**Assignment-01**

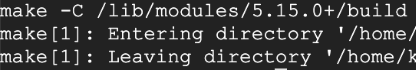
### Steps followed:

1. Retrieve the starter .c file and Mekefile from canvas.  
2. Add the remaining code sections to the file.(struct, definitions and vm features)

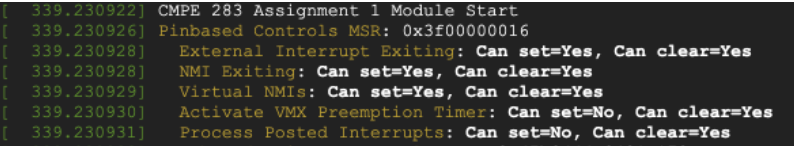
| **MSR Name** | **MSR Index** | **Description** | **References** |
| --- | --- | --- | --- |
| IA32\_VMX\_PINBASED\_CTLS | 0x481 | This MSR is used for pinbased controls if no true controls capability | SDM volume 3C, section 24.6.1 |
| IA32\_VMX\_PROCBASED\_CTLS | 0x482 | This MSR is used for primary procbased controls if no true controls capability | SDM volume 3C, section 24.6.2 |
| IA32\_VMX\_PROCBASED\_CTLS2 | 0x48B | This MSR is used for secondary procbased controls if available | SDM volume 3C, section 24.6.2 |
| IA32\_VMX\_EXIT\_CTLS | 0x483 | This MSR is used for exit controls if no true controls capability | SDM volume 3C, section 24.7.1 |
| IA32\_VMX\_ENTRY\_CTLS | 0x484 | This MSR is used for entry controls if no true controls capability | SDM volume 3C, section 24.8.1 |

1. Create a GitHub repo, fork the torvalds:master repo and create a folder CMPE 283 inside the linux repo.
2. Upload the edits .c file and Makefile to the folder to get started with building the kernel.
3. Create a GCP account and set up a VM compute instance.
4. Create an N2 type- Ubuntu 20.04.3 virtual machine (15 GB RAM, 4 vcpus and 100 GB HDD) in Las Vegas region. Make sure the nested virtualization is enabled on the machine.
5. Clone the github repo created in the VM instance.
6. Run the make command.Running this command initially gave us a lot of package dependency errors.
7. Install the packages by apt-get install <package name>. To name a few, we had to install flex, bison, binutils, libelf and libssl.
8. Had to run make oldconfig and make prepare in order to build the entire kernel in the cloned source code repo.
9. The make modules and make command then ran successfully. It took about 1.5 hours for it to complete.
10. Copy all the modules built to the boot location. (strip debugging information)
11. Run the make install command to install the kernel on the system.
12. Reboot the VM so the the newly installed kernel boots up.
13. Run the make command in the root directory. (the output screenshot is attached below). Include the MODULE\_LICENSE line in the cmpe283-1.c file.
14. Run the sudo insmod cmpe283-1.ko command to insert modules in the kernel.
15. Run the dmesg command to display the output printed on the system message buffer console

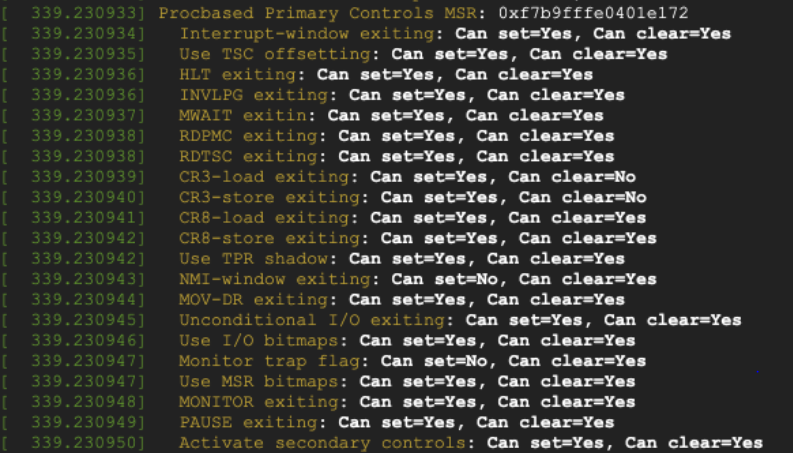
Final make command output with timestamp and kernel version



