# An Introduction to Virtualization Technology

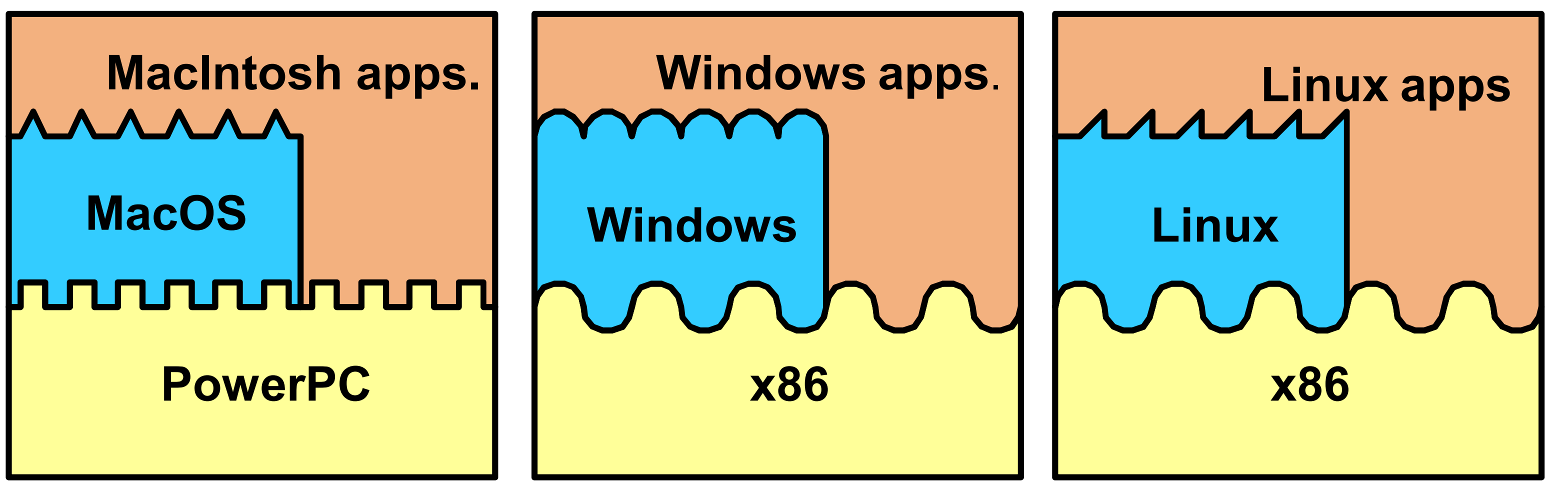
Hadi Salimi
Distributed Systems Lab,
School of Computer Engineering,
Iran University of Science and Technology,
Tehran, Iran
hsalimi@iust.ac.ir

# Introduction

- What is Virtualization?
  - Virtual Reality?
  - Virtual Memory?
  - Java Virtual Machine?
  - VMWare Virtual Machine?
- *Why they are interesting?*
- *They enable innovation in flexible, adaptive software & hardware, security, network computing (and others)*
- *They involve computer architecture in a pure sense*
- Virtualization will be a key part of future computer systems in hardware, system software and application software.

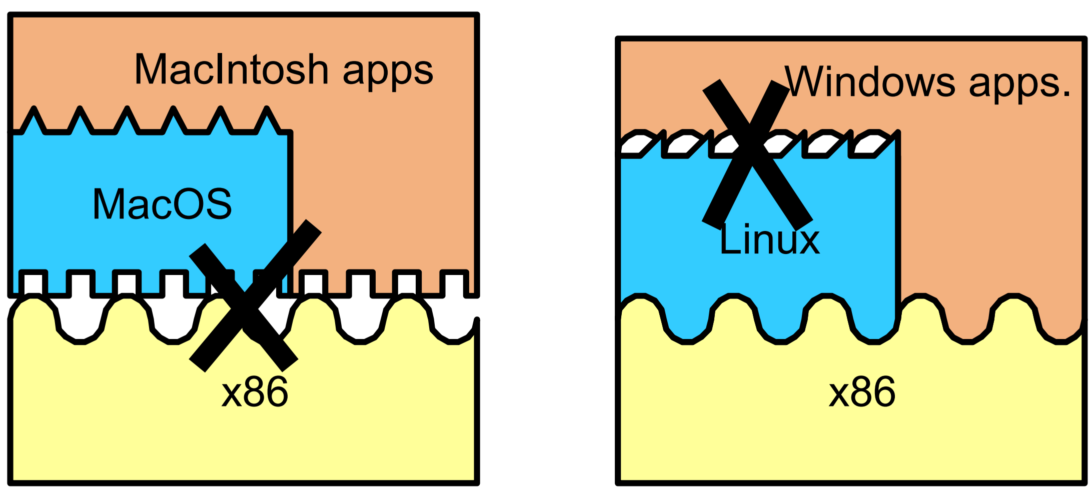
# Advantages of Standard Interfaces

- Major design tasks are decoupled
  - In space and time
- Different hardware and software development schedules
- Software can run on any machine
  *supporting a compatible interface*



# Disadvantages

- Software compiled for one ISA will not run on hardware with a different ISA
  - Apple Mac (PowerPC) binaries on an x86? No !!! ☹
- Even if ISAs are the same, OSes may differ
  - Windows NT applications on a Solaris x86? No !!! ☹
- Binary may not be optimized for the specific hardware platform it runs on
  - Intel Pentium 4 binaries on an AMD Athlon?



# Disadvantages (contd.)

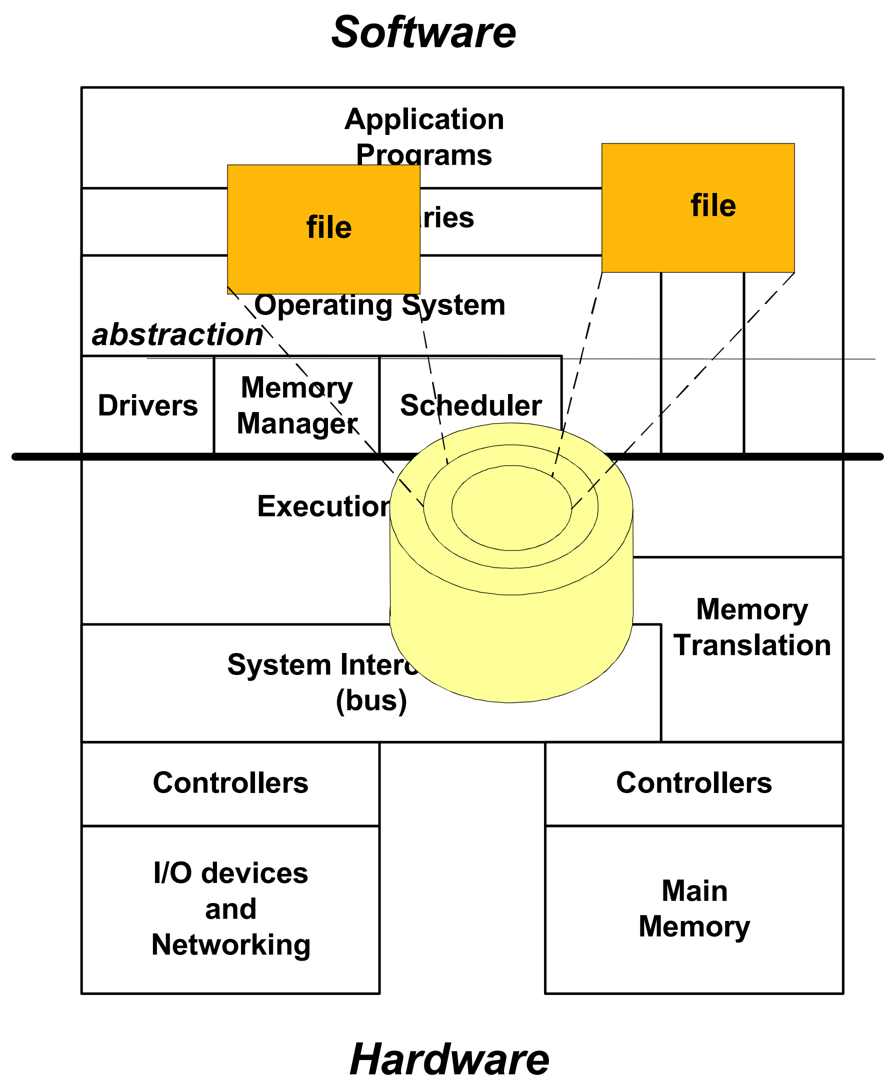
- Innovation may be inhibited by fixed ISA
  - Hard to add new instructions
    - *OR* remove obsolete ones
  - What was the most recent (successful) new ISA?
    Or new OS?
- Difficult for software to interact directly with implementation
  - Performance features
  - Power management
  - Fault tolerance
  - Software is *supposed* to be implementation independent

# Hardware Resources

- Conventional system software manages hardware resources directly
  - An OS manages the physical memory of a specific size
  - I/O devices are managed as physical entities
- Difficult to share resources except through OS
  - All users of hardware must use the same OS
  - All users are vulnerable to attack from other users sharing the resource (via security holes in OS)

