Chapter 1: Introduction
Chapter 13: I/O structure

Prof. Li-Pin Chang
NYCU

WHAT IS AN OPERATING SYSTEM?

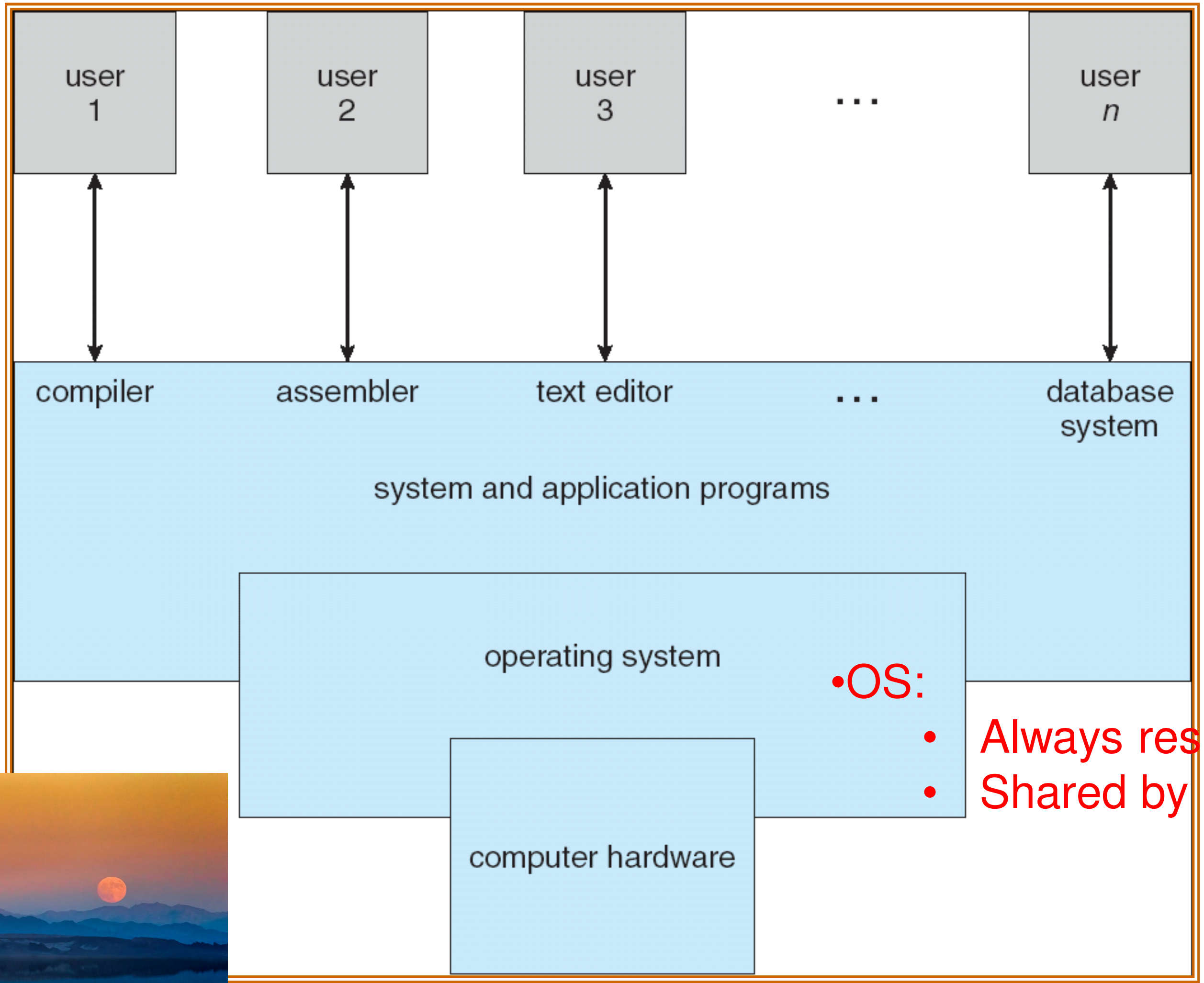
What is an Operating System?

- A program that acts as an intermediary between a user of a computer and the computer hardware
- Operating system goals:
  - Execute user programs and make solving user problems easier
  - Make the computer system convenient to use
  - Use the computer hardware in an efficient manner

Computer System Structure

- Computer system can be divided into four components
- Hardware – provides basic computing resources
  - CPU, memory, I/O devices
- Operating system
  - Controls and coordinates use of hardware among various applications and users
- Application programs – define the ways in which the system resources are used to solve the computing problems of the users
  - Word processors, compilers, web browsers, database systems, video games
- Users
  - Human beings
  - Some systems do not interact with human

Four Components of a Computer System



- OS:
  - Always resident
  - Shared by users

Operating System Definition

- OS is a resource allocator
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use
- OS is a control program
  - Controls (or monitor) execution of programs to prevent errors and improper use of the computer

Operating System Definition (Cont'd)

- No black-and-white definition
  - *"Everything a vendor ships when you order an operating system"* is good approximation
  - Err… how about the browser?
- "The one program running at all times on the computer" is the kernel
  - Everything else is either a system program or an application program
- Example: the UNIX OS
  - POSIX (IEEE 1003.1-2017)
  - A kernel
  - A collection of system programs ~= coreutil + binutil

- Which one(s) of the following are part of operating systems?
    a) The CPU scheduler
    b) Device drivers
    c) Compilers
    d) Word processors

THE THREE PILLARS

Pillar 1: Process Management

- A process is a program in execution It is a unit of work within the system Program is a passive entity, process is an active entity
- Process needs resources to accomplish its task
  - CPU time for execution
  - I/O for communication
  - Files for data storage

- A system has many processes running concurrently on one or more CPUs

Process Management Activities

- The operating system is responsible to the following activities in connection with process management:
  - Creating and deleting processes
  - Suspending and resuming processes
  - Providing mechanisms for process communication
  - Providing mechanisms for process synchronization
  - Providing mechanisms for deadlock handling

