Lab1

4.2 编写`head.S`

```
.extern start_kernel

    .section .text.entry
    .global _start
_start:
    la sp, boot_stack_top       #初始化$sp
    call start_kernel           #调用第一个函数start_kernel

    .section .bss.stack
    .globl boot_stack
boot_stack:
    .space 4096 * 4             #4K大小的栈空间

    .globl boot_stack_top
boot_stack_top:
```

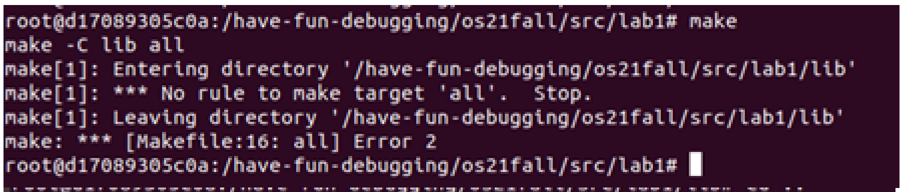
4.3 完善`Makefile`脚本

简单分析一下，这个`makefile`中需要完成的部分有：

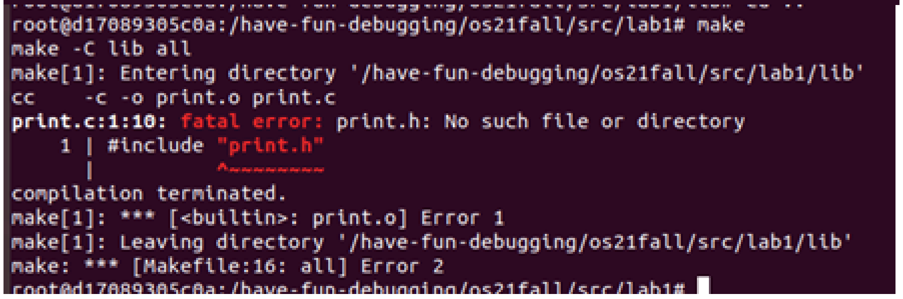
- 指定两个头文件的路径
- 确定目标和依赖
- 设定编译动作

```
VPATH = ../include:../arch/riscv/include    #print.h和sbi.h的路径

objects = print.o                           #虽然只有一个文件，但好像这样看上去比较正儿
八经

all : $(objects)
    $(GCC) -o edit $(objects) $(CFLAG)      #根目录下的makefile在引用这个makefile的时
候执行了all操作，所以需要这样写

print.o : print.c print.h sbi.h
    $(GCC) -c print.c $(CFLAG)

.PHONY : clean
clean :
    -rm all $(objects)
```

