2012543-刘沿辰-PA5报告

学号：2012543
姓名：刘沿辰
专业：计算机科学与技术
指令集：RISC-V 32
日期：2023.6.13

实验目的

- 用整数运算来支持浮点数，由此支持仙剑奇侠传的战斗环节
- 测试NEMU这个教学系统的性能瓶颈，寻找性能提升的问题所在
- 实现JIT，完成性能的提升（未完成）

实验内容

- 第一阶段：浮点数的支持
- 第二阶段：寻找性能突破瓶颈
- 第三阶段：JIT的实现（未完成）

实验过程

第一阶段

这一部分的内容实际上在PA3的定点算数中已经完成，具体代码如下：

```
/* Multiplies a fixedpt number with an integer, returns the
result. */
static inline fixedpt fixedpt_muli(fixedpt A, int B) {
        return (fixedpt)(A * B);
}

/* Divides a fixedpt number with an integer, returns the
result. */
static inline fixedpt fixedpt_divi(fixedpt A, int B) {
        return (fixedpt)(A / B);
}

/* Multiplies two fixedpt numbers, returns the result. */
static inline fixedpt fixedpt_mul(fixedpt A, fixedpt B) {
        return (fixedpt)((A * B) >> 8);
}


/* Divides two fixedpt numbers, returns the result. */
static inline fixedpt fixedpt_div(fixedpt A, fixedpt B) {
        return (fixedpt)((A << 8) / B);
}

static inline fixedpt fixedpt_abs(fixedpt A) {
        return A >= 0 ? A : (-A);
}

static inline fixedpt fixedpt_floor(fixedpt A) {
        if ((int)A == 0x7fffffff || (int)A == 0xffffffff ||
(int)A == 0) return A;
        else if ((int)A < 0) return A | 0xff;
        else if ((int)A > 0) return A & 0xffffff00;
        assert(0);
}

static inline fixedpt fixedpt_ceil(fixedpt A) {
```

```
        if ((int)A == 0x7fffffff || (int)A == 0xffffffff ||
(int)A == 0) return A;
        if ((int)A < 0) return (fixedpt)(-((-(int)A) &
0xffffff00));
        if ((int)A > 0) return (fixedpt)(-((-(int)A) |
0xff));
        assert(0);
}
```

我们可以在PA中进入战斗界面来验证这一实现的正确性：

验证完毕

第二阶段

让仙剑奇侠传跑起来的时候真是兴奋的睡不着，有一种自己搭建了一个逻辑王国的快感。曾经不知在何处，PA告诉我们**抑制住你优化代码的冲动！！！** 原因其实就是我们的优化最终可能根本不在系统的关键路径上，大量的优化时间白白浪费。因此，如果想要优化一段代码，优化那些`hot code`才是更优的方案。

GNU/Linux内核提供了perf工具，可以帮我们有效查看NEMU的性能瓶颈：

```
yyy@yyy-virtual-machine:~/Documents/ics2022$ sudo apt-get install linux-tools-$(
uname -r) linux-tools-generic -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
E: Unable to locate package linux-tools-5.19.0-35-generic
E: Couldn't find any package by glob 'linux-tools-5.19.0-35-generic'
E: Couldn't find any package by regex 'linux-tools-5.19.0-35-generic'
```

我的Ubuntu22.04无法安装性能工具perf，经过多次换源与更新之后依然无法运行。Linux哲学告诉我们：无论我要干什么事情，一定有适合的工具能帮助我，因此我找到了Linux经典性能分析工具top。

```
top - 22:17:47 up  5:15,  1 user,  load average: 1.16, 0.59, 0.48
Tasks: 316 total,   2 running, 314 sleeping,   0 stopped,   0 zombie
%Cpu(s): 26.6 us,  3.2 sy,  0.0 ni, 69.3 id,  0.0 wa,  0.0 hi,  0.9 si,  0.0 st
MiB Mem :   7914.8 total,    246.8 free,   2413.5 used,   5254.4 buff/cache
MiB Swap:   2048.0 total,   2043.7 free,      4.3 used.   5169.9 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
  63013 yyy       20   0  375604 209856  55724 R  99.7   2.6   0:37.48 riscv32-nemu-in
   1885 yyy       20   0 4869212 334916 129920 S  11.0   4.1   8:12.15 gnome-shell
   2210 yyy       20   0  298248  84252  56976 S   7.0   1.0   3:36.64 Xwayland
  23401 root      20   0       0      0      0 I   1.3   0.0   0:01.93 kworker/u256:0-events_unbound
    430 root     -51   0       0      0      0 S   0.7   0.0   0:23.06 irq/16-vmwgfx
  26427 root      20   0       0      0      0 I   0.7   0.0   0:01.29 kworker/3:0-events
     15 root      20   0       0      0      0 I   0.3   0.0   0:09.93 rcu_preempt
  22326 root      20   0       0      0      0 I   0.3   0.0   0:00.70 kworker/1:3-events
  59099 root      20   0       0      0      0 I   0.3   0.0   0:00.27 kworker/3:1-events
  59100 yyy       20   0   22000   4572   3712 R   0.3   0.1   0:00.68 top
      1 root      20   0  168204  13740   8480 S   0.0   0.2   0:17.23 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.13 kthreadd
      3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
      4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_par_gp
```

直接运行仙剑奇侠传，可以看到此时CPU占用率高达99.7%，大量任务被迫中断，为nemu的执行让路。

内存瓶颈?

