1. 检查运行环境：

我们提供的实验工程是一个完整的工程项目，可以通过测试程序，大概需要五分钟。但是发现部分同学的实验环境出现了问题，所以建议同学们先检查一下运行环境；

如何检查?

直接打开工程项目，不用进行任何操作，执行仿真即可；（第一次可能需要较长时间，因为需要综合各IP核）。仿真时间大概五分钟左右。

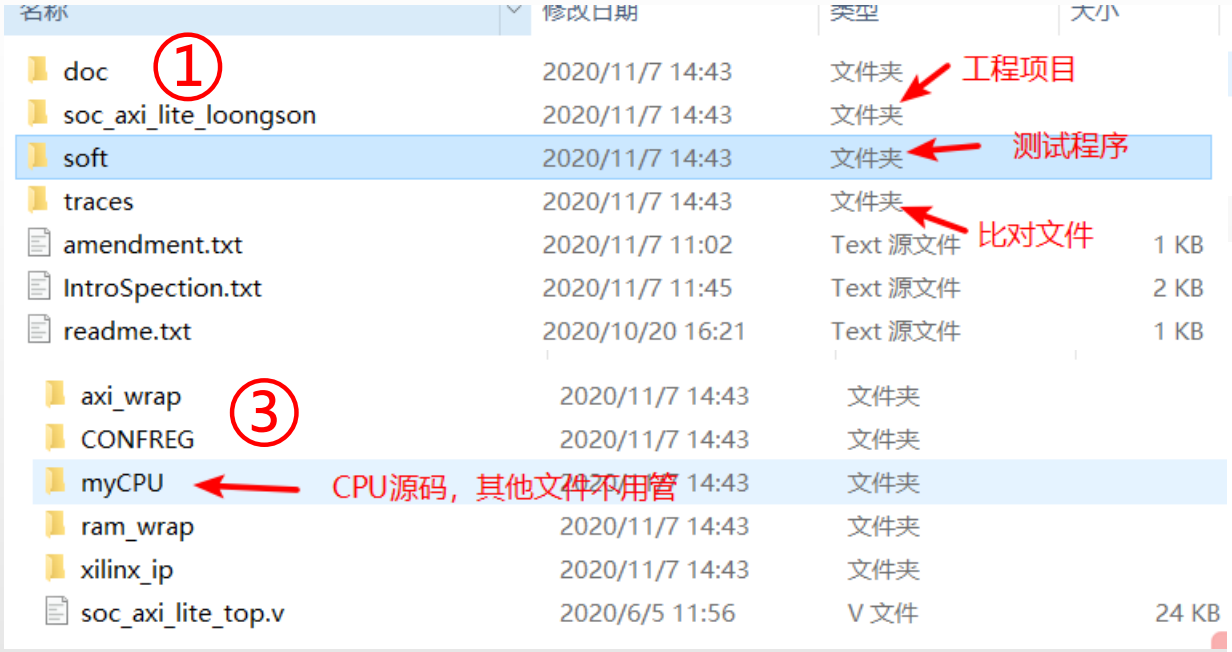
如果TCL控制台输出以下所有信息，则表示运行环境没有问题；

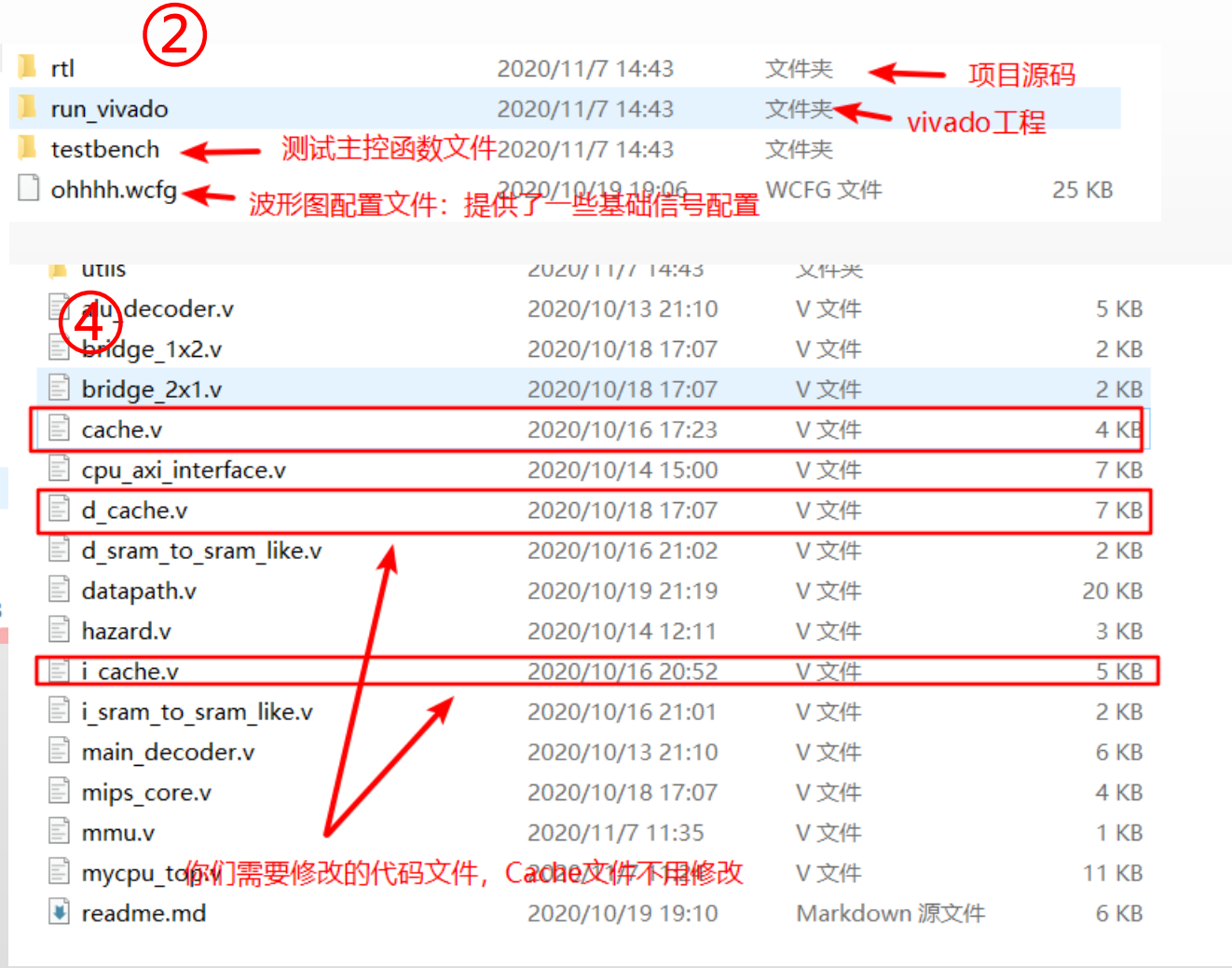
否则，参考之前各助教上传的《实验2运行环境解决办法》解决实验环境问题。

```
run all
================================================================
Test begin!
shell1 test begin.

dgemm PASS!

Total Count(SoC count) = 0x351b4

Total Count(CPU count) = 0x18292

================================================================
Test end!
----PASS!!!
```

体系结构实验2--Cache设计与实现

2. 理解项目框架；阅读理解示例代码





阅读理解示例代码：参考指导书示例代码部分

体系结构实验2--Cache设计与实现

3. 阅读理解示例代码

知识储备：

1. 类sram接口协议：　　了解cache是如何处理请求跟发送请求的。
2. cache基础知识：　　写回，组相联等等。
3. 状态机？要掌握一下；

实验代码要求的逻辑：

1. 类sram接口逻辑：了解类sram接口是如何通信的；
2. cache内部处理逻辑：主要是读命中，读缺失，写缺失，写命中四种情况下的处理逻辑

体系结构实验2--Cache设计与实现

3. 阅读理解示例代码

知识储备：

1. 类sram接口协议：　　了解cache是如何处理请求跟发送请求的。
2. cache基础知识：　　写回，组相联等等。
3. 状态机？要掌握一下；

实验代码要求的逻辑：

1. 类sram接口逻辑：了解类sram接口是如何通信的；
2. cache内部处理逻辑：主要是读命中，读缺失，写缺失，写命中四种情况下的处理逻辑

3. 阅读理解示例代码

T1：学姐“告诉”我她要杯子

----a_req = 1;