中间因为没有写all操作和路径设定不正确出现了诸如此类的一些小问题。

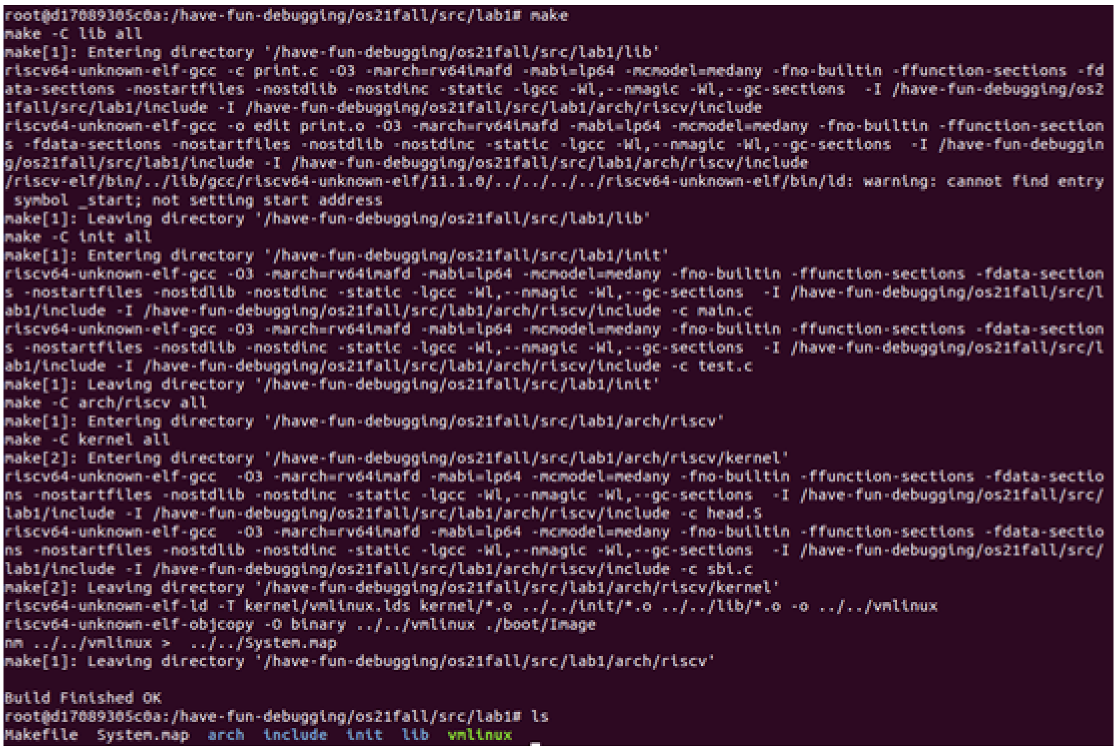


```
root@d17089305c0a:/have-fun-debugging/os21fall/src/lab1# make
make -C lib all
make[1]: Entering directory '/have-fun-debugging/os21fall/src/lab1/lib'
make[1]: *** No rule to make target 'all'.  Stop.
make[1]: Leaving directory '/have-fun-debugging/os21fall/src/lab1/lib'
make: *** [Makefile:16: all] Error 2
root@d17089305c0a:/have-fun-debugging/os21fall/src/lab1# 
```



```
root@d17089305c0a:/have-fun-debugging/os21fall/src/lab1# make
make -C lib all
make[1]: Entering directory '/have-fun-debugging/os21fall/src/lab1/lib'
cc     -c -o print.o print.c
print.c:1:10: fatal error: print.h: No such file or directory
    1 | #include "print.h"
      |          ^~~~~~~~~
compilation terminated.
make[1]: *** [<builtin>: print.o] Error 1
make[1]: Leaving directory '/have-fun-debugging/os21fall/src/lab1/lib'
make: *** [Makefile:16: all] Error 2
root@d17089305c0a:/have-fun-debugging/os21fall/src/lab1# 
```

