**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31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

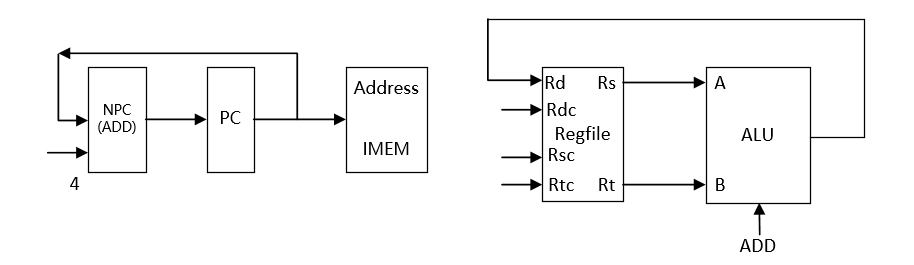
格式：ADD rd, rs, rt

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

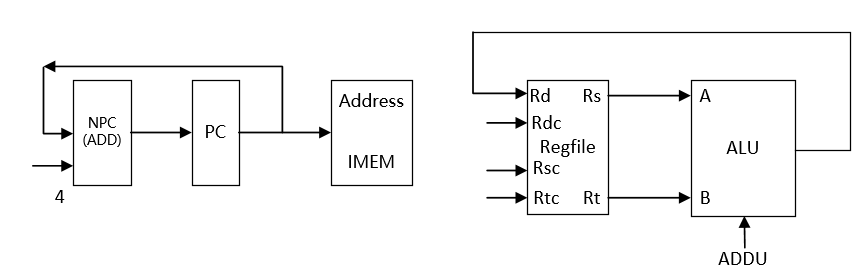
格式：ADDU rd, rs, rt

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

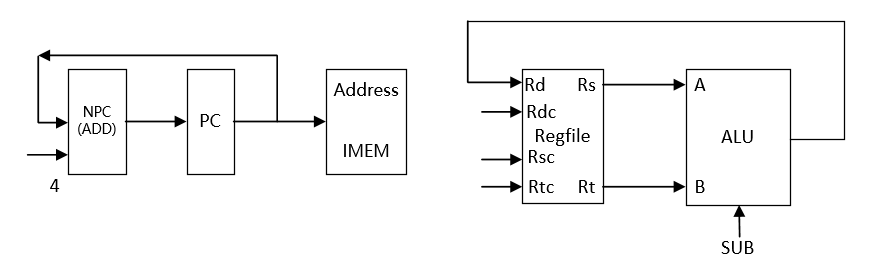
格式：SUB rd, rs, rt

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

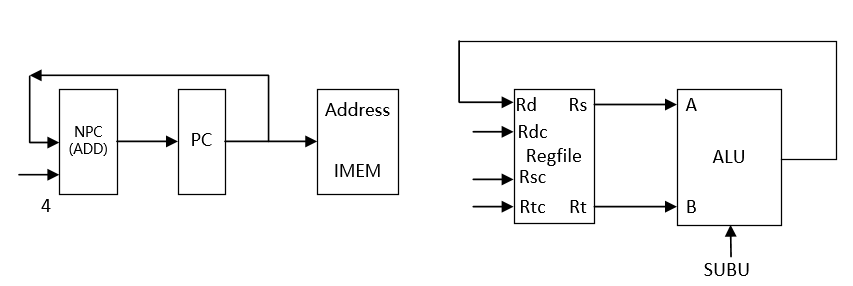
格式：SUBU rd, rs, rt

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

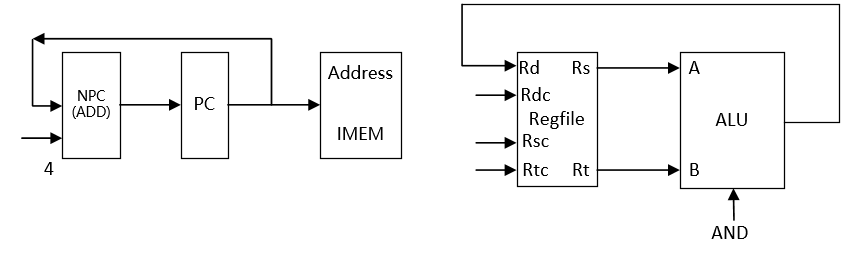
格式：AND rd, rs, rt

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

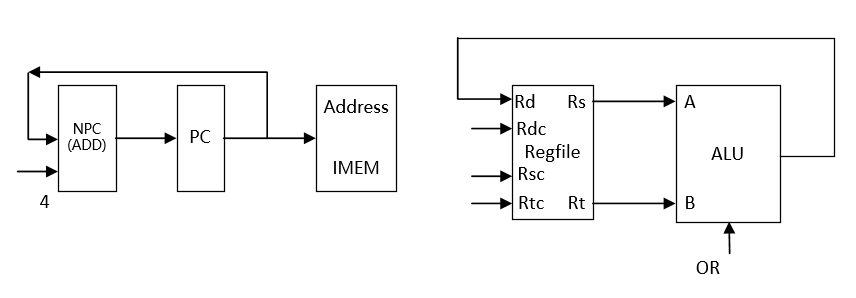
格式：OR rd, rs, rt

