计算机组成原理实验报告

VerilogHDL完成五级流⽔线处理器开发

班级：1619402班

学号：161940205 161940206 161940219 161940234

姓名：裴书玥 曹子钰 阴俊晖 牛纪龙

学院：计算机科学与技术学院/人工智能学院

2021.06.19

**文档目录：**

1. 功能设计说明
2. 模块化和层次化设计说明
3. 具体模块定义
4. 测试代码及结果
5. 课设完成时间安排
6. 心得体会
7. 功能设计说明

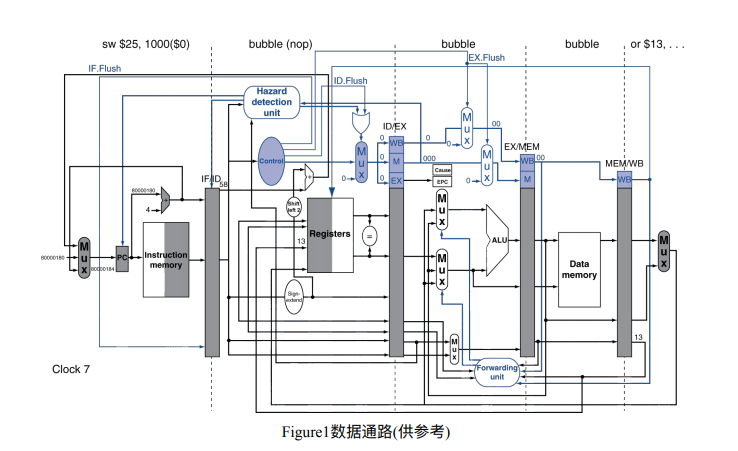
1. 完成以下指令集

a.完成如下指令（若遗漏有基本流⽔proj2的指令，⾃⾏补⻬）（共39条）

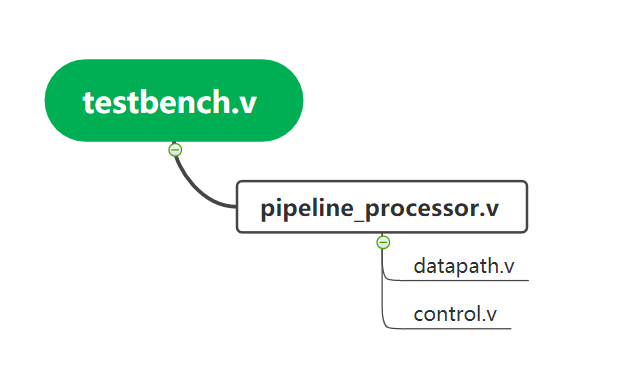
算数运算指令： add , addi , addu , addiu sub , subu slt , slti , sltu , sltiu 逻辑运算指令： and , andi , or , ori lui , nor , xor , xori 移位指令： sll , slv, srl , srlv , sra , srav 分⽀跳转指令： beq , bne bgez , blez j , jr 访存指令： lb , lh , lw sb , sh , sw

