Virtual Machines

Pr. Olivier Gruber

Full-time Professor

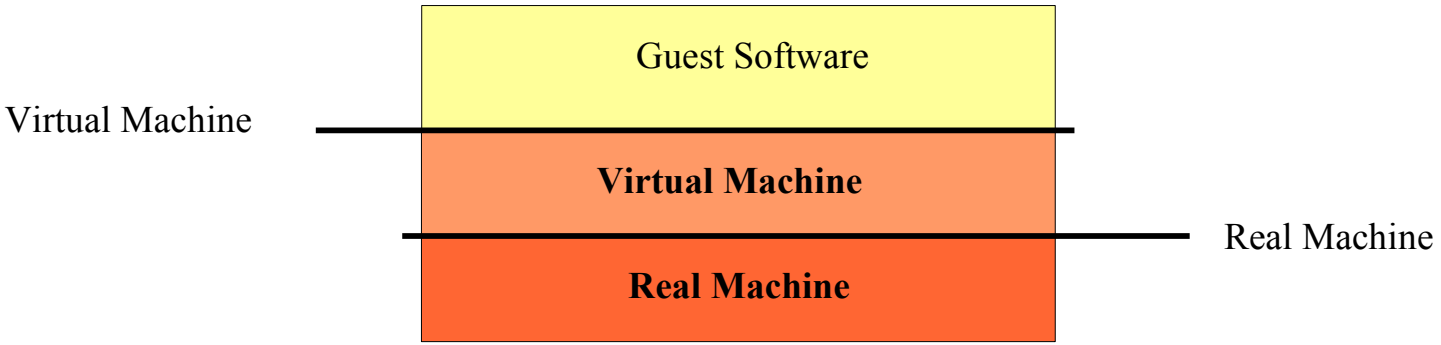
Université Joseph Fourier

Laboratoire d'Informatique de Grenoble

Olivier.Gruber@imag.fr

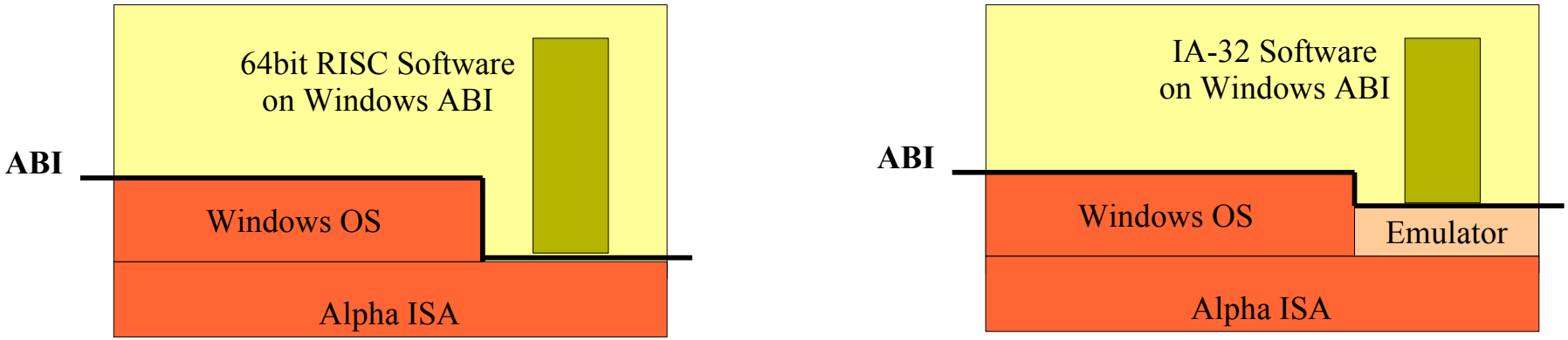
Virtual Machine Basics

- ***Virtual Machines*** versus ***Real Machines***
  - *A virtual machine defines a machine (interface)*
  - *A virtual machine is a machine (implementation)*



Dec Alpha Example

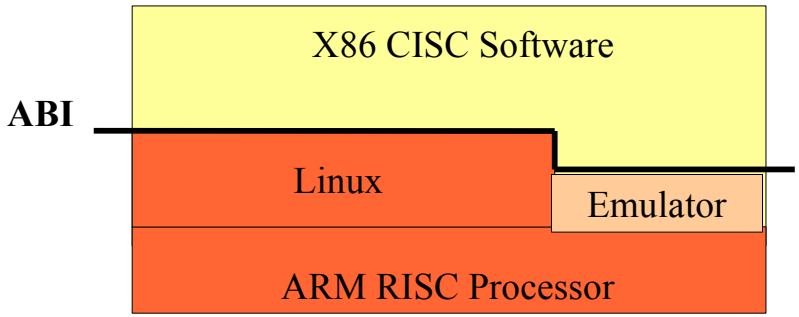
- Digital Alpha machine
  - Early provider of a 64bit RISC processor
    - Challenge: no existing software...
  - Support program binaries compiled to a different ISA / same ABI
    - Same ABI: ported the operating system
    - Different ISA: emulate one instruction set on a different instruction set



