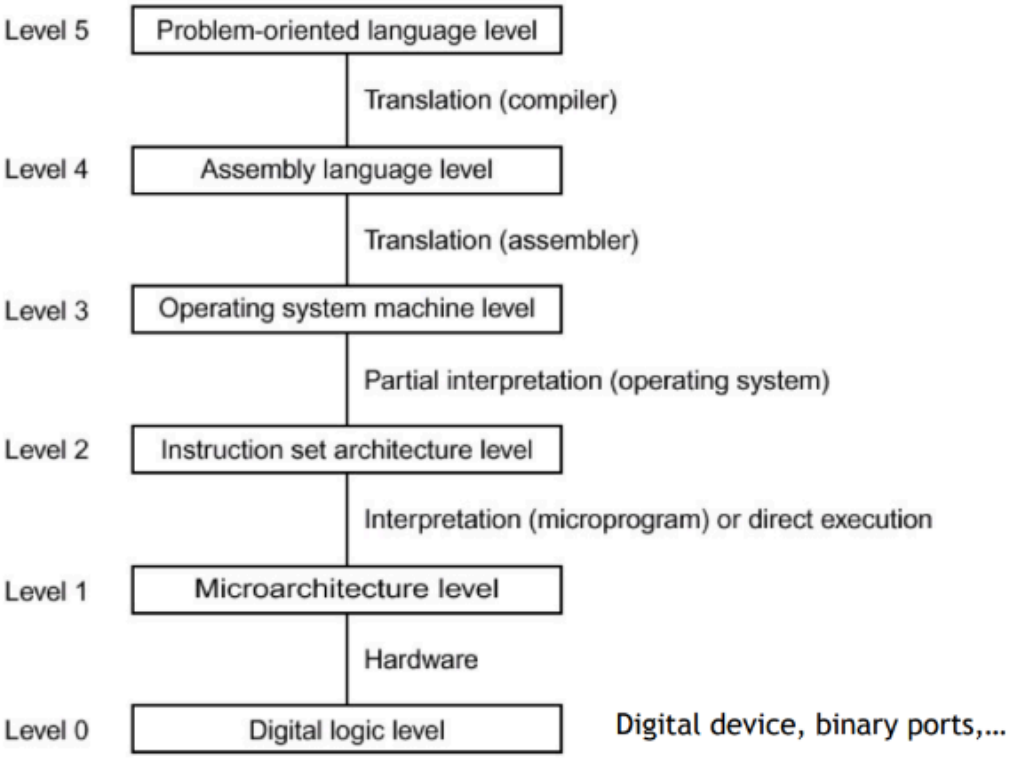
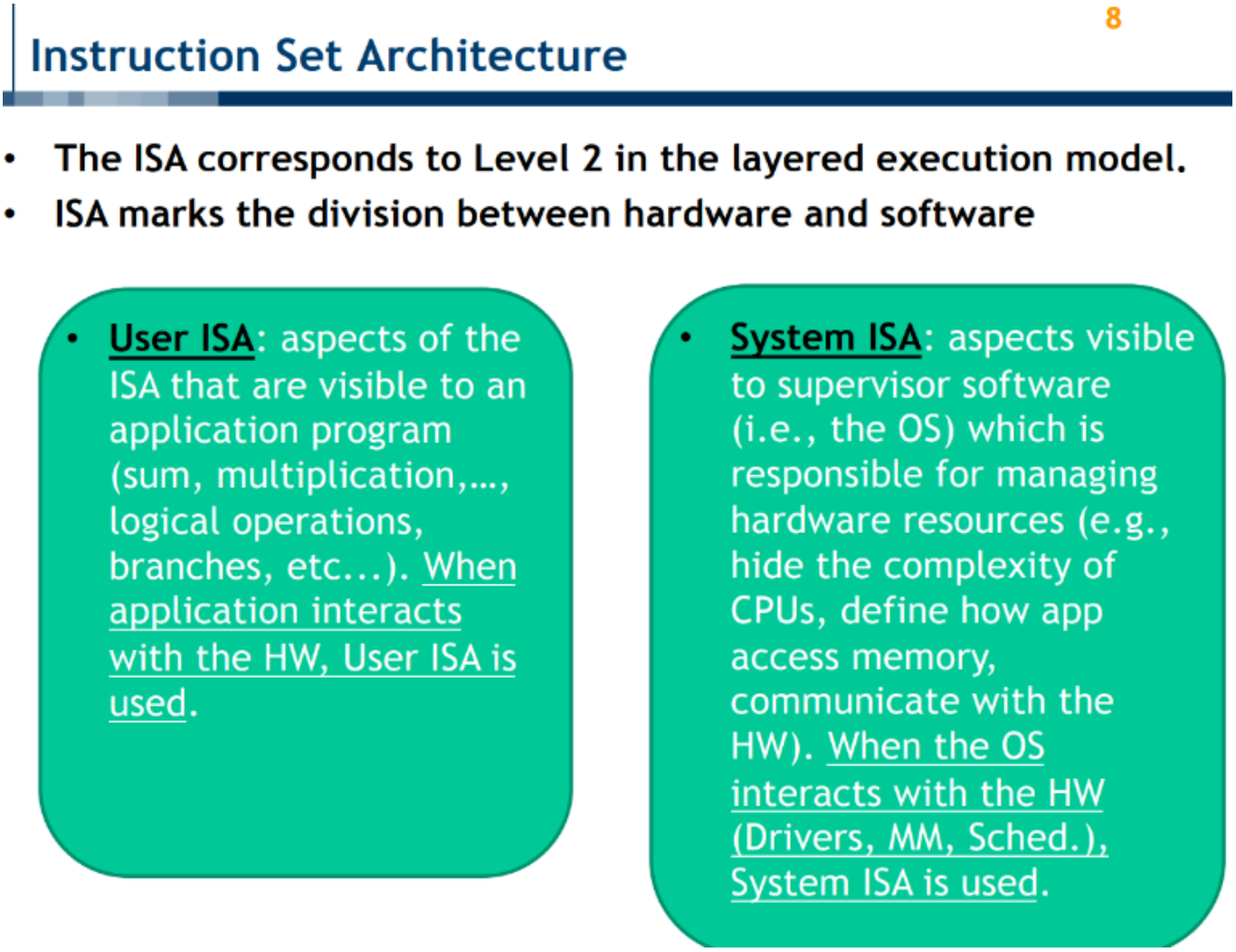
Virtualization