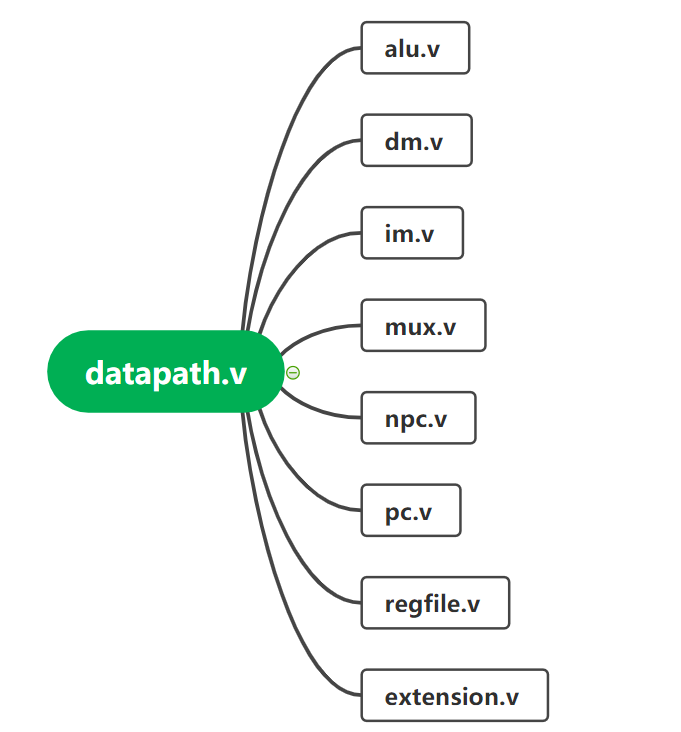
2. 处理器为流⽔线设计，⽀持各类冒险与转发。

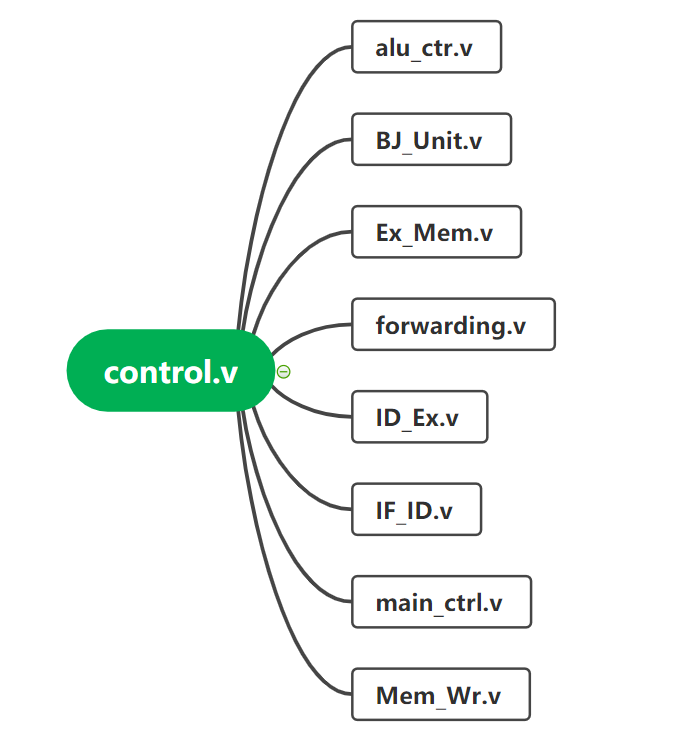
3. 参考的指示图如下（教材 p198）此外还添加了sll , slv, srl , srlv , sra , srav，bgez , blez , lh 指令的实现，做出了一定的修改



1. 模块化和层次设计说明







1. 具体模块定义

1）PC模块定义：

(1) 基本描述

PC 主要功能是完成输出当前指令地址。复位后，PC指向0x0000\_3000，此处为第一条指令的地址。

(2) 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| [31:0] next\_addr | I | 下一条指令地址 |
| clk | I | 时钟信号 |
| rst | I | 重置信号 |
| [31:0] cur\_pc | O | 当前指令地址 |
| IF\_pcHigh4 | O | 指令地址高四位 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 复位 | Reset=1时，将out置为0X0000\_3000 |

2）NPC模块定义：

(1) 基本描述

NPC 主要功能是根据当前指令是否为branch或者jump，来输出下一条指令的地址。

(2) 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| [31:0] IF\_pc\_4 | O | PC+4 |
| [1:0] Ex\_jump | I | 区别J,JR,JAL,JALR指令。  00->not jump  01->j or jal  10->jr |
| [2:0] Mem\_Branch | I | 000->not branch  001->BEQ  010->BNE  011->BGEZ  100->BLEZ |
| [31:0]Mem\_branch\_addr | I | branch指令的跳转地址 |
| [31:0] jr\_addr | I | jr指令的跳转地址 |
| [31:0] j\_addr | I | j指令的跳转地址 |
| [31:0] next\_addr | O | 下一条指令地址 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 输出指令地址 | 根据Ex\_jump和 Mem\_Branch的值输出下一条指令的地址。 |

