同济大学计算机系

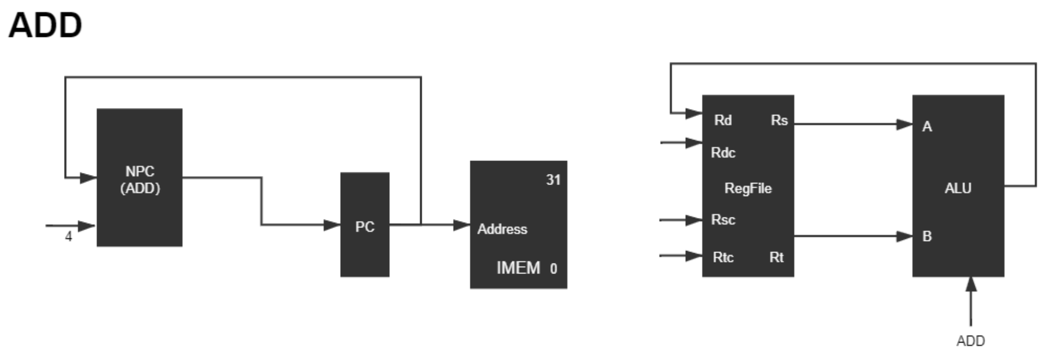
计算机组成原理实验报告

学　　号	**1951108**
姓　　名	刘嘉文
专　　业	信息安全
授课老师	陈永生

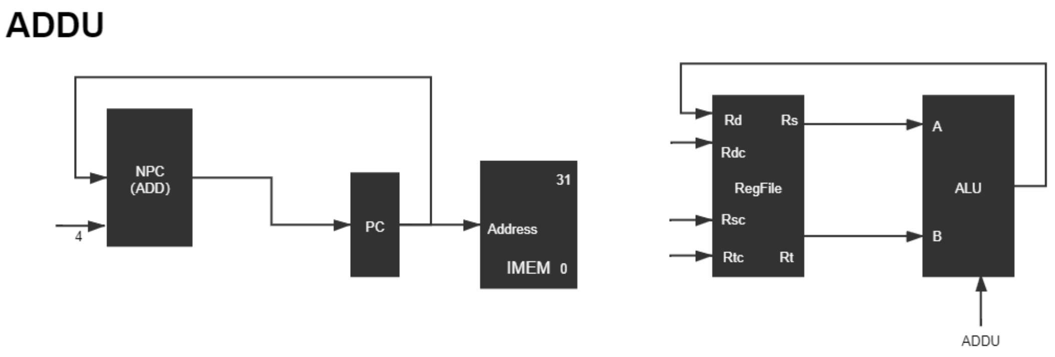
一、实验内容

在本次实验中，我们将使用 Verilog HDL 语言实现 54 条 MIPS 指令的 CPU 的设计和仿真和下板

二、指令流程图

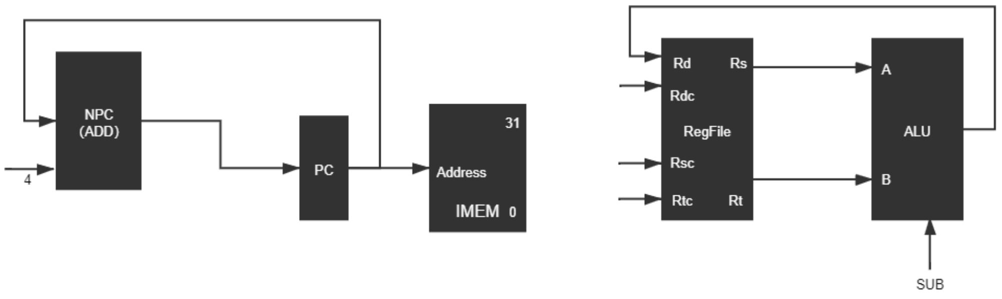


操作：Add rd,rs,rt ;rd←rs+rt,PC←NPC(PC+4)
所需部件：PC、NPC、IMEM、Regfile、ALU



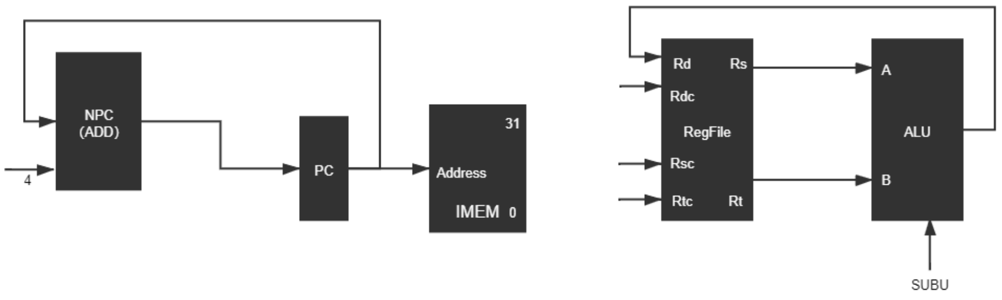
操作：Addu rd,rs,rt ;rd←rs+rt,PC←NPC(PC+4)
所需部件：PC、NPC、IMEM、Regfile、ALU

SUB



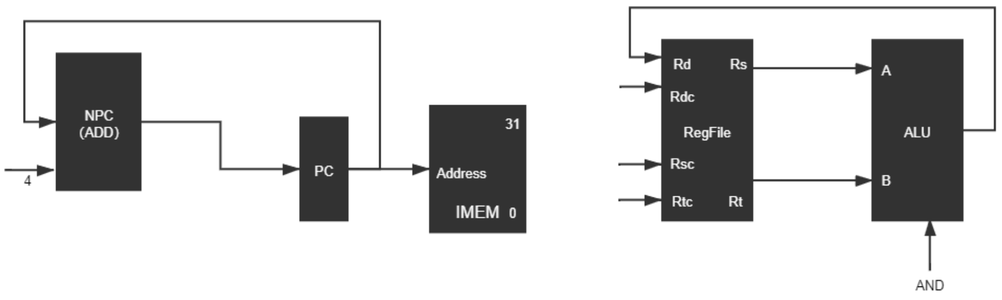
操作：Sub rd,rs,rt ;rd←rs-rt,PC←NPC(PC+4)
所需部件：PC、NPC、IMEM、Regfile、ALU

SUBU



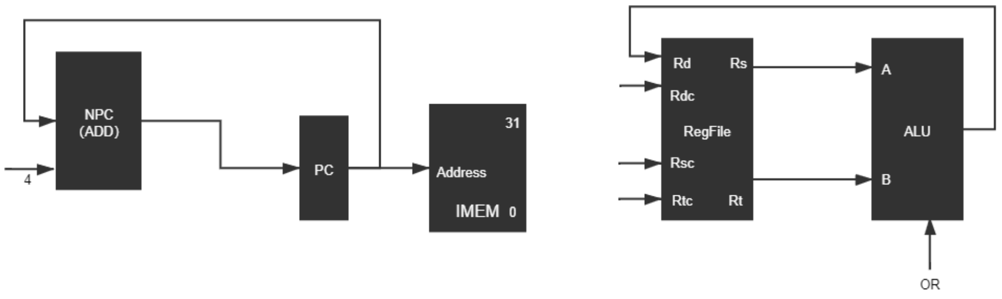
操作：Subu rd,rs,rt ;rd←rs-rt,PC←NPC(PC+4)
所需部件：PC、NPC、IMEM、Regfile、ALU

AND



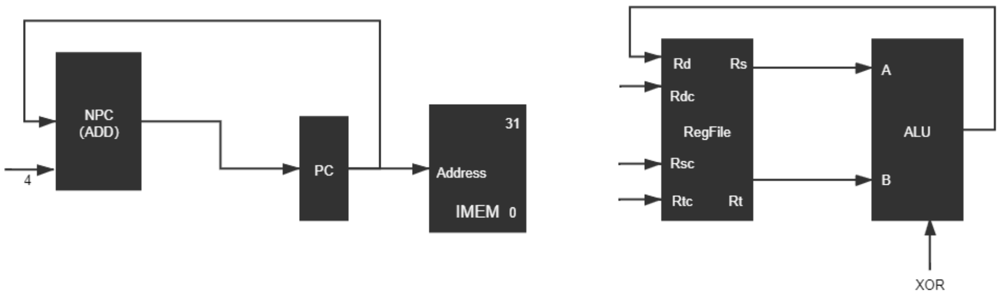
操作：And rd,rs,rt ;rd←rs&rt,PC←NPC(PC+4)
所需部件：PC、NPC、IMEM、Regfile、ALU

OR



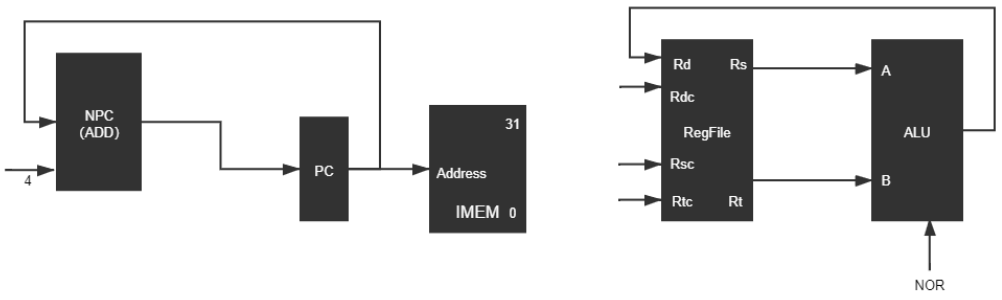
操作：Or rd,rs,rt ;rd←rs||rt,PC←NPC(PC+4)
所需部件：PC、NPC、IMEM、Regfile、ALU

XOR



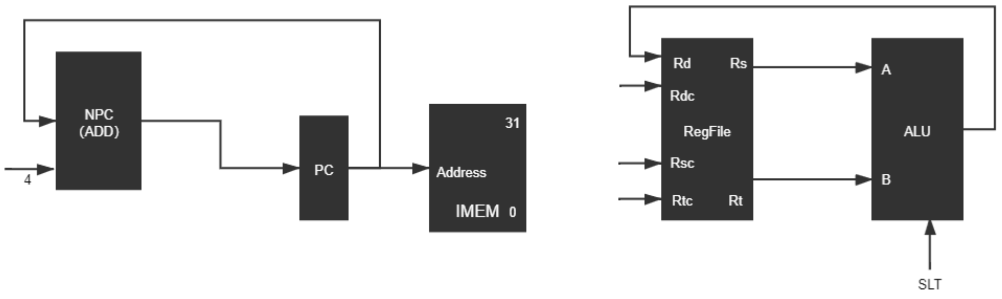
操作：Xor rd,rs,rt ;rd←rs^rt,PC←NPC(PC+4)
所需部件：PC、NPC、IMEM、Regfile、ALU

NOR



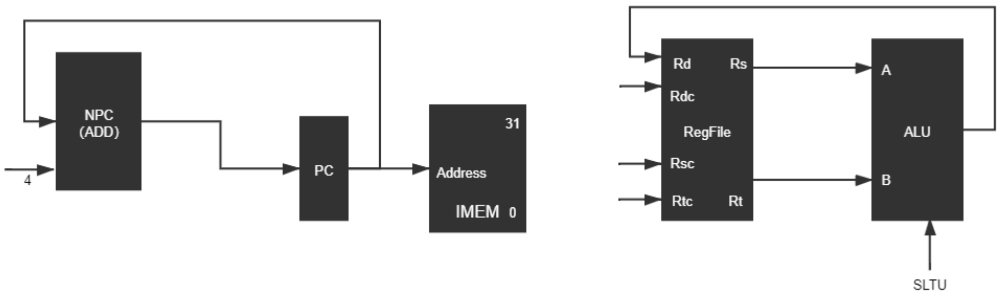
操作：Nor rd,rs,rt ;rd←!(rs^rt),PC←NPC(PC+4)
所需部件：PC、NPC、IMEM、Regfile、ALU

SLT



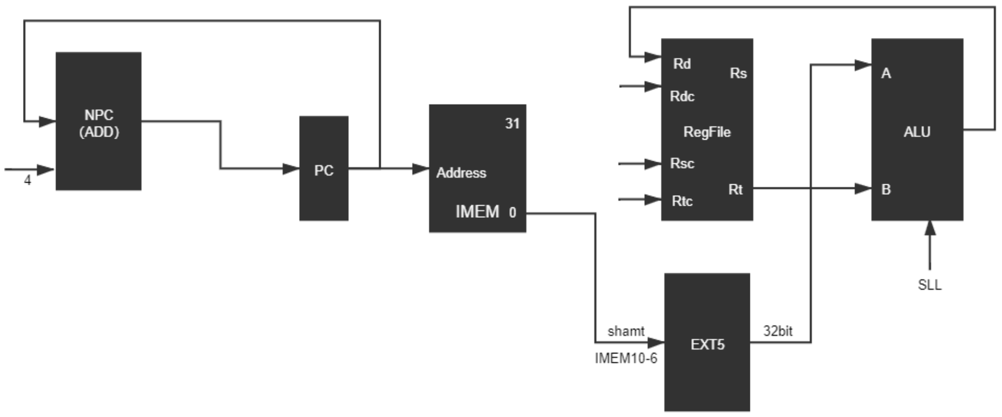
操作：Slt rd,rs,rt ;rd←rs<<rt,PC←NPC(PC+4)
所需部件：PC、NPC、IMEM、Regfile、ALU

SLTU



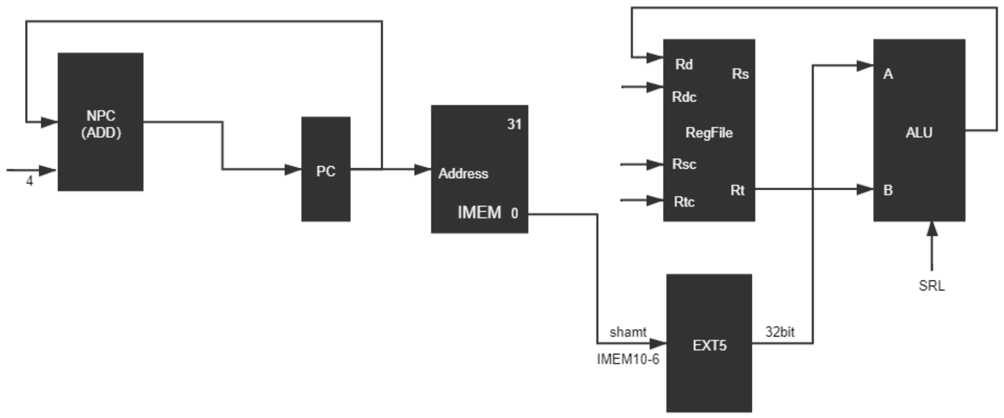
操作：Sltu rd,rs,rt ;rd←rs<<rt,PC←NPC(PC+4)
所需部件：PC、NPC、IMEM、Regfile、ALU

SLL



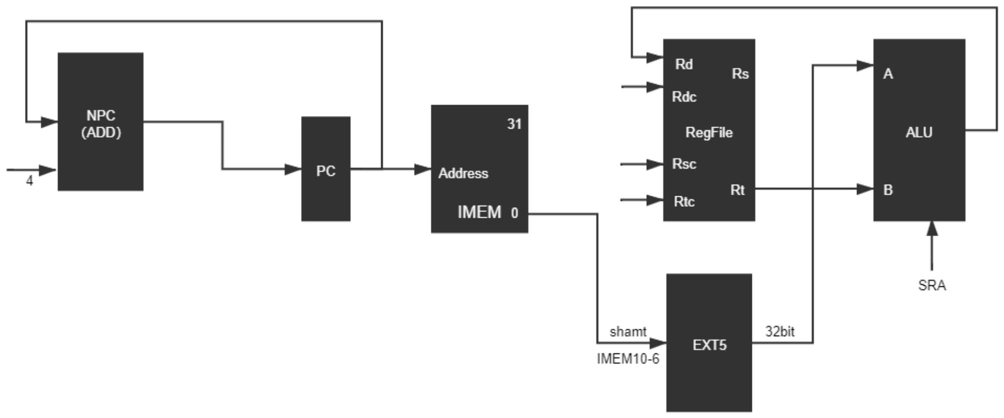
操作：Sll rd,rt,shamt ;rd←rt<<shamt,PC←NPC(PC+4)
所需部件：PC、NPC、IMEM、Regfile、ALU、EXT5

SRL



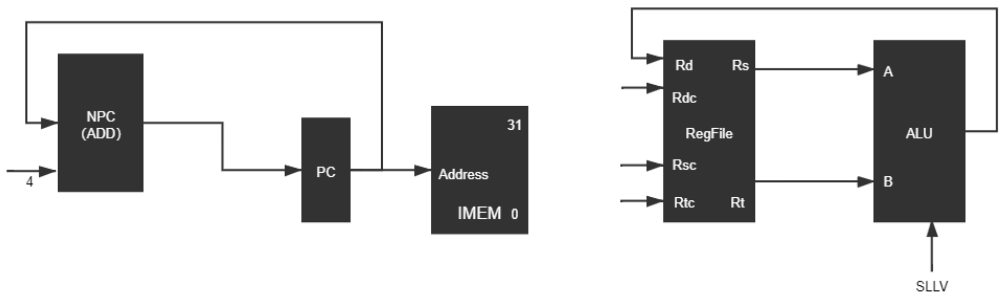
操作：Srl rd,rt,shamt;rd←rt>>shamt,PC←NPC(PC+4)
所需部件：PC、NPC、IMEM、Regfile、ALU、EXT5

SRA



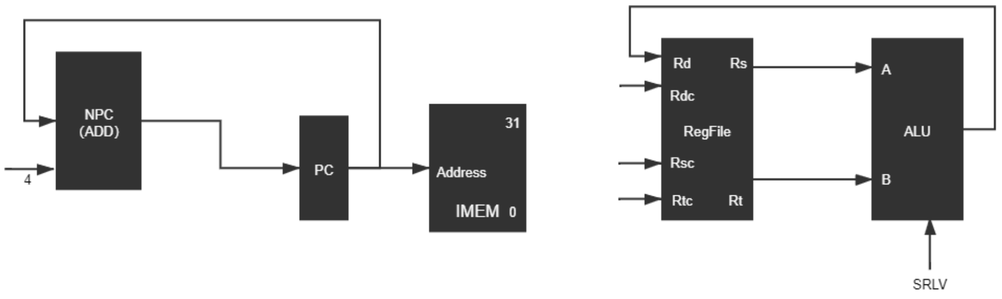
操作：Sra rd,rt,shamt;rd←rt>>>shamt,PC←NPC(PC+4)
所需部件：PC、NPC、IMEM、Regfile、ALU、EXT5

SLLV



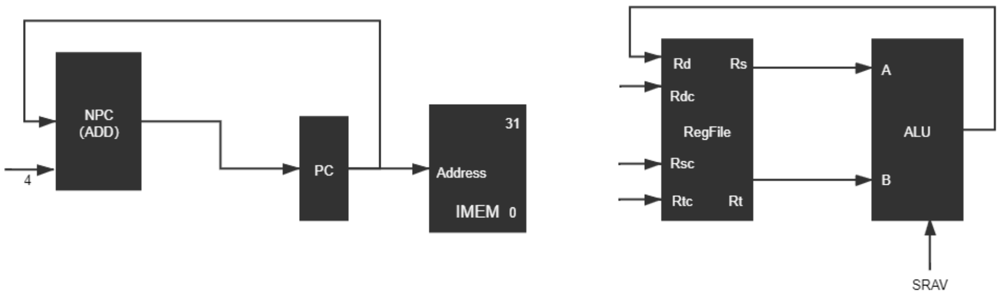
操作：Sllv rd,rs,rt ;rd←rs<<rt,PC←NPC(PC+4)
所需部件：PC、NPC、IMEM、Regfile、ALU

SRLV



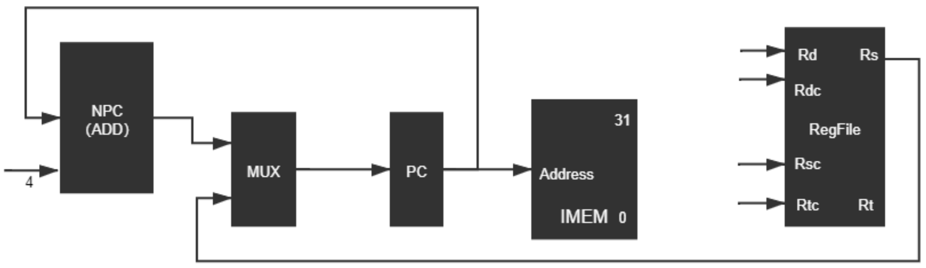
操作：Srlv rd,rs,rt ;rd←rs>>rt,PC←NPC(PC+4)
所需部件：PC、NPC、IMEM、Regfile、ALU

SRAV



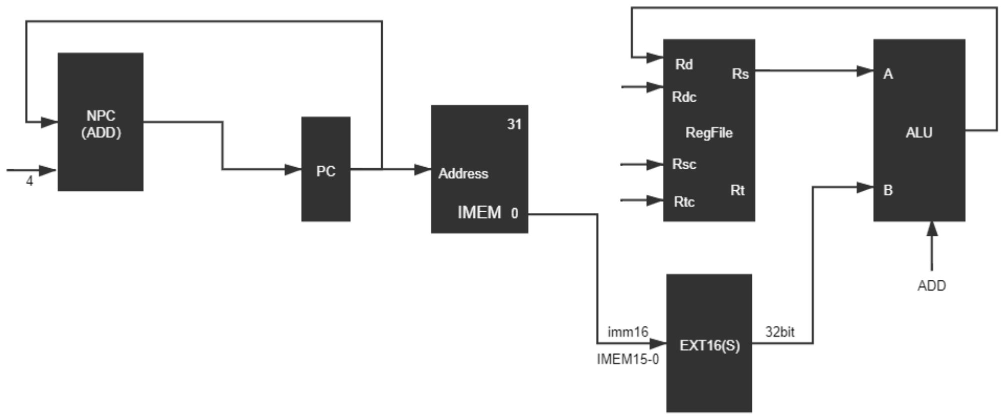
操作：Srav rd,rs,rt ;rd←rs>>>rt,PC←NPC(PC+4)
所需部件：PC、NPC、IMEM、Regfile、ALU

