77

CMPE283 – Virtualization
Midterm Exam

Name: SWATHI KODURI

This exam is governed by the rules posted in the class Canvas discussions forum.

Select / provide the best answer for each question.

1. (20 points) Give an example of two x86 instructions that are sensitive but not privileged and explain why the instructions cause problems if allowed to execute without a VMM.

20

1. POPF : This stands for POP Flags off Stack. It removes the word from top of stack and increments stack pointer by 2 and stores the value in lower 16 bits EFLAGS register. This instruction allows values in the EFLAGS register to be changed. In host OS, this is performed under privilege level 0. Hence when run in virtual environment, it doesn't cause exception. This can change flags that are under control of host OS.

2. SGDT : It stands for Store Global Descriptor Table Register. This register copies the contents of GDT into the memory specified by the operand in 6 byte memory location. This instruction is sensitive and has the capability to read and change sensitive registers or memory locations. Hence, this instruction should be handled by VMM.

5

2. (5 points) False True or False – All VMEXITs require the VMM to emulate at least one guest VM instruction.

3. (5 points) How many levels of the page directory/page table hierarchy does the does the x86 processor traverse when translating a virtual address in 64 bit mode?

5

A. Always 4
B. Always 3
C. Sometimes less than 4
D. None of the above

5

4. (5 points) True True or False – The processor must always reload CR3 when a VMEXIT occurs.

35

5. (15 points) Describe what happens in the processor and VMM when a VM exit occurs. Be specific and thorough.

When a VM exit occurs, the control goes to the hypervisor and guest state will be saved so that we can continue from where we left off when we enter the VM again. As soon as exit takes place, the host state area and guest state area are swapped. Next, we find out which instruction or event has caused the exit. For example, CPUID. Every VMM has a handler which handles the exits. Suppose exit is caused due to CPU_ID, handler_cpuid() switch case is executed.

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

We provide fake CPU information and after exit is handled, we enter guest state and resume our operations by incrementing the Instruction Pointer and perform the next instruction.

15

6. (5 points) False True or False – All VMEXITs occur due to the guest VM executing a prohibited or sensitive instruction.

7. (3 points for each correct answer) For each of the following exit types, provide an instruction that would cause the exit, along with any assumptions or conditions required for the exit to happen. If the exit type is not caused by an instruction, answer "No instruction", but still provide any assumptions required for the exit to happen.

A. VMREAD (Intel VMX exit reason 23)
B. Exception or non-maskable interrupt (Intel VMX exit reason 0)
C. CPUID (Intel VMX exit reason 10)
D. I/O Instruction (Intel VMX exit reason 30)
E. Control register access (Intel VMX exit reason 28)

[P.T.O]

E) Control Register Access: SGDT, SLDT, SIDT, MSR access

3 If the VM executes the following instruction, exit occurs. MOV %RAX, %CR3

CR3 could contain other important information.

A) VMREAD:

3 Exit occurs because VMREAD instruction can read contents of VMCS like GSA and HSA. The GSA stores the state of CR0, CR3, CR4, RIP, etc which are sensitive information the VM should not be given access to sensitive registers.

B) Exception or Non-Maskable Interrupt: #PF, HLT, JMP, INT

3 Always turn on interrupt handling in VM. JMP could cause exit if privilege or permission is not present.

C) CPUID: CPUID

Always exit for eg: turn off thermal monitoring, etc.

D) I/O instruction:

Hardware Input and Output devices should be emulated. By using emulated system calls and pipes, exit to VMM and get actual input value from devices.

NSN?

Virtualization – O...

Name: Sagar Bhoi...

...15

MSR access
ction, exit

...ion.

...nation.

...er
...cti
...n

3 (10 points) Describe how global pages are used and why they are beneficial to an operating system.

When multiple processes are running, context switching takes place to allot resources to different processes. When a process tries to access memory, paging is used. In paging, we map physical addresses to virtual addresses.

