**流水线CPU仿真实验报告**

第33组

成员：李庆锋（组长） 马浩元 李燚

组长联系方式：18568768665

2284817507@qq.com

目　录

[一、设计题目 3](#_Toc53151304)

[二、参与成员及分工 3](#_Toc53151305)

[三、设计内容 3](#_Toc53151306)

[四、设计设备与工具 3](#_Toc53151307)

[五、设计原理及代码实现 4](#_Toc53151308)

[5.1 总体结构 4](#_Toc53151309)

[5.2 主要功能 6](#_Toc53151310)

[5.2.1 跳转指令 6](#_Toc53151311)

[5.2.2 控制相关 6](#_Toc53151312)

[5.2.3 数据相关 7](#_Toc53151313)

[六、仿真测试 8](#_Toc53151314)

[七、实验总结与体会 10](#_Toc53151315)

# 一、设计题目

流水线CPU设计与仿真

# 二、参与成员及分工

|  |  |  |  |
| --- | --- | --- | --- |
| 姓名 | CPU设计分工 | 完成情况 | 贡献度 |
| 李庆锋 | IFS模块编写，设计数据通路图，设计课程实验所需的各类框图、逻辑图，仿真测试，撰写实验报告 | 个人部分完成度高，但对团队的贡献度略低于另两位同学，参与的核心和重难点工作稍少，整体不错 | 29% |
| 马浩元 | IDS模块，EXS模块编写，解决设计过程中各种问题（重点在debug），仿真测试 | 完成度很高，对团队贡献很大 | 37% |
| 李燚 | MES模块，WBS模块编写，设计指令及实现，仿真测试 | 完成度很高，对团队贡献很大 | 34% |

# 三、设计内容

小组同学共同完成一个流水线的CPU，仿真并在精工板上验证。

# 四、设计设备与工具

实验设备：

仿真软件：Vivado\_2019.2

作图软件：<https://app.diagrams.net/>

# 五、设计原理及代码实现

5.1 总体结构

CPU根据五级流水分成五个模块，分别为取指、译码、执行、访存、写回，每条指令执行时，数据依次流经这些模块。五级流水线各阶段的主要工作如下：

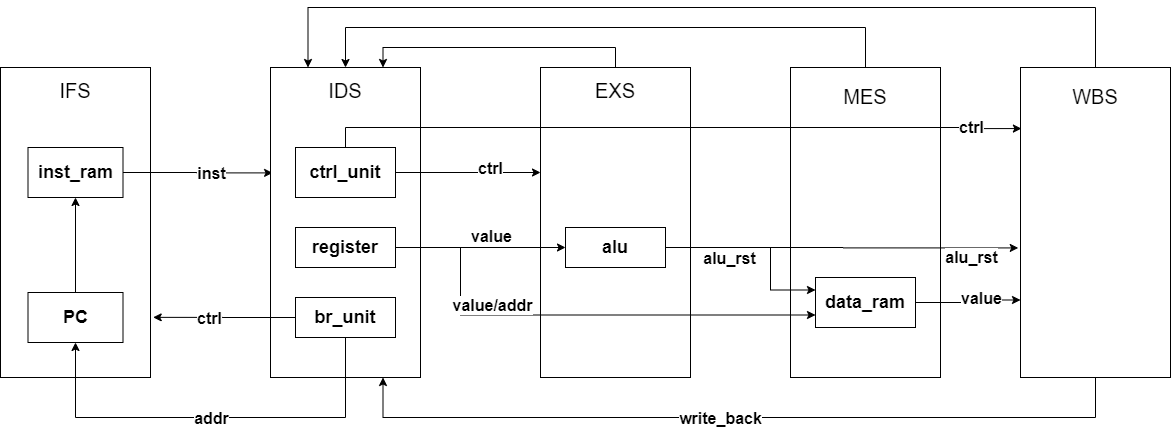
* 取指阶段：从指令存储器读出指令，同时确定下一条指令地址。
* 译码阶段：对指令进行译码，从通用寄存器中读出要使用的寄存器的值，如果指令中含有立即数，则将立即数进行符号扩展或无符号扩展。如果是转移指令且满足转移条件，那么给出转移目标，作为新的指令地址。
* 执行阶段：按照译码阶段给出的操作数、运算类型，进行运算，给出运算结果。如果是Load/Store指令，则另需计算出Load/Store的目标地址。
* 访存阶段：如果是Load/Store指令，那么在此阶段会访问数据存储器，反之，只是将执行阶段的结果向下传递到回写阶段。同时，此阶段还要判断是否有异常需要处理，如果有，那么会清除流水线，然后转移到异常处理例程入口地址处继续执行。
* 回写阶段：将运算结果保存到目标寄存器。

