P3 设计文档

21373459 王晨昊

1.设计内容

1.1 模块组成

本单周期 CPU 的大部分从 E 阶段到 W 阶段先后由 pc, im. controller, cmp, npc, alu, multdiv, dm 八个大模块连接组成。实现了一部分 MIPS 指令。

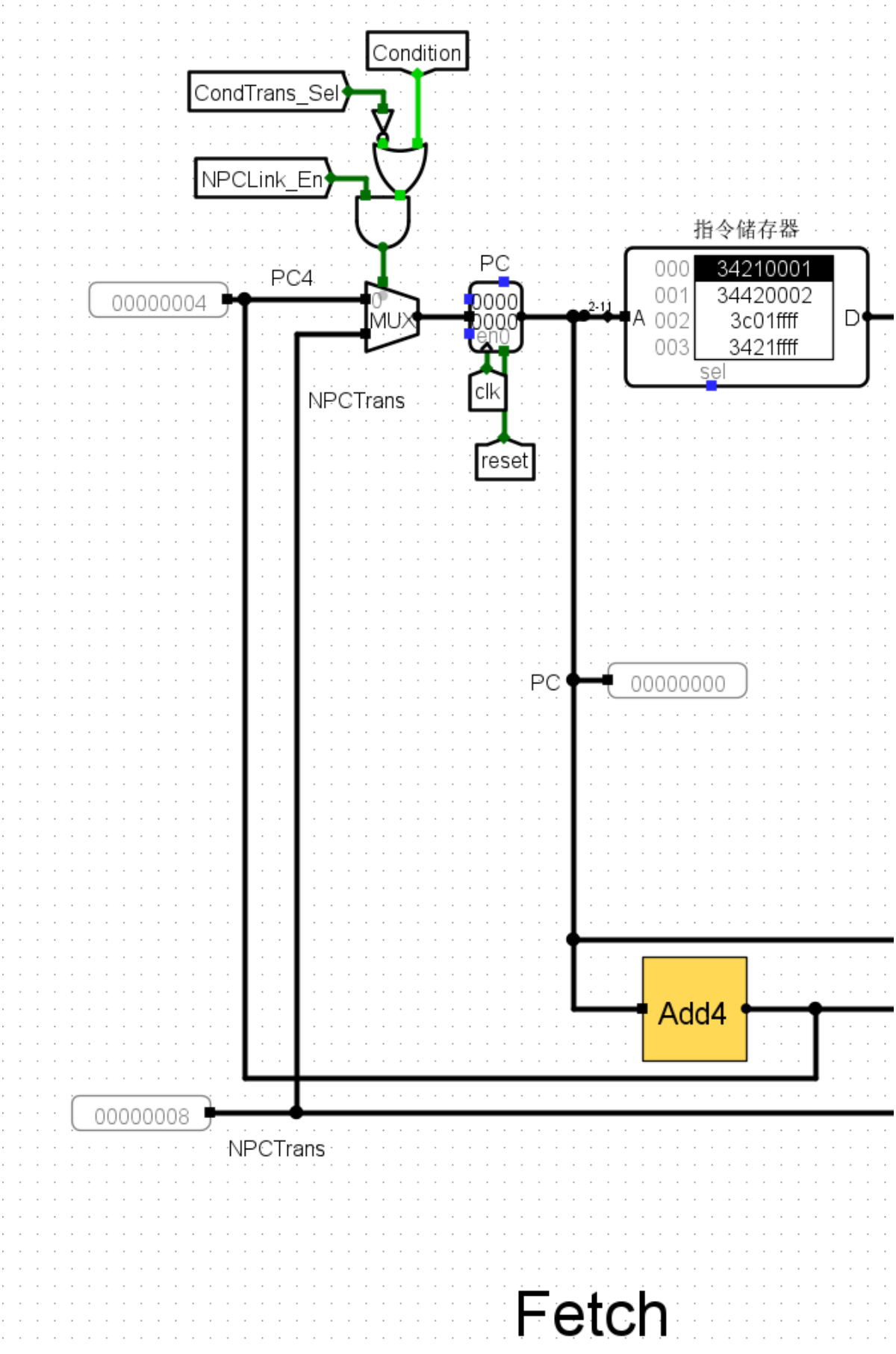
增加了 cmp 模块单独用于简单的比较，产生 Condition 条件信号用于控制有条件的跳转，写寄存器或写内存。

