Name: Anshul Patni **Student ID**: 012415156
Name: Prasun Saxena **Student ID**: 012449775

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

1. Work done by Anshul Patni:
- Downloaded and built the Linux Kernel modules and associated libraries to create a local copy of Linux Kernel.
- Modified the cmpe283-1.c code by adding the custom logic to enable our system to read and give output for capabilities of the various MSRs.
- Staged and committed the cmpe283-1.c code file and Makefile after inserting the module and printing out the buffer from the Kernel.
- Generated a comprehensive diff file after committing the changes to the repository.

2. Work done by Prasun Saxena:
- Tested the machine to check its capability for the VMX virtualization and feature recoginition.
- Tested and verified the proper working of the functionality of code by comparing it with the sample output given to us.
- Simulating the answers for the questions in the report.

Question 2: Describe in detail the steps you used to complete the assignment.
Answer:

This assignment focuses on the creation of a Linux kernel module to query various MSRs to determine the virtualization features present in CPU. The module will then report the features it discovers as system message log.

Prerequisites:
- A machine capable of running Linux
- Features for VMX virtualization exposed

Step followed to complete the assignment-

1. Install git

- sudo apt-get install git

2. Download the Linux kernel source code:

- git clone https://github.com/torvalds/linux.git

3. Build the Linux Kernel using the source code in previous step:

- sudo apt-get install libncurses-dev
- sudo apt-get install libssl-dev
- make menuconfig

- make
- make modules
- make modules_install
- make install

4. Code a .c (C language) file that will be used to find the capabilities of the MSRs:

-gedit cmpe283-1.c

File content/code displayed in the appendix of this file.

5. Creating new kernel module for MSRs:

-gedit Makefile

File content-

```
obj-m += cmpe283-1.o
all:
	make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules
clean:
	make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

-make all

6. Inserting/Loading the specific kernel module into the kernel:

- insmod ./cmpe283-1.ko

7. Verify the message buffer/output from the kernel in the system message log:

- dmesg

8. Commit the changes done to the local repository:

- git add /home/prasun/Desktop/linux/anshul/cmpe283-1.c
/home/prasun/Desktop/linux/anshul/Makefile

- git config --global user.email "anshulpatni19@gmail.com"

- git config --global user.name "AnshulPatni"

- git commit -m “CMPE 283 Assignment 1 - Initial Commit”

9. Creation of a diff file:

- git diff HEAD~1 > cmpe283-1.diff

10. Check the git log to find the details of the most recent commit:

- git log

```
commit 487e2a060765ea198eb9c66744c5799f13120876
Author: AnshulPatni <anshulpatni19@gmail.com>
Date:   Sun Mar 18 23:11:43 2018 -0700

    CMPE 283 Assignment 1 - Initial commit

commit 6f7da290413ba713f0cdd9ff1a2a9bb129ef4f6c
Author: Linus Torvalds <torvalds@linux-foundation.org>
Date:   Sun Jul 2 16:07:02 2017 -0700

    Linux 4.12

commit 401e000ab90d7b81d8ea0735e3ff909548754876
Author: Sylvain 'ythier' Hitier <sylvain.hitier@gmail.com>
Date:   Sun Jul 2 15:21:56 2017 +0200

    moduleparam: fix doc: hwparam_irq configures an IRQ

    Signed-off-by: Sylvain 'ythier' Hitier <sylvain.hitier@gmail.com>
    Signed-off-by: Linus Torvalds <torvalds@linux-foundation.org>
```

Appendix

1. Code

```
/*
 *  cmpe283-1.c - Kernel module for CMPE283 assignment 1
 */
#include <linux/module.h>    /* Needed by all modules */
#include <linux/kernel.h>      /* Needed for KERN_INFO */
#include <asm/msr.h>

#define MAX_MSG 80

/*
 * Model specific registers (MSRs) by the module.
 * See SDM volume 4, section 2.1
 */
#define IA32_VMX_PINBASED_CTLS     0x481

#define IA32_VMX_BASIC_CTLS 0x480

#define IA32_VMX_PROCBASED_CTLS 0x482
```

```
#define IA32_VMX_PROCBASED_CTLS2 0x48B

#define IA32_VMX_EXIT_CTLS 0x483

#define IA32_VMX_ENTRY_CTLS 0x484

#define IA32_VMX_TRUE_PINBASED_CTLS 0x48D

#define IA32_VMX_TRUE_PROCBASED_CTLS 0x48E

#define IA32_VMX_TRUE_EXIT_CTLS 0x48F

#define IA32_VMX_TRUE_ENTRY_CTLS 0x490


/*
 * struct capability_info
 *
 * Represents a single capability (bit number and description).
 * Used by report_capability to output VMX capabilities.
 */
struct capability_info {
        uint8_t bit;
        const char *name;
};


