Lab5: 流水线 CPU 设计

1. 实验目的

- 掌握流水线 CPU 的设计和实现。
- 理解通过流水线并行降低时延提高性能的基本原理。
- （Bonus）掌握流水线中拓展乘法指令的设计和实现。

2. 实验环境

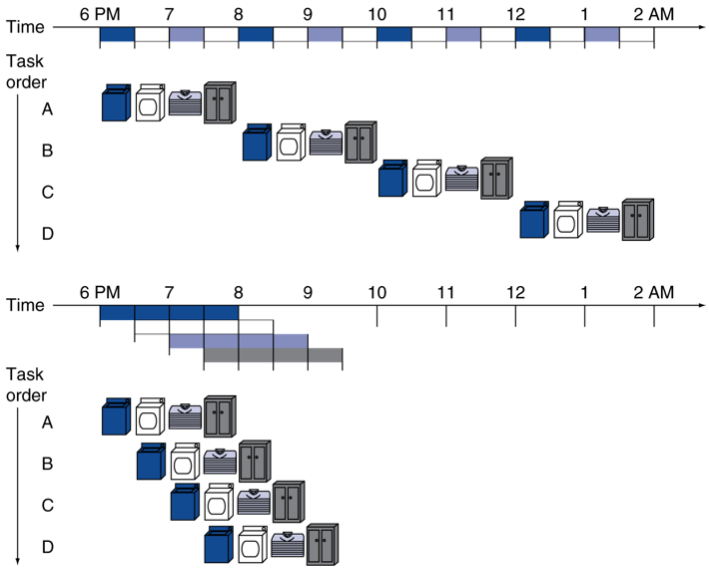
- vivado 2022.2
- 串口通信的单步调试工具。

- XC7A100TCSG324-2L 开发板。

3. 实验原理

3.1 流水线并行

若把执行一条指令的过程比作洗衣服的过程，那么单周期 CPU （上图） 和流水线 CPU （下图）的差别如下所示。



可以看到虽然细一件衣服的时间没有变，但是同时最多可以洗四件衣服，所以效率得到了大大的提高。对于 CPU 来说，从 pc 中存储的指令地址，一直到取出地址，进行计算，将数据存回 regfile 或者数据内存，这一条路径的延时非常长，若将其分成不同阶段，流水线的执行，每条指令执行的时间不变（甚至会变长一点），但是 CPU 会同时执行多条指令，所以运行的效率会得到很大的提升。

经典的五级流水线 CPU，分为如下五级：

- 取指（IF）：根据 pc 取出指令
- 译码（ID）：根据取出的指令进行译码，生成控制信号，取出操作数等。
- 执行（EX）：执行计算或者做比较的过程。
- 访存（MEM）：访问数据内存
- 写回（WB）：将计算结果写回 regfile。

3.2 流水线竞争

竞争或冲突是因为指令的执行由顺序变为并行执行，由于指令之间的一些依赖关系导致了并行执行的指令存在冲突，通常有以下三种冲突。

- 结构竞争：并行的指令需要使用同一结构。
- 数据竞争：需要等待前序的指令完成数据的读写才能执行。
- 控制竞争：分支和跳转指令后续执行的指令需要该条指令执行完成后才能决定。

通常解决流水线竞争的方式就是停顿（stall），即将产生冲突的指令以及之后的指令的执行都停止，等待引发冲突的指令执行完成再继续执行。但也有一些方式可以解决冲突，减少 stall 的时钟周期数。

3.3 结构竞争的解决方式

经典五级流水线中，结构冲突有以下两种。

- IF 级和 MEM 级都要访问内存，产生结构冲突。
- ID 级要读取 regfile，WB 级要写入 regfile。

解决的方式其实很简单：

- 访问内存的冲突，其实将数据内存和指令内存分开已经完美地解决了这个结构竞争，所以咱们实验其实不存在这个冲突。
- 对于读写 regfile 的冲突，若 regfile 使用上升沿写入，则对 regfile 的写入会在下一个时钟周期上升沿才完成，则在下一个时钟周期才能看见修改，我们可以使用 double bump 技术，即 regfile 使用下降沿写入，将一个时钟周期一分为二，前半个周期完成数据的写入，后半个周期完成数据的读取。即可完美解决这个冲突。

