实验报告成绩:	成绩评定日期:

2022 ~ 2023 学年秋季学期

《计算机系统》必修课

课程实验报告

班级：人工智能 2001 班

组长：刘小龙

组员：王怡正 李鸣骁

报告日期：2022.1.2

目录

1. 实验设计

1.1 小组成员工作量划分

姓名	任务分工	任务量占比
刘小龙	添加算术运算、数据移动、逻辑、跳转、访存指令，参与实现 hilo 寄存器、参与实现 stall	50%
王怡正	主要负责在流水线中添加 stall 相关指令，参与 hilo 的相关指令，参与实验报告的编写	25%
李鸣骁	主要负责在流水线中添加 hilo 相关指令，参与实现 stall 的相关指令，参与实验报告的编写	25%

1.3 运行环境及工具

运行环境：装有 Vivado 的 Linux 服务器。 FPGA 的 Family 为 Artix 7，Package 为 fbg676，型号为 xc7a200tfbg676-2。

编程工具：使用 VSCode 编写代码，使用 Vivado 模拟仿真，使用 git 进行版本管理，使用 GitHub 搭建项目仓库。

1.2 总体设计

项目包括 IF.v，ID.v，EX.v，MEM.v，WB.v，hi_lo_reg.v，mycpu_core.v，mycpu_top.v，这部分搭建了一条流水线的基本框架；及位于/lib 目录下的 alu.v，decoder_2_4.v，decoder_5_32.v，decoder_6_64.v，defines.vh，div.v，mmu.v，regfile.v，这部分构建了 ALU 和寄存器，定义了包含总线宽度信息在内的头文件；及位于/lib/mul 目录下的 add.v，fa.v，mul.v，这部分实现了乘法的运算。

