**实验3报告**

第32小组

施璠、袁峥

一、实验任务（10%）

1. 设计一款静态 5级流水简单 MIPS CPU。

2. 本次实验要求延续 lab2实验中的以下要求：

(1) CPU复位从虚拟地址 0xbfc00000处取指。

(2) CPU虚实地址转换采用：虚即是实。

(3) CPU对外访存接口为取指、数据访问分开的同步 SRAM接口。

(4) CPU只实现一个操作模式：核心模式，不要求实现其他操作模式。

(5) 不要求支持例外和中断。

(6) CPU顶层连出写回级的 debug信号，以供验证平台使用。

3. 整个实验中，最后要求实现 MIPS I指令集，除 了 ERET（非 MIPS I）、MTC0、MFC0、BREAK、SYSCALL指令，其余指令均要求实现，共 56条指令。

(1) 要求实现 MIPS架构的延迟槽技术，延迟槽不再设定为 NOP 指令，可能是任意指令。

(2) 控制相关由分支指令造成，通过延迟槽技术可以完美解决。

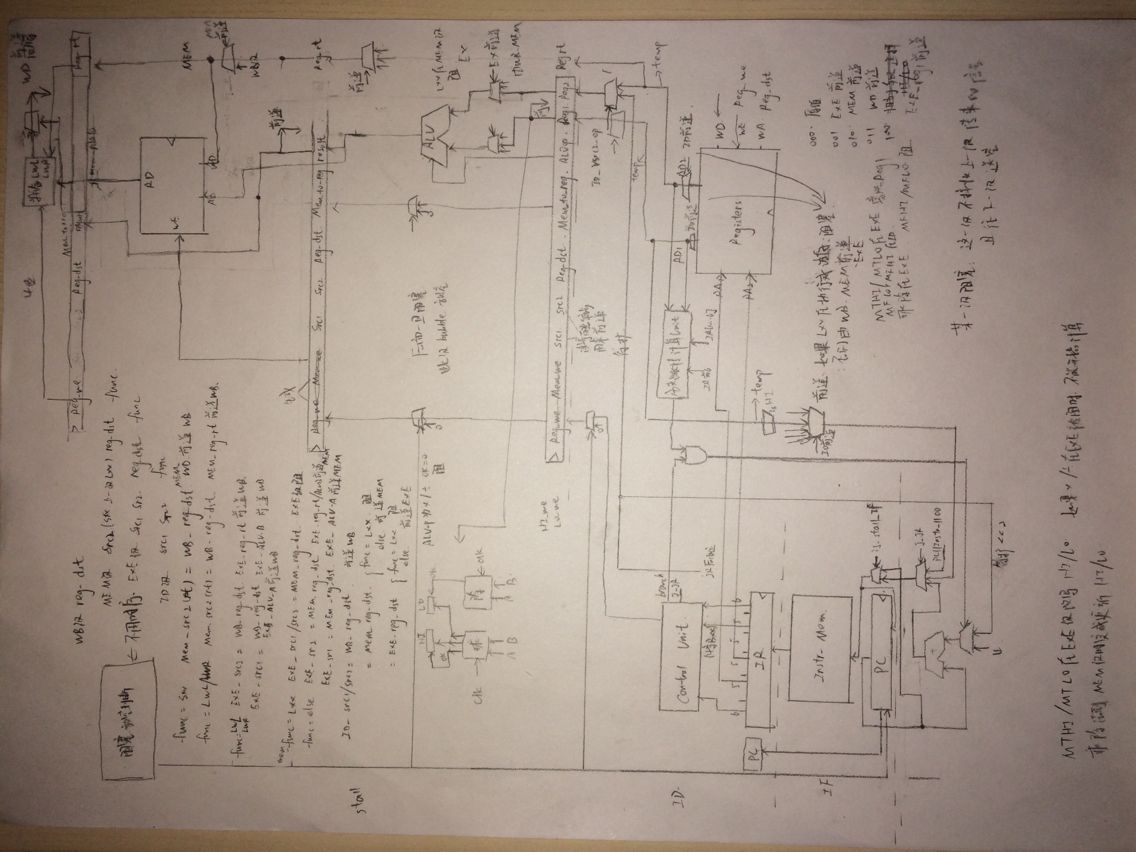
(3) 结构相关即某一级流水停顿了，会阻塞上游的流水级。

(4) 要求数据相关采用前递处理。

(5) 乘除法指令实现可以调用 Xilinx的乘除法 IP，推荐能力有余的同学自行编写乘除法器，乘法采用booth算法+华莱士、除法采用迭代算法。

二、实验设计（30%）

整个CPU设计共分为10个模块，分别为ALU、IF\_stage、ID\_stage、EXE\_stage、MEM\_stage、WB\_stage、next\_pc、regfile、stall和mycpu。其中IF\_stage、ID\_stage、EXE\_stage、MEM\_stage和WB\_stage分别为CPU五级流水的阶段，ALU模块在EXE\_stage级进行算逻运算，next\_pc产生下条指令地址，regfile为寄存器堆，stall为流水线的阻塞总调度，mycpu是整个设计的顶层，负责其他各个模块的整体调度。整体设计图如下：



（一）ALU模块

（1）基本概述

基本采用了在文档LEC02\_Verilog复习提供的alu代码，修改了其中的几处错误，一处是进行slt和sltu运算时，减法没有进行运算，另一处是srl和sra在处理上有些问题，其他地方采用了同样的设计。

（2）接口定义

| **名称** | **方向** | **位宽** | **功能描述** |
| --- | --- | --- | --- |
| aluop | IN | 6 | ALU操作选择信号 |
| vsrc1 | IN | 32 | ALU第一个操作数 |
| vsrc2 | IN | 32 | ALU第二个操作数 |
| result | OUT | 32 | ALU运算结果 |

（二）IF\_stage模块

（1）基本概述

时钟上升沿更新IF\_pc的值，如果IF级没有被阻塞，就用next\_pc，如果IF级被阻塞，就进行复位，也就是传递 bubble。同时IF\_inst记录指令存储器读出的数据。

（2）接口定义

| **名称** | **方向** | **位宽** | **功能描述** |
| --- | --- | --- | --- |
| clk | IN | 1 | 时钟信号 |
| resten | IN | 1 | 复位信号，低电平有效 |
| IF\_stall | IN | 1 | 决定IF级是否阻塞 |
| next\_pc | IN | 32 | 下一条指令地址 |
| inst\_sram\_rdata | IN | 32 | 指令存储器的读数据 |
| inst\_sram\_addr | OUT | 32 | 指令存储器的读地址 |
| IF\_pc | OUT | 32 | IF级的pc |
