CMPE 283 - Assignment 2-3

Name : Mrugesh Jayeshkumar Master

Answer 1.

I implemented code for 0x4ffffffe and 0x4fffffc to calculate the CPU Cycles for exits. I used rdtsc() to store time stamp counter values at the start and end of vmx_handle_exits function in vmx.c. Created atomic64_t variables to store the cpu cycles and pass it to cpuid.c. I created conditions to check if eax is 0x4ffffffe or 0x4fffffc and store required values in ebx and ecx.

Answer 2. Steps:

vmx.c :
1. Created 3 uint64_t variables start_time, end_time and time_diff to store rdtsc().
2. Created int variable to store return of kvm_vmx_exit_handlers.
4. Created extern atomic64_t time_spent variable and used atomic functions to store cpu cycles and send it to cpuid.c
5. start_time = rdtsc() at the start of vmx_handle_exits().
6. After kvm_vmx_exit_handlers is finished, end_time=rdtsc().
7. If exit_reason is between 0 to 69, time_diff = start_time - end_time.
8. Set return of vmx_handle_exit to the variable created in step 2.

cpuid.c :
1. Created atomic64_t variable time_spent to get cpu cycles from vmx.c
2. Used exit_valid array created by Sarthak to store cpu_cycles in ebx and ecx only for valid exits.
3. EXPORTED time_spent variable to make it visible in vmx.c.
4. If eax=0x4ffffffe, store total time_spent for all exits in ebx and ecx.
5. Looping over all the exit_reasons in the array exits_valid and adding them to get total cpu cycles spent for all exits.
6. Store low_bits of time_spent in ebx and high_bits in ecx.
7. If eax=0x4fffffc, get cpu cycles for exit reason specified in ecx and store it in ebx and ecx.
8. For exit_reason specified in ecx, ebx=low_bits of time_spent and ecx=high_bits of time_spent.

Testing
1. Encountered version magic error while trying to modprobe kvm.
2. Recompiling only kvm modules with make -C /lib/modules/5.6.0-rc2-00070-g99a3a1d6faa2-dirty/build M=$(pwd) modules to get rid of version magic error and to avoid recompiling all the modules.
3. modprobe and modprobe -r did not successfully always reload the drivers.
4. Used sudo rmmod and sudo insmod to remove and reload the drivers.
5. Installed qemu and created VM using qemu command line.
6. Created qemu bash script to start the VM using 2 vCPUs and 2048 Memory.
7. VMOS was Arch Linux, so had to create OVMF configuration to load UEFI BIOS.
8. Encountered multiple Kernel Panic issues due to incorrect implementation of atomic variables for counting cpu cycles.
9. VM did not have network connection so copied CPUID from host OS to VM OS.
10. Tested CPUID with leaf 0x4ffffffe to get cpu cycles for all exits.
11. CPU Cycles did not exceed 32bit limit, hence high 32 bits were always 0.
12. Tested CPUID with leaf 0x4fffffc and ecx = exit reasons (10,48,30) to confirm logic works fine.
13. Committed code to github repo.

Answer 3:

With subsequent runs of CPUID, the no of exits for each reason was decreasing. For instance, for EPT violation, the frequency of exits is decreasing with each execution of CPUID.

		1st run	2nd run	3rd run	4th run	5th run	2nd – 1st	3rd – 2nd	4th – 3rd	5th – 4th
48	**EPT Violation**	45946	63368	77797	83690	91641	17422	14429	5893	7951

HLT(12), IO Instruction(30), WRMSR(32) and EPT Violation(48) account for most number of VM Exits.

1st run of CPUID shows total 563435 exits happened during VM Boot.

Answer 4:

Most frequent exits : EPT Violation(48), followed by WRMSR(32)

Least frequent exits : EPT Misconfig(49), VMX Premption timer expired(52)