Hookway and Herdeg. *Digital FX!32: Combining Emulation and Binary Translation*.
Digital Technical Journal, January 1997, pp 3-17
Zheng and Thompson. *PA-RISC to IA-64: Transparent Execution, No Recompilation.*
IEEE Computer, March 2000, pp. 47-53

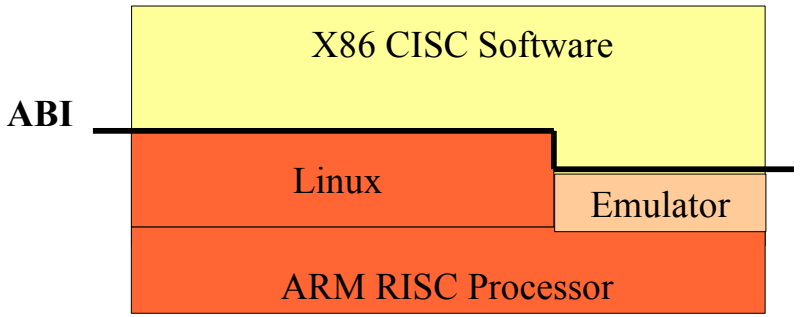
Design Study

- Step One
  - *Using interpretation* (fetch, decode and emulate)
  - Design the emulator for x86 CISC to ARM RISC
  - Running inside a linux process
  - Emulating x86 code, compiled against a compatible linux ABI



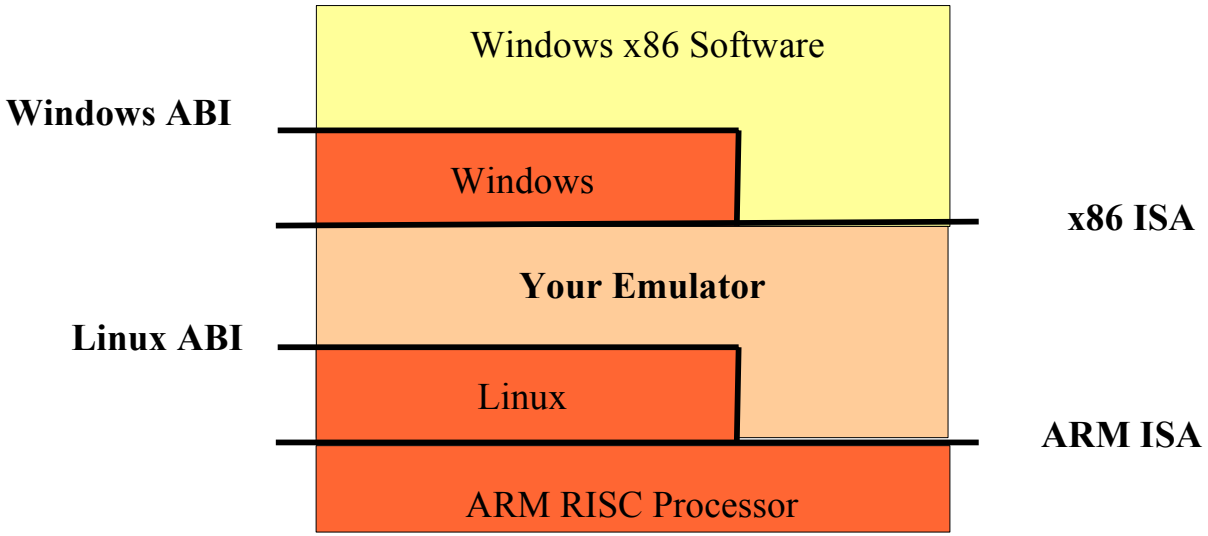
Design Study

- Step One Answers
  - Your emulator runs inside a process
    - Essentially about fetch, decode and emulate
  - Fetching
    - Manage a program counter, being aware of branch operations
  - Decode - Emulate
    - A switch like decoding leading to executing emulation snippets
    - Manage x86 registers
    - Emulate memory over a large malloc
    - Bridge the ABI calls



