






OS-review

 Assign	
 Property	
 tag	homework
 姓名	周鹏宇
 学号	2019K8009929039

Review

1. 关于函数调用

大致就是给的资源做了一个简单的封装，方便我们环境调用，例子如下

1.

```
ifndef _SBI_ASM_H
#define _SBI_ASM_H

.macro SBI_CALL which
    li a7, \which
    ecall
    nop
.endm

#endif /* _SBI_ASM_H */
```

```
#ifndef _ASM_SBI_DEF_H
#define _ASM_SBI_DEF_H

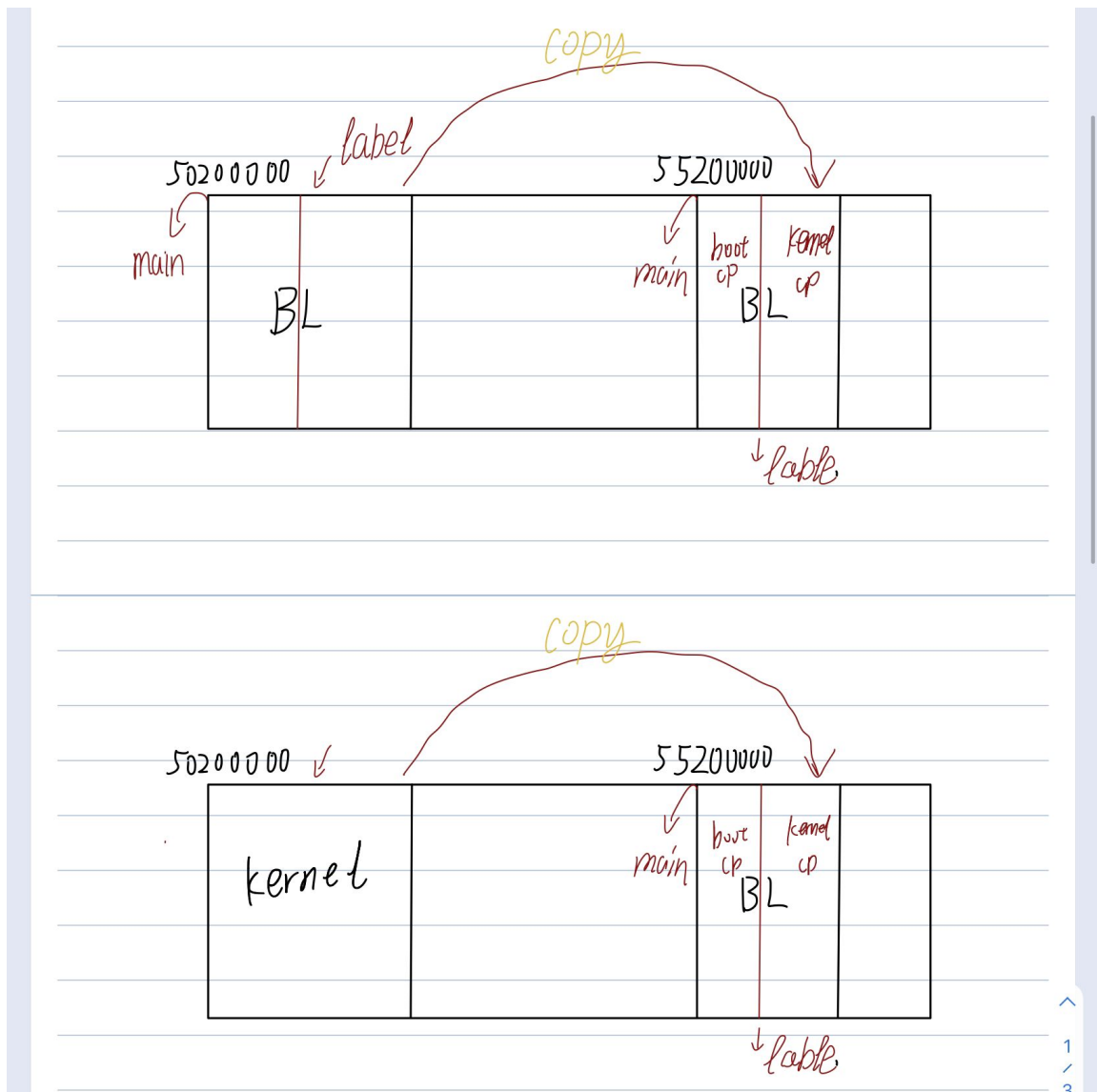
#define SBI_SET_TIMER 0
#define SBI_CONSOLE_PUTCHAR 1
#define SBI_CONSOLE_GETCHAR 2
#define SBI_CLEAR_IPI 3
#define SBI_SEND_IPI 4
#define SBI_REMOTE_FENCE_I 5
#define SBI_REMOTE_SFENCE_VMA 6
#define SBI_REMOTE_SFENCE_VMA_ASID 7
#define SBI_SHUTDOWN 8
#define SBI_CONSOLE_PUTSTR 9
#define SBI_SD_WRITE 10
#define SBI_SD_READ 11
```

```
#endif // _ASM_SBI_DEF_H
```

```
la a0, msg  
SBI_CALL SBI_CONSOLE_PUTSTR
```

```
la a0, msg  
li a7, 1  
ecall  
nop
```

2. 关于bootblock在内存中的排布（涵盖课件和微信中两个关于bootloader的问题）
鉴于对a-core的考虑，排布大致如下



3. 关于组合kernel和bootloader

完全参照createimage函数的框架即可

- 读elf文件
- 读程序头
- 通过write_segment把phdr指示的段放置到正确位置
- 记录大小
- 如果要做双系统还要注意统计不同核的大小和在image中的位置
- 还需要开辟一个可用空间，方便bootloader加载

4. 双系统：

a. bootloader

需要开辟一个分支，并且处理一下输入（getchar）

b. createimage

已经在3.中说了

关于调试

bootload部分

- 对取字不熟，在获取kernel_size时没有启用取半字(0(t0))
- 设计输入的跳转时忘记输入的‘1’和‘0’是按照ascii码进入系统的，应以49和48与之比较
- 对la指令的理解不够，经常混用成li

head部分

- 由于是对BSS段置0，其在内存中，故应当使用s指令

kernel.c部分

- 做大核测试时可声明一个较大的全局变量

createimage.c部分

- 在支持双核时，处理多个输入的kernel文件时，忘记file++，导致反复编译同一个文件
- 其实可以通过将define部分的常量改为变量可以实现函数复用和多核装载