# Programming Assignment 1

161220049 黄奕诚

September 18, 2017

## I. 实验进度

完成所有内容，并把所有“选做”内容完成.

## II.必答题

### 1.理解基础设施

若符合所有假设，且没有实现简易调试器，则该学期我在调试上花费的时间为：500×90×20×30s=270000s=75h≈3days.

若使用简易调试器，则只需 1/3 ×75h=25h≈1day

于是节省了约两天的时间.

### 2.查阅 i386 手册

(a)EFLAGS 寄存器中的 CF 位是什么意思?

阅读范围：P50-52

(b)ModR/M 字节是什么?

阅读范围：P241-248

(c)mov 指令的具体格式是怎么样的?

阅读范围：P345-353

### 3.shell 命令

此时 nemu/目录下的所有.c 文件总共有 3360 行，.h 文件总共有 733 行.我通过在 nemu 目录下输入指令 find . -name *.c | xargs wc -l | sort -n 以及 find . -name *.h | xargs wc -l | sort -n 来完成这一行数统计.

在做 pa1 之前，.c 文件总行数为 2758，.h 文件总行数为 725，因此我的 pa1 的.c 文件行数为 602，.h 文件行数为 8.

除去空行之外，nemu/目录下的所有.c 文件行数为 2817，.h 文件行数为 578.

（以上行数统计时间为 9 月 18 日下午 3 点，可能会在后续的调试与添加功能中发生变化）

### 4.gcc 编译

**(a)作用**

-Wall:使 gcc 产生尽可能多的警告信息.

-Werror:在所有警告的地方停止编译，视为错误

**(b)原因**

作为一名未来（或者说现在）的程序员，我们必须要提高程序的鲁棒性和稳定性，不容许“将就”，只有将每一个可能出现 bug 的点都扫除，才能步步为营，走向最终的成功！

## III.问题&思索&解决方法

**PA1-1**

起初我面对庞大的 nemu 略有茫然，不过通过浏览各个文件，在脑中建立了 nemu 的基本架构，于是很快就上手写 pa1-1 了.这几个指令都很容易，只要找到接口以及具备基本的 C 语言功底就能完成.

**PA1-2**

首先，面临的问题是:我对正则表达式一无所知.于是便通过 man 以及网上的元字符表对正则表达式的查找与替换有了基本的理解，解析字符方面较顺利.

其次,将解析出来的 token 放入 tokens 数组，起初因疏忽忘记判断解析出来的 token 数量是否超过数组的最大容量，后来经 debug 成功改正.

再者，通过分治算法与递归计算表达式的值。因实验讲义已经给出了框架和算法，把它填补并不难，不过在 debug 的过程中，我发现并解决了如下问题：

(a)除法以及取模运算的零除问题.由于是整数的除法，所以不容许除数为 0，作一步判断即可.

(b)括号的配对问题.面对这个式子：(1+1)*(2+2)，我刚开始的程序判其为输入不合法，实际上是因为 check_parentheses 判其为 true 的原因.只要进一步判断式子里是否有多对括号即可.

(c)扩充负数、解引用的功能时，经过思考，发现可以修改 eval 中 p==q 的代码，最后如果剩余两个 token，则不可能再有二元运算符，只会是*NUM 或是-NUM.

**PA1-3**

管理监视点的重要在于对链表的插入和删除操作有正确的实现，因为写过相关程序，所以较为顺利，想说明的关键点在于：对结点或其后继重新赋值时，其自身是否已经发生了改变。注意到这点便可成功实现.

随后只要在 cpu-exec.c 中相应位置调用表达式求值的接口即可，在此之前必须检验输入的表达式的合法性.总体来说 PA1-3 比较容易.

## IV.蓝框思考题

**1.**在 cmd_c()函数中, 调用 cpu_exec()的时候传入了参数-1, 如此就不会进入 for(;n>=0;n--)这个循环，则 nemu_state 不会变为 NEMU_STOP，因此程序可以一直执行到 hit good trap.

**2.**opcode_table 的类型是结构体 opcode_entry.

**3.**区分负号和减号：观察前一个字符是否是数字或右括号，若是，则为减号，反之为负号.
负号是个单目运算符，分裂时要注意将最终 p==q 的情况扩展到 q==p+1 然后判断 p 位置上是否是负号.

**4.**框架代码中定义 wp_pool 等变量的时候使用了关键字 static, static 在此处的含义是：变量作用域只限于该文件，不能被其他文件访问（相当于 java 里面的一种 private 类）.

**5.**关于断点的工作原理的三个问题回答：

a.指令长度为 1 个字节是必须的.如果变成 2 个字节，则不能正常工作，因为这样就无法将一些单字节指令替换成 breakpoint.

b.如果把断点设置在指令的非首字节(中间或末尾), 依然会在执行这个指令时中断.缘由可能是程序在执行过程中仍然经过了被替换成断点的指令而被中断.

c.模拟器和调试器的区别：模拟器是建立在一个计算机系统上，通过模拟来实现另一个计算机系统的功能.而调试器基于中央处理器的异常机制，并由操作系统的异常分发\事件分发的子系统（或模块）负责将其封装处理后，以比较友好的方式与调试器进行实时交互。与 NEMU 相比，GDB 是基于 ptrace 这个系统调用，利用 ptrace 系统调用，在被调试程序和 gdb 之间建立追踪关系，截获信号，查看被调试程序相应的内存地址，并控制被调试的程序继续运行.