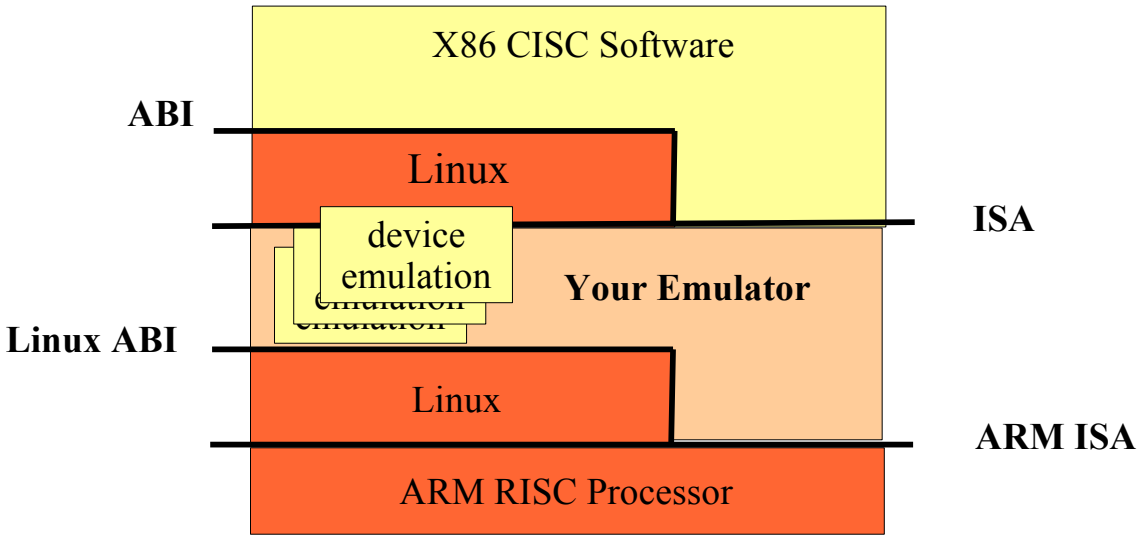
Design Study

- Step Two
  - *Still using interpretation* (fetch, decode and emulate)
  - Design the emulator for **the complete IA-32 ISA**
  - What is different now?



Design Study

- Step Two Answers
  - Just a more complex architected state
  - Registers and memory
  - Complete interrupt/trap emulation
  - Complete device emulation, leverage Linux device drivers



Emulating Devices

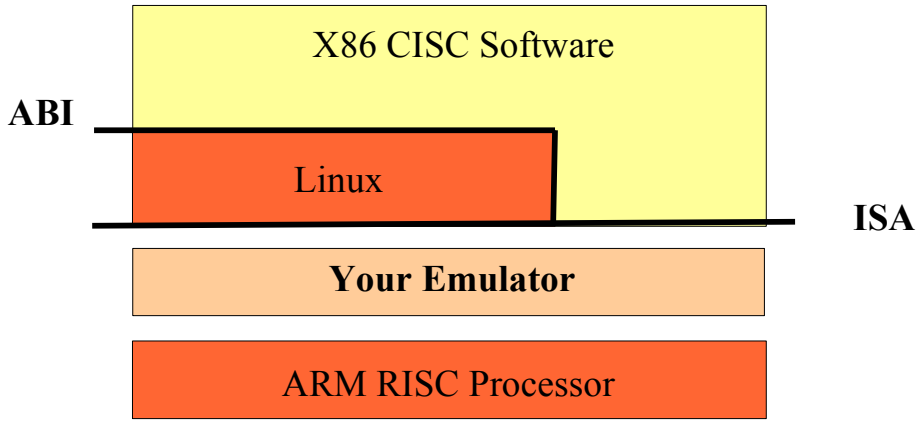
- Serial lines
  - Maybe over sockets or pipes
  - Easy byte in/out
- Screen and Mouse
  - Maybe over SDL
  - A window as the screen → emulate a frame buffer
  - Mouse events → encoding x,y position and buttons
- Keyboard
  - Scan codes / character translation
  - Assumes a keyboard layout

Emulating Devices

- Emulate a hard disk or a network card
  - Maybe over a file, or a real partition
  - Decode I/O requests
    - Automata, input from emulated hardware registers
    - Often organized as a ring buffers (in/out)
  - Issue real I/O on the file
    - Blocking I/O would be a poor emulation
    - Requires multi-threading in your emulation
  - Raise interrupts
    - Race condition with the execution
    - Interrupts only happen in between instructions

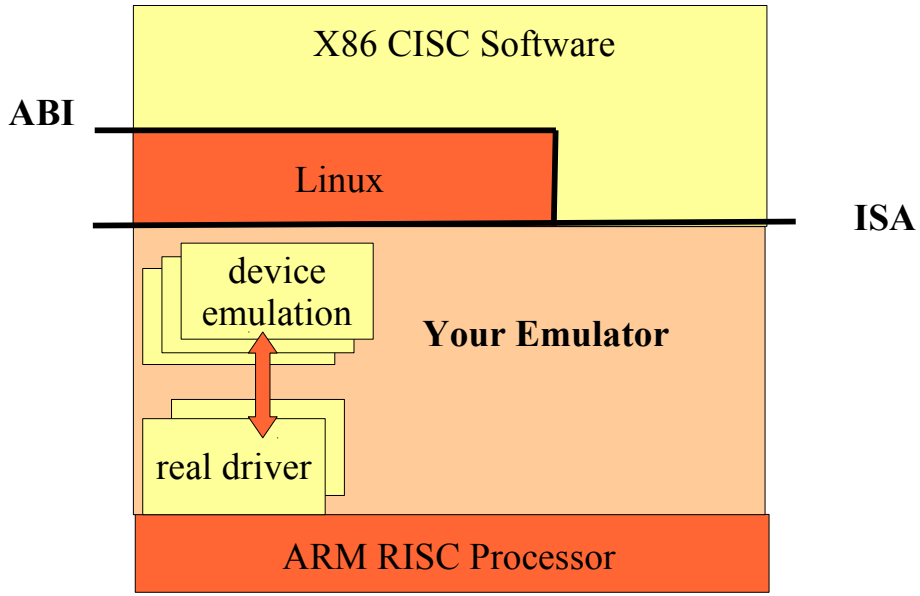
Design Study

- Step Three
  - *Still using interpretation* (fetch, decode and emulate)
  - Still emulating a complete x86 illusion
  - **But a bare metal emulation** (no operating system)
  - What is different now?



