Advanced Computer Architecture

JASHWANTH REDDY GANGULA

SUID : 646254141

**Lab2:**

**Task1 :**

#include <stdlib.h>

#include <stdio.h>

#include <stdbool.h>

struct tcs {

unsigned long enclave\_rsp\_heap;

unsigned long p\_saved\_regs\_e;

bool saved;

unsigned long res1;

unsigned long flags;

unsigned long ossa;

unsigned int cssa;

unsigned int nssa;

unsigned long oentry; // entry point

unsigned long res2;

unsigned long ofsbasgx;

unsigned long ogsbasgx;

unsigned int fslimit;

unsigned int gslimit;

char res3[4024];

};

struct tcs\* p\_tcs;

void enclave\_program() {

printf("We are now in the enclave, cheers!\n");

exit(0);

}

int main() {

p\_tcs = (struct tcs \*)malloc(sizeof(struct tcs));

p\_tcs->oentry = (unsigned long)enclave\_program;

asm volatile (

"mov %0, %%rbx\n\t"

"movl $0x02, %%eax\n\t" // EENTER, instruction type

".byte 0x0f,0x01,0xd7\n\t" // call opcode 01d7

:

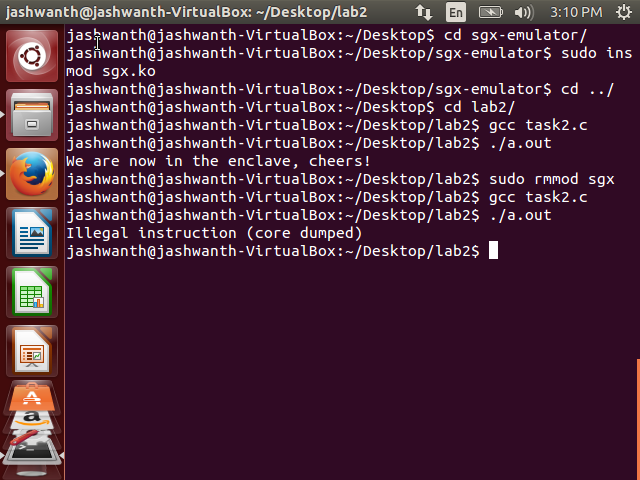
: "r" (p\_tcs)

: "eax", "rbx");

return 0;

}

Task1 and Task2 Screenshots:



**Observation:**

Here Initially we have loaded sgx emulator and hence the program works correctly as if the CPU is SGX-equipped CPU. After unloading the sgx we get illegal instruction as shown in the screenshot above.

**Lab 3:**

**Task1.c**

#include <stdlib.h>

#include <stdio.h>

#include <stdbool.h>

struct tcs {

unsigned long enclave\_rsp\_heap;

unsigned long p\_saved\_regs\_e;

bool saved;

unsigned long res1;

unsigned long flags;

unsigned long ossa;

unsigned int cssa;

unsigned int nssa;

unsigned long oentry; // entry point

unsigned long res2;

unsigned long ofsbasgx;

unsigned long ogsbasgx;

unsigned int fslimit;

unsigned int gslimit;

char res3[4024];

};

struct tcs\* p\_tcs;

void enclave\_program()

{

int arg;

asm volatile("mov %%edx, %0\n\t"

: "=r" ( arg) ::

);

printf("We are now in the enclave and argument "

"is %d,cheers!\n",arg);

exit(0);

}

int main()

{

p\_tcs = (struct tcs \*)malloc(sizeof(struct tcs));

p\_tcs->oentry = (unsigned long)enclave\_program;

int arg = 661;

asm volatile ( "mov %0, %%rbx\n\t"

"movl %1, %%edx\n\t"

"movl $0x02, %%eax\n\t" // EENTER, instruction type

".byte 0x0f,0x01,0xd7\n\t" // call opcode 01d7

:

: "r" (p\_tcs),"r"(arg)

: "eax", "rbx" ,"rdx"

);

return 0;

}

**Task2.c**

#include <stdlib.h>

#include <stdio.h>

#include <stdbool.h>

struct tcs {

unsigned long enclave\_rsp\_heap;

unsigned long p\_saved\_regs\_e;

bool saved;

unsigned long res1;

unsigned long flags;

unsigned long ossa;

unsigned int cssa;

unsigned int nssa;

unsigned long oentry; // entry point

unsigned long res2;

unsigned long ofsbasgx;

unsigned long ogsbasgx;

unsigned int fslimit;

unsigned int gslimit;

char res3[4024];

