cont.)

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

```

graph LR
    user((user)) -- "Interrupt/trap" --> kernel((kernel))
    kernel -- "set user mode" --> user
    
```

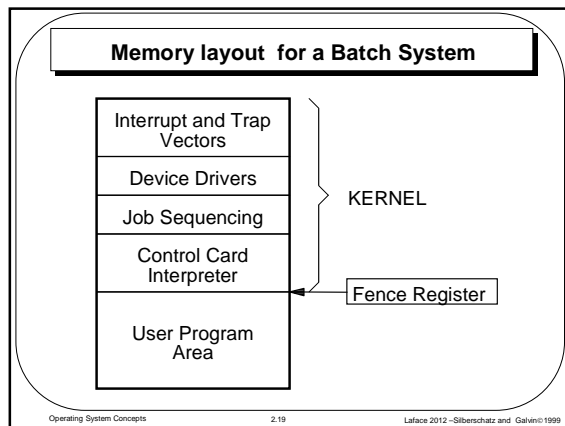
- **Privileged instructions**, that can be issued only in kernel mode.

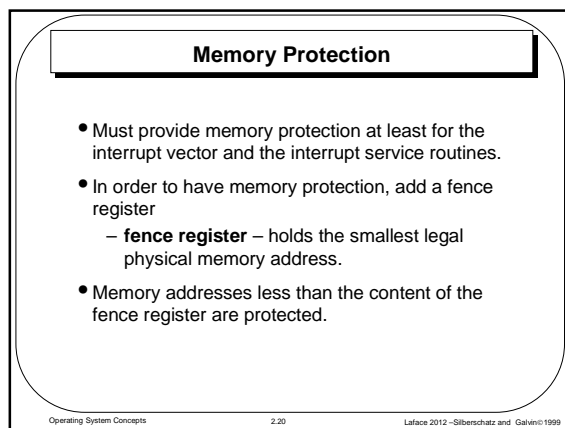
Operating System Concepts
2.17
Laface 2012 –Silberschatz and Galvin©1999

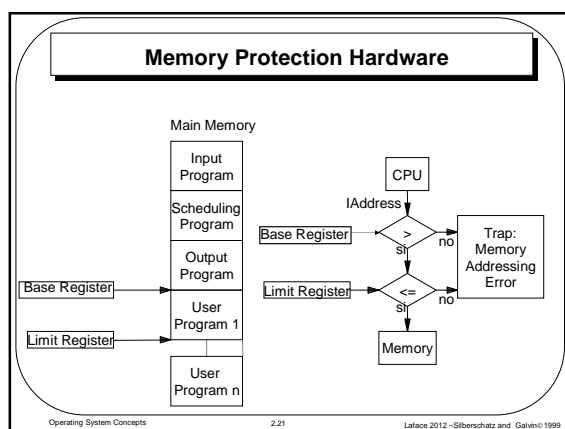
I/O Protection

- All I/O instructions are privileged instructions.
- Must ensure that a user program could never gain control of the computer in kernel mode
- I.e., a user program cannot store a new address in the interrupt vector

Operating System Concepts
2.18
Laface 2012 –Silberschatz and Galvin©1999



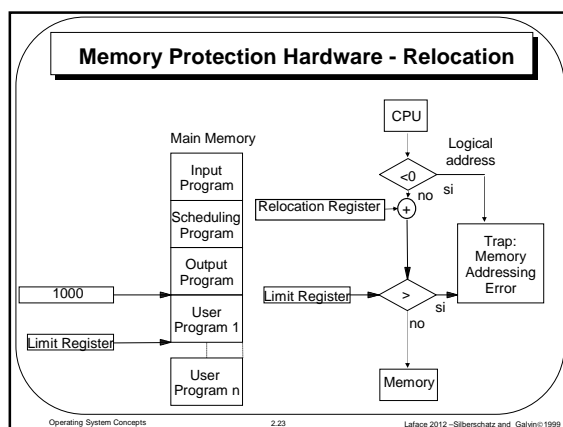




Memory Protection

- 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 2.22 Laface 2012 –Silberschatz and Galvin©1999



Protection Hardware

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

Operating System Concepts 2.24 Laface 2012 –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
- Timer also used to compute the current time
- Load-timer is a privileged instruction

Operating System Concepts

2.25

Laface 2012 –Silberschatz and Galvin©1999

General-System Architecture

- Given the I/O instructions are privileged, how does the user program perform I/O?
- More generally, how does a user program call a kernel function?

Operating System Concepts

2.26

Laface 2012 –Silberschatz and Galvin©1999

General-System Architecture

- 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 kernel 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

2.27

Laface 2012 –Silberschatz and Galvin©1999

