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：钱名锐

组员：刘培栋

报告日期：2023.1.1

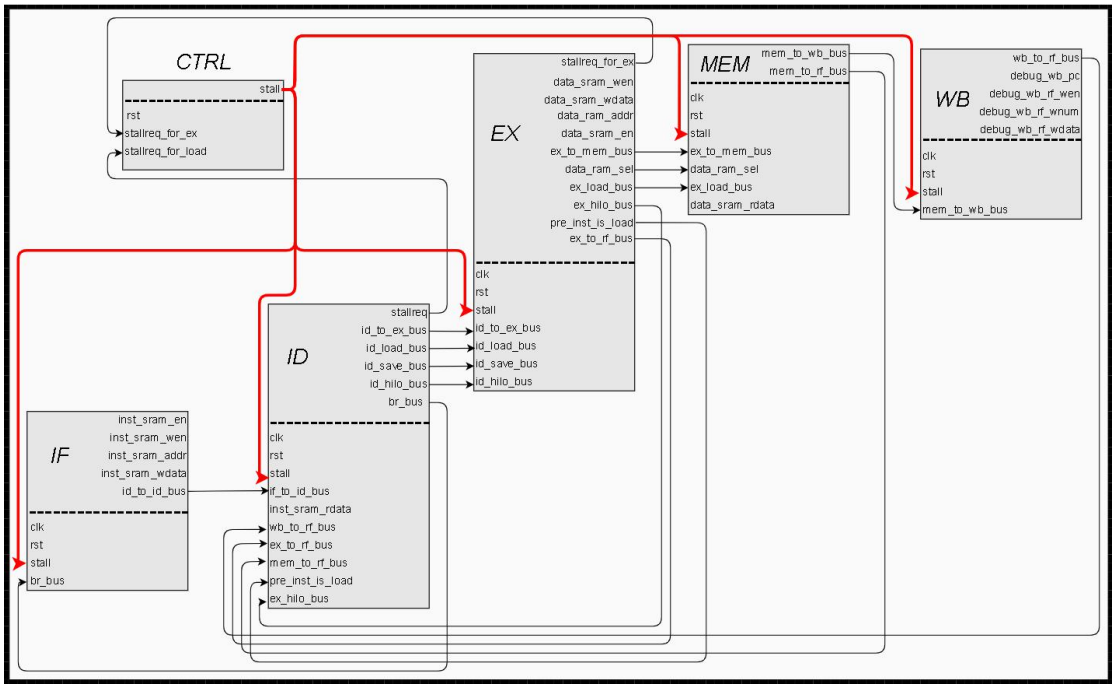
目录

1. 工作量说明

钱名锐：完成所有数据相关通路的处理、完成对所有算术运算指令（包括乘除法）的添加、完成所有数据移动指令、访存指令的添加、完成流水段气泡设计。工作量占比约为 60%。

刘培栋：完成所有移位指令、逻辑运算指令和 J 系与 R 系转移指令的添加。工作量占比约为 40%。

以下是流水段之间的连线图。在本次实验中，我组共完成了 14 条算术运算指令、8 条逻辑运算指令、6 条移位指令、12 条分支跳转指令、4 条数据移动指令和 8 条访存指令共计 52 条指令。另外，还设计了一条用于验收测验的指令 match，不过该条指令并未进行完整的测试。



本次实验使用 VSCode 作为文本编辑环境，配合 Vivado 仿真环境进行 CPU 仿真。

2. 流水段结构说明

2.1 IF 段

IF 段用于从存储器中提取指令与 PC 值，并将提取到的指令内容与 PC 值传入下个流水段。

IF 段中有四个输入端口和五个输出端口。

输入端口中 clk 为时钟信号端口，rst 为复位信号端口，stall 为流水线暂停信号，br_bus 为 ID 段中传来的转移信号及转移地址。

五个输出端口中，if_to_id_bus 输出较为重要，这是 IF 段传给 ID 段的所有数据信息，其中包含指令使能信号 ce_reg 和指令 PC 值 pc_reg。

IF 段中包含一个 next_pc 的选择器。选择器代码如下。

```
assign next_pc = br_e ? br_addr
                      : pc_reg + 32'h4;
```