| IF\_inst | OUT | 32 | IF级的指令 |

（三）ID\_stage模块

（1）基本概述

这一级主要是将IF级传来的指令存入指令寄存器，并将该条指令进行解码，准备好后面流水级需要的控制信号，同时该级要读取寄存器堆的数据，并完成分支指令是否跳转的判断，然后将结果传回next\_pc。

（2）接口定义

| **名称** | **方向** | **位宽** | **功能描述** |
| --- | --- | --- | --- |
| clk | IN | 1 | 时钟信号 |
| resten | IN | 1 | 复位信号，低电平有效 |
| ID\_stall | IN | 1 | 决定ID级是否阻塞 |
| IF\_inst | IN | 32 | IF级往ID级传送的指令 |
| IF\_pc | IN | 32 | IF级往ID级传送的pc |
| ID\_reg\_raddr1 | OUT | 5 | 寄存器堆的读地址1 |
| ID\_reg\_rdata1 | IN | 32 | 寄存器堆的读数据1 |
| ID\_reg\_raddr2 | OUT | 5 | 寄存器堆的读地址2 |
| ID\_reg\_rdata2 | IN | 32 | 寄存器堆的读数据2 |
| ID\_br\_taken | OUT | 1 | 判断如果ID级是分支指令，是否满足跳转条件 |
| ID\_br\_type | OUT | 1 | ID级是否为分支指令 |
| ID\_j\_type | OUT | 1 | ID级是否为跳转指令 |
| ID\_jr\_type | OUT | 1 | ID级是否为寄存器跳转指令 |
| ID\_br\_index | OUT | 16 | 分支指令的偏移地址 |
| ID\_j\_index | OUT | 26 | 跳转指令的跳转地址 |
| ID\_jr\_index | OUT | 32 | 寄存器跳转指令的从寄存器堆读出的跳转地址 |
| ID\_vsrc1\_for | IN | 3 | 寄存器堆读端口1的前递选择信号 |
| ID\_vsrc2\_for | IN | 2 | 寄存器堆读端口2的前递选择信号 |
| HI | IN | 32 | 前递传来的HI寄存器的值 |
| LO | IN | 32 | 前递传来的LO寄存器的值 |
| EXE\_for | IN | 32 | 从EXE级前递传来的值 |
| MEM\_for | IN | 32 | 从MEM级前递传来的值 |
| WB\_for | IN | 32 | 从WB级前递传来的值 |
| ID\_pc\_out | OUT | 32 | ID级往EXE级传递的pc |
| ID\_inst\_out | OUT | 32 | ID级往EXE级传递的指令 |
| ID\_src1\_out | OUT | 5 | ID级指令读寄存器堆时的读地址1 |
| ID\_src2\_out | OUT | 5 | ID级指令读寄存器堆时的读地址2 |
| ID\_dest\_out | OUT | 5 | ID级指令写回时的目的寄存器 |
| ID\_ALUop\_out | OUT | 6 | ID级指令ALU操作数 |
| ID\_vsrc1\_out | OUT | 32 | ID级指令读寄存器堆的读数据1 |
| ID\_vsrc2\_out | OUT | 32 | ID级指令读寄存器堆的读数据1 |
| ID\_reg\_rt\_out | OUT | 32 | ID级指令rt寄存器的读数据（针对LOAD指令） |
| ID\_HI\_we\_out | OUT | 1 | ID级指令HI寄存器的写使能信号 |
| ID\_LO\_we\_out | OUT | 1 | ID级指令LO寄存器的写使能信号 |
| ID\_inst | OUT | 32 | ID级的指令 |

