**流水线CPU设计**

李卓 pb19000064

**实验目的**

理解流水线CPU的结构和工作原理

掌握流水线CPU的设计和调试方法，特别是流水线中数据相关和控制相关的处理

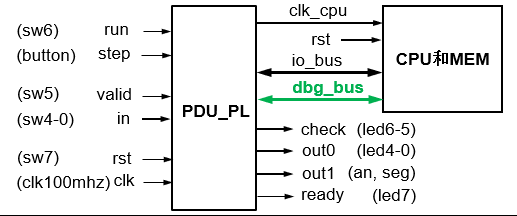
熟练掌握数据通路和控制器的设计和描述方法

**实验原理**

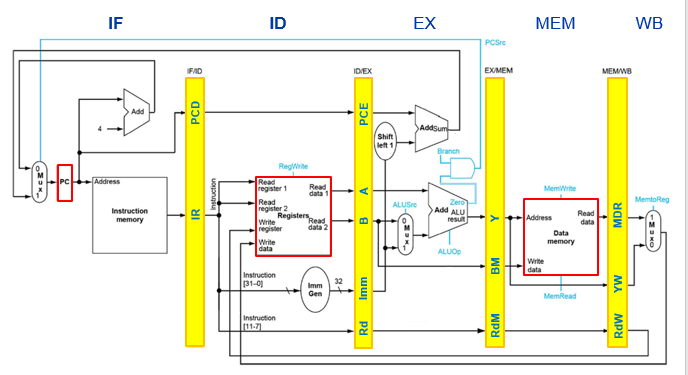
**设计实现5级流水线的RISC-V CPU**

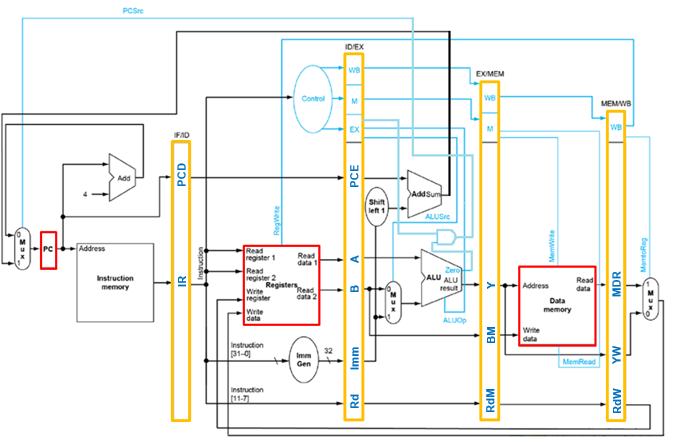
能够执行6条指令：**add, addi, lw, sw, beq, jal**

指令存储器和数据存储器均使用分布式存储器，容量均为256x32位，数据存储器地址为0x0000\_0000 ~ 0x0000\_2ffff，指令存储器地址为0x0000\_3000 ~ 0x0000\_3ffc。



