1. qemu-system-riscv64: emulate a riscv64 cpu

-machine virt 用于指定虚拟机的类型为 virt , virt 是一个基于 kvm 和 qemu 的虚拟化平台 -nographic 禁用图形输出,将虚拟机的控制台重定向到当前终端 -bios default 用于指定虚拟机的BIOS为默认值,这是一个固件程序,用于启动虚拟机并设置其硬件

-device loader, file=bin/ucore.bin, addr=0x80200000: 这是 QEMU 的选项之一,用于指定虚拟机的设备为一个 loader,从指定的文件 bin/ucore.bin 中加载内核,并将其放置在物理地址 0x80200000 处。这个指令将指向 makefile 对应的label,也就是在 makefile 中定义了这些选项,当我们执行 make qemu 时, makefile 会将这些选项传递给 QEMU 仿真器,以启动虚拟机并加载内核。

```
2.
       . = BASE_ADDRESS;
                                  // move current address to BASE_ADDRESS
                                   // link .text sections
       .text : {
           *(.text.kern_entry)
               // link .text.kern_entry first
               // ,to make sure .text.kern_entry label is located at
   BASE_ADDRESS
            *(.text .stub .text.* .gnu.linkonce.t.*)
               // then link other labels under .text sections
       }
       PROVIDE(etext = .); // define symbol etext to be current address
                           // link section .rodata, readonly data
        .rodata : {
            *(.rodata .rodata.* .gnu.linkonce.r.*)
               // link sections (.rodata .rodata.*, .gnu.linkonce.r.*) from any
   object files
       }
        . = ALIGN(0x1000); // Make current address aligned to page size
```

- 3. memset(edata, 0, end edata) 用于初始化内核的数据段。edata: 这是一个指向符号 edata 的指针,它指向的是内核数据段的结尾地址。在编译链接过程中,编译器会将 edata 指向数据段的最后一个已初始化的地址,也就是说,edata 指向数据段中最后一个已经有值的字节的后面一位。0: 这是要用来初始化数据段的值,这里是0,即将数据段中的所有字节都置为0。end edata: 这是要初始化的数据段的大小,它的值等于 end 和 edata 之间的距离。 end 是一个符号,它指向数据段的结束地址,也就是数据段最后一个字节的后面一位。 memset(edata, 0, end edata) 这行代码的作用是: 将内核数据段中从 edata 到 end 的所有字节都设置为0,也就是清空数据段中未被初始化的部分。这是为了确保未被初始化的内存空间中不会出现无效数据,以避免在运行内核时发生未定义的行为。
- 4. cputs() 函数是用于在终端上打印字符串的函数。在 ucore 操作系统中, cputs() 函数通过调用 SBI规范中的 console\_putchar() 函数来实现在终端上打印字符的功能。对于每个传递进来的字符, console\_putchar() 函数会将传递过来的字符放入一个名为 a0 的寄存器中,然后使用 ecall 指令触发SBI异常,将控制权转移给SBI。SBI会根据 a7 寄存器中存储的系统调用号来执行对应的操作,对于 console\_putchar() 函数,它的系统调用号为`SBI\_CONSOLE\_PUTCHAR

```
5. dgy@dgy-ubuntu ~/g/o/l/lab (master) [2]> make qemu
   + cc kern/trap/trap.c
   riscv64-unknown-elf-objcopy bin/kernel --strip-all -0 binary bin/ucore.bin
   Platform Name
   Current Hart
miliciane verabilis
#include <sbi.h>
int kern_init(void) __attribute__((noreturn));
void grade backtrace(void);
static void lab1_switch_test(void);
int kern init(void) {
    extern char edata[], end[];
    memset(edata, 0, end - edata);
    const char *message = "os is loading ...\n";
    cputs(message);
    idt_init();
    intr_enable();
    // asm volatile("ebreak"::);
    // cputs("start testing illegal instructions");
    // asm volatile("mret"::);
    // cputs("end testing illegal instructions");
    cputs("The system will close.\n");
    sbi shutdown();
    clock_init();
    // -----start-----
```

```
void sbi_shutdown(void){
    sbi_call(SBI_SHUTDOWN, 0, 0, 0);
}
```