**同济大学计算机系**

**计算机组成原理实验报告**

****

**学 号 1750844**

**姓 名 周展田**

**专 业 计算机科学与技术**

**授课老师 张冬冬**

1. 实验内容

在本次实验中，将使用Verilog HDL实现54条MIPS指令的CPU的设计、前仿真、后仿真和下板调试运行。

1. 数据通路

（一）单独数据通路：

1 ADD

格式：ADD rd, rs, rt

操作：取指令，rd←rs+rt, PC←NPC(PC+4)

所需部件：PC, NPC, IMEM, Regfile, ALU

输入输出关系：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | | ALU | |
|  |  |  |  | Rd | Rdc | A | B |
| ADD | NPC | PC | PC | ALU | 15-11 | Rs | Rt |



2 ADDU

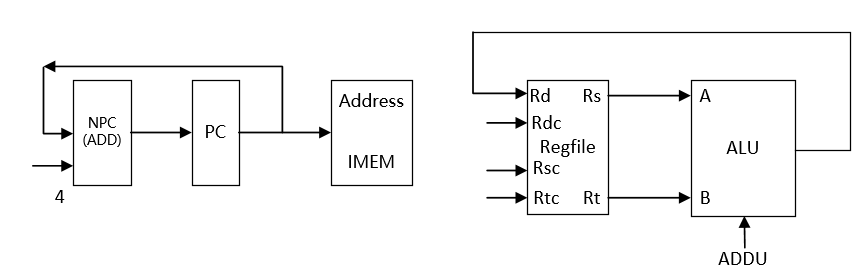
格式：ADDU rd, rs, rt

操作：取指令，rd←rs+rt, PC←NPC(PC+4)

所需部件：PC, NPC, IMEM, Regfile, ALU

输入输出关系：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | | ALU | |
|  |  |  |  | Rd | Rdc | A | B |
| ADDU | NPC | PC | PC | ALU | 15-11 | Rs | Rt |



3 SUB

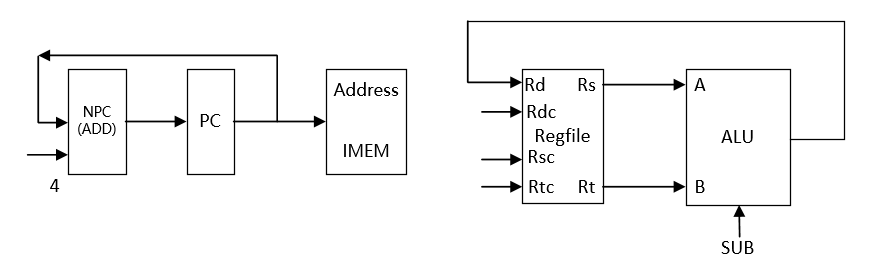
格式：SUB rd, rs, rt

操作：取指令，rd←rs-rt, PC←NPC(PC+4)

所需部件：PC, NPC, IMEM, Regfile, ALU

输入输出关系：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | | ALU | |
|  |  |  |  | Rd | Rdc | A | B |
| SUB | NPC | PC | PC | ALU | 15-11 | Rs | Rt |



4 SUBU

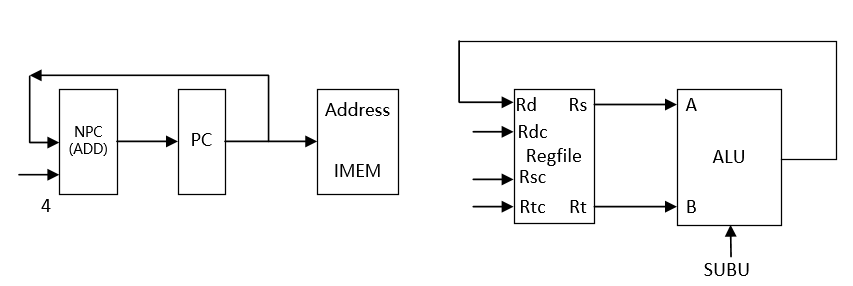
格式：SUBU rd, rs, rt

操作：取指令，rd←rs-rt, PC←NPC(PC+4)

所需部件：PC, NPC, IMEM, Regfile, ALU

输入输出关系：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | | ALU | |
|  |  |  |  | Rd | Rdc | A | B |
| SUBU | NPC | PC | PC | ALU | 15-11 | Rs | Rt |



5 AND

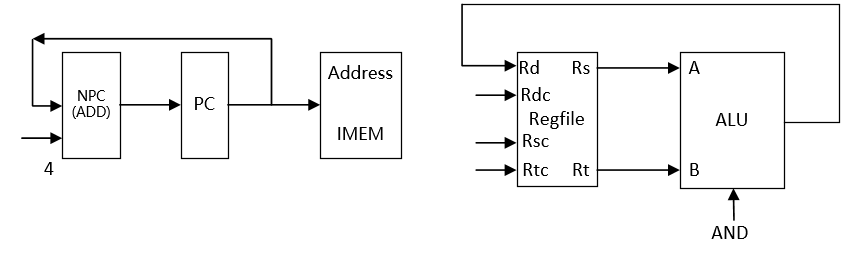
格式：AND rd, rs, rt

操作：取指令，rd←rs & rt, PC←NPC(PC+4)

所需部件：PC, NPC, IMEM, Regfile, ALU

输入输出关系：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | | ALU | |
|  |  |  |  | Rd | Rdc | A | B |
| AND | NPC | PC | PC | ALU | 15-11 | Rs | Rt |



6 OR

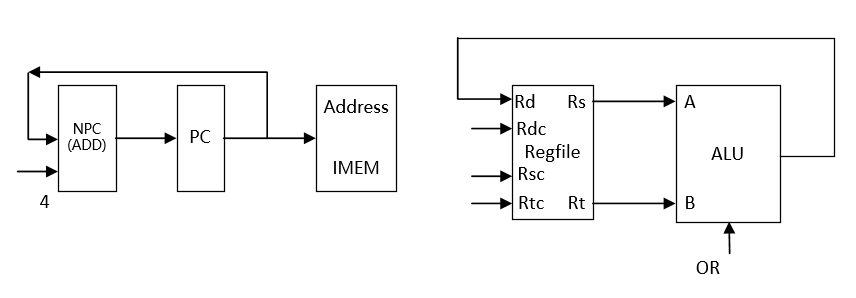
格式：OR rd, rs, rt

操作：取指令，rd←rs | rt, PC←NPC(PC+4)

所需部件：PC, NPC, IMEM, Regfile, ALU

输入输出关系：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | | ALU | |
|  |  |  |  | Rd | Rdc | A | B |
| OR | NPC | PC | PC | ALU | 15-11 | Rs | Rt |



7 XOR

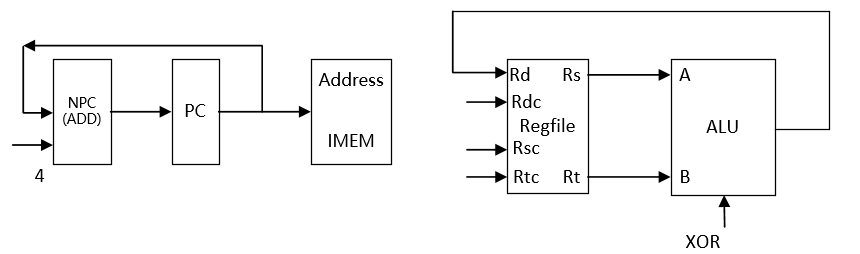
格式：XOR rd, rs, rt

操作：取指令，rd←rs ^ rt, PC←NPC(PC+4)

所需部件：PC, NPC, IMEM, Regfile, ALU

输入输出关系：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | | ALU | |
|  |  |  |  | Rd | Rdc | A | B |
| XOR | NPC | PC | PC | ALU | 15-11 | Rs | Rt |



8 NOR

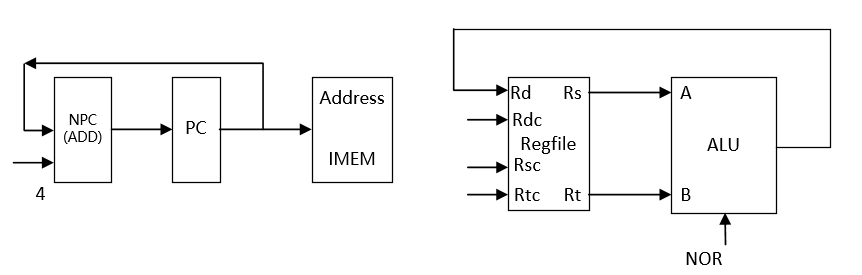
格式：NOR rd, rs, rt

操作：取指令，rd←~(rs | rt), PC←NPC(PC+4)

所需部件：PC, NPC, IMEM, Regfile, ALU

输入输出关系：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | | ALU | |
|  |  |  |  | Rd | Rdc | A | B |
| NOR | NPC | PC | PC | ALU | 15-11 | Rs | Rt |



9 SLT

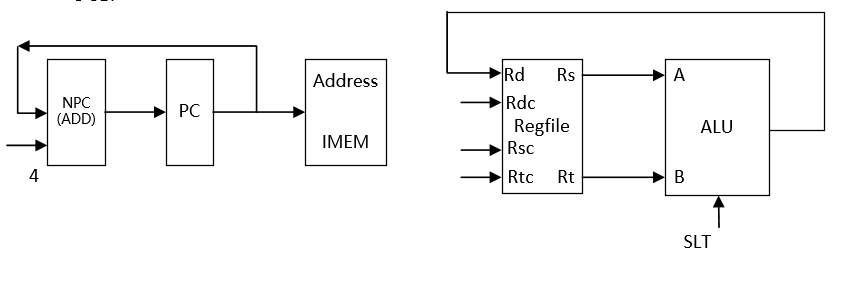
格式：SLT rd, rs, rt

操作：取指令，rd← rs < rt , PC←NPC(PC+4)

所需部件：PC, NPC, IMEM, Regfile, ALU

输入输出关系：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | | ALU | |
|  |  |  |  | Rd | Rdc | A | B |
| SLT | NPC | PC | PC | ALU | 15-11 | Rs | Rt |



10 SLTU

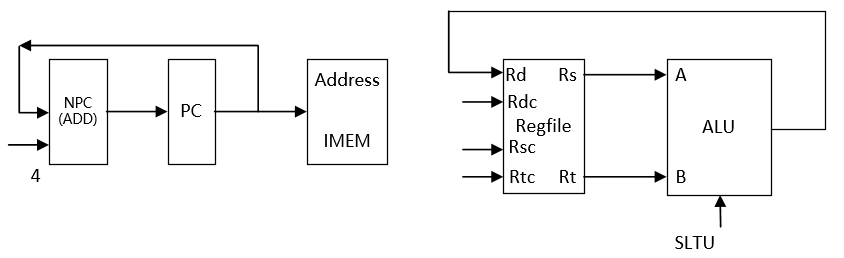
格式：SLTU rd, rs, rt

操作：取指令，rd← rs < rt , PC←NPC(PC+4)

所需部件：PC, NPC, IMEM, Regfile, ALU

输入输出关系：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | | ALU | |
|  |  |  |  | Rd | Rdc | A | B |
| SLTU | NPC | PC | PC | ALU | 15-11 | Rs | Rt |



11 SLL

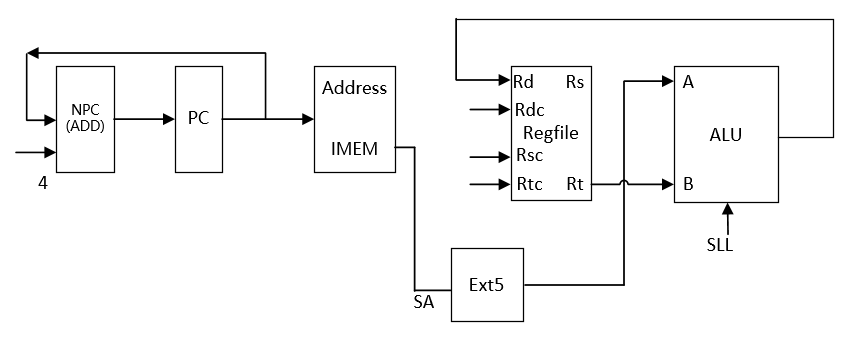
格式：SLL rd, rt, sa

操作：取指令, rd←rt << sa , PC←NPC(PC+4)

所需部件：PC, NPC, IMEM, Regfile, ALU, Ext5

输入输出关系：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | | ALU | | Ext5 |
|  |  |  |  | Rd | Rdc | A | B |  |
| SLL | NPC | PC | PC | ALU | 15-11 | Ext5 | Rt | Sa |



12 SRL

格式：SRL rd, rt, sa

操作：取指令, rd←rt >> sa (logical) , PC←NPC(PC+4)

所需部件：PC, NPC, IMEM, Regfile, ALU, Ext5

输入输出关系：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | | ALU | | Ext5 |
|  |  |  |  | Rd | Rdc | A | B |  |
| SRL | NPC | PC | PC | ALU | 15-11 | Ext5 | Rt | Sa |



13 SRA

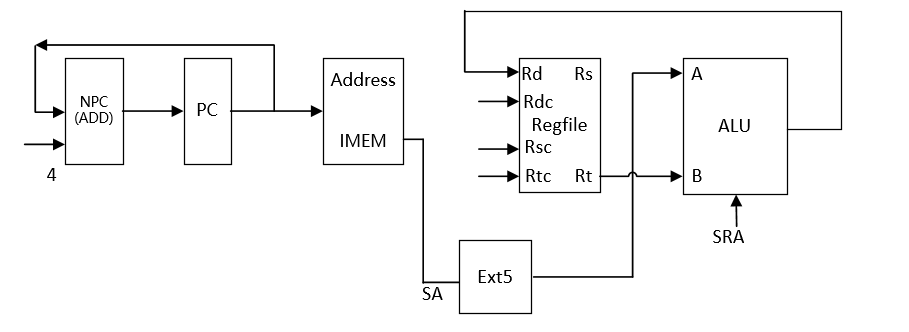
格式：SRA rd, rt, sa

操作：取指令, rd←rt >> sa (arithmetic) , PC←NPC(PC+4)

所需部件：PC, NPC, IMEM, Regfile, ALU, Ext5

输入输出关系：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | | ALU | | Ext5 |
|  |  |  |  | Rd | Rdc | A | B |  |
| SRA | NPC | PC | PC | ALU | 15-11 | Ext5 | Rt | Sa |



14 SLLV

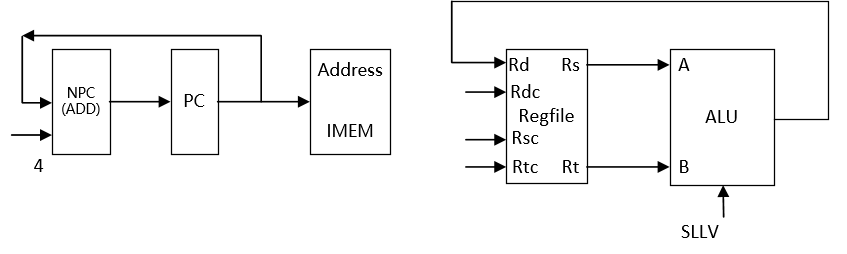
格式：SLLV rd, rt, rs,

操作：取指令, rd←rt << rs , PC←NPC(PC+4)

所需部件：PC, NPC, IMEM, Regfile, ALU

输入输出关系：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | | ALU | |
|  |  |  |  | Rd | Rdc | A | B |
| SLLV | NPC | PC | PC | ALU | 15-11 | Rs | Rt |