流水线CPU数据通路

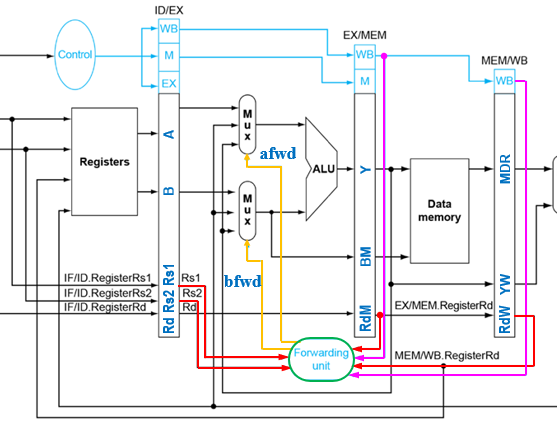




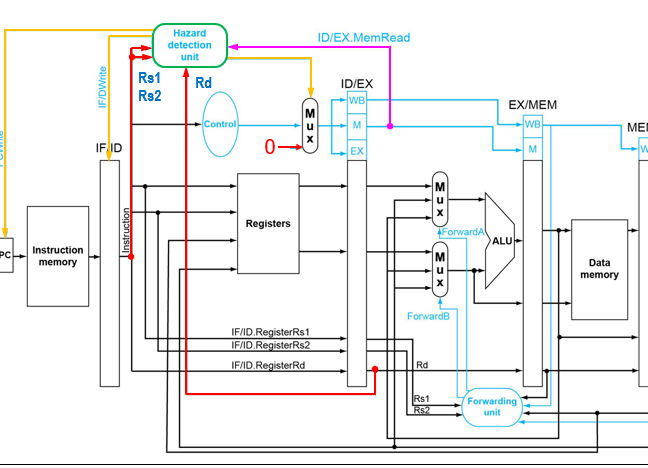
流水线相关及其处理

* 结构相关：当多条指令执行时竞争使用同一资源时
  + 存储器相关处理：哈佛结构（指令和数据存储器分开）
  + 寄存器堆相关处理：同一寄存器读写时，写优先（Write First）
* 数据相关：当一条指令需要等待前面指令的执行结果时
  + 数据定向（Forwarding）：将执行结果提前传递至之前流水段
  + 加载-使用相关（Load-use hazard）：阻止紧随Load已进入流水线的指令流动（Stall），向后续流水段插入空操作（Bubble）
* 控制相关：当遇到转移指令且不能继续顺序执行时
  + 清除（Flush）紧随转移指令已进入流水线的指令 从转移目标处取指令后执行

数据定向（Forwarding）：



load-use hazard:



io\_bus信号

CPU运行时访问开关(sw)、指示灯(led)和数码管(an, seg）

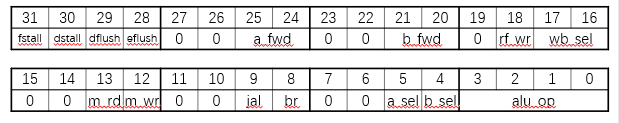
* + io\_addr：I/O外设的地址
  + io\_din：CPU接收来自输入缓冲寄存器（IBR）的sw输入数据
  + io\_dout：CPU向led和seg输出的数据
  + io\_we：CPU向led和seg输出时的使能信号，利用该信号将io\_dout存入输出缓冲寄存器（OBR），再经数码管显示电路将其显示在数码管（an，seg）

debug\_bus信号

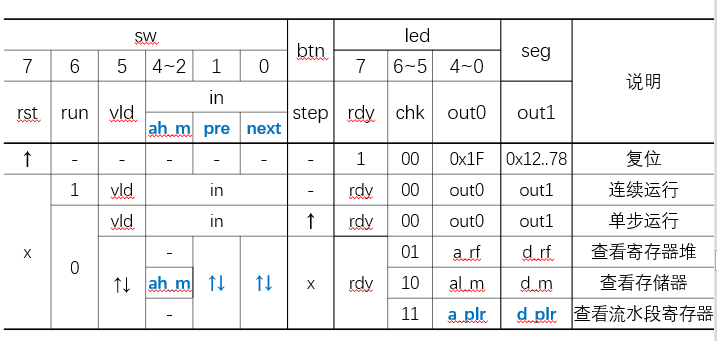
调试时将存储器和寄存器堆内容，以及CPU数据通路状态信息导出显示

* + m\_rf\_addr：存储器(MEM)或寄存器堆(RF)的调试读口地址
  + rf\_data：从RF读取的数据
  + m\_data：从MEM读取的数据
  + 流水段寄存器
    - PC/IF/ID：pcin, pc, pcd, ir
    - ID/EX：pce, a, b, imm, rd, ctrl
    - EX/MEM：y, bm, rdm, ctrlm
    - MEM/WB：yw, mdr, rdw, ctrlw

