Questionnaire & Join MS teams today

- Please help me understand you more

Questionnaire (before 09/13)

How to apply MS account

- And, Join MS teams right after the class
- Code: ig0604f

國立陽明交通大學
NATIONAL YANG MING CHIAO TUNG UNIVERSITY

Institute of Artificial Intelligence Innovation
Department of Computer Science

Operating System

Lecture 01: Introduction

Shuo-Han Chen (陳碩漢),
shch@nycu.edu.tw

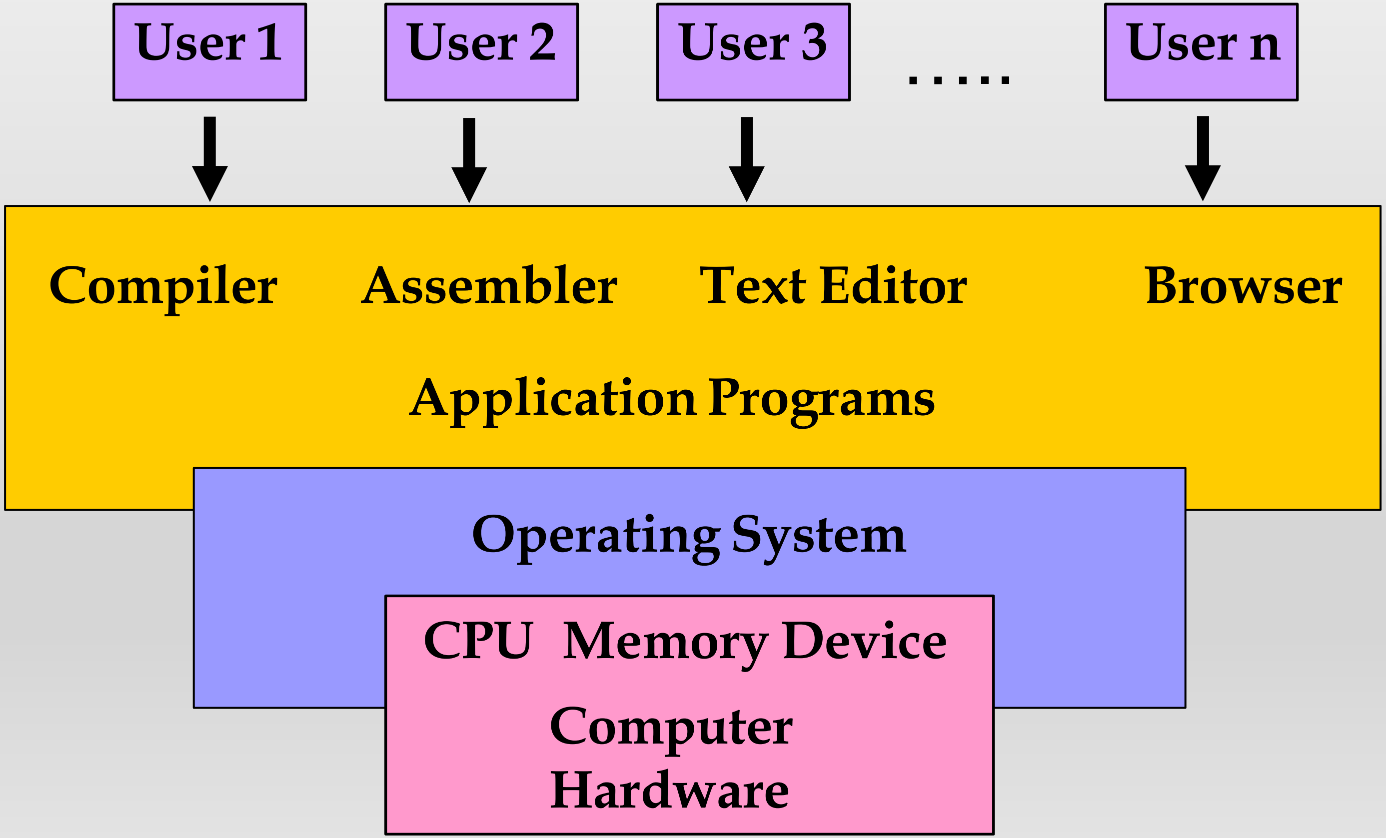
Wed. 10:10 - 12:00 EC115 +
Fri. 11:10 – 12:00 Online

Course Schedule

W	Date	Lecture	Online	Homework
1	Sept. 4	Lec01: Couse Overview & Historical Prospective		
2	Sept. 11	Lec02: Introduction	V	
3	Sept. 18	Lec03: OS Structure	V	HW01
4	Sept. 25	Lec04: Processes Concept	V	
5	Oct. 2	Lec08: Memory Management	V	
6	Oct. 9	Lec09: Virtual Memory Management	V	HW02
7	Oct. 16	Lec05: Process Scheduling	V	
8	Oct. 23	School Midterm Exam		
9	Oct. 30	Lec06: Process Synchronization	V	
10	Nov. 6	Lec07: Deadlocks	V	HW03
11	Nov. 13	Lec10: File System Interface	V	
12	Nov. 20	School Event – No class		
13	Nov. 27	Lec11: File System Implementation	V	HW04
14	Dec. 4	Lec12: Mass Storage System	V	
15	Dec. 11	Lec13: IO Systems	V	
16	Dec. 18	School Final Exam		

Computer System

- Four components:
  - Hardware, OS, Application, User

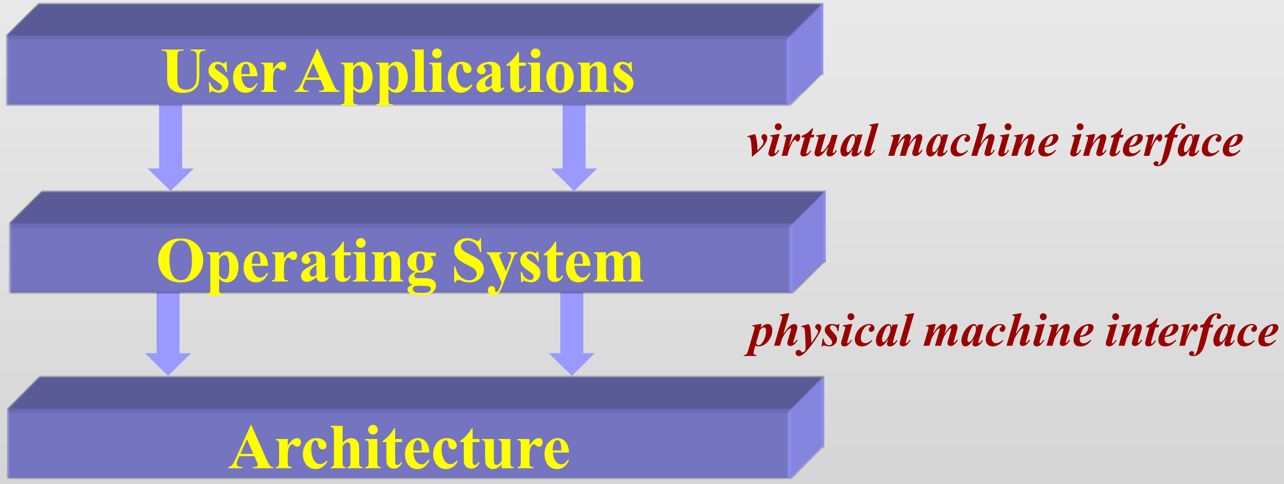


Computer System

- **User** — people, machines, other computers
- **Application** — define the ways in which the system resources are used to solve the computing problems
- **Operating System** — controls and coordinates the use of the hardware/resources
- **Hardware** — provides basic computing resources (CPU, memory, I/O devices)