15 SRLV

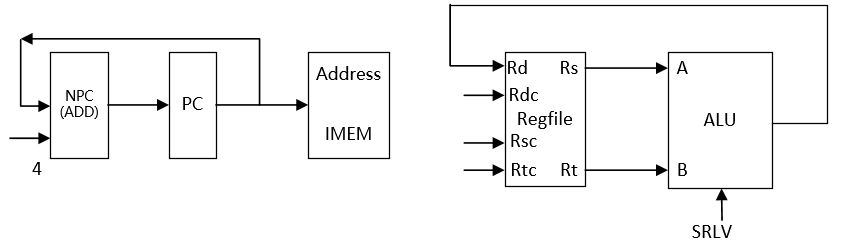
格式：SRLV rd, rt,rs

操作：取指令, rd←rt >> rs (logical) , PC←NPC(PC+4)

所需部件：PC, NPC, IMEM, Regfile, ALU

输入输出关系：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | | ALU | |
|  |  |  |  | Rd | Rdc | A | B |
| SRLV | NPC | PC | PC | ALU | 15-11 | Rs | Rt |



16 SRAV

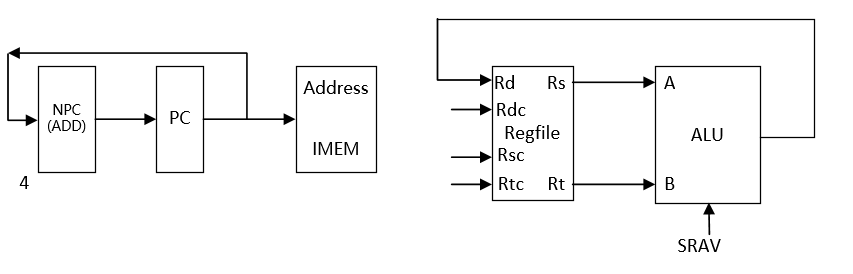
格式：SRAV rd, rt,rs

操作：取指令, rd←rt >> rs (arithmetic) , PC←NPC(PC+4)

所需部件：PC, NPC, IMEM, Regfile, ALU

输入输出关系：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | | ALU | |
|  |  |  |  | Rd | Rdc | A | B |
| SRAV | NPC | PC | PC | ALU | 15-11 | Rs | Rt |



17 JR

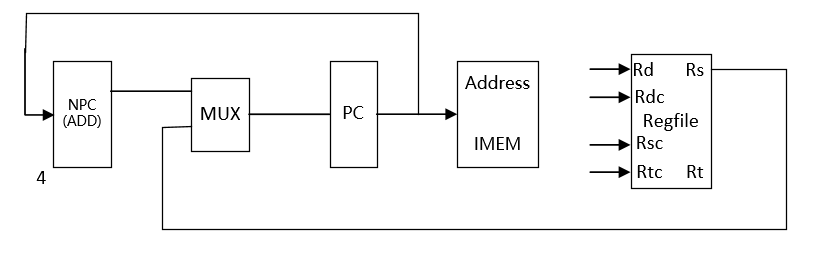
格式：JR rs

操作：取指令, PC←rs, PC←NPC(PC+4)

所需部件：PC, NPC, IMEM, Regfile

输入输出关系：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | | ALU | |
|  |  |  |  | Rd | Rdc | A | B |
| JR | Rs | PC | PC |  |  |  |  |



18 ADDI

格式：ADDI rt, rs, imm16

操作：取指令、rt←rs+imm16(sign\_extend) 、PC←NPC(PC+4)

所需部件：：PC、NPC、IMEM、Rregfile、ALU、Ext16

输入输出关系：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | | ALU | | Ext16 |
|  |  |  |  | Rd | Rdc | A | B |  |
| ADDI | NPC | PC | PC | ALU | 20-16 | Rs | Ext16 | Imm16 |



19 ADDIU

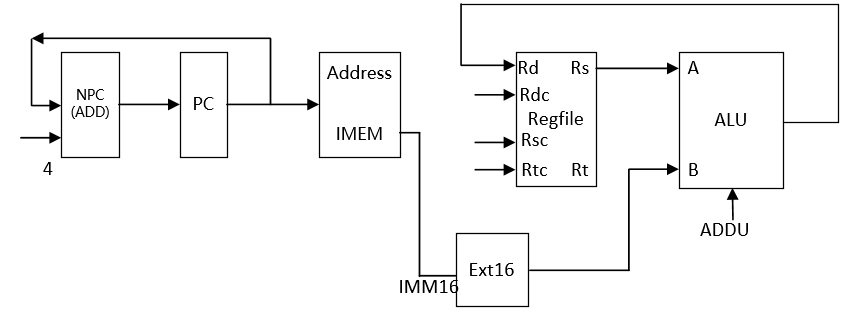
格式：ADDIU rt, rs, imm16

操作：取指令、rt←rs+imm16(sign\_extend) 、PC←NPC(PC+4)

所需部件：PC、NPC、IMEM、Rregfile、ALU、Ext16

输入输出关系：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | | ALU | | Ext16 |
|  |  |  |  | Rd | Rdc | A | B |  |
| ADDIU | NPC | PC | PC | ALU | 20-16 | Rs | Ext16 | Imm16 |



20 ANDI

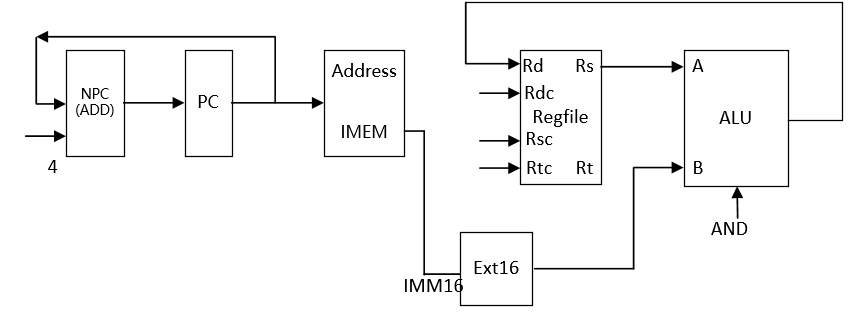
格式：ANDI rt, rs, imm16

操作：取指令、rt←rs&imm16(zero\_extend) 、PC←NPC(PC+4)

所需部件：PC、NPC、IMEM、Rregfile、ALU、Ext16

输入输出关系：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | | ALU | | Ext16 |
|  |  |  |  | Rd | Rdc | A | B |  |
| ANDI | NPC | PC | PC | ALU | 20-16 | Rs | Ext16 | Imm16 |



21 ORI

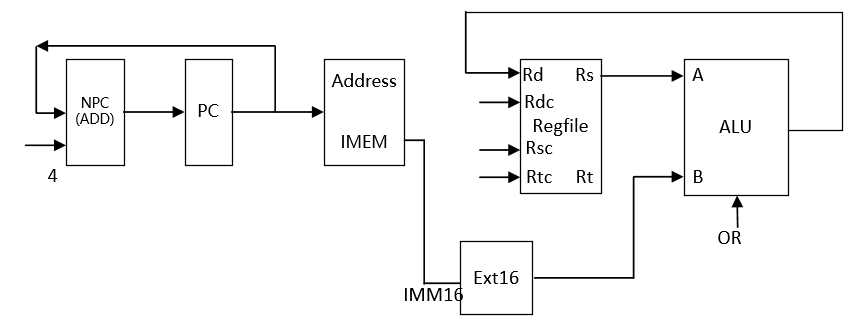
格式：ORI rt, rs, imm16

操作：取指令、rt←rs|imm16(zero\_extend) 、PC←NPC(PC+4)

所需部件：PC、NPC、IMEM、Rregfile、ALU、Ext16

输入输出关系：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | | ALU | | Ext16 |
|  |  |  |  | Rd | Rdc | A | B |  |
| ORI | NPC | PC | PC | ALU | 20-16 | Rs | Ext16 | Imm16 |



22 XORI

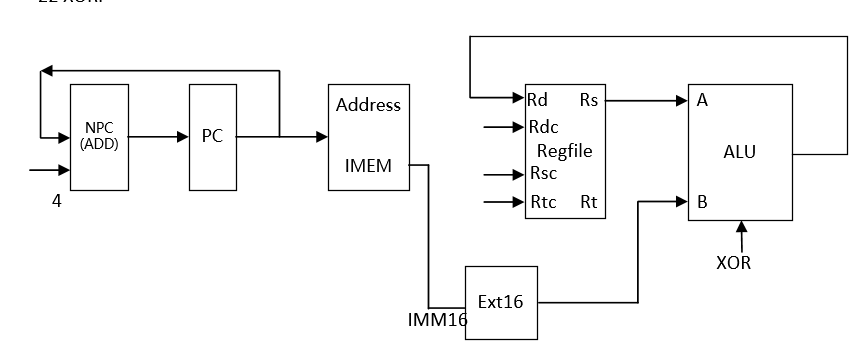
格式：XORI rt, rs, imm16

操作：取指令、rt←rs^imm16(zero\_extend) 、PC←NPC(PC+4)

所需部件：PC、NPC、IMEM、Rregfile、ALU、Ext16

输入输出关系：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | | ALU | | Ext16 |
|  |  |  |  | Rd | Rdc | A | B |  |
| XORI | NPC | PC | PC | ALU | 20-16 | Rs | Ext16 | Imm16 |



23 LW

格式：LW rt, offset(base)

操作：取指令、rt←memory[rs + offset]、PC←NPC(PC+4)

所需部件：PC、NPC、IMEM、ALU、Ext16、DMEM

输入输出关系：

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | | ALU | | Ext16 | DMEM | |
|  |  |  |  | Rd | Rdc | A | B |  | Addr | Data |
| LW | NPC | PC | PC | DMEM  (data) | 20-16 | Rs  (base) | Ext16 | Offset | ALU |  |



24 SW

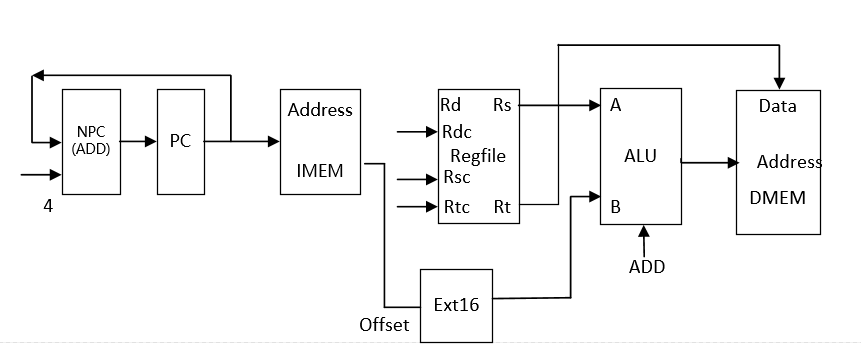
格式：SW rt, offset(base)

操作：取指令、memory[base + offset]←rt、PC←NPC(PC+4)

所需部件：PC、NPC、IMEM、ALU、Ext16、DMEM

输入输出关系：

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | | ALU | | Ext16 | DMEM | |
|  |  |  |  | Rd | Rdc | A | B |  | Addr | Data |
| SW | NPC | PC | PC | DMEM  (data) |  | Rs  (base) | Ext16 | Offset | ALU | Rt |



25 BEQ

格式：BEQ rs, rt, offset

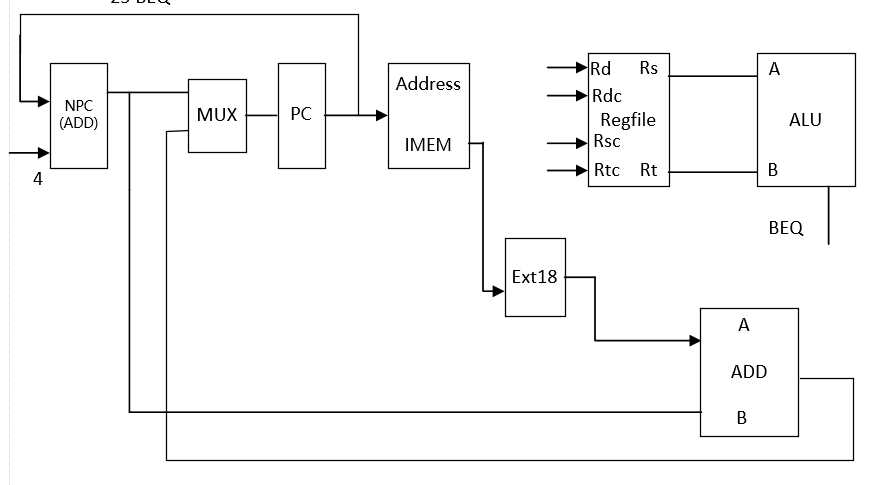
操作: if (rs=rt)PC←NPC + Sign\_ext(offset||02)

else PC← NPC(PC+4)

所需部件：PC、NPC、IMEM、Regfile、ALU、Ext18、ADD

输入输出关系：

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | | ALU | | Ext18 | ADD | |
|  |  |  |  | Rd | Rdc | A | B |  | A | B |
| BEQ | ADD | PC | PC |  |  | Rs | Rt | Offset | Ext18 | NPC |



26 BNE

格式：BNE rs, rt, offset

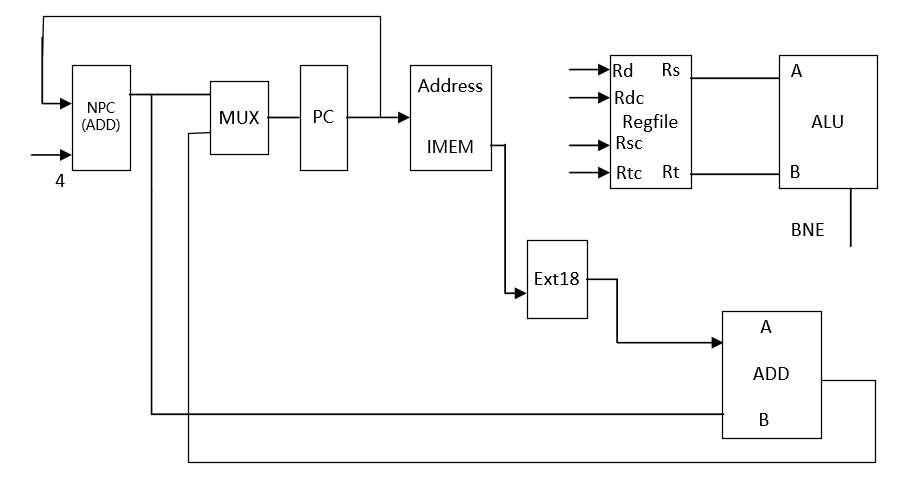
操作：if (rs≠rt)PC←NPC + Sign\_ext(offset||02)

else PC← NPC(PC+4)

所需部件：PC、NPC、IMEM、Regfile、ALU、Ext18、ADD

输入输出关系：

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | | ALU | | Ext18 | ADD | |
|  |  |  |  | Rd | Rdc | A | B |  | A | B |
| BNE | ADD | PC | PC |  |  | Rs | Rt | Offset | Ext18 | NPC |



27 SLTI

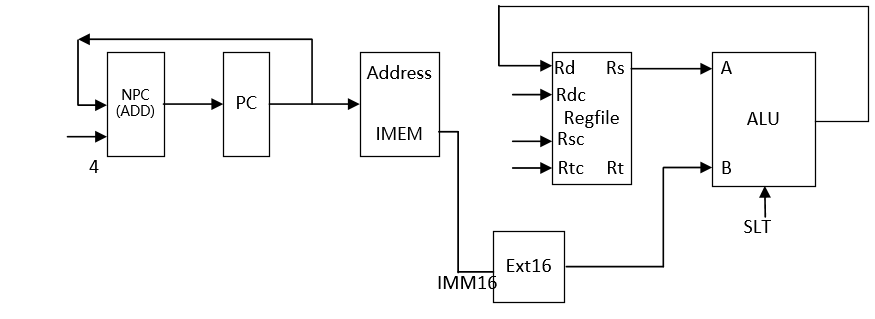
格式：SLTI rt, rs, imm16

操作：取指令、rt←rs < ext.imm16(sign\_extend) 、PC←NPC(PC+4)

