中山大学数据科学与计算机学院本科生实验报告

（2019 学年秋季学期）

课程名称：**计算机组成原理实验**　　任课教师：郭雪梅　　助教：丁文、汪庭葳

年级&班级	2019 级 04 班	专业(方向)	计算机科学与技术（超算方向）
学号	19335112	姓名	李钰
电话	19847352856	Email	1643589912@qq.com
开始日期	2020 年 10 月 23 日	完成日期	2020 年 11 月 6 日星期五

一、实验题目

ALU 功能的实现,以及符号位、标志位的扩展实现

二、实验目的

⑴ 了解运算器的组成结构。

⑵ 掌握运算器的工作原理。

⑶ 掌握数码管的工作原理与使用方法，学会 IP 核封装调用。

三、实验内容

1. 实验步骤

a. 创建display IP核

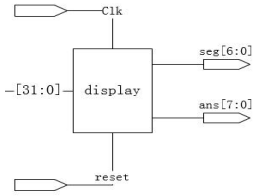
b. 创建调用IP核的工程，实现ALU功能

c. 扩展：加入符号位输出，以及可变宽度形式

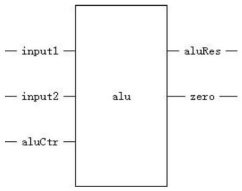
d. 扩展：增加标志位输出

2. 实验原理

display IP核效果图　　ALU输入输出形式　　ALU效果图



input	input1[31:0]
input	input2[31:0]
input	aluCtr[3:0]
output	aluRes[31:0]
output	zero



