

**本科实验报告**

|  |  |
| --- | --- |
| 课程名称： | 计算机体系结构 |
| 姓 名： | 展翼飞 |
| 学 院： | 计算机 |
| 专 业： | 计算机科学与技术 |
| 学 号： | 3190102196 |
| 指导教师： | 王总辉 |

2023 年 10 月 20 日

**浙江大学实验报告**

课程名称： 计算机体系结构 实验类型： 综合

实验项目名称： Lab1：Pipelined CPU supporting RISC-V RV32I Instructions

学生姓名： 展翼飞 专业： 计算机科学与技术 学号： 3190102196

同组学生姓名： 指导老师： 王总辉

实验地点： 曹西301 实验日期： 2023 年 10 月18 日

1. **实验目的和要求**
   1. **实验目的**

• 理解RISC-V RV32I指令

• 掌握执行RV32I指令的流水线CPU的设计方法

• 掌握Pipeline Forwarding Detection和bypass unit设计的方法

• 掌握Predict-not-taken分支设计的1周期停顿的方法

• 掌握执行RV32I指令的流水线CPU程序验证的方法

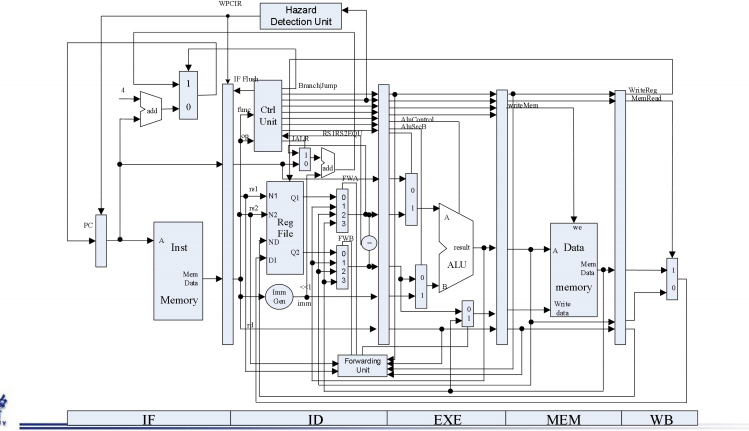
* 1. **实验要求**

设计RV32I指令的流水线CPU

包含数据路径设计，Bypass单元设计和CPU控制器设计。

1. **实验内容和原理**

实现模块： Hazard detection， rv32i core（主要为多路选择器）， cpu controller

Hazard原理：

**1.结构冒险**

\* 原因：在WB阶段写寄存器时同时有ID阶段的指令读寄存器

\* 解决方式：寄存器在时钟上边沿写，下边沿读

**2.数据冒险**

\* 原因：当前指令需要写回寄存器的结果在WB阶段前前就要被之后的指令读相应寄存器

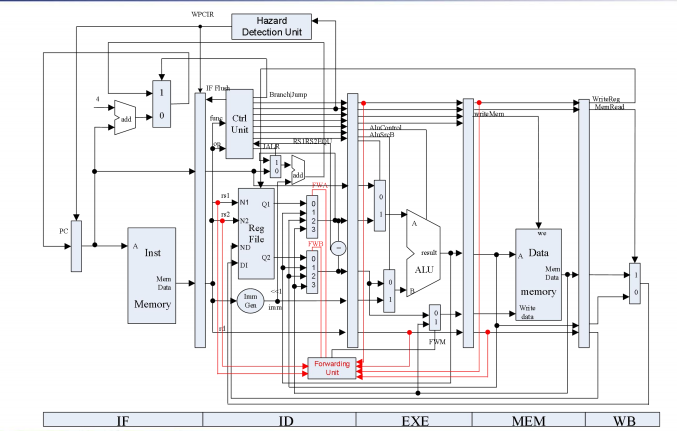
\* 解决方法；

\* R | I | J 或LUI, AUIPC 后读相应寄存器，则从EXE或MEM阶段的ALU结果forward到ID（对应forward control unit控制ID多路选择器的线1 2）

\* L类型指令后紧接S类型指令读相应寄存器，则从MEM阶段的取回值forward到EXE（对应EXE阶段下方准备传给MEM WriteData的多路选择器线1）