JR



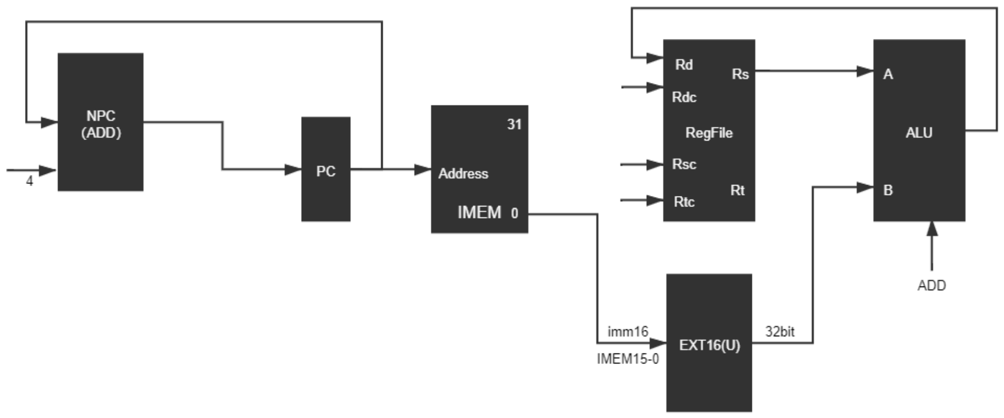
操作：J rs; PC←rs;
所需部件：PC、NPC、IMEM、Regfile

ADDI



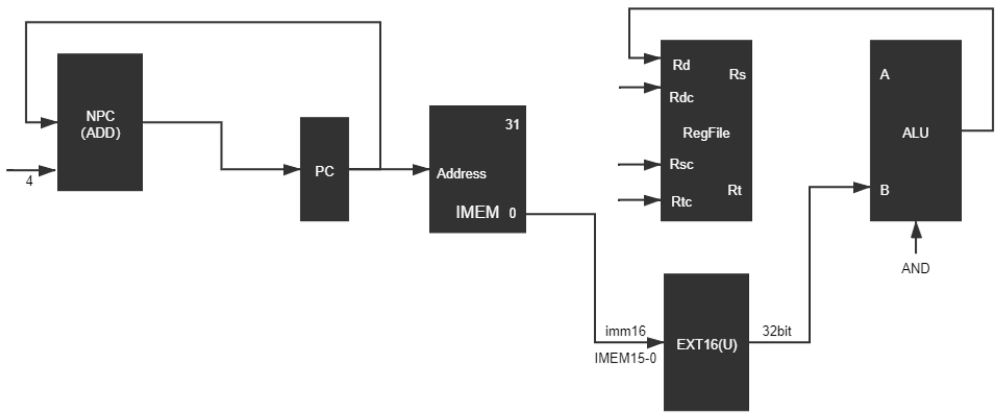
操作：Addi rd,rs,imm ;rd←rs+imm,PC←NPC(PC+4)
所需部件：PC、NPC、IMEM、Regfile、ALU、EXT16(S)

ADDIU



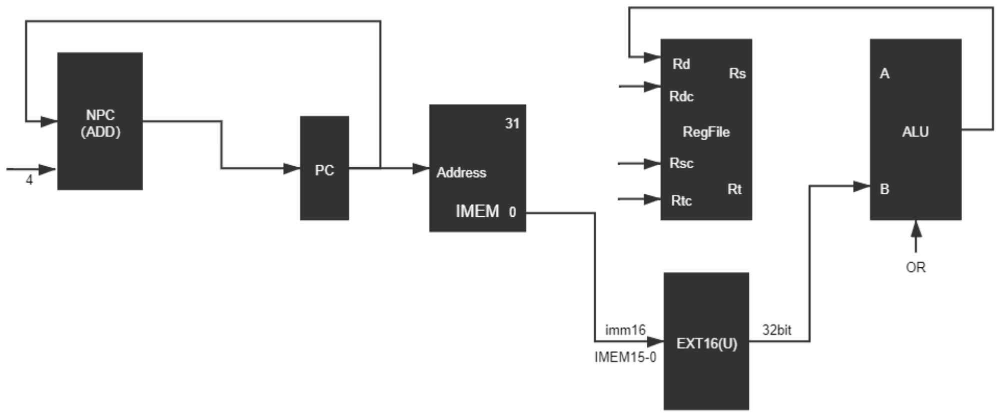
操作：Addiu rd,rs,imm ;rd←rs+imm,PC←NPC(PC+4)
所需部件：PC、NPC、IMEM、Regfile、ALU、EXT16(U)

ANDI



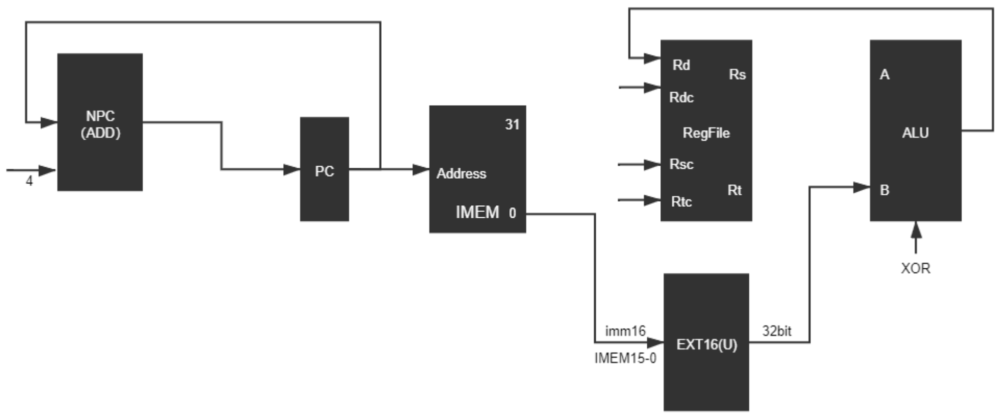
操作：Andi rd,rs,imm ;rd←rs&imm,PC←NPC(PC+4)
所需部件：PC、NPC、IMEM、Regfile、ALU、EXT16(U)

ORI



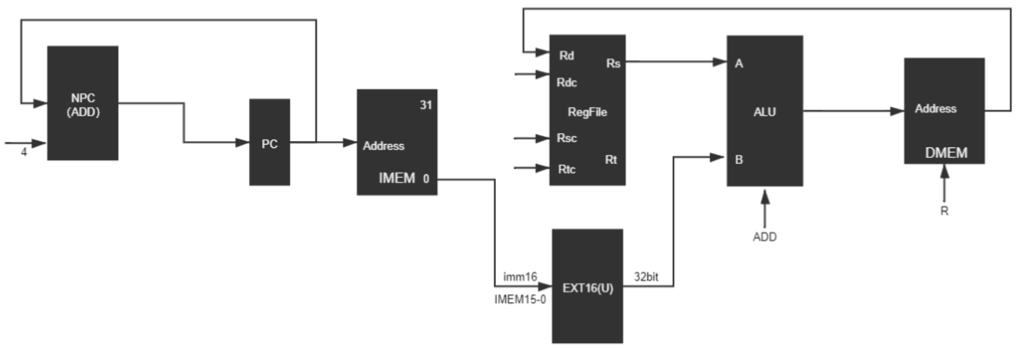
操作：Ori rd,rs,imm ;rd←rs||imm,PC←NPC(PC+4)
所需部件：PC、NPC、IMEM、Regfile、ALU、EXT16(U)

XORI



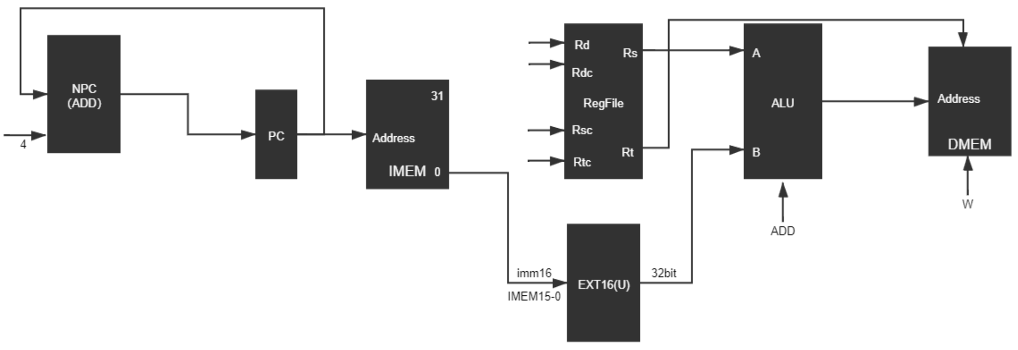
操作：Xori rd,rs,imm ;rd←rs^imm,PC←NPC(PC+4)
所需部件：PC、NPC、IMEM、Regfile、ALU、EXT16(U)

LW



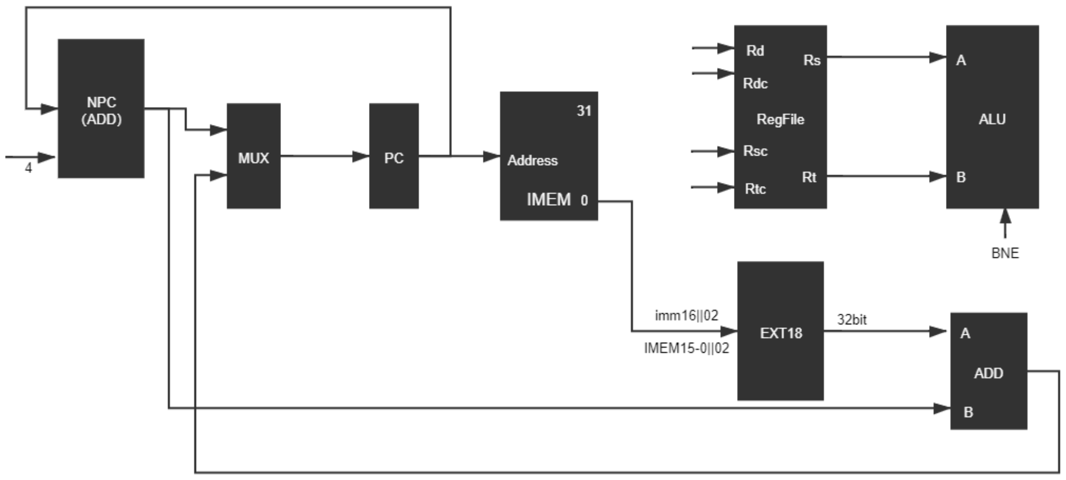
操作：Lw rd,rs,imm ;addr←rs+imm,rd←[addr] ,PC←NPC(PC+4)
所需部件：PC、NPC、IMEM、Regfile、ALU、EXT16(U)、DMEM

SW



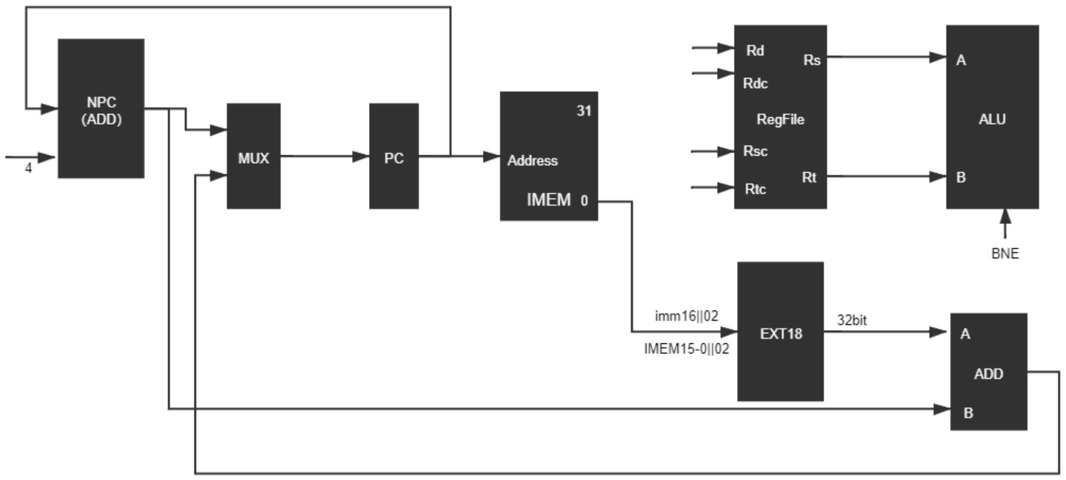
操作：Sw rt,rs,imm ;addr←rs+imm,[addr]←rt ,PC←NPC(PC+4)
所需部件：PC、NPC、IMEM、Regfile、ALU、EXT16(U)、DMEM

BEQ



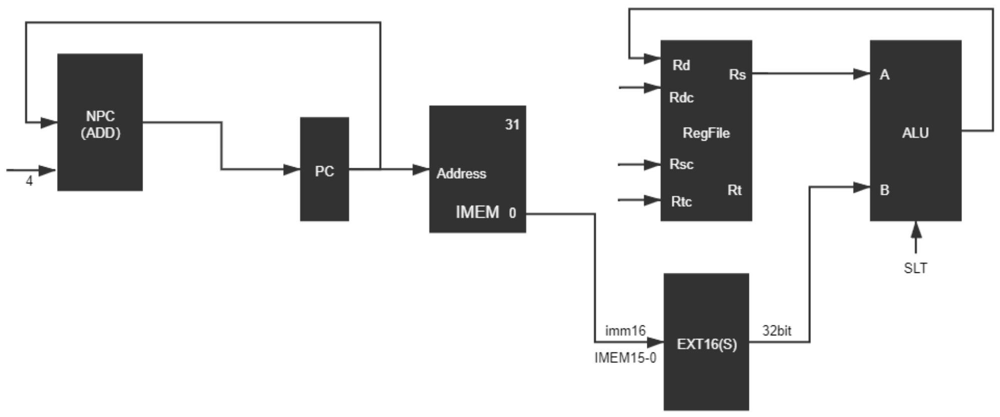
操作：Beq rs,rt,imm ;mux_chose←beq(rs,rt), mux_b←imm+npc ,PC←Mux
所需部件：PC、NPC、IMEM、Regfile、ALU、EXT18、ADD、MUX

BNE



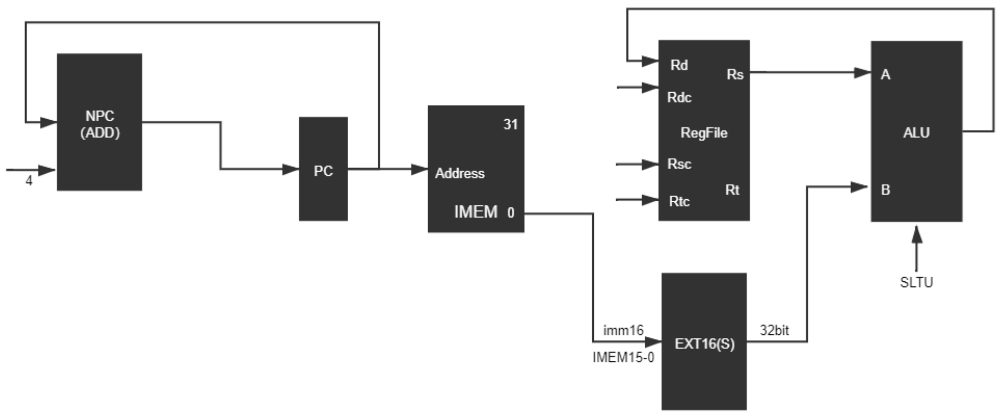
操作：Bne rs,rt,imm ;mux_chose←bne(rs,rt), mux_b←imm+npc ,PC←Mux
所需部件：PC、NPC、IMEM、Regfile、ALU、EXT18、ADD、MUX

SLTI



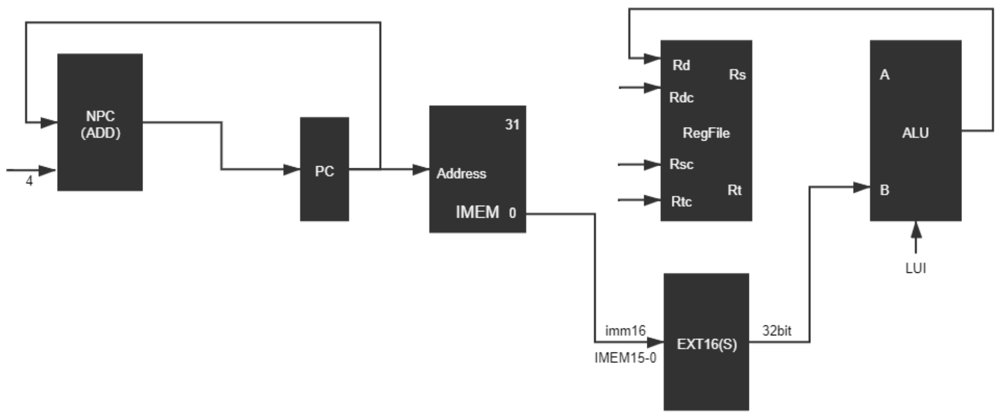
操作：Slti rd,rs,imm ;rd←slt(rs,imm),PC←NPC(PC+4)
所需部件：PC、NPC、IMEM、Regfile、ALU、EXT16(S)

SLTIU



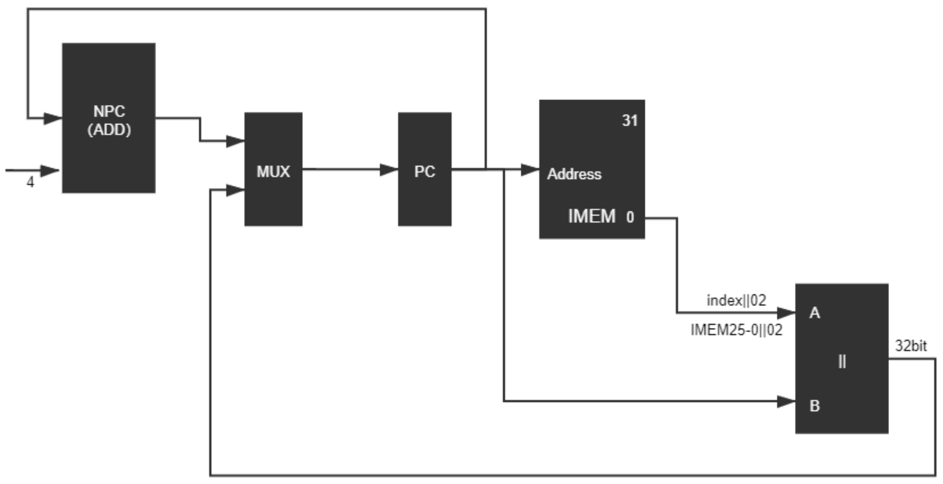
操作：Sltiu rd,rs,imm ;rd←sltu(rs,imm),PC←NPC(PC+4)
所需部件：PC、NPC、IMEM、Regfile、ALU、EXT16(S)

LUI



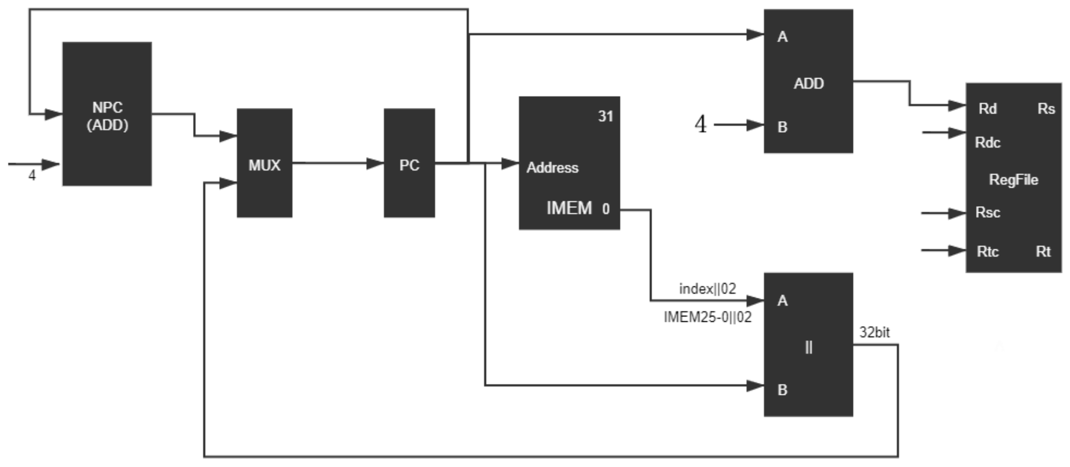
操作：Lui rt,imm ;rd←lui(imm),PC←NPC(PC+4)
所需部件：PC、NPC、IMEM、Regfile、ALU、EXT16(S)

