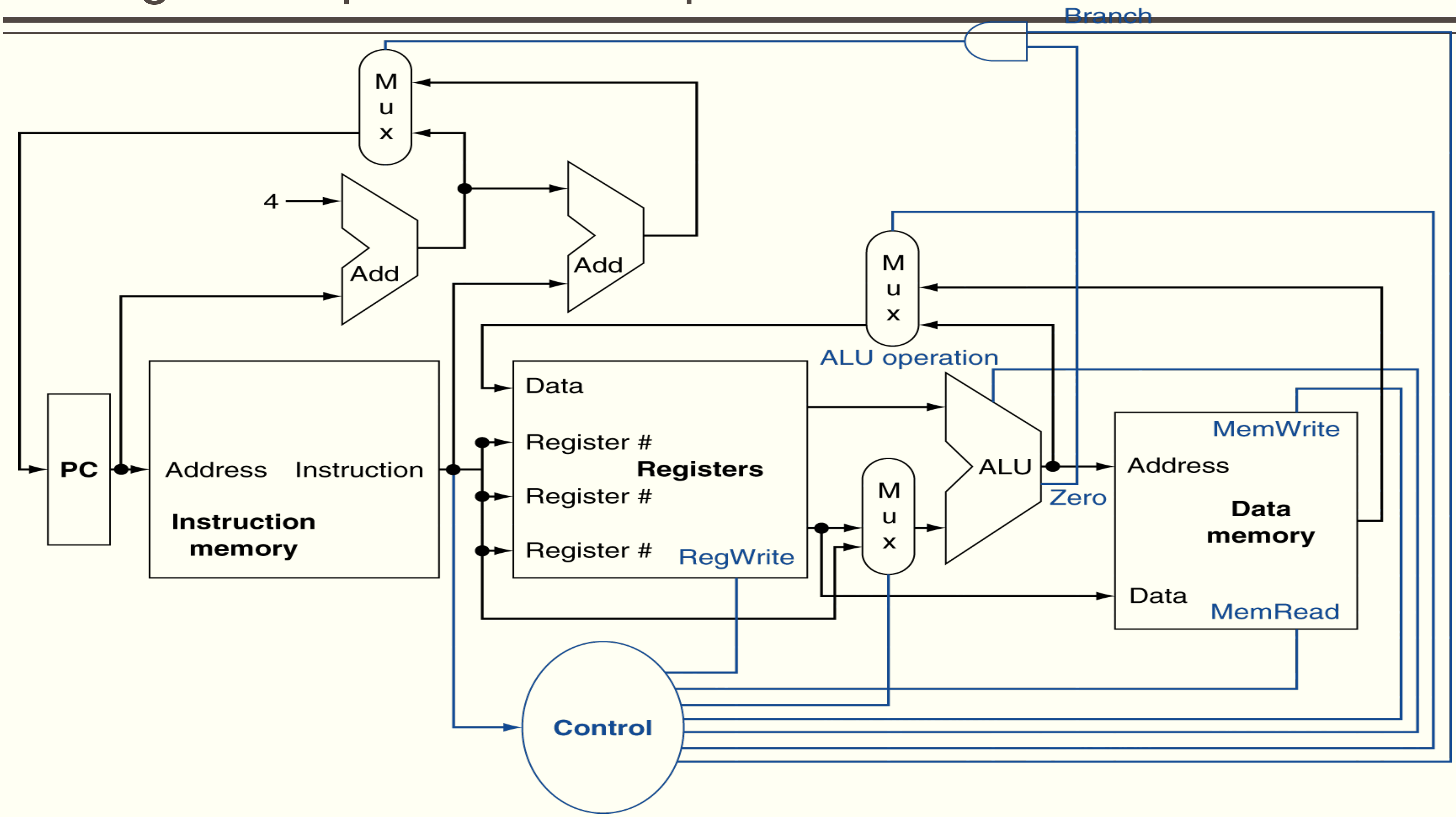
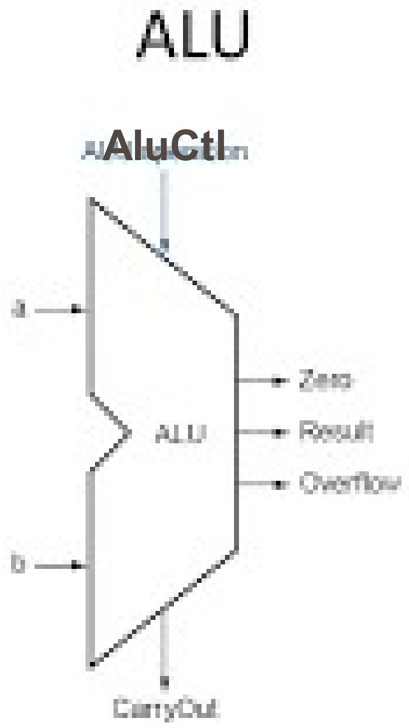
Verilog Description of MIPS processor



