

COS3043

System Fundamentals

Lecture 1

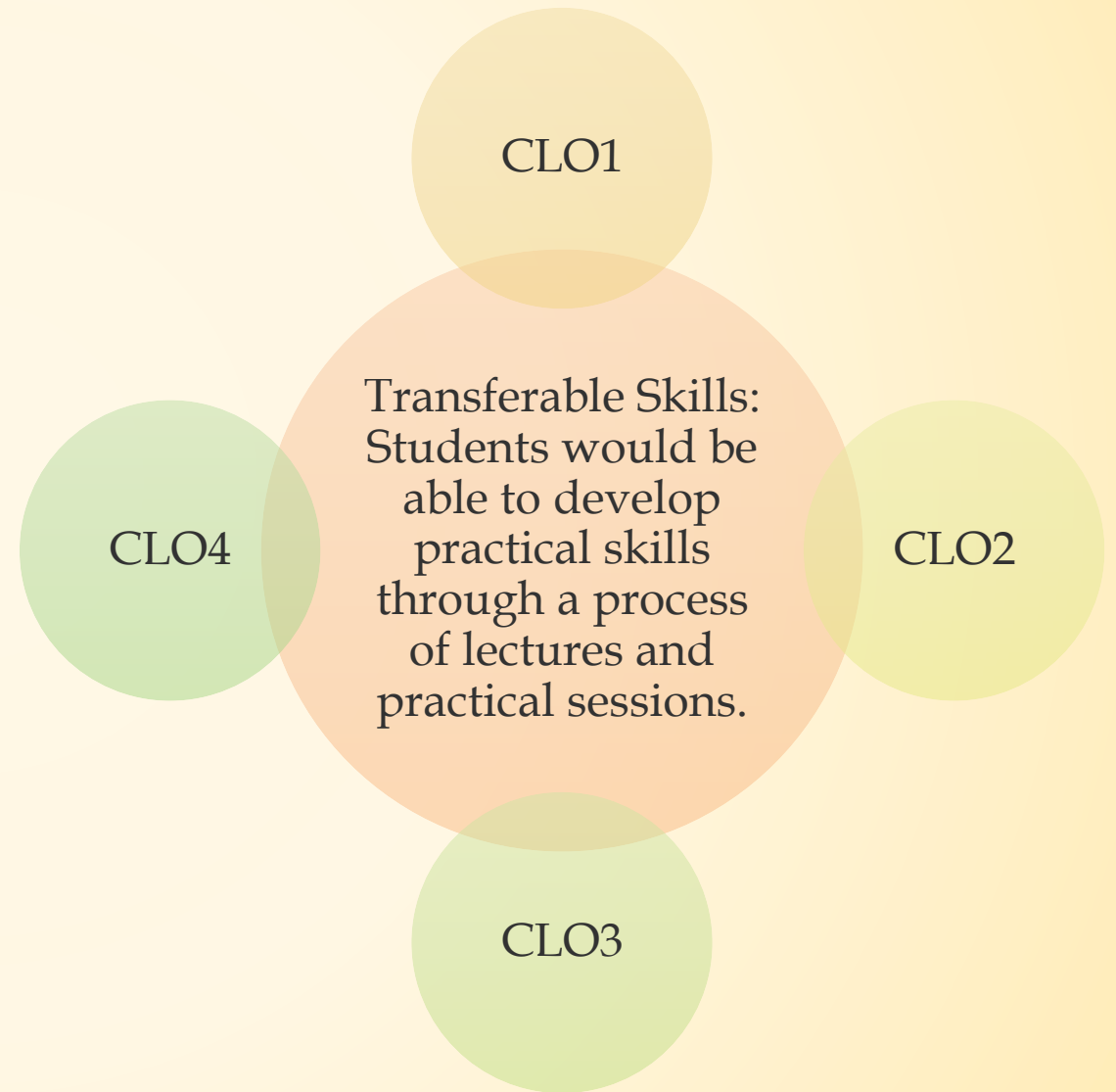
List of Discussion

- Introduction to Course
- Principle of Abstractions
- Hardware Resources
- OS Functionality
- OS Abstraction - Managing the CPU and Memory

Introduction to Course

Course Learning Outcomes

- CLO1: Explain advanced concepts in systems fundamentals theory and implementation.
- CLO2: Evaluate software design issues for advanced computer systems such as multiprocessors or distributed systems.
- CLO3: Discuss current topics in systems fundamentals research by reading and analysis of journal papers.
- CLO4: Implement a correct concurrent program requiring synchronization.



