Introduction to Virtual Machines

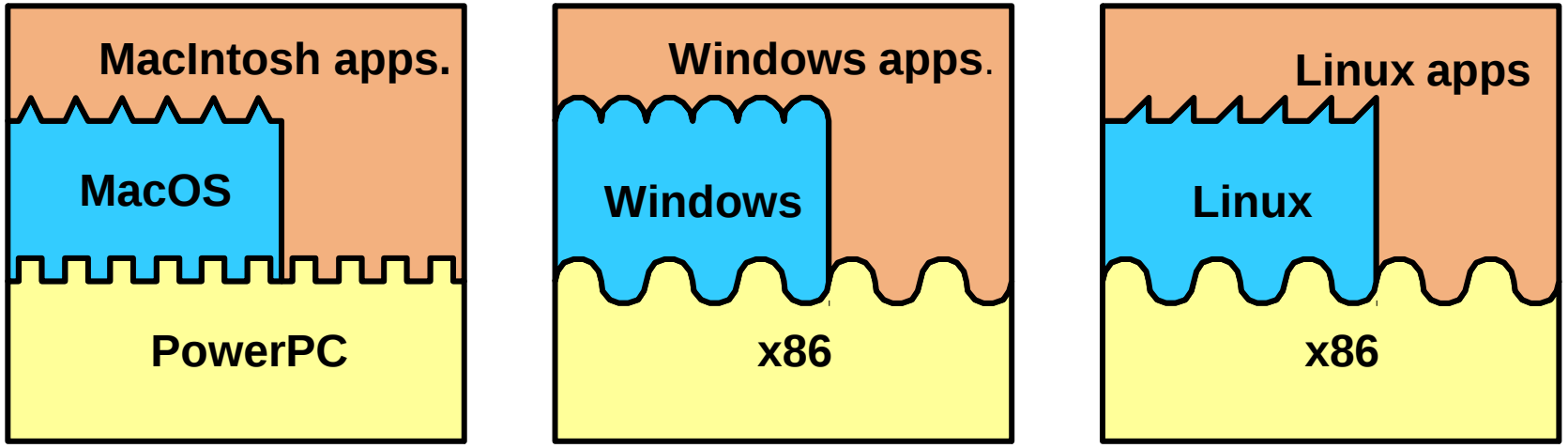
- Introduction
- Abstraction and interfaces
- Virtualization
- Computer system architecture
- Process virtual machines
- System virtual machines

Abstraction

- Mechanism to manage complexity in computer systems.
- Mechanism consists of
  - partition the design of a system into levels
  - allow higher levels to ignore the implementation details of lower levels
- In computer systems, lower levels are implemented in hardware, and higher levels in software.

Interfaces

- An interface defines the communication boundary between two entities
  - hierarchical relationship
  - linear relationship
- Software can run on any machine supporting a compatible interface

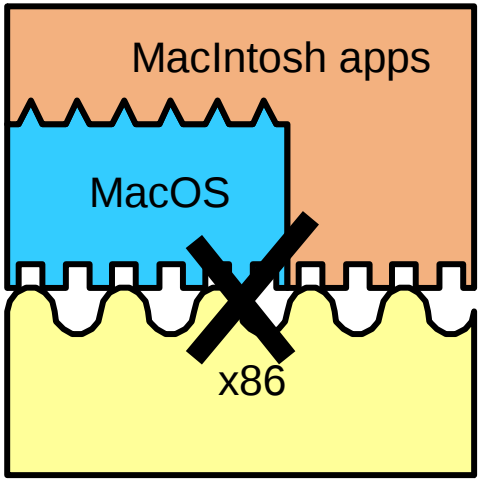


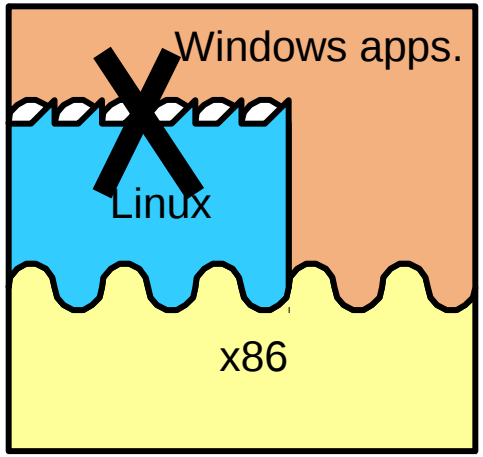
Interfaces – Advantages

- Allows de-coupling of computer design tasks
  - each component provides an abstraction of itself to the outside world
- Work on different components can progress independently
- Helps manage system complexity

Interface – Disadvantages

- Software compiled for one ISA will not run on hardware with a different ISA
  - powerPC binaries on an x86 machine ?
- Even if ISA's are same Oses may differ
  - MS–Windows applications Sun Solaris ?





Interface – Disadvantages (2)

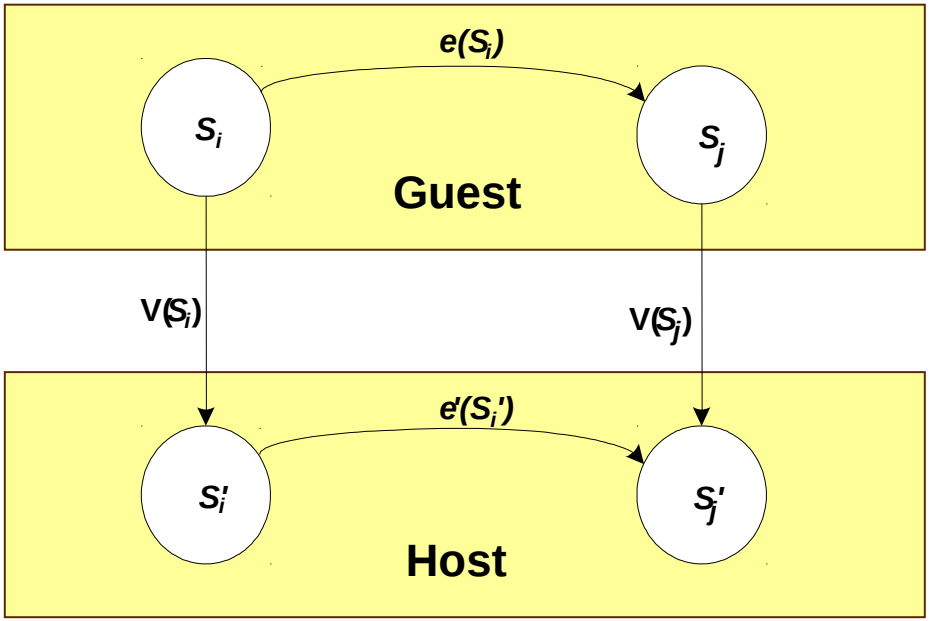
- Binaries may not be optimized for the platform they run on
  - Intel Pentium binaries on AMD Athlon ?
- Innovation may be inhibited by a fixed ISA
  - hard to change instruction sets
- Application software cannot *directly* exploit microprocessor implementation features
  - software *supposed* to be implementation-neutral !

