Lab3 32 位 MIPS 多周期处理器设计

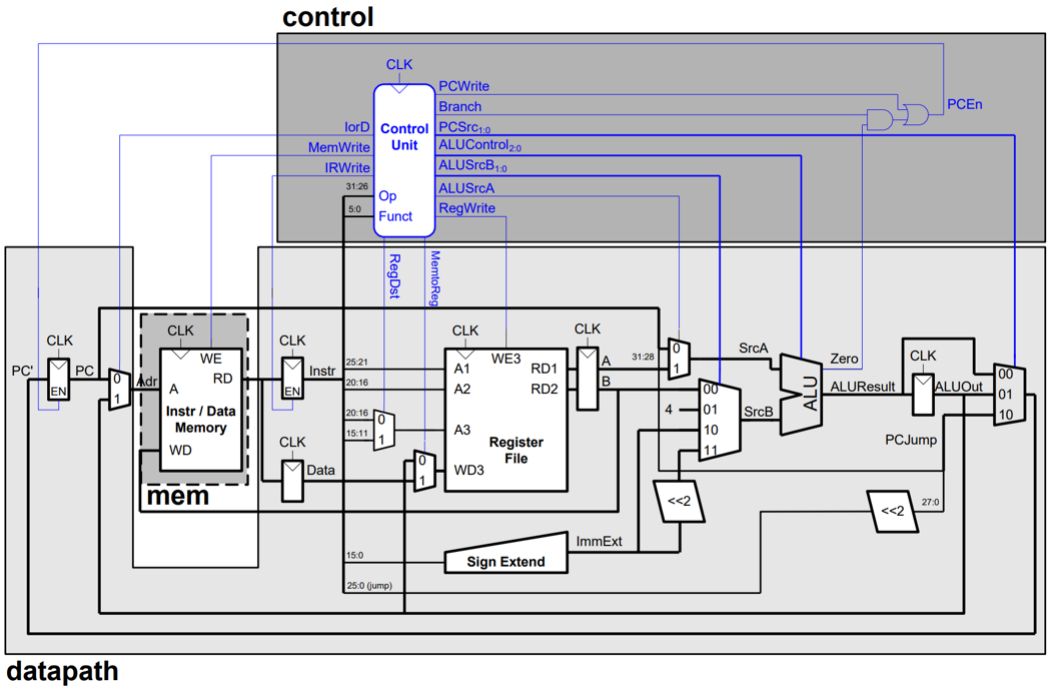
1 实验目的

熟悉 MIPS 多周期 CPU 的工作原理

2 实验过程

2.1 学习 MIP 多周期处理器原理

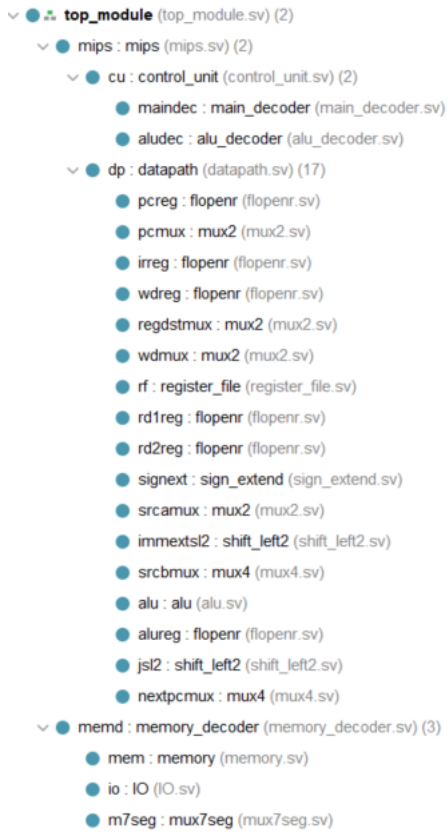
首先在阅读了教材（《计算机组成与体系结构（第三版）》）的第五、六章后，对于 MIPS 多周期处理器的工作原理喝单周期的不同之处我有了更深刻的理解。对于寄存器，ALU，译码器等模块的工作原理以及共同实现的效果有了更深刻的了解。并且对如下图所示的 MIPS 多周期处理器的工作原理以及使用它完成各种指令时各模块的工作方式有了较全面的了解。