3.4 数据竞争的解决方式

数据竞争主要有以下三种类型：

- RAW：写后读
- WAR：读后写
- WAW：写后写

由于五级流水线也是顺序流水线，能保证在前序指令完成写入和读取后才进行写入，所以不存在后面两种数据竞争类型。~~（体系结构课程会学到的乱序流水线会存在这两种竞争，大家可以期待一下，确信）~~
CPU 中只有数据内存和 regfile 两个结构会涉及到写入操作，所以数据竞争主要发生在这两个结构中，又由于数据内存一次只能处理一条指令的操作，一条指令不可能同时读和写内存，且顺序流水线保证了执行的顺序，**所以对数据内存不会出现写后读的竞争。**
则写后读只会发生在寄存器的读写中，可以设想一下在 EX 级或者 MEM 级存在指令要修改某个寄存器，但它是 ID 级指令的一个源寄存器，此时若直接读取就只能读取到旧数据，而不是前序指令修改后写回的数据。
解决的方式主要为 forwarding，即前递。

3.4.1 forwarding

虽然前序指令在 EX 级或 MEM 级，还没将计算结果写回 regfile，但其实计算结果已经出现在了数据通路中，此时只要将其传到 ID 级，通过多路选择器选择正确的结果作为 ID 级从 regfile 中读取到的源数据即可。（当然这个多路选择逻辑也可以在 EX 中实现，具体的逻辑大家可以自行梳理。）
需要注意，若同时需要从 EX 级和 MEM 级前递结果，则需要判断前递最新的数据，即 EX 级的指令是后序指令，其才是最新的值。

3.4.2 load 导致的 stall

若此时 ID 级需要 forwarding EX 级写回目的寄存器的结果，但是此时 EX 级是一条 Load 指令，则还没读取到需要的结果，则此时 IF 和 ID 两级需要因为这个冲突停顿一个时钟周期，当 load 指令流动到 MEM 级，将数据读取出来后前递到 ID 级。这个停顿是无法避免的。若产生了这个停顿，则只需停止 IF 和 ID 两级的指令流动（保持相关寄存器的值不变），并将 EX 级的下一条指令设为无效指令即可。当然还有一种情况的 Load 产生的 stall 是可以通过 forwarding 来解决的，就是 load 指令后面是一条 store 指令，且 store 指令会将刚刚读取到的数据写回数据内存中，这个时候可以将 MEM 中读取到的数据前递到 MEM 级写端前的正确位置，从而减少掉这一 stall。

3.5 控制竞争的解决方式

控制竞争的解决方式主要是预测，预测分支跳转指令到底是跳转还是不跳转，主要有以下两种：

- predict taken：预测这条指令会发生跳转。
- predict not taken：预测这条指令不会发生跳转。

此处只解释 predict not taken 的实现方式，其实非常简单，IF 级中的 pc 正常 +4 即可，当 EX 级检测到这条指令需要跳转时，将原 IF 和 ID 级的指令设为无效指令，并将计算好的跳转目的地址在下一时钟周期写到 pc 中即可。

在其他指令集（如 MIPS）会有**延时槽**技术，理论分析上能在顺序流水线中实现很好的分支预测的效果，要打龙芯杯的同学可以自行搜索一下。

4. 实验步骤

4.1 准备工作

创建工程，从学在浙大下载压缩包 lab5_src.zip，解压后将下列源文件添加到工程中。

```
.
├── DMem.v
├── Defines.vh
├── IMem.v
├── Mem.v
├── UART_TX_CTRL.vhd
├── btn_scan.v
├── data.mem
├── debug_clk.v
├── debug_ctrl.v
├── function.vh
├── inst.mem
├── my_clk_gen.v
├── top.v
└── uart_buffer.v
```

