

CS302 Lab4 Report

何泽安 12011323

2023.3.8

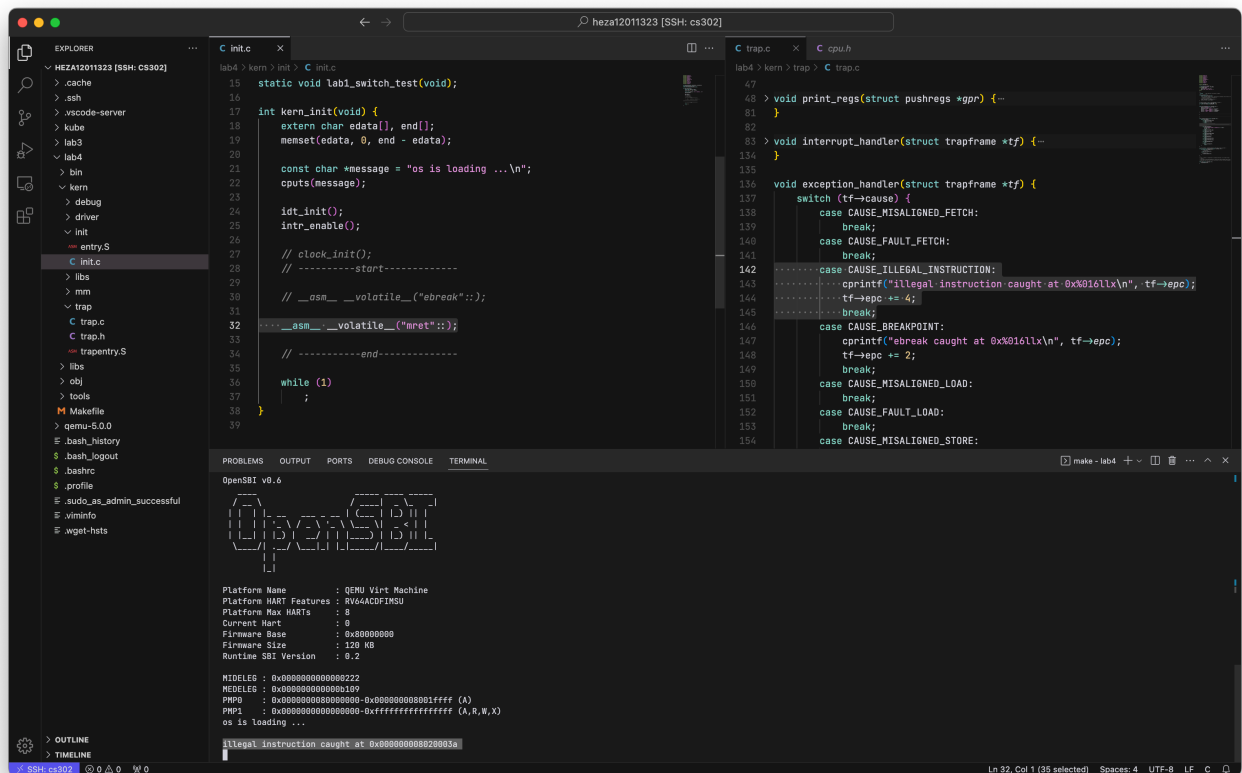
1. How does the OS handle the breakpoint interrupt

In the illustration, we used a `asm volatile("ebreak");` to make a breakpoint signal, when catching an interrupt, OS first save the context, then find the appropriate interruption handler from the `stvec` and call (jump) the handler method. After the method returns, jump back to the place where interrupted, reload the saved context, and use a `sret` to set PC as `spec`, and go back to the program before entering S mode.

2. The functionality of `epc` register

The `epc` (stands for exception program counter) register stores the address of the instruction that caused the exception. The OS can there by set the PC as the value in `epc`, such that it can jump back to the program after handling the exception.

3. Intentionally cause a `ILLEGAL_INSTRUCTION` exception, and handle it



```
lab4 > kern > init > C init.c
15 static void lab1_switch_test(void);
16
17 int kern_init(void) {
18     extern char edata[], end[];
19     memset(edata, 0, end - edata);
20
21     const char *message = "os is loading ... \n";
22     cputs(message);
23
24     idt_init();
25     intr_enable();
26
27     // clock_init();
28     // -----start-----
29     // __asm__ __volatile__ ("mret");
30
31     // -----end-----
32     while (1)
33     {
34     }
35 }
36
37
38
39
```

```
lab4 > kern > trap > C trap.c
47
48 > void print_regs(struct pushregs *gpr) {--
81 }
82
83 > void interrupt_handler(struct trapframe *tf) {--
134
135 void exception_handler(struct trapframe *tf) {
136     switch (tf->cause) {
137         case CAUSE_MISALIGNED_FETCH:
138             break;
139         case CAUSE_FAULT_FETCH:
140             break;
141         case CAUSE_ILLEGAL_INSTRUCTION:
142             printf("illegal instruction caught at 0x%016lx\n", tf->epc);
143             tf->epc += 4;
144             break;
145         case CAUSE_BREAKPOINT:
146             printf("ebreak caught at 0x%016lx\n", tf->epc);
147             tf->epc += 2;
148             break;
149         case CAUSE_MISALIGNED_LOAD:
150             break;
151         case CAUSE_FAULT_LOAD:
152             break;
153         case CAUSE_MISALIGNED_STORE:
154             break;
155     }
156 }
```

```
OpenSBI v0.6
Platform Name      : QEMU Virt Machine
Platform HART Features : RV64ACDFJMSU
Platform Max HARTs   : 8
Current Hart       : 0
Firmware Base      : 0x00000000
Firmware Size      : 120 KB
Runtime SBI Version : 0.2
MIDELG : 0x00000000000000222
MEDELG : 0x0000000000000b109
PRNG : 0x0000000000000000-0x0000000000000000 (A)
PRNG : 0x0000000000000000-0x0000000000000000 (A,R,W,X)
os is loading ...
Illegal instruction caught at 0x0000000000000000
```

We call `mret` in U mode, which will cause an illegal instruction exception, and caught by the handler. The handler use the correct handler by switching the cause. We first print a prompt, then increase the `epc` by 4, since the `mret` instruction is 32 bits long.