Virtualization – Quiz 2

Name: Jay Mansukh Bhorania

ID: 010831470

Guest VM Instruction:

0x1010: mov 0x1080, %rax ;mov content in address 0x1080 to rax

1. (2 points) Which addresses (if any) are checked in the TLB during this instruction, and what are the content of the TLB after the instruction (tag info not needed)?

Answer: a) 0x1000 is checked in virtual TLB and 0x12000 is checked in the processor's TLB during this instruction. **b)** VA 0x1000 to PA 0x12000 is present in virtual TLB and GPA 0x12000 to HPA 0x45000 is present in processor's TLB after the instruction.

2. (2 points) True or False? This instruction will cause a write to memory.

Answer: False. Instruction causes write (a/d bits)

3. (2 points) True or False? This instruction will cause a read from memory.

Answer: True.

4. (2 points) True or False? This instruction will cause a VM exit.

Answer: True. Instruction does not cause VM exit

5. (2 points) True or False? This instruction will cause a #GP (protection violation/ privilege violation).

Answer: False.

References:

1. 64-ia-32-architectures-software-developer-manual-325462 – Chapter 32
2. https://www.cs.cmu.edu/~dga/15-440/F11/lectures/vm-ucsd.pdf

Virtualization Quiz 4

Name: Jay Mansukh Bhorania

ID: 010831470

Q1. Describe how live migration of a virtual machine between hosts works. Be specific- more points will be awarded based on the completeness of your answer.

Answer: Live migration is a process of moving VMs among physical virtual host without any downtime. The main component of a VM is VM's storage and VM's configuration. Often the storage is located at different location and VM's configuration is running on host's processor and memory[2].

Following are the steps that needs to be followed:

- The host needs to create connection with destination host.
- Transfer the state to the destination host as fast ~~as~~ possible.
- Start guest on destination host.
- Now transfer the memory from source host to destination host and keep track of modified pages.
- Transfer the modified pages from the source host to destination host.
- Pause the VM and sync VM image on source and destination host.
- Transfer the traffic to destination host (broadcasting new NIC) and transfer network configuration.
- Shut down source VM and continue the working of guest on destination VM.
- If there is a failure in between any step, continue source host and restart the process of migration.

Q2. Describe which operations need to be intercepted for application virtualization solutions, and how this interception is performed. Be specific – more points will be awarded based on the completeness of your answer.

Answer: In application virtualization, each application has its own set of configurations and are executed in a way that each application sees only its own configurations. Hence host operating system configurations remain unaltered.

Following are the operations that needs to be intercepted:

Create file, delete file, Open file, Create User, Delete User, set system time, Query system time, Open network connection, Load Device Driver, Create new process, List files in folder, Reboot system, Terminate process, Set file permissions and query file attributes [1].

This interception can be done in following ways:

- System call interception- system calls are hooked and intercepted before the underlying kernel can process them and can be handled in any manner we want. This is generally done in the kernel and requires some help from OS.
- Hook a custom driver into the application that will intercept the calls before the control is passed to middleware.
- Another way is that a master process creates a child process in a suspended state. The master process looks through child's memory space to find all the routines that needs to be monitored and rewrites these in place with its own engine [1].

References:

1. CMPE 283 lecture slides – 283_11_appvirt.pdf
2. https://en.wikipedia.org/wiki/Live_migration

Cmpe 283 – Quiz 1

Name: Akshay Mishra

SJSU ID : 011476673

```
701: 55 push %rbp
702: 41 54 push %r12
704: 41 55 push %r13
706: 49 89 fd mov %rdi,%r13
709: 65 48 8b 0c 25 08 00 00 mov %gs:0x8,%rcx
710: 4d 85 ed test %r13,%r13
713: 0f 22 d8 mov %rax,%cr3
716: e7 32 out %eax,$0x32
718: 0d 00 08 00 00 or $0x800,%eax
71d: 0f 30 wrmsr
```

For each instruction, state if an exit is possible for any reason, and give the type of exit and reason if so. You may assume that the page containing the mystery function (eg, the page containing addresses 0x0–0x1000) is present in memory (with all page tables properly configured). The mystery function is running in CPL0. (1 point awarded for each correct instruction)

Instruction	Exit possible (Y/N)?	If exit possible, which exit, and under what circumstances would exit occur?
701	Y	If stack is not present, that will cause page fault
702	Y	If stack is not present, that will cause page fault
704	Y	If stack is not present, that will cause page fault
706	N	
709	Y	Exit can be caused by page fault
710	N	
713	Y	It will not cause the exit while in non-root operation, if CR3-target count is n, only first n CR3-target values are considered; if CR3-target count is 0 then it will cause exit
716	Y	It will cause the conditional exit of the VM

718	N	
71d	Y	It contains 1 bit for each of the MSRs in the address range of 00000000H to 00001FFH. And each bit will indicate if executing WRMSR for that MSR will cause a VM exit or not.