对于以上三者的某个信号来说，使能信号 使能 的值是

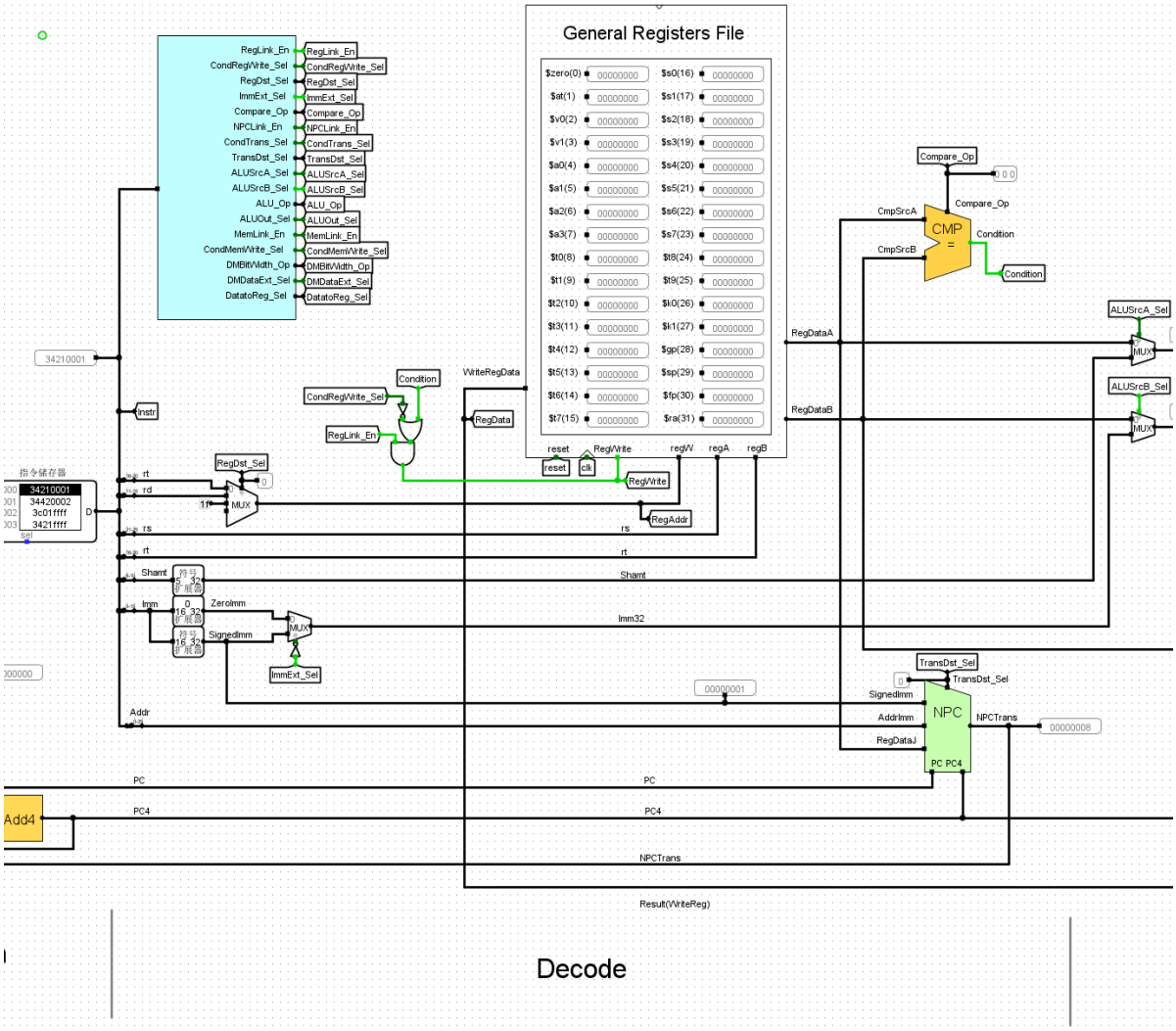
```
assign 使能 = 链接 & ( ~有条件 | 条件为真);
```

