计算机组成原理实验报告

单周期处理器开发

班级：1618001

学号：161810329

姓名：荀正

专业：培优班

2020.4.20

一、功能设计说明

1.完成的指令集：

addiu、slt、beq、addi、sw、sub、j、lw、ori、and、or、add、lui、xor

2.处理器为单周期设计

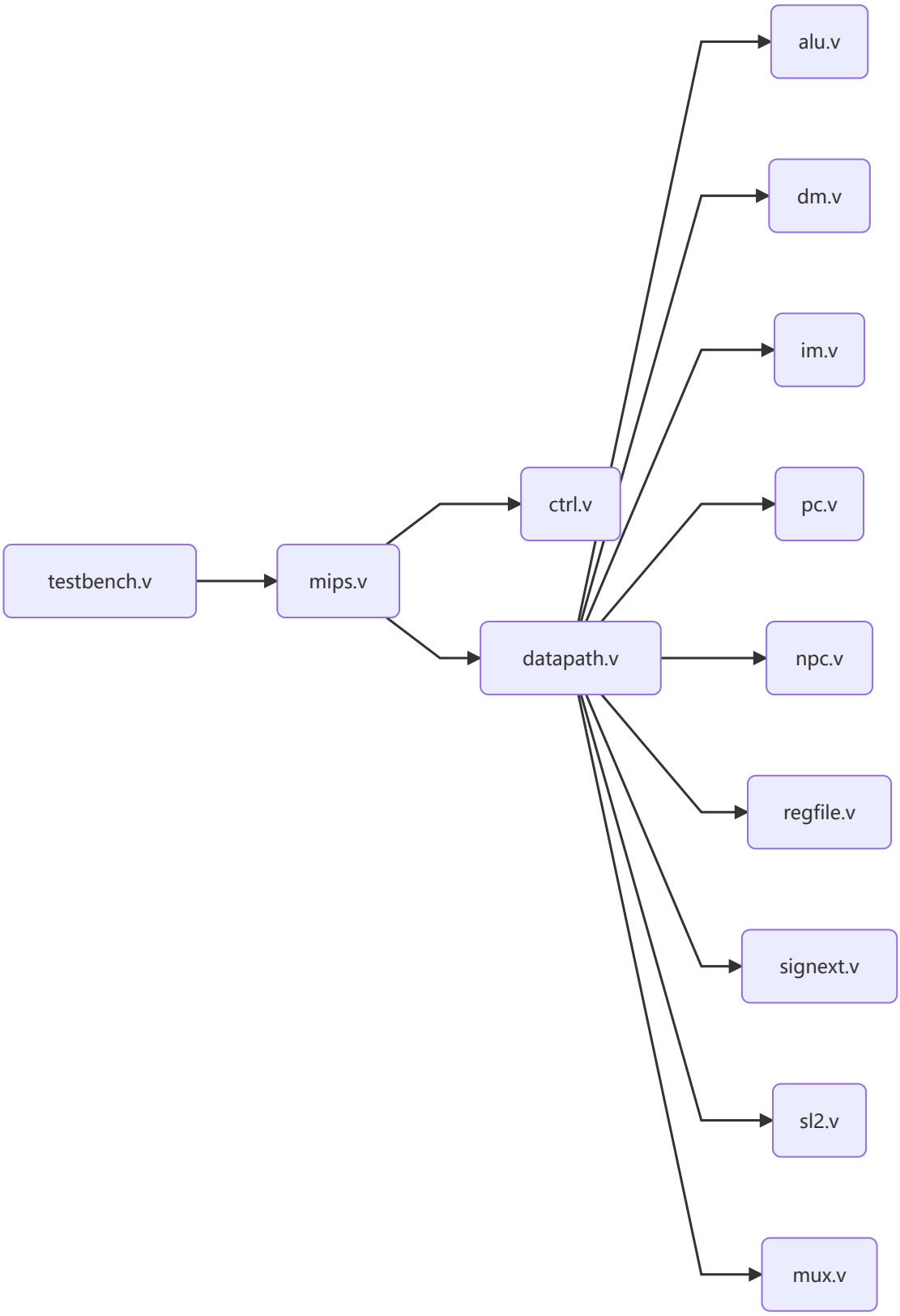
3.功能模块采用教材图设计，在控制信号上，为了应对一些可能出现的拓展，做了一些微调。