\* L类型指令一周期后接读相应寄存器指令，则从MEM阶段的取回值forward到ID（对应forward control unit控制ID多路选择器的线3）

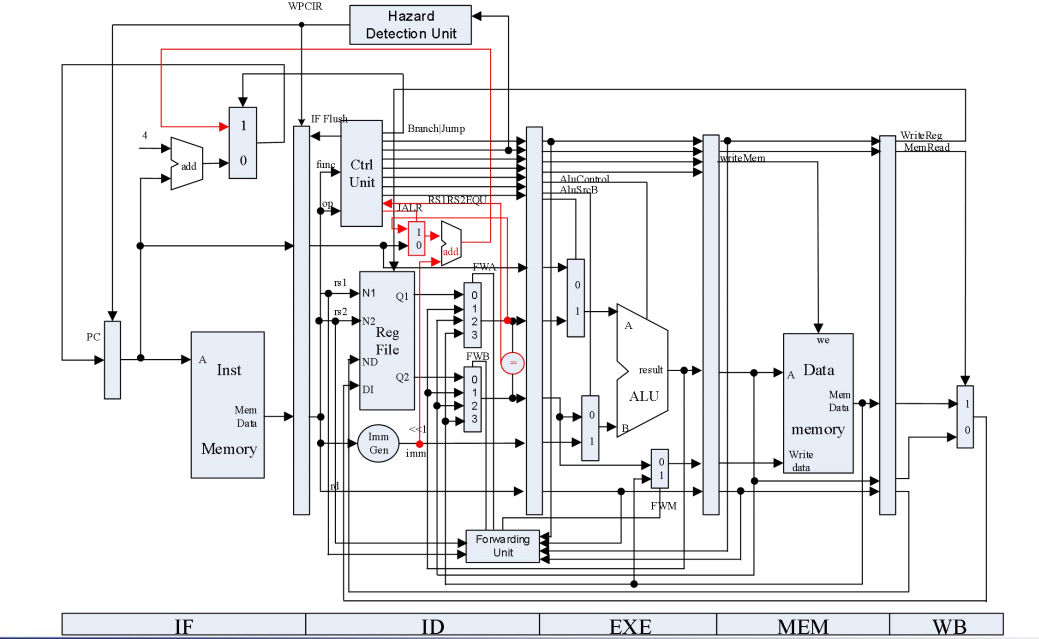
\* L类型指令紧接R | I | J或LUI，AUIPC后读相应寄存器，无法forward（当前指令load时下一条已经在exe），需要Flush ID-EXE并Stall之前的寄存器（L指令在EXE阶段时判断）



1. **控制冒险**

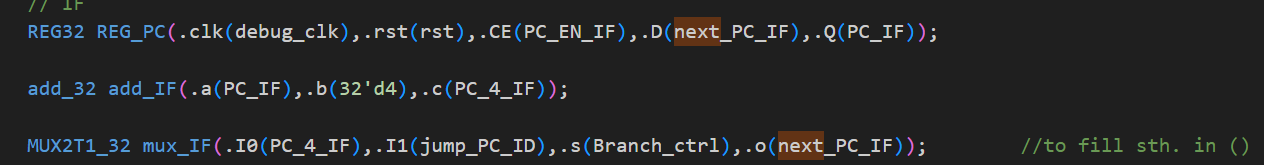
策略：Predict branch taken

ID阶段在检测到跳转指令后由Hazard Control Unit输出控制信号，Flush掉IF-ID寄存器的一个cycle使一个周期后IF取到的地址为ID阶段当前时钟末上边沿产出的需要跳转正确地址

