UNIT - 3.

1. Different layers, interfaces in a computer system

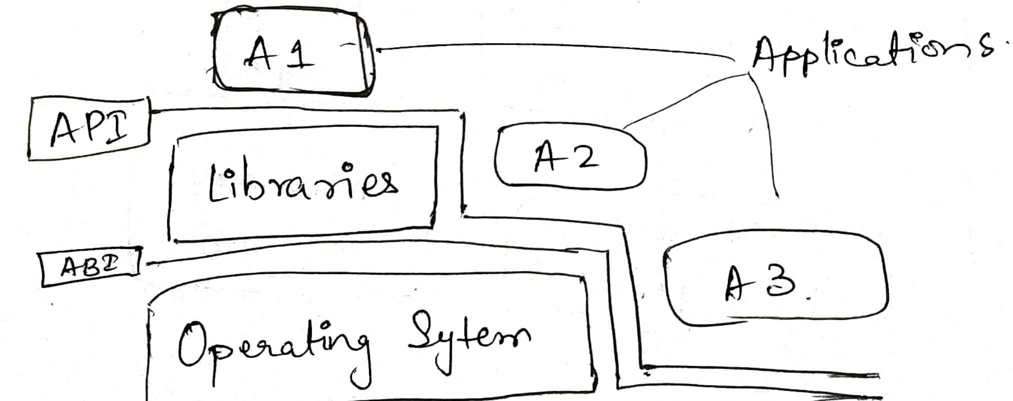
→ Computer systems are complex & have interfaces among the s/w components and the h/w.

* The h/w supports 2 execution modes - privileged/kernel & user.
* The instruction set consists of 2 sets of instr. privileged instr. that can be executed only in kernel mode & non-privileged instr. that can be executed in user mode.

~~There is~~

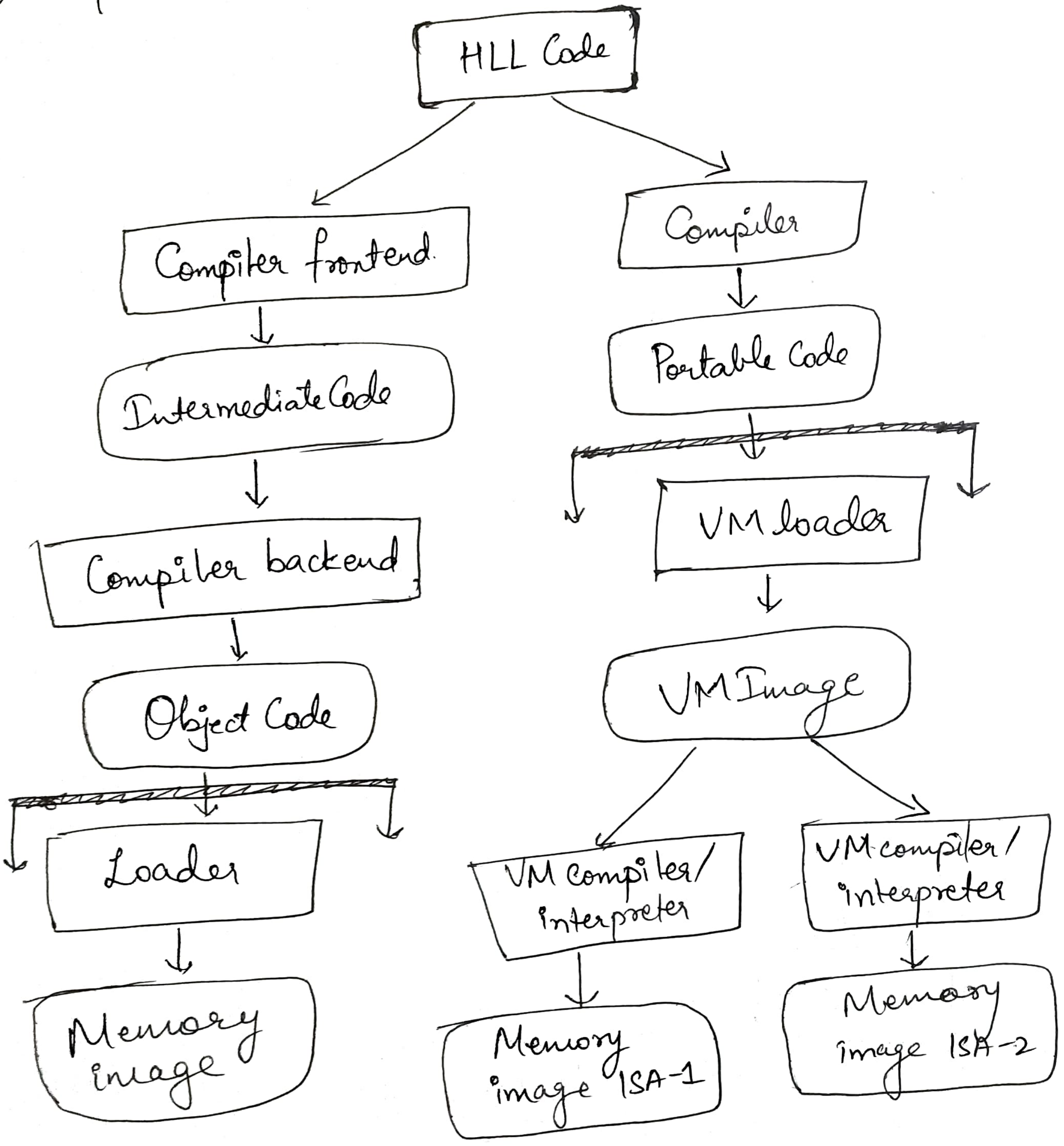
→ Layering:

1) Instruction Set Architecture (ISA) - boundary b/w h/w & s/w.
2) Application Binary Interface (ABI) - allows ensemble which consists of the application & the library modules to access the h/w
3) Application Program Interface (API) - defines the set of instr. the h/w was designed to execute & gives the application access to the ISA; also includes HLL library calls.



* A1 - uses library funcⁿ
* A2 - makes system calls.
* A3 - executes machine instr

③ Compilationprocess of HLL program & portable code



* It is possible to compile an HLL program for a VM environment where portable code is produced & distributed & then converted to binary translators to the ISA of the host system.
* Dynamic binary translation - converts blocks of guest instructions from portable code to the host instruction and leads to a significant performance improvement as such blocks are cached and reused.

Binaries created for specific ISA & specific OS are not portable.

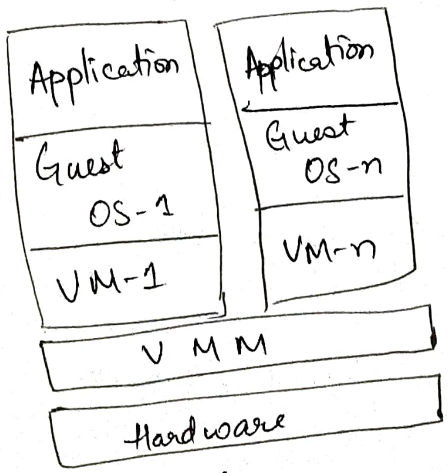
×~~Ext is possible~~

(5) Traditional vs Hybrid vs Hosted VM.

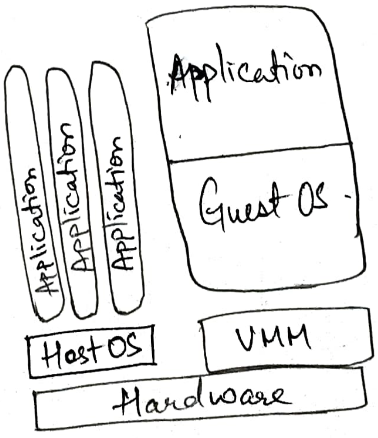
→ Traditional VM - supports multiple VMS.
runs directly on hardware.
also called bare metal VMM
main adv - performance.
Eg: VMWare ESX, ESXi Servers, Xen.

→ Hybrid VM - VMM shares the hardware with existing OS.
- supports multiple VMs.
Eg: VMWare workstation.

