计算机系统结构课程实验
总结报告

实验题目：动、静态流水线设计与性能对比分析

学号：1750844

姓名：周展田

指导教师：喻剑

日期：2019.12.1

一、实验环境部署与硬件配置说明

代码编辑：VScode

IDE: Vivado 2017

仿真工具: Vivado2017, Modelsim10.4

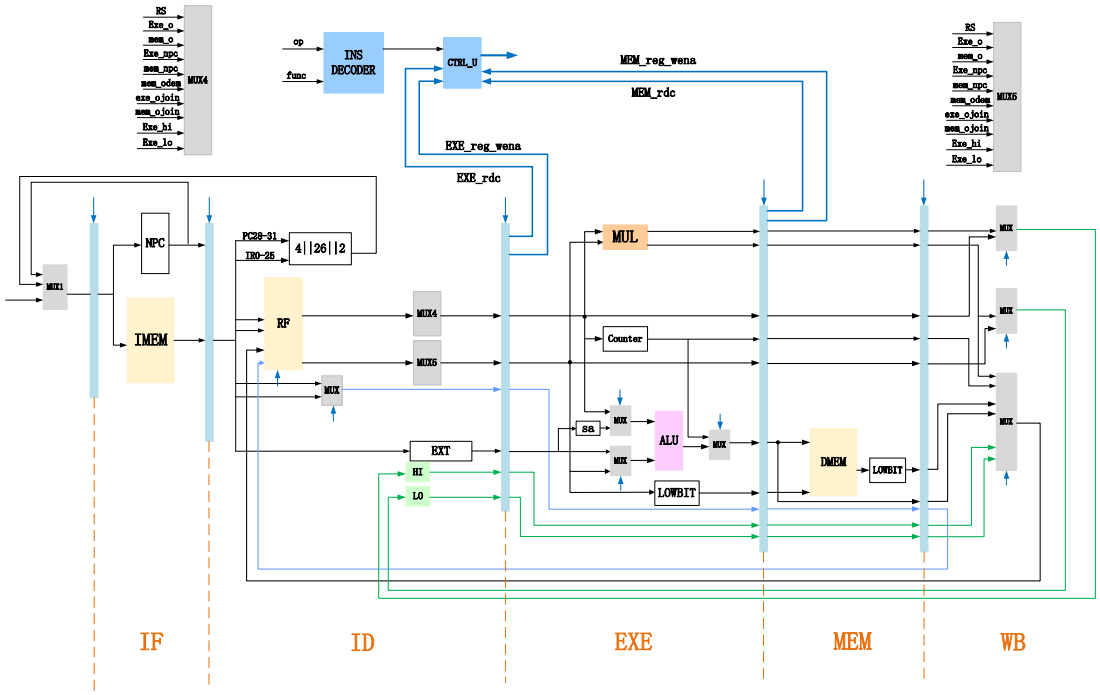
操作系统：Windows10 64 位

开发板：Xilinx N4

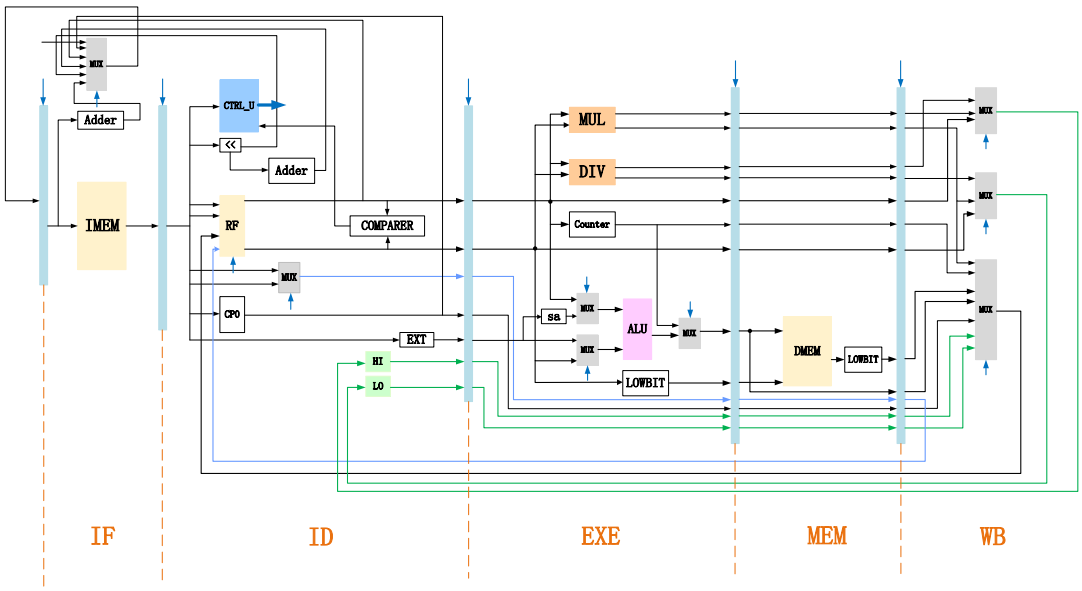
绘图工具：Visio

二、实验的总体结构

1、 动态流水线的总体结构



2、 静态流水线的总体结构

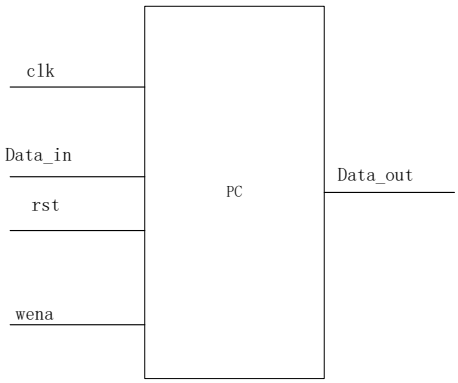


三、总体架构部件的解释说明

1、 动态流水线总体结构部件的解释说明

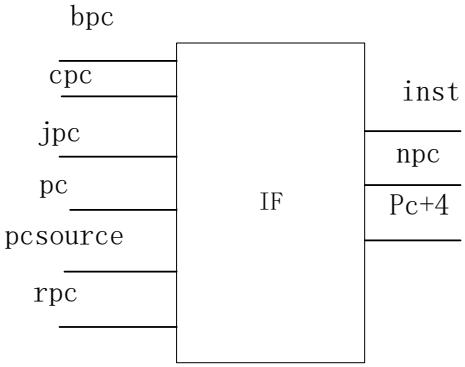
(1)PCreg

PC 寄存器



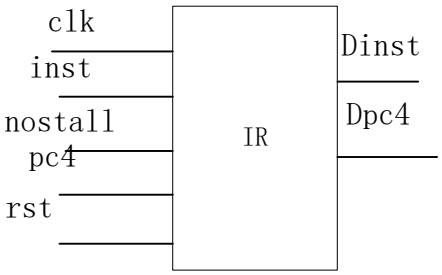
(2)pipeIF

IF 段 ， 输入：PC 各种来源，输出：下一条 PC 和指令



(3)pipeIR

IF 级与 ID 级间的流水寄存器存放取出的指令和 PC+4



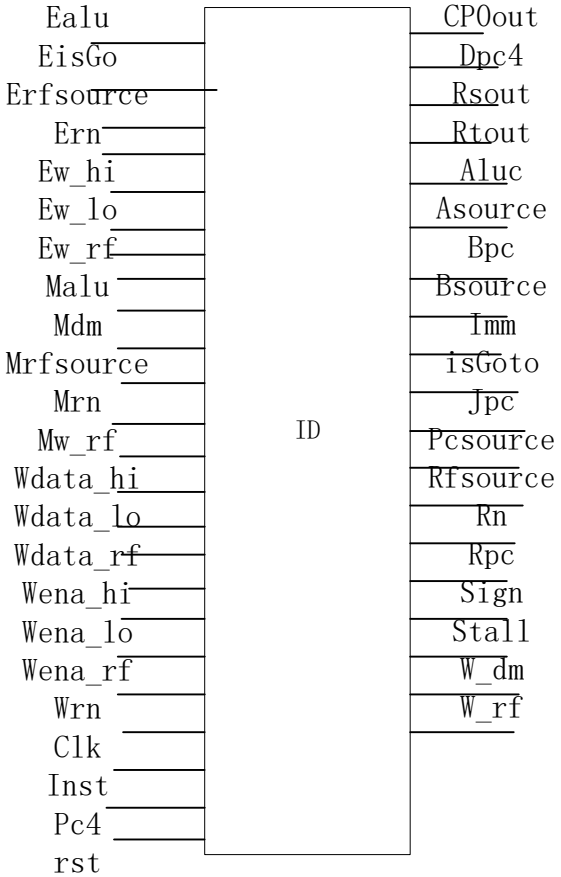
(4)pipeID

ID 级部件

包含了控制单元 CU、寄存器堆、CP0 寄存器组、Hi 寄存器、Lo 寄存器、用于分支指令的比较器、 用于立即数扩展的扩展器、用于计算转移地址的加法器以及多路选择器

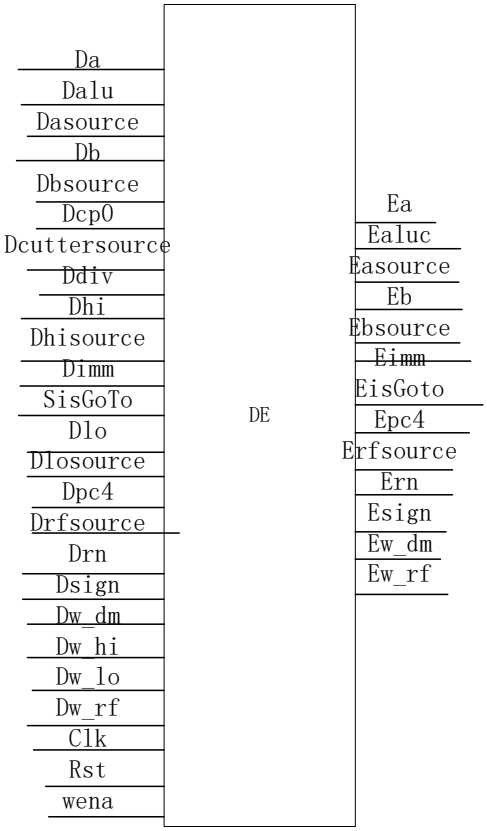
输入：WB 级传回写信号、写地址和写数据，IF 级传递的值

输出：各类控制信号、向 EXE 级传递的各类寄存器读出的值



(5)pipeDEreg

ID 级与 EXE 级间的流水寄存器，传递 ID 级输出的控制信号和读出的数据

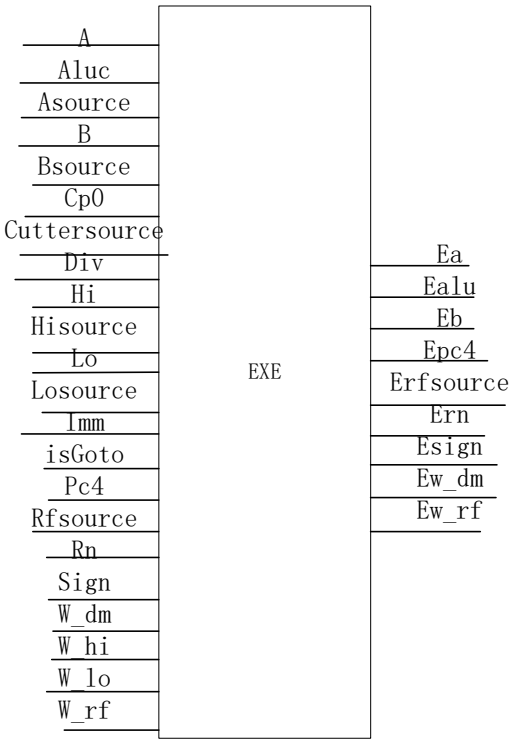


(6)pipeEXE

EXE 段，包含了 ALU 模块、乘法器模块、除法器模块、计算前导零的计算模块和多路选择器。

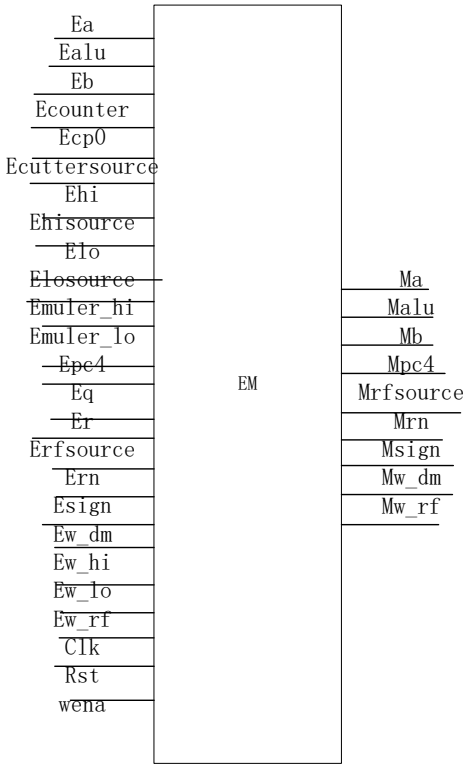
输入：ID 级传递的控制信号以及各类源操作数值

输出：向 MEM 级传递的控制信号以及计算的结果



(7)pipeEMreg

EXE 级与 MEM 级间的流水寄存器，存放 EXE 级产生的计算结果和传递的各类控制信号

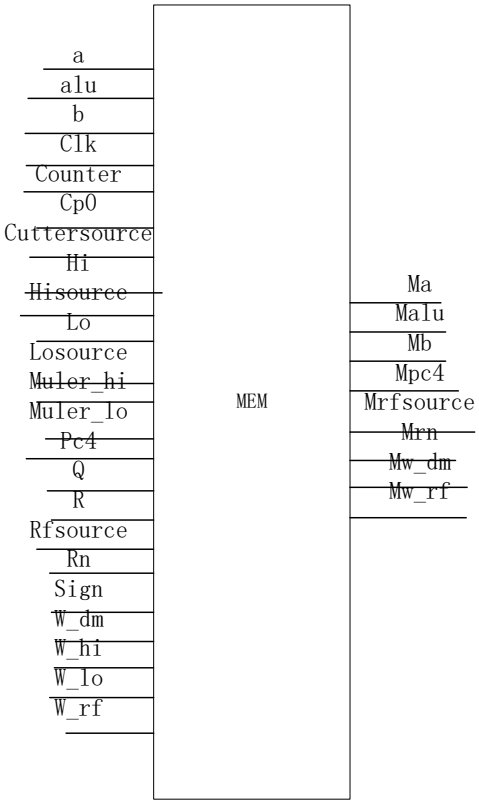


(8)pipeMEM

包含了数据存储器模块、选择数据长度的模块和多路选择器。

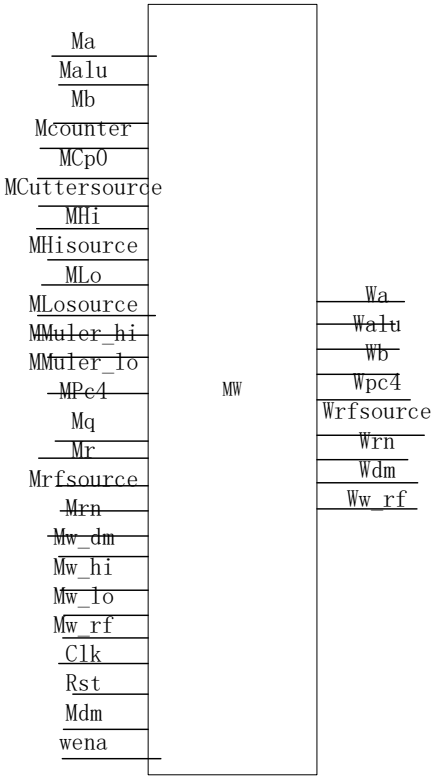
输入：EXE 级的计算结果和传递的控制信号

输出：传递的控制信号与读出的结果。



(9)pipeMWreg

MEM 级与 WB 级间的流水寄存器 ，存放控制信号和各类待写入的结果数值

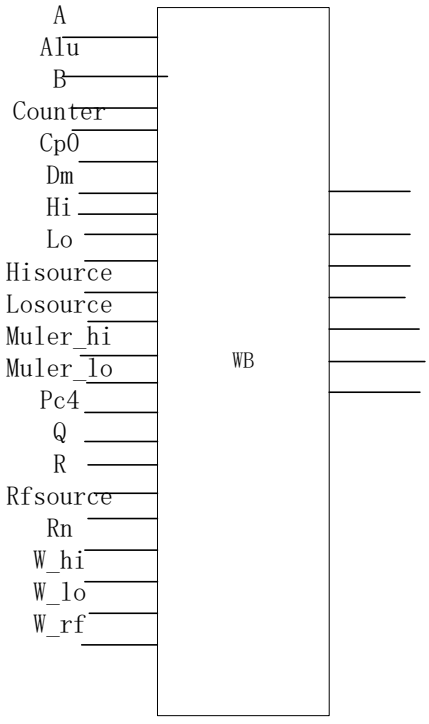


(10)pipeWB

包含了数据存储器模块、选择数据长度的模块和多路选择器。

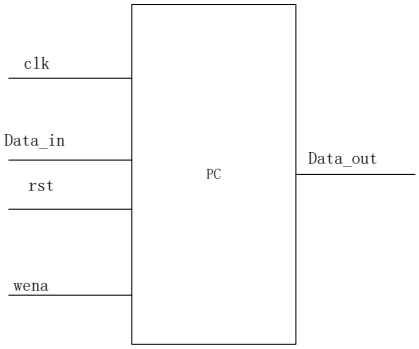
输入：EXE 级的计算结果和传递的控制信号

输出：传递的控制信号与读出的结果。



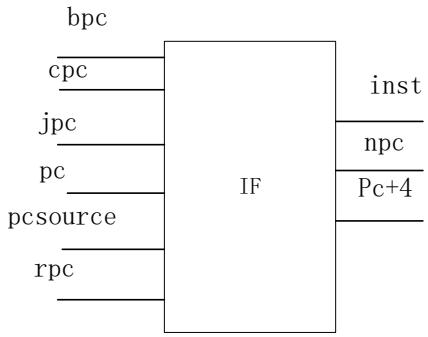
2、 静态流水线总体结构部件的解释说明

(1)PCreg



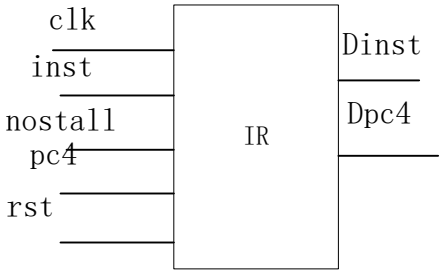
(2)pipeIF

IF 段，输入 PC 各种来源，输出下一条 PC 和指令



(3)pipeIR

IF 级与 ID 级间的流水寄存器存放取出的指令和 PC+4



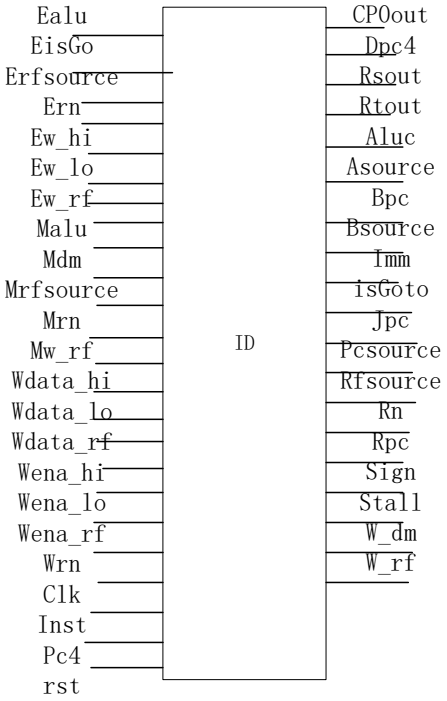
(4)pipeID

ID 级部件

包含了控制单元 CU、寄存器堆、CP0 寄存器组、Hi 寄存器、Lo 寄存器、用于分支指令的比较器、 用于立即数扩展的扩展器、用于计算转移地址的加法器以及多路选择器