所需部件：PC、NPC、IMEM、Rregfile、ALU、Ext16

输入输出关系：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | | ALU | | Ext16 |
|  |  |  |  | Rd | Rdc | A | B |  |
| SLTI | NPC | PC | PC | ALU | 20-16 | Rs | Ext16 | Imm16 |



28 SLTIU

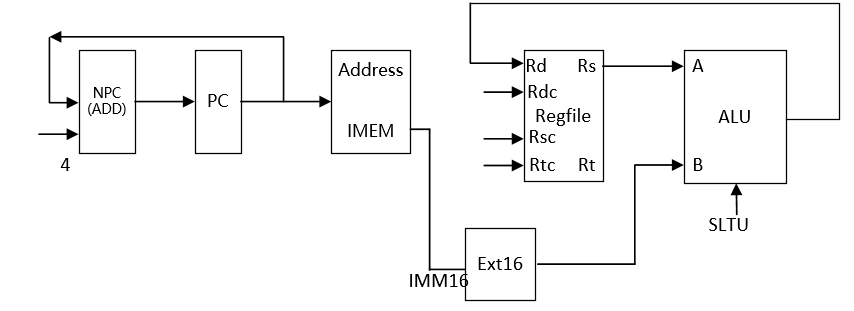
格式：SLTIU rt, rs, imm16

操作：取指令、rt←rs < ext.imm16(sign\_extend) 、PC←NPC(PC+4)

所需部件：PC、NPC、IMEM、Rregfile、ALU、Ext16

输入输出关系：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | | ALU | | Ext16 |
|  |  |  |  | Rd | Rdc | A | B |  |
| SLTIU | NPC | PC | PC | ALU | 20-16 | Rs | Ext16 | Imm16 |



29 LUI

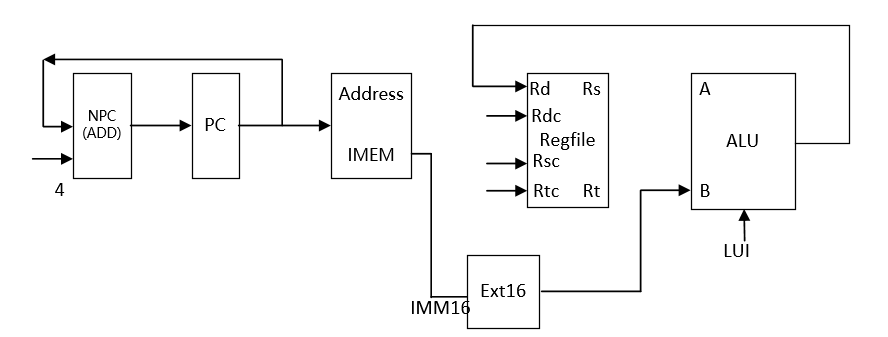
格式：LUI rt, imm16

操作：取指令、rt←imm16||016 、PC←NPC(PC+4)

所需部件：PC、NPC、IMEM、Rregfile、ALU、Ext16

输入输出关系：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | | ALU | | Ext16 |
|  |  |  |  | Rd | Rdc | A | B |  |
| LUI | NPC | PC | PC | ALU | 20-16 |  | Ext16 | Imm16 |



30 J

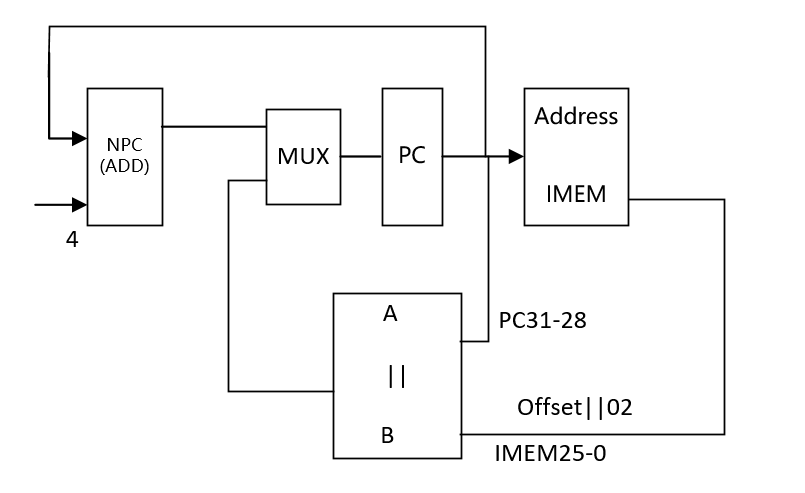
格式：J target

操作：取指令、PC ← PC31-28||instr\_index||02 , PC←NPC(PC+4)

所需部件：PC、NPC、IMEM

输入输出关系：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | | ALU | |  |
|  |  |  |  | Rd | Rdc | A | B |  |
| J | || | PC | PC |  |  |  |  |  |



31 JAL

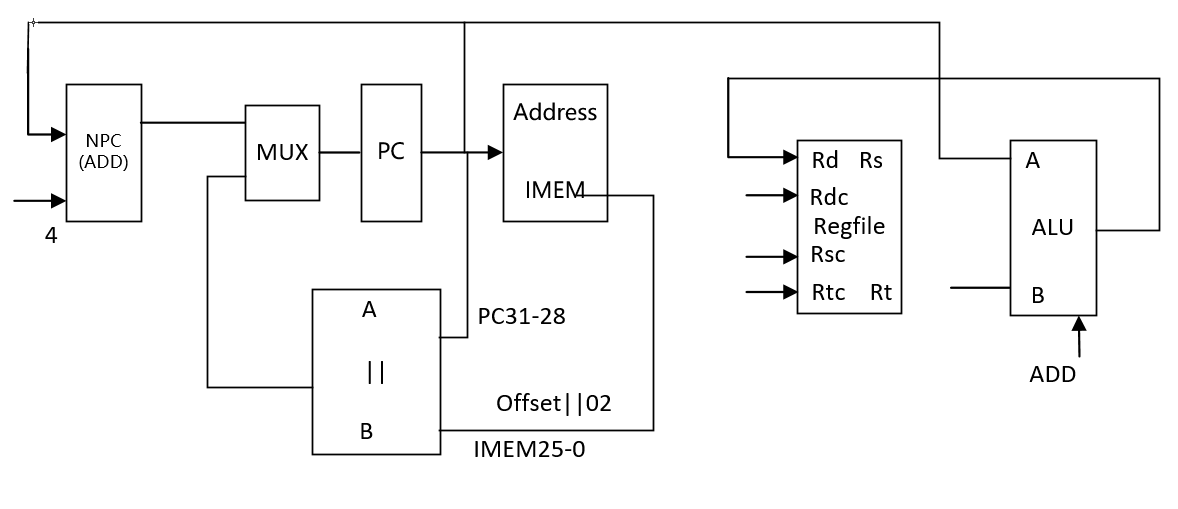
格式：JAL target

操作：取指令、 R[31] ← PC + 8,PC ← PC31-28||instr\_index||02 , PC←PC+4

所需部件：：PC、NPC、IMEM

输入输出关系：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | | ALU | |  |
|  |  |  |  | Rd | Rdc | A | B |  |
| JAL | || | PC | PC | ALU |  | PC | 8 |  |



32 BREAK

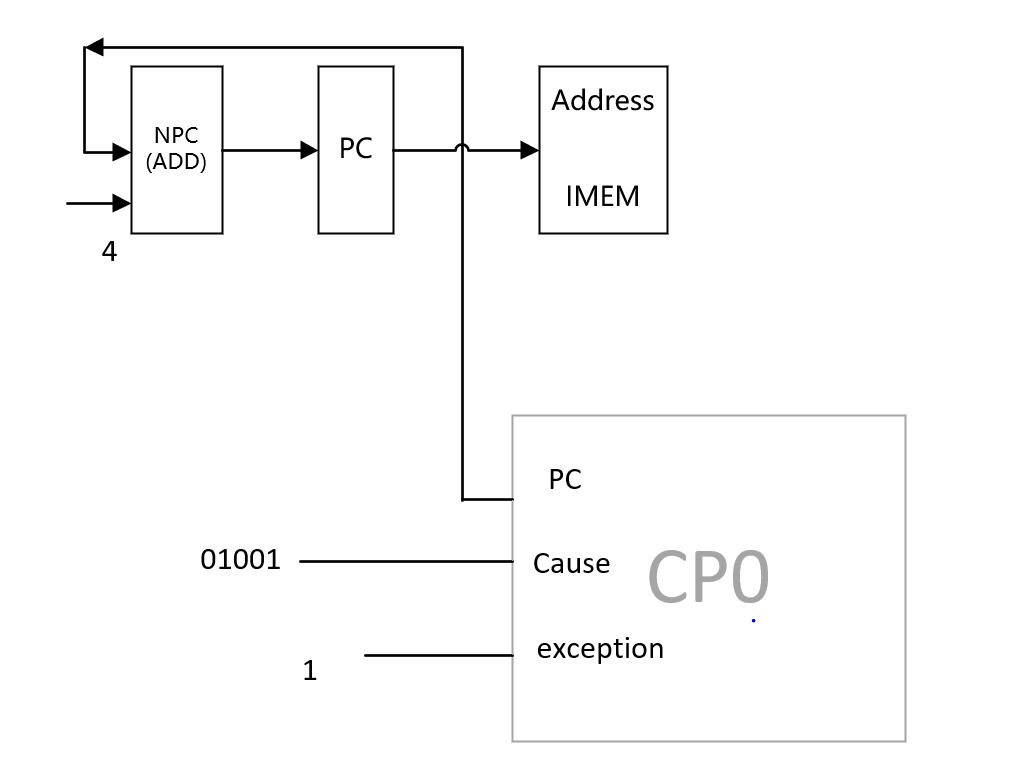
格式：BREAK MIPS32

操作：取指令、 EPC ← PC, cause ← 01001, status<<5, PC←PC+4

所需部件：PC、NPC、CP0

输入输出关系：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM |  | EPC | cause | exception |
| BREAK | NPC | PC | PC |  | PC | 01001 | 1 |



33 SYSCALL

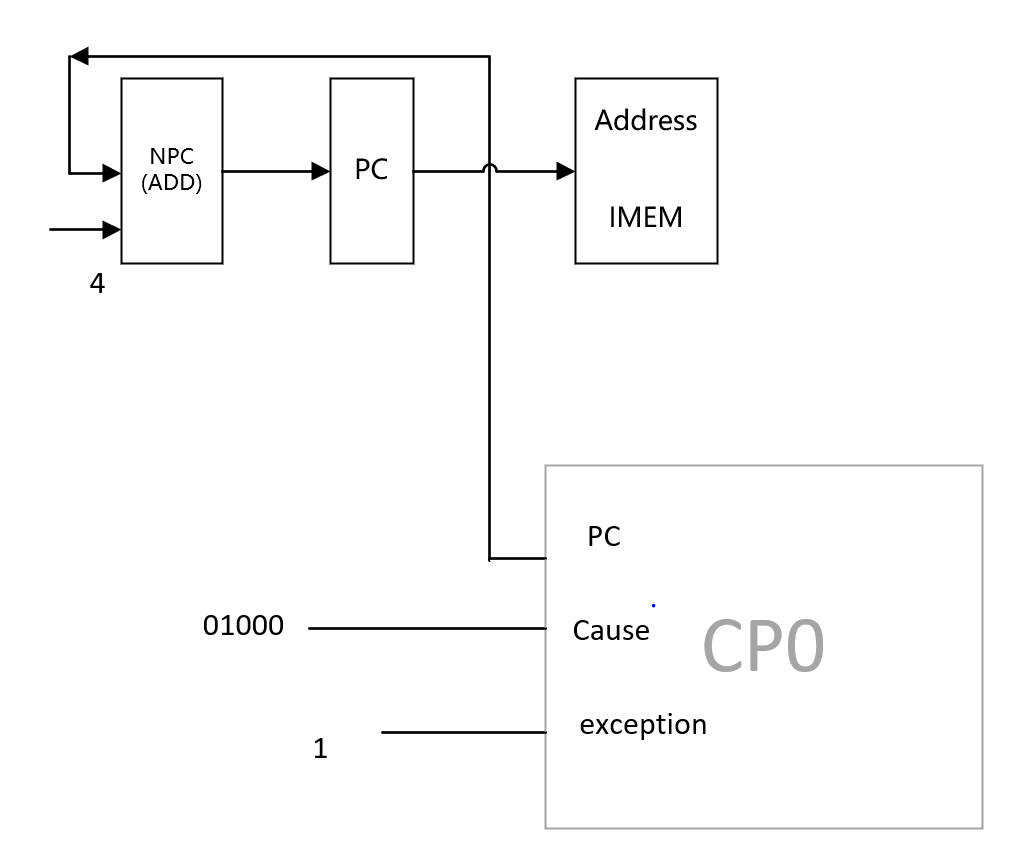
格式：SYSCALL

操作：取指令、 EPC ← PC, cause ← 01000, status<<5, PC←PC+4

所需部件：PC、NPC、CP0

输入输出关系：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | EPC | cause | exception |
| break | NPC | PC | PC | PC | 01000 | 1 |



34 TEQ

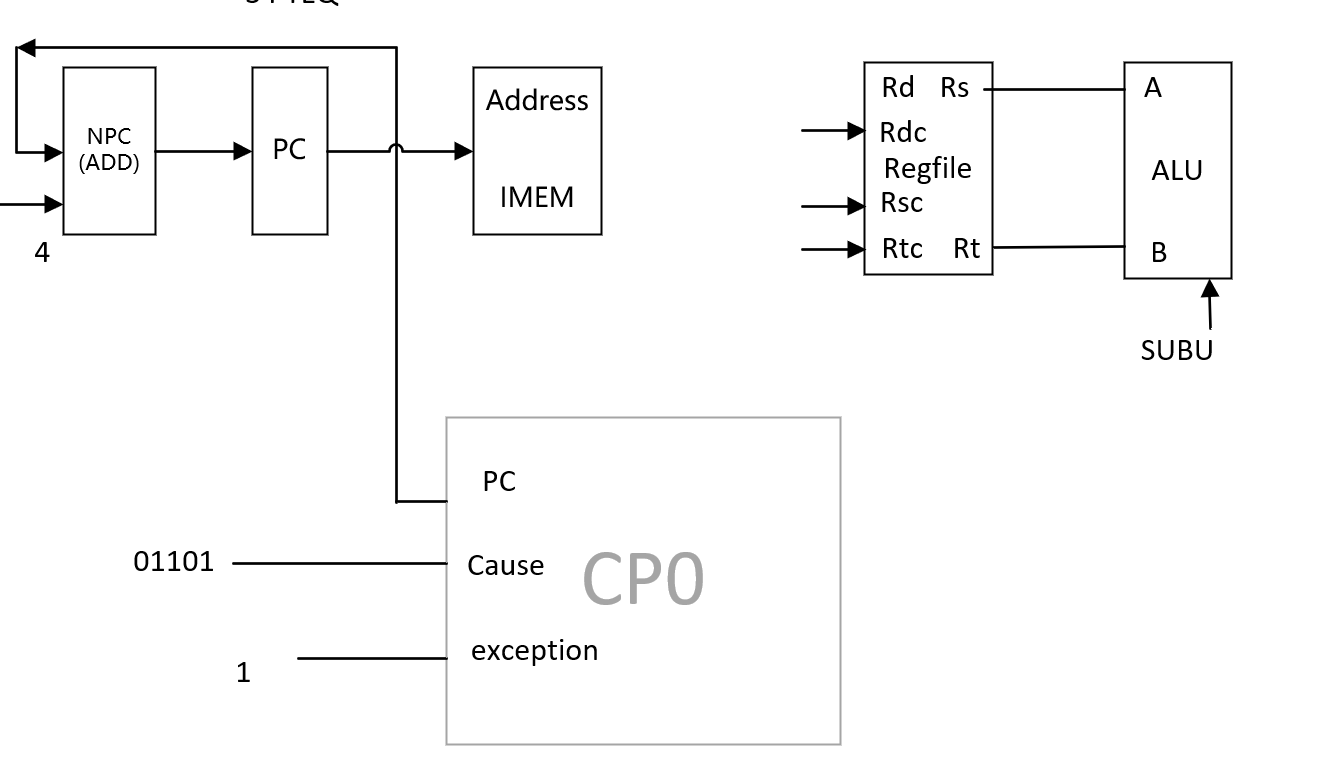
格式：TEQ rs,rt

操作取指令、 R[rs]-R[rt]、EPC ← PC、cause ← 01000, status<<5, PC←PC+4

所需部件：PC、NPC、CP0、IMEM、Regfiles、ALU

输入输出关系：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | EPC | cause | exception | ALU | |
|  |  |  |  |  |  |  | A | B |
| teq | NPC | PC | PC | PC | 01101 | 1 | Rs | Rt |