当 NPCTrans_En 为低电平时，意味着 PC 在下一周期只会被简单地更新为 PC4（即不存在非常规的跳转）。

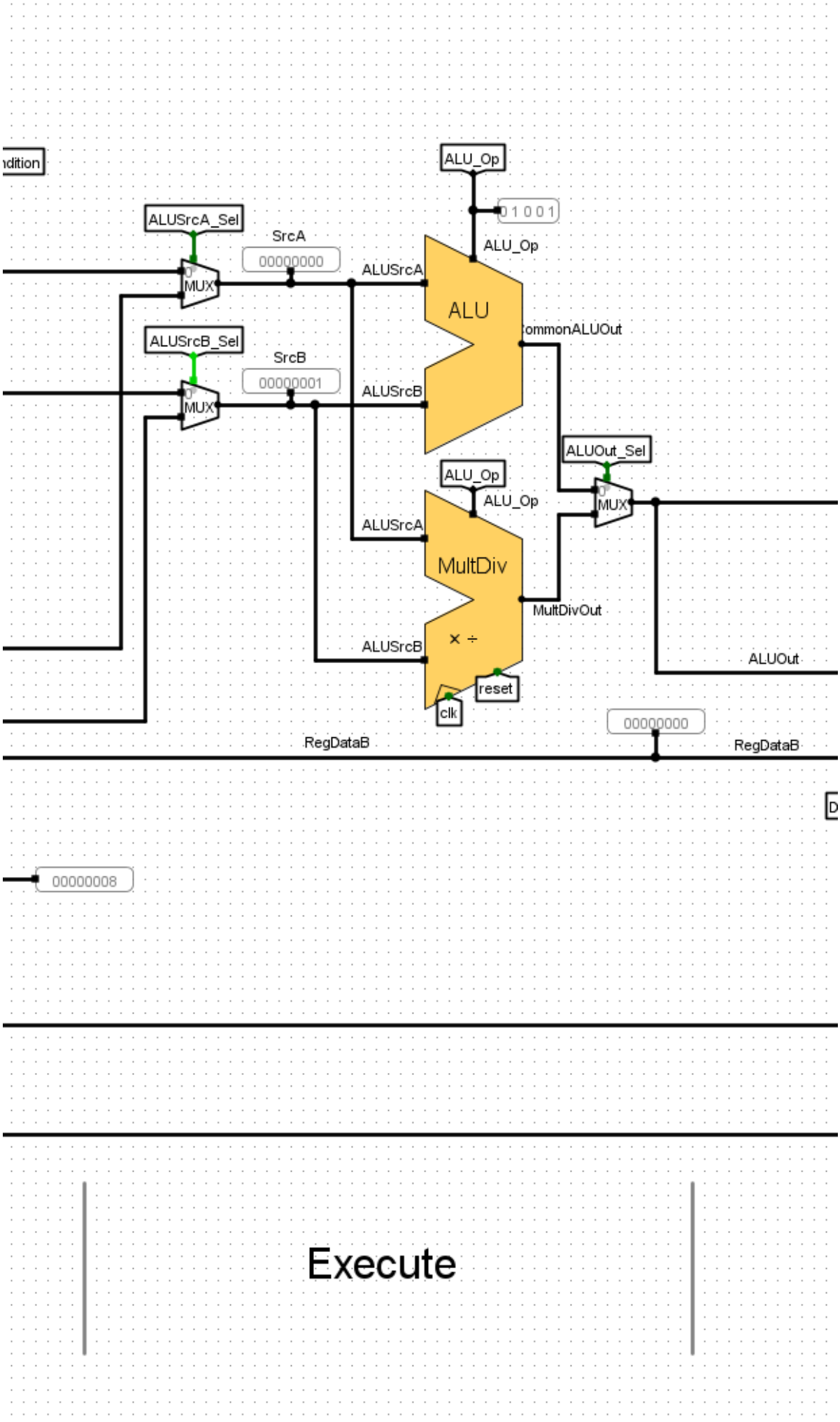
1.1.1 F 阶段


Condition
CondTrans_Sel
NPCLink_En
指令储存器
PC4
PC
00000004
0
MUX
1
0000
0000
en0
clk
reset
NPCTrans
2-11
A
000 34210001
001 34420002
002 3c01ffff
003 3421ffff
D
sel
PC
00000000
Add4
00000008
NPCTrans
Fetch