图1 五级流水线CPU模块示意图

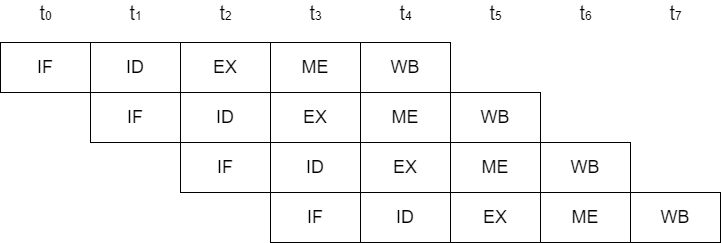
 单周期CPU中，当指令位于某一个特定模块（如MES模块）时，其他模块为空闲状态，会造成浪费。因此引入流水线CPU，对某一个特定模块，当正在执行的某一条特定指令位于其他模块时，该模块转而执行其他指令，而不是闲置。

图2 五级流水线CPU实现简图

以下为顶层模块例化代码，目的是对上述五个阶段进行例化。

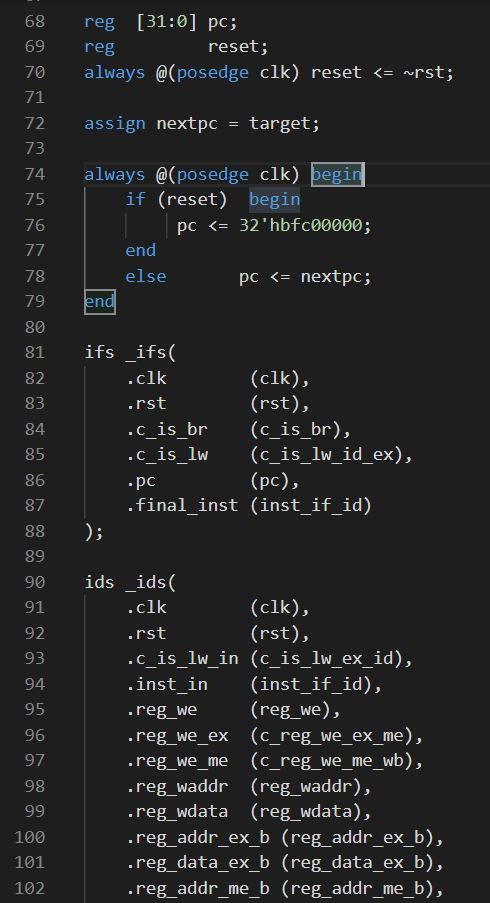
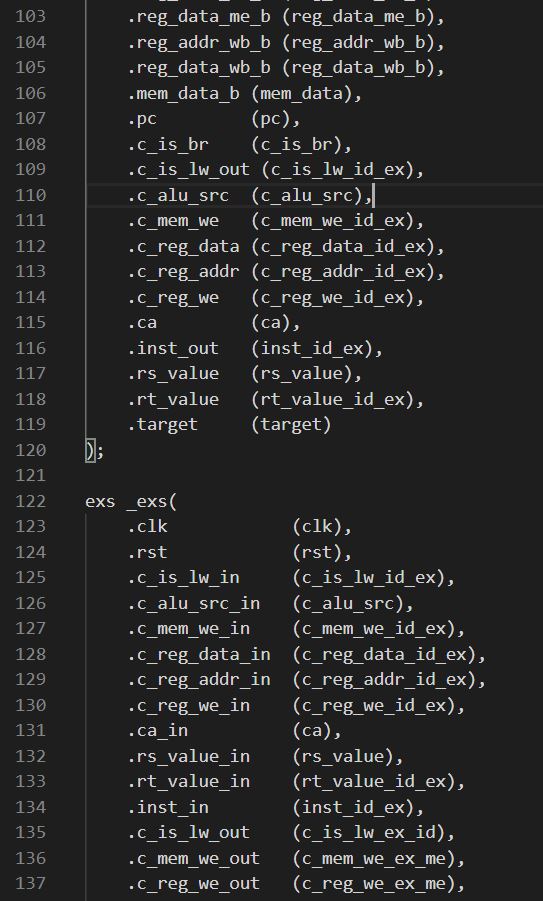
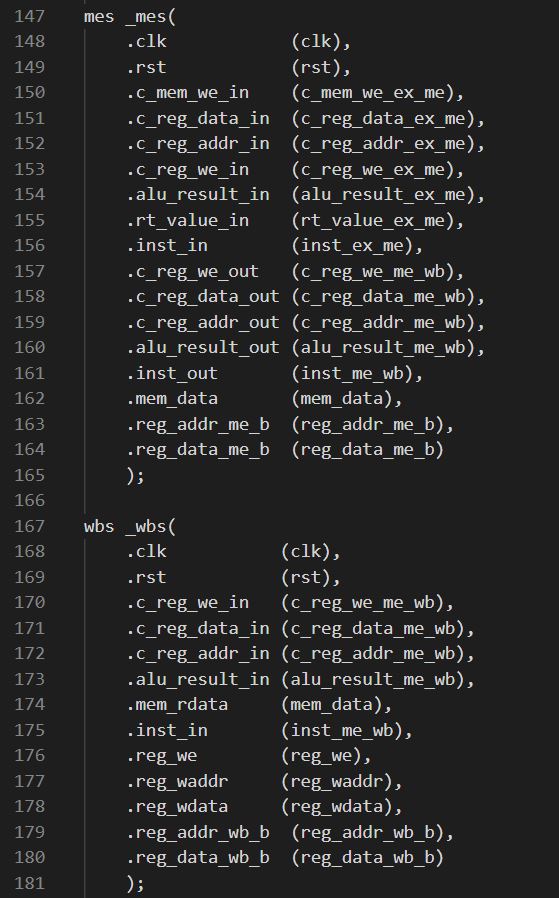


