

Extended OS



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



Virtual Machines

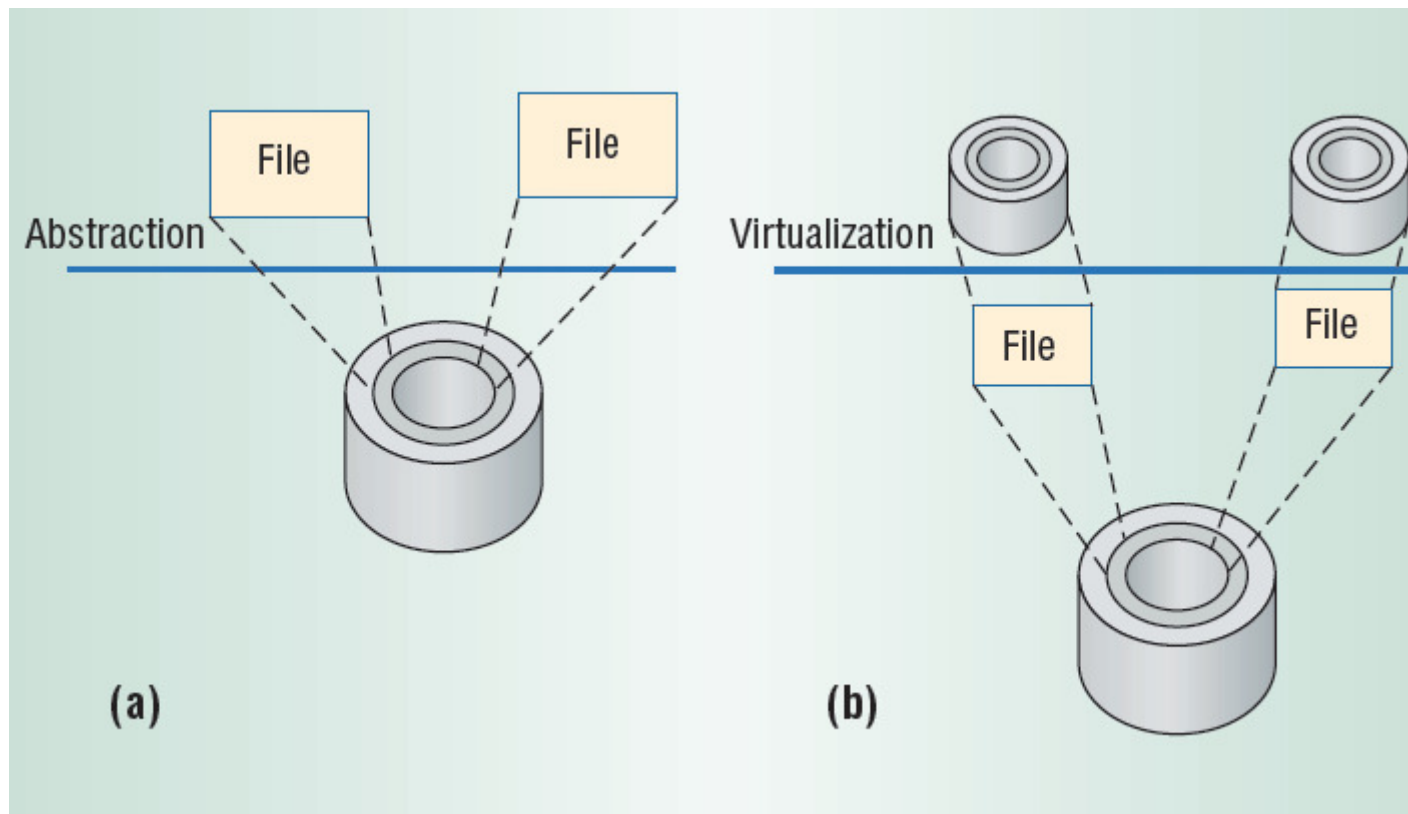
References:

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

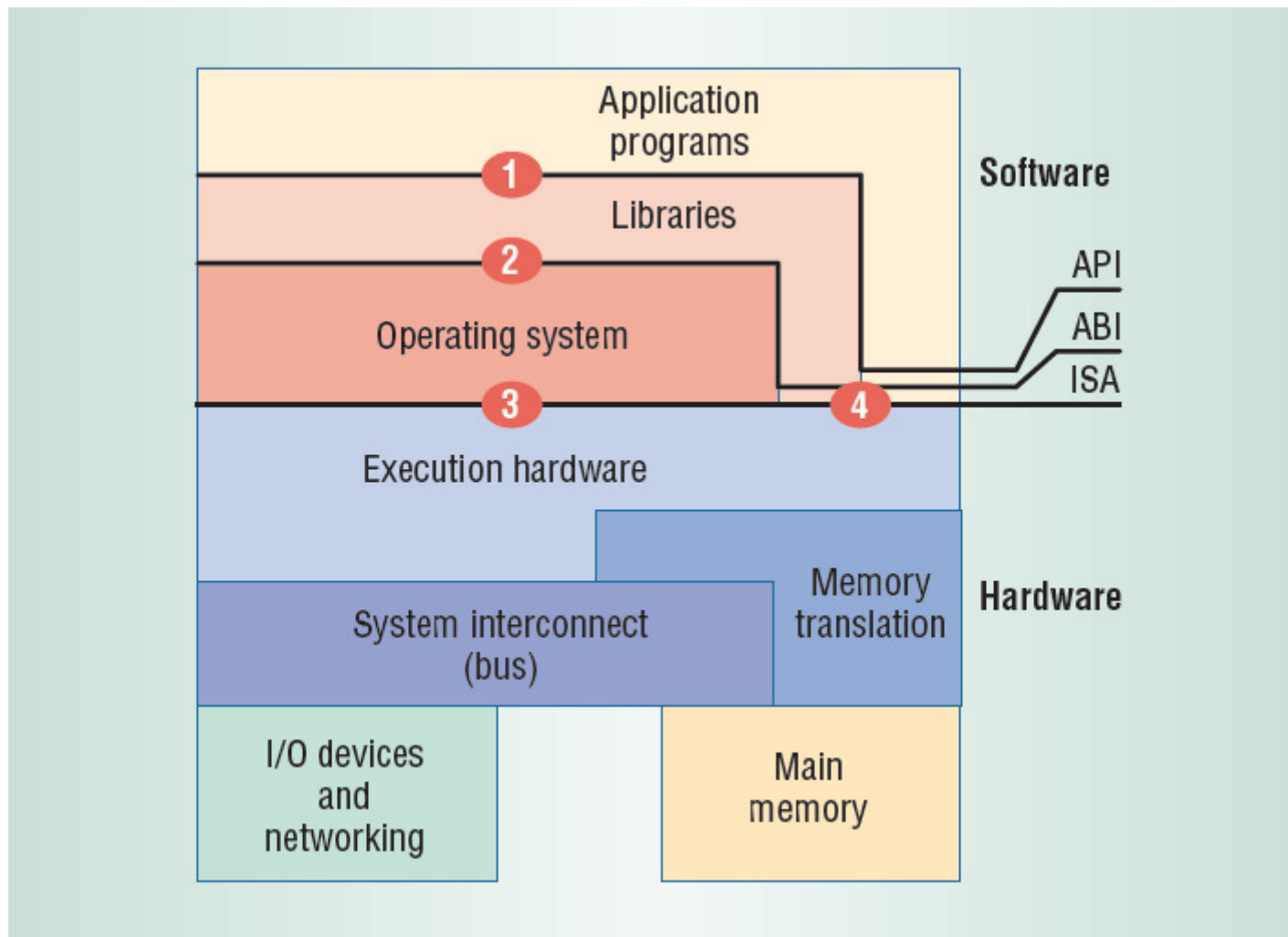
Chapter 8.3 Textbook "Modern Operating Systems"