A Machine is an execution environment capable of running a program.
Physical machines have some HW that runs the program. Instead the virtual machines have the OS to create new instructions to access devices/HW

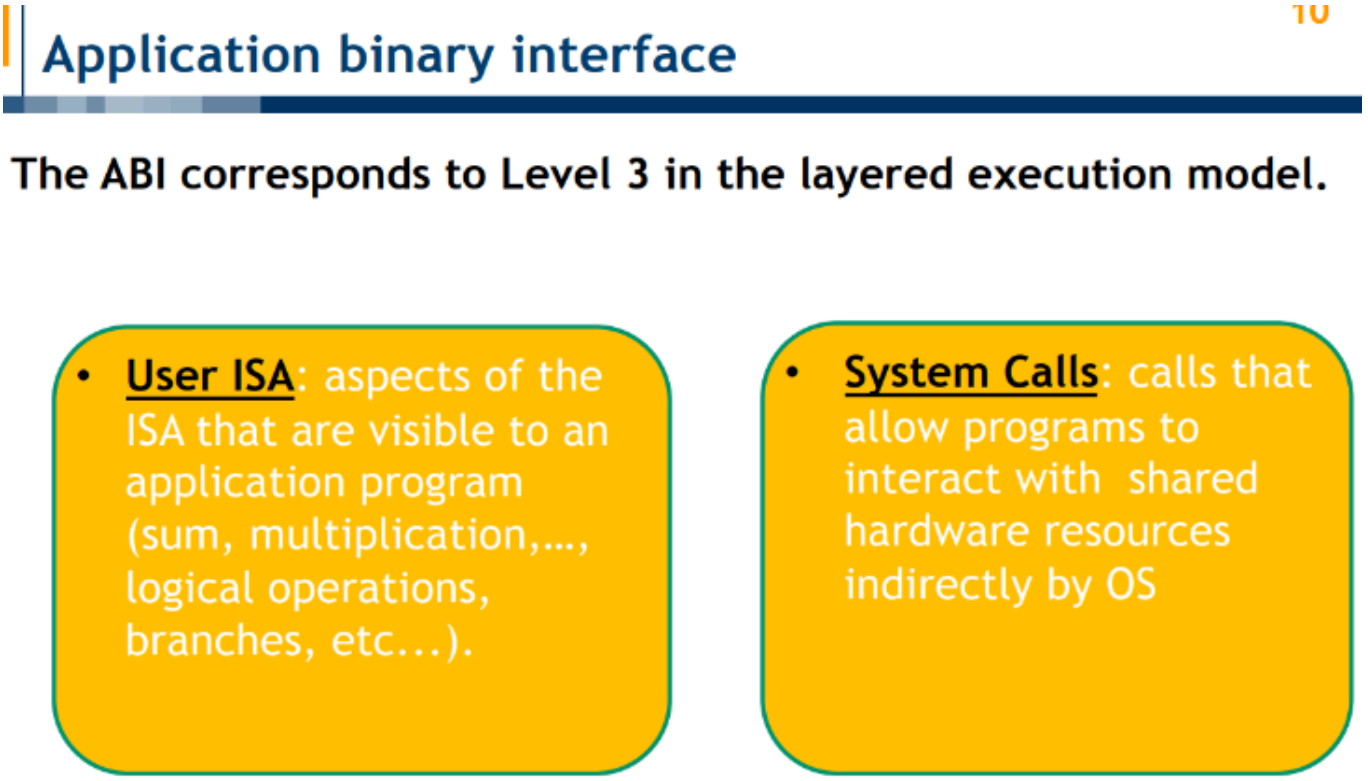
Machine levels

In computer architecture, the set of instructions that a program can use might be structured at different levels.





There are some instructions that only the OS can do(System ISA) and resolve and not the common applications and they are used to converge on situation relative to the memory management, CPU communications...
Level 3 also use applications, it is an extension of CPU using and to perform some task. How can a program send message using only User ISA? They can send the message with common instructions and then exploit the OS to deliver the message/perform task that at low level requires to access System instructions.