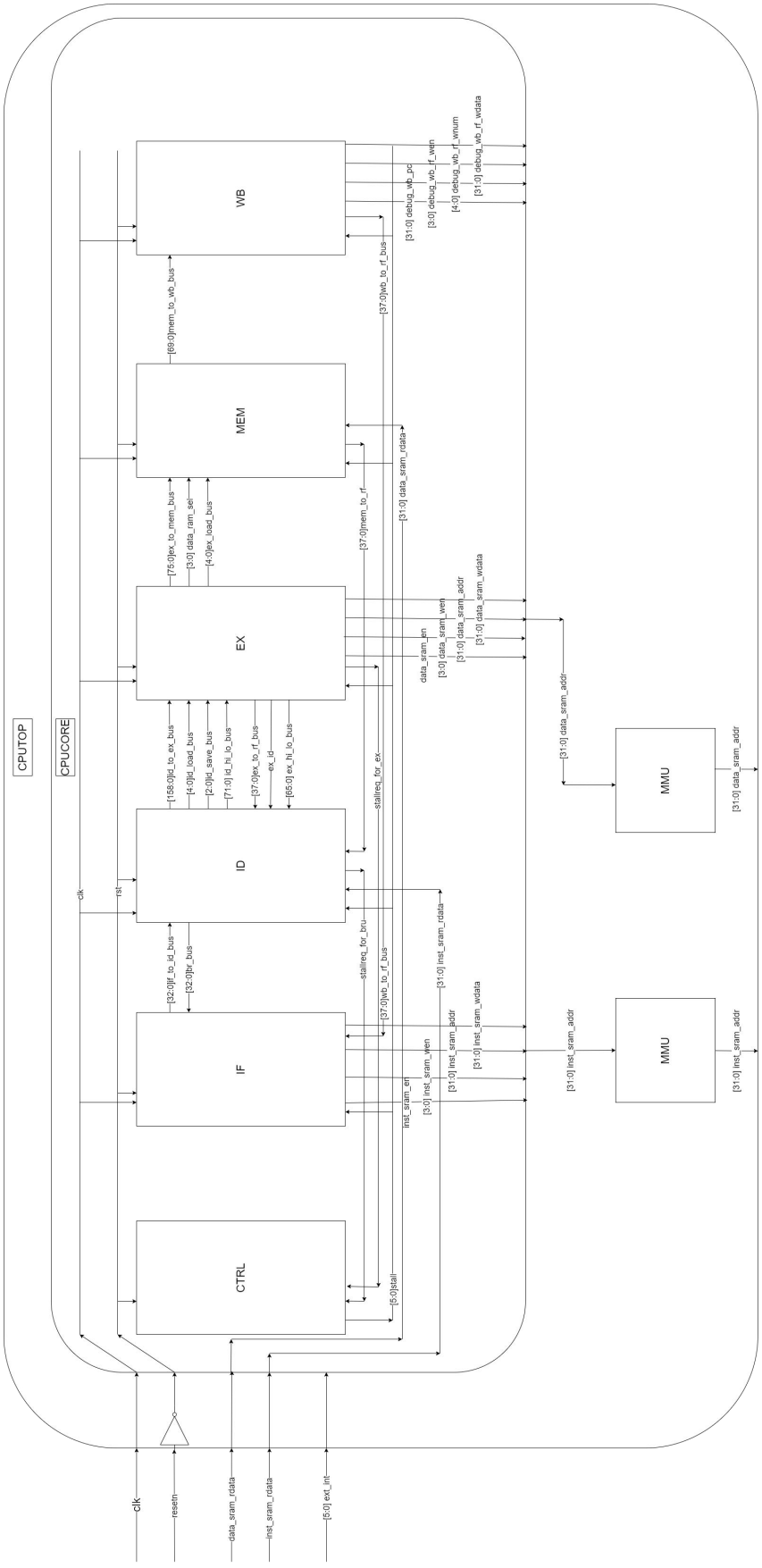


图 1 CPU 流水线示意图

2. 流水线各个阶段的说明

2.1 IF 模块

整体说明：取指令，控制指令延迟槽和跳转指令。接口如图所示。

功能说明：

首先，IF 段会输入时钟信号和复位信号，如果复位信号为真，则复位 pc 值。然后，再判断暂停信号 stall，如果 stall 的值为 1，则暂停延迟槽，即让下一条指令的 pc 值等于当前的 pc 值。再然后，判断 br_bus 的值，若需要跳转，则取出 br_bus 中的地址值赋给 next_pc，然后再把 next_pc 赋给 pc_reg，否则，pc_reg 值为当前 next_pc 的值，next_pc 值还需要加 4。最后，将 pc_reg 的地址发给指令内存，从指令内存中得到相应的 pc 地址对应的值并发给 ID 段。

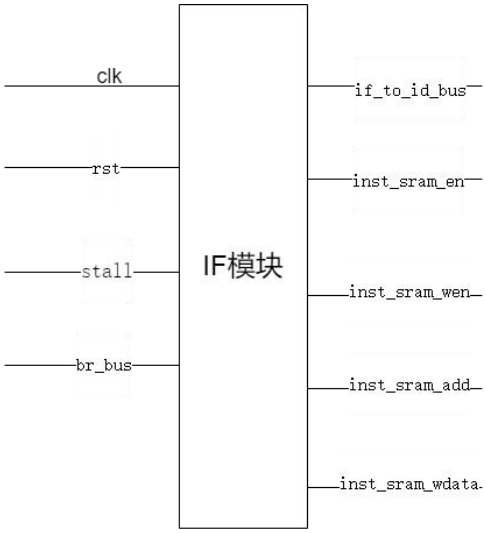


表 1 IF 模块输入输出

序号	接口名	宽度	输入/输出	作用
1	clk	1	输入	时钟信号
2	rst	1	输入	复位信号
3	stall	6	输入	暂停信号，控制指令是否暂停
4	br_bus	33	输入	分支跳转信号，控制延迟槽是否跳转
5	if_to_id_bus	33	输出	IF 段到 ID 段的数据总线
6	inst_sram_en	1	输出	读写使能信号
7	inst_sram_wen	4	输出	写使能信号
8	inst_sram_addr	32	输出	存放指令寄存器的地址
9	inst_sram_wdata	32	输出	存放指令寄存器的数据

