CMPE 283:Virtualization

Assignment 2: Instrumentation via hypercall

Name: Deep Khajanchi (013764686)

Github repo: <https://github.com/deepkhajanchi/linux>

Problem Statement:

1. For CPUID leaf node %eax=0x4FFFFFFF:

Return the total number of exits (all types) in %eax

(2) For CPUID leaf node %eax=0x4FFFFFFD:

Return the number of exits for the exit number provided (on input) in %ecx

This value should be returned in %eax

Steps to follow:

1. With the configuration of Assignment 1, follow the step in the sequence.
2. In the hypervisor code, open the folder linux/arch/x86/kvm.
3. Edit the file cpuid.c and add the the logic function for saving total exit counts to EAX and total time time spent cycles in EBX and ECX respectively.
4. Go to the folder linux/arch/x86/kvm/vmx.
5. Edit the file vmx.c and add the logic .
6. Save your changes and go back to the root folder.
7. Now build the source code using below command. Make sure you are using super user mode.

**sudo make**

1. Reboot the system after successful build.
2. Switch on the guest VM and install the package CPUID in it.

**sudo apt install cupid**

1. Run the belox command to check all the number of exits occurred by passing 0x04FFFFFFF instruction.

**cupid -l 0x04FFFFFFF**

1. To check the the number of exits for the exit number provided (on input) in %ecx.

**cupid -l 0x04FFFFFFD**

1. Now reboot the guest VMand run the command in step 10.
2. Count the total number of exits occur in a full system boot.

* Logic implementation in cupid.c

|  |
| --- |
| //changes in assignment2 |
|  |

|  |
| --- |
| int interruptCounterCpuid = 0; |
|  |

|  |
| --- |
| EXPORT\_SYMBOL\_GPL(interruptCounterCpuid); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| int counterIpt[2][69] ={ |
|  |

|  |
| --- |
| {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68}, |
|  |

|  |
| --- |
| {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}}; |
|  |

|  |
| --- |
| EXPORT\_SYMBOL(counterIpt); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| bool kvm\_cpuid\_customLeaf(struct kvm\_vcpu \*vcpu, u32 \*eax, u32 \*ebx, u32 \*ecx, u32 \*edx){ |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| if(\*eax == 0x4FFFFFFF) { |
|  |

|  |
| --- |
| \*ebx = 0x00000000; |
|  |

|  |
| --- |
| \*ecx = 0x00000000; |
|  |

|  |
| --- |
| \*edx = 0x00000000; |
|  |

|  |
| --- |
| \*eax = interruptCounterCpuid; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| else if(\*eax == 0x4FFFFFFD) { |
|  |

|  |
| --- |
| \*eax = 0x00000000; |
|  |

|  |
| --- |
| if(\*ecx <69){ |
|  |

|  |
| --- |
| \*eax = counterIpt[1][\*ecx]; |
|  |

|  |
| --- |
| \*ebx = 0x00000000; |
|  |

|  |
| --- |
| \*ecx = 0x00000000; |
|  |

|  |
| --- |
| \*edx = 0x00000000; |
|  |

|  |
| --- |
| }else { |
|  |

|  |
| --- |
| \*eax = 0x00000000; |
|  |

|  |
| --- |
| \*ebx = 0x00000000; |
|  |

|  |
| --- |
| \*ecx = 0x00000000; |
|  |

|  |
| --- |
| \*edx = 0xFFFFFFFF; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| else { |
|  |

|  |
| --- |
| \*eax = 0x00000000; |
|  |

|  |
| --- |
| \*ebx = 0x00000000; |
|  |

|  |
| --- |
| \*ecx = 0x00000000; |
|  |

|  |
| --- |
| \*edx = 0x00000000; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| return true; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| EXPORT\_SYMBOL\_GPL(kvm\_cpuid\_customLeaf); |
|  |

