P6.Verilog 开发流水线 cpu（plus）

一、实验目的

使用 Verilog 设计一个支持 50 条指令的流水线 CPU。

二、设计要求

1.处理器使用 Verilog 设计。

2.处理器能支持 MIPS-C3={LB、LBU、LH、LHU、LW、SB、SH、SW、ADD、ADDU、SUB、SUBU、MULT、MULTU、DIV、DIVU、SLL、SRL、SRA、SLLV、SRLV、SRAV、AND、OR、XOR、NOR、ADDI、ADDIU、ANDI、ORI、XORI、LUI、SLT、SLTI、SLTIU、SLTU、BEQ、BNE、BLEZ、BGTZ、BLTZ、BGEZ、J、JAL、JALR、JR、MFHI、MFLO、MTHI、MTLO}指令。

3.处理器为流水线设计，有独立的控制器和冲突模块。

4.处理器需支持延迟槽。

三、实验设计

一、模块设计

1.PC 模块

（1）基本描述

PC 主要功能是完成地址转移工作，当复位信号有效时，将地址变为首地址，否则维持原有地址不变，将地址传输给指令存储器（IM）。

（2）模块接口

信号名	方向	描述
inp	I	输入地址。
Clk	I	时钟信号。
Reset	I	复位信号。
outp[31:0]	O	当前的 32 位地址。
en	I	使能信号。

（3）功能定义

序号	功能名称	功能描述
1	复位	复位信号有效时，PC 被设置为 0x00003000。
2	转移地址	当复位信号为 0 时，PC 将输入的地址转移给 IM。
3	停止转移	当使能信号为 0 时，PC 值固定不变。

2. IM 模块

（1）基本描述

IM 的主要功能是根据当前的地址，将地址相对应的 32 位数据导出并进行后续操作。

（2）模块接口

信号名	方向	描述
add[31:0]	I	输入当前的地址。
cod[31:0]	O	输出地址所对应的 32 位数据。

（3）功能定义

序号	功能名称	功能描述
1	取指令	根据当前输入的地址，从存储器里取出对应的数据进行操作。

3. GRF 模块

（1）基本描述

GPR 主要功能是利用寄存器以实现对数据的取出和存入操作。通过一个 32 位 MIPS 指令对指令中的指定寄存器的值进行读或写操作。

（2）模块接口

信号名	方向	描述
Clk	I	时钟信号。
Reset	I	复位信号。
Rw	I	寄存器写信号，当信号有效时，可以向寄存器里写入数据。
Rr1[25:21]	I	读入数据的寄存器地址 1。

Rr2[20:16]	I	读入数据的寄存器地址 2。
Wr[4:0]	I	写寄存器的地址。
Wd[31:0]	I	写寄存器的数据。
Rd1[31:0]	O	rr1 中数据的输出。
Rd2[31:0]	O	rr2 中数据的输出。
hidata	I	写入 hi 的数据
lodata	I	写入 lo 的数据
hiwrite	I	hi 写控制信号
lowrite	I	lo 写控制信号
hiout	O	hi 的输出数据
loout	O	lo 的输出数据

（3）功能定义

序号	功能名称	功能描述
1	取数	取出 RS 接口和 RT 接口对应的寄存器中存储的值。
2	写数	将指定的 32 位数据写入指定的寄存器中。

4. ALU 模块

（1）基本描述

ALU 的功能是对指定的两个 32 位数进行加、减、或运算以及对两个数进行大小比较。

（2）模块接口

信号名	方向	描述
I1	I	第一个 32 位数据的输入。
I2	I	第二个 32 位数据的输入。
Aluop[3:0]	I	alu 输出数据的控制信号。
R1	O	运算输出结果
sop	I	逻辑移位位数信号控制

（3）功能定义

序号	功能名称	功能描述
1	运算	取出 RS 接口和 RT 接口对应的寄存器中存储的值。并根据当前的信号进行相对应的运算。

5.DM 模块

（1）基本描述

DM 的功能是对数据存储器里指定的地址里的数据进行读或写操作。

（2）模块接口

信号名	方向	描述
Clk	I	时钟信号。
Reset	I	复位信号。
Addr[31:0]	I	写入或读出数据的地址。
Memw	I	写数据操作信号。
Inp[31:0]	I	写入的数据。
Outp[31:0]	O	读出的数据。
EXop[3:0]	I	对半字或者字节写入的选择操作