/*
 * Pinbased capabilities
 * See SDM volume 3, section 24.6.1
 */
struct capability_info pinbased[5] =
{
        { 0, "External Interrupt Exiting" },
        { 3, "NMI Exiting" },
        { 5, "Virtual NMIs" },
        { 6, "Activate VMX Preemption Timer" },
        { 7, "Process Posted Interrupts" }
};

/*
* Procbased Controls
*/
struct capability_info procbased[21] =
{
        {2 , "Interrupt-window" },
        {3 , "Use TSC offsetting" },
        {7 , "HLT exiting" },
        {9 , "INVLPG exiting" },
```

```
        {10 , "MWAIT exiting" },
        {11 , "RDPMC exiting" },
        {12 , "RDTSC exiting" },
        {15 , "CR3-load exiting" },
        {16 , "CR3-store exiting" },
        {19 , "CR8-load exiting" },
        {20 , "CR8-store exiting" },
        {21 , "Use TPR shadow" },
        {22 , "NMI-window exiting" },
        {23 , "MOV-DR exiting" },
        {24 , "Unconditional I/O" },
        {25 , "Use I/O bitmaps" },
        {27 , "Monitor trap flag" },
        {28 , "Use MSR Bitmaps" },
        {29 , "MONITOR exiting" },
        {30 , "PAUSE exiting" },
        {31 , "Activate secondary controls" }
};

/*
* Procbased_ctls2 Controls
*/
struct capability_info procbased_ctls2[23] =
{
        { 0, "Virtualize APIC accesses" },
        { 1, "Enable EPT" },
        { 2, "Descriptor-table exiting" },
        { 3, "Enable RDTSCP" },
        { 4, "Virtualize x2APIC mode" },
        { 5, "Enable VPID" },
        { 6, "WBINVD exiting" },
        { 7, "Unrestricted guest" },
        { 8, "APIC-register virtualization" },
        { 9, "Virtual-interrupt delivery" },
        { 10, "PAUSE-loop exiting" },
        { 11, "RDRAND exiting" },
        { 12, "Enable INVPCID" },
        { 13, "Enable VM functions" },
        { 14, "VMCS shadowing" },
        { 15, "Enable ENCLS exiting" },
        { 16, "RDSEED exiting" },
        { 17, "Enable PML" },
        { 18, "EPT-violation" },
        { 19, "Conceal VMX nonroot operation from Intel PT" },
        { 20, "Enable XSAVES/XRSTORS" },
        { 22, "Mode-based execute control for EPT" },
        { 25, "Use TSC scaling" }
};
```

```
/*
 * Entry Controls
 */
struct capability_info entry[9] =
{
        { 2, "Load debug controls" },
        { 9, "IA-32e mode guest" },
        { 10, "Entry to SMM" },
        { 11, "Deactivate dual-monitor treatment" },
        { 13, "Load IA32_PERF_GLOBAL_CTRL" },
        { 14, "Load IA32_PAT" },
        { 15, "Load IA32_EFER" },
        { 16, "Load IA32_BNDCFGS" },
        { 17, "Conceal VM entries from Intel PT" }
};


/*
 * Exit Controls
 */
struct capability_info exit1[11] =
{
        { 2, "Save debug controls" },
        { 9, "Host addressspace size" },
        { 12, "Load IA32_PERF_GLOB AL_CTRL" },
        { 15, "Acknowledge interrupt on exit" },
        { 18, "Save IA32_PAT" },
        { 19, "Load IA32_PAT" },
        { 20, "Save IA32_EEFR" },
        { 21, "Load IA32_EFER" },
        { 22, "Save VMXpreemption timer value" },
        { 23, "Clear IA32_BNDCFGS" },
        { 24, "Conceal VM exits from Intel PT" },
};



/*
 * report_capability
 *
 * Reports capabilities present in 'cap' using the corresponding MSR values
 * provided in 'lo' and 'hi'.
 *
 * Parameters:
 *  cap: capability_info structure for this feature
 *  len: number of entries in 'cap'
 *  lo: low 32 bits of capability MSR value describing this feature
 *  hi: high 32 bits of capability MSR value describing this feature
 */
```

```
void
report_capability(struct capability_info *cap, uint8_t len, uint32_t lo,
   uint32_t hi)
{
        uint8_t i;
        struct capability_info *c;
        char msg[MAX_MSG];

        memset(msg, 0, sizeof(msg));

        for (i = 0; i < len; i++) {
                c = &cap[i];
                snprintf(msg, 79, "  %s: Can set=%s, Can clear=%s\n",
                   c->name,
                   (hi & (1 << c->bit)) ? "Yes" : "No",
                   !(lo & (1 << c->bit)) ? "Yes" : "No");
                printk(msg);
        }
}

