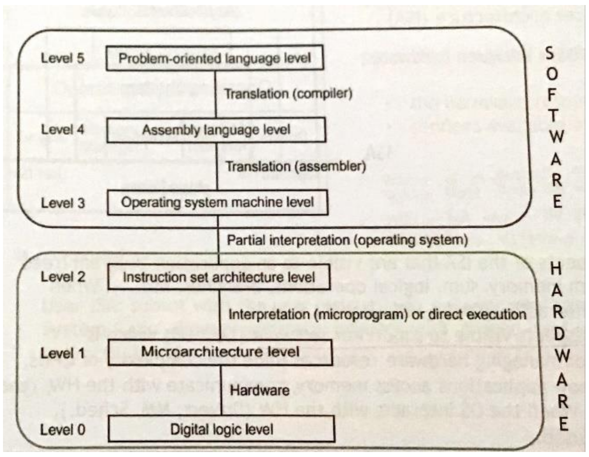
In a machine there are 6 levels of instructions, to move from one to the lower one a translator is needed.


Level 5
Problem-oriented language level
Translation (compiler)
Level 4
Assembly language level
Translation (assembler)
Level 3
Operating system machine level
Partial interpretation (operating system)
Level 2
Instruction set architecture level
Interpretation (microprogram) or direct execution
Level 1
Microarchitecture level
Hardware
Level 0
Digital logic level
SOFTWARE
HARDWARE

Level 2 is that of the **instruction set architecture (ISA)**, which represents the bunch of instructions the machine can execute.
It is divided into:

- User ISA: visible to an application program;
- System ISA: can be seen only by the operative system, used to perform its specific task.

Another important component is the **application binary interface (ABI)** which allows the program to access to hardware resources and other available systems. It is composed by:

- User ISA
- System calls, which allows a program to indirectly interact with Os-managed resources

Not possible to run on instructions that were not meant for that machine, but the **hardware can be virtualised**: Java runs in every architecture for which an interpreter exists, as it uses a specific set of instructions which is translated by the Java Runtime Environment into machine instructions.
The same technique can be used to run on a machine software which is meant for another one, by using **emulators**. They understand the behaviour of the CPU and of the RAM and reproduce it into the physical system on which it is running. Both the ABI and the ISA are reproduced.

A **virtual machine** is a logical abstraction able to provide a virtual execution environment or a full system one. It maps virtual resources in physical ones and it uses the physical machine instructions to execute virtual ones, the **host** is the software that runs in the virtual machine while the **guest** is the underlying platform that supports it. There are two types of virtual machines.

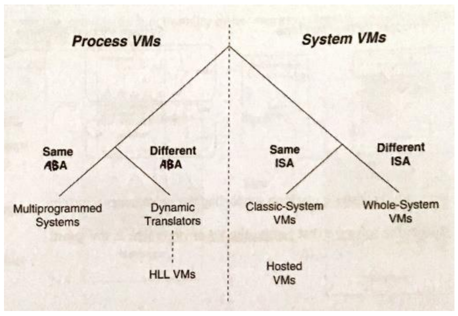
1. Process virtual machines

The ABI provides the interface between the physical machine and the virtual one, onto which it is possible to execute processes but no other operative systems. The virtualisation software is called **runtime software**, it emulates user-level ISA and operating system calls, meaning that it must rely in the software of the physical machine to access the hardware as it supports the level 0-3 of the architecture.

2. System virtual machines

The ISA provides the interface between the two machines, meaning that each machine can run its own operative system and interact independently with resources, either via an underlying OS or directly communicating with the hardware, which includes also the I/O resources. The virtualising software is the **virtual machine monitor** (VMM), it supports the level 0-2 of the architecture, as it does not include the software.

A great problem is the slowing down of the machine due to the need of translating the instructions, but it can be solved (not always) exploiting the fact that the languages of the two machines coincide, which means that they have the same ABI for process VMs or the same ISA for system ones.



- Same architecture: depending in the level, it can be the same IBA is the same ISA
    - Process VMs have the same ABI, this technique is used to allow multiple users to run in the same machine, but it is not considered a real virtual machine.
    - System VMs have the same ISA, the machine is on the bare hardware and the VMM can connect it directly with the I/O and the memory. If the machine runs on top of another operative system, it is called **hosted VM**.
- Different architecture: **emulation** is required, it refers to those techniques developed to allow an application to run in an environment different from that originally intended. Two methods can be implemented in two ways, through interpretation or binary translation, which is faster as it converts blocks of instructions at runtime and caches them.
    - Process VMs: every application has a different execution environment, the VMM translates the byte code of the applications, which are sandboxed, into that of the machine.
    - System VMs: usually they run on top of another OS, in this case the VM must emulate the entire hardware environment.

