# **IRIS Labs Hardware Assignment I**

Q2.

a.)**Control Signals**

**beq:**

branch = 1

memRead = 0

memtoReg = X

ALUOp = 2'b01

memWrite = 0

ALUSrc = 0

regWrite = 0

**sw:**

branch = 0

memRead = 0

memtoReg = X

ALUOp = 2'b00

memWrite = 1

ALUSrc = 1

regWrite = 0

**Lw:**

branch = 0

memRead = 1

memtoReg = 1

ALUOp = 2'b00

memWrite = 0

ALUSrc = 1

regWrite = 1

b.)

**loop: slt x2,x0,x1**

**beq x2,x0,DONE**

**addi x1,x1,-1**

**addi x2,x2,2**

**j loop**

**Done:**

Initially x2 is set to 2 and x1 to 8. As x0 is not initialized, it is always 0.

Slt sets x2 to 1 if x0<x1.

Beq x2,x0,DONE sets pc to DONE when x2 and x0 are equal(they aren't here)

Addi x1,x1,-1 decrement x1 by 1

Addi x2,x2,2 adds 2 to x2, here since x2 was set to 1 now it becomes 3

J loop sets pc back to loop

The given code loops similarly as x1 decrements from 8 all the way to 0.

When x1 is 0 , slt x2,x0,x1 sets x2 to 0

Now as x2 and x0 are equal beq x2,x0,DONE sets pc to DONE and the execution ends.

So, the final value is x2 is always 0 and is independent of the initial value.

c.)

We can implement the functionality to count the number of trailing zeros by either setting ALUOp to an unused value or adding a custom signal CTZ to the control block. If we are considering a new signal then we can set all control signals to 0 except regWrite and CTZ which will be set to 1, CTZ can be directly connected to the ALU where can it can be implemented in the following manner.

module ALU (

input [3:0] ALUCtl,

input CTZ, // New control flag for CTZ operation

input [31:0] A, B,

output reg [31:0] ALUOut,

output zero

);

// Hardware implementable CTZ using a loop from 32 to 0

reg [4:0] ctz\_result;

integer i;

always @(\*) begin

if (CTZ) begin

ctz\_result = 32;

for (i = 31; i >= 0; i = i - 1) begin

if (A[i]) begin

ctz\_result = 31 - i;

break;

end

end

ALUOut = (A == 0) ? 32 : ctz\_result; // If input is zero, return 32

end else begin

case(ALUCtl)

………

endcase

end

end

// Zero flag is high when the ALU result is zero.

assign zero = (ALUOut == 32'd0);

endmodule