};

struct tcs\* p\_tcs;

struct enclave\_args

{

int val;

char\* msg;

};

void enclave\_program()

{

struct enclave\_args \*my\_arg;

asm volatile ("mov %%rdx, %0\n\t"

: "=r" (my\_arg) ::

);

printf("We are now in the enclave and val is %d,"

"msg=%s,cheers!\n",my\_arg->val,my\_arg->msg);

exit(0);

}

int main()

{

p\_tcs = (struct tcs \*)malloc(sizeof(struct tcs));

p\_tcs->oentry = (unsigned long)enclave\_program;

struct enclave\_args args;

args.val = 661;

args.msg = "Computer Architecture!";

asm volatile ( "mov %0, %%rbx\n\t"

"mov %1, %%rdx\n\t"

"movl $0x02, %%eax\n\t" // EENTER, instruction type

".byte 0x0f,0x01,0xd7\n\t" // call opcode 01d7

:

: "r" (p\_tcs),"r"(&args)

: "eax", "rbx" ,"rdx"

);

return 0;

}

**Task3.c:**

#include <stdlib.h>

#include <stdio.h>

#include <stdbool.h>

struct tcs {

unsigned long enclave\_rsp\_heap;

unsigned long p\_saved\_regs\_e;

bool saved;

unsigned long res1;

unsigned long flags;

unsigned long ossa;

unsigned int cssa;

unsigned int nssa;

unsigned long oentry; // entry point

unsigned long res2;

unsigned long ofsbasgx;

unsigned long ogsbasgx;

unsigned int fslimit;

unsigned int gslimit;

char res3[4024];

};

struct tcs\* p\_tcs;

void enclave\_program() {

void \*return\_address;

asm volatile ( "mov %%rcx, %0\n\t"

: "=r" (return\_address) ::

);

printf("We are now in the enclave, cheers!\n");

asm volatile ("mov %0, %%rbx \n\t"

"movl $0x04 , %%eax\n\t" // EEXIT

".byte 0x0f,0x01,0xd7"

:

: "r" (return\_address));

}

int main() {

p\_tcs = (struct tcs \*)malloc(sizeof(struct tcs));

p\_tcs->oentry = (unsigned long)enclave\_program;

asm volatile ("mov %0, %%rbx\n\t"

"movl $0x02, %%eax\n\t" // EENTER, instruction type

".byte 0x0f,0x01,0xd7\n\t" // call opcode 01d7

:

: "r" (p\_tcs)

: "eax", "rbx"

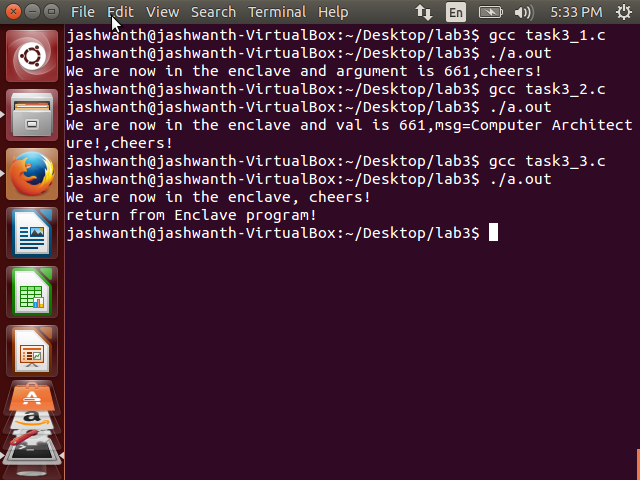
);

printf("return from Enclave program!\n");

return 0;

}

Screenshot of outputs of three tasks:



We are passing the address p\_tcs to the register rbx, 0x02 into the register eax which is same for all the programs. In the first task , we are passing the argument into the edx register. In the second task,

To pass multiple arguments we are holding all the arguments in a single data structure and passing the address of the structure.In the third taks after the execution of enclave program, we are returning to the main program with the help of return address. The main program passes the return address with rcx register , The enclave program after its completion issues EEXIT(0x04) along with the return address to rbx register. Please observe the outputs of all the programs printing correctly in the screenshot above.