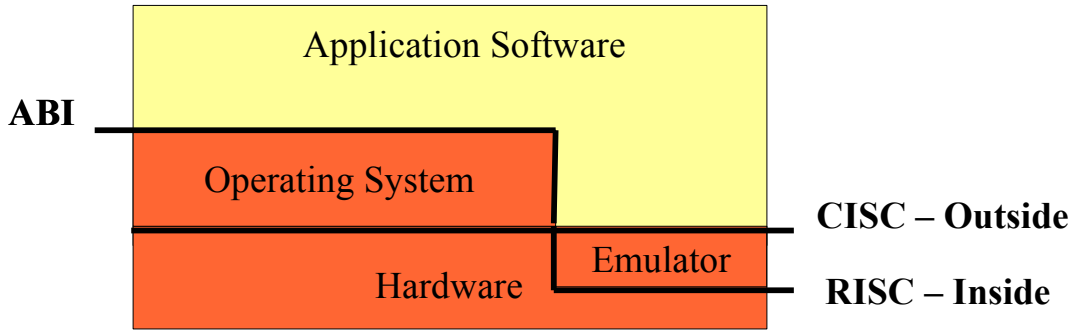
Design Study

- Step Three Answers
  - You can no longer rely on an operating system
  - You still need an IA-32 emulator
  - You also needs drivers for the real devices
  - And you also need an emulation for virtual devices



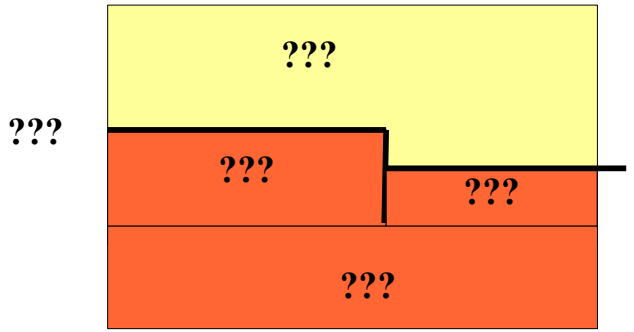
CISC Outside – RISC Inside

- Did you know?
  - Intel Processors are emulators
  - CISC code is translated, optimized, and parallelized on the fly
    - One of the key performance aspects of Intel processors
    - One of the reasons for its high energy consumption
  - Other features are possible
    - Speculative execution for example
  - ***So real machines are not so real after all…***



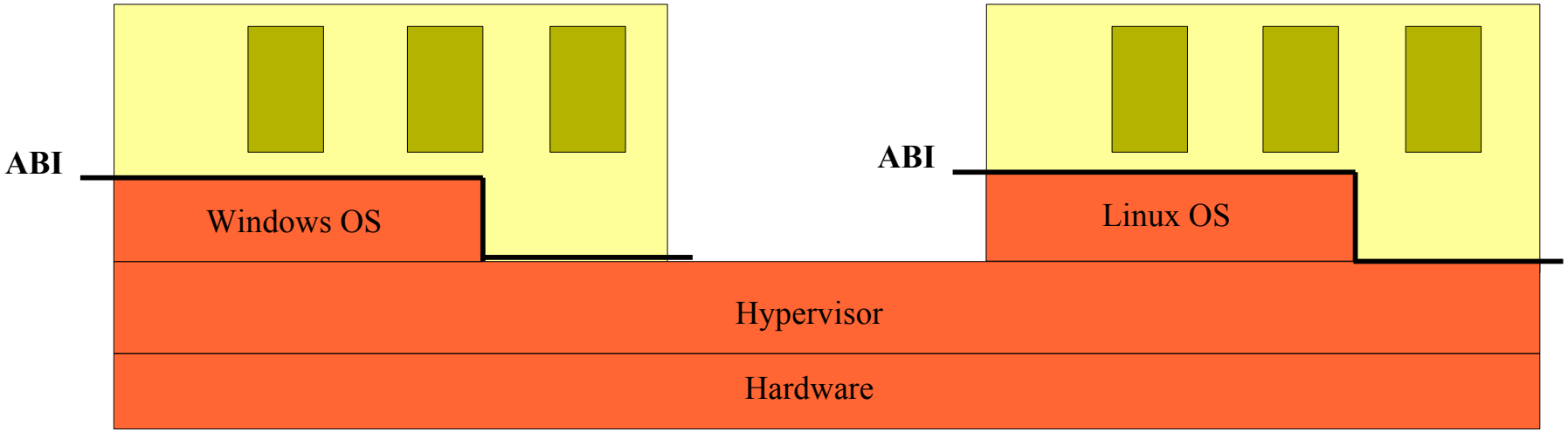
Virtual Machine – So what are they?