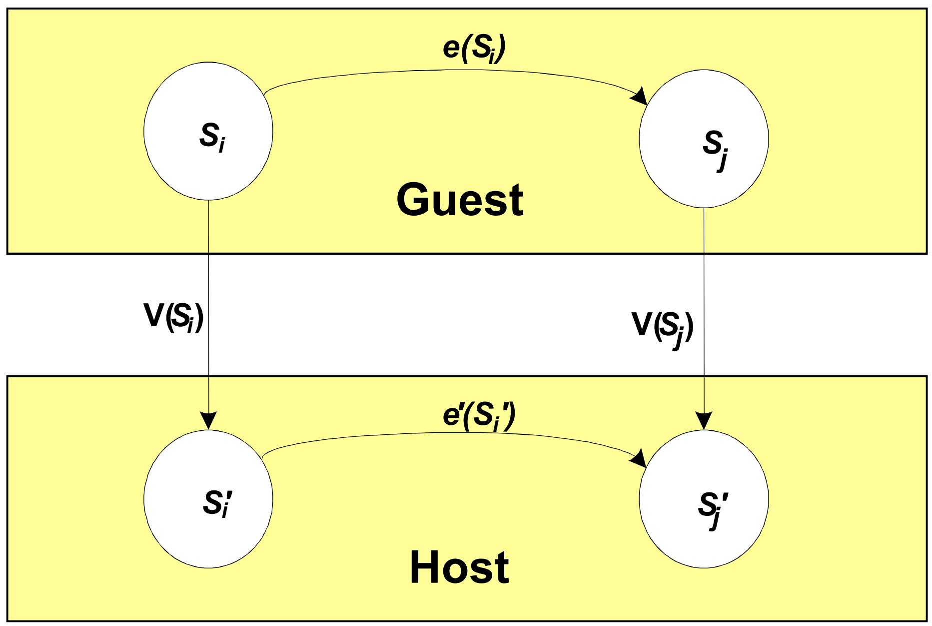
# Abstraction

- Computer systems are built on levels of abstraction
- Higher level of abstraction hide details at lower levels
- Example: files are an abstraction of a disk



# Virtualization

- An isomorphism from guest to host
  - Map guest state to host state
  - Implement "equivalent" functions

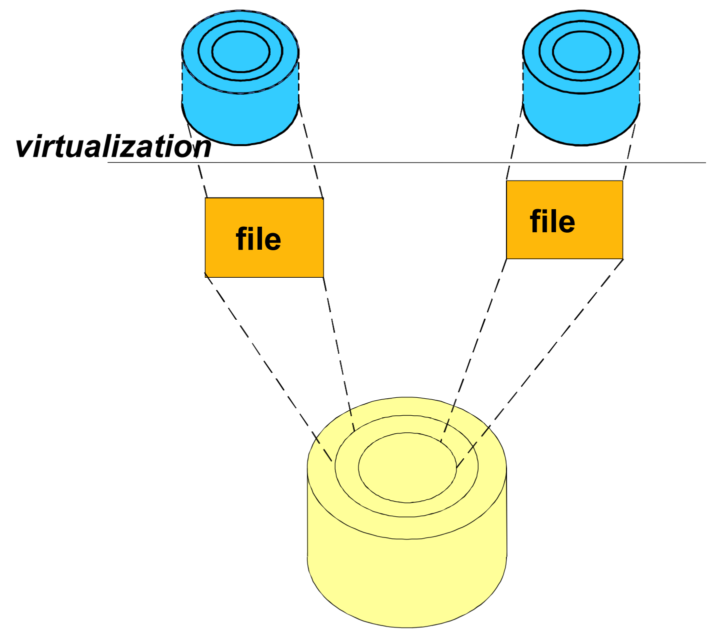


# Virtualization

- Similar to abstraction

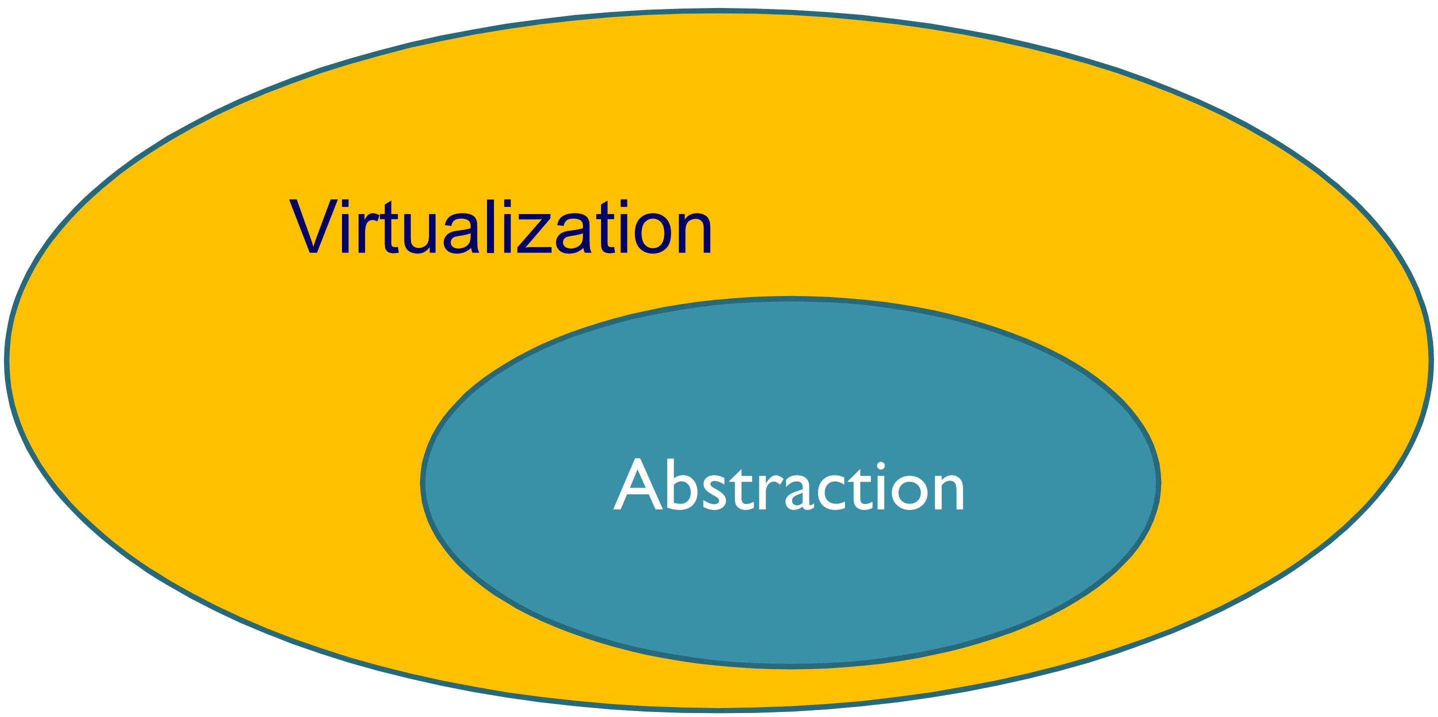
  *Except*

  - Details not necessarily hidden
- Construct Virtual Disks
  - As files on a larger disk
  - Map state
  - Implement functions
- VMs: do the same thing with the whole "machine"



# So …

- In abstraction, the details are hidden necessarily, but this is not true for virtualization.



# The "Machine"

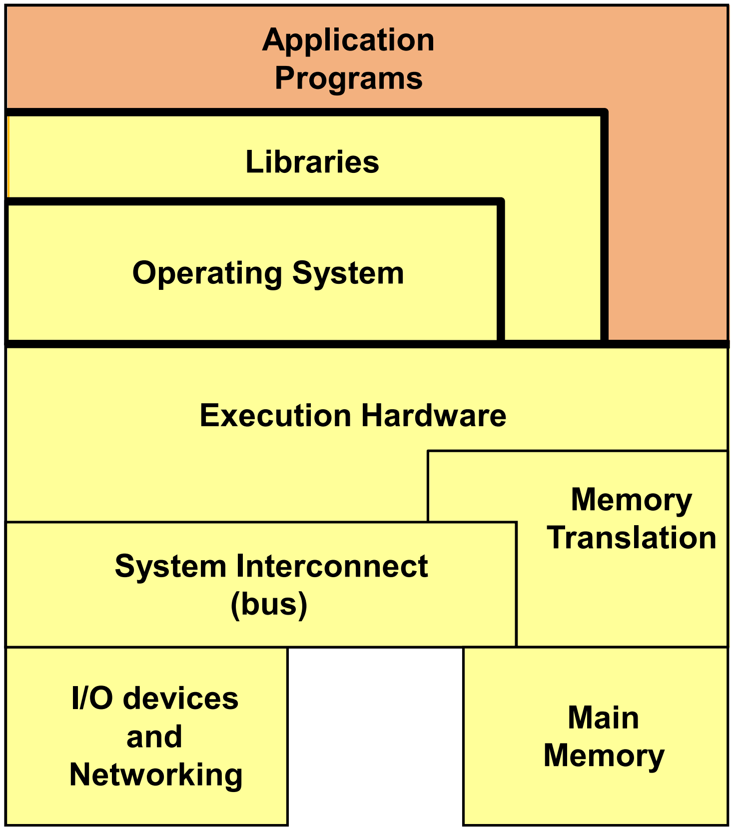
- Different perspectives on what the *Machine* is:
- OS developer
- Compiler developer
- Application programmer