Multiplexer

```
module Mult2to1(In1,In2,Sel,Out);
  input [31:0] In1,In2;
  input Sel;
  output reg [31:0] Out;
  always @(In1,In2,Sel)
  case (Sel)
    0: Out <=In1;
    default: Out <=In2;
  endcase
endmodule
```

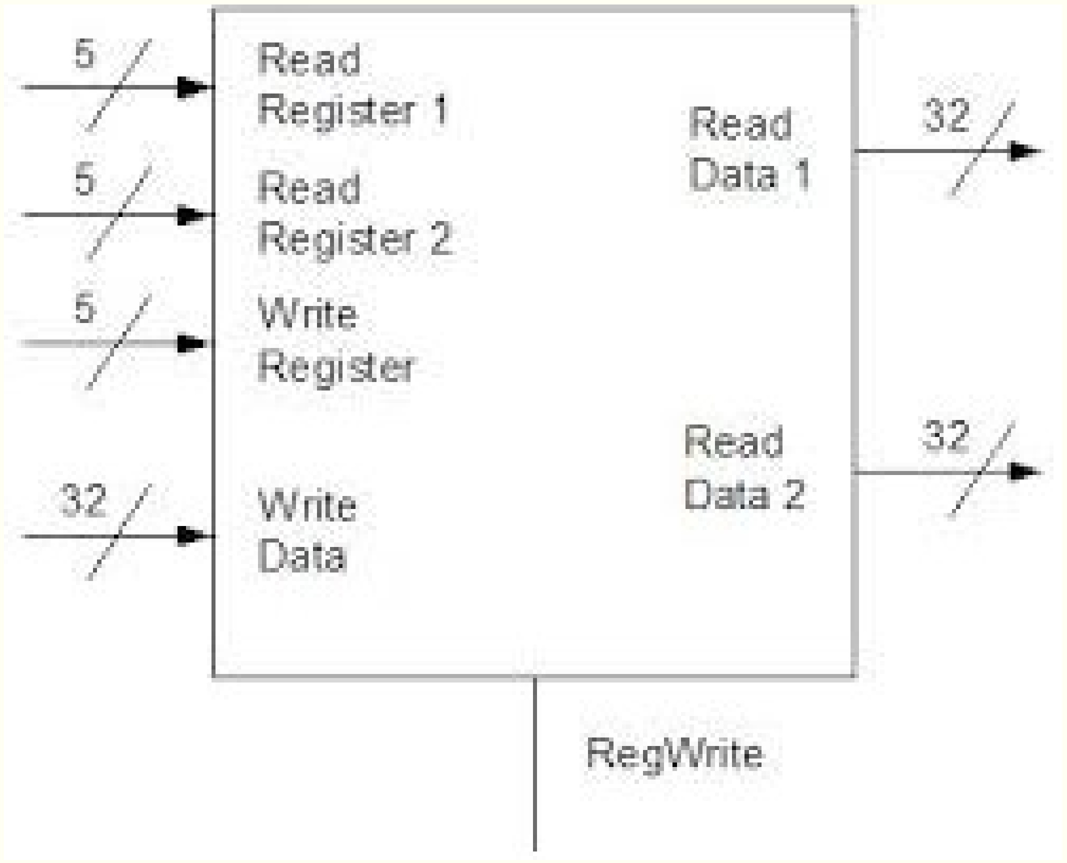
MIPS ALU



MIPS ALU

```
module MPISALU(ALUctl,A,B,ALUOut,Ze);
  input [3:0] ALUctl;
  input [31:0] A,B;
  output reg [32:0] ALUOut;
  output Ze;
  assign Ze = (ALUOut == 0);
  always@(ALUctl,A,B) begin
    case (ALUctl)
      0: ALUOut <= A & B;
      1: ALUOut <= A | B;
      2: ALUOut <= A + B;
      6: ALUOut <= A - B;
      7: ALUOut <= A < B ? 1 : 0;
      12 : ALUOut <= ~(A|B);
      default : ALUOut <= 0;
    endcase
  end
endmodule
```

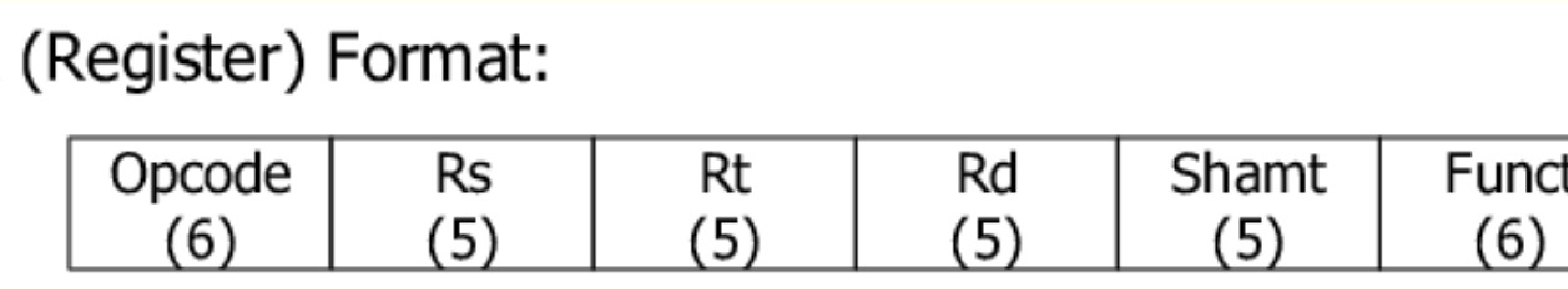
MIPS Register File Verilog Description



MIPS Register file

```
module registerfile(Read1,Read2,WriteReg,
    WriteData, RegWrite, Data1,Data2,Clock);

   input [4:0] Read1,Read2,WriteReg;
   input [31:0] WriteData;
   input RegWrite, Clock;

   output[31:0] Data1, Data2;
   reg[ 31:0] RF [31:0];


   assign Data1 = RF[Read1];
   assign Data2 = RF[Read2];

   always @(posedge Clock)
        if (RegWrite) RF[WriteReg] = WriteData;
endmodule
```

MIPS Instruction Format


(Register) Format:

Opcode (6)	Rs (5)	Rt (5)	Rd (5)	Shamt (5)	Func (6)

add $1, $2, $3

00000000010000110000100000100000

Instruction Fetch

```
module CPU (clock);
        input clock;
        reg[31:0] PC, Regs[0:31], IMemory[0:1023], DMemory[0:1023], IR,ALUOut;
        wire [4:0] rs, rt;
        wire [31:0] Ain, Bin;
        wire [5:0] op;
        assign rs = IR[25:21];
        assign rt = IR[20:16];
        assign op = IR[31:26];
        initial begin
                PC = 0;
        end
        always @ (posedge clock)
        begin
                IR <= IMemory[PC>>2];
                PC <= PC + 4;
```

ADD Instruction

```
always @ (posedge clock)
        begin
                IR <= IMemory[PC>>2];
                PC <= PC + 4;
                if (op == 6'b000000)
                case (IR[5:0])

                        32  : ALUOut <= Ain + Bin;

                endcase
        end


endmodule
```