Virtualization

- Removes some constraints imposed by system interfaces, and increases flexibility
  - improves availability of application software
  - removes the assumption of a single management regime, improving security and failure isolation
- Provide a *different* view to a particular computer resource
  - not necessarily a *simpler* view

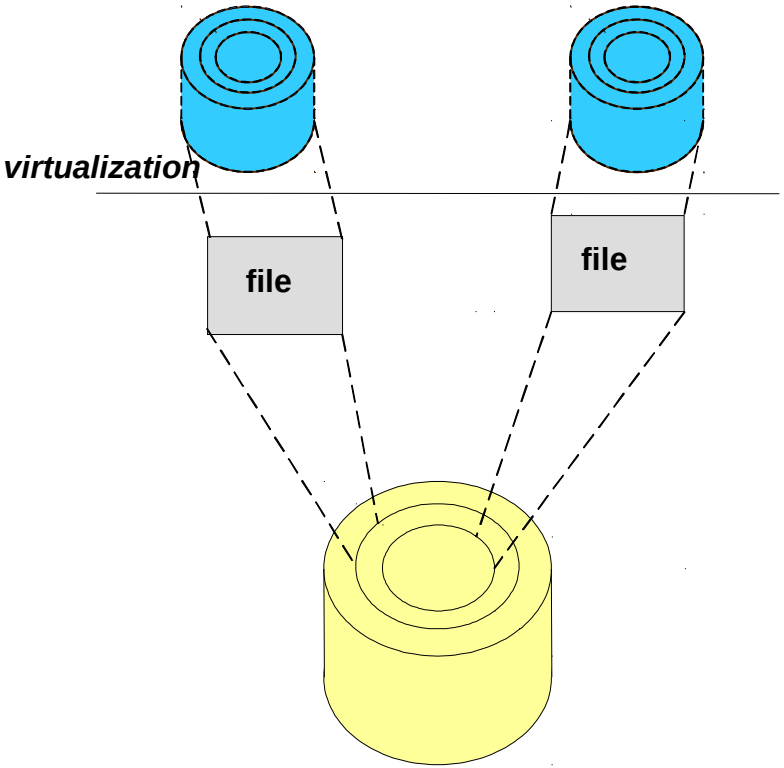
Virtualization (2)

- Virtualization constructs an isomorphism that maps a virtual *guest* system to a real *host*.



Virtualization (3)

- Virtualization Vs. abstraction
  - virtualization does not necessarily hide details



Virtual Machines

- Concept of *virtualization* applied to the entire machine.
- A virtual machine is implemented by adding a layer of software to a real machine to support the desired virtual machine's architecture.
- Virtualization
  - mapping of virtual resources or state to real resources
  - use of real machine instructions to carry out actions specified by the virtual machine instructions

Some Benefits of VMs

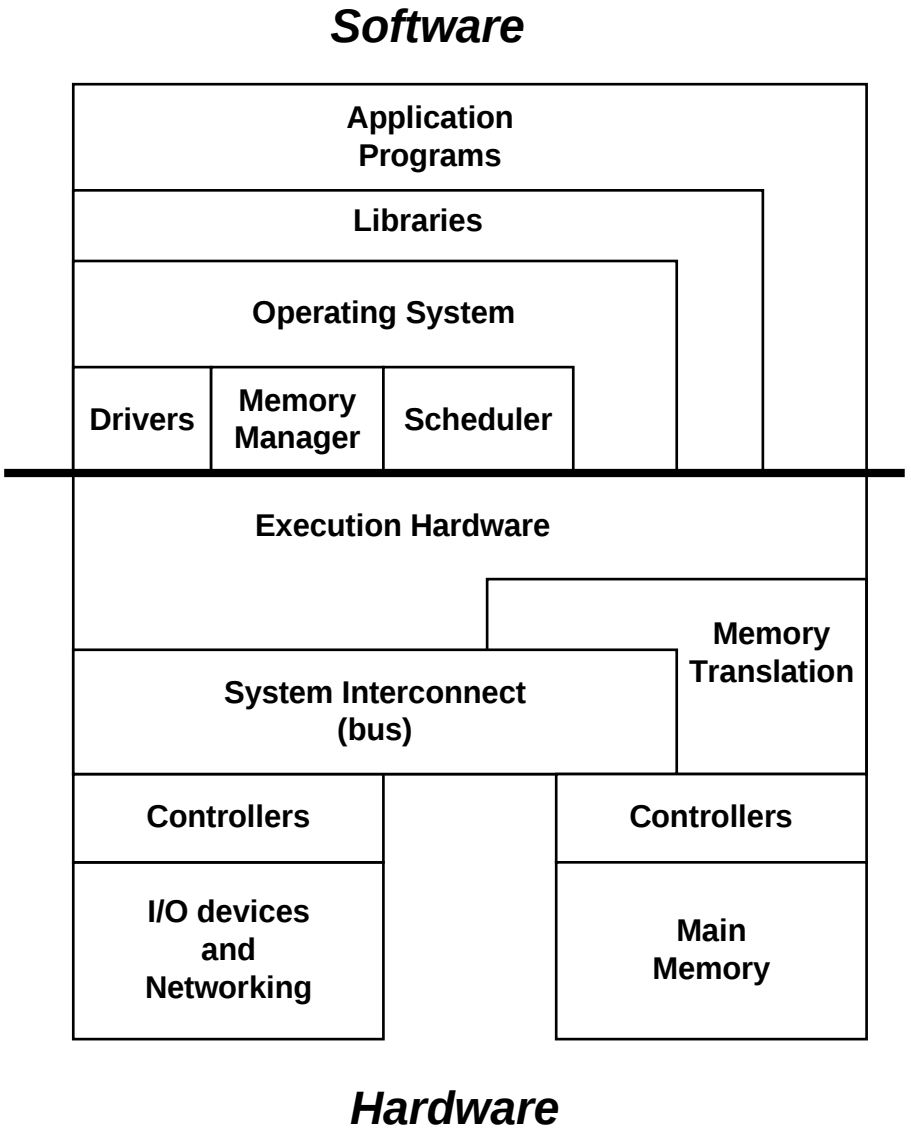
- Flexibility
- Portability
- Isolation
- Security

