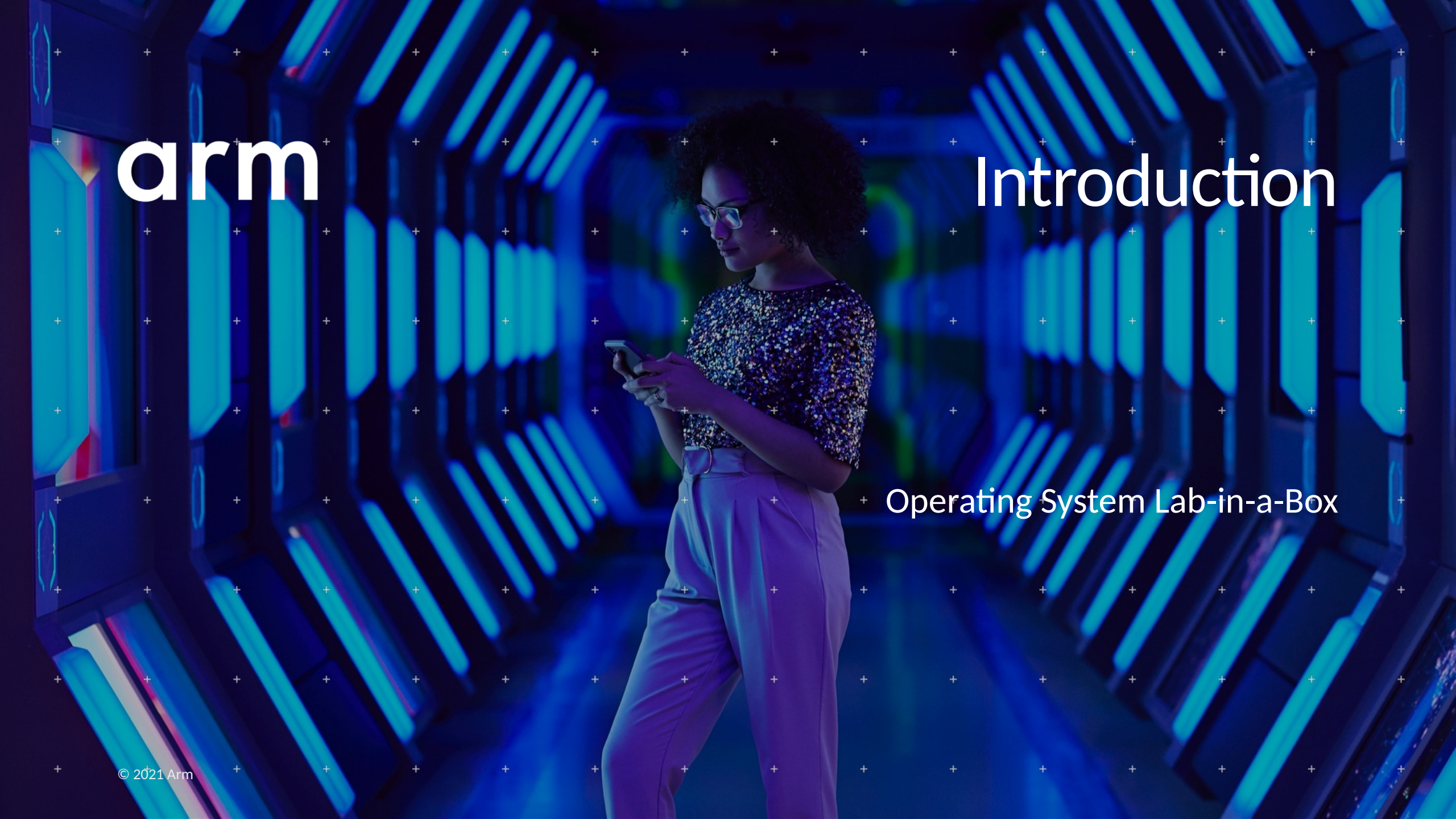
Inclusive Language Commitment

- *Arm is committed to making the language we use inclusive, meaningful, and respectful. Our goal is to remove and replace non-inclusive language from our vocabulary to reflect our values and represent our global ecosystem.*
- *Arm is working actively with our partners, standards bodies, and the wider ecosystem to adopt a consistent approach to the use of inclusive language and to eradicate and replace offensive terms. We recognise that this will take time. This course may contain references to non-inclusive language; it will be updated with newer terms as those terms are agreed and ratified with the wider community. We recognise that some of you will be accustomed to using the previous terms and may not immediately recognise their replacements. Please refer to the following examples:*
  - *When introducing the AMBA AXI Protocols, we will use the term 'Manager' instead of 'Master' and 'Subordinate' instead of 'Slave'.*
- *Contact us at education@arm.com with questions or comments about this course. You can also report non-inclusive and offensive terminology usage in Arm content at terms@arm.com.*



Introduction

Operating System Lab-in-a-Box

Course Overview

- Around 13 lectures
- Covers some of the most important aspects of operating systems
- Particular emphasis on embedded systems and Real Time OS (RTOS)