二、模块化和层次设计说明

模块对应

文件名	对应部件名
alu.v	算数逻辑模块
ctrl.v	控制器模块
datapath.v	数据通路模块
dm.v	数据存储模块 用来存取数据
im.v	指令存储模块 取指令
mips.v	进入的主模块
mux.v	选择器模块 复用
npc.v	计算下一个pc模块
pc.v	pc模块
regfile.v	寄存器模块
signext.v	符号拓展模块
sl2.v	左移模块
testbench.v	测试运行模块

关系图



三、模块定义

1、pc模块定义

1）基本描述

输出当前指令地址并保存下一条指令地址。复位后，PC指向0x0000_3000，即第一条指令的地址。

2) 模块接口

信号名	方向	描述
NPC [31:2]	I	下条指令的地址
clk	I	时钟信号 上升沿触发
rst	I	复位信号 1：复位 0：无效
PC [31:2]	O	30位指令存储器地址

3）功能定义

序号	功能名称	功能描述
1	复位	复位信号有效时，PC设置为0x0000_3000
2	输出NPC	在上升沿的时候输出NPC

2、npc模块定义

1）基本描述

根据当前的指令地址结合分支、跳转等信号计算下一条指令地址并输出给PC

2) 模块接口

信号名	方向	描述
PC [31:0]	I	当前pc
instruction [25:0]	I	当前指令的后26位，j指令使用
beqInstruction [31:0]	I	根据指令计算的beq指令地址，已经左移两位
branch [1:0]	I	分支信号 2'b00：无分支 2'b10：beq 2'b11：bne
jump[1:0]	I	跳转信号 2'b00：无跳转 2'b01：j 2'b10：jr 2'b11：jal
zero	I	alu传来的zero信号，用于branch指令 1：相等 0：不相等
NPC [31:2]	O	下一个指令地址
fourPC[31:2]	O	当前指令加四地址，留作拓展

3）功能定义

序号	功能名称	功能描述
1	输出NPC	计算下条指令并输出
2	输出PC+4	留给jal指令使用

3、im模块定义

1）基本描述

im（instruction memory）的主要功能是存储指令，根据指令输出相应指令。

2) 模块接口

信号名	方向	描述
addr[11:2]	I	下条指令地址
dout[31:0]	O	下条指令

3）功能定义

序号	功能名称	功能描述
1	读取指令并输出	根据输入地址读取指令并输出

4、dm模块定义

1）基本描述

DM的主要功能是数据的存储和读出。

2) 模块接口

信号名	方向	描述
addr [11:2]	I	传入的数据地址
we	I	是否允许写入数据 1：允许写入 0：不允许写入
clk	I	时钟信号
din	I	要存储的数据
dout	O	根据地址取出的数据

3）功能定义

序号	功能名称	功能描述
1	读数据	根据读入的地址寻找相应地址的数据并读出，主要用于lw指令
2	写数据	sw指令将读出的寄存器的值存入dm

5、regfile模块定义

1）基本描述

regfile的主要功能是完成寄存器的读取写数据。

2) 模块接口

信号名	方向	描述
readRegister1 [4:0]	I	指令的[25:21]位，rs段
readRegister2 [4:0]	I	指令的[20:16]位，rt段
writeRegister [4:0]	I	指明往哪个寄存器写数据，由指令的类型决定
writeData	I	指明往寄存器中写入的数据是什么
regWrite	I	决定是否写数据 1：写 0：不写
clk	I	时钟信号
rst	I	复位信号 1：复位，寄存器清零 0：无效
readData1 [31:0]	O	readRegister1对应的寄存器中的数据
readData2 [31:0]	O	readRegister2对应的寄存器中的数据