2.2 ID 模块

整体说明：

对指令进行译码，将结果传给 EX 段，实现寄存器读写，处理数据相关。接口下图所示。

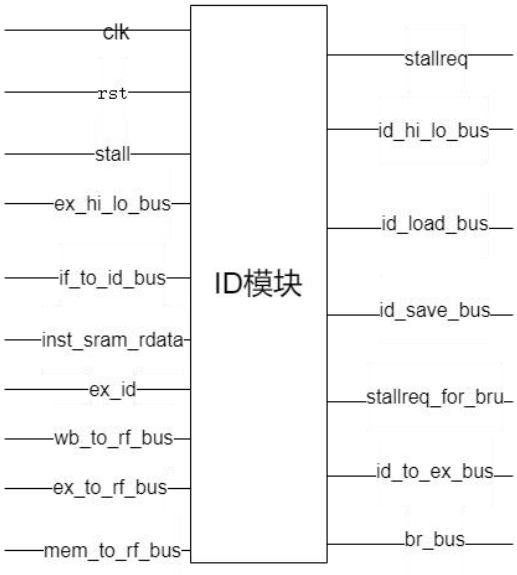
表 2 ID 模块输入输出

序号	接口名	宽度	输入/输	作用
1	clk	1	输入	时钟信号
2	rst	1	输入	复位信号
3	stall	6	输入	暂停信号，控制指令是否暂停
4	stallreq	1	输出	暂停请求信号
5	if_to_id_bus	33	输入	IF 段到 ID 段的数据总线
6	inst_sram_rdata	1	输入	读写使能信号
7	ex_id	1	输入	写使能信号
8	wb_to_rf_bus	38	输入	WB 段存放进寄存器的数据
9	ex_to_rf_bus	38	输入	EX 段存放进寄存器的数据
10	mem_to_rf_bus	38	输入	MEM 段存放进寄存器的数据
11	ex_hi_lo_bus	66	输入	EX 段存放进 hilo 寄存器的数据的总
12	id_hi_lo_bus	72	输出	ID 段存放进 hilo 寄存器的数据的总
13	id_load_bus	5	输出	ID 段执行 load 命令的数据总线
14	id_save_bus	3	输出	ID 段执行 save 命令的数据总线
15	stallreq_for_bru	1	输出	执行 load 命令时的暂停请求
16	id_to_ex_bus	159	输出	ID 段到 EX 段的数据总线
17	br_bus	33	输出	分支跳转信号，控制延迟槽是否跳转

功能说明：

ID 段的执行比较复杂，下面我们分成几部分来分别细述。

第一部分 流水线暂停的判断与实现



ID 段收到来自 CTRL 模块的 stall 值后会判断 stall 的值，当对应 ID 段的部分的 stall 的值为 0 时，也就说明没有流水线暂停，那就将将 IF 段传给 ID 段的 if_to_id_bus 正常地赋值给 if_to_id_bus_r，然后就可以进行接下来的译码、取操作数的部分。但是如果判断到 stall 的值的对应的 ID 段的部分是 1，也就是说此刻发生了访存冲突，需要读取的寄存器中的值还没有获得，还需要在下一个周期才可以从内存中读取出来，且无法通过数据前递解决，则现在需要对流水线的 ID 段进行暂停一个周期，在下个周期获得到需要读取的值后再发给 ID 段。当判断到暂停后就将 if_to_id_bus_r 置为 0，本周期停止，下个周期再恢复正常。由于 if_to_id_bus 不包含指令值，指令的

值即 inst 值是在 ID 段时根据上个周期的 IF 段中的 pc 值从内存中读取到的，是直接从内存获取的，因此 inst 值并没有被置为 0，因此还需要把当前时刻的 inst 值保存一个周期，下 一个周期再使用当前周期的 inst 值，这样才能保证 ID 段和之后所有部分的指令的 pc 值和 inst 值是相互匹配的。

第二部分 指令的译码

一般指令：先依据指令中的特征字段区分指令，同时激活相应的指令对应的 inst_**变量，表示是哪一条指令。根据译码结果，读取通过 regfile 模块读取地址为 rs（inst[25;21]）以及地址为 rt(inst[20:16]的通用寄存器,得到 rdata1 以及 rdata2，并且通过判断是否发生数据相关，从而更改 rdata1 以及 rdata2 的值。同时分析要执行的运算，给对应的 ALU 标识符赋值，其中，0 表示该条指令不采用该 ALU，1 表示采用该 ALU，同时将所有的 ALU 标识符组合起来成为 alu_op，alu_op 为十二位宽，代表 16 种不同的 ALU，并且作为传入 EX 段的一部分。要写入的目的寄存器。rf_we 代表写使能信号，表示该条指令是否用写入通用寄存器，sel_rf_dst[0]表示该指令要将计算结果写入 rd 通用寄存器，sel_rf_dst[1]表示该指令要将计算结果写入 rt 通用寄存器,sel_rf_dst[2]表示该指令要将计算结果写入 31 号通用寄存器。rf_waddr 表示要该条指令的计算结果要写入的通用寄存器的地址，data_ram_en 表示该条指令是否要与内存中取值或者写入值，如果该条指令要从内存中取值或者写入值，那么它将被赋值为 1’b1,data_ram_wen 为四位宽，表示该条指令是否要写入寄存器，如果该条指令要将计算结果的第几个字节写入寄存器，那么对应位置的值设为 1。