35 ERET

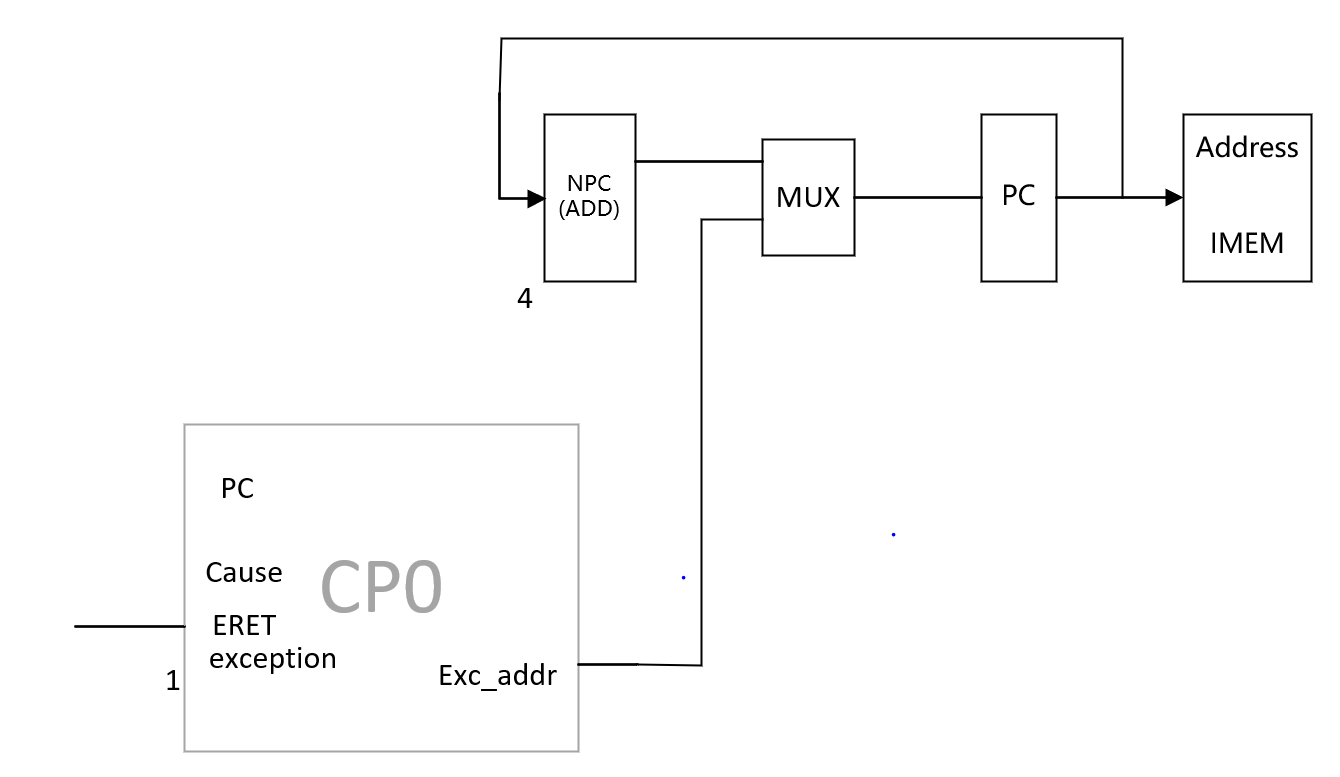
格式：ERET

操作：取指令、 PC ← EPC, PC←PC+4

所需部件: PC、NPC、CP0

输入输出关系：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM |  |
| eret | EPC | PC | PC |  |



36 MFC0

格式：MFC0 rt,rd

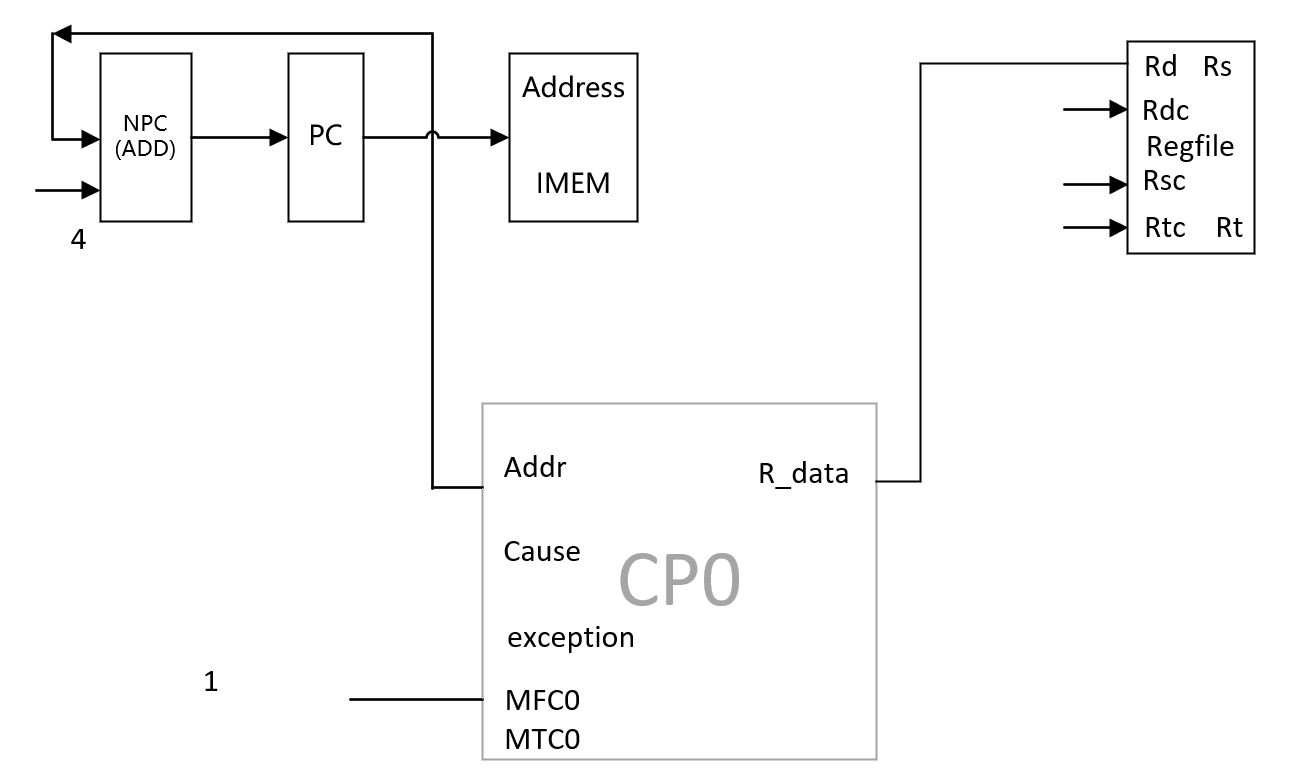
MFC0 rt,rd,sel

操作：取指令、 R[rt] ← CP0 R[rd], PC←PC+4

所需部件：PC、NPC、CP0、Rregfile

输入输出关系：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | RegFile |  |
|  |  |  |  | Rd |  |
| MFC0 | NPC | PC | PC | Rdata |  |



37 MTC0

格式：MTC0 rt,rd

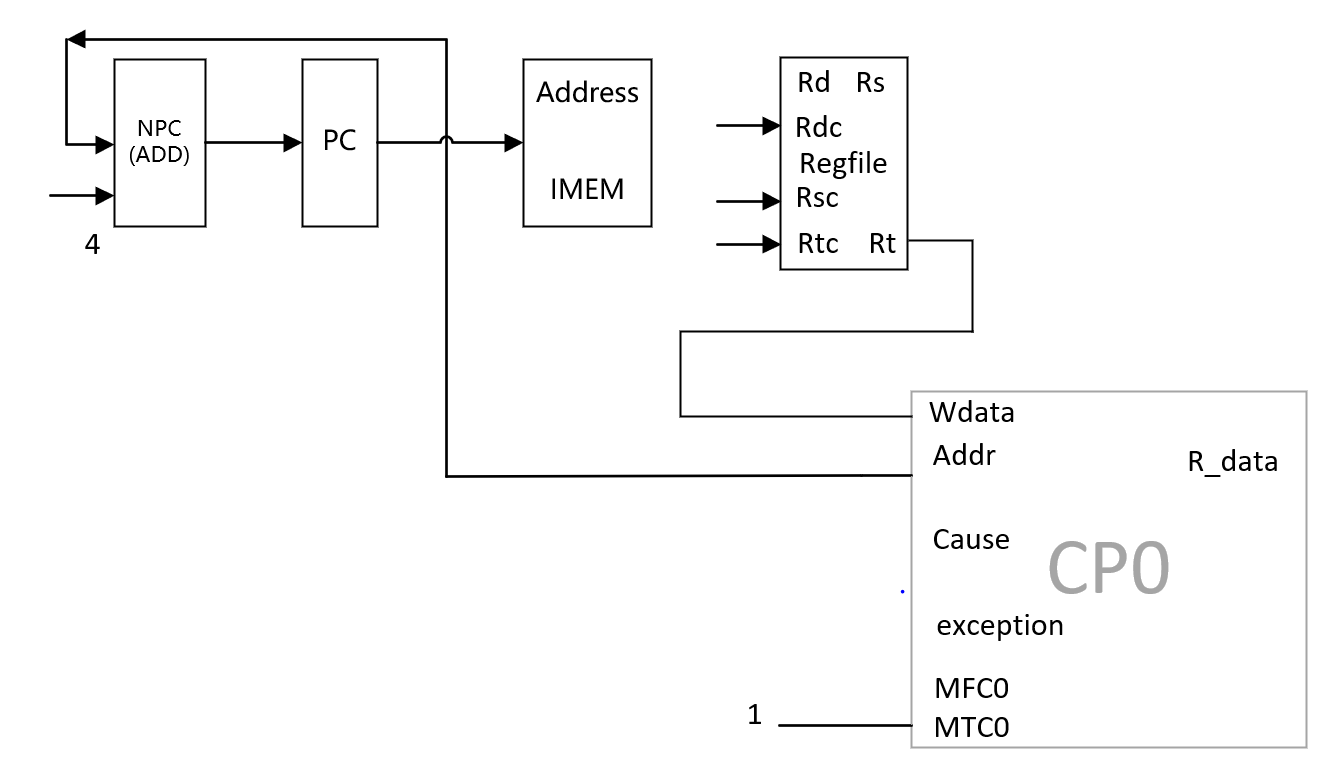
MTC0 rt,rd,sel

操作：取指令、 CP0 R[rd] ←R[rt], PC←PC+4

所需部件：PC、NPC、CP0、Rregfile

输入输出关系：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM |  |  |
| MTC0 | NPC | PC | PC |  |  |



38、Clz rd rs

所需的操作: 取指令、R[rd] ←count\_leading\_zeros R[rs], PC←PC+4

所需器件：PC、NPC、Rregfile



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | RegFile |
|  |  |  |  | Rd |
| clz | NPC | PC | PC | TEMP |

39、DIVU rd rs

所需的操作: 取指令、(HI,LO)←R[rd]/R[rt], PC←PC+4

所需器件：PC、NPC、Rregfile、DIVU、LO、HI



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | DIVU | | LO | HI |
|  |  |  |  | dividend | divisor |  |  |
| divu | npc | pc | pc | rs | rt | q | r |

40、DIV rd rs

所需的操作: 取指令、(HI,LO)←R[rd]/R[rt], PC←PC+4

所需器件：PC、NPC、Rregfile、DIV、LO、HI



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | DIV | | LO | HI |
|  |  |  |  | dividend | divisor |  |  |
| divu | npc | pc | pc | rs | rt | q | r |

41、Lb rt,offset(base);

所需的操作: 取指令、R[rt]←memory[R[base] + offset]、PC←PC+4

所需器件：PC、NPC、IMEM、ALU、Ext16、DMEM



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | RegFile | ALU | | Ext16 | Ext5 | DMEM | |
|  |  |  |  | Rd | A | B |  |  | Addr | Data |
| Lb | NPC | PC | PC | DMEM  (data) | Rs  (base) | Ext16 | offset |  | ALU |  |

42、Lbu rt,offset(base);

所需的操作: 取指令、R[rt]←memory[R[base] + offset]、PC←PC+4

所需器件：PC、NPC、IMEM、ALU、Ext16、DMEM



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | RegFile | ALU | | Ext16 | Ext5 | DMEM | |
|  |  |  |  | Rd | A | B |  |  | Addr | Data |
| Lbu | NPC | PC | PC | DMEM  (data) | Rs  (base) | Ext16 | offset |  | ALU |  |

43、Lhu rt,offset(base);

所需的操作: 取指令、R[rt]←memory[R[base] + offset]、PC←PC+4

所需器件：PC、NPC、IMEM、ALU、Ext16、DMEM



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | RegFile | ALU | | Ext16 | Ext5 | DMEM | |
|  |  |  |  | Rd | A | B |  |  | Addr | Data |
| Lhu | NPC | PC | PC | DMEM  (data) | Rs  (base) | Ext16 | offset |  | ALU |  |

44、Lh rt,offset(base);

所需的操作: 取指令、R[rt]←memory[R[base] + offset]、PC←PC+4

所需器件：PC、NPC、IMEM、ALU、Ext16、DMEM



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | RegFile | ALU | | Ext16 | Ext5 | DMEM | |
|  |  |  |  | Rd | A | B |  |  | Addr | Data |
| Lh | NPC | PC | PC | DMEM  (data) | Rs  (base) | Ext16 | offset |  | ALU |  |

45、Sb rt,offset(base);

所需的操作: 取指令、memory[R[base] + offset] ← R[rt]、PC←PC+4

所需器件：PC、NPC、IMEM、ALU、Ext16、DMEM



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | RegFile | ALU | | Ext16 | Ext5 | DMEM | |
|  |  |  |  | Rd | A | B |  |  | Addr | Data |
| Sb | NPC | PC | PC | DMEM  (data) | Rs  (base) | Ext16 | offset |  | ALU | Rt |

46、Sh rt,offset(base);

所需的操作: 取指令、memory[R[base] + offset] ← R[rt]、PC←PC+4

所需器件：PC、NPC、IMEM、ALU、Ext16、DMEM



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | RegFile | ALU | | Ext16 | Ext5 | DMEM | |
|  |  |  |  | Rd | A | B |  |  | Addr | Data |
| Sh | NPC | PC | PC | DMEM  (data) | Rs  (base) | Ext16 | offset |  | ALU | Rt |

47、mfhi rd

所需的操作: 取指令、 R[rd] ← HI、PC←PC+4

所需器件：PC、NPC、IMEM、RegFile、HI



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | RegFile |
|  |  |  |  | Rd |
| mfhi | NPC | PC | PC | HI |

48、mflo rd

所需的操作: 取指令、 R[rd] ← LO、PC←PC+4

所需器件：PC、NPC、IMEM、RegFile、LO



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | RegFile |
|  |  |  |  | Rd |
| MfLO | NPC | PC | PC | HI |