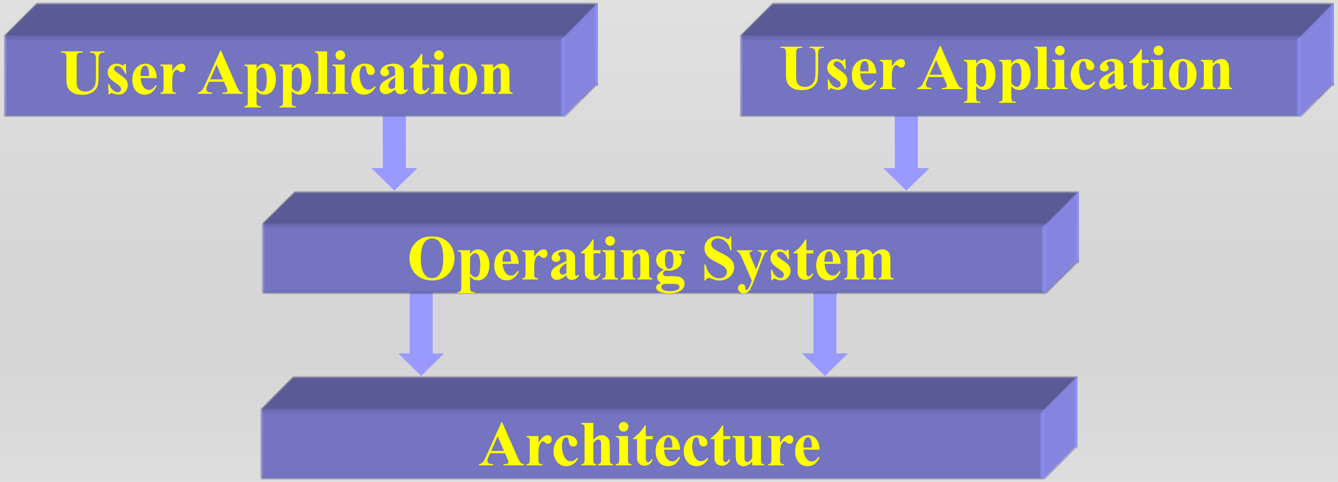
What is an Operating System

- An operating system is the "*permanent*" software that controls/abstracts hardware resources for user applications

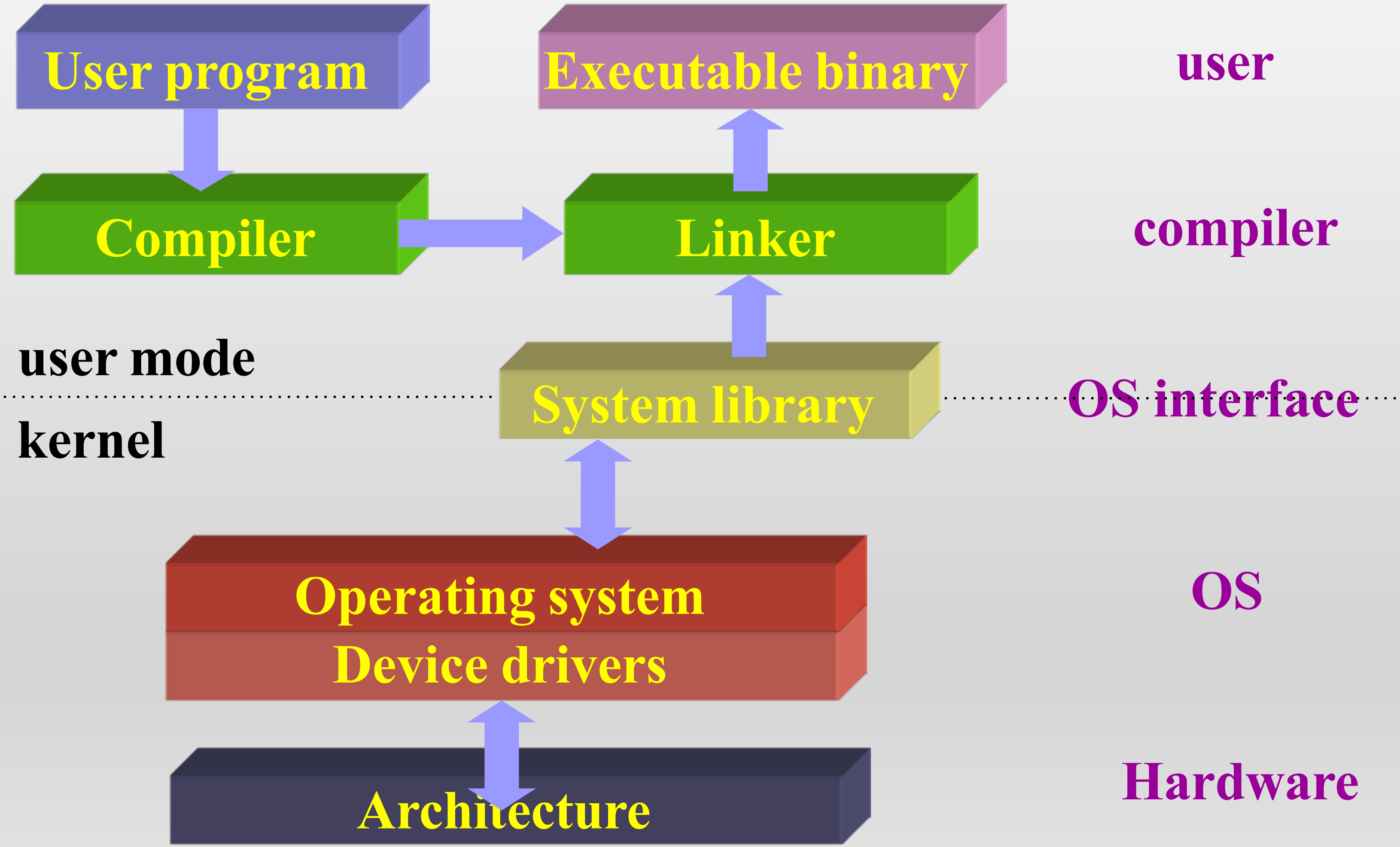


Multi-tasking Operating Systems

- Manages resources and processes to support different user applications
- Provides Applications Programming Interface (API) for user applications



General-Purpose Operating Systems



Definition of an Operating System

- **Resource allocator** – manages and allocates resources to insure efficiency and fairness
- **Control program** – controls the execution of user programs and operations of I/O devices to prevent errors and improper use of computer
- **Kernel** – the one program running at all times (all else being system/application programs)
    - No universally accepted definition

Goals of an Operating System

- **Convenience**
  - make computer system easy to use and compute
  - In particular for small PC
- **Efficiency**
  - use computer hardware in an efficient manner
  - Especially for large, shared, multiuser systems

