Question 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).

Kaustubh Hassan Narasimhan(ID: 013727129)

1. Implemented code for true controls test
2. Implemented code for VM-Entry Controls and VM-Exit Controls MSRs respectively by referring SDM Table 24-12 and 24-10 respectively.
3. Wrote procedure for VM-Entry controls in Question 2 and made the git diff file for submission

Jason Gonsalves(ID: 013740610)

1. Installed VMware Workstation 15 Player and installed Ubuntu using ubuntu-18.04.1-desktop-amd64 disk image file.
2. Implemented code for VM-Primary Processor based controls and VM-Secondary Processor based controls by referring SDM Table 24-6 and 24-7 respectively.
3. Wrote procedure followed for VM-Exit Controls MSR, Primary Processor based VM-execution controls and Secondary Processor based VM-execution controls in Question 2.

Question 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.

1. Download VMware Virtual Station.
2. Install Ubuntu Virtual Machine (ubuntu-18.04.1-desktop-amd64 disk image file).
3. Clone the linux repo
   git clone https://github.com/torvalds/linux.git
4. Change to Linux Directory
   cd linux
5. Copy your kernel config file to the current directory. You'll find it in /boot, and the name will vary but it will generally start with "config". The new name should be ".config"
   copy /boot/config-4.15.0-29-generic .config
6. Execute 'makeoldconfig' and answer 'y' or 'yes' as default answer to all the questions
7. Start a bash shell as a root level user :
   sudo bash

8. Build kernel (link the kernel image, compile individual files for each question you answered during kernel config, install your built kernel and install your kernel modules) using the command:
   make && make modules && make install && make modules_install
9. Reboot and select the new kernel during the boot process (should be 5.0. something)
10. Create a new folder (say cmpe283-1, inside the Linux directory), download the two files i.e. Makefile and the .c file (for checking MSR for Pinbased controls that Professor has given) and copy them into this folder
11. First, we need to detect the presence of true controls, for that we need to check if the 55th bit of IA32_VMX_BASIC MSR(0x480) is set
12. Read this MSR by calling
    rdmsr(IA32_VMX_BASIC, low, high);
    (declare two unsigned integer variables low and high before)
13. Next, check if the 24th bit of high variable is set or not(MSR read is stored in two variables of each 32 bits, therefore 24 bits into high contains 55th bit)
14. We can do this by right shifting high by 23 bits and checking to see if that bit is set by 'AND' ing it with a 1
15. If it's set, we will call a different method(detect_vmx_true_features_pinbased), where we read from IA32_VMX_TRUE_PINBASED_CTLS), otherwise we call the regular method(where we read from IA32_VMX_PINBASED_CTLS)
16. Run "**make**"
17. Check to see whether there's a. ko file in the same directory
18. This is the module that is generated, insert it into the kernel that you built using:
    insmod (name).ko
19. Execute "**dmesg**" to see the kernel message, sample output is shown below -

```
831.051542] True controls available
831.051566] True Pinbased Controls MSR: 0x3f00000016
831.051567]   External Interrupt Exiting: Can set=Yes, Can clear=Yes
831.051568]   NMI Exiting: Can set=Yes, Can clear=Yes
831.051568]   Virtual NMIs: Can set=Yes, Can clear=Yes
831.051569]   Activate VMX Preemption Timer: Can set=No, Can clear=Yes
831.051569]   Process Posted Interrupts: Can set=No, Can clear=Yes
```

