PA2 实验报告

221240075 高歌

实验进度：

完成 PA2 所有内容。

必答题：

1. **RTFSC 请整理一条指令在 NEMU 中的执行过程**

程序被交叉编译成二进制指令被装入 NEMU 的内存中，NEMU 通过一系列函数来取指令，解码指令，执行指令

2. **编译与链接 在 nemu/include/rtl/rtl.h 中，你会看到由 static inline 开头定义的各种 RTL 指令函数. 选择其中一个函数，分别尝试去掉 static，去掉 inline 或去掉两者，然后重新进行编译，你可能会看到发生错误. 请分别解释为什么这些错误会发生/不发生？你有办法证明你的想法吗?**

Static defined **local** variables do not lose their value between function calls. In other words they are global variables, but scoped to the local function they are defined in.<函数内定义的变量用 static 关键字修饰就会被编译器放入静态存储区，应该是被放在.data 表内，其实就相当于一个另类的全局变量了.>

A static **global** variable or a function is “seen” only in the file it’s declared in.<字面意思>

去掉 inline:static 和 static inline 其实没有很大的不同,只是函数的调用方式改变了，我们在 gcc 中加入了-Werror 把所有的警告都当成 error 来处理,把-Werror 去掉就可以了

去掉 static:因为有 inline 关键字的存在所以，程序就像 define 一样会在调用处展开所以定义在头文件中的无 static 有 inline 的函数不会出现多次定义,只要把 makefile 文件中的-Werror 去掉就可以编译链接成功

去掉 static inline:会报错因为头文件会被许多文件引用所以如果去掉 static inline,这个函数就会被多次定义

3.编译与链接

在 nemu/include/common.h 中添加一行 volatile static int dummy; 然后重新编译 NEMU. 请问重新编译后的 NEMU 含有多少个 dummy 变量的实体？你是如何得到这个结果的?

添加上题中的代码后，再在 nemu/include/debug.h 中添加一行 volatile static int dummy; 然后重新编译 NEMU. 请问此时的 NEMU 含有多少个 dummy 变量的实体？与上题中 dummy 变量实体数目进行比较，并解释本题的结果.

修改添加的代码, 为两处 dummy 变量进行初始化:volatile static int dummy = 0; 然后重新编译 NEMU. 你发现了什么问题? 为什么之前没有出现这样的问题? (回答完本题后可以删除添加的代码.)

74 个

74 个.

两个初始化后会出现多次定义的错误，强弱定义的问题