CMPE283 : Virtualization
Assignment 2: Modifying instruction behavior in KVM

Due: 21 Nov 2018 before midnight

In this assignment you will learn how to modify processor instruction behavior inside the KVM hypervisor. This lab assignment is worth up to 30 points and may be done in groups of up to **two people max.** Each team member can receive up to 30 points. It is expected that groups of more than one student will find an equitable way to distribute the work outlined in this assignment. You can form groups from both Monday and Tuesday sections, as both sections will receive the same assignment.

Prerequisites

- You will need a machine capable of running Linux, with VMX virtualization features exposed. You may be able to do this inside a VM, or maybe not, depending on your hardware and software configuration.

The Assignment

Your assignment is to modify the CPUID emulation code in KVM to report back additional information when a special CPUID "leaf function" is called.

- For CPUID leaf function %eax=0x4FFFFFFF:
  - Return the total number of exits (all types) in %eax
  - 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

At a high level, you will need to perform the following:
- Configure a Linux machine, either VM based or on real hardware. You may use any Linux distribution you wish, but it must support the KVM hypervisor.
- Download and build the Linux kernel source code (see below)
- Modify the kernel code with the assignment functionality:
  - Determine where to place the measurement code
  - Create new CPUID leaf 0x4FFFFFFF
    - Report back information as described above
- Create a user-mode program that performs various CPUID instructions required to test your assignment
- Verify proper output

On or before the due date, turn a code listing and answers to the questions below via E-Mail. **Make sure to follow the instructions on diff/code format below.**

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=32149, cycles spent in exit=1028748293

$ ./test_assignment2
CPUID(0x4FFFFFFF), exits=32291, cycles spent in exit=1159390993
```

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.

Building The Kernel

To build the kernel (once you have cloned the Linux git repository), the following sequence of commands can be used (eg, for Ubuntu – other distributions have similar steps but may differ in the installation of the build prerequisites):

```
sudo bash
apt-get install build-essential kernel-package fakeroot libncurses5-dev libssl-dev ccache bison flex libelf-dev
make menuconfig                                    (and then just exit and save the default .config, don't change anything)
make && make modules && make install && make modules-install                 (will take a long time the first time)
reboot
```

Verify that you are using the newer kernel (4.16, etc) after reboot:

```
uname -a
```

After changing the code in KVM for the assignment, you can rebuild using the same "make" sequence of commands above (and it should only take a few minutes, not several hours).

Grading

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

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

Submissions shall be made via email to the email addresses listed in the class syllabus/green sheet. DO NOT WAIT UNTIL LATE ON THE DUE DATE, as email server lags or delays may result in a late submission. Since you have three weeks to complete this assignment, I will not accept "email server outage or delay" as an excuse for late submissions. If you are concerned about this, submit your assignment early. This is one area that I am extremely picky with – even 1 second late will result in a zero score.

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 partners do not copy another group's work without your knowledge, as all group members will be penalized if cheating is found.

Special Notes

I am implementing an automated framework to test these submissions. Therefore, you **must** follow the subsequent submission rules precisely. I will be using a script that will automatically process my mailspool to extract your submissions, and the script will expect the submission email to formatted a specific way, as described below:

- Use a kernel built from the master Linux git repository
  - https://github.com/torvalds/linux.git
  - Record the head commit ID of your tree:
    - For example, if the output of "git log" shows the following:
      commit 89970a04d70c6c9e5e4492fd4096c0b5630a478c
      Merge: 806276b7f07a 3ea3217cf918
      Author: Linus Torvalds <torvalds@linux-foundation.org>
      Date: Wed Mar 29 19:59:49 2017 -0700
    - .. you would record "commit 89970a04d70c6c9e5e4492fd4096c0b5630a478c"
- Use 'git add' and 'git commit' to add your tree changes. **Include all changes required to build your assignment.**
- Submit a plain text file containing a unified diff ("git diff" format) diff file containing your entire submission. Name the diff file "cmpe283-2.diff" in your submission when attaching to the email.

- When submitting, submit **only**:
    - A plain text email (no HTML)
    - A single PDF file attachment containing the answers to the questions (the PDF can have whatever name you want)
    - The diff file in plain text format as described above.
    - Your commit ID as calculated above, by including a line starting with “commit” followed by a space, followed by the SHA value of the commit ID, with no other text on that line.
    - A list of student IDs and names for members of the group, in the following format, one per line:
        - ID <id> <name>
            - (example) ID 0123456789 Larkin, Michael
- The subject line of the email must be “CMPE283 Assignment 2 Submission” (no quotes)
- Do not submit .zip, .tar, .rar, etc. Send me an email with two attachments, a PDF and a .diff file, as described above.
- Do not mangle your diff
- Do not give me commit IDs of your local commits
- Do test your diff before submitting
- Make sure to follow all other instructions in this assignment precisely.
- **Failure to follow the instructions above may result in a zero score for the assignment.**

Questions

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.
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?