- Two goals are sometimes contradictory
- In the past, efficiency is more important

Importance of an Operating System

- System API are the *only* interface between user applications and hardware
  - API are designed for general-purpose, not performance driven
- OS code cannot allow any bug
  - Any break (e.g. invalid access) causes reboot
- The owner of OS technology controls the software & hardware industry
- Operating systems and computer architecture influence each other

Modern Operating Systems

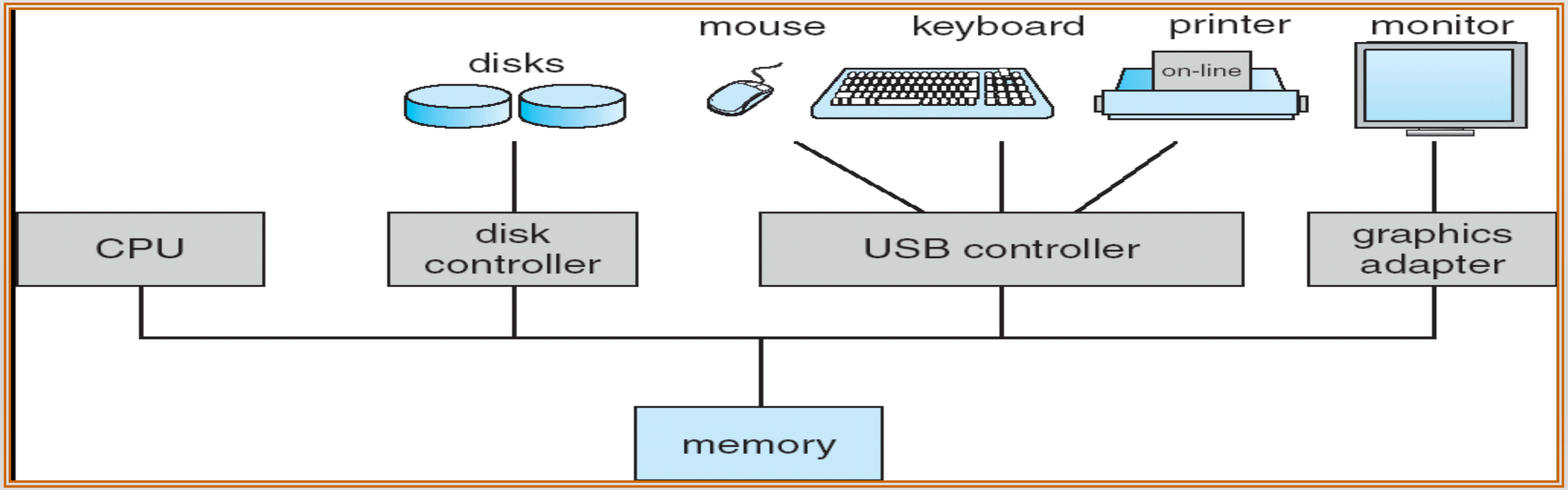
- x86 platform
    - Linux (CentOS, Redhat, openSUSE, Ubuntu, etc)
    - Windows (Windows10, XP, 2000, etc)
- PowerPC platform – Mac OS
- Smartphone Mobile OS
    - Android, iOS, Windows10 Mobile, Ubuntu Touch
- Embedded OS
    - Embedded Linux(Andriod, WebOS), Windows CE
    - Raspberry Pi, Xbox, etc

Review Slides (1)

- Definition of OS?
- Goals of OS?
- Importance of OS?

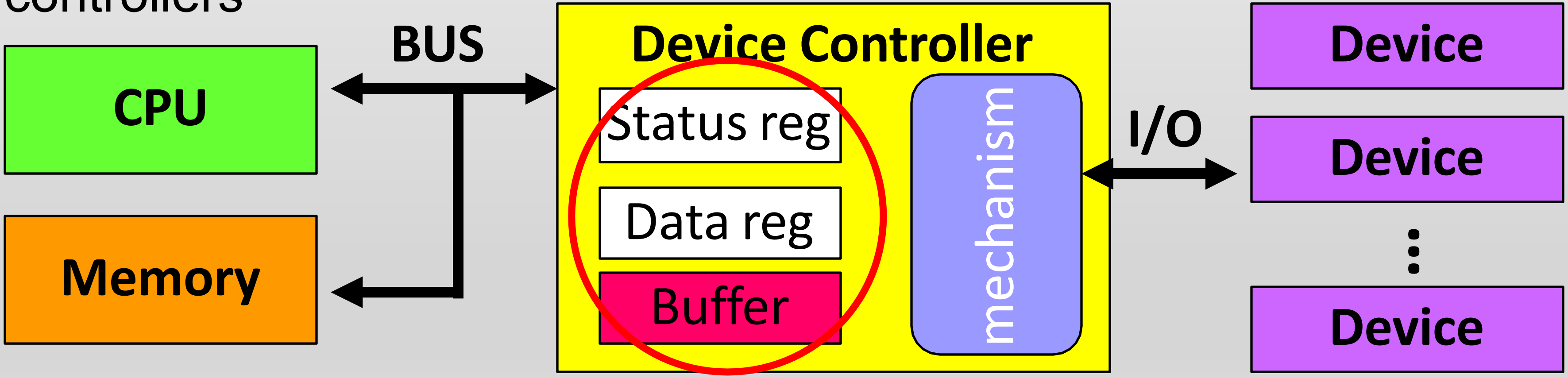
Computer-System Organization

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



Computer-System Operations

- Each device controller is in charge of a particular device type
- Each device controller has a local buffer
- I/O is from the device to controller's local buffer
- CPU moves data from/to memory to/from local buffers in device controllers



Busy/wait output

- Simplest way to program device
  - Use instructions to test when device is ready