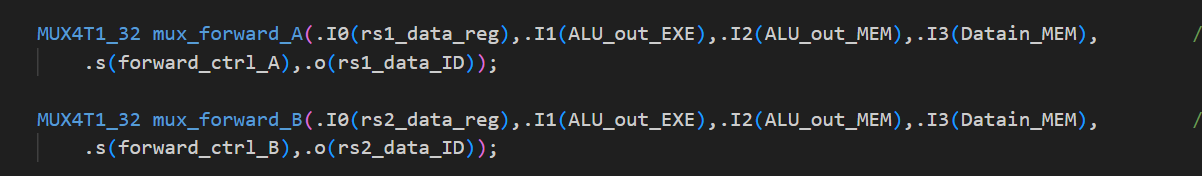


1. 重要部分代码和分析（主要介绍**你写的**代码的逻辑，贴代码+文字描述）
2. **RV32core模块**

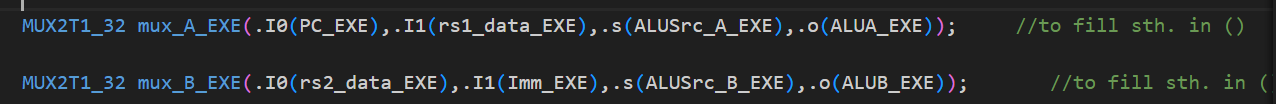
主要为多路选择器的线路填写，理解信号名，各个模块代表的含义，按照流水线CPU线路原理图填写即可



分析：IF阶段取指令地址的多路选择器，输入PC+4与ID阶段得到的跳转地址以及分支控制信号，输出下一条指令的地址



分析：ID阶段forward unit控制的两个多路选择器，输入寄存器读取内容与三种forward信息与控制信号，输出选择器结果，三种forward分别是EXE阶段ALU结果，MEM阶段的ALU结果和MEM阶段的内存读取结果



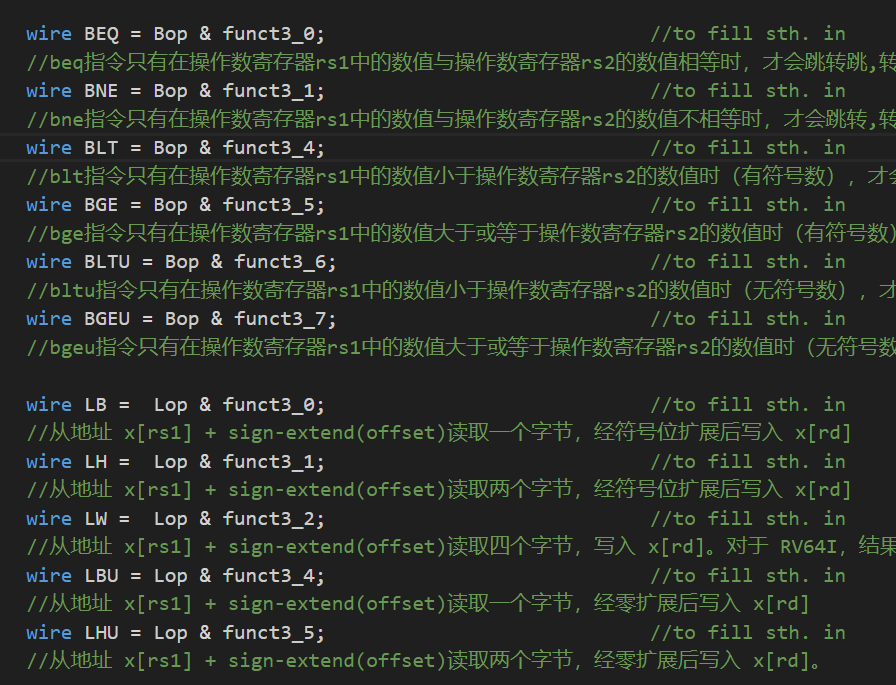
分析：EXE阶段控制ALU输入的两个多路选择器，控制信号由Control unit产生并通过阶段寄存器传递到ALUA\_EXE与ALUB\_EXE



分析：控制load后紧接store指令的forward

1. **ControlUnit 模块**

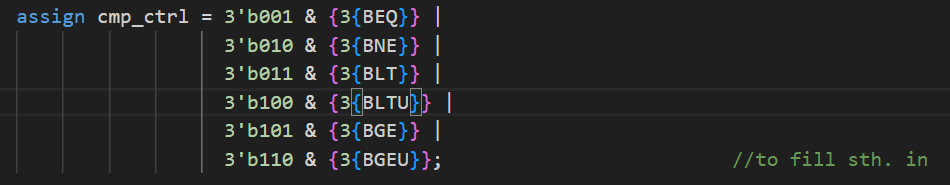
需要根据ID阶段的32位指令读取相应信息，分析指令类型，获取立即数，生成跳转，ALU前多路选择器，冒险类型等控制信号