Abstraction & Virtualisation

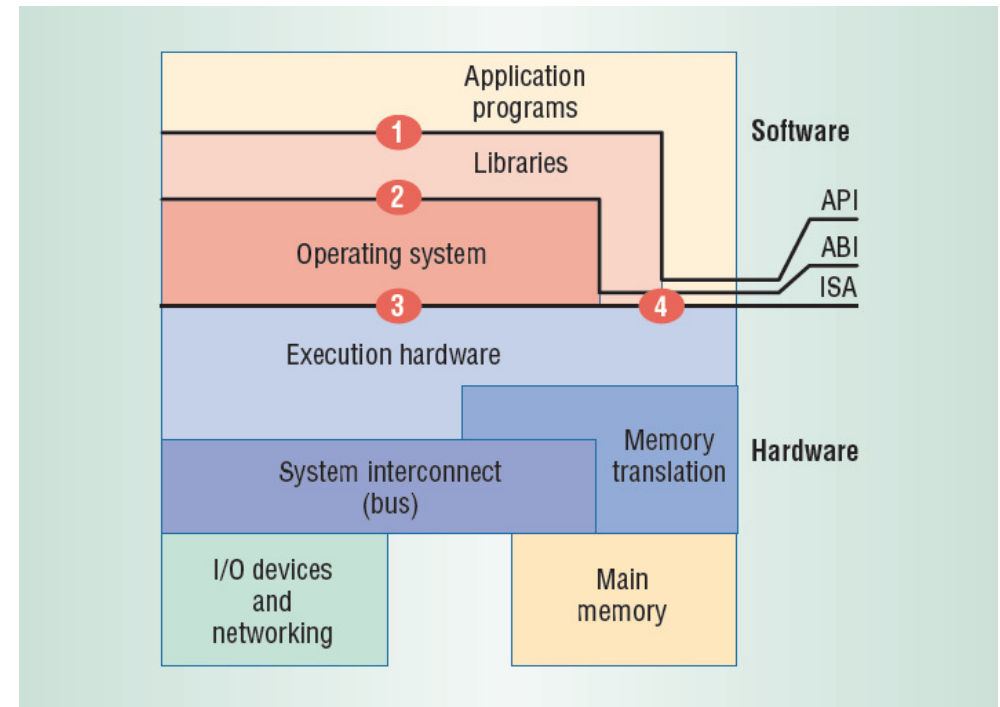


Interface Levels



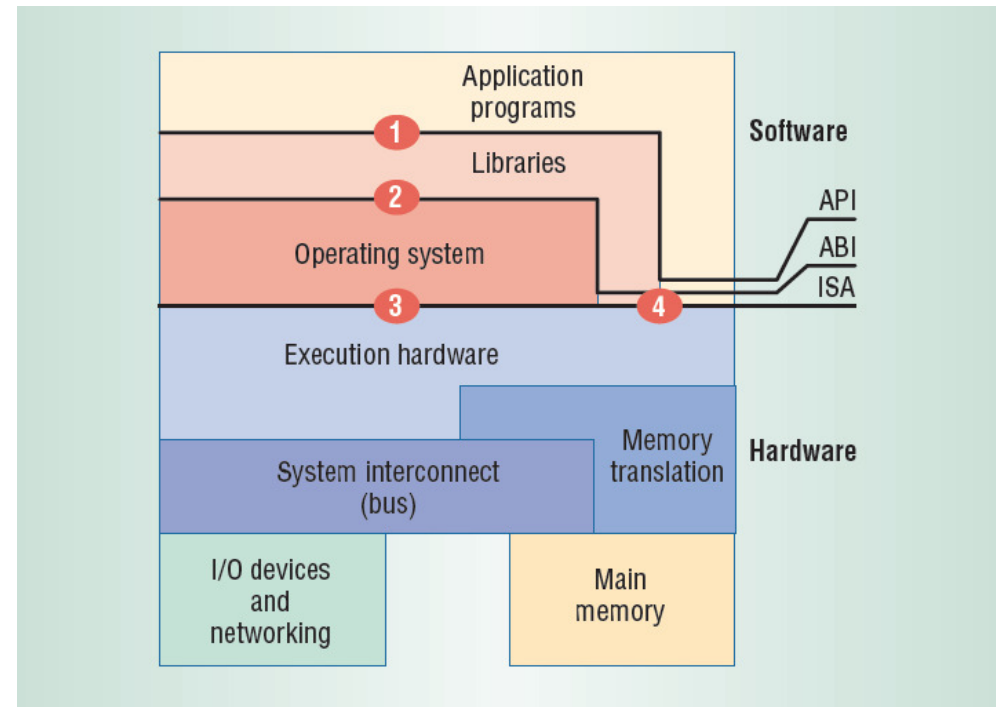
Instruction Set Architecture

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



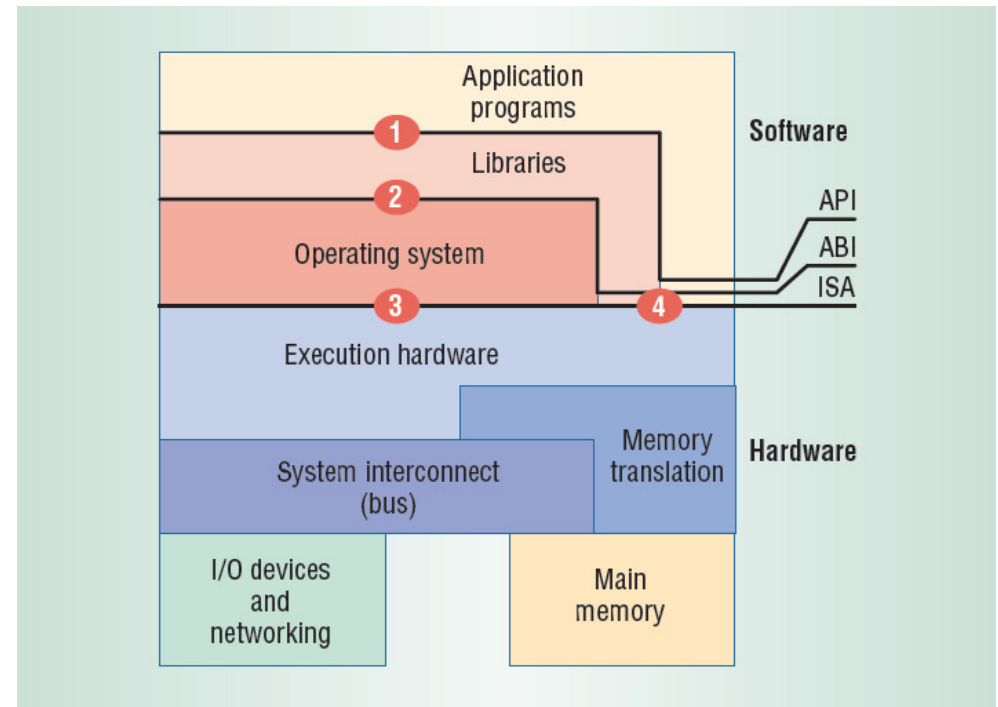
Application Binary Interface