1.1.2 D 阶段

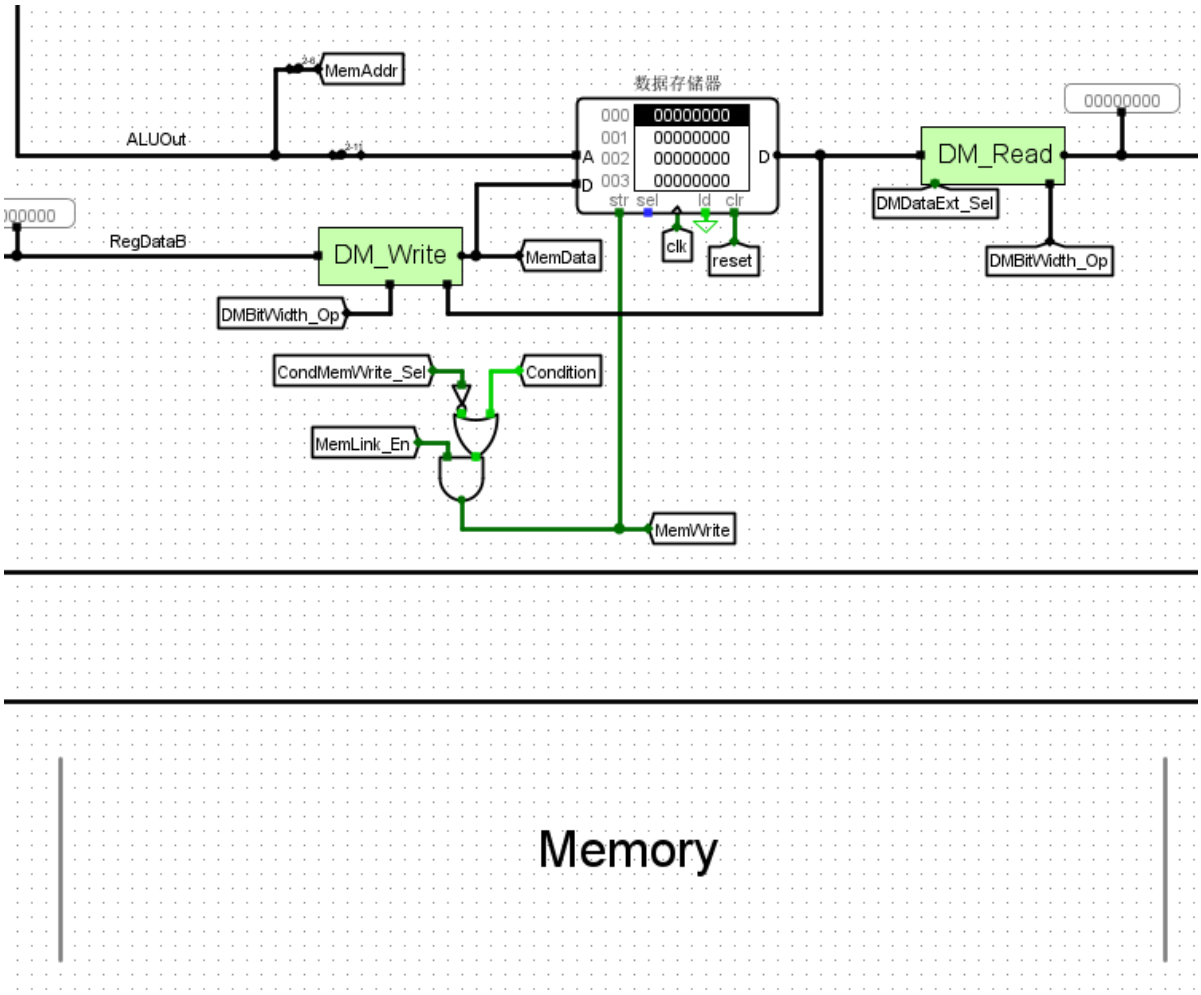

General Registers File
RegLink_En
CondRegWrite_Sel
RegDst_Sel
ImmExt_Sel
Compare_Op
NPCLink_En
CondTrans_Sel
TransDst_Sel
ALUSrcA_Sel
ALUSrcB_Sel
ALU_Op
ALUOut_Sel
MemLink_En
CondMemWrite_Sel
DMBitWidth_Op
DMDataExt_Sel
DatatoReg_Sel
34210001
Instr
Condition
WriteRegData
RegData
reset
clk
RegWrite
regW
regA
regB
RegAddr
rs
rt
Shamt
Imm32
ZeroImm
SignedImm
Addr
PC
PC4
NPCTrans
Result(WriteReg)
Compare_Op
CmpSrcA
CmpSrcB
CMP
Condition
RegDataA
RegDataB
ALUSrcA_Sel
ALUSrcB_Sel
MUX
TransDst_Sel
SignedImm
AddrImm
RegDataJ
NPC
PC PC4
00000008
00000001
Add4
Decode