2.2 完成 MIPS 多周期处理器设计

2.2.1 添加代码框架

根据上次实验中单周期处理器的代码进行修改，写出代码的整体结构，如下图所示：



2.2.2 添加指令集

2.2.2.1 指令集概览

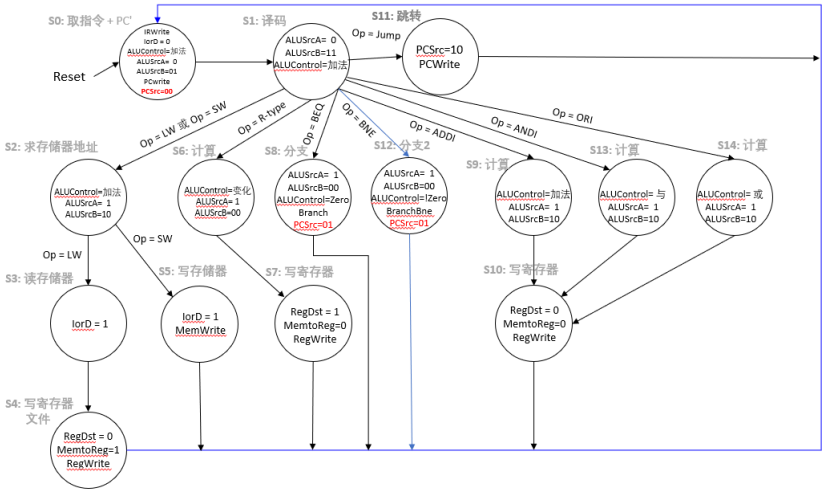
经过编写，目前本 MIPS 多周期处理器实现的指令集如下：

指令	指令类型	功能
add	R	加
sub	R	减
add	R	按位与
or	R	按位或
slt	R	小于则置 1
lw	I	读内存
sw	I	写内存
addi	I	加立即数
andi	I	按位与立即数
ori	I	按位或立即数
beq	I	相等则跳转
j	J	跳转

2.2.2.2 指令集实现

指令集中多数指令都已经在书中给出了其具体实现方式，这里不再赘述。一些新增的指令并不需要对处理器的结构做出大修改，只需要调整译码器的状态机，使其可以识别新的指令。

2.2.2.3 主译码器 FSM



2.2.2.4 ALUOP 编码表

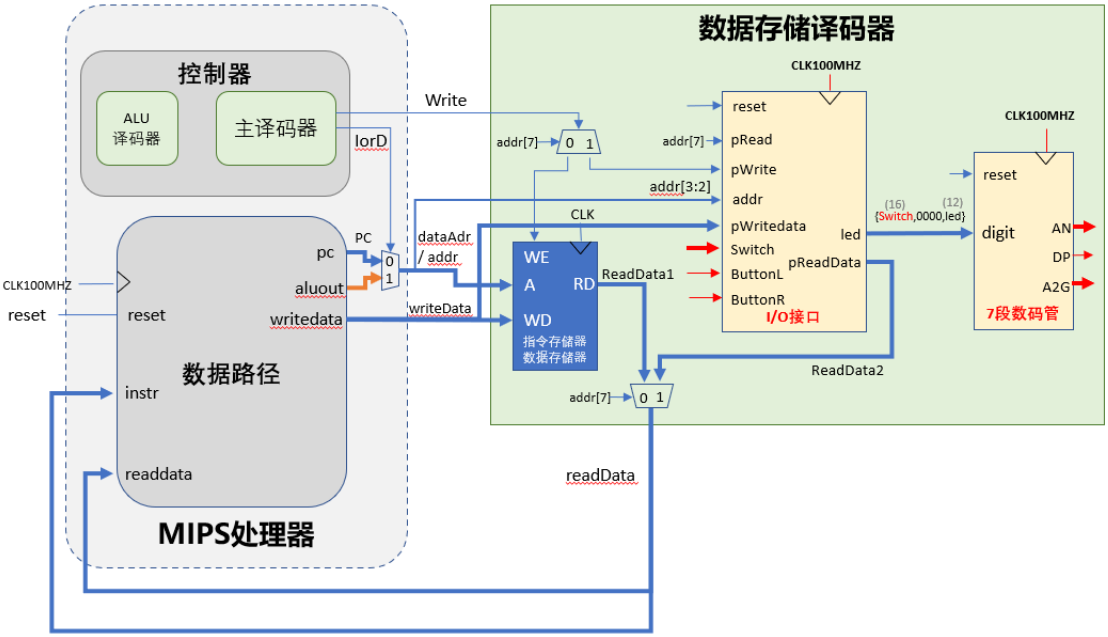
ALUOP	含义
000	加法
001	减法
010	按位与
011	按位或
100	根据 funct 决定
other	未使用