（四）EXE\_stage模块

（1）基本概述

这一级主要是将ID级传来的两个数据进行ALU运算，并将结果送给MEM级，同时如果是进行的乘除法操作，需要多拍才能完成运算，因此需要将后续指令进行阻塞。乘除法完成后，将运算结果送到HI和LO寄存器。

（2）接口定义

| **名称** | **方向** | **位宽** | **功能描述** |
| --- | --- | --- | --- |
| clk | IN | 1 | 时钟信号 |
| resten | IN | 1 | 复位信号，低电平有效 |
| EXE\_stall | IN | 1 | 决定EXE级是否阻塞 |
| ID\_pc | IN | 32 | ID级往EXE级传送的指令 |
| ID\_inst | IN | 32 | ID级往EXE级传送的pc |
| ID\_src1 | IN | 5 | ID级指令的读寄存器地址1 |
| ID\_src2 | IN | 5 | ID级指令的读寄存器地址2 |
| ID\_dest | IN | 5 | ID级指令的写回寄存器地址 |
| ID\_ALUop | IN | 6 | ID级指令的ALU操作选择信号 |
| ID\_vsrc1 | IN | 32 | ID级传递的ALU操作数1 |
| ID\_vsrc2 | IN | 32 | ID级传递的ALU操作数2 |
| ID\_reg\_rt | IN | 32 | ID级传递的rt寄存器的读数据（针对LOAD指令） |
| ID\_HI\_we | IN | 1 | ID级传递的HI寄存器写使能信号 |
| ID\_LO\_we | IN | 1 | ID级传递的LO寄存器写使能信号 |
| EXE\_srcA\_for | IN | 2 | EXE级的ALU操作数1的前递选择信号 |
| EXE\_srcB\_for | IN | 2 | EXE级的ALU操作数2的前递选择信号 |
| EXE\_reg\_rt\_for | IN | 2 | EXE级指令的rt寄存器的前递选择信号 |
| MEM\_for | IN | 32 | 从MEM级前递传来的值 |
| WB\_for | IN | 32 | 从WB级前递传来的值 |
| EXE\_pc\_out | OUT | 32 | EXE级的pc |
| EXE\_inst\_out | OUT | 32 | EXE级的指令 |
| EXE\_src1\_out | OUT | 5 | EXE级指令的读寄存器地址1 |
| EXE\_src2\_out | OUT | 5 | EXE级指令的读寄存器地址2 |
| EXE\_dest\_out | OUT | 5 | EXE级指令的写会寄存器地址 |
| EXE\_result\_out | OUT | 32 | EXE级ALU运算结果 |
| EXE\_reg\_rt\_out | OUT | 32 | EXE级指令rt寄存器的读数据（针对LOAD指令） |
| EXE\_HI\_we\_out | OUT | 1 | EXE级指令的HI寄存器的写使能信号 |
| EXE\_LO\_we\_out | OUT | 1 | EXE级指令的LO寄存器的写使能信号 |
| EXE\_for | OUT | 32 | EXE级向前传送的前递信号 |
| EXE\_inst | OUT | 32 | EXE级指令 |
| EXE\_dest | OUT | 5 | EXE级指令的回写寄存器 |
| EXE\_memaddr | OUT | 32 | EXE级提前送给MEM级的访存读地址 |
| EXE\_data\_sram\_ren | OUT | 1 | EXE级提前送给MEM级的访存读使能信号 |
| HI\_out | OUT | 32 | EXE级乘除法指令运算结果 |
| LO\_out | OUT | 32 | EXE级乘除法指令运算结果 |
| EXE\_mul\_div\_validout | OUT | 1 | EXE级乘除法指令输出有效信号 |