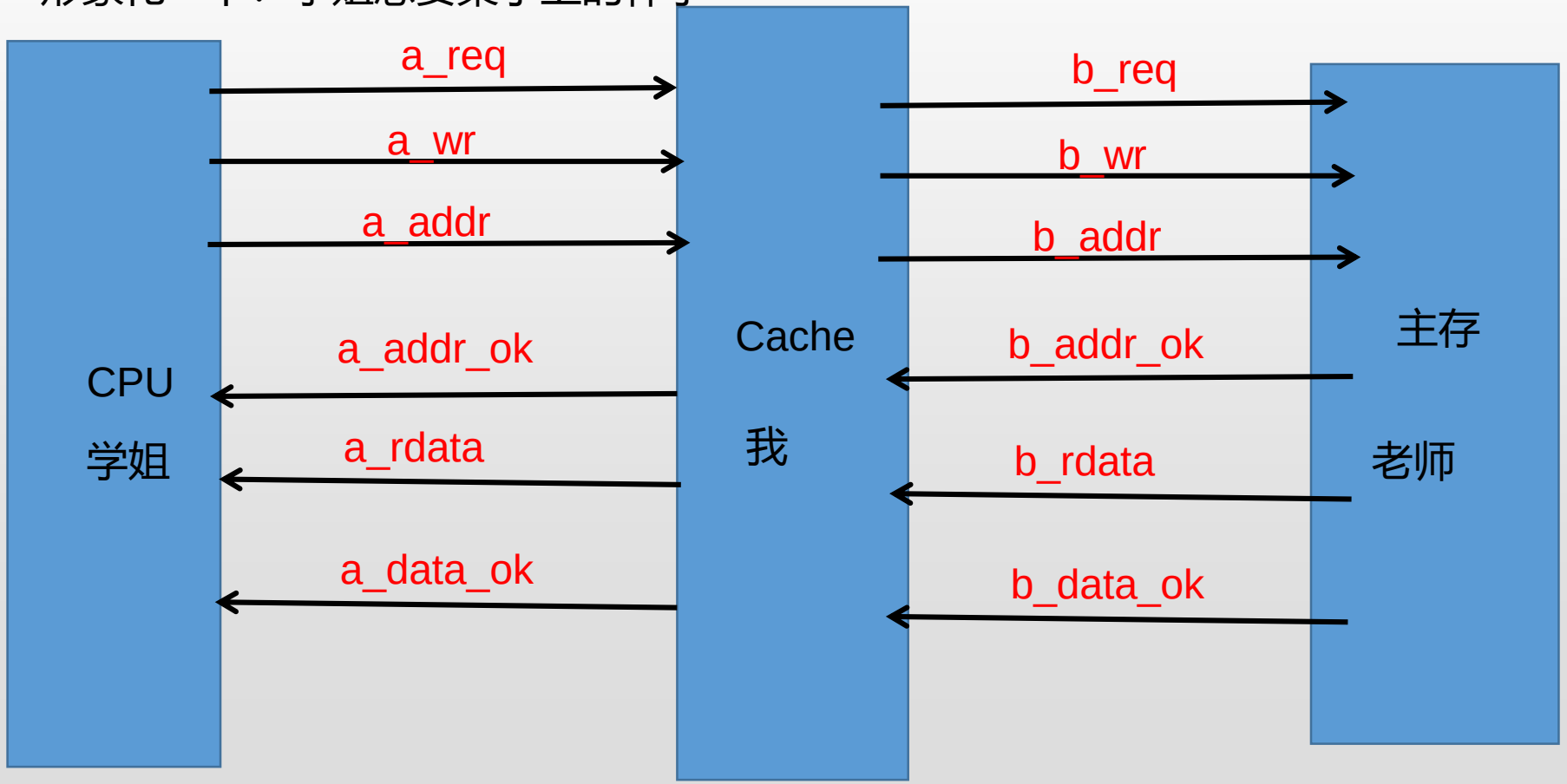
a_wr =0 ;

a_addr = 1;

