

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 (native) and type 2 VMMs (hosted)
- 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 7 – 7.3 Textbook "Modern Operating Systems", 4th ed.
 All of chapter 7, if you're interested.



CSE

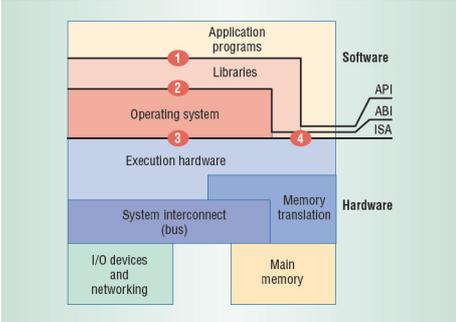
Observations

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



CSE

Interface Levels



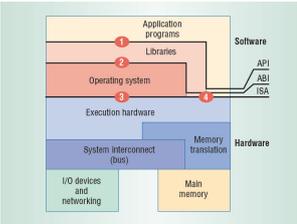
The diagram illustrates the layers of a system and the interfaces between them. The layers are: Application programs, Libraries, Operating system, Execution hardware, System interconnect (bus), Memory translation, I/O devices and networking, and Main memory. The top three layers (Application programs, Libraries, and Operating system) are grouped under 'Software'. The bottom three layers (Execution hardware, System interconnect (bus), and Memory translation) are grouped under 'Hardware'. The bottom two layers (I/O devices and networking, and Main memory) are also grouped under 'Hardware'. Four numbered labels indicate interfaces: 1 between Application programs and Libraries; 2 between Libraries and Operating system; 3 between Operating system and Execution hardware; and 4 between Execution hardware and System interconnect (bus). On the right side, labels for API, ABI, and ISA are shown, with lines indicating their scope across the layers.



CSE

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



This diagram is identical to the 'Interface Levels' diagram, showing the layers of a system and the interfaces between them. It highlights the interface between software and hardware (labels 3 and 4) and the division into privileged and un-privileged parts.



Application Binary Interface

- Interface between programs ↔ hardware + OS
 - Label 2+4
- 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 specific labels 1, 2, 3, and 4 marking key interfaces.

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

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

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

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....

Abstraction versus Virtualisation

The diagram illustrates two ways of mapping files to hardware. In (a) 'Abstraction', two 'File' boxes are connected by dashed lines to a single physical disk icon. In (b) 'Virtualization', two 'File' boxes are connected by dashed lines to two separate virtual disk icons, which are then connected to a single physical disk icon.

Process versus System Virtual Machine

The diagram compares two virtual machine models. (a) 'Process virtual machine' shows a 'Guest' with an 'Application process' and 'Runtime' layer, which runs on 'Virtualizing software' on top of a 'Host' 'OS' and 'Hardware'. (b) 'System virtual machine' shows a 'Guest' with 'Applications' and 'OS' running on 'Virtualizing software' on top of a 'Host' 'OS' and 'Hardware'.

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

```

    graph TD
      A[Java Code (.java)] --> B((JAVAC compiler))
      B --> C[Byte Code (.class)]
      C --> D[JVM]
      D --> E[Windows]
      D --> F[Linux]
      D --> G[Mac]
    
```

THE UNIVERSITY OF NEW SOUTH WALES

Comparing Conventional code execution versus Emulation/Translation

```

    graph TD
      subgraph a
        A1[HLL program] -->|Compiler front end| A2[Intermediate code]
        A2 -->|Compiler back end| A3[Object code]
        A3 -->|Loader| A4[Memory Image]
      end
      subgraph b
        B1[HLL program] -->|Compiler| B2[Portable code]
        B2 -->|VM loader| B3[Virtual memory image]
        B3 -->|VM interpreter/compiler| B4[Host instructions]
      end
      A4 -.->|Distribution| B2
      B2 -.->|Distribution| A3
    
```

THE UNIVERSITY OF NEW SOUTH WALES

Aside: Just In-Time compilation (JIT)

THE UNIVERSITY OF NEW SOUTH WALES

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

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?
 - Performance isolation?
- What about activities requiring “root” privileges

THE UNIVERSITY OF NEW SOUTH WALES

Virtual Machine Monitors

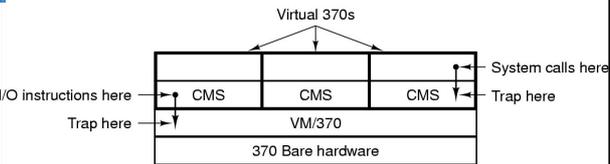
Also termed a *hypervisor*

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



IBM VM/370

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

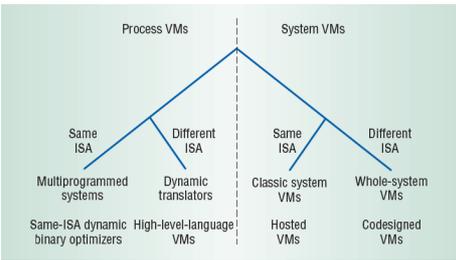



Advantages

- 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



Taxonomy of Virtual Machines




What is System/161?

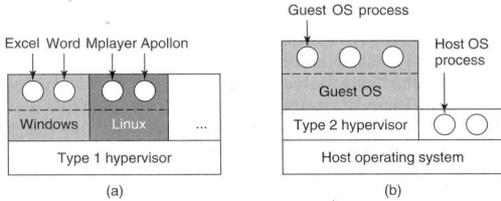



Figure 1-29. (a) A type 1 hypervisor. (b) A type 2 hypervisor.



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
- What happens when guest OS executes native privileged instructions?

(a)

THE UNIVERSITY OF NEW SOUTH WALES

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

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

Approach: Trap & Emulate?

THE UNIVERSITY OF NEW SOUTH WALES

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	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

THE UNIVERSITY OF NEW SOUTH WALES

X86 POPF

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

THE UNIVERSITY OF NEW SOUTH WALES

Cse

Virtual R3000???

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

THE UNIVERSITY OF NEW SOUTH WALES

Cse

Issues

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

THE UNIVERSITY OF NEW SOUTH WALES

Cse

R3000 Virtual Memory Addressing

```

            graph LR
                Processor[Processor] -- Virtual Address --> MMU[Memory Management Unit]
                MMU -- Real Address --> MainMemory[Main Memory]
                MainMemory --- SecondaryMemory[(Secondary Memory)]
                SecondaryMemory -- Disk Address --> MMU
            
```

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

Figure 2.10 Virtual Memory Addressing

THE UNIVERSITY OF NEW SOUTH WALES

36

