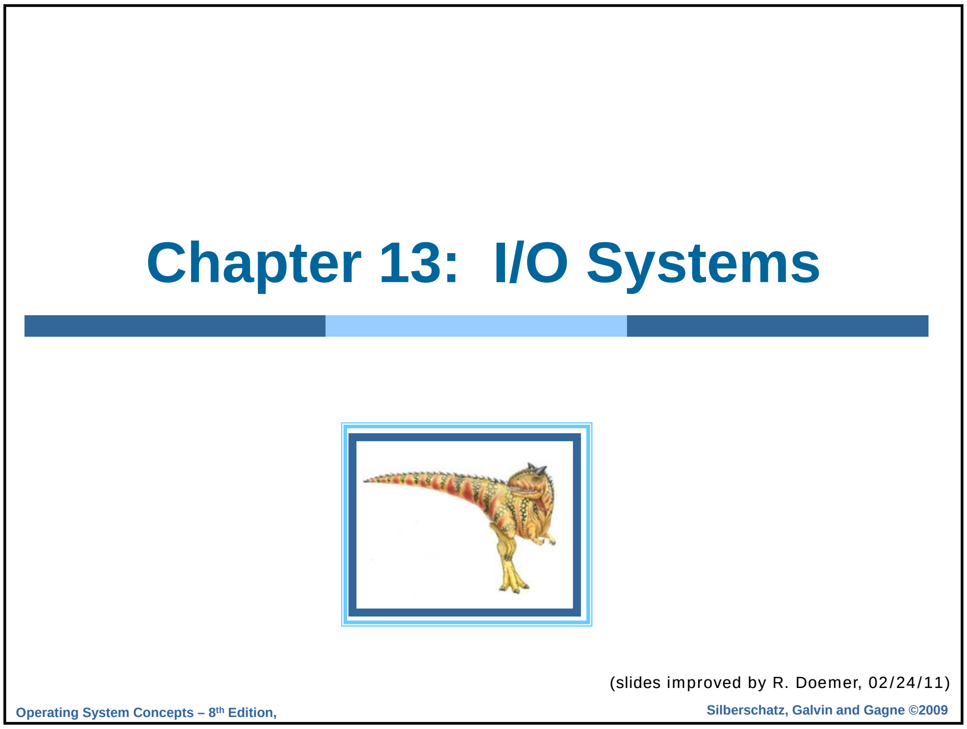
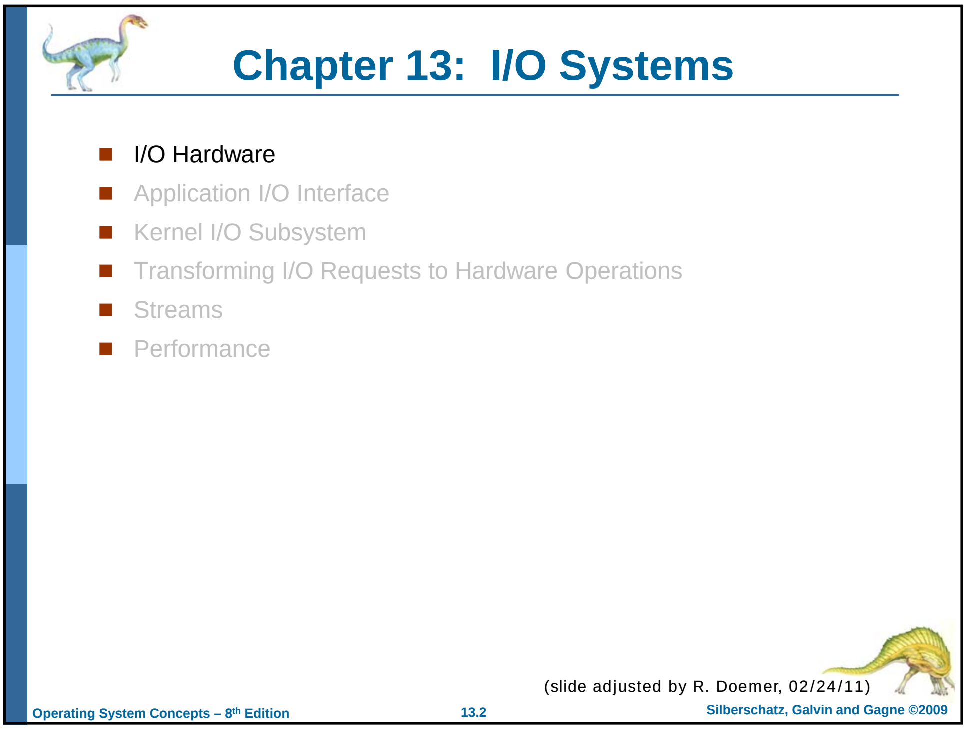