（3）功能定义

序号	功能名称	功能描述
1	写数据	将 inp 写入向 addr 指定的地址中。
2	读数据	将 addr 地址中的数据输出到 outp。
3	选择写入部分	通过 EXop 输入的信号，选择相对应的字节进行存储。

6.EXT 模块

（1）基本描述

EXT 的功能是对一个十六位二进制数进行扩展，并根据扩展信号判断进行符号扩展还是零扩展。

（2）模块接口

信号名	方向	描述
Extd[15:0]	I	需要扩展的 16 位数据。
Extop	I	控制输出的信号。
Exo[31:0]	O	扩展完毕的 32 位数据。

（3）功能定义

序号	功能名称	功能描述
1	扩展	根据扩展信号将 16 位数据变为 32 位数据。

7.乘除模块

（1）基本描述

乘除模块的功能是对计算乘除运算。

（2）模块接口

信号名	方向	描述
Ind1	I	第一个 32 位数据
Ind2	I	第二个 32 位数据
Mulop	I	乘除指令的类型。
Clk	I	时钟信号
Reset	I	复位信号
Hi	O	Hi 输出结果
Lo	O	lo 输出结果
Hiw	O	Hi 写信号
Low	O	Lo 写信号

（3）功能定义

序号	功能名称	功能描述
1	乘除运算	当前指令是乘除指令时，计算乘除指令并按照规定延后输出结果。

8.字节选择模块

（1）基本描述

字节选择模块的功能是当前是 store 型指令时选择字节存储。

（2）模块接口

信号名	方向	描述
Op	I	当前指令 op 段
Two	I	地址的后两位

BE	O	存储的形式。

（3）功能定义

序号	功能名称	功能描述
1	选择存储字节	根据当前指令选择存储字节。

9.DM 输出模块

（1）基本描述

DM 输出模块的功能是对根据指令数据存储器里的输出进行操作。

（2）模块接口

信号名	方向	描述
Data	I	来自 dm 的 32 位数据。
Dmop	I	输出控制信号
a	I	地址的后两位。
Out	O	操作后的数据。

10.npc 模块

（1）基本描述

npc 的功能是对下一次指令的地址进行计算。

（2）模块接口

信号名	方向	描述
dpc[31:0]	I	当前的地址。
Imm[25:0]	I	当前地址对应的数据的后 26 位数
Npc_sel	I	判断指令的类型。
Zero	I	判断两个操作数是否相等
Xiao	I	判断 rs 的值是否小于等于 0
Da	I	判断 rs 的值是否大于等于 0
Dra[31:0]	I	31 号寄存器的值
Npc[31:0]	O	下一条指令地址

（3）功能定义

序号	功能名称	功能描述
1	数据操作	根据指令以及地址后两位进行数据操作。

11.control（control0、control1、control2、control3）模块

（1）基本描述

control 模块的功能是对各部分的运算、存储、读取等进行控制。

（2）模块接口

信号名	方向	描述
Op[5:0]	I	六位 op 信号。
fun[5:0]	I	六位 func 信号。
npc_sel	O	下一位地址输出控制。
extop	O	Ext 输出控制。
aluop	O	Alu 计算控制。
sop	O	乘除指令控制。
mulop	O	移位指令控制
alusrc	O	Alu 计算数据控制。
memwrite	O	DM 写控制。
regdst	O	Grf 写地址控制。
memtoreg	O	Grf 写数据控制。
regwrite	O	Grf 写控制。
loadcontrol	O	DM 数据输出选择
hiwrite	O	Hi 写控制
lowrite	O	Lo 写控制

