Homework 1

Karthik Maganahalli Prakash
B01099629

Q1 (20 points): Answer the following questions on metrics. Give exact values and show how you obtain your answer.

a) How much is 2^{12} in decimals?
   - ⇨ 2^{12}
   - ⇨ 2*2*2*2*2*2*2*2*2*2*2*2
   - ⇨ 4096

b) How much is $\log_2(2048)$ in decimals?
   - ⇨ $\log_2(2048) = \log_2(2^{11})$
   - ⇨ 11

c) How much is $\log_2(4 \times 2^{10} \times 2^{10} \times 4^5)$ in decimals?
   - ⇨ $\log_2(2^2 \times 2^{10} \times 2^{10} \times 2^{(2*5)})$
   - ⇨ $\log_2(2^2 \times 2^{10} \times 2^{10} \times 2^{(10)})$
   - ⇨ $\log_2(2^{32})$
   - ⇨ 32

d) 5Kbps of network bandwidth equals how many bits per second?
   - ⇨ Since 1 Kbps is equal to 1,000 bits per second
   - ⇨ 5Kbps=5×1,000=5,000bits per second

e) 4KiB of storage equals how many bytes?
   - ⇨ We know that: 1KiB=1,024 bytes
   - ⇨ 4×1,024 bytes
   - ⇨ 4,096 bytes

Q2: What are the following? Why are they important?

1. Instruction Set Architecture (ISA)
   - ⇨ A set of instructions that a processor can understand and execute, serving as the interface between software and hardware.
   - ⇨ Importance:
     - o Enables software-hardware interaction
     - o Performance optimization: Influences how efficiently processors execute programs.
2. System ISA
   - ⇨ Refers to the set of instructions used by the **operating system** and the **hardware** for tasks like managing resources, handling exceptions, and interacting with devices.
   - ⇨ Importance:
     - o Provides the instructions that the operating system needs to manage system-level tasks such as memory management, and process control.
       Example: (Hardware interrupts, direct memory access (DMA), or CPU power states.)

3. User ISA
   - ⇨ Refers to the set of instructions available to **user-level programs or applications**. These instructions are generally **restricted** to prevent direct access to critical hardware resources and are more focused on application execution.
   - ⇨ Importance:
     - o User ISA is essential for application developers because it provides a consistent interface to write software.
4. System Calls:
   - ⇨ It's a technique through which **user applications** or **application programs** request services from Operating Systems.
   - ⇨ Examples:
     - o Opening, Writing, Reading Files
     - o Multi-threading programming
     - o Accessing environment variables.
5. Application Binary Interface (ABI)
   - ⇨ Its interface between user programs or application programs and the Operating System converts high-level information to binary format (low-level or Machine level)

Q3: For each of the following virtualization categories, explain what interfaces are virtualized?

1. Process virtual machine:
   - ⇨ **Process Virtual Machine (PVM)** is an interface between individual processes and the operating system that allows a process to create its own virtualized environment. This environment is abstracted from the underlying hardware, enabling the process to execute as if it were running on a separate machine. When the process terminates or exits, this virtual environment is destroyed, and the resources it uses are reclaimed.
   - ⇨ **Example:** JVM, Python runtime environment.

2. System virtual machine:
   - ⇨ **System Virtual Machine (SVM)** virtualizes an entire system, including the operating system, and allows multiple guest OS instances (VMs) to run on top of a single physical host.
   - ⇨ Hypervisors like VMware ESXi, Microsoft Hyper-V, and KVM, where each VM runs its own guest OS (like Windows, Linux) on top of the hypervisor, and the guest OS interacts with virtualized hardware as though it were a physical machine.

Q4: Hypervisors often do everything that traditional operating systems do, such as memory management, CPU scheduling, etc. So, what is the one key aspect by which a hypervisor differs from a traditional Operating System?

- ⇨ A traditional operating system (OS) controls the hardware resources for a single operating system on a single machine, whereas a hypervisor mainly works by establishing an abstraction layer between the physical hardware and several virtual machines (VMs), enabling each VM to run its own operating system on the same physical hardware.

Q5: For system virtual machines, explain how virtual memory addresses are translated to physical addresses when

a. hardware supports EPT/NPT (extended/nested page tables)

- ⇨ **Level 1 Guest Page Table**
  - o The guest operating systems inside the virtual machine use its own memory management system. When a process in the guest OS accesses memory, it uses a Guest Virtual Address (GVA)
  - o The guest OS has its page tables to translate Guest Virtual address to Guest Physical address
- ⇨ **Level 2: Extended Page Table(EPT)**
  - o GPA is not actually the physical address of host machines instead it is an intermediate address that needs to be translated to a host physical address.
  - o The hypervisor maintains a set of page tables called EPT in Intel (NPT Nested pages tables), this table is used to translate GPA's to HPA's
  - o Translation is performed by CPU without involving the hypervisor.