3）功能定义

序号	功能名称	功能描述
1	读寄存器	根据指令读取指定寄存器
2	读数据	根据指令从指定寄存器中取数据
3	写数据	根据指令将数据写入到指定寄存器

6、alu模块定义

1）基本描述

ALU的主要功能是完成数据的运算。ALU可以计算加法、减法、或、与、异或等运算。ALU除了完成相关运算，还可判断beq指令下zero是否为0；slt指令下目的寄存器该存入1还是0。

2) 模块接口

信号名	方向	描述
aluOp[2:0]	I	控制alu执行何种运算 3'b000：加 3'b001：减 3'b010：或 3'b011：slt 3'b100：与 3'b101：异或
data1 [31:0]	I	输入数据一，为rs寄存器中数据
data2 [31:0]	I	输入数据二，根据指令类型取寄存器数据或立即数
zero	O	判断beq指令下两数相减是否为0 1：相等 0：不相等
result [31:0]	O	数据运算结果

3）功能定义

序号	功能名称	功能描述
1	数据运算	根据输入数值和aluOp进行运算
2	判断数据是否相等	用于输出zero
3	比较大小	用于在slt等指令下判断大小

7、mux模块定义

1）基本描述

mux是多路选择器模块，根据选择信号从多个输入信号中选择一个作为输出信号。

2) 模块接口

mux2_32

信号名	方向	描述
a[31:0]	I	第一个32位数据
b[31:0]	I	第二个32位数据
sel	I	选择信号
y[31:0]	O	输出32位数据

mux2_30

信号名	方向	描述
a[31:2]	I	第一个30位数据
b[31:2]	I	第二个30位数据
sel	I	选择信号
y[31:2]	O	输出30位数据

mux3_32

信号名	方向	描述
a[31:0]	I	第一个32位数据
b[31:0]	I	第二个32位数据
c[31:0]	I	第三个32位数据
sel	I	选择信号
y[31:0]	O	输出32位数据

mux3_5

信号名	方向	描述
a[4:0]	I	第一个4位数据
b[4:0]	I	第二个4位数据
c[4:0]	I	第三个4位数据
sel	I	选择信号
y[4:0]	O	输出4位数据

3）功能定义

序号	功能名称	功能描述
1	选择数据	根据选择信号从多个输入中选择一个数据输出

8、sl2模块定义

1）基本描述

将输入的32位数据左移两位。虽然可以设置成自定义位数，感觉在这里有点鸡肋。

2) 模块接口

信号名	方向	描述
data[31:0]	I	输入的要移位的数据
slData[31:0]	O	移位后的数据

3）功能定义

序号	功能名称	功能描述
1	移位	将数据左移两位

9、signext模块定义

1）基本描述

extender的主要功能是扩展16位的立即数。其内部包括0扩展、符号扩展、高位扩展(lui)等扩展。

2) 模块接口

信号名	方向	描述
instruction[15:0]	I	输入的16位拓展数据
extType[1:0]	I	判断拓展的类型 2'b00：符号拓展 2'b01：无符号拓展 2‘b10：lui拓展
signExtNumber[31:0]	O	将数据拓展后的结果

3）功能定义

序号	功能名称	功能描述
1	无符号拓展	无符号拓展
2	有符号拓展	将数的最高位往前复制16个
3	高位拓展	在数后添加16个0

10、ctrl模块定义

1）基本描述

Controller的主要功能是控制各个部件的执行功能及多路选择器。Controller是根据指令决定各个控制信号的。

2) 模块接口

信号名	方向	描述
opcode [5:0]	I	指令的[31:26]位
funct [5:0]	I	指令的[5:0]位
regDst [1:0]	O	决定写入到哪个寄存器 2'b00：写入到指令[20:16]指定的寄存器 2'b01：写入到指令[15:11]指定的寄存器 2'b10：写入到31号寄存器
jump [1:0]	O	跳转信号 2'b00：无跳转 2'b01：j 2'b10：jr 2'b11：jal
branch [1:0]	O	分支信号 2'b00：无分支 2'b10：beq 2'b11：bne
memRead	O	是否允许从dm中读数据，默认可读 1：读 0：不读
memToReg [1:0]	O	写入寄存器数据来源 2'b00：alu的计算结果写入 2'b01：dm读取的数据写入 2'b10：时钟信号写入
aluOp [2:0]	O	控制alu执行何种运算 3'b000：加 3'b001：减 3'b010：或 3'b011：slt 3'b100：与 3'b101：异或
memWrite	O	是否允许往dm中写入数据 1：允许 0：不允许
aluSrc	O	决定alu的第二个数据来源 1：alu的第二个数据来自extender扩展后的数 0：alu的第二个数据来自读出的第二个寄存器的值
regWrite	O	是否往寄存器中写入数据 1：是 0：否
extType [1:0]	O	判断拓展的类型 2'b00：符号拓展 2'b01：无符号拓展 2'b10：lui拓展

