# CMPE\_283\_Assignment1-intel

**1.For each member in your team, provide 1 paragraph detailing what parts of the lab that member implemented / researched.**

Team Members:

Priyanka Devendran- (id:015231411)

* Setup the environment in Mac using VMware Fusion and installed Ubuntu.
* ﻿Discussed and Researched about MSRs to be read in the SDM.
* Contribution: MSR code for controls : Primary Procbased, Secondary Procbased.

Preeti Parihar -(id:015218073)

* Setup the environment in Mac using VMware Fusion 12.x Pro(Student Licence) and ﻿download the ISO disk image file for Ubuntu Desktop(Ubuntu 20.04.1 LTS). Created VM successfully in first attempt by allocating 200GB storage and 4GB RAM.
* ﻿Researched and Discussed about MSRs to be read in the SDM and help each other writing and executing the code.
* Contribution: MSR code for controls: Entry Controls, Exit Controls and Determine if secondary procbased controls are available By checking 31st bit.

**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.**

Prerequisites • A machine capable of running Linux, with VMX virtualization features exposed.

Requirement: To create a Linux kernel module that will query various MSRs to determine virtualization features available in the CPU.

Functionality : • To read various MSRs to ascertain support capabilities/features - Entry / Exit / Procbased / Secondary Procbased / Pinbased controls • For each group of controls above, interpret and output the values read from the MSR to the system via printk(..), including if the value can be set or cleared.

**SETUP the Environment:**

* We installed VMware fusion 12.x Pro in Macbook then Downloaded the ISO disk image for Ubuntu Desktop(Ubuntu 20.04.1 LTS).
* Created Virtual Machine by allocating 200GB storage and 4GB RAM.
* Open virtual machine’s properties page, select Processors & Memory under advanced options, select the checkbox next to Enable hypervisor applications in this virtual machine.
* Now build the linux kernel using the following command:

1. ﻿Install Git:

sudo apt-get update

sudo apt-get install git fakeroot build-essential ncurses-dev xz-utils libssl-dev bc

﻿

1. Check version of current linux kernel:

uname –r

1. Clone the Git repository for the latest linux kernel source code :

git clone <https://github.com/torvalds/linux.git>

1. Change to the cloned directory:

cd linux

1. ﻿Do git log and Note the COMMIT id - ﻿Record the head commit ID of your tree:

git log

sample output:

preeti@ubuntu:~/linux$ git log

commit d76913908102044f14381df865bb74df17a538cb (HEAD -> master, origin/master, origin/HEAD)

Merge: af0041875ce7 24f7bb8863eb

Author: Linus Torvalds <torvalds@linux-foundation.org>

Date: Sat Oct 24 12:46:42 2020 -0700

1. ﻿Change to the Linux folder and configure the modules to be included/excluded:

﻿ make oldconfig

1. ﻿Check for number of processing units available using nproc in our case its 2 and then make modules-install:

﻿ nproc

﻿ make && make modules && make install && make modules-install

1. ﻿Use ﻿update-grub command to automatically look for the /boot folder and adds them to the grub’s config file:

﻿ update-grub

1. Again check for the version and make sure it shows latest kernel version:

﻿ uname –r (sample output: 5.4.0-52-generic)

1. To make sure Environment setup properly run following command and look for vmm:

cat /proc/cpuinfo | more (if you able to see vmm in your output it means environment setup properly, if vmm is not present it means you have to enable it(see bullet point 3 for how to enable) otherwise the output will be 0x0 which is not correct)

Either use above steps to build the kernel once you clone the repo or use this simple approch mentioned in assignment2 pdf available in canvas.

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):

1. sudo bash
2. apt-get install build-essential kernel-package fakeroot libncurses5-dev libssl-dev ccache bison flex libelf-dev
3. uname -a (and note down your kernel version, for example “4.15.0-112-generic”)
4. cp /boot/config-4.15.0-112-generic ./.config (substitute your version obtained from the previous step here though)
5. make oldconfig (and then just use the default for everything, don’t change anything – you can do this by holding down enter)
6. make && make modules && make install && make modules-install (will take a long time the first time)reboot
7. Verify that you are using the newer kernel (5.8, etc) after reboot:

uname -a (sample output: Linux ubuntu 5.4.0-52-generic #57-Ubuntu SMP Thu Oct 15 10:57:00 UTC 2020 x86\_64 x86\_64 x86\_64 GNU/Linux)

**Module Development:**

Following are the steps to be followed:

1. Create a new directory named “cmpe283” by using mkdir in previously cloned linux source folder:

mkdir cmpe283

2. ﻿Save the files cmpe283-1.c and Makefile in cmpe283 directory, from files section of Canvas provided by the Professor(in our case):

3. ﻿Modify the file cmpe283-1.c to implement the assignment functionality.

4. We create different functionality to query all the 5 different MSRs in the file cmpe283-1.c.

5. All the different structures are created with description and the bit position by referring to SDM.

6. To detect the VMX capabilities of CPU, the function report capability() will be called with passing the appropriate parameters to print the different controls.

7. ﻿We need to create the makefile to compile the module, In our case makefile is present in the canvas.

﻿Use below command to compile the module:

cd cmpe283

make

8. When we insert the module in the kernel, the module\_init marco will be invoked, which will in turn call the function init\_module defined in code. Use sudo because we can not make changes in user mode.

Command: sudo insmod ./cmpe283-1.ko

9. Once loaded the VMX features must be logged in the kernel log and this can be checked by using the command: dmesg

10. When we unload the module the rmmod, module\_exit macro will be invoked, which will in turn call the cleanup\_module defined in code.

Command: sudo rmmod ./cmpe283-1.ko

11. When finally implemented all the requirement mentioned in the assignment do git commit.