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

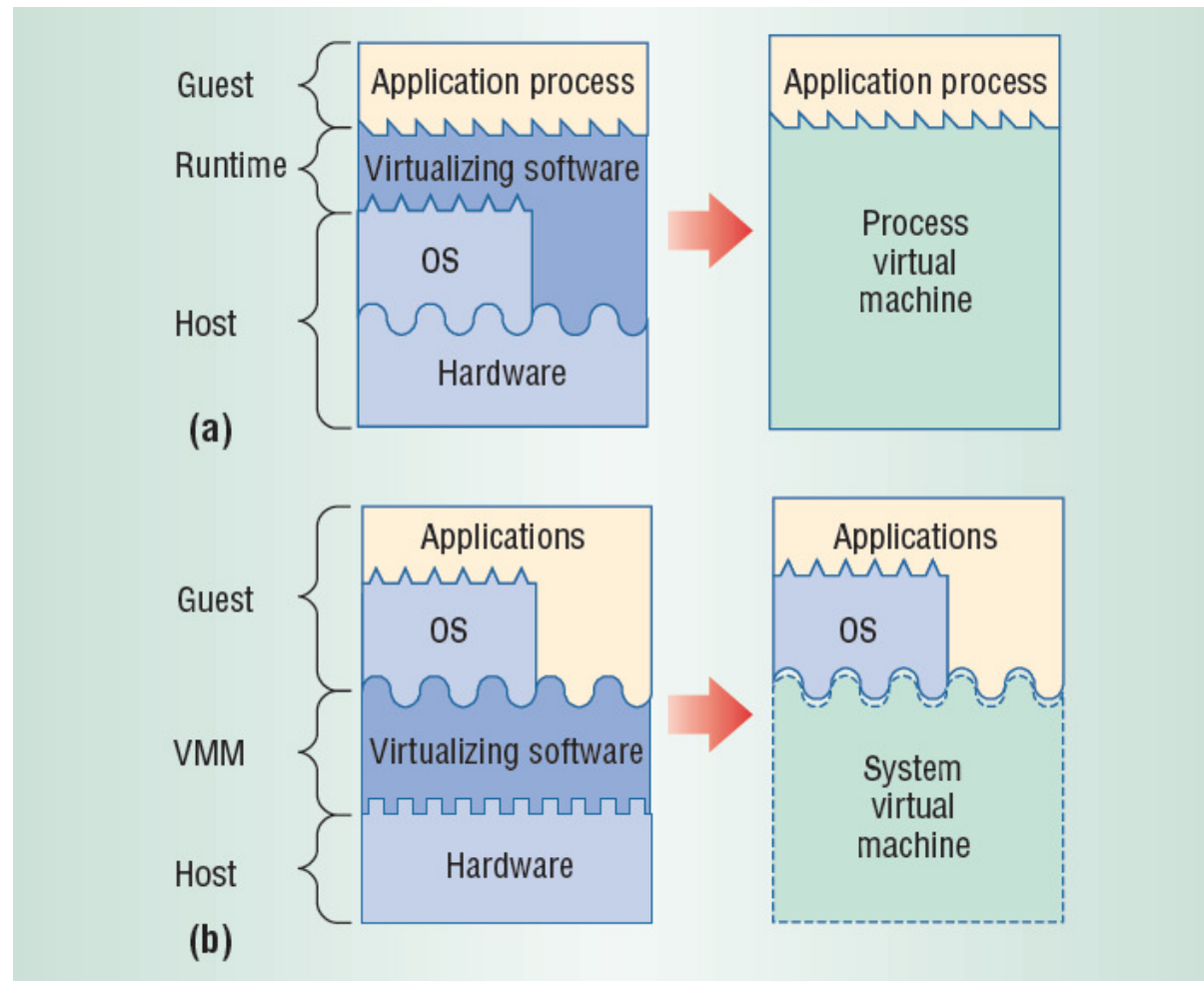


Application Programming Interface

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



Process versus System Virtual Machine



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....