（五）MEM\_stage模块

（1）基本概述

这一级完成数据存储器的读写操作，如果是LOAD指令，读地址在EXE级已经提前送到数据存储器的地址端口，等到MEM级上升沿到来，即完成读取数据，然后将其存入寄存器。如果是STORE指令，在当拍计算出写使能信号，并传递数据存储器的写数据和写地址，在下一拍上升沿时完成数据存储器的更新。

（2）接口定义

| **名称** | **方向** | **位宽** | **功能描述** |
| --- | --- | --- | --- |
| clk | IN | 1 | 时钟信号 |
| resten | IN | 1 | 复位信号，低电平有效 |
| EXE\_pc | IN | 32 | EXE级的pc |
| EXE\_inst | IN | 32 | EXE级的指令 |
| EXE\_src1 | IN | 5 | EXE级指令的读寄存器地址1 |
| EXE\_src2 | IN | 5 | EXE级指令的读寄存器地址2 |
| EXE\_dest | IN | 5 | EXE级指令的寄存器写回地址 |
| EXE\_result | IN | 32 | EXE级ALU运算结果 |
| EXE\_reg\_rt | IN | 32 | EXE级传递的rt寄存器数据（LOAD指令在WB级使用） |
| EXE\_memaddr | IN | 32 | EXE级提前准备的访存读地址 |
| EXE\_data\_sram\_ren | IN | 1 | EXE级提前准备的访存读使能信号 |
| MEM\_reg\_rt\_for | IN | 1 | MEM级rt寄存器传递数据的前递选择信号 |
| WB\_for | IN | 32 | 从WB级前递传来的值 |
| data\_sram\_rdata | IN | 32 | 数据存储器的读数据 |
| data\_sram\_en | OUT | 1 | 数据存储器的片选信号 |
| data\_sram\_wen | OUT | 4 | 数据存储器的写使能信号 |
| data\_sram\_wdata | OUT | 32 | 数据存储器的写数据 |
| data\_sram\_addr | OUT | 32 | 数据存储器的地址 |
| MEM\_pc\_out | OUT | 32 | MEM级的pc |
| MEM\_inst\_out | OUT | 32 | MEM级的指令 |
| MEM\_reg\_we\_out | OUT | 2 | MEM级写地址的最后两位（用于LOAD指令在WB级拼接写回寄存器的值） |
| MEM\_dest\_out | OUT | 5 | MEM级指令的写回寄存器地址 |
| MEM\_memtoreg\_out | OUT | 1 | MEM级指令写回寄存器数据选择信号 |
| MEM\_result\_out | OUT | 32 | MEM级传递的ALU计算结果 |
| MEM\_mem\_rdata\_out | OUT | 32 | MEM级数据存储器的读数据 |
| MEM\_reg\_rt\_out | OUT | 32 | MEM级传递的rt寄存器数据（LOAD指令在WB级使用） |
| MEM\_for | OUT | 32 | MEM级向前传送的前递信号 |
| MEM\_src2 | OUT | 5 | MEM级指令读寄存器的地址2 |

（六）WB\_stage模块

（1）基本概述

将ALU模块的计算结果或者数据存储器读取的结果送到寄存器堆改写相应寄存器。这里需要注意的情况是如果是LOAD指令，需要写回的结果与原寄存器里的数据可能有关，因此将原寄存器中的数据用MEM\_reg\_rt一路送到WB级，并根据访存地址的最后两位（即MEM\_reg\_we信号），在WB级进行拼接后再送到寄存器堆进行改写。