CR3 is used to point to the base physical address of the process page table. ~~Modify~~ Modifying CR3 is expensive because we ~~had~~ have to walk the page table and directory. ~~So~~ So, a cache has been incorporated to avoid page walking The Translation Lookup Buffer (TLB).

Global Pages:

The virtual addresses computed by CR3 are usually split between Kernel space mappings and user space mappings. The Kernel space mappings are identical across different processes, so they are stored seperately in Global Pages. These Global pages do not get flushed because they can be used by different processes. This avoids flushing caches and unnecessary page walks because page walking is expensive and consumes lots of CPU time.

10

Virtualization – Quiz 1

Name: **Sagar Bholte**

SID: **010034115**

The list of instructions in the x86 architecture that are not virtualizable in certain circumstances before the advent of hardware assisted virtualization are as follows:

A] Sensitive Register Instructions:

1] SGDT:

It stands for Store Global Descriptor Table Register. The Store Global Descriptor register copies the contents of the global descriptor table register in the memory specified by the operand in 6 byte memory location. This instruction is sensitive and has the capability to read and change sensitive registers or memory locations. Hence this instruction should be handled.

2] SIDT:

Store Interrupt Descriptor Table register stores the contents of the Interrupt Descriptor Table Register (IDTR) and copies them to memory specified in the destination operand in 6 byte memory location. The Intel processor has only one of the LDTR, GDTR and IDTR register and hence when the OS tries to access two of the registers at the same time it causes error. When the OS in the VM tries to write on one of the above registers whose contents apply to the host OS it causes a trap.

3] SLDT:

Store Local Descriptor Table (LDT) register copies the contents of the local descriptor table register in a 16 or 32-bit general-purpose register or memory specified by the operand. This instruction is sensitive and has the capability to read and change sensitive registers or memory locations. Hence this instruction should be handled.

4] SMSW:

Store Machine Status Word, this instruction stores the machine status word into a general purpose register or memory location. Bits 6 to 15 of CR0 are reserved and are not supposed to be modified. Bit 0 to 5 are unprivileged. If the VMOS checked the MSW to see if it was in real mode, it would incorrectly see that the PE bit is set. This means that the machine is in protected mode. If the VMOS halts or shuts down if in protected mode, it will not be able to run successfully.

5] POPF:

It stands for Pop Flags off Stack. It pops the word from top of the stack, and increment the stack pointer by 2 and stores the value in lower 16 bit EFLAGS register. The POPF instruction allows values in the EFLAGS register to be changed. In the host OS the operation is performed under the privilege level 0 except for the VM, VIP and VIF flags. Hence when run in virtual environment the privilege mode being less doesn't cause exception. And also the modification of certain flags can change flags that are under control of the Operating system.

6] PUSHF:

It stands for Push Flags onto Stack. The PUSHF instruction pushes the lower 16 bits of the EFLAGS register onto the stack and decrements the stack pointer by 2. Pushing the EFLAGS to stack can make them visible and hence when examined can cause the VM to know its mode can cause errors.

B] Protection System Instructions:

1] LAR:

The Load Access Rights instruction loads access rights from a segment descriptor into a general purpose register and sets the ZF flag in the flag register. The source operand can be register or memory location. The LAR command can only be executed in protected mode and IA – 32e mode. Hence this command should be handled.

2] *LSL:*

It stands for Load Segment Limit. It loads the unscrambled segment limit from the segment descriptor specified with the source operand into the destination operand and sets the ZF flag in the EFLAGS register similar to the LAR command. And the command runs in protected mode and IA -32e mode.

3] *VERR:*

Verify segment for reading. Verifies whether the code or data segment specified with the source operand is readable from the current privilege level. The source operand is a 16-bit register or a memory location that contains the segment selector for the segment to be verified. For the VERR command to set the ZF flag the segment should be readable and if the segment is not present in the virtual machines memory bound then it can create a fault. The VERR instruction runs in the privileged mode and since most of the VM's don't run on privileged mode this instruction should be handled.