//till her

* Logic implementation in vmx/vmx.c

|  |
| --- |
| //changes for assignment 2. |
|  |

|  |
| --- |
| extern int interruptCounterCpuid; |
|  |

|  |
| --- |
| extern int counterIpt[2][69]; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| /\* |
|  |

|  |
| --- |
| \* The guest has exits. |
|  |

|  |
| --- |
| \*/ |
|  |

|  |
| --- |
| static int vmx\_handle\_exit(struct kvm\_vcpu \*vcpu) |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| //atomic\_inc(&interruptCounterCpuid); |
|  |

|  |
| --- |
| interruptCounterCpuid = interruptCounterCpuid+1; |
|  |

|  |
| --- |
| struct vcpu\_vmx \*vmx = to\_vmx(vcpu); |
|  |

|  |
| --- |
| u32 exit\_reason = vmx->exit\_reason; |
|  |

|  |
| --- |
| u32 vectoring\_info = vmx->idt\_vectoring\_info; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| if(exit\_reason < 69){ |
|  |

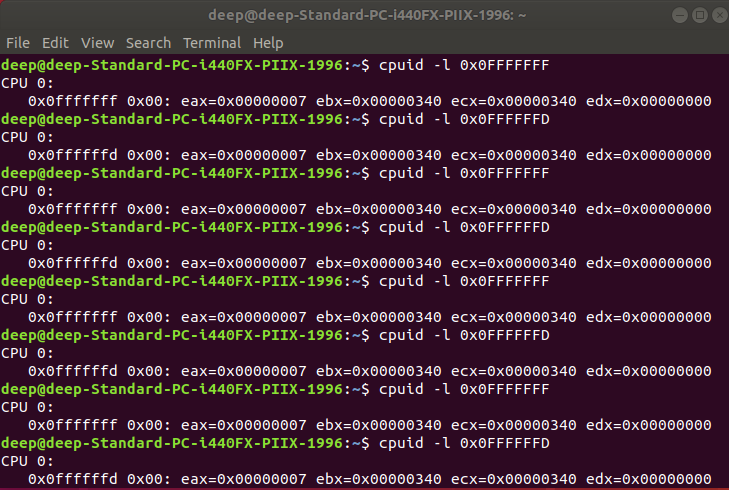
|  |
| --- |
| counterIpt[1][exit\_reason] = counterIpt[1][exit\_reason] + 1; |
|  |

|  |
| --- |
| } |
|  |

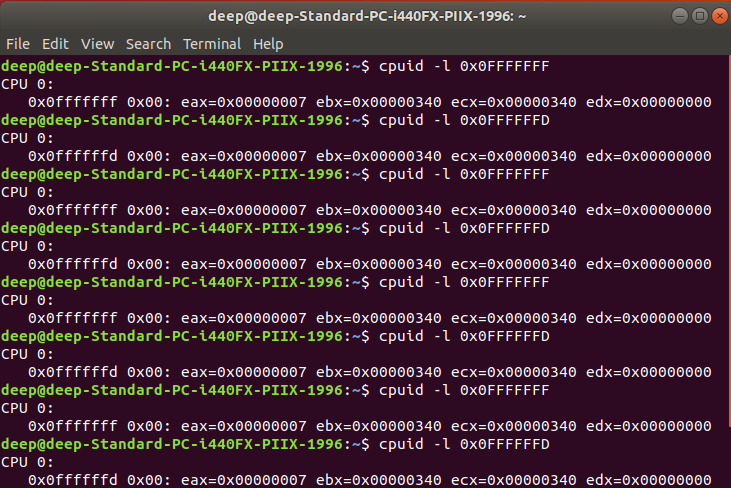
|  |
| --- |
|  |
|  |

//till here

1. Exit counter before reboot



1. Exit counter after reboot



**Question:**

Comment on the frequency of exits – does the number of exits increase at a stable rate? Or are there more exits performed during certain VM operations? Approximately how many exits does a full VM boot entail?

**Answer:**

The no. of exits increases every time but not on a stable rate. The number of exits depends on the different operations, like IO or EPT violation the exits are more in comparison with the other operations. A full VM boot entails approximately **1234590** exits for the source tree.

**Question:**

Of the exit types defined in the SDM, which are the most frequent? Least?

**Answer:**

“External interrupts” are the most frequent and “VMCALL/VMREAD” are the least.