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 2002

组长：赵润松

组员：谭又僮

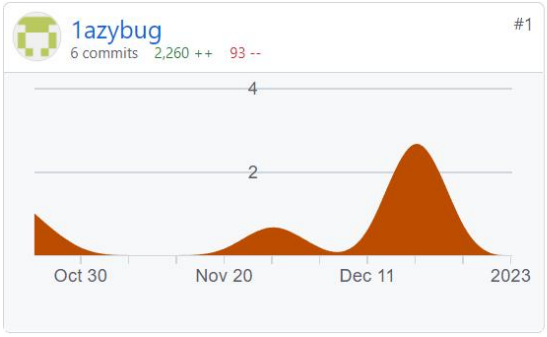
报告日期：2023.1.1

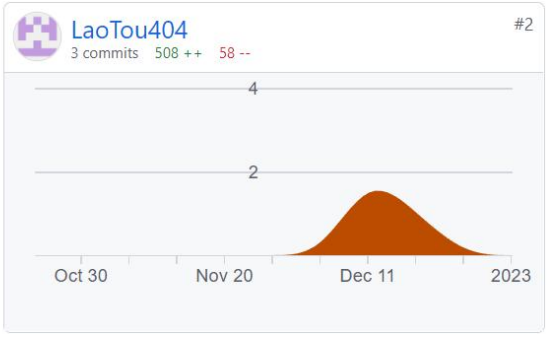
目录

1. 工作量

1.1 比例

赵润松（1azybug：60%）；谭又僮（LaoTou404：40%）





1.2 具体工作

Commit 1-3

添加助教代码

Commit 4（由谭又僮完成）

解决了数据相关的 bug，并添加了 addu。

结构上：

从 EX 段到 ID 段接了一条 bus,MEM 段到 ID 段接了一条 bus,WB 段到 ID 段接了一条 bus、在顶层添加了对应接口、在 define 里加了几条定义。时间太久了，有点记不清，差不多这些。

跑仿真发现没用，错误还是在初始位置。

添加了 rf_data1、rg_data2 来接收数据，解决 bug 从而完成数据相关。

添加了 addu 来测试一下数据相关的正确性。

Commit 5（由谭又僮完成）

添加了 jal,jr bfc00704 to 9fc06348

因为不知道一号点到底要到达什么程度才能过，就一直在添加指令，错误由 bfc00704 提高到了 9fc06348。

Commit 6（由赵润松完成）

添加了 jalr、or、sll、lw、xor、sw；通过一号点；加入了 ID 段的 stall 请求；bus 上添加了 sl_bus，用于传递 Store 和 Load 指令信息。

结构上：

从EX段到ID段接了一条loading线，提示EX段在进行Load操作。

从ID段到CTRL接了一条stallreq线，要求暂停IF段和ID段。

从ID段到EX段和从EX段到MEM段的bus上添加了一个sl_bus，同时在EX段和MEM段增加对sl_bus的解析，作用是向后传递Store和Load指令信息

其他：

在CTRL根据stallreq给stall赋值。

在ID段添加了当stall时让inst保持不变的代码

当EX段是Load指令时，且EX段写入寄存器的值和ID段的rs/rt相同时，要让IF段和ID段暂停，因为EX段计算的是地址，要在MEM段把数据定向到ID段。

Commit 7（由谭又僮完成）

add sltu pass 1 to 7;

add slt pass 8;

add slti pass 9;

add sltiu pass 9 to 12;

add j pass 12 to 15;

add add，addi，sub，and，andi，nor，xori，sllv,sra,srav,srl,srlv pass 15 to 36。

Commit 8（由赵润松完成）

添加了bgez、bgtz、blez、bltz、bltzal、bgezal指令，通过43号点。

Commit 9（由赵润松完成）

添加了div、mflo、mfhi、divu、mult、multu、mthi、mtlo指令；增加了hilo寄存器；增加了EX段的stall请求；bus上添加了hilo_bus，用于传递与hilo寄存器相关的指令信息。解决了引入hilo寄存器的数据相关，通过了58号点。

