Non-examinable Learning Outcomes ☺

- An appreciation that the abstract interface to the system can be at different levels.
  - Virtual machine monitors (VMMs) provide a low-level interface
- An understanding of trap and emulate
- Knowledge of the difference between type 1 (native) and type 2 VMMs (hosted)

1

Virtual Machines

References:

Smith, J.E.; Ravi Nair; , "The architecture of virtual machines," *Computer* , vol.38, no.5, pp. 32- 38, May 2005

Chapter 7 – 7.3 Textbook “Modern Operating Systems”, 4th ed.

All of chapter 7, if you’re interested.

THE UNIVERSITY OF NEW SOUTH WALES

2

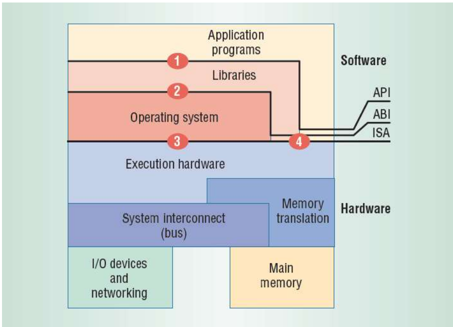
Observations

- Operating systems provide well defined interfaces
  - Abstract hardware details
    - Simplify
    - Enable portability across hardware differences
- Hardware instruction set architectures are another will defined interface
  - Example AMD and Intel both implement (mostly) the same ISA
  - Same software can run on both

THE UNIVERSITY OF NEW SOUTH WALES

3

Interface Levels

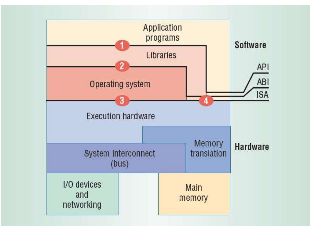


THE UNIVERSITY OF NEW SOUTH WALES

4

Instruction Set Architecture

- Interface between software and hardware
  - label 3 + 4
- Divided between privileged and un-privileged parts
  - Privileged a superset of the un-privileged

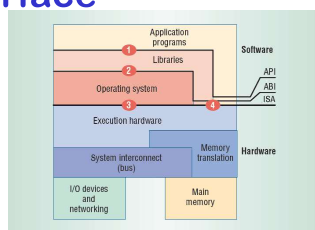


THE UNIVERSITY OF NEW SOUTH WALES

5

Application Binary Interface

- Interface between programs ↔ hardware + OS
  - Label 2+4
- Consists of system call interface + un-privileged ISA

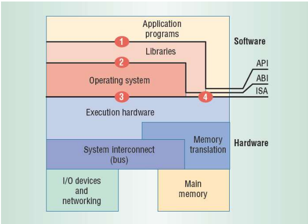


THE UNIVERSITY OF NEW SOUTH WALES

6

Application Programming Interface

- Interface between high-level language ↔ libraries + hardware + OS
- Consists of library calls + un-privileged ISA
  - Syscalls usually called through library.
- Portable via re-compilation to other systems supporting API
  - or dynamic linking



7

Some Interface Goals

- Support deploying software across all computing platforms.
  - E.g. software distribution across the Internet
- Provide a platform to securely share hardware resources.
  - E.g. cloud computing

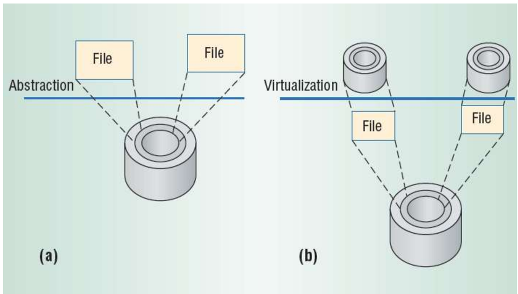
8

OS is an extended virtual machine

- Multiplexes the "machine" between applications
  - Time sharing, multitasking, batching
- Provided a higher-level machine for
  - Ease of use
  - Portability
  - Efficiency
  - Security
  - Etc….