Pillar 2: Memory Management

- All data in memory before and after processing
- All instructions in memory in order to execute
- Memory management activities
  - Allocating and deallocating memory space as needed
  - Deciding which processes (or parts thereof) and data to move into and out of memory (swapping or paging)
  - Allowing programs allocate much more memory than physical memory (virtual memory)
  - Keeping track of which parts of memory are currently being used and by whom (protection)

Pillar 3: File/Storage Management

- OS provides uniform, logical view of information storage
  - Abstracts physical properties to logical storage unit - file
- File-System management
  - Files usually organized into directories
  - Access control on most systems to determine who can access what
- File system basic activities
  - Creating and deleting files and directories
  - Reading and writing files and directories
  - Mapping files from secondary storage to main memory

Storage Management

- Entire speed of computer operation hinges on disk subsystem and its algorithms
- OS activities
  - Storage allocation
  - Free-space management
  - Disk scheduling
- Some storage need not be fast
  - Tertiary storage includes optical storage, magnetic tape
  - Varies between WORM (write-once, read-many-times) and RW (read-write)

I/O STRUCTURE

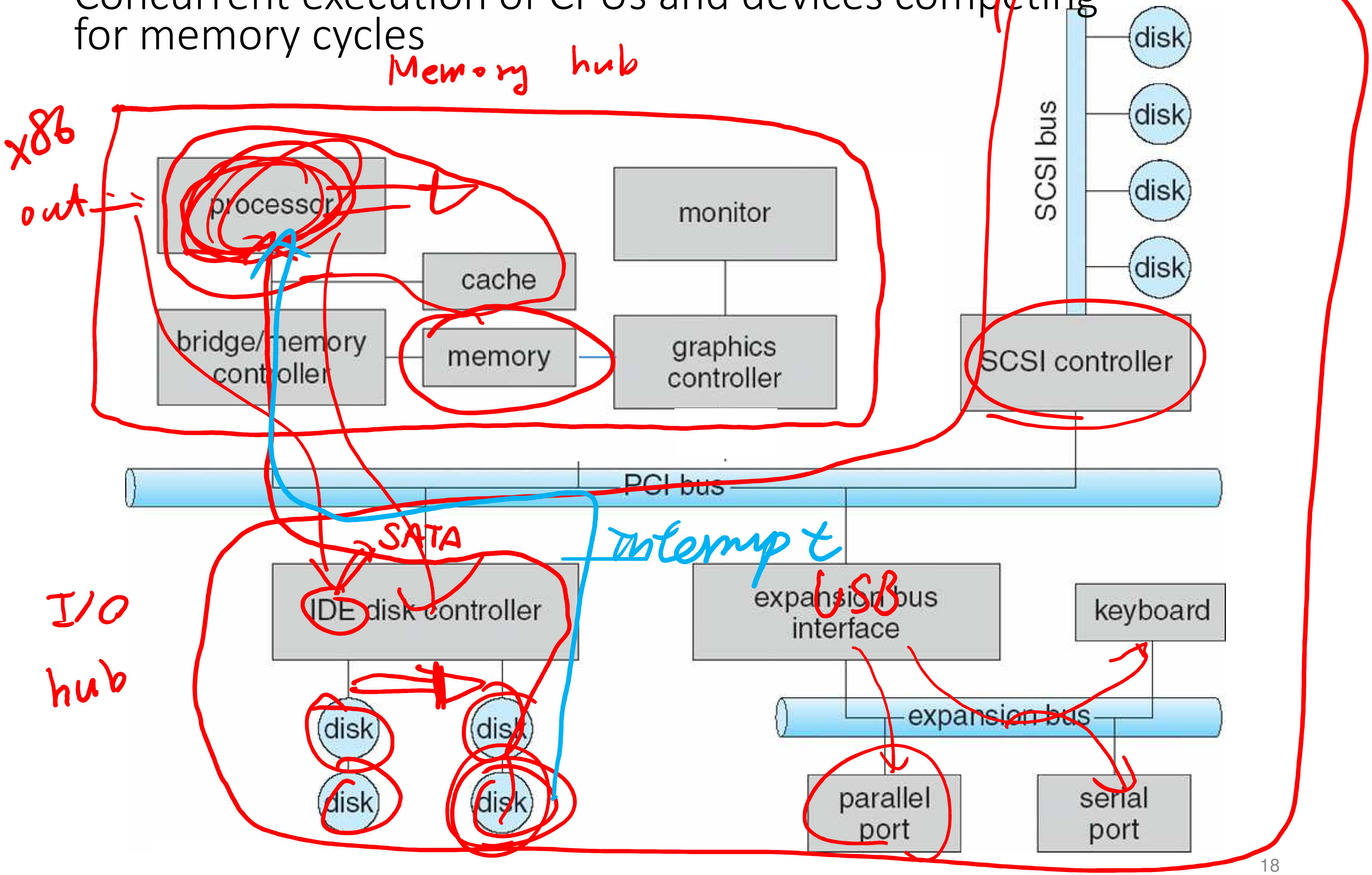
Why Bothering with I/Os?

- Operating system is called synchronously by user programs, but it is also often triggered by asynchronous events, i.e., hardware events from outside of the CPU
- User programs run on the CPU in parallel with operations on I/O devices
- OS strives to optimize CPU utilization and I/O latency at the same time, and the optimization is closely related to how I/O devices behave

Before OS: Computer Startup

- Bootstrap program is loaded at power-up or reboot
  - Typically stored in ROM or EEPROM, generally known as firmware
  - Initializes all aspects of system
  - Loads operating system kernel and starts execution
- Legacy PC bootloading procedure
  - FFFF:0000
  - BIOS
  - MBR (Master Boot Record)
  - Boot Manager
  - Pre-OS
  - OS

- One or more CPUs, device controllers connect through common bus providing access to shared memory
- Concurrent execution of CPUs and devices competing for memory cycles