- Operating Systems
  - Unix – Linux – Windows, almost a 40 year-old design
  - Different approaches exist(ed)
    - Palm-OS – a refreshing approach
    - AS-400 – a different world
    - Mac-OS with micro-kernel technology
- Hypervisors
  - New or old technology?
  - Both of course…
- High-level languages – a different future?
  - Microsoft DotNet – quite a fundamental change
  - Java bare metal or Arduino
  - Google Android



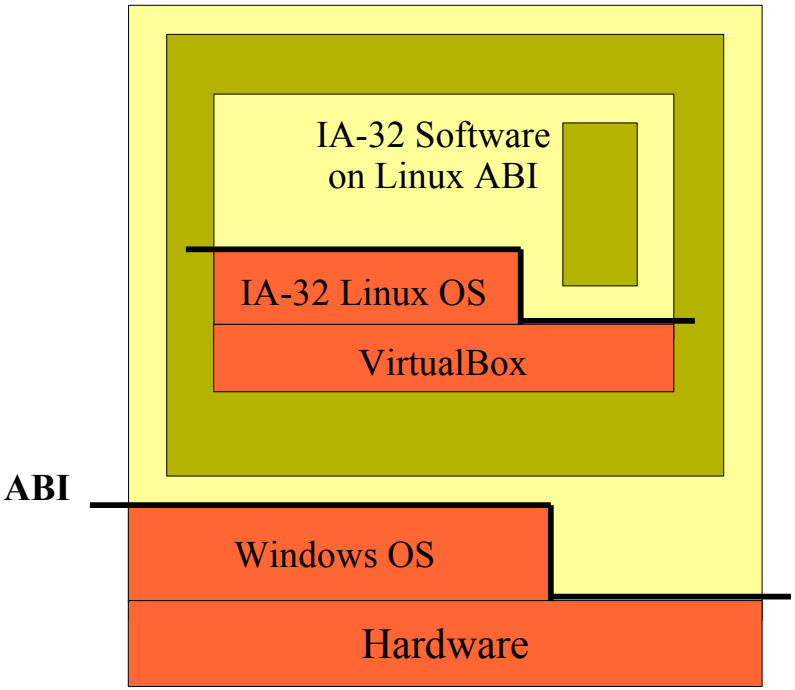
Hypervisor – Basics

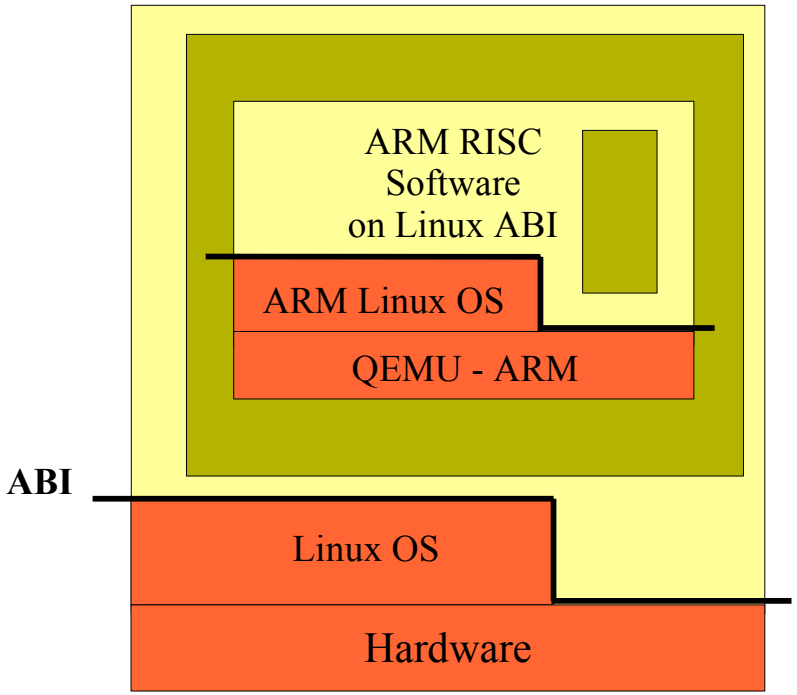
- Bare-metal virtualization (Type-I or System VMs)
  - Such as Vmware ESX or Xen or Nova
  - Multiplex operating systems, often para-virtualized
  - Often virtualize a similar hardware (but not always)
  - Often relies on hardware-assisted virtualization