/*
 * detect_vmx_features
 *
 * Detects and prints VMX capabilities of this host's CPU.
 */
void
detect_vmx_features(void)
{
        uint32_t lo, hi;

        rdmsr(IA32_VMX_BASIC_CTLS, lo, hi);
        pr_info("Basic Controls MSR: 0x%llx\n",
                (uint64_t)(lo | (uint64_t)hi << 32));
        //report_capability(pinbased, 5, lo, hi);

        pr_info("\n");

        if(!((hi>>23)&& 1))
        {

                pr_info("True Controls capability is not set");

                pr_info("\n");

                /* Pinbased controls */
                rdmsr(IA32_VMX_PINBASED_CTLS, lo, hi);
                pr_info("Pinbased Controls MSR: 0x%llx\n",
                        (uint64_t)(lo | (uint64_t)hi << 32));
                report_capability(pinbased, 5, lo, hi);
```

```
		pr_info("\n");

		/* Procbased controls */
		rdmsr(IA32_VMX_PROCBASED_CTLS, lo, hi);
		pr_info("Procbased Controls MSR: 0x%llx\n",
			(uint64_t)(lo | (uint64_t)hi << 32));
		report_capability(procbased, 21, lo, hi);

		pr_info("\n");

		if((hi>>31)&& 1)
		{
			/* Procbased_ctls2 controls */
			rdmsr(IA32_VMX_PROCBASED_CTLS2, lo, hi);
			pr_info("Procbased_ctls2 Controls MSR: 0x%llx\n",
				(uint64_t)(lo | (uint64_t)hi << 32));
			report_capability(procbased_ctls2, 23, lo, hi);

		}
		else
		{
			pr_info("No capability set for “Activate Secondary Controls” control in the
primary procbased controls");
		}

		pr_info("\n");

		/* Exit controls */
		rdmsr(IA32_VMX_EXIT_CTLS, lo, hi);
		pr_info("Exit Controls MSR: 0x%llx\n",
			(uint64_t)(lo | (uint64_t)hi << 32));
		report_capability(exit1, 11, lo, hi);

		pr_info("\n");

		/* Entry controls */
		rdmsr(IA32_VMX_ENTRY_CTLS, lo, hi);
		pr_info("Entry Controls MSR: 0x%llx\n",
			(uint64_t)(lo | (uint64_t)hi << 32));
		report_capability(entry, 9, lo, hi);

	}
	else
	{

		pr_info("True Controls capability is set");

		pr_info("\n");
```

```
		/* True Pinbased controls */
		rdmsr(IA32_VMX_TRUE_PINBASED_CTLS, lo, hi);
		pr_info("True Pinbased Controls MSR: 0x%llx\n",
			(uint64_t)(lo | (uint64_t)hi << 32));
		report_capability(pinbased, 5, lo, hi);

		pr_info("\n");

		/* True Procbased controls */
		rdmsr(IA32_VMX_TRUE_PROCBASED_CTLS, lo, hi);
		pr_info("True Procbased Controls MSR: 0x%llx\n",
			(uint64_t)(lo | (uint64_t)hi << 32));
		report_capability(procbased, 21, lo, hi);

		pr_info("\n");

		/* True Exit controls */
		rdmsr(IA32_VMX_TRUE_EXIT_CTLS, lo, hi);
		pr_info("True Exit Controls MSR: 0x%llx\n",
			(uint64_t)(lo | (uint64_t)hi << 32));
		report_capability(exit1, 11, lo, hi);

		pr_info("\n");

		/* True Entry controls */
		rdmsr(IA32_VMX_TRUE_ENTRY_CTLS, lo, hi);
		pr_info("True Entry Controls MSR: 0x%llx\n",
			(uint64_t)(lo | (uint64_t)hi << 32));
		report_capability(entry, 9, lo, hi);
	}


}

/*
 * init_module
 *
 * Module entry point
 *
 * Return Values:
 *  Always 0
 */
int
init_module(void)
{
	printk(KERN_INFO "CMPE 283 Assignment 1 Module Start\n");

	detect_vmx_features();
```

```
	/*
	 * A non 0 return means init_module failed; module can't be loaded.
	 */
	cleanup_module();
	return 0;
}

/*
 * cleanup_module
 *
 * Function called on module unload
 */
void
cleanup_module(void)
{
	printk(KERN_INFO "CMPE 283 Assignment 1 Module Exits\n");
}
```

Output:

```
[ 9517.403597] CMPE 283 Assignment 1 Module Start
[ 9517.403602] Basic Controls MSR: 0xda040000000012

[ 9517.403605] True Controls capability is set

[ 9517.403610] True Pinbased Controls MSR: 0x7f00000016
[ 9517.403613]   External Interrupt Exiting: Can set=Yes, Can clear=Yes
[ 9517.403617]   NMI Exiting: Can set=Yes, Can clear=Yes
[ 9517.403620]   Virtual NMIs: Can set=Yes, Can clear=Yes
[ 9517.403623]   Activate VMX Preemption Timer: Can set=Yes, Can clear=Yes
[ 9517.403626]   Process Posted Interrupts: Can set=No, Can clear=Yes

[ 9517.403630] True Procbased Controls MSR: 0xfff9fffe04006172
[ 9517.403634]   Interrupt-window: Can set=Yes, Can clear=Yes
[ 9517.403637]   Use TSC offsetting: Can set=Yes, Can clear=Yes
[ 9517.403640]   HLT exiting: Can set=Yes, Can clear=Yes
[ 9517.403643]   INVLPG exiting: Can set=Yes, Can clear=Yes
[ 9517.403645]   MWAIT exiting: Can set=Yes, Can clear=Yes
[ 9517.403648]   RDPMC exiting: Can set=Yes, Can clear=Yes
[ 9517.403651]   RDTSC exiting: Can set=Yes, Can clear=Yes
[ 9517.403654]   CR3-load exiting: Can set=Yes, Can clear=Yes
[ 9517.403657]   CR3-store exiting: Can set=Yes, Can clear=Yes
[ 9517.403661]   CR8-load exiting: Can set=Yes, Can clear=Yes
[ 9517.403664]   CR8-store exiting: Can set=Yes, Can clear=Yes
[ 9517.403667]   Use TPR shadow: Can set=Yes, Can clear=Yes
[ 9517.403670]   NMI-window exiting: Can set=Yes, Can clear=Yes
[ 9517.403673]   MOV-DR exiting: Can set=Yes, Can clear=Yes
[ 9517.403676]   Unconditional I/O: Can set=Yes, Can clear=Yes
```

[9517.403680] Use I/O bitmaps: Can set=Yes, Can clear=Yes
[9517.403683] Monitor trap flag: Can set=Yes, Can clear=Yes
[9517.403686] Use MSR Bitmaps: Can set=Yes, Can clear=Yes
[9517.403689] MONITOR exiting: Can set=Yes, Can clear=Yes
[9517.403692] PAUSE exiting: Can set=Yes, Can clear=Yes
[9517.403696] Activate secondary controls: Can set=Yes, Can clear=Yes

[9517.403700] True Exit Controls MSR: 0x7fffff00036dfb
[9517.403703] Save debug controls: Can set=Yes, Can clear=Yes
[9517.403707] Host addressspace size: Can set=Yes, Can clear=Yes
[9517.403710] Load IA32_PERF_GLOB AL_CTRL: Can set=Yes, Can clear=Yes
[9517.403714] Acknowledge interrupt on exit: Can set=Yes, Can clear=Yes
[9517.403717] Save IA32_PAT: Can set=Yes, Can clear=Yes
[9517.403720] Load IA32_PAT: Can set=Yes, Can clear=Yes
[9517.403723] Save IA32_EEFR: Can set=Yes, Can clear=Yes
[9517.403750] Load IA32_EFER: Can set=Yes, Can clear=Yes
[9517.403765] Save VMXpreemption timer value: Can set=Yes, Can clear=Yes
[9517.403774] Clear IA32_BNDCFGS: Can set=No, Can clear=Yes
[9517.403787] Conceal VM exits from Intel PT: Can set=No, Can clear=Yes

[9517.403817] True Entry Controls MSR: 0xffff000011fb
[9517.403830] Load debug controls: Can set=Yes, Can clear=Yes
[9517.403842] IA-32e mode guest: Can set=Yes, Can clear=Yes
[9517.403855] Entry to SMM: Can set=Yes, Can clear=Yes
[9517.403869] Deactivate dual-monitor treatment: Can set=Yes, Can clear=Yes
[9517.403881] Load IA32_PERF_GLOBAL_CTRL: Can set=Yes, Can clear=Yes
[9517.403898] Load IA32_PAT: Can set=Yes, Can clear=Yes
[9517.403907] Load IA32_EFER: Can set=Yes, Can clear=Yes
[9517.403919] Load IA32_BNDCFGS: Can set=No, Can clear=Yes
[9517.403931] Conceal VM entries from Intel PT: Can set=No, Can clear=Yes
[9517.403944] CMPE 283 Assignment 1 Module Exits