**操作系统Project1——bootloader**

胡力杭 2019k8009926002

1、对较大值的常量的载入。

讲义上提出要使用下面的方法进行载入，但是直接用la指令好像也可以。

lui a0, %hi(const\_value) # 载入常量的高位

addi a0, a0, %lo(const\_value) # 载入常量的低位

使用反汇编之后看出如果用la指令，编译器其实会帮助我们完成这两个指令的转换。

2、gp寄存器的初始化

因为这个操作系统是我们自己书写的，所以开始的时候gp寄存器并没有初始化的操作，如果在bootblock.S文件的反汇编中使用了gp寄存器，就需要我们初始化gp寄存器。

/\* Load the global pointer \*/

    .option push

    .option norelax

    la gp, \_\_global\_pointer$

    .option pop

3、bootloader拷贝完成之后的跳转

完成Acore的时候，我们希望将bootloader放在较高位。这时我们在拷贝完成之后需要跳转到相对应的地址。但是如果是从头开始执行，就会反复拷贝出现错误，所以我们需要跳转到相对应的地址。我们使用了一个新的标签，通过地址相减得到我们需要的偏移量。

la t1, bootloader

    la t2, Location

    la t3, kernel

    sub t2, t2, t3

    add t1, t1, t2

    jr t1

Location:

这里尝试过使用反汇编指令读出Location的地址，但是每次改写bootblock.S的时候都需要重新使用反汇编读出地址，比较麻烦。

4、BIOS中的函数的参数传递

为了实验方便，老师已经定义了sbi.h头文件，可以系统调用其中的一些函数。调用函数的时候我们使用了a0，a1，a2寄存器来传递参数，并且这个顺序和函数定义时的顺序是一样的。

5、有关跳转的j指令

在bootblock.S文件中，我们在拷贝bootloader之后需要进行一个跳转。如果这里直接使用j指令，会导致跳转的地址产生问题。可以使用寄存器进行跳转，这样不会产生跳转地址的偏差。

6、书写习惯的问题

在写代码的时候，我定义参数会用一些无意义的名称，比如a，b，c，这样在debug的时候还需要重新理解自己的代码，浪费了一点时间。