将下列引脚约束文件添加到工程中。

```
.
└── constraint.xdc
```

将 Lab1 中实现的 MACtrl 模块，RV32Core 模块，DispNum 模块及其子模块添加到工程中。

```
MACtrl.v
RV32Core.v
DispNum.v
  Mux8to1_4b.v
  clkdiv.v
... //自己定义的其他子模块
```

此时整个工程结构除了 Core.v 部分已经完整，此时可以将 Lab3 中实现的三个模块以及 Lab4 中实现的数据通路中会出现的模块添加到自己的工程中。随后创建（或者添加）Core.v 文件，在其中实现自己的流水线 CPU 即可。

4.2 流水线 CPU 中需要实现的指令

本次实验需要实现的指令和 Lab4 中第一部分实现的指令一样，即如下指令：

- lw, sw

- beq, bne bge, bgeu, blt, bltu
- addi, slti, sltiu, xori, ori, andi, slli, srli, srai
- add, sub, sll, slt, sltu, xor, srl, sra, or, and
- lui, auipc, jal, jalr

4.3 流水线 CPU 的实现

本次实验推荐大家实现流水线时同时使用上 **double bump，forwarding，predict not taken** 三个技术，三个技术可以参考实验原理部分（**使用这三个技术有 0.5 分 bonus**），当然实在里不清楚的同学也可以直接用 stall 来处理流水线的竞争。这三个技术能极大限度地解决流水线地竞争问题，同时实现非常简单，只需要在流水线中添加部分数据通路即可，比检测每种可能的竞争并且控制复杂的 stall 逻辑要简单很多，**实现简单，性能更高**，所以推荐大家实现。

接下来就是将 Lab4-1 的数据通路分为五级，可以按照以下方式分为五级：

- IF：pc 的赋值逻辑，连接 IMem 得到指令。
- ID：指令解码，生成控制信号，根据指令生成立即数，从 regfile 中读取源数据，根据 EX 和 MEM 级的相关信号实现 Forwarding 的逻辑。
- EX：ALU 和 Comparator，在该级可以判断是否需要跳转，若要跳转则将原 IF 和 ID 级的指令 flush 掉。
- MEM：数据内存
- WB：将数据写回 regfile

可以根据这个划分自行画图或者修改 Lab4 中的数据通路。

对于各级之间用于传递数据的寄存器，根据各级的模块需要的数据进行传递即可，在命名时可以加上诸如 IF, ID, EX, MEM, WB 之类的前缀来区分不同级的寄存器。建议每一级都要包含如下三种寄存器，方便 debug 以及方便将某一级的指令设为无效。

- pc：该级指令对应的 pc。
- inst：该级执行的指令
- valid：该级指令是否有效，由于流水线中会有 bubble，即可能会有某一级指令无效的情况，为了避免无效指令修改 regfile，错误修改流水线数据流动等，在写使能，以及判断 forwarding，判断跳转时都要注意与上 valid 信号，来确保相关数据流动是有效指令引起的。

其他实现细节就不再赘述了，避免误导，具体的实现需要同学们花时间自行梳理。

4.4 Core.v 宏定义处理

本实验的宏定义处理和 Lab4 中 Core.v 的宏定义处理相同，但是注意上板时显示的是 WB 级的指令和地址，则请将 WB 级的 pc 和指令的寄存器声明为如下

```verilog
reg [31:0] wb_pc;
reg [31:0] wb_inst;
```

4.5 仿真和上板测试

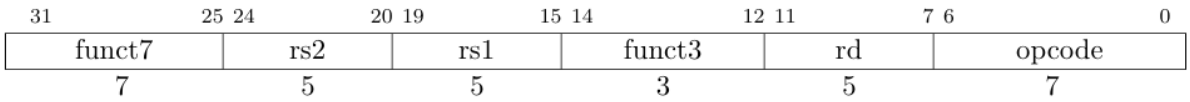
lab5_src 中有两份测试文件 test1.s 和 test.pipeline.s：

- test1.s ： 其实就是 Lab4-1 的测试代码，流水线 CPU 需要保证要求指令的正确执行。
- test.pipeline.s：测试流水线中各种竞争的测试代码。