**Application Program Interface**

- API
- User ISA + library calls

**Application Binary Interface**

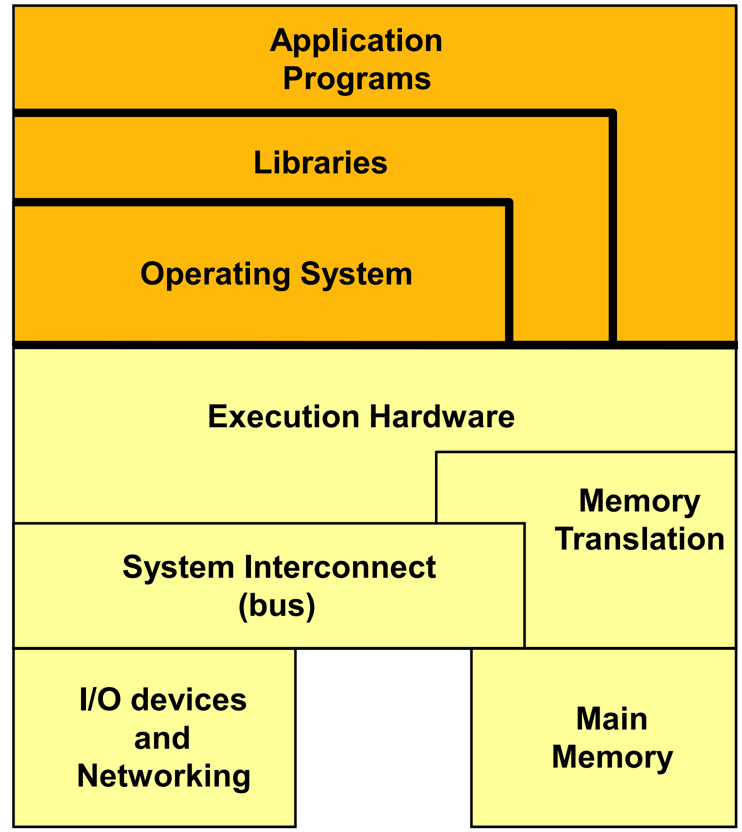
- ABI
- User ISA + OS calls



# The "Machine"

- Different perspectives on what the *Machine* is:
- OS developer

- Instruction Set Architecture
  - ISA
  - Major division between hardware and software