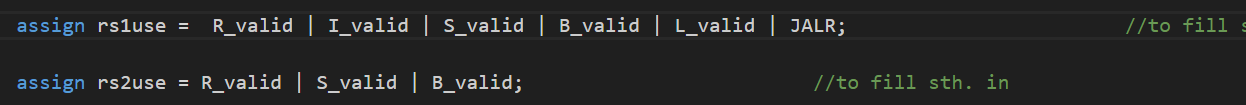
分析：这部分位指令种类读取，根据它们所属类型与function3，opcode等信号即可判断



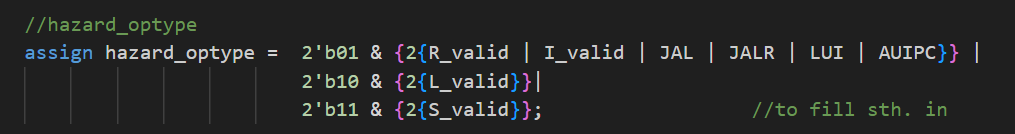
分析：跳转控制信号，为B类型指令加上JAL,JALR



分析：compare控制信号，与compare Unit对应即可



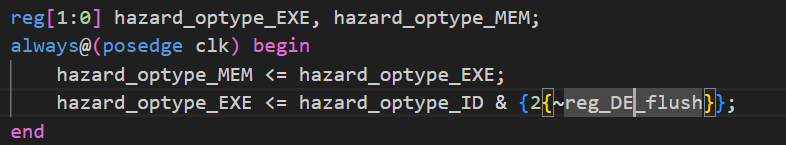
分析：当前指令两条寄存器读取线路使用信号



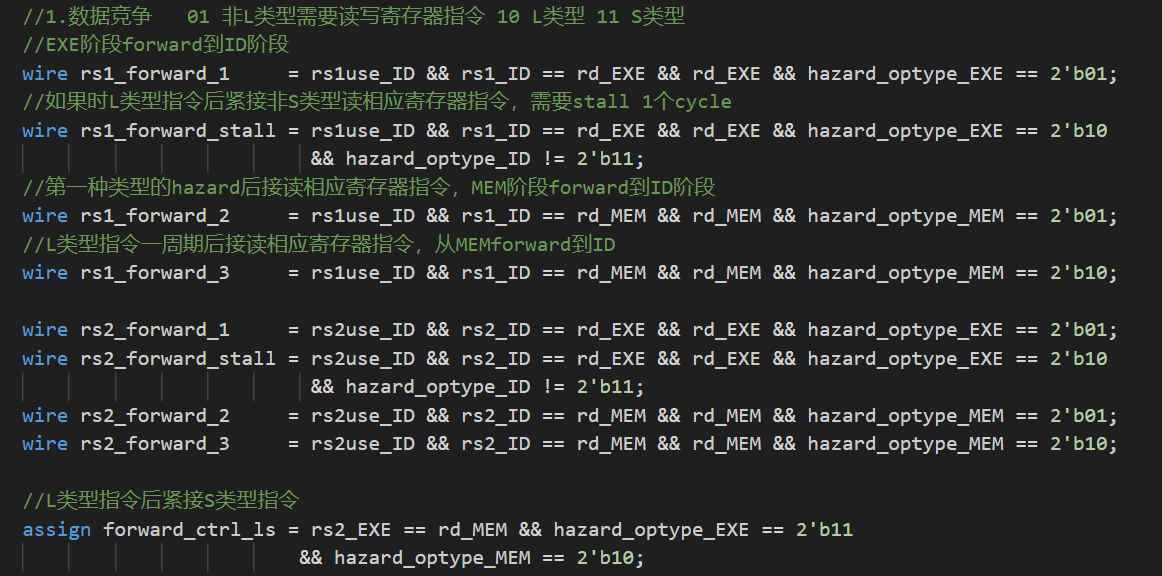
分析：Hazard类型控制信号，01意为非L或S类型的指令，10意为L类型指令，11意为S类型指令，后续输入HazardDetection模块，便于辨别冒险类型，生成冒险相关控制信号