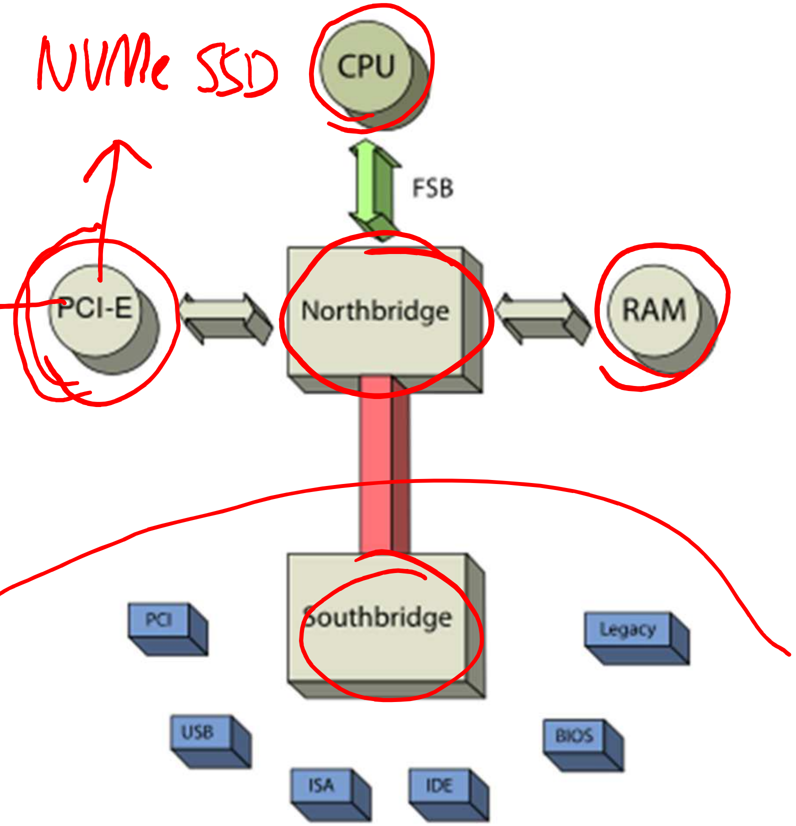


Typical PC Organization

- North bridge (~~memory~~ hub)
  - CPU
  - Memory
  - High-speed PCIe devices, like graphics, NVMe SSDs
- South bridge (IO hub)
  - USB
  - SCSI
  - SATA
  - …

display

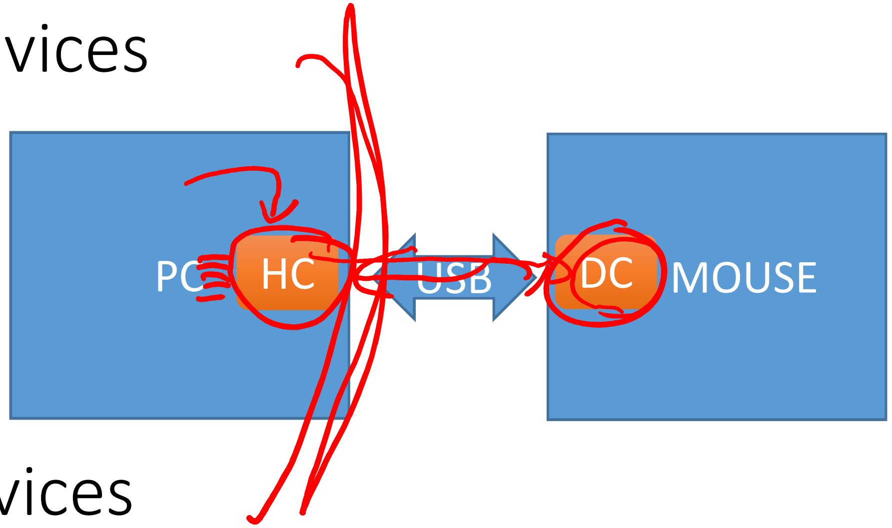
NVMe SSD



https://en.wikipedia.org/wiki/Northbridge_(computing)

I/O Hardware

- Incredible variety of I/O devices
- Common concepts
  - Host controller
  - Bus or interface
  - Device controller
- I/O instructions control devices
  - Direct I/O
  - Memory-mapped I/O



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 address

I/O registers

timer

~~item~~ interrupt

tick

~~Mem addr~~

Example: Set the x86 Interval Timer

```
mov al, 0x36
out 0x43, al     ;tell the PIT which channel we're setting

mov ax, 11931
out 0x40, al     ;send low byte
mov al, ah
out 0x40, al     ;send high byte
```

privilege inst.

direct I/O

in

Channel 0	0x40	System Timer
Channel 1	0x41	DRAM refresh (obsolete)
Channel 2	0x42	Buzzer
Command	0x43	Command I/O register

https://en.wikibooks.org/wiki/X86_Assembly/Programmable_Interval_Timer
https://wiki.osdev.org/Programmable_Interval_Timer

I/O Operation

- I/O devices and CPUs execute concurrently
  - Polling I/O: The CPU waits on an I/O device until completion
  - Interrupt I/O: The CPU will be notified of I/O completion

- Data movement of I/O operations
  - Each device controller has a local buffer
  - Option 1: CPU moves data between main memory and I/O local buffers
  - Option 2: CPU offloads data movement to DMA

Polling

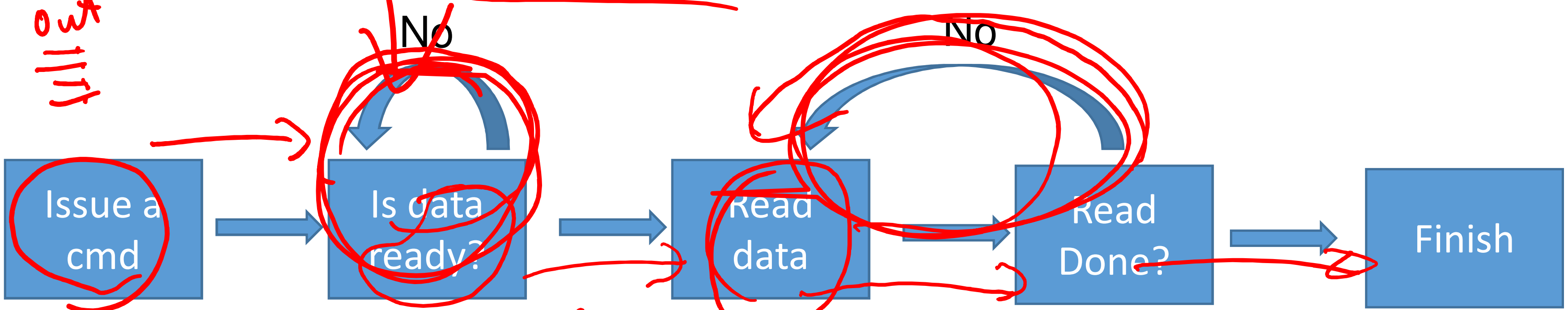
- CPU determines state of device
  - Busy
  - Data ready
  - Error
- Busy-wait cycle to wait for I/O from device