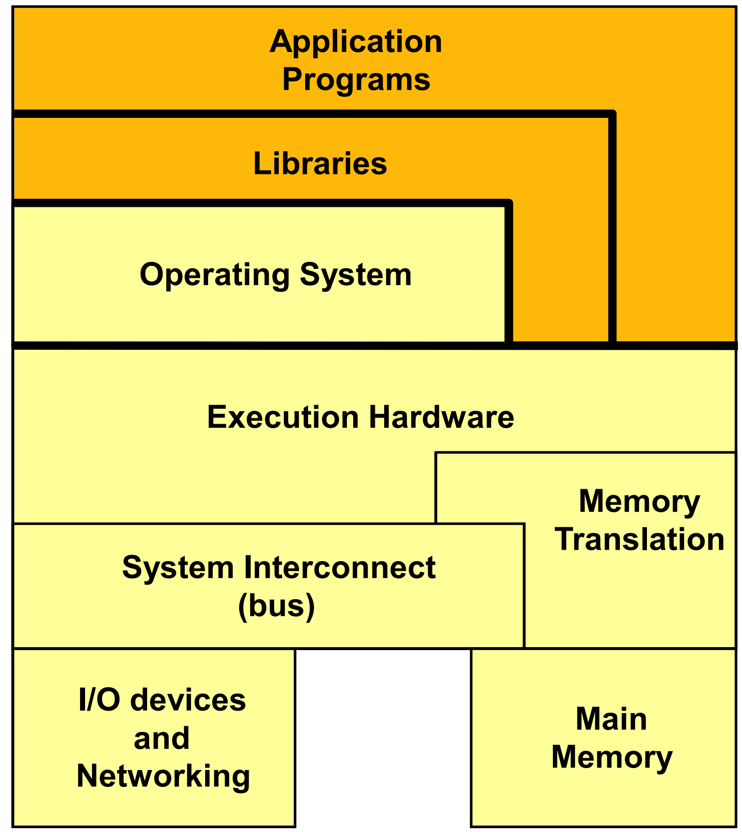


# The "Machine"

- Different perspectives on what the *Machine* is:
- Compiler developer

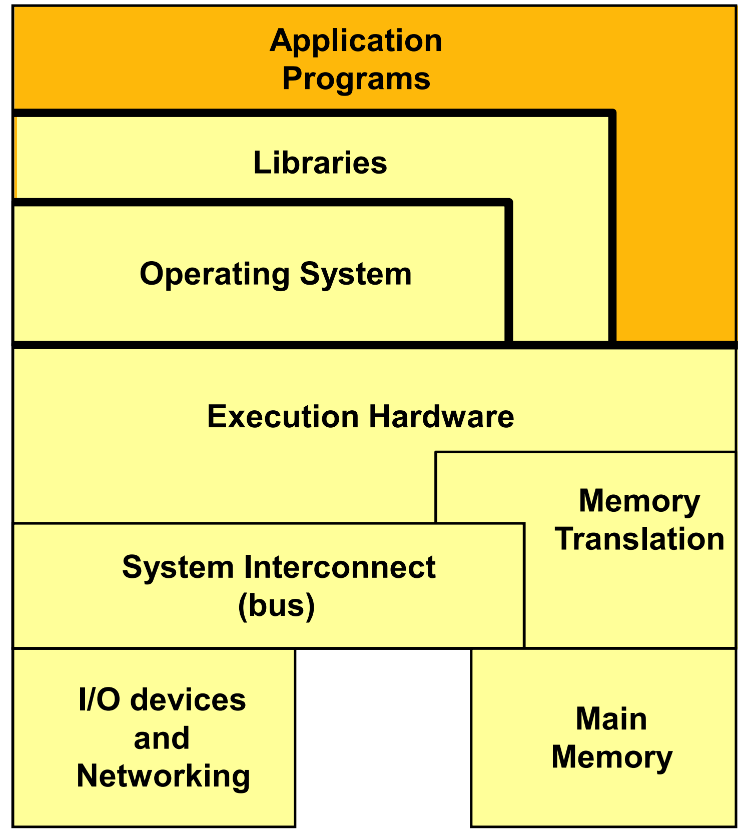
**Application Binary Interface**

- ABI
- User ISA + OS calls



# The "Machine"

- Different perspectives on what the *Machine* is:
- Application programmer



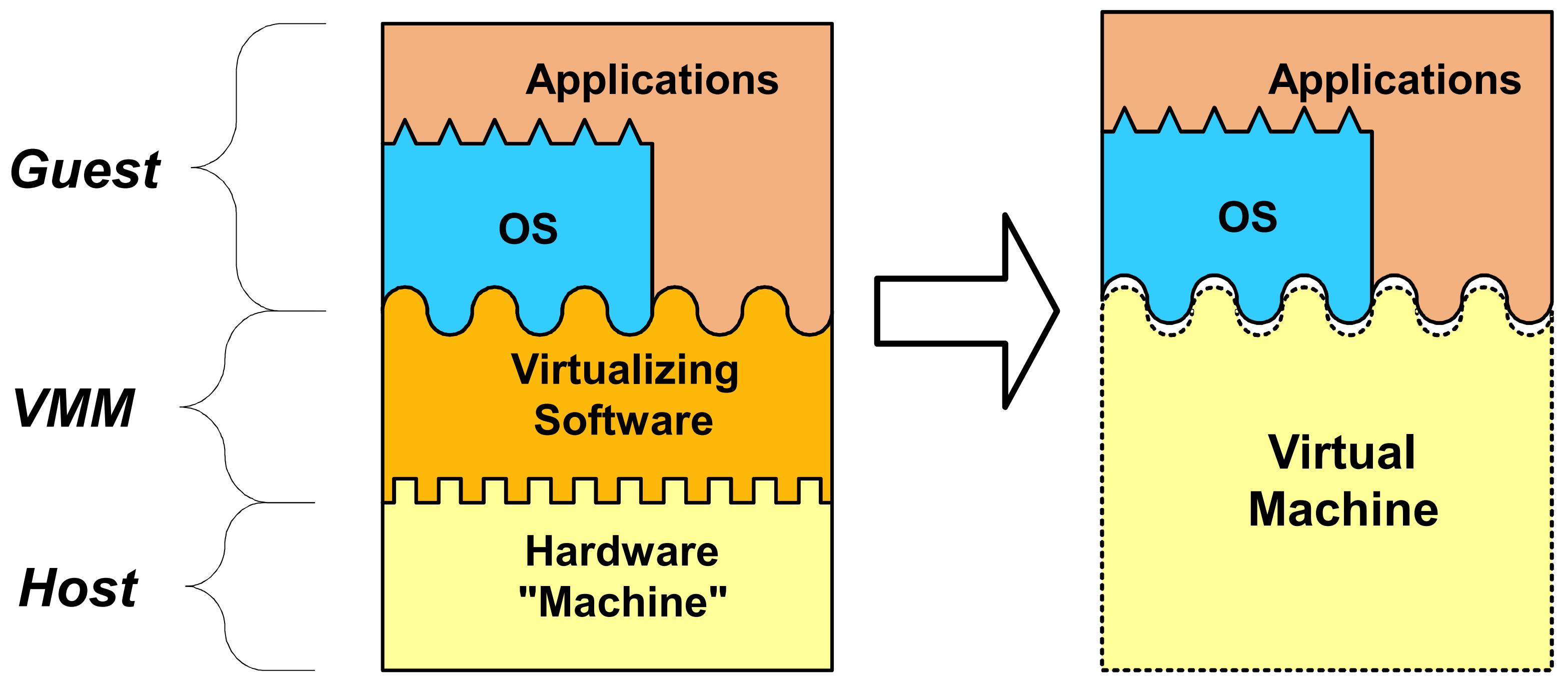

**Application Program Interface**

- API
- User ISA + library calls

# Virtual Machines

- Add *Virtualizing Software* to a *Host* platform and support *Guest* process or system on a *Virtual Machine* (VM)

- Example: System Virtual Machine



# The Family of Virtual Machines

- Lots of things are called "virtual machines"
  - IBM VM/370
  - Java
  - VMware

**Some things *not* called "virtual machines", *are* virtual machines**

- IA-32 EL
- Dynamo
- Transmeta Crusoe

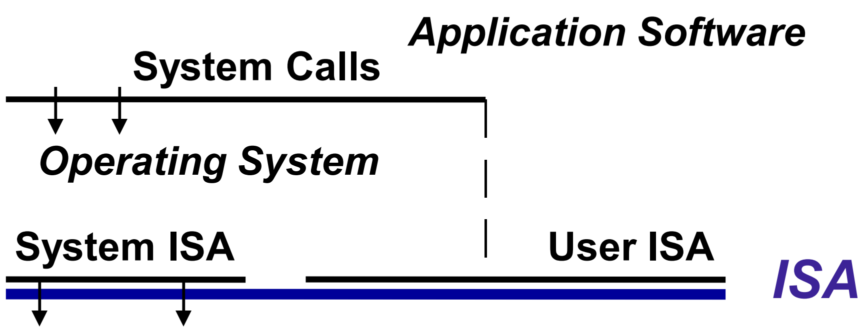
# Taking a Unified View

"The subjects of virtual machines and emulators have been treated as entirely separate. … they have much in common. Not only do the usual implementations have many shared characteristics, but this commonality extends to the theoretical concepts on which they are based"

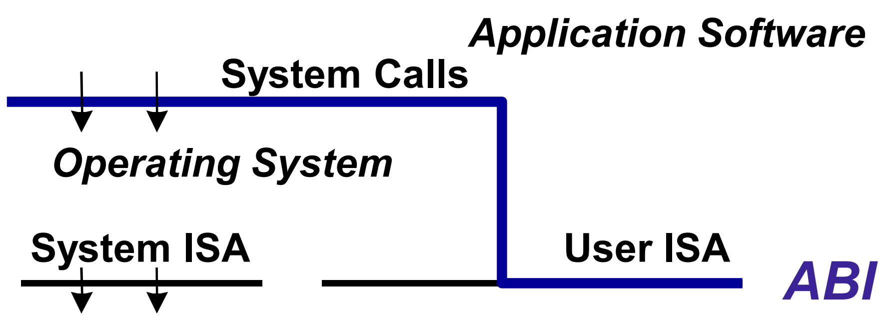
-- Efrem G. Wallach, 1973

# Major Program Interfaces

- ISA Interface -- supports all conventional software

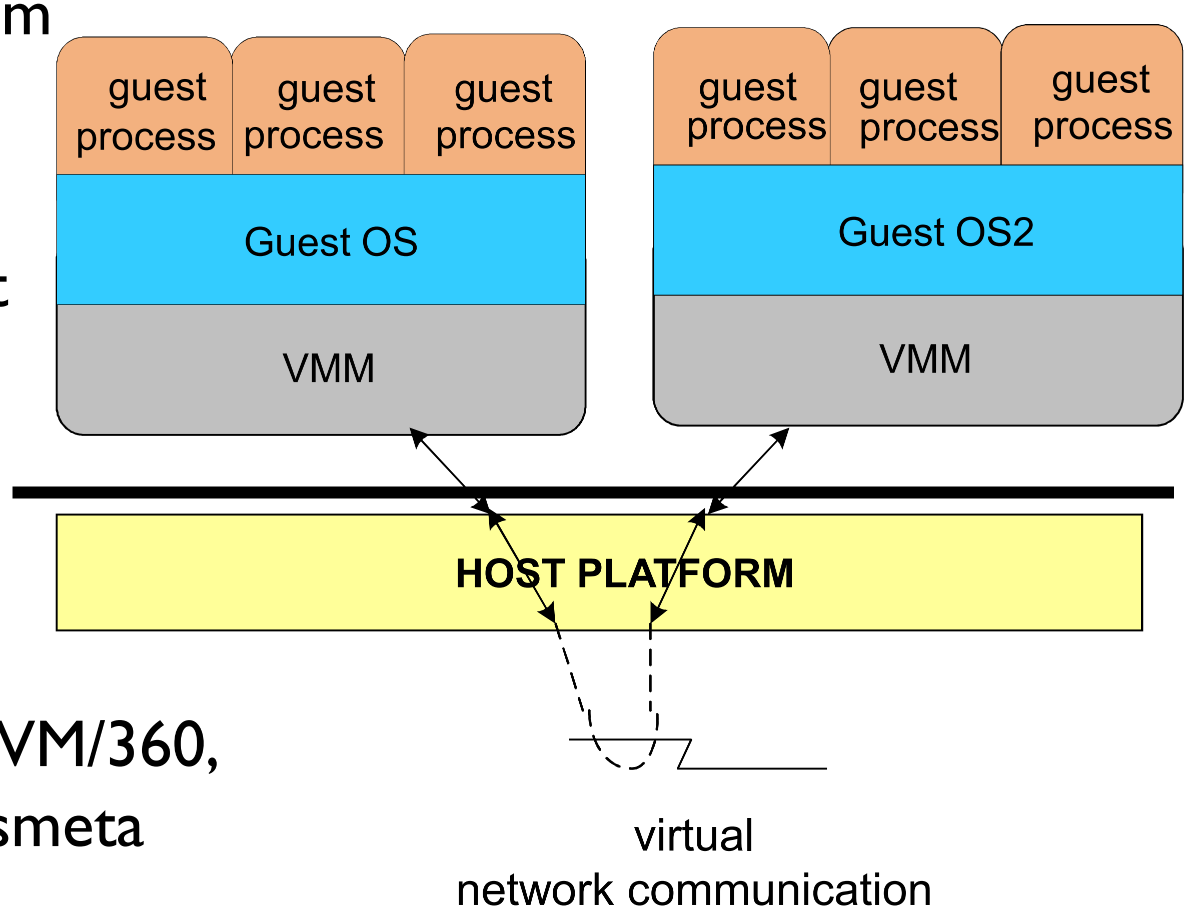



- **Application Binary Interface (ABI)**
  **-- supports application software only**