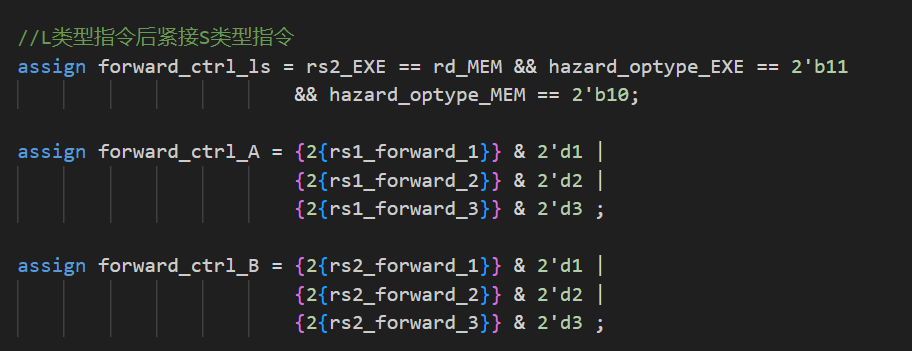
1. **HazardDetection 模块**

通过输入各阶段对寄存器的读写情况，判断冒险类型，输出各阶段寄存器使能与Flush信号



分析：通过寄存器将ID阶段产生的hazard\_optype传递并保留到EXE, MEM阶段，便于判断冒险类型





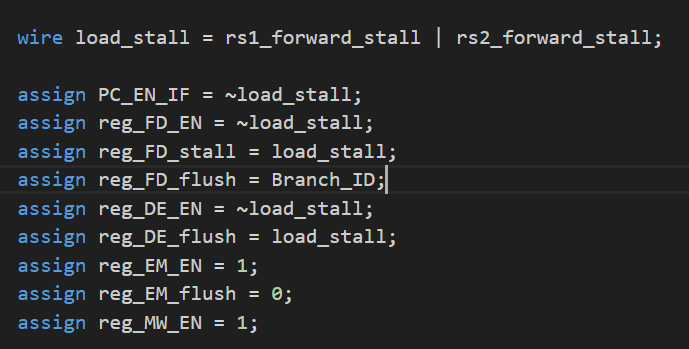
分析：通过ID阶段寄存器读取情况与后续阶段寄存器写入情况判断forward类型，生成forward与stall的控制信号

R | I | J 或LUI, AUIPC 后读相应寄存器，则从EXE或MEM阶段的ALU结果forward到ID（对应forward control unit控制ID多路选择器的线1 2）

L类型指令后紧接S类型指令读相应寄存器，则从MEM阶段的取回值forward到EXE（对应EXE阶段下方准备传给MEM WriteData的多路选择器线1）

L类型指令一周期后接读相应寄存器指令，则从MEM阶段的取回值forward到ID（对应forward control unit控制ID多路选择器的线3）

L类型指令紧接R | I | J或LUI，AUIPC后读相应寄存器，无法forward（当前指令load时下一条已经在exe），需要Flush ID-EXE并Stall之前的寄存器（L指令在EXE阶段时判断）



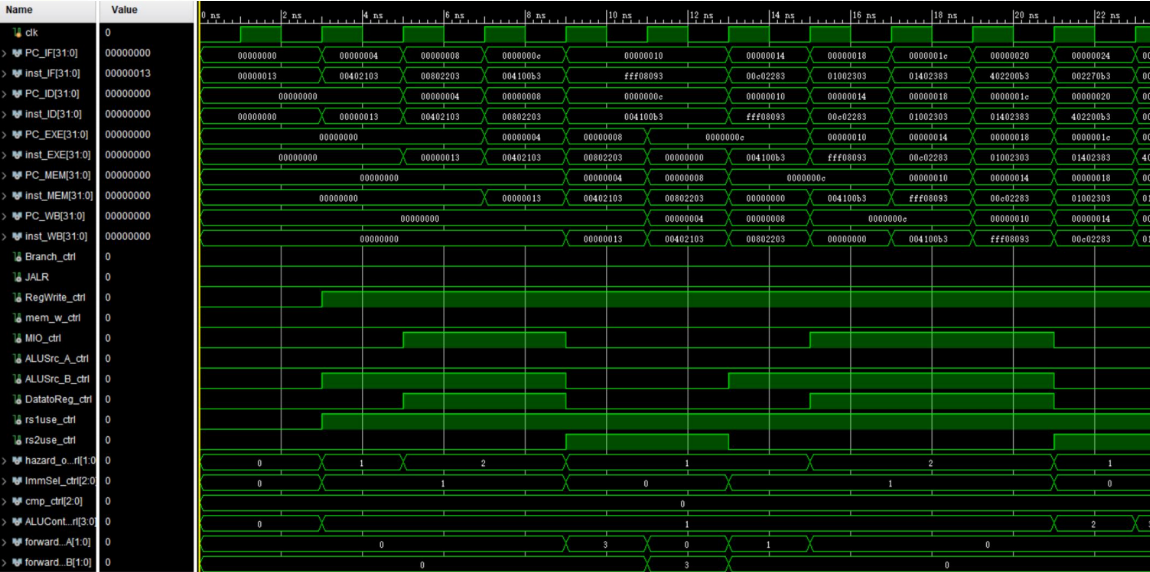
通过forward stall与ID阶段的跳转控制信号得到各阶段寄存器是否需要stall或flush。只有跳转和分支外仅有load后紧接store指令后需要stall。