Computer System Architecture

- Architecture – functionality and appearance of a computer system, but not the details of its implementation
- Implementation – the actual embodiment of an architecture

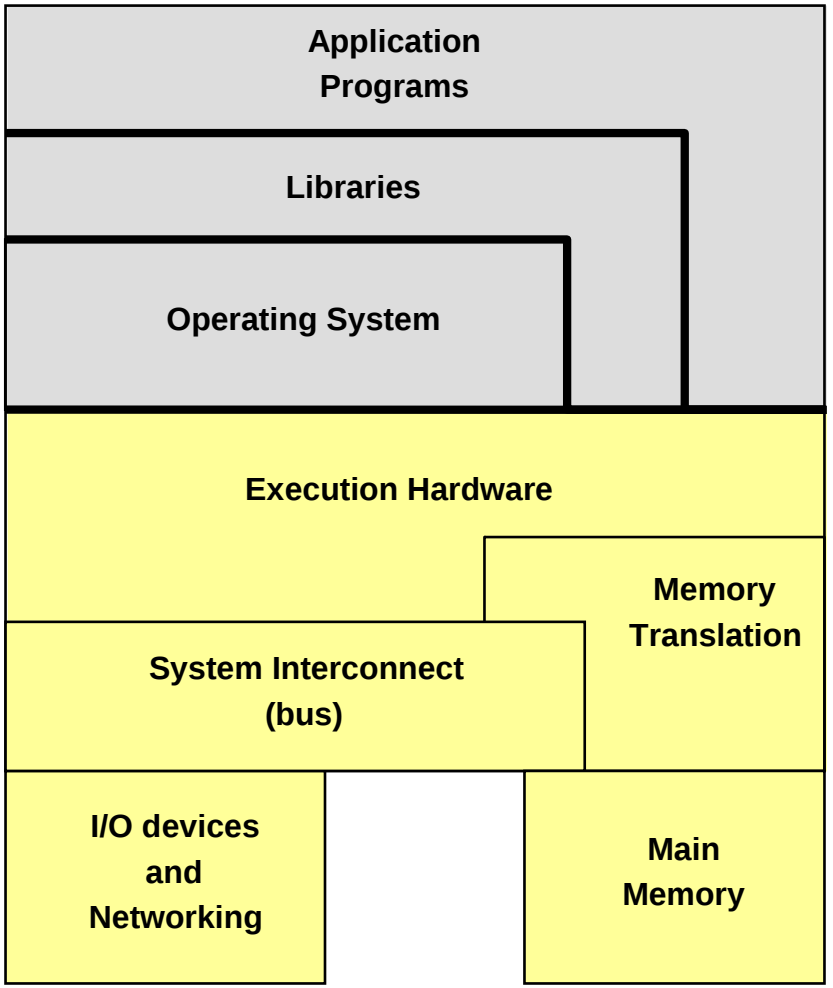
Computer Architecture (2)

- Computer systems are built of levels of abstraction
  - hierarchical abstraction
  - well-defined interfaces