回到根目录，顺利编译。



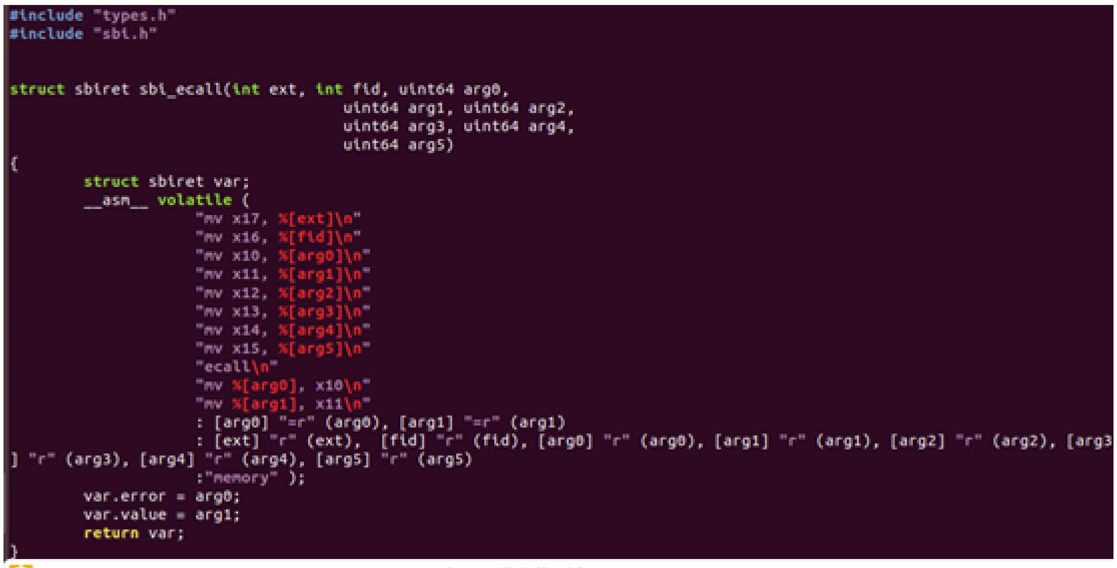
```
root@d17089305c0a:/have-fun-debugging/os21fall/src/lab1# make
make -C lib all
make[1]: Entering directory '/have-fun-debugging/os21fall/src/lab1/lib'
riscv64-unknown-elf-gcc -c print.c -O3 -march=rv64imafd -mabi=lp64 -mcmodel=medany -fno-builtin -ffunction-sections -fd
ata-sections -nostartfiles -nostdlib -nostdinc -static -lgcc -Wl,--nmagic -Wl,--gc-sections  -I /have-fun-debugging/os2
1fall/src/lab1/include -I /have-fun-debugging/os21fall/src/lab1/arch/riscv/include
riscv64-unknown-elf-gcc -o edit print.o -O3 -march=rv64imafd -mabi=lp64 -mcmodel=medany -fno-builtin -ffunction-section
s -fdata-sections -nostartfiles -nostdlib -nostdinc -static -lgcc -Wl,--nmagic -Wl,--gc-sections  -I /have-fun-debuggin
g/os21fall/src/lab1/include -I /have-fun-debugging/os21fall/src/lab1/arch/riscv/include
/riscv-elf/bin/../lib/gcc/riscv64-unknown-elf/11.1.0/../../../../riscv64-unknown-elf/bin/ld: warning: cannot find entry
 symbol _start; not setting start address
make[1]: Leaving directory '/have-fun-debugging/os21fall/src/lab1/lib'
make -C init all
make[1]: Entering directory '/have-fun-debugging/os21fall/src/lab1/init'
riscv64-unknown-elf-gcc -O3 -march=rv64imafd -mabi=lp64 -mcmodel=medany -fno-builtin -ffunction-sections -fdata-section
s -nostartfiles -nostdlib -nostdinc -static -lgcc -Wl,--nmagic -Wl,--gc-sections  -I /have-fun-debugging/os21fall/src/l
ab1/include -I /have-fun-debugging/os21fall/src/lab1/arch/riscv/include -c main.c
riscv64-unknown-elf-gcc -O3 -march=rv64imafd -mabi=lp64 -mcmodel=medany -fno-builtin -ffunction-sections -fdata-section
s -nostartfiles -nostdlib -nostdinc -static -lgcc -Wl,--nmagic -Wl,--gc-sections  -I /have-fun-debugging/os21fall/src/l
ab1/include -I /have-fun-debugging/os21fall/src/lab1/arch/riscv/include -c test.c
make[1]: Leaving directory '/have-fun-debugging/os21fall/src/lab1/init'
make -C arch/riscv all
make[1]: Entering directory '/have-fun-debugging/os21fall/src/lab1/arch/riscv'
make -C kernel all
make[2]: Entering directory '/have-fun-debugging/os21fall/src/lab1/arch/riscv/kernel'
riscv64-unknown-elf-gcc  -O3 -march=rv64imafd -mabi=lp64 -mcmodel=medany -fno-builtin -ffunction-sections -fdata-sectio
ns -nostartfiles -nostdlib -nostdinc -static -lgcc -Wl,--nmagic -Wl,--gc-sections  -I /have-fun-debugging/os21fall/src/
lab1/include -I /have-fun-debugging/os21fall/src/lab1/arch/riscv/include -c head.S
riscv64-unknown-elf-gcc  -O3 -march=rv64imafd -mabi=lp64 -mcmodel=medany -fno-builtin -ffunction-sections -fdata-sectio
ns -nostartfiles -nostdlib -nostdinc -static -lgcc -Wl,--nmagic -Wl,--gc-sections  -I /have-fun-debugging/os21fall/src/
lab1/include -I /have-fun-debugging/os21fall/src/lab1/arch/riscv/include -c sbi.c
make[2]: Leaving directory '/have-fun-debugging/os21fall/src/lab1/arch/riscv/kernel'
riscv64-unknown-elf-ld -T kernel/vmlinux.lds kernel/*.o ../../init/*.o ../../lib/*.o -o ../../vmlinux
riscv64-unknown-elf-objcopy -O binary ../../vmlinux ./boot/Image
nm ../../vmlinux >  ../../System.map
make[1]: Leaving directory '/have-fun-debugging/os21fall/src/lab1/arch/riscv'

Build Finished OK
root@d17089305c0a:/have-fun-debugging/os21fall/src/lab1# ls
Makefile  System.map  arch  include  init  lib  vmlinux
```