在交互模式查看内存，结果如下：

```
top - 22:19:42 up  5:16,  1 user,  load average: 1.12, 0.78, 0.57
Tasks: 316 total,   2 running, 314 sleeping,   0 stopped,   0 zombie
%Cpu(s): 25.8 us,  1.0 sy,  0.0 ni, 72.8 id,  0.0 wa,  0.0 hi,  0.4 si,  0.0 st
MiB Mem : 34.7/7914.8   [||||||||||||||||||||||||||||
MiB Swap:  0.2/2048.0   [
```

此时内存的使用并不多，内存并未到达瓶颈。PS：不得不感慨这些经典游戏的构造之精巧，一个小小的35M空间就足以容纳一个游戏。

当然，也可以用`free`命令查看内存情况：

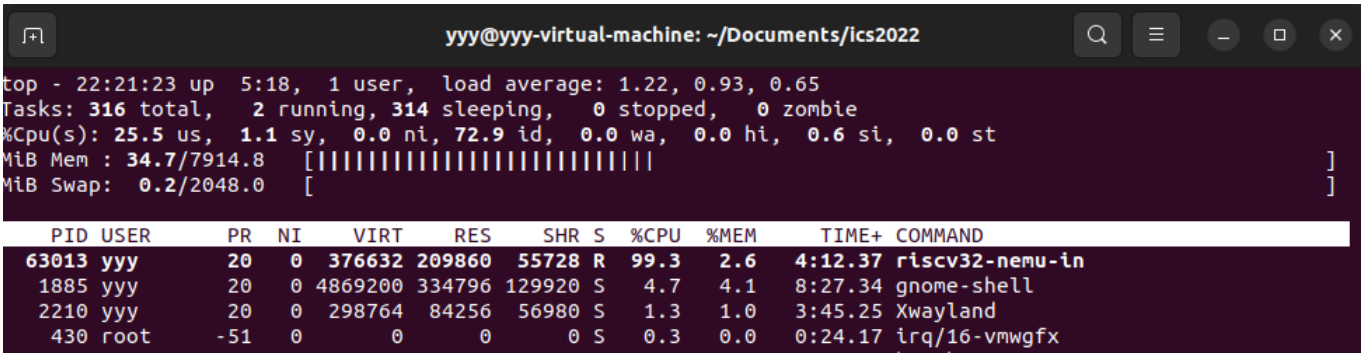
```
yyy@yyy-virtual-machine:~/Documents/ics2022$ free
               total        used        free      shared  buff/cache   available
Mem:         8104752     2475076      249092       42508     5380584     5290340
Swap:        2097148        4400     2092748
```

此时内存完全足够，有大量的内存剩余，这方面暂时不称为瓶颈。

> 突然想起，x86这种复杂指令集的设计之初一定程度上就是为了缩小代码空间，而x86最盛行的上世纪九十年代正好也是仙剑奇侠传诞生的日子。从这里也可以窥探计算机发展的过程，“存储”正在越来越廉价。换言之，“存储”作为当年的计算机“瓶颈”，所有的程序都必须考虑压缩存储空间，殊不知今日的存储价格再次大大降低。不同时期，不同发展状况，系统都有各自的瓶颈。

系统瓶颈?

接下来按照CPU占用率对线程排序，看到如下结果：



- 图中的第三行有一个%wa参数，这个参数代表等待I/O的CPU时间百分比，此时基本为0；
- 而第三行还有一个%id参数，这个参数代表空闲CPU时间百分比，这个参数仅仅达到了72.9（越高越好），说明CPU存在瓶颈！

分析瓶颈

作为占据了99.7%的CPU时间的nemu进程，CPU时间居然还有近1/4空闲！我们来跟踪一下这个进程的问题：

可以使用指令`strace -o output.txt -T -tt -e trace=all -p`来获取所有系统调用的时间，并将其保存到output.txt文件中进行详细分析。

NEMU作为一个模拟器，其一条指令的执行对应着native上百条指令的执行，原本应该很快的硬件操作这个时候成为了性能的大瓶颈，这也导致我们费力实现的诸如多道机制、分页机制等会大幅度拖慢程序的执行速度。还有各种addr_read和指令执行，作为一个硬件程序，在并发实现不完善的时候实在是占用了太多系统资源。

因此，如果将来要进行优化，首先想到的优化方向就是：

- 加快存储速度，但是说实话想不到方向；
- 优化中提到的JIT编译，作为RISC指令集，指令数量少，这方面的实现是占据天然优势的。

第三阶段

`//TODO`