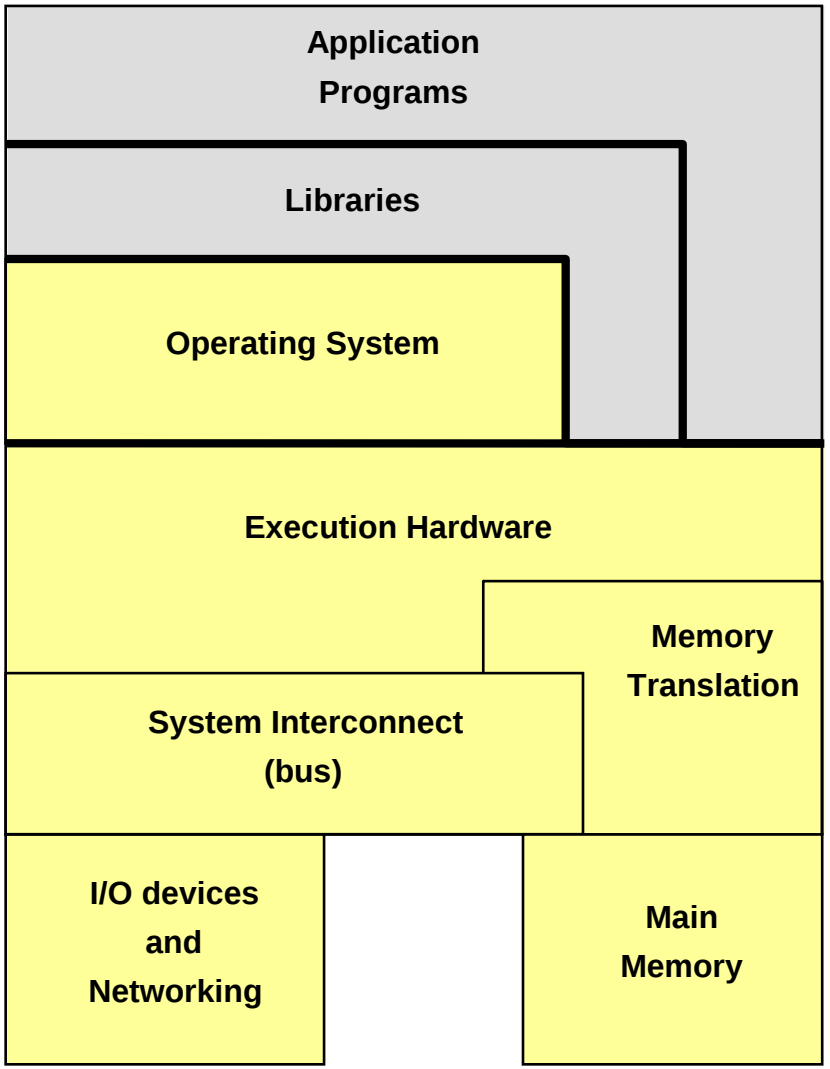
The ISA Interface

- Interface between hardware and software
- Important for
  - OS developer



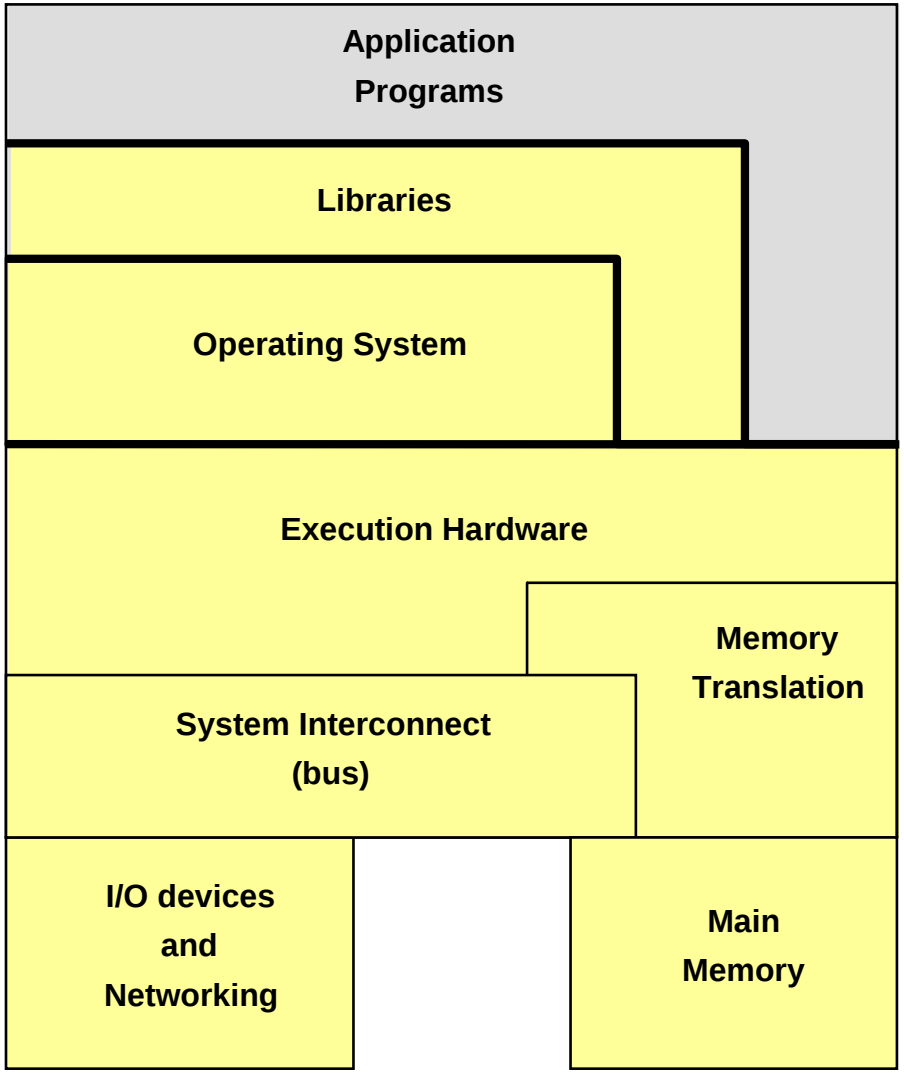
The ABI Interface

- Application Binary Interface (ABI)
  - user ISA + system calls
- Important for
  - compiler writers



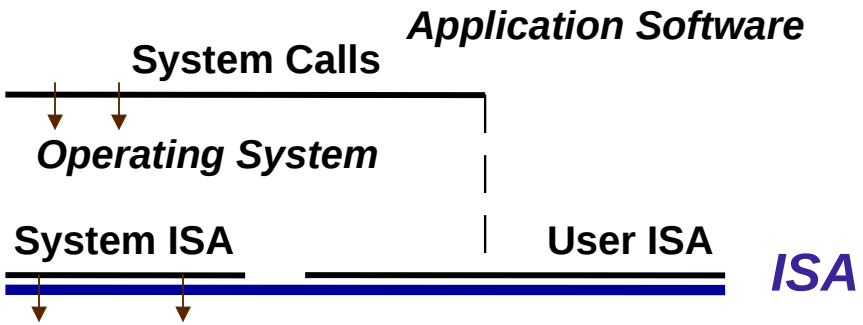
The API Interface

- Application Programming Interface (API)
  - user ISA + library calls
- Important for
  - application programmers