J



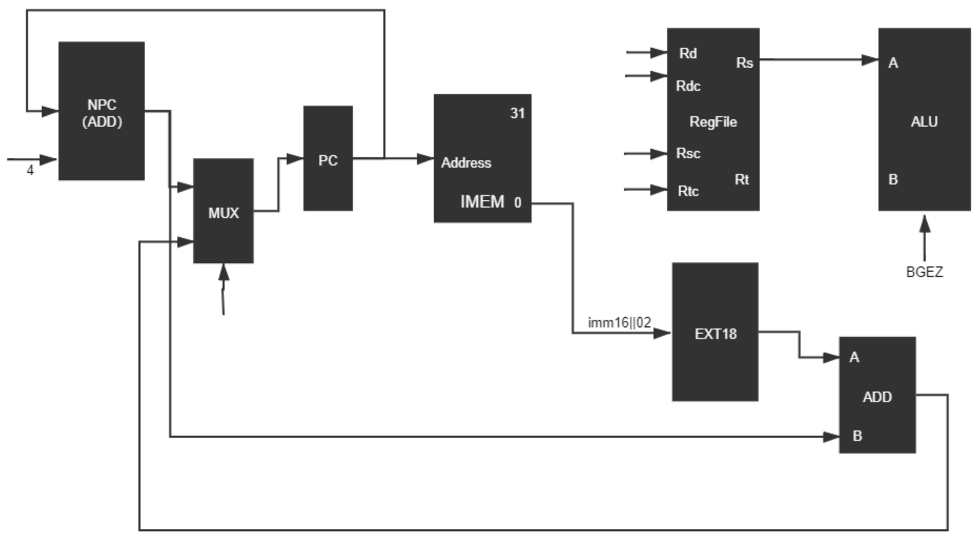
操作：J index; PC←index||00;
所需部件：PC、NPC、IMEM、||

JAL



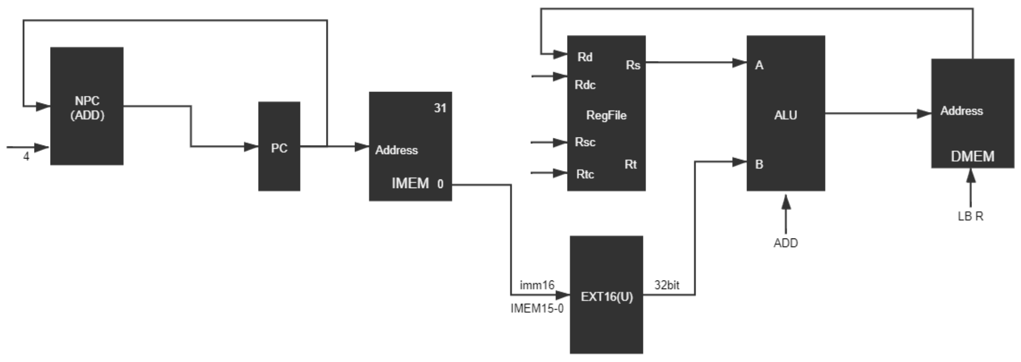
操作：J index; rd←PC+4 PC←index||00
所需部件：PC、NPC、IMEM、||、ADD、Regfile

BGEZ



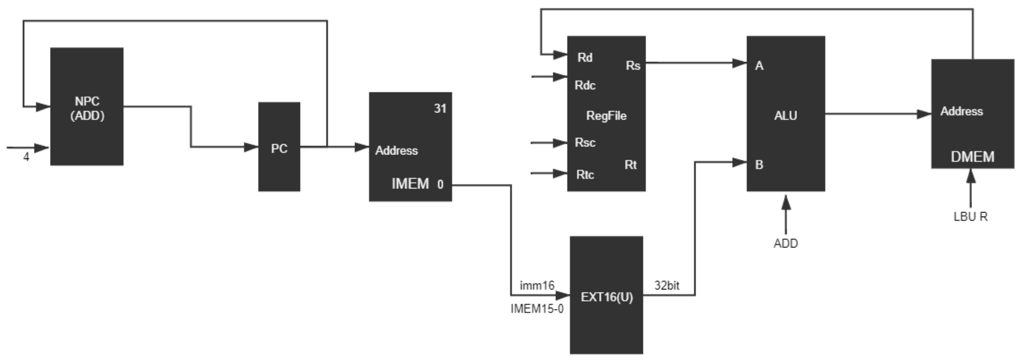
操作：BGEZ rs, offset; B←imm||00 A←Rs BGEZ MUX_C←ALU_out PC←Mux_out;
所需部件：PC、NPC、IMEM、EXT18、Regifile、ALU、MUX

LB



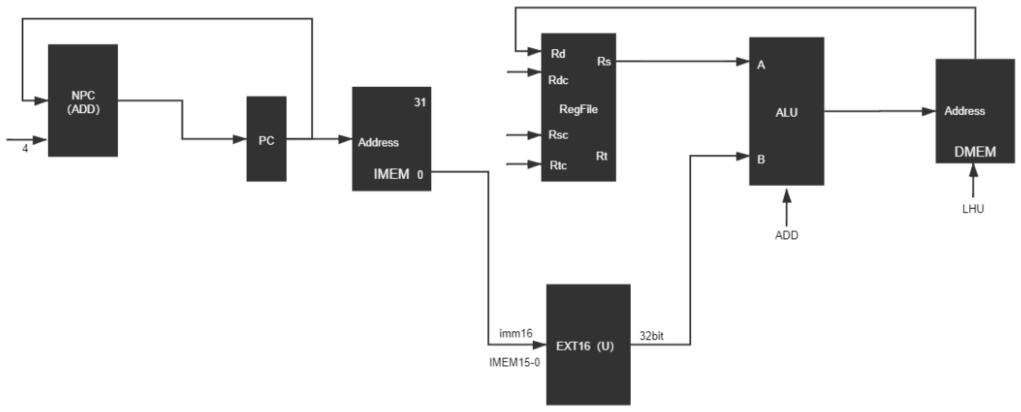
操作：LB rd,rs,imm ;addr←rs+imm,rd←[addr] ,PC←NPC(PC+4)
所需部件：PC、NPC、 IMEM、Regfile、ALU、EXT16(U)、DMEM

LBU



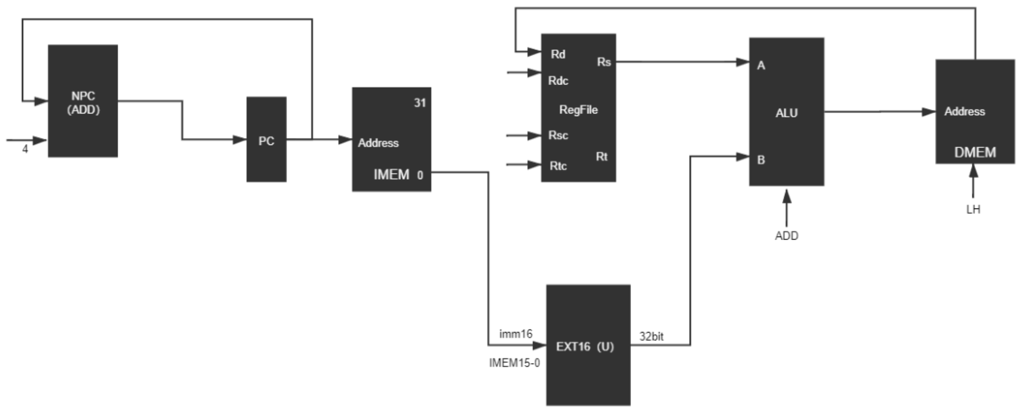
操作：LBU rd,rs,imm ;addr←rs+imm,rd←[addr] ,PC←NPC(PC+4)
所需部件：PC、NPC、 IMEM、Regfile、ALU、EXT16(U)、DMEM

LHU



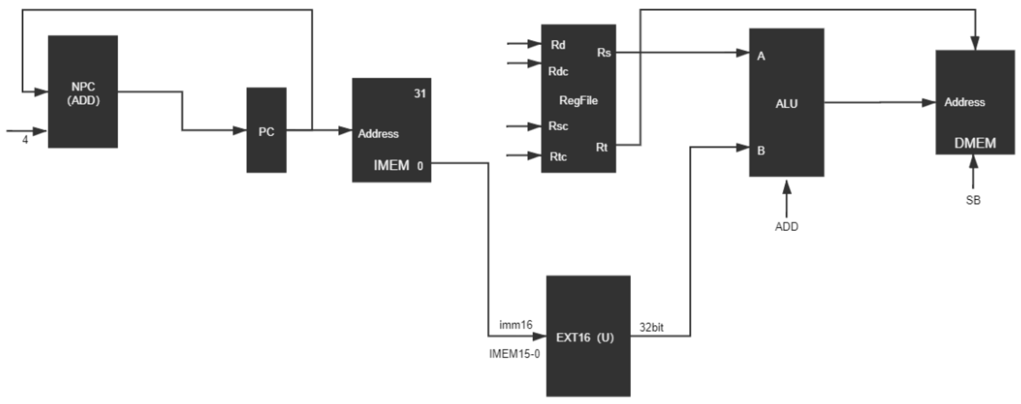
操作：LHU rd,rs,imm ;addr←rs+imm,rd←[addr] ,PC←NPC(PC+4)
所需部件：PC、NPC、IMEM、Regfile、ALU、EXT16(U)、DMEM

LH



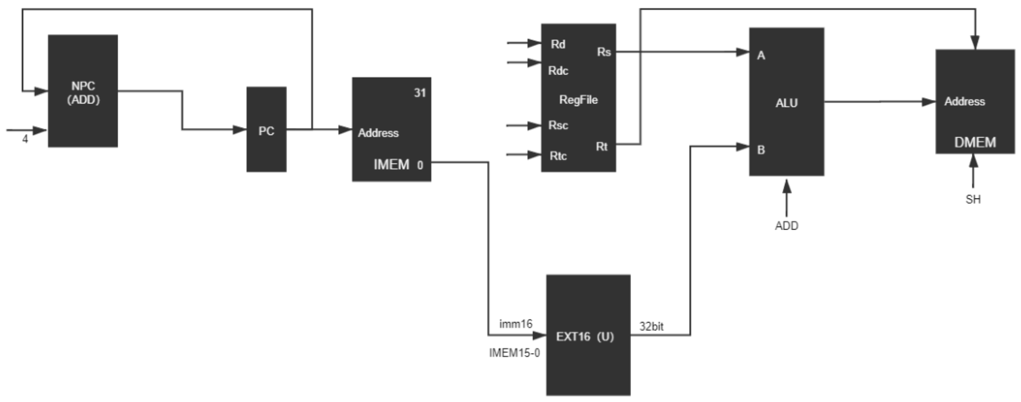
操作：LH rd,rs,imm ;addr←rs+imm,rd←[addr] ,PC←NPC(PC+4)
所需部件：PC、NPC、IMEM、Regfile、ALU、EXT16(U)、DMEM

SB

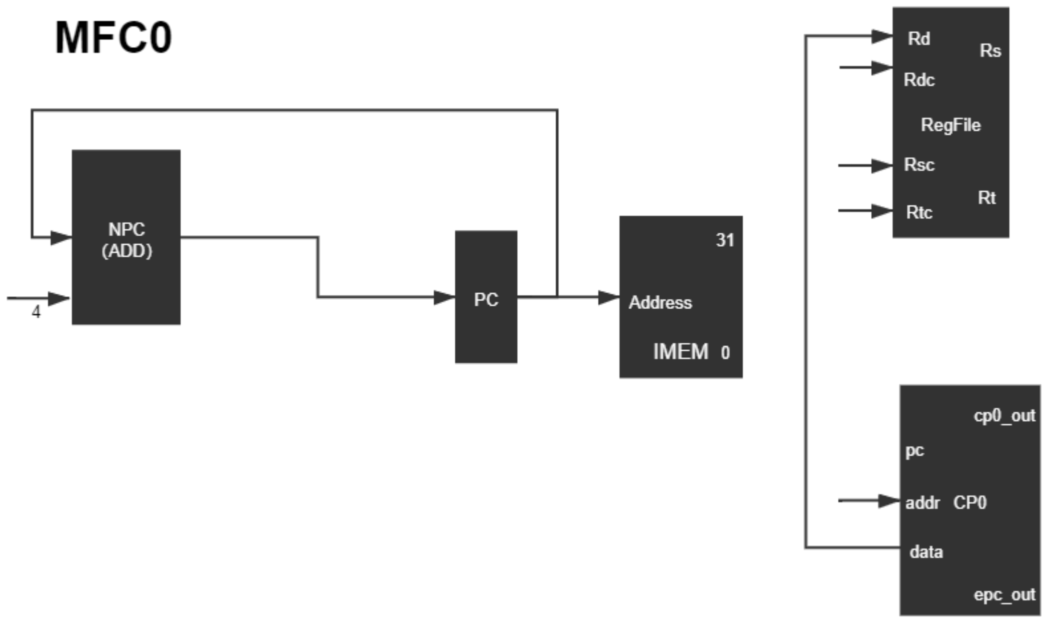


操作：SB rt,rs,imm ;addr←rs+imm,[addr]←rt ,PC←NPC(PC+4)
所需部件：PC、NPC、IMEM、Regfile、ALU、EXT16(U)、DMEM

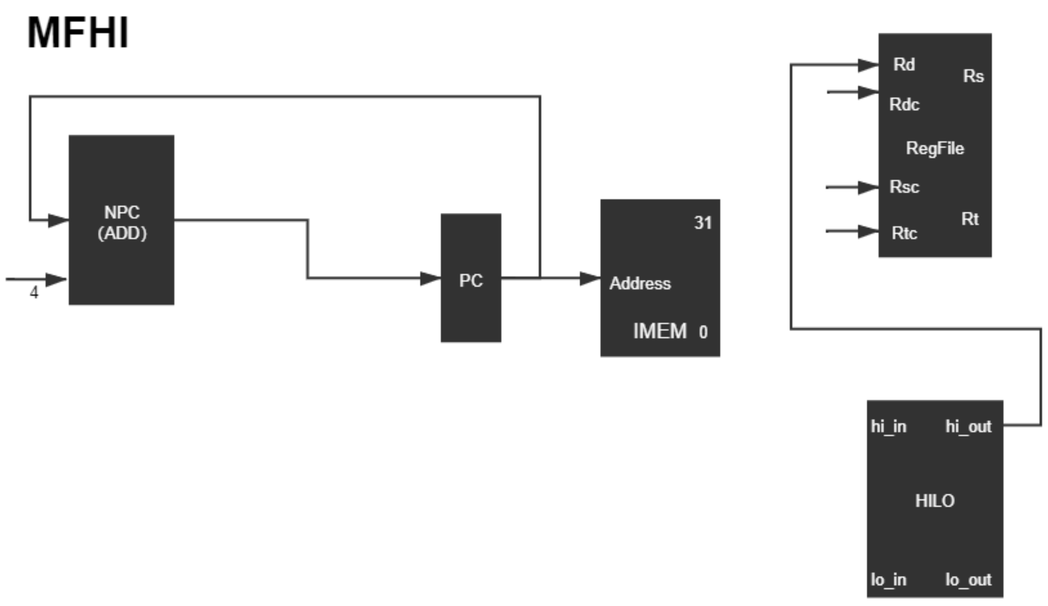
SH



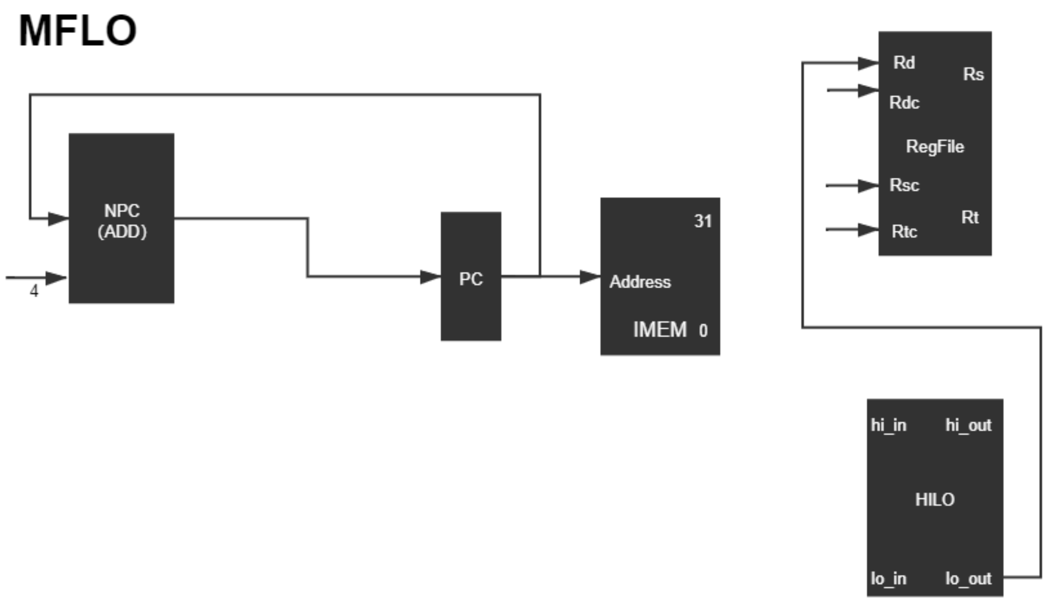
操作：SH rt,rs,imm ;addr←rs+imm,[addr]←rt ,PC←NPC(PC+4)
所需部件：PC、NPC、IMEM、Regfile、ALU、EXT16(U)、DMEM



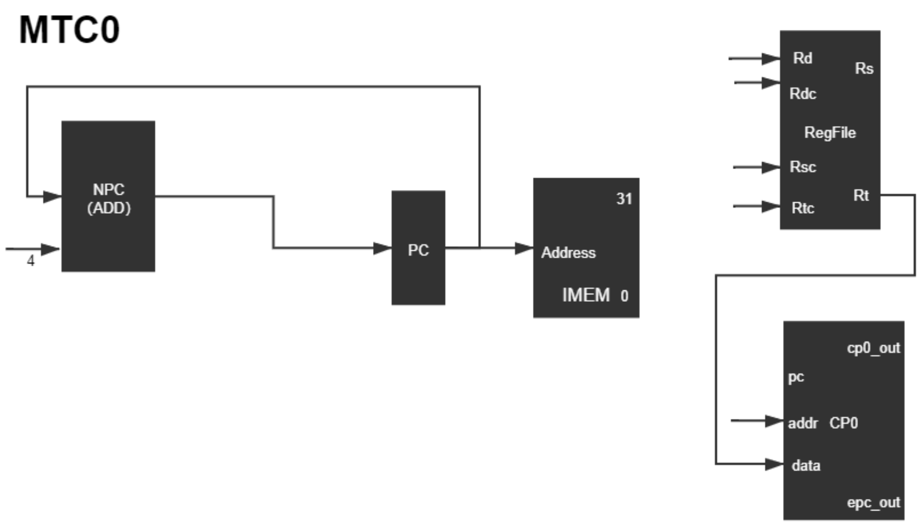
操作：MFC0 rt, rd;rd(Regfile)←rt(CP0),PC←NPC(PC+4)
所需部件：PC、NPC、Regfile、CP0



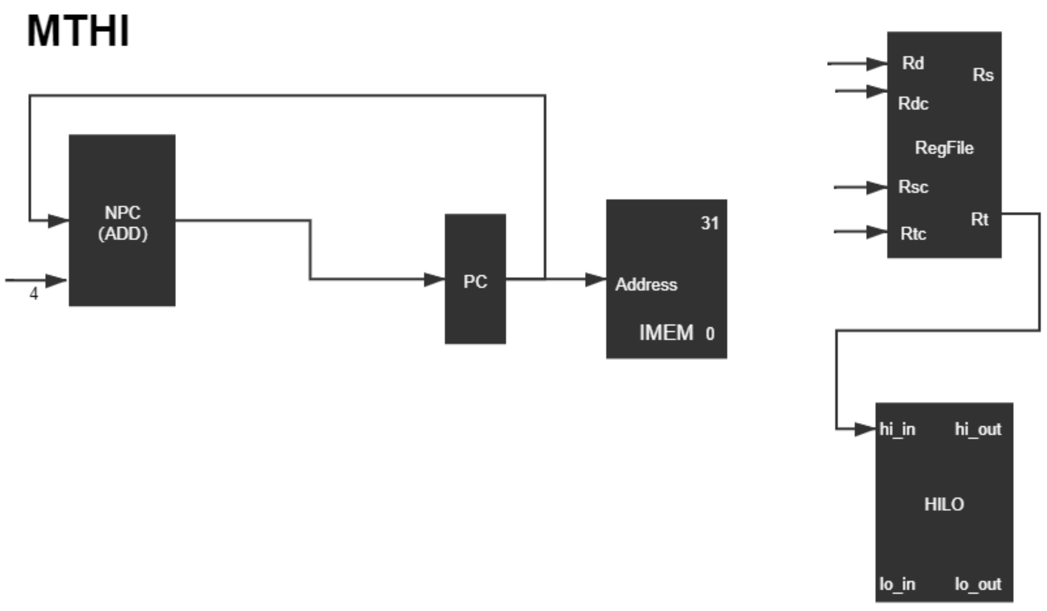
操作：MFHI rd;rd←hi,PC←NPC(PC+4)
所需部件：PC、NPC、Regfile、HILO