(b) hardware only supports traditional (non-nested) page tables.

- ⇨ **Level 1 Guest Page Table**
  - o The guest operating systems inside the virtual machine use its own memory management system. When a process in the guest OS accesses memory, it uses a Guest Virtual Address (GVA)
  - o The guest OS has its page tables to translate Guest Virtual address to Guest Physical address
- ⇨ **Level 2:**
  - o Since the hardware does not support the nested page tables, the hypervisor translates from GPA to HPA
  - o The hypervisor maintains a shadow page table for each guest OS. The shadow page table combines the GVA-to-GPA translation (from the guest page tables) and the GPA-to-HPA translation (managed by the hypervisor) into a single set of page tables that the hardware can use directly.

Q6: When the total memory required by all guest VMs exceeds the host's physical RAM, a **hypervisor** employs which technique(s) to manage resources? Explain your answer.
A) Memory overcommitment
B) Ballooning
C) Swapping
D) Dynamic reallocation of cores

⇨ The correct techniques that are used to manage resources when VMs exceed the host physical RAM are

A. **Memory overcommitment:** Memory overcommitment allows the hypervisor to allocate more memory to VMs than the host physically has available. This is based on the assumption that not all VMs will use their allocated memory simultaneously. (The hypervisor monitors memory usage and relies on techniques like ballooning and swapping to reclaim memory when necessary.)

B. **Ballooning**: Memory ballooning is a virtualization technique that allows virtual machines (VMs) to give back unused memory to the host. This helps prevent overcommitting the host's memory. (Hypervisor uses a special driver (called a balloon driver) installed in the guest OS to reclaim memory from idle or less-active VMs)

C. **Swapping(Memory paging)**: Swapping involves moving inactive or less-used memory pages from RAM to disk (swap space) to free up physical memory for active VMs.

Q7: Compare and contrast traditional virtual memory with memory virtualization used in Virtual Machines (VMs).

Aspect	**Traditional Virtual Memory**	**Memory Virtualization in VMs**
Purpose	Manage memory for processes within a single OS.	Manage memory for multiple VMs on a single host.
Level of Abstraction	Virtualizes memory for processes.	Virtualizes memory for entire operating systems.
Address Spaces	Virtual addresses (per process) → Physical addresses.	Guest virtual addresses → Guest physical addresses → Host physical addresses.

Translation Mechanism	Page tables managed by the OS.	Nested page tables (EPT/NPT) or shadow page tables managed by the hypervisor.
Isolation	Isolates processes within a single OS.	Isolates entire VMs from each other.
Hardware Support	MMU in CPU handles address translation.	Requires hardware support (EPT/NPT) for efficient nested translation.
Overcommitment Handling	Uses swapping to disk (paging).	Uses ballooning, swapping, and memory overcommitment.
Transparency	Processes are unaware of virtual memory management.	Guest OS is unaware of memory virtualization.
Use Case	Single OS running multiple processes.	Multiple VMs running on a single host.

Q8: For intel VTx, why CPU Privilege Levels for Root Mode and Non-Root Mode have the same 4 levels (0, 1, 2, 3) design? Can we reduce the CPU Privilege Levels to just 2 levels (0, 3) for Non-Root Mode?

⇨ Although lowering the Non-Root Mode privilege levels to just two levels (Ring 0 and Ring 3) is technically feasible, doing so would interfere with existing software compatibility, limit flexibility, and possibly make the hypervisor more complex. In both Root Mode and Non-Root Mode, Intel VT-x maintains the 4-level privilege architecture to provide security, compatibility, and flexibility in virtualized settings. The necessity to maintain effective and safe virtualization while supporting a broad variety of operating systems and software is reflected in this design decision.

Q9: Briefly describe the life cycle of a VM.

⇨ The life cycle of a Virtual Machine (VM) consists of the following key stages:

1. **Creation**: The VM is created by the hypervisor, which allocates resources (CPU, memory, storage) and configures virtual hardware (e.g., virtual CPU, network interfaces).
2. **Provisioning**: The VM is provisioned with an operating system (OS) and necessary software, either from a template or through manual installation.
3. **Startup**: The VM is powered on, and the guest OS boots up, initializing its virtual hardware and services.
4. **Running**: The VM operates normally, executing applications and services. The hypervisor manages resource allocation and ensures isolation from other VMs.

5. **Suspension** (optional): The VM can be paused or suspended, saving its current state to disk for later resumption.
6. **Migration** (optional): The VM can be live-migrated to another host for load balancing, maintenance, or fault tolerance.
7. **Shutdown**: The VM is gracefully powered off, and its resources are released.
8. **Deletion**: The VM is removed, and its associated resources (e.g., disk files) are freed or archived.

This cycle repeats as needed for the VM's use case.