Virtualisation must be implemented if the machine needs to execute instructions meant for another ISA or another ABI, depending on the level where it is placed, it is possible to obtain different types:
- Hardware-level virtualisation is placed between the hardware and the operative system;
- Application-level one is put between the operative system and some application, which run in its environment independently and can be also a software.

Virtual machine managers (system VMs, same ISA)

The VMM manages the virtual machines, mediates the access to the hardware resources and handles protected instructions; it is crucial to support cloud computing. There are three terms, which have a slightly different meaning:
- Virtual machine monitor, the software
- Virtual machine manager, the software which takes care of virtualised resources providing a user-friendly interface
- Hypervisor, which runs directly on the hardware

Let's focus on the hypervisor, it can be
- Bare metal (type 1), which takes direct control of the hardware and was sometimes used to provide hacks for certain OS.
    - Monolithic: the drivers are installed on the hypervisor, so they need to be available for that software, it leads to better performances and isolation
    - Microkernel: the drivers are installed on a service virtual machine, so those specific for the OS and also 3rd party ones (even if difficult) can be used, this allows also to keep the hypervisor small
- Hosted (type 2), which is characterised by at least two operative systems, the host where the VMM runs and the guest where the virtual machine runs. It is more flexible and simpler, but special care must be taken to avoid conflicts between the systems and to keep low the resources consumed by the guest.

VMM must ensure the following properties:
- Partition,
- Isolation,
- Encapsulation (the state can be saved in a file),
- Hardware independence

There are three types of virtualisation

1. Para-virtualisation: the interface is similar but not identical to that if the underlying hardware, the guest OS can require the host to execute some instructions that would be virtualised thus making them faster; only some modified guest OS can be installed in these machines.
2. Full virtualisation: the guest OS does not need to be modified, but the hypervisor must mediate, and some hardware support is required
3. Kernel-level virtualisation: the virtualisation is at the OS level, it does not allow multiple OS to run but it can be used to support private servers, it requires some kernel level modification

Implementing full virtualisation

The machine being virtualised is composed of:

- E, executable storage
- M, the system mode (user or kernel)
- P, the program counter
- R, it represents the memory relocation registers which is a pair (l, b) where l is the physical address of the virtual memory and b its bound

Instructions are classified in three categories

1. Privileged, those which need a trap to be executed
2. Control sensitive, which affect the configuration of resources
3. Behaviour sensitive, their result depends on the configuration of the resources

Requirement for system-level VM

1. Efficiency: the instructions that need the VM intervention must be less than those that can be executed without it;
2. Equivalence: the behaviour of the program must be identical to that in a physical machine;
3. Resource control: the VMM hat control completely the virtualised resources.

Popek and Goldberg: theorem

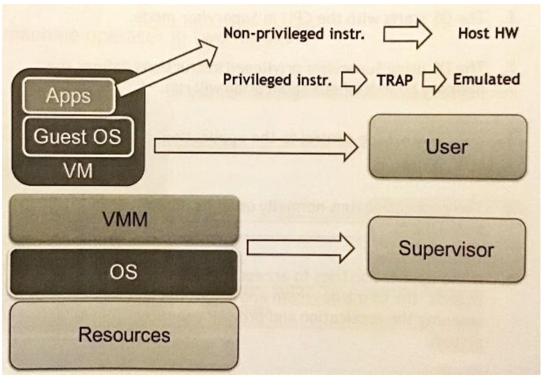
A VMM may be built if the privileged instructions are a subset of control/behaviour sensitive ones

The property must be guaranteed

1. Resource control: instructions that could prevent the correct functioning of the VMM are trapped and handled by the VMM itself
2. Efficiency: non-privileged instructions must not be controlled by the VMM
3. Equivalence: the virtual machine accesses virtual resources in executing sensitive instructions, so they need to be handled by the VMM

AMD and Intel implemented virtualisation by adding a new high-privilege ring for the VMM, so that instructions on a virtual machine can be in two modes:

- VMX root, which can execute a whole set of instructions (VMX) that deal with critical shared resources
- VMX non-root, which is similar to the user mode in non-virtual machines



The guest OS believes to be running at the root level, but it is running at the VMX non-root.