实验报告

211240045 杨镇源

实验进度：

PA4.1 完成了所有必做内容；

PA4.2 完成了所有必做内容；

PA4.3 完成了包括时钟中断之前的所有必做内容，完成了添加前台程序和程序切换功能的基本代码框架，但是内核栈和用户栈有关内容并未完成。

必做题：

必做题 1：分时多任务的具体过程。

通过使用 vme.c 定义的 map 函数，我们在操作系统中完成了程序虚拟地址对具体物理地址的映射。同时我们将该进程页目录基地址在 satp 寄存器中进行保存。

对于不同进程来说，进行 map 操作对应的物理地址不同，且页目录基地址也不相同，因此在不同程序中即使访问了相同的虚拟地址空间，也会通过 mmu_translate 函数，将其映射到不同的物理空间上，从而保证不同进程地址访问的正确性。这一切都是建立在分页机制上。

然后在分时多任务的背景下，硬件每 10ms 强制触发异常，abstract machine 的 cte 在保存当前上下文后，跳转至 nanos-lite 处理，其中的 schedule 函数用于决定下一时间需要调度的进程。然后再将相关进程的上下文返回，确保 hello 和仙剑奇侠传两个进程都能够正常运行。但是实际上 nemu 在一个特定的时间点只能在运行一个进程，由于进程之间切换的时间足够短，可以实现多任务的目标。

必做题 2：理解计算机系统。

首先，操作系统在装载该进程时，会将字符串常量储存在可执行文件的.rodata 段中。.rodata 段中的数据（"abc"）是只读的。若此时对该字符串所在存储区域进行写操作，硬件 mmu 会翻译出该段不可写，进而触发 mmu 异常。该异常被操作系统识别到后，会通过 SIGSEGV 信号将该进程 kill。此处 SIGSEGV 信号是段错误信号，进程被操作系统 kill 后计算机会根据 SIGSEGV 信号报告段错误。

对于 linux 来说，通过硬件的相关机制以及 linux 操作系统的信号处理机制，可以保证段错误的侦测和发生。

至于工具的使用，我认为通过使用 gdb 可以观察到 SIGSEGV 信号被发送至程序，通过 man 2 signal 可以查找该信号触发的具体原因。

实验心得：

在基本完成 PA 后，我个人总体观感，做起来感觉 PA4 明显比 PA1、PA2 和 PA3 难度大。因为对于我来说，如果结合课本来看 PA4 的任务，其实陌生的东西并不多。但实际上理论是理论，实践是实践，需要把讲义和书本上的理论实际搬到 nanos-lite 和 abstract-machine 上，许多细节需要考量。

反思了一下在做 PA4 的时候阻力重重的原因，主要是“细节”二字。各种约定和细节手册上都有明确的规定。但是有些时候因为想偷懒，我就是不愿意去 STFM，而是想当然，按照自己的想法来。这在当时那个板块可能行得通，但是如果到了跑 pal 或者其他需要综合各个板块来运行的项目的时候，就有可能出现 UB。此时 debug 会变得十分困难，程序的运行情况也不如预期。我在解决此类 BUG 的过程中，耗费了大量的时间。经历这一次的 PA 实验，更让我认识到计算机科学与技术是一门标准与创新并重的学科。

再具体到 PA4 来看，其次呢感觉讲义确实有故意忽略一些 key point，这就导致在完成了讲义提到的要求之后也许并不能将测试程序正确地跑起来，需要自己结合以往的经验和程序运行的结果，在各个模块中继续进行相应的修改。这虽然很花时间，但是也确实锻炼了自己的能力。

然后内核栈和用户栈那个地方，讲义虽然讲了很多，但是理解了很久脑袋里依旧是一团浆糊，对我来说过于抽象了 QAQ。我认为这次 PA 最难的部分，应该是栈切换那一块。毕竟虚地址和实地址的映射，在课上也讲过，也更好理解一些。

就说这么些了，做完 PA 好好过年！