



计算机系统基础


实验报告

PA 2

计算机科学与技术系
191220008 陈南曠

2-1:

1、使用 `hexdump` 命令查看测试用例的 文件，所显示的 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？

所显示的 文件的内容对应模拟内存的 Physical Memory 中 Testcase Binary 的部分，包含代码区、只读数据区和可读写数据区。

指令在机器中的表示是一串二进制机器码（用十六进制表示）

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？

会显示 `multiple definition of 'instr_execute_2op'`

```
multiple definition of `instr_execute_2op';
```

同一个工程中不应该有同名函数，若需要使用同名函数而不用 `static`，在程序调用该函数时会无法确定调用同名函数中的哪一个，导致程序无法进行。

故加上 `static` 后：

- (1) 加了 static 后表示该函数失去了全局可见性，只在该函数所在的文件作用域内可见
- (2) 当函数声明为 static 以后,编译器在该目标编译单元内只含有该函数的入口地址,没有函数名,其它编译单元便不能通过该函数名来调用该函数。

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?

```
float a = 1.2, b = 1;
float c = a + b;

if (c == 2.2) ;
else HIT_BAD_TRAP;

c = a - b;
if (c == 0.2) ;
else HIT_BAD_TRAP;
```

1.2 = 00111111100110011001100110011010
1.0 = 00111111100000000000000000000000
2.2 = 01000000000011001100110011001101
0.2 = 00111110010011001100110011001101

1.2 + 1.0 => 右规1次后的结果和2.2比
中间结果:
001111111 10 00110011001100110011010

1.2 - 1.0 => 左规3次后的结果
0.2 无法用二进制精确表示
中间结果:
001111111 00 00110011001100110011010

浮点数运算的精度问题

由于浮点数在机器中以阶码+尾数的方式存储，因此部分浮点数无法精确表示，这些数的尾数是无限循环数串的一部分。当浮点数运算需要进行左规时（如：1.2-1.0），尾数的末端只能用零补齐，进而导致原有的无限循环的尾数部分被丢失，所以结果与预期结果有细微的差距，因而会 fail。

因此，在浮点数相关的程序里，在进行浮点数的大小比较时，需要将 if (A == B) 替换为 if (fabs (A-B) < epsilon)，以消除规格化导致的细微误差。