Lab. page tables

① MMU 什么时候使用？有什么作用？

代码层面直接访问某地址时： e.g. *src = 10;

当执行用户态程序时，satp 指向每个用户进程自己的虚拟内存页表，MMU 对用户代码中对某地址的直接访问用页表查 user va 对应 RAM 地址访问

当 xv6 处于内核态（内核线程），satp 切换到内核页表（全局唯一）上，内核代码访问某地址时 MMU 也会基于内核页表进行翻译，由于 xv6 内核页表与 RAM 一一对应，因此 MMU 的翻译实际上是直接对应访问

MMU 取 src 中地址值
基于当前 satp 指向页表
进行翻译成真正 RAM 地址
读写数据

② 什么时候使用软件模拟的 MMU？

A: e.g.

copyin　copyout.

内核要使用用户虚拟地址时.

因为在内核态时，用户给的 vm addr 在内核页表中是不存在映射的，将用户 vm addr 直接给 mmu 翻译当然出错，需使用软件将其翻译成对应的内核页表地址，由于内核页表与实际 RAM 硬件地址一一对应，直接映射，此时将被翻译过来的内核页表地址再用 MMU 翻译（直接映射）成 RAM 物理地址.

③ copyin 与 copyin_new.

1. 内核与用户进程各自使用各自的页表，每个用户进程都有自己独立的虚拟内存空间与页表。CPU在用户态执行时，satp中指向页表基地址为该用户进程页表的存放地址，对于用户访问的自己的虚拟地址MMU访问satp中用户页表地址找到该页表对用户VA进行翻译。进行读写内核态用 satp 内核页表地址。内核代码访问内核中虚拟内存时（由于内核 ~~Vm与RAM一一对应，因此内核页表将某一对应关系存储~~），MMU访问satp获取内核页表，查询翻译再实际读写RAM

内核页表中"内核虚拟地址空间与RAM"和"用户进程虚拟地址空间与内核虚拟地址空间（⇔RAM）"对应关系不一样，用户VA在内核页表中无效，因此内核想访问用户VA必须在内核中以用户进程页表和用户VA为参数使用软件模拟MMU进行访问用户数据

2. copyin与copyin_new都是内核代码，处于内核态下执行。因此执行这两函数时satp Reg中存的页表地址在修改代码前都是全局唯一的内核页表。

3. 原来的copyin是以用户页表地址、用户虚拟地址为参数，软件模拟MMU得到 user VA 对应的 内核虚拟内存地址，再将该地址交给MMU翻译成RAM地址进行data读写

4. 本Lab想使用另一种 copy to kernel 的方法命名使用 copyin. (copyin_new)