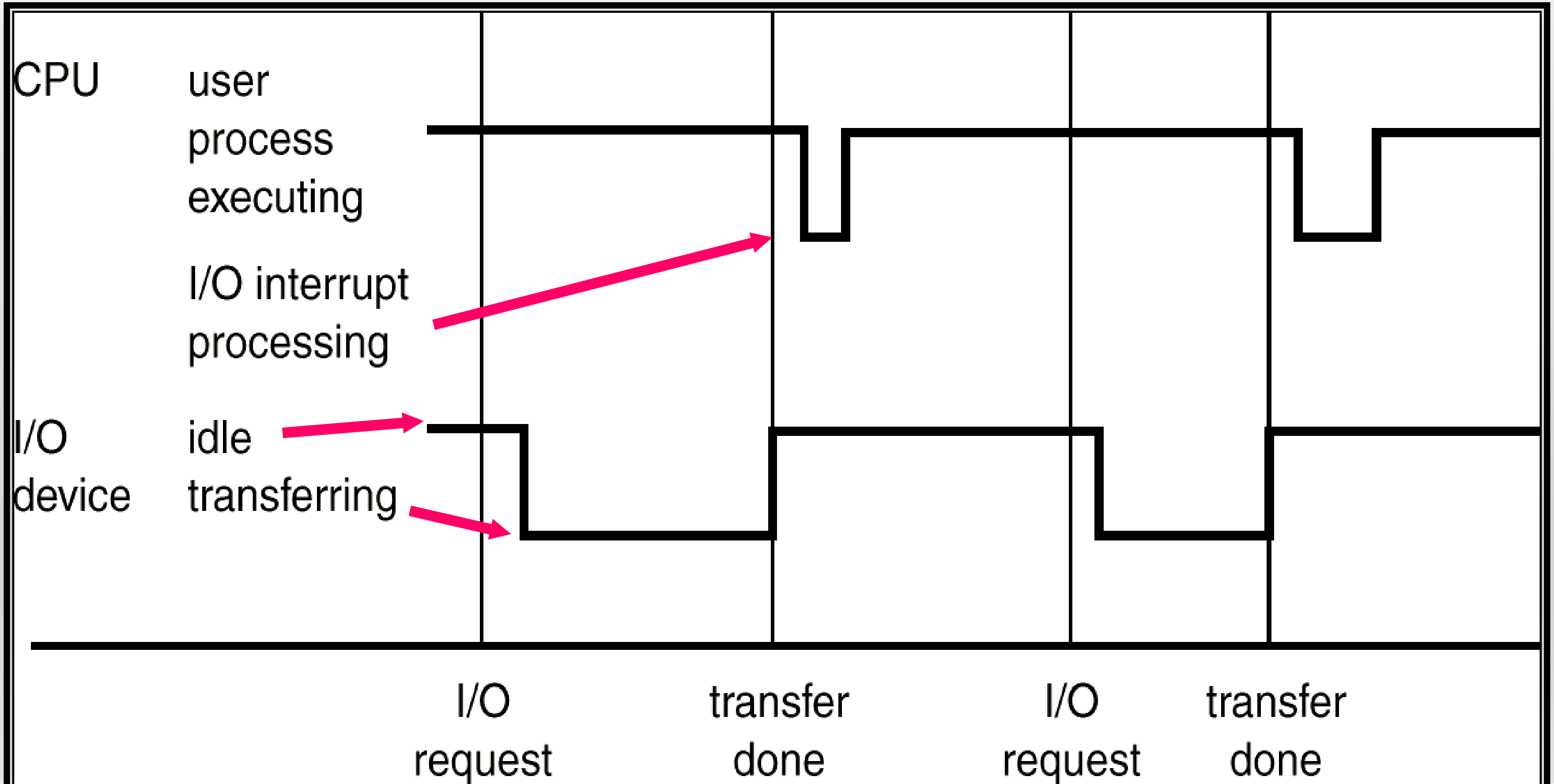
```
#define OUT_CHAR 0x1000          // device data register
#define OUT_STATUS 0x1001        // device status register

current_char = mystring;
while (*current_char != ‘\0’) {
        poke(OUT_CHAR,*current_char);
        while (peek(OUT_STATUS) != 0); // busy waiting
        current_char++;
}
```

Interrupt I/O

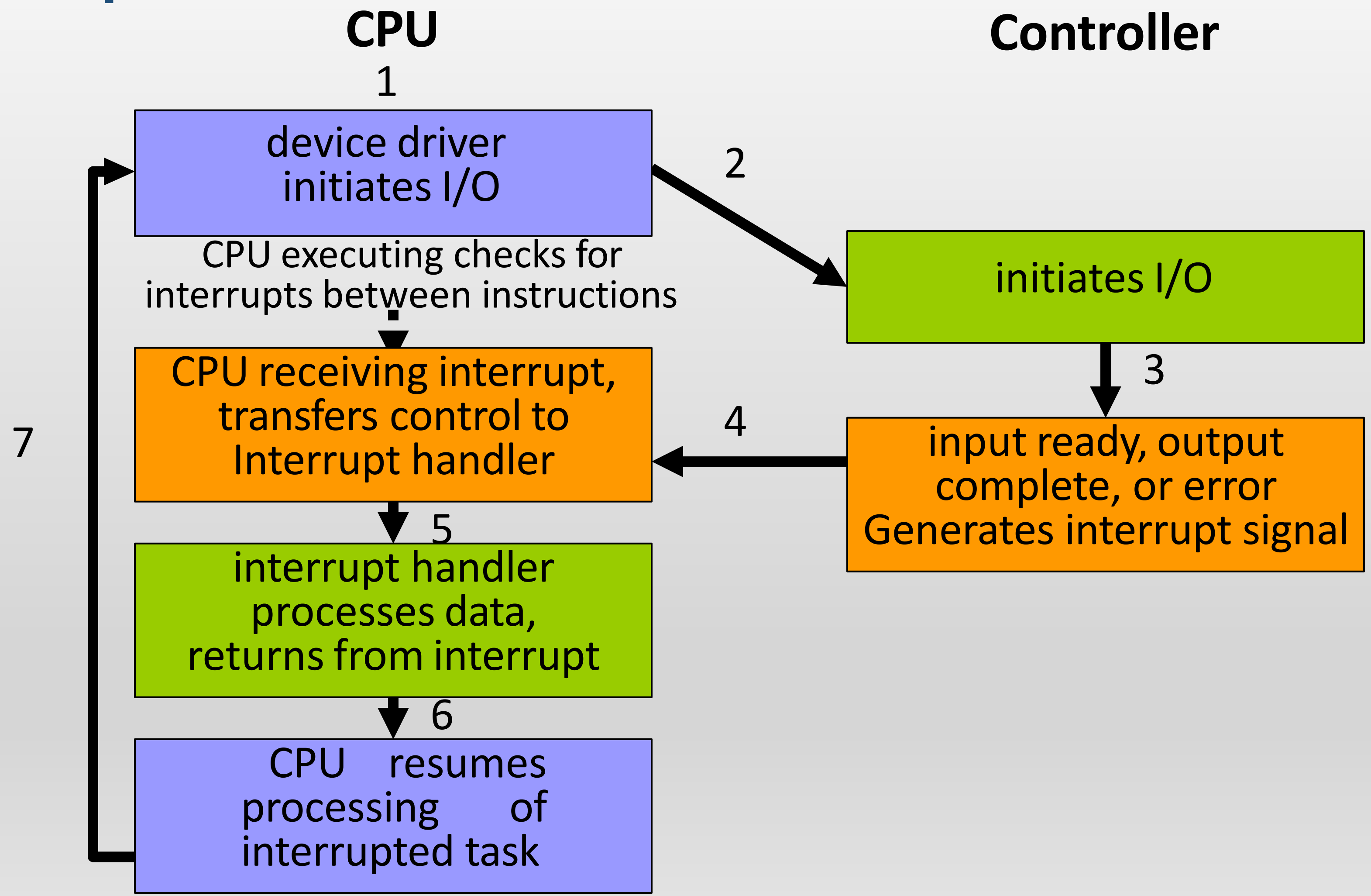
- Busy/wait is very inefficient
  - CPU can’t do other work while testing device
  - Hard to do simultaneous I/O
- Interrupts allow a device to **change the flow of control in the CPU**
  - Causes subroutine call to handle device