M.2 SSD

Gigabit Eth

"in 0x40"

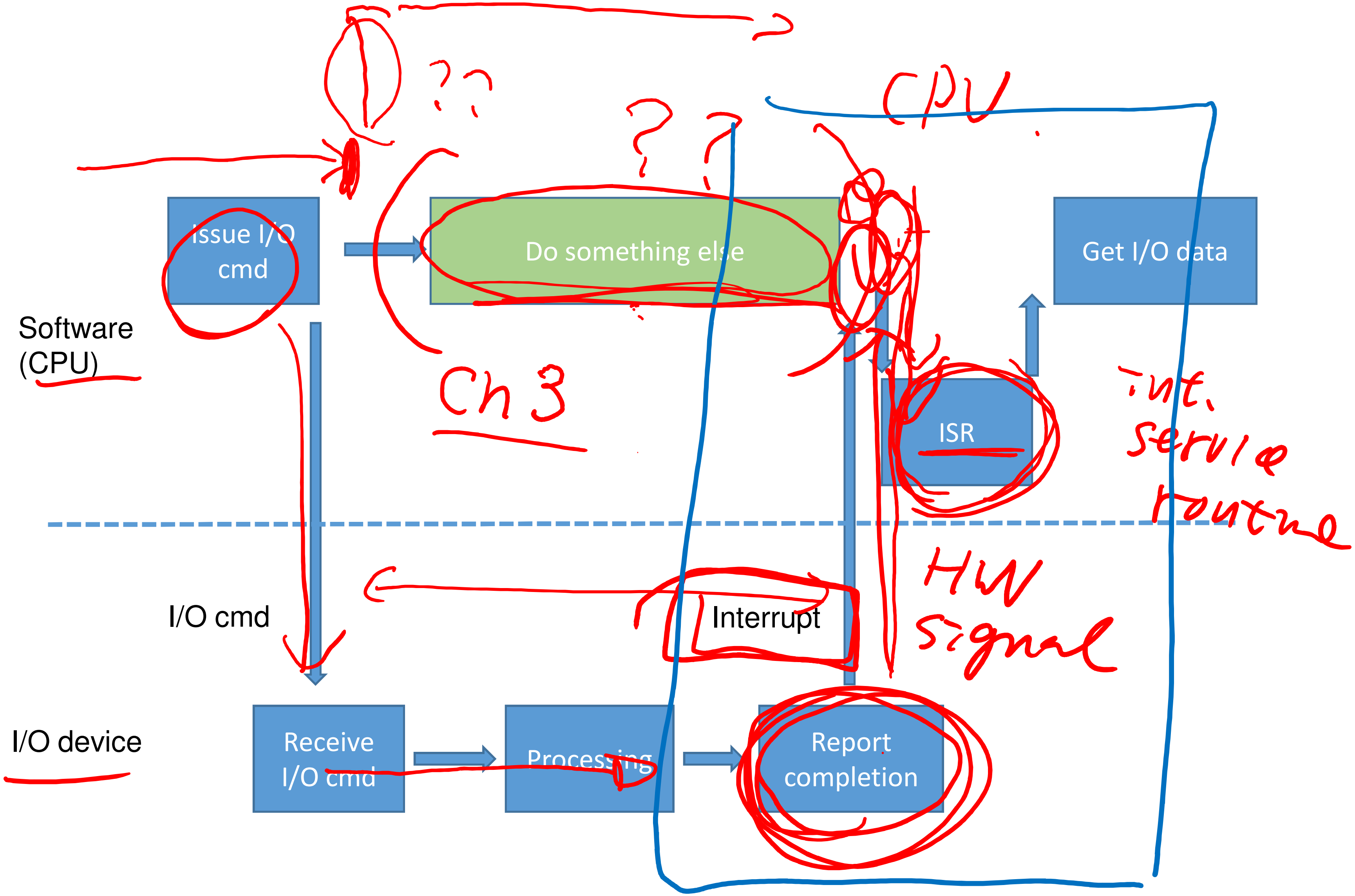
out



delay

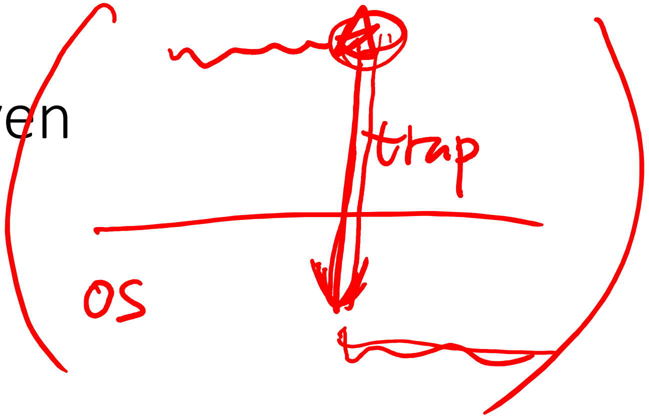
Interrupts

- CPU Interrupt-request line 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 non-maskable
- Interrupt mechanism also used for exceptions


Software (CPU)
Issue I/O cmd
Do something else
Get I/O data
ISR
I/O cmd
Interrupt
I/O device
Receive I/O cmd
Processing
Report completion
??
??
CPU
Ch 3
Int. service routine
HW signal

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
  - A trap is a software-generated interrupt caused either by an error or a user request
  - An IRQ is generated by hardware
- An operating system is interrupt driven