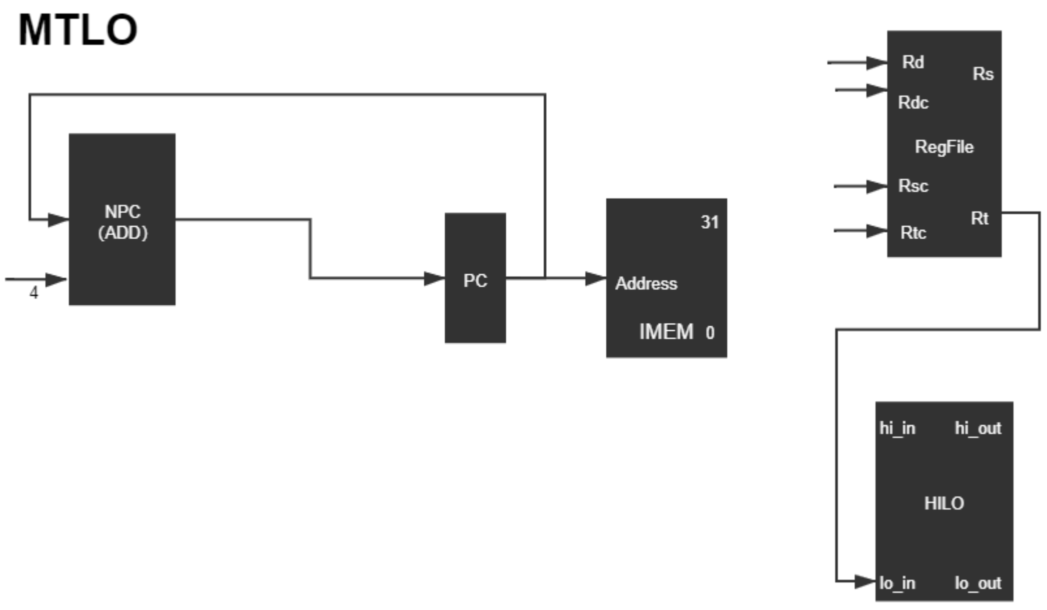
操作：MFHI rd;rd←lo,PC←NPC(PC+4)
所需部件：PC、NPC、Regfile、HILO



操作：MTC0 rt, rd;rd(CP0)←rt(Regfile),PC←NPC(PC+4)
所需部件：PC、NPC、Regfile、CP0

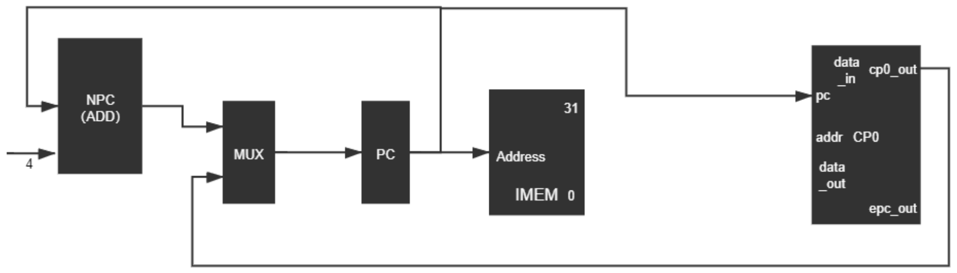


操作：MTHI rt;hi←rt,PC←NPC(PC+4)
所需部件：PC、NPC、Regfile、HILO



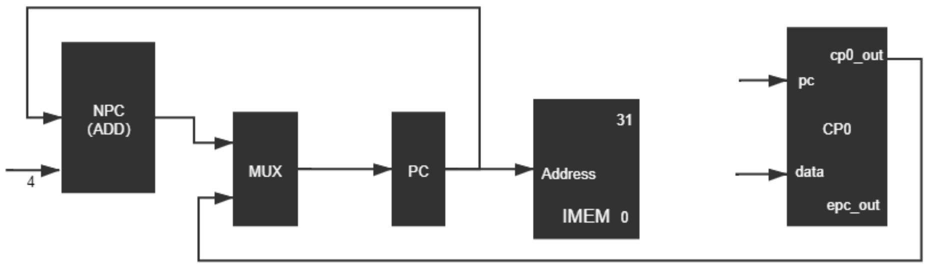
操作：MTLO rt;lo←rt,PC←NPC(PC+4)
所需部件：PC、NPC、Regfile、HILO

SYSCALL



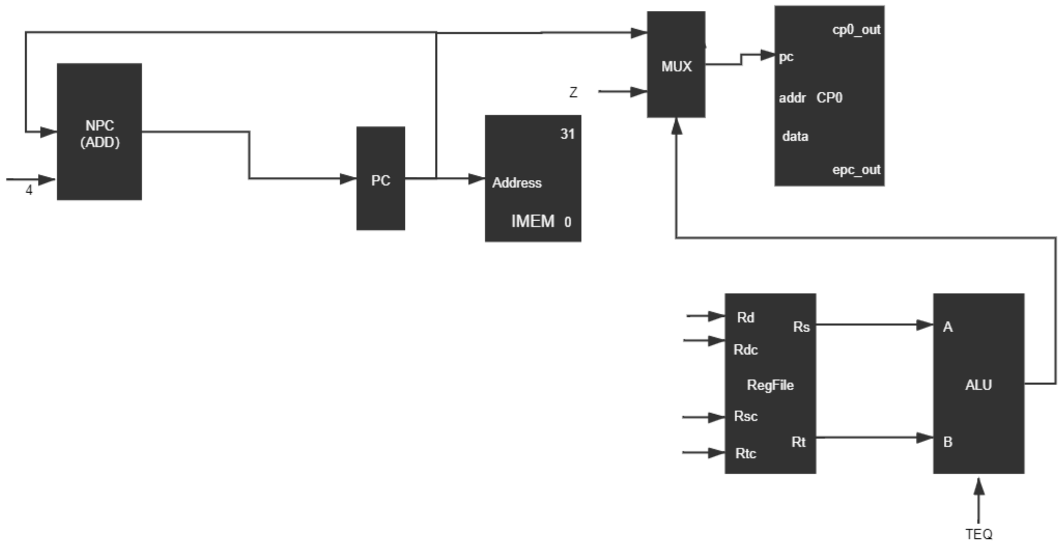
操作：SYSCALL;CP0_pc←PC+4, PC←cp0_out
所需部件：PC、NPC、CP0、MUX

ERET



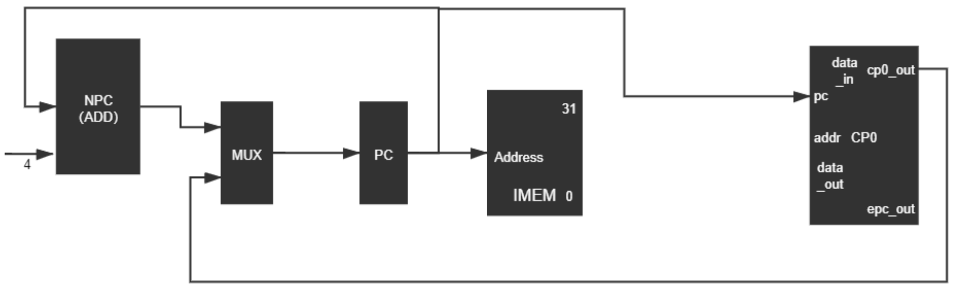
操作：ERET; PC←cp0_out
所需部件：PC、NPC、CP0、MUX

TEQ



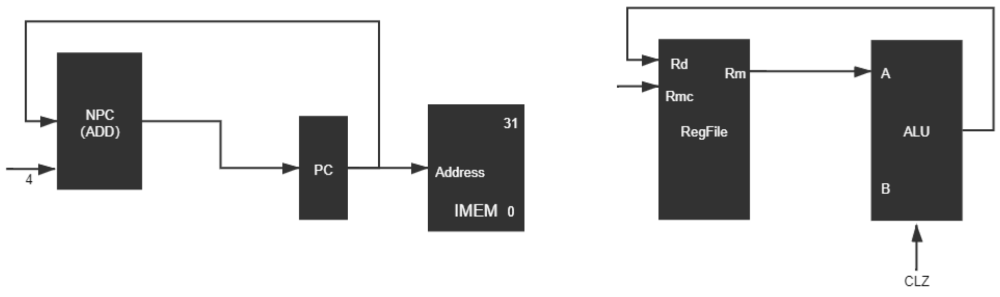
操作：TEQ rs,rt;A←rs,B←rt,TEQ
相等：CP0_pc←PC+4, PC←cp0_out
不相等：PC←PC+4
所需部件：PC、NPC、CP0、MUX、Regfile、ALU

BREAK



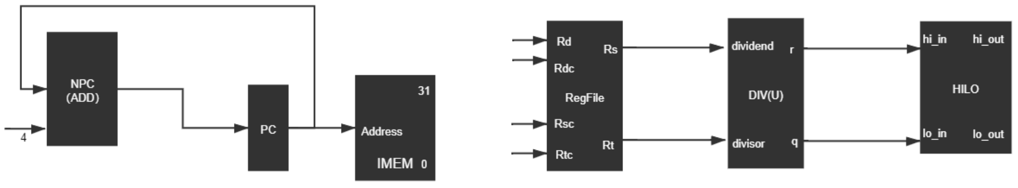
操作：BREAK;CP0_pc←PC+4, PC←cp0_out
所需部件：PC、NPC、CP0、MUX

CLZ



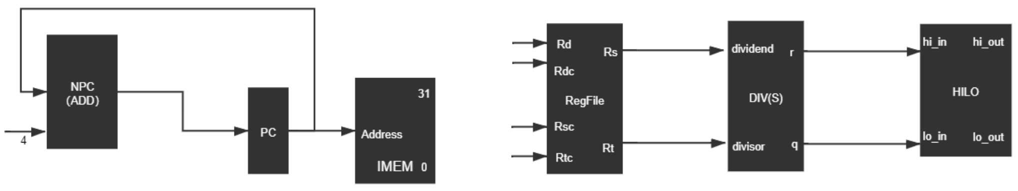
操作：CLZ rd,rs;rd←CLZ(rs),PC←NPC(PC+4)
所需部件：PC、NPC、IMEM、Regfile、ALU

DIVU



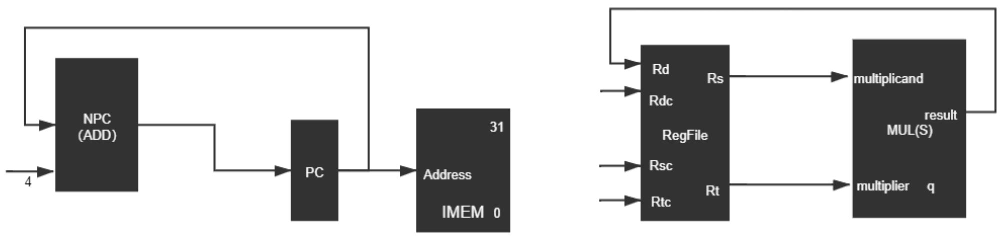
操作：DIVU rs,rt ;hi←rs/rt,lo←rs%rt,PC←NPC(PC+4)
所需部件：PC、NPC、IMEM、Regfile、DIV(U)、HILO

DIV



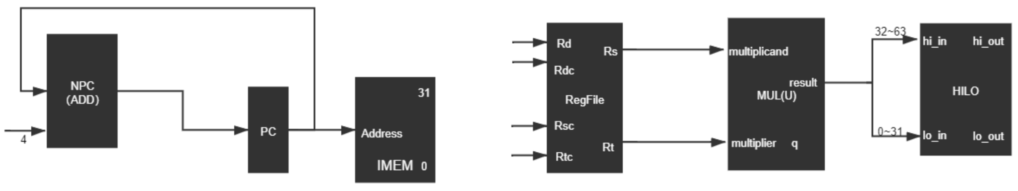
操作：DIVU rs,rt ;hi←rs/rt,lo←rs%rt,PC←NPC(PC+4)
所需部件：PC、NPC、IMEM、Regfile、DIV(S)、HILO

MUL



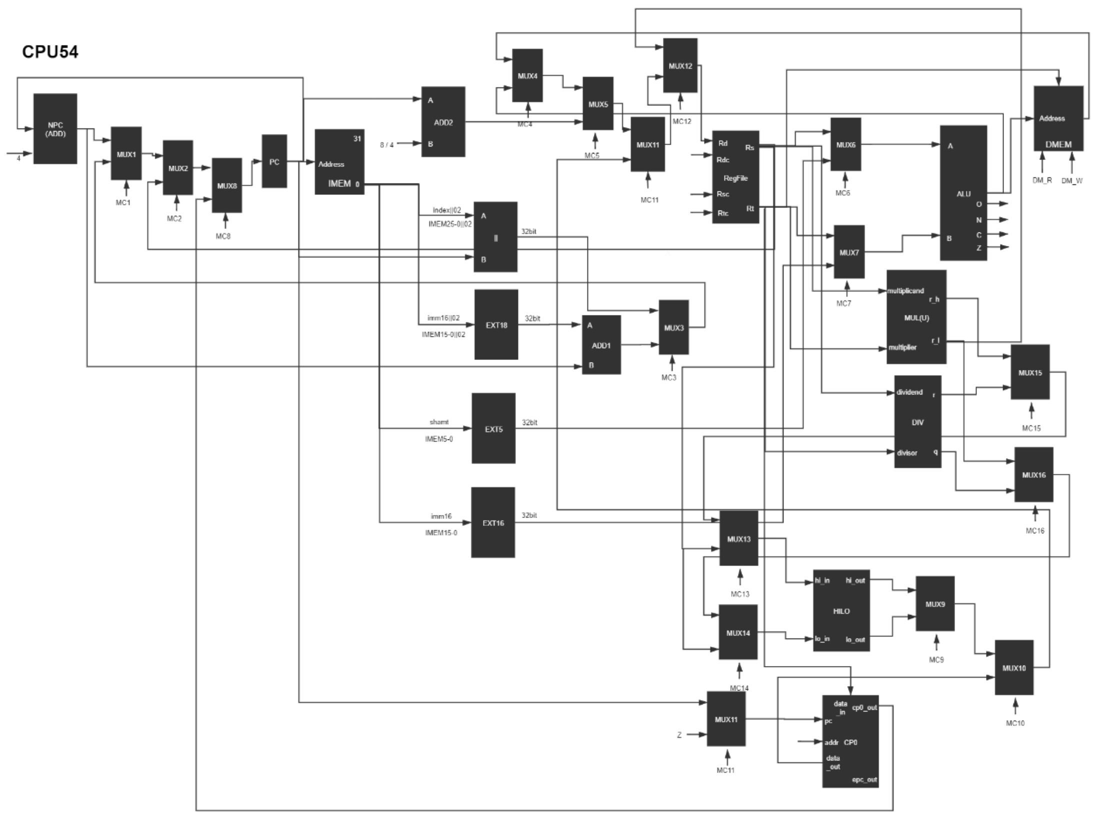
操作：MUL rd,rs,rt ;rd←rs*rt,PC←NPC(PC+4)
所需部件：PC、NPC、IMEM、Regfile、MUL(S)

MULTU



操作：MULTU rs,rt ;hi←rs*rt 高兴,lo←rs*rt 低位,PC←NPC(PC+4)
所需部件：PC、NPC、IMEM、Regfile、MUL(U)

总体数据通路图：

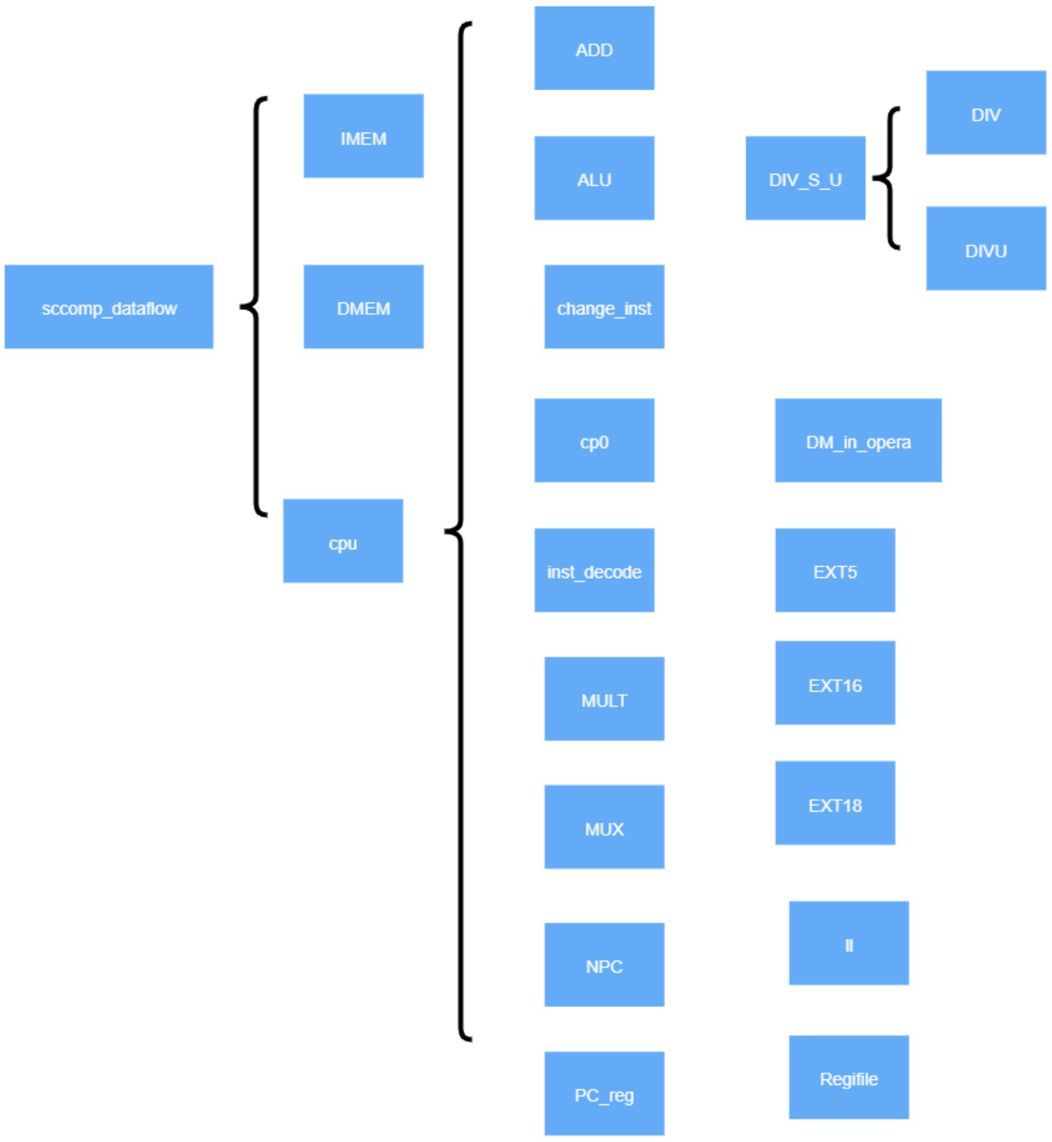


各指令所使用的部件及联系：

	PC	NPC	IMEM	RegFile	ALU		EXT5	EXT16(S)	EXT16(U)	DMEM		EXT18	ADD		||		DIV(U)		DIV(S)		HILO		CP0		MUL(S)		MUL(U)	
				Rd	A	B				Addr	Data		A	B	A	B	dividend	divisor	dividend	divisor	hi_in	lo_in	pc	data	multiplica	multiplier	multiplica	multiplier
ADD	NPC	PC	PC	ALU	Rs	Rt																						
ADDU	NPC	PC	PC	ALU	Rs	Rt																						
SUB	NPC	PC	PC	ALU	Rs	Rt																						
SUBU	NPC	PC	PC	ALU	Rs	Rt																						
AND	NPC	PC	PC	ALU	Rs	Rt																						
OR	NPC	PC	PC	ALU	Rs	Rt																						
XOR	NPC	PC	PC	ALU	Rs	Rt																						
NOR	NPC	PC	PC	ALU	Rs	Rt																						
SLT	NPC	PC	PC	ALU	Rs	Rt																						
SLTU	NPC	PC	PC	ALU	Rs	Rt																						
SLL	NPC	PC	PC	ALU	EXT5	Rt	shamt																					
SRL	NPC	PC	PC	ALU	EXT5	Rt	shamt																					
SRA	NPC	PC	PC	ALU	EXT5	Rt	shamt																					
SLLV	NPC	PC	PC	ALU	Rs	Rt																						
SRLV	NPC	PC	PC	ALU	Rs	Rt																						
SRAV	NPC	PC	PC	ALU	Rs	Rt																						
JR	Rs	PC	PC																									
ADDI	NPC	PC	PC	ALU	Rs	EXT16(S)		Imm16																				
ADDIU	NPC	PC	PC	ALU	Rs	EXT16(S)		Imm16																				
ANDI	NPC	PC	PC	ALU	Rs	EXT16(U)			Imm16																			
ORI	NPC	PC	PC	ALU	Rs	EXT16(U)			Imm16																			
XORI	NPC	PC	PC	ALU	Rs	EXT16(U)			Imm16																			
LW	NPC	PC	PC	DMEM	Rs	EXT16(U)			Imm16	ALU																		
SW	NPC	PC	PC		Rs	EXT16(U)			Imm16	ALU	Rd																	
BEQ	MUX?NPC	PC	PC		Rs	Rt						Imm16||02	EXT18	NPC														
BNE	MUX?NPC	PC	PC		Rs	Rt						Imm16||02	EXT18	NPC														
SLTI	NPC	PC	PC	ALU	Rs	EXT16(S)		Imm16																				
SLTIU	NPC	PC	PC	ALU	Rs	EXT16(S)		Imm16																				
LUI	NPC	PC	PC	ALU		EXT16(S)		Imm16																				
J	||	PC	PC												index||02	PC												
JAL	||	PC	PC	ADD									PC	4	index||02	PC												
CLZ	NPC	PC	PC	ALU	Rs																							
DIVU	NPC	PC	PC	ALU													Rs	Rt			DIV(U) r	DIV(U) q						
ERET	cp0_out	PC	PC																									
JALR	Rs	PC	PC	ADD									PC	4														
LB	NPC	PC	PC	DMEM	Rs	EXT16(U)			Imm16	ALU																		
LBU	NPC	PC	PC	DMEM	Rs	EXT16(U)			Imm16	ALU																		
LHU	NPC	PC	PC	DMEM	Rs	EXT16(U)			Imm16	ALU																		
SB	NPC	PC	PC		Rs	EXT16(U)			Imm16	ALU																		
SH	NPC	PC	PC		Rs	EXT16(U)			Imm16	ALU	Rd																	
LH	NPC	PC	PC	DMEM	Rs	EXT16(U)			Imm16	ALU	Rd																	
MFC0	NPC	PC	PC	CP0																								
MFHI	NPC	PC	PC	hi_out																								
MFLO	NPC	PC	PC	lo_out																								
MTC0	NPC	PC	PC																					Rt				
MTHI	NPC	PC	PC																		Rt							
MTLO	NPC	PC	PC																			Rt						
MUL	NPC	PC	PC	MUL(S)																					Rs	Rt		
MULTU	NPC	PC	PC																								Rs	Rt
SYSCALL	NPC	PC	PC																				PC					
TEQ	NPC	PC	PC		Rs	Rt																	MUX?PC,Z					
BGEZ	MUX?NPC	PC	PC		Rs							Imm16||02	EXT18	NPC														
BREAK	NPC	PC	PC																				PC					
DIV	NPC	PC	PC																Rs	Rt	DIV(S) r	DIV(S) q						