20. Next, we need to modify the code for the other 4 MSRs and add true controls test

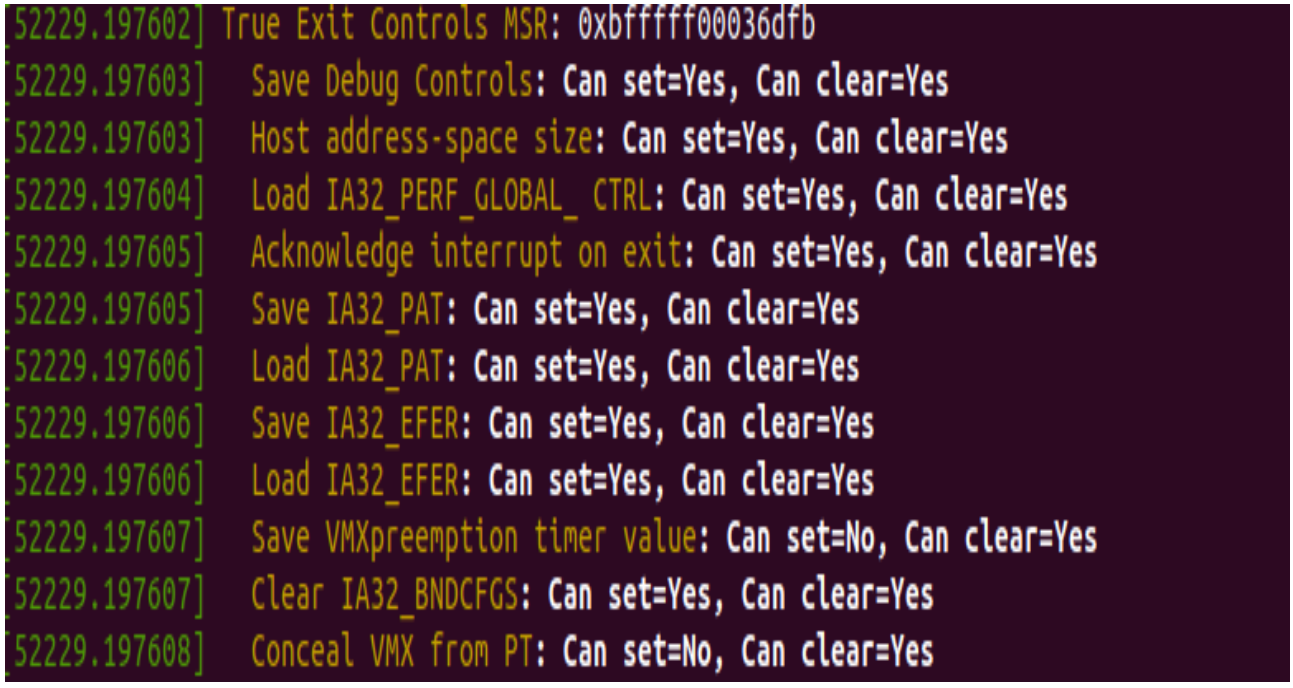
21. All we need to modify is the struct that we passed in the report_capability method, length of the structure and additionally add some code for true controls test

24-12: VM-Entry Controls MSR

22. Detect the presence of true controls(as described above for Pinbased Controls)
23. If it's set, we will call a different method(detect_vmx_true_features_entrycontrols(), where we read from IA32_TRUE_ENTRY_CTLS), otherwise we call the regular method(where we read from IA32_VMX_ENTRY_CTLS)
24. Add the definitions for IA32_VMX_ENTRY_CTLS to be 0x484 and IA32_TRUE_ENTRY_CTLS to be 0x490
25. For adding the struct for entry controls MSR, refer Table **24-12** or Section **24.18.1** of the SDM(please note that there are 9 values for VM-Entry Controls), replace the Bit Positions and the corresponding Names in the C struct
26. Add the name of the struct that you pass when you call the report_capability function in the detect_vmx_features(void) method
27. Make any other suitable functions and run the c file using "make" as described before
28. Insert the module into the kernel(**insmod**) and see the message log(**dmesg**) to see the output
29. Sample output for VM-Entry Controls MSR will look like this -
30. Repeat the process for the other 3 MSRs using the tables in SDM as follows