4] VERW:

Verify segment for writing. Verifies whether the code or data segment specified with the source operand is writable from the current privilege level. The source operand is a 16-bit register or a memory location that contains the segment selector for the segment to be verified. For the VERR command to set the ZF flag the segment should be writable and if the segment is not present in the virtual machines memory bound then it can create a fault. The VERW instruction runs in the privileged mode and since most of the VM's don't run on privileged mode this instruction should be handled.

5] POP:

The POP instruction loads a value from top of the stack to a general purpose register, memory location or segment register. Since the CS register contains the CPL it cannot be used to load the CS register. Since it is operating in the Virtual machine the privilege level would be lower than the machines and this could cause the protection exception. Hence this instruction should be handled.

6] PUSH:

The PUSH instruction allows a general purpose register, memory location or a segment register values to be pushed onto the stack. A process running in CPL 0 can push the CS register to the stack and when finding that the CPL is not 0, the process comes to halt. Hence this instruction should be handled.

7] CALL:

The CALL instruction saves procedure linking information to the stack and branches to the procedure given in its destination operand. Call instruction has can be classified as near calls, far calls to the same privilege level, far calls to a different privilege level and task switches. Near calls and far calls to the same privilege level are not a problem for virtualization. Task switches and far calls to different privilege levels are problems because they involve the CPL, DPL and RPL. If a far call is executed to a different privilege level, the code segment for the procedure being accessed has to be accessed through a call gate. A task uses a different stack for every privilege level. Since the VM normally operates at higher privilege level, these checks will not work correctly when a VMOS tries to access call gates or task gates at CPL 0.

8] JMP:

JMP instruction can perform an unconditional jump. The JMP instruction is similar to the CALL instruction in both the way that it executes and the reasons it prevents virtualization. Such an instruction transfers flow from changing the instruction pointer register. This can cause issue when the operation is tried to perform in privileged mode.

9] INT n:

The INT instruction performs similar to the CALL instruction. The INT n instruction performs a call to the interrupt or exception handler specified by n. It specifies an interrupt vector number 0 to 255 encoded as unsigned 8 bit intermediate value. INT pushes the EFLAGS register onto the stack before pushing the return address. The INT instruction references the protection system many times during its execution. Hence it has to be handled.

10] RET:

The RET instruction has the opposite effect of the CALL instruction. It transfers program control to a return address that is placed on the stack. The RET instruction can be used for three different types of returns: near, far, and inter privilege level returns. The RET examines the privilege levels and access rights of the code and stack segments that are being returned to determine if the operation should be allowed. The DS, ES, FS, and GS segment registers are cleared by the RET instruction if they refer to segments that cannot be accessed by the new privilege level. Since having higher than 0 privilege level can cause issues with RET.

11] STR:

The Store Task Register instruction stores the segment sector from the task register into general purpose register or memory location. The segment can store the segment sector to general purpose and memory locations. Since the contents of the memory location can be examined and there is the chance for the VMOS finds the privilege level to be more than 0 while expecting 0. This can cause problems.

12] MOV:

Two variants of the MOV instruction prevent Intel processor virtualization. These are the two MOV instructions that load and store control registers. The MOV opcode that stores segment registers allows all six of the segment registers to be stored to either a general purpose register or to a memory location. This is a problem because the CS and SS registers both contain the CPL in bits 0 and 1. And the bits can be examined when moved to memory location and halt the system as the VM expects the privilege level to 0 but it would be different.