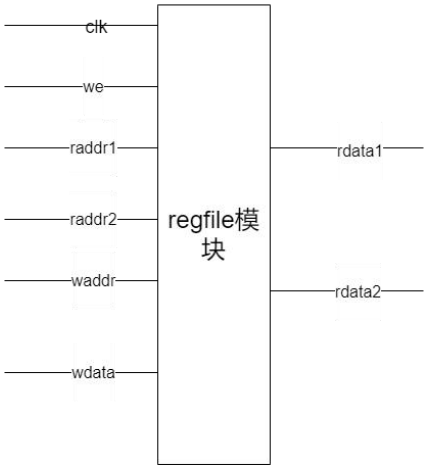
跳转指令：先用 br_e 表示这条指令是否为跳转指令，用 rs_ge_z 表示是否满足 rdata1 的值大于等于 0,用 rs_le_z 表示是否满足 rdata1 的值小于等于 0，用 rs_lt_z 表示是否满足 rdata1 的值小于 0，rs_eq_rt 表示是否满足 rdata1 是否等于 radta2 的值。br_addr 表示跳转后的地址，根据不同的指令对地址做不同的计算，并将结果赋给 br_addr。

第三部分 判断操作数来源

用 sel_alu_src1 和 sel_alu_src2 来判断操作数来源，第一个操作数有三种来源，第二个操作数有四种来源，通过区分不同的指令进而分辨不同的操作数的来源。

传值用 ID 段得到的数据，分别给 id_to_ex_bus 和 br_bus 赋值，其中 br_bus 是传给 IF 段的用于传输跳转指令的判断信号和跳转的地址。

regfile 模块接口如图所示。



regfile 模块说明：

regfile 模块的作用是确定 rs 寄存器以及 rt 寄存器的值，判断 raddr1 是否为零，如果为零，就 把 32’b0 赋值给 rdata1,如果不为零，就把 raddr1 对应的寄存器的值赋值给 rdata1; 判断 raddr2 是否为零，如果为零，就把 32’b0 赋值给 rdata2,如果 不为零，就把 raddr2 对应的寄存器的值赋值 rdata2。

表 3 regfile 模块输入输出

序号	接口名	宽度	输入/输出	作用
1	clk	1	输入	时钟信号
2	we	1	输入	寄存器的写使能信号
3	raddr1	5	输入	读取的第一个数的地址
4	raddr2	5	输入	读取的第二个数的地址
5	rdata1	32	输出	读取的第一个数的值
6	rdata2	32	输出	读取的第二个数的值
7	waddr	5	输入	写入的地址
8	wdata	32	输入	写入寄存器的值

2.3 EX 模块

整体说明：

执行运算、计算地址和计算 ALU 结果。从 ID/EX 流水线寄存器中读取由寄存器 1 传来的值和寄存器 2 传来的值（或寄存器 1 传来的值和符号扩展过后的立即数的值），并用 ALU 将它们相加，结果值存入 EX/MEM 流水线寄存器。其中 ALU 模块已经提供，基本通过给 alu 提供控制信号就可以完成逻辑和算术运算，对于需要访存的指令在此段发出访存请求。接口如右图所示。