3）功能定义

序号	功能名称	功能描述
1	控制	根据指令输出各种数据控制各个部件执行

11、datapath模块定义

1）基本描述

Datapath是以上各个部件的综合，将他们之间的相互连线都连接在一起。

2) 模块接口

信号名	方向	描述
clk	I	时钟信号
rst	I	复位信号 1：复位 0：无效
regDst [1:0]	I	决定写入到哪个寄存器 2'b00：写入到指令[20:16]指定的寄存器 2'b01：写入到指令[15:11]指定的寄存器 2'b10：写入到31号寄存器
jump [1:0]	I	跳转信号 2'b00：无跳转 2'b01：j 2'b10：jr 2'b11：jal
branch [1:0]	I	分支信号 2'b00：无分支 2'b10：beq 2'b11：bne
memRead	I	是否允许从dm中读数据，默认可读 1：读 0：不读
memToReg [1:0]	I	写入寄存器数据来源 2'b00：alu的计算结果写入 2'b01：dm读取的数据写入 2'b10：时钟信号写入
aluOp [2:0]	I	控制alu执行何种运算 3'b000：加 3'b001：减 3'b010：或 3'b011：slt 3'b100：与 3'b101：异或
memWrite	I	是否允许往dm中写入数据 1：允许 0：不允许
aluSrc	I	决定alu的第二个数据来源 1：alu的第二个数据来自extender扩展后的数 0：alu的第二个数据来自读出的第二个寄存器的值
regWrite	I	是否往寄存器中写入数据 1：是 0：否

信号名	方向	描述
extType [1:0]	I	判断拓展的类型 2'b00：符号拓展 2'b01：无符号拓展 2'b10：lui拓展
instruction[31:0]	O	当前执行的指令

3）功能定义

序号	功能名称	功能描述
1	数据传送	实现数据连接

12、mips模块定义

1）基本描述

mips是datapath和controller的组合，可以通过clk和rst来控制mips指令的执行。

2) 模块接口

信号名	方向	描述
clk	I	时钟信号
rst	I	复位信号 1：复位 0：无效

3）功能定义

序号	功能名称	功能描述
1	执行指令	执行相应的指令

四、测试代码及结果

测试代码

```
while:
addiu $t1,$zero,10
addiu $t2,$zero,0x0
addiu $t3,$zero,10
addiu $t4,$zero,1
addiu $t7,$zero,4
addiu $v0,$zero,0
```

```
for:
slt $v1,$v0,$t3
beq $v1,0,after1
addi $v0,$v0,1
sw $t1,($t2)
sub $t1,$t1,$t4
addi $t2,$t2,4
j for



after1:
addiu $t1,$zero,10
addiu $v0,$zero,0
addiu $a0,$zero,0
sub $t3,$t3,$t4



bubble1:
addiu $t2,$zero,0x0
slt $v1,$v0,$t3
beq $v1,0,after2
addi $v0,$v0,1
addiu $a0,$zero,0



bubble2:
slt $v1,$a0,$t3
beq $v1,0,bubble1
addi $a0,$a0,1
lw $t1,($t2)
lw $t5,4($t2)
addi $t2,$t2,4
slt $v1,$t1,$t5
beq $v1,1,bubble2
sub $t2,$t2,$t7
ori $t6,$t1,0
and $t1,$t1,$zero
or $t1,$t1,$t5
and $t5,$t5,$zero
or $t5,$t5,$t6
sw $t1,($t2)
sw $t5,4($t2)
add $t2,$t2,$t7
j bubble2



after2:
addiu $v0,$zero,0
addiu $t2,$zero,0x0

result:
slt $v1,$v0,$s3
beq $v1,0,afterall
addi $v0,$v0,4
```

