实验报告成绩:	成绩评定日期:

2024～2025 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 2202

组长：张恒

组员：王丹烁、张馨文

报告日期：2025.1.5

目录

一、设计分工

组长学号及姓名： 20226451 张恒 **工作占比：**50%

分工：参与实现通路链接，处理数据相关，拓展 stall 功能，增加 HILO 寄存器实现移动指令功能，处理访存操作，完善加载存储指令的实现，编写实验报告。

组员 1 学号及姓名：20226470 王丹烁 **工作占比：**35%

分工：主要负责实现 stall 功能，增加延迟槽来管理指令，实现部分移动指令，编写实验报告。

组员 2 学号及姓名：20226488 张馨文 **工作占比：**15%

分工：参与完成数据相关链路连接，添加全部指令定义，编写实验报告。

二、总体设计

首先实现 EX、MEM 段到 ID 段的连线，数据前推解决部分数据相关问题。接着添加全部指令的简单定义，根据仿真错误进行相关指令的添加，增加部分的跳转指令、分支指令和加载存储指令的相关实现。最后实现 stall 功能处理 load 相关，增加 HILO 寄存器对乘除法以及移动指令进行处理，完善加载存储指令的使用方式，最终通过 8’d64。

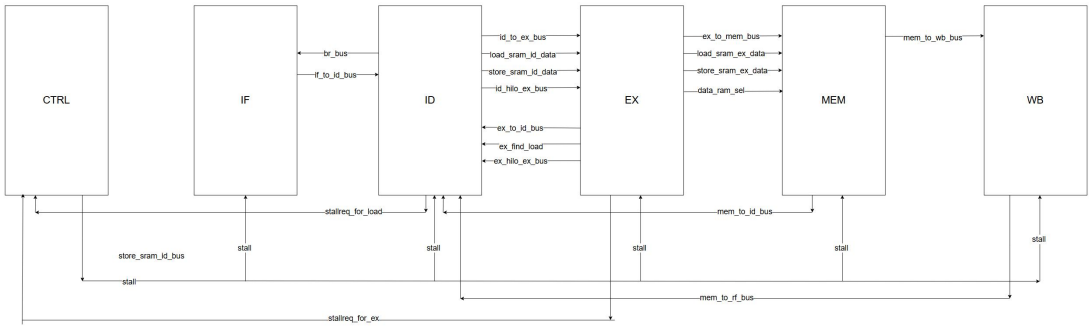

图 1. 流水线之间的连线

三、实验环境

操作系统：Windows11

仿真软件：Vivado 2023.1

编程工具：Visual Studio Code，使用 git 进行版本管理

四、流水线各个阶段说明

1、IF模块

（1）整体功能说明

IF 模块的主要功能是处理指令获取过程。当系统复位时，PC 初始化为特定值。当未发生暂停（stall）状态时，模块会根据分支指令的状态更新 PC。如果有有效的分支指令，则将 PC 设置为分支地址，否则按照正常的顺序递增（PC + 4）。

（2）端口介绍

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
4	br_bus	33	输入	分支跳转信号
5	if_to_id_bus	33	输出	用于将控制信号（ce_reg）和当前 PC 值传递到 ID 模块
6	inst_sram_en	1	输出	指令存储器使能信号
7	inst_sram_wen	4	输出	指示是否写入数据以及写入哪些字节
8	inst_sram_addr	32	输出	指令存储器的地址输入
9	inst_sram_wdata	32	输出	要写入指令存储器的数据

2、ID模块

（1）整体功能说明

ID 模块的主要功能是处理从 IF 阶段传来的数据，将指令解码为控制信号和操作数，并根据数据相关性进行处理，同时实例化寄存器模块，维护通用寄存器和 HILO 寄存器的功能。