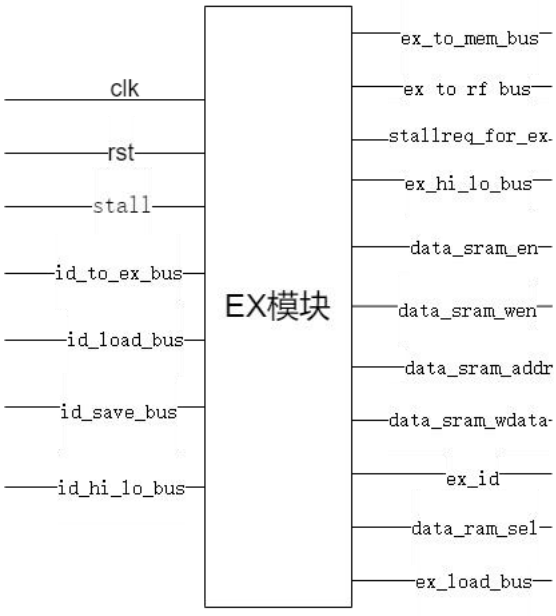


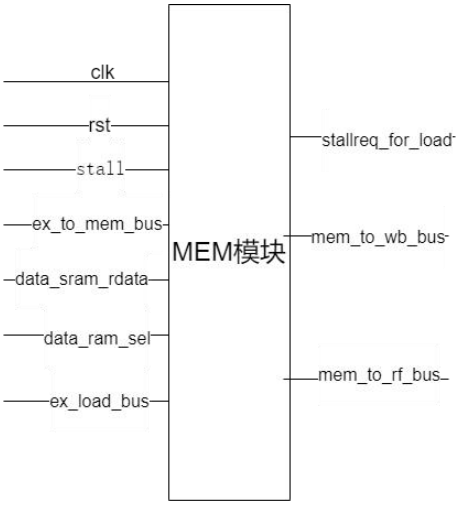
表 4 EX 模块输入输出

序号	接口名	宽度	输入/输出	作用
1	clk	1	输入	时钟信号
2	rst	1	输入	复位信号
3	stall	6	输入	控制暂停信号
4	id_to_ex_bus	169	输入	ID 段传给 EX 段的数据
5	id_load_bus	5	输入	ID 段传递读的数据
6	id_save_bus	3	输入	ID 段传递写的数据
7	ex_to_mem_bus	80	输出	EX 段传给 MEM 段的数据
8	ex_to_rf_bus	38	输出	EX 段传给 regfile 段的数据
9	id_hi_lo_bus	72	输入	ID 段传给 hilo 段的数据
10	ex_hi_lo_bus	66	输出	EX 段传给 hilo 段的数据
11	stallreq_for_ex	1	输出	对 EX 段的 stall 请求
12	data_sram_en	1	输出	内存数据的读写使能信号
13	data_sram_wen	4	输出	内存数据的写使能信号
14	data_sram_addr	32	输出	内存数据存放的地址
15	data_sram_wdata	32	输出	要写入内存的数据
16	ex_id	38	输出	EX 段传给 ID 段的数据
17	data_ram_sel	4	输出	内存数据的选择信号
18	ex_load_bus	5	输出	EX 段读取的数据

功能说明：

MEM 模块，该模块有 18 个输入输出端口，包括 clk，rst，stall，id_to_ex_bus，id_load_bus，id_save_bus，id_hi_lo_bus 等 7 个输入端口和 ex_to_mem_bus，ex_to_rf_bus，ex_hi_lo_bus，stallreq_for_ex，data_sram_en，data_sram_wen，data_sram_addr，data_sram_wdata，ex_id，data_ram_sel，ex_load_bus 等 11 个输出端口，该模块还包含乘除法的部分实现。

2.4 MEM 模块



整体说明：

执行访问内存操作，从 EX/MEM 流水线寄存器中得到地址读取数据寄存器，并将数据存入 MEM/WB 流水线寄存器。接收并处理访存的结果，并选择写回结果对于需要访存的指令在此段接

收访存结果。接口如上图所示。

表 5 MEM 模块输入输出

序号	接口名	宽度	输入/输出	作用
1	clk	1	输入	时钟信号
2	rst	1	输入	复位信号
3	stall	6	输入	控制暂停信号
4	ex_to_mem_bus	80	输入	EX 传给 MEM 段的数据
5	data_sram_rdata	32	输入	从内存中读出来要写入 寄存器的
6	data_ram_sel	4	输入	内存数据的选择信号
7	ex_load_bus	5	输入	EX 段读取的数据
8	stallreq_for_load	1	输出	对 EX 段的 stall 请求
9	mem_to_wb_bus	70	输出	MEM 传给 WB 段的数据
10	mem_to_rf_bus	38	输出	MEM 段传给 regfile 段的数据