49、mthi rd

所需的操作: 取指令、 HI←R[rs]、PC←PC+4

所需器件：PC、NPC、IMEM、RegFile、HI



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | HI |
| mthi | NPC | PC | PC | Rs |

50、mtlo rd

所需的操作: 取指令、 LO←R[rs]、PC←PC+4

所需器件：PC、NPC、IMEM、RegFile、LO



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | LO |
| mtlo | NPC | PC | PC | Rs |

51、MULT rd rs

所需的操作: 取指令、(HI,LO)←R[rs] \* R[rt], PC←PC+4

所需器件：PC、NPC、IMEM、Rregfile、MUL、



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | RegFile | MUL | |
|  |  |  |  | Rd |  | |
| MULT | npc | pc | pc | HI | rs | rt |

52、MULTU rd rs

所需的操作: 取指令、(HI,LO)←R[rs] \* R[rt], PC←PC+4

所需器件：PC、NPC、IMEM、Rregfile、MULU、



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | RegFile | MULU | |
|  |  |  |  | Rd |  | |
| MULTU | npc | pc | pc | HI | rs | rt |

53、Bgez rs,rt,offset ;

所需的操作: if (rs≠rt)PC←NPC + Sign\_ext(offset||02)

else PC← NPC(PC+4)

所需器件：PC、NPC、IMEM、Regfile、ALU、Ext18、ADD



|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | RegFile | ALU | | Ext16 | Ext5 | Ext18 | ADD | |
|  |  |  |  | Rd | A | B |  |  |  | A | B |
| Bgez | ADD | PC | PC |  | Rs | 0 |  |  | Offset | NPC | Ext18 |

54、Jalr rd rs

所需的操作: 取指令、 R[rd] ← PC + 8,PC ← R[rs] , PC←PC+4

所需器件：PC、NPC、IMEM



|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | RegFile | ALU | | Ext16 | Ext5 |
|  |  |  |  | Rd | A | B |  |  |
| Jalr | Rs | PC | PC | ALU | PC | 8 |  |  |

（二）整个数据通路

****

（三）部件表：

**所需部件表**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | PC | NPC | IMEM | RegFile | ALU | | Ext16 | Ext5 | Ext18 | DMEM | |
|  |  |  |  |  | Rd | A | B |  |  |  | Addr | Data |
| 1 | Add | NPC | PC | PC | ALU | Rs | Rt |  |  |  |  |  |
| 2 | Addu | NPC | PC | PC | ALU | Rs | Rt |  |  |  |  |  |
| 3 | Sub | NPC | PC | PC | ALU | Rs | Rt |  |  |  |  |  |
| 4 | Subu | NPC | PC | PC | ALU | Rs | Rt |  |  |  |  |  |
| 5 | and | NPC | PC | PC | ALU | Rs | Rt |  |  |  |  |  |
| 6 | or | NPC | PC | PC | ALU | Rs | Rt |  |  |  |  |  |
| 7 | xor | NPC | PC | PC | ALU | Rs | Rt |  |  |  |  |  |
| 8 | nor | NPC | PC | PC | ALU | Rs | Rt |  |  |  |  |  |
| 9 | slt | NPC | PC | PC | ALU | Rs | Rt |  |  |  |  |  |
| 10 | sltu | NPC | PC | PC | ALU | Rs | Rt |  |  |  |  |  |
| 11 | sllv | NPC | PC | PC | ALU | Rs | Rt |  |  |  |  |  |
| 12 | srlv | NPC | PC | PC | ALU | Rs | Rt |  |  |  |  |  |
| 13 | srav | NPC | PC | PC | ALU | Rs | Rt |  |  |  |  |  |
| 14 | sll | NPC | PC | PC | ALU | Ext5 | Rt |  | sa |  |  |  |
| 15 | srl | NPC | PC | PC | ALU | Ext5 | Rt |  | sa |  |  |  |
| 16 | sra | NPC | PC | PC | ALU | Ext5 | Rt |  | sa |  |  |  |
| 17 | Addi | NPC | PC | PC | ALU | Rs | Ext16 | Imm16 |  |  |  |  |
| 18 | Addiu | NPC | PC | PC | ALU | Rs | Ext16 | Imm16 |  |  |  |  |
| 19 | Andi | NPC | PC | PC | ALU | Rs | Ext16 | Imm16 |  |  |  |  |
| 20 | Ori | NPC | PC | PC | ALU | Rs | Ext16 | Imm16 |  |  |  |  |
| 21 | Xori | NPC | PC | PC | ALU | Rs | Ext16 | Imm16 |  |  |  |  |
| 22 | Slti | NPC | PC | PC | ALU | Rs | Ext16 | Imm16 |  |  |  |  |
| 23 | Sltiu | NPC | PC | PC | ALU | Rs | Ext16 | Imm16 |  |  |  |  |
| 24 | Lui | NPC | PC | PC | ALU |  | Ext16 | Imm16 |  |  |  |  |
| 25 | Lw | NPC | PC | PC | DMEM (data) | Rs (base) | Ext16 | offset |  |  | ALU |  |
| 26 | Lbu | NPC | PC | PC | DMEM (data) | Rs (base) | Ext16 | offset |  |  | ALU |  |
| 27 | Lhu | NPC | PC | PC | DMEM (data) | Rs (base) | Ext16 | offset |  |  | ALU |  |
| 28 | Lh | NPC | PC | PC | DMEM (data) | Rs (base) | Ext16 | offset |  |  | ALU |  |
| 29 | Lhu | NPC | PC | PC | DMEM (data) | Rs (base) | Ext16 | offset |  |  | ALU |  |
| 30 | Sw | NPC | PC | PC | DMEM (data) | Rs (base) | Ext16 | offset |  |  | ALU | Rt |
| 31 | Sb | NPC | PC | PC | DMEM (data) | Rs (base) | Ext16 | offset |  |  | ALU | Rt |
|
| 32 | Sh | NPC | PC | PC | DMEM (data) | Rs (base) | Ext16 | offset |  |  | ALU | Rt |
|

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | PC | NPC | IMEM | RegFile | ALU | | Ext16 | Ext5 | Ext18 | ADD | | || | |
|  |  |  |  |  | Rd | A | B |  |  |  | A | B | A | B |
| 33 | Beq | ADD | PC | PC |  | Rs | Rt |  |  | Offset | NPC | Ext18 |  |  |
| 34 | Bne | ADD | PC | PC |  | Rs | Rt |  |  | Offset | NPC | Ext18 |  |  |
| 35 | Bgez | ADD | PC | PC |  |  |  |  |  | Offset | NPC | Ext18 |  |  |
| 36 | J | || | PC | PC |  |  |  |  |  |  |  |  | PC  31-28 | IMEM 25-0 |
| 37 | Jal | || | PC | PC | ALU | PC | 8 |  |  |  |  |  | PC  31-28 | IMEM 25-0 |
| 38 | jr | rs | PC | PC |  |  |  |  |  |  |  |  |  |  |
| 39 | jalr | rs | PC | PC | ALU | PC | 8 |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | PC | NPC | IMEM | RegFile | ALU | | cp0 | | | |
|  |  |  |  |  | Rd | A | B | wdata | epc | cause | exception |
| 40 | eret | EPC | PC | PC |  |  |  |  |  |  |  |
| 41 | mfc0 | NPC | PC | PC | Rdata |  |  |  |  |  |  |
| 42 | mtc0 | NPC | PC | PC |  |  |  | Rt |  |  |  |
| 43 | syscall | NPC | PC | PC |  |  |  |  | PC | 01000 | 1 |
| 44 | teq | NPC | PC | PC |  | Rs | Rt |  | PC | 01101 | 1 |
| 45 | break | NPC | PC | PC |  |  |  |  | PC | 01001 | 1 |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | PC | NPC | IMEM | RegFile | DIV/MUL | | DIV/MULTU | | LO | HI |
|  |  |  |  |  | Rd | dividend | divisor | dividend | divisor |  |  |
| 46 | clz | NPC | PC | PC | TEMP |  |  |  |  |  |  |
| 47 | divu | NPC | PC | PC |  | rs | rt |  |  | q | r |
| 48 | div | NPC | PC | PC |  |  |  | rs | rt | q | r |
| 49 | mul | NPC | PC | PC |  | rs | rt |  |  |  |  |
| 50 | multu | NPC | PC | PC |  |  |  | rs | rt | r[31:0] | r[63:32] |
| 51 | mfhi | NPC | PC | PC | HI |  |  |  |  |  |  |
| 52 | mflo | NPC | PC | PC | LO |  |  |  |  |  |  |
| 53 | mthi | NPC | PC | PC |  |  |  |  |  |  | Rs |
| 54 | mtlo | NPC | PC | PC |  |  |  |  |  | Rs |  |

1. 模块建模

1 顶层模块sccomp\_dataflow，调用cpu,imem和dram

module sccomp\_dataflow(

input clk\_in,

input reset,

// output [7:0] o\_seg,

// output [7:0] o\_sel

output [31:0] inst,

output [31:0] pc

);

wire [31:0] imm;

wire [31:0] Rt;

wire [31:0] alu\_r;

wire cs;

wire dm\_w;

wire dm\_r;

wire [31:0] ram\_out;

wire clk\_out;

wire [2:0]select;

// wire [31:0] inst;

// wire [31:0] pc;

assign inst=imm;

assign clk\_out=clk\_in;

//divider div(clk\_in,reset,clk\_out);

//seg7x16 seg(.clk(clk\_out),

// .reset(reset),

//.cs(1),

// .i\_data(pc),

//.o\_seg(o\_seg),

// .o\_sel(o\_sel));

imem imem(((pc- 32'h00400000)/4),imm);

//imem im(pc,inst)

cpu sccpu(clk\_out,reset,imm,ram\_out, //input

Rt,select,alu\_r,pc,cs,dm\_w,dm\_r); //output

DMEM dram(

.clk(clk\_out),

.CS(cs), //enable control signal

.DM\_W(dm\_w), //write

.DM\_R(dm\_r), //read

.select(select),

.addr((alu\_r[31:0]-32'h10010000)),

.data\_in(Rt),

.data\_out(ram\_out)

);

endmodule

2 cpu模块 发送控制信号

module cpu(

input clk\_in,

input reset,

input [31:0]imem,

input [31:0] ram\_out,

output [31:0] Rt,

output [2:0]select,//DMEM 8bit 16 bit 32 bit

output [31:0] alu\_r,

output [31:0] pc,

output cs, //dmem control signal

output dm\_w, //dmem write

output dm\_r//dmem read

);

//wire [31:0]Rt;

wire [31:0]npc;

wire M3,M3\_2,M4,M4\_2, M2,M5,M1,M1\_2,M6,M7; //mux2

wire ALUC3,ALUC2,ALUC1,ALUC0;

wire RF\_W; //regfiles write

wire RF\_CLK; //regfiles clk

wire [3:0]ALUC;//alu control

wire [31:0] PC;

wire [31:0] MUX1;

wire [31:0] MUX1\_2;

reg [31:0]Rd;

// wire [31:0] MUX2;

wire [31:0] MUX3;

wire [31:0] MUX3\_2;

wire [31:0] mux\_in\_4;

wire [31:0] MUX4;

wire [31:0] MUX4\_2;

wire [31:0] MUX5;

wire [31:0] Rs; //Rs

wire [31:0] ext18\_sign;

wire [4:0] rdc;

wire [4:0] rsc;

wire [4:0] rtc;

wire [31:0] ext16;

wire [31:0] ext16\_sign;

wire ext16\_sin\_judge;

wire zero;

wire carry;

wire negative;

wire overflow;

assign pc = PC;

wire \_add, \_addu, \_sub, \_subu, \_and, \_or, \_xor, \_nor;

wire \_slt, \_sltu, \_sll, \_srl, \_sra, \_sllv, \_srlv, \_srav, \_jr;

wire \_addi, \_addiu, \_andi, \_ori, \_xori, \_lw, \_sw;

wire \_beq, \_bne, \_slti, \_sltiu, \_lui, \_j, \_jal;

//CP0

wire \_break,\_syscall,\_teq,\_eret,\_mfc0,\_mtc0;

//1~17

assign \_add = (imem[31:26]==6'b000000&&imem[5:0]==6'b100000)?1'b1:1'b0;

assign \_addu = (imem[31:26]==6'b000000&&imem[5:0]==6'b100001)?1'b1:1'b0;

assign \_sub = (imem[31:26]==6'b000000&&imem[5:0]==6'b100010)?1'b1:1'b0;

assign \_subu = (imem[31:26]==6'b000000&&imem[5:0]==6'b100011)?1'b1:1'b0;

assign \_and = (imem[31:26]==6'b000000&&imem[5:0]==6'b100100)?1'b1:1'b0;

assign \_or = (imem[31:26]==6'b000000&&imem[5:0]==6'b100101)?1'b1:1'b0;

assign \_xor = (imem[31:26]==6'b000000&&imem[5:0]==6'b100110)?1'b1:1'b0;

assign \_nor = (imem[31:26]==6'b000000&&imem[5:0]==6'b100111)?1'b1:1'b0;

assign \_slt = (imem[31:26]==6'b000000&&imem[5:0]==6'b101010)?1'b1:1'b0;

assign \_sltu = (imem[31:26]==6'b000000&&imem[5:0]==6'b101011)?1'b1:1'b0;

assign \_sll = (imem[31:26]==6'b000000&&imem[5:0]==6'b000000)?1'b1:1'b0;

assign \_srl = (imem[31:26]==6'b000000&&imem[5:0]==6'b000010)?1'b1:1'b0;

assign \_sra = (imem[31:26]==6'b000000&&imem[5:0]==6'b000011)?1'b1:1'b0;

assign \_sllv = (imem[31:26]==6'b000000&&imem[5:0]==6'b000100)?1'b1:1'b0;

assign \_srlv = (imem[31:26]==6'b000000&&imem[5:0]==6'b000110)?1'b1:1'b0;

assign \_srav = (imem[31:26]==6'b000000&&imem[5:0]==6'b000111)?1'b1:1'b0;

assign \_jr = (imem[31:26]==6'b000000&&imem[5:0]==6'b001000)?1'b1:1'b0;

//18~29

assign \_addi = (imem[31:26]==6'b001000)?1'b1:1'b0;

assign \_addiu = (imem[31:26]==6'b001001)?1'b1:1'b0;

assign \_andi = (imem[31:26]==6'b001100)?1'b1:1'b0;

assign \_ori = (imem[31:26]==6'b001101)?1'b1:1'b0;

assign \_xori = (imem[31:26]==6'b001110)?1'b1:1'b0;

assign \_lw = (imem[31:26]==6'b100011)?1'b1:1'b0;

assign \_sw = (imem[31:26]==6'b101011)?1'b1:1'b0;

assign \_beq = (imem[31:26]==6'b000100)?1'b1:1'b0;

assign \_bne = (imem[31:26]==6'b000101)?1'b1:1'b0;

assign \_slti = (imem[31:26]==6'b001010)?1'b1:1'b0;

assign \_sltiu = (imem[31:26]==6'b001011)?1'b1:1'b0;

assign \_lui = (imem[31:26]==6'b001111)?1'b1:1'b0;

//30 31

assign \_j = (imem[31:26]==6'b000010)?1'b1:1'b0;

assign \_jal = (imem[31:26]==6'b000011)?1'b1:1'b0;

