南開大學
Nankai University

计算机学院

计算机系统设计实验报告

PA5 - 从一到无穷大：程序与性能

姓名：李颖

学号：2110939

专业：计算机科学与技术

2024 年 6 月 10 日

目录

1 实验介绍

通过实现对浮点数的支持等让《仙剑奇侠传》跑得更快，并且能够支持战斗系统。

1.1 实验目的

- 实现浮点数运算
- 通过整数来模拟实数的运算
- 理解 binary scaling 的支持

2 实现浮点数运算

2.1 浮点数的表现形式

在《计算机组成原理》课程中我们知道，浮点数在计算机的存储如下所示：

$$float = (-1)^S + M * 2^E \tag{1}$$

其中，各个字母的含义如下所示。

- S：符号位，当 S 为 0 时表示正数，为 1 时表示负数，第 0 位；
- M：尾数，即浮点数的小数部分，第 1 8 位；
- E：阶码，即科学计数法中的指数部分，第 9 31 位。

2.2 用整数表示实数

由于实现浮点数需要涉及 x87 架构的很多细节，因此可以使用 binary scaling，即通过整数模拟实数的运算，并且把该实数类型称为 FLOAT。

想要实现 FLOAT，首先可以在 navy-apps/apps/pal/src/FLOAT/FLOAT.c 的 f2F() 函数中定义一个 union 来表示这种浮点数，其中，包含符号位、阶码、尾数。

代码如下所示。

```
FLOAT f2F(float a) {
  union float_{
      struct {
  uint32_t m : 23;
  uint32_t e : 8;
  uint32_t signal : 1;
      };
      uint32_t value;

  };
  union float_ f;
  return 0;
```

```
}
```

之后，就可以将二进制数转换为浮点数，具体的步骤如下所示：

1. 用真实的指数减去固定的移码；
2. 若指数大于 7，则需要向左移动；小于 7 则小数位溢出，需要右移；
3. 判断符号位正负，若为负数则向左移动

对应的代码如下所示：

```
FLOAT f2F(float a) {
  union float_ {
    struct {
      uint32_t m : 23;
      uint32_t e : 8;
      uint32_t signal : 1;
    };
    uint32_t value;
  };
  union float_ f;
  f.value = *((uint32_t*)(void*)&a);

  int e = f.e - 127;

  FLOAT result;
  if (e <= 7) {
    result = (f.m | (1 << 23)) >> 7 - e;
  }
  else {
    result = (f.m | (1 << 23)) << (e - 7);
  }
  return f.signal == 0 ? result : (result|(1<<31));
}
```

2.3 实现浮点数的乘除法

对于浮点数乘法，只需要将两个数相乘后右移 16 位，取最后的 32 位结果即可。F_mul_F() 函数的相关代码如下所示。

```
FLOAT F_mul_F(FLOAT a, FLOAT b) {
  return (a * b) >> 16;
}
```

而对于除法来说，需要每次都进行左移，再进行计算，最后得到结果，F_div_F() 函数的代码如下所示。

```
FLOAT F_div_F(FLOAT a, FLOAT b) {
  FLOAT result = Fabs(a) / Fabs(b);
```

```
  FLOAT m = Fabs(a);
  FLOAT n = Fabs(b);
  m = m % n;

  for (int i = 0; i < 16; i++) {
    m <<= 1;
    result <<= 1;
    if (m >= n) {
      m -= n;
      result++;
    }
  }
  if (((a ^ b) & 0x80000000) == 0x80000000) {
    result = -result;
  }
  return result;
}
```

2.4 FLOAT 与 int 相互转换

除了 FLOAT 之间的算术问题，还需要考虑 FLOAT 和 int 之间的相互转换问题。

对于 FLOAT 转化为 int，只需要将其右移 16 位即可。此外，还需要判断符号位。int 转化为 FLOAT 则相反，需要左移 16 位。navy-apps/apps/pal/include/FLOAT.h 中的代码如下所示。

```
static inline int F2int(FLOAT a) {
  if ((a & 0x80000000) == 0) {
    return a >> 16;
  }
  else {
    return -((-a) >> 16);
  }
}

static inline FLOAT int2F(int a) {
  if ((a & 0x80000000) == 0) {
    return a << 16;
  }
  else {
    return -((-a) << 16);
  }
}
```

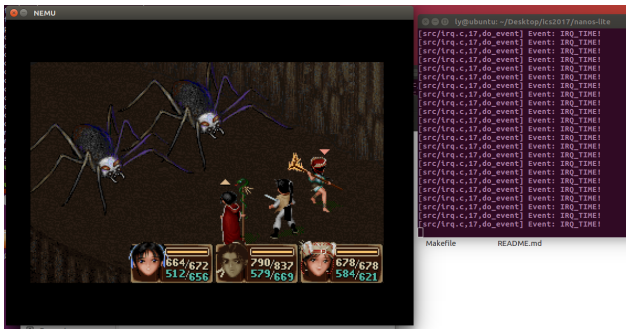
之后，还需要实现 int 和 FLOAT 之间的乘除法运算。仅需要调用刚才的 int2F() 函数和 F2int() 函数即可。补充的函数也在 FLOAT.h 中，代码如下所示。

```
static inline FLOAT F_mul_int(FLOAT a, int b) {
  return F_mul_F(a, int2F(b));
}
```