功能说明：

MEM 模块，该模块有 10 个输入输出端口，包括 clk、rst、stall、ex_to_mem_bus、ex_load_bus 作为输入，并将 mem_to_wb_bus 和 mem_to_rf_bus 作为输出。它能够完成 load 和 store 指令的执行，处理 lb、lbu、lh、lhu、lw，store 指令 sb、sh 等指令。对于 load 指令，它会根据地址最低两位来确定字节选择，并将结果写入 RF 寄存器；而 store 指令则会根据地址最低两位来确定字节写使能，并将数据写入数据 RAM。

2.5 WB 模块

整体说明：

将结果写回寄存器，从 MEM/WB 流水线寄存器中读取数据并将它写回图中部的寄存器堆中。接口如右图所示。

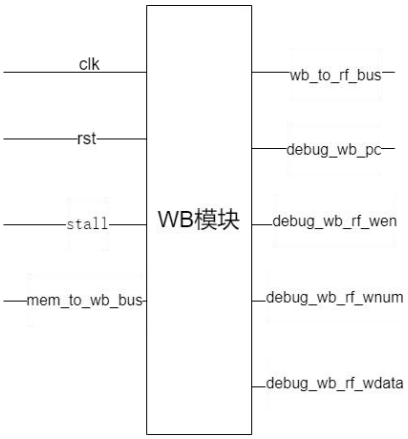


表 6 WB 模块输入输出

序号	接口名	宽度	输入/输出	作用
1	clk	1	输入	时钟信号
2	rst	1	输入	复位信号
3	stall	6	输入	控制暂停信号
4	mem_to_wb_bus	70	输入	MEM 传给 WB 的数据
5	wb_to_rf_bus	38	输出	WB 传给 rf 的数据
6	debug_wb_pc	32	输出	用来 debug 的 pc 值
7	debug_wb_rf_wen	4	输出	用来 debug 的写使能信号
8	debug_wb_rf_wnum	5	输出	用来 debug 的写寄存器地址
9	debug_wb_rf_wdata	32	输出	用来 debug 的写寄存器数据

功能说明：

WB 模块，该模块有 9 个输入输出端口，包括时钟（clk），复位（rst），stall，mem_to_wb_bus，以及调试信号。输出包括 wb_to_rf_bus，以及 debug 用的调试信号。

该模块的主要功能是从 MEM/WB 流水线寄存器中读取数据，并将它写回图中部的寄存器堆中。它首先定义了一个名为 mem_to_wb_bus_r 的寄存器，用于存储从 MEM/WB 流水线寄存器中读取的数据，然后将数据写回寄存器堆中。最后，它将写回的数据通过 wb_to_rf_bus 输出，以便在图中部的寄存器堆中更新数据。

2.6 CTRL 模块

整体说明：

接收各段传递过来的流水线请求信号，从而控制流水线各阶段的运行。接口如右图所示。

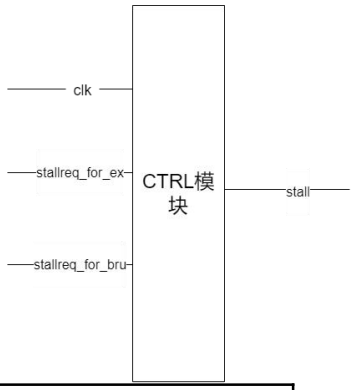


表 7 CTRL 模块输入输出

序号	接口名	宽	输入/输出	作用
1	clk	1	输入	时钟信号
2	stallreq_for_ex	1	输入	执行阶段的指令是否请求流水线暂停
3	stallreq_for_bru	5	输入	Load 命令是否请求流水线暂停
4	stall	6	输出	暂停信号

功能说明：

假设位于流水线第 n 阶段的指令需要多个周期，进而请求流水线暂停，那么需要保持取指令地址 PC 不变，同时保持流水线第 n 阶段及之前的各个阶段的寄存器保持不变，而第 n 阶段后面的指令继续运行。stall[0]为 1 表示没有暂停，1-5 为 1 时分别 代表 if 段、id 段、ex 段、 mem 段、wb 段暂停。

2.7 HILO 寄存器模块

整体说明：

hi 和 lo 属于协处理器，不在通用寄存器的范围内，这两个寄存器主要是在用来处理乘法和除法。以乘法作为示例，如果两个整数相乘，那么乘法的结果低位保存在 lo 寄存器，高位保存在 hi 寄存器。 当然，这两个寄存器也可以独立进行读取和写入。读的时候，使用 mfhi、mflo；写入的时候，用 mthi、mtlo。