Pre-requisites Knowledge

- COS3024 Operating Systems and Concurrency

Why Learning System Fundamentals?

Motivator 1

Core Technology

A computer scientist must know the basic technology and the intricacies involved in a computer system. Without understanding the underlying concepts, it gets very difficult to improve or contribute. - *No Magic!*

Motivator 2

Resources Management

When you can understand the design, architecture and how operating system manages the process of a computer, it gives better idea to produce efficient coding - better programming quality.

Motivator 3

Problem Solving Models

In OS, there are many concepts proposed to deal with the issues of managing processes. Those can be used as models to solve real problems.

Lecturer Contact

- Foo Lye Heng
- Email address: lyeheng.foo@uow.edu.my
- Part-time Lecturer
- [Face-to-face / Meet Online](#)
 - make an appointment by sending me a Microsoft Teams message / an email

Topics

1.	Abstractions 1.1 Hardware Resources 1.2 OS Functionality 1.3 Managing the CPU and Memory
2.	OS Structure 2.1 SPIN Approach 2.2 Exokernel Approach 2.3 L3/L4 Micro-Kernel Approach
3.	Virtualization 3.1 Intro to Virtualization 3.2 Memory Virtualization 3.3 CPU and Device Virtualization
4.	Parallelism 4.1 Shared Memory Machines 4.2 Synchronization 4.3 Communication 4.4 Scheduling
5.	Distributed Systems 5.1 Definitions 5.2 Lamport Clocks 5.3 Latency Limit

6.	Distributed Object Technology 6.1 Spring Operating System 6.2 Java RMI 6.3 Enterprise Java Beans
7.	Design and Implementation of Distributed Services 7.1 Global Memory System 7.2 Distributed Shared Memory 7.3 Distributed File System
8.	System Recovery 8.1 Lightweight Recoverable Virtual Memory 8.2 Rio Vista 8.3 Quicksilver
9.	Internet Scale Computing 9.1 GiantScale Services 9.2 Content Delivery Networks 9.3 MapReduce
10.	Real-Time and Multimedia 10.1 Persistent Temporal Streams

Topics

1.	Abstractions 1.1 Hardware Resources 1.2 OS Functionality 1.3 Managing the CPU and Memory
2.	OS Structure 2.1 SPIN Approach 2.2 Exokernel Approach 2.3 L3/L4 Micro-Kernel Approach
3.	Virtualization 3.1 Intro to Virtualization 3.2 Memory Virtualization 3.3 CPU and Device Virtualization
4.	Parallelism 4.1 Shared Memory Machines 4.2 Synchronization 4.3 Communication 4.4 Scheduling
5.	Distributed Systems 5.1 Definitions 5.2 Lamport Clocks 5.3 Latency Limit

6.	Distributed Object Technology 6.1 Spring Operating System 6.2 Java RMI 6.3 Enterprise Java Beans
7.	Design and Implementation of Distributed Services 7.1 Global Memory System 7.2 Distributed Shared Memory 7.3 Distributed File System
8.	System Recovery 8.1 Lightweight Recoverable Virtual Memory 8.2 Rio Vista 8.3 Quicksilver
9.	Internet Scale Computing 9.1 GiantScale Services 9.2 Content Delivery Networks 9.3 MapReduce
10.	Real-Time and Multimedia 10.1 Persistent Temporal Streams

Assessments

Structure	Marks (%)	Hand-out	Hand-in
Assignment 1	30	Week 2	Week 6
Assignment 2	30	Week 6	Week 11
Final examination	40	Exam Week	

References

- Textbook:
 - Andrew S.Tanenbaum (2016), *Modern Operating System*
 - Abraham Silberschatz (2013) *Operating System Concepts*, 9th Edition, Wiley