（红色：control0；绿色：control1；蓝色：control2；黄色：control3）

（3）功能定义

序号	功能名称	功能描述
1	总体控制	根据输入信号，判断需要控制的数据，从而得到我们想要的结果。

（4）控制器真值表

见附表

12. hazard 模块

（1）基本描述

由于一条指令处理多个周期，可能会造成不同数据间的冲突，hazard 的作用是处理这些冲突数据，使得流水线能够正常工作。

（2）模块接口

信号名	方向	描述
ird[31:0]	I	If/Id 寄存器中的指令
ire[31:0]	I	Id/Ex 寄存器中的指令。
irm[31:0]	I	Ex/mem 寄存器中的指令。
irw[31:0]	I	mem/wb 寄存器中的指令。
Inadd	I	寄存器写地址
Pcen	O	Pc 使能信号
irden	O	If/Id 寄存器使能信号
Ireclr	O	Id/Ex 寄存器复位信号
FORWARDRSD	O	Id 级 rs 选择信号
FORWARDRTD	O	Id 级 rt 选择信号
FORWARDRSE	O	Ex 级 rs 选择信号
FORWARDRTE	O	Ex 级 rt 选择信号
FORWARDRTM	O	Mem 级 rt 选择信号
FORWARDHI	O	HI 选择信号
FORWARDLO	O	LO 选择信号

