Extended OS
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
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

Virtual 370s

System calls here

Trap here

I/O instructions here

CMS

CMS

CMS

VM/370

370 Bare hardware
Advantages

• Legacy OSes (and applications)
• Concurrent OSes
  – Linux – Windows
  – Primary – Backup
• 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??
R3000 Virtual Memory Addressing

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

Figure 2.10 Virtual Memory Addressing
R3000 Translation

Unprivileged (User) Mode

\[ A_{\text{phys}} = \begin{cases} 
  f_{\text{mmu}}(A_{\text{virt}}) : & A_{\text{virt}} < 0x80000000 \\
  & \text{(Privileged (Kernel) Mode)}
\end{cases} \]

Privileged (Kernel) Mode

\[ A_{\text{phys}} = \begin{cases} 
  f_{\text{mmu}}(A_{\text{virt}}) : & A_{\text{virt}} < 0x80000000 \\
  A_{\text{virt}} - 0x80000000 : & 0x80000000 \leq A_{\text{virt}} < 0xA0000000 \\
  A_{\text{virt}} - 0xA0000000 : & 0xA0000000 \leq A_{\text{virt}} < 0xC0000000 \\
  f_{\text{mmu}}(A_{\text{virt}}) : & A_{\text{virt}} \geq 0xC0000000 
\end{cases} \]
R3000 Address Space Layout

- **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 - 0xffffffff virtual = 0x00000000 - 0x1fffffff 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 - 0xbfffffff virtual = 0x00000000 - 0x1fffffff physical
    - MMU not used
  - **NOT** cacheable
  - Only kernel-mode accessible
  - Where devices are accessed (and boot ROM)

- **kseg2**
- **kseg1**
- **kseg0**
- **kuseg**
- Physical Memory

$0x00000000$ to $0xffffffff$
R3000 Address Space Layout

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

```plaintext
address space layout:

- kseg2: 0xffffffff
- kseg1: 0x00000000
- kseg0: 0xc0000000
- kuseg: 0x80000000
```
Issues

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