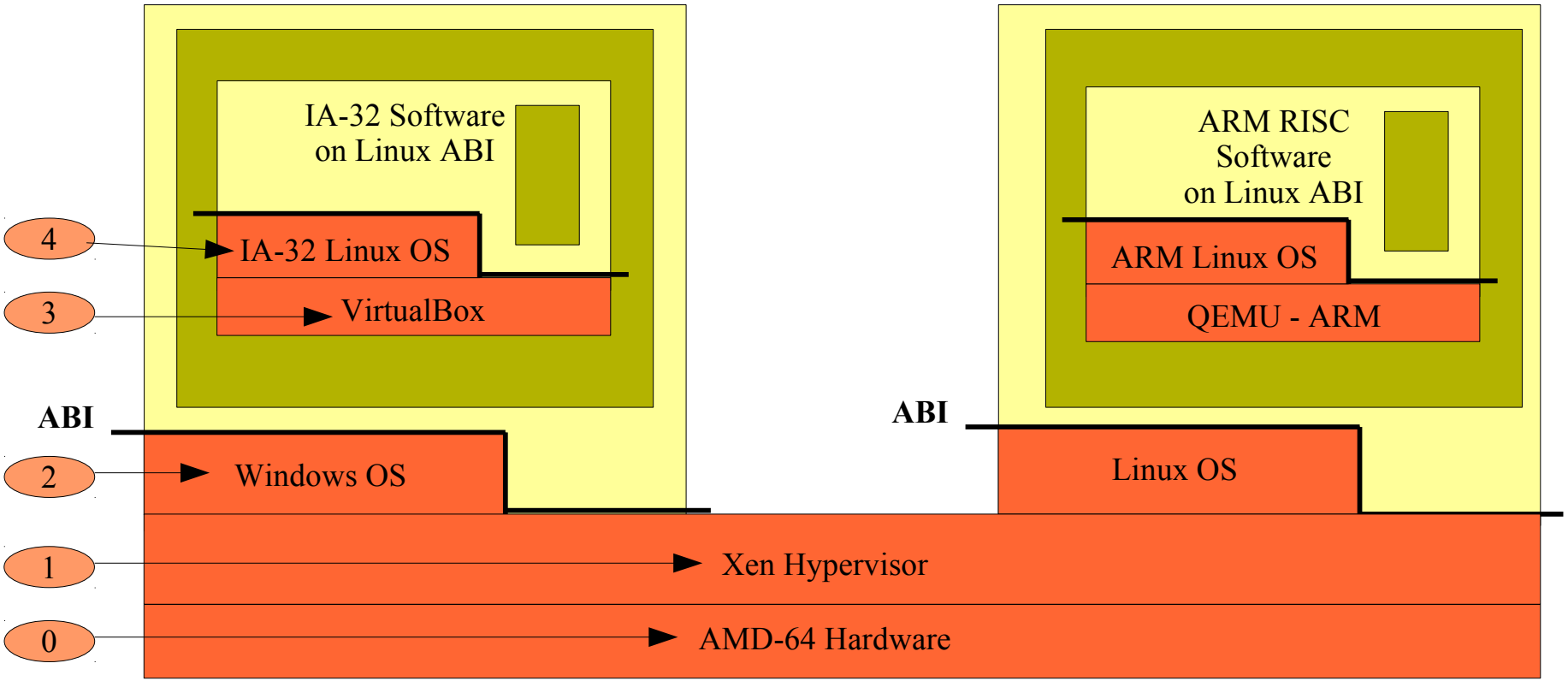
Hypervisor – Basics

- In-Process Hypervisor (Type II or Process VMs)
  - Such as VirtualBox or QEMU
  - Virtualizing a real machine within a process
  - Para-virtualized or out-of-the-box operating systems
  - Same hardware (QEMU/VirtualBox) or not (QEMU)