- Additional:
 - Andrew S.Tanenbaum (2006), *Operating System Design and Implementation*
 - William Stallings (2017), *Operating Systems: Internals and Design Principles*, Pearson Education

Topics

1.	Abstractions 1.1 Hardware Resources 1.2 OS Functionality 1.3 Managing the CPU and Memory
2.	OS Structure 2.1 SPIN Approach 2.2 Exokernel Approach 2.3 L3/L4 Micro-Kernel Approach
3.	Virtualization 3.1 Intro to Virtualization 3.2 Memory Virtualization 3.3 CPU and Device Virtualization
4.	Parallelism 4.1 Shared Memory Machines 4.2 Synchronization 4.3 Communication 4.4 Scheduling
5.	Distributed Systems 5.1 Definitions 5.2 Lamport Clocks 5.3 Latency Limit

6.	Distributed Object Technology 6.1 Spring Operating System 6.2 Java RMI 6.3 Enterprise Java Beans
7.	Design and Implementation of Distributed Services 7.1 Global Memory System 7.2 Distributed Shared Memory 7.3 Distributed File System
8.	System Recovery 8.1 Lightweight Recoverable Virtual Memory 8.2 Rio Vista 8.3 Quicksilver
9.	Internet Scale Computing 9.1 GiantScale Services 9.2 Content Delivery Networks 9.3 MapReduce
10.	Real-Time and Multimedia 10.1 Persistent Temporal Streams

Principle of Abstractions

Definition - Abstraction



- It is a technique for arranging complexity of computer systems by establishing a level of simplicity so that the persons who interact with the system can suppress the more complex details below the current level.
- But why we need such a technique in computing world? Because we only need to deal with information that is relevant or required.
- Example:
 - Programmer doesn't need to care how number is represented in hardware level.

The essence of abstractions is preserving information that is relevant in a given context, and forgetting information that is irrelevant in that context.
– John V. Guttag

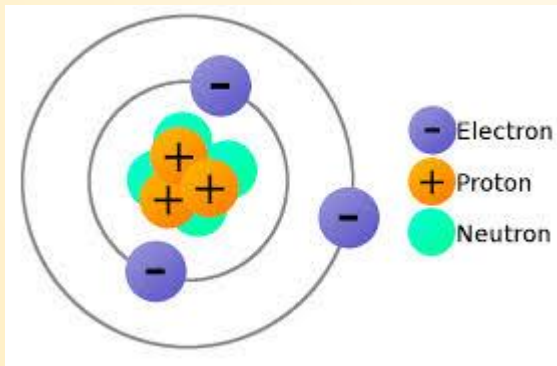
Question?

- Which of the following is NOT abstraction?
 - FLOAT data type
 - Transistor
 - Number of pins coming out from a chip
 - Door Knob
 - Instruction set of a computer processor
 - Location of shoe rack in your house

Abstraction in OS



What kind of
abstractions in
between?



- Application
- System Software (OS, compiler, etc.)

- Instructions Set Architecture
- Machine Organization (Data Path + Control)
- Sequential + Combinational Logic Elements



- Logic Gates
- Transistors

Hardware Resources

Hardware Control

- One of the major roles by OS is to control access of the applications to physical hardware.
- Therefore, the basic understanding of the hardware and how they interact to the OS is imperative to this course.

Question?