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

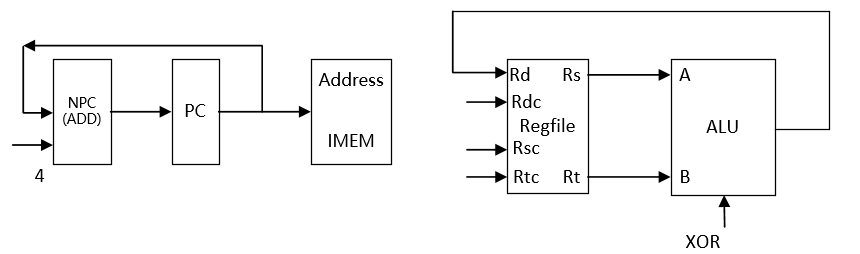
格式：XOR rd, rs, rt

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

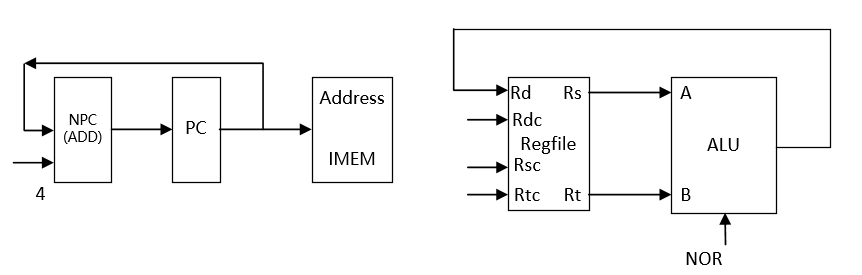
格式：NOR rd, rs, rt

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

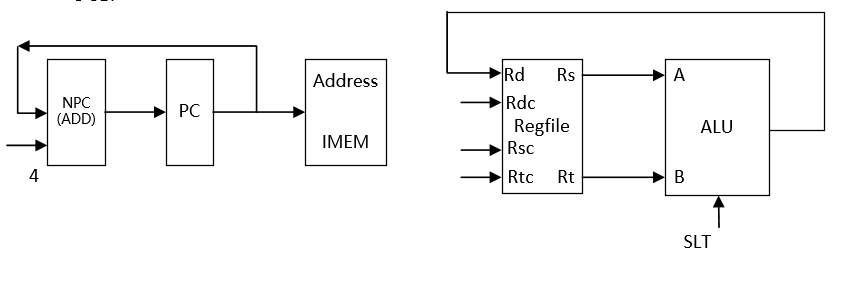
格式：SLT rd, rs, rt

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

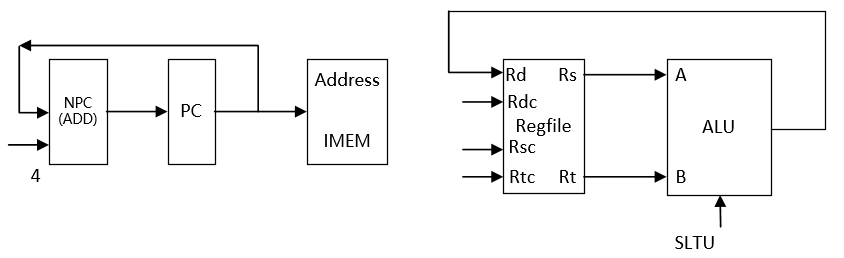
格式：SLTU rd, rs, rt

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

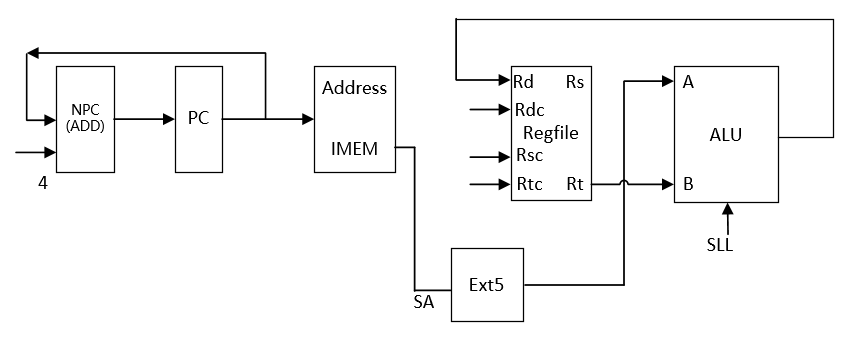
格式：SLL rd, rt, sa

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

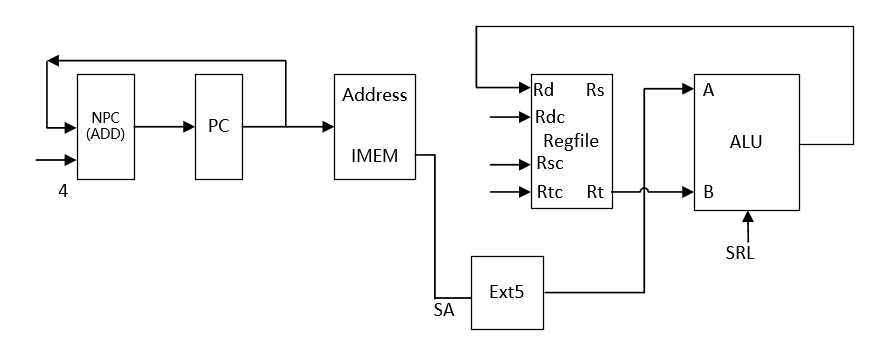
格式：SRL rd, rt, sa

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

