PA3 实验报告

韩加瑞

2023 年 12 月 26 日

1 实验进度

未完成：Flappy Bird 音效、基础设施 (3)、添加开机音乐

2 必做题

2.1 理解上下文结构体的前世今生

c 指向的上下文结构在栈中,这些上下文结构中的成员都是在 ___am_asm_trap 函数中加载到栈中的。
ISA-nemu.h 负责定义上下文结构体成员的结构，trap.S 中的汇编代码负责将前面定义的成员按照一定结构加载到栈上，再调用 ___am_irq_handle 函数进行处理，nemu 中实现的新指令即用于实现 trap.S 中的命令。

2.2 理解穿越时空的旅程

首先在 yield-test 中调用 yield() 函数，这个函数指向 am.h 中定义的 yield() 函数，我选择的是 riscv32 架构，因此这个函数会执行下面两条指令：

```
li a7, -1;
ecall;
```

前一条指令用于设置处理的参数,后一条指令在 nemu 中调用 isa_raise_intr() 函数，记录当前 pc 和异常原因，并返回异常处理函数的入口地址。接下来 nemu 跳转到异常处理函数 ___am_asm_trap 中，在 riscv32 架构中它首先将所有通用寄存器的值加载到栈中，接下来将 mepc、mcause 和 mstatus

寄存器中的值加载到栈中，再调用 ___am_irq_handle 函数根据异常原因进行处理。处理 yiled 异常时会先将 mepc 指向下一条指令，接下来调用 cte_init 注册的异常处理函数，在 yield-test 中即为 simple_trap，根据异常原因输出字符 y 并返回。之后一路返回到 _am_asm_trap 中，先将修改后的 mepc、mcause 和 mstatus 值写入寄存器中，接下来恢复通用寄存器的值。调用 mret 指令返回，mret 指令将 pc 设为 mepc 中的值，程序继续执行下一条指令，结束。

2.3 hello 程序是什么, 它从而何来, 要到哪里去

hello 程序一开始位于 nanos-lite/build 目录下的 ramdisk.img 中，在编译 nanos-lite 时，通过 nanos-lite/src/resources.S 中的

```
.section .data
.global ramdisk_start, ramdisk_end
ramdisk_start:
.incbin "build/ramdisk.img"
ramdisk_end:
```

语句将 ramdisk.img 中 hello 的程序段和数据段装载到可执行文件的.data 节中，并定义了 ramdisk_start 和 ramdisk_end 两个变量来指示 hello 程序的起始和终止位置。

在运行 nanos-lite 程序时，首先调用 init_proc 函数初始化程序，之后调用 naive_uload 程序将原本位于到.data 节的 hello 程序需要加载的 program segment 加载到虚拟内存中对应的地址，并通过 elf 头中的 e_entry 得到第一条指令的地址，并令 entry 函数指向程序第一条指令所在的地址。程序加载结束后，通过执行 entry 函数运行加载到内存中的 hello 程序。

对于 hello 中的每一个字符，首先是 printf 函数将想要输出的字符串格式化，之后调用 malloc 函数申请缓冲区来存放这些字符串，malloc 函数调用 _sbrk 向 nanos-lite 发起系统调用 SYS_brk，检查返回值，若为 1 则返回指向缓冲区的指针以供字符串存放。之后 printf 函数调用 _write 函数并传入文件流描述符、存放字符串的缓冲区指针和输出的字符数量，_write 发起系统调用 SYS_write，nanos-lite 处理系统调用，将字符串输出到串口，并返回输出的字符数量，nemu 读取串口中的字符并将其打印到屏幕上。printf 函数根据返回值判断是否所有字符都被输出，否则继续调用 _write 函数并

传入未被输出的字符串。

2.4 仙剑奇侠传究竟如何运行

PAL 通过调用 fopen、fseek 和 fread 函数来读取 mgo.mkf 中的像素信息，这三个函数分别发起系统调用 sys_open、sys_lseek，sys_read，nanos-lite 在处理这些系统调用时会根据文件路径从 ramdisk 中读取 mgo.mkf 的内容并将其写入缓冲区中返回。在得到 mgo.mkf 中的内容后，PAL 首先对这些信息进行解码得到像素信息，接下来调用 SDL 库中的函数按照帧率更新画布中的像素信息，并调用 SDL_UpdateRect 函数。这个函数首先会根据画布和位置信息生成所需像素的缓冲区，之后调用 NDL_UpdateRect，将缓冲区中的信息写入到文件/dev/fb 中，在 nanos-lite 中会将该缓冲区中的信息通过调用 ___am_gpu_fbdraw 函数传输到 nemu 中的像素缓冲区，并将同步信号置 1，更新屏幕上的像素。