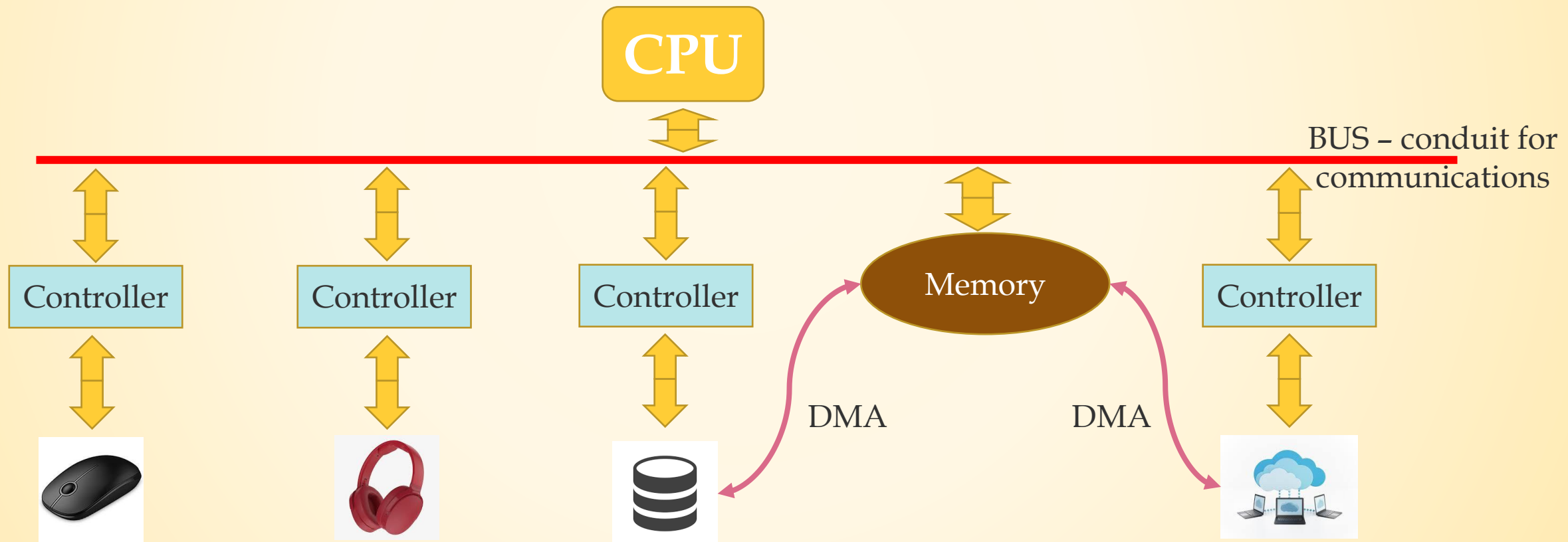


From small to big...

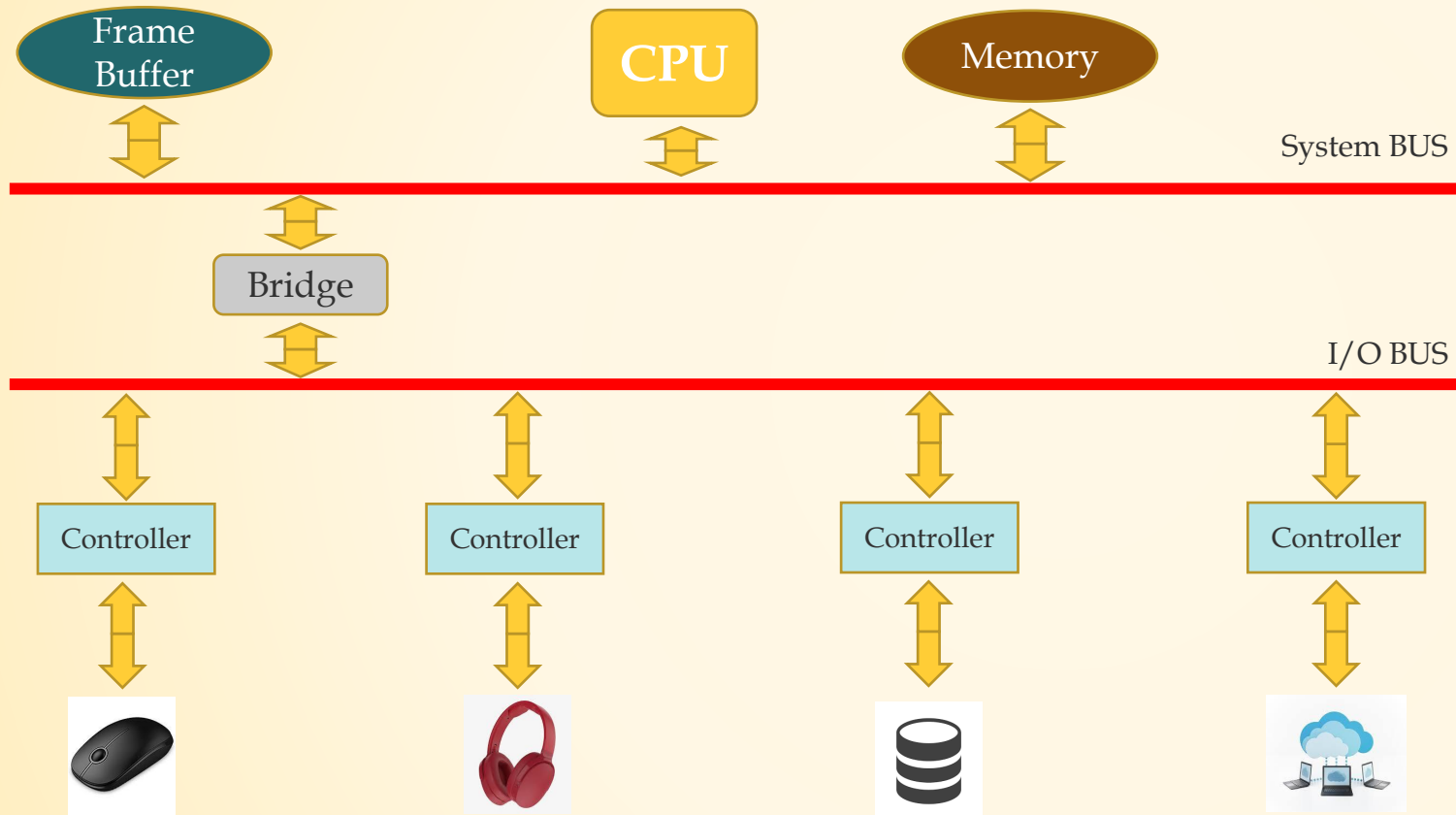
Is internal organization of this hardware continuum vastly different?