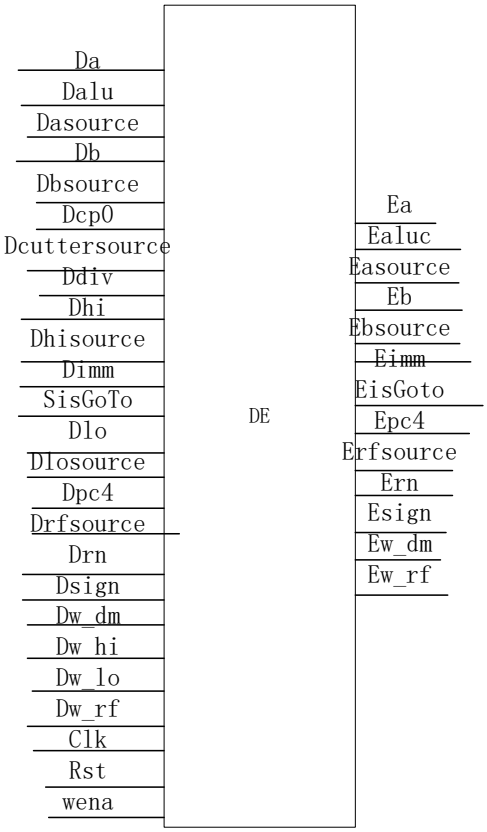
输入：WB 级传回写信号、写地址和写数据，IF 级传递的值

输出：各类控制信号、向 EXE 级传递的各类寄存器读出的值



(5)pipeDEreg

ID 级与 EXE 级间的流水寄存器，传递 ID 级输出的控制信号和读出的数据

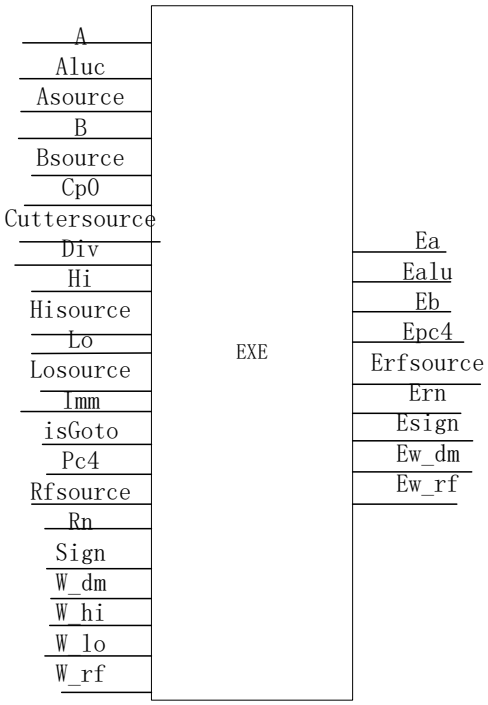


(6)pipeEXE

EXE 段，包含了 ALU 模块、乘法器模块、除法器模块、计算前导零的计算模块和多路选择器。

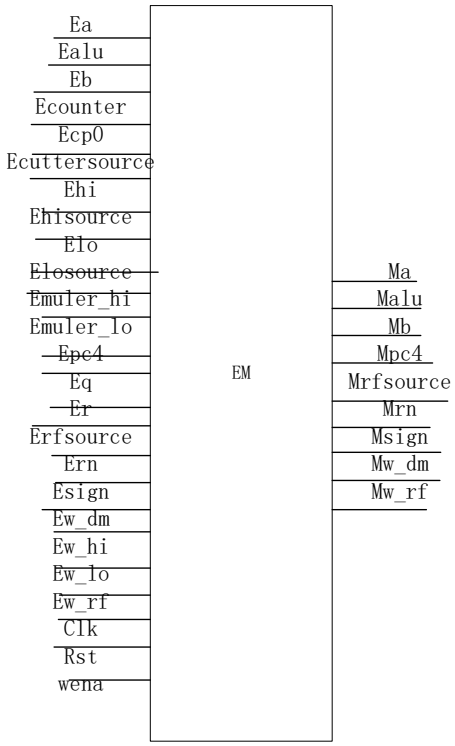
输入：ID 级传递的控制信号以及各类源操作数值

输出：向 MEM 级传递的控制信号以及计算的结果



(7)pipeEMreg

EXE 级与 MEM 级间的流水寄存器，存放 EXE 级产生的计算结果和传递的各类控制信号

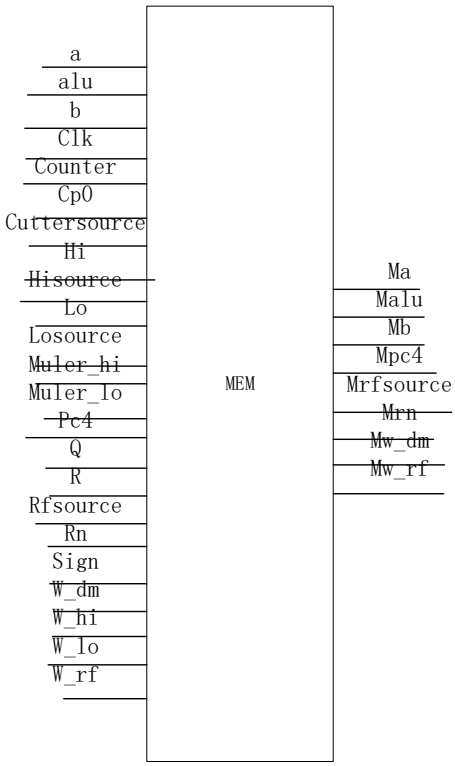


(8)pipeMEM

包含了数据存储器模块、选择数据长度的模块和多路选择器。

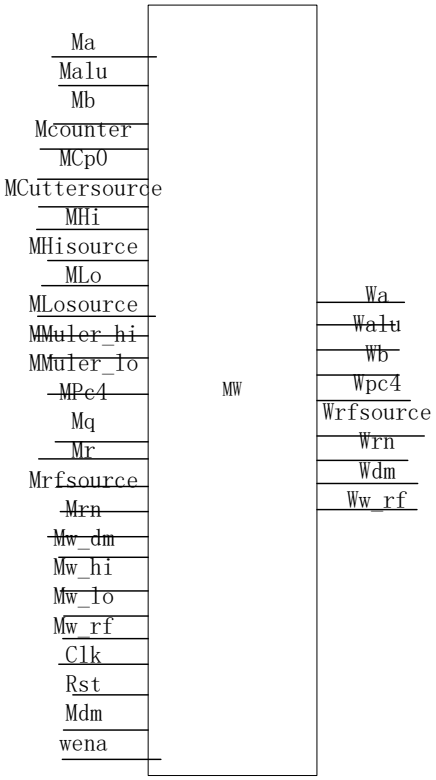
输入：EXE 级的计算结果和传递的控制信号

输出：传递的控制信号与读出的结果。

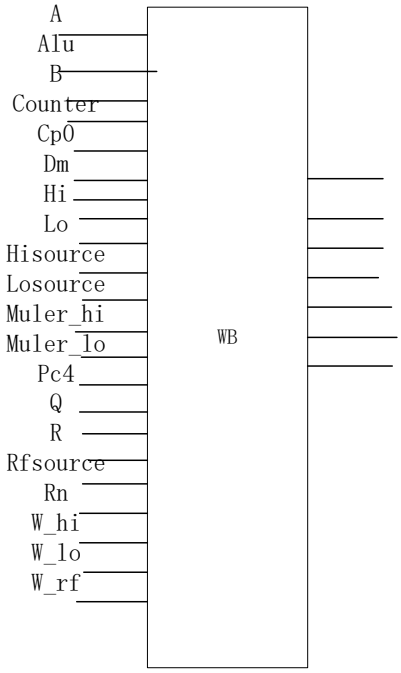


(9)pipeMWreg

MEM 级与 WB 级间的流水寄存器 ， 存放控制信号和各类待写入的结果数值



(10)pipeWB



四、实验仿真过程

1、 动态流水线的仿真过程

(1) 在 Mars 中编写编译汇编程序，导出 16 进制文件

Edit Execute

test.asm

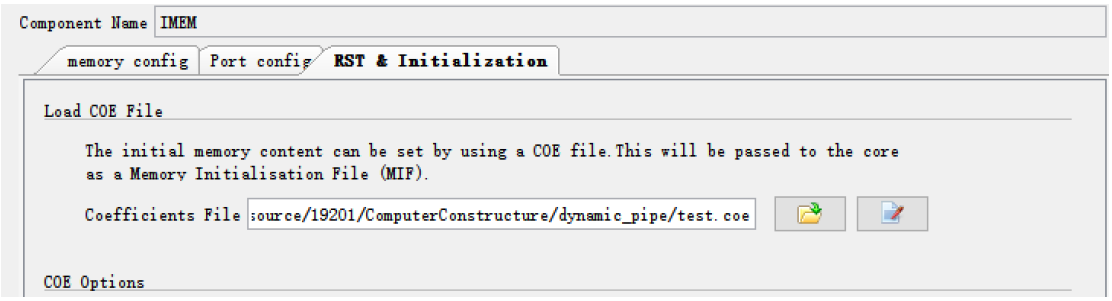
```
.data
A:.space 240
B:.space 240
C:.space 240
D:.space 240
E:.space 240

.text
j main
exc:
nop
j exc
 main:
addi $2,$0,0 #a[i]
addi $3,$0,1 #b[i]
addi $4,$0,0 #c[i]
addi $13,$0,0 #d[i]
addi $5,$0,4 #counter
addi $6,$0,0 #a[i-1]
addi $7,$0,1 #b[i-1]
addi $10,$0,0 #flag for i<20 || i<40
addi $11,$0,240 #sum counts
addi $14,$0,3
addi $29,$0,2#delay counts
addi $30,$0,0
sw $2,A($0)
sw $3,B($0)
sw $2,C($0)
sw $3,D($0)
```

test.coe

E: > 19201 > test.coe

```
08100003
00000000
08100001
20020000
20030001
20040000
200d0000
20050004
20060000
20070001
200a0000
200b00f0
200e0003
201d0002
201e0000
3c011001
00200821
ac220000
3c011001
00200821
ac2300f0
```

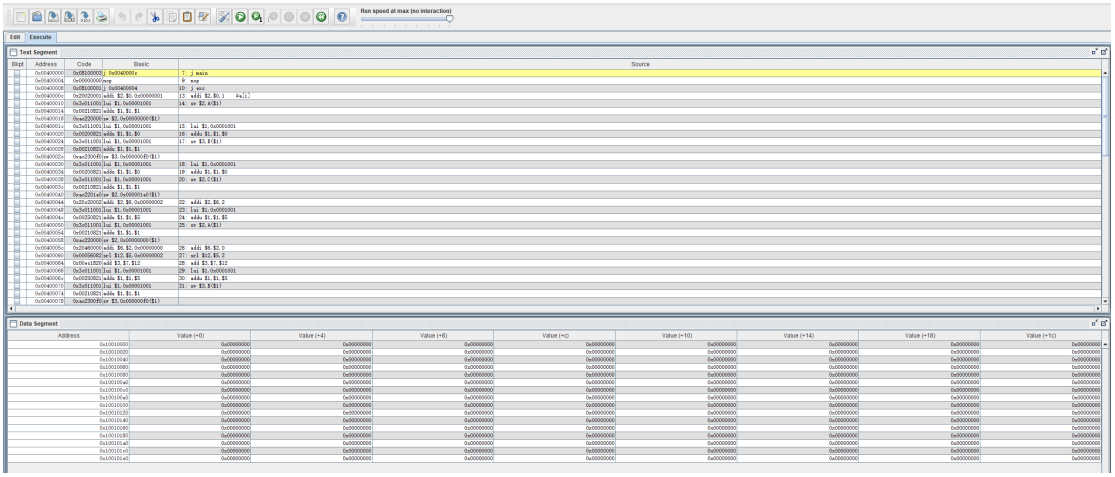
(2)导入到 IP core 中



(3)添加 testbench 文件，进行仿真

2、 静态流水线的仿真过程

(1)在 Mars 中编写编译汇编程序，导出 16 进制文件

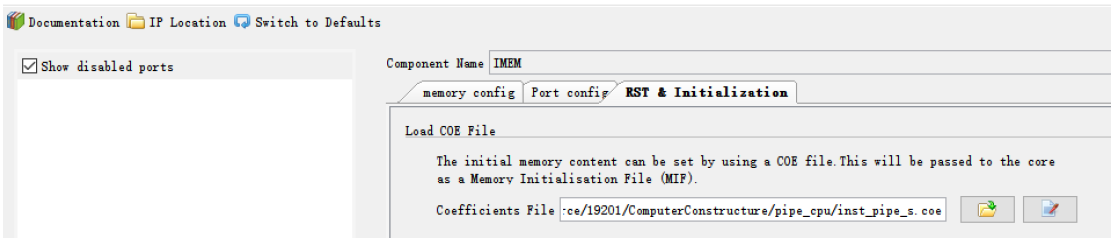


inst_pipe.coe - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
memory_initialization_radix = 16;
memory_initialization_vector =
08100003
00000000
08100001
20020001
3c011001
00210821
ac220000
3c011001
00200821
3c011001
00210821
ac2300f0
3c011001
```

(2)导入到 IP core 中，进行仿真验证

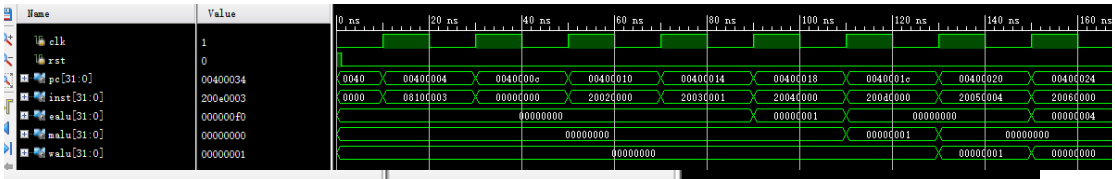


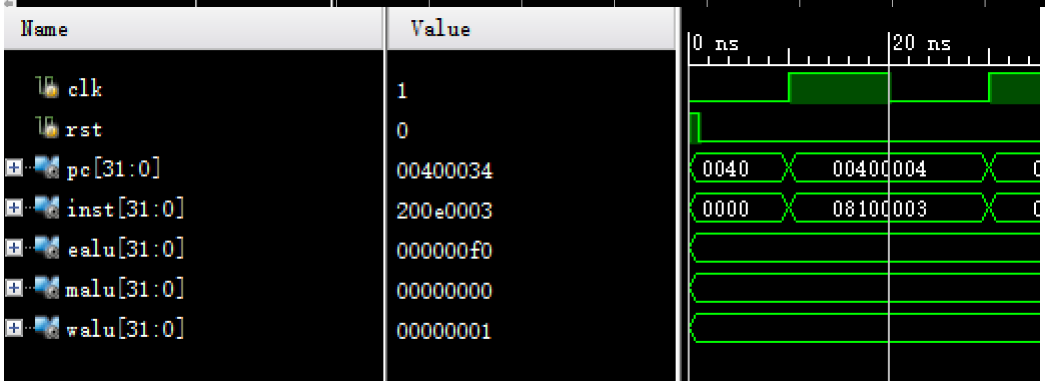
(3)添加 testbench 文件进行仿真

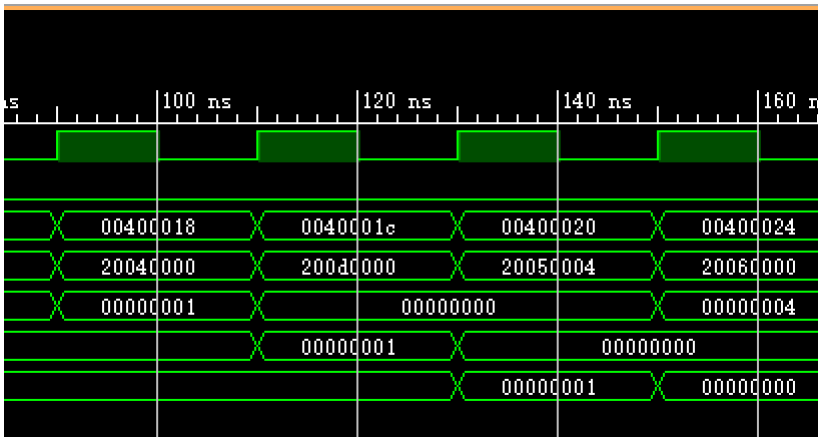
五、实验仿真的波形图及某时刻寄存器值的物理意义

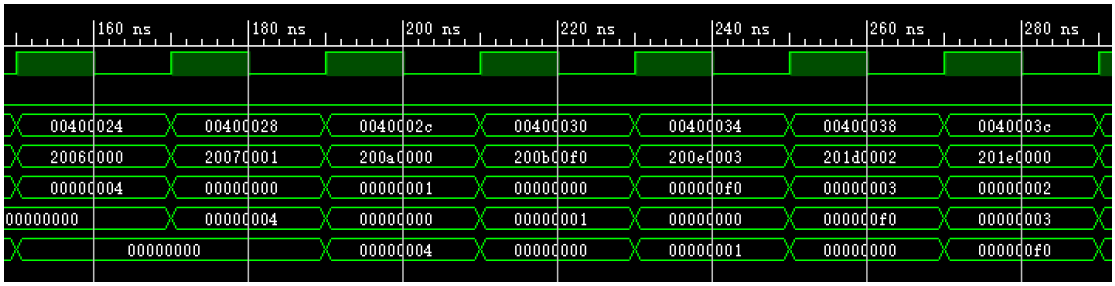
1、 动态流水线的波形图及某时刻寄存器值的物理意义

波形显示了 pc 的值和当前指令，以及各个阶段 alu 的结果





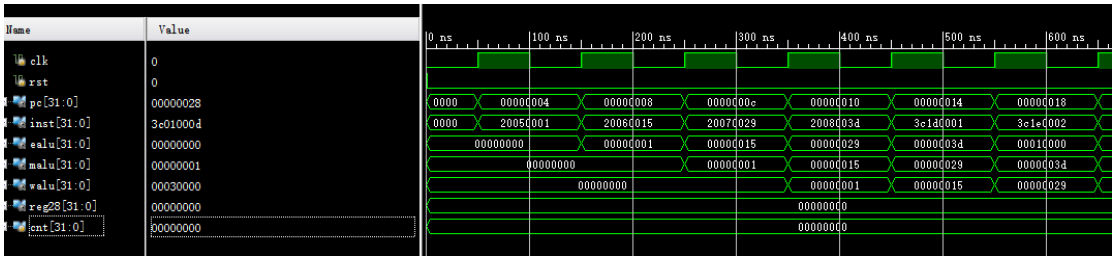




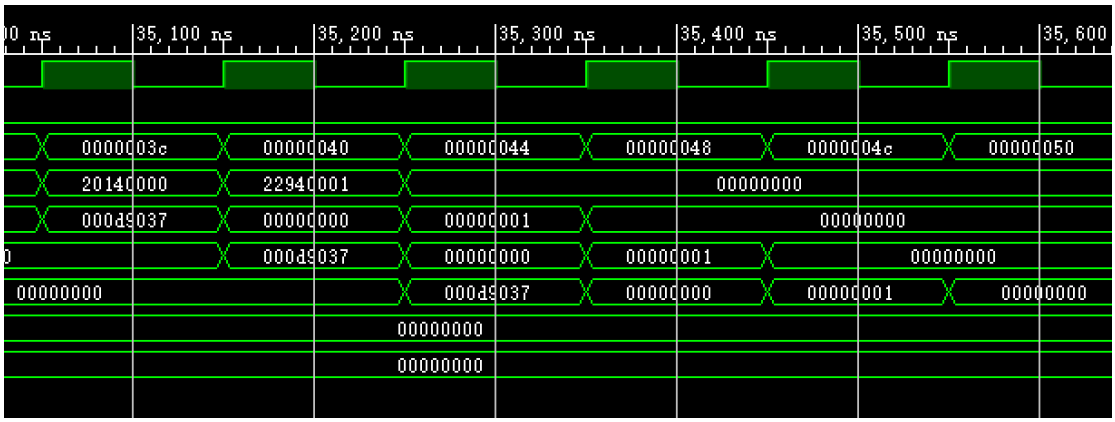
从波形图中可以看到，alu 的结果向下一级流动

2、 静态流水线的波形图及某时刻寄存器值的物理意义

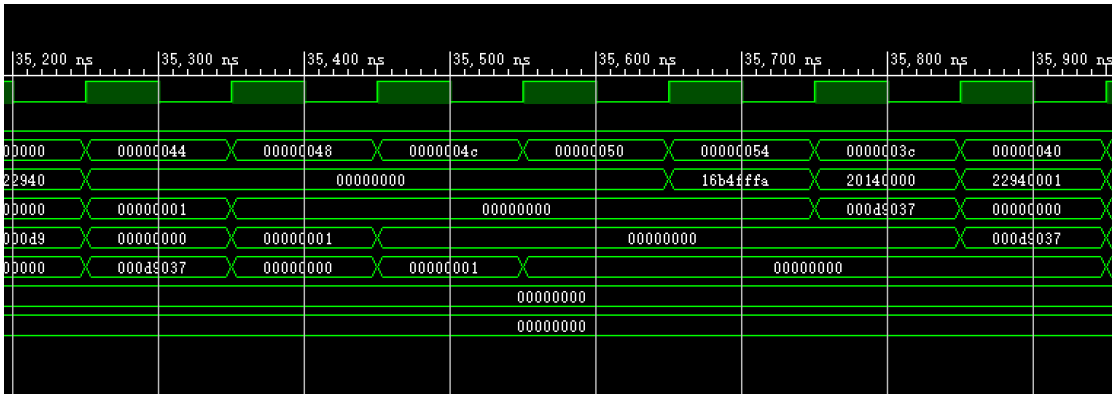
波形显示了 pc 的值和当前指令，以及各个阶段 alu 的结果



alu 的值在不同流水段流动：



流水线排空后执行另一种操作：



六、实验验算数学模型及算法程序

int a[m],b[m],c[m],d[m];

a[0]=0;

b[0]=1;

a[i]=a[i-1]+i;

b[i]=b[i-1]+3i;

$$c[i]=\begin{cases} a[i], & 0\le i\le 19 \\ a[i]+b[i], & 20\le i\le 39 \\ a[i]*b[i], & 40\le i\le 59 \end{cases}$$

$$d[i]=\begin{cases} b[i], & 0\le i\le 19 \\ a[i]*c[i], & 20\le i\le 39 \\ c[i]*b[i], & 40\le i\le 59 \end{cases}$$

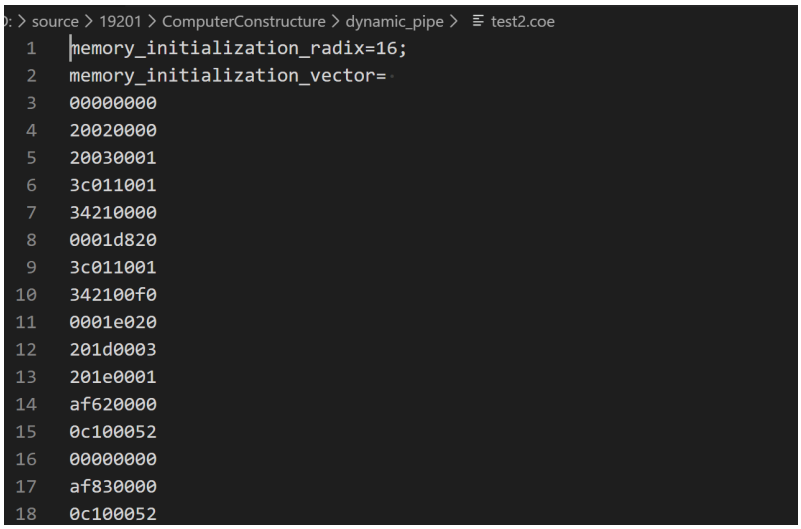
1.编写汇编程序如下：

