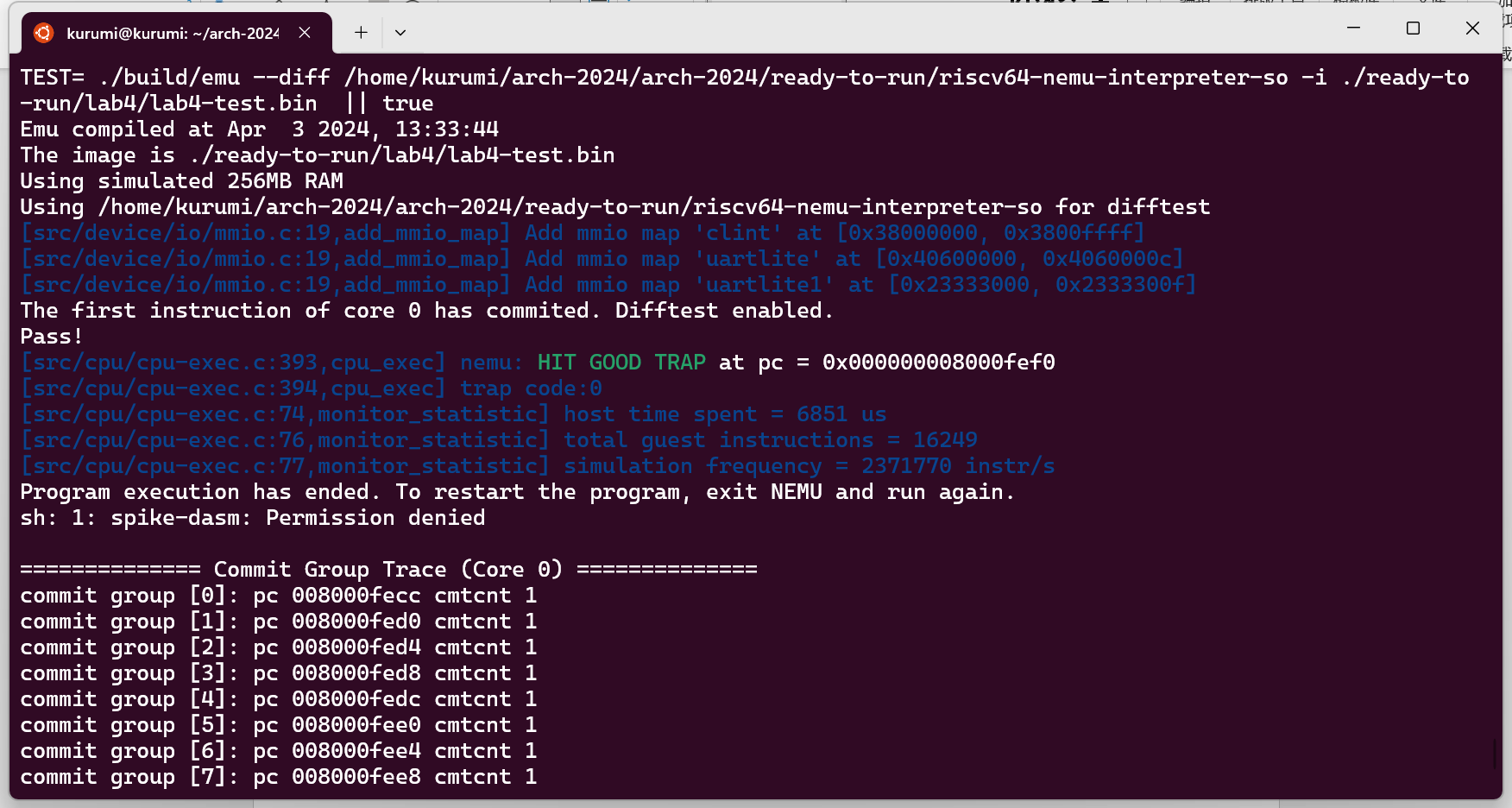
Lab4实验

实验人：谢志康

学号：22307110187

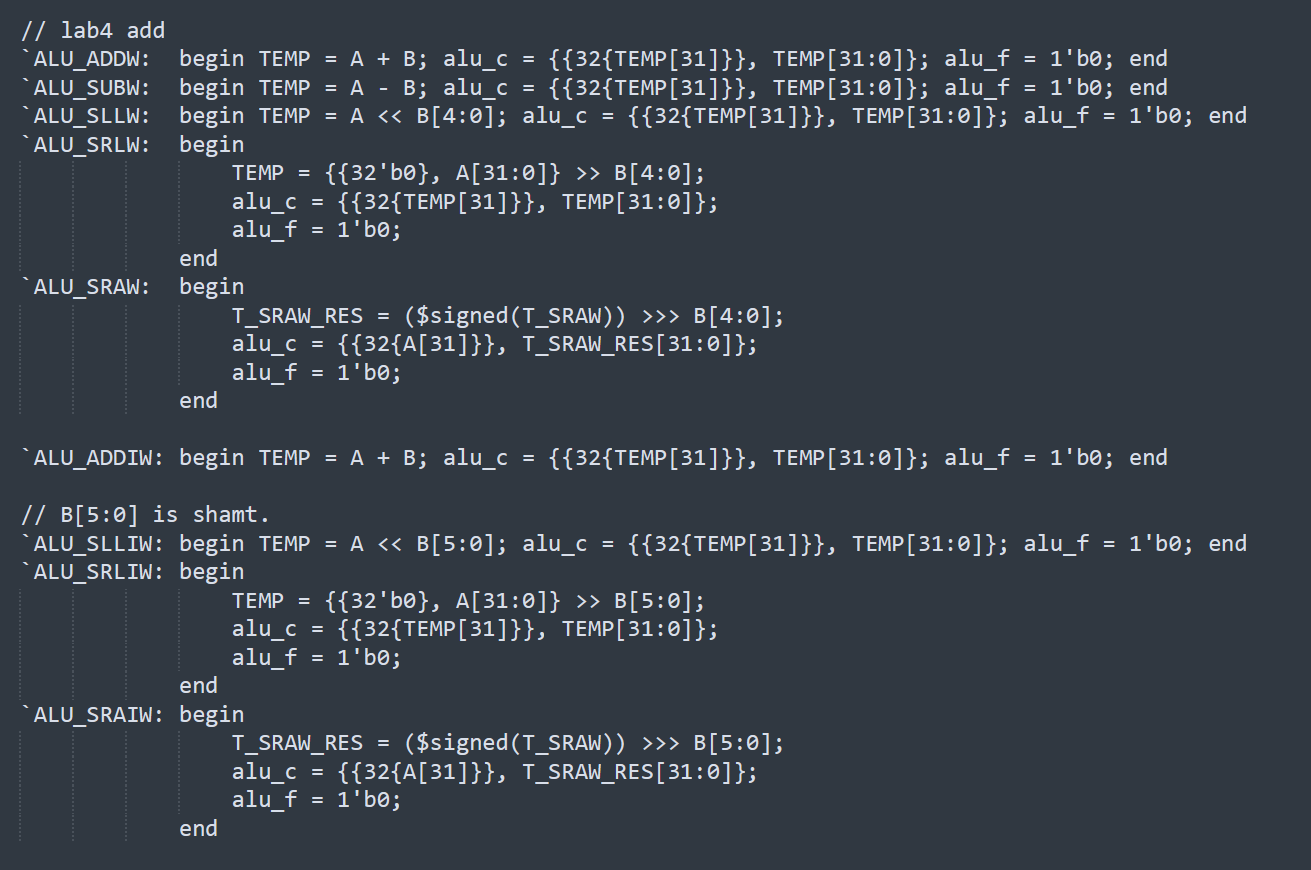
时间：2024.4.3

通过截图：



相较于上一版，添加了几条指令，大部分比较简单直接在ALU里新加几条判断即可。

这一部分不讲了，就是按照指令手册一一完成即可，就是加了以下几行代码。



稍麻烦一点的部分是多周期乘除法器的实现，但其实逻辑和我们lab2实现的访存指令差不多。在我的架构中，原本是有stall，stall\_this\_dbus，stall\_next\_ibus，用于控住CPU，执行完当前指令再往前走。在MEM的访存指令中，我就是用以下逻辑实现：没有访存完（单周期搞不完）就让stall\_this\_dbus为1（stall就为1）了，不让cpu读取下一条指令。

这里的乘除法其实也就是这个逻辑。

具体乘除法器实现上网上很多教学，如何多周期算乘除，总思路就是每周期算一点（如参考<https://blog.csdn.net/qq_44840079/article/details/104790338?spm=1001.2101.3001.6650.3&utm_medium=distribute.pc_relevant.none-task-blog-2~default~CTRLIST~default-3-104790338-blog-100743362.pc_relevant_antiscanv4&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2~default~CTRLIST~default-3-104790338-blog-100743362.pc_relevant_antiscanv4&utm_relevant_index=4>）