Chapter 13: I/O Systems

(slides improved by R. Doemer, 02/24/11)



Chapter 13: I/O Systems

- I/O Hardware
- Application I/O Interface
- Kernel I/O Subsystem
- Transforming I/O Requests to Hardware Operations
- Streams
- Performance

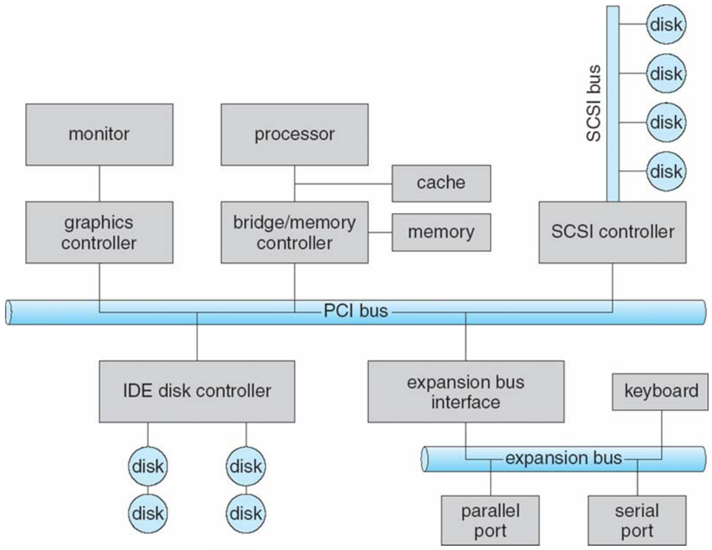
(slide adjusted by R. Doemer, 02/24/11)

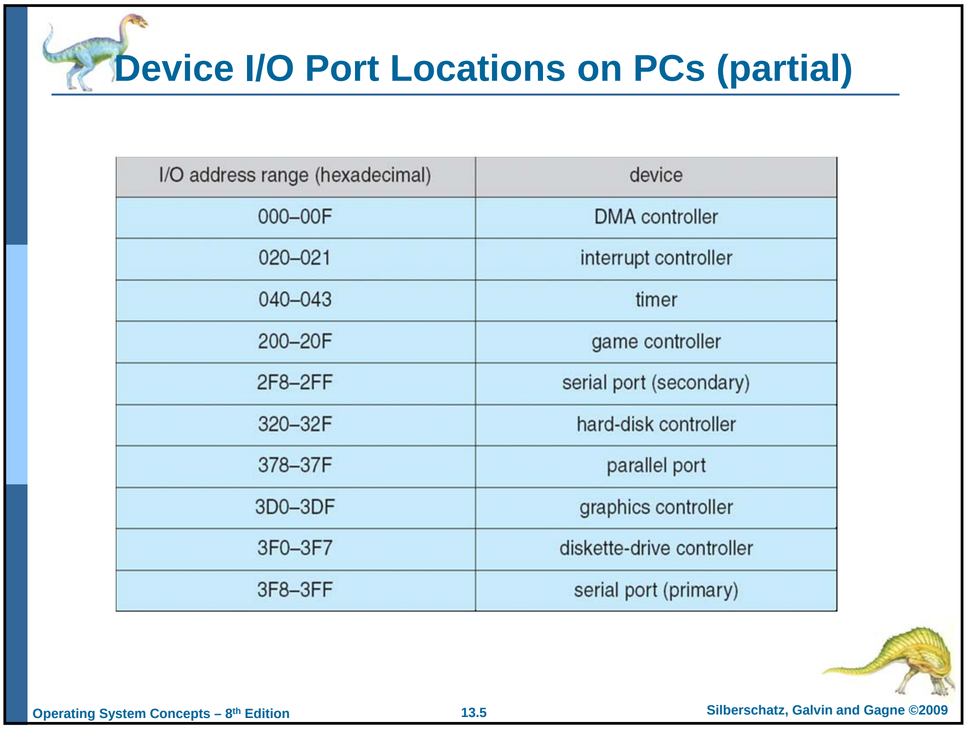
I/O Hardware

- Incredible **variety of I/O devices** with different features
- **Common concepts**
  - **Port**
  - **Bus** (**daisy chain** or shared direct access)
  - **Controller** (host adapter)
- CPU controls devices by use of **I/O instructions**
- Devices have *addresses*, used by either
  - **Direct I/O instructions**, or
  - **Memory-mapped I/O**

(slide adjusted by R. Doemer, 02/24/11)