- Illustrate details using RTOS examples
- Labs and assignments are intended to help you understand both the concepts and implementation of OS components
- Hands on experience in developing multithreaded applications on a RTOS

Brief History

- The Abacus
  - Ancient tools, widely used in various regions
- Logarithms
  - John Napier and Napier’s Bones
- Pascal’s calculator: the Pascaline in 1642
  - Mechanical digital calculator

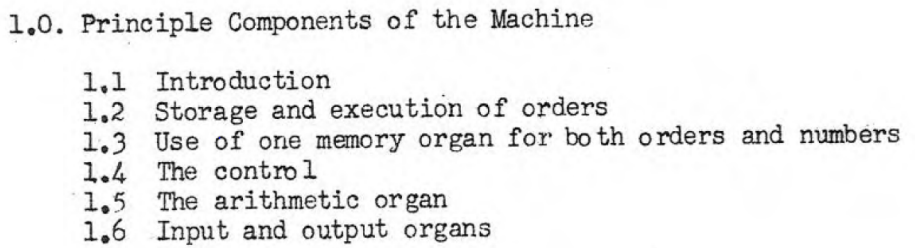
Brief History

- Difference and Analytical Engines
  - Charles Babbage
  - Ada Lovelace
  - Never built
  - Turing completeness machine
  - A precursor for modern computers in some sense
- Punched Card Machinery - 1890
- Vacuum Tubes – 1905
- Relay Calculators – 1935
- Z3 – 1941
- Colossus -1943

Brief History

- Turing Machine
  - On computable numbers, with an application to the Entscheidungsproblem – 1936
  - Intelligent machinery – 1948
  - Model design of CPU

- Preliminary Discussion of the Logical Design of an Electronic Computing Instrument - 1947
  - Arthur Burks, Herman Goldstine, John von Neumann
  - Stored-program computer



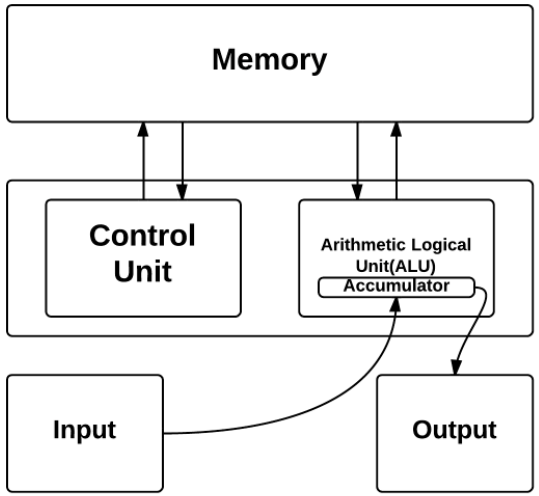
1.0. Principle Components of the Machine

1.1 Introduction
1.2 Storage and execution of orders
1.3 Use of one memory organ for both orders and numbers
1.4 The control
1.5 The arithmetic organ
1.6 Input and output organs

Looks familiar?