（2）端口介绍

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
4	stallreq	1	输出	暂停请求信号
5	if_to_id_bus	33	输入	来自 IF 模块的指令和控制信号总线
6	inst_sram_rdata	1	输入	从指令存储器读取的指令数据
7	wb_to_rf_bus	38	输入	来自写回阶段的数据，包含写回使能信号、写地址和写数据
8	ex_to_id_bus	38	输入	来自执行阶段的数据，包含数据相关的控制信息
9	mem_to_id_bus	38	输入	来自内存阶段的数据，包含数据相关的控制信息
10	load_sram_id_data	5	输出	与加载指令相关的输出数据，表明需要从内存读取哪些数据
11	store_sram_id_data	3	输出	与存储指令相关的输出数据，表明需要向内存写入哪些数据
12	ex_find_load	3	输入	指示执行阶段是否有加载操作
13	stallreq_for_load	1	输出	处理 load 相关加入暂停
14	id_hilo_ex_bus	72	输出	与 HI 和 LO 寄存器相关的输出
15	ex_hile_id_bus	66	输入	来自执行阶段的 HI 和 LO 寄存器值
16	id_to_ex_bus	159	输出	包含指令和 ALU 控制数据，将数据提供给 EX 模块
17	br_bus	33	输出	分支跳转信号，控制延迟槽是否跳转

（3）具体实现

在此模块主要针对四个问题进行解决：

1）最开始在波形图中发现 id_to_ex_bus 的 rdata 部分为 X 态，说明并没有对其正确的赋值，经过层层向前查找，发现当前指令处在 ID 段，需要访问内存中的数据，但是前一条指令正处在 EX 段，这里还发现 regfile 模块在 ID 段根据 WB 段传来的数据进行模块例化操作，因此需要等到前一条指令在 WB 写回寄存器，当前指令才可以读取指令。在这里发现了数据相关问题，也明白了为什么要先完成 EX、MEM 到 ID 的连线。

解决方法：增加两条从 EX、MEM 段中传入的线路，具体实现参考 wb_to_rf_bus，在 ID 段中对从寄存器中读取数据的 rdata 进行处理，定义 tdata 临时数据用来存储 EX、MEM、WB 段的要写入寄存器的值，最后用 tdata 代替 rdata 进行数据的传输，避免数据相关问题。

2）此时遇到对应的 PC 值不匹配，发现问题出在 load，store 这几个指令，在 ID 段中已经定义这些指令，也就是说应当增加 EX、MEM 中有关 sram 读取的功能，因此我需要增加到这两个模块的连线（load_sram_id_data、store_sram_id_data），将 load，store 的信息传递下去。这里发现，项目中所有有

关 sram 字样的变量可以当成已知得到，这些变量的值都是通过内存直接读取，无需关注这些变量的来源。

解决方法：增加两条线（load_sram_id_data、store_sram_id_data），将有关 load 和 store 的操作都打包传递给 EX、MEM 段，在这两段进行有关 load 和 store 操作。

3）发现 load 相关的问题，前一条为 load 指令，当前指令要是访问 load 指令指向的寄存器，会在波形图的 ID 段中的数据发现，此时的 load 指令处在 EX 阶段，还没有读取内存中的值，也没有写回寄存器，这里出现了 load 相关。

解决方法：插入暂停操作，需要在 EX 段中根据 ID 段传过去的有关 load、store 数据判断出是否设计 load 或者 store 操作，然后将数据再通过 ex_find_load 从 EX 段中传回 ID 段，在 ID 段中判断两个寄存器是否与前一条指令的写回地址相同并更改 stallreq_for_load，传回 CTRL 模块进行暂停。这里还要注意对暂停的指令要放入到延迟槽中进行处理并更改 inst 读取的逻辑，不然会发现暂停结束后读取的指令有问题。在解决这一问题的过程中，又发现 PC 值对应错误，经过不断试错（痛苦的过程），找到一个解决方法，即在 test.s 的汇编指令中根据最开始的指令一步一步手算 PC，终于发现某条跳转指令有问题导致波形图中的 PC 和手算正确的 PC 值不同，debug 之后顺利运行。

