huihongxiao GitBook: [master... Latest commit 7bbde8e on Oct 31, 2020 History

1 contributor

26 lines (14 sloc) 2.88 KB Raw Blame

4.7 kvminithart 函数

之后，kvminit函数返回了，在main函数中，我们运行到了kvminithart函数。

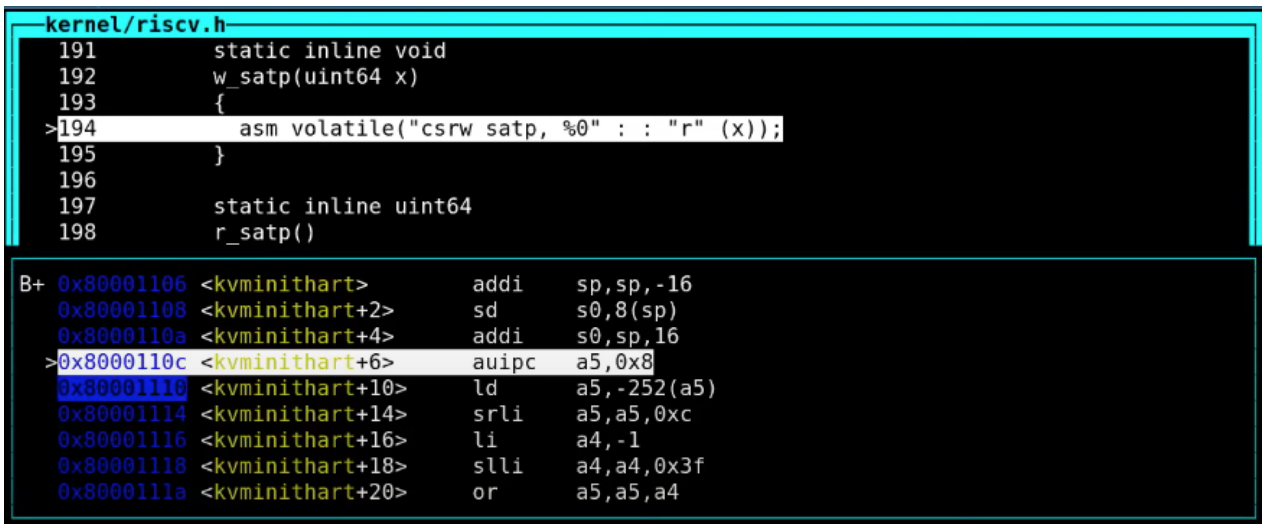
```
// Switch h/w page table register to the kernel's page table,
// and enable paging.
void
kvminithart()
{
  w_satp(MAKE_SATP(kernel_pagetable));
  sfence_vma();
}
```

这个函数首先设置了SATP寄存器，kernel_pagetable变量来自于kvminit第一行。所以这里实际上是内核告诉MMU来使用刚刚设置好的page table。当这里这条指令执行之后，下一个指令的地址会发生什么？

在这条指令之前，还不存在可用的page table，所以也就不存在地址翻译。执行完这条指令之后，程序计数器（Program Counter）增加了4。而之后的下一条指令被执行时，程序计数器会被内存中的page table翻译。

所以这条指令的执行时刻是一个非常重要的时刻。因为整个地址翻译从这条指令之后开始生效，之后的每一个使用的内存地址都可能对应到与之不同的物理内存地址。因为在这条指令之前，我们使用的都是物理内存地址，这条指令之后page table开始生效，所有的内存地址都变成了另一个含义，也就是虚拟内存地址。

这里能正常工作的原因是值得注意的。因为前一条指令还是在物理内存中，而后一条指令已经在虚拟内存中了。比如，下一条指令地址是0x80001110就是一个虚拟内存地址。



为什么这里能正常工作呢？因为kernel page的映射关系中，虚拟地址到物理地址是完全相等的。所以，在我们打开虚拟地址翻译硬件之后，地址翻译硬件会将一个虚拟地址翻译到相同的物理地址。所以实际上，我们最终还是能通过内存地址执行到正确的指令，因为经过地址翻译0x80001110还是对应0x80001110。

管理虚拟内存的一个难点是，一旦执行了类似于SATP这样的指令，你相当于将一个page table加载到了SATP寄存器，你的世界完全改变了。现在每一个地址都会被你设置好的page table所翻译。那么假设你的page table设置错误了，会发生什么呢？有人想回答这个问题吗？

> 学生A回答：你可能会覆盖kernel data。
>
> 学生B回答：会产生page fault。

是的，因为page table没有设置好，虚拟地址可能根本就翻译不了，那么内核会停止运行并panic。所以，如果page table中有bug，你将会看到奇怪的错误和崩溃，这导致了page table实验将会比较难。如果你不够小心，或者你没有完全理解一些细节，你可能会导致kernel崩溃，这将会花费一些时间和精力来追踪背后的原因。但这就是管理虚拟内存的一部分，因为对于一个这么强大的工具，如果出错了，相应的你也会得到严重的后果。我并不是要给你们泼凉水，哈哈。另一方面，这也很有乐趣，经过了page table实验，你们会真正理解虚拟内存是什么，虚拟内存能做什么。