关于提问

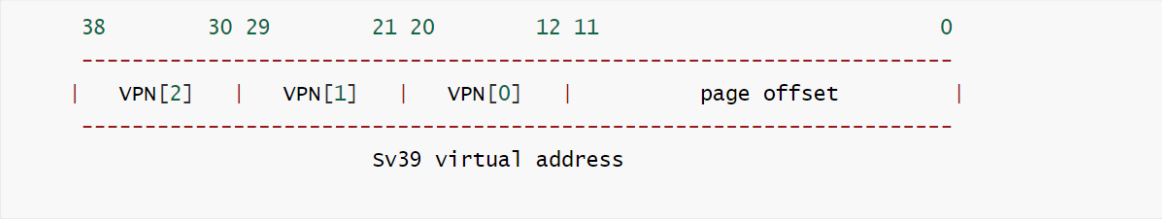
希望大家提问的时候能描述的具体点，直接放一张运行结果图让我们找问题的话我们也看不出个什么。最起码应该给出你相关的代码和你的运行结果，最好可以附一下你觉得错在哪里，你尝试的解决办法。这样沟通起来效率会高一点。

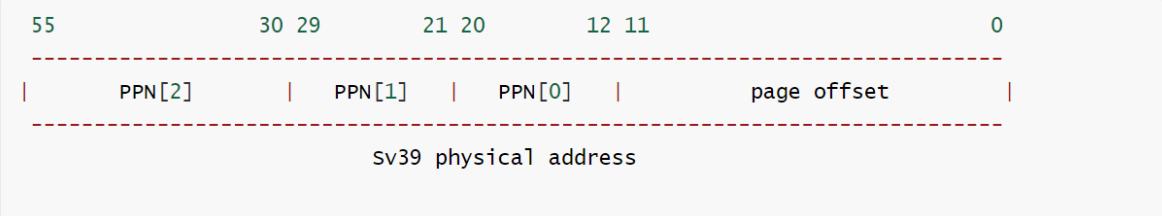
create_mapping代码编写思路

先确定自己逻辑对不对，我这边给个供参考的思路，不会写的同学也可以参考下

实验文档原文：为了增加一个虚拟地址到物理地址的映射，根据Rv39分配方案中的地址翻译规则，首先需要根据根页表基地址以及虚拟地址找到相应页表项的地址，并在该过程中为页表分配物理页面，随后根据物理地址及映射的权限设置页表项具体的值，使得后续MMU工作时能够根据根页表基地址和虚拟地址得到正确的物理地址。

图示：





思路：

```
void create_mapping(uint64_t *pgtbl, uint64_t va, uint64_t pa, uint64_t sz, int
perm){
    //pgtbl(&_end)是三级页表的首地址，要为他保留一页的大小，所以后续空闲物理页面需要从
&_end+0x1000*1开始分配。

    //每次设置一个一级页表项管理4KB大小的映射，将4KB大小的va映射到4KB大小的pa，并在该过程中设
置好二级和三级页表。
    while(sz>0){
        //我们用V位标志该页表项是否有效,一开始时V位为0，当我们设置好页表项后V位为1。
         //MMU过程中先根据vpn[2]和三级页表首地址得到相应的三级页表项，三级页表项中存放的值是
二级页表的首地址。为此，我们需要给二级页表分配物理空间得到其地址，找到三级页表项的位置并将其值设
置为二级页表首地址，随后将V位标志为1，标记这个页表项生效。
        //V位为0，需要设置三级页表项
        if((pgtbl[vpn[2]]&0x1)==0)
        {
            //给二级页表分配空间，pgtbl_2即为二级页表的首地址
            pgtbl_2=(uint64_t *);
            //将二级页表首地址填充到三级页表项的值，并设置好该页表项的权限
            pgtbl[vpn[2]]=(uint64_t);
        }
        //V位为1，可以根据三级页表项的值得到二级页表首地址
```

```
        else{
            pgtbl_2=(uint64_t *)
        }
        //对于二级页表同上
        if((pgtbl_2[vpn[1]]&0x1)==0){}
        else{}
        //找到一级页表项后直接设置即可，其中填充的内容是物理地址
        pgtbl_1[vpn[0]]=(uint64_t);
        //更新映射大小、下一个需要创建映射关系的的pa和va
         sz-=0x1000;
         va+=0x1000;
         pa+=0x1000;
    }
```

注意：

- 注意运算符优先级，<<和>>是低于+/-的，&是高于==的，用括号保证运算的正确顺序，否则容易出现你的结果并不符合你的预期，可以打印个例子出来看一下正不正确。

5	加减法运算符：+　-	从左到右
6	移位运算符：<<　>>	从左到右
7	关系运算符：<<=　>>=	从左到右
8	相等运算符：==　!=	从左到右
9	位运算符 AND：&	从左到右

