实验进度：我已通过 PA2 所有 OJ 测试样例。

必答题：

程序是个状态机：

加法程序状态机用三元组（pc, r1, r2）表示，设加法参数为 a 和 b，x 表示未初始化。则状态机如下所示：

(0, x, x) -> (1, a, x) -> (2, a, b) -> (3, a+b, b)

YEMU 执行一条指令的过程：首先通过 pc 取一条指令，然后对其操作码和操作数译码，再根据其语义执行对应操作，最后更新 pc。

联系：执行指令的过程会导致状态机发生变化。

RTFSC 理解指令在 NEMU 中的执行过程：

1.通过 pc 取得一条指令。
2.识别指令类型，opcode 不同译码模式不同，将各字段以对应译码规则匹配。识别寄存器，立即数，并且对立即数进行预处理。
3.通过译码辅助函数 table 对指令进行具体匹配，部分指令需要匹配多段 code。
4.使用对应执行辅助函数，访问相应寄存器和进行相应读写操作。
5.执行完成，更新 pc 以做下一次取指令准备。

程序如何运行：

1.键盘发送通码/断码，NEMU 将其放在芯片注册的对应地址空间中。
2.AM 从 KBD_ADDR 读出相应信息，并保存在 AM_INPUT_KBYBRD 寄存器里。
3.程序读取输入信息，进行运行判断是否 hit。
4.cpu 依次按照程序逻辑取指令。

5.根据 ISA 的规定对二进制指令进行匹配，译码和执行。
6.重复上述两步直至程序返回结果，将屏幕信息存放入 AM_GPU_FBDRAW。
7.更新下一帧时，AM 通过接口将屏幕信息输出到 NEMU 注册的对应屏幕地址。

编译与链接：

两者都移除会出现重复定义的警告；去除 inline 会发生定义未使用的警告。

static 函数的访问被限制到声明他们的文件，而内联则是将函数体直接嵌入对应调用的地方，阻止了 pc 的跳转。

移除 static inline 的时候时候发生重复定义：无论 gcc 按照什么样的顺序链接，他们进入.o 文件时都会发生冲突。

移除 inline：使用 werror 时是不允许定义未使用的，但是 inline 关键词是建议编译器将这段内容嵌入它的调用点，因此不调用时这个函数相当于是不存在的，所以带有 inline 关键词的函数不会出发 werror 的警告。

编译与链接：

1.如下图所示：

```
yanahashirui@SharkWitch:~/ics2021/nemu/build$ objdump -D riscv32-nemu-interpreter >temp
yanahashirui@SharkWitch:~/ics2021/nemu/build$ find . -name "temp" | xargs grep "<dummy>"| wc -l
33
```

33 个，同样的方法可以知道不加这行代码 dummy 的实体是 0 个。

2&3.会报错：

```
In file included from ../../include/common.h:34,
                 from difftest.cc:2:
/home/yanahashirui/ics2021/nemu/include/debug.h:29:21: error: redefinition of ‘volatile int dummy’
   29 | volatile static int dummy;
      |                     ^~~~~
In file included from difftest.cc:2:
../../include/common.h:11:21: note: ‘volatile int dummy’ previously declared here
   11 | volatile static int dummy;
      |                     ^~~~~
```

common.h 和 debug.h 互相 include，而 static 会让他们重复声明，自然会报错。

了解 Makefile：

make 的工作方式如下：
1.读入 Makefile 和被 include 的其他 Makefile。
2.初始化其中的变量。
3.推到隐式规则，并分析其他规则。
4.为所有目标文件创建依赖关系链。
5.根据依赖关系，决定生成哪些目标，如何生成。
6.执行生成命令。

具体到 hello，其中的 Makefile 定义了 NAME 和 SRCS 两个变量，用于生成文件名称和指定依赖关系。然后同时 include 了 abstract-machine 的 Makefile。
通过阅读 Makefile 代码可以知道，首先利用定义的变量，创建了目标文件目录 build/riscv32-nemu 来存放.d 和.o 文件。
然后找到目标文件，这里只有 SRCS，也就是初始化为 hello.c 的变量被用来初始化 OBJS，这是需要被编译和链接的目标文件。
根据 isa 设定好编译的 flag。
用 gcc 编译.c 文件生成.o 文件（这里没有需要递归链接的.a 库）链接打包生成 img 镜像。elf 文件放在指定的 build 目录下。

选做题：

完成了 difftest 基础设施，通过正确实现的对照来调试，效率极高（尤其对实现指令和检测指令错误很友好）。