1.1.3 E 阶段

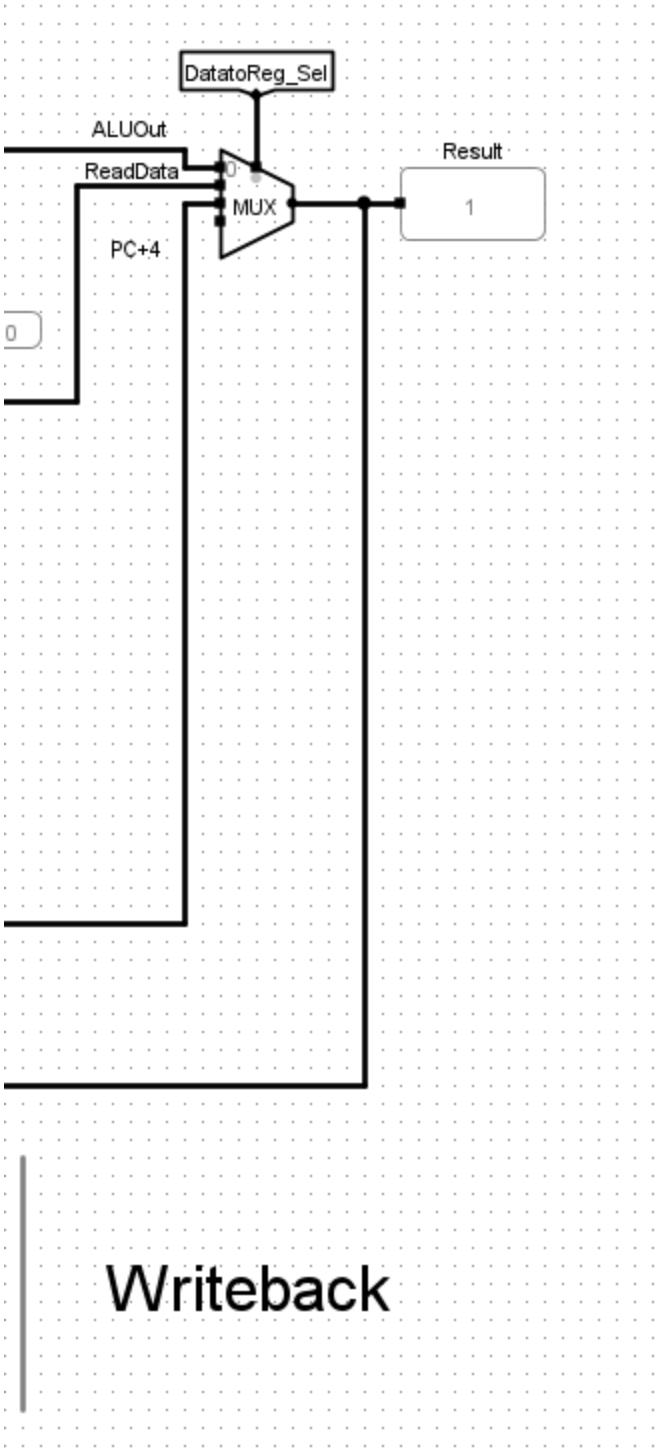

ALU_Op
01001
ALUSrcA_Sel
SrcA
00000000
ALU_Op
ALUSrcA
ALU
ALUSrcB_Sel
SrcB
00000001
ALUSrcB
ALUOut_Sel
ALU_Op
ALU_Op
ALUSrcA
MultDiv
MultDivOut
ALUSrcB
reset
clk
ALUOut
00000000
RegDataB
RegDataB
00000008

Execute

1.1.4 M 阶段



1.1.5 W 阶段



1.2 控制信号

阶段	信号	类型	作用	备注
Decode	RegLink_En	使能	写寄存器	是写寄存器信号RegWrite_En为1的必要条件
	CondRegWrite_Sel	复用	条件写寄存器	若为1，必须在Condition信号为1时才写入寄存器
	RegDst_Sel	复用	选择RegW来源	选择 rt(0) / rd(1) / $31(2) 作为RegW
	ImmExt_Sel	复	选择如何拓展	选择 符号拓展(0) / 0拓展(1)