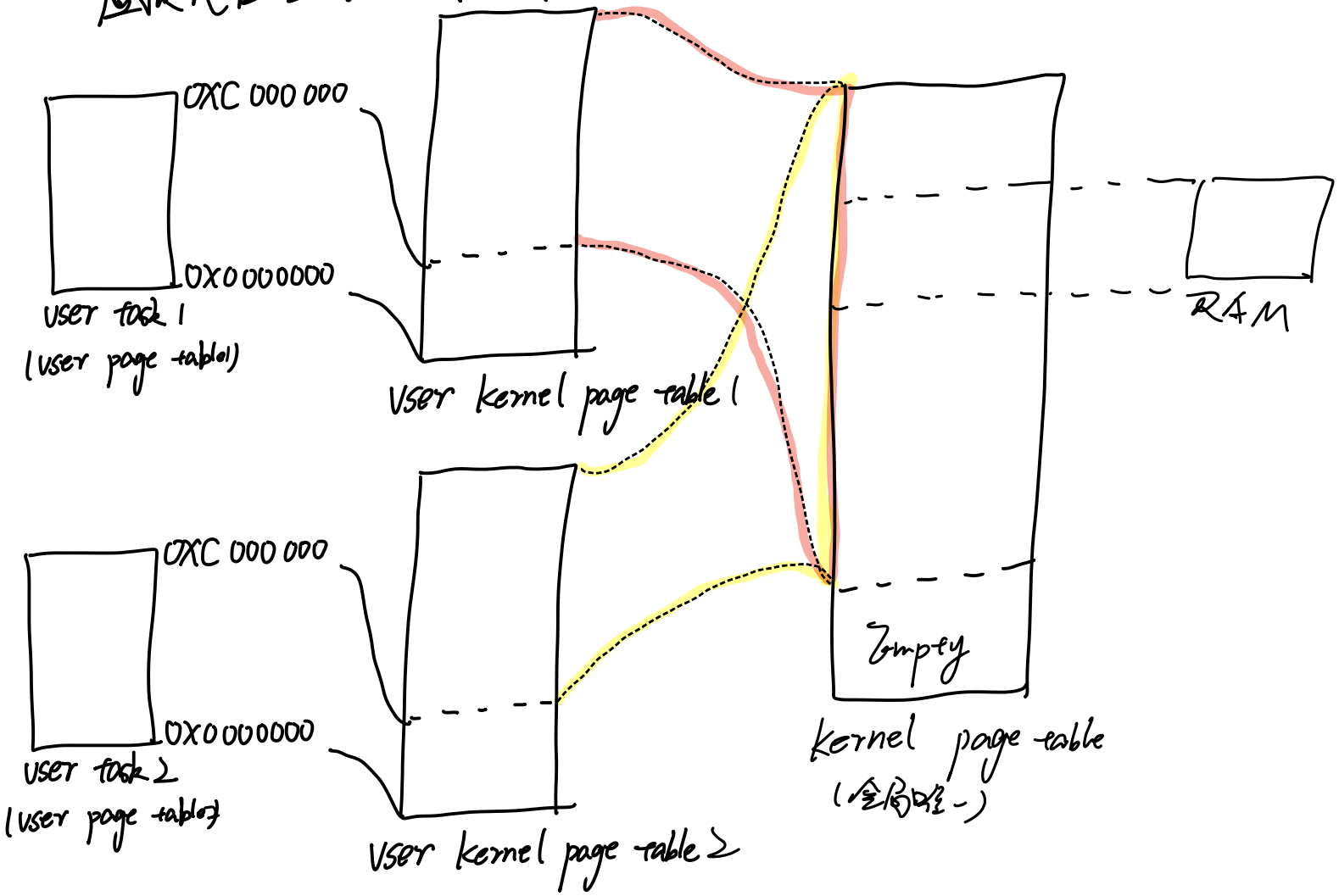
5. copyin_new 是直接基于当前satp指向页表，传入的VA直接进行访问，MMU将该VA查上述satp指向页表，对RAM进行访问。如果什么都不改动，copyin_new直接访问用户态VA是会出错的。因为内核页表上没有该用户态VA与内核虚拟内存的正确对应关系