4.4 补充`sbi.c`

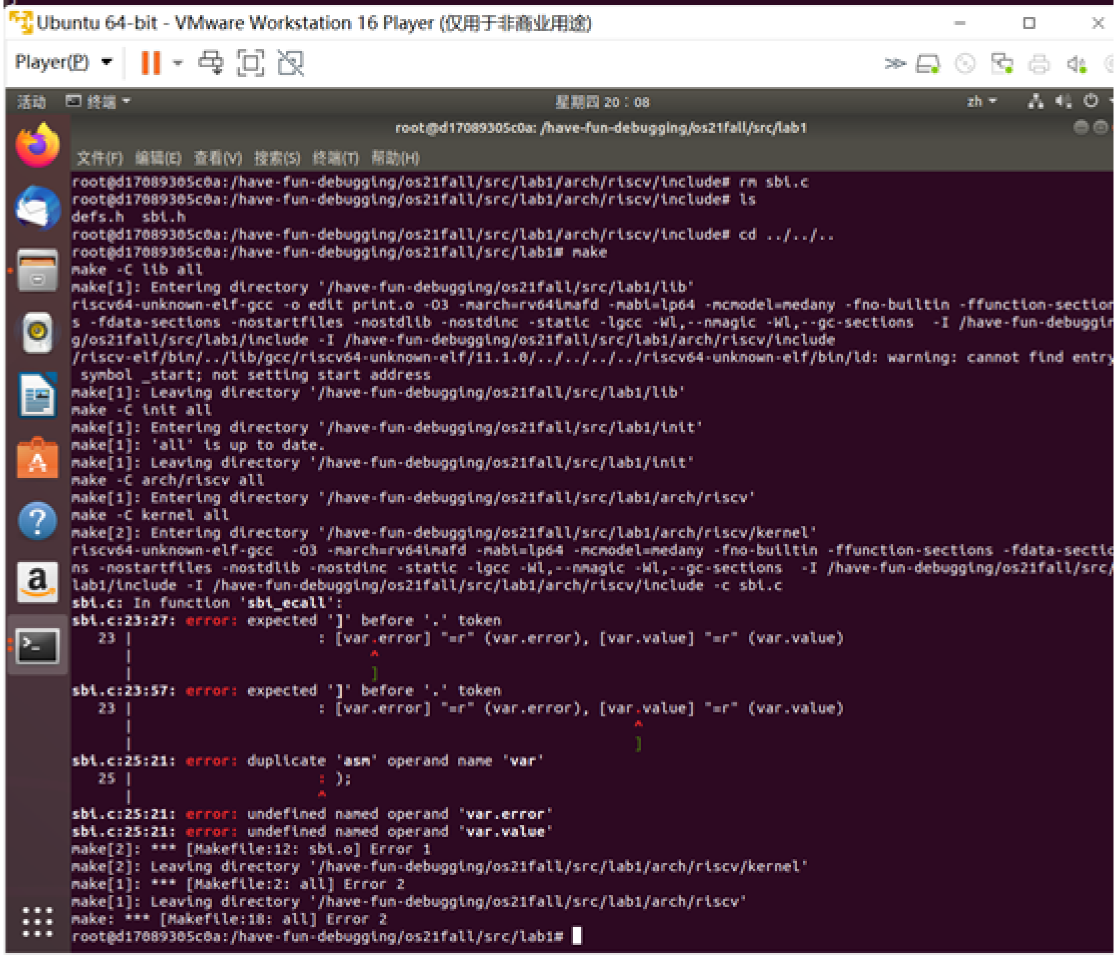
```c
#include "type.h"
#include "sbi.h"

struct sbiret sbi_ecall(int ext, int fid,
                        uint64 arg0, uint64 arg1, uint64 arg2,
                        uint64 arg3, uint64 arg4, uint64 arg5)
{
    struct sbiret var;
    register uint64 a0 asm ("a0") = (uint64)(arg0);
    register uint64 a0 asm ("a1") = (uint64)(arg1);
    register uint64 a0 asm ("a2") = (uint64)(arg2);
    register uint64 a0 asm ("a3") = (uint64)(arg3);
    register uint64 a0 asm ("a4") = (uint64)(arg4);
    register uint64 a0 asm ("a5") = (uint64)(arg5);
    register uint64 a0 asm ("a6") = (uint64)(arg6);
    register uint64 a0 asm ("a7") = (uint64)(arg7);

    asm volatile (
```

```
        "ecall"
        : "+r" (a0), "+r" (a1)
        : "r" (a2), "r" (a3), "r" (a4), "r" (a5), "r" (a6), "r" (a7)
        : "memory");
    var.error = a0;
    var.value = a1;
    return var;
};
```

