练习1

解决bug

由于虚拟机老死机，本人决定毅然放弃使用Vmware，转而去使用wsl（没有图形化界面）。开始时，用vscode链接到wsl子系统，并没有任何影响，可以正常地执行make指令。

但是，当我需要执行make qemu指令时，却报了两个错，一个是没有软连接，在执行

```
sudo ln –s /usr/local/bin/qemu-system-i386  /usr/local/bin/qemu //软连接操作
```

而第二个错：**没有链接终端**，在查阅资料过后，发现qemu和gdb两个工具都需要图形化的终端才能执行，所以在wsl安装了xrdp远程终端界面，解决问题！！！

练习2

```
file bin/kernel //指定调试目标文件，让gdb获得符号信息
target remote :1234 //设置远程连接端口为qemu的运行端口1234，连接到qemu
set architecture i8086 //指定qemu要模拟的硬件架构
b *0x7c00 //在bootloader开始地址0x7c00处下断点
continue //开始调试，执行到刚才指定的断点
/*
进入gdb页面以后
*/
layout asm //显示汇编可视化窗口
b <address>//设置断点
x /<n>i $pc //以十六进制格式打印机器指令,n为想要多少条指令
```

解决Bug

当我"make debug"的时候，显示"other progress is running",因为没有打开别的进程,一直以为是qemu环境没有配置好,重新install 依然报错,最终命令行中输入top，手动kill所有和qemu有关的进程。

```
kill <和qemu有关的进程id>
```

问题解决！！！

练习5

依据代码提示，简单翻译即可打印栈。实现详情如下：

```
void print_stackframe(void) {
     uint32_t ebp=read_ebp();//ebp
     uint32_t eip=read_eip();//eip
     for (int i = 0; ebp != 0 &&i < STACKFRAME_DEPTH; i++)
     {
        //printf value of ebp, eip
        cprintf("EBP:0x%08x \t EIP:0x%08x \t 参数：",ebp,eip);
        //EBP->返回地址->参数
        uint32_t* arguments=(uint32_t*)ebp +2;
        for(int j=0;j<4;j++)
        {
            //参数从右向左压栈，从左向右打印
            cprintf("0x%08x ", arguments[j]);
        }
        cprintf("\n");
        //print_debuginfo(eip-1)->calling function name and line number,
etc.
        print_debuginfo(eip - 1);
        //返回地址+4，栈底不变
        eip = *((uint32_t*)ebp +1);
        ebp = *((uint32_t *)ebp);
     }
}
```

本次练习中没有遇见Bug。

- ss[ebp]指向上一层的ebp
- ss[ebp-4]指向局部变量
- ss[ebp+4]指向返回地址
- ss[ebp+4+4n]指向第n个参数

练习6

```
void idt_init(void) {
     /* LAB1 YOUR CODE : STEP 2 */
     //"extern uintptr_t __vectors[];"全局变量
    extern uintptr_t __vectors[];
    // 在中断向量表中建立索引，SETGATE macro
    for (int i = 0; i < sizeof(idt) / sizeof(struct gatedesc); i ++) {
        SETGATE(idt[i], 0, GD_KTEXT, __vectors[i], DPL_KERNEL);
    }
    //最后从DPL_USER跳转至DPL_KERNEL
    SETGATE(idt[T_SWITCH_TOK], 0, GD_KTEXT, __vectors[T_SWITCH_TOK],
DPL_USER);
    // 让cpu知道idt位置
    lidt(&idt_pd);
}
```

- 对中断向量表进行初始化的函数idt_init。在idt_init函数中，依次对所有中断入口进行初始化。使用mmu.h中的SETGATE宏，填充idt数组内容。
- 在对时钟中断进行处理的部分填写trap函数中处理时钟中断的部分，使操作系统每遇到100次时钟中断后，调用print_ticks子程序，向屏幕上打印一行文字”100 ticks”。