//32-54

wire \_clz,\_divu,\_div,\_multu,\_mul;

wire \_lb,\_lbu,\_lhu, \_lh;

wire \_sb,\_sh;

assign \_clz = (imem[31:26]==6'b011100&&imem[5:0]==6'b100000)?1'b1:1'b0;

assign \_divu = (imem[31:26]==6'b000000&&imem[5:0]==6'b011011)?1'b1:1'b0;

assign \_div = (imem[31:26]==6'b000000&&imem[5:0]==6'b011010)?1'b1:1'b0;

assign \_multu = (imem[31:26]==6'b000000&&imem[5:0]==6'b011001)?1'b1:1'b0;

assign \_mul = (imem[31:26]==6'b011100&&imem[5:0]==6'b000010)?1'b1:1'b0;

assign \_lb = (imem[31:26]==6'b100\_000)?1'b1:1'b0;

assign \_lbu = (imem[31:26]==6'b100\_100)?1'b1:1'b0;

assign \_lh = (imem[31:26]==6'b100\_001)?1'b1:1'b0;

assign \_lhu = (imem[31:26]==6'b100\_101)?1'b1:1'b0;

assign \_sb = (imem[31:26]==6'b101\_000)?1'b1:1'b0;

assign \_sh = (imem[31:26]==6'b101\_001)?1'b1:1'b0;

assign \_mfhi = (imem[31:26]==6'b000000&&imem[5:0]==6'b010000)?1'b1:1'b0;

assign \_mflo = (imem[31:26]==6'b000000&&imem[5:0]==6'b010010)?1'b1:1'b0;

assign \_mthi = (imem[31:26]==6'b000000&&imem[5:0]==6'b010001)?1'b1:1'b0;

assign \_mtlo = (imem[31:26]==6'b000000&&imem[5:0]==6'b010011)?1'b1:1'b0;

assign \_bgez = (imem[31:26]==6'b000001)?1'b1:1'b0;

assign \_jalr = (imem[31:26]==6'b000000&&imem[5:0]==6'b001001)?1'b1:1'b0;

wire [31:0] CLZ;

assign CLZ =Rs[31]==1? 32'h00000000:Rs[30]==1? 32'h00000001:Rs[29]==1? 32'h00000002:Rs[28]==1? 32'h00000003:Rs[27]==1? 32'h00000004:

Rs[26]==1? 32'h00000005:Rs[25]==1? 32'h00000006:Rs[24]==1? 32'h00000007:Rs[23]==1? 32'h00000008:Rs[22]==1? 32'h00000009:

Rs[21]==1? 32'h0000000a:Rs[20]==1? 32'h0000000b:Rs[19]==1? 32'h0000000c:Rs[18]==1? 32'h0000000d:Rs[17]==1? 32'h0000000e:

Rs[16]==1? 32'h0000000f:Rs[15]==1? 32'h00000010:Rs[14]==1? 32'h00000011:Rs[13]==1? 32'h00000012:Rs[12]==1? 32'h00000013:

Rs[11]==1? 32'h00000014:Rs[10]==1? 32'h00000015:Rs[9]==1? 32'h00000016:Rs[8]==1? 32'h00000017:Rs[7]==1? 32'h00000018:

Rs[6]==1? 32'h00000019:Rs[5]==1? 32'h0000001a:Rs[4]==1? 32'h0000001b:Rs[3]==1? 32'h0000001c:Rs[2]==1? 32'h0000001d:

Rs[1]==1? 32'h0000001e:Rs[0]==1? 32'h0000001f:32'h00000020;

//\*\*\*\*\*\*\*\*\*\*\*divu\*\*\*\*\*\*\*\*\*

wire [31:0]divu\_q;

wire [31:0] du\_q;

wire [31:0]divu\_r;

wire [31:0] du\_r;

wire divu\_busy;

DIVU divu(

.dividend(Rs),//

.divisor(Rt),//

.start(\_divu),

.clock(clk\_in),

.reset(reset),

.q(du\_q),

.r(du\_r),

.busy(divu\_busy)

);

//\*\*\*\*\*\*\*\*\*\*\*\*div\*\*\*\*\*\*\*\*\*\*\*\*\*

wire [31:0]div\_q;

wire [31:0] d\_q;

wire [31:0]div\_r;

wire [31:0] d\_r;

wire div\_busy;

DIV div(

.dividend(Rs),

.divisor(Rt),

.start(\_div),

.clock(clk\_in),

.reset(reset),

.q(d\_q),

.r(d\_r),

.busy(div\_busy)

);

assign div\_q=div\_busy?32'bz:d\_q;

assign div\_r=div\_busy?32'bz:d\_r;

//\*\*\*\*\*\*\*\*\*multu\*\*\*\*\*\*\*\*\*\*

wire [63:0]multu\_r;

wire [63:0] mu\_r;

// wire multu\_busy;

MULTU multu(

.clk(clk\_in),

.reset(reset),

.start(\_multu),

.a(Rs),

.b(Rt),

.z(mu\_r)

// .busy(multu\_busy)

);

// assign multu\_r=multu\_busy?64'bz:mu\_r;

assign multu\_r=mu\_r;

//\*\*\*\*\*\*\*\*\*mul\*\*\*\*\*\*\*\*\*\*

wire [63:0]mul\_r;

wire [63:0] m\_r;

assign mul\_r=Rs\*Rt;

assign m\_r=mul\_r[31:0];

wire busy;

// assign busy=mul\_busy||multu\_busy||div\_busy||divu\_busy;

assign busy= div\_busy||divu\_busy;

//\*\*\*\*\*\*\*\*\*\*\*\_lw\*\*\*\*\*\_lb \*\*\*\* \_lbu\*\*\*\* \_lh\*\*\*\_lhu\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*sw\*\*\*sb\*\*\*\*\*\*\*\*\*\*sh\*\*\*\*\*\*\*\*\*

wire [5:0] op = imem[31:26];

reg [31:0] dmem\_data;

always@(\*)begin

case(op)

//lb

6'b100\_000: dmem\_data = {{24{ram\_out[31]}},ram\_out[31:24]};

//lbu

6'b100\_100: dmem\_data = {24'b0,ram\_out[31:24]};

//lh

6'b100\_001: dmem\_data = {{16{ram\_out[31]}},ram\_out[31:16]};

//lhu

6'b100\_101: dmem\_data = {16'b0,ram\_out[31:16]};

//lw

6'b100011: dmem\_data = ram\_out;

default:dmem\_data = ram\_out;

endcase

end

assign select = \_sb?3'b001:(\_sh?3'b010:3'b100);

// case(op)

// //sb

// 6'b101\_000: dmem\_in = {24'b0,Rt[7:0]};

// //sh

// 6'b101\_001: dmem\_in = {16'b0,Rt[15:0]};

// //sw

//// 6'b101\_011: dmem\_in = Rt;

// default:

// endcase

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*LO\*\*\*\*\*\*HI\*\*\*\*\*\*\*

wire [31:0]lo\_out;

reg [31:0]lo\_in;

wire [31:0]hi\_out;

reg [31:0]hi\_in;

wire [11:0] op12 = {imem[31:26],imem[5:0]};

wire LO\_ctrl,HI\_ctrl;

assign L=lo\_in;

assign H=hi\_in;

assign LO\_ctrl=\_divu || \_div||\_mtlo||\_multu?1'b1:1'b0;

assign HI\_ctrl=\_divu || \_div||\_mthi||\_multu?1'b1:1'b0;

always@(\*)begin

case(op12)

//divu

12'b000000\_011011: begin

lo\_in = divu\_q;

hi\_in = divu\_r;

end

//div

12'b000000\_011010: begin

lo\_in = div\_q;

hi\_in = div\_r;

end

//multu

12'b000000\_011001:begin

lo\_in = multu\_r[31:0];

hi\_in = multu\_r[63:32];

end

//mtlo

12'b000000\_010011: lo\_in = Rs;

//mthi

12'b000000\_010001: hi\_in = Rs;

endcase

end

HLreg LO(

.clk(clk\_in),

.rst(reset),

.wena(LO\_ctrl),

.data\_in(lo\_in),

.data\_out(lo\_out)

);

HLreg HI(

.clk(clk\_in),

.rst(reset),

.wena(HI\_ctrl),

.data\_in(hi\_in),

.data\_out(hi\_out)

);

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//CP0

assign \_break = (imem[31:26]==6'b000000&&imem[5:0]==6'b001101)?1'b1:1'b0;

assign \_syscall = (imem[31:26]==6'b000000&&imem[5:0]==6'b001100)?1'b1:1'b0;

assign \_teq = (imem[31:26]==6'b000000&&imem[5:0]==6'b110100)?1'b1:1'b0;

assign \_eret = (imem[31:26]==6'b010000&&imem[5:0]==6'b011000)?1'b1:1'b0;

assign \_mfc0 = (imem[31:26]==6'b010000&&imem[25:21]==5'b00000)?1'b1:1'b0;

assign \_mtc0 = (imem[31:26]==6'b010000&&imem[25:21]==5'b00100)?1'b1:1'b0;

//CP0

wire exception;

wire [4:0] cause;

assign exception = (\_break||\_syscall||(\_teq&&zero)) ?1'b1:1'b0;

assign cause = \_break?5'b01001:(\_syscall?5'b01000:(\_teq?5'b01101:5'b00000));

wire wdata;

assign wdata = Rt;

wire [31:0] rdata;

wire [31:0] status;

wire [31:0] exc\_addr;

CP0 cp0(.clk(clk\_in),

.rst(reset),

.we(1),//new

.mfc0(\_mfc0),

.mtc0(\_mtc0),

.pc(PC),

.addr(imem[15:11]),//.Rd(imem[15:11]),

.data(wdata),//.wdata(wdata),

.exception(exception),

.eret(\_eret),

.cause(cause),

//.intr(0),

.rdata(rdata),

.status(status),

// .reg\_timer\_int(0),

.exc\_addr(exc\_addr)

);

//Control signal expression

assign M3 = \_sll || \_srl || \_sra ;

assign M3\_2 = \_jal||\_jalr ;

assign M4 = \_addi || \_addiu || \_andi || \_ori || \_xori || \_slti || \_sltiu || \_lui || \_lw||\_lb || \_lbu || \_lh ||\_lhu || \_sw ||\_sb||\_sh;

assign M4\_2 = \_jal ||\_jalr;

assign M6 =\_addi||\_addiu||\_andi||\_ori||\_xori|||\_slti||\_sltiu||\_lui||\_lw ||\_lb || \_lbu || \_lh ||\_lhu||\_mfc0;

assign M7 =\_jal;

assign ALUC[3] = \_slt || \_sltu ||\_sllv || \_srlv || \_srav || \_sll || \_srl || \_sra || \_slti || \_sltiu || \_lui ;

assign ALUC[2] = \_and || \_or ||\_xor || \_nor || \_sllv || \_srlv || \_srav || \_sll || \_srl || \_sra || \_andi || \_ori || \_xori ;

assign ALUC[1] = \_add || \_sub ||\_xor || \_nor || \_slt || \_sltu || \_sllv || \_sll || \_addi || \_xori || \_slti || \_sltiu;

assign ALUC[0] = \_sub || \_subu ||\_or || \_nor || \_slt || \_srlv || \_srl || \_ori || \_slti || \_beq || \_bne ||\_teq;//cp0 teq

assign M2 = (!\_lw) && (!\_lb) && (!\_lbu) && (!\_lh) && (!\_lhu);

assign rdc = M6?imem[20:16]:(M7?5'd31:imem[15:11]);

assign RF\_W= (!\_sw)&&(!\_beq)&&(!\_bne)&&(!\_bgez)&&(!\_j)&&(!\_jr)&&(!\_sb)&&(!\_sh)&&(!\_mtc0)&&(!\_eret)&&(!\_syscall)&&(!\_teq)&&(!\_break)&&(!\_divu)&&(!\_div)&&(!\_multu)&&(!\_mthi)&&(!\_mtlo);//&&(!\_break)&&(!\_divu)&&(!\_div)&&(!\_multu)&&(!\_mthi)&&(!\_mtlo)

assign RL\_CLK= ((!\_sw)&&(!\_beq)&&(!\_bne)&&(!\_bgez)&&(!\_j)&&(!\_jal)&&(!\_jalr)&&(!\_jr)&&(!\_sb)&&(!\_sh))&&clk\_in;

assign M5 = (\_beq&&zero) || (\_bne&&(!zero)) ||(\_bgez&&(Rs[31]==1'b0));

assign M1 = (!\_j)&&(!\_jal)&&(!\_jalr) ;

assign M1\_2 = \_jr||\_jalr ;

assign cs = \_lw||\_lb || \_lbu || \_lh ||\_lhu || \_sw||\_sb||\_sh;

assign dm\_r = \_lw||\_lb || \_lbu || \_lh ||\_lhu;

assign dm\_w = \_sw||\_sb||\_sh;

assign ext16\_sin\_judge = \_addi || \_addiu || \_slti||\_sltiu || \_lw||\_lb || \_lbu || \_lh ||\_lhu; //sign\_ext

assign ext18\_sign = {{14{imem[15]}},{imem[15:0],2'h0}};

assign npc = PC + 4;

assign ext16\_sign = {{16{imem[15]}},imem[15:0]};

assign ext16 = {16'h0,imem[15:0]};

assign mux\_in\_4 = ext16\_sin\_judge?ext16\_sign:ext16;

assign rsc = imem[25:21];

assign rtc = imem[20:16];

assign MUX1=M1?MUX5:{PC[31:28],imem[25:0],2'b00}; //mux1

//CP0 eret

assign MUX1\_2=M1\_2?Rs:((\_eret||exception)?exc\_addr:MUX1); //mux1\_2

//CP0 mfc0

//assign MUX2=M2?alu\_r:(\_mfc0?rdata:ram\_out); //mux2 //cp0

wire [6:0] op3 = {M2,\_mfc0,\_clz,\_mfhi,\_mflo,\_multu,\_mul};

always@(\*)begin

case(op3)

//M2=0

7'b0000000: Rd = dmem\_data;

//M2=1

7'b1000000: Rd = alu\_r;

//mfc0

7'b1100000: Rd = rdata;

//clz

7'b1010000: Rd = CLZ;

//mfhi

7'b1001000: Rd = hi\_out;

//mflo

7'b1000100: Rd = lo\_out;

//multu

7'b1000010: Rd = mu\_r[31:0];

//mul

7'b1000001: Rd = m\_r[31:0];

default:Rd = alu\_r;

endcase

end

//assign MUX1\_2=M1\_2?Rs:MUX1; //mux1\_2

//assign MUX2=M2?alu\_r:ram\_out; //mux2

assign MUX3=M3?{27'b0,imem[10:6]}:Rs;

assign MUX3\_2=M3\_2?PC:MUX3;

assign MUX4=M4?mux\_in\_4:Rt;

assign MUX4\_2=M4\_2?32'd4:MUX4;

assign MUX5=M5?ext18\_sign+npc:npc;

pcreg PCreg(

.clk(clk\_in),

.rst(reset),

.ena(1),

.wena(!busy),

.data\_in(MUX1\_2),

.data\_out(PC)

);

regfile cpu\_ref(

.clk(clk\_in),

.rst(reset),

.ena(1),

.we(RF\_W),

.raddr1(rsc), //rsc

.raddr2(rtc),//rtc

.waddr(rdc), //rdc 5bits

.wdata(Rd), //rd

.rdata1(Rs), //rs

.rdata2(Rt) //rt

);

ALU alu(

.a(MUX3\_2),

.b(MUX4\_2),

.aluc(ALUC),

.r(alu\_r),

.zero(zero),

.carry(carry),

.negative(negative),

.overflow(overflow)

);

endmodule

3 dmem模块

module DMEM(

input clk,

input CS,

input DM\_W,

input DM\_R,

input [2:0]select, //100 32bit 010:16bit 001:8

input [31:0] addr,

input [31:0] data\_in,

output [31:0] data\_out

);

wire [7:0] Addr;

assign Addr=addr[7:0];

reg [7:0] num [0:255];

assign data\_out=CS? (DM\_R? {num[Addr],num[Addr+1],num[Addr+2],num[Addr+3]}: 32'h00000000):32'hzzzz\_zzzz;

always@(negedge clk or negedge CS)

begin

if(CS&&DM\_W)

begin

case (select)

3'b001:num[Addr]<=data\_in[7:0];