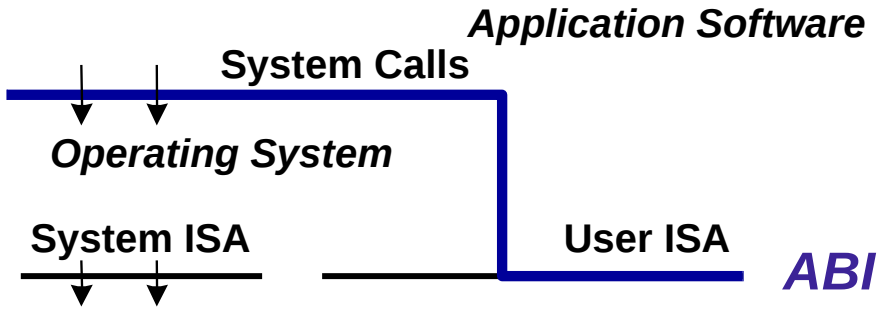


Major Program Interfaces

- ISA – supports all conventional software

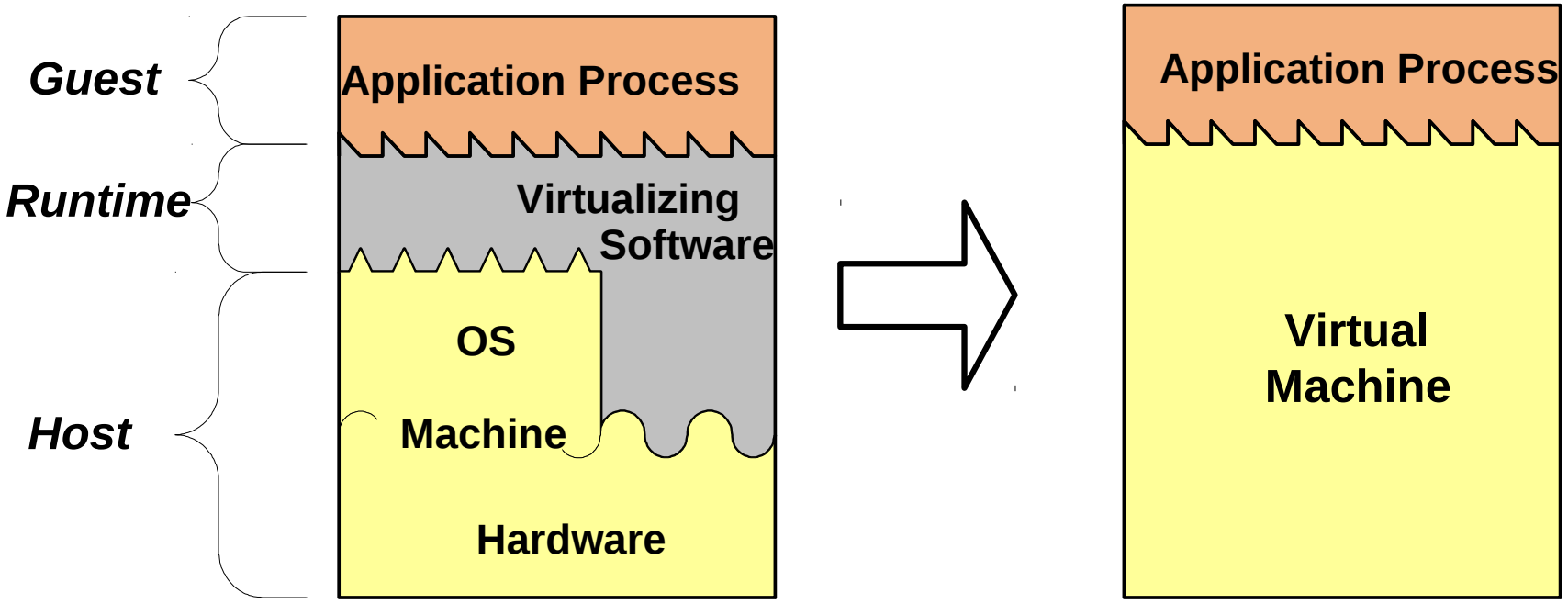


- ABI – supports application software only



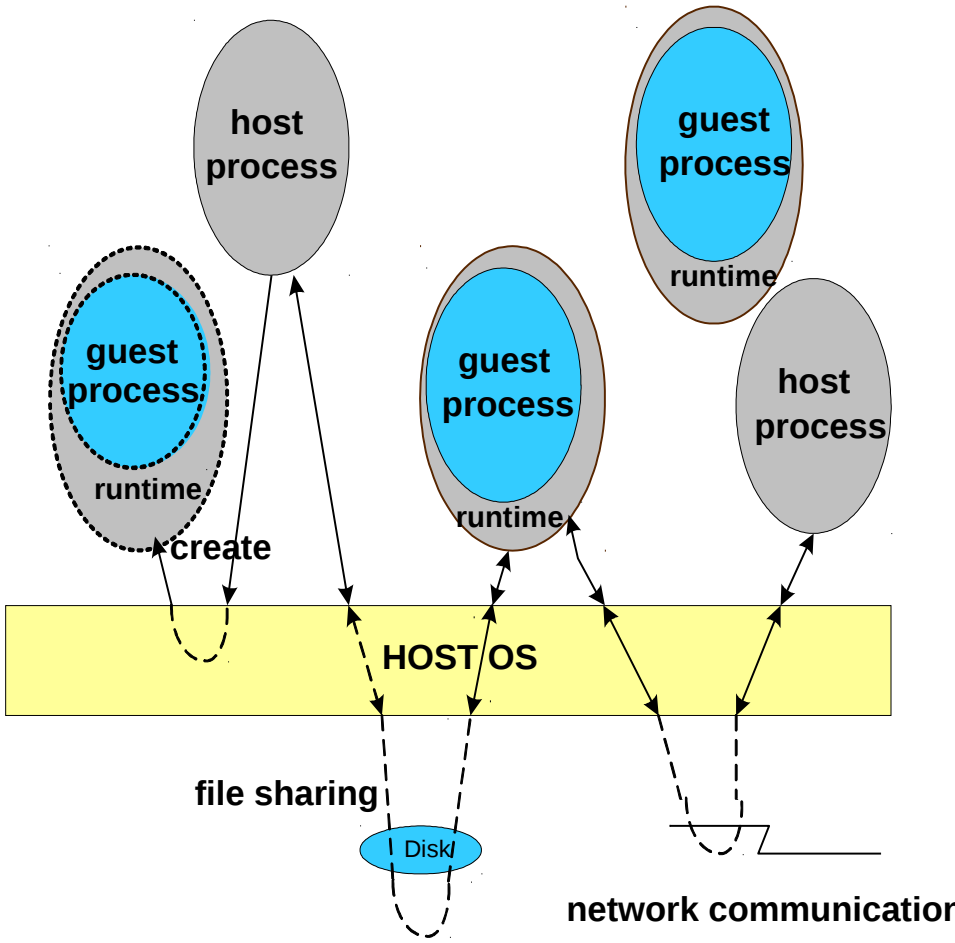
Process Virtual Machines

- *Process virtual machine* is capable of supporting an individual process
  - different *guest* and *host* ISA
  - couple at ABI level via *runtime system*