（3）功能定义

序号	功能名称	功能描述
1	解决冲突	解决产生冲突的数据。

二、模块间连接

```
module mips(clk,reset);
    input clk;
    input reset;
    wire clk,reset;
    wire [31:0] p1,p2,p3,p4,p5,p6;
    wire d1;
    wire [31:0] hda,lda,hda1,lda1;
    wire [31:0] r1,r2,r3,r4;
    wire [31:0] re1,re2;
    wire [31:0] se1,se2,se3,se4;
    wire [31:0] t1,t2,t3,t4,t5;
    wire [31:0] ts1;
    wire [4:0] a1;
    wire [31:0] a2;
    wire [31:0] u1,u2,u3,u4,u5,u6;
    wire [31:0] s1,s2,s3,s4;
    wire [31:0] m1,m2,m3,m4;
    wire [31:0] hrm,lrm,hrw,lrw;
    wire zero,da,xiao;
    wire en,en2,clr,cc1,cc2;
    wire [2:0] h3,h4,memtoreg;
    wire h5,hbm,hbw,lbm,lbw;
    wire [2:0] lc;
    wire [3:0] h1,npc_sel,h2,h6;
    wire [31:0] b1,b2,b3,b4,b5,b6,hire,lore;
    wire [31:0] c1,c2,c3,c4,c5,c6,c7,c8;
    wire [3:0] ext,aluop;
    wire [1:0]regdst,extop,mulop,hh,ll;
    wire alusrc,regwrite,memwrite;
    wire m,d,hiw,low,bu,sop,hp,lp;
    assign zero = (b1==b2)?1:0;
    assign da = ($signed(b1)>=0)?1:0;
    assign xiao = ($signed(b1)<=0)?1:0;
    PC
QPC(.clk(clk),.reset(reset),.inputadd(p6),.outputadd(p1),.en(en));
    IM QIM(.imadd(p1),.cod(p3));
```

```
    add4 qadd4(.inputadd(p1),.outputadd(p4));
    add4 qqadd4(.inputadd(p4),.outputadd(p5));
    reg1
qreg1(.clk(clk),.reset(reset),.pcd(p1),.pce(m1),.ird(p3),.pc4d(p4),.i
re(r1),.pc4e(r2),.pc8d(p5),.pc8e(r3),.en(en2));
    GRF
QGRF(.hiwrite(hp||hbw),.lowrite(lp||lbw),.hidata(hda),.lodata(lda),.h
io(c1),.lio(c2),.inputadd1(r1[25:21]),.address(m4),.inputadd2(r1[20:1
6]),.outputdata1(re1),.outputdata2(re2),.regwrite(regwrite),.clk(clk)
,.reset(reset),.writeadd(a1),.writedata(a2));
    EXT QEXT(.inputdata(r1[15:0]),.extop(extop),.outputdata(r4));
    NPC
QNPC(.pc(r2),.npc_sel(npc_sel),.ra(b6),.npc(p2),.index(r1[25:0]),.zer
o(zero),.xiaoe(xiao),.dae(da));
    reg2
qreg2(.clk(clk),.hie(c1),.him(c3),.loe(c2),.lom(c4),.reset(reset||clr
),.pce(m1),.pcm(m2),.ire(r1),.pc4e(r2),.rse(re1),.rte(re2),.exte(r4),
.irm(s1),.pc4m(s2),.rsm(se1),.rtm(se2),.extm(s4),.pc8e(r3),.pc8m(s3));
    M1 QM1(.A(b4),.B(s4),.sel(alusrc),.C(se3));
    ALU
QALU(.sop(sop),.s(s1[10:6]),.indata1(b3),.indata2(se3),.aluop(aluop),
.outdata(se4));
    reg3
qreg3(.hirm(hrm),.lorm(lrm),.him(hda1),.lom(lda1),.hm(hbm),.lm(lbm),.
hirw(hrw),.lorw(lrw),.hiw(c5),.low(c6),.hw(hbw),.lw(lbw),.clk(clk),.r
eset(reset),.pcm(m2),.pcw(m3),.irm(s1),.pc4m(s2),.aom(se4),.rtm(b4),.
irw(t3),.pc4w(t4),.aow(t1),.rtw(t2),.pc8m(s3),.pc8w(t5));
    DM
QDM(.EX(ext),.address(t1),.add(m3),.inputdata(b5),.clk(clk),.reset(re
set),.memwrite(memwrite),.outputdata(ts1));
    BEEXT QBEEXT(.op(t3[31:26]),.two(t1[1:0]),.EX(ext));
    MUL
QMUL(.ind1(b3),.ind2(se3),.mulop(mulop),.clk(clk),.reset(reset),.m(m)
,.d(d),.hi(hrm),.lo(lrm),.hiw(hbm),.low(lbm),.b(bu));
    reg4
qreg4(.clk(clk),.hiw(c5),.low(c6),.hid(c7),.lod(c8),.reset(reset),.pc
w(m3),.pcd(m4),.irw(t3),.ird(u3),.pc4w(t4),.pc4d(u4),.aow(t1),.aod(u1
),.drw(ts1),.drd(u2),.pc8w(t5),.pc8d(u5));
```

```
    DMEXT QDMEXT(.data(u2),.a(u1[1:0]),.dmop(1c),.outdata(u6));
    MU5 QM2(.A(u1),.B(u6),.C(u5),.D(c7),.E(c8),.op(memtoreg),.F(a2));
    M3
QM3(.A(u3[20:16]),.B(u3[15:11]),.C(5'b11111),.sel(regdst),.D(a1));
    hazard
qhazard(.busy(bu),.hie(hh),.loe(11),.ird(r1),.ire(s1),.irm(t3),.irw(u
3),.pcen(en),.irden(en2),.ireclr(c1r),.FORWARDRSD(h1),.FORWARDRTD(h2)
,.FORWARDJR(h6),.FORWARDRSE(h3),.FORWARDRTE(h4),.FORWARDRTM(h5),.inad
d(a1));
    MU9
MRSD(.A(re1),.B(s3),.C(c3),.D(c4),.E(t1),.F(t5),.G(c5),.H(c6),.I(a2),
.op(h1),.J(b1));
    MU9
MRTD(.A(re2),.B(s3),.C(c3),.D(c4),.E(t1),.F(t5),.G(c5),.H(c6),.I(a2),
.op(h2),.J(b2));
    MU6
MRSE(.A(se1),.B(t1),.C(t5),.D(c5),.E(c6),.F(a2),.op(h3),.G(b3));
    MU6
MRTE(.A(se2),.B(t1),.C(t5),.D(c5),.E(c6),.F(a2),.op(h4),.G(b4));
    MU9
MJRE(.A(re1),.B(c3),.C(c4),.D(s3),.E(t1),.F(t5),.G(c5),.H(c6),.I(a2),
.op(h6),.J(b6));
    M1 MRTM(.A(t2),.B(a2),.sel(h5),.C(b5));
    MU3 MH(.A(c3),.B(c5),.C(c7),.op(hh),.D(hda1));
    MU3 ML(.A(c4),.B(c6),.C(c8),.op(11),.D(lda1));
    M1 pcM1(.A(p4),.B(p2),.sel(d1),.C(p6));
    M1 MH1(.A(u1),.B(hrw),.sel(hbw),.C(hda));
    M1 ML1(.A(u1),.B(lrw),.sel(lbw),.C(lda));
    control0
qcontrol0(.op(r1[31:26]),.fun(r1[5:0]),.rt(r1[20:16]),.npc_sel(npc_se
l),.extop(extop),.mu(d1));
    control1
qcontrol1(.op(s1[31:26]),.fun(s1[5:0]),.alusrc(alusrc),.aluop(aluop),
.sop(sop),.m(m),.d(d),.mulop(mulop));
    control2
qcontrol2(.op(t3[31:26]),.fun(t3[5:0]),.memwrite(memwrite));
    control3
qcontrol3(.op(u3[31:26]),.ins(u3),.fun(u3[5:0]),.regwrite(regwrite),.
```

```
memtoreg(memtoreg),.regdst(regdst),.loadcontrol(lc),.hiwrite(hp),.low
rite(lp));
endmodule
```