三、 模块建模

整体模块结构如下：



顶层模块：

```
`timescale 1ns / 1ps


module sccomp_dataflow(
    input clk_in,
    input reset,
```

```verilog
    output wire [31:0] inst,
    output wire [31:0] pc,
    //);

output wire [31:0]im_addr,
output wire mc1,
output wire mc2,
output wire mc3,
output wire mc4,
output wire mc5,
output wire mc6,
output wire mc7,
output wire mc8,
output wire mc9,
output wire mc10,
output wire mc11,
output wire mc12,
output wire mc13,
output wire mc14,
output wire mc15,
output wire mc16,
output wire [31:0]mux1_out,
output wire [31:0]mux2_out,

output wire [31:0]mux3_out,

output wire [31:0]mux4_out,

output wire [31:0]mux5_out,

output wire [31:0]mux6_out,

output wire [31:0]mux7_out,

output wire [31:0]mux8_out,

output wire [31:0]mux9_out,

output wire [31:0]mux10_out,

output wire [31:0]mux11_out,

output wire [31:0]mux12_out,

output wire [31:0]mux13_out,

output wire [31:0]mux14_out,

output wire [31:0]mux15_out,

output wire [31:0]mux16_out,

output wire [4:0]Rsc,

output wire [4:0]Rdc,

output wire [4:0]Rtc,

output wire rf_en,

output wire [31:0]ext16_out,

output wire DM_R,

output wire DM_W,
output wire [31:0]DM_addr,

output wire [31:0]DM_in,

output wire [31:0]DM_out,

output wire [5:0] inst_chose,

output wire [31:0]imm,

output wire cp0_R,

output wire cp0_W);

//assign pc=32'h1234;


assign im_addr=(pc-32'h0040_0000)>>2;

wire[31:0] a_dm_addr;

assign DM_addr = (a_dm_addr-32'h1001_0000)>>2;

IMEM_t imem(.a(im_addr),.spo(inst));

cpu sccpu(clk_in,reset,inst,DM_out,pc,DM_R,DM_W,a_

dm_addr,DM_in,

mc1,mc2,mc3,mc4,mc5,mc6,mc7,mc8,mc9,mc10,mc11,mc12

,mc13,mc14,mc15,mc16,

mux1_out,mux2_out,mux3_out,mux4_out,mux5_out,mux6_

out,mux7_out,mux8_out,mux9_out,mux10_out,mux11_out

,mux12_out,mux13_out,mux14_out,mux15_out,mux16_out

,

Rsc,Rdc,Rtc,rf_en,imm,inst_chose,cp0_R,cp0_W);
```

```
//*/
DMEM dmem(.clk(clk_in),.ena(1),.DM_R(DM_R),.DM_W(D
M_W),.addr(a_dm_addr),.d_addr(DM_addr),.DM_in(DM_i
n),.inst_chose(inst_chose),.DM_out(DM_out));
//*/
endmodule
```

CPU 模块:

```
`timescale 1ns / 1ps

module  cpu (
    input clk,
    input rst,
    input [31:0]inst,
    input [31:0]DM_out,
    output [31:0]pc,
    //output DM_chose,
    output wire DM_R,
    output wire DM_W,
    output [31:0]DM_addr,
    output wire [31:0]DM_in,
    output wire mc1,
    output wire mc2,
    output wire mc3,

    output wire mc4,

    output wire mc5,

    output wire mc6,

    output wire mc7,

    output wire mc8,

    output wire mc9,

    output wire mc10,

    output wire mc11,

    output wire mc12,

    output wire mc13,

    output wire mc14,

    output wire mc15,

    output wire mc16,

    output wire [31:0]mux1_out,

    output wire [31:0]mux2_out,

    output wire [31:0]mux3_out,

    output wire [31:0]mux4_out,

    output wire [31:0]mux5_out,

    output wire [31:0]mux6_out,

    output wire [31:0]mux7_out,

    output wire [31:0]mux8_out,
    output wire [31:0]mux9_out,
    output wire [31:0]mux10_out,
    output wire [31:0]mux11_out,
    output wire [31:0]mux12_out,
    output wire [31:0]mux13_out,
    output wire [31:0]mux14_out,
    output wire [31:0]mux15_out,
    output wire [31:0]mux16_out,
    output wire [4:0]Rsc,
    output wire [4:0]Rdc,
    output wire [4:0]Rtc,
    output wire rf_en,
    output wire [31:0]ext16_out,
    output wire [5:0] inst_chose,
    output wire cp0_R,
    output wire cp0_W
    );
wire [31:0] hi_in;
wire [31:0] lo_in;
wire ho_rst;
wire H_R,L_R;
wire H_W,L_W;
wire [31:0] hi_out;
wire [31:0] lo_out;


//wire cp0_R;
//wire cp0_W;
wire exp;
wire eret;
wire [4:0]cause;
wire [31:0]rdata;
wire [31:0]exc_addr;

wire mul_is_signed;
wire [31:0]r_h;
wire [31:0]r_l;
wire mul_work;

wire div_start;
wire div_is_signed;
wire [31:0]div_q;
wire [31:0]div_r;
wire div_busy;
```

```
//wire [4:0]inst_chose;

wire [31:0]pc_now;

wire [31:0]npc;

NPC anpc(pc_now,div_busy,npc);

//wire [31:0]mux1_out;

//wire [31:0]mux2_out;

//wire [31:0]mux3_out;


//wire mc1,mc2,mc3;

//wire mc4,mc5,mc6,mc7;

wire[31:0]ll_out;

wire[31:0]ext18_out;

wire[31:0]add1_out;


//wire [4:0]Rsc;

//wire [4:0]Rdc;

//wire [4:0]Rtc;

wire [4:0]shamt;

wire [15:0]imm;

instr_decode decode(.inst(inst),.inst_chose(inst_c

hose),.Rsc(Rsc),.Rdc(Rdc),.Rtc(Rtc),.shamt(shamt),

.imm(imm));
```

```
wire[31:0] alu_out;


//wire rf_en;
wire is_signed;
wire [31:0] Rs;
wire [31:0] cp0_out;
change_inst change(.inst_chose(inst_chose),.alu_re
s(alu_out[0]),.div_busy(div_busy),
.MC1(mc1),.MC2(mc2),.MC3(mc3),.MC4(mc4),.MC5(mc5),
.MC6(mc6),.MC7(mc7),.MC8(mc8),.MC9(mc9),.MC10(mc10
),.MC11(mc11),.MC12(mc12),.MC13(mc13),.MC14(mc14),
.MC15(mc15),.MC16(mc16),
.DM_R(DM_R),.DM_W(DM_W),.rf_en(rf_en),.is_signed(i
s_signed),.mul_is_signed(mul_is_signed),.mul_work(
mul_work),
.H_R(H_R),.L_R(L_R),.H_W(H_W),.L_W(L_W),.cp0_R(cp0
_R),.cp0_W(cp0_W),.exp(exp),.eret(eret),.cause(cau
se),
.div_start(div_start),.div_is_signed(div_is_signed
));
ll ll_(.inst(inst),.pc(pc_now),.ll_out(ll_out));
EXT18 ext18(.EXT_in(imm),.EXT18_out(ext18_out));
```

```
ADD add1(.A(ext18_out),.B(npc),.ADD_out(add1_out))
;
MUX mux3(.A(ll_out),.B(add1_out),.flag(mc3),.mux_o
ut(mux3_out));
MUX mux1(.A(npc),.B(mux3_out),.flag(mc1),.mux_out(
mux1_out));
MUX mux2(.A(mux1_out),.B(Rs),.flag(mc2),.mux_out(m
ux2_out));
MUX mux8(.A(mux2_out),.B(exc_addr),.flag(mc8),.mux
_out(mux8_out));


PC_reg pc_reg(.clk(clk),.rst(rst),.ena(1'b1),.data
_in(mux8_out),.data_out(pc_now));
//assign pc_now = rst?32'h0040_0000:npc;
wire[31:0]add2_out;
ADD add2(.A(pc_now),.B(4),.ADD_out(add2_out));//??
?8 还是 4�?????
wire [31:0]ext5_out;
EXT5 ext5(.EXT_in(shamt),.EXT5_out(ext5_out));
//wire [31:0]ext16_out;
EXT16 ext16(.is_signed(is_signed),.EXT_in(imm),.EX
T16_out(ext16_out));
```

```verilog
wire [31:0] Rd;
wire [31:0] Rt;
//wire [31:0]mux5_out;
regfile cpu_ref(.clk(clk),.ena(rf_en),.rst(rst),.Rdc(Rdc),.Rd(mux12_out),.Rsc(Rsc),.Rtc(Rtc),.Rs(Rs),.Rt(Rt));

//wire [31:0]mux6_out;
//wire [31:0]mux7_out;
MUX mux6(.A(Rs),.B(ext5_out),.flag(mc6),.mux_out(mux6_out));
MUX mux7(.A(Rt),.B(ext16_out),.flag(mc7),.mux_out(mux7_out));
ALU alu(.rst(rst),.a(mux6_out),.b(mux7_out),.aluc(inst_chose),.r(alu_out),.zero(zero),.carry(carry),.negative(negative),.overflow(overflow));

//wire [31:0]mux4_out;
MUX mux4(.A(DM_out),.B(alu_out),.flag(mc4),.mux_out(mux4_out));
MUX mux5(.A(mux4_out),.B(add2_out),.flag(mc5),.mux_out(mux5_out));

HILO ho(.hi_in(mux13_out),.lo_in(mux14_out),.rst(rst),.ena(1),.H_R(H_R),.L_R(L_R),.H_W(H_W),.L_W(L_W),.hi_out(hi_out),.lo_out(lo_out));
MUX mux9(.A(hi_out),.B(lo_out),.flag(mc9),.mux_out(mux9_out));
MUX mux10(.A(mux9_out),.B(rdata),.flag(mc10),.mux_out(mux10_out));
MUX mux11(.A(mux5_out),.B(mux10_out),.flag(mc11),.mux_out(mux11_out));
MUX mux12(.A(r_l),.B(mux11_out),.flag(mc12),.mux_out(mux12_out));
MUX mux13(.A(mux15_out),.B(Rs),.flag(mc13),.mux_out(mux13_out));
MUX mux14(.A(mux16_out),.B(Rs),.flag(mc14),.mux_out(mux14_out));

wire [31:0]status;
wire intr;
wire set;
wire timer_int;
cp0 CP_0(.clk(clk),.rst(rst),.mfc0(cp0_R),.mtc0(cp
0_W),.pc(pc_now),.Rd(Rsc),.wdata(Rt),.expection(ex
p),
.eret(eret),.cause(cause),.intr(intr),.rdata(rdata
),.status(status),.timer_int(timer_int),.exc_addr(
exc_addr));

MULT_ mult(.clk(clk),.reset(rst),.mul_is_signed(mu
l_is_signed),.mul_work(mul_work),.a(Rs),.b(Rt),.z_
h(r_h),.z_l(r_l));

MUX mux15(.A(r_h),.B(div_r),.flag(mc15),.mux_out(m
ux15_out));
MUX mux16(.A(r_l),.B(div_q),.flag(mc16),.mux_out(m
ux16_out));
DM_in_opera in_opera(.inst_chose(inst_chose),.Rt(R
t),.alu_out(alu_out),.DM_in(DM_in));
DIV_ div_(.dividend(Rs),.divisor(Rt),.start(div_st
art),.clock(clk),.reset(rst),
.div_is_signed(div_is_signed),.q(div_q),.r(div_r),
.busy(div_busy));
```

```
assign pc=pc_now;
assign DM_addr = (DM_R||DM_W)?alu_out:32'hz ;
endmodule
```

寄存器模块

```
module regfile (
    input clk,
    input ena,
    input rst,
    input [4:0] Rdc,
    input [31:0] Rd,
    input [4:0] Rsc,
    input [4:0] Rtc,
    output  [31:0] Rs,
    output [31:0] Rt
);
    reg [31:0] array_reg[31:0];
    assign Rs = array_reg[Rsc];
    assign Rt = array_reg[Rtc];
```

```verilog
always @(negedge clk or posedge rst)
begin
    if(rst)
    begin
        array_reg[0]  <= 32'b0;
        array_reg[1]  <= 32'b0;
        array_reg[2]  <= 32'b0;
        array_reg[3]  <= 32'b0;
        array_reg[4]  <= 32'b0;
        array_reg[5]  <= 32'b0;
        array_reg[6]  <= 32'b0;
        array_reg[7]  <= 32'b0;
        array_reg[8]  <= 32'b0;
        array_reg[9]  <= 32'b0;
        array_reg[10] <= 32'b0;
        array_reg[11] <= 32'b0;
        array_reg[12] <= 32'b0;
        array_reg[13] <= 32'b0;
        array_reg[14] <= 32'b0;
        array_reg[15] <= 32'b0;
        array_reg[16] <= 32'b0;
        array_reg[17] <= 32'b0;
```

```verilog
            array_reg[18] <= 32'b0;
            array_reg[19] <= 32'b0;
            array_reg[20] <= 32'b0;
            array_reg[21] <= 32'b0;
            array_reg[22] <= 32'b0;
            array_reg[23] <= 32'b0;
            array_reg[24] <= 32'b0;
            array_reg[25] <= 32'b0;
            array_reg[26] <= 32'b0;
            array_reg[27] <= 32'b0;
            array_reg[28] <= 32'b0;
            array_reg[29] <= 32'b0;
            array_reg[30] <= 32'b0;
            array_reg[31] <= 32'b0;
        end
        else if(ena && Rdc != 5'b0)
        begin
            array_reg[Rdc]<=Rd;
        end
    end
endmodule
```

ALU 模块：

```
`timescale 1ns / 1ps


module ALU(
    input rst,
    input [31:0] a,
    input [31:0] b,
    input [5:0] aluc,
    output reg [31:0] r,
    output reg zero,
    output  reg carry,
    output reg negative,
    output  reg overflow
);

```

```
    parameter ADD =0 ;
    parameter  ADDU=1 ;
    parameter  SUB=2 ;
    parameter  SUBU=3 ;
    parameter  AND=4 ;
    parameter  OR=5 ;
    parameter  XOR=6 ;
    parameter  NOR=7 ;
    parameter  SLT=8 ;
    parameter  SLTU=9 ;
    parameter  SLL=10 ;
    parameter  SRL=11 ;
    parameter  SRA=12 ;
    parameter  SLLV=13 ;
    parameter  SRLV=14 ;
    parameter  SRAV=15 ;
    parameter  JR= 16;
    parameter  ADDI=17 ;
    parameter  ADDIU=18 ;
    parameter  ANDI=19 ;
    parameter  ORI=20 ;
    parameter  XORI=21 ;
    parameter  LW=22 ;
    parameter  SW=23 ;
    parameter  BEQ=24 ;
    parameter  BNE=25 ;
    parameter  SLTI=26 ;
    parameter  SLTIU=27 ;
    parameter  LUI=28 ;
    parameter  J=29 ;
    parameter  JAL=30 ;
    parameter   JALR=31;//跳转到寄存器的值 pc 加 4 放
入 31 号寄存器
    parameter   BGEZ=32;
    parameter   LB=33;//取字节
    parameter   LBU=34;//取字节无
    parameter   LHU=35;//取半字（无符号
    parameter   LH=36;//有
    parameter   SB=37;//存字节
    parameter   SH=38;//存半字
    parameter   MFC0=39;
    parameter   MFHI=40;
    parameter   MFLO=41;
    parameter   MTC0=42;
    parameter   MTHI=43;
    parameter   MTLO=44;
    parameter   SYSCALL=45;
    parameter   ERET=46;//中断 cp0 statu cause
    parameter   TEQ=47;
    parameter   BREAK=48;
    parameter   CLZ=49;//ALU
    parameter   DIVU=50;//divu
    parameter   DIV=51;//div
    parameter   MULT=52;
    parameter   MULTU=53;
reg [35:0]t;

always@(*)
begin
    if(rst)
    begin
        zero=0;
        carry=0;
        negative=0;
        overflow=0;
        r=32'hz;
    end

    else

    begin

        casex(aluc)

        ADDU , ADDIU:

        begin

            t<=a+b;

            r<=a+b;

            if(t>36'h0ffffffff)

                carry<=1;

            else

                carry<=0;

            if(r==0)

                zero=1;

            else

                zero=0;

            if(r[31]==1)

                negative=1;

            else

                negative=0;

        end

        ADD,ADDI,SW,LW,LB,LBU,LH,LHU,SB,SH:
```

```
    begin
        r=$signed(a)+$signed(b);
        if(a[31]==0&&b[31]==0)
        begin
            t=a+b;
            if(t[31]==1)
                overflow=1;
            else
                overflow=0;
        end
        if(a[31]==1&&b[31]==1)
        begin
            t=a+b;
            if(t[31]==0)
                overflow=1;
            else
                overflow=0;
        end
        if(a[31]+b[31]==1)
            overflow=0;
        if(r==0)
            zero=1;
        else
            zero=0;
        if(r[31]==1)
            negative=1;
        else
            negative=0;
    end
    SUBU:
    begin
        r=a-b;
        if(a>=b)
            carry=0;
        else
            carry=1;
        if(r==0)
            zero=1;
        else
            zero=0;
        if(r[31]==1)
            negative=1;
        else
            negative=0;
```

```
end
SUB:
begin
r=$signed(a)-$signed(b);
t=a-b;
if(a[31]==0&&b[31]==1)
begin
    if(t[31]==1)
        overflow<=1;
    else
        overflow<=0;
end
if(a[31]==1&&b[31]==0)
begin
    if(t[31]==0)
        overflow<=1;
    else
        overflow<=0;
end
if(!(a[31]^b[31]))
    overflow=0;
if(r==0)
```

```
    zero<=1;
else
    zero<=0;
if(r[31]==1)
    negative<=1;
else
    negative<=0;
end
AND,ANDI:
begin
    r=a&b;
if(r==0)
    zero<=1;
else
    zero<=0;
if(r[31]==1)
    negative<=1;
else
    negative<=0;
end
OR,ORI:
begin
```

```
    r=a|b;
if(r==0)
    zero<=1;
else
    zero<=0;
if(r[31]==1)
    negative<=1;
else
    negative<=0;
end
XOR,XORI:
begin
r<=a^b;
if(r==0)
    zero<=1;
else
    zero<=0;
if(r[31]==1)
    negative<=1;
else
    negative<=0;
end
```

```
NOR:
begin
r<=~(a|b);
if(r==0)
    zero<=1;
else
    zero<=0;
if(r[31]==1)
    negative<=1;
else
    negative<=0;
end
LUI:
begin
r<={b[15:0],16'b0};
if(r==0)
    zero<=1;
else
    zero<=0;
if(r[31]==1)
    negative<=1;
else
```

```
    negative<=0;
end
//SLT
SLT,SLTI:
begin
r=($signed(a)<$signed(b))?1:0;
zero=(a==b)?1:0;
negative=(r==1)?1:0;
end
//SLTU
SLTU,SLTIU:
begin
r=(a<b)?1:0;
zero=(a==b)?1:0;
carry=(r==1)?1:0;
negative=r[31];
end
SRA,SRAV:
begin
r=$signed(b)>>>a;
zero=(r==0)?1:0;
negative=r[31];
```

```
carry=(a>0&&a<33)?b[a-1]:b[31];

end

SLL,SLLV:

begin

r=b<<a;

zero=(r==0)?1:0;

carry=(a>0&&a<33)?b[32-a]:0;

negative=r[31];

end

SRL,SRLV:

begin

r=b>>a;

zero=(r==0)?1:0;

carry=(a>0&&a<33)?b[a-1]:0;

negative=r[31];

end

BEQ,BNE,TEQ:

begin

if(a==b)

    r<=1;

else

    r<=0;
```

```
            zero<=1;
        end
        BGEZ:
        begin
            if(a[31]==0)
            begin
                r<=1;
            end
            else begin
                r<=0 ;
            end
        end
        CLZ:
        begin
            if(a[31]==1)
                r<=0;
            else if(a[30]==1)
                r<=1;
            else if(a[29]==1)
                r<=2;
            else if(a[28]==1)
                r<=3;
            else if(a[27]==1)
                r<=4;
            else if(a[26]==1)
                r<=5;
            else if(a[25]==1)
                r<=6;
            else if(a[24]==1)
                r<=7;
            else if(a[23]==1)
                r<=8;
            else if(a[22]==1)
                r<=9;
            else if(a[21]==1)
                r<=10;
            else if(a[20]==1)
                r<=11;
            else if(a[19]==1)
                r<=12;
            else if(a[18]==1)
                r<=13;
            else if(a[17]==1)
                r<=14;
            else if(a[16]==1)
                r<=15;
            else if(a[15]==1)
                r<=16;
            else if(a[14]==1)
                r<=17;
            else if(a[13]==1)
                r<=18;
            else if(a[12]==1)
                r<=19;
            else if(a[11]==1)
                r<=20;
            else if(a[10]==1)
                r<=21;
            else if(a[9]==1)
                r<=22;
            else if(a[8]==1)
                r<=23;
            else if(a[7]==1)
                r<=24;
            else if(a[6]==1)
                r<=25;
            else if(a[5]==1)
                r<=26;
            else if(a[4]==1)
                r<=27;
            else if(a[3]==1)
                r<=28;
            else if(a[2]==1)
                r<=29;
            else if(a[1]==1)
                r<=30;
            else if(a[0]==1)
                r<=31;
            else
                r<=32;
        end
        default:
            r=32'hz;
        endcase
    end
end
endmodule
```