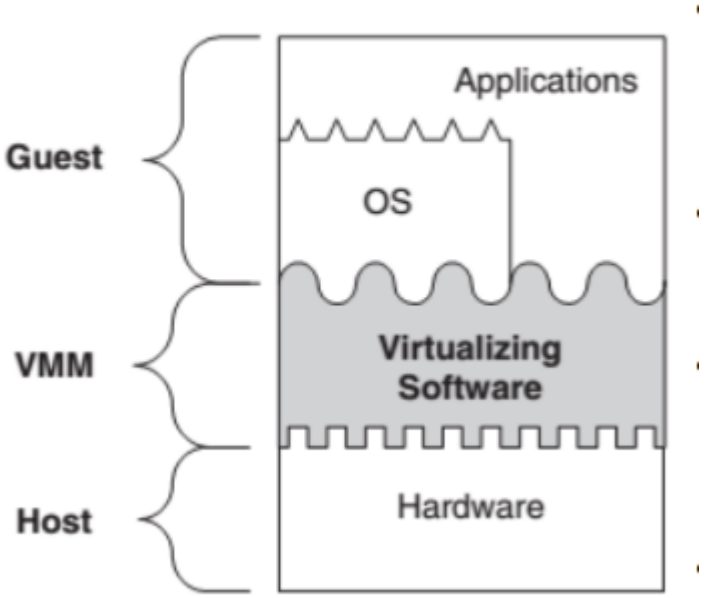


One machine level can only run instructions that were meant for it
A Virtual Machine (VM) is a logical abstraction able to provide a virtualized execution environment. More specifically, a VM:

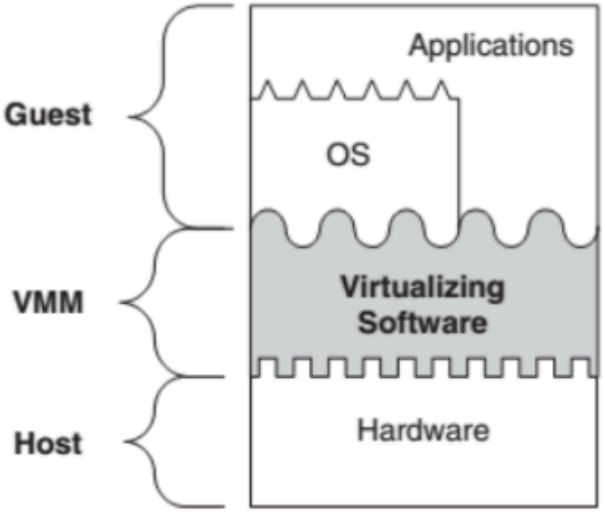
- (provides) Identical Software behavior
- (consists in a) Combination of physical machine and virtualizing software
- (may appear as) Different resources than physical machine
- (may result in) Different level of performance
  Its tasks are:
- To map virtual resources or states to corresponding physical ones
- To use physical machine instructions/calls to execute the virtual ones.
  Two types of Virtual Machines:
- System VMs
- Process VMs
  The software that virtualize the environment tries to map the virtual resources on the physical resources.
  The VM support the Level 0-2 of the architecture.