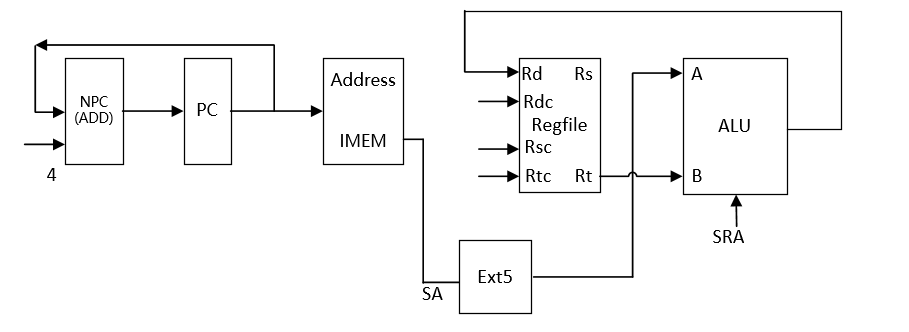
格式：SRA rd, rt, sa

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

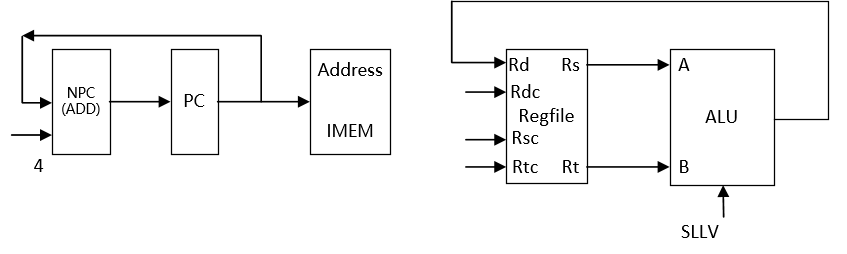
格式：SLLV rd, rt, rs,

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

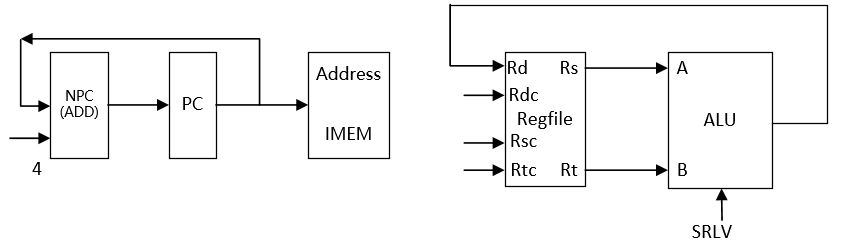
格式：SRLV rd, rt,rs

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

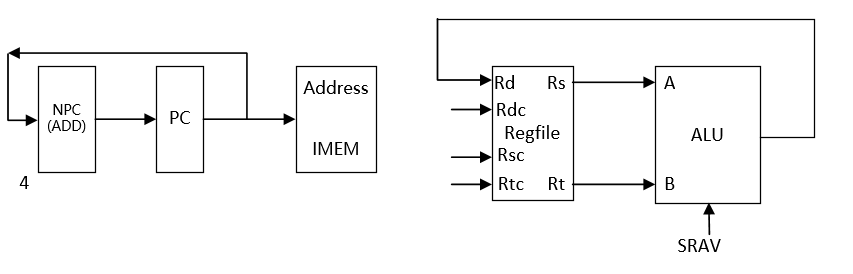
格式：SRAV rd, rt,rs

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

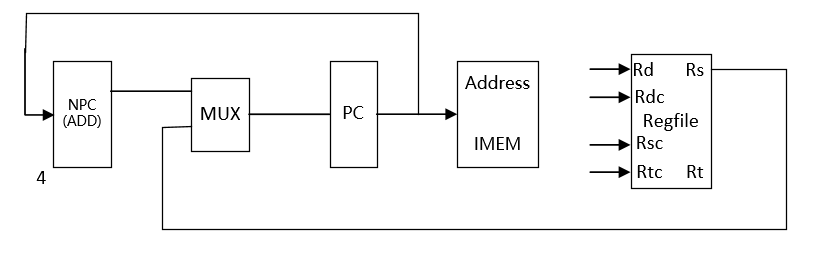
格式：JR rs

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

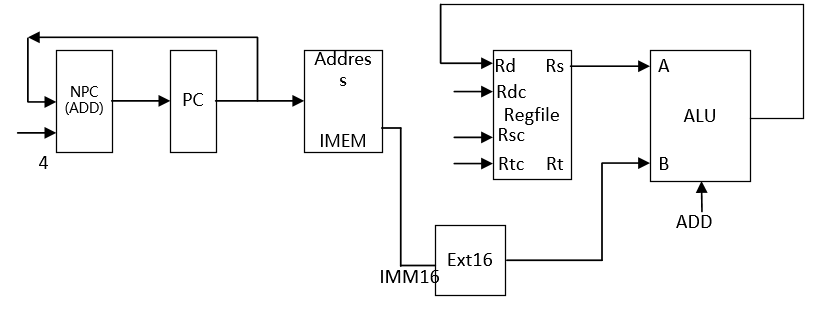
格式：ADDI rt, rs, imm16

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

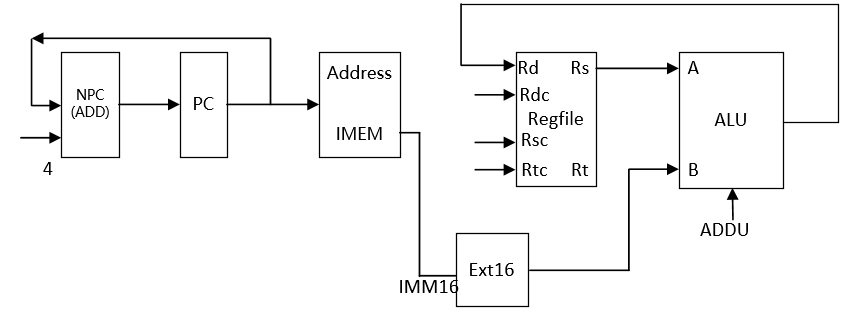
格式：ADDIU rt, rs, imm16

