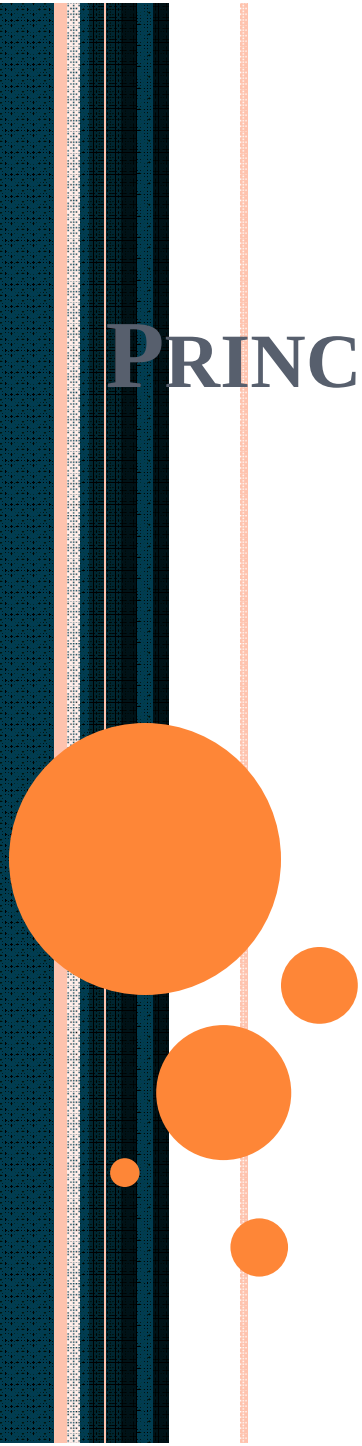
PRINCIPLES OF OPERATING SYSTEMS



LECTURE 32
I/O HARDWARE

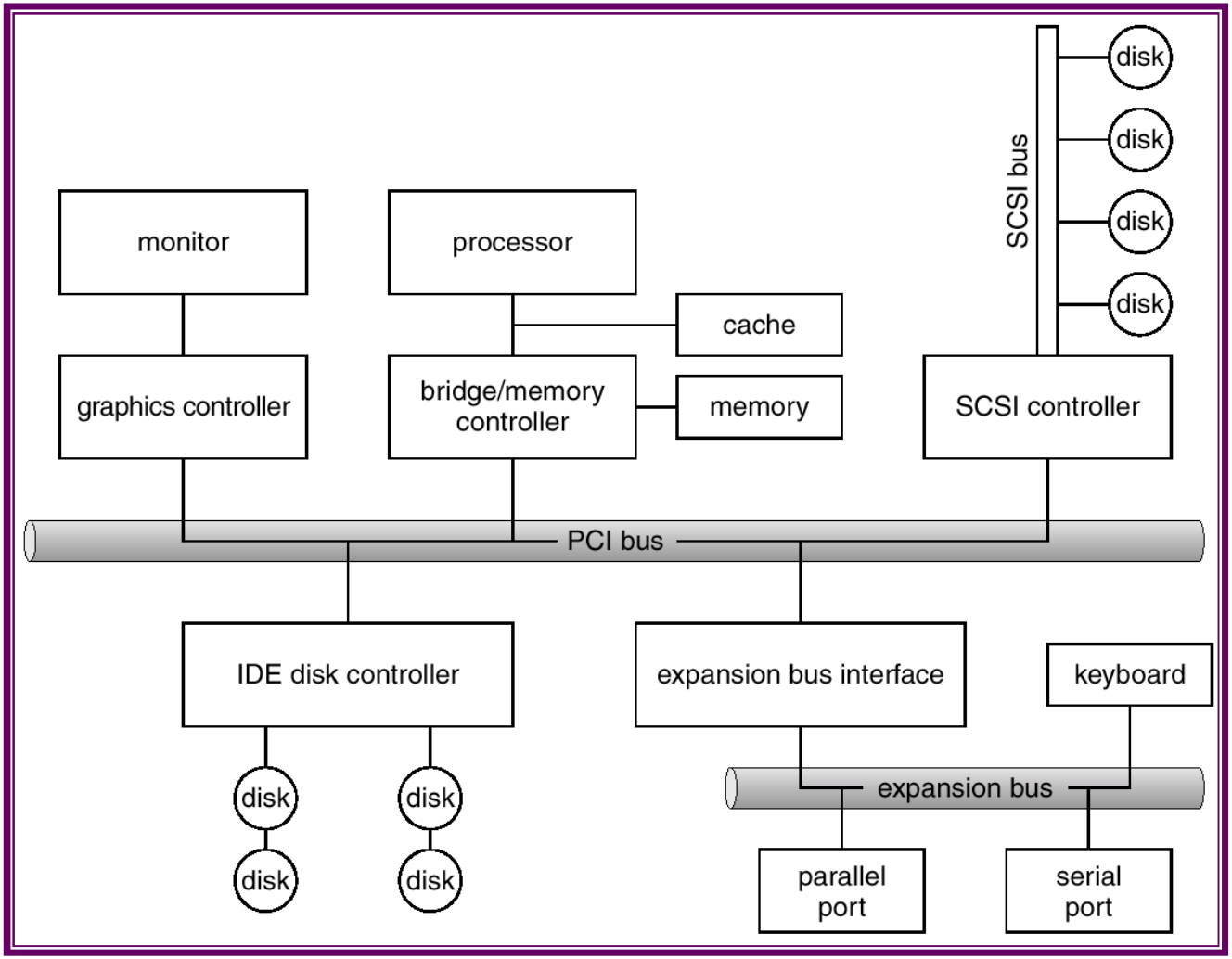
I/O Systems

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

I/O Hardware

- Incredible variety of I/O devices
- Common concepts
  - Port **- basic interface to CPU - status, control, data**
  - Bus (daisy chain or shared direct access) **- main and specialized local (ex: PCI for main and SCSI for disks)**
  - Controller (host adapter) **- HW interface between Device and Bus - an adapter card or mother board module Controller has special purposes registers (commands, etc.) which when written to causes actions to take place - may be memory mapped**
- I/O instructions control devices **- ex: in, out for Intel**
- Devices have addresses, used by
  - Direct I/O instructions **- uses I/O instructions**
  - Memory-mapped I/O **- uses memory instructions**

A Typical PC Bus Structure



Device I/O Port Locations on PCs (partial)

Various ranges for a device includes both control and data ports

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)

Polling

- **Handshaking**
- Determines state of device
  - `command-ready`
  - `busy`
  - `Error`
- Busy-wait cycle to wait for I/O from device
  When not busy - set data in data port, set command in control port and let ‘er rip
- **Not desirable if excessive - since it is a busy wait which ties up CPU & interferes with productive work**
- **Remember CS220 LABs**