# System Virtual Machines

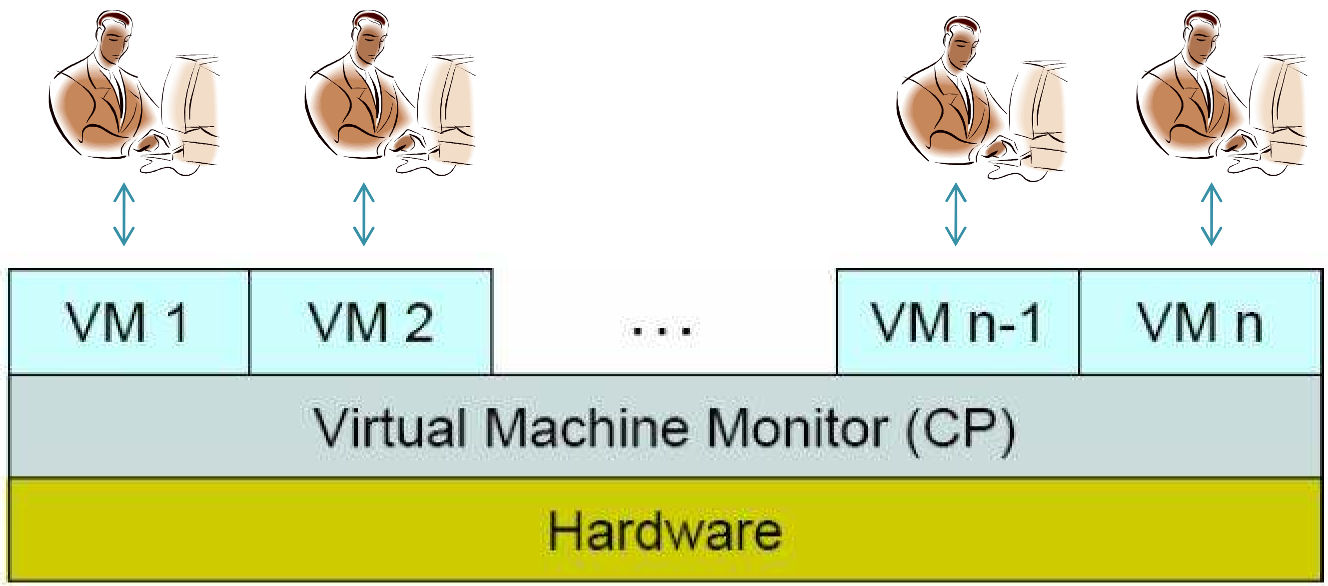
- Provide a system environment
- Constructed at ISA level
- Persistent
- Examples: IBM VM/360, VMware, Transmeta Crusoe



# Virtualization Technology (VT)

- An old technology from late 60's


- Was first coined by IBM to multiplex the power of mainframes



# VT (Cont.)

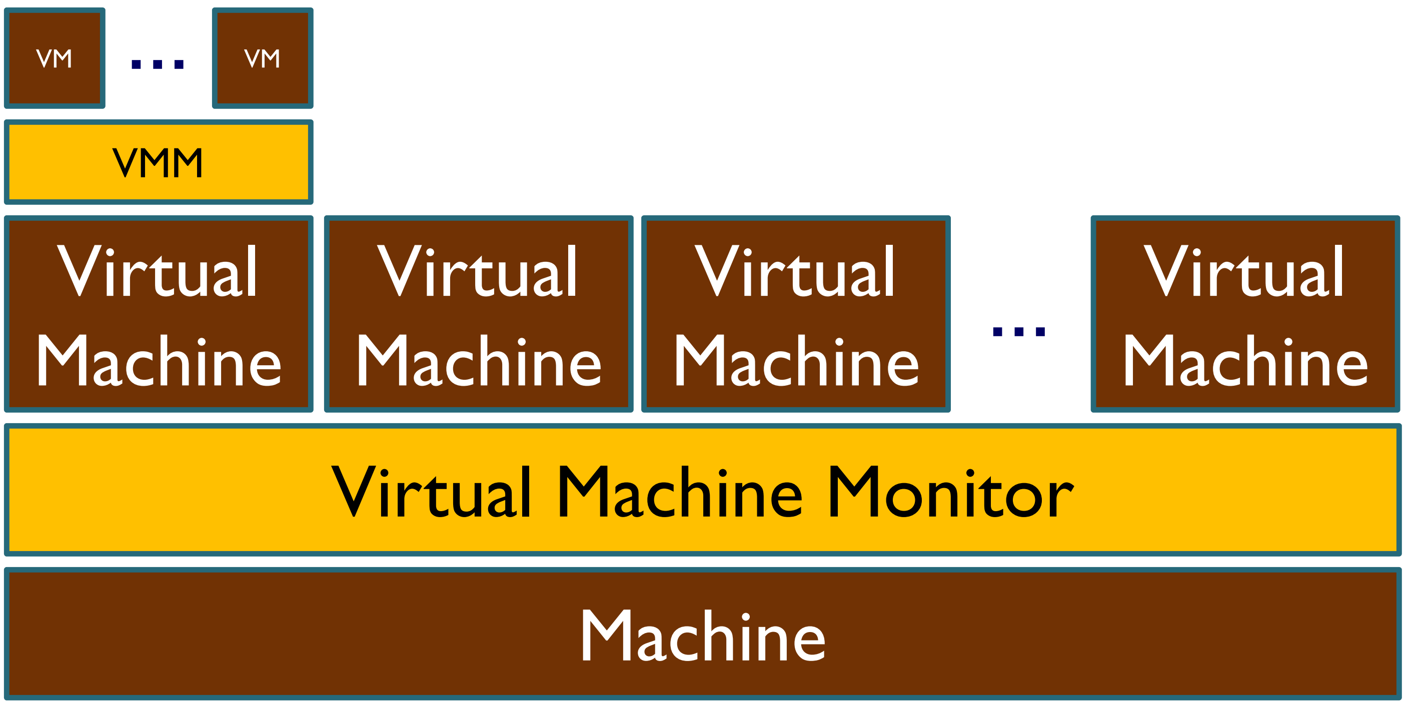
- Was dormant for decades because of its overhead

- Has became active after recent advanced in hardware  and software technologies

- Two main concepts:
  - Virtual Machine (VM)
  - Virtual Machine Monitor (VMM)

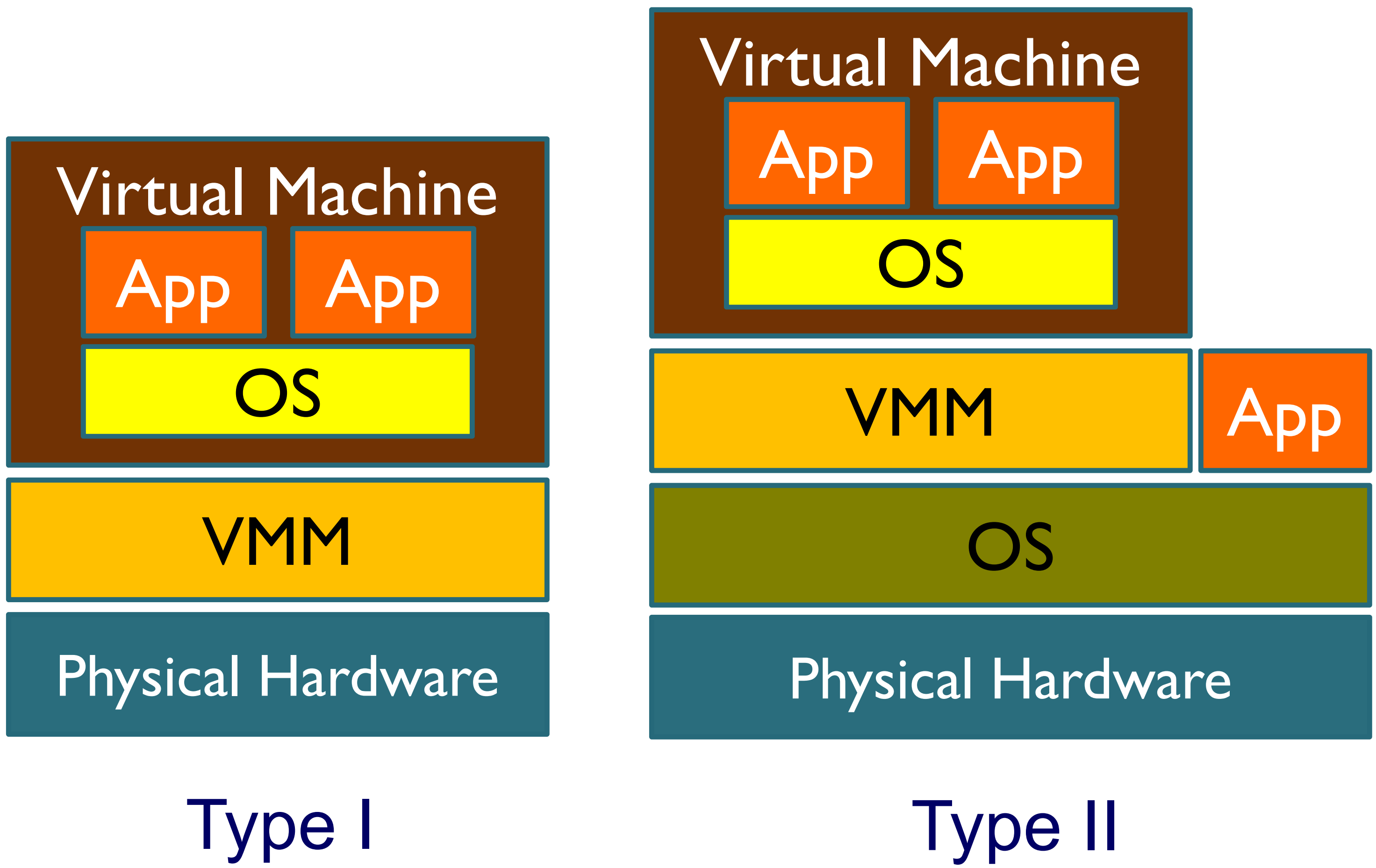
# Basic Concepts

- Virtualization basic concepts [GOL73]:
  - Virtual Machine
  - Virtual Machine Monitor



# VT Categories

- Types of VMM:



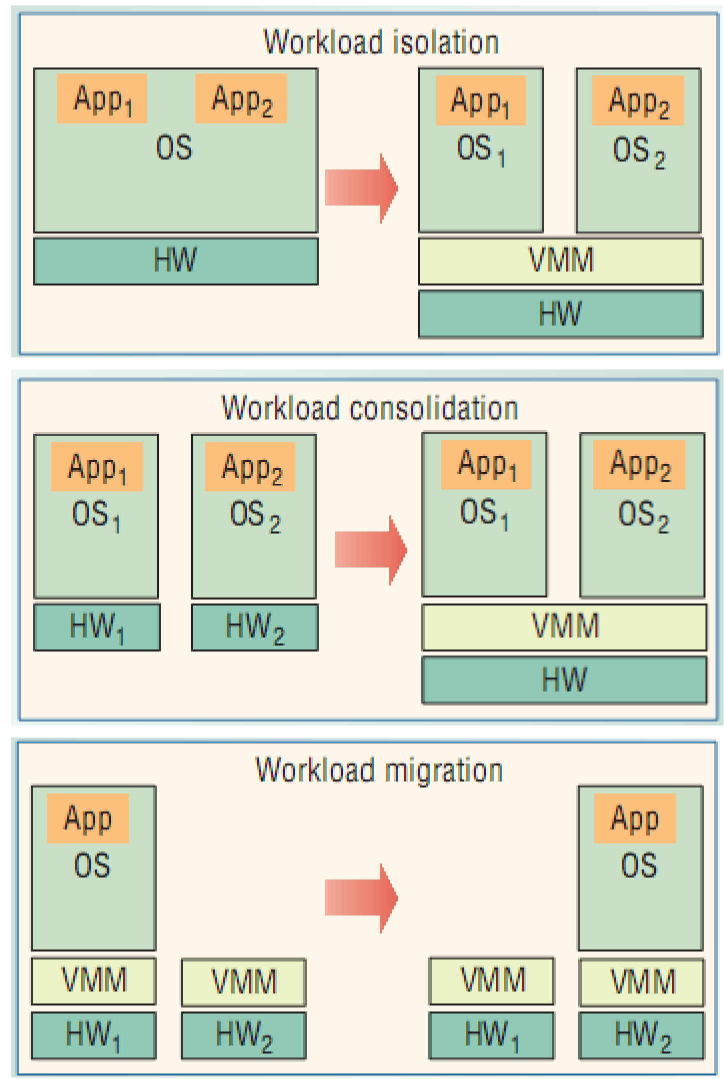
# Advantages of VT

- Three main advantages of VT:

- Isolation

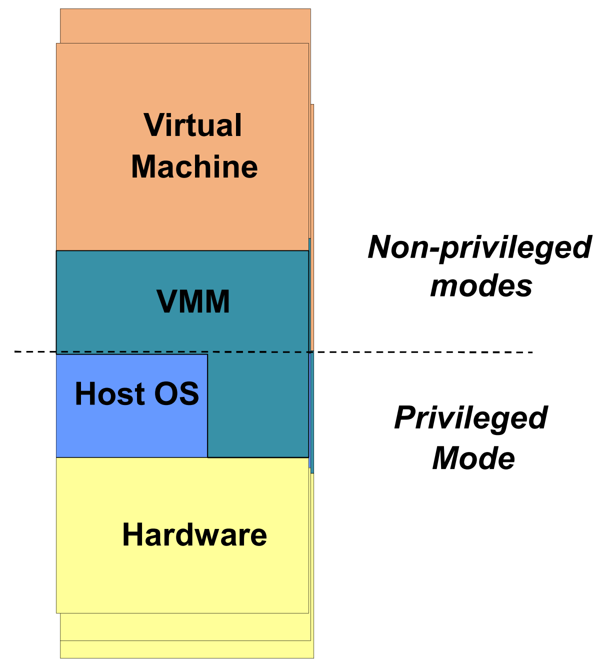
- Consolidation

- Migration



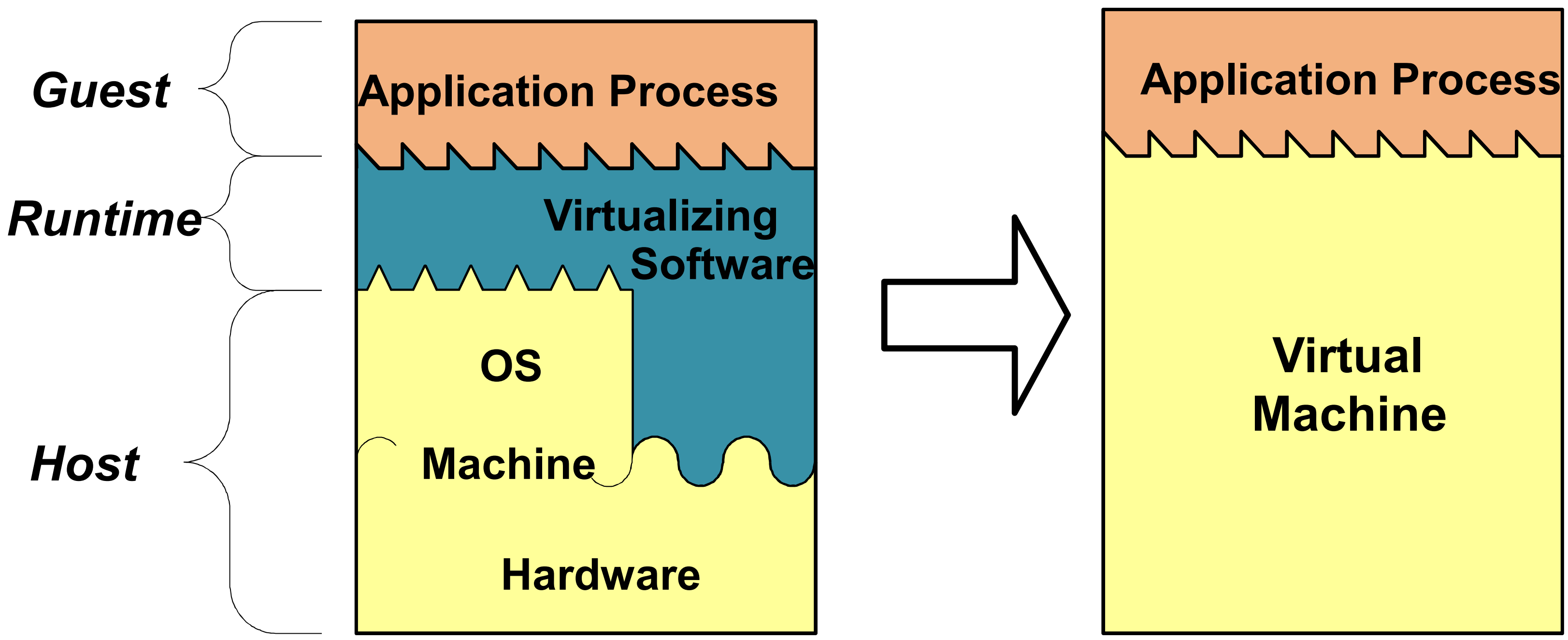
# System Virtual Machines

- Native VM System
  - VMM privileged mode
  - Guest OS user mode
  - Example: classic IBM VMs
- User-mode Hosted VM
  - VMM runs as user application
- Dual-mode Hosted VM
  - Parts of VMM privileged; parts non-privileged
  - Example VMware