此选择器在有 ID 段传入的转移信号时优先选择转移地址，如果没有转移信号，则将 PC+4，用以代表下一个指令的 PC 值。

2.2 ID 段

ID 段主要用于指令译码，发出转移使能信号和转移地址。

ID 段除了常规的复位信号、时钟信号输入和流水线暂停信号输入，还包含很多数据 bus 的输入。如从 IF 段传入的 if_to_id_bus、从 EX、MEM、WB 段分别传回的用以解决数据相关问题的 ex_to_rf_bus、mem_to_rf_bus 和 wb_to_rf_bus，其中 wb_rf_bus 不仅用以解决数据相关问题，还负责正常的寄存器数据写回。pre_inst_is_load 是由 EX 段传回的数据，用以负责判断当前 ID 段处理的指令的上一条是否是 load 指令，用以决定是否要暂停流水线解决数据相关问题。ex_hilo_bus 是从 EX 段传入的关于 hi 和 lo 寄存器的相关数据，也负责在 ID 段进行 hi 和 lo 寄存器的存取。

关于 ID 段的输出端口方面，除了常规的 id_to_ex_bus 将 ID 段的部分数据传到 EX 段以供使用以外，还存在 id_load_bus、id_save_bus 和 id_hilo_bus 这三个数据 bus，分别将 EX 段中可能会用到的 load 信息、save 信息和 hilo 信息传出。br_bus 是 ID 段解码后得出的转移信息，包括转移使能信号和转移地址，ID 段会将其传至 IF 段。最后一个输出是 stallreq，这个信号在 ID 段中用于判断是否存在 load 相关，如存在时，将流水线暂停一个时钟周期，该信号会由 ID 段发出，并传入 CTRL 段。

ID 段的结构相较于其他流水段复杂得多。首先，设置了一个指令内容选择器，目的在于当流水线暂停结束后，需要沿用原本的指令内容时，能够保存记录暂停之前处理的指令。

```
assign inst = ce ? flag ? pre_inst : inst_sram_rdata : 32'b0;
```

然后，ID 段会将获取的指令分段解码，得到 opcode、rs、rt、rd、sa、func、imm、base、offset、instr_index 等值。获取各段的值之后，将会通过 opcode、func、sa 等值确定当前处理指令具体是哪一条指令，并将该条指令使能。与此同时，将通过 regfile 模块对寄存器进行前面处理完的指令的寄存器写回、hilo 寄存器的更新与读取以及当前指令寄存器的读取。

有了指令时能信号之后，便能够获取当前指令的一些使能信号，例如指令中是否用到 alu 模块、两操作数分别来自哪里、是否需要写寄存器等等。这些使能信号都将通过 id_to_ex_bus 传入 EX 段进行后续操作。

为了处理 RAW 数据相关问题，仅仅获取当前指令的寄存器读取内容并不可行，因为当前读取的寄存器内容可能已经被更新，因此需要通过从后三流水段传回的 ex_to_rf_bus、mem_to_rf_bus、wb_to_rf_bus 与当前寄存器数据进行选择。选择器如下。

```
assign data1 = ((ex_rf_we && rs == ex_rf_waddr) ? ex_rf_wdata : 32'b0)
|

((!(ex_rf_we && rs == ex_rf_waddr) && (mem_rf_we && rs ==
mem_rf_waddr)) ? mem_rf_wdata : 32'b0) |

((!(ex_rf_we && rs == ex_rf_waddr) && !(mem_rf_we && rs ==
mem_rf_waddr) && (wb_rf_we && rs == wb_rf_waddr)) ? wb_rf_wdata : 32'b0)
|

(((ex_rf_we && rs == ex_rf_waddr) || (mem_rf_we && rs == mem_rf_waddr)
|| (wb_rf_we && rs == wb_rf_waddr)) ? 32'b0 : rdata1);
```

该选择器为第一个操作数的选择器，第二个操作数的选择器结构与之基本相同。该选择器在条件满足时优先选取 EX 段传回的数据 bus，其次选择 MEM 段传回的数据 bus，再次选择 WB 段传回的数据 bus，最后选择当前从寄存器中读取的数据。该选择器本身是以并行选择的形式写的，但是后面发现使用优先选取的方式会更为简单，只是已经使用这种并行形式完成了相关处理并能够成功运行，于是就没有去修改这个选择器了。

接下来还需要处理 load 指令的数据相关问题，当 load 指令的下一条指令需要用到 load 指令写回的寄存器的数据时，流水线需要暂停一个时钟周期来解决数据相关问题。

```
assign stallreq = (pre_inst_is_load & ex_rf_we & (rs == ex_rf_waddr
| rt == ex_rf_waddr)) ? 1'b1 : 1'b0;
```

数据移动指令中需要对 hi、lo 寄存器进行写回，也可能触发数据相关问题，这时需要对读取的 hi、lo 寄存器的值进行选择。

```
assign hi = hi_we ? hi_wdata : hi_r;

assign lo = lo_we ? lo_wdata : lo_r;
```