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

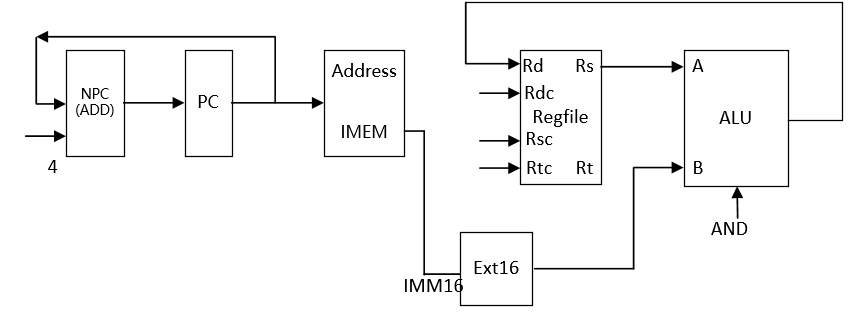
格式：ANDI rt, rs, imm16

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

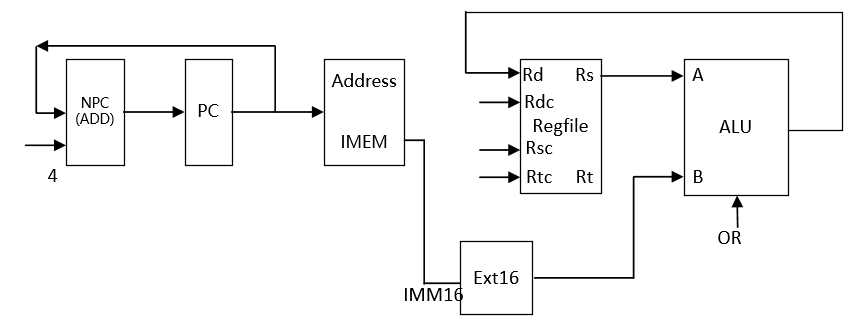
格式：ORI rt, rs, imm16

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

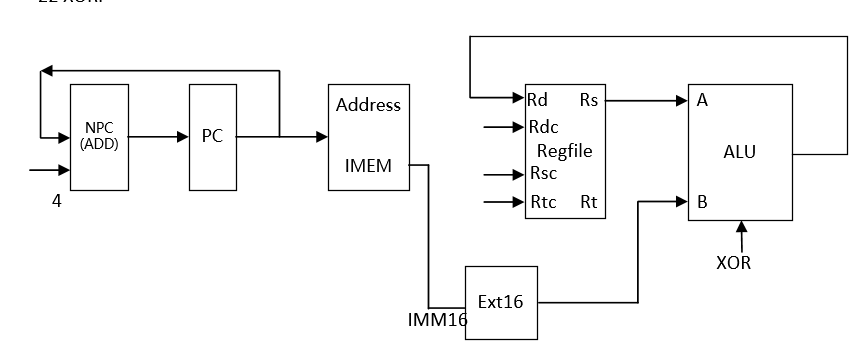
格式：XORI rt, rs, imm16

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

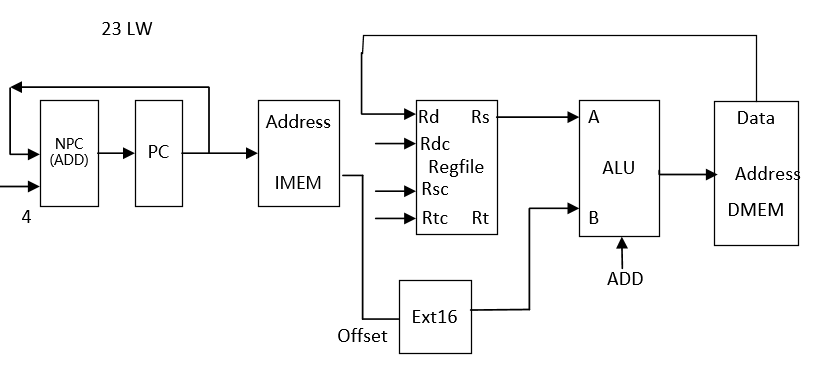
格式：LW rt, offset(base)

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

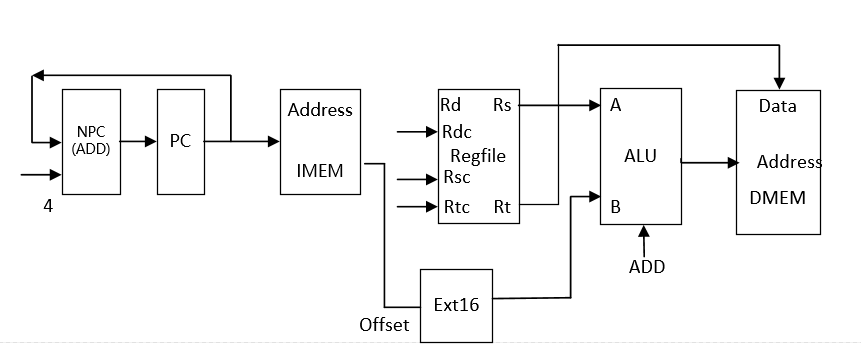
格式：SW rt, offset(base)

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

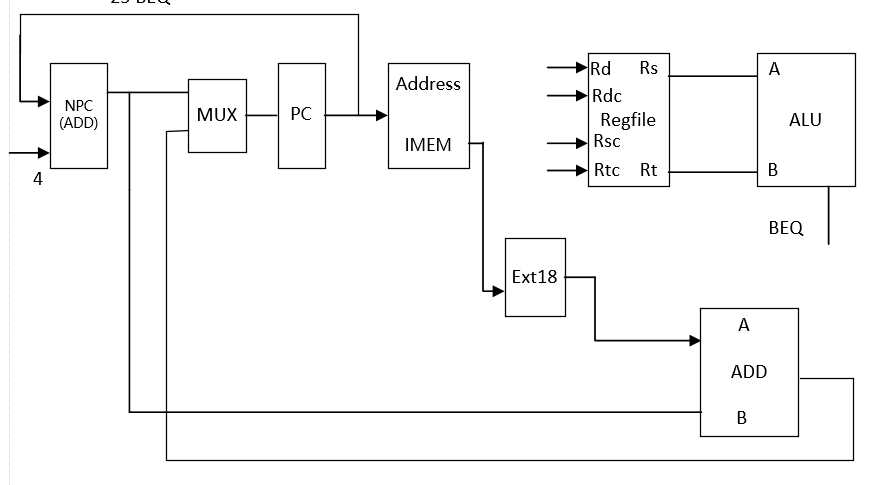
操作: if (rs=rt)PC←NPC + Sign\_ext(offset||02)