和通用寄存器不同，mfhi、mflo 是在执行阶段才开始从 hi、lo 寄存器获取数值的。写入则和通用寄存器一样，也是在写回的时候完成的。

可以直接改写 lib 下的 regfile.v，也可以添加 hiloreg.v，创建 u_hi_lo_reg，但是 MEM、WB 也要跟着改，这里我们采用第二种方法，即添加 hiloreg.v 文件。接口如右图所示。

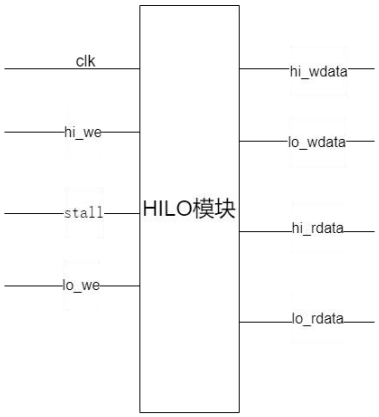


表 8 HILO 寄存器输入输出

序号	接口名	宽度	输入/输出	作用
1	clk	1	输入	时钟信号
2	stall	6	输入	控制暂停信号
3	hi_we	1	输入	hi 寄存器的写使能信号
4	lo_we	1	输入	lo 寄存器的写使能信号
5	hi_wdata	32	输出	Hi 寄存器写的数据
6	lo_wdata	32	输出	Lo 寄存器写的数据
7	hi_rdata	32	输出	Hi 寄存器读的数据
8	lo_rdata	32	输出	Lo 寄存器读的数据

功能说明：

当 hi_we 和 lo_we 均为 1 时，寄存器 reg_hi 和 reg_lo 同时将 hi_wdata 和 lo_wdata 写入。当 hi_we 为 0，lo_we 为 1 时，reg_lo 将 lo_wdata 写入；当 hi_we 为 1，lo_we 为 0 时，reg_hi 将 hi_wdata 写入。hi_rdata 和 lo_rdata 分别输出 reg_hi 和 reg_lo 中的数据。

3. 实验感受及建议

3.1 刘小龙部分

在实验中我熟练的掌握了 GitHub 的使用，能够用它搭建仓库、审阅代码、管理版本，极大的提高了我们的工作效率。

在 debug 时，我们在波形图添加可能有问题的数值，查看提示 pc 值附近目标的波形图。这使我得以定位出错的位置及原因。

3.2 王怡正部分

本次实验我深入了解了流水线的整体运行过程，把课堂中学习到的知识真正代入到了实践中。

本次实验虽然我负责的任务可能不多，但是还是需要把所有代码的整体运行逻辑搞懂，真正了解流水线每一步的具体运行，才能在其中插入相关指令，有很多不懂的地方都需要上网上查阅相关资料才能实现。

这次实验同时也让我明白了团队合作的重要性，要想使任务完成的更加成功，必须要分工明确并且和队友多多交流，体现出团队的价值。

总之，这次实验让我了解并学习了一门新的编程方法，更加深入的探究了流水线的运行逻辑和具体细节，领悟到了团队的强大力量。

3.3 李鸣骁部分

首先是关于实验内容的感悟，本次实验采用和之前完全不同的平台和内容，考核方式也是别出心裁，这些都使我学会了很多新技能、新方法。GitHub 的使用提高了小组代码同步的效率；CG 实验平台的使用避免了软件安装调试的麻烦；在实验 debug 过程中更是对 CPU 五级流水线的进一步熟悉与运用。

其次，是对小组合作的感悟。在实验过程中，我们小组分工明确，队友之间互帮互助，使我体会到了团队合作的重要性，同时，这也是我经历过最为高效的团队合作。

总之，这是一次十分愉悦的实验经历，在实验过程中不仅能体验到团队合作的有趣，更能体会到实验设计的用心，不论是新颖的实验方式还是老师助教的认真负责，这一切的一切构成了本次计算机系统实验，相信在今后我们也能更加优秀，收获更多。

4. 参考资料

1、张晨曦 著《计算机体系结构》（第二版） 高等教育出版社

2、雷思磊 著《自己动手写 CPU》 电子工业出版社

3、（美）DavidA.Patterson、John L.Hennessy 著 《计算机组 成与设计：硬件、软件接口（原书第 4 版）》

4、Yale N.Patt 著 《计算机系统概论（原书第 2 版）》

5、龙芯杯官方的参考文档