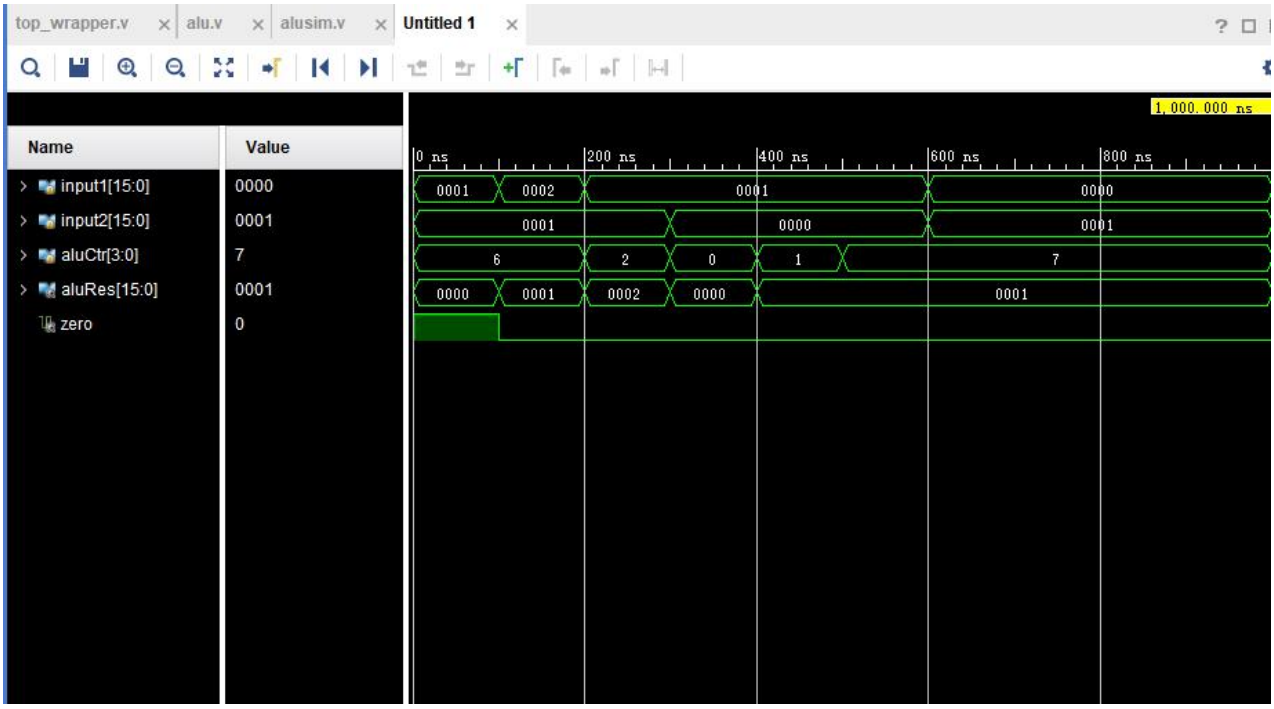
ALU功能图

0000	与
0001	或
1100	异或
0110	减
0010	加
0111	小于

四、实验结果

1. ALU 功能

仿真波形



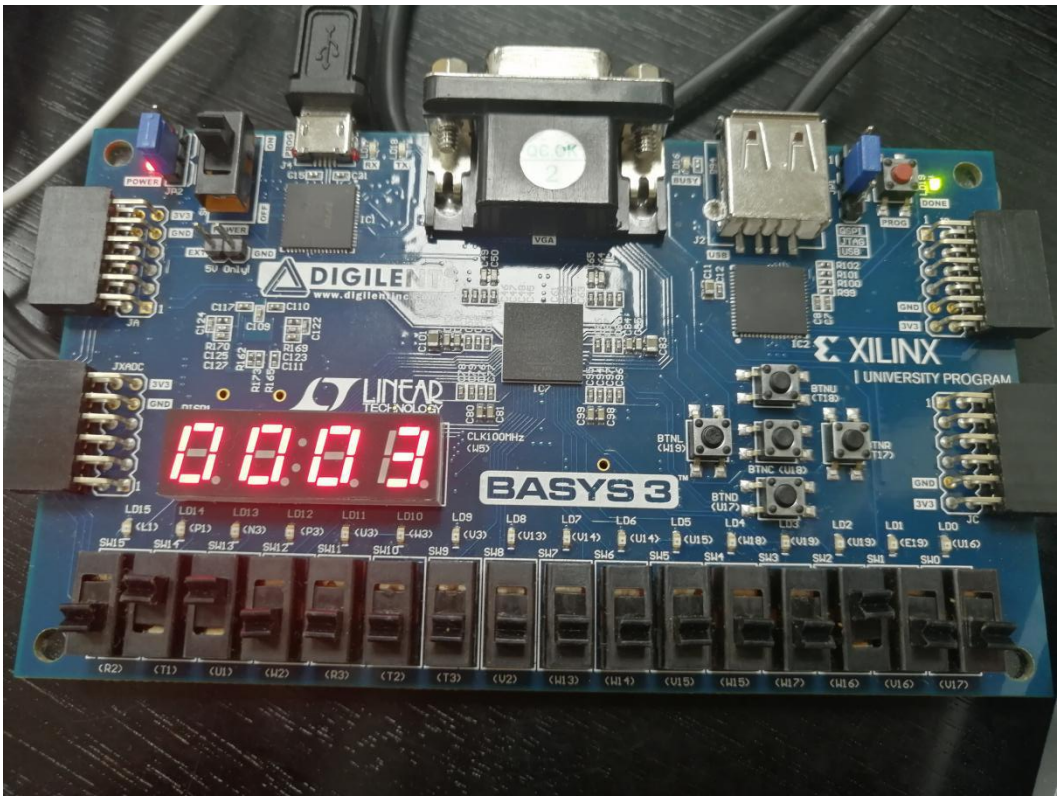
分析波形图

aluCtr = 6时　进行减法运算，　即0001 - 0001 = 0000；　0002 - 0001 = 0001

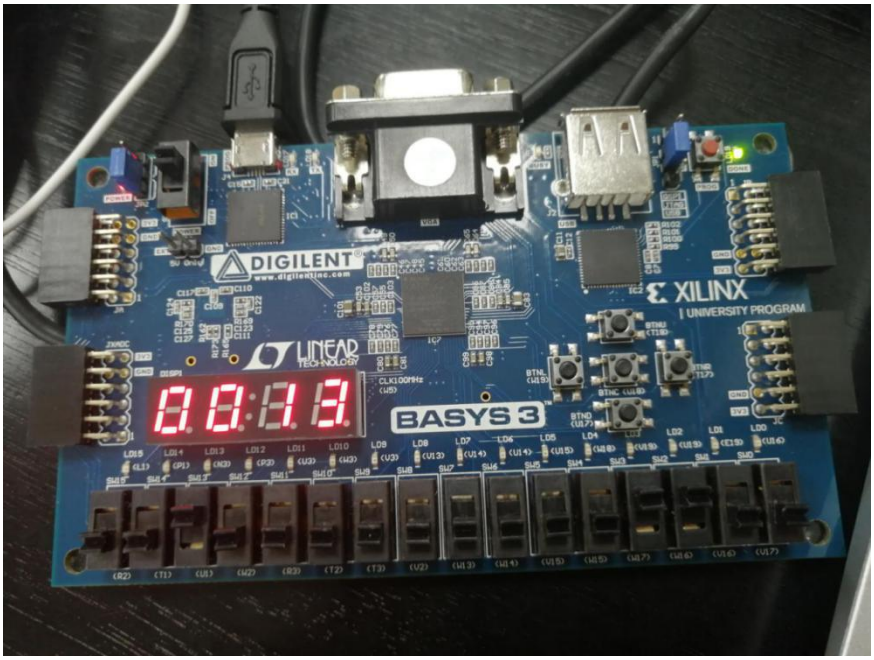
aluCtr = 2时　进行加法运算，　即0001 + 0001 = 0002

实物模拟

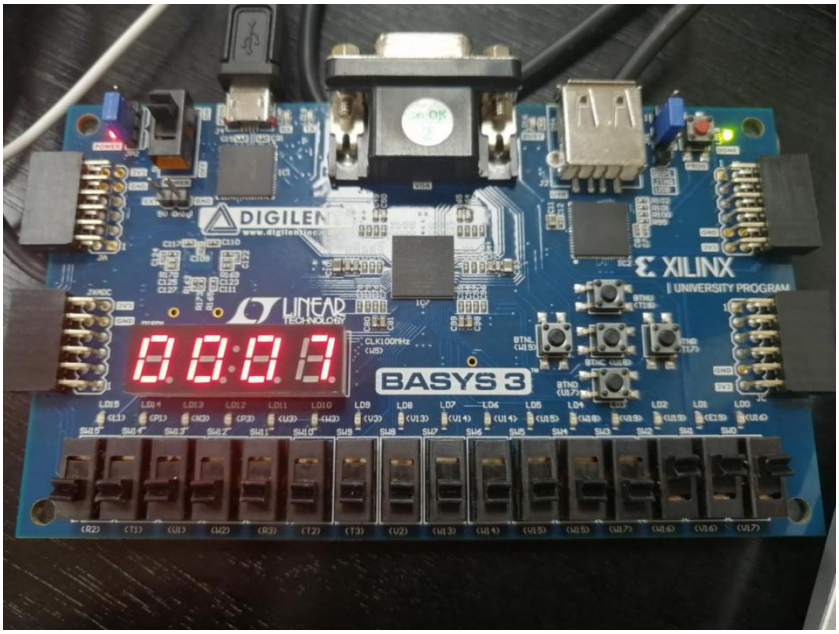
Input1初始设置为7



aluCtr = 0110 时，做减法运算， input2 = 4 ，最终结果为7 - 4 = 3



aluCtr = 0010时，做加法运算，input2 = 12， 结果为19， 十六进制显示13



AluCtr = 0000时，做与的运算，input2 为7，结果为7

扩展1 实现符号位扩展

仿真效果

扩展二　增加标志位输出

ZF, //0标志位，运算结果为0(全零)则置1，否则置0

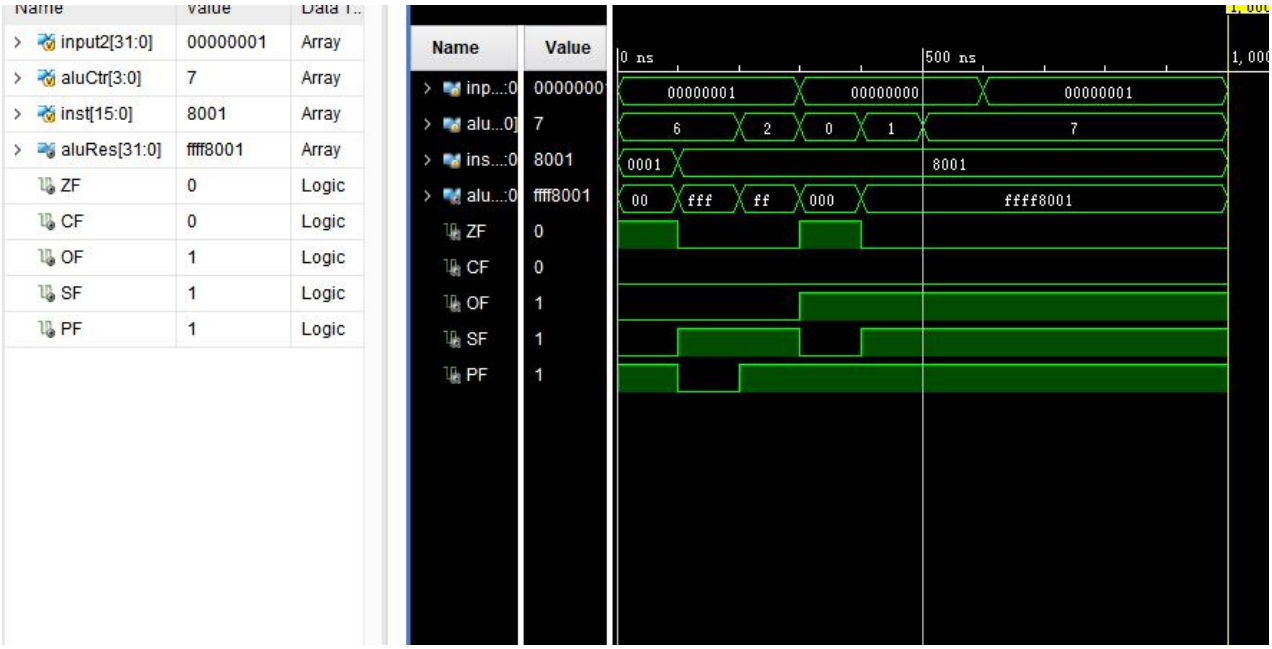
CF, //进借位标志位,

OF, //溢出标志位，对有符号数运算有意义，溢出则OF=1，否则为0

SF, //符号标志位，与F的最高位相同

PF; //奇偶标志位，F有奇数个1，则PF=1，否则为0

仿真如下



五、实验感想

通过这次实验，我掌握了数码管的构造原理，使用方法等；以及封装 IP 核的方法、调用 IP 核的相关操作，以及实现 alu 的基本功能，增添符号位扩展、标志位等功能。

做实验中遇到的问题：

① 调用 IP 核时，create wrapper 的是 block design 之后的文件，生成的 wrapper 文件可以被当作头文件，做适当修改后应用。

② 仿真时，要把新增的仿真文件设置成头文件。

③ 实物模拟时，input1 初始化的时候要在顶层文件中声明 wire input input1，再做赋值，不然赋值会失败。

④ 生成约束文件时，所有引脚都需要 fixed

⑤ 写标志位扩展时，要悉知标志位的计算方法

附录（流程图，注释过的代码）：

Ip 核代码：module display(clk,data,sm_wei,sm_duan);

```
input clk;

input [15:0] data;

output [3:0] sm_wei;

output [6:0] sm_duan;

//分频

integer clk_cnt;

reg clk_400Hz;

always @(posedge clk)

  if(clk_cnt==32'd100000)


begin clk_cnt <= 1'b0; clk_400Hz <= ~clk_400Hz;

end else clk_cnt <= clk_cnt + 1'b1;

//位控制


reg [3:0]wei_ctrl=4'b1110;

always @(posedge clk_400Hz)
```

```
wei_ctrl <= {wei_ctrl[2:0],wei_ctrl[3]}; //位控制

reg [3:0]duan_ctrl;


always @(wei_ctrl or data)

case(wei_ctrl)

4'b1110:duan_ctrl=data[3:0];

4'b1101:duan_ctrl=data[7:4];

4'b1011:duan_ctrl=data[11:8];

4'b0111:duan_ctrl=data[15:12];

default:duan_ctrl=4'hf;

  endcase


//解码模块

reg [6:0]duan;

always @(duan_ctrl)

case(duan_ctrl)

4'h0:duan=7'b100_0000;//0

4'h1:duan=7'b111_1001;//1

4'h2:duan=7'b010_0100;//2

4'h3:duan=7'b011_0000;//3

4'h4:duan=7'b001_1001;//4

4'h5:duan=7'b001_0010;//5

4'h6:duan=7'b000_0010;//6
```

```
4'h7:duan=7'b111_1000;//7

4'h8:duan=7'b000_0000;//8

4'h9:duan=7'b001_0000;//9

4'ha:duan=7'b000_1000;//a

4'hb:duan=7'b000_0011;//b

4'hc:duan=7'b100_0110;//c

4'hd:duan=7'b010_0001;//d

4'he:duan=7'b000_0111;//e

4'hf:duan=7'b000_1110;//f

// 4'hf:duan=7'b111_1111;//不显示

default : duan = 7'b100_0000;//0

endcase


//----------------------------------------------------------

assign sm_wei = wei_ctrl;

assign sm_duan = duan;

endmodule
```

alu 代码：

顶层文件：

```
`timescale 1 ps / 1 ps


module top_wrapper
  (
```

```
input clk,
input reset, //复位信号（连接一个按键）
output [6:0] seg,//段码
output [3:0] sm_wei,//哪个数码管
input [15:0] input2,
input [3:0] aluCtr
);
wire [15:0]input1;
//wire [15:0] input2;
assign input1 = 16'h0x7;
//assign input2[15:8] = {8{input2[7]}};
//wire aluCtr;
wire zero;
wire[15:0] aluRes;
alu alu(.input1(input1),
.input2(input2),
.aluCtr(aluCtr),
.zero(zero),
.aluRes(aluRes));
    top top_i
            (.clk_0(clk),
              .data_0(aluRes),
              .sm_duan_0(seg),
```

```
        .sm_wei_0(sm_wei));
endmodule



//wire[15:0] expand;
// ALU 控制信号线



// 实例化符号扩展模块
//signext signext(.inst(inst[15:0]), .data(expand));
//.....实例化数码管显示模块



alu：module alu(
input [15:0] input1,
input [15:0] input2,
input [3:0] aluCtr,
output reg[15:0] aluRes,
output reg zero
);

always @(input1 or input2 or aluCtr) // 运算数或控制码变化时操作
```

```
begin
case(aluCtr)
4'b0110: //  减
begin
aluRes = input1 - input2;
if(aluRes == 0)
zero = 1;
else
zero = 0;
end
4'b0010: //  加
aluRes = input1 + input2;
4'b0000: //  与
aluRes = input1 & input2;
4'b0001: //  或
aluRes = input1 | input2;
4'b1100: //  异或
aluRes = ~(input1 | input2); //这里我认为应该是(~input1 & input2) | (input1 & ~input2)
4'b0111: //  小于设置
begin
if(input1<input2)
aluRes = 1;
end
```

```
default:
aluRes = 0;
endcase
end
endmodule
```

内容拓展 1 代码：

Verilog 代码如下，细节见注释：

```
module signext(
input [15:0] inst, // 输入 16 位
output [31:0] data // 输出 32 位
);
// 根据符号补充符号位
// 如果符号位为 1，则补充 16 个 1，即 16’h ffff
// 如果符号位为 0，则补充 16 个 0
assign data = inst[15:15]?{16'hffff,inst}:{16'h0000,inst};
endmodule
```

alu：

```
module alu(
input [15:0] inst,
input [31:0] input2,
```

```
input [3:0] aluCtr,
output reg[31:0] aluRes,
output reg zero

);
wire [31:0]input1;
always @(input1 or input2 or aluCtr)
 // 运算数或控制码变化时操作
begin
case(aluCtr)
4'b0110: // 减
begin
aluRes = input1 - input2;
if(aluRes == 0)
zero = 1;
else
zero = 0;
end
4'b0010: // 加
aluRes = input1 + input2;
4'b0000: // 与
aluRes = input1 & input2;
4'b0001: // 或
```

```
aluRes = input1 | input2;

4'b1100: // 异或

aluRes = ~(input1 | input2);

4'b0111: // 小于设置

begin

if(input1<input2)

aluRes = 1;

end

default:

aluRes = 0;

endcase

end

signext signext(.inst(inst[15:0]), .data(input1));

endmodule
```

仿真：

```
module alusim;

// Inputs

reg [31:0] input2;

reg [3:0] aluCtr;

reg [15:0] inst; // 输入 16 位

// Outputs
```

```
wire [31:0] aluRes;

wire zero;


// Instantiate the Unit Under Test (UUT)

alu uut (

.inst(inst),

.input2(input2),

.aluCtr(aluCtr),

.aluRes(aluRes),

.zero(zero)

);

initial begin

// Initialize Inputs

input2 = 1;

aluCtr = 4'b0110;

inst=16'b0000_0000_0000_0001; //符号扩展的输入

#100;

input2 = 1;

aluCtr = 4'b0110;

inst=16'b1000_0000_0000_0001;

#100

input2 = 1;

aluCtr = 4'b0010;
```

```
#100
input2 = 0;
aluCtr = 4'b0000;
#100
input2 = 0;
aluCtr = 4'b0001;
#100
input2 = 0;
aluCtr = 4'b0111;
#100
input2 = 1;
aluCtr = 4'b0111;
end
endmodule
```

内容拓展 2 代码：

alu 修改：

```
module alu(
input [15:0] inst,
input [31:0] input2,
input [3:0] aluCtr,
output reg[31:0] aluRes,
```

```
output reg
ZF,CF,OF,SF,PF
);
wire [31:0]input1;
always @(input1 or input2 or aluCtr)
 // 运算数或控制码变化时操作
begin
case(aluCtr)
4'b0110: // 减
begin
{CF,aluRes}= input1 - input2;
if(aluRes == 0)
ZF = 1;
else
ZF = 0;//全为 0，则 ZF=1
OF = input1[31]^input2[31]^aluRes[31]^aluCtr;//溢出标志公式
SF = aluRes[31];//符号标志,取 F 的最高位
PF = ~^ aluRes;//奇偶标志，F 有奇数个 1，则 F=1；偶数个 1，则 F=0
end
4'b0010: // 加
begin
{CF,aluRes} = input1 + input2;
if(aluRes == 0)
```

```
ZF = 1;
else
ZF = 0;//全为 0，则 ZF=1
OF = input1[31]^input2[31]^aluRes[31]^aluCtr;//溢出标志公式
SF = aluRes[31];//符号标志,取 F 的最高位
PF = ~^ aluRes;//奇偶标志，F 有奇数个 1，则 F=1；偶数个 1，则 F=0
end
4'b0000: // 与
begin
aluRes = input1 & input2;
if(aluRes == 0)
ZF = 1;
else
ZF = 0; //全为 0，则 ZF=1
OF = input1[31]^input2[31]^aluRes[31]^aluCtr;//溢出标志公式
SF = aluRes[31];//符号标志,取 F 的最高位
PF = ~^ aluRes;//奇偶标志，F 有奇数个 1，则 F=1；偶数个 1，则 F=0
end
4'b0001: // 或
begin
aluRes = input1 | input2;
if(aluRes == 0)
ZF = 1;
```

```
else
ZF = 0;//全为 0，则 ZF=1
OF = input1[31]^input2[31]^aluRes[31]^aluCtr;//溢出标志公式
SF = aluRes[31];//符号标志,取 F 的最高位
PF = ~^ aluRes;//奇偶标志，F 有奇数个 1，则 F=1；偶数个 1，则 F=0
end
4'b1100: // 异或
begin
aluRes = ~(input1 | input2);
if(aluRes == 0)
ZF = 1;
else
ZF = 0;//全为 0，则 ZF=1
OF = input1[31]^input2[31]^aluRes[31]^aluCtr;//溢出标志公式
SF = aluRes[31];//符号标志,取 F 的最高位
PF = ~^ aluRes;//奇偶标志，F 有奇数个 1，则 F=1；偶数个 1，则 F=0
end
4'b0111: // 小于设置
begin
if(input1<input2)
aluRes = 1;
ZF = 0;//全为 0，则 ZF=1
OF = input1[31]^input2[31]^aluRes[31]^aluCtr;//溢出标志公式
```

```
SF = aluRes[31];//符号标志,取 F 的最高位
PF = ~^ aluRes;//奇偶标志，F 有奇数个 1，则 F=1；偶数个 1，则 F=0
end
default:
begin
aluRes = 0;
ZF = 0;//全为 0，则 ZF=1
OF = input1[31]^input2[31]^aluRes[31]^aluCtr;//溢出标志公式
SF = aluRes[31];//符号标志,取 F 的最高位
PF = ~^ aluRes;//奇偶标志，F 有奇数个 1，则 F=1；偶数个 1，则 F=0
end
endcase
end

signext signext(.inst(inst[15:0]), .data(input1));
endmodule
```

仿真：

```
module alusim;
// Inputs
reg [31:0] input2;
reg [3:0] aluCtr;
reg [15:0] inst; // 输入 16 位
```

```
// Outputs
wire [31:0] aluRes;
wire ZF;
wire CF;
wire OF;
wire SF;
wire PF;

// Instantiate the Unit Under Test (UUT)
alu uut (
.inst(inst),
.input2(input2),
.aluCtr(aluCtr),
.aluRes(aluRes),
.ZF(ZF),
.CF(CF),
.OF(OF),
.SF(SF),
.PF(PF)
);
initial begin
// Initialize Inputs
input2 = 1;
```

```
aluCtr = 4'b0110;

inst=16'b0000_0000_0000_0001; //符号扩展的输入

#100;

input2 = 1;

aluCtr = 4'b0110;

inst=16'b1000_0000_0000_0001;

#100

input2 = 1;

aluCtr = 4'b0010;

#100

input2 = 0;

aluCtr = 4'b0000;

#100

input2 = 0;

aluCtr = 4'b0001;

#100

input2 = 0;

aluCtr = 4'b0111;

#100

input2 = 1;

aluCtr = 4'b0111;

end

endmodule
```