


CSE


Extended OS



CSE

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 and type 2 VMMs
- An appreciation of some of the issues in virtualising the R3000



CSE

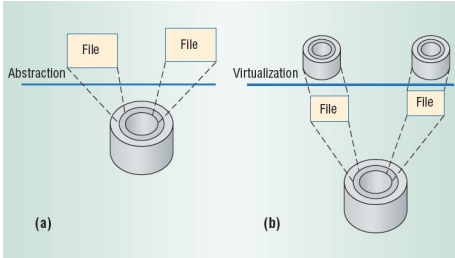

Virtual Machines

References:
 Smith, J.E.; Ravi Nair; , "The architecture of virtual machines,"
Computer , vol.38, no.5, pp. 32- 38, May 2005
 Chapter 8.3 Textbook "Modern Operating Systems"



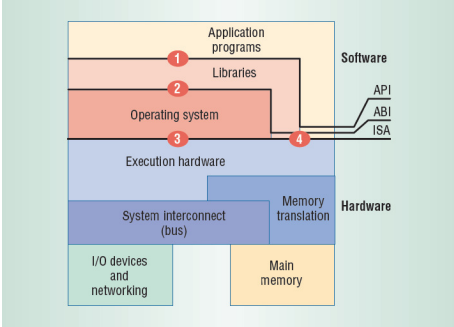

CSE

Abstraction & Virtualisation

CSE

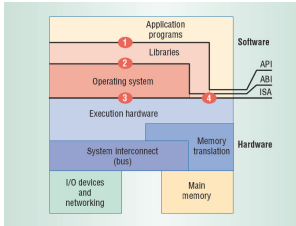

Interface Levels

CSE

Instruction Set Architecture

- Interface between software and hardware
- Divided between privileged and un-privileged parts

Application Binary Interface

- Interface between programs hardware + OS
- Consists of system call interface + un-privileged ISA

The diagram shows a layered architecture. At the top is 'Software' containing 'Application programs' and 'Libraries'. Below this is the 'Operating system'. The 'Execution hardware' layer sits above the 'Hardware' layer. The hardware layer includes 'System interconnect (bus)', 'Memory translation', 'I/O devices and networking', and 'Main memory'. Arrows indicate the flow of data and control between these layers, with numbered points 1 through 4 highlighting key interfaces.

Application Programming Interface

- Interface between programs hardware + OS
- Consists of library calls + un-privileged ISA
 - Syscalls usually called through library.

This diagram is identical to the one for the Application Binary Interface, showing the same layered architecture from software to hardware.

Process versus System Virtual Machine

The diagram compares two types of virtual machines. (a) Process virtual machine: A 'Guest' application process runs on 'Virtualizing software' (OS) which runs on 'Host' hardware. (b) System virtual machine: 'Applications' run on a 'Guest' OS, which runs on 'Virtualizing software' (OS) on 'Host' hardware. Both diagrams show the flow of data and control between the guest and host environments.

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....

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

Conventional versus Emulation/Translation

The flowchart compares two paths for executing an HLL program. (a) Conventional compilation: HLL program → Compiler front end → Intermediate code → Compiler back end → Object code → Loader → Memory Image. (b) Emulation/translation: HLL program → Compiler → Portable code → VM loader → Virtual memory image → VM interpreter/compiler → Host instructions. Arrows indicate the flow of data and control between these steps.

Aside: Just In-Time compilation (JIT)

main binary
func(C)
func(2C)

main(C) {
...
func(C)
func(2C)
}

THE UNIVERSITY OF NEW SOUTH WALES

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

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?
- What about activities requiring “root” privileges

