**module** ifu**(**clk**,**reset**,**npc\_sel**,**zero**,**insout**,**jpcnew**);**

**input** clk**,**reset**,**zero**;** //时钟，重置，0（用于比较）

**input** **[**1**:**0**]**npc\_sel**;** //选择t0(addu,subu,ori,lw,sw)还是t1(beq)

**output** **[**31**:**0**]**insout**;** //输出的指令

**output** **[**31**:**0**]**jpcnew**;** //j指令后的新地址

**reg** **[**31**:**0**]**pc**;** //pc计数器

**wire** **[**31**:**0**]** pcnew**;** //pc计数器的新值

**wire** **[**31**:**0**]** t1； //传输beq指令的地址码=(pc+4)接imm16经符号位扩展后的32位后左移2位

**wire** **[**31**:**0**]** t0**;** //addu,subu,ori,lw,sw均采用pc+4的地址码

**wire** **[**31**:**0**]** extout**;** //传输imm16经符号位扩展后的32位后左移2位后的值(beq)

**wire** **[**31**:**0**]** immext**;** //传输imm16经符号位扩展后的32位的值(beq)

**wire** **[**31**:**0**]** jbranch**;** //传输j采用伪直接寻址即为 pc+4后的值+imm16左移2位变为28位数+00 所获得的32位地址码

**wire** **[**31**:**0**]** jpcnew**;** //执行j指令后pc的新值

**wire** **[**15**:**0**]** imm16**;** //16位立即数

**wire** **[**25**:**0**]** imm26**;** //26位立即数

**reg** **[**7**:**0**]** im**[**1023**:**0**];** //声明1KB的im(命令存储器)，8根线，包含1024个的数组

**initial** **begin**

$readmemh**(**"code.txt"**,**im**);** //读取指令

**end**

**assign** insout **=** **{**im**[**pc**[**9**:**0**]],** im**[**pc**[**9**:**0**]+**1**],** im**[**pc**[**9**:**0**]+**2**],** im**[**pc**[**9**:**0**]+**3**]};**

//取指令，一次取8位，连续去四次，合计32位为一条指令

**assign** imm16 **=** insout**[**15**:**0**];** //传输指令第1位至第16位，用于lui,ori,sw,lw的立即数和beq的扩展

**assign** imm26 **=** insout**[**25**:**0**];** //传输指令第1位至第26位，用于j指令的26位的地址码

**assign** immext **=** **{{**16**{**imm16**[**15**]}},**imm16**};**

//beq指令符号扩展，{16{imm16[15]}意为将imm16的第16位(即符号位)扩展16次，{{16{imm16[15]}},imm16}最后在连接imm16

**assign** extout **=** immext**<<**2**;** //beq,j都需要左移2位

**assign** jbranch **=** **{**t0**[**31**:**28**],**imm26**,**2'b0**};**

//j采用伪直接寻址即为 pc+4后的值+imm16左移2位变为28位数+00

**always@(posedge** clk**,posedge** reset**)**

/\*

1、always语句有两种触发方式。第一种是电平触发，例如always @(a or b or c)，a、b、c均为变量，当其中一个发生变化时，下方的语句将被执行。

2、第二种是沿触发，例如always @(posedge clk or negedge rstn)，即当时钟处在上升沿或下降沿时，语句被执行。

\*/

**begin**

**if(**reset**)** pc **=** 32'h0000\_3000**;** //当Reset信号为高电平时（同步），重置PC为0x00003000

**else** pc **=** pcnew**;** //当时钟上升沿到来时将pcnew写入PC内部，并且输出

**end**

**assign** pcnew **=** **(**npc\_sel**==**2'b10**||**npc\_sel**==**2'b11**)?**jbranch**:(**npc\_sel**&&**zero**)?**t1**:**t0**;**

**assign** jpcnew **=** pc**+**8**;** //计算j指令后pc的新值

**assign** t0 **=** pc**+**4**;** //addu,subu,ori,lw,sw均采用pc+4的顺序寻址

**assign** t1 **=** t0**+**extout**;** //计算beq指令的地址码=(pc+4)接imm16经符号位扩展后的32位后左移2位

**endmodule**

**module** DataMemory**(**clk**,**MemWr**,**DataIn**,**Addr**,**DataOut**);**

**input** clk**,**MemWr**;**

**input** **[**31**:**0**]** DataIn**,**Addr**;**

**output** **reg** **[**31**:**0**]**DataOut**;**

**reg** **[**7**:**0**]**memory**[**0**:**1023**];**

**reg** **[**31**:**0**]** address**;**

**integer** i**;**

**initial** **begin**

**for** **(**i**=**0**;** i**<**1024**;** i**=**i**+**1**)**

memory**[**i**]** **<=** 0**;**

**end**

**always** **@(**MemWr **or** DataIn **or** Addr **or** **posedge** clk**)**

**begin**

address **=** Addr **<<** 2**;**

DataOut **=** **(**memory**[**address**]<<**24**)+(**memory**[**address**+**1**]<<**16**)+(**memory**[**address**+**2**]<<**8**)+**memory**[**address**+**3**];**