Interrupt I/O Timeline

- Interrupt time line for I/O on a single process



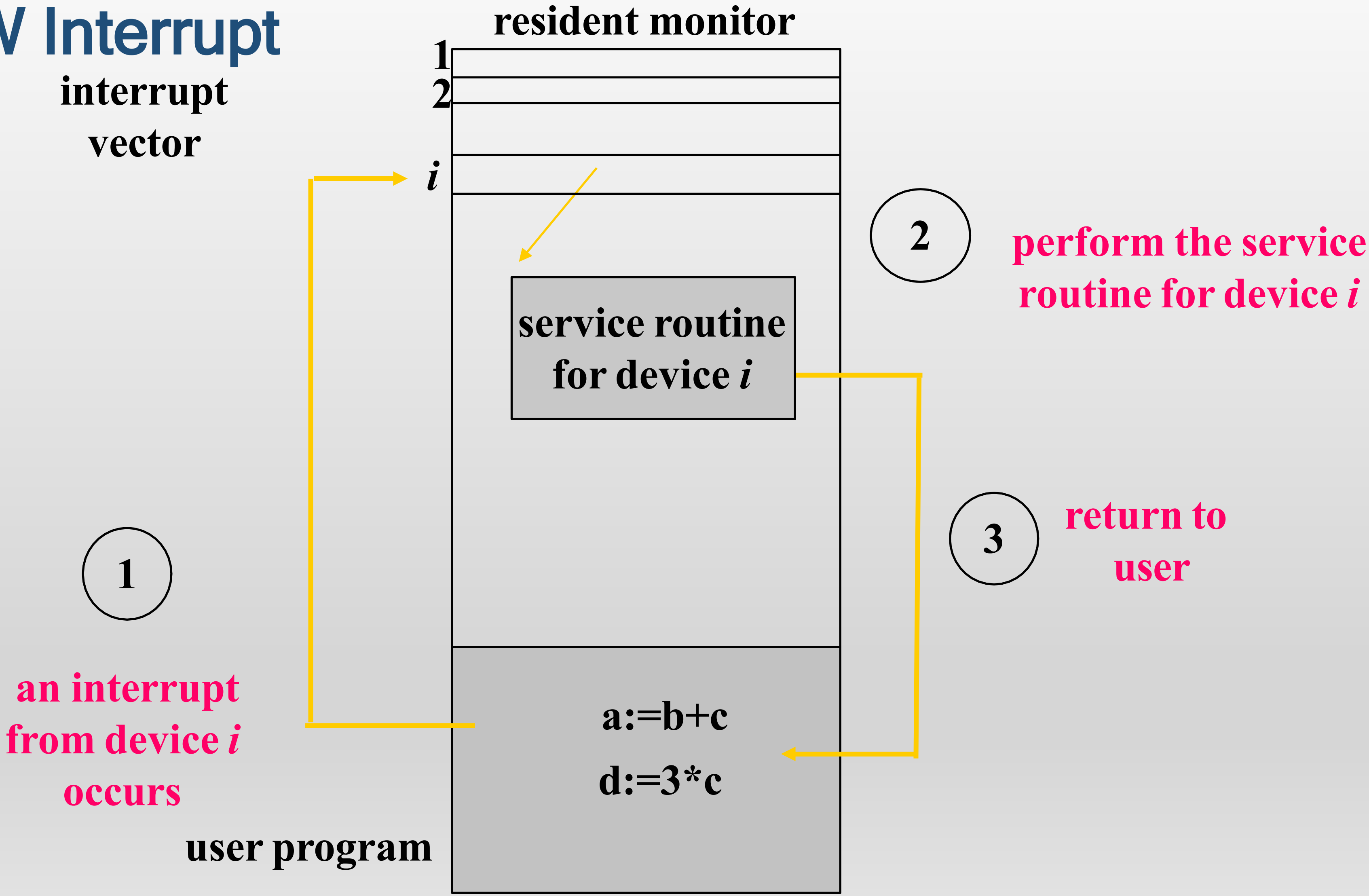
Interrupt-Driven I/O



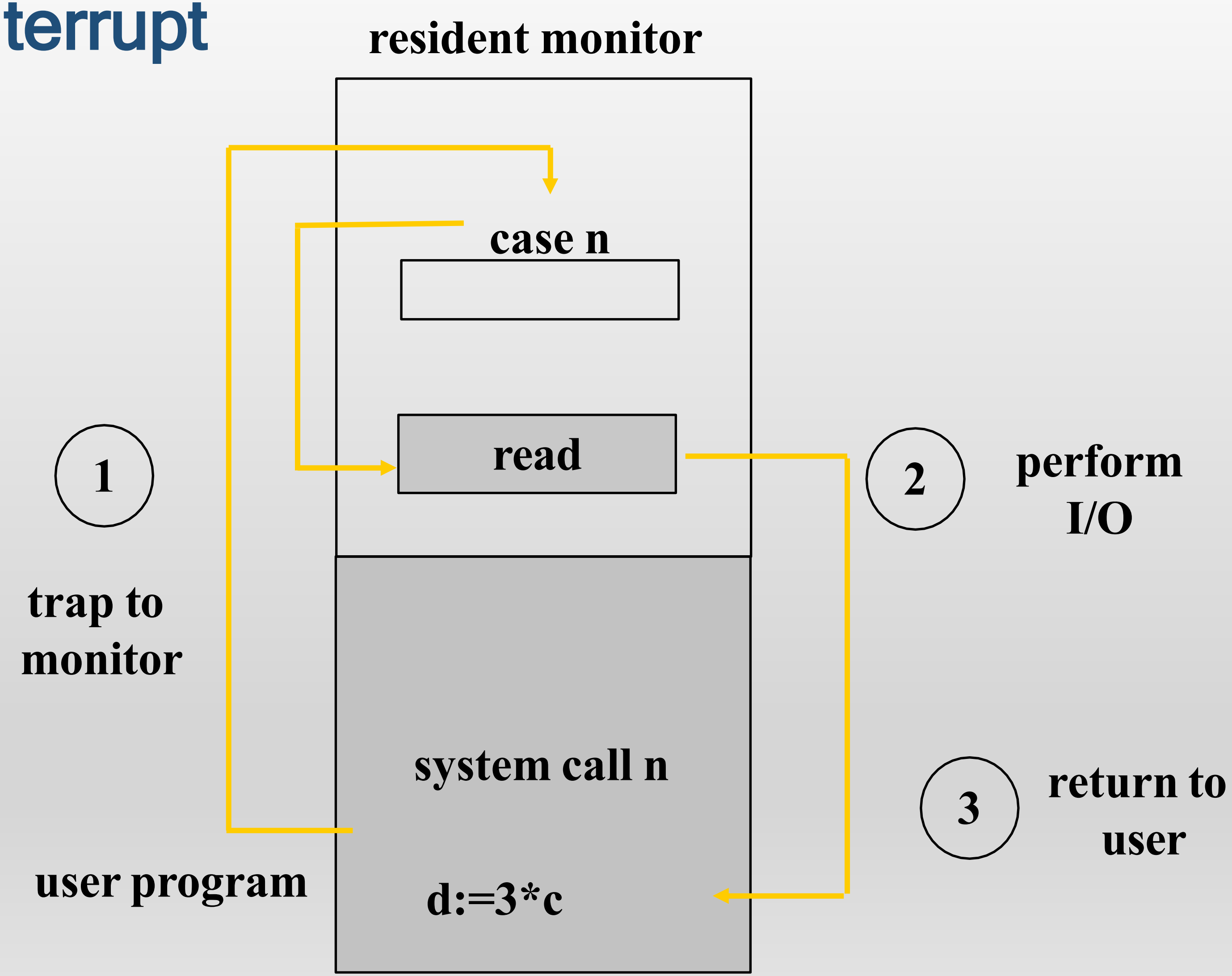
Interrupt

- Modern OS are interrupt driven
- The occurrence of an event is signaled by an interrupt from either hardware or software.
    - Hardware may trigger an interrupt at any time by sending a **signal** to CPU
    - Software may trigger an interrupt either by an error (division by zero or invalid memory access) or by a user request for an operating system service (**system call**)
- Software interrupt also called trap

