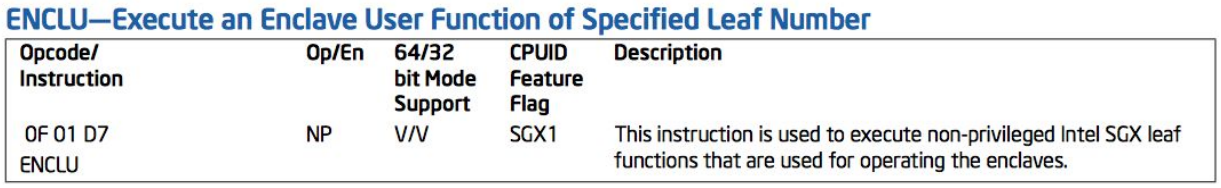
1 Lab overview

The purpose of the lab is to help students to gain hands-on experience on CPU instructions, by using concrete examples and writing real-world C programs. In this lab, students will be given an incomplete C program; their task is to fill in the program with important arguments and constants so that the program could work. We assume students have finished Lab 1 and gained basic knowledge about assembly language.

An important thing to complete the tasks is to have a basic understanding of SGX and the format of Enclave instructions. The SGX manual can be found in here

https://software.intel.com/sites/default/files/managed/48/88/329298-002.pdf

However, reading through the entire document would be burdensome and time consuming. Here we provide several key information for you to complete the task. Specifically, in Section 5.2, you can find the opcode for the ENCLU instruction. Also, in table 5.2, you can find the register usage for each Enclave leaf functions.



ENCLU—Execute an Enclave User Function of Specified Leaf Number

Opcode/ Instruction	Op/En	64/32 bit Mode Support	CPUID Feature Flag	Description
0F 01 D7 ENCLU	NP	V/V	SGX1	This instruction is used to execute non-privileged Intel SGX leaf functions that are used for operating the enclaves.

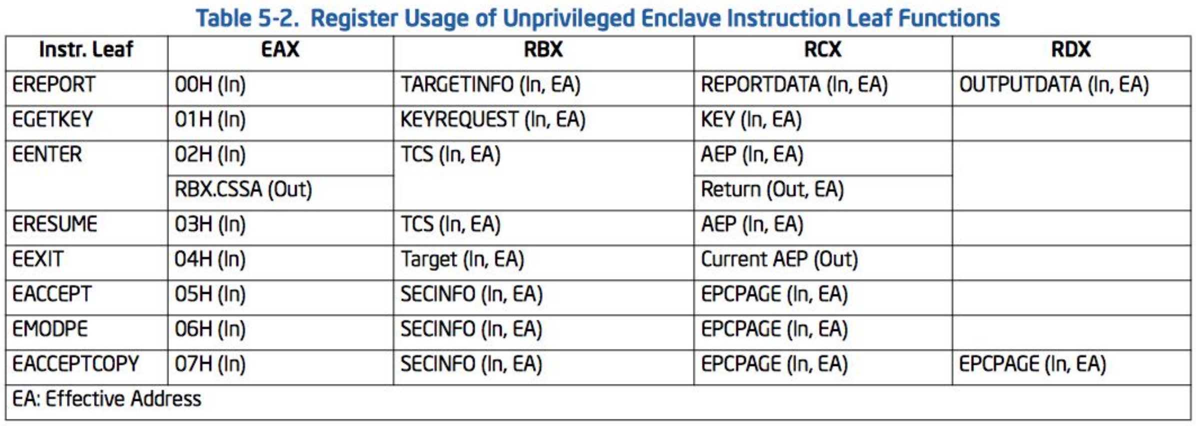


Table 5-2. Register Usage of Unprivileged Enclave Instruction Leaf Functions

Instr. Leaf	EAX	RBX	RCX	RDX
EREPORT	00H (In)	TARGETINFO (In, EA)	REPORTDATA (In, EA)	OUTPUTDATA (In, EA)
EGETKEY	01H (In)	KEYREQUEST (In, EA)	KEY (In, EA)	
EENTER	02H (In)	TCS (In, EA)	AEP (In, EA)	
	RBX.CSSA (Out)		Return (Out, EA)	
ERESUME	03H (In)	TCS (In, EA)	AEP (In, EA)	
EEXIT	04H (In)	Target (In, EA)	Current AEP (Out)	
EACCEPT	05H (In)	SECINFO (In, EA)	EPCPAGE (In, EA)	
EMODPE	06H (In)	SECINFO (In, EA)	EPCPAGE (In, EA)	
EACCEPTCOPY	07H (In)	SECINFO (In, EA)	EPCPAGE (In, EA)	EPCPAGE (In, EA)
EA: Effective Address				

For example, in order to correctly issue an EENTER instruction, you need to first load EAX register with leaf number (0x02) and load RBX register with the address of TCS data structure. (For simplicity, we ignore other arguments for this lab.)

2 Lab Tasks

For each of the lab task, please capture the screen-shot of the key step and include it in your lab report.

2.1 Initial setup

Program list 1 shows an incomplete C program. This program demonstrates how to issue a CPU instruction called EENTER to enter enclave program.

```
#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 $0x???, %%eax\n\t" // EENTER, instruction type
                  ".byte 0x0f,0x01,0xd7\n\t" // call opcode 01d7
                  :
                  : "r" (???)
                  : "eax", "rbx"
                  );
      return 0;
}
```

Program List 1

2.2 Task 1

Replace the question mark (in red) with concrete numbers and variable names, compile the program and run it, so that it throws the following output.

```
We are now in the enclave, cheers!
```

Before you run the program, use the following command to load SGX emulator in your system, you could find it under sgx_emuluator directory. (See Lab 0 for details)

```
sudo insmod sgx.ko
```

2.2 Task 2

The EENTER instruction only exists on a SGX-equipped CPU. So why can you run the instruction even if there's no SGX support on your machine? Well, this is because we've emulated the SGX instruction (by software) for you.

Run the following command to unload SGX emulator from system, re-run the above program, describe whatever you've observed.

```
sudo rmmod sgx
```