（2）接口定义

| **名称** | **方向** | **位宽** | **功能描述** |
| --- | --- | --- | --- |
| clk | IN | 1 | 时钟信号 |
| resten | IN | 1 | 复位信号，低电平有效 |
| MEM\_pc | IN | 32 | MEM级传递的pc |
| MEM\_inst | IN | 32 | MEM级传递的指令 |
| MEM\_reg\_we | IN | 2 | MEM级传递的访存地址最后两位，用于与MEM\_reg\_rt信号结合，进行LOAD指令的写回数据拼接 |
| MEM\_dest | IN | 5 | MEM级传递的写回寄存器号 |
| MEM\_memtoreg | IN | 1 | MEM级传递的写回数据选择信号 |
| MEM\_result | IN | 32 | MEM级传递的ALU运算结果 |
| MEM\_mem\_rdata | IN | 32 | MEM级访存的读数据 |
| MEM\_reg\_rt | IN | 32 | MEM级传递的rt寄存器数据（用于LOAD指令） |
| WB\_reg\_write\_en | OUT | 1 | WB级写回写使能信号 |
| WB\_reg\_wdata | OUT | 32 | WB级写回数据 |
| WB\_reg\_addr | OUT | 5 | WB级指令写回寄存器号 |
| WB\_pc | OUT | 32 | WB级的pc |
| WB\_for | OUT | 32 | WB级向前传送的前递信号 |

（七）next\_pc模块

（1）基本概述

根据ID级送出的分支跳转相关信号，计算下一条指令地址，并提前送到IF级的指令存储器读地址。

（2）接口定义

| **名称** | **方向** | **位宽** | **功能描述** |
| --- | --- | --- | --- |
| IF\_pc | IN | 32 | IF级的pc |
| ID\_br\_taken | IN | 1 | ID级分值跳转是否成功 |
| ID\_br\_type | IN | 1 | ID级指令是否为分支指令 |
| ID\_j\_type | IN | 1 | ID级指令是否为跳转指令 |
| ID\_jr\_type | IN | 1 | ID级指令是否为寄存器跳转指令 |
| ID\_br\_index | IN | 16 | ID级分支指令偏移地址 |
| ID\_j\_index | IN | 26 | ID级跳转指令跳转地址 |
| ID\_jr\_index | IN | 32 | ID级寄存器跳转指令跳转地址 |
| inst\_sram\_en | OUT | 1 | 指令存储器读使能信号 |
| next\_pc | OUT | 32 | 下一条指令地址 |

（八）regfile模块

（1）基本概述

该寄存器堆为异步读同步写的，在WB级写寄存器堆，在后一上升沿完成更新；在ID级读寄存器堆。

（2）接口定义

| **名称** | **方向** | **位宽** | **功能描述** |
| --- | --- | --- | --- |
| clk | IN | 1 | 时钟信号 |
| raddr1 | IN | 5 | 寄存器堆读地址1 |
| rdata1 | OUT | 32 | 寄存器堆读数据1 |
| raddr2 | IN | 5 | 寄存器堆读地址2 |
| rdata2 | OUT | 32 | 寄存器堆读数据2 |
| we | IN | 1 | 寄存器堆写使能信号，高低平有效 |
| waddr | IN | 5 | 寄存器堆写地址 |
| wdata | IN | 32 | 寄存器堆写数据 |

（九）stall模块

（1）基本概述

该模块整体控制整个五级流水的阻塞或运行，并提供不同端口的前递选择信号。

整个设计中需要用到前递选择的地方一共有6处，分别是ID级中两个寄存器读端口读出的数据，EXE级ALU模块两个输入口的数据，以及EXE级和MEM级传递的reg\_rt信号（该信号是用于LOAD指令在WB级与原rt寄存器值进行拼接）。

五级流水中，可能会出现阻塞的有ID级、IF级和EXE级。出现阻塞的情况如下：

1. MEM级中是LOAD指令，EXE级指令的寄存器读地址和MEM级寄存器写回地址相同时，EXE级及前面流水阻塞等待。
2. MEM级是STORE指令且EXE级是LOAD指令，EXE级及前面流水阻塞等待，因为LOAD指令的访存地址需要提前送到MEM级的数据存储器的地址端口，但此时STORE指令还在用该端口写数据存储器，因此LOAD指令需要阻塞等待。
3. EXE级是乘除法指令，且乘除法结果有效信号为0，即还没有完成运算，EXE级及前面的流水阻塞等待。
4. MEM级中是LOAD指令，ID级指令的寄存器读地址和MEM级寄存器写回地址相同时，ID级及前面流水阻塞等待。
5. EXE级中是LOAD指令，ID级指令的寄存器读地址和EXE级寄存器写回地址相同时，ID级及前面流水阻塞等待。
6. EXE级是乘除法指令，ID级是MFHI/MFLO指令，即使乘除法完成运算，ID级和IF级仍需要阻塞等待，因为HI和LO寄存器需要等到下一拍上升沿才能完成更新。

