Project 4 实验报告

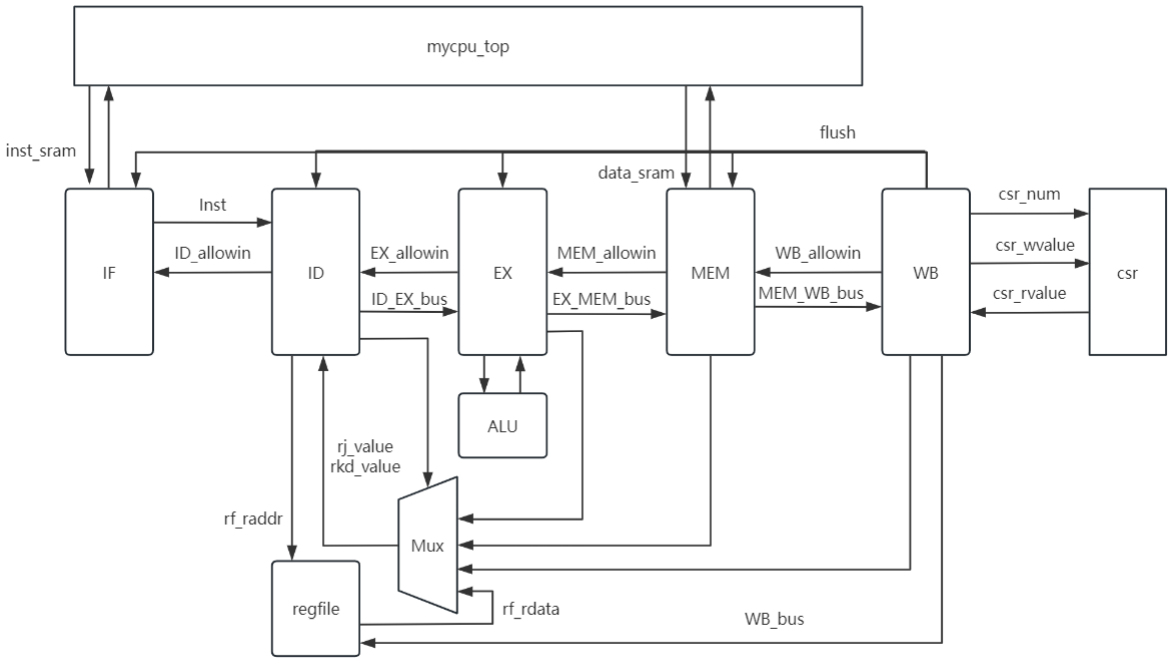
钱翰林 周远扬 石曜铭

11/11/2025

1 小组分工情况说明

- 钱翰林: CSR 内部实现，调试
- 周远扬: 中断，CSR 指令译码与实现
- 石曜铭: CSR 指令的实现，调试流水线

2 处理器结构设计图



3 主要设计点

3.1 CSR 内部实现

首先按照指令手册 7.1 节控制状态寄存器一览表去定义 `csr_num`，并对照手册对各个控制状态寄存器进行分段。然后阅读教材，参考书上的代码完成 CSR 各个域的更新逻辑，其中 ECFG、EENTRY、SAVE0 ~ 3、TID、TCFG 仅会被 CSR 读写指令更新；PRMD、ERA、BRA 在触发异常时也会被更新；CRMD 在触发异常和 ertn 指令执行时也会被更新；ESTAT 的各个域的更新逻辑存在区别，需要一一设计；定时器计数器 TVAL 在软件开启 timer 的使能时将 `csr_ tcfg_initval` 更新到 `timer_cnt` 中，软件关闭 timer 的使能时，`timer_cnt` 不更新。最后每个 CSR 重新拼合出一个 32 位宽的值，根据 `csr_num` 进行选择并返回。

3.2 CSR 相关指令

本实验在流水线中加入了 CSR 相关指令的支持，包括 csrrd、csrwr 和 csrxchg，不同指令对应不同的信号，传入 csr 分别完成相关读写操作，并且在指令期间阻塞，保证异常安全。

3.3 各异常与中断

在实验中，完成了取指地址错（ADEF）、地址非对齐（ALE）、断点（BRK）、系统调用（SYS）和指令不存在（INE）异常以及各种中断。对于异常，分别来自流水线的各个阶段或者指令，将异常信号在 WB 阶段传给整个流水线，并且拉低各阶段的 valid，重新在 IF 取新指令完成异常处理，另外实现 ertn 指令完成异常恢复，流程类似。

4 调试

4.1 误将复位后的全零指令视为不存在

解决方案：对 IF 阶段添加 `if_to_id_valid` 信号，只有当信号拉高时，才会启动对指令不存在的判断。

4.2 EX 阶段状态转移不完备

- 原先对 EX 阶段状态转移的设计中，默认了 ID 阶段是单周期，当添加 CSR 指令后，ID 阶段可能会有阻塞，导致与 EX 阶段期望的状态不符。

  解决方案：在 EX 的状态机中添加与 ID 的握手信号（在代码实现中表现为 EX 状态的 *valid* 信号），仅当 *valid* 拉高时才进行状态的转移。

- 没考虑到 EX 在中间两个等待状态的过程中，被 $flush$ 的情况。

  解决方案：在被 $flush$ 后统一转移到 *readygo* 状态，这是符合逻辑的。

4.3 CSR 相关指令引起的数据冲突解决不完善

在 rdcntid, rdcntvl.w, rdcntvh.w 指令中，rdcntid 需要读 TID，而结果在 WB 才能得到，所以也不能通过前递解决，需要在 ID 阶段阻塞。对于 rdcntvl.w, rdcntvh.w 指令，结果在 EX 中得到，因此可以添加进 EX 阶段的 `compute_result` 中，跟随原有的数据前递即可。

我们注意到，在通过 exp11 的代码中，EX 阶段的前递数据直接连在了 `alu_result` 而不是更完全的 `compute_result` 中，理论上，如果有一条乘除法指令与下一条指令有数据冲突，我们之前设计的处理器行为是不正确的，因为并没有把结果前递。但这仍然能通过 exp11 的测试，可能是测试中不包含针对这种情形的样例所致。