6. Lab基于(OS)提出了解决方法并让我们实现：

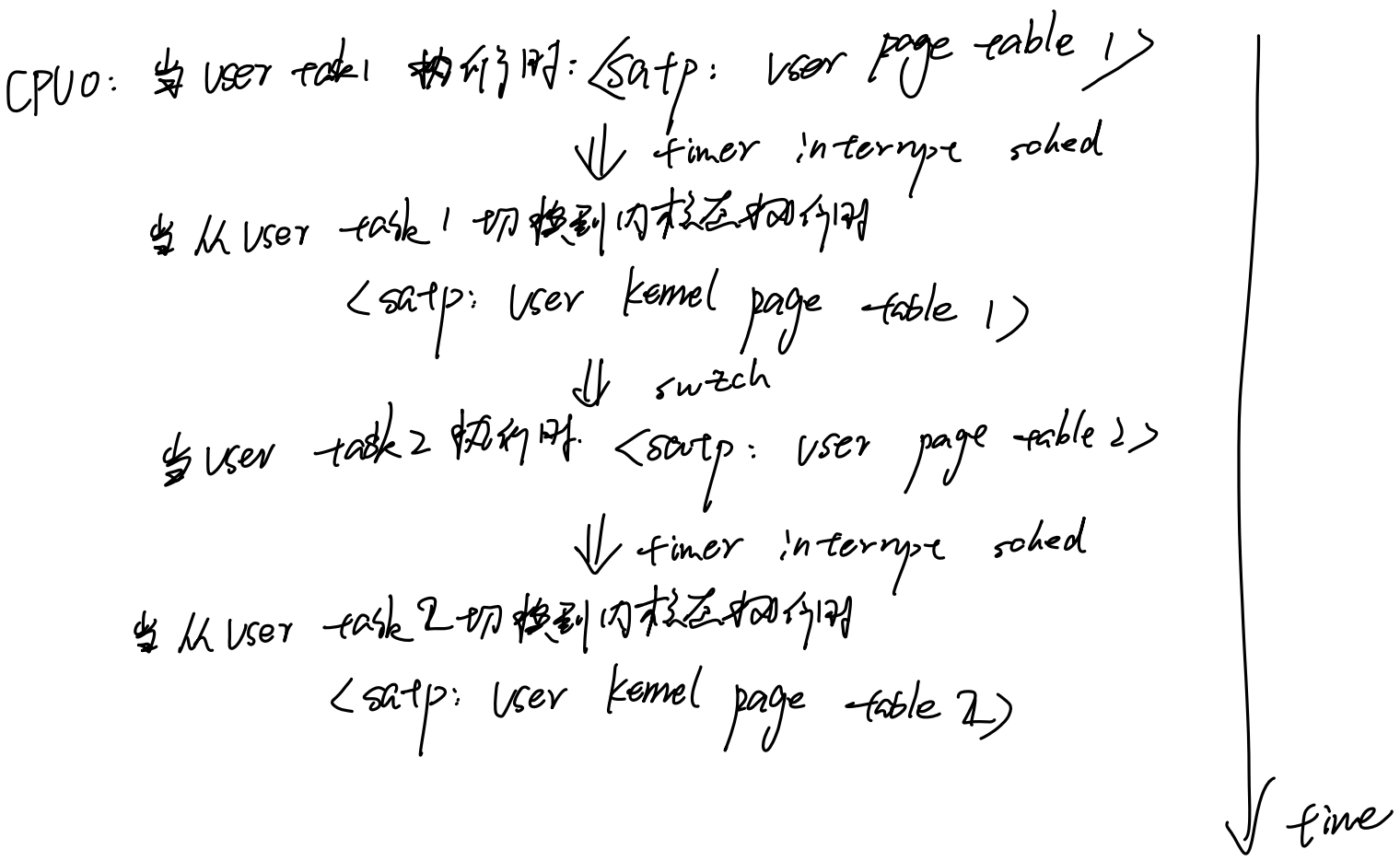
⇒ 规定此Lab中用户虚拟内存使用大小小于 0xC000000 bytes.

由于内核虚拟地址空间自0xC000000以下是没有被内核自身所使用的. 因此我们让(每个用户进程)有一个"用户内核页表", 当从该用户进程切换到内核态时. SOS从用户虚拟地址空间映射关系的用户页表地址切换到前面新增的"用户内核页表".

这个用户内核页表自 0xC000000 以上都与唯一的内核页表保持一致. 以下的内容因用户程序不同而映射到用户虚存页表映射的内核虚拟地址空间页表页表项.



.
.
.


CPU0: 当 user task1 被切换时: <satp: user page table 1>
⇓ timer interrupt sched
当从 user task1 切换到内核态被切换时
<satp: user kernel page table 1>
⇓ swtch
当 user task2 被切换时. <satp: user page table 2>
⇓ timer interrupt sched
当从 user task2 切换到内核态被切换时
<satp: user kernel page table 2>
time