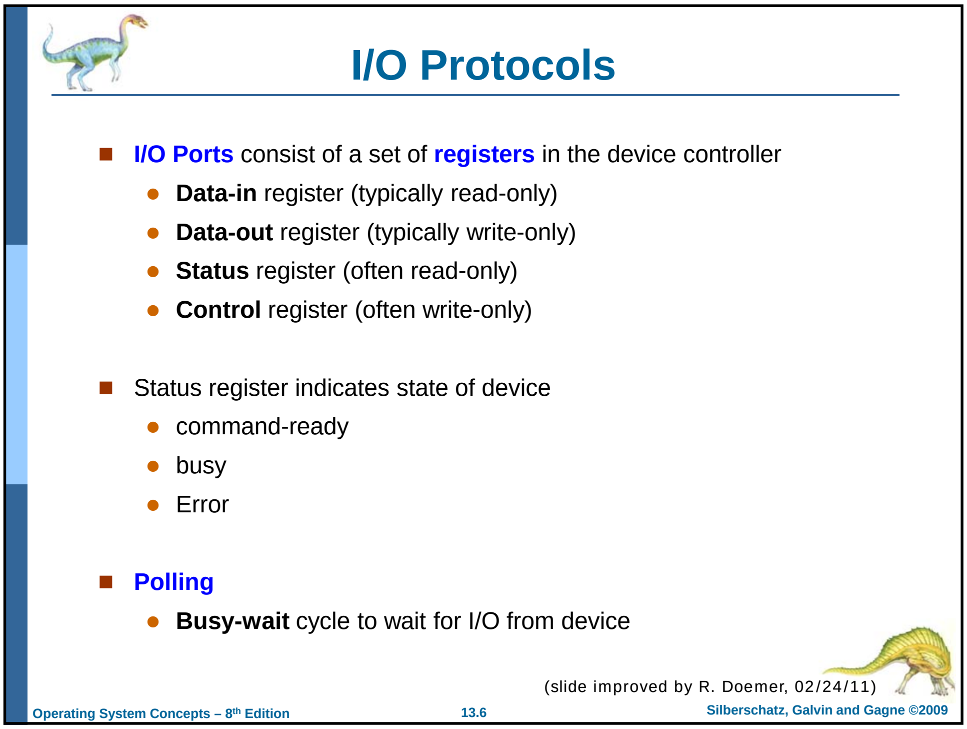
A Typical PC Bus Structure





Device I/O Port Locations on PCs (partial)

I/O address range (hexadecimal)	device
000–00F	DMA controller
020–021	interrupt controller
040–043	timer
200–20F	game controller
2F8–2FF	serial port (secondary)
320–32F	hard-disk controller
378–37F	parallel port
3D0–3DF	graphics controller
3F0–3F7	diskette-drive controller
3F8–3FF	serial port (primary)



I/O Protocols

- **I/O Ports** consist of a set of **registers** in the device controller
  - **Data-in** register (typically read-only)
  - **Data-out** register (typically write-only)
  - **Status** register (often read-only)
  - **Control** register (often write-only)
- Status register indicates state of device
  - command-ready
  - busy
  - Error
- **Polling**
  - **Busy-wait** cycle to wait for I/O from device

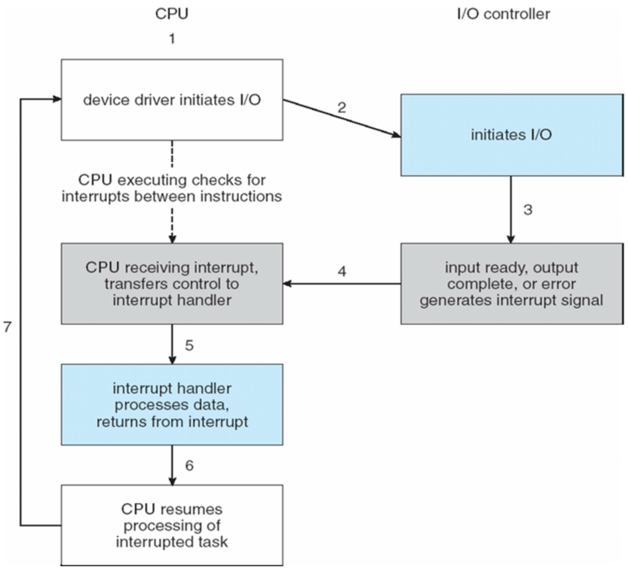
(slide improved by R. Doemer, 02/24/11)