四、Mips 程序测试

```
addi $sp,$0,0
addi $sp,$sp,0x0100
addi $s0,0,0
lui $t1, 0xabcd
sw $t1,0($s0)
addi $s0,$s0,0x0004
bgtz $t1,a
bgez $t1,b
lui $t1,0xefeb
mult $t1,$s0
mflo $s0
sw $t0,0($s1)
addi $s1,$s1,0
b:
slti $t2,$t1,0
beq $t2,$0,4
lui $9,0xdcba
a:
sw $s1, 0($t1)
blez $t1, d
lui $s0,0x3246
d:
lui $s0,0x3246
ori $s0,$s0,0xff3a
lui $s1,0x1889
ori $s1,$s1,0xde51
addu $s2,$s1,$s0
sw $s2,0($t4)
sh $s2,4($t4)
sb $s2,8($t4)
sb $s2,12($t4)
sw $s2,16($t4)
```

```
lui $s0,0x2345
ori $s0,$s0,0x1111
lui $s1,0x0321
ori $s2,$s1,0x6666
lui $s3,0x2344
sw $t7,0($sp)
addu $sp,$sp,$28
sw $17,0($29)
addu $29,$29,$2
ori $s3,$s3,0x6dea
lui $s4,0xaed1
subu $s1,$s2,$s3
addu $s4,$s1,$s2
and $s7,$s2,$s3
ori $s3,$s1,$s3
addu $s5,$s2,$s1
beq $s5,$s7,loop
subu $s1,$s2,$s3
addu $s7,$s1,$s3
addu $s5,$s2,$s1
beq $s5,$s7,loop
subu $s1,$s2,$s3
lui $t1,0x8888
nor $s6,$s1,$s2
subu $s1,$s2,$s3
lw $s1,0($t7)
beq $s1,$s3,loop2
lui $t1,0xac05
lui $t2,0x5388
ori $t1,$t1,0x2333
beq $t1,$t2,loop2
addu $s7,$s2,$s1
sllv $s1,$s2,$s7
lh $s1,0($t7)
bgez $s1 ,loop2
addu $t0,$s1,$s2
ori $s3,$t0,0x0003
beq $s4,$s3,loop2
```

```
addiu $t2,$s1,0x4132
lw $s1,0($t7)
srav $t5,$s1,$s3
xori $s2,$s1,0x2453
lw $s2,0($t7)
beq $s1,$s3,loop2
ori $t3,$s2,0x3946
add $t0,$s1,$t3
addi $t2,$s4,0x3946
lui $t1,0x8888
j loop
sw $s0,0($t7)
sllv $s1,$s2,$s3
lw $s1,0($t7)
beq $s1,$s3,loop2
sh $s1,4($t7)
sw $s2,8($t7)
loop2:
sw $s3,12($t7)
lh $s1,0($t7)
jalr $s1,$s5
lw $s1,0($t7)
addiu $t5,$s1,0x42ae
beq $s1,$s3,loop2
addu $t6,$s1,$t0
andi $s2,$s1,0x2453
lw $s2,0($t7)
ori $s2,$zero,0x0004
addu $t7,$t8,$s2
sw $t6,0($t7)
jr $ra
loop:
sw $k0,100($a0)
lw $s4,4($k0)
jal loo
add $s6,$s5,$s4
jal end
loo:
```

```
lw $s2,0($t7)
addu $s2,$s2,$k0
ori $s4,$s1,0x0004
subu $t9,$s4,$s2
beq $s2,$t9,loop
beq $zero,$zero,loop2
nop
end:
sw $s4,8($t4)
lh $s4,4($t4)
addu $s4,$s4,$s1
sub $s4,$s4,$s3
```