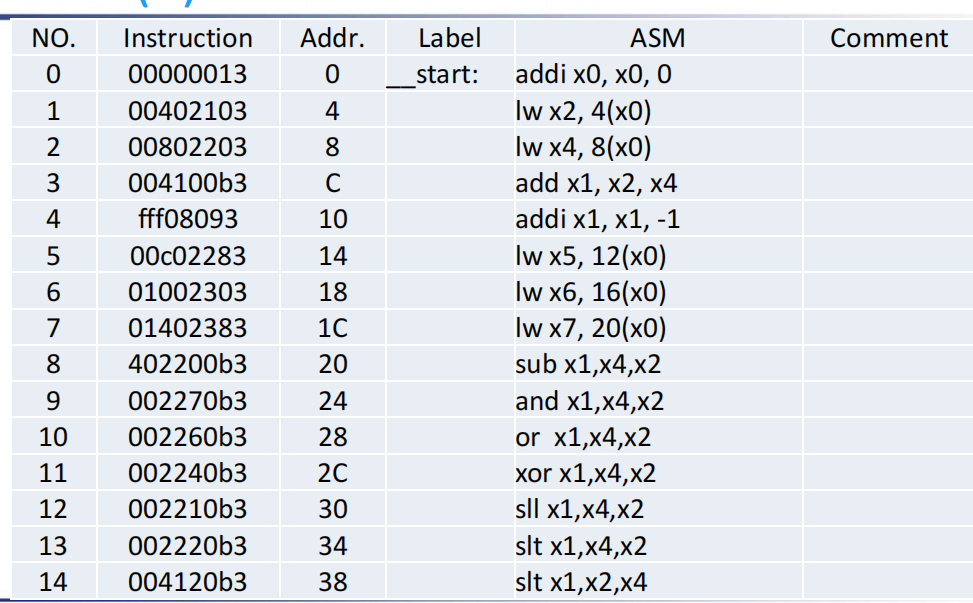
如果load后紧接store，则L指令在EXES指令在ID阶段时得到判断，则需要EXE指令后的IF到ID，ID到EXE均Stall一个cycle，即阶段寄存器使能置0。

如果ID阶段判断到需要跳转，则采取Branch predict taken策略，flush IF到ID的阶段寄存器并使PC取指令寄存器stall一个cycle等待分支控制信号与正确的指令地址。