HW Interrupt



SW Interrupt



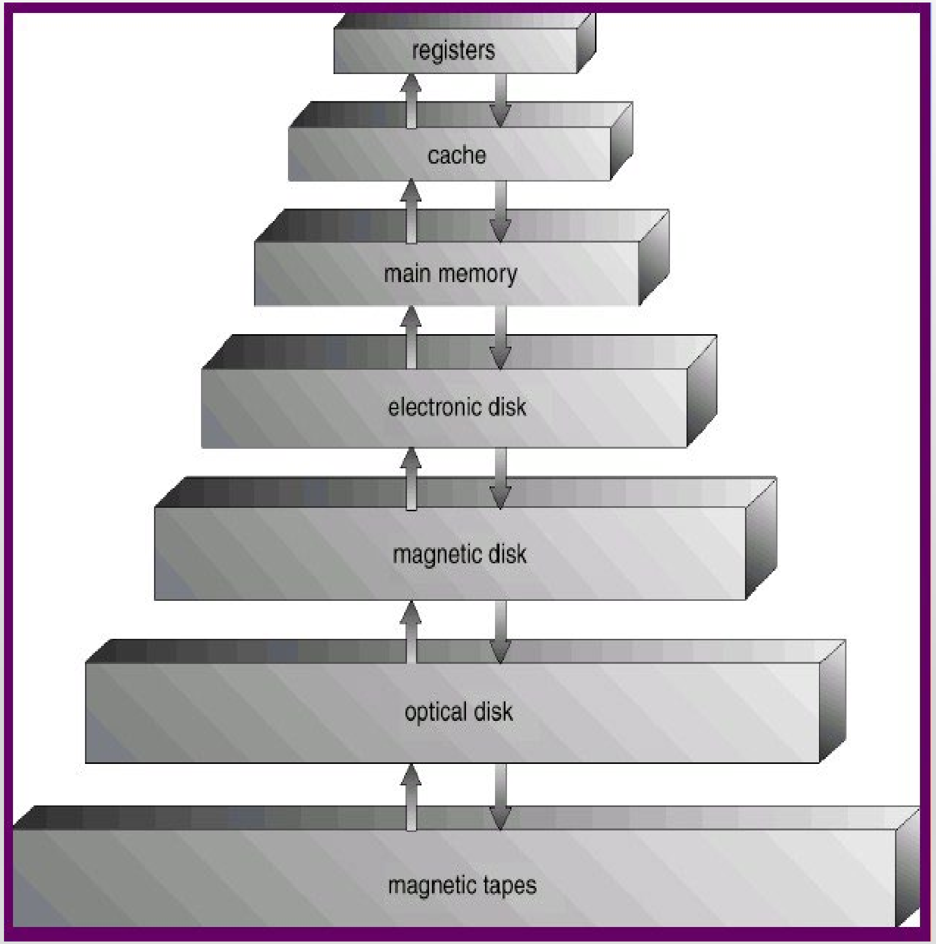
Common Functions of Interrupts

- Interrupt transfers control to the interrupt service routine generally, through the ***interrupt vector***, which contains the **addresses** (function pointer) of all the service (i.e. interrupt handler) 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

Review Slides (2)

- What is interrupt and how does it work?
- What is the difference between trap and interrupt?

Storage-Device Hierarchy



Storage-Device Hierarchy

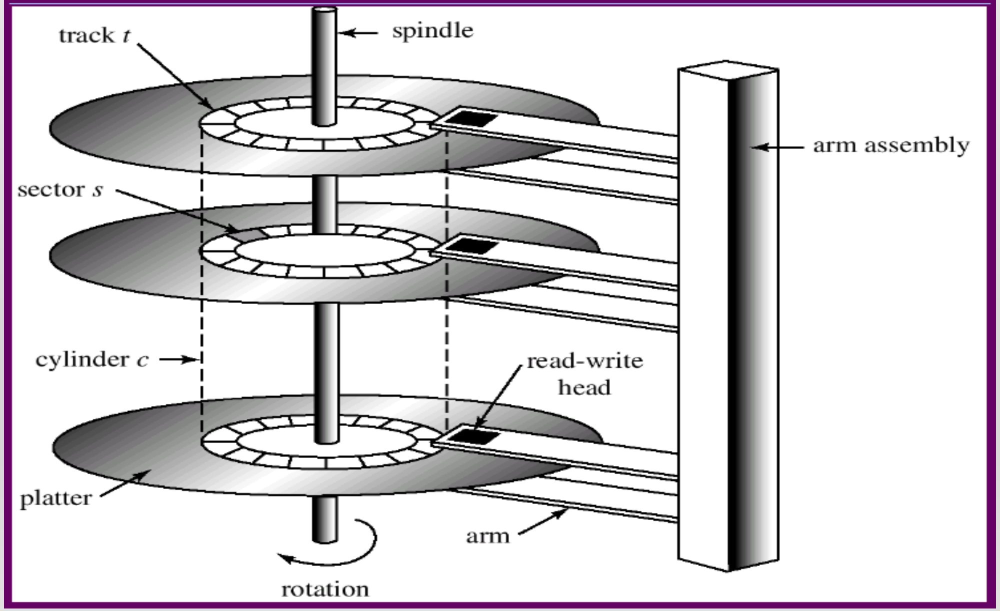
- Storage systems organized in hierarchy.
  - Speed , Cost, Volatility
- **Main memory** – only large storage media that the CPU can access directly
  - RAM: Random Access Memory
- Secondary storage – extension of main memory that provides large nonvolatile storage capacity
  - Magnetic disk

RAM: Random-Access Memory

- DRAM (Dynamic RAM):
  - Need only one transistor
  - Consume less power
  - values must be periodically refreshed
  - Access Speed: >= 30ns
- SRAM (Static RAM):
  - Need six transistors
  - Consume more power
  - Access Speed: 10ns~30ns
  - usage: cache memory

Disk Mechanism

- Speed of magnetic disk
    - Transfer time = data size / transfer rate
    - Positioning time (random access time)
        - seek time (cylinder) + rotational latency (sector)