Virtualization – Quiz 3

Name: **Sagar Bhoite**

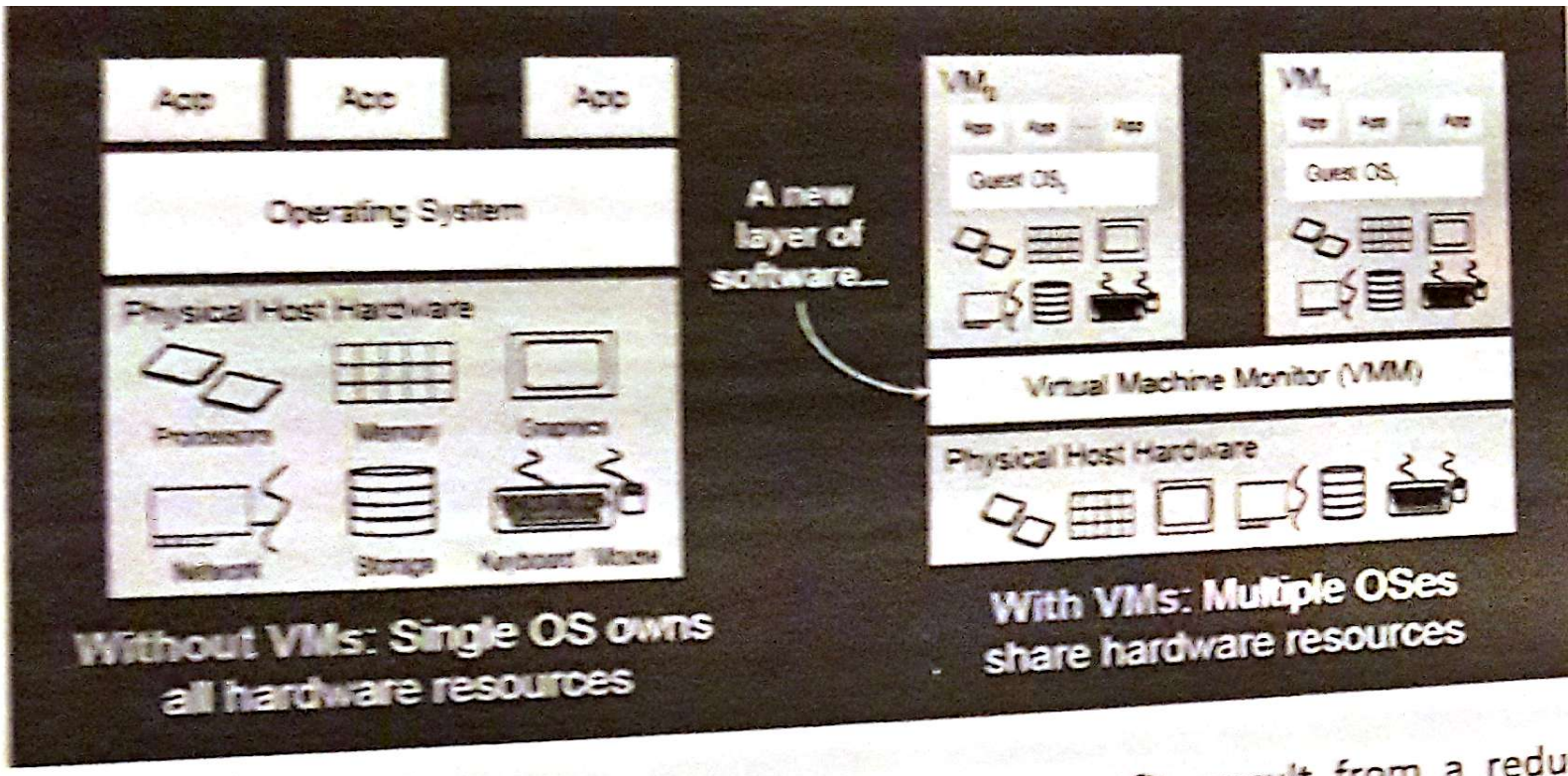
SID: **010034115**

Read Time Stamp Counter (RDTSC) loads the current value of the processor's time-stamp counter into the EDX, EAX registers. The EDX register is loaded with the high-order 32 bits and the EAX register is loaded with the low-order 32 bits. The time stamp disable (TSD) flag in register CR4 restricts the use of the RDTSC instruction as follows. When the flag is clear, the RDTSC instruction can be executed at any privilege level; when the flag is set, the instruction can only be executed at privilege level 0.

