Control ROM

By Brian and Cody

Why is the component important?

- It drives the various operations of the processor
- It controls which components are enabled based on the instruction
- It takes the instruction sent from the instruction memory and uses it to direct all other components to accomplish the task

What instructions does the component support

- All of them
- ADD
- AND
- MOVL
- MOVS
- CMP
- JA
- JALR
- JALR 1 1
- This component is required for instructions to function because this is where instructions are processed

Implementation

- This component uses data flow style.
- Using data flow allows values to be assigned for the output based on the output from the decoder which is a set instruction
- We used conditional statements and the assign statement to assign the values

Test Cases

```
module CONTROLROMTEST();

reg [63:0] in;
wire [6:0] out;

CONTROLROM DUT(in, out);









initial
begin
$display("AND Input: 0000000000000000000000010000000000000000000000000000000000000000");
in <= 64'b0000000000000000000000010000000000000000000000000000000000000000;
#1000
$display("Output: %d%d%d%d%d%d%d", out[6], out[5], out[4], out[3], out[2], out[1], out[0]);

$display("ADD Input: 0000100000000000000000000000000000000000000000000000000000000000");
in <= 64'b0000100000000000000000000000000000000000000000000000000000000000;
#1000
$display("Output: %d%d%d%d%d%d%d", out[6], out[5], out[4], out[3], out[2], out[1], out[0]);

$display("MOVL Input: 0000000000100000000000000000000000000000000000000000000000000000");
in <= 64'b0000000000100000000000000000000000000000000000000000000000000000;
#1000
$display("Output: %d%d%d%d%d%d%d", out[6], out[5], out[4], out[3], out[2], out[1], out[0]);

$display("MOVS Input: 0000000000010000000000000000000000000000000000000000000000000000");
in <= 64'b0000000000010000000000000000000000000000000000000000000000000000;
#1000
$display("Output: %d%d%d%d%d%d%d", out[6], out[5], out[4], out[3], out[2], out[1], out[0]);

$display("CMP Input: 0000000000000000000000000000000000000000000000000000000000001000");
in <= 64'b0000000000000000000000000000000000000000000000000000000000001000;
#1000
$display("Output: %d%d%d%d%d%d%d", out[6], out[5], out[4], out[3], out[2], out[1], out[0]);

$display("JA Input: 0000000000000010000000000000000000000000000000000000000000000000");
in <= 64'b0000000000000010000000000000000000000000000000000000000000000000;
#1000
$display("Output: %d%d%d%d%d%d%d", out[6], out[5], out[4], out[3], out[2], out[1], out[0]);









end

endmodule
```

Test Cases

```
AND Input: 00000000000000000000001000000000000000000000000000000000000000
Output: 011110x
ADD Input: 00001000000000000000000000000000000000000000000000000000000000
Output: 011100x
MOVL Input: 00000000001000000000000000000000000000000000000000000000000000
Output: 0010010
MOVS Input: 00000000000100000000000000000000000000000000000000000000000000
Output: xx00011
CMP Input: 00000000000000000000000000000000000000000000000000000000001000
Output: xx01x0x
JA Input: 00000000000000100000000000000000000000000000000000000000000000
Output: xx0xx0x
```

Issues

- We had to learn the syntax for conditional statements, binary numbers, and the assign block
- We had to finish the decoder to get the values for the conditional statements

Gate Delay

- Structurally the Control ROM is seven NOR gates working in parallel. Each NOR has a delay of 2 and since the gates do not rely on each other the total delay for the Control ROM is 2.

The End