根据指令提取信息模块:

```
`timescale 1ns / 1ps


module instr_decode (

    input [31:0]inst,

    output reg [5:0]inst_chose,

    output reg [4:0]Rsc,

    output reg [4:0]Rdc,

    output reg [4:0]Rtc,

    output reg[4:0]shamt,

    output reg [15:0]imm

);

    parameter ADD =0 ;

    parameter  ADDU=1 ;

    parameter  SUB=2 ;

    parameter  SUBU=3 ;

    parameter  AND=4 ;

    parameter  OR=5 ;

    parameter  XOR=6 ;

    parameter  NOR=7 ;

    parameter  SLT=8 ;
    parameter  SLTU=9 ;
    parameter  SLL=10 ;
    parameter  SRL=11 ;
    parameter  SRA=12 ;
    parameter  SLLV=13 ;
    parameter  SRLV=14 ;
    parameter  SRAV=15 ;
    parameter  JR= 16;
    parameter  ADDI=17 ;
    parameter  ADDIU=18 ;
    parameter  ANDI=19 ;
    parameter  ORI=20 ;
    parameter  XORI=21 ;
    parameter  LW=22 ;
    parameter  SW=23 ;
    parameter  BEQ=24 ;
    parameter  BNE=25 ;
    parameter  SLTI=26 ;
    parameter  SLTIU=27 ;
    parameter  LUI=28 ;
    parameter  J=29 ;
    parameter  JAL=30 ;
    parameter   JALR=31;//跳转到寄存器的
�?? pc�??4 放入 31 号寄存器
    parameter   BGEZ=32;
    parameter   LB=33;//取字�??
    parameter   LBU=34;//取字节无
    parameter   LHU=35;//取半字（无符�??
    parameter   LH=36;//�??
    parameter   SB=37;//存字�??
    parameter   SH=38;//存半�??
    parameter   MFC0=39;
    parameter   MFHI=40;
    parameter   MFLO=41;
    parameter   MTC0=42;
    parameter   MTHI=43;
    parameter   MTLO=44;
    parameter   SYSCALL=45;
    parameter   ERET=46;//中断 cp0 statu cause
    parameter   TEQ=47;
    parameter   BREAK=48;
    parameter   CLZ=49;//ALU
    parameter   DIVU=50;//divu
    parameter   DIV=51;//div
    parameter   MUL=52;

    parameter   MULTU=53;

always @(*)

begin

    if(inst[31:26]==6'b0000_00)

    begin

        case(inst[5:0])

        6'b1000_00:inst_chose=0;

        6'b1000_01:inst_chose=1;

        6'b1000_10:inst_chose=2;

        6'b1000_11:inst_chose=3;

        6'b1001_00:inst_chose=4;

        6'b1001_01:inst_chose=5;

        6'b1001_10:inst_chose=6;

        6'b1001_11:inst_chose=7;

        6'b1010_10:inst_chose=8;

        6'b1010_11:inst_chose=9;

        6'b0000_00:inst_chose=10;

        6'b0000_10:inst_chose=11;

        6'b0000_11:inst_chose=12;

        6'b0001_00:inst_chose=13;

        6'b0001_10:inst_chose=14;
        6'b0001_11:inst_chose=15;

        6'b0010_00:inst_chose=16;

        6'b0010_01:inst_chose=31;

        6'b0100_00:inst_chose=MFHI;

        6'b0100_10:inst_chose=MFLO;

        6'b0100_01:inst_chose=MTHI;

        6'b0100_11:inst_chose=MTLO;

        6'b0011_00:inst_chose=SYSCALL;

        6'b1101_00:inst_chose=TEQ;

        6'b0011_01:inst_chose=BREAK;

        6'b0110_01:inst_chose=MULTU;

        6'b0110_11:inst_chose=DIVU;

        6'b0110_10:inst_chose=DIV;

        default:inst_chose=inst[31:26];

        endcase

    end

    else

    begin

        if(inst[31:21]==11'b0100_0000_000)

            inst_chose=MFC0;

        else if(inst[31:21]==11'b0100_0000_100)

            inst_chose=MTC0;
        else if(inst[31:21]==11'b0100_0010_000)
            inst_chose=ERET;
        else if(inst[31:26]==6'b0111_00&&inst[5:0]
==6'b1000_00)
            inst_chose=CLZ;
        else
        begin
            case (inst[31:26])
                6'b0010_00:inst_chose=17;
                6'b0010_01:inst_chose=18;
                6'b0011_00:inst_chose=19;
                6'b0011_01:inst_chose=20;
                6'b0011_10:inst_chose=21;
                6'b1000_11:inst_chose=22;
                6'b1010_11:inst_chose=23;
                6'b0001_00:inst_chose=24;
                6'b0001_01:inst_chose=25;
                6'b0010_10:inst_chose=26;
                6'b0010_11:inst_chose=27;
                6'b0011_11:inst_chose=28;
                6'b0000_10:inst_chose=29;
                6'b0000_11:inst_chose=30;
                6'b0000_01:inst_chose=BGEZ;
                6'b1000_00:inst_chose=LB;
                6'b1001_00:inst_chose=LBU;
                6'b1001_01:inst_chose=LHU;
                6'b1010_00:inst_chose=SB;
                6'b1010_01:inst_chose=SH;
                6'b1000_01:inst_chose=LH;
                6'b0111_00:inst_chose=MUL;
                default:inst_chose=5'bz;
            endcase
        end
    end

    if(inst_chose==MTC0||inst_chose==MFC0)
        Rsc<=inst[15:11];
    else if (inst_chose==BGEZ||(inst_chose>=LB&&inst_chose<=LH)||(inst_chose==MTHI||inst_chose==MTLO||inst_chose==TEQ)||inst_chose==CLZ||inst_chose==MUL||inst_chose==MULTU||inst_chose==DIV||inst_chose==DIVU)
    begin
        Rsc<=inst[25:21];
    end
    else if (((inst[31:26]!=6'b0000_00 && inst[31:26]!=6'b0000_10  && inst[31:26]!=6'b0000_11  && inst[31:26]!=6'b0011_11) || (inst[31:26]==6'b0000_00 && inst[5:0]!=6'b0000_00 &&inst[5:0]!=6'b0000_10 && inst[5:0]!=6'b0000_11))||inst_chose==JALR)
    begin
        Rsc<=inst[25:21];
    end
    else
    begin
        Rsc<=6'bz;
    end

    if(inst_chose==SB||inst_chose==SH||inst_chose==MTC0||inst_chose==TEQ||inst_chose==MUL||inst_chose==MULTU||inst_chose==DIV||inst_chose==DIVU)
        Rtc<=inst[20:16];
    else if ((inst[31:26]==6'b0000_00 && inst[5:0]!=6'b0010_00)||inst[31:26]==6'b0001_00||inst[31:26]==6'b0001_01||inst[31:26]==6'b1010_11)
    begin
```

```
        Rtc<=inst[20:16];
    end
    else
    begin
        Rtc<=6'bz;
    end

    if(inst_chose==JALR)
    begin
        Rdc<=31;
    end
    else if(inst_chose==CLZ||inst_chose==MUL)
        Rdc<=inst[15:11];
    else if(inst_chose>=LB&&inst_chose<=LH||(inst_
chose==MFC0))
    begin
        Rdc<=inst[20:16];
    end
    else
    begin
        if((inst[31:26]==6'b0000_00 && inst[5:0]!=
6'b0010_00)||(inst_chose==MFHI||inst_chose==MFLO))
```

```
        begin
            //Rdc<=inst[20:16];
            Rdc<=inst[15:11];
        end
        else if(inst[31:26]!=6'b0000_00&&inst[31:26]!=6'b0000_10&&inst[31:26]!=6'b0000_11&&inst[31:26]!=6'b1010_11&&inst_chose!=SB&&inst_chose!=SH)
        begin
            Rdc<=inst[20:16];
        end
        else if (inst[31:26]==6'b0000_11)
        begin
            Rdc=31;
        end
        else
        begin
            Rdc<=6'bz;
        end
    end
    if(inst[31:26]==6'b0000_00 && (inst[5:0]==6'b0000_00||inst[5:0]==6'b0000_10||inst[5:0]==6'b0000_11))
    begin
        shamt<=inst[10:6];
    end
    else begin
        shamt <= 5'bz;
    end

    if ((inst[31:26]!=6'b0000_10  && inst[31:26]!=6'b0000_11  && inst[31:26]!=6'b0000_00 )||(inst_chose>=LB && inst_chose<=SH))
    begin
        imm<=inst[15:0];
    end
    else
    begin
        imm<=16'bz;
    end

end
```

```
endmodule
```

根据指令进行信号赋值的模块:

```
`timescale 1ns / 1ps


module change_inst (

    input[5:0] inst_chose,

    input alu_res,

    input div_busy,

    output  MC1,

    output  MC2,

    output  MC3,

    output  MC4,

    output  MC5,

    output  MC6,

    output  MC7,

    output  MC8,

    output  MC9,

    output  MC10,

    output  MC11,

    output  MC12,

    output  MC13,

    output  MC14,
    output  MC15,
    output  MC16,
    output  DM_R,
    output  DM_W,
    output  rf_en,
    output is_signed,
    output mul_is_signed,
    output mul_work,
    output  H_R,
    output  L_R,
    output  H_W,
    output  L_W,
    output  cp0_R,
    output  cp0_W,
    output  exp,
    output  eret,
    output reg [4:0]cause,
    output div_start,
    output div_is_signed
);
    parameter ADD =0 ;
    parameter  ADDU=1 ;
    parameter  SUB=2 ;
    parameter  SUBU=3 ;
    parameter  AND=4 ;
    parameter  OR=5 ;
    parameter  XOR=6 ;
    parameter  NOR=7 ;
    parameter  SLT=8 ;
    parameter  SLTU=9 ;
    parameter  SLL=10 ;
    parameter  SRL=11 ;
    parameter  SRA=12 ;
    parameter  SLLV=13 ;
    parameter  SRLV=14 ;
    parameter  SRAV=15 ;
    parameter  JR= 16;
    parameter  ADDI=17 ;
    parameter  ADDIU=18 ;
    parameter  ANDI=19 ;
    parameter  ORI=20 ;
    parameter  XORI=21 ;
    parameter  LW=22 ;
    parameter  SW=23 ;
    parameter  BEQ=24 ;
    parameter  BNE=25 ;
    parameter  SLTI=26 ;
    parameter  SLTIU=27 ;
    parameter  LUI=28 ;
    parameter  J=29 ;
    parameter  JAL=30 ;
    parameter   JALR=31;//跳转到寄存器的值 pc 加 4 放
入 31 号寄存器
    parameter   BGEZ=32;
    parameter   LB=33;//取字节
    parameter   LBU=34;//取字节无
    parameter   LHU=35;//取半字（无符号
    parameter   LH=36;//有
    parameter   SB=37;//存字节
    parameter   SH=38;//存半字
    parameter   MFC0=39;
    parameter   MFHI=40;
    parameter   MFLO=41;
    parameter   MTC0=42;
    parameter   MTHI=43;
    parameter   MTLO=44;
    parameter   SYSCALL=45;
    parameter   ERET=46;//中断 cp0 statu cause
    parameter   TEQ=47;
    parameter   BREAK=48;
    parameter   CLZ=49;//ALU
    parameter   DIVU=50;//divu
    parameter   DIV=51;//div
    parameter   MUL=52;
    parameter   MULTU=53;
    assign MC1 =((inst_chose>=ADD && inst_chose<=S
RAV) || (inst_chose>=ADDI && inst_chose<=SW) ||
    (inst_chose>=SLTI && inst_chose<=LUI) ||(inst_
chose==BEQ && !alu_res))||(inst_chose==BNE && alu_
res)||
    (inst_chose==BGEZ && !alu_res)||(inst_chose>=L
B);
    assign MC2 =((inst_chose>=ADD && inst_chose<=S
RAV) || (inst_chose>=ADDI && inst_chose<=SW) ||
    inst_chose==BEQ || inst_chose==BNE ||(inst_cho
se>=SLTI && inst_chose<=LUI)||inst_chose==J ||
    inst_chose==JAL||inst_chose>=BGEZ) ;
    assign MC3 = ((inst_chose==BEQ && !alu_res))||
((inst_chose==BNE && alu_res))||inst_chose==J
    ||inst_chose==JAL||(inst_chose==BGEZ && !alu_r
es);
    assign MC4 = (inst_chose==LW||(inst_chose>=LB&
&inst_chose<=LH));
    assign MC5 =((inst_chose>=ADD && inst_chose<=S
RAV)||(inst_chose>=ADDI && inst_chose<=XORI)|| ins
t_chose==LW||(inst_chose>=SLTI && inst_chose<=LUI)
||(inst_chose>=LB&&inst_chose<=LH)||inst_chose==CL
Z) ;
    assign MC6 =(inst_chose>=ADD && inst_chose<=SL
TU)|| (inst_chose>=SLLV && inst_chose<=SRAV)||(ins
t_chose>=ADDI && inst_chose<=SLTIU)||(inst_chose==
BGEZ||(inst_chose>=LB && inst_chose<=SH)||inst_cho
se==TEQ||inst_chose==CLZ);
    assign MC7 = (inst_chose>=ADD && inst_chose<=S
RAV)||inst_chose==BEQ||inst_chose==BNE||inst_chose
==TEQ;
    assign MC8 =!(inst_chose==ERET||inst_chose==SY
SCALL||inst_chose==BREAK||(inst_chose==TEQ&&alu_re
s)) ;
```

```
    assign MC9 = (inst_chose==MFHI) ;
    assign MC10 = (inst_chose==MFHI||inst_chose==M
FLO);
    assign MC11 = !(inst_chose==MFHI||inst_chose==
MFLO||inst_chose==MFC0);
    assign MC12 = (inst_chose==MUL);//!(inst_chose
==MFHI||inst_chose==MFLO);
    assign MC13 = !(inst_chose==MTHI);
    assign MC14 = !(inst_chose==MTLO);
    assign MC15 = (inst_chose==MUL||inst_chose==MU
LTU);
    assign MC16 = (inst_chose==MUL||inst_chose==MU
LTU);
    assign DM_R = (inst_chose==LW||inst_chose==LB|
|inst_chose==LH||inst_chose==LHU||inst_chose==LBU)
;
    assign DM_W = (inst_chose==SW||inst_chose==SB|
|inst_chose==SH);
    assign rf_en =((inst_chose>=ADD && inst_chose<
=SRAV) || (inst_chose>=ADDI && inst_chose<=LW)||
    (inst_chose>=SLTI && inst_chose<=LUI)||inst_chose==JAL||inst_chose==JALR||(inst_chose>=LB&&inst_chose<=LH)||
    (inst_chose>=MFC0&&inst_chose<=MFLO)||inst_chose==CLZ||inst_chose==MUL);
    assign is_signed=!((inst_chose>=ANDI)&&(inst_chose<=SW));
    assign H_R = (inst_chose==MFHI);
    assign L_R = (inst_chose==MFLO);
    assign H_W = (inst_chose==MTHI||inst_chose==MUL||inst_chose==MULTU||inst_chose==DIV||inst_chose==DIVU);
    assign L_W = (inst_chose==MTLO||inst_chose==MUL||inst_chose==MULTU||inst_chose==DIV||inst_chose==DIVU);
    assign cp0_R=(inst_chose==MFC0);
    assign cp0_W =(inst_chose==MTC0) ;
    assign exp = (inst_chose==SYSCALL||inst_chose==BREAK||(inst_chose==TEQ&&alu_res));
    assign eret = (inst_chose==ERET);
    assign mul_is_signed = (inst_chose==MUL);
    assign mul_work = (inst_chose==MUL||inst_chose
==MULTU);
    assign div_start =(inst_chose==DIV||inst_chose
==DIVU)&&!div_busy ;
    assign div_is_signed =(inst_chose==DIV) ;
    always @(*)
    begin
    if(inst_chose==SYSCALL)
        cause<=5'b1000;
    else if(inst_chose==BREAK)
        cause<=5'b1001;
    else if(inst_chose==TEQ&&alu_res)
        cause<=5'b1101;
    end
endmodule
```

内存模块:

```
`timescale 1ns / 1ps


module DMEM (
    input   clk,
    input   ena,
    input   DM_R,
    input   DM_W,
    input   [31:0]addr,
    input   [31:0]d_addr,
    input   [31:0]DM_in,
    input   [5:0]inst_chose,
    output  reg [31:0]DM_out
);
    parameter  LW=22 ;
    parameter  SW=23 ;
    parameter   LB=33;//取字节
    parameter   LBU=34;//取字节无
    parameter   LHU=35;//取半字（无符号
    parameter   LH=36;//有
    parameter   SB=37;//存字节
    parameter   SH=38;//存半字
    reg [31:0] DM_data[640:0];//[31:0];
    always @(*)//(negedge clk)// or negedge ena)
    begin
        if (DM_W)
        begin
            DM_data[d_addr]<=DM_in;
        end
        if(DM_R)
        begin

            if(inst_chose==LW)
                DM_out<=DM_data[d_addr];
            else if(inst_chose==LHU)
            begin
                if(addr[1:0]==2'b00)
                    DM_out<={16'b0,DM_data[d_addr]
[15:0]};
                else if(addr[1:0]==2'b10)
                    DM_out<={DM_data[d_addr][31:16
],16'b0};
            end
            else if(inst_chose==LH)
            begin
                if(addr[1:0]==2'b00)
                begin
                    if(DM_data[d_addr][15]==1)
                        DM_out<={16'hffff,DM_data[
d_addr][15:0]};
                    else begin
```

```
                        DM_out<={16'b0,DM_data[d_a
ddr][15:0]};
                    end
                end
                else if(addr[1:0]==2'b10)
                begin
                    if(DM_data[d_addr][31]==1)
                        DM_out<={16'hffff,DM_data[
d_addr][31:16]};
                    else begin
                        DM_out<={16'b0,DM_data[d_a
ddr][31:16]};
                    end
                end
            end
            else if(inst_chose==LB)
            begin
                if(addr[1:0]==2'b00)
                begin
                    if(DM_data[d_addr][7]==1)
                        DM_out<={24'hffffff,DM_dat
a[d_addr][7:0]};
```

```
                    else
                        DM_out<={24'b0,DM_data[d_a
ddr][7:0]};
                end
                else if(addr[1:0]==2'b01)
                begin
                    if(DM_data[d_addr][15]==1)
                        DM_out<={24'hffffff,DM_dat
a[d_addr][15:8]};
                    else
                        DM_out<={24'b0,DM_data[d_a
ddr][15:8]};
                end
                else if(addr[1:0]==2'b10)
                begin
                    if(DM_data[d_addr][23]==1)
                        DM_out<={24'hffffff,DM_dat
a[d_addr][23:16]};
                    else
                        DM_out<={24'b0,DM_data[d_a
ddr][23:16]};
                end
                else if(addr[1:0]==2'b11)
                begin
                    if(DM_data[d_addr][31]==1)
                        DM_out<={24'hffffff,DM_dat
a[d_addr][31:24]};
                    else
                        DM_out<={24'b0,DM_data[d_a
ddr][31:24]};
                end
            end
            else if(inst_chose==LBU)
            begin
                if(addr[1:0]==2'b00)
                begin
                    DM_out<={24'b0,DM_data[d_addr]
[7:0]};
                end
                else if(addr[1:0]==2'b01)
                begin
                    DM_out<={24'b0,DM_data[d_addr]
[15:8]};
                end
                else if(addr[1:0]==2'b10)
                begin
                    DM_out<={24'b0,DM_data[d_addr]
[23:16]};
                end
                else if(addr[1:0]==2'b11)
                begin
                    DM_out<={24'b0,DM_data[d_addr]
[31:24]};
                end
            end
        end
        else begin
            DM_out<=32'hz ;
        end
    end
endmodule
```

5 位数拓展模块：

```
module EXT5 (
    input [4:0]EXT_in,
    output [31:0]EXT5_out
);
```

```
    assign EXT5_out ={27'b0000_0000_0000_0000_0000
_0000_00,EXT_in} ;
endmodule
```

16 位数拓展模块：

```
module EXT16 (
    input is_signed,
    input [15:0]EXT_in,
    output [31:0]EXT16_out
);
    assign EXT16_out =is_signed && EXT_in[15]?{16'
b1111_1111_1111_1111,EXT_in}:{16'b0000_0000_0000_0
000,EXT_in};
endmodule
```

18 位数拓展模块：

```
module EXT18 (
    input [15:0]EXT_in,
    output [31:0]EXT18_out
);
    assign EXT18_out =EXT_in[15]?{14'b1111_1111_11
11_11,EXT_in,2'b00}:{14'b0000_0000_0000_00,EXT_in,
2'b00} ;
endmodule
```

加法模块：

```
module ADD (

    input [31:0]A,

    input [31:0]B,

    output [31:0]ADD_out

);

    assign ADD_out =A + B ;

endmodule
```

拼接模块：

```
module ll (

    input [31:0]inst,

    input [31:0]pc,

    output [31:0]ll_out

);

    assign ll_out ={pc[31:28],inst[25:0],2'b00} ;

endmodule
```