3'b010:

begin

num[Addr]<=data\_in[15:8];

num[Addr+1]<=data\_in[7:0];

end

3'b100:

begin

num[Addr]<=data\_in[31:24];

num[Addr+1]<=data\_in[23:16];

num[Addr+2]<=data\_in[15:8];

num[Addr+3]<=data\_in[7:0];

end

endcase

end

end

endmodule

4 ALU负责完成各类计算

module ALU(

input [31:0] a, //OP1

input [31:0] b, //OP2

input [3:0] aluc, //controller

output [31:0] r, //result

output zero,

output carry,

output negative,

output overflow);

parameter Addu = 4'b0000; //r=a+b unsigned

parameter Add = 4'b0010; //r=a+b signed

parameter Subu = 4'b0001; //r=a-b unsigned

parameter Sub = 4'b0011; //r=a-b signed

parameter And = 4'b0100; //r=a&b

parameter Or = 4'b0101; //r=a|b

parameter Xor = 4'b0110; //r=a^b

parameter Nor = 4'b0111; //r=~(a|b)

parameter Lui1 = 4'b1000; //r={b[15:0],16'b0}

parameter Lui2 = 4'b1001; //r={b[15:0],16'b0}

parameter Slt = 4'b1011; //r=(a-b<0)?1:0 signed

parameter Sltu = 4'b1010; //r=(a-b<0)?1:0 unsigned

parameter Sra = 4'b1100; //r=b>>>a

parameter Sll = 4'b1110; //r=b<<a

parameter Srl = 4'b1101; //r=b>>a

parameter bits=31;

parameter ENABLE=1,DISABLE=0;

reg [32:0] result;

wire signed [31:0] sa=a,sb=b;

always@(\*)begin

case(aluc)

Addu: begin

result=a+b;

end

Subu: begin

result=a-b;

end

Add: begin

result=sa+sb;

end

Sub: begin

result=sa-sb;

end

Sra: begin

if(a==0) {result[31:0],result[32]}={b,1'b0};

else {result[31:0],result[32]}=sb>>>(a-1);

end

Srl: begin

if(a==0) {result[31:0],result[32]}={b,1'b0};

else {result[31:0],result[32]}=b>>(a-1);

end

Sll: begin

result=b<<a;

end

And: begin

result=a&b;

end

Or: begin

result=a|b;

end

Xor: begin

result=a^b;

end

Nor: begin

result=~(a|b);

end

Sltu: begin

result=a<b?1:0;

end

Slt: begin

result=sa<sb?1:0;

end

Lui1,Lui2: result = {b[15:0], 16'b0};

default:

result=a+b;

endcase

end

assign r=result[31:0];

assign carry = result[32];

assign zero=(r==32'b0)?1:0;

assign negative=result[31];

assign overflow=result[32];

endmodule

5 regfiles模块

module regfile(

input clk,

input rst,

input ena,

input we,

input [4:0] raddr1,

input [4:0] raddr2,

input [4:0] waddr,

input [31:0] wdata,

output wire [31:0] rdata1,

output wire [31:0] rdata2

);

reg [31:0]array\_reg[31:0];

assign rdata1 = ena?array\_reg[raddr1]:32'bz;

assign rdata2 = ena?array\_reg[raddr2]:32'bz;

always @(negedge clk or posedge rst)

//always @(posedge clk or posedge rst)

begin

if (rst) begin

array\_reg[0] <= 32'h0;

array\_reg[1] <= 32'h0;

array\_reg[2] <= 32'h0;

array\_reg[3] <= 32'h0;

array\_reg[4] <= 32'h0;

array\_reg[5] <= 32'h0;

array\_reg[6] <= 32'h0;

array\_reg[7] <= 32'h0;

array\_reg[8] <= 32'h0;

array\_reg[9] <= 32'h0;

array\_reg[10] <= 32'h0;

array\_reg[11] <= 32'h0;

array\_reg[12] <= 32'h0;

array\_reg[13] <= 32'h0;

array\_reg[14] <= 32'h0;

array\_reg[15] <= 32'h0;

array\_reg[16] <= 32'h0;

array\_reg[17] <= 32'h0;

array\_reg[18] <= 32'h0;

array\_reg[19] <= 32'h0;

array\_reg[20] <= 32'h0;

array\_reg[21] <= 32'h0;

array\_reg[22] <= 32'h0;

array\_reg[23] <= 32'h0;

array\_reg[24] <= 32'h0;

array\_reg[25] <= 32'h0;

array\_reg[26] <= 32'h0;

array\_reg[27] <= 32'h0;

array\_reg[28] <= 32'h0;

array\_reg[29] <= 32'h0;

array\_reg[30] <= 32'h0;

array\_reg[31] <= 32'h0;

end

else if ((ena&we)&& (waddr != 0))

array\_reg[waddr] <= wdata;

end

endmodule

6 pc寄存器

module pcreg(clk,rst,ena,data\_in,data\_out);

input clk;

input rst;

input ena;

input [31:0]data\_in;

output reg[31:0]data\_out;

wire [31:0]data;

always@(\*)

begin

if(ena==1)

data\_out=data;

else

data\_out=data\_out;

end

Asynchronous\_D\_FF DFF0(.CLK(clk),.D(data\_in[0]),.RST\_n(rst),.Q1(data[0]));

Asynchronous\_D\_FF DFF1(.CLK(clk),.D(data\_in[1]),.RST\_n(rst),.Q1(data[1]));

Asynchronous\_D\_FF DFF2(.CLK(clk),.D(data\_in[2]),.RST\_n(rst),.Q1(data[2]));

Asynchronous\_D\_FF DFF3(.CLK(clk),.D(data\_in[3]),.RST\_n(rst),.Q1(data[3]));

Asynchronous\_D\_FF DFF4(.CLK(clk),.D(data\_in[4]),.RST\_n(rst),.Q1(data[4]));

Asynchronous\_D\_FF DFF5(.CLK(clk),.D(data\_in[5]),.RST\_n(rst),.Q1(data[5]));

Asynchronous\_D\_FF DFF6(.CLK(clk),.D(data\_in[6]),.RST\_n(rst),.Q1(data[6]));

Asynchronous\_D\_FF DFF7(.CLK(clk),.D(data\_in[7]),.RST\_n(rst),.Q1(data[7]));

Asynchronous\_D\_FF DFF8(.CLK(clk),.D(data\_in[8]),.RST\_n(rst),.Q1(data[8]));

Asynchronous\_D\_FF DFF9(.CLK(clk),.D(data\_in[9]),.RST\_n(rst),.Q1(data[9]));

Asynchronous\_D\_FF DFF10(.CLK(clk),.D(data\_in[10]),.RST\_n(rst),.Q1(data[10]));

Asynchronous\_D\_FF DFF11(.CLK(clk),.D(data\_in[11]),.RST\_n(rst),.Q1(data[11]));

Asynchronous\_D\_FF DFF12(.CLK(clk),.D(data\_in[12]),.RST\_n(rst),.Q1(data[12]));

Asynchronous\_D\_FF DFF13(.CLK(clk),.D(data\_in[13]),.RST\_n(rst),.Q1(data[13]));

Asynchronous\_D\_FF DFF14(.CLK(clk),.D(data\_in[14]),.RST\_n(rst),.Q1(data[14]));

Asynchronous\_D\_FF DFF15(.CLK(clk),.D(data\_in[15]),.RST\_n(rst),.Q1(data[15]));

Asynchronous\_D\_FF DFF16(.CLK(clk),.D(data\_in[16]),.RST\_n(rst),.Q1(data[16]));

Asynchronous\_D\_FF DFF17(.CLK(clk),.D(data\_in[17]),.RST\_n(rst),.Q1(data[17]));

Asynchronous\_D\_FF DFF18(.CLK(clk),.D(data\_in[18]),.RST\_n(rst),.Q1(data[18]));

Asynchronous\_D\_FF DFF19(.CLK(clk),.D(data\_in[19]),.RST\_n(rst),.Q1(data[19]));

Asynchronous\_D\_FF DFF20(.CLK(clk),.D(data\_in[20]),.RST\_n(rst),.Q1(data[20]));

Asynchronous\_D\_FF DFF21(.CLK(clk),.D(data\_in[21]),.RST\_n(rst),.Q1(data[21]));

Asynchronous\_D\_FF1 DFF22(.CLK(clk),.D(data\_in[22]),.RST\_n(rst),.Q1(data[22]));//reset==1

Asynchronous\_D\_FF DFF23(.CLK(clk),.D(data\_in[23]),.RST\_n(rst),.Q1(data[23]));

Asynchronous\_D\_FF DFF24(.CLK(clk),.D(data\_in[24]),.RST\_n(rst),.Q1(data[24]));

Asynchronous\_D\_FF DFF25(.CLK(clk),.D(data\_in[25]),.RST\_n(rst),.Q1(data[25]));

Asynchronous\_D\_FF DFF26(.CLK(clk),.D(data\_in[26]),.RST\_n(rst),.Q1(data[26]));

Asynchronous\_D\_FF DFF27(.CLK(clk),.D(data\_in[27]),.RST\_n(rst),.Q1(data[27]));

Asynchronous\_D\_FF DFF28(.CLK(clk),.D(data\_in[28]),.RST\_n(rst),.Q1(data[28]));

Asynchronous\_D\_FF DFF29(.CLK(clk),.D(data\_in[29]),.RST\_n(rst),.Q1(data[29]));

Asynchronous\_D\_FF DFF30(.CLK(clk),.D(data\_in[30]),.RST\_n(rst),.Q1(data[30]));

Asynchronous\_D\_FF DFF31(.CLK(clk),.D(data\_in[31]),.RST\_n(rst),.Q1(data[31]));

Endmodule

7 CP0

module CP0(

input clk,

input rst,

input we,

input mfc0, // CPU instruction is Mfc0

input mtc0, // CPU instruction is Mtc0

input [31:0]pc,

input [4:0] addr, // Specifies Cp0 register

input [31:0] data, // Data from GP register to replace CP0 register

input exception, //1的话就是异常

input eret,

input [4:0] cause,

output [31:0] rdata, // Data from CP0 register for GP register

output [31:0] status,

output [31:0]exc\_addr // Address for PC at the beginning of an exception

);

reg [31:0] CP0 [31:0];

reg [31:0] status\_tmp;

integer i;

assign rdata=(mfc0==1)?CP0[addr]:32'bz;

assign exc\_addr=(exception && (eret==1))?CP0[14]:((exception && (eret==0) )?1:32'bz);

assign status=CP0[12];

always @(negedge clk or posedge rst)

begin

if(rst==1)

begin

for(i=0;i<=11;i=i+1)

CP0[i]=32'b0;

CP0[12]={28'b0,4'b1};

for(i=13;i<=31;i=i+1)

CP0[i]=32'b0;

end

else begin

if(we) begin

if(!exception)

begin

// if(mfc0)//读

// rdata=CP0[addr];

if(mtc0)//写

begin

CP0[addr]=data;

end

end

if(exception)

begin

status\_tmp=CP0[12];

if(!eret)

begin

CP0[14]=pc;

CP0[12]={CP0[12],5'b0};

CP0[13][6:2]=cause;

end

else begin

CP0[12]=status\_tmp;

end

end

end

end

end

endmodule

8HILO

module HLreg(

input clk,

input rst,

input wena,

input [31:0]data\_in,

output[31:0]data\_out

);

reg [31:0]data;

always @(posedge clk or posedge rst)

if (rst)

data <= 32'h00000000;

else if ( wena)

data <= data\_in;

assign data\_out = data;

endmodule

9Div

module DIV(

input [31:0]dividend,

input [31:0]divisor,

input start,

input clock,

input reset,

output [31:0]q,

output [31:0]r,

output reg busy

);

reg [4:0]count;

reg [31:0]reg\_q;

reg [31:0]reg\_r;

reg [31:0]reg\_b;

reg q\_sign;

reg r\_sign;

reg a\_sign;

wire [32:0]sub\_add=r\_sign?{reg\_r,reg\_q[31]}+{1'b0,reg\_b}:{reg\_r,reg\_q[31]}-{1'b0,reg\_b};

assign r=a\_sign?(-(r\_sign?reg\_r+reg\_b:reg\_r)):(r\_sign?reg\_r+reg\_b:reg\_r);

assign q=q\_sign?-reg\_q:reg\_q;

always@(negedge clock or posedge reset)

begin

if(reset)

begin count<=0;busy<=0;end

else if(start&&(!busy))begin

count<=0;reg\_q<=dividend[31]?-dividend:dividend;

reg\_r<=0;

reg\_b<=divisor[31]?-divisor:divisor;

r\_sign<=0;busy<=1'b1;

q\_sign<=dividend[31]^divisor[31];

a\_sign<=dividend[31];

end

else if(busy)begin

reg\_r<=sub\_add[31:0];

r\_sign<=sub\_add[32];

reg\_q<={reg\_q[30:0],~sub\_add[32]};

count<=count+1;

if(count==31)busy<=0;

end

end

endmodule

10 Divu

module DIVU(

input [31:0] dividend,

input [31:0] divisor,

input start,

input clock,

input reset,

output [31:0] q,

output [31:0] r,

output reg busy

);

reg [4:0]count;

reg [31:0] reg\_q;

reg [31:0] reg\_r;

reg [31:0] reg\_b;

reg r\_sign;

wire [32:0] sub\_add = r\_sign?({reg\_r,q[31]} + {1'b0,reg\_b}):({reg\_r,q[31]} - {1'b0,reg\_b}); //加、减法器

assign r = r\_sign? reg\_r + reg\_b : reg\_r;

assign q = reg\_q;

always @ (negedge clock or posedge reset)begin

if (reset == 1) begin //重置

count <=5'b0;

busy <= 0;

end

else begin

if (start&&(!busy)) begin //开始除法运算，初始化

reg\_r <= 32'b0;

r\_sign <= 0;

reg\_q <= dividend;

reg\_b <= divisor;

count <= 5'b0;

busy <= 1'b1;

end else if (busy)

begin //循环操作

reg\_r <= sub\_add[31:0]; //部分余数

r\_sign <= sub\_add[32]; //如果为负，下次相加

reg\_q <= {reg\_q[30:0],~sub\_add[32]};

count <= count +5'b1; //计数器加一

if (count == 5'b11111) busy <= 0; //结束除法运算

end

end

end

endmodule

11Multu

module MULTU(

input clk,

input reset,

input start,

input [31:0] a,

input [31:0] b,

output [63:0] z

// output reg busy

);

wire [65:0] temp;

assign temp={1'b0,a}\*{1'b0,b};

assign z=temp[63:0];

//// 申请寄存器

// reg [63:0] temp;

////32位

// reg [63:0] stored0;

// reg [63:0] stored1;

// reg [63:0] stored2;

// reg [63:0] stored3;

// reg [63:0] stored4;

// reg [63:0] stored5;

// reg [63:0] stored6;

// reg [63:0] stored7;

// reg [63:0] stored8;

// reg [63:0] stored9;

// reg [63:0] stored10;

// reg [63:0] stored11;

// reg [63:0] stored12;

// reg [63:0] stored13;

// reg [63:0] stored14;

// reg [63:0] stored15;

// reg [63:0] stored16;

// reg [63:0] stored17;

// reg [63:0] stored18;

// reg [63:0] stored19;

// reg [63:0] stored20;

// reg [63:0] stored21;

// reg [63:0] stored22;

// reg [63:0] stored23;

// reg [63:0] stored24;

// reg [63:0] stored25;

// reg [63:0] stored26;

// reg [63:0] stored27;

// reg [63:0] stored28;

// reg [63:0] stored29;

// reg [63:0] stored30;

// reg [63:0] stored31;

////第一级相加

// reg [63:0] add0\_1;

// reg [63:0] add2\_3;

// reg [63:0] add4\_5;

// reg [63:0] add6\_7;

// reg [63:0] add8\_9;

// reg [63:0] add10\_11;

// reg [63:0] add12\_13;

// reg [63:0] add14\_15;

// reg [63:0] add16\_17;

// reg [63:0] add18\_19;