4）定义好 hilo_reg 模块后，需要在当前 ID 模块实例化一个 hilo 模块，而实例化需要的数据需要在 EX 段中进行乘除法或者移动指令进行判断得到，而在 EX 段中又需要 ID 段中的译码出来的乘除法操作或者移动指令操作来进行处理，根据一个问题逐步思考，又发现回到了原地，参考很多资料后实现了这一部分。

解决方法：利用 id_hilo_ex_bus 和 ex_hilo_id_bus 这两条线解决上述的问题，这里以 mfhi 和 mflo 两条指令进行分析，对于 mflo、mfhi 两条指令，需要将 hi lo 的值在 ID 段读入并传送到 EX 段，并写回通用寄存器中，此时就需要将 ex_result 表示为 hi lo 的值，因此需要利用 id_hilo_ex_bus 传到 EX 段的数据。在 EX 段中进行乘除法操作时，将乘法的高 32 为写入 hi，低 32 位写入 lo，除法将余数写入 hi，将商写入 lo，这时就出现了数据相关问题，需要将此时计算的结果通过 ex_hilo_id_bus 传回到 ID 段，确保 ID 段得到最新的值。

3、EX模块

（1）整体功能说明

对 ID 段译码后的信号以及操作数等信息，在 ALU 中进行计算

（2）端口介绍

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
4	id_to_ex_bus	169	输入	ID 段传给 EX 段的数据
5	ex_to_mem_bus	76	输出	传递给 MEM 阶段的数据，包括 ALU 结果、内存使能信号
6	ex_to_id_bus	38	输出	用于处理数据相关
7	load_sram_id_data	5	输入	来自 ID 阶段的加载指令相关数据
8	store_sram_id_data	3	输入	来自 ID 阶段的存储指令相关数据
9	load_sram_ex_data	5	输出	需要从内存加载的数据
10	store_sram_ex_data	3	输出	需要保存到内存的数据
11	ex_find_load	1	输出	用于说明是否存在加载指令的信号
12	data_ram_sel	4	输出	内存写入的字节选择信号
13	id_hilo_ex_bus	72	输入	ID 和 EX 间的关于 HILO 寄存器数据
14	ex_hilo_id_bus	66	输出	ID 和 EX 间的关于 HILO 寄存器数据
15	stallreq_for_ex	1	输出	请求除法暂停
16	data_sram_en	1	输出	与内存交互的信号
17	data_sram_wen	4	输出	指示是否写入数据以及写入哪些字节
18	data_sram_addr	32	输出	指令存储器的地址输入
19	data_sram_wdata	32	输出	要写入指令存储器的数据

（3）具体实现

在此模块中同样根据 ID 段中发现的四个问题进行处理，解决方法有很多重复地方，不再赘述。

1）处理数据相关，参考 wb_to_rf 的实现。

2）关于 load 和 store 的操作，在这一模块中进行判断，通过 ex_find_load 返回到 ID 段来处理 load 相关。由于在加入全部指令时增加了对 lb，lh，lw 这些指令需要位数的判断，在增加 HILO 寄存器后，通过波形图查找到问题所在即可，这里的 lb，lh，lw 这些读取相关位数的问题，参考文档中的实现。

3）对乘除法的操作，将其对应的位数拆分写到 HILO 寄存器中，判断乘法操作数是否是无符号操作，加入除法的暂停指令。完成 ID 段有关 HILO 数据的打包传输。

4、MEM模块

（1）整体功能说明

主要功能是执行内存相关的操作。接收来自 EX 阶段的数据，执行必要的存储或读取操作，并将结果传递给 WB 阶段。还处理数据相关性，以确保不会出现数据相关问题。

（2）端口介绍

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
4	ex_to_mem_bus	76	输入	包含需要执行的内存操作的信息
5	mem_to_id_bus	38	输出	传递给 ID 阶段的数据，用于处理数据相关
6	load_sram_ex_data	5	输入	来自 EX 阶段的加载指令的数据
7	store_sram_ex_data	3	输入	来自 EX 阶段的存储指令的数据
8	data_ram_sel	3	输入	EX 阶段生成的内存字节选择信号
9	data_sram_rdata	32	输入	从内存中读取的数据
10	mem_to_wb_bus	70	输出	传递给 WB 阶段的数据，包含写回寄存器的信息