**if** **(**MemWr **==** 1**)**

**begin**

address **=** Addr **<<** 2**;**

memory**[**address**]** **=** DataIn**[**31**:**24**];**

memory**[**address**+**1**]** **=** DataIn**[**23**:**16**];**

memory**[**address**+**2**]** **=** DataIn**[**15**:**8**];**

memory**[**address**+**3**]** **=** DataIn**[**7**:**0**];**

**end**

**end**

**endmodule**

**module** ext**(**imm**,**ExtOp**,**Extout**);**

**input** **[**15**:**0**]** imm**;**

**input** ExtOp**;**

**output** **[**31**:**0**]**Extout**;**

**assign** Extout**[**15**:**0**]** **=** imm**;**

**assign** Extout**[**31**:**16**]** **=** ExtOp**?(**imm**[**15**]?**16'hffff**:**16'h0000**):**16'h0000**;**

//1为符号扩展

**endmodule**

**module** Control**(**op**,**func**,**RegDst**,**ALUSrc**,**MemtoReg**,**RegWr**,**MemWr**,**nPC\_sel**,**ExtOp**,**ALUctr**,**add8**);**

**input** **[**5**:**0**]**op**,**func**;** //界定具体指令

**output** RegDst**;** //仅有aadu,subu用到rd寄存器才会选择1

**output** ALUSrc**;** //控制ALU的输入是来自寄存器堆还是扩展单元

**output** MemtoReg**;**

**output** RegWr**;**

**output** MemWr**;**

**output** **[**1**:**0**]** nPC\_sel**;**

**output** ExtOp**,**add8**;**

**output** **[**2**:**0**]** ALUctr**;**

**wire** **[**2**:**0**]**ADD**,**Or**,**Subtract**;**

**assign** RegDst **=** **(**op**==**6'b000000**)?**1**:**0**;**

**assign** ALUSrc **=** **(**op**==**6'b000000**||**op**==**6'b000100**)?**0**:**1**;** //addu,subu,beq选择0

**assign** MemtoReg **=** **(**op**==**6'b100011**)?**1**:**0**;** //仅有lw选择1

**assign** RegWr **=** **(**op**==**6'b101011**||**op**==**6'b000100**)?**0**:**1**;** //仅有sw,beq选择0

**assign** MemWr **=** **(**op**==**6'b101011**)?**1**:**0**;** //仅有sw选择0

**assign** nPC\_sel **=** **(**op**==**6'b000011**)?**2'b11**:(**op**==**6'b000010**)?**2'b10**:(**op**==**6'b000100**)?**2'b01**:**2'b00**;**

//依次对应jal,j,beq

**assign** ExtOp **=** **(**op**==**6'b001101**)?**0**:**1**;** //ori指令选择0

**assign** ADD **=** 3'b000**;**

**assign** Subtract **=** 3'b001**;**

**assign** Or **=** 3'b100**;**

**assign** ALUctr **=** **(**op**==**6'b001111**)?**3'b011**:(**op**==**6'b001101**)?**Or**:(**func**==**6'b100011**||**op**==**6'b000100**)?**Subtract**:**ADD**;**

//依次对应lui,ori,subu,addu

**endmodule**

**module** MUX2X5**(**A**,**B**,**Sel**,**Out**);**

//insout[20:16],insout[15:11],RegDst,rw

**input** **[**4**:**0**]**A**,**B**;**

**input** Sel**;**

**output** **[**4**:**0**]**Out**;**

**function** **[**4**:**0**]**select**;**

**input** **[**4**:**0**]**A**,**B**;**

**input** Sel**;**

**case(**Sel**)**

1'b0**:**select **=** A**;**

1'b1**:**select **=** B**;**

**endcase**

**endfunction**

**assign** Out **=** select**(**A**,**B**,**Sel**);**

**endmodule**

**module** MIPS**(**clk**,**reset**);**

**input** reset**,**clk**;**

**wire** **[**31**:**0**]**insout**,**jpcnew**;**

**wire** **[**4**:**0**]**rw**,**rt**;**

**wire** **[**2**:**0**]**ALUctr**;**

**wire** **[**1**:**0**]**nPC\_sel**;**

**wire** RegWr**,**RegDst**,**ExtOp**,**ALUSrc**,**MemWr**,**MemtoReg**,**zero**,**add8**;**

**wire** **[**31**:**0**]**busA**,**busB**,**busO**,**busE**,**busB1**,**DataO**,**busW**;**

**assign** rt **=** **(**nPC\_sel**==**2'b11**)?**5'b11111**:**insout**[**20**:**16**];**

Control cu0**(**insout**[**31**:**26**],**insout**[**5**:**0**],**RegDst**,**ALUSrc**,**MemtoReg**,**RegWr**,**MemWr**,**nPC\_sel**,**ExtOp**,**ALUctr**,**add8**);**

ifu i1**(**clk**,**reset**,**nPC\_sel**,**zero**,**insout**,**jpcnew**);**

MUX2X5 M1**(**insout**[**20**:**16**],**insout**[**15**:**11**],**RegDst**,**rw**);**