- Hardware: interrupt request (IRQ)
  - I/O completion, timer, etc
- Software: Trap
  - divide by zero, access violation, system call

不正常

正常 trap

Legacy PC interrupts

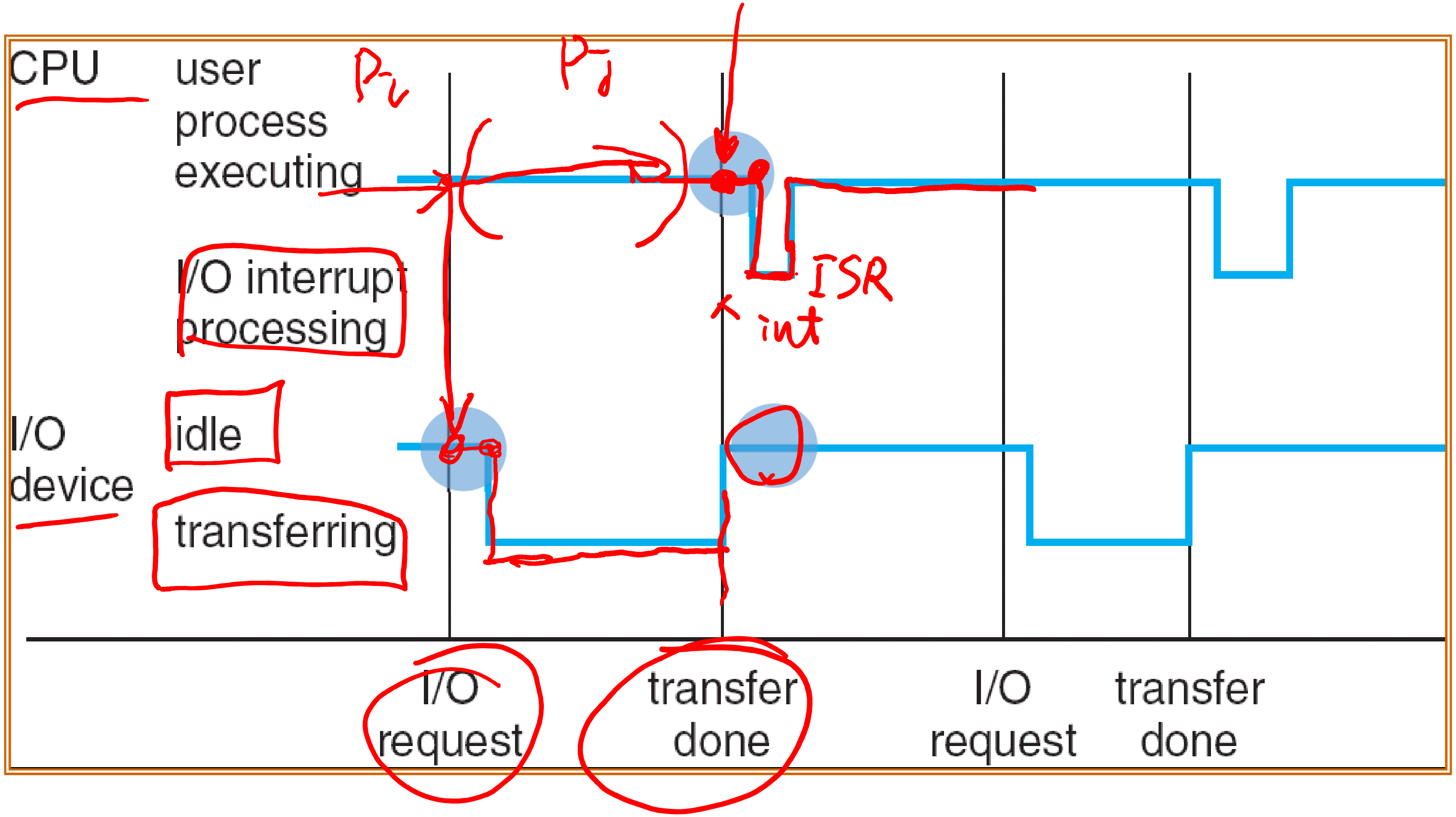
Interrupt	Function
00h	Divide By Zero
01h	Single Step
02h	Nonmaskable Interrupt (NMI)
03h	Breakpoint Instruction
04h	Overflow Instruction
05h	Print Screen
05h	Bounds Exception (80286, 80386)
06h	Invalid Op Code (80286, 80386)
07h	Math Coprocessor Not Present
08h	Double Exception Error (80286, 80386) (AT Only)
08h	System Timer - IRQ 0
09h	Keyboard - IRQ 1
09h	Math Coprocessor Segment Overrun (80286, 80386) (AT Only)
0Ah	IRQ 2 - Cascade from Second programmable Interrupt Controller
0Ah	Invalid Task Segment State (80286, 80286) (AT Only)
0Ah	IRQ 2 - General Adapter Use (PC Only)
0Bh	IRQ 3 - Serial Communications (COM 2)
0Bh	Segment Not Present (80286, 80386)
0Ch	IRQ 4 - Serial Communications (COM 1)
0Ch	Stack Segment Overflow (80286, 80386)
0Dh	Parallel Printer (LPT 2) (AT Only)
0Dh	IRQ 5 - Fixed Disk (XT Only)
0Dh	General Protection Fault (80286, 80386)
0Eh	IRQ 6- Diskette Drive Controller
0Eh	Page Fault (80386 Only)
0Fh	IRQ 7 - Parallel printer (LPT 1)