The processor monotonically increments the time-stamp counter MSR every clock cycle and resets it to 0 whenever the processor is reset. If the Virtual machine has a program that calculates the time taken by individual function then the expected result and the actual result will differ because the clock of the virtual machine is running at a different rate from the actual time. So most common use is to have two RDTSC instructions with a small amount of code between them, taking the difference of the times as the elapsed time (number of cycles) for the code sequence.

The RDTSC instruction is not a serializing instruction. It does not necessarily wait until all previous instructions have been executed before reading the counter. Similarly, subsequent instructions may begin execution before the read operation is performed.

RDTSC is use to get fine-grained timing information, where the overhead of a virtualization trap would be quite significant. Hence it needs to be virtualized.



ons:

nsolidation & containment: Server consolidation benefits result from a redu
systems and related recurring costs (power, cooling, rack space, etc.). Server c
l approach to workload virtualization, whereby new projects are provisioned w
physical servers, thus deferring hardware purchases.

velopment optimization: Rapidly provisioning test & development servers by re

Continuity: Reducing the cost and complexity of business continuity (high ava
entire systems into single files that can be replicated

In **real mode**, addresses are generated by adding an address offset to the value of a segment register shifted left four bits. As the segment register and address offset are 16 bits long this results in a 20-bit address. This is the origin of the one megabyte (2^20) limit in real mode.

In **protected mode**, the segment registers contain an index into a table of segment descriptors. Each segment descriptor contains the start address of the segment, to which the offset is added to generate the address. In addition, the segment

10 minutes

Score

6 out of 10

answers are hidden.

quiz: 6 out of 10
ar 14 at 6:44pm
took 10 minutes.

Question 1

2 / 2 pts

1. Which of the following instructions would a hypervisor running on an Intel x86 CPU use to enter a virtual machine?

- VMLAUNCH
- VMSTART
- World Switch
- VMENTER

Question 2

2 / 2 pts

2. Which of the following statements most accurately represents the treatment of the VMCS content?

- The hypervisor and processor both play roles in populating VMCS content
- Neither the hypervisor nor the processor are responsible for populating VMCS content
- The processor is solely responsible for populating VMCS content
- The hypervisor is solely responsible for populating VMCS content

Question 3

0 / 2 pts

3. The CR3 control register stores the location of the page location (eg, page table structures) currently in use by the processor?

- True
- **False**

Incorrect

Question 5

0 / 2 pts

5. Which answer best describes the responsibilities of the hypervisor author when using VMX controls?

- The hypervisor author is required to handle all exits requested by the selected controls
- The hypervisor author must enable the "exit on HLT" control, unconditionally
- **The hypervisor author can select any set of controls desired, regardless of which physical CPU is being used**
- The hypervisor author must enable all controls provided by the CPU in use

GUEST VM PAGE TABLE				
VA	PA	Page Metadata Bits		
		P	A	D
0x1000	0x12000	P	X	
0x2000	0x14000	P	X	
0x4000	0x13000	P	X	X
0x5000	0x9000	P	X	X
0x6000	0x6000	P	X	X
0x7000	0x7000	P	X	

NESTED PAGE TABLE				
GPA	HPA	Page Metadata Bits		
		P	A	D
0x6000	0x40000	P	X	X
0x7000	0x4A000	P	X	
0x9000	0x44000	P	X	X
0x12000	0x45000	P	X	
0x13000	0x51000	P	X	X
0x14000	0x55000	P	X	

Question: Consider the following sequence of guest VM instructions:

```
149c <mystery>:
149c:       0f 32                                   rdmsr
149e:       89 04 25 54 41 01 00                    mov    %eax,0x14154
14a5:       5a                                      pop    %rdx
14a6:       59                                      pop    %rcx
14a7:       0f 01 04 25 38 41 01 00                 sgdtl  0x14138
14af:       0f 01 0c 25 28 41 01 00                 sidtl  0x14128
14b7:       0f 00 04 25 48 41 01 00                 sldtl  0x14148
14bf:       0f 00 0c 25 52 41 01 00                 strl   0x14152
14c7:       0f 01 3c 25 00 10 02 00                 invlpg 0x21000
14cf:       c3                                      retq
```

