`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company: Meow

// Engineer: Jamison Bauer, Aros Aziz, Kylee Flukiger, Dezeray Kowalski

//

// Create Date: 12:54:08 09/04/2018

// Design Name:

// Module Name: ALU

// Project Name: CPU

// Target Devices:

// Tool versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module ALU(In1, In2, Out, aluControl, Flags);

input [15:0] In1, In2;

input [15:0] aluControl;

reg [7:0] Opcode;

reg [7:0] Immediate;

output reg [15:0] Out;

output reg [4:0] Flags; // zero, carry, overflow, negative, low

integer bitPosition;

parameter Cin = 1'b1;

parameter ADD = 8'b0000\_0101;

parameter ADDI = 8'b0101\_xxxx;

parameter ADDU = 8'b0000\_0110;

parameter ADDUI = 8'b0110\_xxxx;

parameter ADDC = 8'b0000\_0111;

parameter ADDCU = 8'b0000\_0100;

parameter ADDCUI = 8'b0001\_xxxx;

parameter ADDCI = 8'b0111\_xxxx;

parameter SUB = 8'b0000\_1001;

parameter SUBI = 8'b1001\_xxxx;

parameter CMP = 8'b0000\_1011;

parameter CMPI = 8'b1011\_xxxx;

parameter CMPU = 8'b0000\_1101;

parameter CMPUI = 8'b0010\_xxxx;

parameter AND = 8'b0000\_0001;

parameter OR = 8'b0000\_0010;

parameter XOR = 8'b0000\_0011;

parameter NOT = 8'b0000\_1111;

parameter LSH = 8'b0000\_1000;

parameter LSHI = 8'b0011\_xxxx;

parameter RSH = 8'b0000\_1010;

parameter RSHI = 8'b1110\_xxxx;

parameter ALSH = 8'b0000\_1100;

parameter ARSH = 8'b0000\_1110;

parameter NOP = 8'b0000\_0000;

always @(In1, In2, aluControl)

begin

Flags = 5'b00000;

bitPosition = 0;

Opcode = {aluControl [15:12], aluControl [7:4]};

Immediate = {aluControl[11:8], aluControl [3:0]};

case (Opcode[7:4])

4'b0000: // non immediate instructions

case (Opcode[3:0])

ADD[3:0]: // for unsigned set overflow when set carry.

begin

{Flags[3], Out} = In1 + In2; ///Set carry this way for everything

if( (~In1[15] & ~In2[15] & Out[15]) | (In1[15] & In2[15] & ~Out[15]) ) Flags[2] = 1'b1;

end

ADDU[3:0]:

begin

{Flags[3],Out} = In1 + In2;

if (Out < In1 && Out < In2) Flags[2] = 1'b1; //set overflow

end

ADDC[3:0]: // TODO: can this set carry flag? if both inputs have a 1 in the msb than carry is 1

begin

{Flags[3],Out} = In1 + In2 + {15'b0, Cin};

if( (~In1[15] & ~In2[15] & Out[15]) | (In1[15] & In2[15] & ~Out[15]) ) Flags[2] = 1'b1;

end

ADDCU[3:0]:

begin

{Flags[3],Out} = In1 + In2 + {15'b0, Cin};

if (Out < In1 && Out < In2) Flags[2] = 1'b1; //set overflow

end

SUB[3:0]: // in msb 0-1, set carry

begin

Out = In1 - In2;

if ( (In1[15] & ~In2[15] & Out[15]) | (~In1[15] & In2[15] & Out[15]) ) Flags[2] = 1'b1;

//if (Out[15] == 1'b1) Flags[1] = 1'b1; //negative flag

if (In1 < In2) Flags[3] = 1'b1; // is this correct?

end

CMP[3:0]:

begin

if( $signed(In1) < $signed(In2) ) Flags[1:0] = 2'b11;

if( In1 == In2) Out = 8'b0000\_0000;

else Out = 8'b0000\_0001;

end

CMPU[3:0]:

begin

if( In1 < In2 ) Flags[1:0] = 2'b11;

if( In1 == In2) Out = 8'b0000\_0000;

else Out = 8'b0000\_0001;

end

AND[3:0]:

begin

Out = In1 & In2;

// Out = 16'b0000\_0000\_0000\_0000;

// for (bitPosition = 15; bitPosition < 0; bitPosition = bitPosition - 1)

// begin

// if ( In1[bitPosition] == 1'b1 & In2[bitPosition] == 1'b1) Out[bitPosition] = 1'b1;

// end

end

OR[3:0]:

begin

Out = In1 | In2;

// Out = 16'b0000\_0000\_0000\_0000;

// for (bitPosition = 15; bitPosition < 0; bitPosition = bitPosition - 1)

// begin

// if ( In1[bitPosition] == 1'b1 | In2[bitPosition] == 1'b1) Out[bitPosition] = 1'b1;

// end

end

XOR[3:0]:

begin

Out = In1 ^ In2;

// Out = 16'b0000\_0000\_0000\_0000;

// for (bitPosition = 15; bitPosition < 0; bitPosition = bitPosition - 1)

// begin

// if ( In1[bitPosition] == 1'b1 ^ In2[bitPosition] == 1'b1) Out[bitPosition] = 1'b1;

// end

end

NOT[3:0]:

begin

Out = ~In1;

end

LSH[3:0]:

begin

Out = In1 << 1;

end

RSH[3:0]:

begin

Out = In1 >> 1;

end

ALSH[3:0]:

begin

Out = In1 << 1;

end

ARSH[3:0]:

begin

if (In1[15] == 1'b1)

begin

Out = In1 >> 1;

Out[15] = 1'b1;

end

else

begin

Out = In1 >> 1;

end

end

NOP[3:0]:

begin

Out = 4'b0000;

Flags = 5'b00000;

end

default:

begin

Out = 4'b0000;

Flags = 5'b00000;

end

endcase

//immediate instructions

ADDI[7:4]:

begin

{Flags[3], Out} = In1 + {{8{Immediate[7]}}, {Immediate[7:0]}}; // 3.6 in text book

if( (~In1[15] & ~In2[15] & Out[15]) | (In1[15] & In2[15] & ~Out[15]) ) Flags[2] = 1'b1;

end

ADDUI[7:4]:

begin

{Flags[3], Out} = In1 + {{8'b0000\_0000}, {Immediate[7:0]}};

if (Out < In1 && Out < In2) Flags[3:2] = 2'b11; //set carry and overflow

// set carry flag

end

ADDCUI[7:4]:

begin

{Flags[3], Out} = In1 + {{8'b0000\_0000}, {Immediate[7:0]}} + 1'b1;

if (Out < In1 && Out < In2) Flags[3:2] = 2'b11; //set carry and overflow

end

ADDCI[7:4]:

begin

{Flags[3], Out} = In1 + {{8{Immediate[7]}}, {Immediate[7:0]}} + {15'b0, Cin};

if ( (~In1[15] & ~In2[15] & Out[15]) | (In1[15] & In2[15] & ~Out[15]) ) Flags[2] = 1'b1;

end

SUBI[7:4]:

begin

Out = In1 - {{8{Immediate[7]}}, {Immediate[7:0]}};

if ( (~In1[3] & ~In2[3] & Out[3]) | (In1[3] & In2[3] & ~Out[3]) ) Flags[2] = 1'b1;

//if (Out[15]) Flags[1] = 1'b1; //negative flag

if (In1 < In2) Flags[3] = 1'b1; // is this correct?

end

CMPI[7:4]:

begin

if( $signed(In1) < $signed(Immediate) ) Flags[1:0] = 2'b11;

if( In1 == In2) Out = 8'b0000\_0000;

else Out = 8'b0000\_0001;

end

CMPUI[7:4]: //same as previous?

begin

if( In1 < Immediate ) Flags[1:0] = 2'b11;

if( In1 == Immediate) Out = 8'b0000\_0000;

else Out = 8'b0000\_0001;

end

LSHI[7:4]: // shift by immediate specified amount

begin

Out = In1 << Immediate;

end

RSHI[7:4]: // shift by immediate specified amount

begin

Out = In1 >> Immediate;

end

default:

begin

Out = 4'b0000;

Flags = 5'b00000;

end

endcase

if (Out == 8'b0000\_0000 && Opcode != 8'b0000\_0000) //If output is 0, except for in nop case:

begin

Flags[4] = 1'b1;

end

end

endmodule