2.2.2.5 ALU 译码器真值表

ALUop	Funct	ALUControl
000	x	加
001	x	减
010	x	按位与
011	x	按位或
100	100000	加
100	100010	减
100	100100	按位与
100	100101	按位或
100	101010	小于则置 1
other	x	未使用

2.3 测试代码

为了便于测试，本 CPU 还实现了 IO 接口，可通过开关实时控制 CPU 中运算的数值。原理和上次单周期的版本一致，为在内存中高位设置一块接口区，其中包含两个操作数的值、一个结果的值和一个状态数，当访问到内存中的高位时，改为从接口区读取数值，同时在代码中通过判断状态数决定是否进行跳转，从而使输入可以即时反映到代码。具体则是通过扩展 DataMemory 实现。最后添加和 7 段数码管的衔接，从而将结果显示出来。结构图如下图所示：



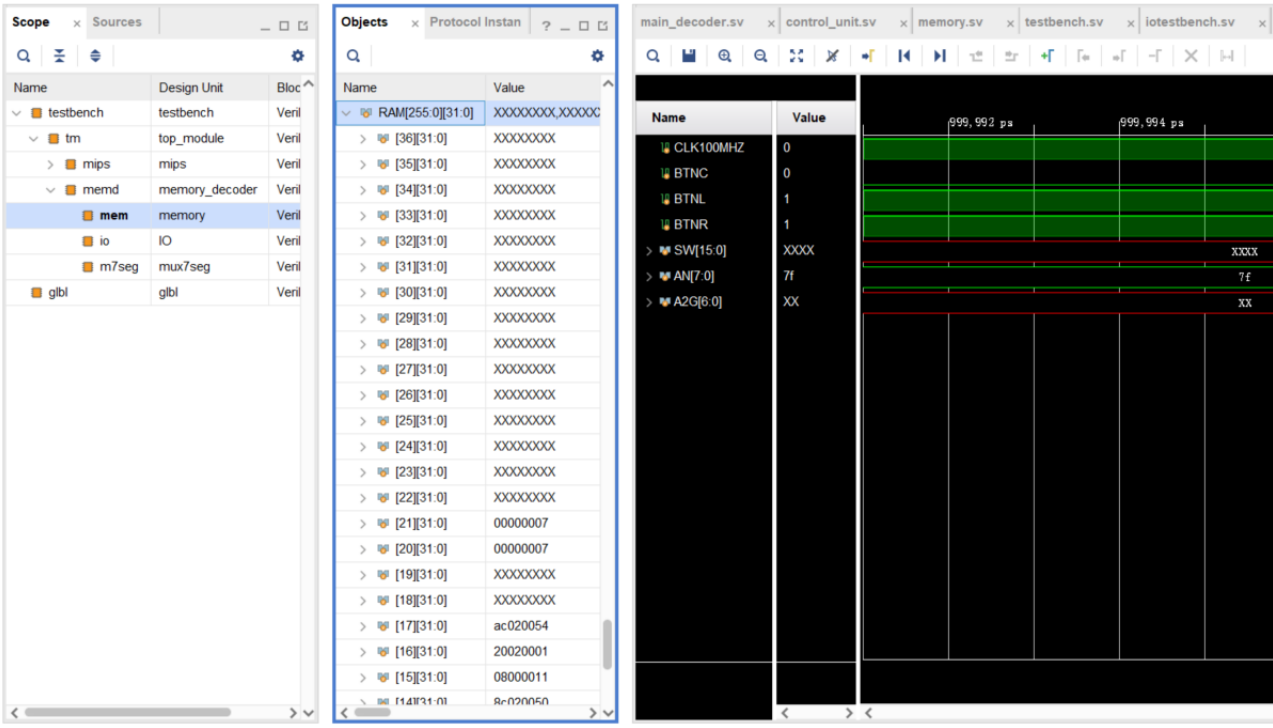
data_memory_decoder: 将下列三个模块衔接起来；

data_memory: 修改前的 DataMemory，扩展了内存用来进行 IO 接口；

IO: 判断状态数，存储三个数的数值，并将数值传输给下一个模块；

mux7seg: 用 7 段数码管显示三个数；

接下来先进行实验材料中 PPT 上的 MIPS 测试代码的仿真，测试结果如下图：



可以看到在运行结束后内存中的 21 位（之所以是 21，是因为将指令和数据存在一个内存中后，便于地址对齐所导致，在转换为实际地址时乘以 4 即可）存入了 7，与代码设计相符。