For each instruction, state if an exit is possible for any reason, and give the type of exit an reason if so. You may assume the following:

- The page containing the mystery function (eg, the page containing addresses 0x100 – 0x2000), is present in memory with RX permission (Read + Execute)
- The page 0x14000-0x15000 is present in memory with R/W permission
- The page 0x21000-0x22000 is marked as "not present" in the current page table.
- The mystery function is running in CPL0.
- The paging structures for the above tables are properly present in memory

Instruction	Exit possible (Y/N)	If exit possible, which exit, and under what circumstances would exit occur
149c	Yes	This contains one bit for each MSR address in the range 00000000H to 00001FFFH. The bit determines whether an execution of RDMSR applied to that MSR causes a VM exit
149e	No	
14a5	Yes?	#PF: Page Fault Exception. Stack might not be present) /Accessing Memory
14a6	Yes?	#PF: Page Fault Exception. Stack might not be present) /Accessing Memory
14a7	Yes	instruction cause VM exits if the "descriptor-table exiting" VM-execution control is 1.
14af	Yes	instruction cause VM exits if the "descriptor-table exiting" VM-execution control is 1.
14b7	Yes	instruction cause VM exits if the "descriptor-table exiting" VM-execution control is 1

14bf	Yes	instruction cause VM exits if the "descriptor-table exiting" VM-execution control is 1.
14c7	Yes	The INVLPG instruction causes a VM exit if the "INVLPG exiting" VM-execution control is 1.
14cf	Yes	If return instruction pointer is not within the segment limit page fault exits.

Question: Describe how live migration of a virtual machine between hosts works.

Answer: There are two ways through which live migration between hosts can work:

1) Managed Migration : - This type of migration is performed by daemon thread running in the management of the source and destination VM. It is responsible for creating a new VM on the destination machine and co-ordinates transfer of live system over the network.

- The control software performs copying rounds which runs the complete scan of VM's memory pages. Although all the pages are transferred in this the first round, in subsequent rounds it is restricted to pages that were previously dirtied. Page tables managed by guest OS are exposed to real physical address to fill TLB so they need not be mapped.

- Xen inserts shadow page table underneath the running OS in the dirtied log pages. And these shadow tables are populated on demand by translating sections of guest page table. As translation is simple for dirty logging, it initially reads only mappings and if the guest tries to modifies it the resulting page fault is trapped by Xen.

- When the bitmap is copied to the control software, the xen's bitmap is cleared and shadow page tables are destroyed. When it determines that pre-copy phase is no longer valid the OS sends a control message requesting to suspend itself. Xen informs the control software and the dirty bitmap is scanned one last time for remaining inconsistent memory pages.

- Once the final information is received at the destination, the VM state on the source machine is discarded. Control software on the destination machine scans the memory map and rewrites the guest page table to reflect the addresses of the memory pages that it has been allocated. Execution is then resumed by starting the new VM at old VM checkpoint.

- The OS then restarts its device drivers and updates its notion of wallclock time. As the transfer of pages is OS – agnostic any guest OS is supported which required by a small virtualized stub to handle resumption.

2) Self Migration: - Self migration places the majority of the implementation within the migration of OS. In this design no modifications are needed either to Xen or to the management software running on the source machine, although a migration stub must run on the destination machine to listen for incoming migration requests, create an appropriate empty VM, and receive the migrated system state.

Self migration is much more harder because the OS must continue to run in order to transfer its final state. So in order to overcome this difficulty, we logically check point the OS on entry to a final two-stage stop and copy phase. Here, the 1st stage disables all OS

activity except migration and then performs final scan of dirty bitmap, which then clears the appropriate bit of each page transferred. Finally all the pages still marked dirty are copied to a shadow buffer. The final stage then transfers the contents of the shadow buffer where page updates are ignored during transfer.

Question: Describe which operations need to be intercepted for application virtualization solutions, and how this interception is performed.