		用	立即数	得到imm32
	Compare_Op[2:0]	操作数	比较方式	决定cmp模块比较的方式
	NPCLink_En	使能	PC跳转	是PC跳转信号NPCTrans_En为1的必要条件
	CondTrans_Sel	复用	条件PC跳转	若为1，必须在Condition信号为1时才跳转PC
	TransDst_Sel[1:0]	复用	选择NPCTrans来源	选择 偏移跳转(0) / 立即数跳转(1) / 寄存器跳转(2) 作为RegW
Execute	ALUSrcA_Sel	复用	选择ALUSrcA来源	选择 RegDataA(0) / Shamt(1) 作为ALUSrcA
	ALUSrcB_Sel	复用	选择ALUSrcB来源	选择 RegDataA(0) / Imm32(1) 作为ALUSrcB
	ALU_Op[4:0]	操作数	运算方式	决定alu和multdiv模块运算的方式
	ALUOut_Sel	复用	选择从何处获得ALUOut	选择 常规ALU(0) / 乘除模块(1) 得到ALUOut
Memory	MemLink_En	使能	写内存	是写内存信号MemWrite_En为1的必要条件
	CondMemWrite_Sel	复用	条件写内存	若为1，必须在Condition信号为1时才写内存
	DMBitWidth_Op	操作数	内存位宽	决定存取内存时的位宽大小
	DMDataExt_Sel	复用	运算方式	读内存的数据将经过 符号拓展(0) / 0拓展(1) 得到MemData
Writeback	DatatoReg_Sel	复用	选择WriteRegData来源	选择 ALUOut(0) / MemData(1) / PC4(2) 写寄存器

2 测试方案

文件提交的几个.asm 文件囊括了大部分常见的样例组合，包括大数据，小数据，正负数的测试。在实际测试中还使用了网上寻找的一个 mips 反汇编探针(mips probe)库。

此外，grf 文件的模块被我魔改成了可以从主电路直接看到内部数据的高交互性设计。电路中注释完备，探针丰富，易于检查。

下段代码用于基础的检验:

```
v2.0 raw
3c101234
36105678
36310001
00000000
36520006
02519820
36940007
12930002
36b50114
36d60514
36f70111
02934822
01344820
ae49fffe
8e8afffd
340b0001
340c0001
340d0006
116d0003
01cb7020
016c5820
1108fffc
ae8e0001
8e420002
```

```
lui $s0,0x1234
ori $s0,$s0,0x5678

ori $s1,1
nop
ori $s2,6
add $s3,$s2,$s1
ori $s4,7
beq $s4,$s3,lab
ori $s5,0x114
ori $s6,0x514
lab:
ori $s7,0x111
sub $t1,$s4,$s3
add $t1,$t1,$s4
sw $t1, -2($s2)
lw $t2, -3($s4)

ori $t3,$0,1
ori $t4,$0,1
ori $t5,$0,6
start:
beq $t3,$t5,end
add $t6,$t6,$t3
```

```
add $t3,$t3,$t4
beq $t0,$t0,start
end:
sw $t6 1($s4)
lw $v0 2($s2)
```

3 思考题

3.1

就本 CPU 而言，ifu(pc 和 im), dm, grf 起到存储功能， 而其余的 controller, alu, ext 起到状态转移功能。

3.2

合理。ROM 不能被写，只能被读，这种特点与 im 一致。RAM 能对某一个地址读或写，这种特点与 dm 一致。寄存器堆能够做到在不同的地址使多个不同的数据同时被写或者被读，这样才能满足 grf 执行指令时需要获得复数个操作数的需求。

3.3

实现时添加了 npc 模块用于计算将要跳转的地址。增加了 cmp 模块在 Decode 阶段判断条件，用于决定分支是否发声，寄存器是否条件写入等功能。增加了 multdiv 模块进行有时序参与的计算。

3.4

未实现 srl 指令时，controller 并不会为 nop 分配任何 NPCLink_En / RegLink_En / MemLink_En 信号，因此不会对电路状态带来影响。但实际上，即便是在实现 srl 指令后，它也只会将 32 位的 0 写入 0 号寄存器，而这是无效的操作。

3.5

让 DM 起始地址为 0，然后永远只截取 PC 的低 11 位并进行 0 拓展后再作为地址读取 IM 即可。

3.6

beq 没有向前跳转的例子。没有连续循环跳转的例子。缺乏特大绝对值数的加减法。没有尝试写入 0 寄存器的操作。dm 地址范围太小。