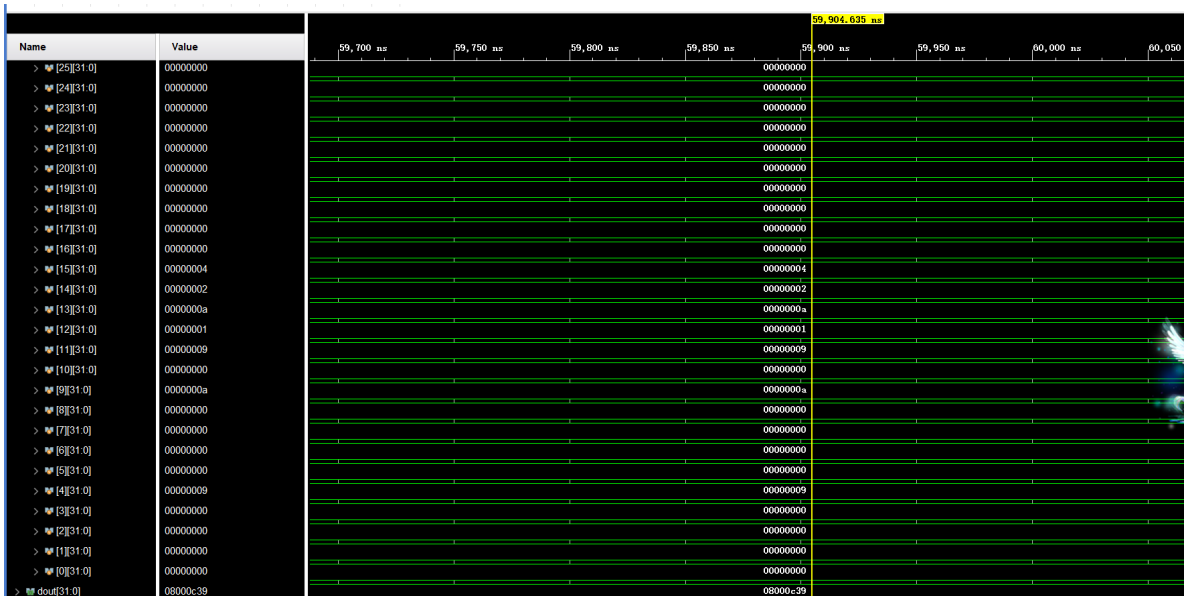
```
lw $t1 ($t2)
addi $t2,$t2,4
j result


afterall:
lui $t1,0x0100
xor $t1,$t1,$t1
addiu $t1,$zero,10
j end



end:
j end
```

运行结果



五、实验时间安排

周五花了约5小时看了一些verilog的基础知识并且看着图和学姐的代码开始思考这个系统的流程以及自己应该如何架构。

周六完成了一些组件的开发，约9小时。

周日完成datapath和ctrl的代码，开始调试，一共花了约11个小时。

总时间大概30小时，包含以前的一些知识的学习。

六、心得体会

在做的过程中，我一直当cpu是一个大的时序电路，通过控制器生成的控制信号来告诉各个组件们应该做什么事情，所以这就需要提前顶层设计以及模块接口的设计，这样才能够通过datapath这样一个串将全局串起来。

在写verilog的时候，感觉verolog和其他语言还是有不一样的地方，感觉像是电路的抽象化，先定义模块就像定义了一些组件，就像电路实验用到的什么00芯片等，然后例化就是取出一块具体的芯片，然后通过各种数据（导线）将他们连起来。同时要注意代码规范，尤其是datapath的数据，要注意接对。调试的过程中遇到一个bug就是因为datapath的数据连接错了，导致dm中数据没有。

在自己给ctrl写指令对应的控制信号的时候，我都要对着图在自己脑子里过一下这个指令应该是怎么执行的，让我对原来的理论知识理解的更加深刻。

感觉这样的一个周末，有点充实，有点肝。最后模拟出结果的时候还是蛮开心的。