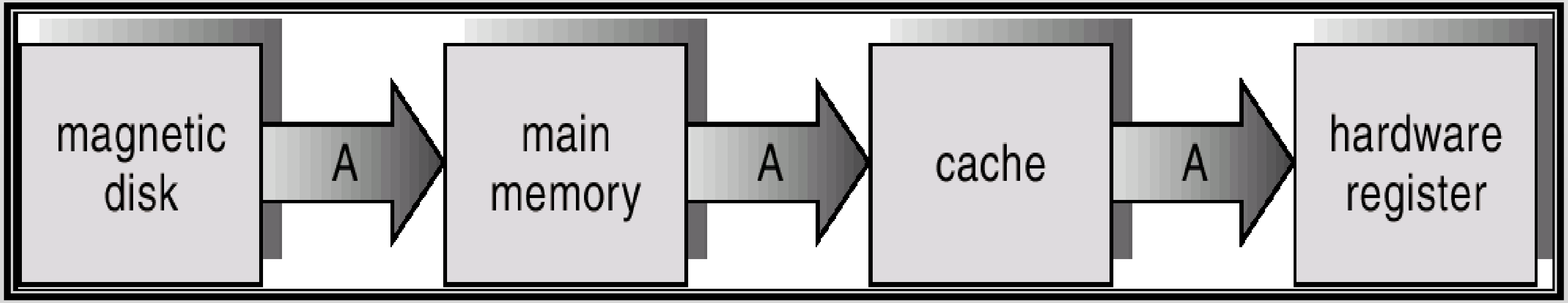


Performance of Various Levels of Storage

Level	1	2	3	4
Name	registers	cache	main memory	disk storage
Typical size	< 1 KB	> 16 MB	> 16 GB	> 100 GB
Implementation technology	custom memory with multiple ports, CMOS	on-chip or off-chip CMOS SRAM	CMOS DRAM	magnetic disk
Access time (ns)	0.25 – 0.5	0.5 – 25	80 – 250	5,000.000
Bandwidth (MB/sec)	20,000 – 100,000	5000 – 10,000	1000 – 5000	20 – 150
Managed by	compiler	hardware	operating system	operating system
Backed by	cache	main memory	disk	CD or tape

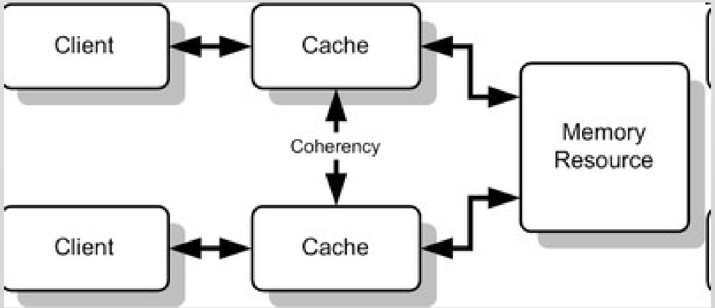
Caching

- Information in use copied from slower to faster storage temporarily
- Faster storage (cache) checked first to determine if information is there
    - If it is, information used directly from the cache (fast)
    - If not, data copied to cache and used there



Coherency and Consistency Issue

- The same data may appear in different levels
  - Issue: Change the copy in register make it inconsistent with other copies
- Single task accessing:
  - No problem, always use the Highest level copy
- Multi-task accessing:
  - Need to obtain the most recent value
- Distributed system:
  - Difficult b.c. copies are on different computers



Review Slides (3)

- Why storage hierarchy?
- Caching? involved issues?

Hardware Protection

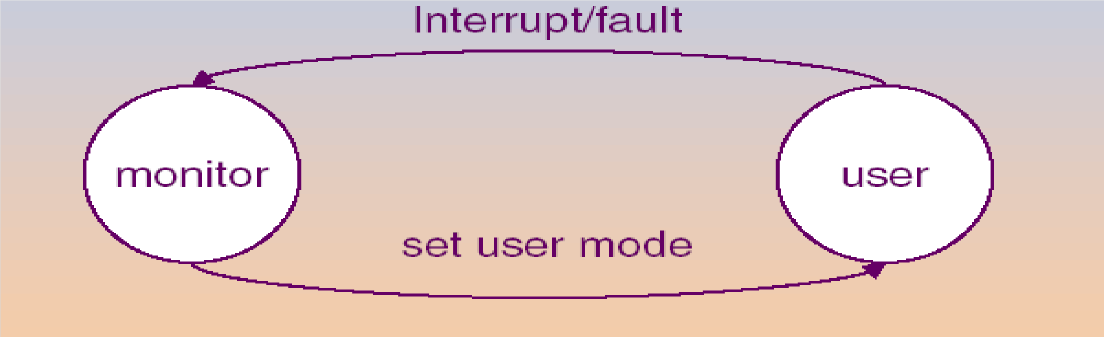
- Dual-Mode Operation
- I/O Protection
- Memory Protection
- CPU Protection

Dual-Mode Operation

- What to protect?
  - Sharing system resources requires OS 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
  - User mode – execution done on behalf of a user
  - Monitor mode (also kernel mode or system mode) – execution done on behalf of operating system

Dual-Mode Operation (Cont'd)

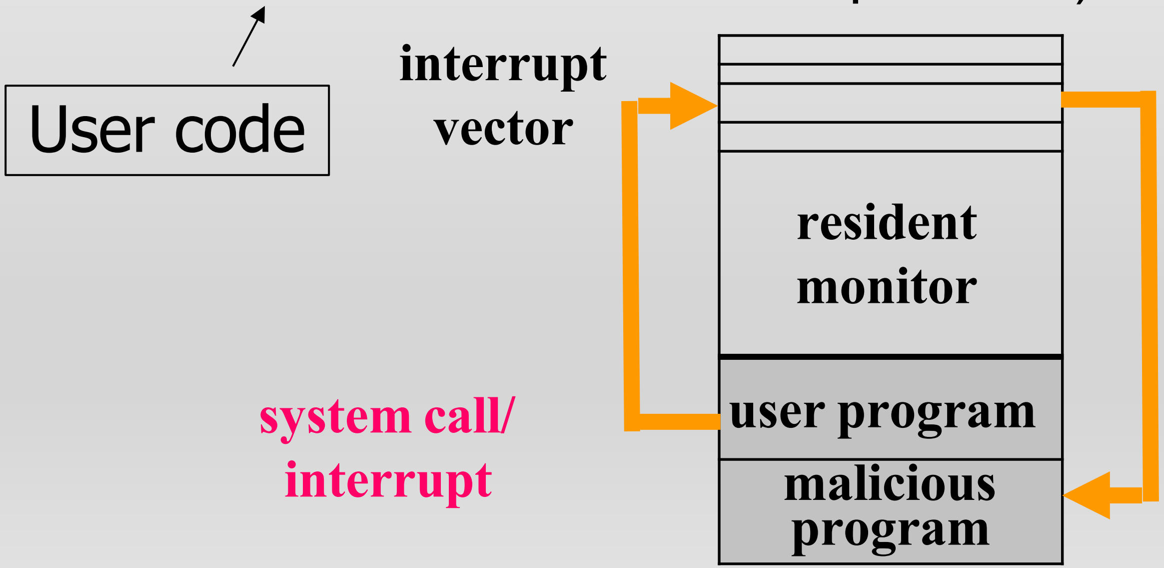
- Mode bit added to computer hardware to indicate the current mode: kernel (0) or user (1)
- When an interrupt/trap or fault occurs, hardware switches to monitor mode



- Privileged instructions
  - Executed only in monitor mode
  - Requested by users (system calls)