Hardware Resources in Computer System

- Basically, there are **same regardless** of scale.
- For instance: **processor, memory, input and output peripherals.**



Organization With I/O BUS



- System BUS > I/O BUS in bandwidth
- Bridge
 - to connect System & I/O BUS
 - to control access of memory for I/O directly or indirectly
- Frame buffer - a high speed device such as graphic display for refreshing screen swiftly.

Summary



- The specifics such as computation power, memory capacity and number & types of I/O devices may vary from one to another computer systems.
- For example, smartphones or tablets may have limited I/O capabilities. But on the other hand, a scientific super computer may connect to thousands of hard disks or CPU.
- However, the organization of a computer system is consistent enough that many of key operating system concepts are applied regardless of size or capacity.

OS Functionality

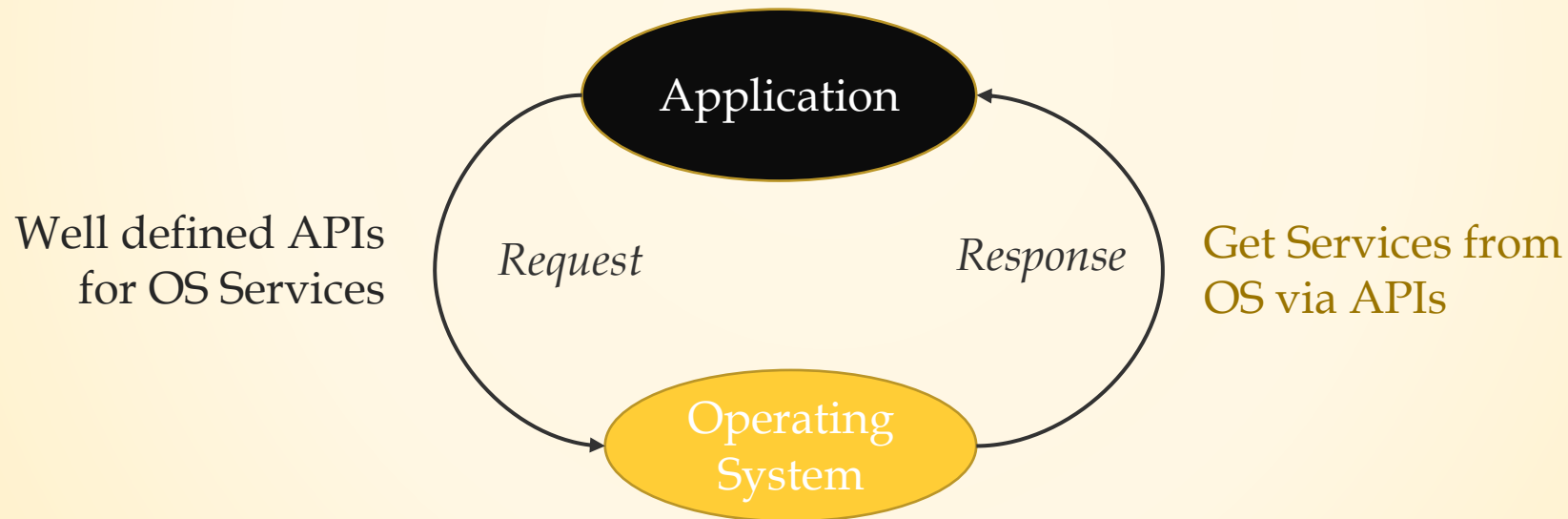
Review Question?