hilo寄存器的输入线接在EX段，输出线接在ID段，因此解决数据相关只需要拉一条EX段到ID段的定向即可。

Commit 10（由赵润松完成）

添加了 lb、lbu、lbu、lh、lhu、sb、sh；通过 64 号点。

其他

match 指令（由赵润松完成）

在 alu 里添加了 match 计算单元，在 alu_op 里添加了 op_match，完成了对 match 指令的接线。

AXI 接口封装（由谭又僮尝试）

尝试了一下，没什么进展，问了一下助教，短时间做不完，就放弃了。

2. 总体设计及连线：

在助教代码基础上增加了以下连线（连线作用见 1.2）：

数据相关：

添加了 EX 段到 ID 段的定向

添加了 MEM 段到 ID 段的定向

添加了 WB 段到 ID 段的定向

EX 段到 ID 段的定向添加了 hilo 寄存器的写入信息（解决 hilo 寄存器的数据相关）。

Stall 线：

从 EX 段到 ID 段接了一条 loading 线

从 ID 段到 CTRL 接了一条 stallreq 线

Bus 线：

从 ID 段到 EX 段和从 EX 段到 MEM 段的 bus 上添加了一个 sl_bus

从 ID 段到 EX 段添加了一个 hilo_bus

hilo 寄存器:

hilo 寄存器的输入线接在 EX 段，输出线接在 ID 段

3. 完成的指令

subu、jal、jr、jalr、addu、bne、sll、or、sw、xor、lw、sltu、slt、slti、sltiu、j、add、addi、sub、and、andi、nor、xori、sllv、sra、srav、srl、srlv、bltz、blez、bgtz、bgez、bgezal、bltzal、divu、div、mfhi、mflo、mult、multu、mthi、mtlo、lb、lbu、lh、lhu、sb、sh、match 共 50 条指令。

4. 程序运行环境及使用工具

CG 平台提供的运行环境和使用工具

5. IF 段说明

从指令存储器中获取指令,并传给 ID 段，计算下一个 PC 值。

输入：

clk：时钟信号

rst: 复位信号

stall: 暂停信号

br_bus: 分支信号

输出：

if_to_id_bus 从 IF 段到 ID 段的 bus

Inst_sram_en 读取指令使能线

Inst_sram_wen 无用

Inst_sram_addr 需要读取的指令的地址

Inst_sram_wdata 无用

6. ID 段说明

对指令进行译码；用寄存器组（包括 hilo 寄存器）读取数据和写入 WB 段的数据（hilo 寄存器的数据在 EX 段传回）；选择 ALU 单元的两个操作数和操作；选择是否使能数据存储器，和可以写入的地方；选择是否可写入寄存器；选择存储的寄存器；向后传递信息，向 IF 段传递分支信息。

输入：

Clk：时钟信号

Rst：复位信号

Loading：EX 段是否执行 Load 指令

Stall：暂停信号

If_to_id_bus：IF 段到 ID 段的 bus

Ex_to_id_bus：EX 段定向到 ID 段的数据信息

Mem_to_id_bus：MEM 段定向到 ID 段的数据信息

Wb_to_id_bus：WB 段定向到 ID 段的数据信息

Inst_sram_rdata：指令存储器向 CPU 传递的指令

Wb_to_rf_bus：WB 段到寄存器的 bus

Hi_o：hilo 寄存器的 hi 寄存器输出

Lo_o：hilo 寄存器的 lo 寄存器输出

输出：

Stallreq：向 CTRL 发出 stall 请求

Id_to_ex_bus：ID 段到 EX 段的 bus

Br_bus：向 IF 段传递分支信号

7. EX 段说明

使用 ALU 模块算出结果，写入 hilo 寄存器，或传到 MEM 段，或写入数据存储器。

输入：

Clk：时钟信号

Rst：复位信号

Stall：暂停信号

Id_to_ex_bus：ID 段传过来的数据

输出：

Ex_to_mem_bus：EX 段到 MEM 段的 bus

Ex_to_id_bus：EX 定向到 ID 段的数据信息(包括 hilo 寄存器的信息)

