Conditions of ISA Virtualizability

References

• HSSV book, Chapter 2
• Smith & Nair, Chapter 8
Popek-Goldberg Requirements for ISA Virtualizability

- Given a computer that meets [this] basic architectural model, under which precise conditions can a VMM be constructed, so that the VMM:
  - provides **virtualization**
    - can execute one or more virtual machines;
  - provides **safety**
    - i.e. is in complete control of the machine at all times;
  - provides **equivalence**
    - i.e. supports arbitrary, unmodified, and potentially malicious operating systems designed for that same architecture; and
  - maintains **performance**
    - be efficient to show at worst a small decrease in speed?
Equivalence: Virtualization as Isomorphism

• Each guest state & transition must have a corresponding mapping to a host state & transition.
Safety: Resource Control

- Issue: How to retain control of resources in the VMM?
- Timer interval control performed by VMM
- Guest OS not allowed to read the timer value
  - Guest OS sees a virtual interval timer
- VMM also gains control whenever guest OS executes privileged instructions.

*Figure 8.4* Actions Taken by the VMM in Retiring One Virtual Machine and Activating the Next Virtual Machine
Instruction Types

- Non-privileged: Do not cause traps
- Privileged: Cause Traps
- Sensitive: Change/depend upon system state
- Innocuous: Not “sensitive”
Conditions of ISA Virtualizability

- Theorem: A computer architecture is **fully virtualizable** if the set of sensitive instructions for that computer is a subset of the set of privileged instructions.
Execution of Privileged Instruction

Diagram:
- **Instruction trap occurs** → **Dispatcher**
  - **Privileged Instruction** → **Interpreter Routine 1**
  - **Privileged Instruction** → **Interpreter Routine 2**
  - **Privileged Instruction** → ... → **Interpreter Routine n**
  - **Allocators**
    - **Privileged Instruction**
      - These instructions desire to change machine resources, e.g., load relocation bounds register
      - They do not change machine resources but access privileged resources, e.g., *IN*, *OUT*, *Write TLB*
Handling Privileged Instructions in a Guest OS

Guest OS code in VM (user mode)

- Privileged instruction (LPSW)
- Next instruction (target of LPSW)

VMM code (privileged mode)

- Dispatcher
- LPSW Routine:
  - Change mode to privileged
  - Check privilege level in VM
  - Emulate instruction
  - Compute target
  - Restore mode to user
  - Jump to target
Hybrid VMM Requirements

• A hybrid virtual machine monitor may be constructed for any conventional third-generation machine in which the set of user-sensitive instructions are a subset of the set of privileged instructions.

• User-sensitive instructions
  • Instructions that are control or behavior-sensitive only in supervisor mode  
  • E.g. JRST in PDP-10 or pop in x86 fail silently in user mode.

• Hybrid VMM interprets in software 100% of the instructions in guest-supervisor mode.
Hybrid VMM example:
Dynamic binary patching in early VMWare ESX server