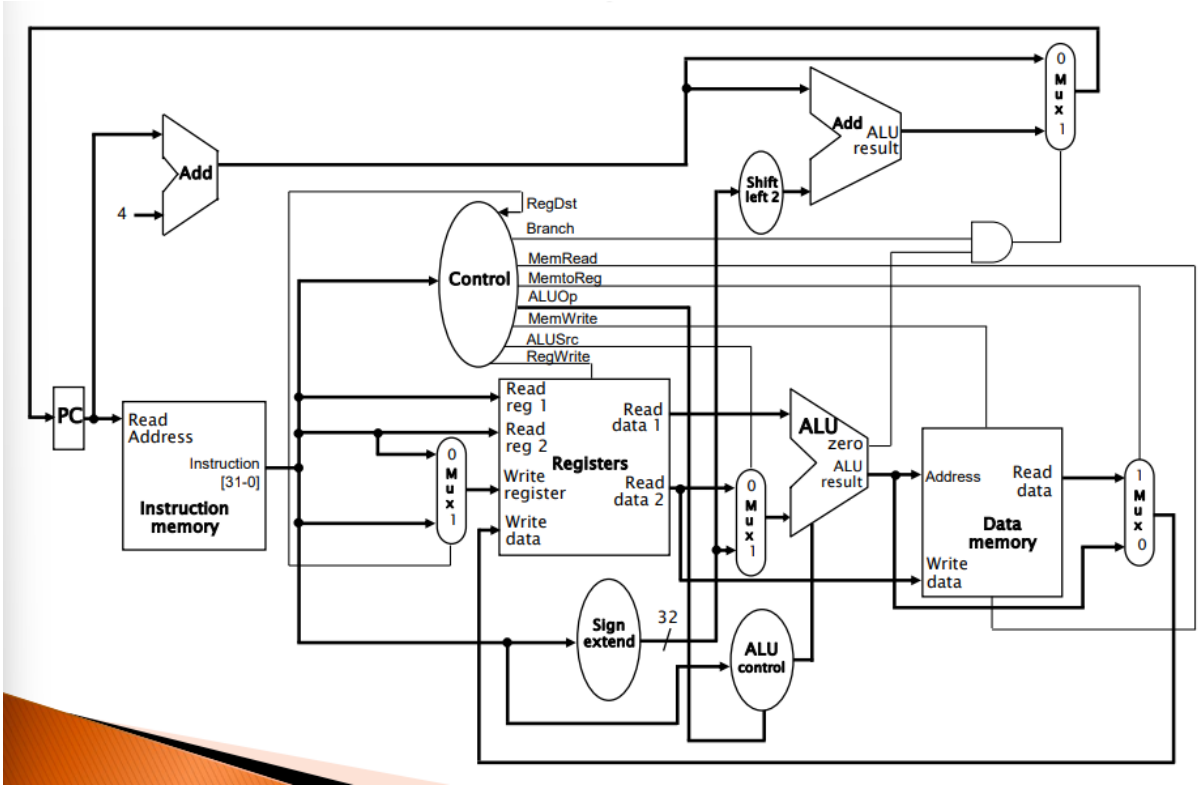
CSE 331 COMPUTER ORGANIZATION

FINAL PROJECT DOCUMENTATION

BARIŞ BATUHAN BOLAT

210104004029



- I try to implement above design of a Single Cycle Processor.
- I used ALU from HW2

TABLES FOR CONTROL UNITES

CONTROL UNIT

Instruction	oppcode	ALUop	RegDst	Branch	MemRead	MemtoReg	ALUSrc	MemWrite	RegWrite
R-TYPE	000000	111	1	0	0	0	0	0	1
addi	000010	101	0	0	0	0	1	0	1
subtri	000011	110	0	0	0	0	1	0	1
andi	000100	000	0	0	0	0	1	0	1
ori	000101	001	0	0	0	0	1	0	1
lw	001000	101	0	0	1	1	1	0	1
sw	010000	101	X	0	0	X	1	1	0
lb	001001	101	0	0	1	1	1	0	1
sb	010001	101	X	0	0	X	1	1	0
slti	000111	100	0	0	0	0	1	0	1
beq	100011	110	X	1	0	X	0	0	0
jump	111000	XXX	X	0	0	X	X	0	0
jal	111001	XXX	X	0	0	X	X	0	0
move	100000	101	0	0	0	0	1	0	1