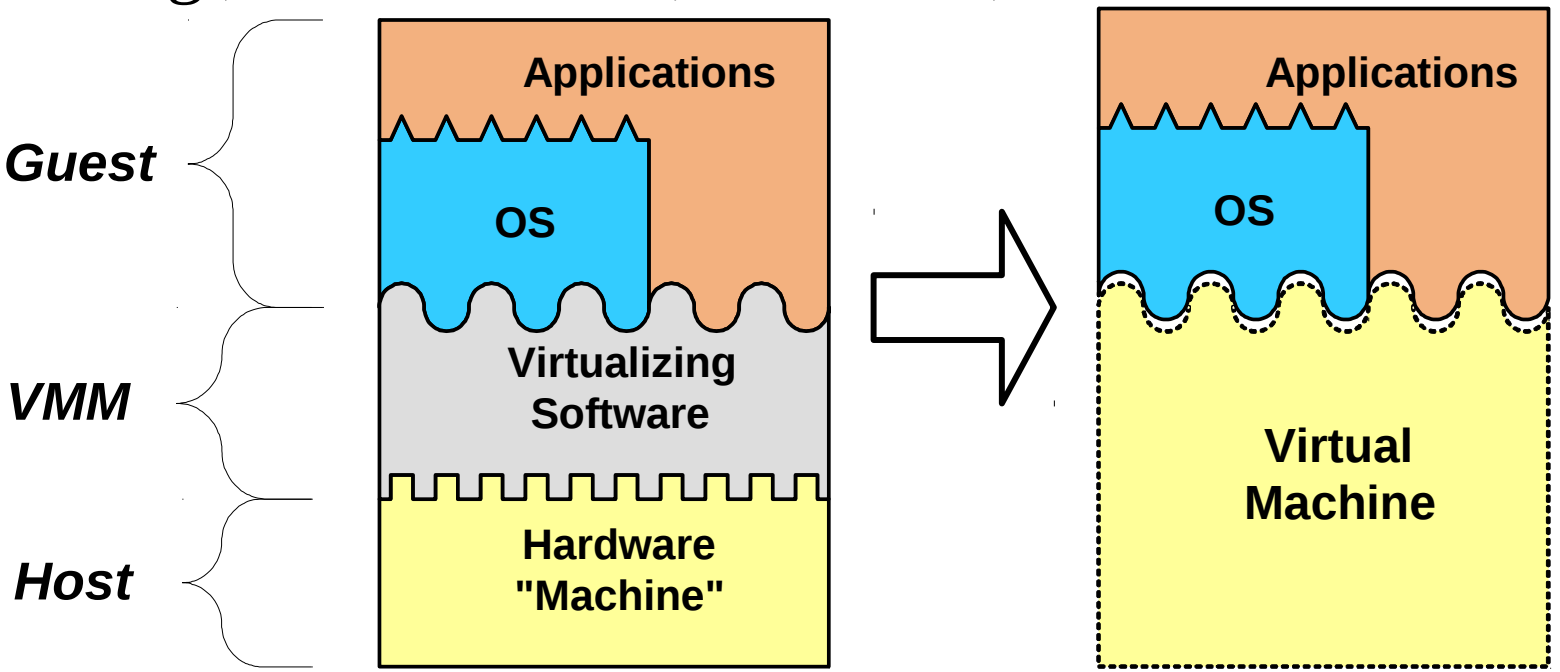
Process Virtual Machines (2)

- Constructed at ABI level
- *Runtime* manages guest process
- Runtime communicates with host OS
- Guest processes may intermingle with host processes
- As a practical matter, binaries built for same OS
- Dynamic optimizers are a special case
- Examples: IA-32 EL, FX!32, Dynamo



System Virtual Machines

- *System Virtual Machine* capable of supporting an OS with potentially many user processes
  - couple at *ISA* level
  - eg., IBM VM/360, VMWare, Transmeta Crusoe



PVM – Multiprogramming

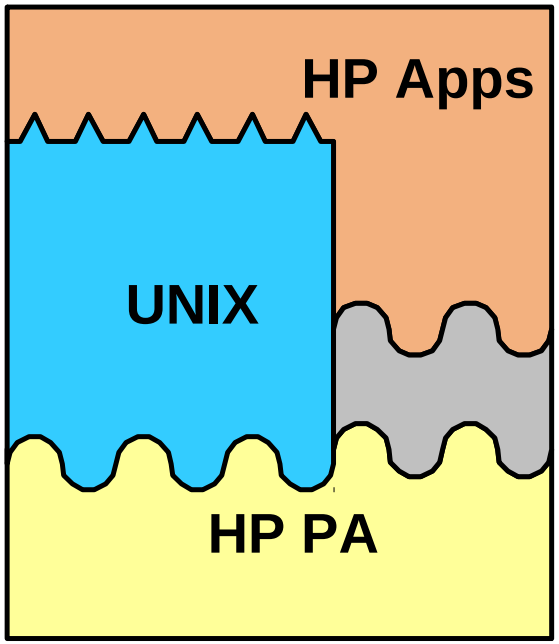
- PVM provided by a multi-process OS for each concurrently executing application.
- Combination of the OS system call interface, and the user-level ISA.
- Each process is given the illusion of having the complete machine to itself.

PVM – Emulators

- Execute binaries compiled to a different instruction set than that executed by the host's hardware.
- Interpretation
  - low startup overhead
  - high steady-state per instruction emulation overhead

PVM – Dynamic Translators

- Run-time translation of blocks of source instructions to equivalent target instructions.
  - high start-up translation overhead
  - fast steady-state execution
- Uses a *code cache* to store translated blocks of code for reuse.
- e.g., Digital's FX!32 system, Aries system, Intel IA-32 EL system