Data_sram_en：数据存储器使能信号

Data_sram_wen：数据存储器写的地方（四个字节选若干个）

Data_sram_addr：数据寄存器使能的地址

Data_sram_wdata：要向数据寄存器写入的数据

Loading：向 ID 段说明 EX 段的是 Load 指令，若想定向需要 stall 一个时钟周期

Hilo_we：hilo 寄存器写的位置

Hi_i：写入 hi 寄存器的数据

Lo_i：写入 lo 寄存器的数据

Stallreq_for_ex：向 CRTL 发送 stall 请求

8. MEM 段说明

从存取器中读取数据

输入：

Clk：时钟信号

Rst：复位信号

Stall：暂停信号

Ex_to_mem_bus：EX 到 MEM 段的 bus

Data_sram_rdata：数据存储器向 CPU 发送的数据

输出：

Mem_to_wb_bus：MEM 段到 WB 段的 bus

Mem_to_id_bus：MEM 段定向到 ID 段的数据

9. WB 段说明

写回寄存器，输出调试信息。

输入：

Clk：时钟信号

Rst：复位信号

Stall：暂停信号

Mem_to_wb_bus：MEM 到 WB 段的 bus

输出：

Wb_to_rf_bus：WB 段到寄存器的 bus

Wb_to_id_bus：WB 段定向到 ID 段的数据

Debug_wb_pc：WB 段的 PC 值（调试用）

Debug_wb_rf_wen：WB 段的寄存器写能信号（调试用）

Debug_wb_rf_wnum：WB 段写入寄存器的地址（调试用）

Debug_wb_rf_wdata：WB 段写入寄存器的数据（调试用）

10. 组员的实验感受

10.1 赵润松的实验感受及改进建议

这次实验最大的收获就是把 github 玩熟练了，我们组在整个实验过程中严格按照 github 的工作流程提交，在 github 的使用上十分顺利。

在实验过程中，遇到了很多困难。在前期，不知道如何着手，虽然应该学习了 Verilog 语言，但代码不知道，评估方式不知道，模块之间的连线图没有。在

中期，下定决心花了两整天的时间把所有代码看了一遍，把《动手学 CPU》的原理看了一遍，终于懂要怎么做了。但是评估方式仍然是一团迷雾，为什么错了？为什么停在这个地方？这些问题困扰着我，往往花了好几个小时在那个波形图上面 debug，最后发现：没有 bug，继续往下加指令就能继续动。我在 ID 段加了 stall 保留原指令的操作之后，波形图就一直停不下来了，然后一直 debug，最后发现没有 bug，停不下来是因为之后的指令没加导致的。很多时间都花在了无意义的 debug 上，这是由于不清楚评估方式导致的。在后期，全都搞懂后，用远少于前面的时间就一直过到了 64 号点。可以说是 CPU 结构，数据存储器和 hilo 寄存器给我整明白了，突出一个会者不难。

虽然这次实验顺利结束，我也懂了很多，我却开心不起来。因为这些知识在我以后的学术生涯和工作生涯都用不上，我却花费了 57 个学时去做这件事，相当于一门多的课时了。如果我用这些时间去读论文多好。我最讨厌的事是为了绩点去学一样东西，更讨厌的是不得不这么做。

我的改进建议：可以考虑给不参加龙芯杯的同学准备另一条出路，把这个项目作为一个选做作业。另一方面，可以考虑给出示例代码，前期带一带同学们，等后期同学们懂了再自己上手，不然前期真的是没头苍蝇。

10.2 谭又僮的实验感受及改进建议

熟练掌握了 GitHub 的一些基础操作，可喜可贺可喜可贺，也对计算机系统有了更深刻的认识。所作即所得，从这段经历中收获很多，也很感谢老师和助教的帮助。

花了三天学 Verilog 语言，顺利做完了第一个小任务（count10...）。本来觉得没那么难。到了实验的时候，直接懵了，就不知道从哪下手，再加上我和赵润松轮流新冠，写起实验真的是折磨，甚至到了最后一次助教上实验课的时候我们组都没过一号点。最后痛定思痛，又认真看了一遍代码，找到了一些思路，最开始被数据相关卡住过，回看了助教的录屏后解决了。理清思路以后，过点速度直线上升，没几天就做完了。

总的来说，虽然老师和助教都很耐心和负责，但是由于前期引导不足（对于我是这样），导致遇到问题的时候我既不能完成实验的继续又很难提出问题向老师和助教请教，最后陷入死循环；而没有问题的时候，进度有飞快。当然，也可能是因为疫情的原因，打乱了老师和助教的计划，可以理解。还是要再次感谢一下老师和助教的帮助。

改进建议：我不知道以上我锻炼自己（折磨）的过程，是否是老师想要看到的。虽然从结果来看，我学到了知识，也完成了实验。但我希望我可以得到更多的前期引导。

11. 参考资料

课上 PPT

A03_“系统能力培养大赛”MIPS 指令系统规范_v1.01.pdf

《自己动手写 CPU》