- Select the choices which are functionalities of an OS:
 - OS is a resource manager.
 - OS provides consistent interface to the hardware resources such as CPU, memory, storage etc..
 - OS schedules applications on the CPU.
 - OS stores personal information such as IC number, email address, credit card number etc..



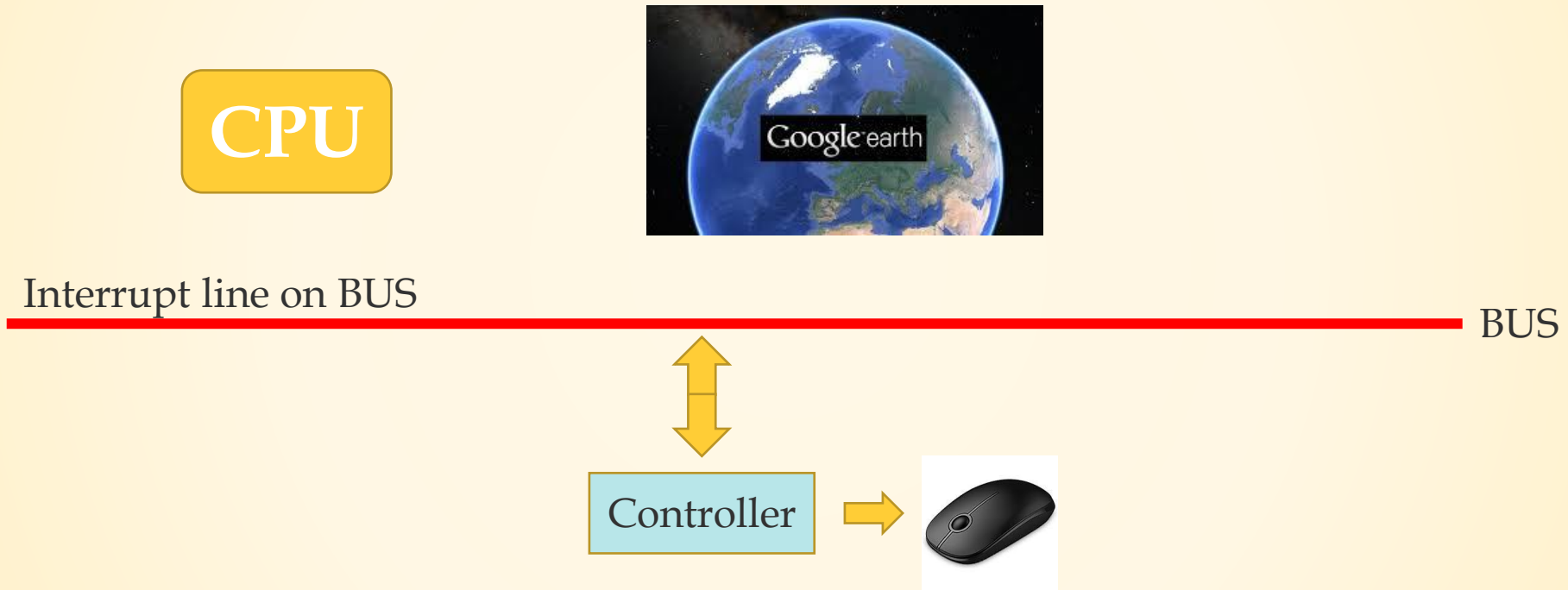
What Is An OS?