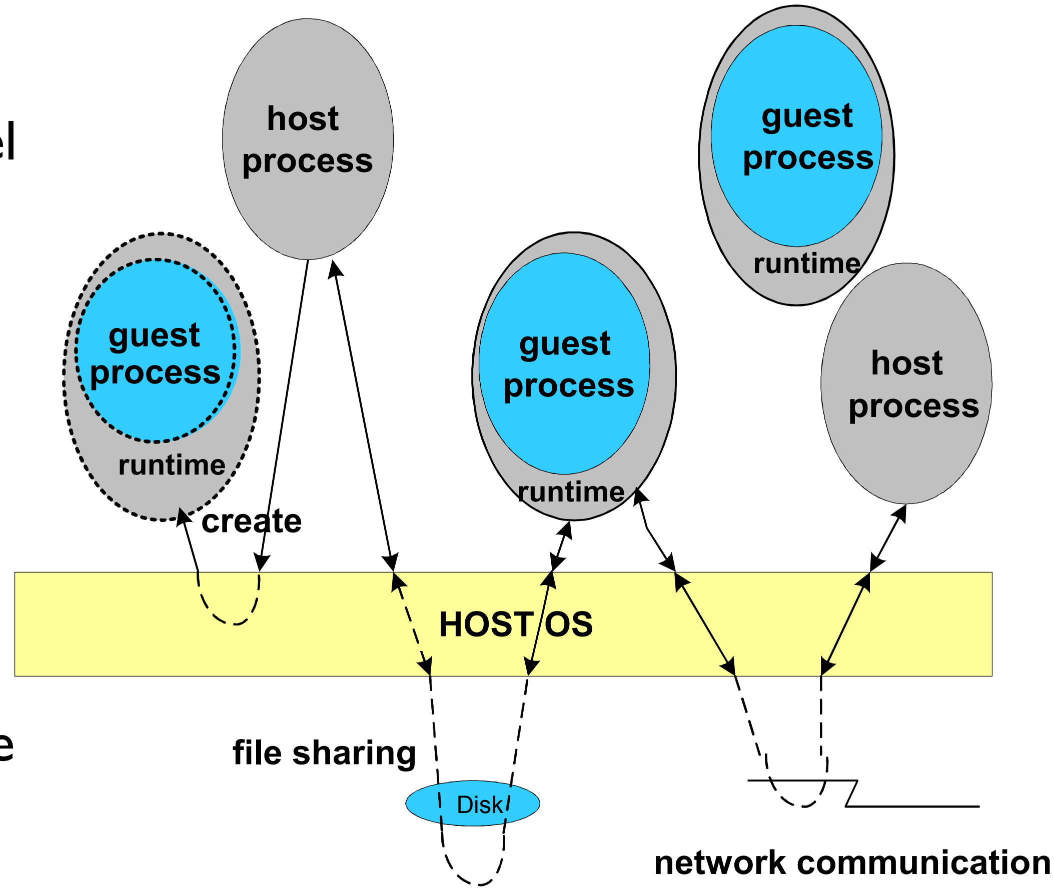
# Process VMs

- Execute application binaries with an ISA *different* from hardware platform
- Couple at ABI level via *Runtime System*
- Examples: IA-32 EL, FX!32



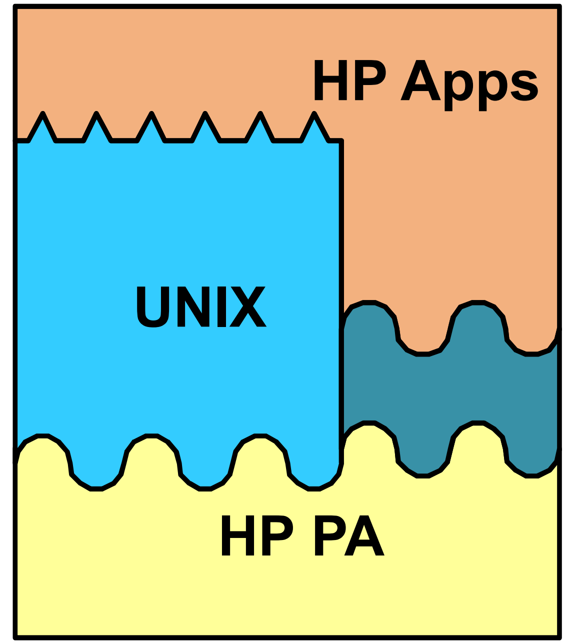
# Process Virtual Machines

- Constructed at ABI level
- *Runtime* manages guest process
- Not persistent
- Guest processes may intermingle with host processes
- As a practical matter, guest and host OSes are often the same
- Dynamic optimizers are a special case
- Examples: IA-32 EL, FX!32, Dynamo



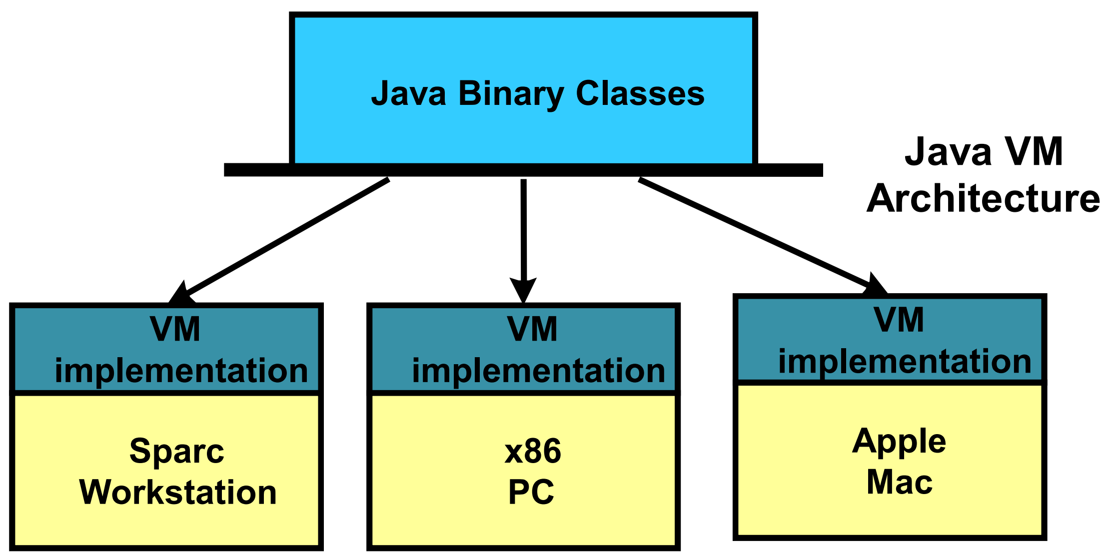
# Same-ISA Dynamic Binary Optimizers

- Optimize Binary at Runtime
- An ABI level optimization
- A type of Process VM
- Example HP Dynamo
  - Can optimize for dynamic properties of program
  - Can optimize for a specific processor implementation



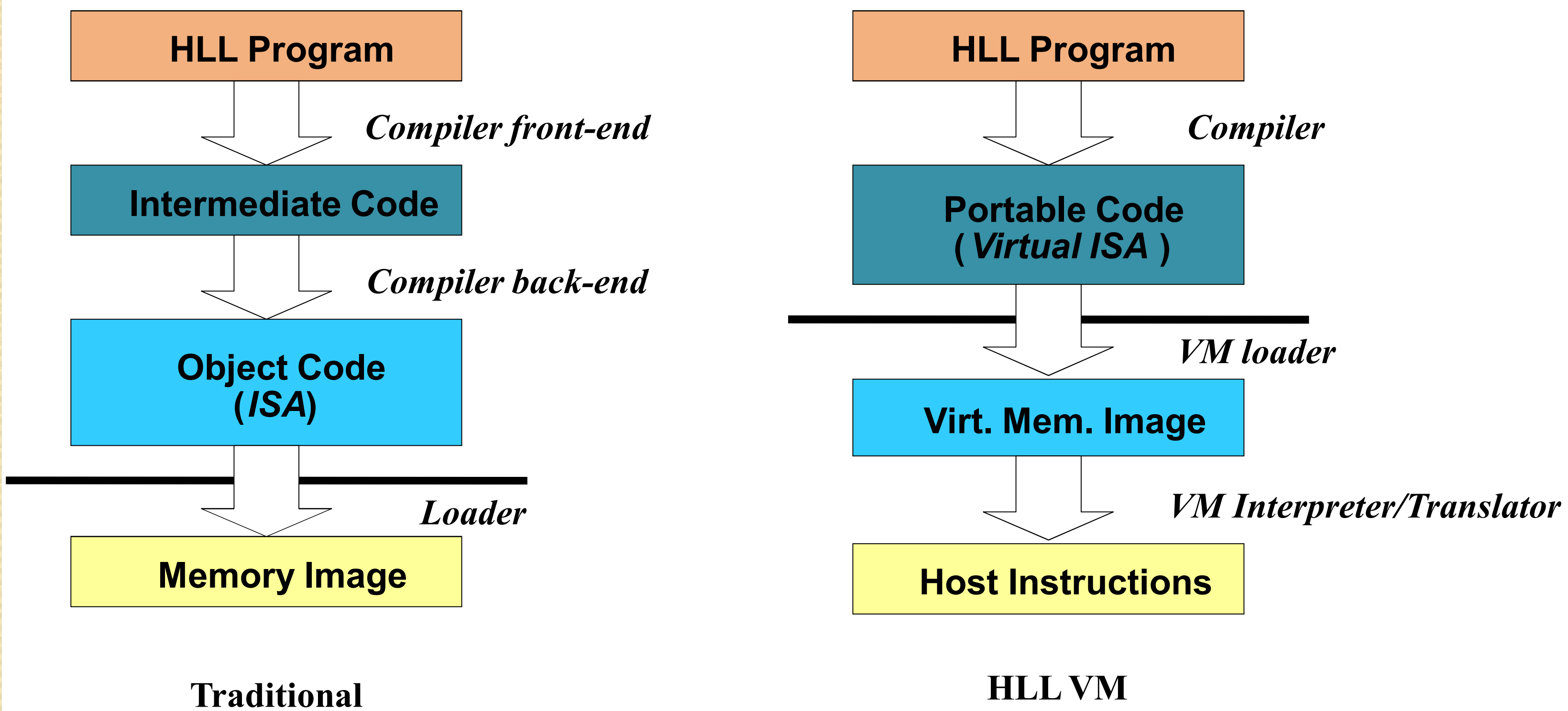
# HLL VMs

- Java and CLI are recent examples
- Binary class files are distributed
- “ISA” is part of binary class format
- OS interaction via APIs (part of VM platform)