其中 hi_wdata 和 lo_wdata 来自 ex_hilo_bus，是从 EX 段传回 ID 段的数据 bus。

此外，当 ID 段处理的指令为分支跳转指令时，还需要处理 br_e 转移使能信号以及 br_addr 转移地址，并将其作为 br_bus 传回 IF 段进行跳转。

ID 段之后的工作便是将各类数据 bus 传到其他流水段。如将 id_to_ex_bus、id_load_bus、id_save_bus 和 id_hilo_bus 传入 EX 段进行后续处理。
id_to_ex_bus 中主要包括当前指令 PC 值、指令内容、alu 使能信号、指令相关使能信号、访存使能信号、寄存器写回信号、指令中两个操作数的值。id_load_bus 中主要包括 5 个 load 指令的使能信号。id_save_bus 中包含 3 个 save 指令的使能信号。id_hilo_bus 中包含 4 个数据移动指令使能信号、2 个乘法使能信号、2 个除法使能信号、读取到的 hi、lo 寄存器的值。

以上即为 ID 段的主要结构。

2.3 EX 段

EX 段主要用于完成指令中的计算操作。

EX 段的输入也包括常规的时钟信号、复位信号和流水线暂停信号。从 ID 段输出的 id_to_ex_bus、id_load_bus、id_save_bus、id_hilo_bus 也会作为 EX 段的输入端口。

EX 段的输出端口主要包括暂停信号 stallreq_for_ex，该信号在流水线处理除法指令 div 和 divu 时发出，会在流水线处理除法指令时将流水线保持停止。data_ram_sel 信号用于访存指令，传入 MEM 段用于决定访存指令需要用到的字节位置及数量。data_sram_en、data_sram_wen、data_sram_addr 和 data_sram_wdata 输出端口传出，用于进行内存中内容的读写。此外，EX 段的输出端口还需要传出 ex_to_rf_bus、ex_hilo_bus 数据 bus，以及 pre_inst_is_load 信号给 ID 段，ex_to_mem_bus、ex_load_bus 给 MEM 段。

EX 段结构较为简单，首先从输入的数据 bus 中读取数据，通过 bus 中的使能信号，确定操作符、操作数，通过 alu 模块进行计算并得出 alu_result。当处理指令为数据移动指令中的 mfhi 和 mflo 时，EX 段的运行结果为 hi 和 lo 寄存器的数据。接下来通过选择器来确定 EX 段的最终结果。

```
assign ex_result = inst_mfhi ? hi :
                   inst_mflo ? lo : alu_result;
```

接下来，EX 段将处理访存指令。利用 alu 模块计算出的访存指令的目标地址，确定访存指令涉及的字节数及字节位置，获得 data_ram_sel 信号传给 MEM 段。data_sram_wen、data_sram_addr、data_sram_wdata 三个信号得出之后传出 EX 段，进行内存的写入。进一步的访存指令内容将在 MEM 段实现。

接下来是乘除法模块，从 ID 段传入的 id_hilo_bus 中包含乘除法的使能信号，通过该信号决定是否启动乘除法模块，并获得计算结果 mul_result，div_result。由于乘除法的特性，两者为 64 位二进制数，因此需要保存在 hi、lo 寄存器中。hi 和 lo 写数据的选择器如下。

```
assign hi_wdata = inst_mthi ? rf_rdata1 :
                  inst_mult|inst_multu ? mul_result[63:32] :
                  inst_div|inst_divu ? div_result[63:32] :
                  32'b0;
```

```
assign lo_wdata = inst_mtlo ? rf_rdata1 :
                  inst_mult|inst_multu ? mul_result[31:0] :
                  inst_div|inst_divu ? div_result[31:0] :
                  32'b0;
```

最后，EX 段需要传出数据 bus。首先是 ex_to_mem_bus，其中包含指令 PC 值、内存读写使能信号、数据来源信号、写寄存器使能信号、写寄存器数据以及 EX 段得出的 ex_result。ex_to_rf_bus 需要传回写寄存器的使能和数据给 ID 段用于处理数据相关。ex_load_bus 将给 MEM 段传 load 指令的使能信号，用于在 MEM 段读取内存数据。ex_hilo_bus 会将 hi、lo 寄存器的写使能信号和写的数据内容传回给 ID 段，用于处理数据相关和写 hi、lo 寄存器。

2.4 MEM 段

MEM 段主要用于访存中 load 指令的最终完成，该段主要将会从内存中读取数据，并放入寄存器写回数据中。

MEM 段输入端也有时钟信号端口、复位信号端口以及流水线暂停信号端口。除此以外，MEM 段还有接收 EX 段传入的 ex_to_mem_bus 和 ex_load_bus 的端口、字节位置及数量 data_ram_sel 端口以及从内存中读取到的数据 data_sram_rdata 端口。