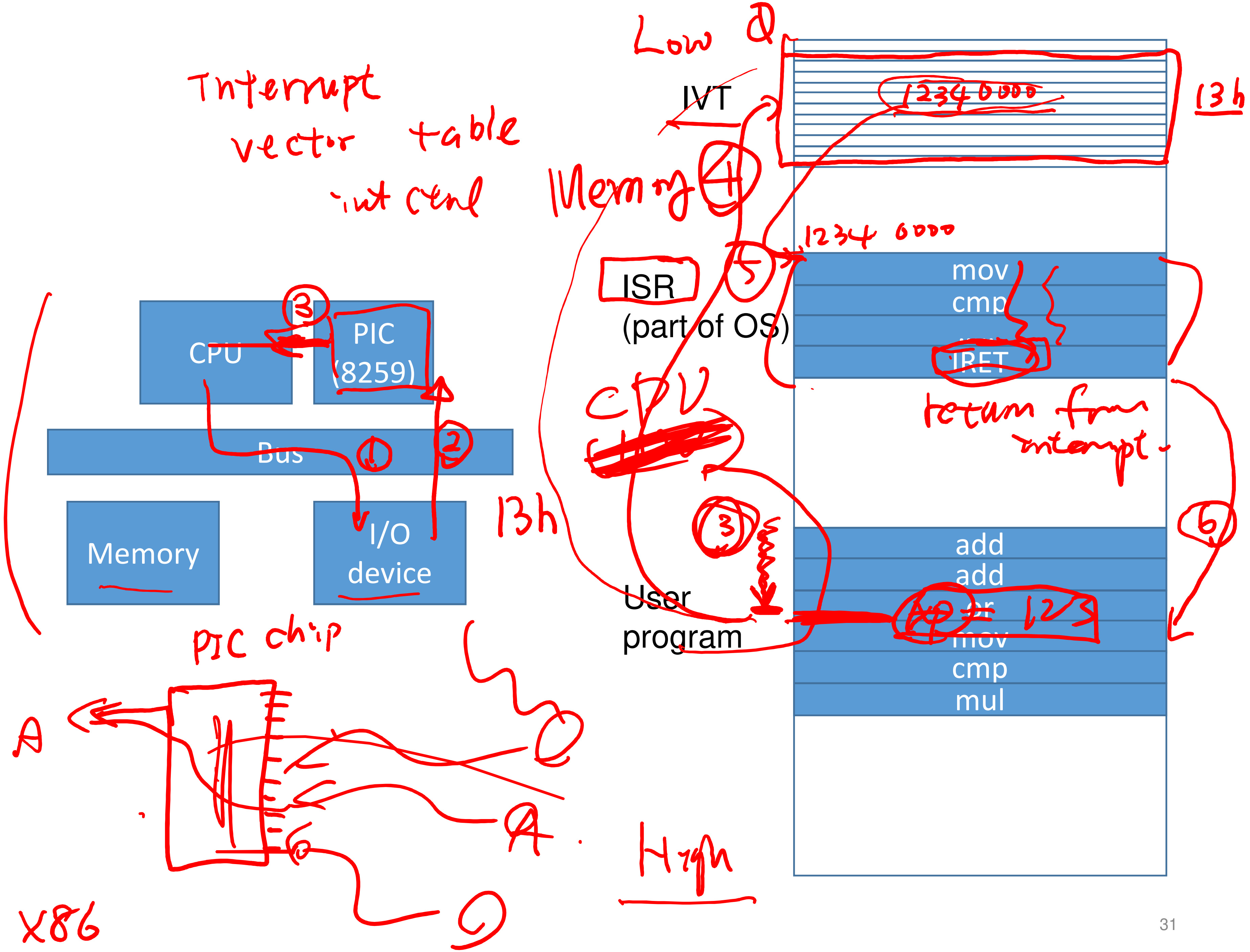
⋮

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:
  - Reading I/O registers
  - vectored interrupt system
- Separate segments of code determine what action should be taken for each type of interrupt

