

pa 2-1

1. 使用 hexdump 命令查看测试用例的.img 文件，所显示的.img 文件的内容对应模拟内存的哪一个部分？指令在机器中表示的形式是什么？

用 hexdump 查看的 img 文件（以 16 进制显示）：

```
pa221220074@b1db3f1c05ac:~/pa_nju/testcase/bin$ hexdump add.img
00000000 00e9 0000 5500 e589 ec83 c710 f445 0000
00000010 0000 45c7 00fc 0000 eb00 c748 f845 0000
00000020 0000 34eb 458b 8bfc 8514 3000 0003 458b
00000030 8bf8 8504 3000 0003 0c8d 8b02 f445 508d
00000040 8901 f455 048b 2085 0330 3900 74c1 b806
00000050 0001 0000 ff82 f845 458b 83f8 07f8 c476
00000060 45ff 8bfc fc45 f883 7607 83b0 fc7d 7408
00000070 b806 0001 0000 8382 f87d 7408 b806 0001
00000080 0000 b882 0000 0000 b882 0000 0000 c3c9
00000090 0000 0000 0000 0000 0000 0000 0000 0000
*
00010000 0014 0000 0000 0000 7a01 0052 7c01 0108
00010010 0c1b 0404 0188 0000 001c 0000 001c 0000
00010020 efe5 ffff 008b 0000 4100 080e 0285 0d42
00010030 0205 c587 040c 0004 0000 0000 0000 0000
00010040 0000 0000 0000 0000 0000 0000 0000 0000
*
00030000 0000 0000 0001 0000 0002 0000 ffff 7fff
00030010 0000 8000 0001 8000 fffe ffff ffff ffff
00030020 0000 0000 0001 0000 0002 0000 ffff 7fff
00030030 0000 8000 0001 8000 fffe ffff ffff ffff
00030040 0001 0000 0002 0000 0003 0000 0000 8000
00030050 0001 8000 0002 8000 ffff ffff 0000 0000
00030060 0002 0000 0003 0000 0004 0000 0001 8000
00030070 0002 8000 0003 8000 0000 0000 0001 0000
00030080 ffff 7fff 0000 8000 0001 8000 fffe ffff
00030090 ffff ffff 0000 0000 fffd 7fff fffe 7fff
00030a00 0000 8000 0001 8000 0002 8000 ffff ffff
00030b00 0000 0000 0001 0000 fffe 7fff ffff 7fff
00030c00 0001 8000 0002 8000 0003 8000 0000 0000
00030d00 0001 0000 0002 0000 ffff 7fff 0000 8000
00030e00 fffe ffff ffff ffff 0000 0000 fffd 7fff
00030f00 fffe 7fff ffff 7fff fffc ffff fffd ffff
00031000 ffff ffff 0000 0000 0001 0000 fffe 7fff
00031100 ffff 7fff 0000 8000 fffd ffff fffe ffff
00031200
```

.img 文件对应模拟内存中的 0x30000 之后的部分，即装载后的指令集

NEMU模拟程序执行：1. 装载程序

1.1 NEMU初始化模拟内存（PA 2-1的装载）



指令在机器中表示的形式为二进制 01 序列，然后通过 opcode 的对应来解码。

2. 如果去掉 instr_execute_2op()函数前面的 static 关键字会发生什么情况？为什么？

在不同的指令中 `instr_excute_2op()` 函数具有不同的实现 需要分别写 `static` 关键字限定了函数只能在这个文件内部使用，具有封装性。
如果去掉 `static` 则会报错 `multiple definition`

```
collect2: error: ld returned 1 exit status
make[1]: *** [Makefile:13: nemu] Error 1
make[1]: Leaving directory '/home/pa221220074/pa_nju/nemu'
cd testcase && make
```

3. 为什么 `test-float` 会 fail? 以后在写和浮点数相关的程序的时候要注意什么?

test-float.c

网络正常 心情如何 > 顺利 遇到挑战 受挫 绝

```
1 #include "trap.h"
2
3 int main()
4 {
5
6     float a = 1.2, b = 1;
7     float c = a + b;
8     if (c == 2.2)
9         ;
10    else
11        HIT_BAD_TRAP;
12    c = a * b;
13    if (c == 1.2)
14        ;
15    else
16        HIT_BAD_TRAP;
17
18    c = a / b;
19    if (c == 1.2)
20        ;
21    else
22        HIT_BAD_TRAP;
23
24    c = a - b;
25    if (c == 0.2) // this will fail, and also fails for native program,
26                  interesting, can be used as a quiz
27        ;
28    else
29        HIT_BAD_TRAP;
30
31    HIT_GOOD_TRAP;
32    return 0;
33 }
```

在浮点数的运算中，由于精度问题会存在一些误差，但只要结果与标准答案非常接近即可，因此不应该用等号去判断是否计算正确，而是应该设置一个比较范围（可以用 计算结果减去标准结果趋于 0 来判断）