I/O Protection

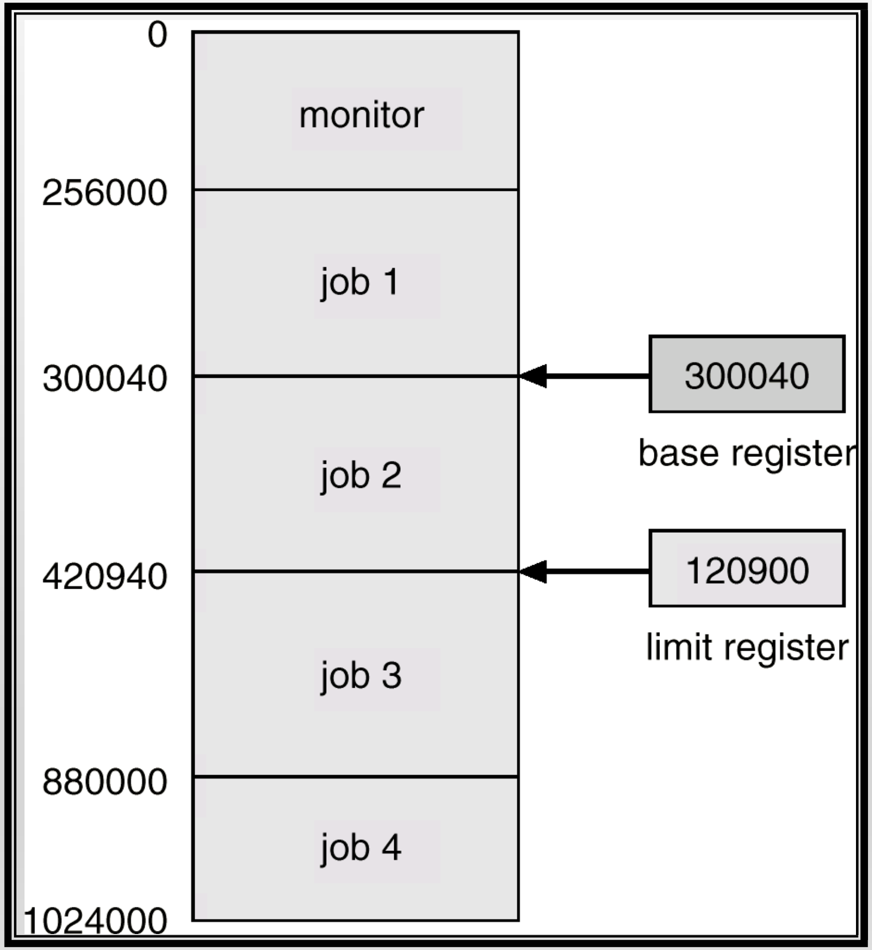
- All I/O instructions are privileged instructions
  - Any I/O device is shared between users
- Must ensure that a user program could never gain control of the computer in monitor mode (i.e., a user program that, as part of its execution, stores a **new address** in the interrupt vector)



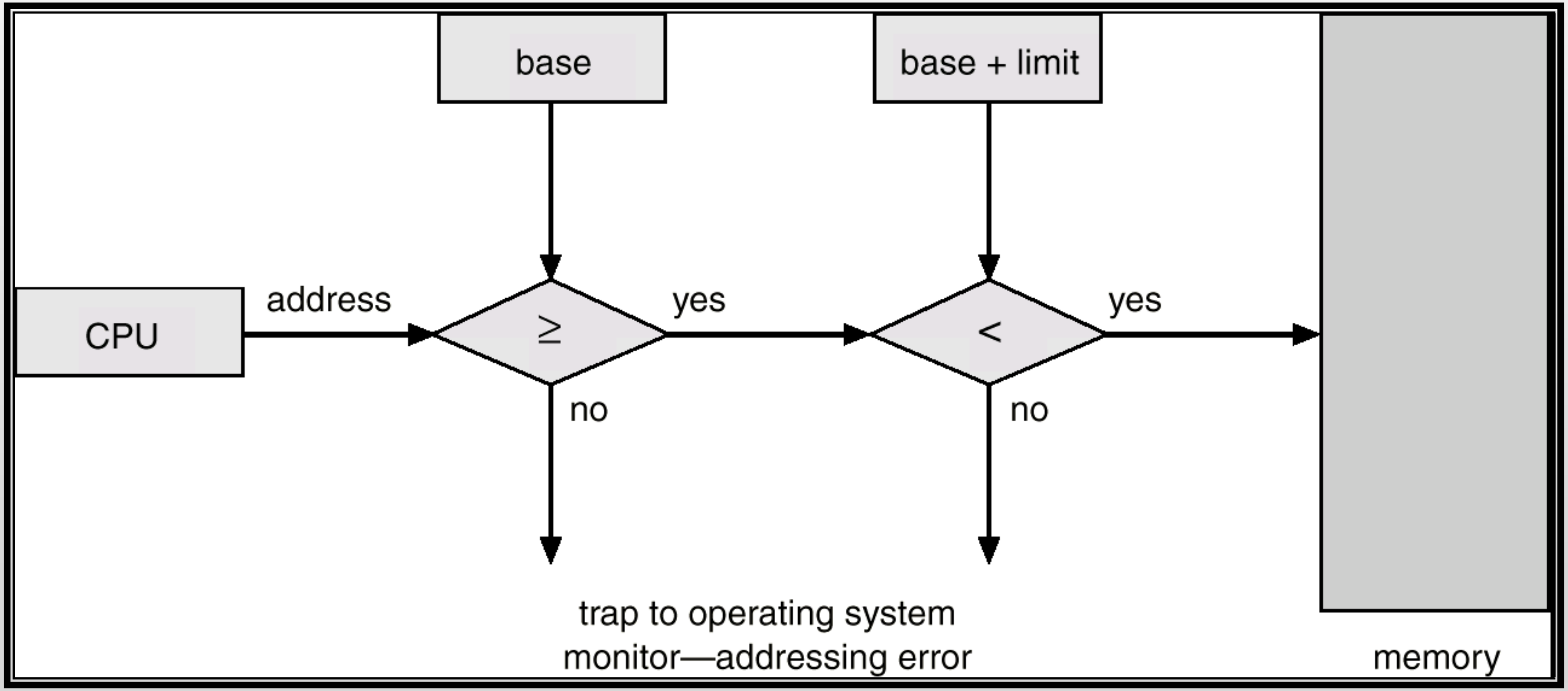
Memory Protection

- Protect
  - Interrupt vector and the interrupt service routines
  - Data access and over-write from other programs
- HW support: two registers for legal address determination:
  - **Base register** – holds the smallest legal physical memory address
  - **Limit register** – contains the size of the range
- Memory outside the defined range is protected

Use of Base and Limit Register



Hardware Address Protection



CPU Protection

- Prevent user program from not returning control
  - getting stuck in an infinite loop
  - not calling system services
- HW support: Timer—interrupts computer after specified period
  - Timer is decremented every clock tick
  - When timer reaches the value 0, an interrupt occurs
- Timer commonly used to implement time sharing
- Load-timer is a privileged instruction

Review Slides (4)

- Dual-mode Operation?
- CPU protection?
- Memory protection?

Reading Material & HW

- Chap 1
- Problem set
    - 1.8: What is the purpose of interrupt? How does an interrupt differ from trap? Can traps be generated intentionally by a user program? If so, for what purpose?
    - 1.10: some computer systems do not provide a privileged mode of operation in hardware. Is it possible to construct a secure operating system for these computer systems? Give arguments both that it is and that it is not possible.
    - Why dual mode operation can protect the system?

Q & A

Thank you for your attention