二路选择器模块：

```
module MUX (

    input[31:0] A,

    input[31:0] B,

    input flag,

    output[31:0]mux_out
);
    assign mux_out=flag?A:B;
endmodule
```

NPC 模块：

```
module NPC (
    input [31:0]data_in,
    output [31:0]data_out
);
    assign data_out =data_in + 4 ;
endmodule
```

PC 寄存器模块：

```
`timescale 1ns / 1ps


module PC_reg (
    input clk,
    input ena,
    input rst,
    input [31:0]data_in,
    output reg [31:0]data_out
);
    always @ (posedge rst or negedge clk) begin
        if (rst) begin
```

```
            data_out <= 32'h0040_0000;
        end else if (ena) begin
            data_out <= data_in;
        end
    end
endmodule
```

对于 SB、SH 这种需要处理的

```
`timescale 1ns / 1ps

module DM_in_opera (
    input [5:0]inst_chose,
    input [31:0]Rt,
    input [31:0]alu_out,
    output reg [31:0]DM_in
);
    parameter  LW=22 ;
    parameter  SW=23 ;
    parameter   LB=33;//取字�?
    parameter   LBU=34;//取字节无
    parameter   LHU=35;//取半字（无符�?
    parameter   LH=36;//�?
    parameter   SB=37;//存字�?
    parameter   SH=38;//存半�?
always @(*)
begin
    if(inst_chose==SW)
        DM_in<=Rt;
    else if(inst_chose==SH)
    begin
        if(alu_out[1:0]==2'b00)
            DM_in<={16'b0,Rt[15:0]};
        else if(alu_out[1:0]==2'b10)
            DM_in<={Rt[15:0],16'b0};
    end
    else if(inst_chose==SB)
    begin
        if(alu_out[1:0]==2'b00)
            DM_in<={24'b0,Rt[7:0]};
        else if(alu_out[1:0]==2'b01)
            DM_in<={16'b0,Rt[7:0],8'b0};
        else if(alu_out[1:0]==2'b10)
            DM_in<={8'b0,Rt[7:0],16'b0};
        else if(alu_out[1:0]==2'b11)
            DM_in<={Rt[7:0],24'b0};
    end

    else

        DM_in<=32'bz;

end

endmodule
```

CP0 模块

```
`timescale 1ns / 1ps


module cp0 (

    input clk,

    input rst,

    input mfc0,

    input mtc0,

    input [31:0]pc,

    input [4:0]Rd,

    input [31:0]wdata,

    input expection,

    input eret,

    input [4:0]cause,

    input intr,

    output [31:0]rdata,
    output [31:0]status,
    output reg timer_int,
    output [31:0]exc_addr
);
    parameter STATUS=12;
    parameter CAUSE=13;
    parameter EPC=14;
    parameter STATUS_SYSCALL=8;
    parameter STATUS_BREAK=9;
    parameter STATUS_TEQ=10;

    parameter SYSCALL = 5'b1000;
    parameter BREAK = 5'b1001;
    parameter TEQ = 5'b1101;

    reg [31:0] cp0_register[31:0];

    assign rdata=(mfc0||eret)?cp0_register[Rd]:32'
bz;//
    assign status=cp0_register[STATUS];
    assign exc_addr=expection?32'h00400004:(eret?c
p0_register[EPC]:32'hz);
```

```
    always @ (posedge clk or posedge rst)//
    begin
        if(rst)
        begin
            cp0_register[0] <= 32'b0;
            cp0_register[1] <= 32'b0;
            cp0_register[2] <= 32'b0;
            cp0_register[3] <= 32'b0;
            cp0_register[4] <= 32'b0;
            cp0_register[5] <= 32'b0;
            cp0_register[6] <= 32'b0;
            cp0_register[7] <= 32'b0;
            cp0_register[8] <= 32'b0;
            cp0_register[9] <= 32'b0;
            cp0_register[10] <= 32'b0;
            cp0_register[11] <= 32'b0;
            cp0_register[12] <= 32'b00000000111000
00001;
            cp0_register[13] <= 32'b0;
            cp0_register[14] <= 32'b0;
            cp0_register[15] <= 32'b0;
            cp0_register[16] <= 32'b0;
            cp0_register[17] <= 32'b0;
            cp0_register[18] <= 32'b0;
            cp0_register[19] <= 32'b0;
            cp0_register[20] <= 32'b0;
            cp0_register[21] <= 32'b0;
            cp0_register[22] <= 32'b0;
            cp0_register[23] <= 32'b0;
            cp0_register[24] <= 32'b0;
            cp0_register[25] <= 32'b0;
            cp0_register[26] <= 32'b0;
            cp0_register[27] <= 32'b0;
            cp0_register[28] <= 32'b0;
            cp0_register[29] <= 32'b0;
            cp0_register[30] <= 32'b0;
            cp0_register[31] <= 32'b0;
        end
        else
        begin
            if(mtc0)
            begin
            cp0_register[Rd] <= wdata;
            end


            if(expection)

            begin

                cp0_register[CAUSE][6:2]<=cause;

                cp0_register[STATUS]<={cp0_registe
r[STATUS][26:0],5'b0};

                cp0_register[EPC]<=pc+32'h4;

            end

            else

            if(eret)

            begin

                cp0_register[STATUS]<=cp0_register
[STATUS]>>5;

            end

        end

    end

endmodule  //cpo0
```

HILO 模块

```
`timescale 1ns / 1ps


module HILO (
    input [31:0] hi_in,

    input [31:0] lo_in,

    input rst,

    input ena,

    input H_R,

    input L_R,

    input H_W,

    input L_W,

    output [31:0] hi_out,

    output [31:0] lo_out

);

    reg [31:0]hi;

    reg [31:0]lo;

    assign hi_out =H_R?hi:32'hz;

    assign lo_out =L_R?lo:32'hz;

    always @(*)

    begin

        if(rst)

        begin

            hi<=0;

            lo<=0;

        end
        else
        begin
            if(H_W)
            begin
                hi<=hi_in;//hi_in;
            end
            if(L_W)
            begin
                lo<=lo_in;
            end
        end
    end
endmodule
```

乘法器模块

```
`timescale 1ns / 1ps


/*module MULTU2(
    input clk,
    input reset,
    input [1:0]a,
    input [1:0]b,
    output [3:0]z
```

```
);
    reg [3:0]t0,t1;
    always@(*)
    if(reset)
    begin
        t0<=0;
        t1<=0;
    end
    else
    begin
        t0<=b[0]?{2'b0,a}:4'b0;
        t1<=b[1]?{1'b0,a,1'b0}:4'b0;
    end
    assign z = t0+t1;
endmodule


module MULTU4(
    input clk,
    input reset,
    input [3:0]a,
    input [3:0]b,
    output [7:0]z
    );
    wire [3:0]s1,s2,s3,s4;
    MULTU2 uut0(clk,reset,a[3:2],b[3:2],s1);
    MULTU2 uut1(clk,reset,a[3:2],b[1:0],s2);
    MULTU2 uut2(clk,reset,a[1:0],b[3:2],s3);
    MULTU2 uut3(clk,reset,a[1:0],b[1:0],s4);
    assign z = {s1,4'b0}+{2'b0,s2,2'b0}+{2'b0,s3,2
'b0}+{4'b0,s4};
endmodule*/

module MULTU4(
    input clk,
    input reset,
    input [3:0]a,
    input [3:0]b,
    output [7:0]z
    );
    reg [7:0]t0,t1,t2,t3,add1,add2;
    reg f;
    always@(posedge clk or posedge reset)
    if(reset)
    begin
```

```
        f <= 0;
        t0<=0;
        t1<=0;
        t2<=0;
        t3<=0;
        add1<=0;
        add2<=0;
    end
    else
    begin
        t0 <= b[0] ? {4'b0,a} :8'b0;
        t1 <= b[1] ? {3'b0,a,1'b0} :8'b0;
        t2 <= b[2] ? {2'b0,a,2'b0} :8'b0;
        t3 <= b[3] ? {1'b0,a,3'b0} :8'b0;
        f<=1;
    end
    assign z = f ? t0+t1+t2+t3 : 8'b0;
endmodule

module MULTU8(
    input clk,
    input reset,
    input [7:0]a,
    input [7:0]b,
    output [15:0]z
    );
    wire [7:0]s1,s2,s3,s4;
    MULTU4 uut0(clk,reset,a[7:4],b[7:4],s1);
    MULTU4 uut1(clk,reset,a[7:4],b[3:0],s2);
    MULTU4 uut2(clk,reset,a[3:0],b[7:4],s3);
    MULTU4 uut3(clk,reset,a[3:0],b[3:0],s4);
    assign z = {s1,8'b0}+{4'b0,s2,4'b0}+{4'b0,s3,4
'b0}+{8'b0,s4};
endmodule

module MULTU16(
    input clk,
    input reset,
    input [15:0]a,
    input [15:0]b,
    output [31:0]z
    );
    wire [15:0]s1,s2,s3,s4;
    MULTU8 uut0(clk,reset,a[15:8],b[15:8],s1);
    MULTU8 uut1(clk,reset,a[15:8],b[7:0],s2);
    MULTU8 uut2(clk,reset,a[7:0],b[15:8],s3);
    MULTU8 uut3(clk,reset,a[7:0],b[7:0],s4);
    assign z = {s1,16'b0}+{8'b0,s2,8'b0}+{8'b0,s3,
8'b0}+{16'b0,s4};
endmodule

module MULTU32(
    input clk,
    input reset,
    input [31:0]a,
    input [31:0]b,
    output [63:0]z
    );
    wire [31:0]s1,s2,s3,s4;
    MULTU16 uut0(clk,reset,a[31:16],b[31:16],s1);
    MULTU16 uut1(clk,reset,a[31:16],b[15:0],s2);
    MULTU16 uut2(clk,reset,a[15:0],b[31:16],s3);
    MULTU16 uut3(clk,reset,a[15:0],b[15:0],s4);
    assign z = {s1,32'b0}+{16'b0,s2,16'b0}+{16'b0,
s3,16'b0}+{32'b0,s4};
endmodule
```

```
module MULT_ (
    input clk,
    input reset,
    input mul_is_signed,
    input mul_work,
    input [31:0]a,
    input [31:0]b,
    output [31:0]z_h,
    output [31:0]z_l
);
    wire [63:0]s1,s2,s3,s4;
    wire [127:0]temp;
    wire [31:0]ta,tb;
    wire [31:0]aa;
    wire [31:0]bb;
    assign aa =mul_work?a:32'hz;
    assign bb =mul_work?b:32'hz;
    assign ta={a[31],a[31],a[31],a[31],a[31],a[31]
,a[31],a[31],a[31],a[31],a[31],a[31],a[31],a[31],a
[31],a[31],a[31],a[31],a[31],a[31],a[31],a[31],a[3
1],a[31],a[31],a[31],a[31],a[31],a[31],a[31],a[31]
,a[31]};
    assign tb={b[31],b[31],b[31],b[31],b[31],b[31]
,b[31],b[31],b[31],b[31],b[31],b[31],b[31],b[31],b
[31],b[31],b[31],b[31],b[31],b[31],b[31],b[31],b[3
1],b[31],b[31],b[31],b[31],b[31],b[31],b[31],b[31]
,b[31]};
    MULTU32 uut0(clk,reset,ta,tb,s1);
    MULTU32 uut1(clk,reset,ta,bb,s2);
    MULTU32 uut2(clk,reset,aa,tb,s3);
    MULTU32 uut3(clk,reset,aa,bb,s4);
    assign temp = {s1,64'b0}+{32'b0,s2,32'b0}+{32'
b0,s3,32'b0}+{64'b0,s4};
    assign z_l=mul_is_signed? temp[31:0]:s4[31:0];
    assign z_h =mul_is_signed?temp[63:32]:s4[63:32
] ;
endmodule
```

有符号除法器模块

```
`timescale 1ns / 1ps


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 [5:0] count;
    wire ready;
    reg [31:00] reg_q;
    reg [31:00] reg_r;
    reg [31:00] reg_b;
    wire [31:00] reg_r2;
    reg r_sign,sign;
    wire [32:0] temp=r_sign?({reg_r,reg_q[31]}+{1'
b0,reg_b}):
                                ({reg_r,reg_q[31]}
-{1'b0,reg_b});
    assign reg_r2=r_sign?reg_r+reg_b:reg_r;
    assign r=dividend[31]?(~reg_r2+1):reg_r2;
    assign q=(divisor[31]^dividend[31])?(~reg_q+1)
:reg_q;

    always @(posedge clock or posedge reset)begin
    if(reset)begin
        count<=0;
        busy<=0;
    end
    else begin
        if(start)begin
            r_sign<=0;
            reg_r<=32'b0;
            if(dividend[31]==1) begin
                reg_q<=~dividend+1;
            end
            else reg_q<=dividend;
            if(divisor[31]==1)begin
                reg_b<=~divisor+1;
            end
            else reg_b<=divisor;
            count<=0;
            busy<=1;
        end
        else if(busy)begin
            reg_q<={reg_q[30:0],~temp[32]};
            r_sign<=temp[32];
            reg_r<=temp[31:0];
            count<=count+1;
            if(count==31)busy<=0;
        end
    end
    end
endmodule
```

无符号除法器模块

```
`timescale 1ns / 1ps

module DIVU(
input [31:0]dividend, //被除数
input [31:0]divisor, //除数
input start, //启动除法运算
input clock,
input reset,
output [31:0]q, //商
output [31:0]r, //余数
output reg busy //除法器忙标志

);

reg[5:0]count;

reg [31:0] reg_q;

reg [31:0] reg_r;

reg [31:0] reg_b;

wire [32:0] temp ={reg_r,reg_q[31]} - {1'b0,reg_b}

;//进行减法

assign r = reg_r;

assign q = reg_q;

always @ (posedge clock or posedge reset)begin

if (reset == 1) begin //重置

count <=3'b0;

busy <= 0;

end else begin

if (start) begin //开始除法运算，初始化

reg_r <= 32'b0;

reg_q <= dividend;

reg_b <= divisor;

count <= 5'b0;

busy <= 1'b1;

end else if (busy) begin //循环操作
```

```
reg_r <= temp[32]?{reg_r[30:0],reg_q[31]}:temp[31:0];//部分余数
reg_q <= temp[32]?{reg_q[30:0],1'b0}:{reg_q[30:0],1'b1};
count <= count +1; //计数器加+1
if (count == 31) busy <= 0; //结束除法运算
end
end
end
endmodule
```

将无符号和有符号除法器整合在一起的模块

```
`timescale 1ns / 1ps


module DIV_ (
    input [31:0] dividend,
    input [31:0] divisor,
    input start,
    input clock,
    input reset,
    input div_is_signed,
    output [31:0] q,
    output [31:0] r,
    output busy
);
    wire [31:0]q_s;
    wire [31:0]r_s;
    wire [31:0]q_u;
    wire [31:0]r_u;
    wire busy_s,busy_u;
    DIV div(dividend,divisor,start,clock,reset,q_s
,r_s,busy_s);
    DIVU divu(dividend,divisor,start,clock,reset,q
_u,r_u,busy_u);
    assign q =div_is_signed?q_s:q_u;
    assign r =div_is_signed?r_s:r_u;
    assign busy =div_is_signed?busy_s:busy_u;
endmodule
```

四、测试模块建模

```
`timescale 1ns / 1ps
module test;
```

```
/*
reg [31:0]a;
wire [31:0]d;
IMEM im(a,d);
initial begin
a=32'h0040_0000;
end//*/

reg clk,rst;
wire [31:0]inst;
//wire [31:0]pc_in;
wire [31:0]DM_out;
wire [31:0]pc;
//wire DM_chose;
//wire DM_R,DM_W;
//wire[31:0]DM_addr;
wire[31:0]DM_in;
wire[31:0]im_addr;
wire mc1,mc2,mc3,mc4,mc5,mc6,mc7;
wire [31:0]mux1_out;
wire [31:0]mux2_out;
wire [31:0]mux3_out;
wire [31:0]mux4_out;
wire [31:0]mux5_out;
wire [31:0]mux6_out;
wire [31:0]mux7_out;
wire [4:0]Rsc;
wire [4:0]Rdc;
wire [4:0]Rtc;
wire rf_en;
wire [31:0]ext16_out;
wire DM_R,DM_W;
wire [31:0]DM_addr;
wire [4:0] inst_chose;
integer file_open;
integer counter;
sccomp_dataflow sc(clk,rst,inst,pc);//,im_addr,mc1
,mc2,mc3,mc4,mc5,mc6,mc7,mux1_out,mux2_out,mux3_ou
t,mux4_out,mux5_out,mux6_out,mux7_out,Rsc,Rdc,Rtc,
rf_en,ext16_out,DM_R,DM_W,DM_addr,DM_in,DM_out,ins
t_chose);
//CPU31 sccpu(clk,rst,inst,DM_out,pc,DM_chose,DM_R
,DM_W,DM_addr,DM_in);
//PC_reg pc_r(clk,1,rst,pc_in,pc);
```

```
initial begin

    file_open = $fopen("output.txt", "w+");

        // Initialize Inputs

        clk = 0;

        rst = 1;

        //pc 初始值 32'h00400000

        //inst 初始值 32'h08100004




        // Wait 200 ns for global reset to finish

        #20;

        rst = 0;



    end


    always begin

    #3;

    clk = ~clk;

    if(clk == 1'b1 && rst == 0)

    begin
```

```
                $fdisplay(file_open, "pc: %h", sc.
pc);
                $fdisplay(file_open, "instr: %h",
sc.inst);
                $fdisplay(file_open, "regfile0: %h
", sc.sccpu.cpu_ref.array_reg[0]);
                $fdisplay(file_open, "regfile1: %h
", sc.sccpu.cpu_ref.array_reg[1]);
                $fdisplay(file_open, "regfile2: %h
", sc.sccpu.cpu_ref.array_reg[2]);
                $fdisplay(file_open, "regfile3: %h
", sc.sccpu.cpu_ref.array_reg[3]);
                $fdisplay(file_open, "regfile4: %h
", sc.sccpu.cpu_ref.array_reg[4]);
                $fdisplay(file_open, "regfile5: %h
", sc.sccpu.cpu_ref.array_reg[5]);
                $fdisplay(file_open, "regfile6: %h
", sc.sccpu.cpu_ref.array_reg[6]);
                $fdisplay(file_open, "regfile7: %h
", sc.sccpu.cpu_ref.array_reg[7]);
                $fdisplay(file_open, "regfile8: %h
", sc.sccpu.cpu_ref.array_reg[8]);
```

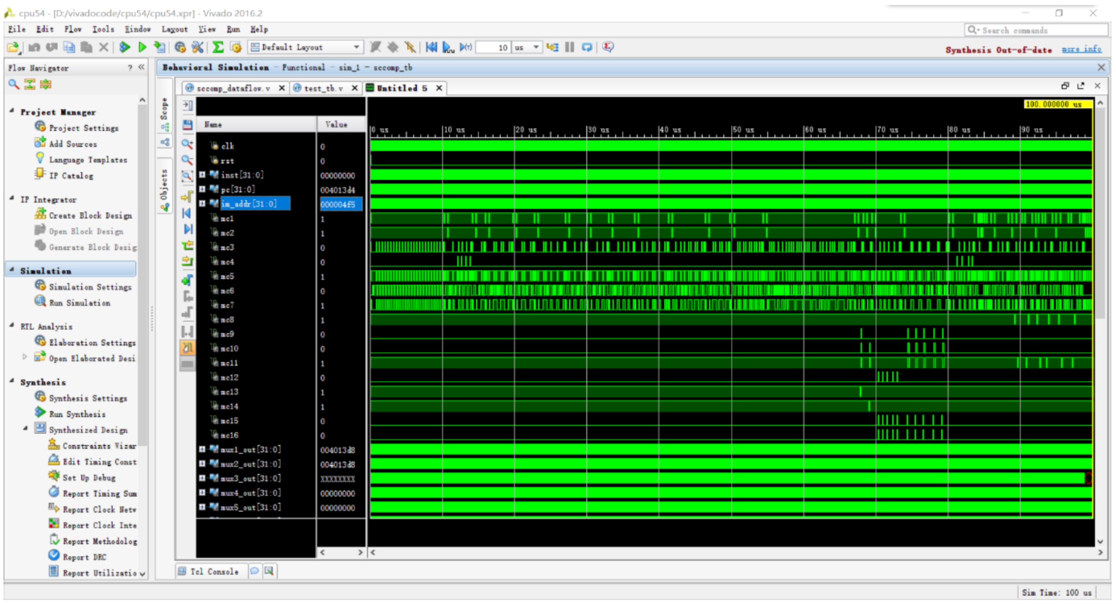
```
                $fdisplay(file_open, "regfile9: %h
", sc.sccpu.cpu_ref.array_reg[9]);
                $fdisplay(file_open, "regfile10: %
h", sc.sccpu.cpu_ref.array_reg[10]);
                $fdisplay(file_open, "regfile11: %
h", sc.sccpu.cpu_ref.array_reg[11]);
                $fdisplay(file_open, "regfile12: %
h", sc.sccpu.cpu_ref.array_reg[12]);
                $fdisplay(file_open, "regfile13: %
h", sc.sccpu.cpu_ref.array_reg[13]);
                $fdisplay(file_open, "regfile14: %
h", sc.sccpu.cpu_ref.array_reg[14]);
                $fdisplay(file_open, "regfile15: %
h", sc.sccpu.cpu_ref.array_reg[15]);
                $fdisplay(file_open, "regfile16: %
h", sc.sccpu.cpu_ref.array_reg[16]);
                $fdisplay(file_open, "regfile17: %
h", sc.sccpu.cpu_ref.array_reg[17]);
                $fdisplay(file_open, "regfile18: %
h", sc.sccpu.cpu_ref.array_reg[18]);
                $fdisplay(file_open, "regfile19: %
h", sc.sccpu.cpu_ref.array_reg[19]);
```

```
                    $fdisplay(file_open, "regfile20: %
h", sc.sccpu.cpu_ref.array_reg[20]);
                    $fdisplay(file_open, "regfile21: %
h", sc.sccpu.cpu_ref.array_reg[21]);
                    $fdisplay(file_open, "regfile22: %
h", sc.sccpu.cpu_ref.array_reg[22]);
                    $fdisplay(file_open, "regfile23: %
h", sc.sccpu.cpu_ref.array_reg[23]);
                    $fdisplay(file_open, "regfile24: %
h", sc.sccpu.cpu_ref.array_reg[24]);
                    $fdisplay(file_open, "regfile25: %
h", sc.sccpu.cpu_ref.array_reg[25]);
                    $fdisplay(file_open, "regfile26: %
h", sc.sccpu.cpu_ref.array_reg[26]);
                    $fdisplay(file_open, "regfile27: %
h", sc.sccpu.cpu_ref.array_reg[27]);
                    $fdisplay(file_open, "regfile28: %
h", sc.sccpu.cpu_ref.array_reg[28]);
                    $fdisplay(file_open, "regfile29: %
h", sc.sccpu.cpu_ref.array_reg[29]);
                    $fdisplay(file_open, "regfile30: %
h", sc.sccpu.cpu_ref.array_reg[30]);
```