Note: Please don't delete anything, if you find something not correct highlight it with RED. True/False whichever not sure have highlighted in Orange. Please confirm

14. Describe the processor and VMM's behavior when a guest VM executes "CPUID".

→ When guest VM executes "CPUID" the VM exit take place and control goes to the hypervisor. VMM records information about the cause of the VM in the VM-exit information fields and update VM-entry control fields.

→ VMM saves the processor state in the guest state area of VMCS. This includes control registers, debug registers, segment registers, RIP, RSP, RFLAGS etc.

→ Save MSRs in the VM-exit MSR-store area. They are used to control and report on processor performance

→ Load processor state based on host state area and some VM-exit controls. This includes host control register, debug register, MSRs host table & descriptor-table register, RIP, RSP, RFLAGS, page-directory pointer table entries.

→ Load MSRs from the VM-exit MSR load ~~area~~ area.

→ Then VMM look for EAX and see which function was called.

→ Every exit instruction has a handler. Here handler_cpuid() switch case is executed from

```
switch(exit reason)
{ case CPUID: =

}
```

→ If its 0, return a string to GPRs. If its 1, we call its handler.

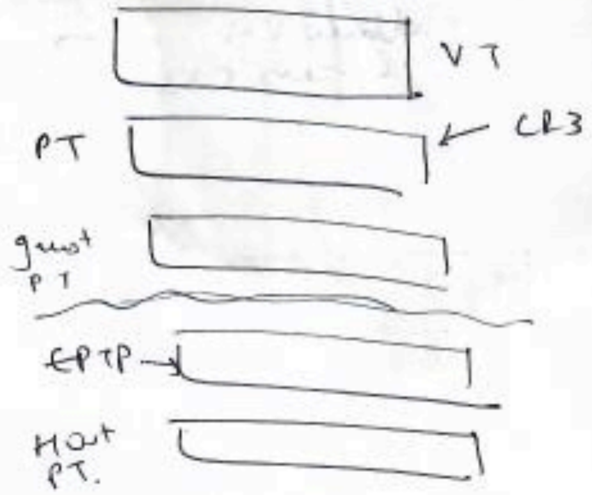
12

→ If it's 1, we do real CPUID and take away the bits we don't want the guest OS to see and put them back and then return.

·. of CR3 loaded:
simultaneously

→ multiple threads may have same PT.
→ multiple processor may have same PT.

Benefit	Drawback
→ allow VM to access CR3	→ many trips to memory
	→ more TLB pressure

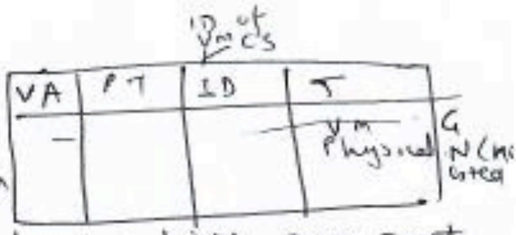


$\boxed{\text{SDM V3: 32.1, 32.3}}$

EPTP violation → page fault

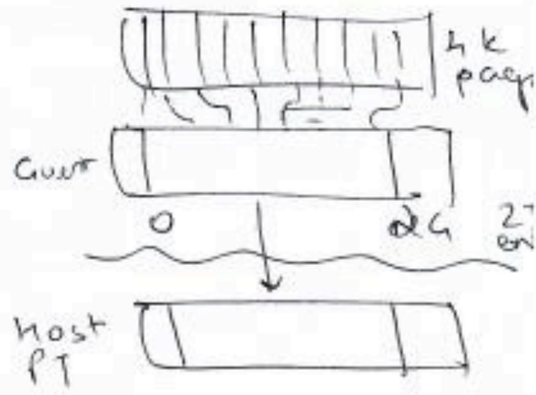
→ PT = no. of ~~tables~~ processes
→ No. of Host physical table
= no. of. VM.

→ 1G largest page size for This processor family.



some entries flush vmswitch while independent entries are not flushed.

→ Intel introduced large pages bcz of hypervisor (he feels)


4k
page
Guest
0
2G
2
en
Host PT