验收和上板的现象都在测试文件的注释中。

4.6 乘法指令的拓展（Bonus）

这个部分是 bonus 部分，让大家把自己的乘法器接到自己的流水线 CPU 上，并支持乘法指令 mul。mul 指令的格式如下所示。



其中 opcode = 7'b0110011, func3 = 3'b000, func7 = 7'b0000001
这里我们规定实现的乘法指令 mul 的功能为如下伪代码所示，当然实际 risc-v 中 mul 指令的功能不是这样，这仅为本次实验要求。该指令格式支持使用 venus 汇编出指令的机器码。

```c
reg[rd] = signed(reg[rs1][15:0]) * signed(reg[rs2][15:0]);
```

这里提供一个拓展的思路供大家参考：将 Lab2 实现的乘法器模块添加到工程中，将其作为 ALU 实现的一种操作，将乘法器的细节隐藏在了 ALU 中
大家可以参考如下步骤拓展：

- 在控制单元中添加对乘法指令的解码，数据通路的控制信号按照 R-type 指令的走即可，并将 alu_ctrl 赋值为乘法运算的编号。
- 修改 ALU 的接口，添加 clk，rst，valid 输入信号，添加 alu_stall 输出信号。输入信号只需将 CPU 时钟，复位信号，EX 级的 valid 信号接入即可。
- 在 ALU 中接入 Mul 模块，Mul 模块的 start 信号当 alu_ctrl 为乘法操作且 valid 为 1 时为 1即可，注意只用将两个源操作数的低 16 位作为乘法器的源操作数。
- 将乘法器的输出接到 ALU 结果的多路选择器中，根据 alu_ctrl 选择结果即可。
- alu_stall 作为告诉整个 CPU 在执行乘法运算的信号，它的作用是在要执行乘法运算时将整个

CPU 停顿，其只需要在 mul_start （输入 Mul 模块的 start 信号）为 1 且 mul_finish （输出Mul 模块的 finish 信号）为 0 时为 1 即可。则当遇到乘法指令且没有计算完时 CPU 就会被 stall，在乘法器计算完成的那个周期 CPU 就会继续流动，并将乘法器的输出结果传到 MEM 级。

- 实现当 alu_stall 为 1 时整个流水线 stall 的逻辑，流水线不流动即数据通路中所有的寄存器值不变，则在数据通路中所有寄存器的赋值逻辑（除了复位的赋值逻辑）前与上 alu_stall 为 1 的条件即可。

乘法指令的测试程序在 test_mul.s 中，里面用的是 Lab2 的乘法的测试样例，需要完成该测试程序的**仿真**和**上板验收**才算完成这个 Bonus。

5. 验收测试

本实验需要完成 test1.s 和 test_pipeline.s 两个测试程序的验证才算完成该实验。对于完成 Bonus 的同学还需要完成 test_mul.s 程序的测试才算拿到 Bonus。验收时请提前讲多个比特流生成好再验收。
验收流程：

1. 查看工程结构。
2. 上板查看现象。

验收时排队时请标明你完成的 bonus，按照如下格式之一排队：

- xxx lab5
- xxx lab5 forwarding + predict not taken
- xxx lab5 mul
- xxx lab5 forwarding + predict not taken + mul

6. 思考题

无。~~想不出来了，大家一起摆吧。~~

7. 实验报告要求

本次实验**需要撰写实验报告**，本次实验以及后续实验的格式按照学在浙大上发布的 **报告模板.docx** 进行书写。
本次实验报告的一些必须项：

- 实验报告的第一部分，结合文字和核心代码，介绍一下整个流水线 CPU 的实现过程。做了 Bonus 的同学也将拓展乘法指令的过程也通过文字和核心代码展示一下。
- 实验报告的第二部分需要展示两个（或三个）测试程序的仿真波形图，并使用**文字**对其进行一定的讲解，随后展示两个（或三个）测试程序上板后串口的输出信息的截图和七段数码管显示的照片，看看是否和正确的验收现象相同。
- 实验报告的第三部分写一下本次实验的心得。

请认真完成实验报告，明显敷衍的完成实验报告将会扣取一定分数！