（2）接口定义

| **名称** | **方向** | **位宽** | **功能描述** |
| --- | --- | --- | --- |
| WB\_reg\_dst | IN | 5 | WB级指令写回寄存器号 |
| MEM\_src2 | IN | 5 | MEM级指令读寄存器地址2（用于LOAD指令） |
| MEM\_reg\_dst | IN | 5 | MEM级指令写回寄存器号 |
| MEM\_inst | IN | 32 | MEM级指令 |
| EXE\_src1 | IN | 5 | EXE级指令读寄存器地址1 |
| EXE\_src2 | IN | 5 | EXE级指令读寄存器地址2 |
| EXE\_reg\_dst | IN | 5 | EXE级指令写回寄存器号 |
| EXE\_inst | IN | 32 | EXE级指令 |
| ID\_src1 | IN | 5 | ID级指令读寄存器地址1 |
| ID\_src2 | IN | 5 | ID级指令读寄存器地址2 |
| ID\_inst | IN | 32 | ID级指令 |
| EXE\_mul\_div\_validout | IN | 1 | EXE级乘除法结果有效信号 |
| IF\_stall | OUT | 1 | IF级是否阻塞 |
| ID\_stall | OUT | 1 | ID级是否阻塞 |
| EXE\_stall | OUT | 1 | EXE级是否阻塞 |
| ID\_vsrc1\_for | OUT | 3 | ID级读寄存器输出口1前递选择信号 |
| ID\_vsrc2\_for | OUT | 2 | ID级读寄存器输出口2前递选择信号 |
| EXE\_srcA\_for | OUT | 2 | EXE级ALU输入口1前递选择信号 |
| EXE\_srcB\_for | OUT | 2 | EXE级ALU输入口2前递选择信号 |
| EXE\_reg\_rt\_for | OUT | 2 | EXE级reg\_rt信号前递选择信号 |
| MEM\_reg\_rt\_for | OUT | 1 | MEM级reg\_rt信号前递选择信号 |

（十）mycpu模块

该模块是整个设计的顶层，进行不同模块之间的连接，同时存有HI和LO寄存器，根据EXE级传递的信号（乘除法指令完成信号）在时钟上升沿进行更新。

三、实验过程（60%）

（一）实验流水账

9月30日下午2点到5点，阅读文档A05\_“体系结构研讨课”MIPS指令系统规范、A07\_交叉编译工具链安装、LEC02\_Verilog复习、LEC03\_CPU实验开发环境使用说明、LEC04\_仿真调试说明和Lab03\_静态3级流水简单MIPS CPU实现。