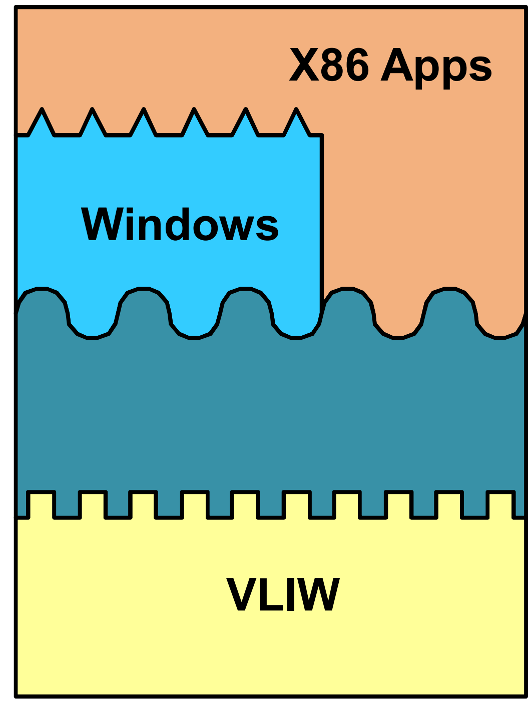
# High Level Language Virtual Machines

- Raise the level of abstraction
  - User higher level virtual ISA
  - OS abstracted as standard libraries
- Process VM (or API VM)

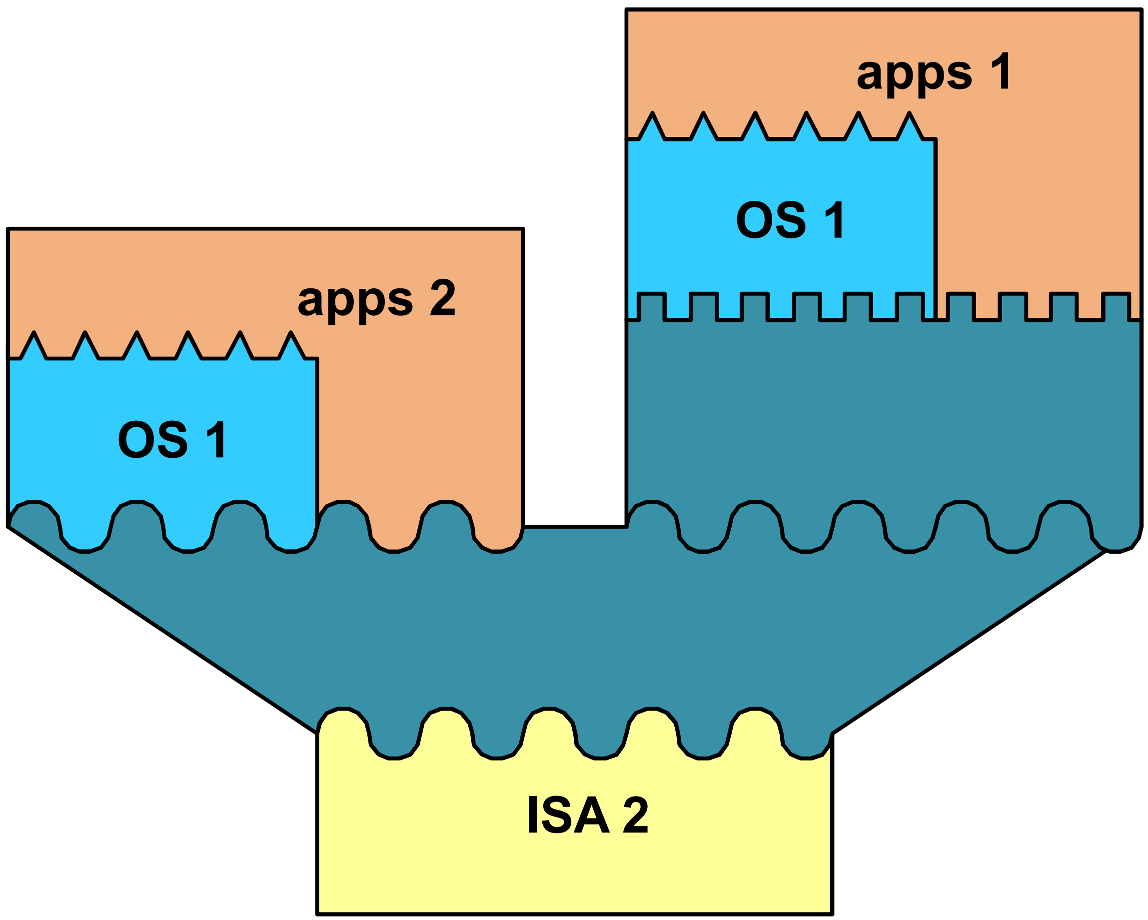


# Co-Designed VMs

- Perform both translation and optimization
- VM provides interface between standard ISA software and implementation ISA
- Primary goal is performance or power efficiency
- Use proprietary implementation ISA
- Transmeta Crusoe and IBM Daisy best-known examples



# Composition



# Composition: Example

Java application

↓ JVM

Linux x86

↓ VMware

Windows x86

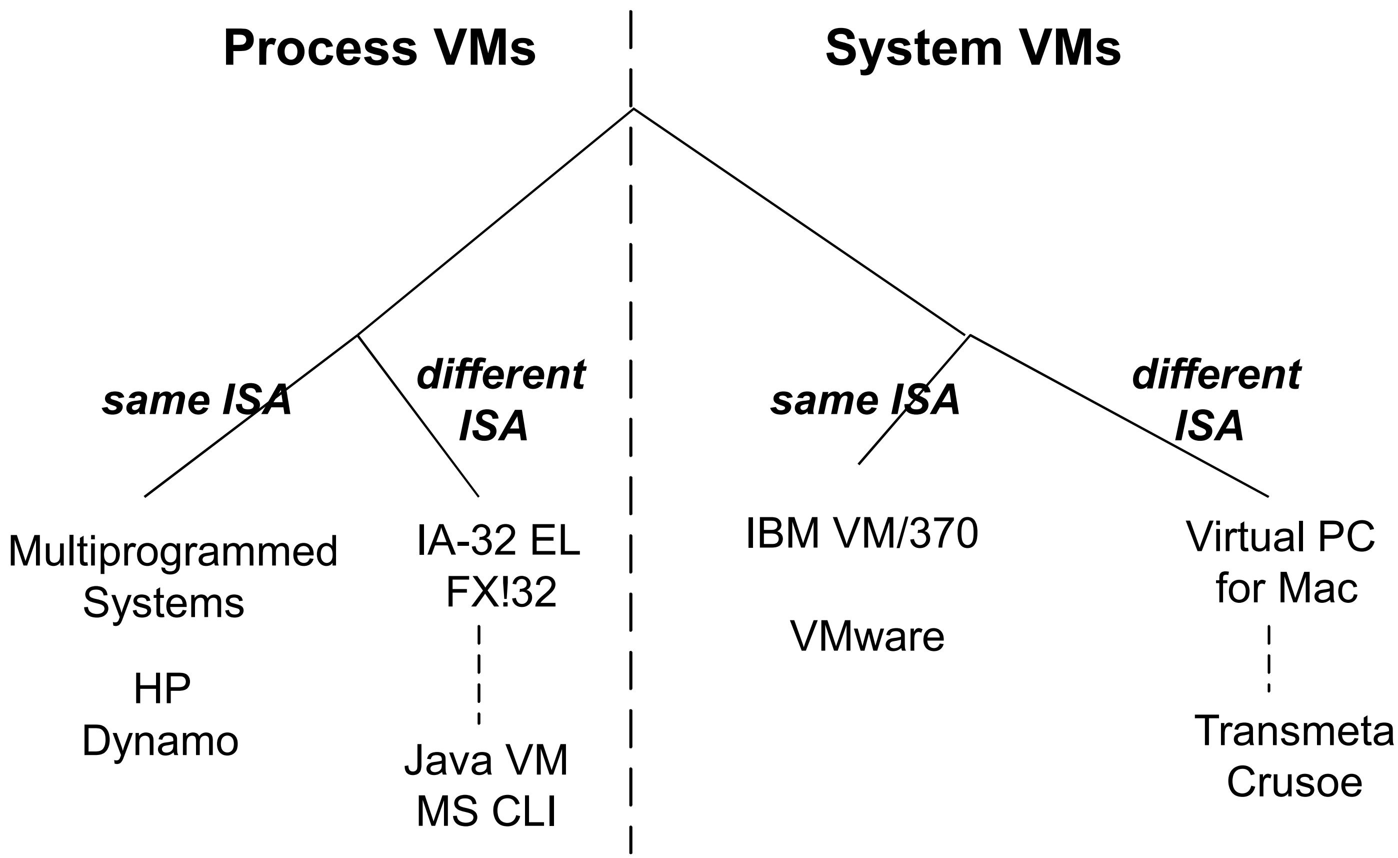
↓ Code Morphing

Crusoe VLIW

# Summary (Taxonomy)

VM type (Process or System)

Host/Guest ISA same or different



Any Questions?