Answer: Following operations need to be intercepted for application virtualization solutions:

- Create File
- Open File
- Delete File
- Create User
- Delete User
- Query System Time
- Set System Time
- Open Network Connection
- Load Device Driver
- Create New Process
- List Files in Folder
- Reboot System
- Terminate System
- Set File Permissions
- Query File Attributes

There are two ways by which interception is performed:

1) First approach is system call interception where calls like (open, create, delete file) are hooked and intercepted before the underlying kernel actually process them.
   - So the system intercepts the call and examines the information for resources requested.
   - Then it either passes the request to the OS or changes the namespace/ID in order to "segregate the resource" before passing the new request to the underlying OS.

2) Second Approach is Process Injection-The whole process is explained as follows:
   - A master or launch process creates a child process in suspended state.
   - Before the process starts, the master process "reaches into" the child's memory space.
   - The parent process would then scan the child's memory space and look for the signature of routines that needs monitoring and finally rewrites those in place with call to its own engine.

CMPE283 – Virtualization
Sample Exam

Name:_____

You may consult the Intel SDM reference guide during this exam, if needed.

Select the best answer for each question numbered 1-5.

1. _____ Under what circumstances will a processor wake from a HLT instruction?

A. A non-masked interrupt occurs
B. A system call occurs
C. A process context switch occurs
D. Either B or C can cause a wake
E. None of A, B, or C

Updated Question: In a non-virtualized environment using an intel cpu newer than 80386, the "move to CR3" instruction flushes which of the following?

2. _____ The "move to CR3" instruction flushes which of the following?

A. All non-global translations from the processor's TLB
B. All translations from the processor's TLB
C. All non-global translations from all processors' TLBs
D. All translations from all processors' TLBs.
E. None of the above

3. _____ Which of the following operations does the system BIOS typically perform during startup?

A. Configuring paging
B. Determining process scheduling characteristics
C. Booting secondary processors
D. None of A, B, or C
E. Answers A, B, and C are all operations performed by the BIOS during startup

4. True _____ True or False? VMX root mode provides the virtualization systems architect with the capability to detect previously non trappable sensitive instructions?

5. False _____ True or False? Kernel area address translations present in a process' page directory (page tables) must be global translations?

Consider the following instructions.

```
(A)  SUB  %RAX,     $0x28     # subtract 0x28 (hex) from the value in the RAX register
(B)  MOV  %RAX,     %CR3      # move the content of the RAX register to the CR3 register
(C)  LGDT *%RAX               # load the GDT register with the content of the memory
                                location pointed to by RAX
(D)  VMXON     *%RAX          # enable VMX operation on the current CPU
(E)  SGDT *%RAX               # store the content of the GDT register to the memory
                                location pointed to by RAX
(F)  CPUID                    # gather CPU information
(G)  MOV  %RAX,     mem_loc   # move the content of the RAX register to mem_loc
```

6. Which instructions from the above set A-G, if any, may fault with a protection violation? Under what circumstances would each fault (if any)? in a non virtualized environment

B => if current privilege is greater than 0.
C => if current privilege is greater than 0.
=> memory location pointed is not accessible (read-only)
D => if current privilege is greater than 0.
=> memory location pointed is not accessible (read-only)
G => memory location pointed is not accessible (read-only)

Not in midterm

7. Which instructions from the above set A-G, if any, should a VMX-enabled VMM intercept while running guest VM code? Why?

C => yes if shadow paging, no if nested paging.
D => yes, must exit
E => yes, must exit
F => yes, must exit

8. Which instructions from the above set A-G are sensitive but not trappable instructions?

E

Consider again the following instruction:

MOV %RAX, mem_loc
64 bits

You may assume the following:

- The page containing 'mem_loc' is regular memory and is marked 'present' in the current page table
- The page containing the MOV instruction and the page containing 'mem_loc' are defined as CPL=3
- No other privilege or page protection violations occur during the MOV operation
- Nothing is previously cached in memory
- This is the first time the page containing 'mem_loc' is being accessed
- All the paging and MMU structures are fully present and resident in memory
- The operation is occurring in a non-virtualized environment.

not in mid-term

9. What is the maximum number of memory accesses that might occur to complete the operation? Partial credit given for partially-correct results, provided an explanation is given.