机器码：201d0000 23bd0100 22100000 3c09abcd ae090000 22100004 1d200009
05210005 3c09efeb 01300018 00008012 ae280000 22310000 292a0000
1140fffe 3c09dcba ad310000 19200001 3c103246 3c103246 3610ff3a
3c111889 3631de51 02309021 ad920000 a5920004 a1920008 a192000c
ad920010 3c102345 36101111 3c110321 36326666 3c132344 afaf0000
03bce821 afb10000 03a2e821 36736dea 3c14aed1 02538823 0232a021
02339825 0251a821 12b70029 02538823 0233b821 0251a821 12b70025
02538823 3c098889 0232b027 02538823 8df10000 12330015 3c09ac05
3c0a5388 35292333 112a0011 0251b821 02f28804 02324021 35130003
1293000c 262a4132 8df10000 02336821 12330008 364b3946 022b4020
228a3946 3c098888 08000c57 adf00000 a5f10004 adf20008 adf3000c
85f10000 02a08809 02287021 32322453 8df20000 34120004 03127821
adee0000 03e00008 ac9a0064 8f540004 0c000c5c 02b4b020 0c000c63
8df20000 025a9021 36340004 0292c823 1259fff6 1000ffeb 00000000
ad940008 85940004 0291a021 0293a022

预期结果：

0x4ad0dd8b	0x4ad0dd8a	0x00000004	0x03216666	0x4ad0dd8b
0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0xffffffff
$t1	9	0x80000000
$t2	10	0xffffff00
$t3	11	0x00000017
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x4ad0dd8a
$t7	15	0x00000004
$s0	16	0x23451111
$s1	17	0x4ad0dd8b
$s2	18	0x4ad0dd8a
$s3	19	0x03216666
$s4	20	0x47af54af
$s5	21	0x03216666
$s6	22	0x032243f1
$s7	23	0x03216666
$t8	24	0x00000000
$t9	25	0xb52f2279
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x00001800
$sp	29	0x00000110
$fp	30	0x00000000
$ra	31	0x000032dc
pc		0x00003304
hi		0x00000000
lo		0x00000000

实际结果：

000032DC	00000000
00000110	00000000
00000000	00000000
B52F2279	00000000
03216666	032243F1
03216666	47AF54AF
03216666	4AD0DD8A
4AD0DD8B	23451111
00000004	4AD0DD8A
00000000	00000000
00000017	FFFFFF00
80000000	FFFFF00000
00000000	00000000
00000000	00000000
00000000	00000000
00000000	00000000

00000000	00000000
00000000	4AD0DD8B
03216666	00000004
4AD0DD8A	4AD0DD8B

五、思考题

1.为什么需要有单独的乘除法部件而不是整合进 ALU？为何需要有独立的 HI、LO 寄存器？

在实际的乘除运算中，按照乘法规则，需要先进行多位与 1 位的乘除运算，计算出需要所有相乘除的结果后再进行加减操作，与 ALU 中的其他操作不同，ALU 中的结果立即就可以得到，而乘除运算的结果不可能一次得到，所以需要单独定义模块

2.参照你对延迟槽的理解，试解释“乘除槽”。

在执行乘除法的时候，为了提高流水线的效率，可以执行一些与乘除法无关的指令。

3.为何上文文末提到的 lb 等指令使用的数据扩展模块应在 MEM/WB 之后，而不能在 DM 之后?