类sram接口

示例场景：Mips-Core想要内存地址为0的数据

形象化一下：学姐想要桌子上的杯子



信号	电平
a_req	1
a_wr	0
a_addr	1
a_addr_ok	0
a_rdata	0
a_data_ok	0

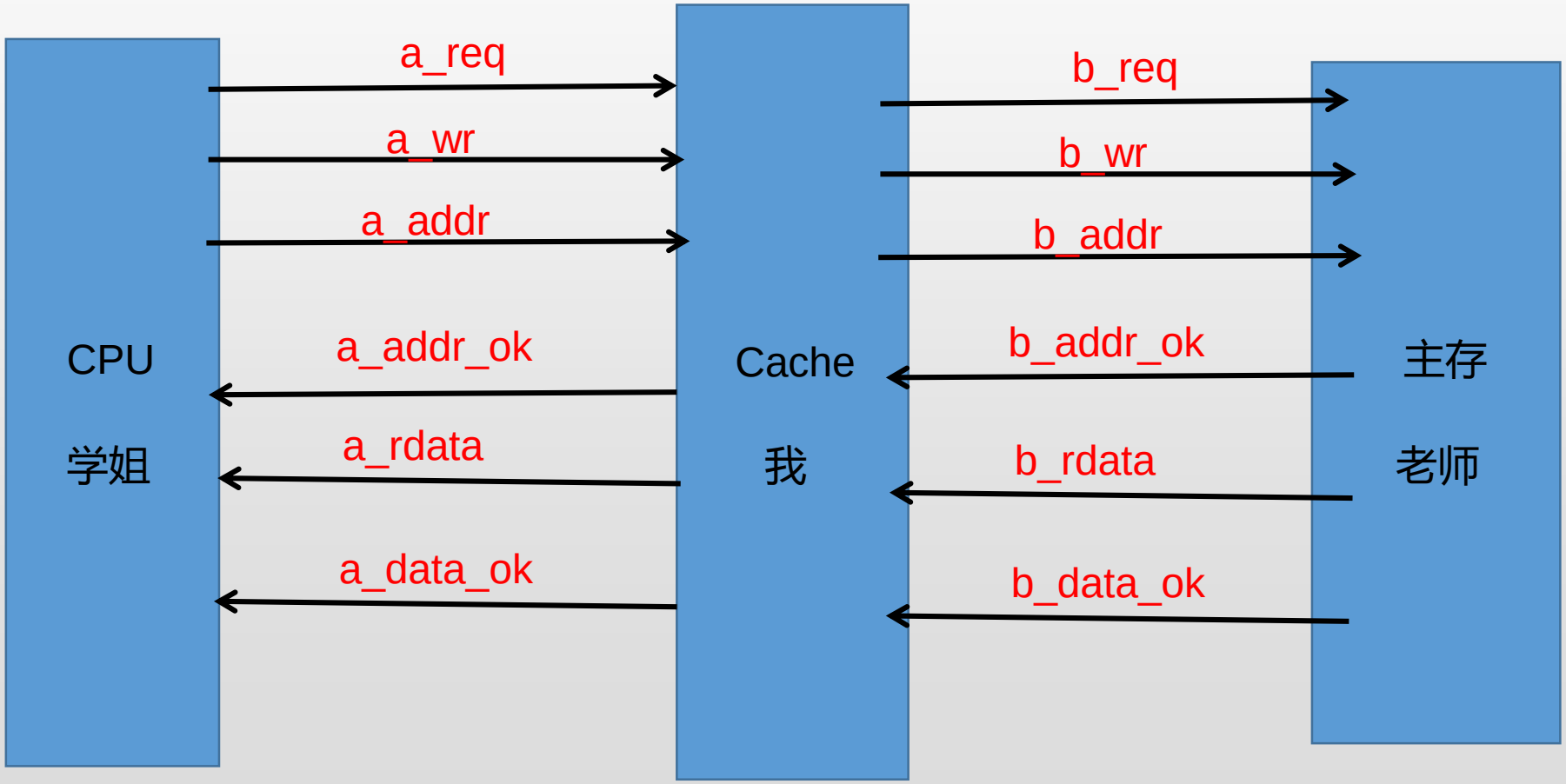
示例场景：Mips-Core想要内存地址为0的数据
形象化一下：学姐想要桌子上的杯子

第一种情况：我有杯子

T2：
我听到了学姐的请求，回复“收到”
-----将a_addr_ok 置为 1；
然后发现我有杯子，然后把杯子给她，并说给你了
-----赋值 a_rdata；并将 a_data_ok 置 1

