实现

1. 重写sys_get_time和sys_task_info：这部分主要的内容就是将传进来的指针由虚地址转换成物理地址。可以直接用translated_byte_buffer转换，但是我一开始并没有意识到，而是将其中用作地址转换的部分提出来放到了两个syscall实现中，然后用unsafe的指针访问把值存进去。同时我也考虑了sys_write的时候可能写到不合法的内存的情况，但是测试的时候似乎没有这个考点。
2. 实现map：每次我们拿到map的区间，先处理port拿到permission，然后对于拿到的区间，去当前进程的memory_set中对于拿到的每个虚拟页，去所有的maparea中查data_frames是否已经有了映射。如果有了，说明之前已映射过，map失败。如果均没有，则我们先查找所有maparea中vpn_range里和当前区间有交集的部分，为他们添加映射（用map_one），然后对于剩下的部分新创建一段maparea插入到memory_set中。
3. 实现unmap：每次我们拿到unmap的区间时，我们先去地址空间中查找所有的maparea是否没有其中页的映射，如果发现有的页没有映射，则unmap失败，否则我们在和当前区间有交集的那些maparea中的data_frames里删除映射（调用unmap_one），然后如果存在maparea的data_frames已经空了，则直接删除这个map_area。

问答题

1.请列举 SV39 页表页表项的组成，描述其中的标志位有何作用?

63 — 48	47 — 28	27 — 19	18 — 10	9 — 7	6	5	4 — 1	0
Reserved	PPN[2]	PPN[1]	PPN[0]	*Reserved for SW*	D	R	Type	V
16	20	9	9	3	1	1	4	1

sv39页表项的结构如上。其中各个部分的作用如下：

V：页表项有效位。

R/W/X: 可读、可写、可执行位。

U：用户态是否可访问。

G：全局地址代换。

A：是否被访问过。

D：是否被修改。

Reserved for SW：保留位。

三个PPN：3段ppn，实现三级页表的查找。

2.缺页

1. 请问哪些异常可能是缺页导致的?

Priority	Exception Code	Description
Highest	3	Instruction address breakpoint
	12	Instruction page fault
	1	Instruction access fault
	2	Illegal instruction
	0	Instruction address misaligned
	8, 9, 11	Environment call
	3	Environment break
	3	Load/Store/AMO address breakpoint
	6	Store/AMO address misaligned
	4	Load address misaligned
	15	Store/AMO page fault
	13	Load page fault
	7	Store/AMO access fault
Lowest	5	Load access fault

如上图，来自riscv手册。可以看到有三种可能的异常：Store/AMO page fault、Load page fault、Instruction page fault.

2. 发生缺页时，描述相关重要寄存器的值，上次实验描述过的可以简略

   切换sp等寄存器，还保存了sstatus、sepc等寄存器，不多赘述；并且从trapContext加载了新的satp，切换到内核态，并且保存了旧的satp，供之后恢复使用。

3. 这样做有哪些好处?

   lazy策略一定程度上可以提升效率。类似copy on write的想法，在用到的时候才真正的加载，这样如果在整个运行过程中没有访问，则不必花费这部分开销；且根据设计，可以做到实际分配的内存大小等于实际用到的内存大小，这样如果申请了很大的内存而用了很小的一部分，则实际分配的只有用到小的这部分，不会浪费内存，提升了时间、空间上的效率。

4. 处理 10G 连续的内存页面，对应的 SV39 页表大致占用多少内存 (估算数量级即可)?

   由 https://rcore-os.github.io/rCore-Tutorial-Book-v3/chapter4/3sv39-implementation-1.html，SV39多级页表实际用到S字节时，页表大约小号S/512左右。估算过程如下：假设有S字节需要映射的内存，则我们映射这S字节大小的内存，需要S/(4096 * 512)个一级页表（每个页表项映射一个页，4096B，而每个二级页表可以映射512个一级页表），这部分一级页表需要S/(4096 * 512) * 4096 = S/512B的内存；对于二级页表，类似的，需要S/(4096 * 512 * 512)* 4096 = 4S/Ki B的内存，相比一级页表可以忽略，因此大约需要S/512 B的内存。带入计算，因此此时页表大致占用10GB / 512 = 20MB左右。

5. 请简单思考如何才能实现 Lazy 策略，缺页时又如何处理？描述合理即可，不需要考虑实现

   即我们mmap的调用的时候，只在地址空间中添加这部分虚地址的范围，但是不分配具体的frames，这样表示这部分地址已经在合法可访问地址空间中。而实际访问时，因为没有分配实际物理页帧，因此会触发异常。此时调用alloc等方法为用户实际分配物理页帧即可。

6. 此时页面失效如何表现在页表项(PTE)上?

   pte的Valid位被置0。

3.双页表与单页表

1. 在单页表情况下，如何更换页表?

   还是切换satp。只不过因为同一个进程的用户、内核页表放到了同一张表里，因此切换页表的时机不再是trap的时候，而在切换进程的时候（大概在switch的时候）。

2. 单页表情况下，如何控制用户态无法访问内核页面？（tips:看看上一题最后一问）

   将内核态地址空间对应的页表项U置0。

3. 单页表有何优势？（回答合理即可）

在内存占用区别不大的情况下（不同应用查satp时，内核态的三级页表物理页帧可以是一样的），在系统调用的时候不用切换页表，从而获得更高的缓存效率。

4. 双页表实现下，何时需要更换页表？假设你写一个单页表操作系统，你会选择何时更换页表（回答合理即可）？

   双页表时在trap.S，即从用户态调到内核态时切换；如果我写单页表，如前所述，会在switch的时候切换页表。