ALU CONTROL UNIT

Instruction	ALUop	Func.Code	ALUctr
AND	111	000100	000
OR	111	000101	001
XOR	111	XXXXXX	010
NOR	111	XXXXXX	011
LESS THAN	111	000111	100
ADD	111	000010	101
SUB	111	000011	110
andi	000	XXXXXX	000
ori	001	XXXXXX	001
slti	100	XXXXXX	100
addi, lb, sb, lw, sw	101	XXXXXX	101
subi, beq	110	XXXXXX	110

TESTBENCHS

Instruction Block

```
initial begin
   clk = 0;
   pc = 0;
   #50;
   pc = 4;
   #50;
   pc = 8;
   #50;
   pc = 12;
   #50;
   pc = 16;
   #50;
   pc = 20;
   #50;
   pc = 24;
   #50;
   pc = 28;
   #1000;

   $finish;
end

always #5 clk = ~clk;

initial $monitor ($time, " pc = %d, instruction = %b", pc, instruction);
```

Output

```
pc =           0, instruction = 00001000000001000000000000000101
pc =           4, instruction = 00001000000001000000000000001000
pc =           8, instruction = 10001100000001000000000000000110
pc =          12, instruction = 10000001000001000000000000000000
pc =          16, instruction = 11100100000000000000000000001010
pc =          20, instruction = 01000010000000100000000000000000
pc =          24, instruction = 00001010000100000000000000000100
pc =          28, instruction = 00001101000010000000000000000001
```

Memory Block

```
always #5 clk = ~clk;

initial begin
    clk = 0;
    write_data = 0;
    read_reg1 = 0;
    read_reg2 = 0;
    write_reg = 0;
    regWrite = 0;
end

initial begin
    @(posedge clk);

    write_data = 42;
    write_reg = 1;
    regWrite = 1;
    @(posedge clk);

    write_data = 17;
    write_reg = 2;
    @(posedge clk);

    regWrite = 0;
    @(negedge clk);

    read_reg1 = 1;
    read_reg2 = 2;
    @(negedge clk);

    $display("Read data 1: %d", read_data1);
    $display("Read data 2: %d", read_data2);

    $finish;
end
```

Output

```
# memWrite:1,memRead=0,address=     x,write_data=         x,read_data=         x
# memWrite:1,memRead=0,address=     0,write_data=        15,read_data=         x
# memWrite:1,memRead=0,address=     4,write_data=        92,read_data=         x
# memWrite:0,memRead=0,address=     4,write_data=        92,read_data=         x
# memWrite:0,memRead=0,address=     8,write_data=        66,read_data=         x
# memWrite:0,memRead=1,address=     0,write_data=        66,read_data=        15
# memWrite:0,memRead=1,address=     4,write_data=        66,read_data=        92
# memWrite:0,memRead=1,address=     8,write_data=        66,read_data=        92
# memWrite:0,memRead=1,address=     8,write_data=        66,read_data=         0
```

Register Block

```
initial begin
   clk <= 0;
   forever begin
      #10 clk <= ~clk;
   end
end

initial begin
       memwrite=1;
       memRead=0;
   #20 write_data = 32'd15;
       address = 18'd0;
   #20 write_data= 32'd92;
       address = 18'd4;
   #20 memwrite=0;
   #20 write_data = 32'd66;
       address = 18'd8;
   #20 memRead=1;
       address=18'd0;
   #20 address=18'd4;
   #30 address=18'd8;
end


initial begin
   $monitor("memwrite:%b,memRead=%b,address=%d,write_data=%d,read_data=%d",memWrite,memRead,address,write_data,read_data);
end
```

Output

```
Read data 1:          42
Read data 2:          17
```

MIPS PROCESSOR

```
    initial begin
       clock <= 0;
       prog_ctr = 32'b00000000000000000000000000000000;
       forever begin
          #10 clock <= ~clock;
       end
    end

  // dump the waveforms to a file
  initial begin
    $dumpfile("mips_test.vcd");
    $dumpvars(0, mips_testbench);
  end

always @(negedge clock) begin
   #60 prog_ctr <= updated_pc;
end

  // run the simulation for 1000 ns
initial begin
   #1000;
   $finish();
end
```

- I didn't output image to report because program doesn't work correctly every time.