信号	电平
a_req	1
a_wr	0
a_addr	1
a_addr_ok	1
a_rdata	杯子
a_data_ok	1

这是读命中



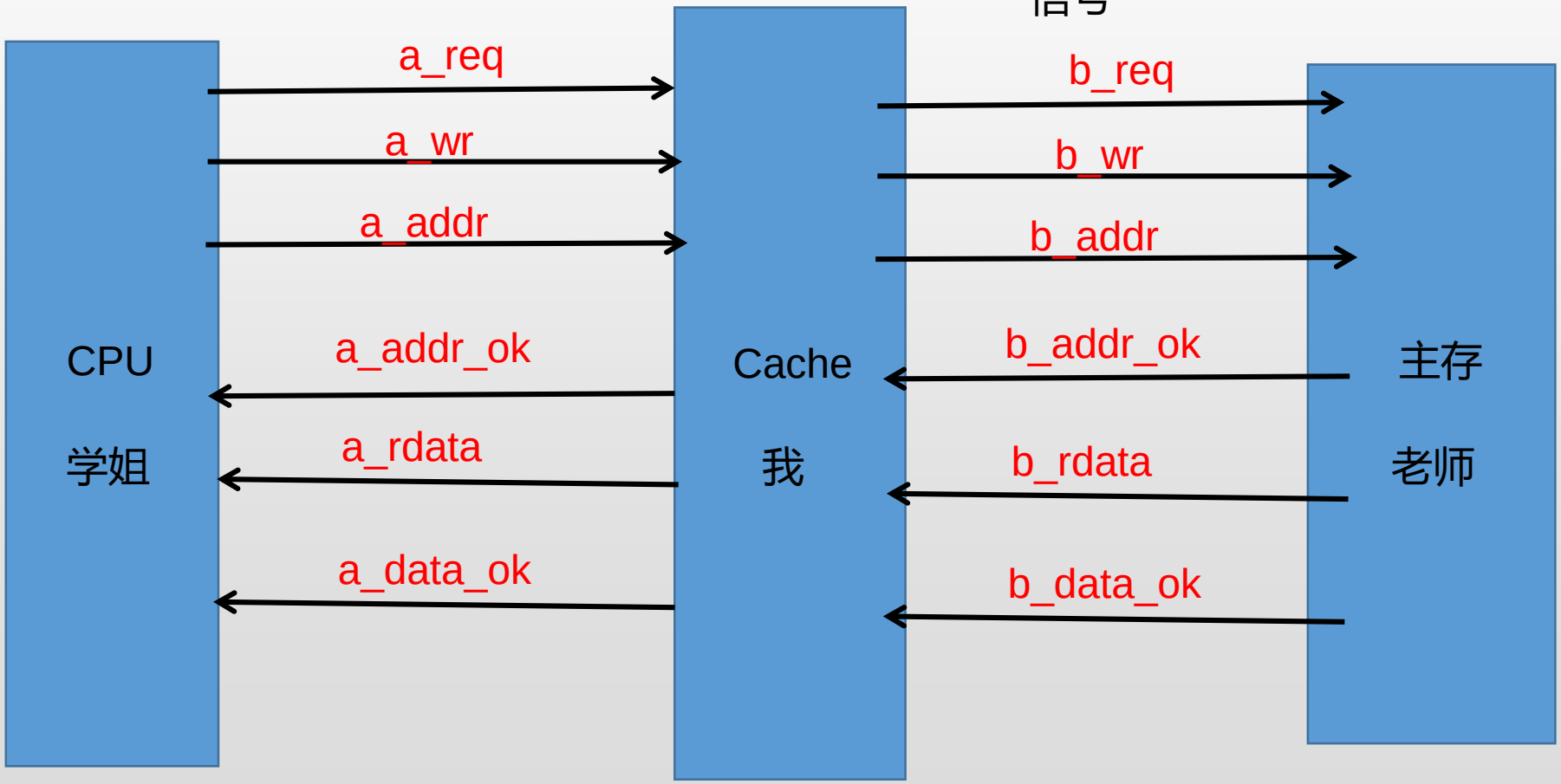
示例场景：Mips-Core想要内存地址为0的数据
形象化一下：学姐想要桌子上的杯子

第一种情况：我有杯子

T3：
学姐听到我回复后，不再发请求；
-----a_req = 0;
然后拿到杯子，初始化她的信号；
-----a_wr = 0;
a_addr = 0;
我在给出杯子后，也初始化我得信号

信号	电平
a_req	0
a_wr	0
a_addr	0
a_addr_ok	0
a_rdata	0
a_data_ok	0

这是读命中



体系结构实验2--Cache设计与实现

4. 设计cache，实现代码

完成指定的实验要求；最低要求实现写回的cache。具体信息参考实验指导书--提升cache性能部分；
顺便提醒一句，如果只实现写回的cache，那么仅仅需要修改d_cache文件即可。

4. 调试

调试机制：trace比对调试；
使用正确的工程项目运行测试程序，并将要修改通用寄存器的指令的执行情况写入到trace文件中。然后运行自己实现的工程项目，当发现当前指令要修改通用寄存器的时候，将该指令的执行情况跟trace文件中记录的执行情况作比对。如果比对结果一样，CPU继续运行，如果不一样，则将错误相关信息输出到TCL中。

trace文件解释：

1. 第一行：访存阶段的写寄存器使能；
2. 第二行：指令的地址；
3. 第三行：该指令写哪个寄存器；
4. 第四行：该指令写入寄存器的数据；