Brief History

- Von Neumann Architecture
  - First Draft of a Report on the EDVAC – 1945
  - A standard for a modern computer
  - Both data and program are stored in the memory



- ENIAC - 1946
  - Electronic Numerical Integrator And Computer
  - Took weeks to program the plug board
  - Vacuum tubes, 150 kW
  - Decimal

Suggested Reading

- First Draft of a Report on the EDVAC
- Preliminary Discussion of the Logical Design of an Electronic Computing Instrument
  - For the fanatics only

How did the OS come into existence

- No OS
  - Manually transferred programs by using switches
  - Simple interface like blinking lights on a panel
  - Sign-up sheets as a scheduling tool
  - Single user had to set up everything
- Then an operator
  - Programs written in punched card
  - Operator will handle the task submitted by users and return the result to them
  - Standard card libraries
- Early OS: batch system
  - Monitor, permanently resident in the memory
  - Handle a batch of jobs without user interaction
  - Problematic

How did the OS come into existence

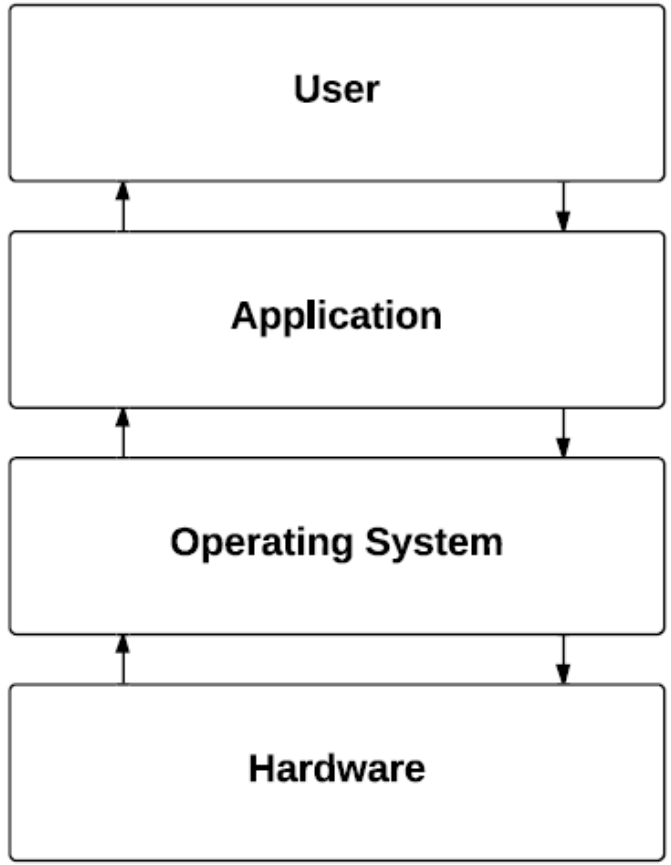
- Multiprogramming
  - Multiprogramming shortens the time that the CPU is blocked by I/O, thus, better utilisation
  - OSs have to be more sophisticated in terms of handling memory for various tasks and schedule the next task once the CPU is blocked by I/O
  - The very essentials of OS
  - Burroughs MCP – 1963
  - IBM's System/360

- Time sharing system
  - Users would like to interact with the computer as well
  - Interactive terminal access: time slices for users