图3 五级流水线CPU总体结构代码实现

5.2 主要功能

5.2.1 跳转指令

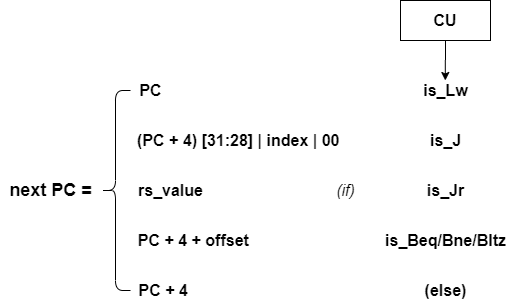
所有的跳转操作都封装在跳转控制模块bru，下一条指令的地址根据控制单元cu生成的控制信号生成。

图4 下一条指令地址示意图

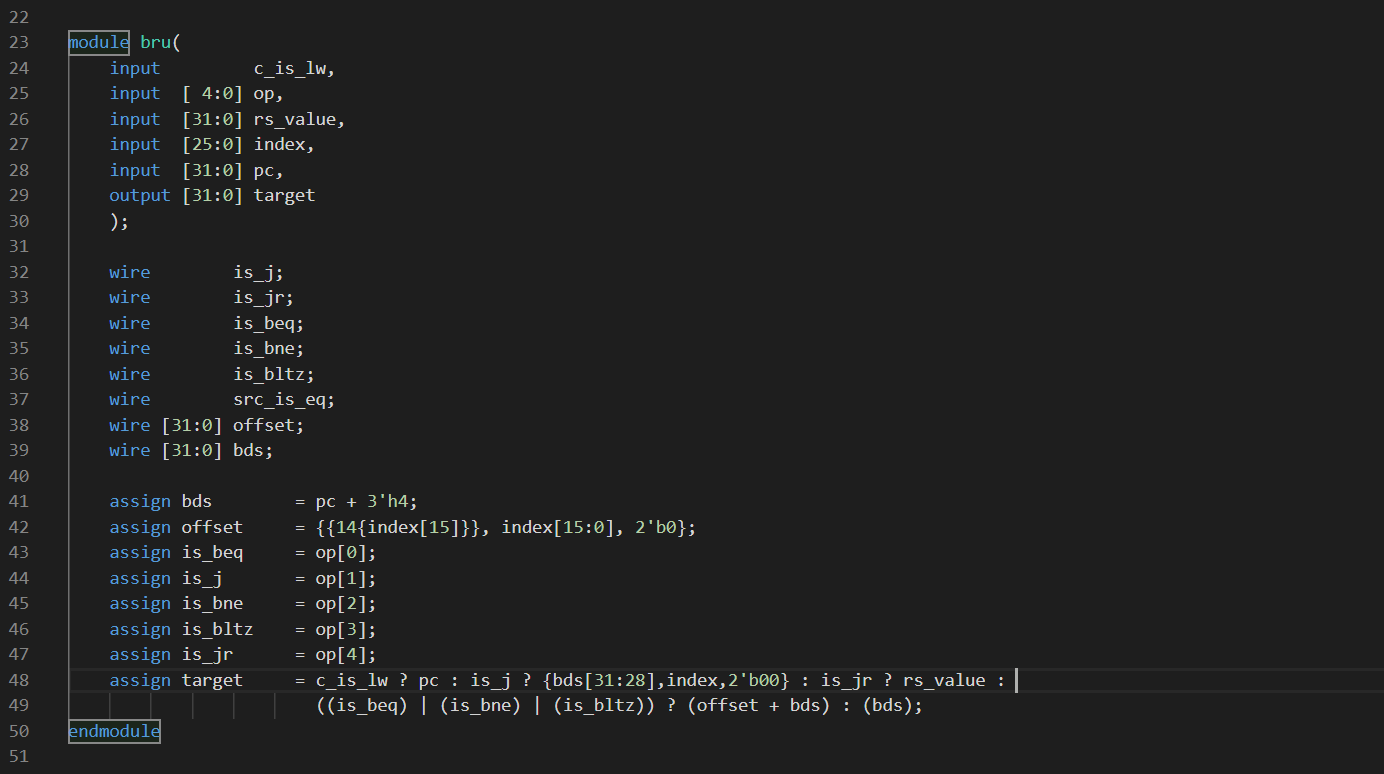
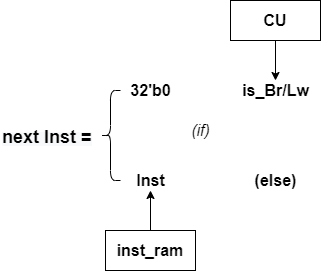
bru模块具体代码实现如下：