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

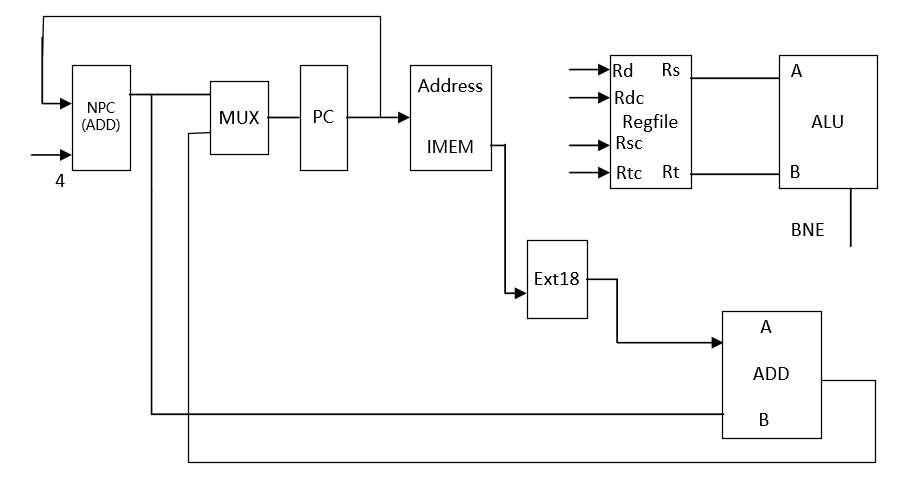
操作：if (rs≠rt)PC←NPC + Sign\_ext(offset||02)