9月30日晚上7点到11点，阅读文档REF01\_计算机体系结构\_第五章\_静态流水线，并开始绘制设计图。

10月1日上午10点到晚上11点，绘制设计图。

10月2日下午2点到晚上11点，编写代码。

10月3日上午10点到12点，编写代码。

10月3日下午2点到晚上11点，调试程序，并通过仿真。

10月4日下午2点到晚上11点，处理乘除法部分，编写除法器程序。

10月9日下午3点到晚上11点，发现前递部分在综合时时序有问题，进行修改调试，并通过上板测试。

（二）错误记录

1、错误1

（1）错误现象

在仿真时发现波形上出现一些X或者Z信号。

（2）分析定位过程

在波形图中和程序中找到出现X信号的位置，并结合具体代码确定该X信号是由哪个信号引起的，然后再这样一路往前追踪，直到找到源头。

（3）错误原因

X信号的出现是由于在多路选择时可能某一路信号没有准备好，或者定义的信号位数前后不一致。

Z信号的出现是由于在顶层调度时某一接口的信号漏接。

（4）修正效果

根据设计的需要和真正的想法修改信号的连接，成功解决该类问题。

（5）归纳总结

解决此类问题主要是要求在编写程序时更加细心，特别在顶层接线要仔细，不要写错变量名和信号长度等。

2、错误2

（1）错误现象

ALU中slt、sltu、sra和srl出错。

（2）分析定位过程

跟踪到EXE级中的ALU模块信号，结合输入数据并手动计算，发现计算结果错误。

（3）错误原因

之前直接采用了文档LEC02\_Verilog复习提供的alu代码，里面在这几条指令的处理上有错误。

（4）修正效果

重新分析了计算过程，并在减法选择信号中加入了slt、sltu信号，并重写了sra和srl指令的操作。

（5）归纳总结

不要偷懒，复用别的代码前还需要自己先分析和仔细检查一遍。

3、错误3

（1）错误现象

仿真通过后在综合和布局布线时报错，报的错误有time loop 和 time violation。

（2）分析定位过程

一开始看到time loop的critical warning时很费解，反复检查了上下文发现并没有什么明显的错误，然后进一步检查了布局布线的时序报告，发现不能满足时序要求，接着查找到了超时路径的信息，发现有一些信号从数据存储器一路连到了指令存储器。在与邢老师的交流下，重新对数据前递有了一些理解，某一拍计算出来的结果必须先存到寄存器然后才能前递，因此修改了设计图。

（3）错误原因

错误原因主要是对数据前递的理解不够深刻，导致在时序上没有注意，从而使得从数据存储器读出数据直接被送到了指令存储器，该路径超过所要求的流水级时间。

（4）修正效果

根据要求重新修改了设计图，原来在读数据存储器时没有提前给地址，而是将读出的数据直接送到回写级，因此该数据没有经过寄存器保存，如果同时将该数据进行前递那么就会出现路径过长的状况。因此此处修改为提前送指令存储器的读地址，然后将读出的数据存入寄存器，从而在下一拍再进行前递，这样就可以解决这个问题。

（5）归纳总结

没有对数据前递进行深刻理解，在以后必须充分了解了以后再进行设计和编写代码，同时在设计是不能够只管仿真的结果，也应该注意综合和布局布线是否合理。

4、错误4

（1）错误现象

访问指令ram的时候，得不到应有的指令。

（2）分析定位过程

根据波形，发现在取值阶段inst\_sram\_en被置为0.

（3）错误原因

对de\_br\_taken信号的错误理解，并写成了只有在只有在de\_br\_taken有效时，可以访问指令ram。

（4）修正效果

修改时把inst\_sram\_en恒置为1，保证指令ram一直可以读，通过后面的控制信号决定要不要更新nextpc

（5）归纳总结

同步ram和异步ram相比延迟了一周产生结果，在取值阶段的实际中，需要提前取出nextpc，那么在真正的取值阶段就可以得到当前所需的指令。 同样地，在br操作中需要在译码阶段就判断是否需要跳转。

5、错误5

（1）错误现象

仿真阶段jal指令发生错误

（2）分析定位过程

本来应该写回pc+8, 但是mycpu没有写回值。通过回想mips指令集，发现在这条指令上理解和设计发生偏差。

（3）错误原因

在译码阶段没有给jal指令设置控制信号，错误地认为在此阶段产生信号控制nextpc而后面阶段不需要再执行。所以在运行过程中并没有把pc+8写回31号寄存器。考虑到后面地指令执行已经根据译码级设计好的de\_op传递的信号进行控制，对信号进行修改工作量较大，所以直接给新增了一位控制信号用来判断jal指令。相应地需要修改jal指令在执行阶段的两个操作数和alu控制信号。

（4）修正效果

按上述方法修改jal指令的相关操作后，可以得到正确结果。

（5）归纳总结

设计之初要对指令有深入了解，不要盲目动手写代码，要先设计好，否则修改起来很麻烦。

6、错误6

（1）错误现象

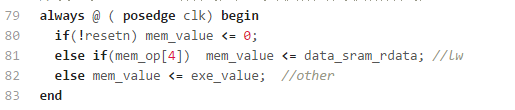
LW指令执行后，load写回的值不是ram中的值，而是alu计算后的数据

（2）分析定位过程

在设计中，控制load的写入选择的信号，是在译码阶段产生的op逐级传递的。根据波形显示，此信号将alu的计算结果写会到了寄存器中，也就是此处的信号错了。追踪波形，发现此处的控制信号已经是下一条指令的信号，而不是当前指令的信号。考虑到在访存阶段对输出mem\_value使用了时序逻辑，而控制信号也是在此阶段由时序逻辑产生，从而找到了错误的地方。