```
sll $0, $0, 0
addi $2, $0, 0    # a[0]=0
addi $3, $0, 1    # b[0]=1; $4 存储数组 c, $5 存储数组 d；$6 存储 3i
addi $27, $0, 0x10010000    #数组 c 存储的起始地址为 0
addi $28, $0, 0x100100f0    #数组 d 存储的起始地址为 240
addi $29, $0, 3
addi $30, $0, 1      #i

sw $2,0($27)
jal display
sll $0,$0,0
sw $3,0($28)
jal display
sll $0,$0,0
addi $27, $27, 4
addi $28, $28, 4

addi $26, $0, 19      #循环次数
loop1:
add $2, $2, $30       #计算 a[i]=a[i]=a[i-1]+i;
mult $29, $30            #计算 3*i;
mflo $6                  #$6=3*i;
add $3,$3,$6             #计算 b[i]=b[i]=b[i-1]+3*i;
addi $30, $30, 1      #i++
sw $2,0($27)
jal display
sll $0,$0,0
```

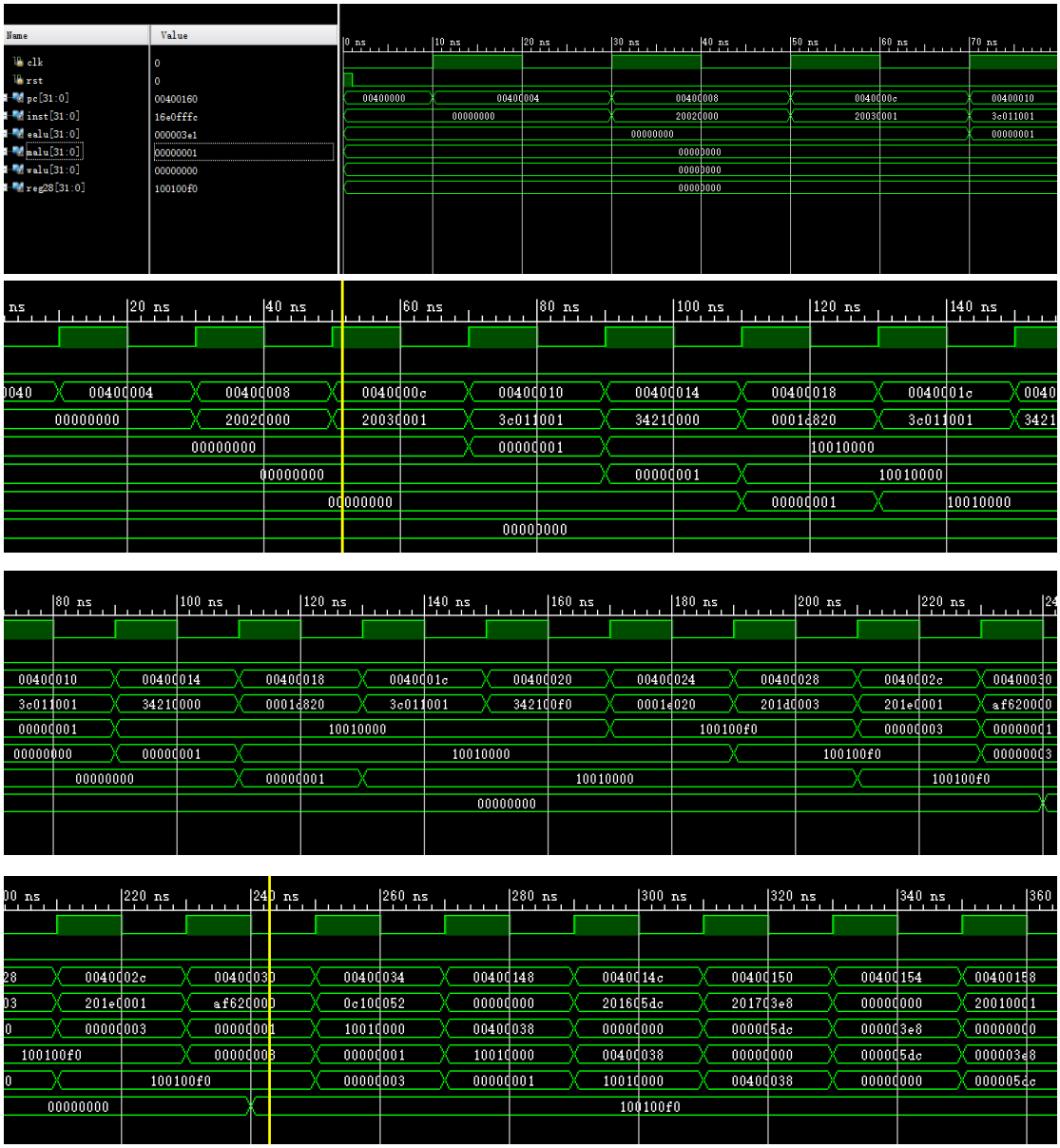
```
sw $3,0($28)
jal display
sll $0,$0,0
addi $27, $27, 4
addi $28, $28, 4
subi $26,$26,1
bne $26,$0,loop1
sll $0, $0, 0

addi $26, $0, 20      #循环次数
loop2:
add $2, $2, $30       #计算 a[i]=a[i]=a[i-1]+i;
mult $29, $30           #计算 3*i;
mflo $6                  #$6=3*i;
add $3,$3,$6             #计算 b[i]=b[i]=b[i-1]+3*i;
add $4, $2, $3          #计算 c[i]=a[i]+b[i];
mult $4, $2
mflo $5
addi $30, $30, 1      #i++
sw $4,0($27)
jal display
sll $0,$0,0
sw $5,0($28)
jal display
sll $0,$0,0
addi $27, $27, 4
addi $28, $28, 4
subi $26,$26,1
bne $26,$0,loop2
sll $0, $0, 0

addi $26, $0, 20      #循环次数
loop3:
add $2, $2, $30       #计算 a[i]=a[i]=a[i-1]+i;
mult $29, $30           #计算 3*i;
mflo $6                  #$6=3*i;
add $3,$3,$6             #计算 b[i]=b[i]=b[i-1]+3*i;
mult $2, $3
mflo $4
mult $4, $3
mflo $5
addi $30, $30, 1      #i++
sw $4,0($27)
jal display
```

```
sll $0,$0,0
sw $5,0($28)
jal display
sll $0,$0,0
addi $27, $27, 4
addi $28, $28, 4
subi $26,$26,1
bne $26,$0,loop3
sll $0, $0, 0
j end
sll $0, $0, 0

display:
addi $22,$0,0x05dc
loop_for_display2:
addi $23,$0,0x03e8
loop_for_display1:
sll         $0,$0,0
subi $23,$23,0x0001
bne    $23,$0,loop_for_display1
sll         $0,$0,0
subi $22,$22,0x0001
bne $22,$0,loop_for_display2
sll         $0,$0,0
jr         $31
sll $0,$0,0
end:
sll $0,$0,0
j end
sll $0,$0,0
```

2.导出为 coe



3.添加到 IP core 中，进行仿真



七、实验验算程序下板测试过程与实现

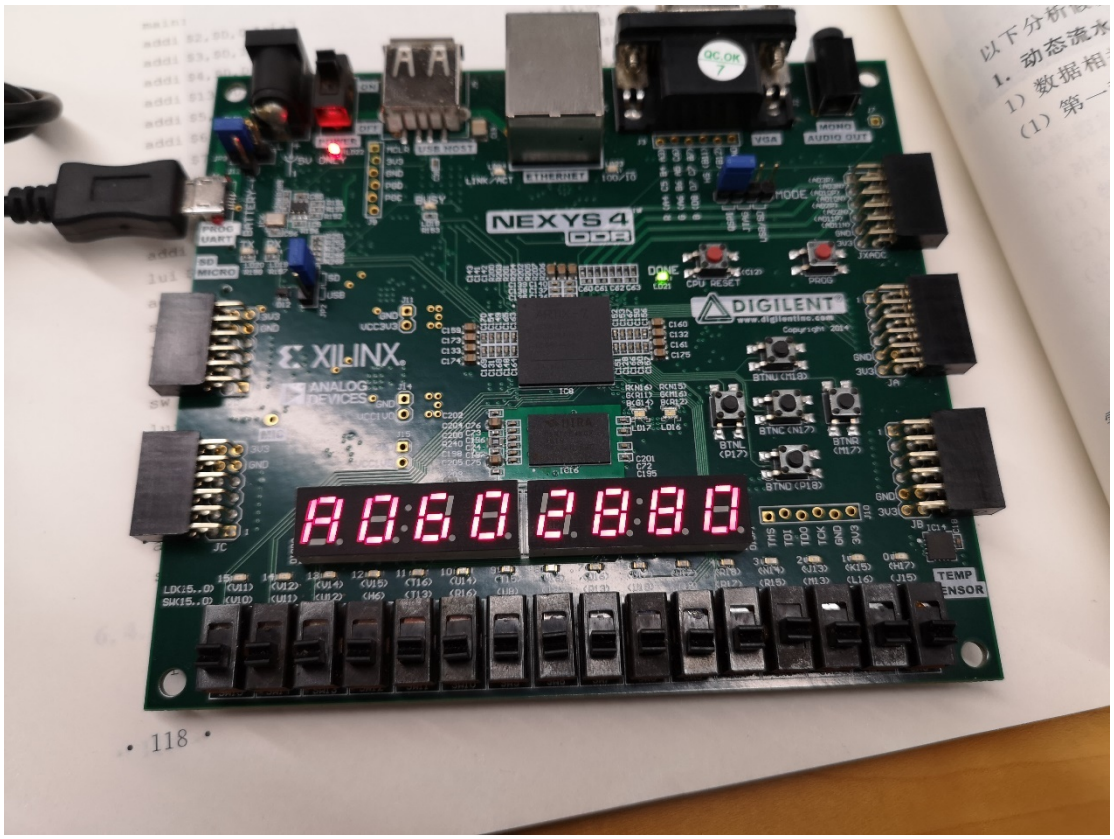
1.编写顶层文件接口和数码管接口

top.v 和 seg7x16.v

2.配置时钟分频，此处使用了 IP core clk_wiz

3.配置约束文件

4.下板结果：



八、流水线的性能指标定性分析（包括：吞吐率、加速比、效率及相关与冲突分析）

1、 动态流水线的性能指标定性分析

(1)吞吐率：

(14+230+298+323)/950=91.05%

(2)效率：

(4*(12+11*19+13*20+1+13*20+1)+2*59+3*(3*20+2*20+19))/950*5

=(2972+118+357)/4750=72.56%

(3)加速比：

和单周期 CPU 相比：863*5/950=4.54

和静态流水线 CPU 相比：2850/950=3.00

(4)冲突分析：

数据冲突：采用数据定向法解决数据冲突。将所有在 WB 级要写回寄存器的数据定向到 ID 级，通过一个多路选择器选择相应的数据作为寄存器读出的数据。

控制冲突：使用延迟槽解决数据冲突，延迟槽统一使用无关指令 sll $0,$0,0。

部件冲突：

存储器的数据冲突：将数据和指令分开，分别用两个存储器存储。同一部件的读写冲突采用上升沿读，下降沿写的方式解决。

2、 静态流水线的性能指标定性分析

(1)吞吐率：

(12+225+299+318)/2850=33.41%

(2)效率：

3648/2850*5=25.60%

(3)加速比：

和单周期 CPU 相比：854*5/2850=1.498

(4)冲突分析：

指令冲突：

不同连接方式的指令因为执行方式的不同，所需要的计算资源也不一样，因此需要将指令按照连接方式分类，通用类指令之间按流水线方式执行，不同类的指令之间需要等待。

数据冲突：

静态流水线 CPU 只需要考虑同一种连接方式的指令之间的数据冲突。

采用数据定向法解决数据冲突，将 EXE 级的 ALU 运算结果和 MEM 级的 ALU 运算结果牵一根线回到 ID 级，ID 级使用一个多路选择器选择相应的数据存入 ID/EXE 流水线寄存器中。（图中红色虚线框）

控制冲突：

在分支指令下加一个延迟槽即可，延迟槽指令统一使用 sll $0,$0,0。

部件冲突：

存储器的数据冲突：将数据和指令分开，分别用两个存储器存储。同一部件的读写冲突采用上升沿读，下降沿写的方式解决。

3、两者性能差异分析

1.性能差异比较

	静态流水线	动态流水线
吞吐率	91.05%	33.41%
效率	72.56%	25.60%
加速比 （与单周期相比）	4.54	1.498

2.分析：

当遇到不同连接方式的指令交替出现的时候，静态流水线 CPU 需要停顿，其效率接近单周期 CPU 的效率。当连续出现同一种连接方式的指令时，静态流水线 CPU 不需要停顿，其执行效率接近动态流水线的效率。动态流水线 CPU 不需要考虑指令的连接方式，无论指令连接方式是否一致，统一按流水线方式执行，不需要停顿。

九、总结与体会

本次实验完成了动态流水线 CPU 的改造，对动态流水线有了更深的理解，通过对两种不同流水线的分析，对流水线的理论计算有了更多认识，对加速比、吞吐率、相关冲突等概念理解更加深入。

十、附件（所有程序）

1、 动态流水线的设计程序

```
//下板的顶层模块
module top(clk,rst,o_seg,o_sel);
    input clk               ;//时钟
    input rst               ;//复位
    output [7:0] o_seg      ;//七段数码管的显示
    output [7:0] o_sel      ;
    wire [31:0] reg28;
    wire locked;
    wire clk_o1,clk_o2;
    clk_wiz_0 c(.clk_in1(clk),.reset(rst),.locked(locked),.clk_out1(clk
_o1));
    seg7x16 s(.clk(clk),.reset(rst&(~locked)),.cs(1'b1),.i_data(reg28),
.o_seg(o_seg),.o_sel(o_sel));
    pipeCPU CPU(.clk(clk_o1),.rst(rst&(~locked)),.reg28(reg28));
endmodule

//CPU
module pipeCPU(clk,rst,reg28,
    _pc,_inst,_ealu,_malu,_walu
);
    input clk;
    input rst;
    output [31:0]reg28;
    //test
    output [31:0]_pc;
    output [31:0]_inst;
    output [31:0]_ealu;
    output [31:0]_malu;
    output [31:0]_walu;
```

```
    wire [31:0]npc,pc4,pc;
    wire stall;
    //PC
    pcreg PC(.clk(clk),.data_in(npc),.rst(rst),.wena(~stall),.data_out(
pc));

    wire [31:0] bpc,jpc,rpc,cpc;
    wire [31:0] inst;
    wire [2:0] pcsource;
    //IF
    pipeIF IF(.bpc(bpc),.cpc(),.jpc(jpc),.rpc(rpc),
              .pc(pc),.pcsource(pcsource),.inst(inst),
              .npc(npc),.pc4(pc4));
    wire [31:0] Dinst_in,Dpc4_in;
    wire [31:0] Dpc4_out;
    //IR
    pipeIR IR(.clk(clk),.pc4(pc4),.inst(inst),
              .nostall(~stall),.rst(rst),
              .Dinst(Dinst_in),.Dpc4(Dpc4_in));
    wire EisGoto;
    wire [ 4:0] Ern;
    wire Ew_hi,Ew_lo,Ew_rf;
    wire [ 4:0] Mrn;
    wire Mw_rf;
    wire [31:0] Wdata_hi,Wdata_lo,Wdata_rf;
    wire Wena_hi,Wena_lo,Wena_rf;
    wire [ 4:0] Wrn;
    wire [31:0] Dcp0out;
    wire [31:0] Hiout,Loout,Rsout,Rtout;
    wire [ 3:0] aluc;
    wire asource,bsource;
    wire [1:0] cuttersource;
    wire [31:0] imm;
    wire isGoto;
    wire [ 4:0] rn;
    wire [2:0] Drfsource;
    wire sign;
    wire w_dm,w_hi,w_lo,w_rf;
    wire [31:0]Ealu_o,Malu_o;
    wire [ 2:0] Erfsource_o,Mrfsource_o;
    wire [31:0] Ern_o;
```

```
    wire [ 4:0] Mrn_o;
    wire [31:0] Malu;
    wire [31:0] Mdm_o;

    wire [31:0] Emuler_hi,Emuler_lo;
    wire [31:0] Mmuler_hi,Mmuler_lo;
    wire [31:0] Emuler_hi_o,Emuler_lo_o;
    wire [31:0] Mmuler_hi_o,Mmuler_lo_o;

    //ID
    pipeID ID(.EisGoto(EisGoto),.Ern(Ern_o),
              .Ew_hi(Ew_hi),.Ew_lo(Ew_lo),.Ew_rf(Ew_rf),
              .Emuler_hi(Emuler_hi_o),.Emuler_lo(Emuler_lo_o),
              .Mrn(Mrn_o),.Mw_rf(Mw_rf),
              .Wdata_hi(Wdata_hi),.Wdata_lo(Wdata_lo),.Wdata_rf(Wdata_r
f),
              .Wena_hi(Wena_hi),.Wena_lo(Wena_lo),.Wena_rf(Wena_rf),
              .Wrn(Wrn),
              .Ealu(Ealu_o),.Malu(Malu),
              .Mdm(Mdm_o),
              .Mmuler_hi(Mmuler_hi_o),.Mmuler_lo(Mmuler_lo_o),
              .Erfsource(Erfsource_o),.Mrfsource(Mrfsource_o),
              .clk(clk),.inst(Dinst_in),.pc4(Dpc4_in),.rst(rst),
              .CP0out(Dcp0out),
              .Dpc4(Dpc4_out),
              .Hiout(Hiout),.Loout(Loout),.Rsout(Rsout),.Rtout(Rtout),
              .aluc(aluc),.asource(asource),.bsource(bsource),
              .bpc(bpc),.cpc(),
              .cuttersource(),
              .div(),
              .reg28(reg28),
              .hisource(),
              .imm(imm),.isGoto(isGoto),
              .jpc(jpc),
              .losource(),.pcsource(pcsource),
              .rfsource(Drfsource),.rn(rn),.rpc(rpc),
              .sign(sign),.stall(stall),
              .w_dm(w_dm),.w_hi(),.w_lo(),.w_rf(w_rf));

    wire [4:0]Ealuc;
    wire [31:0] Ea,Eb,Eimm,Epc4;
    wire Easource,Ebsource,Esign,Ew_dm;
```

```
    wire [2:0] Erfsource_in;

    //DEreg
    pipeDEreg DE(.Daluc(aluc),
                 .Da(Rsout),.Db(Rtout),
                 .Dasource(asource),.Dbsource(bsource),
                 .Dcp0(Dcp0out),.Dcuttersource(),
                 .Ddiv(),
                 .Dhi(),.Dhisource(),
                 .Dimm(imm),
                 .DisGoto(isGoto),
                 .Dlo(),.Dlosource(),
                 .Dpc4(Dpc4_out),
                 .Drfsource(Drfsource),.Drn(rn),
                 .Dsign(sign),
                 .Dw_dm(w_dm),
                 .Dw_hi(w_hi),.Dw_lo(w_lo),
                 .Dw_rf(w_rf),
                 .clk(clk),.rst(rst),.wena(),
                 .Ealuc(Ealuc),
                 .Ea(Ea),.Eb(Eb),
                 .Easource(Easource),.Ebsource(Ebsource),
                 .Ecp0(),.Ecuttersource(),
                 .Ediv(),
                 .Ehi(),.Ehisource(),
                 .Eimm(Eimm),
                 .EisGoto(EisGoto),
                 .Elo(),.Elosource(),
                 .Epc4(Epc4),
                 .Erfsource(Erfsource_in),.Ern(Ern),
                 .Esign(Esign),
                 .Ew_dm(Ew_dm),
                 .Ew_hi(),.Ew_lo(),
                 .Ew_rf(Ew_rf));


    wire [31:0] Ea_o,Eb_o;
    wire EisGoto_o;
    wire [31:0] Epc4_o;
    wire Esign_o,Ew_dm_o,Ew_rf_o;

