PA3 实验报告

21124001 潘昕田

2022 年 12 月 21 日

1 实验进度

1. PA3 阶段 1 完成
2. PA3 阶段 2 完成（实现对 ELF 文件的 ISA 的检查）
3. PA3 阶段 3 完成（实现屏幕居中显示，nterm 内置指令 echo 和 help，以及 nemu 的快照）

2 必答题

2.1 理解上下文结构体的前世今生

c 指向的结构体在 ___am_asm_trap 这一函数的栈上，___am_asm_trap 则在 trap.S 里定义。上述新指令的实现，其中，ecall 是为了能够接受异常号并通过 mtvec 内存储的 ___am_asm_trap 的地址跳入 ___am_asm_trap。csrrw 等指令的实现是为了能够正确地向系统寄存器写入或者读取。$ISA-nemu 则定义了对应指令集的上下文结构体的具体组成。这个结构体的赋值在 ___am_asm_trap 中实现。通过将其地址值赋予 a0 寄存器（riscv32 函数传参的第一个地址值）从而使得 ___am_irq_handler 能够调用该结构体。

2.2 理解穿越时空的旅程

根据约定，ecall 通过 a7 寄存器获取异常号，因此 yield 首先利用 li 伪指令向 a7 写入 yield 的异常号，接着调用 ecall，nemu 执行 ecall 时会往 mcause 写入异常号，往 mepc 写入当前 pc 值，接着取出 mtvec 的值（即 ___am_asm_trap 的入口地址），接着跳入 ___am_asm_trap，

___am_asm_trap 在将相应的寄存器值存入上下文结构体后跳入 ___am_irq_handler，___am_irq_handler 通过上下文结构体中 mcause 的值识别出自陷的异常号，进而封装为 EVENT_YIELD 事件，进而转入 nanos-lite 中 do_event 函数，识别出 EVENT_YIELD 事件后系统仅将上文结构体中存储的 mepc 的值加 4 接着一路返回到 ___am_asm_trap 中，___am_asm_trap 通过上下文结构体恢复上下文后，再调用 mret 通过 mepc 中存储的 pc 值恢复 cpu 的 pc 值，此时 pc 值已加 4，将执行 ecall 后的指令，完成整个 yield 调用过程。

2.3 hello 程序是什么，它从而何来，要到哪里去

hello 程序的 ELF 文件一开始存储在 ramdisk.img 上，其代码，数据等均存储在 ELF 文件中，通过 naive_uload 加载到内存中，之所以能加载到正确的位置，是因为 ELF 文件内规定了相应的加载地址，它的第一条指令在 _start 函数中，由 navy-apps 提供该环境，而 ELF 文件中含有对于程序 entry 的地址的描述，读取 entry 的地址并将其转换为 entry 函数即可执行函数，第一个 Hello World 通过 write 直接调用系统调用 write 然后向 stdout 输出对应的值，而 nanos-lite 直接调用 serial_write 通过 am 的 ioe 向屏幕输出，后续的字符则通过 printf 输出，printf 通过 malloc 申请一块缓冲区，而 malloc 会调用系统调用 brk 来进行堆区管理，之后将字符逐一写入缓冲区，而写入换行符则会直接强制刷新，即此时直接调用系统调用 write 向 stdout 输出。

2.4 仙剑奇侠传究竟如何运行

库函数负责读取”mgo.mkf” 文件并解析其相关内容，而读取文件的操作需要 nanos-lite 中的文件系统的支持，而库函数的文件读取操作需要通过 libos 中的系统调用来实现，libos 的系统调用的实现涉及到异常处理，而异常处理则需要 nemu 的指令支持（当然库函数的执行本身也需要 nemu 的支持），上下文的切换则需要 am 的支持，这样，库函数的文件读取才能顺利地从 libos 跳转至 ___am_asm_trap，再跳转至 ___am_irq_handler，进而跳转至 do_event，再调用 do_syscall，之后调用 fs_read 来实现文件读取，最后再返回原来的仙剑程序去执行。

3 选做题

3.1 从加 4 操作看 CISC 和 RISC

CISC 采取硬件实现加 4 可以降低软件的复杂度，节约程序的空间大小，但提升了硬件的复杂度；RISC 通过软件实现加 4 则是降低了硬件的复杂度，代价则是增加了程序占据的空间，提升了软件的复杂度。CISC 在早期内存存储较小时具有一定的优势，而现今的内存大小则不存在如此苛刻的的节约空间大小的需求，降低硬件复杂度可以更方便实现集成度更高，速度更快的芯片，因此就现今的情况而言 RISC 具有更大的优势，然而 CISC 占据个人 PC 市场时间长，大量软件都依 CISC 架构，因此 RISC 架构目前依旧需要一定的时间取代 CISC。

3.2 如何识别不同格式的可执行文件?

GNU/Linux 加载程序需要 ELF 文件，而 Windows 使用的 PE，二者不兼容，PE 并不存在 ELF 文件所需的魔数，因此在 GNU/Linux 下执行一个从 Windows 拷过来的可执行文件, 将会报告" 格式错误"。

3.3 冗余的属性?

FileSiz 小于 MemSiz 是因为存在未初始化的全局变量，这些变量通常默认赋值为 0，不需要在 ELF 文件中特别记录。

3.4 为什么要清零?

[VirtAddr + FileSiz, VirtAddr + MemSiz) 存储的是未初始化的全局变量，这些变量通常默认赋值为 0。

4 总结

pa3 的完成前半段较为顺利，最后一部分，即仙剑的完善以及部分选做的实现则十分困难，由于感染了新冠，导致很多想要实现和完善的部分均没有很好去完成，这也算是 PA3 最后的一大遗憾吧，不过整个 PA3 还是让我更为深入地了解了计算机系统是如何实现异常机制的，尽管距离真实的操作系统距离依然较远，但是让我感受到了底层机制的实现的基本方法，还是颇有收获。