PA2 必答题

1）请整理一条指令在 NEMU 中的执行过程.

一条指令执行的主要函数就是 exec_once,这个函数主要做两件事情，一是进行取指、译码、计算等（isa_exec）函数，二是进行 pc 的更新（update_pc）函数。

isa_exec 函数：它主要模拟了一个 CPU 取指、译码以及执行的过程，首先它调用 instr_fetch 函数来访问内存:读取 pc 对应位置的指令，并就算出如果进行序列执行的时候的 pc 的值：seq_pc。然后再调用 idex 函数，这个函数会根据指令的 opcode 来调用 opcode_table 里面相应的译码以及执行函数，拿 LUI 指令来举例，idex 会先调用它的 decode_U 函数，这个函数通过 make_DHelper（U）来声明，他的主要作用就是把立即数低位填 0 加载到 id_src,然后再确定 id_dest 对应的寄存器。再调用他的 exec_lui 函数，这个函数通过 make_EHelper(lui)声明，它的作用就是把 id_src 的值放在 id_dest 对应的寄存器之中。

update_pc 函数：当 isa_exec 执行的指令最后不进行跳转时，此函数会让当前的 pc 置为 seq_pc，否则置为跳转的 pc。至此，一条指令的执行过程完毕。

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?

去掉 static 之后，函数不会报错，因为这里的 static 作用是避免内联失败时函数体报重复定义的错，但我们设计的 rtl 函数都是比较简单，没有 for 循环等语句的函数，所以都能内联成功。

去掉 inline 之后，会报函数没有被引用的错误，其实这只是一个 warning，但是因为我们编译时加了-error 参数，所以最后报了 error 的错误。

全部都去掉之后，就会报函数重复定义的错误，因为这个函数是在 rtl 文件里面定义的，这个头文件被多个文件引用了，导致了函数体的多次定义

3）在 nemu/include/common.h 中添加 volatile static int dummy;后重新编译 NEMU。重新编译后的 NEMU 有 1 个 dummy 变量的实体，因为在这里用

volatile 定义了一个 dummy。上一问题条件下在 nemu/include/debug.h 中添加 volatile static int dummy。重新编译后的 NEMU 有 1 个 dummy 变量的实体。因为两个文件中都使用了 volatile 进行 dummy 的定义，所以不会发生冲突。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，会报错。因为当 volatile 修饰的 dummy 被赋予了确定的值之后，两个 dummy 都希望在栈中建立名为 dummy 的静态变量，此时就会有重复定义的冲突。

4）了解 Makefile

以 make run 为例，他会首先找到 run 这个 target 对应的目标：即$(call git_commit, "run") 和$(NEMU_EXEC)，此时首先会通过 git 提交一个记录，然后在生成 NEMU_EXEC 这个项目，可以看到生成 NEMU_EXEC 需要$(BINARY) $(ARGS) $(IMG)，IMG 和 ARGS 都是编译的时候的一些参数或者是输出文件等，BINARY 就是我们的目标文件，BINARY 生成的过程中最主要的过程如下：@$(LD) -O2 -rdynamic $(SO_LDLAGS) -o $@ $^ -lSDL2 -lreadline -ldl，它就是用 gcc -O2 来生成我们的最终项目，其中 OBJS 就是我们的源文件。