Interrupts

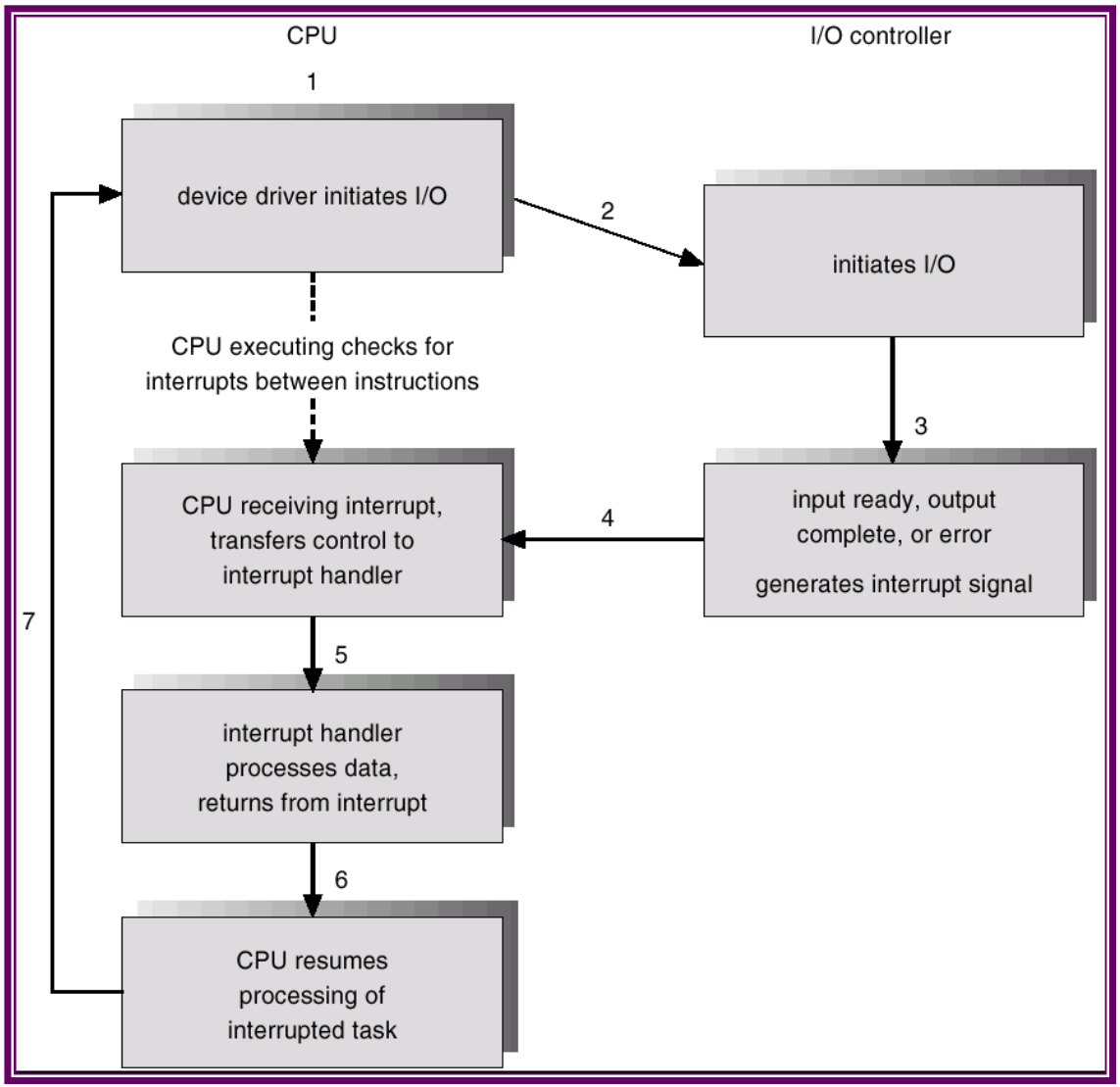
- CPU Interrupt request line **(IRQ)** triggered by I/O device

- Interrupt handler receives interrupts
- Maskable to ignore or delay some interrupts
- Interrupt vector to dispatch interrupt to correct handler
  - Based on priority
  - Some unmaskable
- Interrupt mechanism also used for exceptions
- **Application can go away after I/O request, but is til responsible for transferring data to memory when it becomes available from the device.**
- **Can have “nested” interrupts (with Priorities)**
- **See Instructors notes: “Use of Interrupts and DMA”**
- **Soft interrupts or “traps” generated from OS in system calls.**

Interrupt-Driven I/O Cycle

Go away & do

Something else ==>



Intel Pentium Processor Event-Vector Table

Interrupts 0-31 are non-maskable - cannot be disabled

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
19Ð31	(Intel reserved, do not use)
32Ð255	maskable interrupts

Direct Memory Access

- **With pure interrupt scheme, CPU was still responsible for transferring data from controller to memory (on interrupt) when device mad it available.**

- **Now DMA will do this - all CPU has to do is set up DMA and user the data when the DMA-complete interrupt arrives. … Interrupts still used - but only to signal DMA Complete.**

- Used to avoid programmed I/O for large data movement
- Requires DMA controller
- Bypasses CPU to transfer data directly between I/O device and memory
- **Cycle stealing: interference with CPU memory instructions during DMA transfer. - DMA takes priority - CPU pauses on memory part of word.**

Six Step Process to Perform DMA Transfer