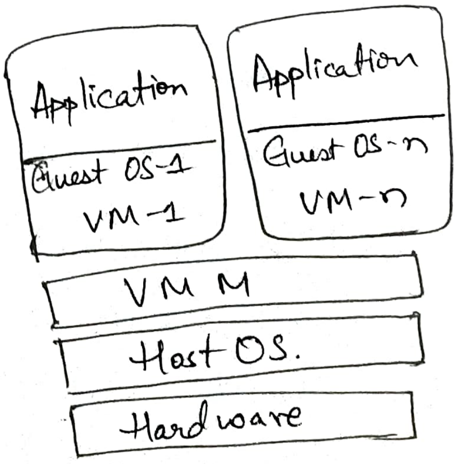
→ Hosted VM - VM runs on top of an existing OS.
- main adv - VM is easier to build & install.
- VM could use several components of OS.
- disadv - increased overhead & associated perf. penalty
Eg: User-mode Linux.



Traditional VMs.



Hybrid VM



Hosted VM.

⑦ Conditions of effective virtualization. Basic approaches to Processor virtualization.

→ Conditions for effective virtualization.

* a program running under the VMM should exhibit a behaviour essentially identical to that demonstrated when running on an equivalent machine directly.
* The VMM should be in complete control of the virtualized resources.
* A statistically significant fraction of machine instr. must be executed without the intervention of VMM.

→ There are 2 basic approaches to processor virtualization full virtualization & para virtualization.

↳ Full virtualization

* a guest OS can run unchanged under the VMM as if it was running directly on the h/w platform.
* ~~[illegible]~~ requires a virtualizable architecture.
* Eg: VMWare

↳ Para Virtualization :

* a guest OS is modified to use only instrs. that can be virtualized.
* Some aspects of h/w can't be virtualized thus need para v.
* It has improved performance
* presents a simpler interface
* Eg: Xen, Denaly

How does virtualization stimulate the interface of physical objects?

→ Virtualization simulates the interface of a physical object by 4 ways-

1. Multiplexing - creates multiple virtual objects from one instance of a physical object.
   Eg: processor is multiplexed among a no. of threads.

2. Aggregation - create one virtual object from multiple physical objects.
   Eg: no. of physical disks are aggregated into a RAID disk.

3. Emulation - construct virtual object from a different type of physical object.
   Eg: physical disk emulates a random access memory.

4. Multiplexing & Emulation -
   Eg: virtual memory with paging multiplexes real memory and disk, & a virtual address emulates real address.
   Eg: TCP emulates a reliable bit pipe & multiplexes a physical communication channel & a processor.

Q. Paravirtualization in x86-64 Itanium through vBlades VMM.

→ The goal of vBlades was to create a VMM for the Itanium family of IA64 Intel processor, 14 capable of supporting the executing of multiple OS in isolated protection domains with security & privacy enforced by the hardware.

→ Itanium processor hardware supports 4 privilege rings PL0, PL1, PL2, and PL3.

→ Privilege instructions executed at PL0, applications run at PL3.
PL2 and PL4 rings generally not used.

→ VMM uses ring compression and runs itself at PL0 & PL1, and forces guest OS to run at PL2.

→ Itanium was selected because of its multiple functional units and multithreading support.

→ Itanium processor has →
30 functional units,
6 General purpose ALUs.
2 Integer units, 1 shift unit,
4 cache units
6 multimedia units
2 parallel shift units, etc.

→ The hardware supports 64 bit addressing. It has 32 64 bit general purpose registers (R0 – R31).