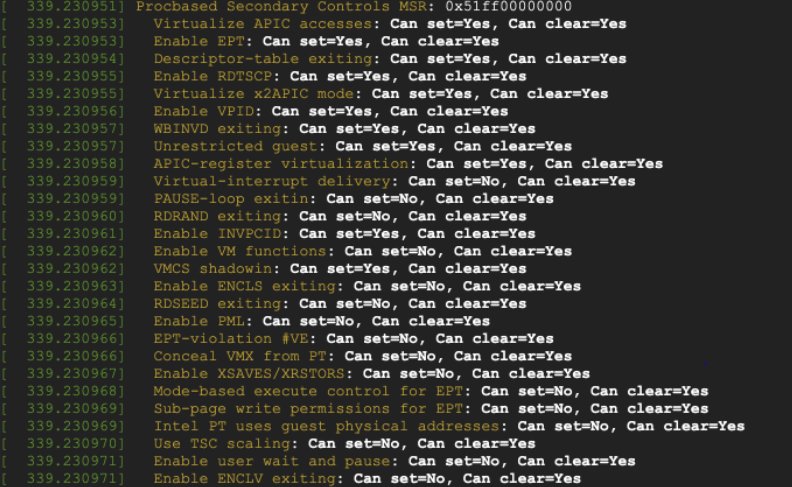
Pinbased control Output



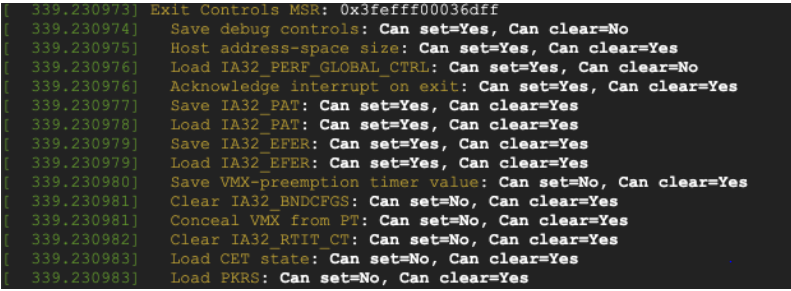
Primary Procbased control output



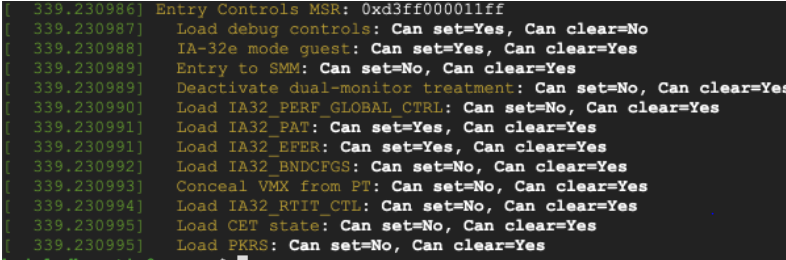
Secondary Procbased control output



Exit controls output:



Entry controls output



## Assignment-02

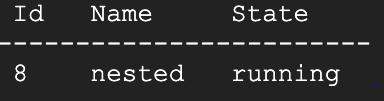
### Steps followed:

1. Update the files arch/x86/kvm/vmx/vmx.c and arch/x86/kvm/cpuid.c
2. Rebuild the kernel using make modules command and make INSTALL\_MOD\_STRIP=1 modules\_install && make install.
3. Run lsmod | grep kvm to check if the kvm modules are preloaded.
4. If they are already present remove them using rmmod kvm and rmmod kvm\_intel commands.
5. Run modprobe kvm and modprobe kvm\_intel commands to reload edited kvm modules.
6. Optional- We installed GUI for out Ubuntu host for a friendly UI and ease fo work. Also, we wanted to use Virtual machine manager to created the nested VM. GUI can be enabled on Ubuntu VM using below commands:([ref](https://subscription.packtpub.com/book/big-data-and-business-intelligence/9781788474221/1/ch01lvl1sec15/installing-and-configuring-ubuntu-desktop-for-google-cloud-platform))  
   $ sudo apt-get install gnome-shell  
   $ sudo apt-get install ubuntu-gnome-desktop  
   $ sudo apt-get install autocutsel  
   $ sudo apt-get install gnome-core  
   $ sudo apt-get install gnome-panel  
   $ sudo apt-get install gnome-themes-standard
7. Install xrdp through the below commands- we need this to be able to access Ubuntu in GUI mode.  
   sudo apt-get update  
   sudo apt-get install -y xrdp  
   sudo apt-get install -y xfce4  
   sudo service xrdp restart
8. Login to the host VM using RDP to have graphical interface.
9. To enable the kvm module in the host and install necessary packages, run the below commands from host terminal: ([ref](https://www.tecmint.com/install-kvm-on-ubuntu/))  
   sudo apt install qemu qemu-kvm qemu-system qemu-utils  
   sudo apt install libvirt-clients libvirt-daemon-system virtinst
10. Run Virtual Machine Manager and create a new VM inside the host. (download the iso file or guest VM as a prerequisite)
11. Install Guest OS once the VM is created and login to the nested VM.
12. Install CPUID using sudo apt install cpuid if it is an Ubuntu VM
13. Run the command cpuid -l 0x4FFFFFFF to verify the output.
14. Run the test bash script to produce results and print number of exits.
15. Run the test2 bash script to produce number of cycles in ebx and ecx registers when eax=0x4ffffffe.

Output screen that verifies that kvm is installed on Ubuntu host



Output screen that shows nested VM created on KVM Host



Output screen that shows number of exits when eax=0x4fffffff