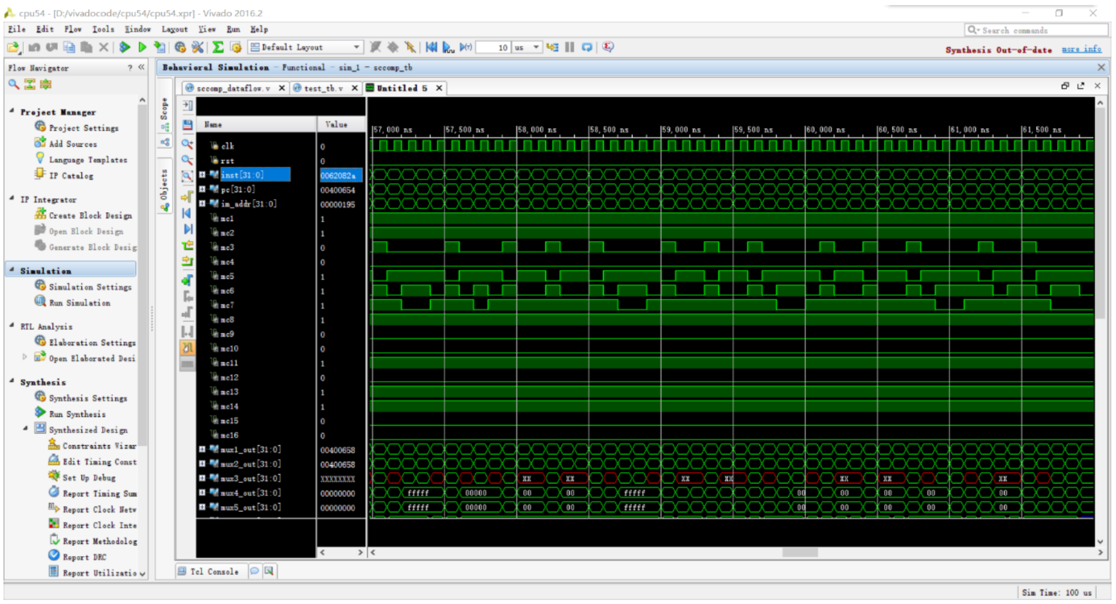
```
                $fdisplay(file_open, "regfile31: %h", sc.sccpu.cpu_ref.array_reg[31]);
        end
    end
endmodule
```

五、实验结果

前仿真截图，调试过程将大量结果都输出出来好检查：

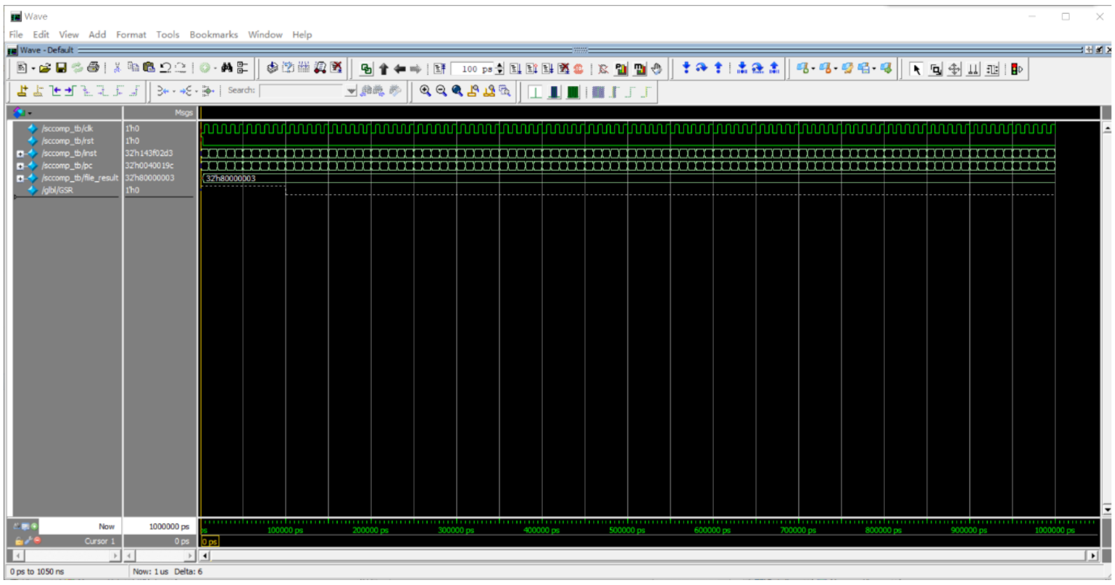


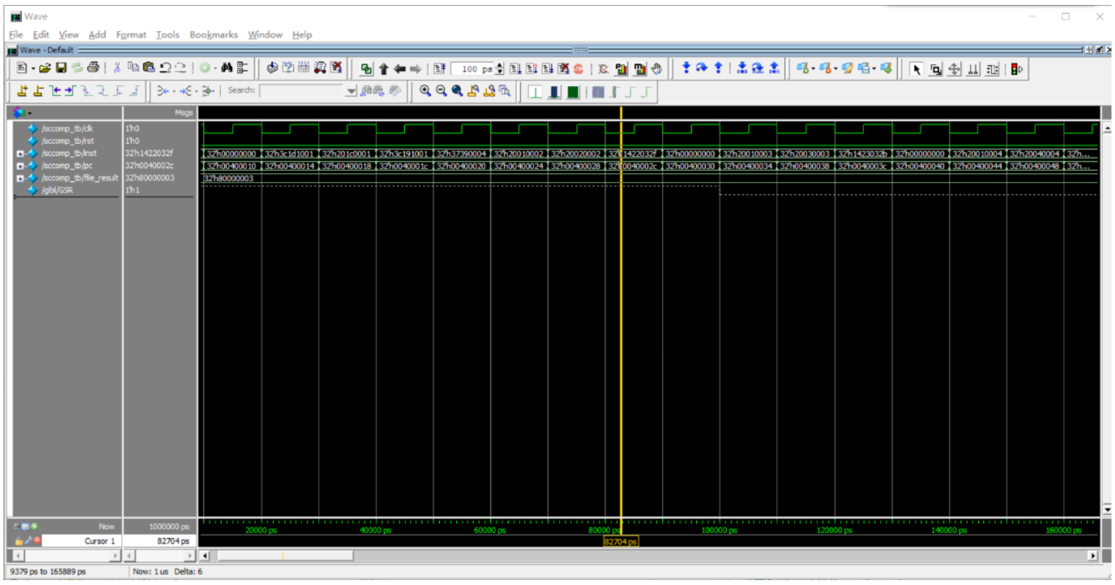
```
Name                Value
clk                 0
rst                 0
inst[31:0]          00000000
pc[31:0]            004013d4
im_addr[31:0]       000004f5
mc1                 1
mc2                 1
mc3                 0
mc4                 0
mc5                 1
mc6                 0
mc7                 1
mc8                 1
mc9                 0
mc10                0
mc11                1
mc12                0
mc13                1
mc14                1
mc15                0
mc16                0
mux1_out[31:0]      004013d8
mux2_out[31:0]      004013d8
mux3_out[31:0]      XXXXXXXX
mux4_out[31:0]      00000000
mux5_out[31:0]      00000000
```

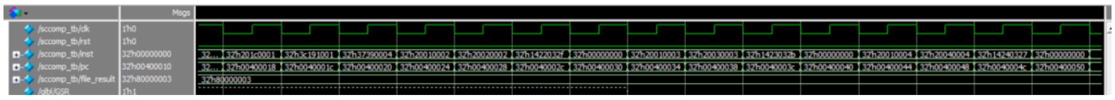


```
Name                Value
clk                 0
rst                 0
inst[31:0]          0062082a
pc[31:0]            00400654
im_addr[31:0]       00000195
mc1                 1
mc2                 1
mc3                 0
mc4                 0
mc5                 1
mc6                 1
mc7                 1
mc8                 1
mc9                 0
mc10                0
mc11                1
mc12                0
mc13                1
mc14                1
mc15                0
mc16                0
mux1_out[31:0]      00400658
mux2_out[31:0]      00400658
mux3_out[31:0]      XXXXXXXX
mux4_out[31:0]      00000000
mux5_out[31:0]      00000000
```

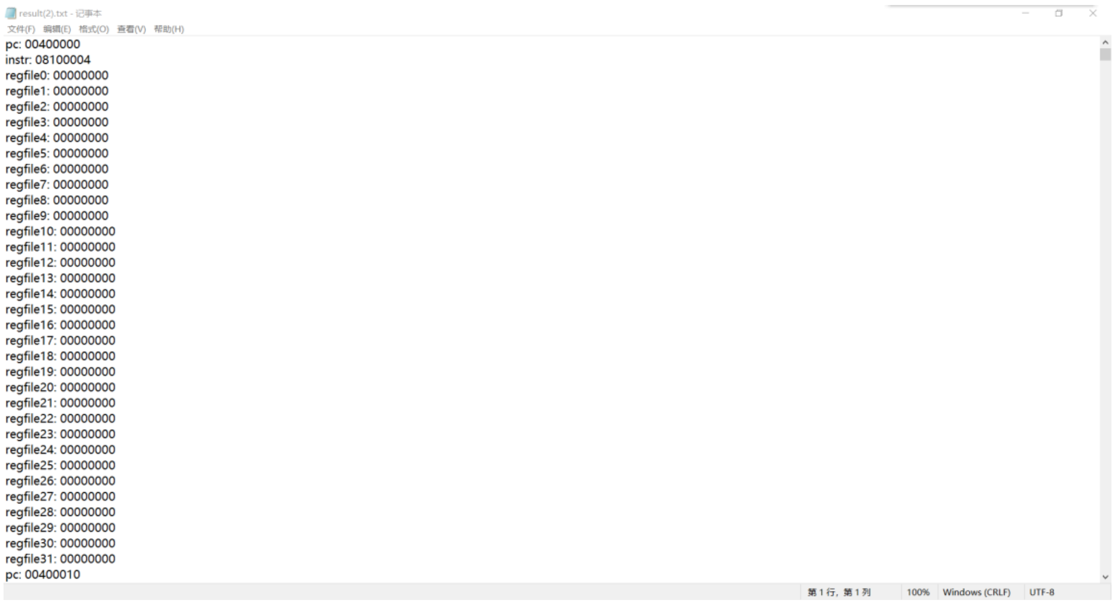
按提交要求得到的前仿真截图：


```
/sccomp_tb/clk          1'h0
/sccomp_tb/rst          1'h0
/sccomp_tb/inst         32'h143f02d3
/sccomp_tb/pc           32'h0040019c
/sccomp_tb/file_result  32'h80000003
/glbl/GSR               1'h0

Now       1000000 ps
Cursor 1  0 ps
0 ps to 1050 ns    Now: 1 us  Delta: 6
```


```
/sccomp_tb/clk          1'h0
/sccomp_tb/rst          1'h0
/sccomp_tb/inst         32'h1422032f
/sccomp_tb/pc           32'h0040002c
/sccomp_tb/file_result  32'h80000003
/glbl/GSR               1'h1

inst: 32'h00000000 32'h3c1d1001 32'h201c0001 32'h3c191001 32'h37390004 32'h20010002 32'h20020002 32'h1422032f 32'h00000000 32'h20010003 32'h20030003 32'h1423032b 32'h00000000 32'h20010004 32'h20040004 32'h...
pc:   32'h00400010 32'h00400014 32'h00400018 32'h0040001c 32'h00400020 32'h00400024 32'h00400028 32'h0040002c 32'h00400030 32'h00400034 32'h00400038 32'h0040003c 32'h00400040 32'h00400044 32'h00400048 32'h...
file_result: 32'h80000003

Now       1000000 ps
Cursor 1  82704 ps
9379 ps to 165889 ps    Now: 1 us  Delta: 6
```


```
/sccomp_tb/clk          1'h0
/sccomp_tb/rst          1'h0
/sccomp_tb/inst         32'h00000000
/sccomp_tb/pc           32'h00400010
/sccomp_tb/file_result  32'h80000003
/glbl/GSR               1'h1

inst: 32... 32'h201c0001 32'h3c191001 32'h37390004 32'h20010002 32'h20020002 32'h1422032f 32'h00000000 32'h20010003 32'h20030003 32'h1423032b 32'h00000000 32'h20010004 32'h20040004 32'h14240327 32'h00000000
pc:   32... 32'h00400018 32'h0040001c 32'h00400020 32'h00400024 32'h00400028 32'h0040002c 32'h00400030 32'h00400034 32'h00400038 32'h0040003c 32'h00400040 32'h00400044 32'h00400048 32'h0040004c 32'h00400050
file_result: 32'h80000003
```

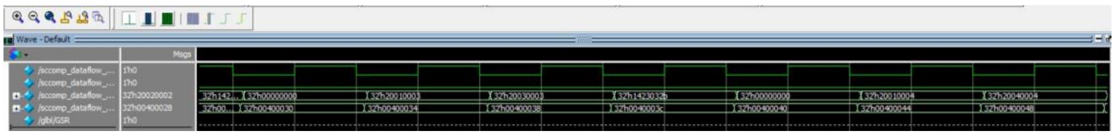
将寄存器的内容用文件输出出来



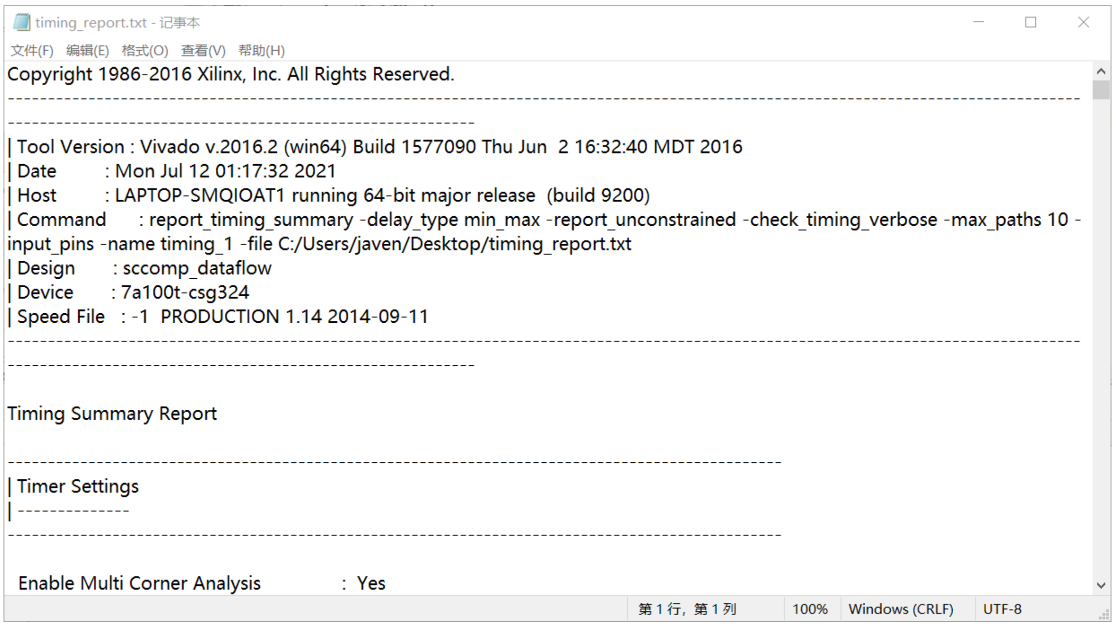
```
pc: 00400000
instr: 08100004
regfile0: 00000000
regfile1: 00000000
regfile2: 00000000
regfile3: 00000000
regfile4: 00000000
regfile5: 00000000
regfile6: 00000000
regfile7: 00000000
regfile8: 00000000
regfile9: 00000000
regfile10: 00000000
regfile11: 00000000
regfile12: 00000000
regfile13: 00000000
regfile14: 00000000
regfile15: 00000000
regfile16: 00000000
regfile17: 00000000
regfile18: 00000000
regfile19: 00000000
regfile20: 00000000
regfile21: 00000000
regfile22: 00000000
regfile23: 00000000
regfile24: 00000000
regfile25: 00000000
regfile26: 00000000
regfile27: 00000000
regfile28: 00000000
regfile29: 00000000
regfile30: 00000000
regfile31: 00000000
pc: 00400010
```

可以在网站上 AC

后仿真截图：

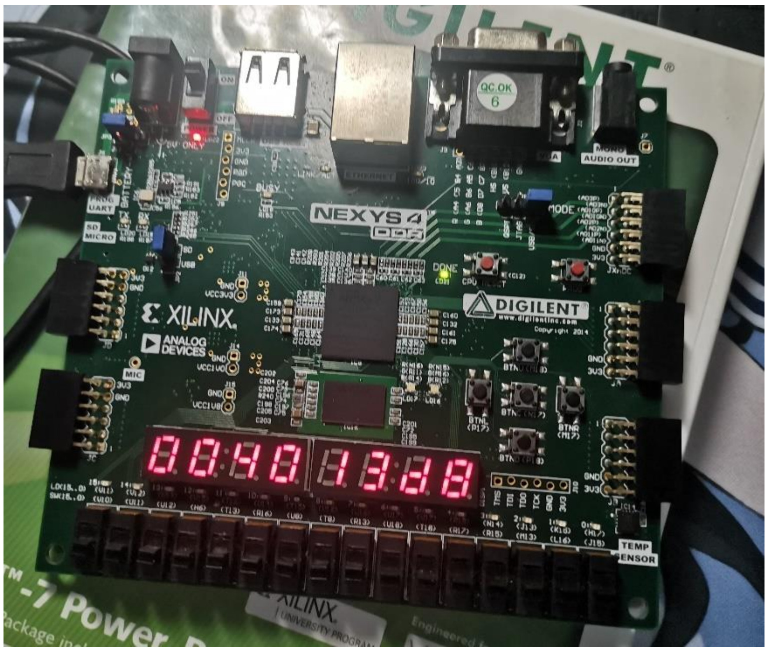


可以看到相比于前仿真后仿真上面每一个指令都相较于 PC 后移了一些位置，即加入了门电路的延迟。
得到的时序报告：


timing_report.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
Copyright 1986-2016 Xilinx, Inc. All Rights Reserved.
--------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------
| Tool Version : Vivado v.2016.2 (win64) Build 1577090 Thu Jun  2 16:32:40 MDT 2016
| Date         : Mon Jul 12 01:17:32 2021
| Host         : LAPTOP-SMQIOAT1 running 64-bit major release  (build 9200)
| Command      : report_timing_summary -delay_type min_max -report_unconstrained -check_timing_verbose -max_paths 10 -
input_pins -name timing_1 -file C:/Users/javen/Desktop/timing_report.txt
| Design       : sccomp_dataflow
| Device       : 7a100t-csg324
| Speed File   : -1  PRODUCTION 1.14 2014-09-11
--------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------

Timing Summary Report

------------------------------------------------------------------------------------------
| Timer Settings
| --------------
------------------------------------------------------------------------------------------

 Enable Multi Corner Analysis              :  Yes
```

下板图片：



数码管显示的 PC 值能稳定在 004013d8 的原因是最后是陷入了 4013d4,4013d8，4013dc 的死循环，这样数字叠加的结果就是如上了

六、心得

本实验是在原有 31 条单周期 CPU 的基础上得到的 54 条单周期 CPU，画剩余 23 条指令的流程图就大概花费了半天的时间，然后写这些并进行基础的测试花了大概 2 天，写的过程是将指令一条一条往上加的，过程虽然比较慢，但感觉还比较扎实。后年为了提交去 debug 又大概花了 1 天的时间去找 bug，找到了不限于选择器的指令赋值有错误，CP0 的 pc 读入读出有错误等等，印象最深的是中断后往 cp0 存 pc，因为原本是将 npc 的结果直接拿过来放入 cp0 相应的寄存器里面的，但是这时候的 pc 的预备值已经变成了 04 只不过还没到下降沿读取而已，所以 npc 的值也变成了 08，即存入的变成了 08，后来的解决办法就是将当前 pc 值传入，然后在内部进行+4 存入。

后面又有后仿真和下板，在这里大概花了 2 天的时间去 debug，主要是它综合一次实在是太慢了，从头开始到下板，大概需要 20 分钟的时间，所以 debug 就很慢，然后中途发现有些错误似乎是因为 vivado 缓存的原因，有时候一直搞不定但是就是有 bug，这时候新建一个工程竟然就好了，感觉自己对 vivado 的理解还不够透彻。