



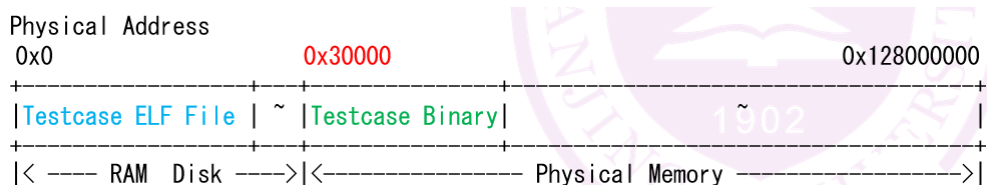
# PA2-1实验报告

191220163 计算机科学与技术系 张木子苗

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？

`hexdump`主要用来查看“二进制”文件的十六进制编码，查看文件后可发现：

- (1) 文件直接从模拟内存中的物理内存，0x30000处开始



带有RAM Disk时的NEMU模拟内存划分方式



- (2) 指令在机器中的表现形式为二进制位串，用hexdump看到的是可以解析的16进制码，汇编语言是对应机器语言的助记符。
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？  
`static`关键字将函数 `instr_execute_2op()` 的作用域限制在当前源文件，即该函数只可被当前源文件内的其他函数调用，不能被其他文件的函数调用。  
如果将其去掉，由于 `instr_execute_2op()` 函数在多个源文件中出现，其定义有多种，会出现重定义现象："multiple definition of `instr_execute_2op`"
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代码段如下：

```
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 = 0 01111111 00110011001100110011010

1.0 = 0 01111111 000000000000000000000000

2.2 = 0 10000000 00011001100110011001101

0.2 = 0 01111100 10011001100110011001101

**1.2 + 1.0 =>**

不需要对阶，中间结果：

0 01111111 10 00110011001100110011010

右规1次后的结果和2.2比，相同

**1.2 - 1.0 =>**

不需要对阶，中间结果：

0 01111111 0 00110011001100110011010

左规3次后的结果和0.2比，不同，Hit Bad Trap

原因：0.2 无法用二进制精确表示，浮点数的精度问题导致运算过程中发生偏差，不能用 "==" 判断结果

**这提醒我们以后写和浮点数相关的程序时，要注意浮点数的精度问题**