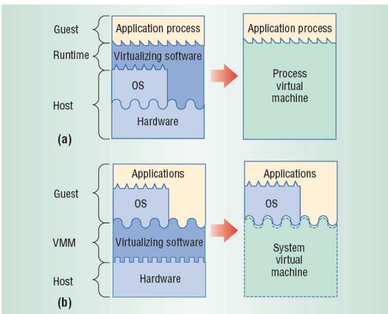
9

Abstraction versus Virtualisation



10

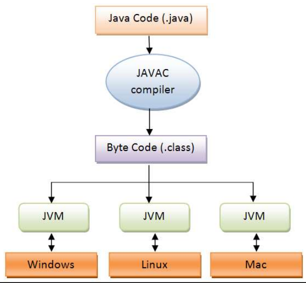
Process versus *System* Virtual Machine



11

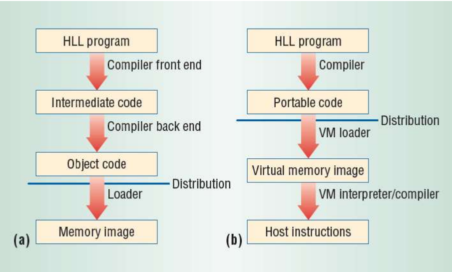
JAVA – Higher-level Virtual Machine

- write a program once, and run it anywhere
  - Architecture independent
  - Operating System independent
- Language itself was clean, robust, garbage collection
- Program compiled into bytecode
  - Interpreted or just-in-time compiled.
  - Lower than native performance



12

Comparing Conventional code execution versus Emulation/Translation



THE UNIVERSITY OF NEW SOUTH WALES

13

Aside: Just In-Time compilation (JIT)

THE UNIVERSITY OF NEW SOUTH WALES

14

JAVA and the Interface Goals

- Support deploying software across all computing platforms. ✔
- Provide a platform to securely share hardware resources. ✘

THE UNIVERSITY OF NEW SOUTH WALES

15

Issues

- Legacy applications
- No isolation nor resource management between applets
- Security
  - Trust JVM implementation? Trust underlying OS?
- Performance compared to native?

THE UNIVERSITY OF NEW SOUTH WALES

16

Is the OS the “right” level of extended machine?

- Security
  - Trust the underlying OS?
- Legacy application and OSs
- Resource management of existing systems suitable for all applications?
  - Performance isolation?
- What about activities requiring “root” privileges

THE UNIVERSITY OF NEW SOUTH WALES

17

Virtual Machine Monitors

Also termed a *hypervisor*

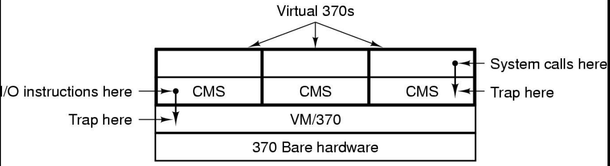
- Provide scheduling and resource management
- Extended “machine” is the actual machine interface.

THE UNIVERSITY OF NEW SOUTH WALES

18

IBM VM/370

- CMS a light-weight, single-user OS
- VM/370 multiplex multiple copies of CMS



THE UNIVERSITY OF NEW SOUTH WALES

19

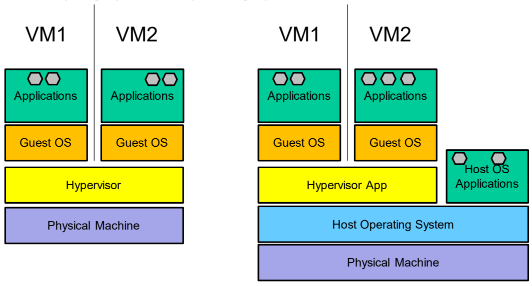
Advantages

- Legacy OSes (and applications)
- Legacy hardware
- Server consolidation
  - Cost saving
  - Power saving
- Server migration
- Concurrent OSes
  - Linux – Windows
  - Primary – Backup
    - High availability
- Test and Development
- Security
  - VMM (hopefully) small and correct
- Performance near bare hardware
  - For some applications

THE UNIVERSITY OF NEW SOUTH WALES

20

Native (Type 1) vs. Hosted (Type 2) Hypervisor

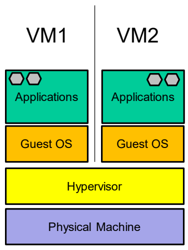


THE UNIVERSITY OF NEW SOUTH WALES

21

Type 1 (Native) Hypervisor

- Hypervisor (VMM) runs in most privileged mode of processor
  - Manage hardware directly
  - Also termed classic…, bare-metal…, native…
- Guest OS runs in non-privileged mode
  - Hypervisor implements a virtual kernel-mode/virtual user-mode
  - Or, CPU provides three privilege levels (e.g. Intel VT-x)
- What happens when guest OS executes native privileged instructions?