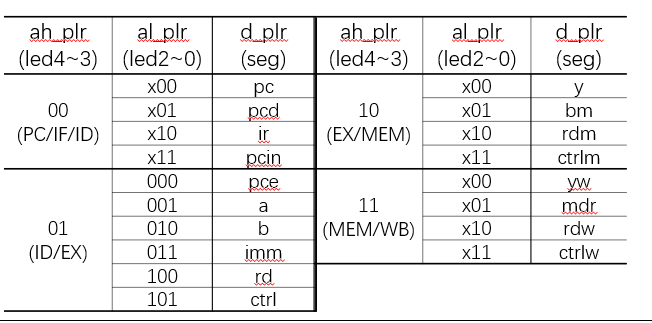
ctrl 信号内容：



pdu 运行方式：

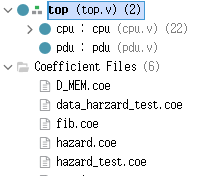


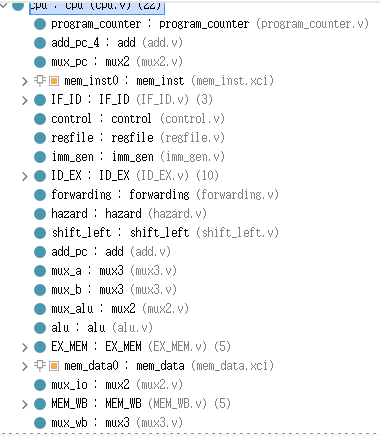
查看流水段基寄存器：

利用vld (sw5)切换chk (led6~5) = 11。使用pre (sw1)和next (sw0)分别修改流水段寄存器地址ah\_plr和al\_plr， out1 (seg)显示对应寄存器数据(d\_plr)  


**实验过程**

源文件结构：





hazard模块

fstall 阻塞pc

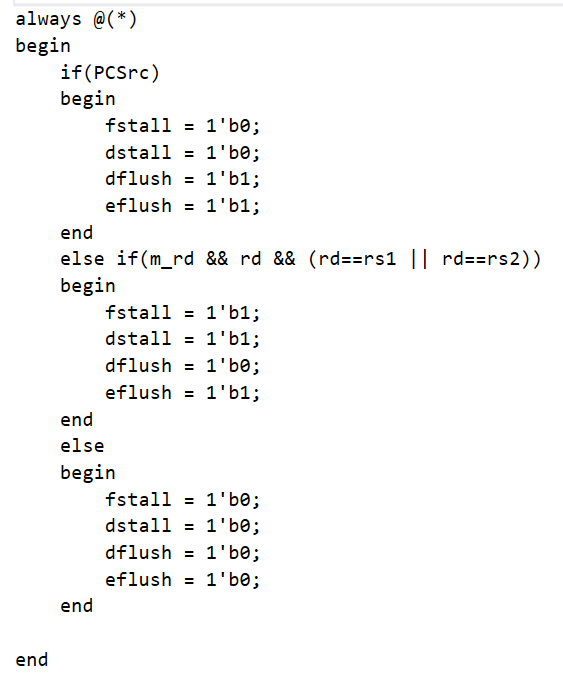
dstall 阻塞 if-id流水段

dflush 清空 if-id流水段

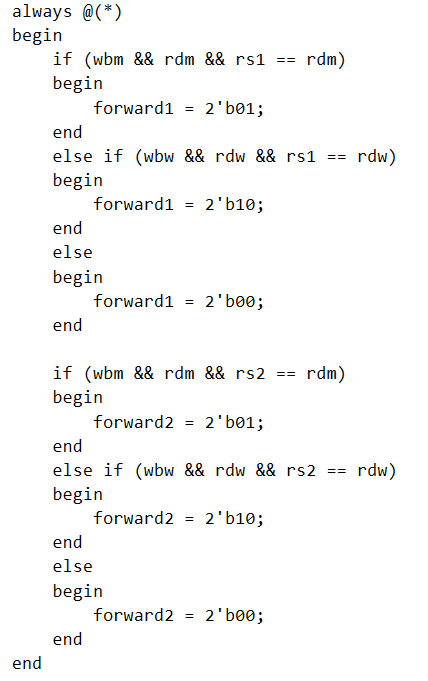
eflush 清空 id-ex流水段

当pcsrc为一时，代表即将发生指令跳转，jal 和beq都是在ex段判断完成，需要清空 提前进入if-id流水段id-ex流水段 的指令

当 m\_rd等于一且 rd 与rs相等时，代表发生load-use hazard，阻塞pc和if-id流水段,并清空 即将ex的指令



forwarding模块：

  
 forward1控制alu\_a所在的mux， forward2控制alu\_b所在的mux, forward为00时正常读入，01时选择从mem段来的数据， 10时选择从wb段来的数据

**实验代码**

所有源文件和测试文件都在仓库中

https://gitee.com/zhuoli628/cod-lab5.git