我这里就以乘法器举例，64位，我们分64个周期，每个周期算一位，之后移位，下一周期再算下一位（always\_ff实现，每个时钟上升沿计算一位，非阻塞赋值）最后得到结果。时钟控制的逻辑和MEM访存一样，我新开了个stall\_this\_alu\_mul，表示mul没算完将它控住（具体说来，也就是mul模块中有个count，发现是mul，count开始设为start，每周期算一位然后count++，直到等于64，算完，stall恢复，cpu才开始继续做事。

细节上也就是注意符号位并且特殊情况（除0、溢出），特判解决即可。符号上我是开了个sign模块，先把俩操作数都变成正的，要是两者同号乘除运算后就是正，vice versa。所以之后都拿正的算（sign的情况，unsign拿原操作数），再看看是否sign使能为1，是的话取反加一恢复即可。

最终完成。

（ps：之前在群里问有没有助教在if楼想去问问题是因为遇到了个冲突当时脑子没转过来不知道咋解决了，但其实很简单。就是因为，我原本有个alu，乘除我新开了俩alu分别做事，然后最后该把谁的结果写到寄存器呢？当时没判断，我以为它自己会做事，但其实会冲突，显然，alu没算乘除结果是0，alu\_mul算乘法是一个结果，该把谁写回去，并不是自动把非零写回去，所以会有冲突。当时觉得写的很对了不知道为啥有一个使能一直为0，看波形图发现没到除法那64个周期就开始了，百思不得其解。后面想清楚是这个问题，新加了一个mux模块决定最终alu计算结果就可以过了）