3）MUX模块定义：

(1) 基本描述

实现任意位二选一或三选一数据选择器，默认情况下是32位。

(2) 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| [4:0] Ex\_Rt | I | Rt寄存器中数据 |
| [4:0] Ex\_Rd | I | Rd寄存器中数据 |
| Ex\_RegDst | I | 指定目的寄存器  1->Rd  0->Rt |
| [4:0] Ex\_Rw | O | Rd目的寄存器中数据 |

4) main\_ctrl模块定义:

(1) 基本描述

根据输入的指令高6位(Op字段)，指令的低6位（func字段）,输出ALUctr，Branch，Jump，RegDst，ALUSrc,MemtoReg，RegWr，MemWr，ExtOp，MemRead，ALUshf，R31Wr控制信号。为了统一方便管理，选择了集中式的控制器。

(2) 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| [5:0] op | I | op字段，即指令高6位 |
| [5:0] func | I | func字段，即指令低6位 |
| [1:0] ID\_jump | I | Jump指令控制信号 |
| [3:0] ID\_ALUop | O | ALU的控制信号 |
| ID\_RegDst | O | 目的寄存器控制信号 |
| [1:0] ID\_ALUsrc | O | ALU数据来源控制信号 |
| ID\_MemWr | O | 存储器写控制信号 |
| [1:0] ID\_ExtOp1, ID\_ExtOp2, ID\_ExtOp3 | O | 立即数扩展控制信号 |
| ID\_MemRead | O | 存储器读控制信号 |
| ID\_ALUShamtSrc | O | ALU移位控制信号 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 生成各种控制信号 | 根据输入的op， func，输出各种控制信号。 |

5)IF\_ID流水线寄存器模块定义:

(1) 基本描述：

IF和ID阶段之间的过程寄存器

(2) 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| rst | I | 重置信号 |
| [3:0] IF\_pcHigh4 | I | IF阶段指令高四位 |
| [3:0] ID\_pcHigh4 | O | ID阶段指令高四位 |
| [31:0] IF\_instruction | I | IF阶段的指令 |
| [31:0] ID\_instruction | O | ID阶段的指令 |
| [31:0] IF\_pc\_4 | I | IF阶段PC+4信号 |
| [31:0] ID\_pc\_4 | O | ID阶段PC+4信号 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | IF/ID过程寄存器 | 将信息从IF阶段传递到ID阶段 |

6)ID\_EX流水线寄存器模块定义:

(1) 基本描述：

ID阶段和EX阶段的过程寄存器。

(2) 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| rst | I | 重置信号 |
| [31:0] ID\_npc | I | ID段npc |
| [31:0] ID\_busA | I | ID阶段总线A的数据 |
| [31:0] ID\_busB | I | ID阶段总线B的数据 |
| [4:0] ID\_Rs | I | ID阶段的rs寄存器编号 |
| [4:0] ID\_Rt | I | ID阶段的rt寄存器编号 |
| [4:0] ID\_Rd | I | ID阶段的rd寄存器编号 |
| [15:0] ID\_imm16 | I | ID阶段的扩展后的立即数 |
| ID\_RegWr | I | ID阶段写回寄存器的控制信号 |
| ID\_RegDst | I | ID阶段的判断写回寄存器编号的信号 |
| [1:0] ID\_ALUsrc | I | ID阶段ALU源操作数信号 |
| ID\_MemWr | I | ID阶段存储器写信号 |
| [1:0] ID\_MemtoReg | I | ID阶段的MemtoReg信号 |
| [3:0] ID\_ALUop | I | ID阶段的ALU控制信号 |
| ID\_MemRead | I | ID阶段的存储器读信号 |
| ID\_ALUShamtSrc | I | 传入ALU的移位数的控制信号 |
| [4:0] Ex\_Rs | O | Ex阶段的rs寄存器编号 |
| [4:0] Ex\_Rt | O | Ex阶段的rt寄存器编号 |
| [4:0] Ex\_Rd, | O | Ex阶段的rw寄存器编号 |
| [31:0] Ex\_busA | O | Ex阶段的总线A数据 |
| [31:0] Ex\_busB | O | Ex阶段的总线B数据 |
| [15:0] Ex\_imm16 | O | Ex阶段扩展后的立即数 |
| Ex\_RegWr | O | Ex阶段的寄存器写信号 |
| Ex\_RegDst | O | Ex阶段目的寄存器信号 |
| [1:0] Ex\_ALUsrc | O | Ex阶段ALU源操作数信号 |
| Ex\_MemWr | O | Ex阶段的存储器写信号 |
| [1:0]Ex\_MemtoReg | O | Ex阶段的存储器写入寄存器信号 |
| [3:0] Ex\_ALUop | O | Ex阶段ALU控制信号 |
| Ex\_MemRead | O | Ex阶段存储器读信号 |
| Ex\_ALUShamtSrc | O | Ex阶段指令的shamt字段 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | ID和Ex阶段的过程寄存器 | 在时钟上升沿到来时，将信息传递到Ex阶段 |

7)Ex\_Mem流水线寄存器模块定义:

(1) 基本描述：

将信号从Ex阶段传递到Mem阶段

(2) 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| rst | I | 重置信号 |
| [31:0] Ex\_ALUout | I | Ex阶段Alu计算结果 |
| 31:0] Ex\_datain | I | Ex阶段总线B的数据 |
| 4:0] Ex\_Rw | I | Ex阶段rd寄存器号 |
| Ex\_RegWr | I | Ex阶段寄存器写信号 |
| [1:0] Ex\_MemtoReg | I | Ex阶段存储器到寄存器信号 |
| Ex\_MemWr | I | Ex的存储器写信号 |
| [31:0] Mem\_ALUout | O | Mem阶段的Alu计算结果 |
| [31:0] Mem\_datain | O | Mem阶段的总线B数据 |
| [4:0] Mem\_Rw | O | Mem阶段的rd寄存器编号 |
| Mem\_RegWr | O | Mem阶段的寄存器写信号 |
| [1:0] Mem\_MemtoReg | O | Mem阶段存储器写回寄存器信号 |
| Mem\_MemWr | O | Mem的存储器写信号 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | Ex到Mem的过程寄存器 | 随之时钟信号，将数据传递到Mem阶段中 |

8)Mem\_Wr流水线寄存器模块定义:

(1) 基本描述

Mem阶段过渡到Wr阶段的过程寄存器，将Mem阶段的一些信号存储起来并传送到Wr阶段。

(2) 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| rst | I | 重置信号 |
| [31:0] Mem\_dataout | I | 内存读出数据 |
| [31:0] Mem\_ALUout | I | Alu计算结果 |
| [31:0] Mem\_npc | I | npc读出数据 |
| [4:0] Mem\_Rw | I | rw寄存器号 |
| Mem\_RegWr | I | RegWr信号 |
| [1:0] Mem\_MemtoReg | I | 写回寄存器来源的控制信号 |
| [31:0] Wr\_dataout | O | 内存输出结果 |
| [31:0] Wr\_ALUout | O | ALU计算结果 |
| [31:0] Wr\_npc | O | npc读出的数据 |
| [4:0] Wr\_Rw | O | rw寄存器号 |
| Wr\_RegWr | O | 是否写回寄存器的控制信号 |
| [1:0] Wr\_MemtoReg | O | 写回寄存器数据来源的控制信号 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 输出Wr阶段所需数据 | 在时钟上升沿到来时，将所需的Mem数据传入Wr阶段。 |