    //EXE
```

```
    pipeEXE EXE(.aluc(Ealuc),.a(Ea),.b(Eb),
                .asource(Easource),.bsource(Ebsource),
                .cp0(),.cuttersource(),
                .div(),
                .hi(),.hisource(),
                .imm(Eimm),
                .isGoto(EisGoto),
                .lo(),.losource(),
                .pc4(Epc4),
                .rfsource(Erfsource_in),.rn(Ern),
                .sign(Esign),
                .w_dm(Ew_dm),.w_hi(),.w_lo(),.w_rf(Ew_rf),
                .Ealu(Ealu_o),.Ea(Ea_o),.Eb(Eb_o),
                .Ecounter(),
                .Ecp0(),.Ecuttersource(),
                .Ehi(),.Ehisource(),
                .EisGoto(EisGoto_o),
                .Elo(),.Elosource(),
                .Emuler_hi(Emuler_hi_o),.Emuler_lo(Emuler_lo_o),
                .Epc4(Epc4_o),
                .Eq(),.Er(),
                .Erfsource(Erfsource_o),.Ern(Ern_o),
                .Esign(Esign_o),
                .Ew_dm(Ew_dm_o),.Ew_hi(),.Ew_lo(),.Ew_rf(Ew_rf_o));

    wire [31:0] Ma,Mb,Mcounter,Mcp0,Mpc4;
    wire [ 1:0] Mcuttersour;
    wire [31:0] Mhi,Mlo;
    wire [ 1:0] Mhisource,Mlosource;
    wire [31:0] Mq,Mr;
    wire [ 2:0] Mrfsource_in;
    wire Msign,Mw_dm,Mw_hi,Mw_lo;

    //EMreg
    pipeEMreg EM(.Ealu(Ealu_o),
                 .Ea(Ea_o),.Eb(Eb_o),
                 .Ecounter(),
                 .Ecp0(),
                 .Ecuttersource(),
                 .Ehi(),.Ehisource(),
                 .Elo(),.Elosource(),
                 .Emuler_hi(Emuler_hi_o),.Emuler_lo(Emuler_lo_o),
```

```
                .Epc4(Epc4_o),
                .Eq(),.Er(),
                .Erfsource(Erfsource_o),.Ern(Ern_o),
                .Esign(Esign_o),
                .Ew_dm(Ew_dm_o),
                .Ew_hi(),.Ew_lo(),
                .Ew_rf(Ew_rf_o),
                .clk(clk),.rst(rst),.wena(),
                .Malu(Malu),
                .Ma(Ma),.Mb(Mb),
                .Mcounter(),
                .Mcp0(),
                .Mcuttersource(),
                .Mhi(),.Mhisource(),
                .Mlo(),.Mlosource(),
                .Mmuler_hi(Mmuler_hi),.Mmuler_lo(Mmuler_lo),
                .Mpc4(Mpc4),
                .Mq(),.Mr(),
                .Mrfsource(Mrfsource_in),.Mrn(Mrn),
                .Msign(Msign),
                .Mw_dm(Mw_dm),
                .Mw_hi(),.Mw_lo(),
                .Mw_rf(Mw_rf));

    wire [31:0] Ma_o,Mb_o,Mcounter_o,Mcp0_o,Mpc4_o;
    wire [ 1:0] Mcuttersour_o;
    wire [31:0] Mhi_o,Mlo_o;
    wire [ 1:0] Mhisource_o,Mlosource_o;
    wire [31:0] Mq_o,Mr_o;
    wire Msign_o,Mw_hi_o,Mw_lo_o,Mw_rf_o;

    //MEMreg
    pipeMEMreg MEM(.clk(clk),
               .alu(Malu),.a(Ma),.b(Mb),
               .counter(),
               .cp0(),.cuttersource(),
               .hi(),.hisource(),
               .lo(),.losource(),
               .muler_hi(Mmuler_hi),.muler_lo(Mmuler_lo),
               .pc4(Mpc4),.q(),.r(),
               .rfsource(Mrfsource_in),.rn(Mrn),
               .sign(Msign),
```

```
            .w_dm(Mw_dm),.w_hi(),.w_lo(),.w_rf(Mw_rf),
            .Malu(Malu_o),.Ma(Ma_o),.Mb(Mb_o),
            .Mcounter(),.Mcp0(),
            .Mdm(Mdm_o),
            .Mhi(),.Mhisource(),
            .Mlo(),.Mlosource(),
            .Mmuler_hi(Mmuler_hi_o),.Mmuler_lo(Mmuler_lo_o),
            .Mpc4(Mpc4_o),.Mq(),.Mr(),
            .Mrfsource(Mrfsource_o),.Mrn(Mrn_o),
            .Msign(Msign_o),
            .Mw_hi(),.Mw_lo(),.Mw_rf(Mw_rf_o));

wire [31:0] Walu,Wa,Wb;
wire [31:0] Wcounter_Wcp0,Wpc4;
wire [ 1:0] Wcuttersour;
wire [31:0] Whi,Wlo;
wire [ 1:0] Whisource,Wlosource;
wire [31:0] Wmuler_hi,Wmuler_lo;
wire [31:0] Wq,Wr;
wire [ 2:0] Wrfsource;
wire [ 4:0] Wrn_in;
wire [31:0] Wdm;
wire Wsign,Ww_hi,Ww_lo,Ww_rf;

//MWreg
pipeMWreg MW(.Malu(Malu_o),
             .Ma(Ma_o),.Mb(Mb_o),
             .Mcounter(),
             .Mcp0(),
             .Mcuttersource(),
             .Mdm(Mdm_o),
             .Mhi(),.Mhisource(),
             .Mlo(),.Mlosource(),
             .Mmuler_hi(Mmuler_hi_o),.Mmuler_lo(Mmuler_lo_o),
             .Mpc4(Mpc4_o),.Mq(),.Mr(),
             .Mrfsource(Mrfsource_o),.Mrn(Mrn_o),
             .Mw_hi(),.Mw_lo(),.Mw_rf(Mw_rf_o),
             .clk(clk),.rst(rst),.wena(),
             .Walu(Walu),
             .Wa(Wa),.Wb(Wb),
             .Wcounter(),
             .Wcp0(),
```

```
                .Wdm(Wdm),
                .Whi(),.Whisource(),
                .Wlo(),.Wlosource(),
                .Wmuler_hi(Wmuler_hi),.Wmuler_lo(Wmuler_lo),
                .Wpc4(Wpc4),
                .Wq(),.Wr(),
                .Wrfsource(Wrfsource),.Wrn(Wrn_in),
                .Ww_hi(),.Ww_lo(),
                .Ww_rf(Ww_rf));

    //WB
    pipeWB WB(.alu(Walu),.a(Wa),.b(Wb),
              .counter(),.cp0(),
              .dm(Wdm),
              .hi(),.hisource(),
              .lo(),.losource(),
              .muler_hi(Wmuler_hi),.muler_lo(Wmuler_lo),
              .pc4(Wpc4),.q(),.r(),
              .rfsource(Wrfsource),.rn(Wrn_in),
              .w_hi(),.w_lo(),.w_rf(Ww_rf),
              .Wdata_hi(Wdata_hi),.Wdata_lo(Wdata_lo),.Wdata_rf(Wdata_r
f),
              .Wrn(Wrn),
              .Ww_hi(Wena_hi),.Ww_lo(Wena_lo),.Ww_rf(Wena_rf));
    //for test
    assign _pc=pc;
    assign _inst=Dinst_in;
    assign _ealu=Ealu_o;
    assign _malu=Malu_o;
    assign _walu=Walu;
endmodule
```

```
module pcreg(
input clk,
input rst,
input wena,
input [31:0]data_in,
output[31:0]data_out
);
reg     [31:0]data;
```

```
always @(posedge clk or posedge rst)
    if (rst)
        data <= 32'h00400000;
    else if (wena)
        data <= data_in;
assign data_out = data;
endmodule
```

```
//IF including imem,cla
module pipeIF(bpc,cpc,jpc,pc,pcsource,rpc,inst,npc,pc4);
    input [31:0] bpc        ;
    input [31:0] cpc        ;
    input [31:0] jpc        ;
    input [31:0] pc         ;
    input [ 2:0] pcsource   ;
    input [31:0] rpc        ;
    output [31:0] inst      ;
    output [31:0] npc       ;
    output [31:0] pc4       ;
    assign pc4=pc+4;
    mux4x32 next_pc(.a(pc4),.b(bpc),.c(rpc),.d(jpc),.pcsource(pcsource[
1:0]),.y(npc));
    pipeIMEM imem(.pc(pc),.inst(inst));
endmodule
```

```
//IMEM
module pipeIMEM(pc,inst);
    input [31:0] pc;
    output [31:0] inst;
    IMEM i(.a({2'b00,pc[31:2]}),.spo(inst));
endmodule
```

```
//IR
module pipeIR(clk,pc4,inst,nostall,rst,Dinst,Dpc4);
    input clk           ;//时钟
    input [31:0] pc4    ;//pc+4
    input [31:0] inst   ;//指令
    input nostall       ;//
    input rst           ;//复位信号
    output [31:0] Dinst;//存的指令
    output [31:0] Dpc4 ;//存的 pc+4
```

```
    Reg pcplus4(.data_in(pc4),.clk(clk),.rst(rst),.wena(nostall),.data_
out(Dpc4));
    Reg instruction(.data_in(inst),.clk(clk),.rst(rst),.wena(nostall),.
data_out(Dinst));
endmodule
```

```
//ID
module pipeID(Ealu,Ecp0,Ehi,Ehisource,
              EisGoto,Elo,Elosource,
              Emuler_hi,Emuler_lo,Eq,Er,
              Erfsource,Ern,Ew_hi,Ew_lo,Ew_rf,
              Malu,Mdm,Mrfsource,
              Mmuler_hi,Mmuler_lo,
              Mrn,Mw_rf,Mw_hi,Mw_lo,
              Wdata_hi,Wdata_lo,Wdata_rf,
              Wena_hi,Wena_lo,Wena_rf,Wrn,
              clk,inst,pc4,rst,
              CP0out,
              Dpc4,
              Hiout,Loout,Rsout,Rtout,
              aluc,asource,bsource,
              bpc,cpc,
              cuttersource,
              div,
              reg28,
              hisource,
              imm,isGoto,
              jpc,
              losource,pcsource,
              rfsource,rn,rpc,
              sign,stall,
              w_dm,w_hi,w_lo,w_rf);
    input [31:0] Ealu           ;//前推 alu
    input [31:0] Ecp0           ;
    input [31:0] Ehi            ;
    input [ 1:0] Ehisource      ;
    input EisGoto               ;//从 EXE 传回的 isGoto(ejal)
    input [31:0] Elo            ;
    input [ 1:0] Elosource      ;
    input [31:0] Emuler_hi      ;
    input [31:0] Emuler_lo      ;
    input [31:0] Eq             ;
```

```
    input [31:0] Er             ;
    input [ 2:0] Erfsource      ;
    input [ 4:0] Ern            ;//
    input Ew_hi                 ;//写 hi 使能信号
    input Ew_lo                 ;//写 lo 使能信号
    input Ew_rf                 ;//写 rf 使能信号,ewreg

    input [31:0] Malu           ;//前推 alu
    input [31:0] Mdm            ;
    input [ 4:0] Mrn            ;//
    input [ 2:0] Mrfsource      ;
    input [31:0] Mmuler_hi      ;
    input [31:0] Mmuler_lo      ;
    input Mw_rf                 ;//写 rf 的信号，mwreg
    input Mw_hi                 ;//写 rf 的信号，mwreg
    input Mw_lo                 ;//写 rf 的信号，mwreg

    input [31:0] Wdata_hi       ;//写入 hi
    input [31:0] Wdata_lo       ;//写入 lo
    input [31:0] Wdata_rf       ;//写入 rf 的数,会在 WB 周期返回，写入 rn
    input Wena_hi               ;//hi 写使能?
    input Wena_lo               ;//lo 写使能?
    input Wena_rf               ;//rf 写使能?
    input [ 4:0] Wrn            ;//Wrn
    input clk                   ;//时钟
    input [31:0] inst           ;//指令
    input [31:0] pc4            ;//pc+4
    input rst                   ;//复位信号
   output [31:0] CP0out         ;//cp0 输出
    output [31:0] Dpc4          ;//pc+4 输出
    output [31:0] Hiout         ;//Hi 寄存?
    output [31:0] Loout         ;//Lo 寄存?
    output [31:0] Rsout         ;//Rs 输出
    output [31:0] Rtout         ;//Rt 输出
    output [ 3:0] aluc          ;//alu 控制信号
    output asource              ;//alu 的 a 选择信号,shift
    output bsource              ;//alu 的 b 选择信号,aluimm
    output [31:0] bpc           ;//beq 的跳转 pc
    output [31:0] cpc           ;//
    output [ 1:0] cuttersource  ;//
    output div                  ;//
    output [ 1:0] hisource      ;//hi 的?选择信号
```

```
    output [ 1:0] losource      ;//lo 的?选择信号
    output [31:0] imm           ;//扩展后的立即?
    output isGoto               ;//jal
    output [31:0] jpc           ;//跳转的 pc
    output [ 2:0] pcsource      ;//pc 的?选择信号
    output [ 2:0] rfsource      ;//写回 rf 的来?
    output [ 4:0] rn            ;//
    output [31:0] rpc           ;//寄存器堆得到的下?
    output sign                 ;
    output stall                ;//停止
    output w_hi                 ;//写 hi 信号  ,1?0 不写
    output w_lo                 ;//写 lo 信号  ,1?0 不写
    output w_dm                 ;//写 dmem 信号,1?0 不写(wmem)
    output w_rf                 ;//写 rf 信号  ,1?0 不写(wreg)
    output [31:0]reg28;
    wire [ 5:0] op,func;//op and func
    wire [ 4:0] rs,rt,rd;//3 个寄存器地址
    assign func = inst[ 5: 0];
    assign op   = inst[31:26];
    assign rs   = inst[25:21];
    assign rt   = inst[20:16];
    assign rd   = inst[15:11];

    wire reg_rt;//0 选 rd 作为目的寄存?,1 选 rt 作为目的寄存?
    wire sext;//是否符号扩展
    wire zero;
    wire [31:0] qa,qb;
    wire [31:0] Rsout0,Rtout0;
    assign zero = (Rsout==Rtout);
    wire [1:0] fwda0,fwda1,fwdb0,fwdb1;
    wire delay;
    //cu
    pipeIDcu cu(.op1(rs),.op2(rt),.op(op),.func(func),.rd(rd),.zero(zero),
                .EisGoto(EisGoto),
                .Erfsource(Erfsource),.Mrfsource(Mrfsource),
                .Ew_rf(Ew_rf),.Mw_rf(Mw_rf),
                .Ern(Ern),.Mrn(Mrn),
                .isGoto(isGoto),
                .aluc(aluc),.asource(asource),.bsource(bsource),
                .pcsource(pcsource),.rfsource(rfsource),
                .w_dm(w_dm),.w_rf(w_rf),
```

```
                .reg_rt(reg_rt),
                .sext(sext),.stall(stall),
                .fwda0(fwda0),.fwdb0(fwdb0),
                .fwda1(fwda1),.fwdb1(fwdb1),
                .delay(delay));
    assign rn=reg_rt?rt:rd;

//regfile rf(.clk(~clk),.rst(rst),.we(Wena_rf),.raddr1(rs),.raddr2(rt),
      //.waddr(Wrn),.wdata(Wdata_rf),.rdata1(qa),.rdata2(qb));
    regfile rf (.clk(~clk),.rst(rst),.we(Wena_rf),.raddr1(rs),.raddr2(r
t),
                .waddr(Wrn),.wdata(Wdata_rf),.rdata1(qa),.rdata2(qb),.r
eg28(reg28));

    mux4x32 a0(.a(qa),.b(Ealu),.c(Malu),.d(Mdm),.pcsource(fwda0),.y(Rso
ut0));
    mux4x32 b0(.a(qb),.b(Ealu),.c(Malu),.d(Mdm),.pcsource(fwdb0),.y(Rto
ut0));
    mux4x32 a1(.a(Rsout0),.b(Emuler_lo),.c(Mmuler_lo),.d(32'h0000),.pcs
ource(fwda1),.y(Rsout));
    mux4x32 b1(.a(Rtout0),.b(Emuler_lo),.c(Mmuler_lo),.d(32'h0000),.pcs
ource(fwdb1),.y(Rtout));
    wire [31:0] hi,lo;
    wire [16:0] ext16;//16 位符号扩展
    wire [31:0] br_offset;//寄存器的输出 qa,qb,branch_offset 分支偏移
    wire e;
    assign e = sext&inst[15];//符号扩展
    assign ext16 = {16{e}};
    assign imm = {ext16,inst[15:0]};//16 位扩展,32 位的立即数
    assign br_offset = {imm[29:0],2'b00};//18 位扩展,branch_offset 分支
偏移,跳转地址
    assign bpc=pc4+br_offset;
    assign rpc   = Rsout-32'h00400000;
    assign jpc   = {pc4[31:28],inst[25:0],2'b00};
    assign Dpc4  = pc4;
    assign sign  = sext;
endmodule
```

```
//IDcu
module pipeIDcu(op1,op2,op,func,rd,zero,
                EisGoto,
                Ew_rf,Mw_rf,
```

```
                Ern,Mrn,
                Erfsource,Mrfsource,
                isGoto,
                aluc,asource,bsource,
                pcsource,rfsource,
                w_dm,w_rf,w_hi,w_lo,
                reg_rt,
                sext,stall,
                fwda0,fwdb0,
                fwda1,fwdb1,
                delay);
    input [4:0] op1         ;//第一个操作数，即 rs
    input [4:0] op2         ;//第二个操作数，即 rt
    input [5:0] op          ;//指令前六位，操作码
    input [5:0] func        ;//指令后六位
    input [4:0] rd          ;
    input zero              ;//rs,rt 相等
    input EisGoto           ;
    input Ew_rf             ;
    input Mw_rf             ;
    input [4:0]Ern          ;
    input [4:0]Mrn          ;
    input [2:0] Erfsource   ;
    input [2:0] Mrfsource   ;
    output isGoto           ;
    output [3:0] aluc       ;//aluc 的选择信号
    output asource          ;
    output bsource          ;
    output [2:0] pcsource   ;//pc 的五选一选择信号
    output [2:0] rfsource   ;
    output w_dm             ;
    output w_rf             ;
    output w_hi             ;
    output w_lo             ;
    output reg_rt           ;
    output sext             ;//是否符号扩展
    output stall            ;
    output reg [1:0] fwda0  ;
    output reg [1:0] fwdb0  ;
    output reg [1:0] fwda1  ;
    output reg [1:0] fwdb1  ;
    output delay            ;
```

```
    //31 条指令译码
    wire r_type=~|op;
    wire i_add  = r_type &  func[5] & ~func[4] & ~func[3] &~func[2] & ~
func[1] & ~func[0];
    wire i_addu = r_type &  func[5] & ~func[4] & ~func[3] &~func[2] & ~
func[1] &  func[0];
    wire i_sub  = r_type &  func[5] & ~func[4] & ~func[3] &~func[2] &
func[1] & ~func[0];
    wire i_subu = r_type &  func[5] & ~func[4] & ~func[3] &~func[2] &
func[1] &  func[0];
    wire i_and  = r_type &  func[5] & ~func[4] & ~func[3] & func[2] & ~
func[1] & ~func[0];
    wire i_or   = r_type &  func[5] & ~func[4] & ~func[3] & func[2] & ~
func[1] &  func[0];
    wire i_xor  = r_type &  func[5] & ~func[4] & ~func[3] & func[2] &
func[1] & ~func[0];
    wire i_nor  = r_type &  func[5] & ~func[4] & ~func[3] & func[2] &
func[1] &  func[0];
    wire i_slt  = r_type &  func[5] & ~func[4] &  func[3] &~func[2] &
func[1] & ~func[0];
    wire i_sltu = r_type &  func[5] & ~func[4] &  func[3] &~func[2] &
func[1] &  func[0];
    wire i_sll  = r_type & ~func[5] & ~func[4] & ~func[3] &~func[2] & ~
func[1] & ~func[0];
    wire i_srl  = r_type & ~func[5] & ~func[4] & ~func[3] &~func[2] &
func[1] & ~func[0];
    wire i_sra  = r_type & ~func[5] & ~func[4] & ~func[3] &~func[2] &
func[1] &  func[0];
    wire i_sllv = r_type & ~func[5] & ~func[4] & ~func[3] & func[2] & ~
func[1] & ~func[0];
    wire i_srlv = r_type & ~func[5] & ~func[4] & ~func[3] & func[2] &
func[1] & ~func[0];
    wire i_srav = r_type & ~func[5] & ~func[4] & ~func[3] & func[2] &
func[1] &  func[0];
    wire i_jr   = r_type & ~func[5] & ~func[4] &  func[3] &~func[2] & ~
func[1] & ~func[0];
    wire i_addi = ~op[5] & ~op[4] &  op[3] & ~op[2] & ~op[1] & ~op[0];
    wire i_addiu= ~op[5] & ~op[4] &  op[3] & ~op[2] & ~op[1] &  op[0];
    wire i_andi = ~op[5] & ~op[4] &  op[3] &  op[2] & ~op[1] & ~op[0];
    wire i_ori  = ~op[5] & ~op[4] &  op[3] &  op[2] & ~op[1] &  op[0];
    wire i_xori = ~op[5] & ~op[4] &  op[3] &  op[2] &  op[1] & ~op[0];
```