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

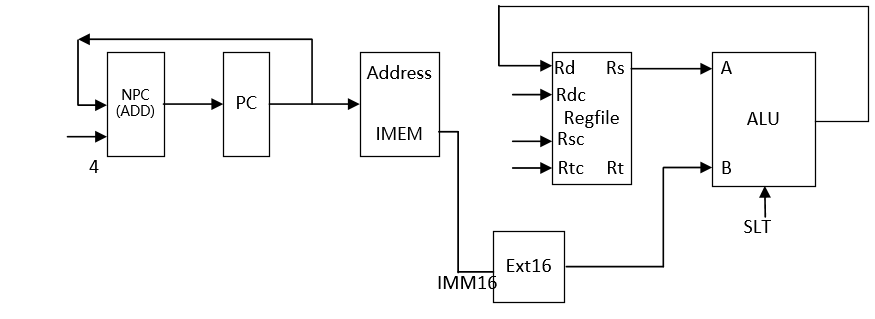
格式：SLTI rt, rs, imm16

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

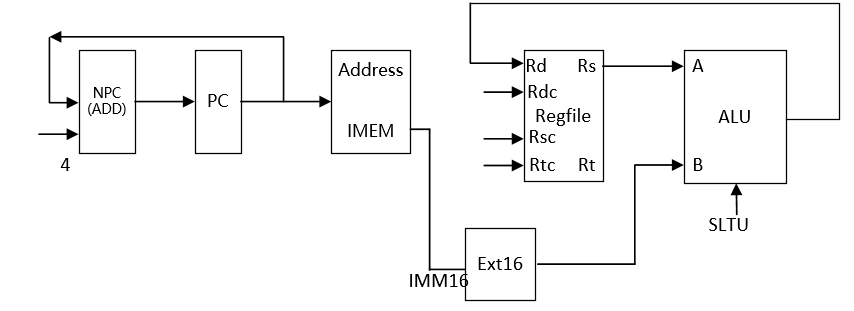
格式：SLTIU rt, rs, imm16

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

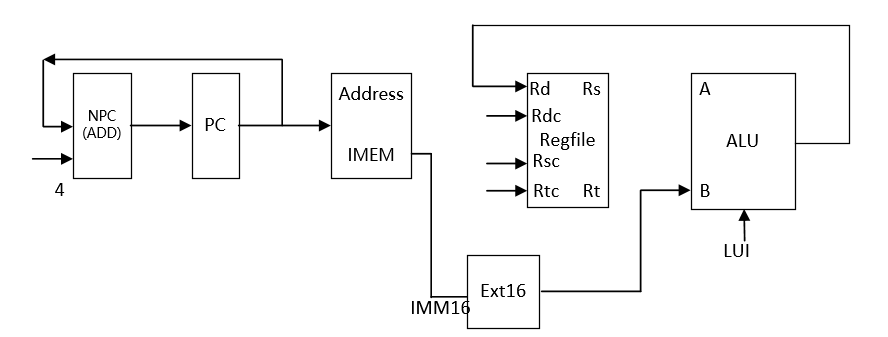
格式：LUI rt, imm16

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

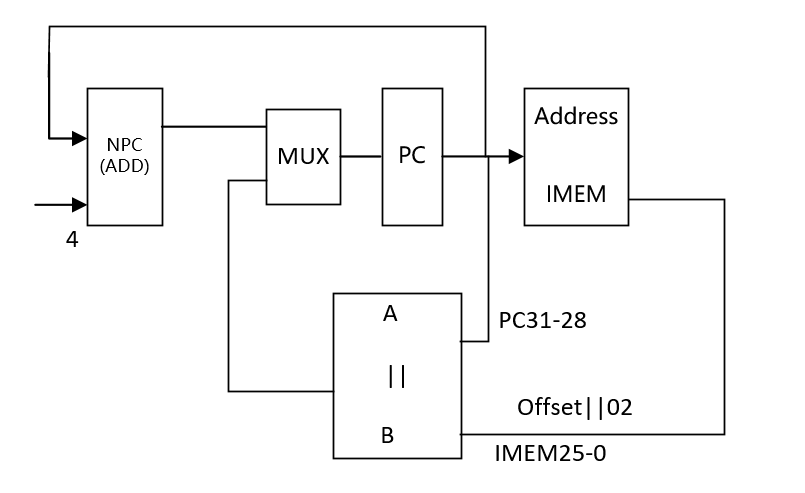
格式：J target

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

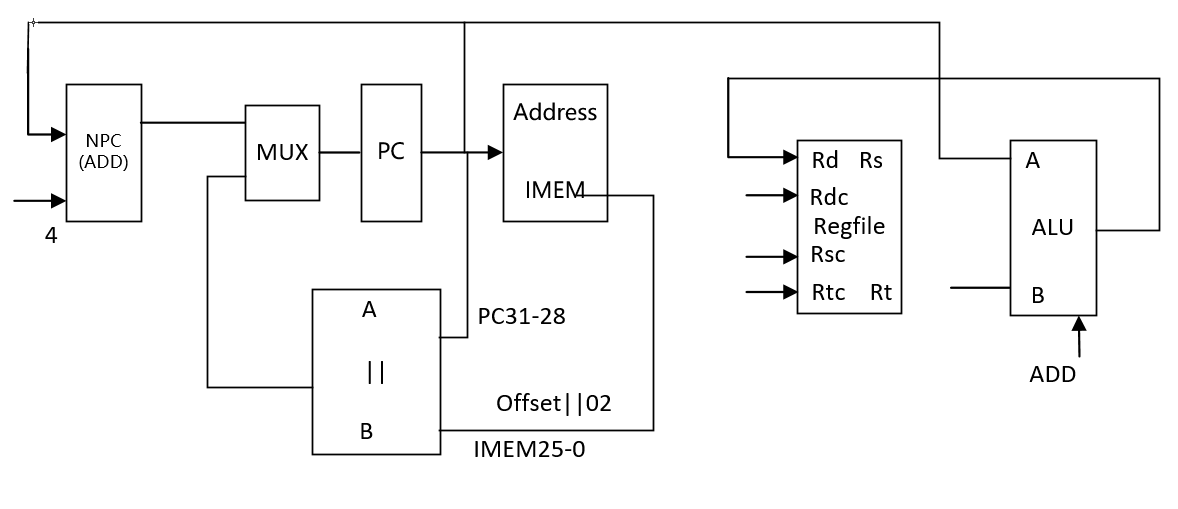
格式：JAL target

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

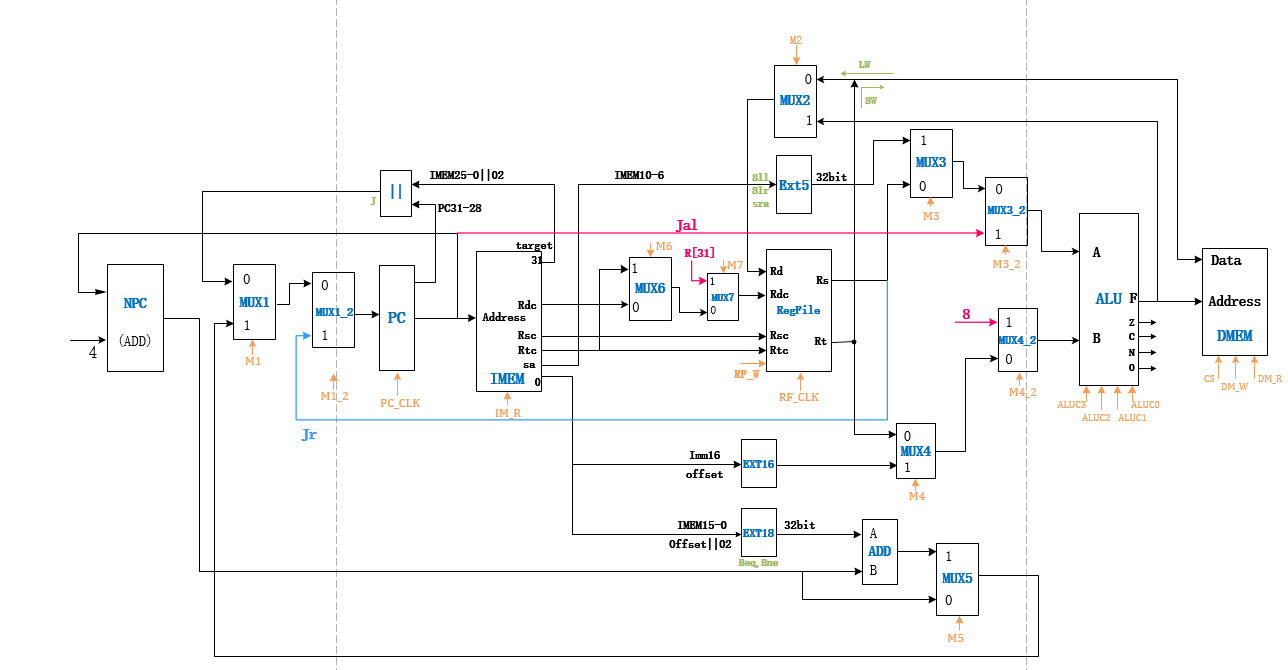
所需部件：：PC、NPC、IMEM

输入输出关系：

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



（二）整个数据通路



（三）部件表：

**31条指令CPU所需部件及数据通路图**

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

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | PC | NPC | IMEM | RegFile | | ALU | | Ext18 | ADD | | || | |
|  |  |  |  |  | Rd | Rdc | A | B |  | A | B | A | B |
| 27 | Beq | ADD | PC | PC |  |  | Rs | Rt | Offset | NPC | Ext18 |  |  |
| 28 | Bne | ADD | PC | PC |  |  | Rs | Rt | Offset | NPC | Ext18 |  |  |
| 29 | J | || | PC | PC |  |  |  |  |  |  |  | PC  31-28 | IMEM 25-0 |
| 30 | Jal | || | PC | PC | ALU |  | PC | 8 |  |  |  | PC  31-28 | IMEM 25-0 |
| 31 | jr | rs | PC | PC |  |  |  |  |  |  |  |  |  |

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 [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;

assign inst=imm;

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

//imem im(pc,inst)

//IMEM imem(1,((pc- 32'h00400000)/4),imm);