- perm不包含V位，设置时是（perm<<1）+1，不能写perm<<1+1原因同上。
- 将分配的物理地址填写到页表项中时，需要先>>12再<<10，因为我们是一个页表项负责映射4KB的大小，只需要前44位作为PPN，后12位作为页内的offset不需要考虑。
- 注意什么时候要取值，什么时候要取地址，可以用(uint64_t)和(uint64_t*)放到前面作为标志。
- page_count需要全局更新，不能在这个函数里定义，不然会导致覆盖，且page_count的初始值应该是1，因为第一个页是给三级页表准备的。

查bug思路

首先大部分同学来问问题都是这个只有warning的界面，基本就是映射没成功，映射成功后起码会输出这行“ZJU OS LAB4...”。

```
qemu-system-riscv64: warning: No -bios option specified. Not loading a firmware.
qemu-system-riscv64: warning: This default will change in a future QEMU release. Please u
se the -bios option to avoid breakages when this happens.
qemu-system-riscv64: warning: See QEMU's deprecation documentation for details.
```

- 一般来说是sfence.vma刷新，MMU机制生效后因为映射不对导致页错误。可以尝试下在sfence.vma后面加上“li a0,10""call　puti"
- 也有可能paging_init就没有完成，在create_mapping()函数途中访问越界之类。在call paging_init后面加个“li a0,11""call　puti"
- 运行看看，没有输出就是你错误了

然后看映射为什么出错：

- 先确定下自己在paging_init中的地址写对没有，0数对了吗，第三步UART映射那边映射大小设为0x1000即可。
- 去head.S把第一个时间中断的时间间隔调高一点，调成10000000000，这样可以在打印的时候不被时钟中断切掉，确定自己映射写对之后（打印出"ZJU OS LAB4..."）后再改回来。

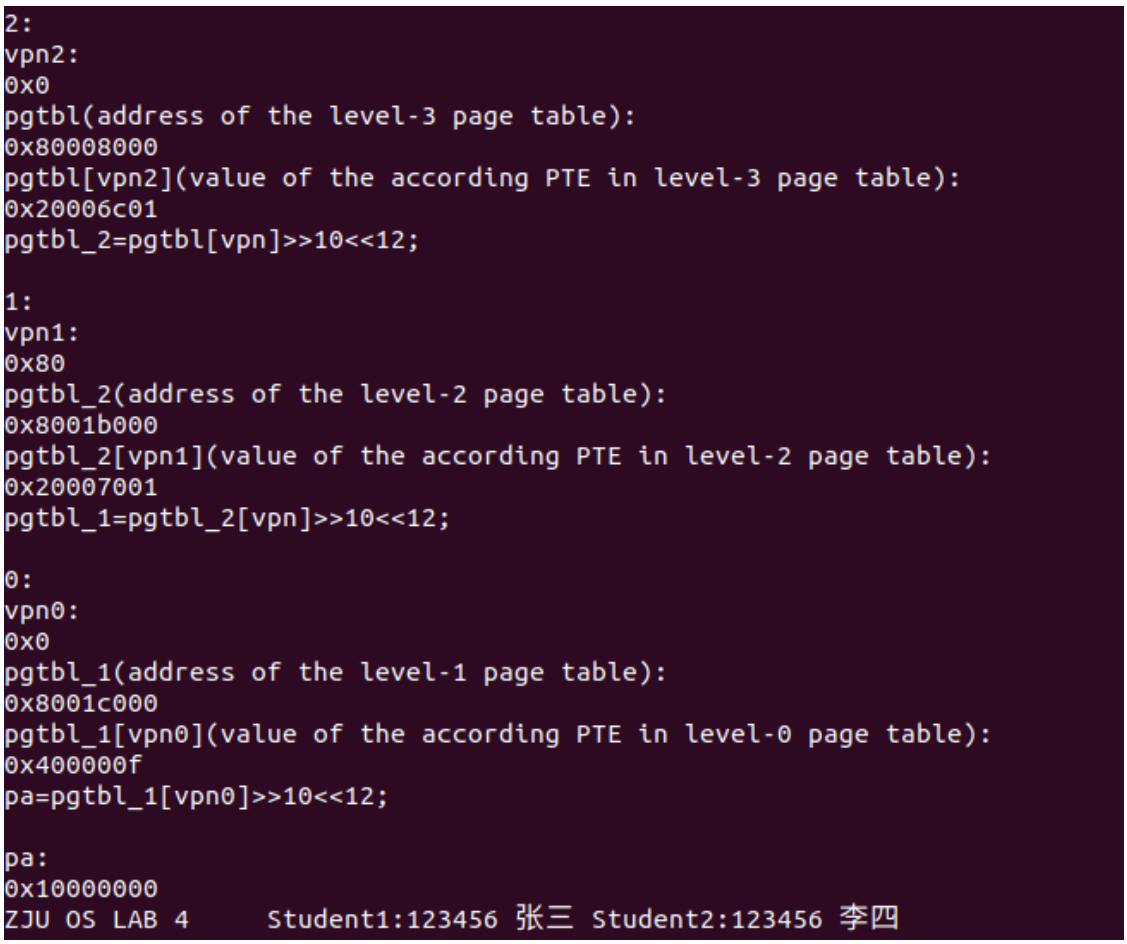
```
clean_loop:
        sb t3, 0(t1)              #往t1的内存中写入t3的内容（每次1个字节）
        addi t1, t1, 1            #t1的值往后1个字节
        bne t1, t2, clean_loop    #比较，如果t1和t2的值不等，则继续往内存中存t3

        #初始化mtimecmp寄存器
        li t1, 0x200bff8          #t1=mtime的地址
        li t2, 0x2004000          #t2=mtimecmp的地址
        ld t4, 0(t1)              #读mtime的8字节的内容到t4中
        li t3, 10000000000
        add t4, t4, t3            #t4加上1000000
        sd t4, 0(t2)              #将a0的值存入地址为t2的内存中（8字节）
```

- 在每一个create_mapping()后面加上puts("finished")　看能否成功输出
    - 如果不能，那就是在create_mapping()中途可能访问越界，自己在里面输出一下中途的信息，看一下上面说的注意点
    - 如果能成功输出，那映射起码是做出来了，模仿MMU机制的过程确认一下自己的映射对不对，如果不对再进函数看下。有一个比较简陋的检查方法：如果按照上面那个写法，输出一下每次create_mapping后的page_count数，应该是10-19-21（三次映射大小为0x1000000\0x1000000\0x1000时），如果你的page count特别大的话应该就有问题。
    - 在paging_init()最后加上这段模仿MMU机制的内容作为测试（这里是简化过的逻辑没有判断有效位是否为1），我这边也提供一下按上面那个思路写出来的代码的运行结果，你也可以根据这段内容反推一下你需要在create_mapping里做什么。

```
uint64_t testva=0x10000230;
testva=testva&0x7fffffffff;
testva=testva>>12;
uint64_t vpn2,vpn1,vpn0;
vpn0=testva&0x1ff;
vpn1=(testva>>9)&0x1ff;
vpn2=(testva>>18)&0x1ff;
uint64_t* p =pgtbl;
puts("\n2:\nvpn2:\n");
putullHex((uint64_t)vpn2);
puts("\npgtbl(address of the level-3 page table):\n");
putullHex((uint64_t)p);
puts("\npgtbl[vpn2](value of the according PTE in level-3 page
table):\n");
putullHex((uint64_t)p[vpn2]);
puts("\npgtbl_2=pgtbl[vpn]>>10<<12;\n");
p=(p[vpn2]>>10)<<12;
puts("\n1:\nvpn1:\n");
putullHex((uint64_t)vpn1);
puts("\npgtbl_2(address of the level-2 page table):\n");
putullHex((uint64_t)p);
puts("\npgtbl_2[vpn1](value of the according PTE in level-2 page
table):\n");
putullHex((uint64_t)p[vpn1]);
```

```
puts("\npgtbl_1=pgtbl_2[vpn]>>10<<12;\n");
p=(p[vpn1]>>10)<<12;
puts("\n0:\nvpn0:\n");
putullHex((uint64_t)vpn0);
puts("\npgtbl_1(address of the level-1 page table):\n");
putullHex((uint64_t)p);
puts("\npgtbl_1[vpn0](value of the according PTE in level-0 page
table):\n");
putullHex((uint64_t)p[vpn0]);
puts("\npa=pgtbl_1[vpn0]>>10<<12;\n");
p=(p[vpn0]>>10)<<12;
puts("\npa:\n");
putullHex((uint64_t)p);
puts("\n");
```

- 
```
2:
vpn2:
0x0
pgtbl(address of the level-3 page table):
0x80008000
pgtbl[vpn2](value of the according PTE in level-3 page table):
0x20006c01
pgtbl_2=pgtbl[vpn]>>10<<12;

1:
vpn1:
0x80
pgtbl_2(address of the level-2 page table):
0x8001b000
pgtbl_2[vpn1](value of the according PTE in level-2 page table):
0x20007001
pgtbl_1=pgtbl_2[vpn]>>10<<12;

0:
vpn0:
0x0
pgtbl_1(address of the level-1 page table):
0x8001c000
pgtbl_1[vpn0](value of the according PTE in level-0 page table):
0x400000f
pa=pgtbl_1[vpn0]>>10<<12;

pa:
0x10000000
ZJU OS LAB 4     Student1:123456 张三 Student2:123456 李四
```

- 这个对的话，权限看下设置对没有，权限也对的话应该没问题了，还不行的话直接来找我。