24-10: VM-Exit Controls MSR

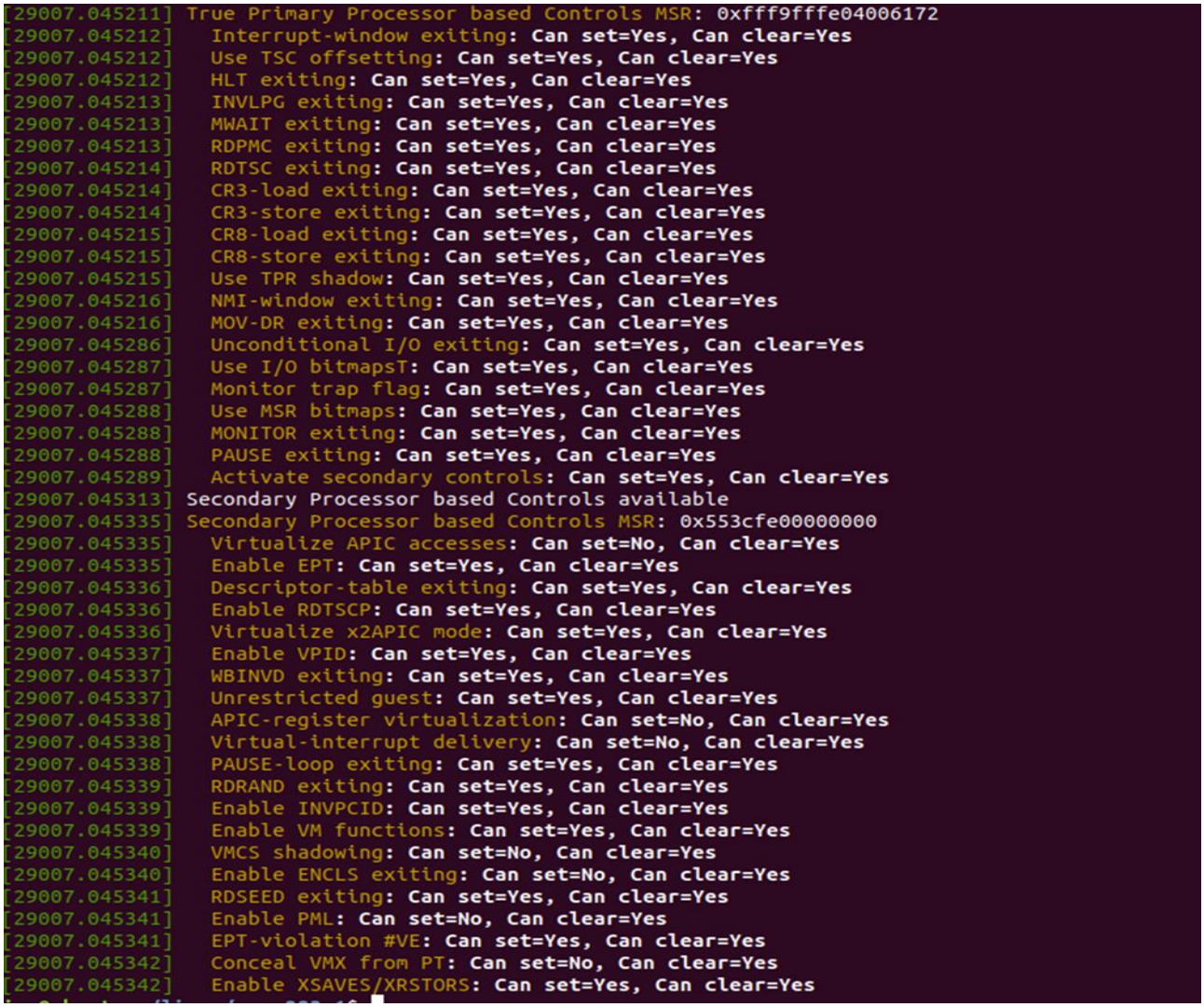
31. Detect the presence of true controls(as mentioned above for Pinbased Controls MSR)
32. If it's set, we will call a different method(detect_vmx_true_features, where we read from IA32_TRUE_EXIT_CTLS), otherwise we call the regular method(where we read from IA32_VMX_EXIT_CTLS)
33. Add the definitions for IA32_VMX_EXIT_CTLS to be 0x483 and IA32_TRUE_EXIT_CTLS to be 0x48F
34. For adding the struct for exit controls MSR, refer Table **24-10** or Section **24.7.1** of the SDM (please note that there are 11 values for VM-Exit Controls), replace the Bit Positions and the corresponding Names in the C struct, also change the name of the struct
35. Add the name of the struct that you pass when you call the report_capability function in the detect_vmx_features(void) method
36. Make any other suitable functions and run the c file using "make" as described before
37. Insert the module into the kernel(**insmod**) and see the message log(**dmesg**) to see the output



24-6: Processor based VM-execution controls(Primary and Secondary)

38. Detect the presence of true controls(as mentioned above for Pinbased Controls MSR)

39. Add the definitions for IA32_VMX_PROCBASED_CTLS to be 0x484 and IA32_TRUE_PROCBASED_CTLS to be 0x490 and and IA32_VMX_PROCBASED_CTLS2 to be 0x48B
40. For adding the struct for primary processor based MSR, refer Table **24-6** or Section **24.6.2** of the SDM (please note that there are 21 values for Primary Processor based VM-execution Controls) and 24-7 for Secondary Processor based controls(there are 24 values for Secondary Processor based VM-Execution based controls)
41. Add the Bit Positions and the corresponding Names in the C struct, also change the name of the struct
42. Add the name of the struct that you pass when you call the report_capability function in the detect_vmx_features(void) method
43. Make any other suitable functions and run the c file using “make” as described before
44. Check for the presence of Secondary Processor based controls by checking to see whether the 63rd bit of reading IA32_VMX_PROCBASED_CTLS is set or not



45. If it’s set, we will call a method for printing Secondary Processor based controls
46. Insert the module into the kernel(**insmod**) and see the message log(**dmesg**) to see the output
47. Document your results and send to Professor