图5 bru模块代码实现

5.2.2 控制相关

由于流水执行，跳转指令在译码阶段被识别时，下一条指令已经取指开始执行，如不处理，会导致寄存器值被更改。这一问题被称为控制相关。

可以采用跳转指令后插入空指令的方法处理控制相关。

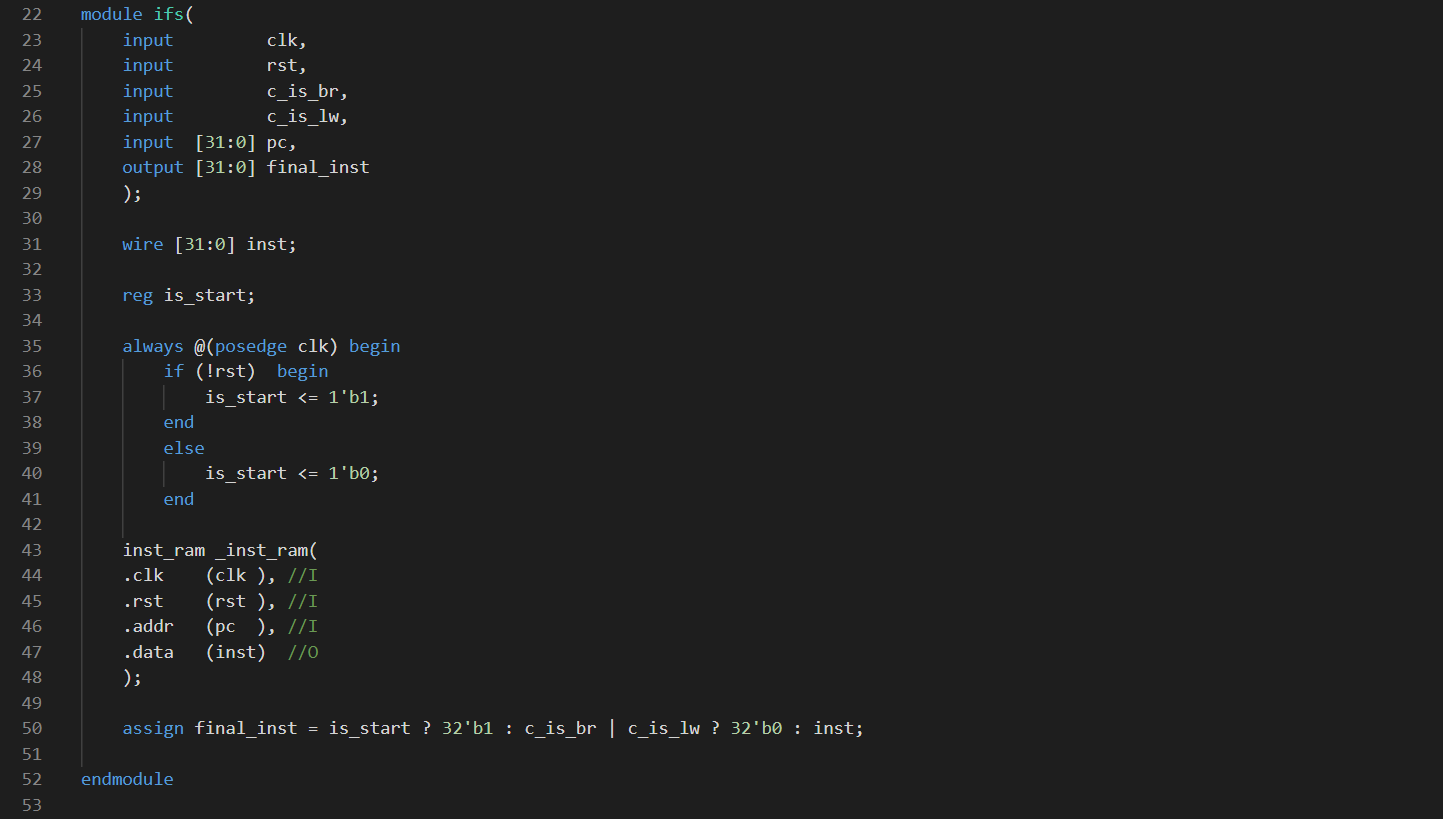
图6 处理控制相关的IFS模块示意图

图7 IFS模块代码实现

5.2.3 数据相关

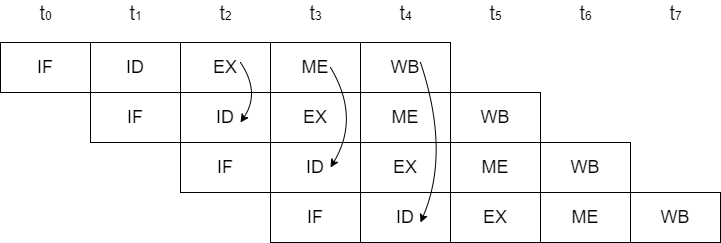
下一条指令读寄存器时，前三条指令还进行写回，如不处理，所读取的内容将不是最新的，这一问题被称为数据相关。采用数据前推的方式解决。

