CMPE283 : Virtualization
Assignment 2: Instrumentation via hypercall
Assignment 3: Instrumentation via hypercall (cont'd)

Due: 22 Nov 2021 before midnight (Assignment 2)
Due: 29 Nov 2021 before midnight (Assignment 3)

In these assignments you will learn how to modify processor instruction behavior inside the KVM hypervisor. Each lab assignment is worth up to 25 points and may be done in groups of up to **two people max.** Each team member can receive up to 25 points per assignment. It is expected that groups of more than one student will find an equitable way to distribute the work outlined in these assignments.

Prerequisites

- You will need a working assignment 1 configuration.

The Assignment

Your assignment is to modify the CPUID emulation code in KVM to report back additional information when special CPUID leaf nodes are requested.

- For CPUID leaf node %eax=0x4FFFFFFF:
  - Return the total number of exits (all types) in %eax
- For CPUID leaf node %eax=0x4FFFFFFE:
  - Return the high 32 bits of the total time spent processing all exits in %ebx
  - Return the low 32 bits of the total time spent processing all exits in %ecx
    - %ebx and %ecx return values are measured in processor cycles, across **all** VCPUs
- 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
- For CPUID leaf node %eax=0x4FFFFFFC:
  - Return the time spent processing the exit number provided (on input) in %ecx
    - Return the high 32 bits of the total time spent for that exit in %ebx
    - Return the low 32 bits of the total time spent for that exit in %ecx

For leaf nodes 0x4FFFFFFD and 0x4FFFFFFC, if %ecx (on input) contains a value not defined by the SDM, return 0 in all %eax, %ebx, %ecx registers and return 0xFFFFFFFF in %edx. For exit types not enabled in KVM, return 0s in all four registers.

At a high level, you will need to perform the following:

- Start with your assignment 1 environment
- Modify the kernel code with the assignment functionality:
  - Determine where to place the measurement code (for exit counts and # cycles)
  - Create new CPUID leaf 0x4FFFFFFF, 0x4FFFFFFE, 0x4FFFFFFD, 0x4FFFFFFC
    - Report back information as described above
- Create a user-mode program that performs various CPUID instructions required to test your assignment
  - Pro tip: This can be achieved on ubuntu by installing the 'cpuid' package
  - Run this user mode program in the inner VM
  - There is no need to insmod anything like assignment 1 did
- Verify proper output

For assignment 2, select two of the four above CPUID leaf nodes and implement the functionality described, and answer the questions below.

For assignment 3, complete the remaining two CPUID leaf nodes and answer the questions below.

On or before the due date(s), commit to your github repo and send the answers to the questions below via a README.md file in your repo.

You should make a test program that exercises the functionality in your hypervisor modification. Since CPUID can be called at any privilege level, you can make a simple usermode program to do this. A sample test output might look like:

```
$ ./test_assignment2
CPUID(0x4FFFFFFF), exits=18840
CPUID(0x4FFFFFFE), total time in vmm: 3871824973 cycles

$ ./test_assignment2 1
CPUID(0x4FFFFFFD), exit number 1 exits=321
```

Note: I will not be running your test code. I have my own test code, and I'll be running that. Thus, there is no need for you to include your test code in your submission.

Grading

Each assignment will be graded and points awarded based on the following:

- 20 points for the implementation and code producing the output above
- 5 points for the answers to the questions below

You should ensure that you "commit early, commit often" to your github repository. Make sure you make your repository public. I will be cloning your repository and checking out code as of the due date, so any commits made after the due date will not be considered.

I will be comparing all submissions to ensure no collaboration has taken place. Make sure you do not copy another group's work. If you copy another group's work, members of both groups will receive an F in the class and be reported to the department chair for disciplinary action. If you are working in a group, make sure your partner does not copy another group's work without your knowledge, as all group members will be penalized if cheating is found.

Special Notes

I will be cloning your repository and using it to build a complete kernel, so please make sure your tree builds. If I cannot build your code, I cannot grade it, and you will receive a zero score for the assignment.

Questions – Provide answers in a README.md in your repo.

For both assignment 2 and 3:

1. For each member in your team, provide 1 paragraph detailing what parts of the lab that member implemented / researched. (You may skip this question if you are doing the lab by yourself).
2. Describe in detail the steps you used to complete the assignment. Consider your reader to be someone skilled in software development but otherwise unfamiliar with the assignment. Good answers to this question will be recipes that someone can follow to reproduce your development steps.

   Note: I may decide to follow these instructions for random assignments, so you should make sure they are accurate.

For whichever assignment you chose to complete CPUID 0x4FFFFFFD and 0x4FFFFFFC:

3. 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?
4. Of the exit types defined in the SDM, which are the most frequent? Least?