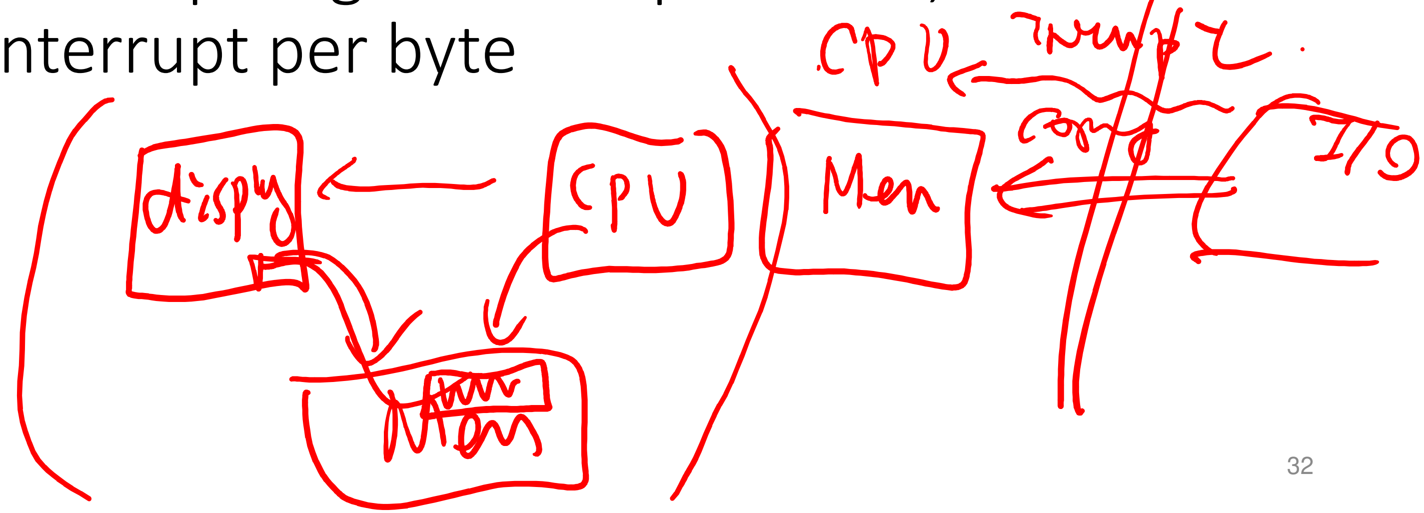
Interrupt Timeline



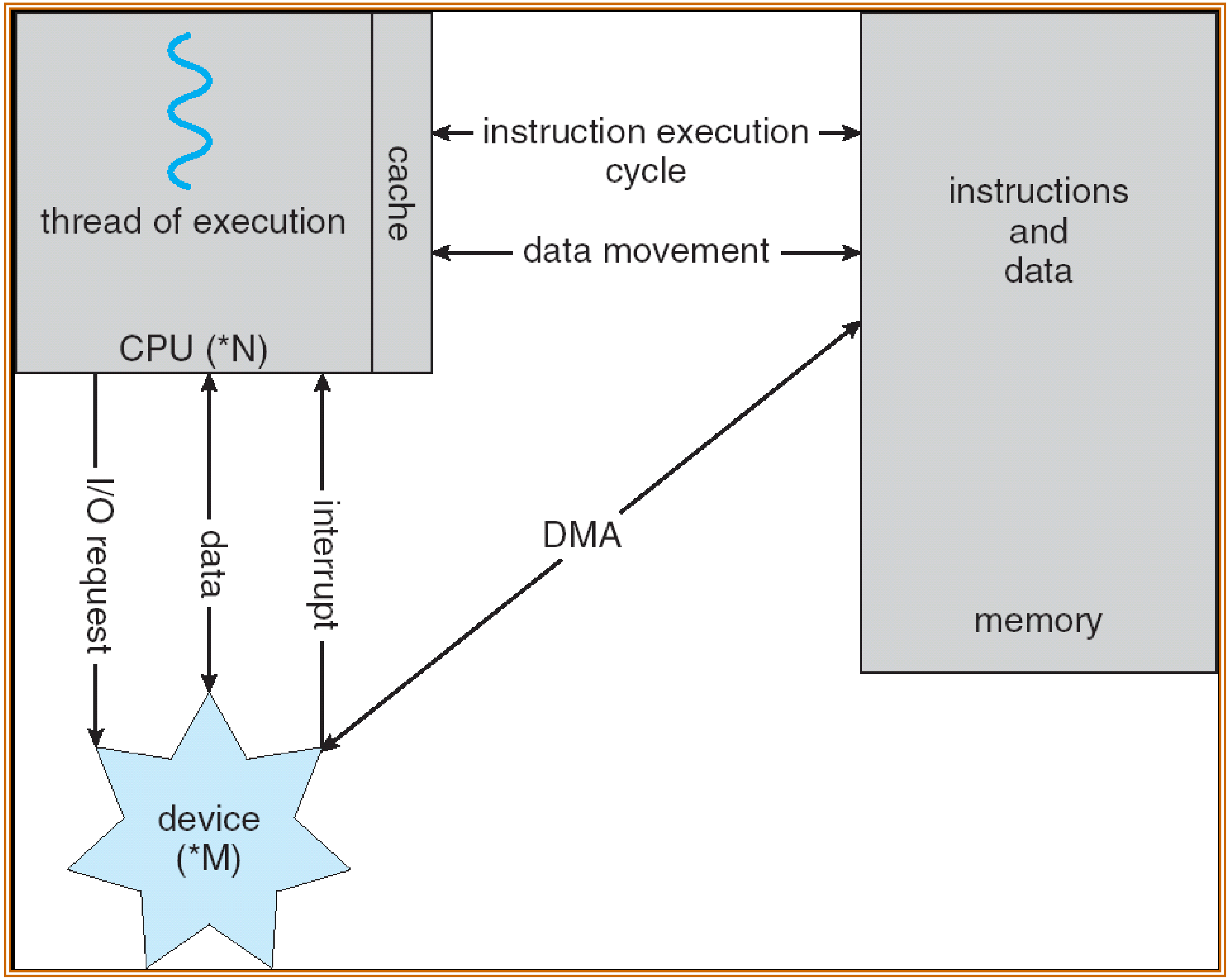

Interrupt vector table
int cntl
CPU
PIC
(8259)
Bus
Memory
I/O
device
PIC chip
A
X86
13h
Low @
IVT
Memory
1234 0000
13h
ISR
(part of OS)
mov
cmp
IRET
return from interrupt
CPU
add
add
mov
cmp
mul
User
program
High

Direct Memory Access

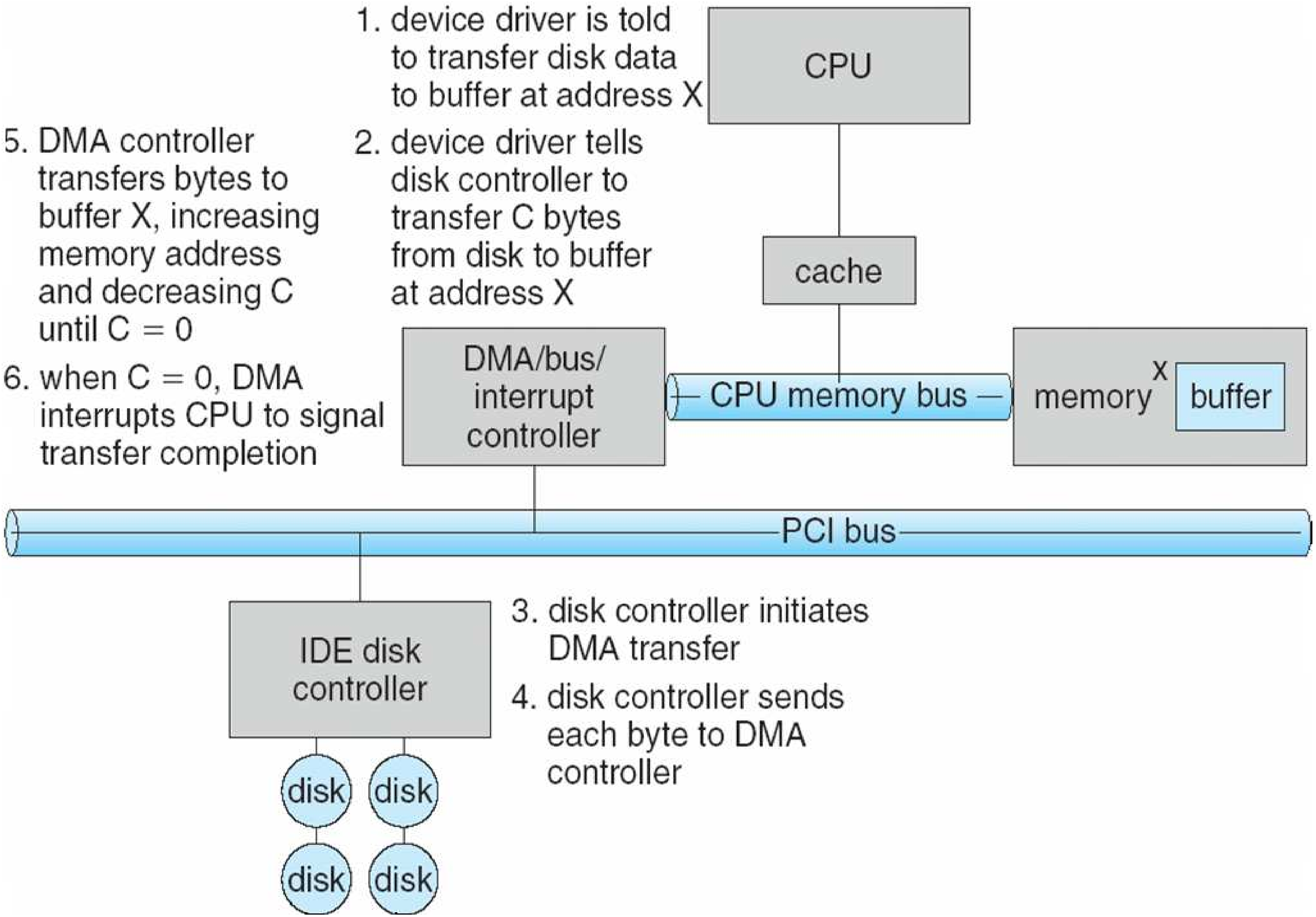
- Used for high-speed I/O devices able to transmit information at close to memory speeds
- Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention
- Only on interrupt is generated per block, rather than the one interrupt per byte