一开始采用了如图的写法，结果参数互相冲突。



```
#include "types.h"
#include "sbi.h"


struct sbiret sbi_ecall(int ext, int fid, uint64 arg0,
                                    uint64 arg1, uint64 arg2,
                                    uint64 arg3, uint64 arg4,
                                    uint64 arg5)
{
        struct sbiret var;
        __asm__ volatile (
                        "mv x17, %[ext]\n"
                        "mv x16, %[fid]\n"
                        "mv x10, %[arg0]\n"
                        "mv x11, %[arg1]\n"
                        "mv x12, %[arg2]\n"
                        "mv x13, %[arg3]\n"
                        "mv x14, %[arg4]\n"
                        "mv x15, %[arg5]\n"
                        "ecall\n"
                        "mv %[arg0], x10\n"
                        "mv %[arg1], x11\n"
                        : [arg0] "=r" (arg0), [arg1] "=r" (arg1)
                        : [ext] "r" (ext),  [fid] "r" (fid), [arg0] "r" (arg0), [arg1] "r" (arg1), [arg2] "r" (arg2), [arg3
] "r" (arg3), [arg4] "r" (arg4), [arg5] "r" (arg5)
                        :"memory" );
        var.error = arg0;
        var.value = arg1;
        return var;
}
```



```
root@d17089305c0a:/have-fun-debugging/os21fall/src/lab1/arch/riscv/include# rm sbi.c
root@d17089305c0a:/have-fun-debugging/os21fall/src/lab1/arch/riscv/include# ls
defs.h  sbi.h
root@d17089305c0a:/have-fun-debugging/os21fall/src/lab1/arch/riscv/include# cd ../../..
root@d17089305c0a:/have-fun-debugging/os21fall/src/lab1# make
make -C lib all
make[1]: Entering directory '/have-fun-debugging/os21fall/src/lab1/lib'
riscv64-unknown-elf-gcc -o edit print.o -O3 -march=rv64imafd -mabi=lp64 -mcmodel=medany -fno-builtin -ffunction-sectio
s -fdata-sections -nostartfiles -nostdlib -nostdinc -static -lgcc -Wl,--nmagic -Wl,--gc-sections  -I /have-fun-debuggi
g/os21fall/src/lab1/include -I /have-fun-debugging/os21fall/src/lab1/arch/riscv/include
/riscv-elf/bin/../lib/gcc/riscv64-unknown-elf/11.1.0/../../../../riscv64-unknown-elf/bin/ld: warning: cannot find entry
 symbol _start; not setting start address
make[1]: Leaving directory '/have-fun-debugging/os21fall/src/lab1/lib'
make -C init all
make[1]: Entering directory '/have-fun-debugging/os21fall/src/lab1/init'
make[1]: 'all' is up to date.
make[1]: Leaving directory '/have-fun-debugging/os21fall/src/lab1/init'
make -C arch/riscv all
make[1]: Entering directory '/have-fun-debugging/os21fall/src/lab1/arch/riscv'
make -C kernel all
make[2]: Entering directory '/have-fun-debugging/os21fall/src/lab1/arch/riscv/kernel'
riscv64-unknown-elf-gcc  -O3 -march=rv64imafd -mabi=lp64 -mcmodel=medany -fno-builtin -ffunction-sections -fdata-sectio
ns -nostartfiles -nostdlib -nostdinc -static -lgcc -Wl,--nmagic -Wl,--gc-sections  -I /have-fun-debugging/os21fall/src/
lab1/include -I /have-fun-debugging/os21fall/src/lab1/arch/riscv/include -c sbi.c
sbi.c: In function 'sbi_ecall':
sbi.c:23:27: error: expected ']' before '.' token
   23 |                      : [var.error] "=r" (var.error), [var.value] "=r" (var.value)
      |                            ^
      |                            ]
sbi.c:23:57: error: expected ']' before '.' token
   23 |                      : [var.error] "=r" (var.error), [var.value] "=r" (var.value)
      |                                                          ^
      |                                                          ]
sbi.c:25:21: error: duplicate 'asm' operand name 'var'
   25 |                      : );
      |                      ^
sbi.c:25:21: error: undefined named operand 'var.error'
sbi.c:25:21: error: undefined named operand 'var.value'
make[2]: *** [Makefile:12: sbi.o] Error 1
make[2]: Leaving directory '/have-fun-debugging/os21fall/src/lab1/arch/riscv/kernel'
make[1]: *** [Makefile:2: all] Error 2
make[1]: Leaving directory '/have-fun-debugging/os21fall/src/lab1/arch/riscv'
make: *** [Makefile:18: all] Error 2
root@d17089305c0a:/have-fun-debugging/os21fall/src/lab1# 
```