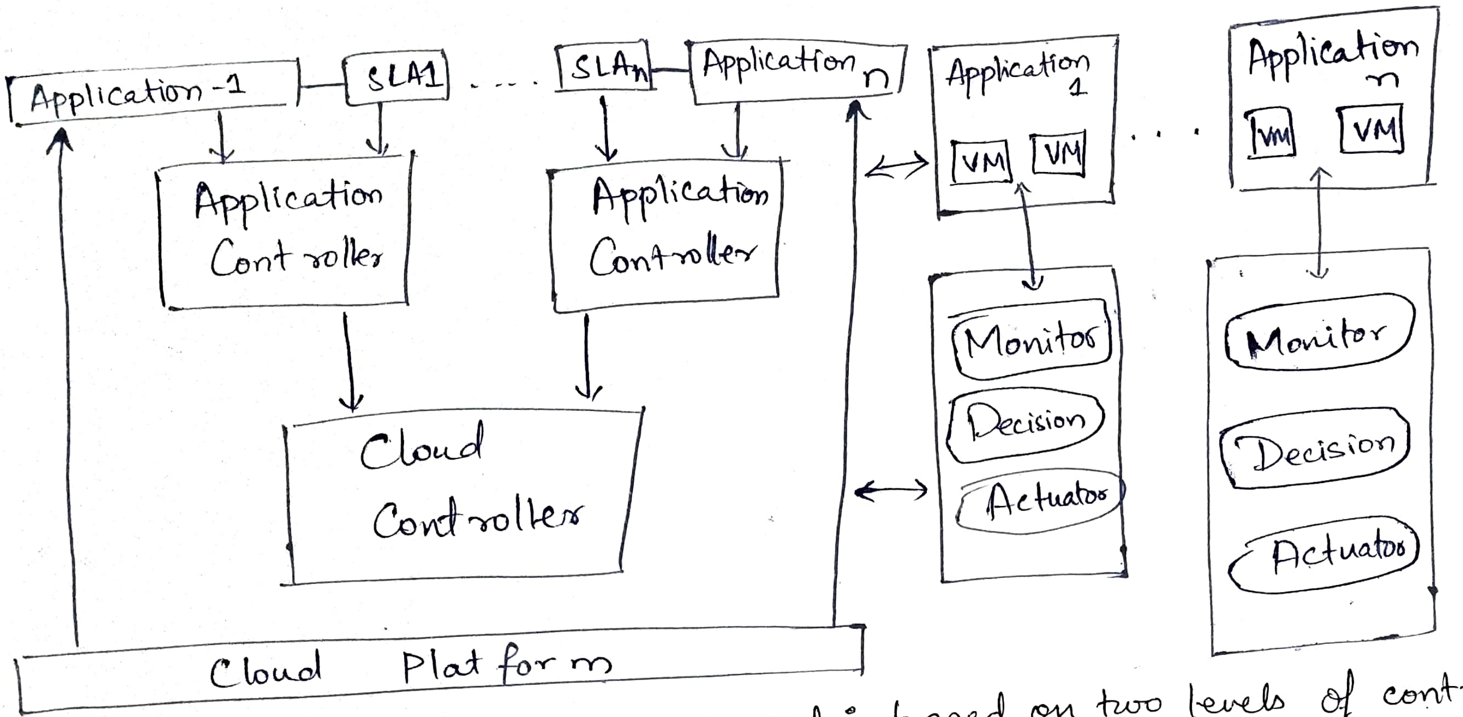
→ Itanium processor supports isolation of address spaces of different processes with 8 privileged region registers.

→ The Processor Abstraction Layer (PAL) firmware allows the caller to set the values in the region register.

The VMM intercepts the privileged instructions issued by the guest OS to its PAL & partitions the set of address spaces among the guest OSs to ensure isolation.

→ Each guest is limited to ~~[illegible]~~ 218 addr spaces.

Q. 2 Level Control Architecture



* The automatic resource management is based on two levels of controllers one for the service provider & one for the application.
* Main components of a control system → i/p, o/p and system controllers.
* Inputs are offered workload & policies for admission control, capacity allocation, load balancing, energy optimization & QoS guarantees in cloud.
* Sensors are used to estimate relevant measures of performance & controllers that implement several policies.
* Controller uses feedback provided by sensors to stabilize the system.
* output is resource allocation to individual appl's.

Virtualization of x86 Architecture.

1. Ring deprivileging – VMMs force the OS. & appl^{n}s to run at a privilege level greater than 0.
2. Ring aliasing – a guest OS is forced to run at privilege level other than ~~sto~~ that it was initially designed for.
3. Address Space compression – VMM uses parts of guest addr. space to store several system data structures.
4. Non-faulting Access to privileged state – several stored instruction can be executed only at level 0 privilege because they operate on data structures that control CPU operation.
   * They fail silently when executed at privilege levels other ~~than~~ than 0..
5. Guest System Calls – cause transitions to / from privilege level 0. It must be emulated by VMM.
6. Interrupt virtualization – in response to physical interrupt, VMM generates "virtual interrupt" & delivers it later to the targed guest OS which can mask interrupts.
7. Access to hidden State – elements of the system state are hidden. There is no mechanism for saving & restoring hidden components when there is a context switch frome one VM to another VM.
8. Ring Compression – paging and segmentation protect VMM code from being overwritten by guest OS & applications.
   * Systems running in 64 ~~bit~~ mode can only use paging, but paging doesnot distinguish b/w privilege levels 0,1,& 2.
   * Thus guest OS must run at privilege level 3. (0/3/3 mode).