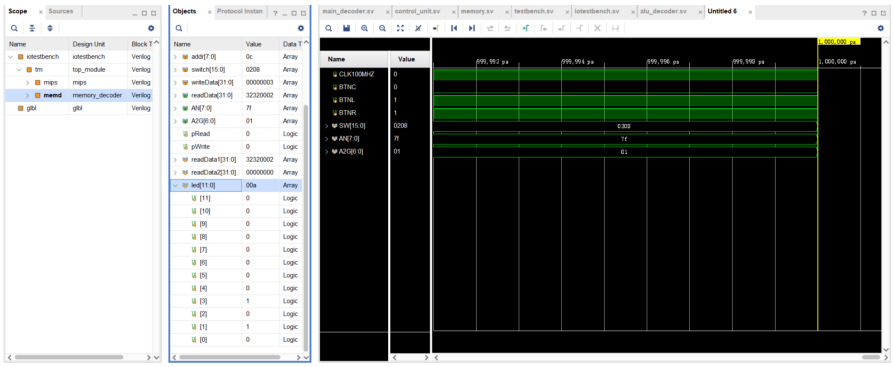
然后进行关于 IO 接口的 MIPS 测试代码的仿真，其中 IO 接口的 MIPS 测试代码如下：

```
main:
    addi $s0, $0, 0
    sw   $s0, 0x80($0)
chkSwitch:
    lw   $s1, 0x80($0)
    andi $s2, $s1, 0x2
    beq  $s2, $0, chkSwitch
    lw   $s3, 0x88($0)
    lw   $s4, 0x8C($0)
    add  $s5, $s4, $s3
chkLED:
    lw   $s1, 0x80($0)
    andi $s2, $s1, 0x1
    beq  $s2, $0, chkLED
    sw   $s5, 0x84($0)
    j    chkSwitch
```

测试仿真代码如下图：

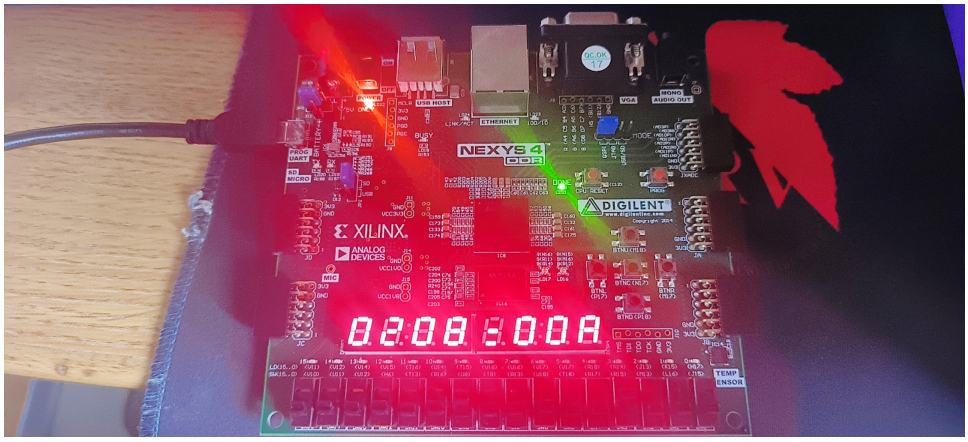
```
module iotestbench();
logic CLK100MHZ, BTNC, BTNL, BTNR;
logic [15:0] SW;
logic [7:0] AN;
logic [6:0] A2G;

initial
begin
    #0;
    CLK100MHZ = 0;
    BTNC = 1;
    #2;
    BTNC = 0;
    #2;
    BTNL = 1;
    BTNR = 1;
    #2;
    SW = 16'b00000010_00001000;
end

always
    begin
        #5;
        CLK100MHZ = ~CLK100MHZ;
    end

top_module tm
(
CLK100MHZ, BTNC, BTNL, BTNR, SW, AN, A2G
);

endmodule
```

测试结果如下图：



可以看到在运行结束后 led 变量存入了 0000000a，即十进制的 10，与代码中应存入模拟输入的 2+8 的结果相符。

最后进行实机测试，测试结果如下图：



3 实现体会

在进行 MIPS 多周期处理器实验的过程中，我深深感受到了 CPU 的复杂与精妙。通过此次实验，我不仅对计算机体系结构有了更深的理解，还掌握了多周期处理器的工作原理。在实现多周期处理器的过程中，我遇到了不少困难，比如时序控制和状态转换的实现。通过反复调试，我发现了许多设计中的细节问题，并逐一解决。通过这次 MIPS 多周期处理器实验，我不仅掌握了多周期处理器的设计与实现方法，还培养了细致严谨的科研态度和解决问题的能力。