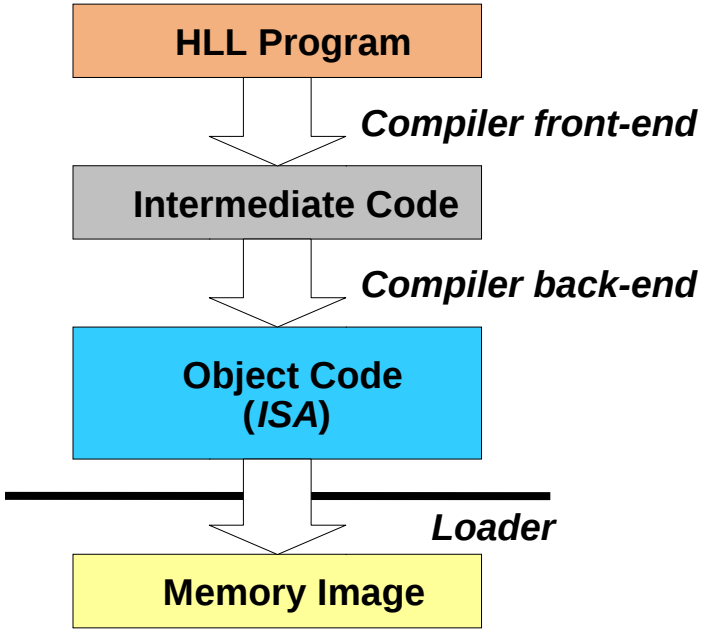


PVM – Same ISA Binary Optimizers

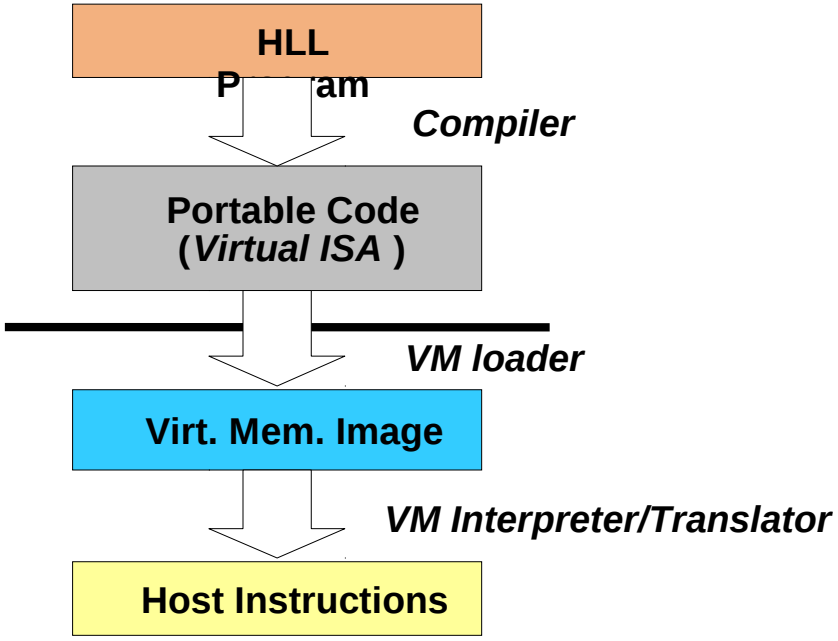
- Same source and target ISAs.
- Main task is the optimization of the source binary
  - ABI level optimization
  - may also collect performance profiles
  - may also enhance security
- e.g., Dynamo system, developed at Hewlett-Packard.

PVM – High Level Language VM

- A HLL is designed for VM execution
  - minimize hardware-specific and OS-specific features that could compromise portability