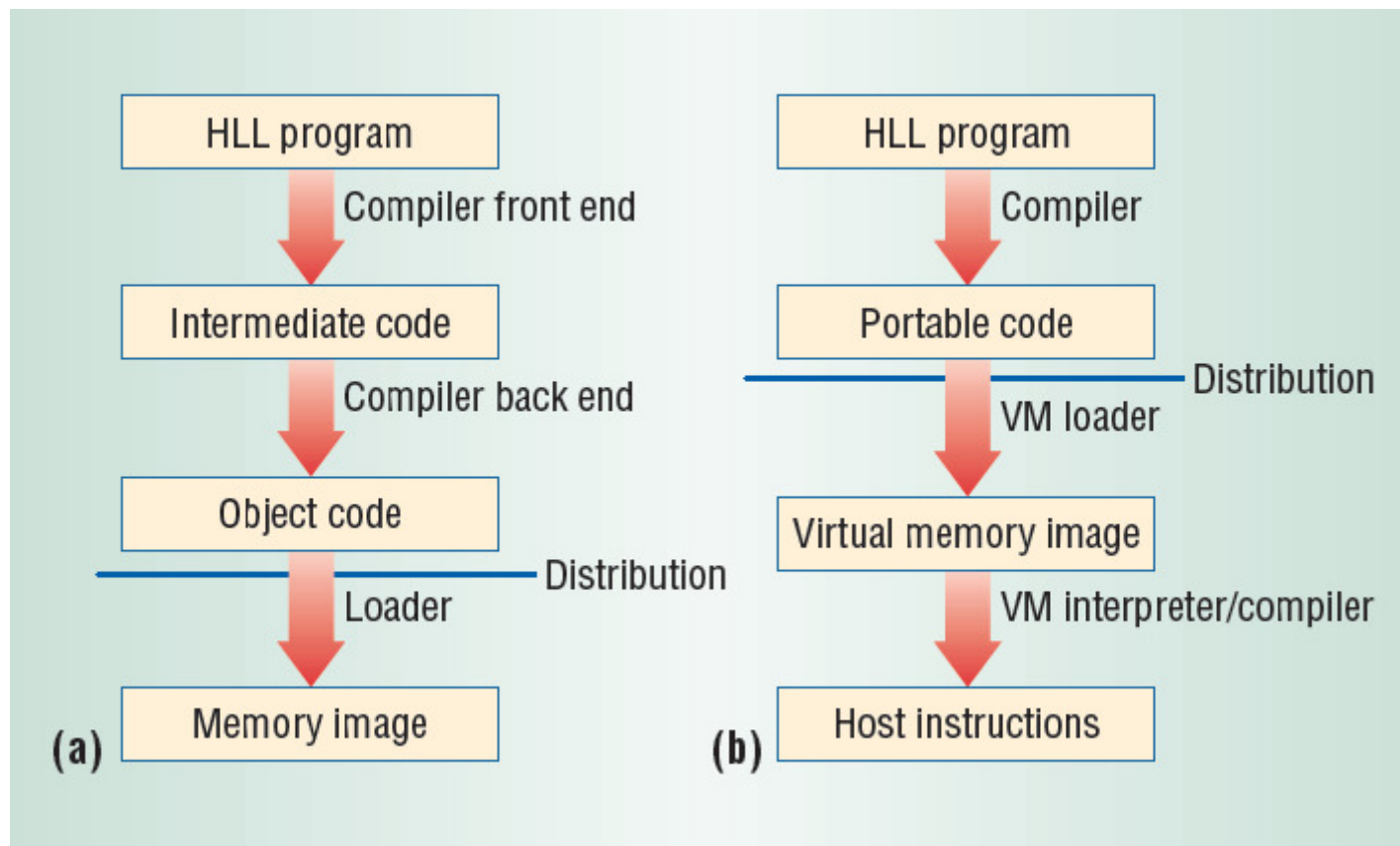


JAVA – Higher-level Virtual Machine

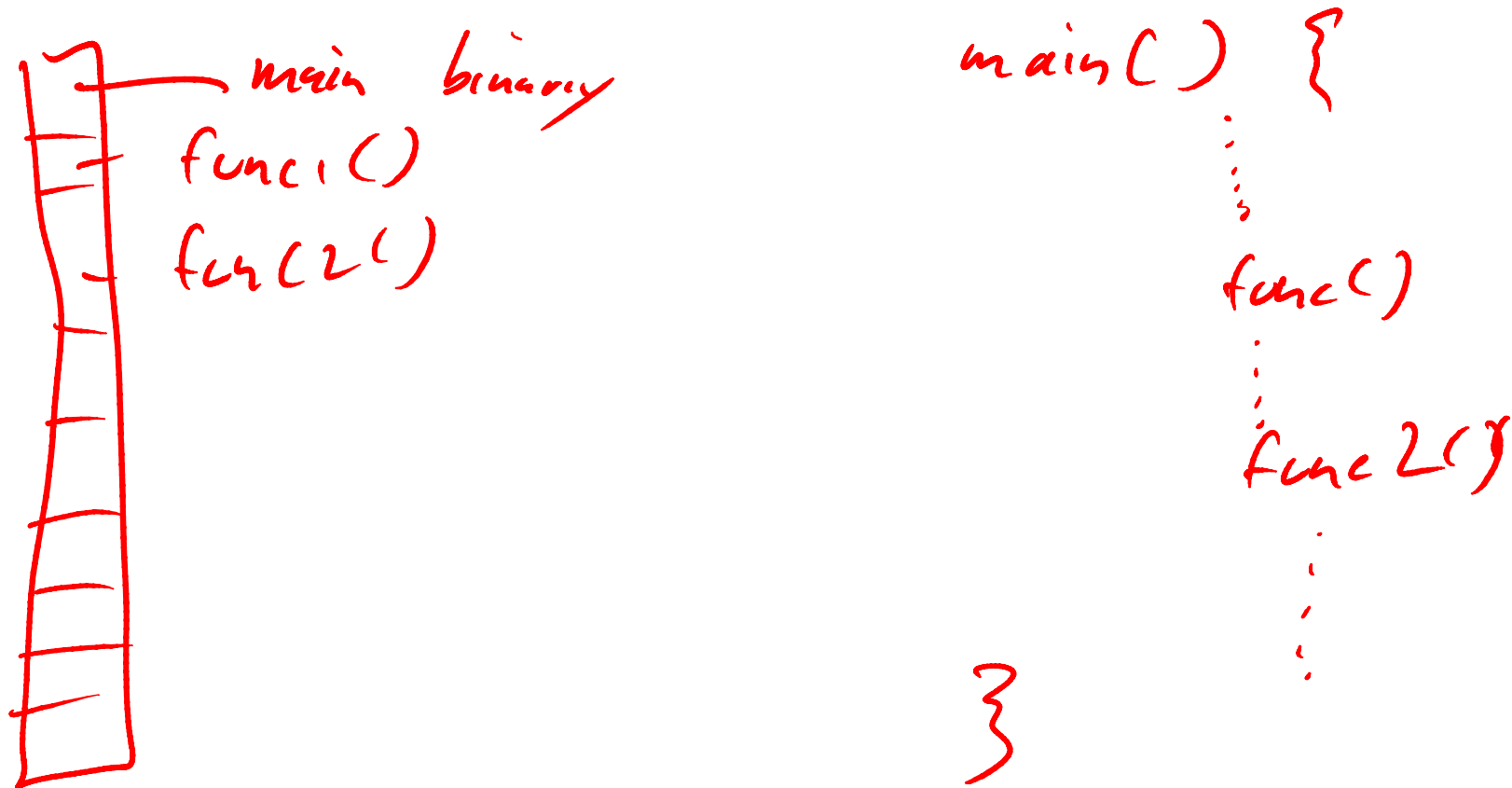
- write a program once, and run it anywhere
 - Architecture independent
 - Operating System independent
- Language itself was clean, robust, garbage collection
- Program compiled into bytecode
 - Interpreted or just-in-time compiled.
 - Lower than native performance



Conventional versus Emulation/Translation



Aside: Just In-Time compilation (JIT)



Issues

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

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

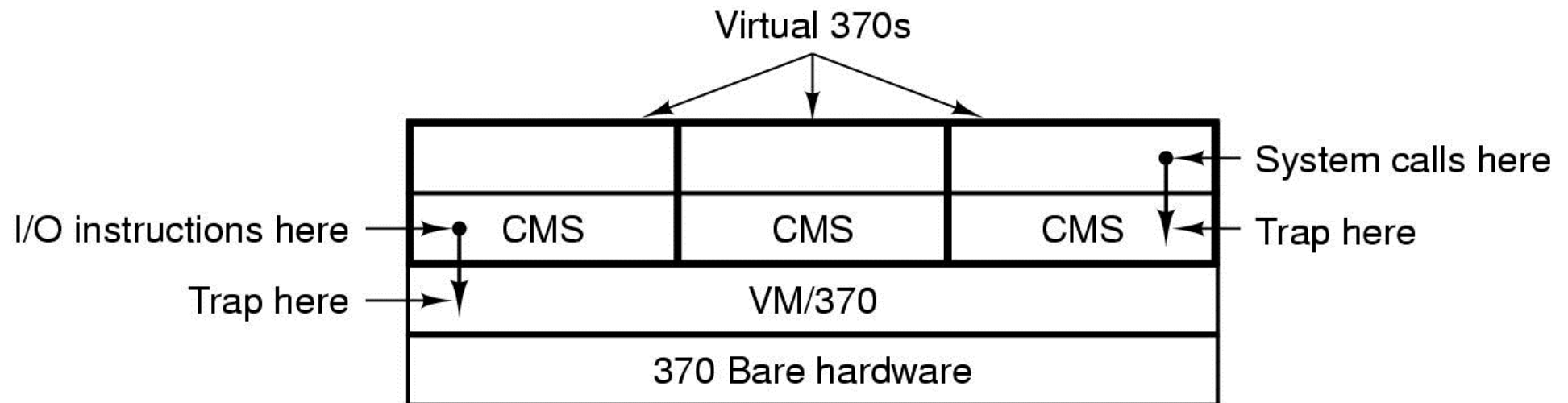


Virtual Machine Monitors

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



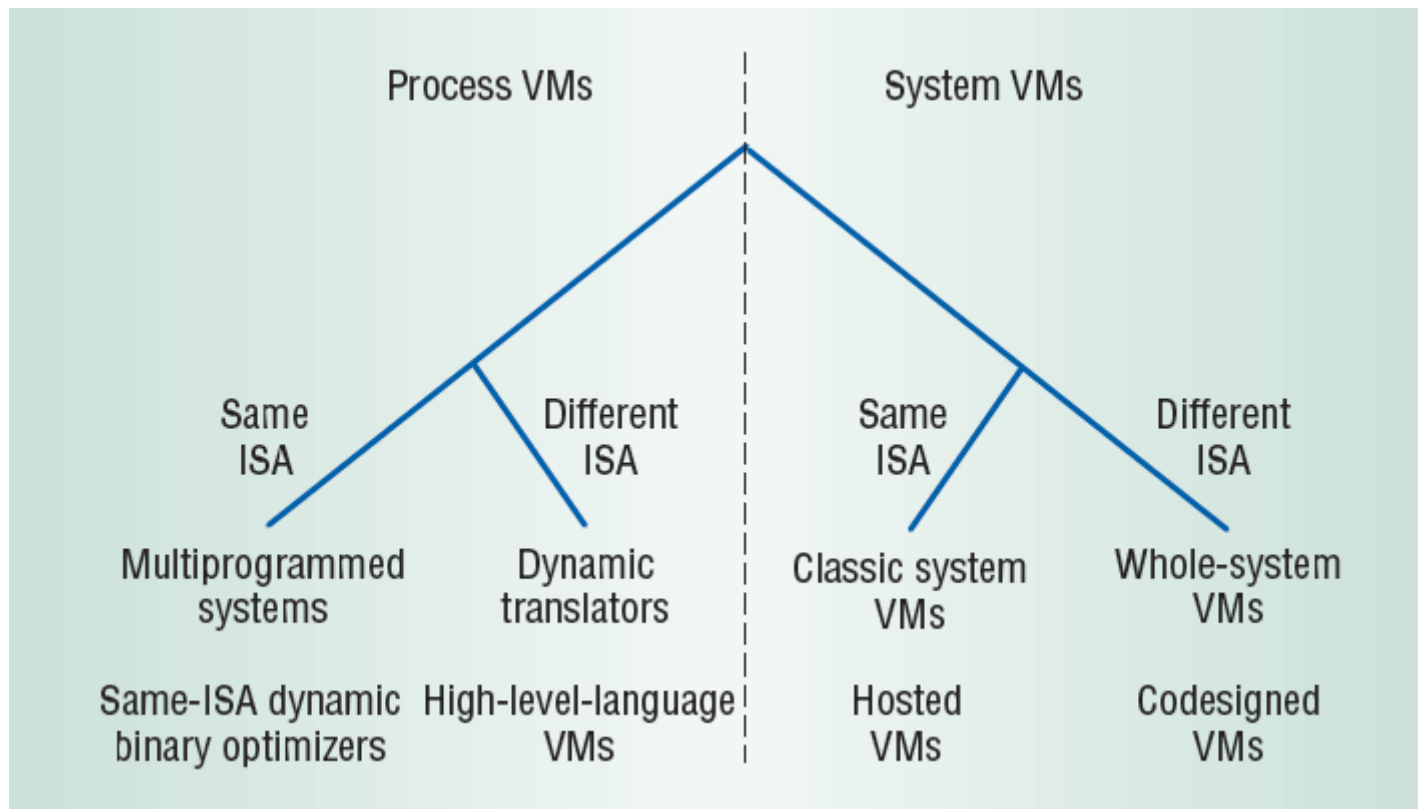
IBM VM/370



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





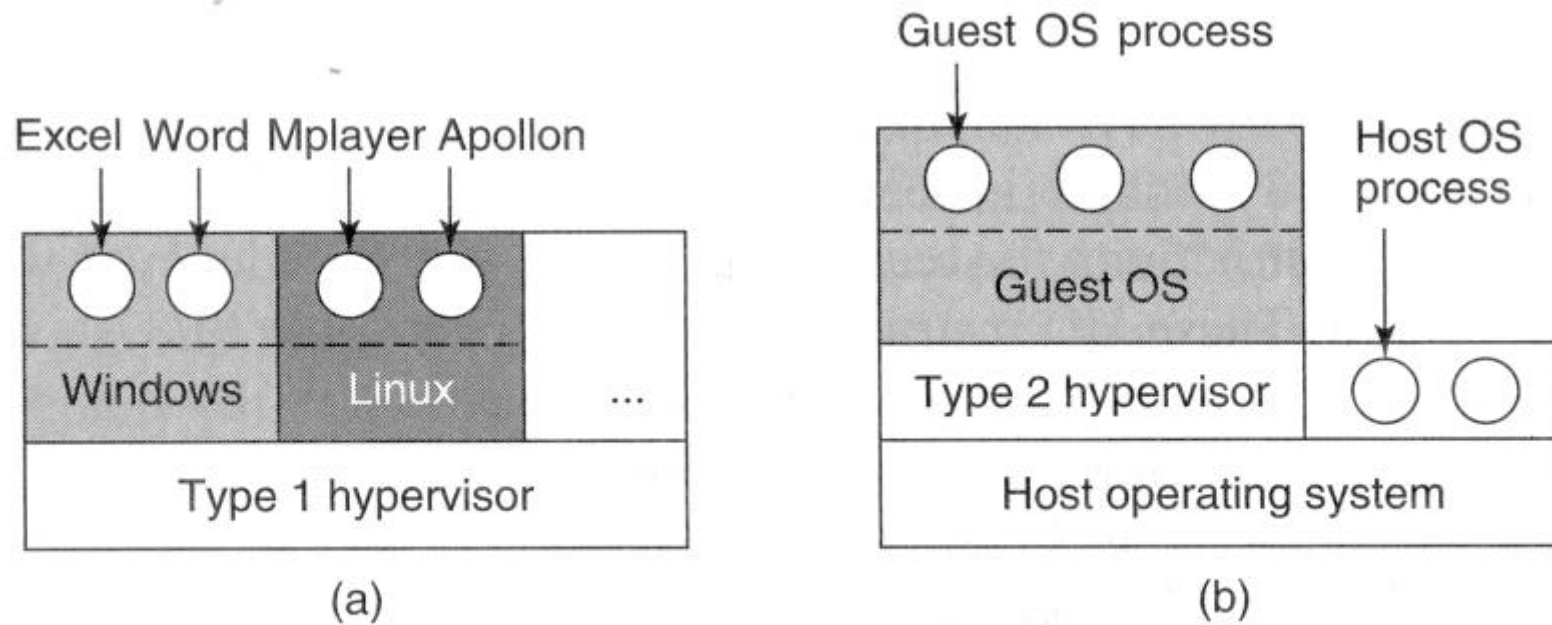


Figure 1-29. (a) A type 1 hypervisor. (b) A type 2 hypervisor.

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 Virtual Memory Addressing

- MMU
 - address translation in hardware