图8 数据前推示意

特殊地，对于LW指令，由于LW指令改写寄存器的写入数据在mem阶段才能确定，因此单纯的数据回推无法解决相关问题，需要在指令后插入一条空指令。为了简化设计，对于所有LW指令，无论是否出现数据相关，均在其后插入空指令（图10）。由于下一条指令仍需执行，所以此时pc的值要保持不变。

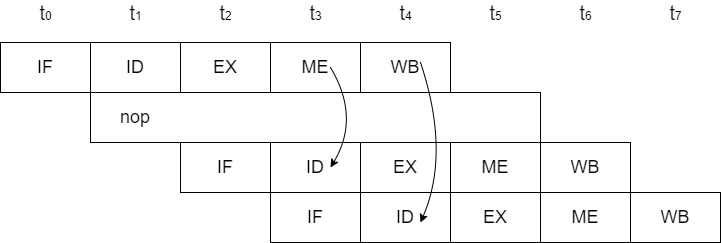
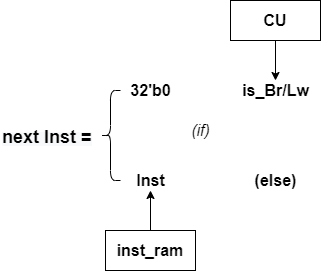
（nextpc = pc if（is\_lw））

图9 LW指令的数据相关解决示意 图10 LW指令后插入空指令

以下为IDS模块结构，目的是用于处理数据相关问题。

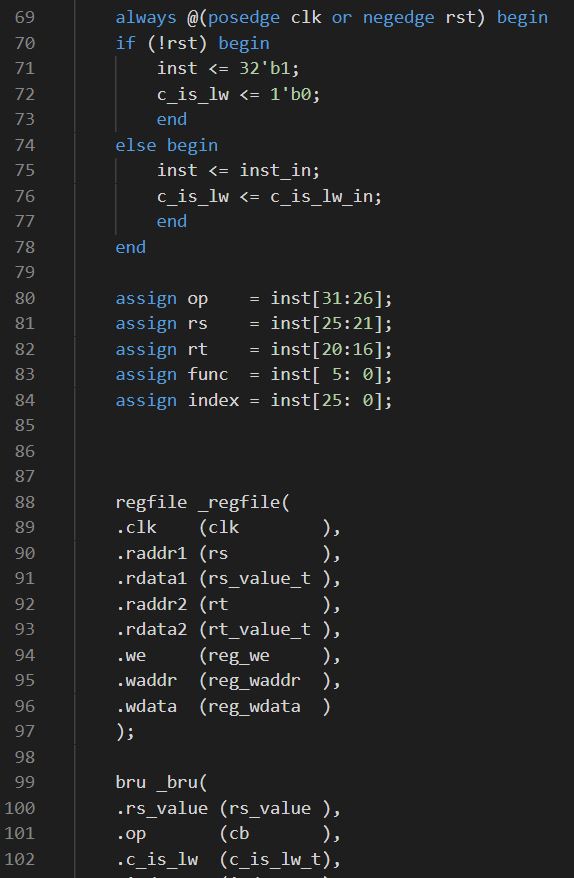
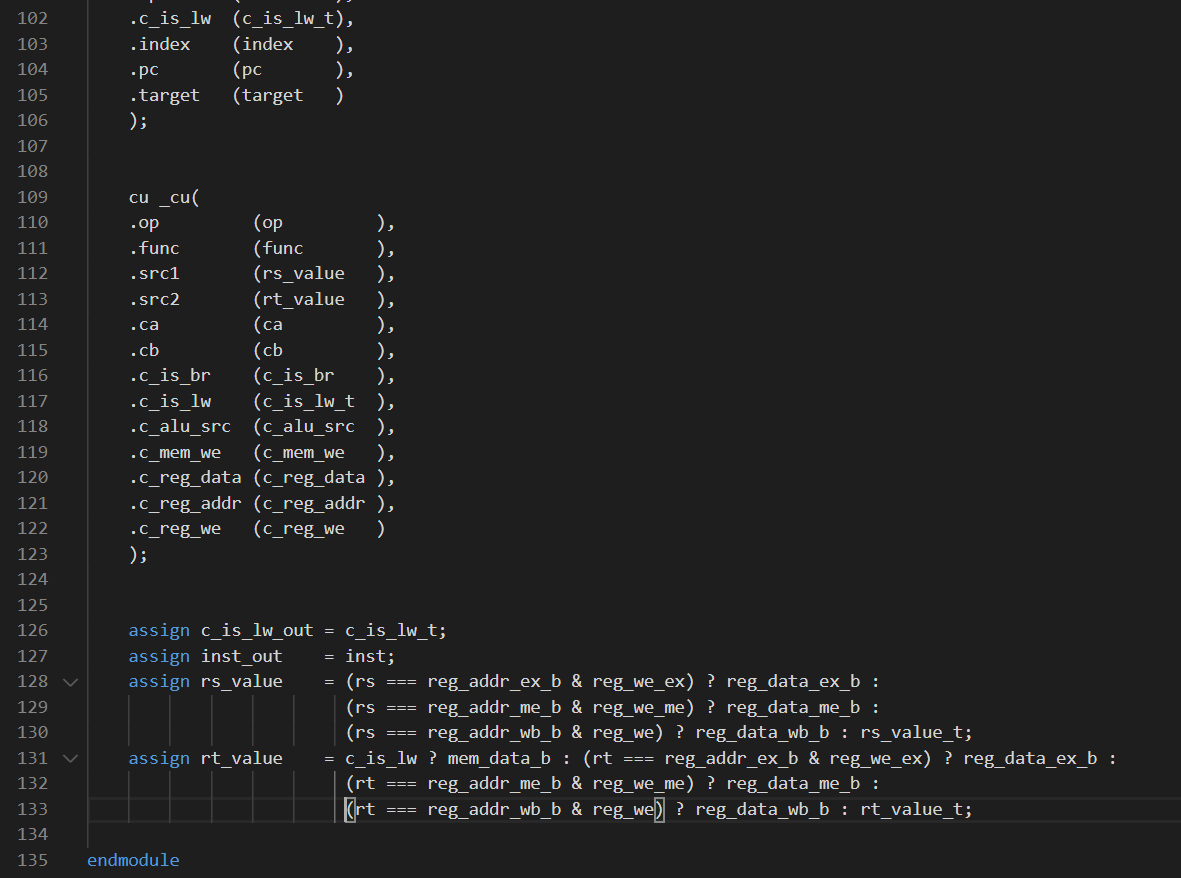