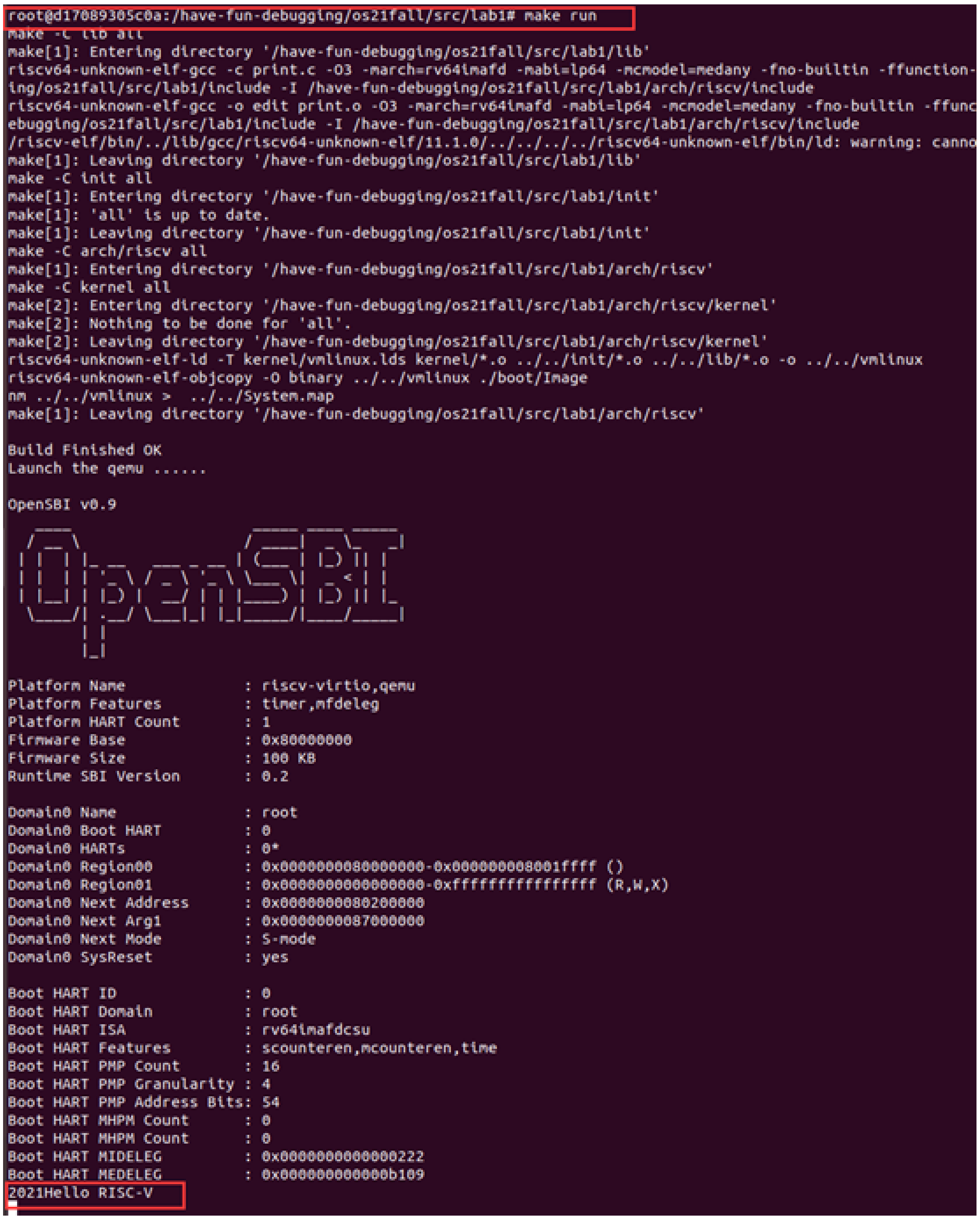
Ubuntu 64-bit - VMware Workstation 16 Player (仅用于非商业用途)
Player(P)
活动
终端
星期四 20：08
zh
root@d17089305c0a: /have-fun-debugging/os21fall/src/lab1
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