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

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

DMEM dram(

.clk(clk\_in),

.CS(cs), //enable control signal

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

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

.Addr((alu\_r[9:0]-32'h10010000)/4),

.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 [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]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 RL\_CLK; //regfiles clk

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

wire [31:0] PC;

wire [31:0] mux\_out\_1;

wire [31:0] mux\_out\_1\_2;

wire [31:0] mux\_out\_2;

wire [31:0] mux\_out\_3;

wire [31:0] mux\_out\_3\_2;

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

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

wire [31:0] mux\_out\_4\_2;

wire [31:0] mux\_out\_5;

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;

//assign RDC\_T = rdc;

assign mux\_out\_2\_T=mux\_out\_2;

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;

//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;

wire exception;

wire [4:0] cause;

wire wdata;

assign wdata = Rt;

wire [31:0] rdata;

wire [31:0] status;

wire [31:0] exc\_addr; //un

//Control signal expression

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

assign M3\_2 = \_jal ;

assign M4 = \_addi || \_addiu || \_andi || \_ori || \_xori || \_slti || \_sltiu || \_lui || \_lw || \_sw ;

assign M4\_2 = \_jal ;

assign M6 =\_addi||\_addiu||\_andi||\_ori||\_xori|||\_slti||\_sltiu||\_lui||\_lw;//||\_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;

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

assign RF\_W= (!\_sw)&&(!\_beq)&&(!\_bne)&&(!\_j)&&(!\_jr);

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

assign M5 = (\_beq&&zero) || (\_bne&&(!zero)) ;

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

assign M1\_2 = \_jr ;

assign cs = \_lw || \_sw;

assign dm\_r = \_lw;

assign dm\_w = \_sw;

assign ext16\_sin\_judge = \_addi || \_addiu || \_slti||\_sltiu || \_lw; //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 mux\_out\_1=M1?mux\_out\_5:{PC[31:28],imem[25:0],2'b00}; //mux1

assign mux\_out\_1\_2=M1\_2?Rs:mux\_out\_1; //mux1\_2

assign mux\_out\_2=M2?alu\_r:ram\_out; //mux2

assign mux\_out\_3=M3?{27'b0,imem[10:6]}:Rs;

assign mux\_out\_3\_2=M3\_2?PC:mux\_out\_3;

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

assign mux\_out\_4\_2=M4\_2?32'd4:mux\_out\_4;

assign mux\_out\_5=M5?ext18\_sign+npc:npc;

pcreg PCreg(

.clk(clk\_in),

.rst(reset),

.ena(1),

//.wena(1),

.data\_in(mux\_out\_1\_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(mux\_out\_2), //rd

.rdata1(Rs), //rs

.rdata2(Rt) //rt

);

ALU alu(

.a(mux\_out\_3\_2),

.b(mux\_out\_4\_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, //enable control signal

input DM\_W, //write

input DM\_R, //read

input [9:0] Addr,

input [31:0] Data\_in,

output [31:0] Data\_out

);

reg [31:0] num [0:31];

assign Data\_out=CS? (DM\_R? num[Addr]: 32'h00000000):32'bzzzz\_zzzz\_zzzz\_zzzz\_zzzz\_zzzz\_zzzz\_zzzz;

always@(negedge clk or negedge CS)

begin

if(CS&&DM\_W)num[Addr]<=Data\_in;

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

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

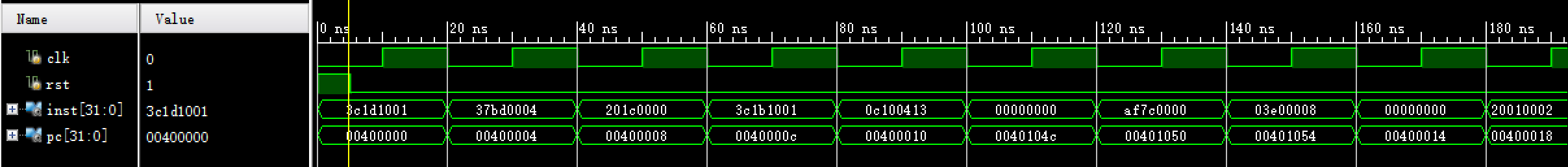
data\_out <= mem[Addr];