```
    wire i_lw   =  op[5] & ~op[4] & ~op[3] & ~op[2] &  op[1] &  op[0];
    wire i_sw   =  op[5] & ~op[4] &  op[3] & ~op[2] &  op[1] &  op[0];
    wire i_beq  = ~op[5] & ~op[4] & ~op[3] &  op[2] & ~op[1] & ~op[0];
    wire i_bne  = ~op[5] & ~op[4] & ~op[3] &  op[2] & ~op[1] &  op[0];
    wire i_slti = ~op[5] & ~op[4] &  op[3] & ~op[2] &  op[1] & ~op[0];
    wire i_sltiu= ~op[5] & ~op[4] &  op[3] & ~op[2] &  op[1] &  op[0];
    wire i_lui  = ~op[5] & ~op[4] &  op[3] &  op[2] &  op[1] &  op[0];
    wire i_j    = ~op[5] & ~op[4] & ~op[3] & ~op[2] &  op[1] & ~op[0];
    wire i_jal  = ~op[5] & ~op[4] & ~op[3] & ~op[2] &  op[1] &  op[0];

    //54 指令译码
    wire c0_type  = ~op[5] &  op[4] & ~op[3] & ~op[2] & ~op[1] & ~op[0]
;
    wire i_mfc0   = c0_type&~op1[4]&~op1[3] &~op1[2]&~op1[1]&~op1[0];
    wire i_mtc0   = c0_type&~op1[4]&~op1[3] & op1[2]&~op1[1]&~op1[0];
    wire i_eret   = c0_type& op1[4]&~op1[3] &~op1[2]&~op1[1]&~op1[0]&~f
unc[5]& func[4] & func[3]&~func[2] & ~func[1] & ~func[0];
    wire i_syscall= r_type & ~func[5] & ~func[4] &  func[3] & func[2] &
 ~func[1] & ~func[0];
    wire i_break  = r_type & ~func[5] & ~func[4] &  func[3] & func[2] &
 ~func[1] &  func[0];
    wire i_teq    = r_type &  func[5] &  func[4] & ~func[3] & func[2] &
 ~func[1] & ~func[0];
    wire i_jalr   = r_type & ~func[5] & ~func[4] &  func[3] &~func[2] &
 ~func[1] &  func[0];
    wire i_divu     = r_type & ~func[5] &  func[4] &  func[3] &~func[2]
 &  func[1] &  func[0];
    wire i_div      = r_type & ~func[5] &  func[4] &  func[3] &~func[2]
 &  func[1] & ~func[0];
    wire i_mulu     = r_type & ~func[5] &  func[4] &  func[3] &~func[2]
 & ~func[1] &  func[0];
    wire i_mfhi     = r_type & ~func[5] &  func[4] & ~func[3] &~func[2]
 & ~func[1] & ~func[0];
    wire i_mflo     = r_type & ~func[5] &  func[4] & ~func[3] &~func[2]
 &  func[1] & ~func[0];
    wire i_mthi     = r_type & ~func[5] &  func[4] & ~func[3] &~func[2]
 & ~func[1] &  func[0];
    wire i_mtlo     = r_type & ~func[5] &  func[4] & ~func[3] &~func[2]
 &  func[1] &  func[0];
    wire i_clz      =~op[5] &  op[4] &  op[3] &  op[2] & ~op[1] & ~op[0
]&  func[5] & ~func[4] & ~func[3] &~func[2] & ~func[1] & ~func[0];
```

```
    wire i_mul      =~op[5] &  op[4] &  op[3] &  op[2] & ~op[1] & ~op[0
]& ~func[5] & ~func[4] & ~func[3] &~func[2] &  func[1] & ~func[0];
    wire i_bgez     =~op[5] & ~op[4] & ~op[3] & ~op[2] & ~op[1] &  op[0
]&  ~op2[4] & ~op2[3]  & ~op2[2]  &~op2[1]  & op2[0];
    wire i_lb       = op[5] & ~op[4] & ~op[3] & ~op[2] & ~op[1] & ~op[0
];
    wire i_lbu      = op[5] & ~op[4] & ~op[3] &  op[2] & ~op[1] & ~op[0
];
    wire i_lh       = op[5] & ~op[4] & ~op[3] & ~op[2] & ~op[1] &  op[0
];
    wire i_lhu      = op[5] & ~op[4] & ~op[3] &  op[2] & ~op[1] &  op[0
];
    wire i_sb       = op[5] & ~op[4] &  op[3] & ~op[2] & ~op[1] & ~op[0
];
    wire i_sh       = op[5] & ~op[4] &  op[3] & ~op[2] & ~op[1] &  op[0
];

    //alu control
    assign aluc[0]=i_sub|i_subu|i_or|i_nor|i_slt|i_srl|i_srlv|i_ori|i_b
eq|i_bne|i_slti|i_clz|i_bgez;
    assign aluc[1]=i_add|i_sub|i_xor|i_nor|i_slt|i_sltu|i_sll|i_sllv|i_
addi|i_xori|i_lw|i_sw|i_slti|i_sltiu|i_clz|i_lb|i_lbu|i_sb|i_lh|i_lhu|i
_sh;
    assign aluc[2]=i_and|i_or|i_xor|i_nor|i_sll|i_srl|i_sra|i_sllv|i_sr
lv|i_srav|i_andi|i_ori|i_xori|i_clz;
    assign aluc[3]=i_slt|i_sltu|i_sll|i_srl|i_sra|i_sllv|i_srlv|i_srav|
i_slti|i_sltiu|i_lui|i_clz|i_bgez;

    assign pcsource[2]=1'b0;
    assign pcsource[1]=i_jr|i_j|i_jal|i_jalr;
    assign pcsource[0]=(i_beq&zero)|(i_bne&~zero)|i_j|i_jal;
    assign delay =(i_beq&zero)|(i_bne&~zero);
    assign isGoto = i_jal;
    assign rfsource[2]=1'b0;
    assign rfsource[1]=i_mul|i_mulu;
    assign rfsource[0]=i_lw|i_lb|i_lbu|i_lh|i_lhu;
    assign asource=i_sll|i_srl|i_sra;
    assign bsource=i_addi|i_andi|i_ori|i_xori|i_lw|i_lui|i_sw;
    assign w_dm=(i_sw|i_sb|i_sh)&(~stall);
    assign w_rf=(i_add|i_addu|i_sub|i_subu|i_and|i_or|i_xor|i_nor|
                i_slt|i_sltu||i_sll|i_srl|i_sra|i_sllv|i_srlv|i_srav|
                i_jr|i_addi|i_addiu|i_andi|i_ori|i_xori|i_lw|i_slti|
```

```
                i_sltiu|i_lui|i_jal|i_mfc0|i_mfhi|i_mflo|i_jalr|i_clz|
                i_lb|i_lbu|i_lh|i_lhu|i_mul)&(~stall);
    assign w_hi=i_mul|i_mulu;
    assign w_lo=i_mul|i_mulu;
    assign reg_rt=(((i_addi|i_addiu)|(i_andi|i_ori))|((i_xori|i_lw)|(i_
sw|i_beq)))|
                  ((i_bne|i_slti)|(i_sltiu|i_lui))|i_lh|i_lhu|i_lb|i_lb
u|i_sh|i_sb|i_mfc0;
    //i_addi|i_lw|i_sw|
    assign sext=((i_addi|i_addiu)|(i_slti|i_sltiu))|((i_lui|i_lw)|i_sw)
|i_lh|i_lb|i_sh|i_sb|i_lbu|i_lhu|i_beq|i_bne|i_mul;
    wire i_rs= i_add | i_sub | i_and | i_or | i_xor | i_jr | i_addi|
               i_andi| i_ori | i_xori| i_lw | i_sw  | i_beq| i_bne |
               i_mul | i_mulu;
    wire i_rt= i_add | i_sub | i_and | i_or | i_xor | i_sll | i_srl|
               i_sra | i_sw  | i_beq | i_bne| i_mul | i_mulu;

    assign stall = (Ew_rf & Erfsource[0] & (Ern!=0) & ((i_rs&(Ern==op1)
) | (i_rt&(Ern==op2))));

    always@(Ew_rf or Mw_rf or Ern or Mrn or Erfsource[0] or Mrfsource[0
] or op1 or op2)
    begin
        fwda0=2'b00;
        fwda1=2'b00;
        if(Ew_rf&(Ern!=0)&(Ern==op1)&~Erfsource[0])
        begin
            fwda0=2'b01;//选择 Ealu 的结果
        end
        else
        begin
            if(Mw_rf&(Mrn!=0)&(Mrn==op1)&~Mrfsource[0])
            begin
                fwda0=2'b10;
            end
            else
            begin
                if(Mw_rf&(Mrn!=0)&(Mrn==op1)&Mrfsource[0])
                begin
                fwda0=2'b11;
                end
            end
```

```
        end
        if(Ew_rf&(Ern!=0)&(Ern==op1)&Erfsource[1])
        begin
            fwda0=2'b01;//选择 Ealu 的结果
        end
        else if(Mw_rf&(Mrn!=0)&(Mrn==op1)&Mrfsource[1])
        begin
            fwda1=2'b10;
        end

        fwdb0=2'b00;
        fwdb1=2'b00;
        if(Ew_rf&(Ern!=0)&(Ern==op2)&~Erfsource[0])
        begin
            fwdb0=2'b01;//选择 Ealu 的结果
        end
        else
        begin
            if(Mw_rf&(Mrn!=0)&(Mrn==op2)&~Mrfsource[0])
            begin
                fwdb0=2'b10;
            end
            else
            begin
                if(Mw_rf&(Mrn!=0)&(Mrn==op2)&Mrfsource[0])
                begin
                fwdb0=2'b11;
                end
            end
        end
        if(Ew_rf&(Ern!=0)&(Ern==op2)&Erfsource[1])
        begin
            fwdb1=2'b01;//选择 Ealu 的结果
        end
        else if(Mw_rf&(Mrn!=0)&(Mrn==op2)&Mrfsource[1])
        begin
            fwdb1=2'b10;
        end
    end
endmodule
```

```
//regfile
```

```
module regfile(clk,rst,we,raddr1,raddr2,waddr,wdata,rdata1,rdata2,reg28
);
    input clk               ;//下降沿写入数据
    input rst               ;//高电平全部寄存器置零
    input we                ;//高电平可以写，低电平可以读
    input [ 4:0] raddr1     ;//读的地址
    input [ 4:0] raddr2     ;
    input [ 4:0] waddr      ;//写的地址
    input [31:0] wdata      ;//写的数据，下降沿写入
    output [31:0] rdata1    ;
    output [31:0] rdata2    ;
    output [31:0] reg28     ;
    wire [31:0] select;

    wire [1023:0]data;
    decoder d(waddr,we,select);//选择
    //32 个寄存器
    Reg u0 (.clk(clk),.rst(rst||(!waddr)),.wena(select[0]&&waddr&&we),.
data_in(wdata),.data_out(data[0*32+:32]));
    Reg u1 (.clk(clk),.rst(rst),.wena(select[ 1]&&we),.data_in(wdata),.
data_out(data[ 1*32+:32]));
    Reg u2 (.clk(clk),.rst(rst),.wena(select[ 2]&&we),.data_in(wdata),.
data_out(data[ 2*32+:32]));
    Reg u3 (.clk(clk),.rst(rst),.wena(select[ 3]&&we),.data_in(wdata),.
data_out(data[ 3*32+:32]));
    Reg u4 (.clk(clk),.rst(rst),.wena(select[ 4]&&we),.data_in(wdata),.
data_out(data[ 4*32+:32]));
    Reg u5 (.clk(clk),.rst(rst),.wena(select[ 5]&&we),.data_in(wdata),.
data_out(data[ 5*32+:32]));
    Reg u6 (.clk(clk),.rst(rst),.wena(select[ 6]&&we),.data_in(wdata),.
data_out(data[ 6*32+:32]));
    Reg u7 (.clk(clk),.rst(rst),.wena(select[ 7]&&we),.data_in(wdata),.
data_out(data[ 7*32+:32]));

    Reg u8 (.clk(clk),.rst(rst),.wena(select[ 8]&&we),.data_in(wdata),.
data_out(data[ 8*32+:32]));
    Reg u9 (.clk(clk),.rst(rst),.wena(select[ 9]&&we),.data_in(wdata),.
data_out(data[ 9*32+:32]));
    Reg u10(.clk(clk),.rst(rst),.wena(select[10]&&we),.data_in(wdata),.
data_out(data[10*32+:32]));
    Reg u11(.clk(clk),.rst(rst),.wena(select[11]&&we),.data_in(wdata),.
data_out(data[11*32+:32]));
```

```
    Reg u12(.clk(clk),.rst(rst),.wena(select[12]&&we),.data_in(wdata),.
data_out(data[12*32+:32]));
    Reg u13(.clk(clk),.rst(rst),.wena(select[13]&&we),.data_in(wdata),.
data_out(data[13*32+:32]));
    Reg u14(.clk(clk),.rst(rst),.wena(select[14]&&we),.data_in(wdata),.
data_out(data[14*32+:32]));
    Reg u15(.clk(clk),.rst(rst),.wena(select[15]&&we),.data_in(wdata),.
data_out(data[15*32+:32]));

    Reg u16(.clk(clk),.rst(rst),.wena(select[16]&&we),.data_in(wdata),.
data_out(data[16*32+:32]));
    Reg u17(.clk(clk),.rst(rst),.wena(select[17]&&we),.data_in(wdata),.
data_out(data[17*32+:32]));
    Reg u18(.clk(clk),.rst(rst),.wena(select[18]&&we),.data_in(wdata),.
data_out(data[18*32+:32]));
    Reg u19(.clk(clk),.rst(rst),.wena(select[19]&&we),.data_in(wdata),.
data_out(data[19*32+:32]));
    Reg u20(.clk(clk),.rst(rst),.wena(select[20]&&we),.data_in(wdata),.
data_out(data[20*32+:32]));
    Reg u21(.clk(clk),.rst(rst),.wena(select[21]&&we),.data_in(wdata),.
data_out(data[21*32+:32]));
    Reg u22(.clk(clk),.rst(rst),.wena(select[22]&&we),.data_in(wdata),.
data_out(data[22*32+:32]));
    Reg u23(.clk(clk),.rst(rst),.wena(select[23]&&we),.data_in(wdata),.
data_out(data[23*32+:32]));

    Reg u24(.clk(clk),.rst(rst),.wena(select[24]&&we),.data_in(wdata),.
data_out(data[24*32+:32]));
    Reg u25(.clk(clk),.rst(rst),.wena(select[25]&&we),.data_in(wdata),.
data_out(data[25*32+:32]));
    Reg u26(.clk(clk),.rst(rst),.wena(select[26]&&we),.data_in(wdata),.
data_out(data[26*32+:32]));
    Reg u27(.clk(clk),.rst(rst),.wena(select[27]&&we),.data_in(wdata),.
data_out(data[27*32+:32]));
    Reg u28(.clk(clk),.rst(rst),.wena(select[28]&&we),.data_in(wdata),.
data_out(data[28*32+:32]));
    Reg u29(.clk(clk),.rst(rst),.wena(select[29]&&we),.data_in(wdata),.
data_out(data[29*32+:32]));
    Reg u30(.clk(clk),.rst(rst),.wena(select[30]&&we),.data_in(wdata),.
data_out(data[30*32+:32]));
    Reg u31(.clk(clk),.rst(rst),.wena(select[31]&&we),.data_in(wdata),.
data_out(data[31*32+:32]));
```

```
    selector s1(clk,data,raddr1,rdata1);
    selector s2(clk,data,raddr2,rdata2);
    assign reg28=data[28*32+:32];
endmodule
```

```
//ID EXE reg
module pipeDEreg(
    input [ 3:0] Daluc,
    input [31:0] Da                 ,//pipeID 的 Rsout
    input [31:0] Db                 ,//pipeID 的 Rsout
    input Dasource                  ,
    input Dbsource                  ,
    input [31:0] Dcp0               ,
    input [ 1:0] Dcuttersource      ,
    input Ddiv                      ,
    input [31:0] Dhi                ,
    input [ 1:0] Dhisource          ,
    input [31:0] Dimm               ,
    input DisGoto                   ,
    input [31:0] Dlo                ,
    input [ 1:0] Dlosource          ,
    input [31:0] Dpc4               ,
    input [ 2:0] Drfsource          ,
    input [ 4:0] Drn                ,//pipeID 的 rn
    input Dsign                     ,
    input Dw_dm                     ,
    input Dw_hi                     ,
    input Dw_lo                     ,
    input Dw_rf                     ,
    input clk                       ,
    input rst                       ,
    input wena                      ,
    output reg [ 3:0] Ealuc         ,
    output reg [31:0] Ea            ,
    output reg [31:0] Eb            ,
    output reg Easource             ,
    output reg Ebsource             ,
    output reg [31:0] Ecp0          ,
    output reg [ 1:0] Ecuttersource ,
    output reg Ediv                 ,
    output reg [31:0] Ehi           ,
    output reg [ 1:0] Ehisource     ,
```

```
    output reg [31:0] Eimm          ,
    output reg EisGoto              ,
    output reg [31:0] Elo           ,
    output reg [ 1:0] Elosource     ,
    output reg [31:0] Epc4          ,
    output reg [ 2:0] Erfsource     ,
    output reg [ 4:0] Ern           ,
    output reg Esign                ,
    output reg Ew_dm                ,
    output reg Ew_hi                ,
    output reg Ew_lo                ,
    output reg Ew_rf
    );
    always@(posedge rst or posedge clk)
    begin
        if(rst==1)
        begin
            Ealuc           <= 0;
            Ea              <= 0;
            Eb              <= 0;
            Easource        <= 0;
            Ebsource        <= 0;
            Ecp0            <= 0;
            Ecuttersource   <= 0;
            Ediv            <= 0;
            Ehi             <= 0;
            Ehisource       <= 0;
            Eimm            <= 0;
            EisGoto         <= 0;
            Elo             <= 0;
            Elosource       <= 0;
            Epc4            <= 0;
            Erfsource       <= 0;
            Ern             <= 0;
            Esign           <= 0;
            Ew_dm           <= 0;
            Ew_hi           <= 0;
            Ew_lo           <= 0;
            Ew_rf           <= 0;
        end
        else
        begin
```

```
            Ealuc           <= Daluc            ;
            Ea              <= Da               ;
            Eb              <= Db               ;
            Easource        <= Dasource         ;
            Ebsource        <= Dbsource         ;
            Ecp0            <= Dcp0             ;
            Ecuttersource   <= Dcuttersource    ;
            Ediv            <= Ddiv             ;
            Ehi             <= Dhi              ;
            Ehisource       <= Dhisource        ;
            Eimm            <= Dimm             ;
            EisGoto         <= DisGoto          ;
            Elo             <= Dlo              ;
            Elosource       <= Dlosource        ;
            Epc4            <= Dpc4             ;
            Erfsource       <= Drfsource        ;
            Ern             <= Drn              ;
            Esign           <= Dsign            ;
            Ew_dm           <= Dw_dm            ;
            Ew_hi           <= Dw_hi            ;
            Ew_lo           <= Dw_lo            ;
            Ew_rf           <= Dw_rf            ;
        end
    end
endmodule
```

```
//EXE including ALU,mul,div,cla
module pipeEXE(aluc,a,b,
               asource,bsource,
               cp0,cuttersource,
               div,
               hi,hisource,
               imm,
               isGoto,
               lo,losource,
               pc4,
               rfsource,rn,
               sign,
               w_dm,w_hi,w_lo,w_rf,
               Ealu,Ea,Eb,
               Ecounter,
               Ecp0,Ecuttersource,
```

```
            Ehi,Ehisource,
            EisGoto,
            Elo,Elosource,
            Emuler_hi,Emuler_lo,
            Epc4,
            Eq,Er,
            Erfsource,Ern,
            Esign,
            Ew_dm,Ew_hi,Ew_lo,Ew_rf);
    input [ 3:0] aluc           ;
    input [31:0] a              ;
    input [31:0] b              ;
    input asource               ;
    input bsource               ;
    input [31:0] cp0            ;
    input [ 1:0] cuttersource   ;
    input div                   ;
    input [31:0] hi             ;
    input [ 1:0] hisource       ;
    input [31:0] imm            ;
    input isGoto                ;//暂时认为是 jal_ena?
    input [31:0] lo             ;
    input [ 1:0] losource       ;
    input [31:0] pc4            ;
    input [ 2:0] rfsource       ;
    input [ 4:0] rn             ;
    input sign                  ;
    input w_dm                  ;
    input w_hi                  ;
    input w_lo                  ;
    input w_rf                  ;
    output [31:0] Ealu          ;
    output [31:0] Ea            ;
    output [31:0] Eb            ;
    output [31:0] Ecounter      ;
    output [31:0] Ecp0          ;
    output [ 1:0] Ecuttersource ;
    output [31:0] Ehi           ;
    output [ 1:0] Ehisource     ;
    output EisGoto              ;
    output [31:0] Elo           ;
    output [ 1:0] Elosource     ;
```

```
    output [31:0] Emuler_hi     ;
    output [31:0] Emuler_lo     ;
    output [31:0] Epc4          ;
    output [31:0] Eq            ;
    output [31:0] Er            ;
    output [ 2:0] Erfsource     ;
    output [ 4:0] Ern           ;
    output Esign                ;
    output Ew_dm                ;
    output Ew_hi                ;
    output Ew_lo                ;
    output Ew_rf                ;
    wire [31:0] alua,alub;
    wire [31:0] ext5;
    assign ext5 = {27'b0000_0000_0000_0000_0000_0000_000,imm[10:6]};
    //mux2x32(a,b,s,y);
    mux2x32 alu_a(.a(a),.b(ext5),.s(asource),.y(alua));
    mux2x32 alu_b(.a(b),.b(imm),.s(bsource),.y(alub));
    assign Ern = rn|{5{isGoto}};//jal??31 号寄存器
    wire [31:0] pc8;
    assign pc8=pc4+4;
    assign Esign=sign;
    assign Ew_dm=w_dm;
    assign Ew_rf=w_rf;
    assign Ew_hi=w_hi;
    assign Ew_lo=w_lo;
    assign Ehi=hi;
    assign Elo=lo;
    assign Er=32'h0000;//没有写除法器
    assign Eq=32'h0000;
    assign Ea=a;
    assign Eb=b;
    assign EisGoto=isGoto;
    assign Erfsource=rfsource;
    assign Epc4=pc4;
    wire [31:0] alu_temp;//ALU 的输?
    ALU alu_unit(.a(alua),.b(alub),.aluc(aluc),.r(alu_temp),.zero(),.ca
rry(),.negative(),.overflow());
    mux2x32 alu_res(.a(alu_temp),.b(pc8),.s(isGoto),.y(Ealu));
    wire [63:0]muls_res,mulu_res;
    mult_gen_0 muls(.A(a),.B(b),.P(muls_res));
    mult_gen_1 mulu(.A(a),.B(b),.P(mulu_res));
```

```
    mux2x32 mul_su_hi(.a(muls_res[63:32]),.b(mulu_res[63:32]),.s(~sign)
,.y(Emuler_hi));
    mux2x32 mul_su_lo(.a(muls_res[31:00]),.b(mulu_res[31:00]),.s(~sign)
,.y(Emuler_lo));
endmodule
```

```
//ALU
module ALU(a,b,aluc,r,zero,carry,negative,overflow);
    input [31:0] a  ;
    input [31:0] b  ;
    input [3:0] aluc;
    output[31:0] r  ;//result
    output zero     ;
    output carry    ;
    output negative ;
    output overflow ;
    //
    wire [31:0] d_add;
    wire d_c,d_n,d_o;

    //Adder
    Adder add(aluc,a,b,d_add,d_c,d_n,d_o);
    wire [31:0] d_and=a&b;
    wire [31:0] d_or=a|b;
    wire [31:0] d_xor=a^b;
    wire [31:0] d_nor=~(a|b);
    wire [31:0] d_lui={b[15:0],16'h0};
    wire [31:0] d_slt_s=($signed(a)<$signed(b))?1:0;
    wire d_slt_s_z=(a==b)?1'b1:1'b0;
    wire d_slt_s_n=d_slt_s[0];
    wire [31:0] d_slt_u=(a<b)?1:0;
    wire d_slt_u_c=d_slt_u[0];
    wire d_slt_u_z=(a==b)?1'b1:1'b0;
    wire [31:0]  sra=$signed(b)>>>a;
    wire [31:0]temp_sra=$signed(b)>>>(a-1);
    wire sra_c=temp_sra[0];
    wire [31:0] sll=$unsigned(b)<<a;
    wire [31:0]temp_sll=$unsigned(b)<<(a-1);
    wire sll_c=temp_sll[31];
    wire [31:0] srl=$unsigned(b)>>a;
    wire [31:0]temp_srl=$unsigned(b)>>>(a-1);
    wire srl_c=temp_srl[0];
```

```
    reg [31:0]temp;
    reg tz,tc,to,tn;
    always@(*)
    begin
        if(!{aluc[3],aluc[2]})//0123
        begin
            temp<=d_add;
            tc<=d_c;
            tn<=d_n;
            to<=d_o;
        end
        else if(aluc==4'b0100) temp<=d_and;
        else if(aluc==4'b0101) temp<=d_or;
        else if(aluc==4'b0110) temp<=d_xor;
        else if(aluc==4'b0111) temp<=d_nor;
        else if({aluc[3],aluc[2],aluc[1]}==3'b100) temp<=d_lui;
        else if(aluc==4'b1011) temp<=d_slt_s;
        else if(aluc==4'b1010)begin
            temp<=d_slt_u;
            tc<=d_slt_u_c;
        end
        else if(aluc==4'b1100)begin
            temp<=sra;
            tc<=sra_c;
        end
        else if(aluc==4'b1111||aluc==4'b1110) begin
            temp<=sll;
            tc<=sll_c;
        end
        else if(aluc==4'b1101)begin
            temp<=srl;
            tc<=srl_c;
        end
    end
   assign r=temp;
    assign zero=(aluc==4'b1010)?d_slt_u_z:(aluc==4'b1011)?d_slt_s_z:!r;
    assign overflow=to;
    assign negative=(aluc==4'b0010||aluc==4'b0011)?tn:(aluc==4'b1011)?d
_slt_s_n:temp[31];
    assign carry=tc;
endmodule
```

```
//EXE MEM reg
module pipeEMreg(Ealu,
                 Ea,Eb,
                 Ecounter,
                 Ecp0,
                 Ecuttersource,
                 Ehi,Ehisource,
                 Elo,Elosource,
                 Emuler_hi,Emuler_lo,
                 Epc4,
                 Eq,Er,
                 Erfsource,Ern,
                 Esign,
                 Ew_dm,
                 Ew_hi,Ew_lo,
                 Ew_rf,
                 clk,rst,wena,
                 Malu,
                 Ma,Mb,
                 Mcounter,
                 Mcp0,
                 Mcuttersource,
                 Mhi,Mhisource,
                 Mlo,Mlosource,
                 Mmuler_hi,Mmuler_lo,
                 Mpc4,
                 Mq,Mr,
                 Mrfsource,Mrn,
                 Msign,
                 Mw_dm,
                 Mw_hi,Mw_lo,
                 Mw_rf);

    input [31:0] Ealu              ;
    input [31:0] Ea                ;
    input [31:0] Eb                ;
    input [31:0] Ecounter          ;
    input [31:0] Ecp0              ;
    input [ 1:0] Ecuttersource     ;
    input [31:0] Ehi               ;
    input [ 1:0] Ehisource         ;
    input [31:0] Elo               ;
```

```
    input [ 1:0] Elosource          ;
    input [31:0] Emuler_hi          ;
    input [31:0] Emuler_lo          ;
    input [31:0] Epc4               ;
    input [31:0] Eq                 ;
    input [31:0] Er                 ;
    input [ 2:0] Erfsource          ;
    input [ 4:0] Ern                ;
    input Esign                     ;
    input Ew_dm                     ;
    input Ew_hi                     ;
    input Ew_lo                     ;
    input Ew_rf                     ;
    input clk                       ;
    input rst                       ;
    input wena                      ;
    output reg [31:0] Malu          ;
    output reg [31:0] Ma            ;
    output reg [31:0] Mb            ;
    output reg [31:0] Mcounter      ;
    output reg [31:0] Mcp0          ;
    output reg [ 1:0] Mcuttersource ;
    output reg [31:0] Mhi           ;
    output reg [ 1:0] Mhisource     ;
    output reg [31:0] Mlo           ;
    output reg [ 1:0] Mlosource     ;
    output reg [31:0] Mmuler_hi     ;
    output reg [31:0] Mmuler_lo     ;
    output reg [31:0] Mpc4          ;
    output reg [31:0] Mq            ;
    output reg [31:0] Mr            ;
    output reg [ 2:0] Mrfsource     ;
    output reg [ 4:0] Mrn           ;
    output reg Msign                ;
    output reg Mw_dm                ;
    output reg Mw_hi                ;
    output reg Mw_lo                ;
    output reg Mw_rf                ;

    always@(posedge rst or posedge clk)
    begin
        if(rst==1)
```

```
        begin
            Malu            <= 0;
            Ma              <= 0;
            Mb              <= 0;
            Mcounter        <= 0;
            Mcp0            <= 0;
            Mcuttersource   <= 0;
            Mhi             <= 0;
            Mhisource       <= 0;
            Mlo             <= 0;
            Mlosource       <= 0;
            Mmuler_hi       <= 0;
            Mmuler_lo       <= 0;
            Mpc4            <= 0;
            Mq              <= 0;
            Mr              <= 0;
            Mrfsource       <= 0;
            Mrn             <= 0;
            Msign           <= 0;
            Mw_dm           <= 0;
            Mw_hi           <= 0;
            Mw_lo           <= 0;
            Mw_rf           <= 0;
        end
        else
        begin
            Malu            <= Ealu             ;
            Ma              <= Ea               ;
            Mb              <= Eb               ;
            Mcounter        <= Ecounter         ;
            Mcp0            <= Ecp0             ;
            Mcuttersource   <= Ecuttersource    ;
            Mhi             <= Ehi              ;
            Mhisource       <= Ehisource        ;
            Mlo             <= Elo              ;
            Mlosource       <= Elosource        ;
            Mmuler_hi       <= Emuler_hi        ;
            Mmuler_lo       <= Emuler_lo        ;
            Mpc4            <= Epc4             ;
            Mq              <= Eq               ;
            Mr              <= Er               ;
            Mrfsource       <= Erfsource        ;
```

```
            Mrn             <= Ern              ;
            Msign           <= Esign            ;
            Mw_dm           <= Ew_dm            ;
            Mw_hi           <= Ew_hi            ;
            Mw_lo           <= Ew_lo            ;
            Mw_rf           <= Ew_rf            ;
        end
    end
endmodule
```

```
//MEM WB reg
module pipeMWreg(
    input [31:0] Malu           ,
    input [31:0] Ma             ,
    input [31:0] Mb             ,
    input [31:0] Mcounter       ,
    input [31:0] Mcp0           ,
    input [ 1:0] Mcuttersource  ,
    input [31:0] Mdm            ,
    input [31:0] Mhi            ,
    input [ 1:0] Mhisource      ,
    input [31:0] Mlo            ,
    input [ 1:0] Mlosource      ,
    input [31:0] Mmuler_hi      ,
    input [31:0] Mmuler_lo      ,
    input [31:0] Mpc4           ,
    input [31:0] Mq             ,
    input [31:0] Mr             ,
    input [ 2:0] Mrfsource      ,
    input [ 4:0] Mrn            ,
    input Mw_hi                 ,
    input Mw_lo                 ,
    input Mw_rf                 ,
    input clk                   ,
    input rst                   ,
    input wena                  ,
    output reg [31:0] Walu      ,
    output reg [31:0] Wa        ,
    output reg [31:0] Wb        ,
    output reg [31:0] Wcounter  ,
    output reg [31:0] Wcp0      ,
    output reg [31:0] Wdm       ,
```

```
    output reg [31:0] Whi       ,
    output reg [ 1:0] Whisource ,
    output reg [31:0] Wlo       ,
    output reg [ 1:0] Wlosource ,
    output reg [31:0] Wmuler_hi ,
    output reg [31:0] Wmuler_lo ,
    output reg [31:0] Wpc4      ,
    output reg [31:0] Wq        ,
    output reg [31:0] Wr        ,
    output reg [ 2:0] Wrfsource ,
    output reg [ 4:0] Wrn       ,
    output reg Ww_hi            ,
    output reg Ww_lo            ,
    output reg Ww_rf
    );
    always@(posedge rst or posedge clk)
    begin
        if(rst==1)
        begin
            Walu        <= 0;
            Wa          <= 0;
            Wb          <= 0;
            Wcounter    <= 0;
            Wcp0        <= 0;
            Wdm         <= 0;
            Whi         <= 0;
            Whisource   <= 0;
            Wlo         <= 0;
            Wlosource   <= 0;
            Wmuler_hi   <= 0;
            Wmuler_lo   <= 0;
            Wpc4        <= 0;
            Wq          <= 0;
            Wr          <= 0;
            Wrfsource   <= 0;
            Wrn         <= 0;
            Ww_hi       <= 0;
            Ww_lo       <= 0;
            Ww_rf       <= 0;
        end
        else
        begin
```

```
            Walu        <= Malu         ;
            Wa          <= Ma           ;
            Wb          <= Mb           ;
            Wcounter    <= Mcounter     ;
            Wcp0        <= Mcp0         ;
            Wdm         <= Mdm          ;
            Whi         <= Mhi          ;
            Whisource   <= Mhisource    ;
            Wlo         <= Mlo          ;
            Wlosource   <= Mlosource    ;
            Wmuler_hi   <= Mmuler_hi    ;
            Wmuler_lo   <= Mmuler_lo    ;
            Wpc4        <= Mpc4         ;
            Wq          <= Mq           ;
            Wr          <= Mr           ;
            Wrfsource   <= Mrfsource    ;
            Wrn         <= Mrn          ;
            Ww_hi       <= Mw_hi        ;
            Ww_lo       <= Mw_lo        ;
            Ww_rf       <= Mw_rf        ;
        end
    end
endmodule
```

```
//DMEM ip
module pipeDMEM(in,addr,clk,wmem,out);
    input [31:0]in      ;
    input [31:0]addr    ;
    input clk           ;
    input wmem          ;
    output [31:0] out   ;
    DMEM d(.a(addr),.d(in),.clk(clk),.we(wmem),.spo(out));
endmodule
```

```
//MEM WB reg
module pipeMWreg(
    input [31:0] Malu           ,
    input [31:0] Ma             ,
    input [31:0] Mb             ,
    input [31:0] Mcounter       ,
    input [31:0] Mcp0           ,
    input [ 1:0] Mcuttersource  ,
```

```
    input [31:0] Mdm            ,
    input [31:0] Mhi            ,
    input [ 1:0] Mhisource      ,
    input [31:0] Mlo            ,
    input [ 1:0] Mlosource      ,
    input [31:0] Mmuler_hi      ,
    input [31:0] Mmuler_lo      ,
    input [31:0] Mpc4           ,
    input [31:0] Mq             ,
    input [31:0] Mr             ,
    input [ 2:0] Mrfsource      ,
    input [ 4:0] Mrn            ,
    input Mw_hi                 ,
    input Mw_lo                 ,
    input Mw_rf                 ,
    input clk                   ,
    input rst                   ,
    input wena                  ,
    output reg [31:0] Walu      ,
    output reg [31:0] Wa        ,
    output reg [31:0] Wb        ,
    output reg [31:0] Wcounter  ,
    output reg [31:0] Wcp0      ,
    output reg [31:0] Wdm       ,
    output reg [31:0] Whi       ,
    output reg [ 1:0] Whisource ,
    output reg [31:0] Wlo       ,
    output reg [ 1:0] Wlosource ,
    output reg [31:0] Wmuler_hi ,
    output reg [31:0] Wmuler_lo ,
    output reg [31:0] Wpc4      ,
    output reg [31:0] Wq        ,
    output reg [31:0] Wr        ,
    output reg [ 2:0] Wrfsource ,
    output reg [ 4:0] Wrn       ,
    output reg Ww_hi            ,
    output reg Ww_lo            ,
    output reg Ww_rf
    );
    always@(posedge rst or posedge clk)
    begin
        if(rst==1)
```

```
        begin
            Walu        <= 0;
            Wa          <= 0;
            Wb          <= 0;
            Wcounter    <= 0;
            Wcp0        <= 0;
            Wdm         <= 0;
            Whi         <= 0;
            Whisource   <= 0;
            Wlo         <= 0;
            Wlosource   <= 0;
            Wmuler_hi   <= 0;
            Wmuler_lo   <= 0;
            Wpc4        <= 0;
            Wq          <= 0;
            Wr          <= 0;
            Wrfsource   <= 0;
            Wrn         <= 0;
            Ww_hi       <= 0;
            Ww_lo       <= 0;
            Ww_rf       <= 0;
        end
        else
        begin
            Walu        <= Malu         ;
            Wa          <= Ma           ;
            Wb          <= Mb           ;
            Wcounter    <= Mcounter     ;
            Wcp0        <= Mcp0         ;
            Wdm         <= Mdm          ;
            Whi         <= Mhi          ;
            Whisource   <= Mhisource    ;
            Wlo         <= Mlo          ;
            Wlosource   <= Mlosource    ;
            Wmuler_hi   <= Mmuler_hi    ;
            Wmuler_lo   <= Mmuler_lo    ;
            Wpc4        <= Mpc4         ;
            Wq          <= Mq           ;
            Wr          <= Mr           ;
            Wrfsource   <= Mrfsource    ;
            Wrn         <= Mrn          ;
            Ww_hi       <= Mw_hi        ;
```

```
            Ww_lo       <= Mw_lo        ;
            Ww_rf       <= Mw_rf        ;
        end
    end
endmodule
```

```
//WB
module pipeWB(alu,a,b,
              counter,cp0,
              dm,
              hi,hisource,
              lo,losource,
              muler_hi,muler_lo,
              pc4,q,r,
              rfsource,rn,
              w_hi,w_lo,w_rf,
              Wdata_hi,Wdata_lo,Wdata_rf,
              Wrn,
              Ww_hi,Ww_lo,Ww_rf);
    input [31:0] alu            ;//alu 的输出，实际是 Malu
    input [31:0] a              ;
    input [31:0] b              ;
    input [31:0] counter        ;
    input [31:0] cp0            ;
    input [31:0] dm             ;
    input [31:0] hi             ;
    input [ 1:0] hisource       ;
    input [31:0] lo             ;
    input [ 1:0] losource       ;
    input [31:0] muler_hi       ;
    input [31:0] muler_lo       ;
    input [31:0] pc4            ;
    input [31:0] q              ;
    input [31:0] r              ;
    input [ 2:0] rfsource       ;
    input [ 4:0] rn             ;
    input w_hi                  ;
    input w_lo                  ;
    input w_rf                  ;
    output [31:0] Wdata_hi      ;
    output [31:0] Wdata_lo      ;
    output [31:0] Wdata_rf      ;
```

```
    output [ 4:0] Wrn           ;
    output Ww_hi                ;
    output Ww_lo                ;
    output Ww_rf                ;
    wire [31:0] Wdata_rf0;
    mux2x32 alu_mem(.a(alu),.b(dm),.s(rfsource[0]),.y(Wdata_rf0));
    mux2x32 am_mul(.a(Wdata_rf0),.b(muler_lo),.s(rfsource[1]),.y(Wdata_
rf));
    assign Wrn=rn;
    assign Ww_rf=w_rf;
endmodule
```

2、 静态流水线的设计程序

```
module pipeCPU(clk,rst,reg28);
    input clk;
    input rst;
    output [31:0]reg28;
    //test
    output [31:0]inst_t;
    output [31:0]pc_t;
    output [31:0]ID_t,
    output [31:0]EXE_t,
    output [31:0]WB_t,
    output [31:0]IR_t,
    output [31:0]IF_t
    wire [31:0]npc,pc4,pc;
    wire stall;
    wire [31:0] bpc,jpc,rpc,cpc;
    wire [31:0] inst;
    wire [2:0] pcsource;
    wire [31:0] Dinst_in,Dpc4_in;
    wire [31:0] Dpc4_out;
    pcreg PC(.clk(clk),.data_in(npc),
             .rst(rst),.wena(~stall),.data_out(pc));
    PipeIF IF(.bpc(bpc),.cpc(),.jpc(jpc),.rpc(rpc),
              .pc(pc),.pcsource(pcsource),.inst(inst),
              .npc(npc),.pc4(pc4));
    PipeIR IR(.clk(clk),.pc4(pc4),.inst(inst),
              .nostall(~stall),.rst(rst),
              .Dinst(Dinst_in),.Dpc4(Dpc4_in));
    wire [31:0] Wdata_hi,Wdata_lo,Wdata_rf;
```

```
    wire Wena_hi,Wena_lo,Wena_rf;
    wire [ 4:0] Wrn;
    wire [31:0] Dcp0out;
    wire [31:0] Hiout,Loout,Rsout,Rtout;
    wire [ 3:0] aluc;
    wire asource,bsource;
    wire [1:0] cuttersource;
    wire [31:0] imm;
    wire isGoto;
    wire [ 4:0] rn;
    wire [2:0] Drfsource;
    wire EisGoto;
    wire [ 4:0] Ern;
    wire Ew_hi,Ew_lo,Ew_rf;
    wire [ 4:0] Mrn;
    wire Mw_rf;
    wire sign;
    wire [31:0] Ern_o;
    wire [ 4:0] Mrn_o;
    wire [31:0] Malu;
    wire [31:0] Mdm_o;
    wire w_dm,w_hi,w_lo,w_rf;
    wire [31:0]Ealu_o,Malu_o;
    wire [2:0] Erfsource_o,Mrfsource_o;
    PipeID ID(.EisGoto(EisGoto),.Ern(Ern_o),
              .Ew_hi(Ew_hi),.Ew_lo(Ew_lo),.Ew_rf(Ew_rf),
              .Mrn(Mrn_o),.Mw_rf(Mw_rf),
              .Wdata_hi(Wdata_hi),.Wdata_lo(Wdata_lo),.Wdata_rf(Wdata_r
f),
              .Wena_hi(Wena_hi),.Wena_lo(Wena_lo),.Wena_rf(Wena_rf),
              .Wrn(Wrn),
              .aluc(aluc),.asource(asource),.bsource(bsource),
              .bpc(bpc),.cpc(),
              .cuttersource(),
              .div(),
              .hisource(),
              .imm(imm),.isGoto(isGoto),
              .Ealu(Ealu_o),.Malu(Malu),
              .Mdm(Mdm_o),
              .Erfsource(Erfsource_o),.Mrfsource(Mrfsource_o),
              .clk(clk),.inst(Dinst_in),.pc4(Dpc4_in),.rst(rst),
              .CP0out(Dcp0out),
```

