实验三：5 段流水 CPU 设计实验报告

杨健邦 515030910223

一、实验目的：

1. 理解计算机指令流水线的协调工作原理，初步掌握流水线的设计和实现原理。
2. 深刻理解流水线寄存器在流水线实现中所起的重要作用。
3. 理解和掌握流水段的划分、设计原理及其实现方法原理。
4. 掌握运算器、寄存器堆、存储器、控制器在流水工作方式下，有别于实验一的设计和实现方法。
5. 掌握流水方式下，通过 I/O 端口与外部设备进行信息交互的方法。与外部设备进行信息交互。

二、实验内容：

1. 采用 Verilog 在 quartus Ⅱ 中实现基本的具有 20 条 MIPS 指令的 5 段流水 CPU 设计。
2. 利用实验提供的标准测试程序代码，完成仿真测试。
3. 采用 I/O 统一编址方式，即将输入输出的 I/O 地址空间，作为数据存取空间的一部分，实现 CPU 与外部设备的输入输出端口设计。实验中可采用高端地址。
4. 利用设计的 I/O 端口，通过 lw 指令，输入 DE2 实验板上的按键等输入设备信息。即将外部设备状态，读到 CPU 内部寄存器。
5. 利用设计的 I/O 端口，通过 sw 指令，输出对 DE2 实验板上的 LED 灯等输出设备的控制信号（或数据信息）。即将对外部设备的控制数据，从 CPU 内部的寄存器，写入到外部设备的相应控制寄存器（或可直接连接至外部设备的控制输入信号）。
6. 利用自己编写的程序代码，在自己设计的 CPU 上，实现对板载输入开关或按键的状态输入，并将判别或处理结果，利用板载 LED 灯或 7 段 LED 数码管显示出来。
7. 例如，将一路 4bit 二进制输入与另一路 4bit 二进制输入相加，利用两组分别 2 个 LED 数码管以 10 进制形式显示“被加数”和“加数”，另外一组 LED 数码管以 10 进制形式显示“和”等。（具体任务形式不做严格规定，同学可自由创意）。
8. 在实现 MIPS 基本 20 条指令的基础上，实现 Y86 相应的基本指令。
9. 在实验报告中，汇报自己的设计思想和方法；并以汇编语言的形式，提供以上两种指令集（MIPS 和 Y86）应用功能的程序设计代码，并提供程序主要流程图。

三、预习内容：

1. 实验前仔细阅读 DE1-SOC User Manual 及相关用户应用数据手册，学习并掌握 其板载相关资源的工作原理、连接方式、和应用注意事项。
2. 根据课程所讲 5 段流水 CPU 设计原理，提前设计并仿真实现相关设计代码。原 理，提前设计并仿真实现相关设计代码。

四、实验器材：

1. Altera-DE2 实验板套件 1 套
2. 万用表 1 台
3. 示波器 1 台

五、设计思想和方法：

1. 基本功能的编写：根据老师所给的电路图以及实验指导书上面的顶层文件进行 模块化编写，共分为 5 个阶段（组合逻辑），5 组寄存器，在每个时钟上升沿时，将每个阶段产生的输出存到后面一组寄存器中。如果有需要 stall 的寄存器，则可以使用 dffe32.v 模块，被 stall 的寄存器不读入新值。
2. 时钟的设计：50MHZ 的时钟可能会太快，导致指令 rom 和数据 ram 无法读出 来，需要 pll 模块把时钟的频率放慢一些。在同一个时钟上升沿同时读 rom 和 ram，也有可能导致读不出来，可以采用上一次单周期 CPU 二分频时钟的方法， 使得 ram 和 rom 不会同时读或写。
3. 利用转发解决部分 data hazard：在 decode 阶段，如果 da 或 db 需要 exe 或者 mem 阶段产生但没写回寄存器文件的数据，可将这些数据直接转发给 alu 做运算，使用数据的优先级规则是：越近的阶段产生的数据越优先转发。

```
always @ (ewreg or mwreg or ern or mrn or em2reg or mm2reg or rs or rt )begin
                fwda =2'b00;// no hazard
                if (ewreg & (ern != 0 ) & (ern == rs) & ~em2reg)begin
                        fwda = 2'b01; //select exe_alu
                end else begin
```

```
                    if (mwreg & (mrn != 0 ) & (mrn == rs) & ~mm2reg) begin
                        fwda = 2'b10;//select mem_alu
                    end else begin
                        if (mwreg & (mrn != 0) & (mrn == rs) & mm2reg)
begin
                            fwda = 2'b11;//select mem_lw
                        end
                    end
                end
```

4. 无法用转发处理的 data hazard：当 lw 指令的下一条指令需要马上用到 lw 所读入的寄存器时，无法用转发解决。在 decode 的时候可以检测到这个 hazard，然后要将 PC 的寄存器，fetch/decode 的寄存器给 stall，然后插入一个 bubble。Stall 的方法解决当 wpcir 为 0 的时候，寄存器不读入新的值。插入 bubble 的方法，将在 decode 阶段的指令的 wreg 和 m2reg 和 wmem 三个信号量置为 0，这样这条指令就不会产生任何实际效果，相当于 bubble。