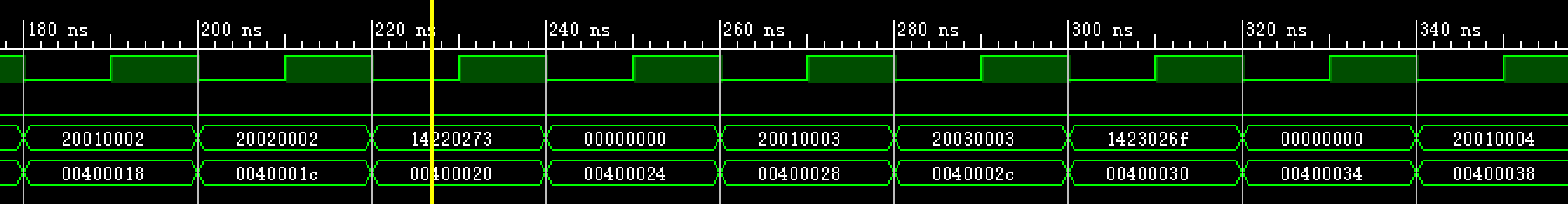
end

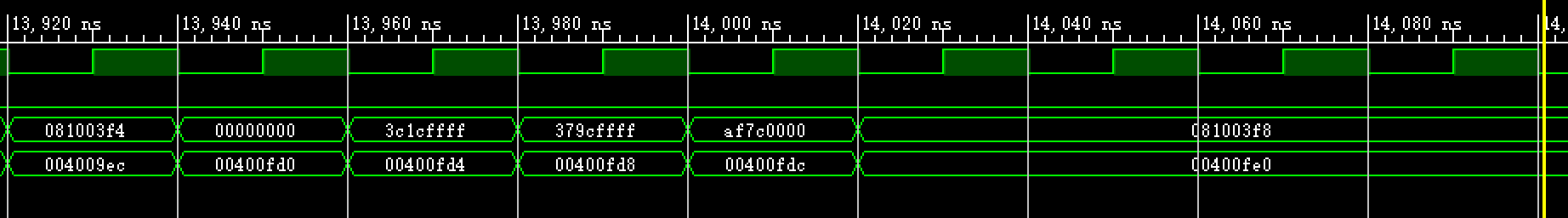
endmodule

使用$readmemh初始化IMEM，读取相应的指令，进行测试

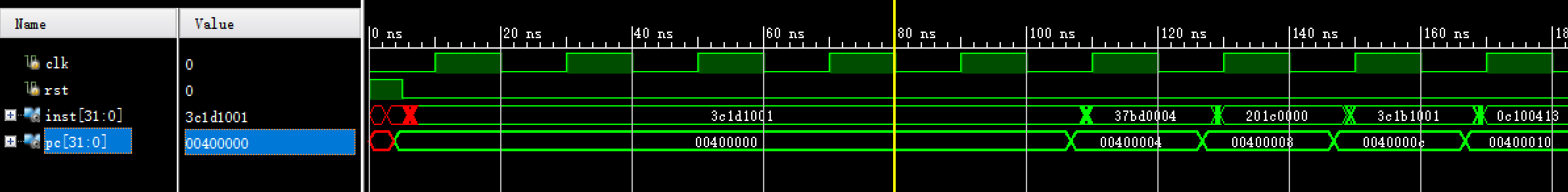
1. 实验结果
2. 前仿真测试

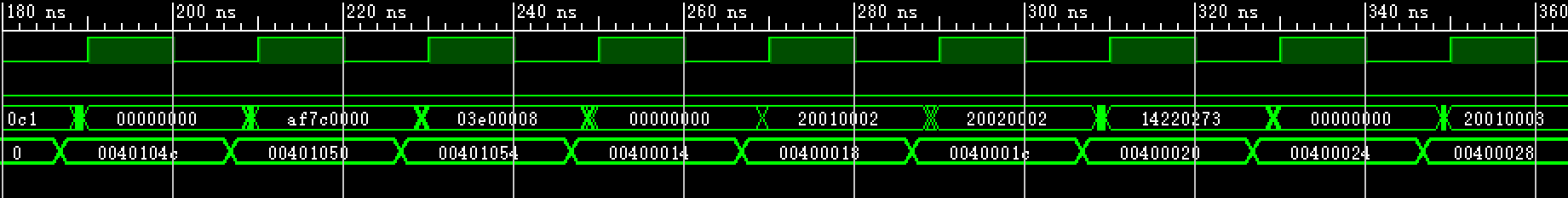


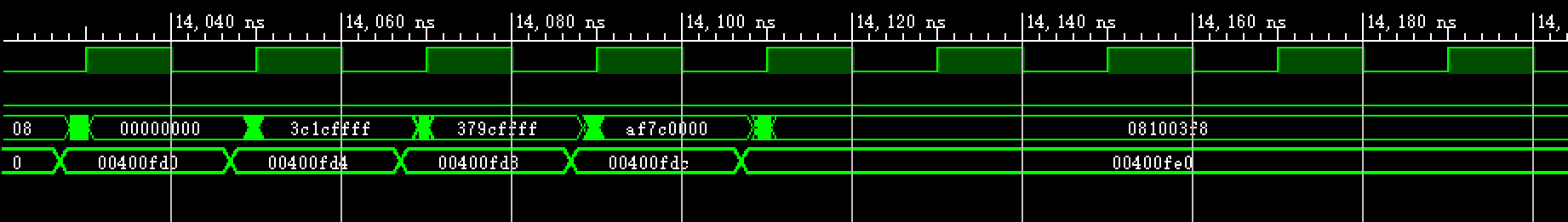




1. 后仿真测试

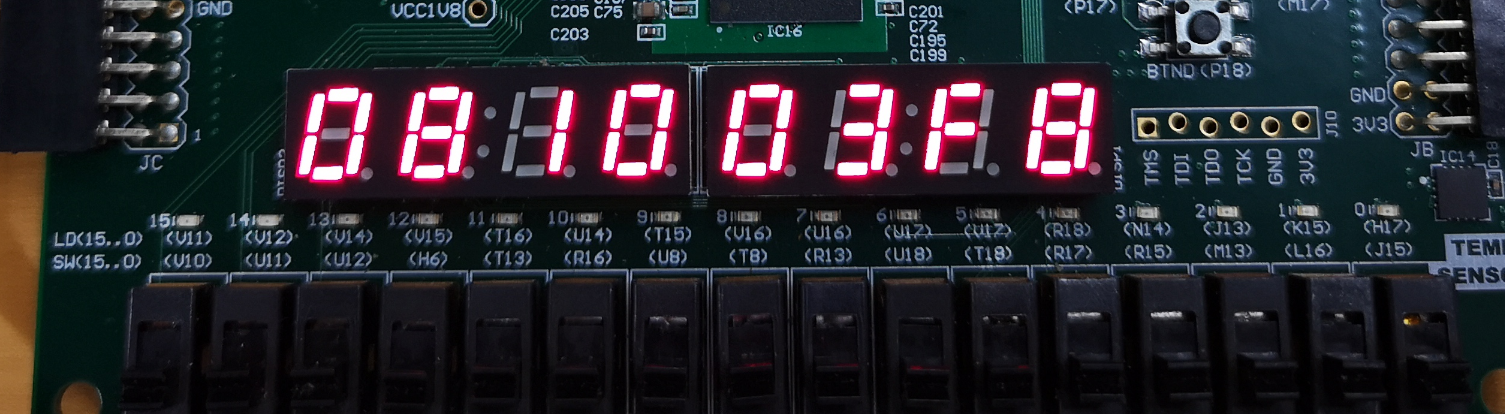






1. 下板测试

Inst:



PC:

