CHIP HalfAdder {

IN a, b; // 1-bit inputs

OUT sum, // Right bit of a + b

carry; // Left bit of a + b

PARTS:

Xor(a=a, b=b, out=sum);

And(a=a, b=b, out=carry);

}

CHIP FullAdder {

IN a, b, c; // 1-bit inputs

OUT sum, // Right bit of a + b + c

carry; // Left bit of a + b + c

PARTS:

HalfAdder(a=a, b=b, sum=s1, carry=c1);

HalfAdder(a=s1, b=c, sum=sum, carry=c2);

Or(a=c1, b=c2, out=carry);

}

CHIP Add16 {

IN a[16], b[16];

OUT out[16];

PARTS:

HalfAdder(a=a[0], b=b[0], sum=out[0], carry=c0);

FullAdder(a=a[1], b=b[1], c=c0, sum=out[1], carry=c1);

FullAdder(a=a[2], b=b[2], c=c1, sum=out[2], carry=c2);

FullAdder(a=a[3], b=b[3], c=c2, sum=out[3], carry=c3);

FullAdder(a=a[4], b=b[4], c=c3, sum=out[4], carry=c4);

FullAdder(a=a[5], b=b[5], c=c4, sum=out[5], carry=c5);

FullAdder(a=a[6], b=b[6], c=c5, sum=out[6], carry=c6);

FullAdder(a=a[7], b=b[7], c=c6, sum=out[7], carry=c7);

FullAdder(a=a[8], b=b[8], c=c7, sum=out[8], carry=c8);

FullAdder(a=a[9], b=b[9], c=c8, sum=out[9], carry=c9);

FullAdder(a=a[10], b=b[10], c=c9, sum=out[10], carry=c10);

FullAdder(a=a[11], b=b[11], c=c10, sum=out[11], carry=c11);

FullAdder(a=a[12], b=b[12], c=c11, sum=out[12], carry=c12);

FullAdder(a=a[13], b=b[13], c=c12, sum=out[13], carry=c13);

FullAdder(a=a[14], b=b[14], c=c13, sum=out[14], carry=c14);

FullAdder(a=a[15], b=b[15], c=c14, sum=out[15], carry=msb);

}

CHIP Inc16 {

IN in[16];

OUT out[16];

PARTS:

Add16(a=in, b[0]=true, out=out);

}

CHIP ALU {

IN

x[16], y[16], // 16-bit inputs

zx,

nx,

zy,

ny,

f,

no;

OUT

out[16],

zr,

ng;

PARTS:

Mux16(a=x, b=false, sel=zx, out=zerox);

Not16(in=zerox, out=notx);

Mux16(a=zerox, b=notx, sel=nx, out=negatex);

Mux16(a=y, b=false, sel=zy, out=zeroy);

Not16(in=zeroy, out=noty);

Mux16(a=zeroy, b=noty, sel=ny, out=negatey);

And16(a=negatex, b=negatey, out=xandy);

Add16(a=negatex, b=negatey, out=xaddy);

Mux16(a=xandy, b=xaddy, sel=f, out=outputxy);

Not16(in=outputxy, out=notoutput);

Mux16(a=outputxy, b=notoutput, sel=no, out=out, out[0..7]=half1, out[8..15]=half2, out[15]=ng);

Or8Way(in=half1, out=zero1);

Or8Way(in=half2, out=zero2);

Or(a=zero1, b=zero2, out=zero3);

Not(in=zero3, out=zr);

}