（3）具体实现

MEM 段中主要实现了数据前推解决数据相关问题以及 load 和 store 对不同字节的访问情况，然后扩展成 32 位写回寄存器中。具体看 Q&A 即可实现。

5、WB模块

（1）整体功能说明

主要功能是写回寄存器，输出调试信息。

（2）端口介绍

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

4	mem_to_wb_bus	70	输入	来自内存阶段的数据总线，包含写回所需的信息
5	wb_to_rf_bus	38	输出	处理数据相关
6	debug_wb_pc	32	输出	调试
7	debug_wb_rf_wen	4	输出	调试
8	debug_wb_rf_wnum	5	输出	调试
9	debug_wb_rf_wdata	32	输出	调试

（3）具体实现

WB 段中主要实现了数据前推解决数据相关问题。

五、实验感受及改进意见

5.1 组长-张恒

在本次实验中，我熟悉了 MIPS 五级流水线的执行过程、Verilog 语言的编写、Vivado 的仿真调试，也熟练掌握了使用 git 对代码进行版本管理，这对我完成整个项目来说起到了非常大的作用。完成整个项目的过程中，无疑是大大提升了我的计算机系统的能力，虽然最后回顾发现实验并不是很难，参考教材中的讲解加网络上很多的开源代码就可以很快完成，但从 0 到 1 的突破对我来说是对自己能力的肯定以及拓宽了我在计算机领域的视野。实验入手很难，花了大量的时间弄明白如何通过波形图调试，明白如何完成不同模块之间的连线，找到问题所在和修改方向，光是自学入门并通过第一个点，就已经用了三节课时加很多课余时间，让我很激动也很有成就感，但是花费的时间和带给我的收获是否成正比？

总体来说，实验过程很有趣，也顺带复习了一下期末（但不多），以通过点数为目标的确更加激励我完成实验，逐层递进实现相关的指令让我越来越想探索并解决接下来的可能会遇到的问题。

改进意见：希望可以给出一个比较详细的入门教学，比如如何利用《自己动手做 CPU》发现问题解决问题，身边好多人都是因为看到 500 多页的 pdf 就打退堂鼓了，但其实对实验有帮助的可能就 100 多页。

5.2 组员-王丹烁

首先对于实验内容的体会，这次实验采用了与以往截然不同的平台和内容，其独特的考核方式让我学习到了许多新技能和方法。使用 GitHub 显著提升了小组间代码同步的效率，而 CG 实验平台的引入则免去了软件安装和调试的烦恼。在调试过程中，我也对 CPU 的五级流水线有了更深入的熟悉和应用。 其次，我对小组合作的认识也有了新的提升。在实验期间，我们的小组分工明确，队友之间相互帮助，使我深刻体会到团队合作的重要性。这也是我经历过的最为高效的

团队协作之一。 总的来说，这次实验经历十分愉快。在实验过程中，我不仅享受到了团队合作的乐趣，还能领略到精心设计的实验方案。不论是新颖的实验方式助教的认真负责，这一切都构成了这次计算机系统实验的成功。我相信，未来我们能够更加优秀，收获更多的成果！

5.3 组员-张馨文

在这次实验中，虽然我负责的任务相对较少，但我意识到必须深入理解所有代码的整体运行逻辑，只有完全掌握流水线每个步骤的具体运作，才能有效地在其中加入相关指令。为了实现这一目标，我常常需要上网查阅资料，以解答其中的疑惑。 这次实验也让我深刻体会到了团队合作的重要性，为了确保任务的顺利完成，明确的分工和与队友的频繁交流是至关重要的，这也充分展示了团队合作的价值。 总而言之，这次实验让我接触到了一种新的编程语言，让我更加深入地探讨了流水线的运行逻辑和具体细节，也让我领悟到团队的力量是多么强大。

六、参考文档

1、雷思磊 著《自己动手写 CPU》 电子工业出版社