9）ALU模块定义：

(1) 基本描述

实现addu,Subu,slt,and,nor,or,xor,sll,srl,sltu,sra,lui这12种计算。

(2) 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| [3:0] Ex\_ALUctr | I | ALU控制信号 |
| [31:0] A | I | 4种来源，寄存器直接读，指令的shf，上一条指令ALU计算得到，上两条指令ALU计算得到 |
| [31:0] B | I | 4种来源，寄存器直接读，指令中立即数，上一条指令ALU计算得到，上两条指令ALU计算得到。 |
| reg [31:0] Ex\_ALUout | O | ALU计算结果。 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 输出计算结果 | 根据ALU控制信号，输出计算结果 |

10）IM模块定义：

(1) 基本描述

指令存储器，通过addr传入，将指令输出。

(2) 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| [11:2] cur\_pc | I | 地址 |
| [31:0] IF\_instruction | O | 对应的指令 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 输出指令 | 根据传入addr，输出地址对应的指令 |

11）regfile模块定义：

(1) 基本描述

由输入的两个寄存器地址，读出相应的值，根据RegWr写信号和Rw，判断遇到时钟上升沿时是否将输入的数据写入寄存器。在寄存器中保存数值。

(2) 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Clk | I | 时钟信号 |
| Wr\_RegWr | I | 写使能端 |
| [4:0] ID\_Ra | I | Rs寄存器地址 |
| [4:0] ID\_Rb | I | Rt寄存器地址 |
| [4:0] Wr\_Rw | I | Rd寄存器地址 |
| [31:0] ID\_busB | I | Rd寄存器应存储的值 |
| rst | I | 重置信号 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 在寄存器中保存值 | 存值到寄存器 |
| 2 | 将数值写入寄存器 | 把busW的值写入寄存器rd |
| 3 | 输出寄存器的值 | 输出地址rs rt中保存的值 |

12）extension模块定义：

1. 基本描述：

主要功能：将立即数进行拓展

（2）模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| [1:0] Ex\_ExtOp1，Ex\_ExtOp2，Mem\_ExtOp3 | I | 符号位扩展控制信号  1->功能：半字0扩展、半字符号扩展、lui  2->功能：不扩展、半字符号扩展、字节符号扩展  3->功能：不扩展、半字符号扩展、半字0扩展、字节符号扩展、字节0扩展 |
| [15:0] Ex\_imm16 | I | 输入16位立即数 |
| [31:0] Ex\_imm32 | O | 输出结果 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 输出拓展后结果 | 根据ExpOp控制信号，输出立即数符号位扩展或0扩展的结果 |

1. 测试代码及结果

1. asm代码（test\_with.asm）

test\_addi:

add $s1,$0,0

add $s2,$0,0

add $s0,$0,100

addi $s3,$1,-100

test\_addiu:

addiu $s4,$s0,-1000

test\_andi:

andi $s5,$s0,1010

test\_ori:

ori $s1,$s2,100

test\_xori:

xori $s5,$s2,100

test\_sltiu:

sltiu $s4,$s5,100

test\_add:

add $s1,$s2,$s3

test\_addu:

add $s1,$s1,$s2

test\_sub:

sub $s1,$s2,$s3

test\_subu:

subu $s3,$s2,$s2

test\_and:

and $s4,$s5,$s2

test\_or:

or $s2,$s3,$s1

test\_nor:

nor $s2,$s3,$s4

test\_xor:

xor $s4,$s3,$s2

test\_slt:

slt $s0,$s2,$s3

test\_sltu:

sltu $s1,$s3,$s5

save\_test\_result:

add $s6,$0,100

sw $s0,($s6)

sw $s1,4($s6)

sw $s2,8($s6)

sw $s3,12($s6)

sw $s4,16($s6)

sw $s5,20($s6)

addi $t0, $0, 0xffffffff

addi $t1,$0, 0

sw $t0,($t1)

addi $t1,$t1,4

sh $t0,($t1)

addi $t1,$t1,4

sb $t0,($t1)

lw $t2,($0)

addi $t1,$t1,4

sw $t2,($t1)

lh $t2,($0)