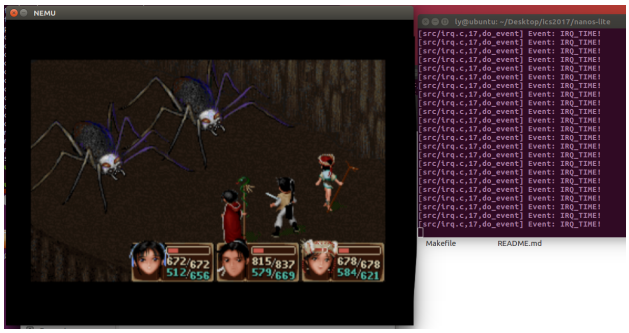
```
4
5 static inline FLOAT F_div_int(FLOAT a, int b) {
6   return F_div_F(a, int2F(b));
7 }
```

3 实验结果

运行《仙剑奇侠传》，并且一直进行到战斗部分，战斗界面如下所示。



进行了几个回合后，发现战斗能正常进行，双方都可以正常减少血量。



4 其他

在实现过程中发现有指令未完善。

4.1 ror 指令

group2 中的 ror 指令并未实现，因此，首先在 all-instr.h 中注册 ror 指令。

```
1 make_EHelper(ror);
```

在 exec.c 中修改 gp2，如下所示。

```
1   /* 0xc0, 0xc1, 0xd0, 0xd1, 0xd2, 0xd3 */
2 make_group(gp2,
3     EX(rol), EX(ror), EMPTY, EMPTY,
4     EX(shl), EX(shr), EMPTY, EX(sar))
```

在 logic.c 中补充执行函数。

```
make_EHelper(ror){
    rtl_li(&t0,31);
  rtl_and(&id_src->val,&t0,&id_src->val);
    rtl_li(&t1,(8*id_dest->width)-id_src->val);
    rtl_shr(&t2,&id_dest->val,&id_src->val);
    rtl_shl(&t3,&id_dest->val,&t1);
  rtl_or(&t2,&t2,&t3);

  if(id_src->val==1){
    rtl_msb(&t3,&t2,id_dest->width);
    rtl_shl(&t1,&t2,&id_src->val);
    rtl_msb(&t1,&t1,id_dest->width);
    rtl_xor(&t1,&t1,&t3);
    rtl_set_OF(&t1);
  }

  operand_write(id_dest,&t2);
    print_asm_template2(ror);
};
```

4.2 shld 指令与 shrd 指令

在查询手册的过程中，发现移位指令 shld 与 shrd 未实现。因此，补充 all-instr.h，注册这两条指令。

```
make_EHelper(mov_r2cr);
make_EHelper(mov_cr2r);

make_EHelper(shrd);
make_EHelper(shld);
```

在 exec.c 的 opcode_table 的 2-bytes 处修改 0x20 处的内容如下。

```
/* 0x20 */  IDEXW(G2E,mov_cr2r,4), EMPTY,IDEXW(E2G,mov_r2cr,4), EMPTY,
```

在 logic.c 中实现 shrd 和 shld 的执行函数。

```
make_EHelper(shld)
{
  //TODO PA5
  rtl_shl(&t0,&id_dest->val,&id_src->val);
  rtl_li(&t2,id_src2->width);
  rtl_shli(&t2,&t2,3);
  rtl_subi(&t2,&t2,id_src->val);
  rtl_shr(&t2,&id_src2->val,&t2);
  rtl_or(&t0,&t0,&t2);
  operand_write(id_dest,&t0);
```

```
  rtl_update_ZFSF(&t0,id_dest->width);
  print_asm_template3(shld);
}

make_EHelper(shrd)
{
  //TODO PA5
  rtl_shr(&t0,&id_dest->val,&id_src->val);
  rtl_li(&t2,id_src2->width);
  rtl_shli(&t2,&t2,3);
  rtl_subi(&t2,&t2,id_src->val);
  rtl_shl(&t2,&id_src2->val,&t2);
  rtl_or(&t0,&t0,&t2);
  operand_write(id_dest, &t0);
  rtl_update_ZFSF(&t0, id_dest->width);
  print_asm_template3(shrd);
}
```

在 system.c 中实现 mov_r2cr 和 mov_cr2r 的执行函数。

```
make_EHelper(mov_r2cr) {
//  Log("%d",id_dest->reg);
  switch(id_dest->reg){
    case 0:
    cpu.CR0.val=id_src->val;
    break;
  case 3:
    cpu.CR3.val=id_src->val;
    break;
    default:Log("gg");assert(0);
  }

  print_asm("movl %%%s,%%cr%d", reg_name(id_src->reg, 4), id_dest->reg);
}

make_EHelper(mov_cr2r) {

switch(id_dest->reg){
    case 0:
    t0=cpu.CR0.val;
    rtl_sr(id_src->reg,4,&t0);
    break;
    case 2:
    t0=cpu.CR3.val;
    rtl_sr(id_src->reg,4,&t0);
    break;
    default:Log("gg");assert(0);
}

```

```
30   print_asm("movl %%cr%d,%%%s", id_src->reg, reg_name(id_dest->reg, 4));
31
32 #ifdef DIFF_TEST
33   diff_test_skip_qemu();
34 #endif
35 }
```

5 实验总结

至此，本学期的学习告一段落。在这门课程中，我不仅巩固了计算机组成原理的基础知识，更对计算机系统的整体架构有了全面的了解。通过课堂上老师的悉心讲解和一系列精心设计的实验作业，我收获了无比丰富的知识和实践经验。

当我在实验手册的指引下，逐步将《仙剑奇侠传》这款经典游戏在计算机上成功运行起来时，那种喜悦和成就感无法用言语表达。在实验中，我深刻体会到了 Nanos-lite、AM 和 NEMU 这些工具之间的协同工作和相互依赖，它们共同构建了一个完整而高效的计算机系统。

最后，我要特别感谢老师和所有助教的辛勤付出。他们的指导和帮助让我能够顺利完成本学期的学习，收获满满。我将带着这段美好而充实的学习经历，继续努力学习，不断提升自己的能力和水平。