（3）错误原因

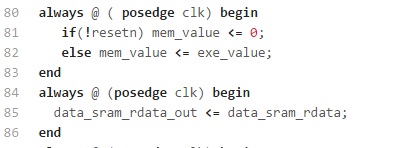
访存阶段对mem\_value的赋值过程如下，当时的设计为了保持住lw的值，在写回过程中，如果是lw操作，就直接把此阶段的mem\_value传给wb\_rf\_wdata，也就是写回数据。在mem\_value的赋值中产生了竞争冒险，在时钟上升沿到来时，mem\_op可能是旧值，也可能是此阶段的新值，从而导致了信号的错误。



（4）修正效果

修改方法是把mem\_value理解成上一拍中exe\_value传递下来的值，而对wb\_rf\_wdata的选择放到写回操作中执行，设置一个寄存器，保存访存阶段中data\_sram\_rdata的值，并把它传递给写回周期。

访存周期中的修改：



写回周期中的修改：



这样修改后，保证了wb\_rf\_wdata是根据指令控制信号选择的正确的值。

（5）归纳总结

在设计时要综合考虑，避免竞争冒险，对信号的产生和使用要有清晰的理解和认识，然后再动手写代码。

四、实验总结

（一）组员：袁峥

这次实验总体来说还算顺利，就是花费的时间有点多，先花了一天阅读了相关的文档，然后花了一天半画完了设计图，花了一天编写程序，花了半天进行调试。目前整个实验的完成情况是还缺少自己编写的乘法器模块，其他内容应该都已经全部完成。在进行整体设计的时候，主要碰到的问题有以下几处：

1、在前递部分花了很长时间思考，具体哪些地方需要前递，以及前递信号如何准备，这方便考虑起来比较麻烦而且容易出错。最后参考了文档REF01\_计算机体系结构\_第五章\_静态流水线中的处理办法，根据实际情况进行了调度。最后在综合部分还是出现了问题，前递前没有把数据存入寄存器，导致路径过长。在与老师交流后认知到了错误并进行了改正。

2、一开始在设计时思考，如果延迟槽指令也是分支跳转指令怎么办，并且百思不得其解，后来在MIPS手册中看到，规范中保证了这种情况不会发生，也就解决了这个问题。

3、乘除法器操作如果放在一拍内完成会导致EXE级过长，因此需要切分流水，但是具体需要切分多少拍不清楚。在咨询老师后决定乘法器2拍完成，除法器可以用16拍或者32拍完成。乘法器目前还没有完成编写，除法器使用原码加减交替法完成，具体内容在下次实验报告中阐述。

4、整个流水控制中怎么进行阻塞。一开始准备在每两级流水之间相互联系，但后来发现这种方式可能过于繁琐，因此新加了一个stall模块来整体控制流水级的阻塞，而被阻塞的流水级往下一级传递的信号全部为0，本级流水寄存器在下一拍时钟上升沿时也不进行更新，仍然保存旧值，这样也就完成了阻塞的操作。

5、LWR、LWL、SWL、SWR这四条指令的理解上花了很长时间，反复查看了MIPS手册，并且结合计算机体系结构课的课件和配图，最后理解了具体操作。

6、LOAD指令在写回寄存器时可能会需要与指令中的rt寄存器原来的值进行拼接，因此在整个流水中需要将rt寄存器的值一直传递到WB级，其中也会需要用到前递，并在WB级与从数据存储器中读取的数据进行拼接，然后再送到寄存器堆的写数据完成数据更新。

（二）组员：施璠

这次实验是和队友分开做的，在队友差不多把后面几个阶段都写完了，我就完全没有压力地开始了第一个阶段。考虑到之前的多周期实验的写法有点乱，本次实验没有在之前的多周期基础上改，而是根据流水分模块重新写。本来想这一阶段把所有的指令都实现的，但是写到后面发现有一些要处理的问题还挺复杂的，虽然译码级把所有指令都包含了，但是执行级访存级和写回级都只考虑了第一个阶段要实现地指令。在本次实验中，学习了有关五级流水的知识，也反应了一些问题，比如开始写实验之前没有完整的设计思路，想到哪里写到哪里。结果就是写了一天bug，又改了一天bug。在debug的过程中，不断发现此前考虑的漏洞，补救起来就有点麻烦。希望以后想通了再开始写。