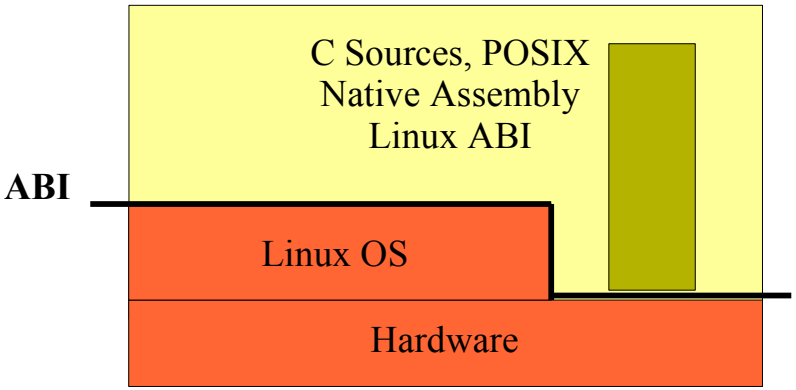


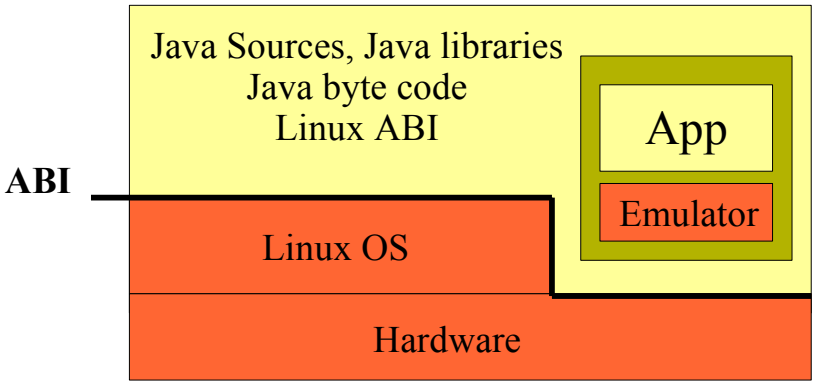
Hypervisor – Overview



High-Level Virtual Machine Basics

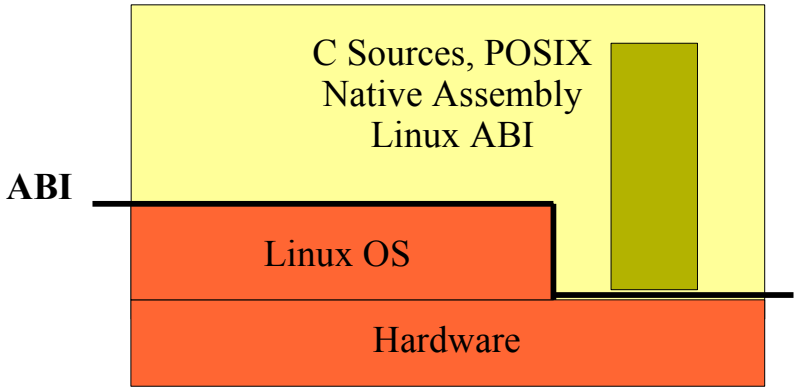
- High-level language virtual machines
  - Many are just about yet another language
  - Such as Python, Perl, JavaScript, or Java
- Virtualize
  - Provide a different instruction set
  - Sometimes provide different libraries

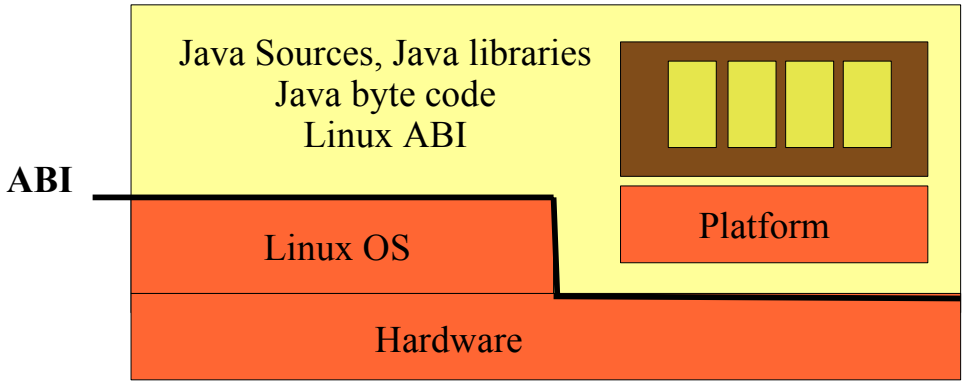




High-Level Virtual Machine Basics

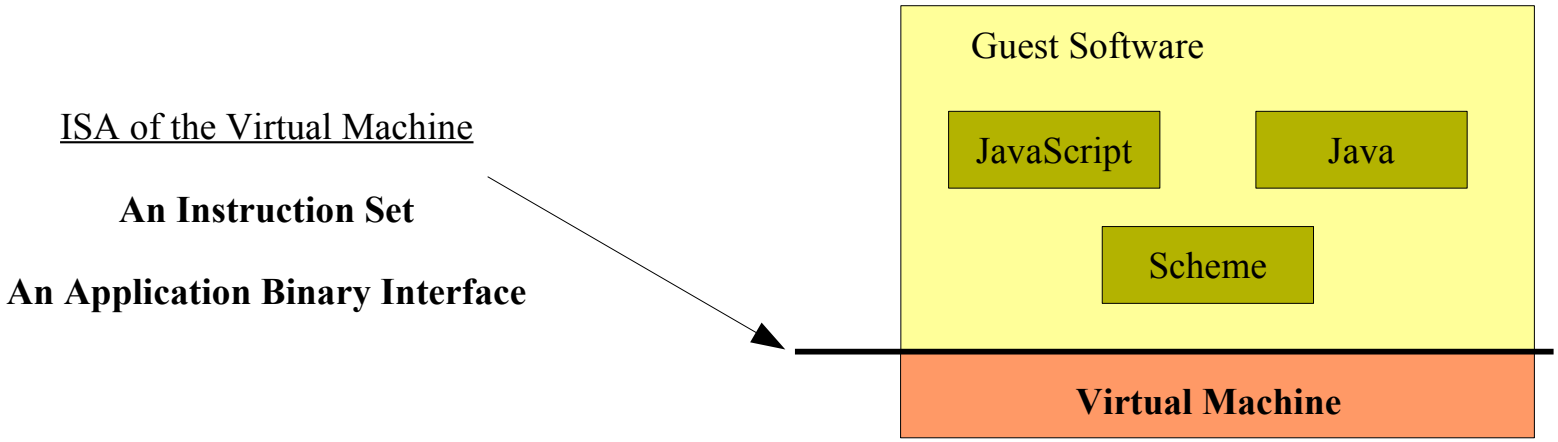
- But some are full-fledge platforms
  - Microsoft C# and DotNet platform
  - Eclipse Rich Client Platform, based on OSGi
  - JONAS Web Server, based on OSGi, a servlet and bean containers