```
            .Dpc4(Dpc4_out),
            .Hiout(Hiout),.Loout(Loout),.Rsout(Rsout),.Rtout(Rtout),
            .rfsource(Drfsource),.rn(rn),.rpc(rpc),
            .sign(sign),.stall(stall),
            .jpc(jpc),
            .losource(),.pcsource(pcsource),
            .reg28(reg28),
            .w_dm(w_dm),.w_hi(),.w_lo(),.w_rf(w_rf));
    wire [4:0]Ealuc;
    wire [31:0] Ea,Eb,Eimm,Epc4;
    wire Easource,Ebsource,Esign,Ew_dm;
    wire [2:0] Erfsource_in;
    PipeDEreg DE(.Daluc(aluc),
                 .Da(Rsout),.Db(Rtout),
                 .Dasource(asource),.Dbsource(bsource),
                 .Ecp0(),.Ecuttersource(),
                 .Ediv(),
                 .Ehi(),.Ehisource(),
                 .Eimm(Eimm),
                 .EisGoto(EisGoto),
                 .Elo(),.Elosource(),
                 .Epc4(Epc4),
                 .Dcp0(Dcp0out),.Dcuttersource(),
                 .Ddiv(),
                 .Dhi(),.Dhisource(),
                 .Dimm(imm),
                 .DisGoto(isGoto),
                 .Dlo(),.Dlosource(),
                 .Dpc4(Dpc4_out),
                 .Drfsource(Drfsource),.Drn(rn),
                 .Dsign(sign),
                 .Dw_dm(w_dm),
                 .Dw_hi(),.Dw_lo(),
                 .Dw_rf(w_rf),
                 .clk(clk),.rst(rst),.wena(),
                 .Ealuc(Ealuc),
                 .Ea(Ea),.Eb(Eb),
                 .Easource(Easource),.Ebsource(Ebsource),
                 .Erfsource(Erfsource_in),.Ern(Ern),
                 .Esign(Esign),
                 .Ew_dm(Ew_dm),
                 .Ew_hi(),.Ew_lo(),
```

```
                .Ew_rf(Ew_rf));
    wire Esign_o,Ew_dm_o,Ew_rf_o;
    wire [31:0] Ea_o,Eb_o;
    wire EisGoto_o;
    wire [31:0] Epc4_o;
    PipeEXE EXE(.aluc(Ealuc),.a(Ea),.b(Eb),
                .asource(Easource),.bsource(Ebsource),
                .cp0(),.cuttersource(),
                .div(),
                .hi(),.hisource(),
                .Ecounter(),
                .Ecp0(),.Ecuttersource(),
                .Ehi(),.Ehisource(),
                .EisGoto(EisGoto_o),
                .Elo(),.Elosource(),
                .Emuler_hi(),.Emuler_lo(),
                .Epc4(Epc4_o),
                .Eq(),.Er(),
                .Erfsource(Erfsource_o),.Ern(Ern_o),
                .imm(Eimm),
                .isGoto(EisGoto),
                .lo(),.losource(),
                .pc4(Epc4),
                .rfsource(Erfsource_in),.rn(Ern),
                .sign(Esign),
                .w_dm(Ew_dm),.w_hi(),.w_lo(),.w_rf(Ew_rf),
                .Ealu(Ealu_o),.Ea(Ea_o),.Eb(Eb_o),
                .Esign(Esign_o),
                .Ew_dm(Ew_dm_o),.Ew_hi(),.Ew_lo(),.Ew_rf(Ew_rf_o));

    wire [31:0] Mmuler_hi,Mmuler_lo;
    wire [31:0] Mq,Mr;
    wire [ 2:0] Mrfsource_in;
    wire Msign,Mw_dm,Mw_hi,Mw_lo;
    wire [31:0] Ma,Mb,Mcounter,Mcp0,Mpc4;
    wire [ 1:0] Mcuttersour;
    wire [31:0] Mhi,Mlo;
    wire [ 1:0] Mhisource,Mlosource;
    PipeEMreg EM(.Ealu(Ealu_o),
                 .Ea(Ea_o),.Eb(Eb_o),
                 .Ecounter(),
                 .Ecp0(),
```

```
                .Esign(Esign_o),
                .Ew_dm(Ew_dm_o),
                .Ew_hi(),.Ew_lo(),
                .Ew_rf(Ew_rf_o),
                .clk(clk),.rst(rst),.wena(),
                .Malu(Malu),
                .Ma(Ma),.Mb(Mb),
                .Mcounter(),
                .Mcp0(),
                .Mcuttersource(),
                .Mhi(),.Mhisource(),
                .Mlo(),.Mlosource(),
                 .Ecuttersource(),
                .Ehi(),.Ehisource(),
                .Elo(),.Elosource(),
                .Emuler_hi(),.Emuler_lo(),
                .Epc4(Epc4_o),
                .Eq(),.Er(),
                .Erfsource(Erfsource_o),.Ern(Ern_o),
                .Mmuler_hi(),.Mmuler_lo(),
                .Mpc4(Mpc4),
                .Mq(),.Mr(),
                .Mw_hi(),.Mw_lo(),
                .Mrfsource(Mrfsource_in),.Mrn(Mrn),
                .Msign(Msign),
                .Mw_dm(Mw_dm),
                .Mw_rf(Mw_rf));
  wire [ 1:0] Mhisource_o,Mlosource_o;
   wire [31:0] Mmuler_hi_o,Mmuler_lo_o;
   wire [31:0] Mq_o,Mr_o;
   wire [31:0] Ma_o,Mb_o,Mcounter_o,Mcp0_o,Mpc4_o;
   wire [ 1:0] Mcuttersour_o;
   wire [31:0] Mhi_o,Mlo_o;
   wire [31:0] Walu,Wa,Wb;
   wire [31:0] Wcounter_Wcp0,Wpc4;
   wire [ 1:0] Wcuttersour;
   wire [31:0] Whi,Wlo;
   wire [ 1:0] Whisource,Wlosource;
   wire [31:0] Wmuler_hi,Wmuler_lo;
   wire [31:0] Wq,Wr;
   wire [ 2:0] Wrfsource;
   wire [ 4:0] Wrn_in;
```

```
    wire [31:0] Wdm;
    wire Wsign,Ww_hi,Ww_lo,Ww_rf;
    wire Msign_o,Mw_hi_o,Mw_lo_o,Mw_rf_o;
    PipeMEMreg MEM(.clk(clk),
                .alu(Malu),.a(Ma),.b(Mb),
                .counter(),
                .cp0(),.cuttersource(),
                .w_dm(Mw_dm),.w_hi(),.w_lo(),.w_rf(Mw_rf),
                .Malu(Malu_o),.Ma(Ma_o),.Mb(Mb_o),
                .Mcounter(),.Mcp0(),
                .Mdm(Mdm_o),
                .Mhi(),.Mhisource(),
                .Mlo(),.Mlosource(),
                .Mmuler_hi(),.Mmuler_lo(),
                .Mpc4(Mpc4_o),.Mq(),.Mr(),
                .hi(),.hisource(),
                .lo(),.losource(),
                .muler_hi(),.muler_lo(),
                .pc4(Mpc4),.q(),.r(),
                .rfsource(Mrfsource_in),.rn(Mrn),
                .sign(Msign),
                .Mrfsource(Mrfsource_o),.Mrn(Mrn_o),
                .Msign(Msign_o),
                .Mw_hi(),.Mw_lo(),.Mw_rf(Mw_rf_o));
    PipeMWreg MW(.Malu(Malu_o),
                  .Ma(Ma_o),.Mb(Mb_o),
                  .Mcounter(),
                  .Mcp0(),
                  .Mcuttersource(),
                  .Mdm(Mdm_o),
                  .Mhi(),.Mhisource(),
                  .Mlo(),.Mlosource(),
                  .clk(clk),.rst(rst),.wena(),
                  .Walu(Walu),
                  .Wa(Wa),.Wb(Wb),
                  .Wcounter(),
                  .Wcp0(),
                  .Wdm(Wdm),
                  .Whi(),.Whisource(),
                  .Wlo(),.Wlosource(),
                  .Wmuler_hi(),.Wmuler_lo(),
                  .Mmuler_hi(),.Mmuler_lo(),
```

```
                .Mpc4(Mpc4_o),.Mq(),.Mr(),
                .Mrfsource(Mrfsource_o),.Mrn(Mrn_o),
                .Mw_hi(),.Mw_lo(),.Mw_rf(Mw_rf_o),
                .Wpc4(Wpc4),
                .Wq(),.Wr(),
                .Wrfsource(Wrfsource),.Wrn(Wrn_in),
                .Ww_hi(),.Ww_lo(),
                .Ww_rf(Ww_rf));
    PipeWB WB(.alu(Walu),.a(Wa),.b(Wb),
              .counter(),.cp0(),
              .dm(Wdm),
              .pc4(Wpc4),.q(),.r(),
              .rfsource(Wrfsource),.rn(Wrn_in),
              .w_hi(),.w_lo(),.w_rf(Ww_rf),
              .Wdata_hi(Wdata_hi),.Wdata_lo(Wdata_lo),.Wdata_rf(Wdata_r
f),
              .hi(),.hisource(),
              .lo(),.losource(),
              .muler_hi(),.muler_lo(),
              .Wrn(Wrn),
              .Ww_hi(Wena_hi),.Ww_lo(Wena_lo),.Ww_rf(Wena_rf));
    assign _pc=pc;
    assign _inst=Dinst_in;
    assign _ealu=Ealu_o;
    assign _malu=Malu_o;
    assign _walu=Walu;
    ///test
    assign inst_t=inst;
    assign pc_t=pc;
endmodule
```

```
module PipeDEreg(
    input [ 3:0] Daluc              ,
    input [31:0] Da                 ,//pipeID??Rsout
    input [31:0] Db                 ,//pipeID??Rsout
    input Dasource                  ,
    input Dbsource                  ,
    input [31:0] Dcp0               ,
    input [ 4:0] Drn                ,//pipeID??rn
    input Dsign                     ,
    input Dw_dm                     ,
```

```
    input Dw_hi                     ,
    input Dw_lo                     ,
    input Dw_rf                     ,
    input clk                       ,
    input rst                       ,
    input wena                      ,
    input [ 1:0] Dcuttersource      ,
    input Ddiv                      ,
    input [31:0] Dhi                ,
    input [ 1:0] Dhisource          ,
    input [31:0] Dimm               ,
    input DisGoto                   ,
    input [31:0] Dlo                ,
    input [ 1:0] Dlosource          ,
    input [31:0] Dpc4               ,
    input [ 2:0] Drfsource          ,
    output reg [ 1:0] Elosource     ,
    output reg [31:0] Epc4          ,
    output reg [ 2:0] Erfsource     ,
    output reg [ 4:0] Ern           ,
    output reg [ 3:0] Ealuc         ,
    output reg [31:0] Ea            ,
    output reg [31:0] Eb            ,
    output reg Easource             ,
    output reg Ebsource             ,
    output reg [31:0] Ecp0          ,
    output reg [ 1:0] Ecuttersource ,
    output reg Ediv                 ,
    output reg [31:0] Ehi           ,
    output reg [ 1:0] Ehisource     ,
    output reg [31:0] Eimm          ,
    output reg EisGoto              ,
    output reg [31:0] Elo           ,
    output reg Esign                ,
    output reg Ew_dm                ,
    output reg Ew_hi                ,
    output reg Ew_lo                ,
    output reg Ew_rf
    );
    always@(posedge rst or posedge clk)
    begin
        if(rst==1)
```

```
        begin
            Elo             <= 0;
            Elosource       <= 0;
            Epc4            <= 0;
            Erfsource       <= 0;
            Ern             <= 0;
            Esign           <= 0;
            Ew_dm           <= 0;
            Ew_hi           <= 0;
            Ew_lo           <= 0;
            Ew_rf           <= 0;
            Ealuc           <= 0;
            Ea              <= 0;
            Eb              <= 0;
            Easource        <= 0;
            Ebsource        <= 0;
            Ecp0            <= 0;
            Ecuttersource   <= 0;
            Ediv            <= 0;
            Ehi             <= 0;
            Ehisource       <= 0;
            Eimm            <= 0;
            EisGoto         <= 0;
        end
        else
        begin
            Elo             <= Dlo              ;
            Elosource       <= Dlosource        ;
            Epc4            <= Dpc4             ;
            Erfsource       <= Drfsource        ;
            Ern             <= Drn              ;
            Esign           <= Dsign            ;
            Ew_dm           <= Dw_dm            ;
            Ew_hi           <= Dw_hi            ;
            Ew_lo           <= Dw_lo            ;
            Ew_rf           <= Dw_rf            ;
            Ealuc           <= Daluc            ;
            Ea              <= Da               ;
            Eb              <= Db               ;
            Easource        <= Dasource         ;
            Ebsource        <= Dbsource         ;
            Ecp0            <= Dcp0             ;
```

```
            Ecuttersource   <= Dcuttersource    ;
            Ediv            <= Ddiv             ;
            Ehi             <= Dhi              ;
            Ehisource       <= Dhisource        ;
            Eimm            <= Dimm             ;
            EisGoto         <= DisGoto          ;
        end
    end
endmodule
```

```
module PipeDEreg(
    input [ 3:0] Daluc              ,
    input [31:0] Da                 ,//pipeID??Rsout
    input [31:0] Db                 ,//pipeID??Rsout
    input Dasource                  ,
    input Dbsource                  ,
    input [31:0] Dcp0               ,
    input [ 4:0] Drn                ,//pipeID??rn
    input Dsign                     ,
    input Dw_dm                     ,
    input Dw_hi                     ,
    input Dw_lo                     ,
    input Dw_rf                     ,
    input clk                       ,
    input rst                       ,
    input wena                      ,
    input [ 1:0] Dcuttersource      ,
    input Ddiv                      ,
    input [31:0] Dhi                ,
    input [ 1:0] Dhisource          ,
    input [31:0] Dimm               ,
    input DisGoto                   ,
    input [31:0] Dlo                ,
    input [ 1:0] Dlosource          ,
    input [31:0] Dpc4               ,
    input [ 2:0] Drfsource          ,
    output reg [ 1:0] Elosource     ,
    output reg [31:0] Epc4          ,
    output reg [ 2:0] Erfsource     ,
    output reg [ 4:0] Ern           ,
    output reg [ 3:0] Ealuc         ,
```

```
    output reg [31:0] Ea            ,
    output reg [31:0] Eb            ,
    output reg Easource             ,
    output reg Ebsource             ,
    output reg [31:0] Ecp0          ,
    output reg [ 1:0] Ecuttersource ,
    output reg Ediv                 ,
    output reg [31:0] Ehi           ,
    output reg [ 1:0] Ehisource     ,
    output reg [31:0] Eimm          ,
    output reg EisGoto              ,
    output reg [31:0] Elo           ,
    output reg Esign                ,
    output reg Ew_dm                ,
    output reg Ew_hi                ,
    output reg Ew_lo                ,
    output reg Ew_rf
    );
    always@(posedge rst or posedge clk)
    begin
        if(rst==1)
        begin
            Elo             <= 0;
            Elosource       <= 0;
            Epc4            <= 0;
            Erfsource       <= 0;
            Ern             <= 0;
            Esign           <= 0;
            Ew_dm           <= 0;
            Ew_hi           <= 0;
            Ew_lo           <= 0;
            Ew_rf           <= 0;
            Ealuc           <= 0;
            Ea              <= 0;
            Eb              <= 0;
            Easource        <= 0;
            Ebsource        <= 0;
            Ecp0            <= 0;
            Ecuttersource   <= 0;
            Ediv            <= 0;
            Ehi             <= 0;
            Ehisource       <= 0;
```

```
            Eimm            <= 0;
            EisGoto         <= 0;
        end
        else
        begin
            Elo             <= Dlo              ;
            Elosource       <= Dlosource        ;
            Epc4            <= Dpc4             ;
            Erfsource       <= Drfsource        ;
            Ern             <= Drn              ;
            Esign           <= Dsign            ;
            Ew_dm           <= Dw_dm            ;
            Ew_hi           <= Dw_hi            ;
            Ew_lo           <= Dw_lo            ;
            Ew_rf           <= Dw_rf            ;
            Ealuc           <= Daluc            ;
            Ea              <= Da               ;
            Eb              <= Db               ;
            Easource        <= Dasource         ;
            Ebsource        <= Dbsource         ;
            Ecp0            <= Dcp0             ;
            Ecuttersource   <= Dcuttersource    ;
            Ediv            <= Ddiv             ;
            Ehi             <= Dhi              ;
            Ehisource       <= Dhisource        ;
            Eimm            <= Dimm             ;
            EisGoto         <= DisGoto          ;
        end
    end
endmodule
```

```
module PipeEXE(aluc,a,b,
               asource,bsource,
               rfsource,rn,
               cp0,cuttersource,
               div,
               hi,hisource,
               imm,
               isGoto,
               lo,losource,
               pc4,
```

```
                sign,
                w_dm,w_hi,w_lo,w_rf,
                Ealu,Ea,Eb,
                Epc4,
                Eq,Er,
                Erfsource,Ern,
                Esign,
                Ecounter,
                Ecp0,Ecuttersource,
                Ehi,Ehisource,
                EisGoto,
                Elo,Elosource,
                Emuler_hi,Emuler_lo,
                Ew_dm,Ew_hi,Ew_lo,Ew_rf);
    input [ 3:0] aluc           ;
    input [31:0] a              ;
    input [31:0] b              ;
    input asource               ;
    input bsource               ;
/**/input [31:0] cp0            ;
/**/input [ 1:0] cuttersource   ;
/**/input div                   ;
/**/input [31:0] hi             ;
/**/input [ 1:0] hisource       ;
    input [31:0] imm            ;
/**/input isGoto                ;
/**/input [31:0] lo             ;
/**/input [ 1:0] losource       ;
    input [31:0] pc4            ;
    input [ 2:0] rfsource       ;
    input [ 4:0] rn             ;
    input sign                  ;
    input w_dm                  ;
/**/input w_hi                  ;
/**/input w_lo                  ;
    input w_rf                  ;
    output [31:0] Ealu          ;
    output [31:0] Ea            ;
    output [31:0] Eb            ;
    output [31:0] Ecounter      ;
    output [31:0] Ecp0          ;
/**/output [ 1:0] Ecuttersource ;
```

```
/**/output [31:0] Ehi           ;
/**/output [ 1:0] Ehisource     ;
/**/output EisGoto              ;
/**/output [31:0] Elo           ;
/**/output [ 1:0] Elosource     ;
/**/output [31:0] Emuler_hi     ;
/**/output [31:0] Emuler_lo     ;
    output [31:0] Epc4          ;
/**/output [31:0] Eq            ;
/**/output [31:0] Er            ;
/**/output [ 2:0] Erfsource     ;
    output [ 4:0] Ern           ;
    output Esign                ;
    output Ew_dm                ;
/**/output Ew_hi                ;
/**/output Ew_lo                ;
    output Ew_rf                ;

    wire [31:0] alua,alub;
    wire [31:0] ext5;
    assign ext5 = {27'b0000_0000_0000_0000_0000_0000_000,imm[10:6]};


    //mux2x32(a,b,s,y);
    mux2x32 alu_a(.a(a),.b(ext5),.s(asource),.y(alua));
    mux2x32 alu_b(.a(b),.b(imm),.s(bsource),.y(alub));
    assign Ern = rn|{5{isGoto}};//jal 写 31 号寄存器
    wire [31:0] pc8;
    cla32 pc8_(.a(pc4),.b(32'h4),.sub(1'b0),.s(pc8));//pc4+4,jal 的跳转地
址
    assign Esign=sign;
    assign Ew_dm=w_dm;
    assign Ew_rf=w_rf;
    assign Ea=a;
    assign Eb=b;
    assign EisGoto=isGoto;
    assign Erfsource=rfsource;
    assign Epc4=pc4;
    wire [31:0] alu_temp;//ALU 的输出
    ALU alu_unit(.a(alua),.b(alub),.aluc(aluc),.r(alu_temp),.zero(),.ca
rry(),.negative(),.overflow());
    mux2x32 alu_res(.a(alu_temp),.b(pc8),.s(isGoto),.y(Ealu));
```