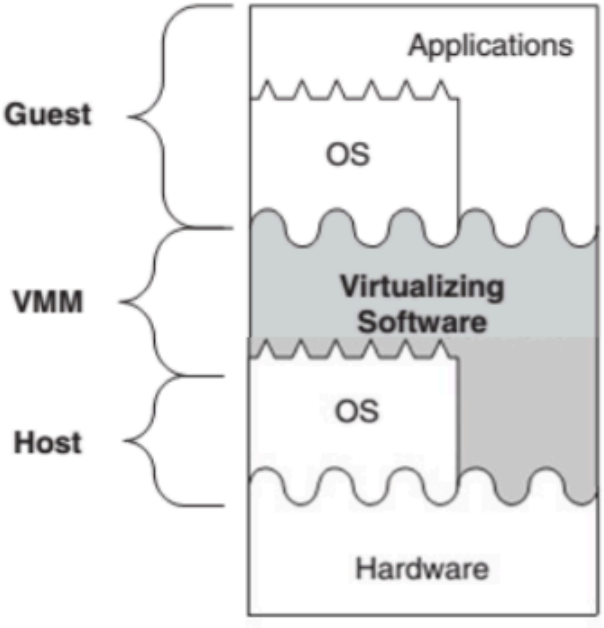
System VMs

System Virtual Machine

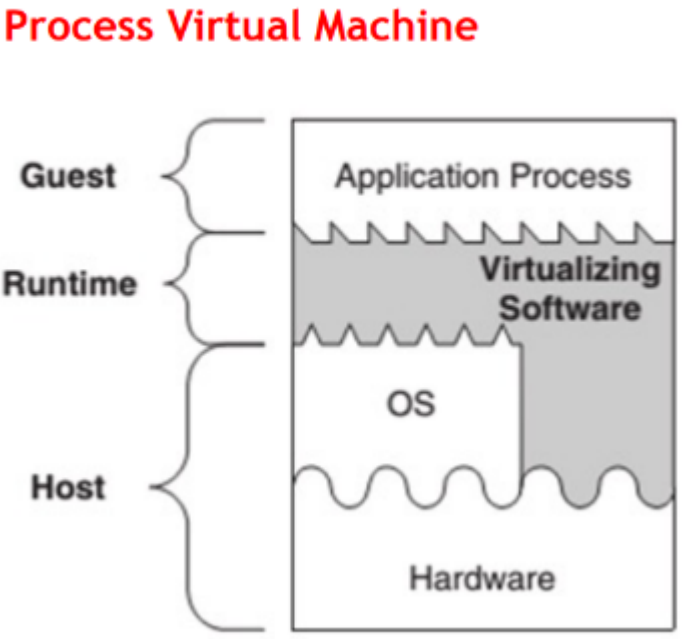


Provide a complete system environment that can support an operating system (potentially with many user processes)
It provides operating system running in it access to underlying hardware resources (networking, I/O, a GUI).
The VM supports the operating system as long as the system environment is alive.
Virtualizing software placed between hardware and software (emulates the ISA interface seen by software)
The virtualization software is called VMM (Virtual Machine Monitor)
The VMM can provide its functionality either working directly on the hardware, or running on another OS. The VMM will exploit the opportunities given by the OS.

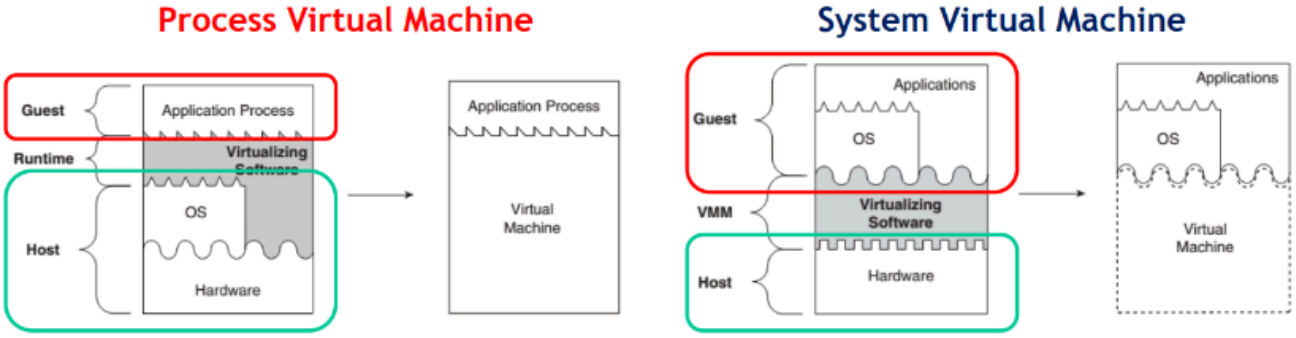




Process VMs



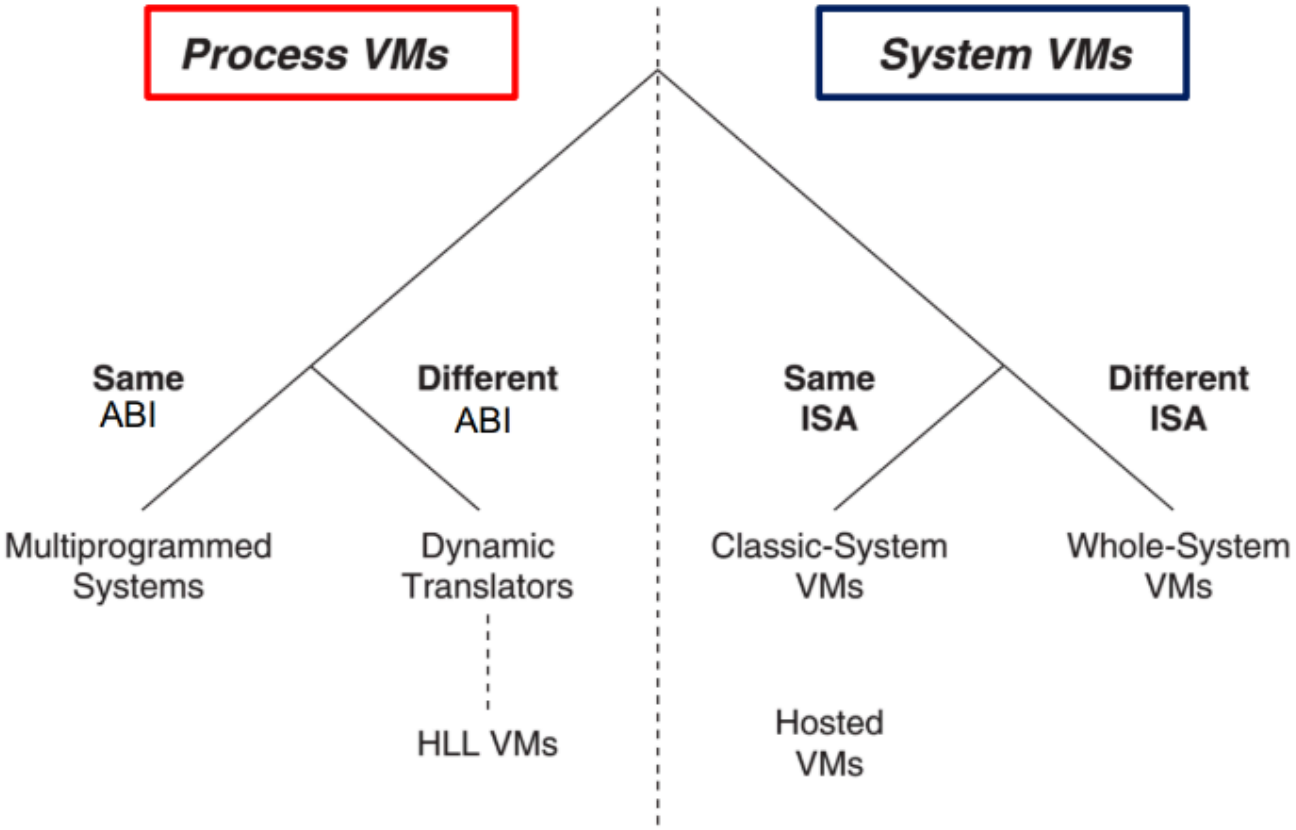
Able to support an individual process
The virtualizing software is placed at the ABI interface, on top of the OS/hardware combination.
The virtualizing software emulates both user-level instructions and operating system calls.
The virtualization software is usually called Runtime Software.
Since there isn't an OS in the system so the HW will relay on a OS underneath