MEM 段的输出端口较少，只有输出传给 WB 段的 mem_to_wb_bus 以及用于处理数据相关问题的传给 ID 段的 mem_to_rf_bus。

MEM 段首先是从传入的 data_ram_sel 数据中获取将要读取内存的字节数以及字节位置，分别从中获取三个数据，也就是 b_data，h_data 和 w_data。

```
assign b_data = data_ram_sel_r[3] ? data_sram_rdata[31:24] :
                data_ram_sel_r[2] ? data_sram_rdata[23:16] :
                data_ram_sel_r[1] ? data_sram_rdata[15:8] :
                data_ram_sel_r[0] ? data_sram_rdata[7:0] :
                8'b0;
assign h_data = data_ram_sel_r[2] ? data_sram_rdata[31:16] :
```

```
                    data_ram_sel_r[0] ? data_sram_rdata[15:0] :
                    16'b0;
assign w_data = data_sram_rdata;
```

接下来，则要通过从 EX 段传入的 ex_load_bus 来获取 load 指令使能信号，并通过该使能信号确定将要用哪一个内存数据，并传给 mem_result。

```
assign mem_result = inst_lw ? w_data :
                    inst_lb ? {{24{b_data[7]}}, b_data} :
                    inst_lbu ? {24'b0, b_data} :
                    inst_lh ? {{16{h_data[15]}}, h_data} :
                    inst_lhu ? {16'b0, h_data} :
                    32'b0;
```

最后判断寄存器写回数据需要的是内存数据还是 EX 段获取的计算数据，并赋给 rf_wdata。

进入 MEM 段的输出阶段，首先是将指令 PC 值、写回使能、写回地址以及写回数据整合为 mem_to_wb_bus 传给 WB 段，最后需要将写回使能、写回地址以及写回数据整合为 mem_to_rf_bus 同步传给 ID 段以处理数据相关问题。

2.5 WB 段

由于该流水线中将寄存器写回放在了 ID 段进行，原本使用 WB 段进行的写回也就不存在了。于是该流水线中的 WB 段只是简单的传递数据，同时兼顾 debug 比对机制的处理。

WB 段的输入端口包括时钟信号、复位信号、流水线暂停信号以及从 MEM 段传入的 mem_to_wb_bus 数据 bus。输出端口主要为 wb_to_rf_bus 这一数据通路，该数据 bus 主要用于传给 ID 段进行寄存器数据写回以及处理数据相关问题。余下的四个输出端口 debug_wb_pc、debug_wb_rf_wen、debug_wb_rf_wnum、debug_wb_rf_wdata 用于传出给比对机制进行寄存器写回比对。

WB 段在读取数据 bus 之后，根据所得数据对 debug_wb_pc、debug_wb_rf_wen、debug_wb_rf_wnum、debug_wb_rf_wdata 四个输出端口进行了赋值并传出。

2.6 CTRL 模块

CTRL 模块不属于五段流水，但该模块用于接收 ID 段和 EX 段发出的流水线暂停请求，并对流水线各段发出暂停信号 stall，是流水线中非常重要的模块。

输入端口为复位信号、从 ID 段传入的 stallreq_for_load 以及从 EX 段出入的 stallreq_for_ex。stall_for_load 主要用于处理 load 数据相关时将流水线的暂停，stallreq_for_ex 主要用于 EX 段处理除法时将流水线的暂停。

接收这些输入后，输出端口则是向五段流水发送流水线暂停信号 stall，以便能够控制流水线的暂停与运行。

3. 感受及改进意见

钱名锐：

本次实验通过实操编写五段流水线，将课堂中学到的有关流水线的知识付诸实践，能够进一步加深对流水线知识的记忆与理解，我认为是一次比较有趣且有意义的实验。改进意见方面，关于时间的分配方面可能会因为学校的安排而受到或多或少的影响，答疑课设置的比较前，导致前期实验由于期末安排等缘故没有做多少内容就安排了答疑，而到后面可能更需要答疑课的时候安排却已经结束了。这一点不知道有没有更好的弹性安排。

刘培栋：

本次实验让我对 CPU 的认识更加全面，并且了解了 CPU 的一些设计流程和工作原理。在这次的实验过程中，我感受到了计算机软硬件之间的联系，了解到流水线的高性能和巧妙，并且感受到了数学的重要性。实验内容和课程内容紧密相连，让我对课上学到的知识有了更深的理解，同时掌握了一定的技能，收获颇丰。

4. 参考资料

《自己动手写 CPU》——雷思磊

RUNOOB Verilog 教程