// reg [63:0] add20\_21;

// reg [63:0] add22\_23;

// reg [63:0] add24\_25;

// reg [63:0] add26\_27;

// reg [63:0] add28\_29;

// reg [63:0] add30\_31;

////第二级相加

// reg [63:0] add0t1\_2t3;

// reg [63:0] add4t5\_6t7;

// reg [63:0] add8t9\_10t11;

// reg [63:0] add12t13\_14t15;

// reg [63:0] add16t17\_18t19;

// reg [63:0] add20t21\_22t23;

// reg [63:0] add24t25\_26t27;

// reg [63:0] add28t29\_30t31;

////第三级相加

// reg [63:0] add0123\_4567;

// reg [63:0] add891011\_12131415;

// reg [63:0] add16171819\_20212223;

// reg [63:0] add24252627\_28293031;

////第四级相加

// reg [63:0] addP1;

// reg [63:0] addP2;

// reg [2:0] count;

//always @(negedge clk or negedge reset)

//begin

//// reset 置零

//if(reset) begin

//busy<=0;

//count<=0;

//temp <= 0;

//stored0 <= 0;

//stored1 <= 0;

//stored2 <= 0;

//stored3 <= 0;

//stored4 <= 0;

//stored5 <= 0;

//stored6 <= 0;

//stored7 <= 0;

//stored8 <= 0;

//stored9 <= 0;

//stored10 <= 0;

//stored11 <= 0;

//stored12 <= 0;

//stored13 <= 0;

//stored14 <= 0;

//stored15 <= 0;

//stored16 <= 0;

//stored17 <= 0;

//stored18 <= 0;

//stored19 <= 0;

//stored20 <= 0;

//stored21 <= 0;

//stored22 <= 0;

//stored23 <= 0;

//stored24 <= 0;

//stored25 <= 0;

//stored26 <= 0;

//stored27 <= 0;

//stored28 <= 0;

//stored29 <= 0;

//stored30 <= 0;

//stored31 <= 0;

////第一级相加

//add0\_1 <= 0;

//add2\_3 <= 0;

//add4\_5 <= 0;

//add6\_7 <= 0;

//add8\_9 <= 0;

//add10\_11 <= 0;

//add12\_13 <= 0;

//add14\_15 <= 0;

//add16\_17 <= 0;

//add18\_19 <= 0;

//add20\_21 <= 0;

//add22\_23 <= 0;

//add24\_25 <= 0;

//add26\_27 <= 0;

//add28\_29 <= 0;

//add30\_31 <= 0;

//add0t1\_2t3 <= 0;

//add4t5\_6t7 <= 0;

//add8t9\_10t11 <= 0;

//add12t13\_14t15 <= 0;

//add16t17\_18t19 <= 0;

//add20t21\_22t23 <= 0;

//add24t25\_26t27 <= 0;

//add28t29\_30t31 <= 0;

//add0123\_4567 <= 0;

//add891011\_12131415 <= 0;

//add16171819\_20212223 <= 0;

//add24252627\_28293031 <= 0;

//addP1 <= 0;

//addP2 <= 0;

//end

//else if (start&&(!busy))begin

//busy<=1;

//end

//else if(busy)begin

//count<= count + 1'b1;

//if(count==6)busy<=0;

//stored0 <= b[0]? {32'b0, a} : 64'b0;

//stored1 <= b[1]? {31'b0, a, 1'b0} :64'b0;

//stored2 <= b[2]? {30'b0, a, 2'b0} :64'b0;

//stored3 <= b[3]? {29'b0, a, 3'b0} :64'b0;

//stored4 <= b[4]? {28'b0, a, 4'b0} :64'b0;

//stored5 <= b[5]? {27'b0, a, 5'b0} :64'b0;

//stored6 <= b[6]? {26'b0, a, 6'b0} :64'b0;

//stored7 <= b[7]? {25'b0, a, 7'b0} :64'b0;

//stored8 <= b[8]? {24'b0, a, 8'b0} :64'b0;

//stored9 <= b[9]? {23'b0, a, 9'b0} :64'b0;

//stored10 <= b[10]? {22'b0, a, 10'b0} :64'b0;

//stored11 <= b[11]? {21'b0, a, 11'b0} :64'b0;

//stored12 <= b[12]? {20'b0, a, 12'b0} :64'b0;

//stored13 <= b[13]? {19'b0, a, 13'b0} :64'b0;

//stored14 <= b[14]? {18'b0, a, 14'b0} :64'b0;

//stored15 <= b[15]? {17'b0, a, 15'b0} :64'b0;

//stored16 <= b[16]? {16'b0, a, 16'b0} :64'b0;

//stored17 <= b[17]? {15'b0, a, 17'b0} :64'b0;

//stored18 <= b[18]? {14'b0, a, 18'b0} :64'b0;

//stored19 <= b[19]? {13'b0, a, 19'b0} :64'b0;

//stored20 <= b[20]? {12'b0, a, 20'b0} :64'b0;

//stored21 <= b[21]? {11'b0, a, 21'b0} :64'b0;

//stored22 <= b[22]? {10'b0, a, 22'b0} :64'b0;

//stored23 <= b[23]? {9'b0, a, 23'b0} :64'b0;

//stored24 <= b[24]? {8'b0, a, 24'b0} :64'b0;

//stored25 <= b[25]? {7'b0, a, 25'b0} :64'b0;

//stored26 <= b[26]? {6'b0, a, 26'b0} :64'b0;

//stored27 <= b[27]? {5'b0, a, 27'b0} :64'b0;

//stored28 <= b[28]? {4'b0, a, 28'b0} :64'b0;

//stored29 <= b[29]? {3'b0, a, 29'b0} :64'b0;

//stored30 <= b[30]? {2'b0, a, 30'b0} :64'b0;

//stored31 <= b[31]? {1'b0, a, 31'b0} :64'b0;

////第一级相加

//add0\_1 <= stored0 + stored1;

//add2\_3 <= stored2 + stored3;

//add4\_5 <= stored4 + stored5;

//add6\_7 <= stored6 + stored7;

//add8\_9 <= stored8 + stored9;

//add10\_11 <= stored10 + stored11;

//add12\_13 <= stored12 + stored13;

//add14\_15 <= stored14 + stored15;

//add16\_17 <= stored16 + stored17;

//add18\_19 <= stored18 + stored19;

//add20\_21 <= stored20 + stored21;

//add22\_23 <= stored22 + stored23;

//add24\_25 <= stored24 + stored25;

//add26\_27 <= stored26 + stored27;

//add28\_29 <= stored28 + stored29;

//add30\_31 <= stored30 + stored31;

////第二级相加

//add0t1\_2t3 <= add0\_1 + add2\_3;

//add4t5\_6t7 <= add4\_5 + add6\_7;

//add8t9\_10t11 <= add8\_9 + add10\_11;

//add12t13\_14t15 <= add12\_13 + add14\_15;

//add16t17\_18t19 <= add16\_17 + add18\_19;

//add20t21\_22t23 <= add20\_21 + add22\_23;

//add24t25\_26t27 <= add24\_25 + add26\_27;

//add28t29\_30t31 <= add28\_29 + add30\_31;

////第三级相加

//add0123\_4567 <= add0t1\_2t3 + add4t5\_6t7;

//add891011\_12131415 <= add8t9\_10t11 + add12t13\_14t15;

//add16171819\_20212223 <= add16t17\_18t19 + add20t21\_22t23;

//add24252627\_28293031 <= add24t25\_26t27 + add28t29\_30t31;

////第四级相加

//addP1 <= add0123\_4567 + add891011\_12131415;

//addP2 <= add16171819\_20212223 + add24252627\_28293031;

//temp <= addP1 + addP2;

//end

//end

//assign z = temp;

endmodule

1. 测试模块建模

1 cpu\_tb

module CPU\_tb();

reg clk;

reg rst;

wire [31:0] inst;

wire [31:0] pc;

//integer file\_output;

//integer counter = 0;

sccomp\_dataflow uut(.clk\_in(clk),.reset(rst),.inst(inst),.pc(pc)//,.addr(addr)

);

initial

begin

// file\_output = $fopen("E:/result.txt");

clk = 0;

rst = 1;

#5;

rst = 0;

end

always

begin

#10;

clk = ~clk;

//if (clk == 1'b0)

//begin

//if (counter == 1500)

//begin

//$fclose(file\_output);

//end

//#4

// counter = counter + 1;

// if (counter > 600)

// begin

//$fdisplay(file\_output,"pc:%h",pc-32'h00400000);

// $fdisplay(file\_output,"pc:%h",pc);

// $fdisplay(file\_output,"instr:%h",uut.inst);

//$fdisplay(file\_output,"regfile0: %h",uut.sccpu.cpu\_ref.array\_reg[0]);

//$fdisplay(file\_output,"regfile1: %h",uut.sccpu.cpu\_ref.array\_reg[1]);

//$fdisplay(file\_output,"regfile2: %h",uut.sccpu.cpu\_ref.array\_reg[2]);

//$fdisplay(file\_output,"regfile3: %h",uut.sccpu.cpu\_ref.array\_reg[3]);

//$fdisplay(file\_output,"regfile4: %h",uut.sccpu.cpu\_ref.array\_reg[4]);

//$fdisplay(file\_output,"regfile5: %h",uut.sccpu.cpu\_ref.array\_reg[5]);

//$fdisplay(file\_output,"regfile6: %h",uut.sccpu.cpu\_ref.array\_reg[6]);

//$fdisplay(file\_output,"regfile7: %h",uut.sccpu.cpu\_ref.array\_reg[7]);

//$fdisplay(file\_output,"regfile8: %h",uut.sccpu.cpu\_ref.array\_reg[8]);

//$fdisplay(file\_output,"regfile9: %h",uut.sccpu.cpu\_ref.array\_reg[9]);

//$fdisplay(file\_output,"regfile10: %h",uut.sccpu.cpu\_ref.array\_reg[10]);

//$fdisplay(file\_output,"regfile11: %h",uut.sccpu.cpu\_ref.array\_reg[11]);

//$fdisplay(file\_output,"regfile12: %h",uut.sccpu.cpu\_ref.array\_reg[12]);

//$fdisplay(file\_output,"regfile13: %h",uut.sccpu.cpu\_ref.array\_reg[13]);

//$fdisplay(file\_output,"regfile14: %h",uut.sccpu.cpu\_ref.array\_reg[14]);

//$fdisplay(file\_output,"regfile15: %h",uut.sccpu.cpu\_ref.array\_reg[15]);

//$fdisplay(file\_output,"regfile16: %h",uut.sccpu.cpu\_ref.array\_reg[16]);

//$fdisplay(file\_output,"regfile17: %h",uut.sccpu.cpu\_ref.array\_reg[17]);

//$fdisplay(file\_output,"regfile18: %h",uut.sccpu.cpu\_ref.array\_reg[18]);

//$fdisplay(file\_output,"regfile19: %h",uut.sccpu.cpu\_ref.array\_reg[19]);

//$fdisplay(file\_output,"regfile20: %h",uut.sccpu.cpu\_ref.array\_reg[20]);

//$fdisplay(file\_output,"regfile21: %h",uut.sccpu.cpu\_ref.array\_reg[21]);

//$fdisplay(file\_output,"regfile22: %h",uut.sccpu.cpu\_ref.array\_reg[22]);

//$fdisplay(file\_output,"regfile23: %h",uut.sccpu.cpu\_ref.array\_reg[23]);

//$fdisplay(file\_output,"regfile24: %h",uut.sccpu.cpu\_ref.array\_reg[24]);

//$fdisplay(file\_output,"regfile25: %h",uut.sccpu.cpu\_ref.array\_reg[25]);

//$fdisplay(file\_output,"regfile26: %h",uut.sccpu.cpu\_ref.array\_reg[26]);

//$fdisplay(file\_output,"regfile27: %h",uut.sccpu.cpu\_ref.array\_reg[27]);

//$fdisplay(file\_output,"regfile28: %h",uut.sccpu.cpu\_ref.array\_reg[28]);

//$fdisplay(file\_output,"regfile29: %h",uut.sccpu.cpu\_ref.array\_reg[29]);

//$fdisplay(file\_output,"regfile30: %h",uut.sccpu.cpu\_ref.array\_reg[30]);

//$fdisplay(file\_output,"regfile31: %h",uut.sccpu.cpu\_ref.array\_reg[31]);

// end

//end

end

endmodule

在测试时，只需要修改cpu\_tb以及IMEM的相应文件路径即可

module IMEM(

input IM\_R, //read

input [9:0] Addr,

output reg [31:0] data\_out

);

reg [31:0] mem[1023:0];

initial begin

$readmemh("E:/31test/intrs/\_4\_jr.txt",mem);

end

always @ (\*)

begin

if(!IM\_R)

begin

data\_out <= 32'hzzzz\_zzzz;

end

else // read data

data\_out <= mem[Addr];

end

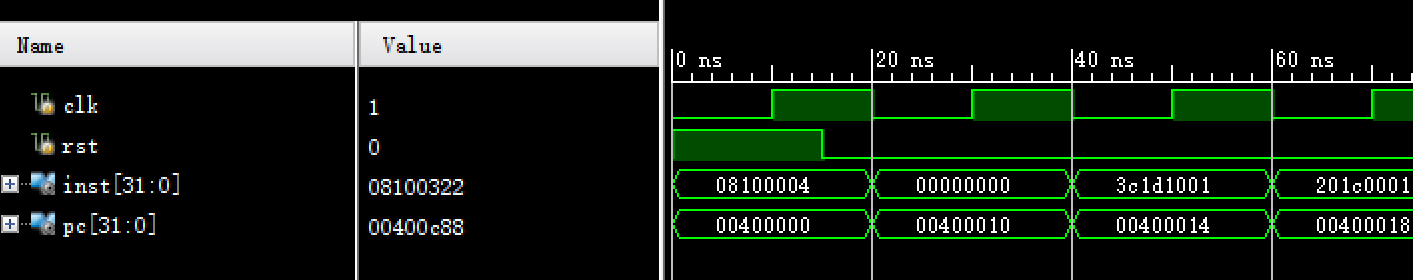
endmodule

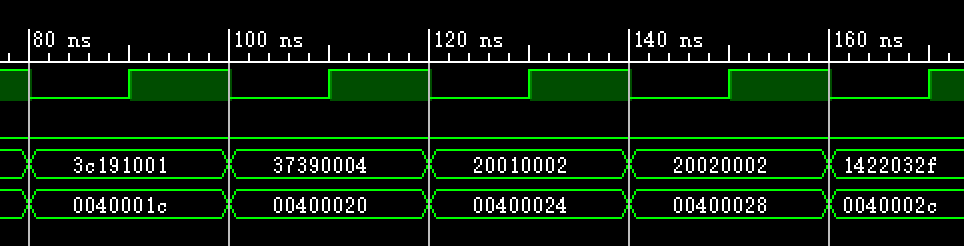
使用$readmemh初始化IMEM，读取相应的指令，进行测试

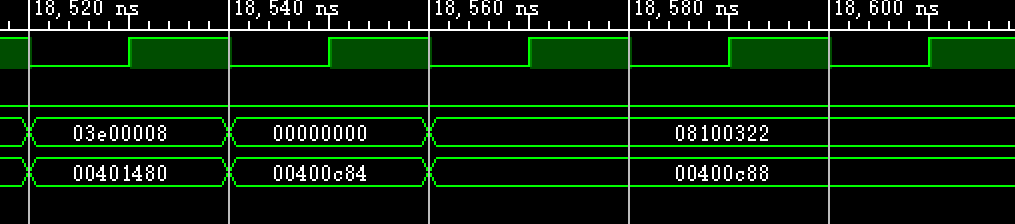
1. 实验结果

1、前、后仿真测试

通过改变imem的初始化文件，并最后添加IP核后，前仿真顺利通过，后仿真观察波形，与前仿真相同







2、下板测试

观察下板结果pc的值，下板通过