Host: the underlying platform supporting the environment/system

Guest: the software that runs in the VM environment as the guest.

Different types of Virtualization



- Same ABI/ASI: Host = Guest
- DIfferent ABI/ASI: Host != Guest
  Process VMs are running at Level 3, System VMs are running at level 2

Multiprogrammed systems

Same ABI / OS(Things running at Guest level are equal to the ones running at system level):

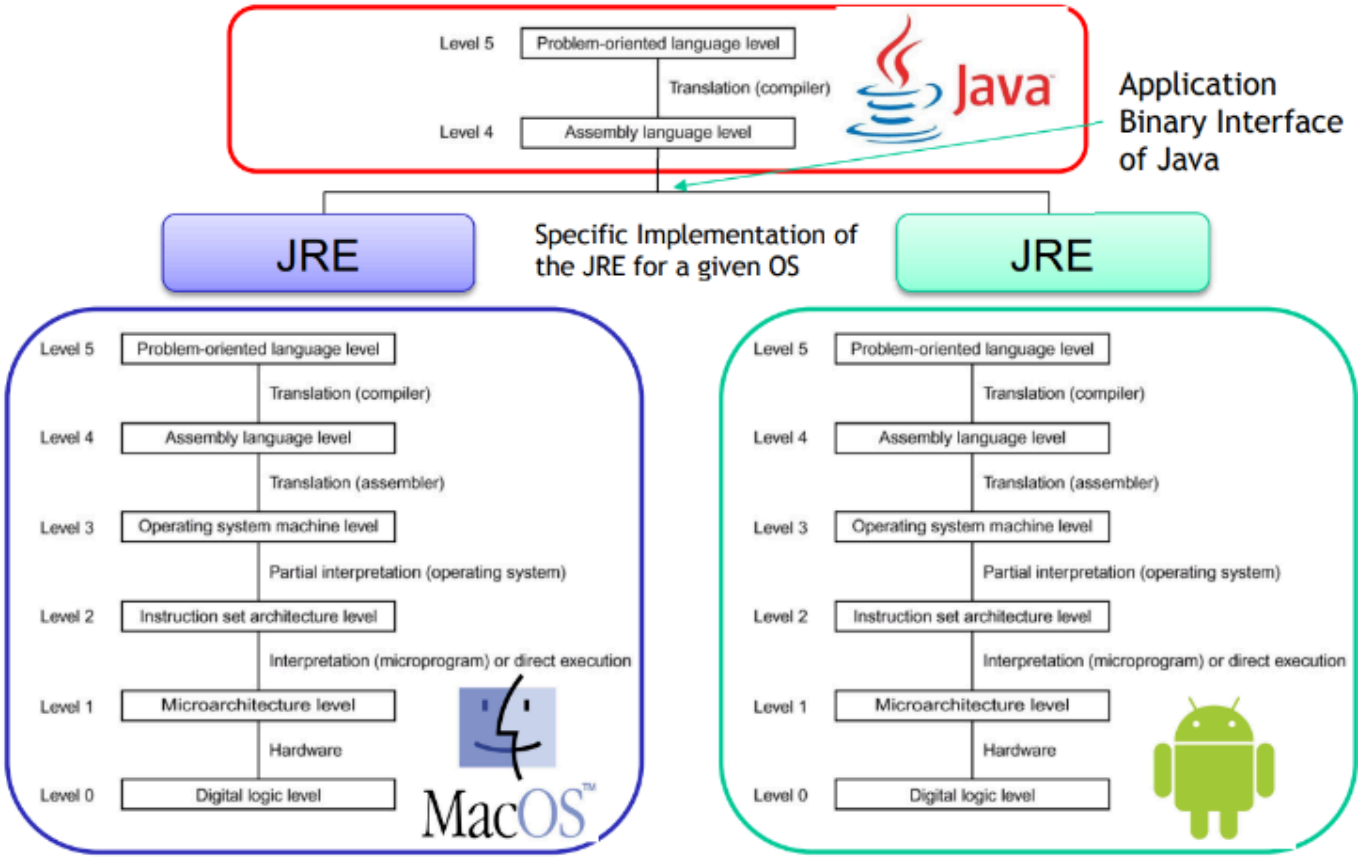
- VM formed by OS call interface + user ISA
- Common approach of all modern OS for multi user support
- Task/Process Manager
- Each user process is given:
  - the illusion of having a complete machine to itself.
  - its own address space and is given access to a file structure
- OS timeshares and manages HW resources to permit this
- Arguable is a real VM
  Emulation refers to those software technologies developed to allow an application (or OS) to run in an environment different from that originally intended. It is required when the VMs have a different ISA / ABI from the architecture where they are running on.
  An emulator reads all the bytes included in the memory of system it is going to reproduce.
  Interpretation:
- Interpreter program fetches, decodes, emulate the execution of individual source instruction;