- An OS:
 - Manages access to the hardware resources.
 - Arbitrates among competing requests.



Example


- What happens when you click your mouse? 



- ✓ Interruption
- ✓ Answering - OS
- ✓ Passing the call to appropriate program for action

OS Abstraction - Managing the CPU and Memory

Introduction

- Multiple application can run simultaneously and seamlessly in a computer system most of the time.
- All the applications need access to the physical hardware such as CPU and memory with these rules:
 - Share nicely 
 - No hogging the CPU
 - No over-writing each other
- That is why an OS is important in such situation to protect these application from one another and to protect an application while trying to manage the resources in 'fair' manner.

Question?

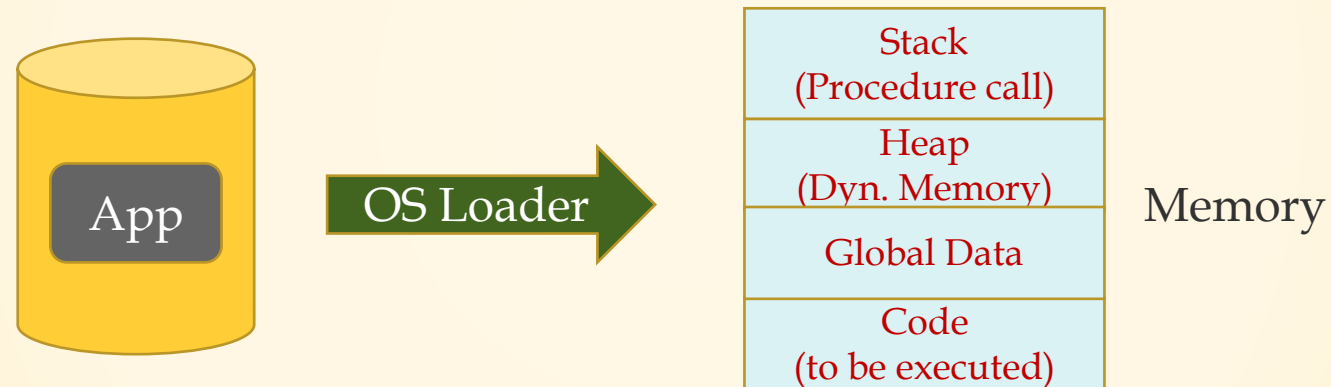
- Computer seemingly runs several programs at the same time: e-mail, music, web browsing.... If it is only one CPU, how is that possible?

Possible Answer:

1. There are **multiple cores** in CPU, so each core runs an application.
1. For **different time unit, different applications are running** on a CPU under the management of an OS.

Catering to Resource Requirements

- An OS knows enough about an application at the time of launching.
- Thus from disk, OS can create a memory footprint for that particular application.



- Application can ask for additional memory at runtime.
- Although OS is 'broker' managing the memory requirement, it won't take away precious resources from app. 🗨️

The Modern OS

- The modern OS is a complex software in the sense that it deals with a lot of different calls which some of them are completely outside of the user control.
- For example, a network message pops in while the scheduled anti virus software is in its full gear to scan for virus.
- A good OS should provide an application with the resources it is asking for such as more memory or right to access to files as quickly and quietly as possible.