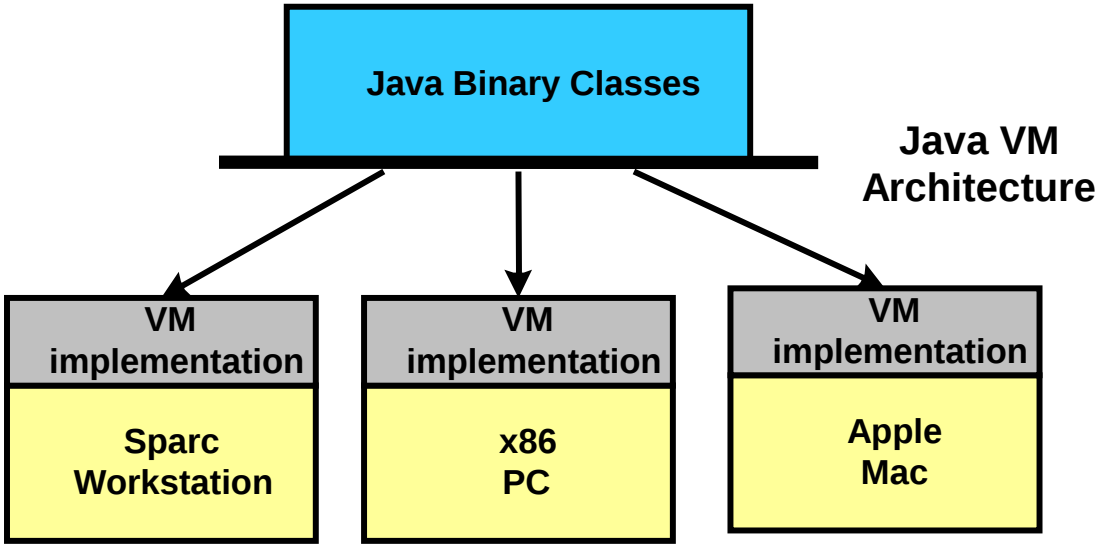
Traditional



HLL VM

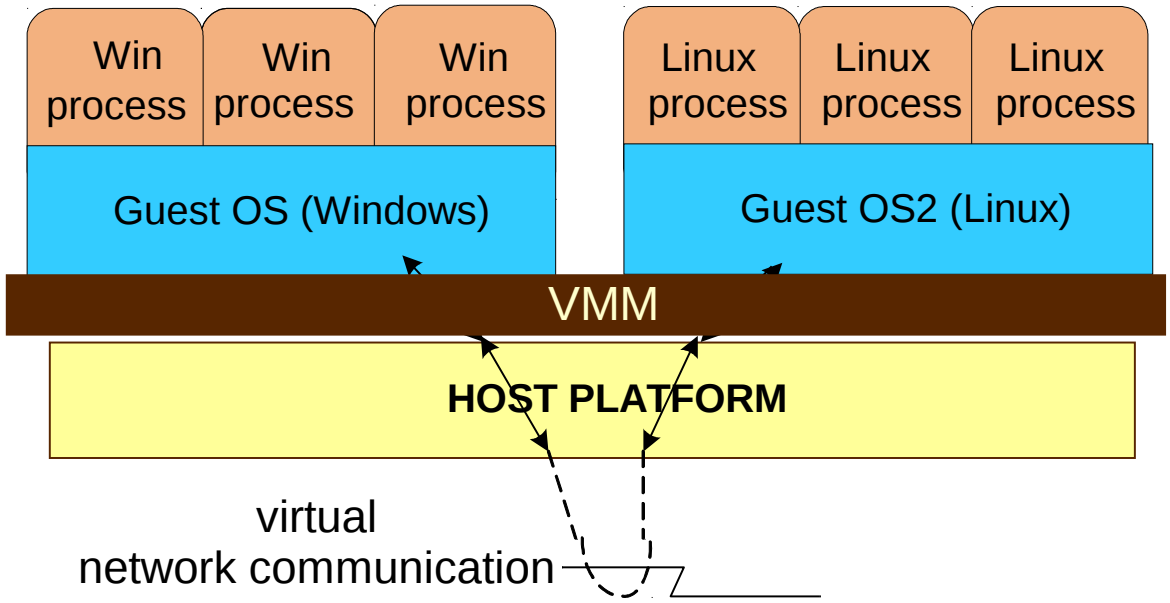
PVM – High Level Language VM

- Binary class files are distributed
  - ISA part of class file (no real implementation)
- OS interaction via API
- e.g., Java, Microsoft CLI



Classic System Virtual Machine

- Original meaning of the term *virtual machine*
  - all guest and host software use the same ISA
  - VMM runs on bare hardware (most privileged mode)
  - VMM intercepts and implements all the privileged operations for the guest OS.



Hosted System Virtual Machine

- Virtualizing software is built on top of an existing host OS.
- Advantages
  - installation is like installing application programs
  - host OS provides device driver support
- Drawbacks
  - less efficient

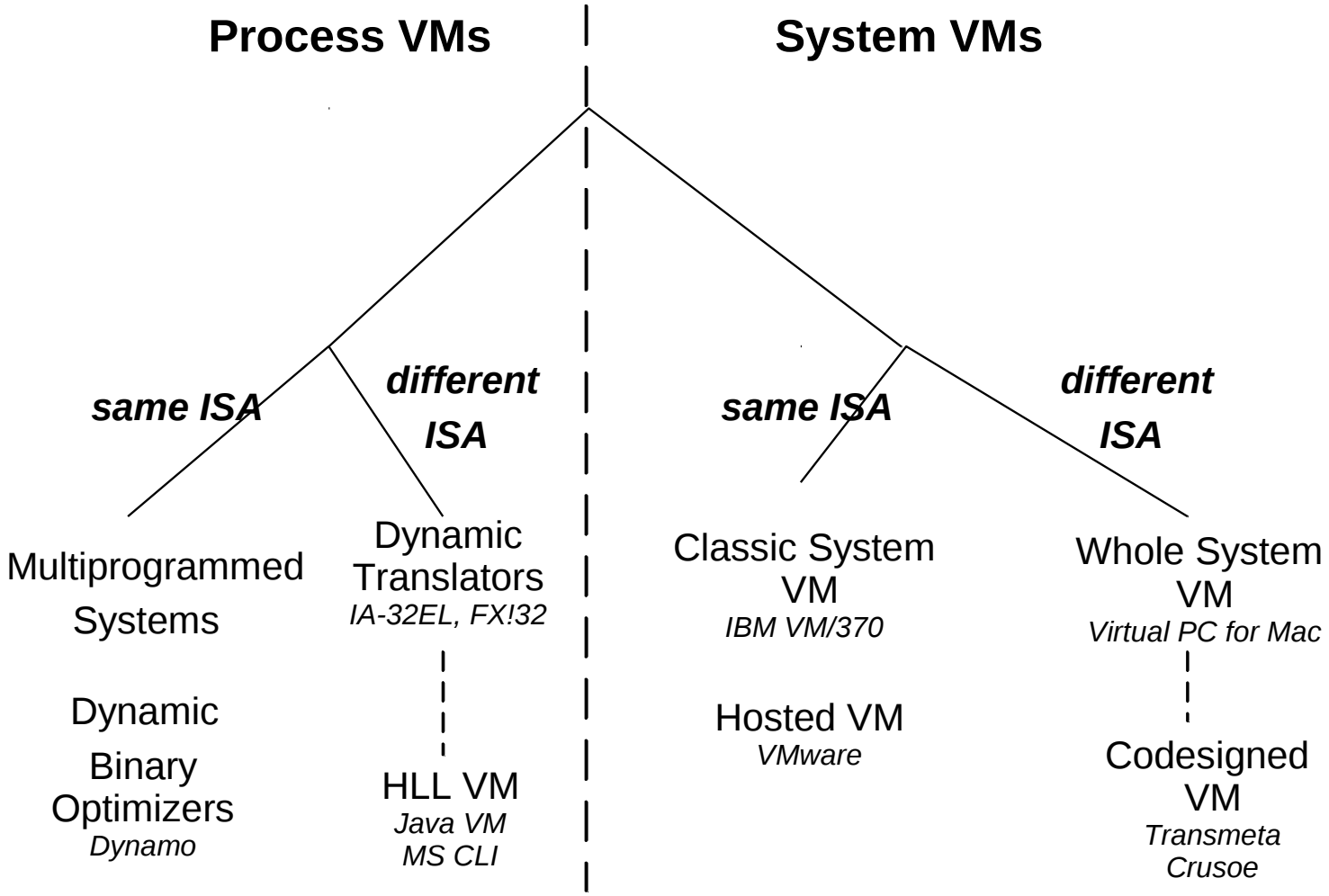
Whole System VMs

- Different ISA for guest and host systems.
  - both application and OS code require emulation
- Implemented by placing the VMM and the guest software on top of a conventional host OS running on the hardware
- e.g., Virtual PC

Codesigned Virtual Machines

- VMs designed to enable innovative ISAs and/or hardware implementations for improved performance, power efficiency, etc.
- Similar hardware virtualization is common for microprocessors, such as Pentium IV.
- Software VM is part of the hardware design
  - applications/OS never directly execute native ISA instructions
- e.g., Transmeta Crusoe processor

VM Taxonomy


Process VMs
System VMs
same ISA
different ISA
same ISA
different ISA
Multiprogrammed Systems
Dynamic Binary Optimizers
Dynamo
Dynamic Translators
IA-32EL, FX!32
HLL VM
Java VM
MS CLI
Classic System VM
IBM VM/370
Hosted VM
VMware
Whole System VM
Virtual PC for Mac
Codesigned VM
Transmeta Crusoe