- Can be a slow process;
- E.g., the Space Invaders

High-Level Language VM

Goal: to have an isolated execution environment for each application (or for different instances of the same application)
VM Task:

- Translates application byte code to OS-specific executable.
- Minimize HW/OS-specific feature for platform independence
  Applications run normally but
- Sandboxed
- Can “migrate”
- Are not conflicting one another
- Are not “installed” in a strict sense
  Example: Java Virtual Machine. Java can work on every architecture for which an interpreter, called the Java Runtime Environment (JRE), exists

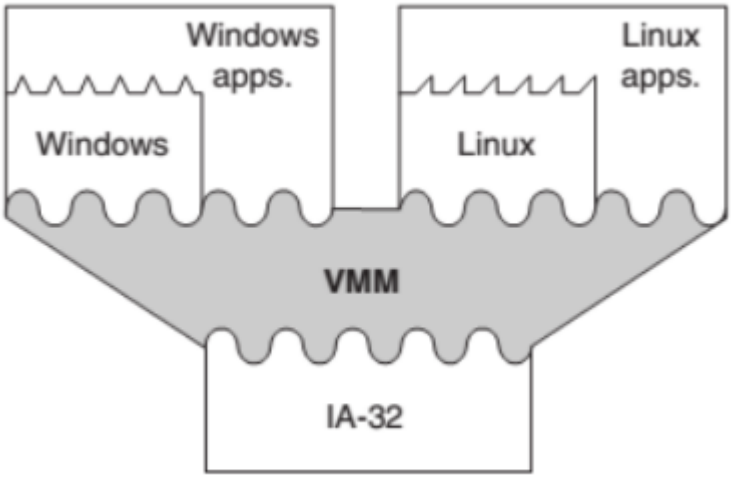


System VM

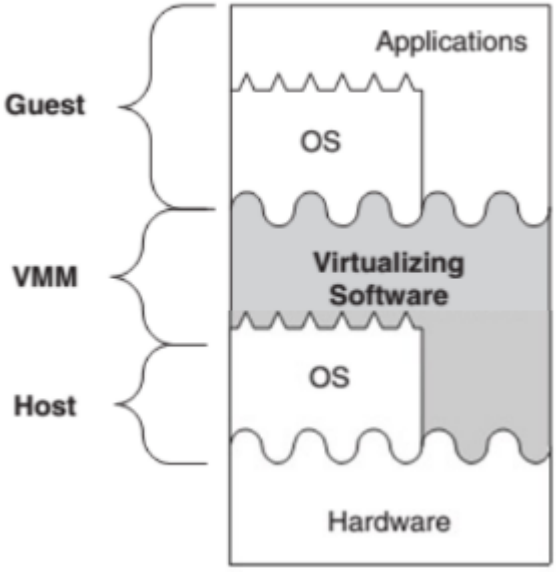
System VM for same ISA:

- The VMM is on bare hardware, and virtual machines fit on top
- The VMM can intercept guest OS’s interaction with hardware resources

- The most efficient VM architecture(HW executes the same instructions of VMs)
- Two different OSs on the same on the same HW
  Exploit the fact that the Virtual environment has some characteristics that also the Host OS has so you can run the VM on the same HW partitioning the resources and having two OS running at the same time.

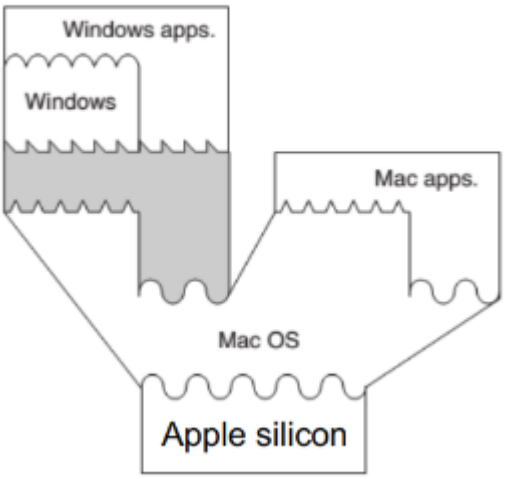


Hosted VM: virtualizing software is on top of an existing host operating system.