High-Level Virtual Machine Basics

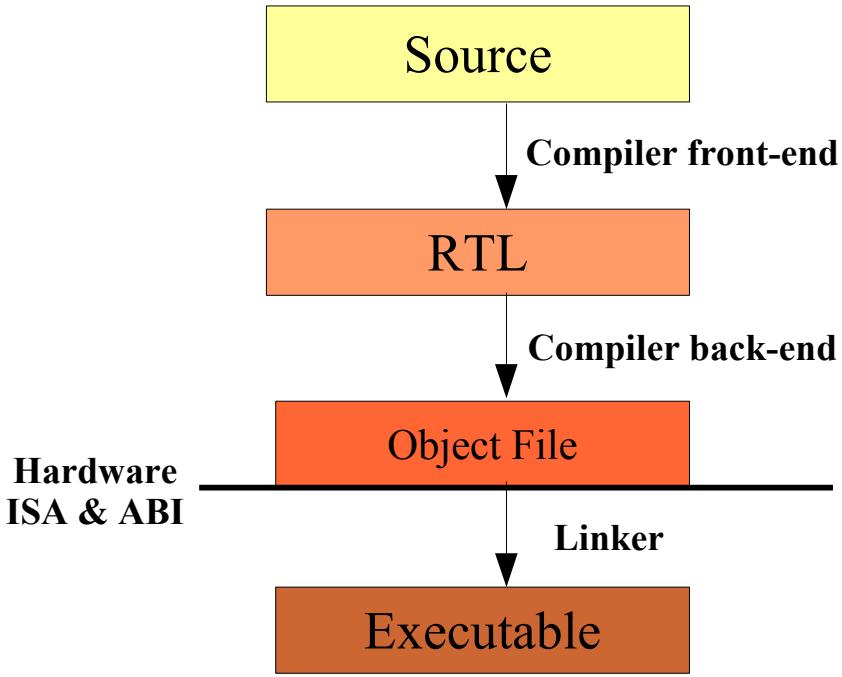
- Towards language independence
  - Microsoft Common Language Infrastructure
    - Common Language Runtime (CLR) and Common Type System (CTS)
  - The Java Virtual Machine is going in the same direction
    - Already a target for many languages (JavaScript, Scheme, Perl, Python, etc.)
  - LLVM Project
    - Interesting path



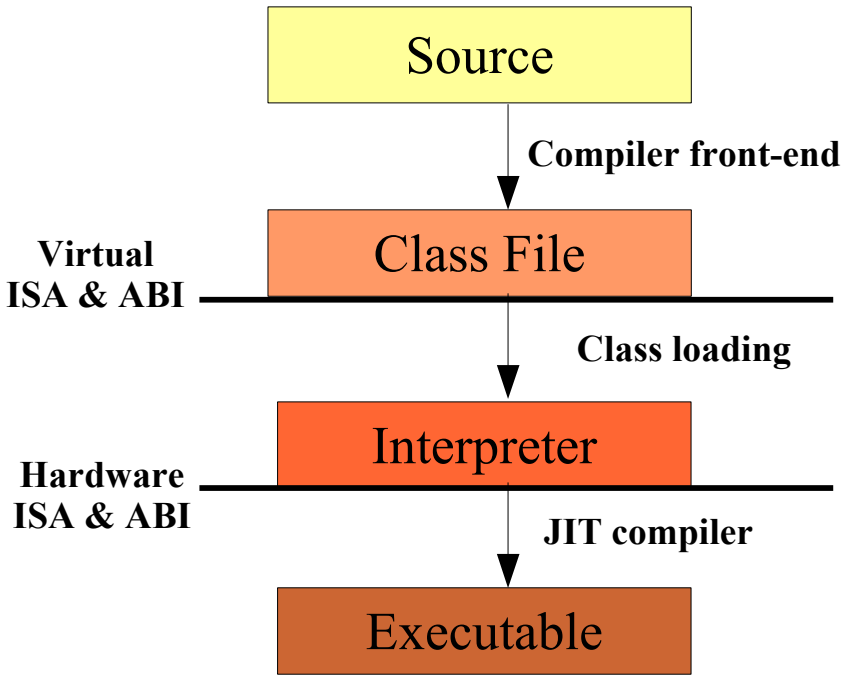
Compilation & Linking

- A shift of responsabilities...

Traditional Language Compilation & Linking Chain



High-Level Language Compilation & Linking Chain



Virtual Machine – Everywhere