- OS and Personal Computer

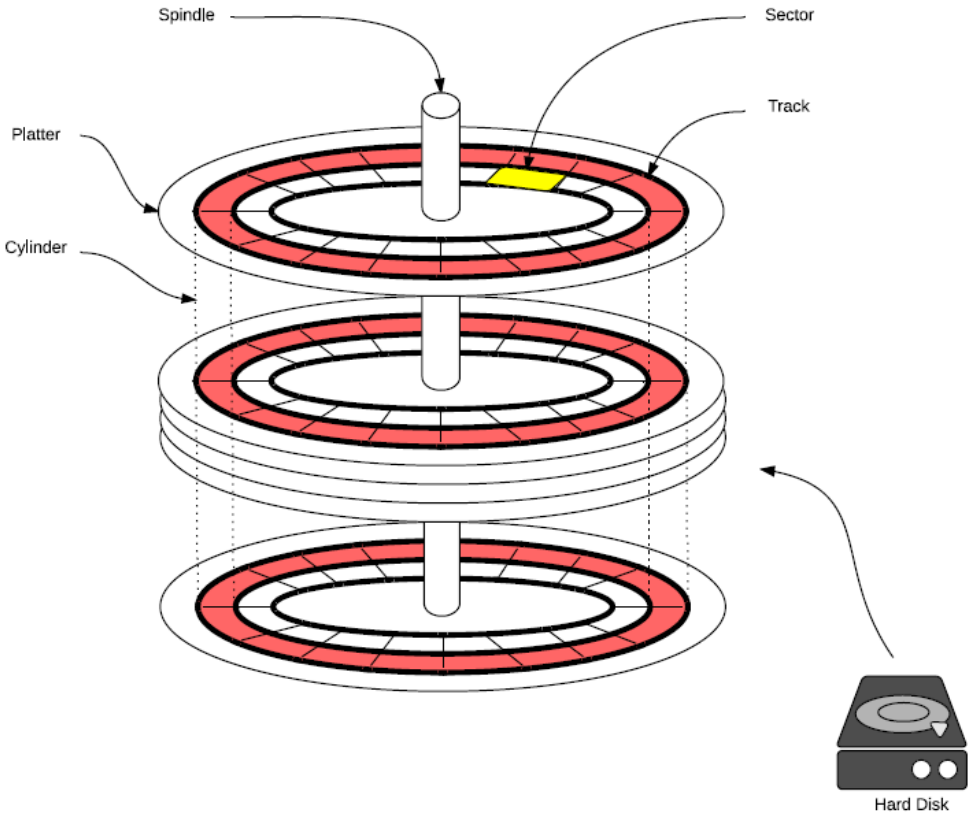
OS Does the Plumbing

- OS provides an abstraction of the messy hardware
  - Hardware is detailed and specific
  - Manipulating hardware requires not only programming knowledge, but also understanding of the hardware
  - User/Programmer does not have to care about the “pipes”, simply turn on and off the “tap”
  - More productive



An Example: Writing to Disk

- Data from memory to the device buffer
  - Reading/writing from/to a disk is usually slower than reading/writing from/to memory
  - *Read(DataMemAddr, Size, DeviceId);*
- Move the read/write head to the right area, identify the platter, track
  - How does a disk work?
  - *Locate(DeviceId, PlatterId, TrackId);*
- Finally write to the track cluster
  - *Write(DeviceId, Cluster);*



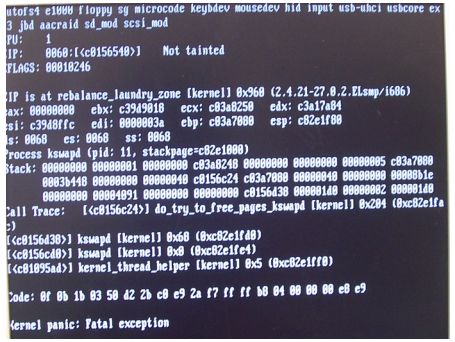
An Example: Writing to Disk

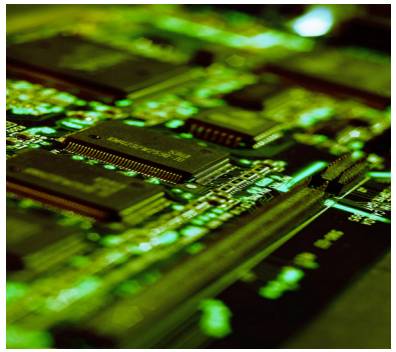
- OSs offer a simplified *Write* function that encapsulates the aforementioned commands
- Also, a logical address instead of the physical address is provided:
  - *Write(DataMemAddr, Size, DeviceId, LogAddr);*