在处理数据冲突时可能会更方便，因为 load 型指令的转发要在 W 级进行。

4.举例说明并分析何时按字节访问内存相对于按字访问内存性能上更有优势。（Hint： 考虑 C 语言中字符串的情况）

在 c 语言中，一个字符代表着一个字节，在存储字符的情况下，存储字节比存储字要方便，如果要存储字，则需要把四个字符拼接起来，如果不够四个字符还要有别的操作，如果要存储字节，则一次只需要存储一个字节，不会产生额外的操作。

5.如何概括你所设计的 CPU 的设计风格？为了对抗复杂性你采取了哪些抽象和规范手段？

规划者型风格

设计多路选择器的控制信号，分类需要寄存器数据的指令，和需要向寄存器里存储数据的指令，尽力把每一种情况想的周到，把每一个数据的来源进行认真思考。还需要思考一些普通的地方是否需要考虑转发。

6.你对流水线 CPU 设计风格有何见解？

控制者风格：尽管简单，但是不容易理解，处理冲突还需要额外的工作量。

规划者风格：尽管易于理解，但是实际的代码实现比控制者的要高很多。而且在新增指令的时候也要考虑很多地方是否需要修改。

7. 在本实验中你遇到了哪些不同指令组合产生的冲突？你又是如何解决的？相应的测试样例是什么样的？请有条理的罗列出来。

暂停解决：

B 与 R 类型指令在 Ex 上产生的冲突：add $s5,$s2,$s1　bne $s5,$s7,loop

B 与 I 类型指令在 Ex 上产生的冲突：andi $s3,$s2,0x0003 beq $s4,$s3,loop2

Beq 与 lw 类型指令在 Ex 上产生的冲突：lh $s1,0($t7)

bgez $s1 ,loop2

Beq 与 lw 类型指令在 mem 上产生的冲突：lw $s1,0($t7)

addiu $t5,$s1,0x42ae

beq $s1,$s3,loop2

R 指令与 lw 类型指令在 Ex 上产生的冲突：sub $s1,$s2,$s3

lb $s1,0($t7)

I 指令与 l 类型指令在 Ex 上产生的冲突：xori $s2,$s1,0x2453

lw $s2,0($t7)

s 指令与 l 指令在 Ex 上产生的冲突：sb $k0,101($a0)

lh $s4,4($k0)

转发解决：

B 与 R 类型指令在 mem 上产生的冲突 rs：sllv $s1,$s2,$s3

lw $s1,0($t7)

beq $s1,$s3,loop2

B 与 I 类型指令在 mem 上产生的冲突 rs：lui $t1,0x5388

ori $t1,$t1,0x2333

beq $t1,$t2,loop

B 与 R 类型指令在 wb 上产生的冲突 rt：subu $s7,$s2,$s3

addu $s3,$s1,$s3

and $s5,$s2,$s1

beq $s5,$s7,loop

B 与 I 类型指令在 wb 上产生的冲突 rt：lui $t1,0xac05

lui $t2,0x5388

ori $t1,$t1,0x2333

bne $t1,$t2,loop2

B 与 lw 指令在 wb 上产生的冲突 rs：lb $s2,0($t7)

ori $s4,$s1,0x0004

sub $t9,$s4,$s2

bltz $s2,$t9,loop

R 类型指令与 R 类型指令在 mem 上的冲突 rs：　　subu $s1,$s2,$s3

addiu $s7,$s1,0xad21

R 类型指令与 i 类型指令在 mem 上的冲突 rt：　　addu $t0,$s1,$s2

ori $s3,$t0,0x0003

R 与 R 在 wb 上的冲突 rt：　　subu $s1,$s2,$s3

addu $s7,$s1,$s3

srav $s5,$s2,$s1

R 与 I 在 wb 上的冲突 rt：　　addu $s2,$s2,$k0

andi $s4,$s1,0x0004

subu $t9,$s4,$s2

sw 指令与 R 类型指令在 wb 上产生的冲突：addu $t7,$t8,$s2

sb $t6,0($t7)

addu $t6,$s1,$t0

ori $s2,$s1,0x2453

sw 指令与 I 类型指令在 wb 上产生的冲突：addu $t7,$t8,$s2

sb $t6,0($t7)

xori $t6,$s1,0x3125

ori $s2,$s1,0x2453