I/O Protocols

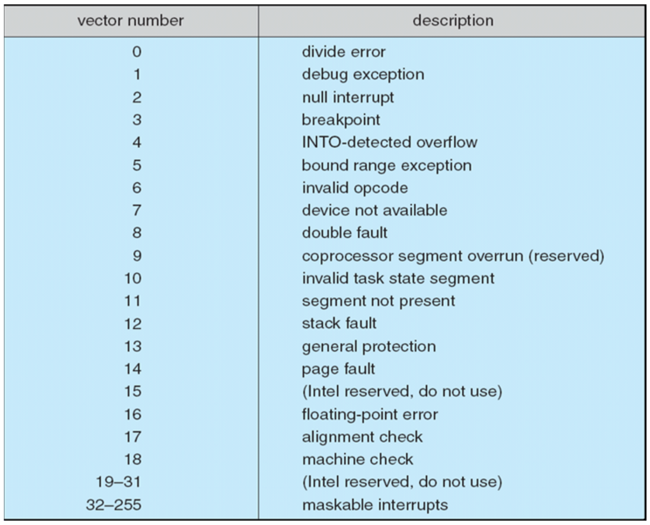
- **Interrupt**-driven I/O
  - CPU **interrupt-request line** is triggered by I/O device
  - **Interrupt handler** receives interrupts
    - Some interrupts are **maskable** to ignore or delay some interrupts
    - Some **nonmaskable**
  - **Interrupt vector** is used to dispatch interrupt to correct handler
    - Often based on priority
- Interrupt mechanism is also used for exceptions (traps)

(slide improved by R. Doemer, 02/24/11)

Interrupt-Driven I/O Cycle


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

Intel Pentium Processor Event-Vector Table

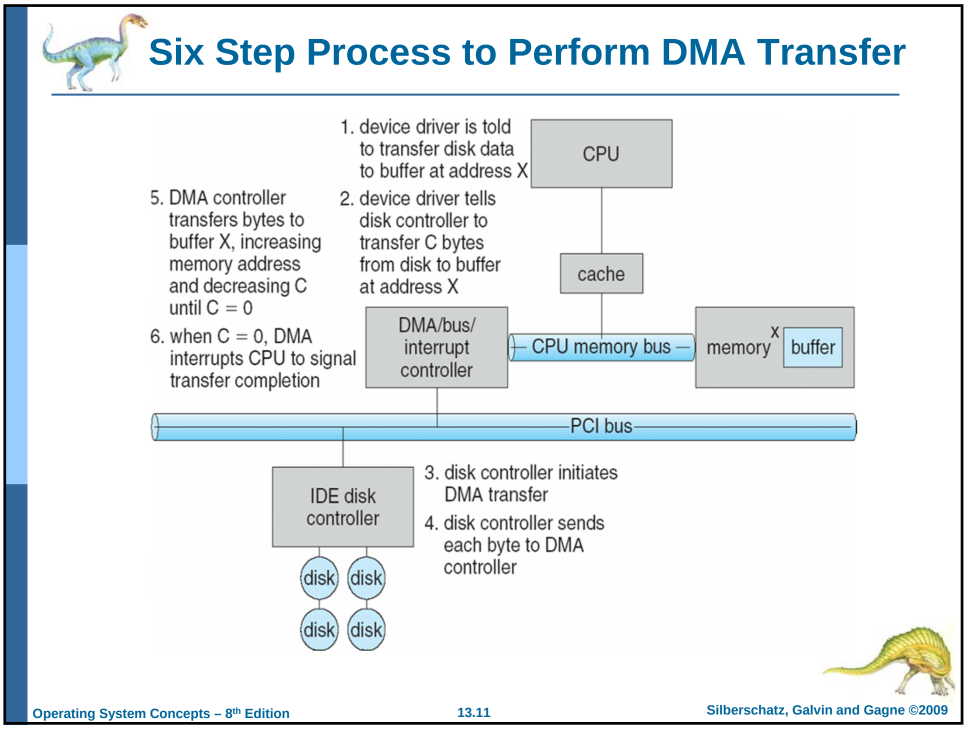


vector number	description
0	divide error
1	debug exception
2	null interrupt
3	breakpoint
4	INTO-detected overflow
5	bound range exception
6	invalid opcode
7	device not available
8	double fault
9	coprocessor segment overrun (reserved)
10	invalid task state segment
11	segment not present
12	stack fault
13	general protection
14	page fault
15	(Intel reserved, do not use)
16	floating-point error
17	alignment check
18	machine check
19–31	(Intel reserved, do not use)
32–255	maskable interrupts

Direct Memory Access

- Polling and Interrupt-based I/O requires heavy CPU usage
- CPU load can be reduced by **Direct Memory Access (DMA)**
  - Used to avoid **programmed I/O** for large data movement
  - Requires **DMA controller**
  - Bypasses CPU to transfer data directly between I/O device and main memory

(slide improved by R. Doemer, 02/24/11)



Six Step Process to Perform DMA Transfer

1. device driver is told to transfer disk data to buffer at address X
2. device driver tells disk controller to transfer C bytes from disk to buffer at address X
3. disk controller initiates DMA transfer
4. disk controller sends each byte to DMA controller
5. DMA controller transfers bytes to buffer X, increasing memory address and decreasing C until C = 0
6. when C = 0, DMA interrupts CPU to signal transfer completion