 - management of translation is software

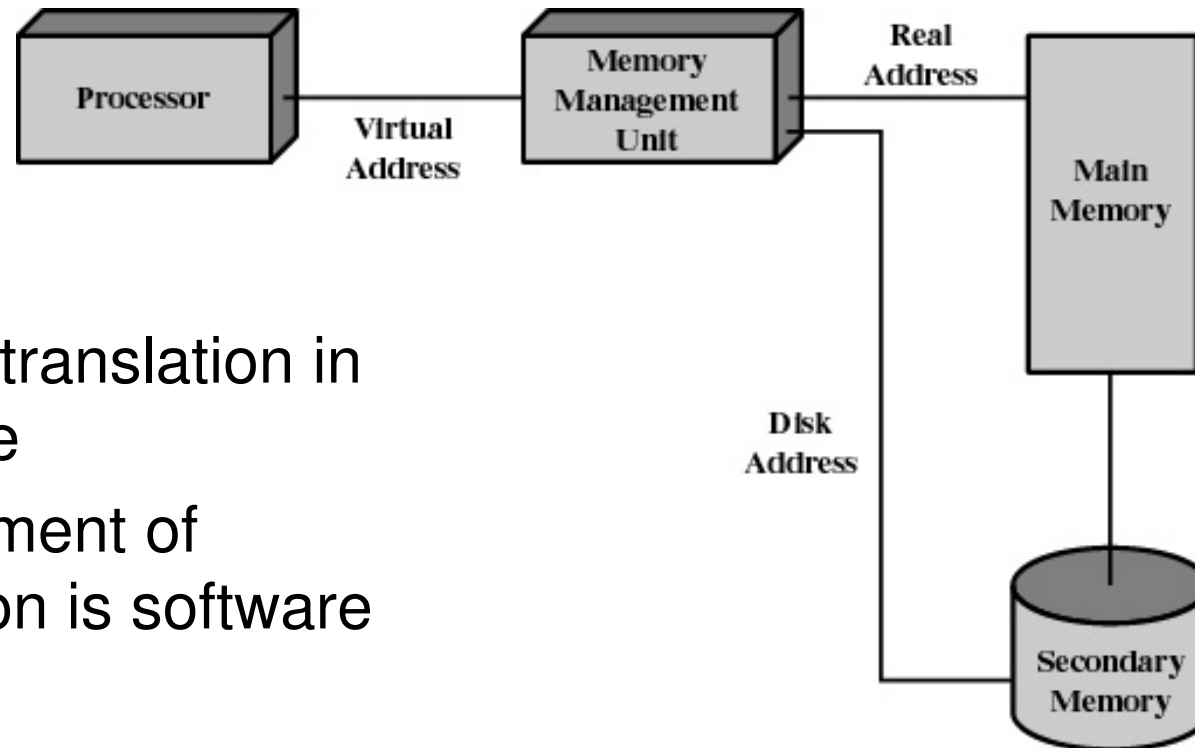


Figure 2.10 Virtual Memory Addressing

R3000 Address Space Layout

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

0xFFFFFFFF

kseg2

0xC0000000

kseg1

0xA0000000

kseg0

0x80000000

kuseg

0x00000000



R3000 Address Space Layout

- kseg0:
 - 512 megabytes
 - Fixed translation window to physical memory
 - 0x80000000 - 0x9fffffff virtual = 0x00000000 - 0x1ffffff physical
 - MMU not used
 - Cacheable
 - Only kernel-mode accessible
 - Usually where the kernel code is placed