THE UNIVERSITY OF NEW SOUTH WALES

Virtual Machine Monitors

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

THE UNIVERSITY OF NEW SOUTH WALES

IBM VM/370

Virtual 370s

I/O instructions here

Trap here

System calls here

Trap here

CMS CMS CMS

VM/370

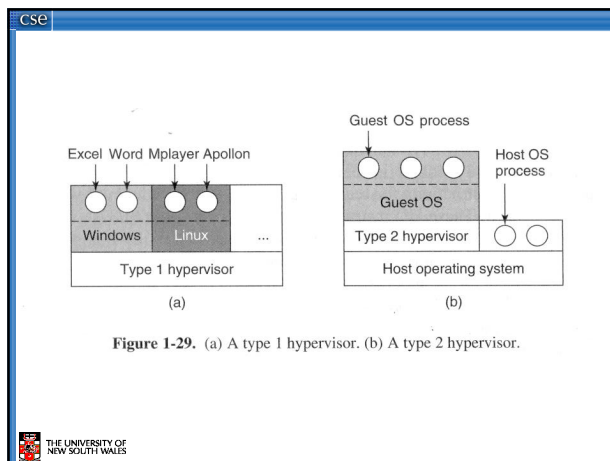
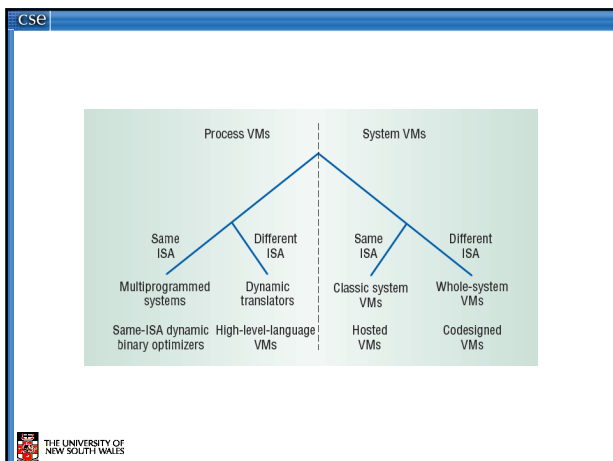
370 Bare hardware

THE UNIVERSITY OF NEW SOUTH WALES

Advantages

- Legacy OSes (and applications)
- Server consolidation
- 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

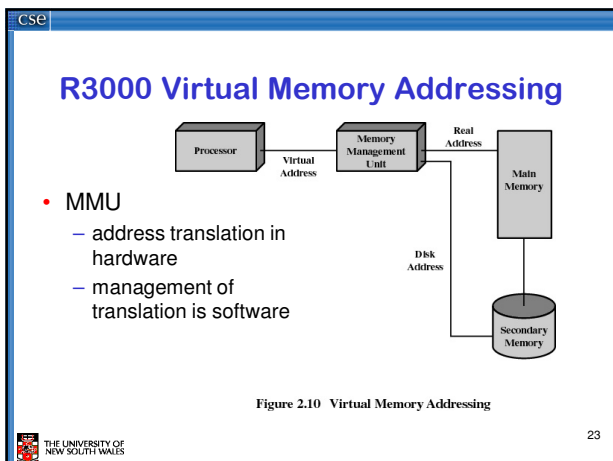


Virtual R3000???

- Interpret
 - System/161
 - slow
 - JIT dynamic compilation
- Run on the real hardware??

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.



R3000 Address Space Layout

0xFFFFFFFF	kseg2
0xC0000000	
0xA0000000	kseg1
0x80000000	kseg0
0x00000000	kuseg

- kuseg:
 - 2 gigabytes
 - MMU translated
 - Cacheable
 - user-mode and kernel mode accessible

R3000 Address Space Layout

- kseg0:
 - 512 megabytes
 - Fixed translation window to physical memory
 - 0x80000000 - 0x9ffffff virtual = 0x00000000 - 0x1ffffff physical
 - MMU not used
 - Cacheable
 - Only kernel-mode accessible
 - Usually where the kernel code is placed

R3000 Address Space Layout

- kseg1:
 - 512 megabytes
 - Fixed translation window to physical memory
 - 0xa0000000 - 0xbffffff virtual = 0x00000000 - 0x1ffffff physical
 - MMU not used
 - **NOT** cacheable
 - Only kernel-mode accessible
 - Where devices are accessed (and boot ROM)

R3000 Address Space Layout

- kseg2:
 - 1024 megabytes
 - MMU translated
 - Cacheable
 - Only kernel-mode accessible

Issues

- Privileged registers (CP0)
- Privileged instructions
- Address Spaces
- Exceptions (including syscalls, interrupts)
- Devices

Approach: Trap & Emulate?

wfc0 r1, cause C0

hardware