```
endmodule
```

```
module PipeID(EisGoto,Ern,Ew_hi,Ew_lo,Ew_rf,
              Mrn,Mw_rf,
              Wdata_hi,Wdata_lo,Wdata_rf,
              Wena_hi,Wena_lo,Wena_rf,Wrn,
              Ealu,Malu,
              Mdm,
              Erfsource,Mrfsource,
              clk,inst,pc4,rst,
              CP0out,
              losource,pcsource,
              reg28,
              rfsource,rn,rpc,
              sign,stall,
              Dpc4,
              Hiout,Loout,Rsout,Rtout,
              aluc,asource,bsource,
              bpc,cpc,
              cuttersource,
              div,
              hisource,
              imm,isGoto,
              jpc,
              w_dm,w_hi,w_lo,w_rf);
    input EisGoto               ;//从 EXE 传回的 isGoto(ejal)
    input [ 4:0] Ern            ;//
    input Ew_hi                 ;//写 hi 使能信号
    input Ew_lo                 ;//写 lo 使能信号
    input Ew_rf                 ;//写 rf 使能信号,ewreg
    input [2:0] Erfsource       ;
    input [ 4:0] Mrn            ;//
    input Mw_rf                 ;//写 rf 的信号，mwreg
    input [2:0] Mrfsource       ;
    input [31:0] Wdata_hi       ;//写入 hi 的数?
    input [31:0] Wdata_lo       ;//写入 lo 的数?
    input [31:0] Wdata_rf       ;//写入 rf 的数?,会在 WB 周期返回，写?rn
    input Wena_hi               ;//hi 写使能信?
    input Wena_lo               ;//lo 写使能信?
    input Wena_rf               ;//rf 写使能信?
    input [ 4:0] Wrn            ;//Wrn
```

```
    input clk                   ;//时钟
    input [31:0] inst           ;//指令
    input [31:0] pc4            ;//pc+4
    input rst                   ;//复位信号
    input [31:0] Ealu           ;//前推 alu 的结果
    input [31:0] Malu           ;//前推 alu 的结果
    input [31:0] Mdm            ;
    output [31:0] CP0out        ;//cp0 输出
    output [31:0] Dpc4          ;//pc+4 输出
    output [31:0] Hiout         ;//Hi 寄存器输出
    output [31:0] Loout         ;//Lo 寄存器输出
    output [31:0] Rsout         ;//Rs 输出
    output [31:0] Rtout         ;//Rt 输出
    output [ 3:0] aluc          ;//alu 控制信号
    output asource              ;//alu 的 a 选择信号,shift
    output bsource              ;//alu 的 b 选择信号,aluimm
    output [31:0] bpc           ;//beq 的跳转 pc
    output [31:0] cpc           ;//
    output [ 1:0] cuttersource  ;//
    output div                  ;//
    output [ 1:0] hisource      ;//hi 的多选择信号
    output [ 1:0] losource      ;//lo 的多选择信号
    output [31:0] imm           ;//扩展后的立即数
    output isGoto               ;//jal
    output [31:0] jpc           ;//跳转的 pc
    output [ 2:0] pcsource      ;//pc 的多选择信号有5种，需要3位
    output [31:0] reg28         ;//
    output [ 2:0] rfsource      ;//写回 rf 的来源有多种
    output [ 4:0] rn            ;//rt,rd 选择得到的结果
    output [31:0] rpc           ;//寄存器堆得到的下一条指令
    output sign                 ;//符号位
    output stall                ;//停顿
    output w_hi                 ;//写 hi 信号  ,1 写 0 不写
    output w_lo                 ;//写 lo 信号  ,1 写 0 不写
    output w_dm                 ;//写 dmem 信号,1 写 0 不写(wmem)
    output w_rf                 ;//写 rf 信号  ,1 写 0 不写(wreg)
    wire [ 5:0] op,func;//op and func
    wire [ 4:0] rs,rt,rd;
    wire reg_rt;
    wire sext;
    wire zero;
    wire [31:0] qa,qb;
```

```
    wire [1:0] fwda,fwdb;
    wire delay;
    assign zero = (Rsout==Rtout);
    assign func = inst[ 5: 0];
    assign op   = inst[31:26];
    assign rs   = inst[25:21];
    assign rt   = inst[20:16];
    assign rd   = inst[15:11];
    pipeIDcu cu(.op1(rs),.op2(rt),.op(op),.func(func),.rd(rd),.zero(zero),
                .EisGoto(EisGoto),
                .Erfsource(Erfsource),.Mrfsource(Mrfsource),
                .Ew_rf(Ew_rf),.Mw_rf(Mw_rf),
                .Ern(Ern),.Mrn(Mrn),
                .isGoto(isGoto),
                .aluc(aluc),.asource(asource),.bsource(bsource),
                .pcsource(pcsource),.rfsource(rfsource),
                .w_dm(w_dm),.w_rf(w_rf),.reg_rt(reg_rt),
                .sext(sext),.stall(stall),
                .fwda(fwda),.fwdb(fwdb),
                .delay(delay));
    mux2x5 des_reg(.a(rd),.b(rt),.ena(reg_rt),.o(rn));
    wire [16:0] ext16;//16 位扩展
    wire [31:0] br_offset;//寄存器的输出 qa,qb,branch_offset 分支偏移
    wire e;
    assign e = sext&inst[15];//符号扩展
    assign ext16 = {16{e}};
    regfile rf (.clk(~clk),.rst(rst),.we(Wena_rf),.raddr1(rs),.raddr2(rt),
                .waddr(Wrn),.wdata(Wdata_rf),.rdata1(qa),.rdata2(qb),.reg28(reg28));
    mux4x32 a(.a(qa),.b(Ealu),.c(Malu),.d(Mdm),.pcsource(fwda),.y(Rsout));
    mux4x32 b(.a(qb),.b(Ealu),.c(Malu),.d(Mdm),.pcsource(fwdb),.y(Rtout));

    assign imm = {ext16,inst[15:0]};//16 位扩展,32 位的立即
    assign br_offset = {imm[29:0],2'b00};//18 位扩展,branch_offset 分支偏移,跳转地址
    cla32 br_adr(.a(pc4),.b(br_offset),.sub(1'b0),.s(bpc));//beq，bne 的跳转地址
    CP0 cp0(.clk(clk),.rst(rst),.wsta(wsta),.wcau(wcau),
```

```
            .wepc(wepc),.mfc0(mfc0),.mtc0(mtc0),.exc(exc),
            .inta(inta),.pc(pc),.npc(npc),.mux1(mux1),
            .wdata(wdata),.alu_mem(alu_mem),.cause(cause),
            .selpc(selpc),.sta(sta),.rdata(rdata),
            .exc_addr(exc_addr));

    assign rpc   = Rsout-32'h00400000;
    assign jpc   = {pc4[31:28],inst[25:0],2'b00};
    assign Dpc4  = pc4;
    assign sign  = sext;

    Reg hi(.data_in(Wdata_hi),.clk(clk),.rst(rst),.wena(w_hi),.data_out
(Hiout));
    Reg lo(.data_in(Wdata_lo),.clk(clk),.rst(rst),.wena(w_lo),.data_out
(Loout));
endmodule
module pipeIDcu(op1,op2,op,func,rd,zero,
                EisGoto,
                Ew_rf,Mw_rf,
                Ern,Mrn,
                Erfsource,Mrfsource,
                isGoto,
                aluc,asource,bsource,
                pcsource,rfsource,
                w_dm,w_rf,reg_rt,
                sext,stall,
                fwda,fwdb,
                sext,stall,
                delay);
    input EisGoto           ;
    input Ew_rf             ;
    input Mw_rf             ;
    input [4:0]Ern          ;
    input [4:0] op1         ;
    input [4:0] op2         ;
    input [5:0] op          ;
    input [5:0] func        ;
    input [4:0] rd          ;
    input zero              ;
    input [4:0]Mrn          ;
    input [2:0] Erfsource   ;
    input [2:0] Mrfsource   ;
```

```
    output isGoto           ;
    output [3:0] aluc       ;
    output asource          ;
    output bsource          ;
    output [2:0] pcsource   ;
    output [2:0] rfsource   ;
    output w_dm             ;
    output w_rf             ;
    output reg_rt           ;
    output sext             ;
    output stall            ;
    output reg [1:0] fwda   ;
    output reg [1:0] fwdb   ;
    output delay            ;
    //31ÌõÖ¸ÁîÒëÂë
    wire r_type=~|op;
    wire i_addi = ~op[5] & ~op[4] &  op[3] & ~op[2] & ~op[1] & ~op[0];
    wire i_addiu= ~op[5] & ~op[4] &  op[3] & ~op[2] & ~op[1] &  op[0];
    wire i_andi = ~op[5] & ~op[4] &  op[3] &  op[2] & ~op[1] & ~op[0];
    wire i_ori  = ~op[5] & ~op[4] &  op[3] &  op[2] & ~op[1] &  op[0];
    wire i_xori = ~op[5] & ~op[4] &  op[3] &  op[2] &  op[1] & ~op[0];
    wire i_lw   =  op[5] & ~op[4] & ~op[3] & ~op[2] &  op[1] &  op[0];
    wire i_sw   =  op[5] & ~op[4] &  op[3] & ~op[2] &  op[1] &  op[0];
    wire i_beq  = ~op[5] & ~op[4] & ~op[3] &  op[2] & ~op[1] & ~op[0];
    wire i_bne  = ~op[5] & ~op[4] & ~op[3] &  op[2] & ~op[1] &  op[0];
    wire i_slti = ~op[5] & ~op[4] &  op[3] & ~op[2] &  op[1] & ~op[0];
    wire i_sltiu= ~op[5] & ~op[4] &  op[3] & ~op[2] &  op[1] &  op[0];
    wire i_lui  = ~op[5] & ~op[4] &  op[3] &  op[2] &  op[1] &  op[0];
    wire i_j    = ~op[5] & ~op[4] & ~op[3] & ~op[2] &  op[1] & ~op[0];
    wire i_jal  = ~op[5] & ~op[4] & ~op[3] & ~op[2] &  op[1] &  op[0];
    /******************/
    wire i_sll  = r_type & ~func[5] & ~func[4] & ~func[3] &~func[2] & ~
func[1] & ~func[0];
    wire i_srl  = r_type & ~func[5] & ~func[4] & ~func[3] &~func[2] &
func[1] & ~func[0];
    wire i_sra  = r_type & ~func[5] & ~func[4] & ~func[3] &~func[2] &
func[1] &  func[0];
    wire i_sllv = r_type & ~func[5] & ~func[4] & ~func[3] & func[2] & ~
func[1] & ~func[0];
    wire i_srlv = r_type & ~func[5] & ~func[4] & ~func[3] & func[2] &
func[1] & ~func[0];
```

```
    wire i_srav = r_type & ~func[5] & ~func[4] & ~func[3] & func[2] &
func[1] &  func[0];
    wire i_jr   = r_type & ~func[5] & ~func[4] &  func[3] &~func[2] & ~
func[1] & ~func[0];
     wire i_add  = r_type &  func[5] & ~func[4] & ~func[3] &~func[2] &
~func[1] & ~func[0];
    wire i_addu = r_type &  func[5] & ~func[4] & ~func[3] &~func[2] & ~
func[1] &  func[0];
    wire i_sub  = r_type &  func[5] & ~func[4] & ~func[3] &~func[2] &
func[1] & ~func[0];
    wire i_subu = r_type &  func[5] & ~func[4] & ~func[3] &~func[2] &
func[1] &  func[0];
    wire i_and  = r_type &  func[5] & ~func[4] & ~func[3] & func[2] & ~
func[1] & ~func[0];
    wire i_or   = r_type &  func[5] & ~func[4] & ~func[3] & func[2] & ~
func[1] &  func[0];
    wire i_xor  = r_type &  func[5] & ~func[4] & ~func[3] & func[2] &
func[1] & ~func[0];
    wire i_nor  = r_type &  func[5] & ~func[4] & ~func[3] & func[2] &
func[1] &  func[0];
    wire i_slt  = r_type &  func[5] & ~func[4] &  func[3] &~func[2] &
func[1] & ~func[0];
    wire i_sltu = r_type &  func[5] & ~func[4] &  func[3] &~func[2] &
func[1] &  func[0];
    //54Ö¸ÁîÒëÂë

    wire i_eret   = c0_type& op1[4]&~op1[3] &~op1[2]&~op1[1]&~op1[0]&~f
unc[5]& func[4] & func[3]&~func[2] & ~func[1] & ~func[0];
     wire c0_type  = ~op[5] &  op[4] & ~op[3] & ~op[2] & ~op[1] & ~op[0
];
    wire i_mfc0   = c0_type&~op1[4]&~op1[3] &~op1[2]&~op1[1]&~op1[0];
    wire i_mtc0   = c0_type&~op1[4]&~op1[3] & op1[2]&~op1[1]&~op1[0];
    wire i_divu     = r_type & ~func[5] &  func[4] &  func[3] &~func[2]
 &  func[1] &  func[0];
    wire i_div      = r_type & ~func[5] &  func[4] &  func[3] &~func[2]
 &  func[1] & ~func[0];
    wire i_multu    = r_type & ~func[5] &  func[4] &  func[3] &~func[2]
 & ~func[1] &  func[0];
    wire i_mfhi     = r_type & ~func[5] &  func[4] & ~func[3] &~func[2]
 & ~func[1] & ~func[0];
    wire i_mflo     = r_type & ~func[5] &  func[4] & ~func[3] &~func[2]
 &  func[1] & ~func[0];
```

```
    wire i_mthi     = r_type & ~func[5] &  func[4] & ~func[3] &~func[2]
 & ~func[1] &  func[0];
    wire i_mtlo     = r_type & ~func[5] &  func[4] & ~func[3] &~func[2]
 &  func[1] &  func[0];
    wire i_syscall= r_type & ~func[5] & ~func[4] &  func[3] & func[2] &
 ~func[1] & ~func[0];
    wire i_break  = r_type & ~func[5] & ~func[4] &  func[3] & func[2] &
 ~func[1] &  func[0];
    wire i_teq    = r_type &  func[5] &  func[4] & ~func[3] & func[2] &
 ~func[1] & ~func[0];
    wire i_jalr   = r_type & ~func[5] & ~func[4] &  func[3] &~func[2] &
 ~func[1] &  func[0];
    /*****************************************/
     wire i_lb       = op[5] & ~op[4] & ~op[3] & ~op[2] & ~op[1] & ~op[
0];
    wire i_lbu      = op[5] & ~op[4] & ~op[3] &  op[2] & ~op[1] & ~op[0
];
    wire i_lh       = op[5] & ~op[4] & ~op[3] & ~op[2] & ~op[1] &  op[0
];
    wire i_lhu      = op[5] & ~op[4] & ~op[3] &  op[2] & ~op[1] &  op[0
];
    wire i_sb       = op[5] & ~op[4] &  op[3] & ~op[2] & ~op[1] & ~op[0
];
    wire i_sh       = op[5] & ~op[4] &  op[3] & ~op[2] & ~op[1] &  op[0
];
    /*****************************************/
    wire i_clz      =~op[5] &  op[4] &  op[3] &  op[2] & ~op[1] & ~op[0
]&  func[5] & ~func[4] & ~func[3] &~func[2] & ~func[1] & ~func[0];
    wire i_mul      =~op[5] &  op[4] &  op[3] &  op[2] & ~op[1] & ~op[0
]& ~func[5] & ~func[4] & ~func[3] &~func[2] &  func[1] & ~func[0];
    wire i_bgez     =~op[5] & ~op[4] & ~op[3] & ~op[2] & ~op[1] &  op[0
]&  ~op2[4] & ~op2[3]  & ~op2[2]  &~op2[1]  & op2[0];
    assign pcsource[2]=1'b0;
    assign pcsource[1]=i_jr|i_j|i_jal|i_jalr;
    assign pcsource[0]=(i_beq&zero)|(i_bne&~zero)|i_j|i_jal;
    assign aluc[0]=i_sub|i_subu|i_or|i_nor|i_slt|i_srl|i_srlv|i_ori|i_b
eq|i_bne|i_slti|i_clz|i_bgez;
    assign aluc[1]=i_add|i_sub|i_xor|i_nor|i_slt|i_sltu|i_sll|i_sllv|i_
addi|i_xori|i_lw|i_sw|i_slti|i_sltiu|i_clz|i_lb|i_lbu|i_sb|i_lh|i_lhu|i
_sh;
    assign aluc[2]=i_and|i_or|i_xor|i_nor|i_sll|i_srl|i_sra|i_sllv|i_sr
lv|i_srav|i_andi|i_ori|i_xori|i_clz;
```

```
    assign aluc[3]=i_slt|i_sltu|i_sll|i_srl|i_sra|i_sllv|i_srlv|i_srav|
i_slti|i_sltiu|i_lui|i_clz|i_bgez;
    assign delay =(i_beq&zero)|(i_bne&~zero);
    assign isGoto = i_jal;
    assign rfsource[2]=1'b0;
    //assign rfsource[1]=1'b0;
    assign rfsource[1]=1'b0;
    assign rfsource[0]=i_lw|i_lb|i_lbu|i_lh|i_lhu;
    assign asource=i_sll|i_srl|i_sra;
    assign bsource=i_addi|i_andi|i_ori|i_xori|i_lw|i_lui|i_sw;
    assign sext=((i_addi|i_addiu)|(i_slti|i_sltiu))|((i_lui|i_lw)|i_sw)
|i_lh|i_lb|i_sh|i_sb|i_lbu|i_lhu|i_beq|i_bne;
    wire i_rs= i_add | i_sub | i_and | i_or | i_xor | i_jr | i_addi|
               i_andi| i_ori | i_xori| i_lw | i_sw  | i_beq| i_bne ;
    wire i_rt= i_add | i_sub | i_and | i_or | i_xor | i_sll | i_srl|
               i_sra | i_sw  | i_beq | i_bne ;

    assign w_dm=(i_sw|i_sb|i_sh)&(~stall);
    assign w_rf=(i_add|i_addu|i_sub|i_subu|i_and|i_or|i_xor|i_nor|
                 i_slt|i_sltu||i_sll|i_srl|i_sra|i_sllv|i_srlv|i_srav|
                 i_jr|i_addi|i_addiu|i_andi|i_ori|i_xori|i_lw|i_slti|
                 i_sltiu|i_lui|i_jal|i_mfc0|i_mfhi|i_mflo|i_jalr|i_clz|
                 i_lb|i_lbu|i_lh|i_lhu|i_mul)&(~stall);
    assign reg_rt=(((i_addi|i_addiu)|(i_andi|i_ori))|((i_xori|i_lw)|(i_
sw|i_beq)))|
                    ((i_bne|i_slti)|(i_sltiu|i_lui))|i_lh|i_lhu|i_lb|i_lb
u|i_sh|i_sb|i_mfc0;
    assign stall = (Ew_rf & Erfsource[0] & (Ern!=0) & ((i_rs&(Ern==op1)
) | (i_rt&(Ern==op2)));

    always@(Ew_rf or Mw_rf or Ern or Mrn or Erfsource[0] or Mrfsource[0
] or op1 or op2)
    begin
        fwda=2'b00;
        if(Ew_rf&(Ern!=0)&(Ern==op1)&~Erfsource[0])
        begin
            fwda=2'b01;
        end
        else
        begin
            if(Mw_rf&(Mrn!=0)&(Mrn==op1)&~Mrfsource[0])
            begin
```

```
                fwda=2'b10;
            end
            else
            begin
                if(Mw_rf&(Mrn!=0)&(Mrn==op1)&Mrfsource[0])
                begin
                fwda=2'b11;
                end
            end
        end
        fwdb=2'b00;
        if(Ew_rf&(Ern!=0)&(Ern==op2)&~Erfsource[0])
        begin
            fwdb=2'b01;
        end
        else
        begin
            if(Mw_rf&(Mrn!=0)&(Mrn==op2)&~Mrfsource[0])
            begin
                fwdb=2'b10;
            end
            else
            begin
                if(Mw_rf&(Mrn!=0)&(Mrn==op2)&Mrfsource[0])
                begin
                fwdb=2'b11;
                end
            end
        end
    end
endmodule
```

```
module PipeWB(alu,a,b,
              counter,cp0,
              dm,
              hi,hisource,
              lo,losource,
              muler_hi,muler_lo,
              pc4,q,r,
              rfsource,rn,
              w_hi,w_lo,w_rf,
```

```
            Wdata_hi,Wdata_lo,Wdata_rf,
            Wrn,
            Ww_hi,Ww_lo,Ww_rf);
    input w_hi                  ;
    input w_lo                  ;
    input w_rf                  ;
    input [31:0] alu            ;//alu 的输出，实际是 Malu
    input [31:0] a              ;
    input [31:0] b              ;
    input [31:0] counter        ;
    input [31:0] cp0            ;
    input [31:0] dm             ;
    input [31:0] hi             ;
    input [ 1:0] hisource       ;
    input [31:0] lo             ;
    input [ 1:0] losource       ;
    input [31:0] muler_hi       ;
    input [31:0] muler_lo       ;
    input [31:0] q              ;
    input [31:0] r              ;
    input [ 2:0] rfsource       ;
    input [31:0] pc4            ;
    input [ 4:0] rn             ;
    output Ww_hi                ;
    output Ww_lo                ;
    output Ww_rf                ;
    output [31:0] Wdata_hi      ;
    output [31:0] Wdata_lo      ;
    output [31:0] Wdata_rf      ;
    output [ 4:0] Wrn           ;
    mux2x32 alu_mem(.a(alu),.b(dm),.s(rfsource[0]),.y(Wdata_rf));
    assign Wrn=rn;
    assign Ww_rf=w_rf;
endmodule
```

其余部件代码与动态流水线相同