1、代码修改

在os/syscall.c中添加了判断：当str的范围不在用户代码地址范围或用户栈地址范围的话，返回-1

2、思考题

1

RustSBI version 0.1.1

IllegalInstruction -> 空指针访问

IllegalInstruction -> 访问S态寄存器

2

1、a0为trapframe地址，a1为用户页表地址

2、

清理虚实地址转换的cache

为了处理虚存和页表相关的内容

不会错误，因为现在没有页表

3、

因为a0当前保存的trapframe地址，不能丢失trapframe地址的信息

112(a0)代表a0的地址

sscratch存了a0的值

4、

sret

因为在usertrapret()函数里设置了sepc和sstatus

5、

第六项

因为前五项是内核态提供的内容，不由用户态提供

除了a0都保存了

因为不能丢失trapframe地址

6、jr t0

7、

t0的值是usertrap的地址

在从S态返回U态时，usertrapret()将trapframe->kernel_trap设定为usertrap的地址

3

Interrupt	Exception Code	Description
0	0	Instruction address misaligned
0	1	Instruction access fault
0	2	Illegal instruction
0	3	Breakpoint
0	4	Load address misaligned
0	5	Load access fault
0	6	Store/AMO address misaligned
0	7	Store/AMO access fault
0	8	Environment call from U-mode
0	9	Environment call from S-mode
0	11	Environment call from M-mode
0	12	Instruction page fault
0	13	Load page fault
0	15	Store/AMO page fault

mcause 最高位为1则为中断，反之为异常

重要寄存器：

CSR 名	该 CSR 与 Trap 相关的功能
sstatus	`SPP` 等字段给出 Trap 发生之前 CPU 处在哪个特权级（S/U）等信息
sepc	当 Trap 是一个异常的时候，记录 Trap 发生之前执行的最后一条指令的地址
scause	描述 Trap 的原因
stval	给出 Trap 附加信息
stvec	控制 Trap 处理代码的入口地址

4

不需要

加速方法：不存不需要的寄存器，比如本实验中的tp