18

BONUS CHALLENGE QUESTION:

10. In the scenario described in question 9, does your answer change if the assumption that the paging and MMU structures are fully present and resident in memory is removed? If so, why, and what is the new answer?

arbitarily many

Mid term - 2

Sample mid-term. Its not a trick question.

1. *Prior to Hardware assited virtualization – could you run a virtualization on x86*
    a. *False. It was done by VMware*
2. *A.*
3. *B Lazy switch – leave context switch ...*
4. *A. – virtual space is always 32 bits. For 64-bit – Not enough information provided.*
5. *C . 32-bit mde, the right ans is D*
6. *True.*
7. *True.*
8. *A. Intrupt Descipter table.*
9. *D*
10. *B. Store full state GSA and RSA*
11. *Give example. Enable "must enable the controls" eg CPUID. Compatible with all CPU , so use old controls*
    a. *Nested paging is advanced.*
12. *CPUID – Always going to exit. Hyperviser main loop. Emulate CPUID eg turn off thermal monitoring. Advanced RIP.*

Mid term -3

1. *VMM hide / mask CPU features '*

5. A

6. True

7. D

9. A, D

11. Global Pages are used to map Kernal address base. Less overhead in flushing TBL.

Sample exam

Sample Quiz

1,. A; 2. A; 3. D; 4. True; 5. False;

5. Global translations is not must.

6. Protection violation #GP.

Page fault, previledgs level zero

B, E,

Anytime memory is touched, Protection violation can occur

7. eg nested paging. DEF . VMXON and CPUID cannt be turned off
8.
9. 20+
10. Page fault. Unknown access

```
Populate VMCS(hs, gs, ctrls)
While(1) {
        Select VMCS/VCPU
        Vmlaunch/resume

        ? which exit
        switch(exit reason){
                case HLT:

                case CPUID:

        }

        .
        .
        .
        events – check for panding interrupt
}
```

Stop on exit i.e. ctnrls
Min # of controls – 6-7
Max # cntrl - 61
Average - 40

Overhead in RED. Therefore, the fastest hyperviser, reduce the number of exit and reduce the RED.

283: Week 6 Contd..

KVM is the hypervisor that is in Linux source code

Every VMM has a handler that handles the exits

Processor based controls control things inside the CPU

Pin based controls control things outside the process. Like I/O events, like interrupts

VMCs has a GSA and a HSA

1. Before a VM is started populate the VMCS. VMCS(hs, gs, controls)
2. While(1)
    1. (Select VMCS/VPCU) (based on scheduling algorithm)
    2. VM Launch / VM Resume. What process is chosen to run in the VM.
    3. An exit has stopped the process (Reduce the number of exits so that the HV can use time effectively. There is an overhead in all the time that we use to handle each kind of exit)
    4. Which Exit?
    5. Check the exit that stopped a process: Switch (exit reason)
        1. give each exit something to do
            1. kvm has about 40 diff types of exit
            2. example: handler_cpuid()
        2. events (emulation for all the interrupts). Event Injection
    6. Advance the RIP and perform the next instruction.

How to channel the interrupts into the VM?

We just use an emulated version of the device (example: sound card) in the VM.

Something like a virtual sound card. Look at the Guest VM memory, take it to the main memory, give it to the actual sound card, play the sound, give response to the guest VM.

What happens if guest VM did While (1);

You're screwed unless your interrupt exiting is turned on. Always, Always, Turn on Interrupt Exiting.

What happens when there is an exit:

10/18/20

Answered above in the flow.

How many exits are there?

Around 40.

How many exits should you handle?

Architecture says you must handle 6 or 7 exits

What happens to RIP?

Incremented. But in certain exits you do not increment the RIP.

How to shut down VM?

While ("live" cpus/vmcs) !shutdown)

Note: Send answer to professor. Free evaluation!