105



thread of execution
cache
CPU (*N)
instruction execution cycle
data movement
instructions and data
memory
I/O request
data
interrupt
DMA
device (*M)


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

From Ch13

Synchronous I/O vs. Asynchronous I/O

- After I/O starts, control returns to the program only upon I/O completion (blocking call; sync I/O)
  - Example: regular I/O reading
    read() or fread()

- After I/O starts, control returns to the program without waiting for I/O completion (non-blocking call; async I/O)
  - Example: regular I/O writing
    write() or fwrite()

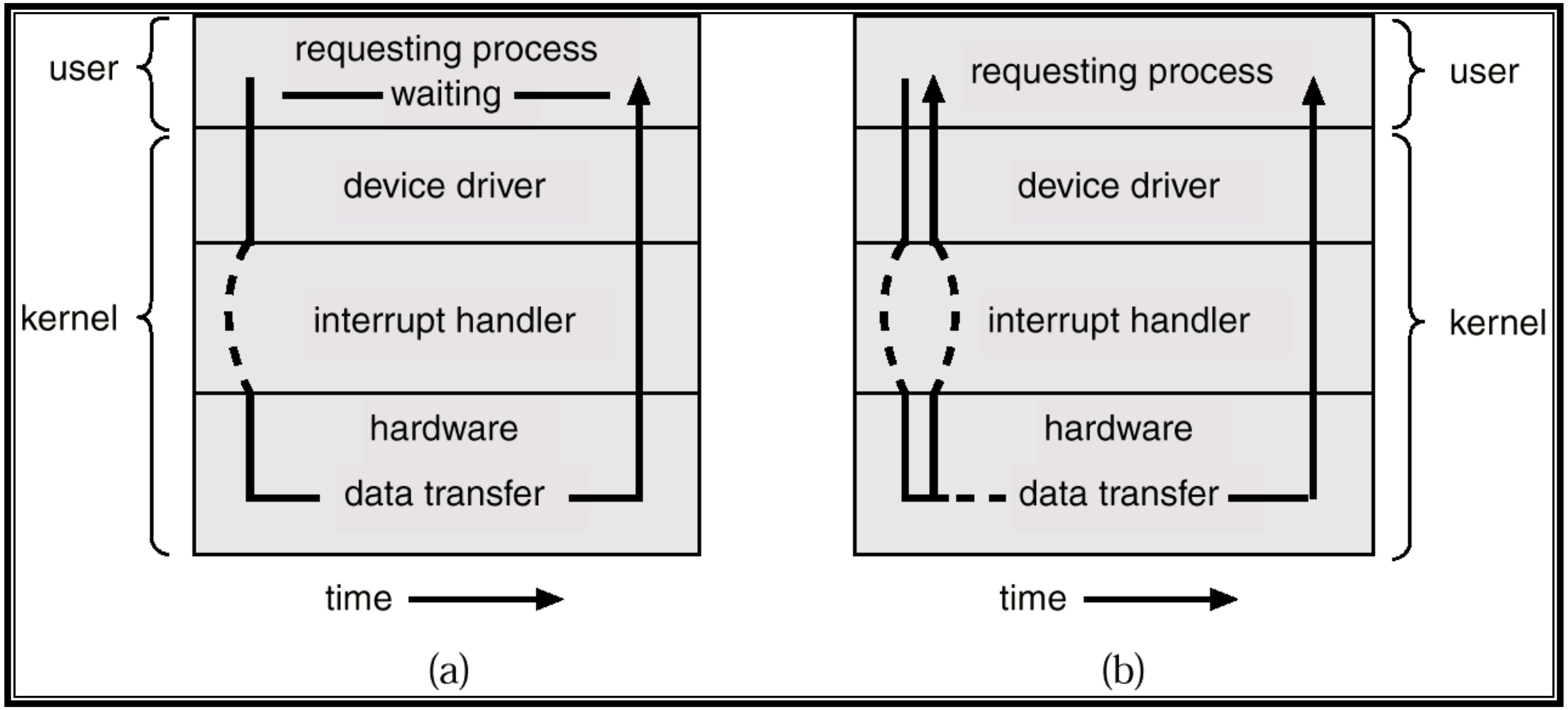
I/O Structure (Async vs Sync)

Synchronous

Blocking I/O

Asynchronous

Non-blocking I/O



- Are the following operations synchronous or asynchronous?
  - fread()
  - fwrite()
  - fsync()
  - aio_read()

End of Chapter 1