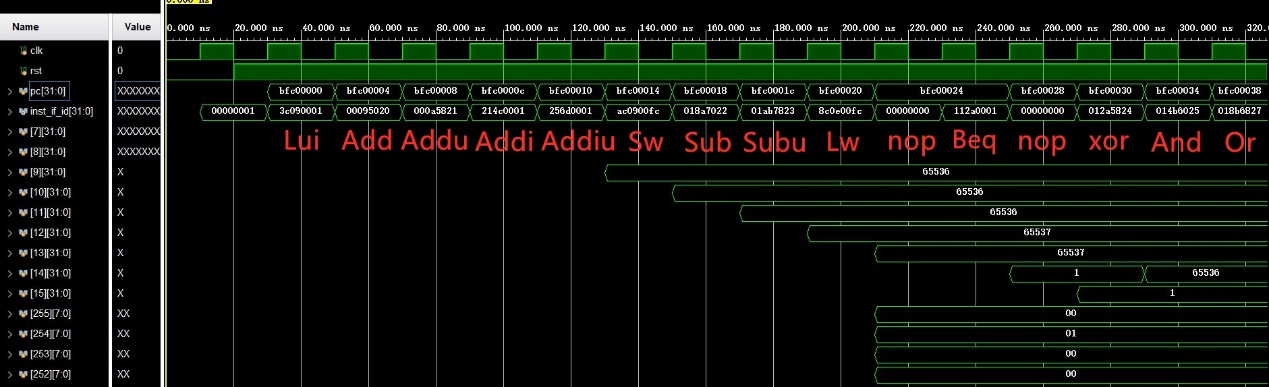
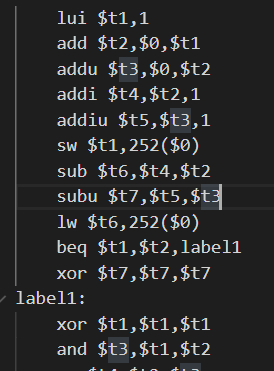
图11 IDS模块结构