1. **仿真结果和分析**

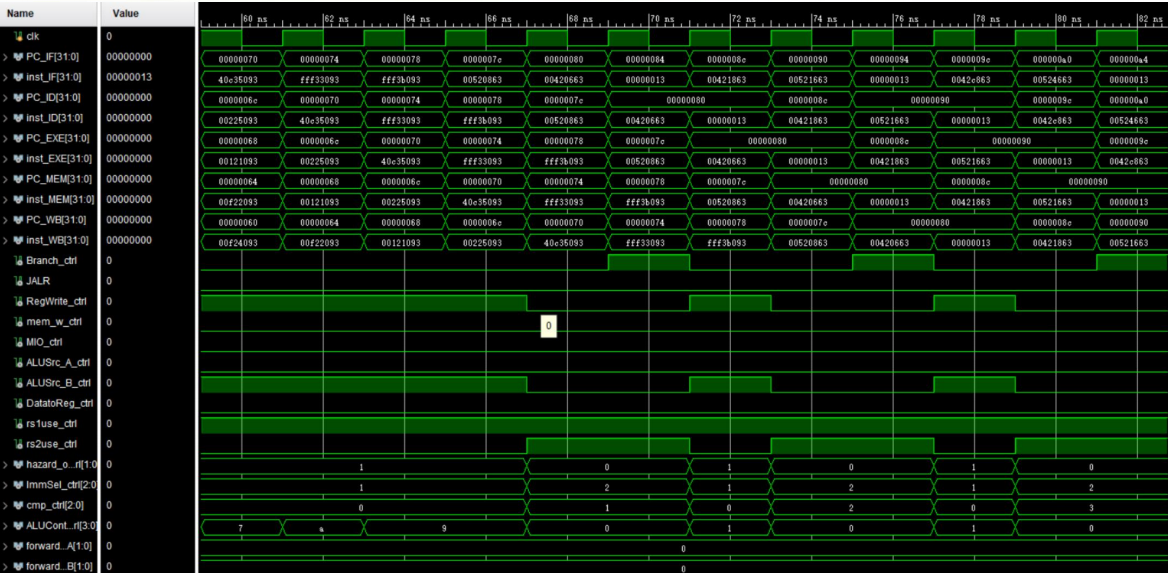
数据冒险：

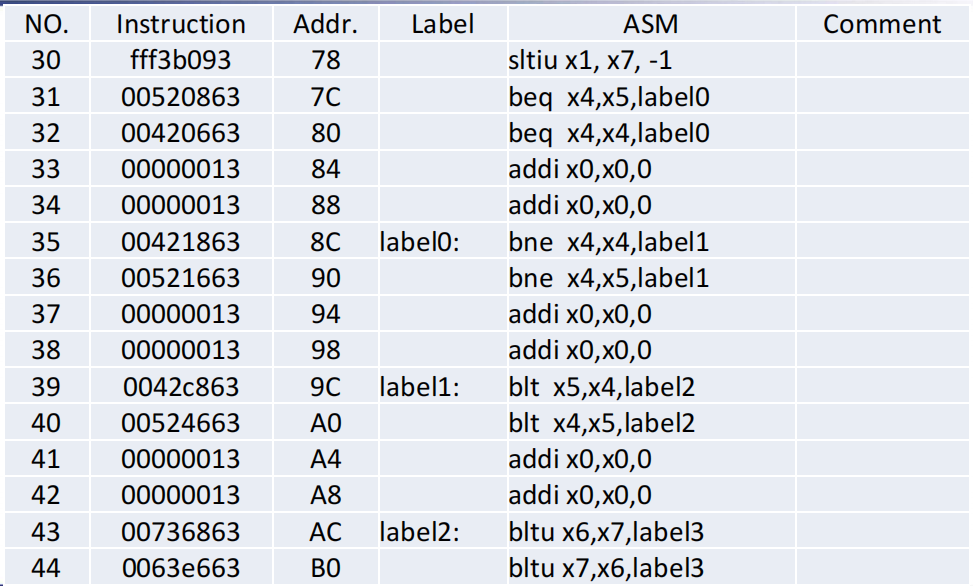




lw后第二条指令读相应寄存器：从上图可以看出t = 9ns时指令add x1, x2, x4正处于ID阶段，且lw x2, 4(x0)处于MEM阶段，发生了lw后的第二条指令读相应寄存器，hazard optype为1，forward为3，与理论相对应

控制冒险：





分析：t = 69s时beq x4, x5 label0 处于ID阶段，分支控制信号Branch\_ctrl = 1，信号无误，此时判断发生跳转，PC在下一周期跳到8C，即label0，结果无误

1. **下板结果（简要概括即可）**

因基础较差，未能及时完成实验结果下板，但可以预测与仿真结果大体一致

1. **讨论与心得（optional）**
2. 本次实验主要为理解五级流水线CPU各级的信号通路，理解三种冒险与解决方法，熟悉基本的指令与指令解码方式。重点和难点在于信号通路的梳理与冒险的解决，要合理设计控制信号，注意时钟周期与阶段寄存器的使能与冲刷控制。