addi $t1,$t1,4

sw $t2,($t1)

lhu $t2,($0)

addi $t1,$t1,4

sw $t2,($t1)

lb $t2,($0)

addi $t1,$t1,4

sw $t2,($t1)

lbu $t2,($0)

addi $t1,$t1,4

sw $t2,($t1)

lui $t3,0xffff

addi $t1,$t1,4

sw $t2,($t1)

sll $t3,$t3,2

addi $t1,$t1,4

sw $t3,($t1)

sra $t3,$t3,2

addi $t1,$t1,4

sw $t3,($t1)

srl $t3,$t3,2

test\_j:

addi $t2,$0,60

addi $t2,$t2,60

beq $t2,120,test\_beq

bne $t2,180,test\_j

sw $t3,($t1)

test\_beq:

jal test\_jal

j end

test\_jal:

jr $ra

end:

1. 生成的16进制文件(test\_with.txt)

20110000

20120000

20100064

2033ff9c

2614fc18

321503f2

36510064

3a550064

2eb40064

02538820

02328820

02538822

02529823

02b2a024

02719025

02749027

0272a026

0253802a

0275882b

20160064

aed00000

aed10004

aed20008

aed3000c

aed40010

aed50014

2008ffff

20090000

ad280000

21290004

a5280000

21290004

a1280000

8c0a0000

21290004

ad2a0000

840a0000

21290004

ad2a0000

940a0000

21290004

ad2a0000

800a0000

21290004

ad2a0000

900a0000

21290004

ad2a0000

3c0bffff

21290004

ad2a0000

000b5880

21290004

ad2b0000

000b5883

21290004

ad2b0000

000b5882

200a003c

214a003c

20010078

102a0003

200100b4

142afffa

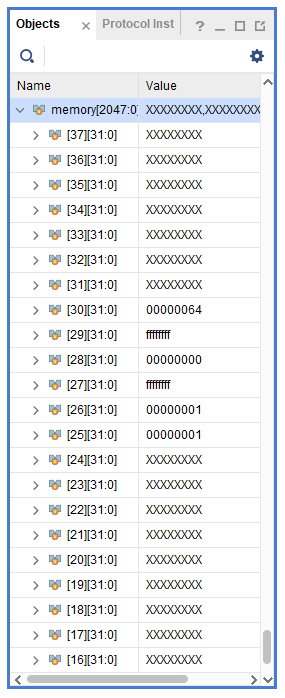
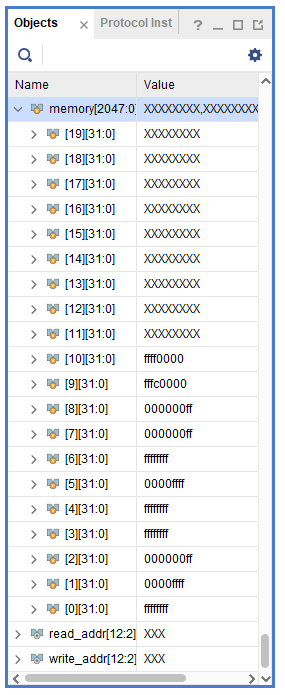
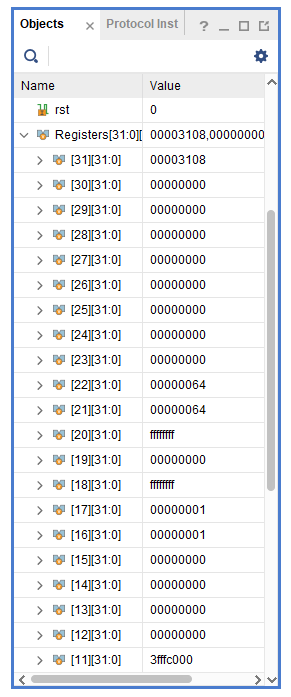
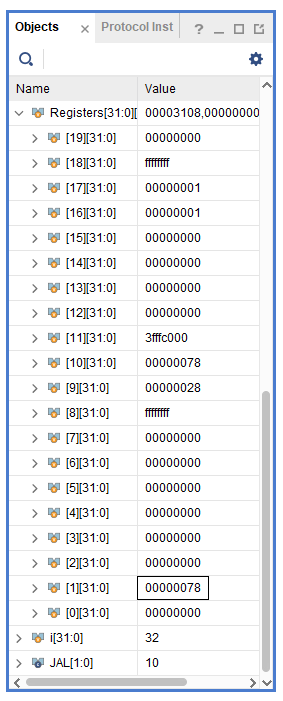
ad2b0000