Processor Related OS Abstraction

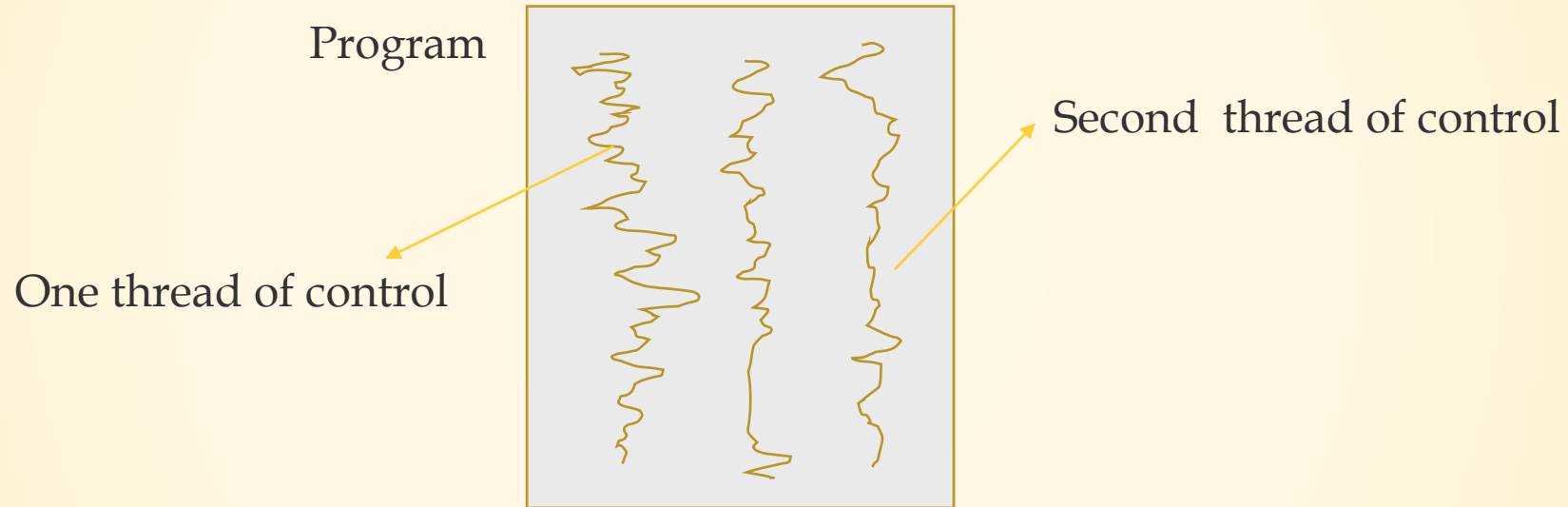
- Program: \Rightarrow Static image loaded into memory
- Process: \Rightarrow A program in execution

Process = Program + Running State (*evolves as the program executes*)



Different Between Process & Thread

- Analogy of reading newspaper:
 - A Program = Newspaper on dining table



Process = Program + State of all the threads executing in the program

- Another example: Web browser
 - Thread to render and display a page
 - Thread to fetch data from web server

Process VS Thread

COMPARISON BASIS	PROCESS	THREAD
Definition	A process is a program under execution i.e an active program.	A thread is a lightweight process that can be managed independently by a scheduler.
Context Switching Time	Processes require more time for context switching as they are heavier.	Threads require less time for context switching as they are lighter than processes.
Memory	Processes are totally independent and don't share memory.	A thread may share some memory with its peer threads.
Communication	Communication between processes requires more time than between threads.	Communication between threads requires less time than between processes.
Blocked	If a process gets blocked, remaining processes can continue execution.	If a user level thread gets blocked, all of its peer threads also get blocked.

Process VS Thread (continue)

COMPARISON BASIS	PROCESS	THREAD
Resources	Processes require more resources than threads.	Threads generally need less resources than processes.
Dependency	Individual processes are independent of each other.	Threads are parts of a process and so are dependent.
Data and Code Sharing	Processes have independent data and code segments.	A thread shares the data segment, code segment, files etc. with its peer threads.
Treatment by OS	All the different processes are treated separately by the operating system.	All user level peer threads are treated as a single task by the operating system.
Time for creation	Processes require more time for creation.	Threads require less time for creation.
Time for termination	Processes require more time for termination.	Threads require less time for termination.

Memory Related OS Abstraction

- How one program (ex: e-mail) is protected from misbehavior of the other application (ex: web browser)?
- This is where memory related OS abstraction comes into play.
 - Address space for each process distinct from one another.
 - Implement the abstraction according to hardware capabilities.