Whole-system VMs:

- Virtualize all software: ISAs are different so both application and OS code require emulation, e.g., via binary translation. (no native execution possible)
- Usually implement the VMM and guest software on top of a conventional host OS running on the hardware.
- the VM software must emulate the entire hardware environment and all the guest ISA operations to equivalent OS call to the Host.
- Example: MS Word and Windows running on a Apple Silicon (Arm base M1 and M2, not x86 family

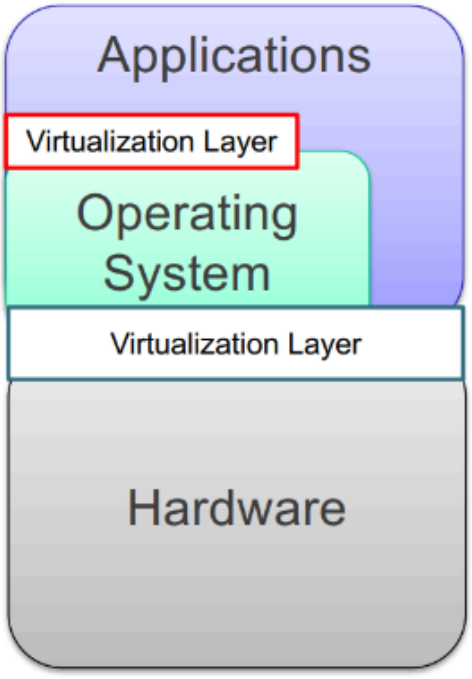


How is Virtualization Implemented?

Given a typical layered architecture of a system...

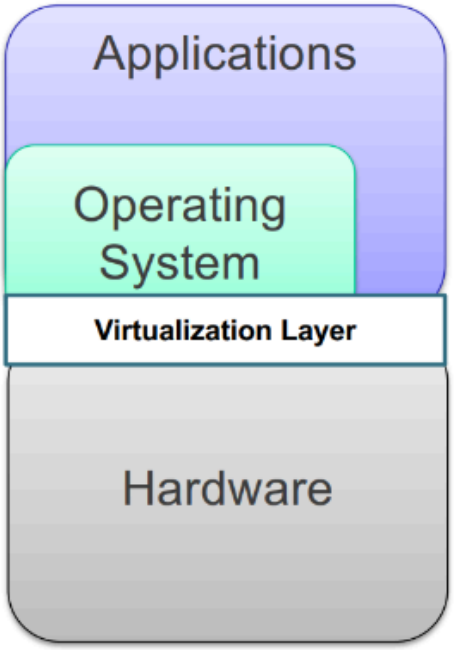
... by adding layers between execution stack layers.

Depending on where the new layer is placed, we obtain different types of Virtualization.



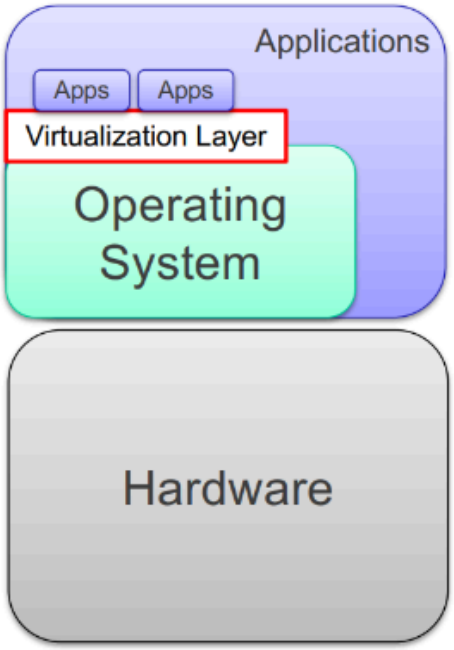
Hardware-level virtualization:

- Virtualization layer is placed between hardware and OS.
- The interface seen by OS and application might be different from the physical one



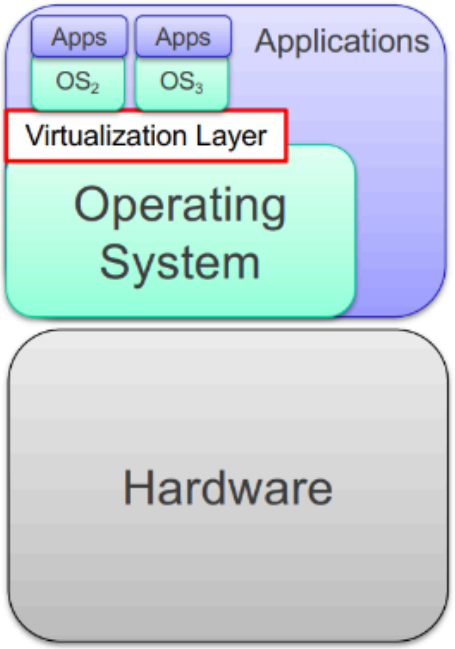
Application-level virtualization:

- A virtualization layer is placed between the OS and some applications
  - E.g.: JVM (Java Virtual Machine)
- Provides the same interface to the applications.
- Applications run in their environment, independently from OS



System-level virtualization:

- The virtualization layer provides the interface of a physical machine to a secondary OS and a set of application running in it, allowing them to run on top of an existing OS.
- Placed between the system's OS and other OS
  - E.g.: VMware Wks/Player, VirtualBox
- Enable several OSs to run on a single HW



We want virtualization to have application running on non original environment having the same performance. This is at the base of cloud computing.
VMs enjoy the following important properties:

- Partitioning: divide the resources of my computer on different OS/applications. Partitioning of resources between the different VM
- Isolation: Fault tolerance and security (at the hardware level). Advanced resource control to guarantee performance (managed by the hypervisor).
- Encapsulation: The entire state of a VM can be saved in a file. A VM is a file, it can be copied and moved as a file, simplifying replication and migration.
- HW-independence: All VMs sees exactly the same virtual hardware, regardless of the host where they are running on. This simplify provisioning and migration of VMs.