RegFile R1**(**clk**,**RegWr**,**busA**,**busB**,**busW**,**insout**[**25**:**21**],**rt**,**rw**,**jpcnew**);**

ext E1**(**insout**[**15**:**0**],**ExtOp**,**busE**);**

MUX2X32 M2**(**busB**,**busE**,**ALUSrc**,**busB1**);**

ALU A1**(**busA**,**busB1**,**ALUctr**,**zero**,**busO**);**

DataMemory D1**(**clk**,**MemWr**,**busB**,**busO**,**DataO**);**

MUX2X32 M3**(**busO**,**DataO**,**MemtoReg**,**busW**);**

**endmodule**

**module** MUX2X32**(**A**,**B**,**Sel**,**Out**);**

//M2: busB,busE,ALUSrc,busB1

//M3: busO,DataO,MemtoReg,busW

**input** **[**31**:**0**]**A**,**B**;**

**input** Sel**;**

**output** **[**31**:**0**]**Out**;**

**function** **[**31**:**0**]**select**;**

**input** **[**31**:**0**]**A**,**B**;**

**input** Sel**;**

**case(**Sel**)**

1'b0**:**select **=** A**;**

1'b1**:**select **=** B**;**

**endcase**

**endfunction**

**assign** Out **=** select**(**A**,**B**,**Sel**);**

**endmodule**

**module** test**;**

**reg** clk**,**reset**;**

MIPS ct1**(**clk**,**reset**);**

**initial**

**begin**

clk **=** 1**;**

reset **=** 0**;**

**#**5 reset **=** 1**;**

**#**5 reset **=** 0**;**

**end**

**always**

**begin**

**#**30 clk**=~**clk**;**

**end**

**endmodule**

**module** RegFile**(**clk**,**RegWr**,**busA**,**busB**,**busW**,**RA**,**RB**,**RW**,**jpcnew**);**

**input** **[**4**:**0**]** RA**;** //相当于rs

**input** **[**4**:**0**]** RB**;** //相当于rt

**input** **[**4**:**0**]** RW**;** //rd或rt

**input** **[**31**:**0**]** busW**;**

**input** **[**31**:**0**]** jpcnew**;** //连接自IFU

**input** clk**;**

**input** RegWr**;**

**output** **[**31**:**0**]** busA**;**

**output** **[**31**:**0**]** busB**;**

**reg** **[**31**:**0**]** register**[**0**:**31**];**

**integer** i**;**

**initial** **begin**

**for** **(**i **=** 0**;** i **<** 32**;** i **=** i**+**1**)**

register**[**i**]** **<=** 0**;**

**end**

**assign** busA **=** register**[**RA**];**

**assign** busB **=** register**[**RB**];**

**always** **@(**RegWr **or** RW **or** busW **or** **posedge** clk**)**

**begin**

**if** **(**RegWr **&&** RW**!=**5'b0**)** //addu,subu需要取rd,并且满足不能是零寄存器

register**[**RW**]** **=** busW**;** //rd = rs +/- rt

**if(**RB**==**5'b11111**)**

register**[**RB**]** **=** jpcnew**;** //rt写入j采用伪直接寻址即为 pc+4后的值+imm16左移2位变为28位数+00 所获得的32位地址码

**end**

**endmodule**

**module** ALU**(**busA**,**busB**,**ALUctr**,**zero**,**Out**);**

**input** **[**31**:**0**]** busA**,**busB**;**

**input** **[**2**:**0**]** ALUctr**;** //可控制lui,ori,subu,addu

**output** **reg** zero**;**

**output** **reg** **[**31**:**0**]**Out**;**

**wire** **[**31**:**0**]**A**;**

**wire** **[**31**:**0**]**B**;**

**assign** B **=** busB**;**

**assign** A **=** busA**;**

**always@(**busA**,**busB**,**ALUctr**,**A**,**B**)**

**begin**

**case(**ALUctr**)** //addu

3'b000**:begin**

Out **=** A**+**B**;**

zero **=** **(**Out**==**0**)?**1**:**0**;**

**end**

3'b001**:begin** //subu

Out **=** A**-**B**;**

zero **=** **(**Out**==**0**)?**1**:**0**;**

**end**

3'b100**:begin** //ori进行按位或运算

Out **=** A**|**B**;**

zero **=** **(**Out**==**0**)?**1**:**0**;**

**end**

3'b011**:begin** //lui把一个16位的立即数填入到寄存器的高16位，写入rt

Out **=** B**<<**16**;**

zero **=** **(**Out**==**0**)?**1**:**0**;**

**end**

**endcase**

**end**

**endmodule**