4.5 `puts()` 和 `puti()`

`sbi_ecall` 函数中，第三个传入 `arg0` 的参数就是待打印字符的ascii码。

```
#include "print.h"
#include "sbi.h"

void puts(char *s){
    int i=0;
    while(s[i++]!='\0')
    {
        sbi_ecall(0x1, 0x0, s[i], 0, 0, 0, 0, 0);
    }
}

void puti(int x)
{
    char s[100];
    int i = 0;
    if (x < 0) {
        sbi_ecall(0x1, 0x0, '-', 0, 0, 0, 0, 0);
        x = 0 - x;
    }
    for (; x/10 != 0; i++) {
        s[i] = x%10 + '0';
        x /= 10;
    }
    s[i] = x + '0';
    for (; i >= 0; i--) {
        sbi_ecall(0x1, 0x0, s[i], 0, 0, 0, 0, 0);
    }
}
```

成功编译运行。



```
root@d17089305c0a:/have-fun-debugging/os21fall/src/lab1# make run
make -C lib all
make[1]: Entering directory '/have-fun-debugging/os21fall/src/lab1/lib'
riscv64-unknown-elf-gcc -c print.c -O3 -march=rv64imafd -mabi=lp64 -mcmodel=medany -fno-builtin -ffunction-
ing/os21fall/src/lab1/include -I /have-fun-debugging/os21fall/src/lab1/arch/riscv/include
riscv64-unknown-elf-gcc -o edit print.o -O3 -march=rv64imafd -mabi=lp64 -mcmodel=medany -fno-builtin -ffunc
ebugging/os21fall/src/lab1/include -I /have-fun-debugging/os21fall/src/lab1/arch/riscv/include
/riscv-elf/bin/../lib/gcc/riscv64-unknown-elf/11.1.0/../../../../riscv64-unknown-elf/bin/ld: warning: canno
make[1]: Leaving directory '/have-fun-debugging/os21fall/src/lab1/lib'
make -C init all
make[1]: Entering directory '/have-fun-debugging/os21fall/src/lab1/init'
make[1]: 'all' is up to date.
make[1]: Leaving directory '/have-fun-debugging/os21fall/src/lab1/init'
make -C arch/riscv all
make[1]: Entering directory '/have-fun-debugging/os21fall/src/lab1/arch/riscv'
make -C kernel all
make[2]: Entering directory '/have-fun-debugging/os21fall/src/lab1/arch/riscv/kernel'
make[2]: Nothing to be done for 'all'.
make[2]: Leaving directory '/have-fun-debugging/os21fall/src/lab1/arch/riscv/kernel'
riscv64-unknown-elf-ld -T kernel/vmlinux.lds kernel/*.o ../../init/*.o ../../lib/*.o -o ../../vmlinux
riscv64-unknown-elf-objcopy -O binary ../../vmlinux ./boot/Image
nm ../../vmlinux >  ../../System.map
make[1]: Leaving directory '/have-fun-debugging/os21fall/src/lab1/arch/riscv'

Build Finished OK
Launch the qemu ......

OpenSBI v0.9
   ____                    _____ ____ _____
  / __ \                  / ____|  _ \_   _|
 | |  | |_ __   ___ _ __ | (___ | |_) || |
 | |  | | '_ \ / _ \ '_ \ \___ \|  _ < | |
 | |__| | |_) |  __/ | | |____) | |_) || |_
  \____/| .__/ \___|_| |_|_____/|____/_____|
        | |
        |_|

Platform Name             : riscv-virtio,qemu
Platform Features         : timer,mfdeleg
Platform HART Count       : 1
Firmware Base             : 0x80000000
Firmware Size             : 100 KB
Runtime SBI Version       : 0.2

Domain0 Name              : root
Domain0 Boot HART         : 0
Domain0 HARTs             : 0*
Domain0 Region00          : 0x0000000080000000-0x000000008001ffff ()
Domain0 Region01          : 0x0000000000000000-0xffffffffffffffff (R,W,X)
Domain0 Next Address      : 0x0000000080200000
Domain0 Next Arg1         : 0x0000000087000000
Domain0 Next Mode         : S-mode
Domain0 SysReset          : yes

Boot HART ID              : 0
Boot HART Domain          : root
Boot HART ISA             : rv64imafdcsu
Boot HART Features        : scounteren,mcounteren,time
Boot HART PMP Count       : 16
Boot HART PMP Granularity : 4
Boot HART PMP Address Bits: 54
Boot HART MHPM Count      : 0
Boot HART MHPM Count      : 0
Boot HART MIDELEG         : 0x0000000000000222
Boot HART MEDELEG         : 0x000000000000b109
2021Hello RISC-V
```

4.6 修改`defs`

```
#ifdef _DEFS_H_
#define _DEFS_H_

#include "types.h"

#define csr_read(csr)
({
    register uint64 __v;    \
    asm volatile ("csrr" " %0, " #csr   \
                  : "=r" (__v)          \
                  : : "memory");        \
    __v;
})

#define csr_write(scr, val)
({
    uint64 __v = (uint64)(val);
    asm volatile ("csrw " #csr ", %0")  \
                    : : "r" (__v)       \
                    : "memory");        \
})

#endif
```

4.7 思考题

请总结一下 RISC-V 的 calling convention，并解释 Caller / Callee Saved Register 有什么区别?