Virtual Machine Managers(System VMs – Same ISA)

Virtual Machine Manager
An application that:

- manages the virtual machines
- mediates access to the hardware resources on the physical host system
- intercepts and handles any privileged or protected instructions issued by the virtual machines
- This type of virtualization typically runs virtual machines whose operating system, libraries, and utilities have been compiled for the same type of processor and instruction set as the physical machine on which the virtual systems are running.
- Crucial to support Cloud Computing
  Three terms are used to identify the same thing:
  • Virtual Machine Manager

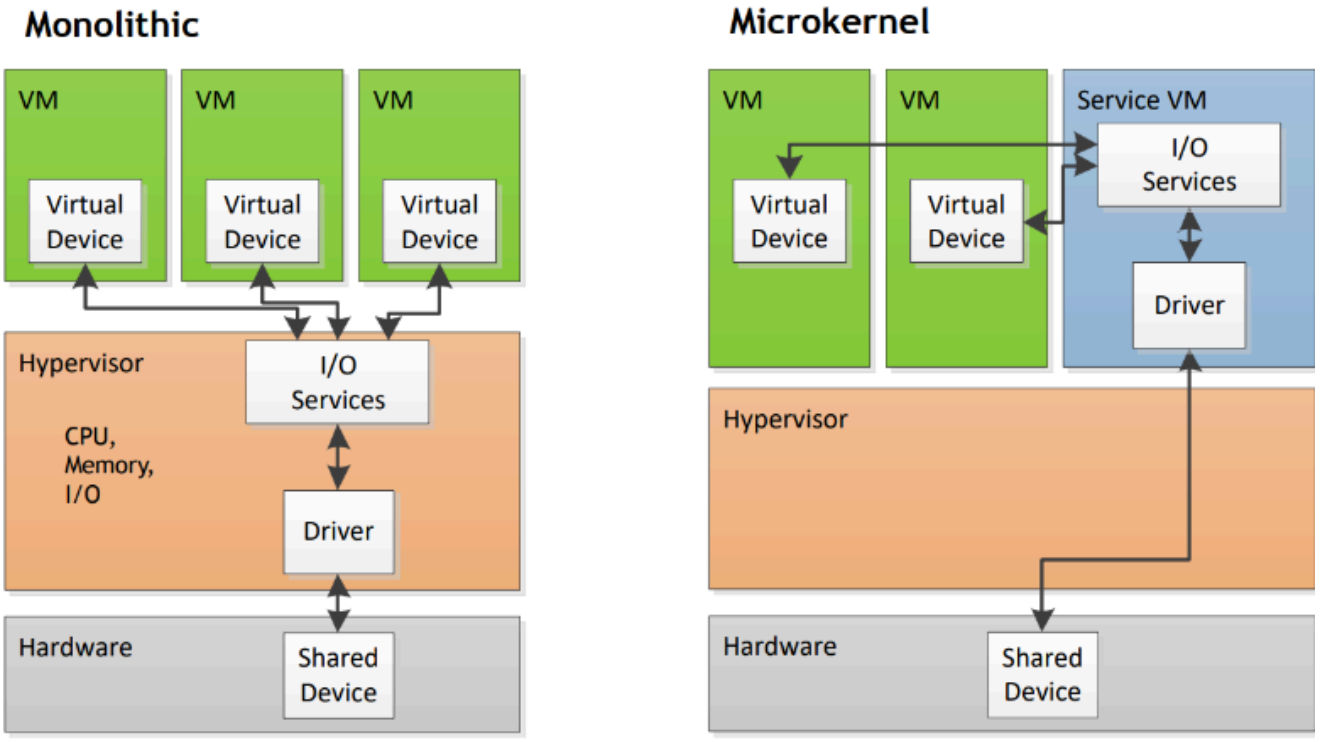
• Virtual Machine Monitor
• Hypervisor
Some authors gives slightly different meanings to the three:

- Virtual Machine Monitor: The virtualization software
- Hypervisor:A virtualization software that runs directly on the hardware
- Virtual Machine Manager:A VMM or Hypervisor that is also used to create, configure and maintain virtualized resources.It provides a user-friendly interface to the underlying virtualization software

Hypervisor types

There are two types depending on where they are running:

- Type 1: they are running directly on the Host HW. The only OS is the one in the Guest VM



The Monolithic have better performance, better performance isolation but can only run on HW for which the hypervisor has driver. The Microkernel has a smaller hypervisor, leverages driver ecosystem of an existing OS and can use 3^{rd} party driver, the VM system can use all the HW in the system

- Type 2: they run inside the Host OS. It can use all the feature of the Host OS at any level. It is the most flexible but it has the most problems has it runs two OS on the same machine.