0xffffffff

kseg2

0xc0000000

kseg1

0xa0000000

kseg0

0x80000000

kuseg

0x00000000



The diagram illustrates the R3000 address space layout. It is a vertical stack of four segments: kseg2 (purple, top), kseg1 (green), kseg0 (blue), and kuseg (white, bottom). Address markers are placed to the left of the stack: 0xffffffff at the top, 0xc0000000 at the boundary between kseg2 and kseg1, 0xa0000000 at the boundary between kseg1 and kseg0, 0x80000000 at the boundary between kseg0 and kuseg, and 0x00000000 at the bottom. A green box labeled 'Physical Memory' is positioned below the kuseg segment. A curved arrow points from the 0x80000000 address in the kseg0 segment down to the Physical Memory box, indicating a fixed translation window.

R3000 Address Space Layout

- kseg1:
 - 512 megabytes
 - Fixed translation window to physical memory
 - 0xa0000000 - 0xbfffffff virtual = 0x00000000 - 0x1fffffff physical
 - MMU not used
 - **NOT** cacheable
 - Only kernel-mode accessible
 - Where devices are accessed (and boot ROM)

0xffffffff

kseg2

0xc0000000

kseg1

0xa0000000

kseg0

0x80000000

kuseg

0x00000000

Physical Memory

R3000 Address Space Layout

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

0xffffffff

kseg2

0xc0000000

kseg1

0xa0000000

kseg0

0x80000000

virtual

kuseg

0x00000000

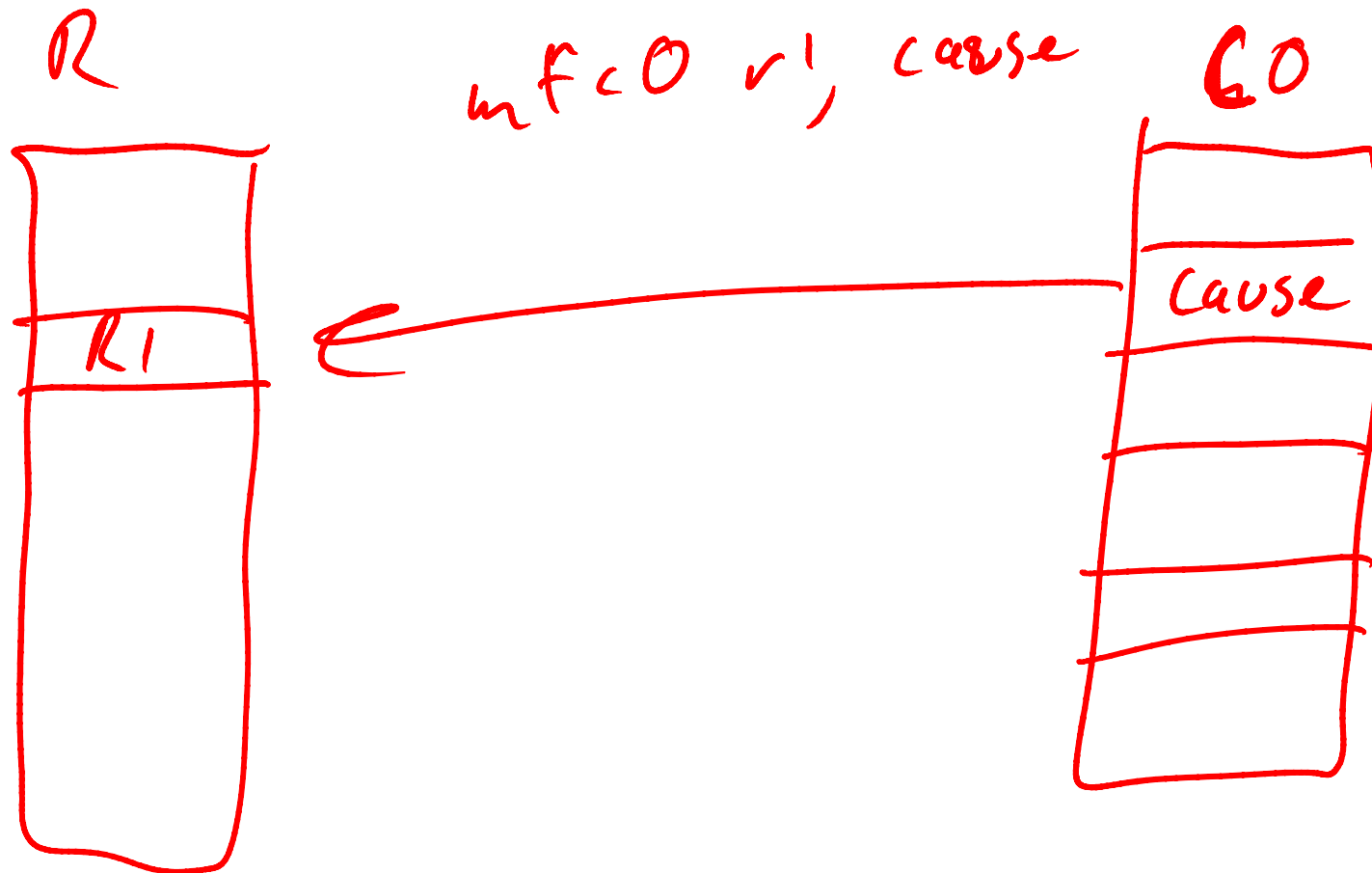


Issues

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



Approach: Trap & Emulate?



Linux

OS/161

CO Linux CO OS/161

CO

Hardaway