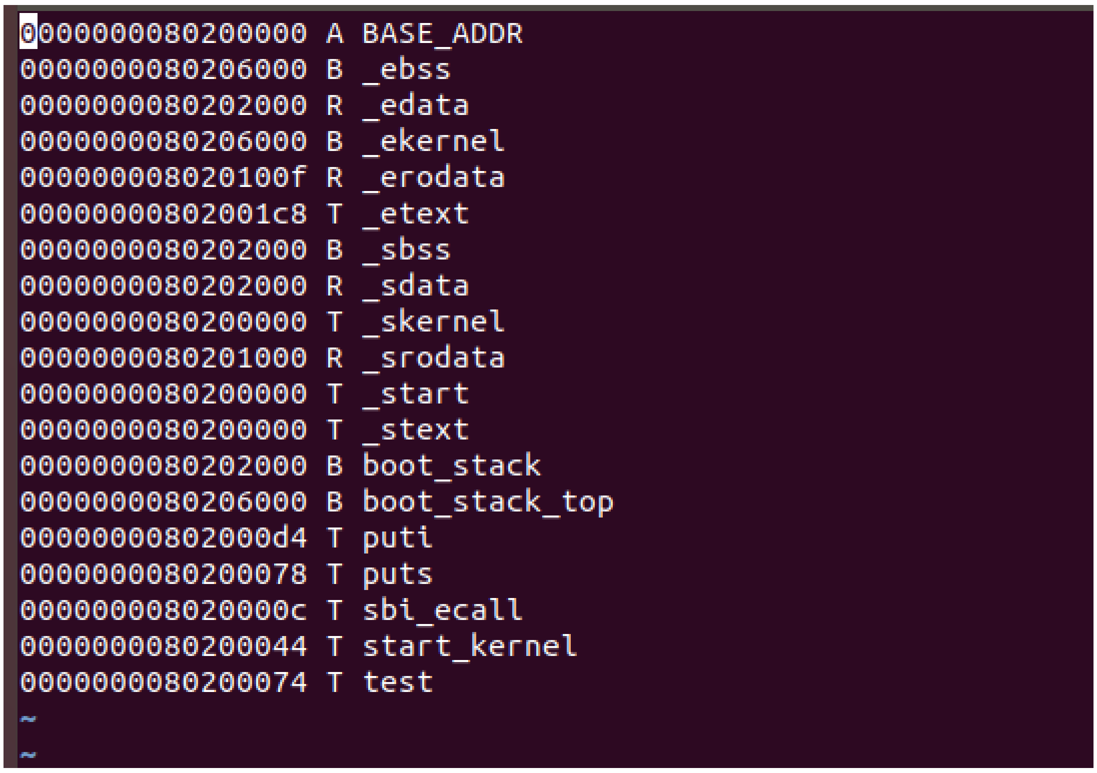
calling convention:

- 把函数参数放到函数能访问的地方
- 拿到 memory 中的资源， 获取函数需要的局部存储资源，按需保存寄存器
- 运行函数中的指令
- 把值写到 memory/register 中， 将返回值存储到调用者能够访问到的位置，恢复寄存器，释放局部存储资源

caller / callee-saved register 的区别

- caller-saved: 发生调用时可以从这些寄存器里读数据并操作
- callee-saved: 存储在调用返回前需要保存的数值，等调用结束后再重新读入

编译之后，通过 System.map 查看 vmlinux.lds 中自定义符号的值



```
0000000080200000 A BASE_ADDR
0000000080206000 B _ebss
0000000080202000 R _edata
0000000080206000 B _ekernel
000000008020100f R _erodata
00000000802001c8 T _etext
0000000080202000 B _sbss
0000000080202000 R _sdata
0000000080200000 T _skernel
0000000080201000 R _srodata
0000000080200000 T _start
0000000080200000 T _stext
0000000080202000 B boot_stack
0000000080206000 B boot_stack_top
00000000802000d4 T puti
0000000080200078 T puts
000000008020000c T sbi_ecall
0000000080200044 T start_kernel
0000000080200074 T test
~
~
```