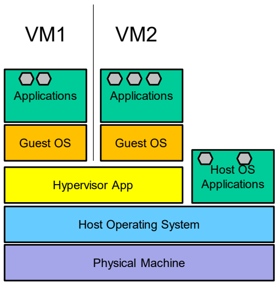


THE UNIVERSITY OF NEW SOUTH WALES

22

Type 2 (Hosted) Hypervisor

- Hypervisor runs as user-mode process above the privileged host OS
  - Also termed hosted hypervisor
- Again, provides a virtual kernel-mode and virtual user-mode
- Can leverage device support of existing host OS.
- What happens when guest OS execute privileged instructions?

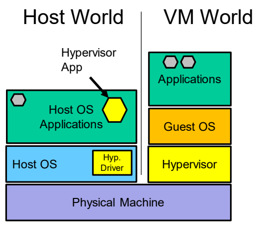


THE UNIVERSITY OF NEW SOUTH WALES

23

Hosted Hypervisor Details

- Jeremy Sugerman, Ganesh Venkitachalam and Beng-Hong Lim, "Virtualizing I/O Devices on VMware Workstation's Hosted Virtual Machine Monitor", USENIX ATC 2001
- Hypervisor application installs driver (part of the hypervisor) into the Host OS
- Driver intercepts hypervisor related activities from Hyp. App.
- It "world switches" when guest OS needs to runs
  - Unloads Host OS state from processor
  - Loads hypervisor state and gives it control of machine
- Hypervisor "world switches" when Host OS is needed
  - Regularly to allow interactivity with Host OS.
  - When hypervisor needs Host OS service (e.g. file system)



THE UNIVERSITY OF NEW SOUTH WALES

24

Gerald J. Popek and Robert P. Goldberg (1974). "Formal Requirements for Virtualizable Third Generation Architectures". Communications of the ACM 17 (7): 412 –421.

- Sensitive Instructions
  - The instructions that attempt to change the configuration of the processor.
  - The instructions whose behaviour or result depends on the configuration of the processor.
- Privileged Instructions
  - Instructions that trap if the processor is in user mode and do not trap if it is in system mode.
- Theorem
  - Architecture is virtualisable if sensitive instructions are a subset of privileged instructions.

THE UNIVERSITY OF NEW SOUTH WALES

25

Example: mtc0/mfc0 MIPS

- mfc0: load a value in the system coprocessor
  - Can be used to observer processor configuration
- mtc0: store a value in the system coprocessor
  - Can be used to change processor configuration
- Example: disable interrupts

```
mfc0 r1, C0_Status
andi r1, r1, CST_IEc
mtc0 r1, C0_Status
```

- Sensitive?
- Privileged?

THE UNIVERSITY OF NEW SOUTH WALES

26

Approach: Trap & Emulate?

THE UNIVERSITY OF NEW SOUTH WALES

27

Example: cli/sti x86

- CLI: clear interrupt flag
  - Disable interrupts
- STI: set interrupt flags
  - Enable interrupts
- Sensitive?
- Privileged?

THE UNIVERSITY OF NEW SOUTH WALES

28

X86 POPF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	ID	VIP	VIF	AC	VM	RF	0	NT	IOPL		OF	DF	IF	TF	SF	ZF	0	AF	0	PF	1	CF

- Pop top of stack and store in EFLAGS register
  - IF bit disables interrupts

THE UNIVERSITY OF NEW SOUTH WALES

29

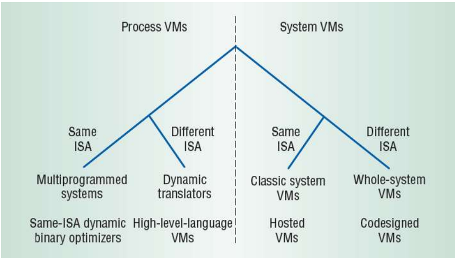
X86 POPF

- Is not privileged (does not trap)
  - In kernel mode – enable/disables interrupts
  - In user-mode – silently ignored
- POPF is not virtualisable
- X86 (pre VT extensions) is not virtualisable

THE UNIVERSITY OF NEW SOUTH WALES

30

Taxonomy of Virtual Machines



31

What is System/161?

32