- An even more abstracted function is common
  - Treat memory and the disk as the same file storage
  - A unified file identification *FieId*
  - And then use a library such as “C stdio”
  - Write an integer variable, *datum*, onto the disk at an implicit offset from the beginning of the file
  - *fprintf(FileId, “%d”, datum);*
  - “Everything is a file” philosophy (Unix)

More examples

- User's view: copy, paste, send, save…
- Application developer's view: malloc(), fork(), open()…
- OS programmer's view: read-disk, start-printer, track-mouse…



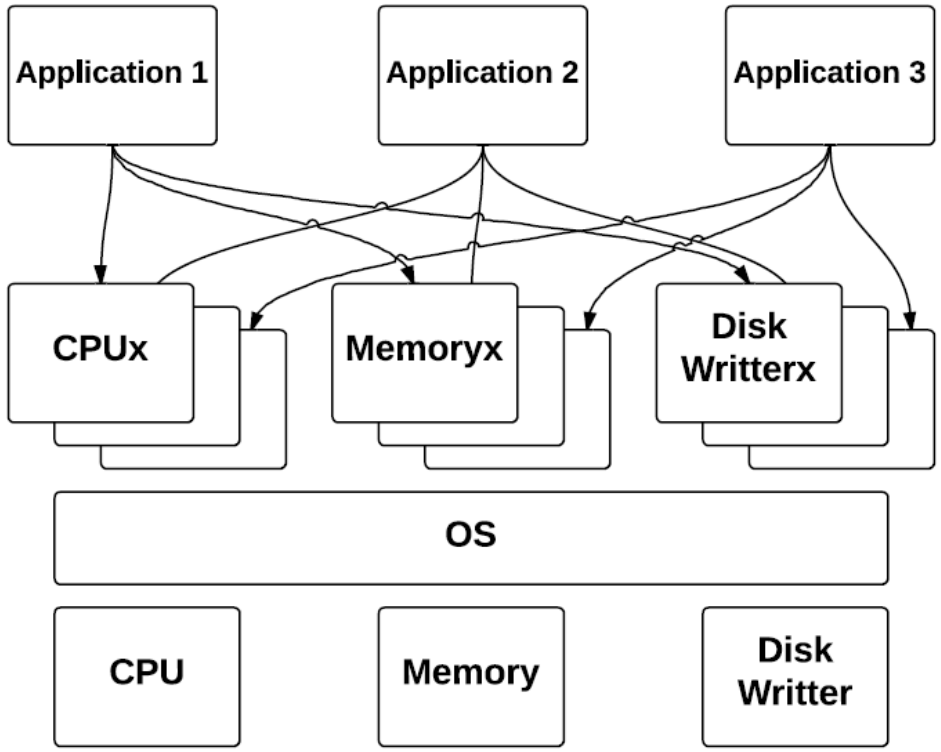


User applications – service request – OS – hardware instructions – hardware

- "All problems in computer science can be solved by another level of abstraction"
- OS provides an abstract programming environment in which specific applications can be further developed to leverage the hardware
- The lowest level of abstraction – an abstraction for hardware

OS Resource Management

- OS also plays the role of resource manager
  - Process management
  - Memory management
- Always limited hardware resources
- How to leverage and optimise the use of hardware
- Time-multiplexing and space-multiplexing
- Control the execution of user applications and I/O devices to avoid errors and misuses



A Dynamic Topic

- OS is a traditional computer science topic, yet very dynamic with new challenges:
  - Post-PC devices and their OS: iOS vs Android
  - Concurrency and parallelism
  - Distributed computing and cloud computing
  - Real time operating system, embedded systems
  - Security and reliability
  - Scalability: How to build the next generation of OS? (90's: below 10 million SLOC, Windows XP 45 million SLOC)
  - OS for bio-computers, quantum computers? (Abstraction and resource management needed)

Next

- OS overview
  - Basic components
  - OS structures
  - Types of OS's
- RTOS