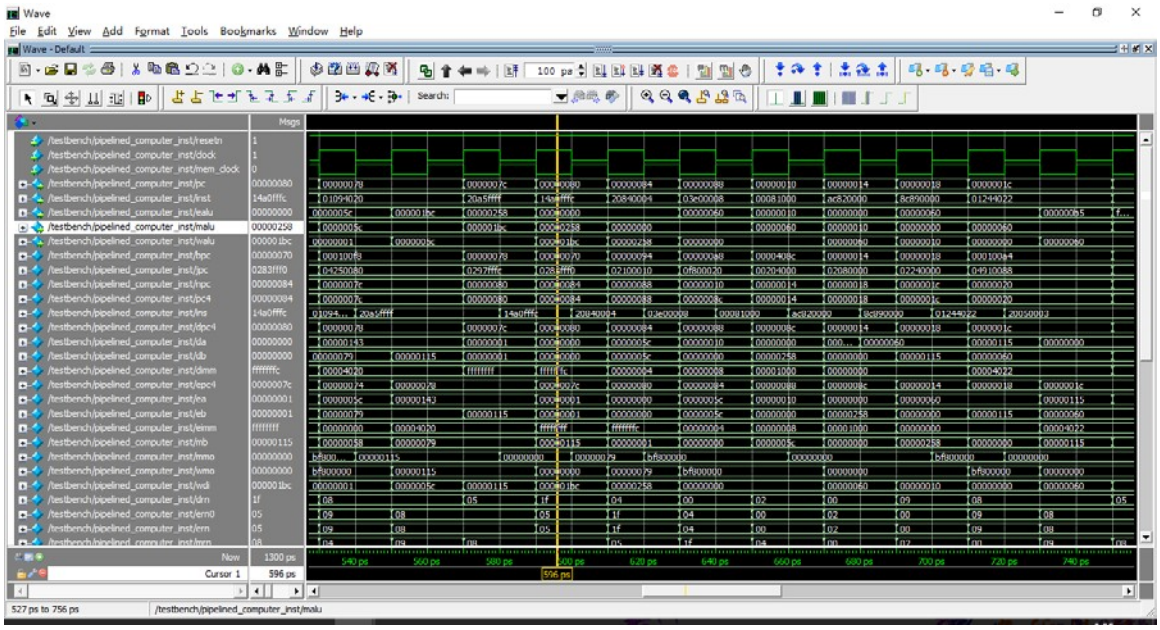
```
assign wpcir = ~ (ewreg & em2reg & (ern != 0 ) & (i_rs & (ern == rs) | i_rt & (ern ==
rt)));
```

5. 处理 Control hazard：分支预测后一条指令执行，真正的 pc 只有当跳转相关的指令到达 decode 之后才能得到，这就意味着分支预测错误只会导致跳转指令的后一条指令多执行。通过在汇编中每个跳转指令后面，插入一条 nop 指令（全 0）来解决，这样多执行的指令不会产生任何影响。

```
0 : 20020080; % (00) addi $2, $0, 10000000b # address 80h %
1 : 20030084; % (04) addi $3, $0, 10000100b # address 84h %
2 : 20010088; % (08) addi $1, $0, 10001000h # address 88h %
3 : 200400c0; % (0c) addi $4, $0, 11000000b # address c0h %
4 : 200500c4; % (10) addi $5, $0, 11000100b # address c4h %
5 : 8c860000; % (14) loop: lw $6, 0($4) # input data from [c0h] %
6 : 8ca70000; % (18)       lw $7, 0($5) # input data from [c4h] %
```

```
7 : 00c74020; % (1c) add $6, $7, $8 # $8 = $6 + $7 %
8 : ac460000; % (20) sw $6, 0($2) # output [C0h] to [80h] %
9 : ac670000; % (24) sw $7, 0($3) # output [c4h] to [84h] %
A : ac280000; % (28) sw $8, 0($1) # output [c0h] + [c4h] to [88h] %
B : 08000005; % (2c) j loop # loop %
C : 00000000;
```

6. IO 设计：板上的 10 个开关分成 2 组，每组都作为一个输入端口。6 个数码管分成 3 组，每组都作为一个输出端口，显示的是十进制的数字，由于数量的限制，只能显示十位和个位的数字，通过一个 num2sevseg 模块来转换。
7. Modelsim 仿真：



六、体会和教训：

1. Mif 文件中，指令的标号是用十六进制表示的，9 的后面是 A，10 代表的是第 16 条指令。
2. 如果在 modelSim 里面仿真成功，而板子上面却失败，可以考虑一下是时钟的问题。50MHZ 的时钟可能会太快，导致指令 rom 和数据 ram 无法读出来，需要 pll 模块把时钟的频率放慢一些。在同一个时钟上升沿同时读 rom 和 ram，也有可能导致读不出来，可以采用上一次单周期 CPU 二分频时钟的方法，使得 ram 和 rom 不会同时读或写。