# 六、仿真测试

本次课程设计的CPU仿真共实现31条指令，其中22条属于计算类型，4条属于数据存取与传送类型，5条跳转类型，为了简化对测试结果的分析，只对pc地址与寄存器数据（及存储器数据）进行验证。

（一）计算、访存、运算指令

每一个指令的执行共花费5个CPU周期，图示第5个周期结束后写回Lui指令的运算结果，置T1值为2^16=65536。其他指令以此类推。运算到LW指令时，插入一个空指令，同时保证PC值不变，以便于其后Beq等一系列指令的顺利执行。执行到Beq指令时，判断T1=T2=65536，跳转到label指令集xor、and、……

图12 计算、访存、运算指令及其仿真结果

（二）跳转指令

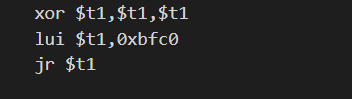
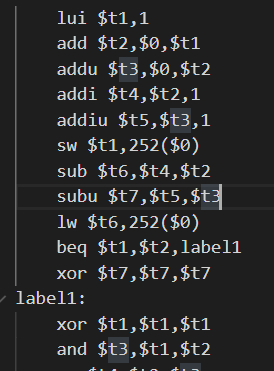
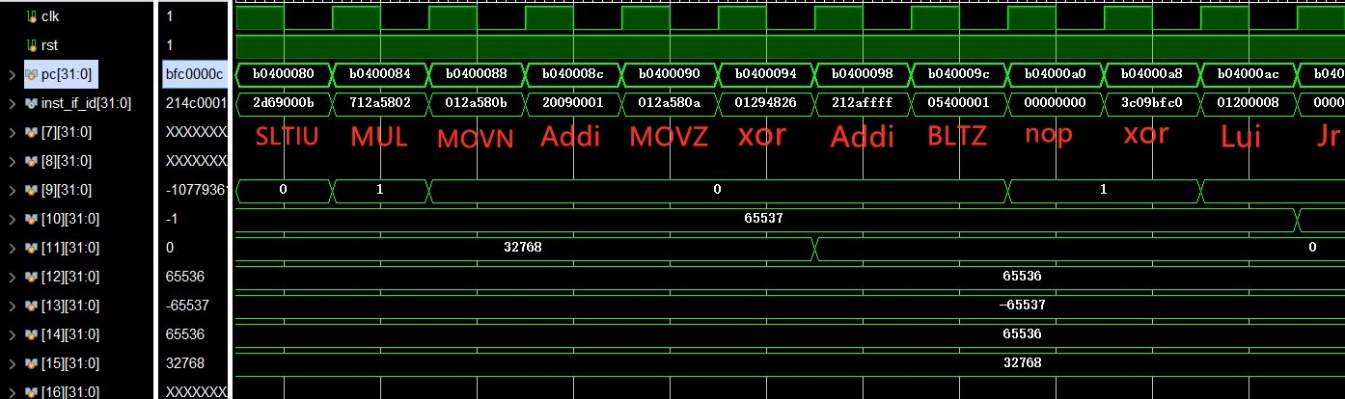
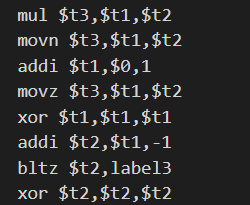
执行到Jr 指令时，其后插一条空指令nop，此时对应$t1 = bfc00000，即跳转到目标地址（程序的开始位置），开始循环执行。

图13 跳转指令仿真结果

（三）传送指令

执行到MOVN指令时，判定此时T2=65537!=0，则在5个周期后将T1赋值给T3。

图14 传送指令仿真结果

# 七、实验总结与体会