0c000c43

08000c44

03e00008

1. 测试结果如下(regfile与dm在运行结束后的数据值)



1. 实验时间安排

1、在完成了单周期的基础上，积累了一些语法知识和verilog技巧，复习流水线时的冒险转发机制——2小时

2、写出段寄存器以及冒险，转发等模块——4小时

3、 将各个模块用数据通路整合起来，并在其中修改接口

使得整体更加完善，并最终完善整体的数据通路——6小时

4、写控制器，将各个部件的控制信号组合在一起，达到控制所有部件的运行的目的——6小时

5、利用vivado工具进行最后的调试——4小时

6、写实验报告——2小时

。。。

其实没有统计用时，主要调bug特别耗时间，总计用时起码40h+吧。

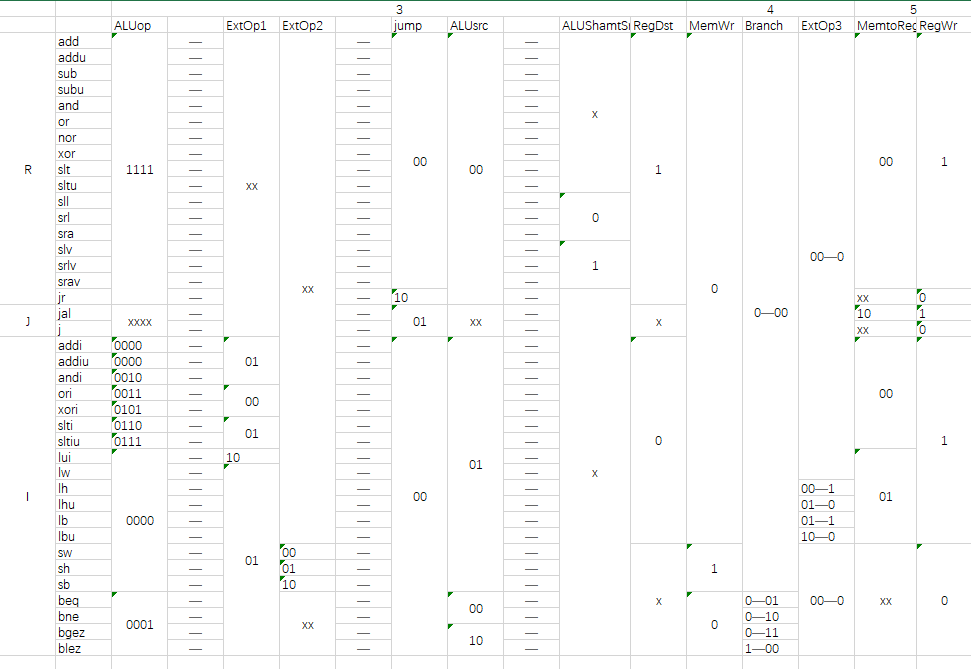
1. 心得体会

1．将任务细化后，可以化繁为简。

2. 流水与单周期很大程度上的不同，要时刻记得哪些变量，哪些模块，属于哪个阶段。所以在写时最好在命名变量时采用 “流水线级名称\_名称”的形式以免混淆

3.在写main\_ctrl.v时需要使用excel表格来确认每条指令控制信号的数据。

如下图所示



在完成课设过程中，我们对于流水线工作的流程更加了解，提升了自我。