```
module ControlOld(opcode, ALUSrc, ALUOp, RegDst, MemWrite, MemRead, Beq, Bne, Jump, MemToReg, RegWrite);

input [5:0] opcode; // 6-bit operation code
output reg ALUSrc, RegDst, MemWrite, MemRead, Beq, Bne, Jump, MemToReg, RegWrite; // Output control lines
output reg [1:0] ALUOp; // 2-bit intermediate output for controlling ALU


//sw; 01 = beq; 10 = arithmetic; 11 = Jump; 00 = lw
always @(*) begin
ALUSrc = 0;  RegDst = 0;  MemWrite = 0;   MemRead = 0;  Beq = 0;   Bne = 0;  Jump = 0;  MemToReg = 0;  RegWrite = 0;
case(opcode)

6'b 000101: begin//bne
      ALUOp = 2'b 01;
      Bne = 1;
    end
6'b 000100: //beq
    begin
      ALUOp = 2'b 01;
      Beq = 1;
    end
6'b 100011: begin//lw
      ALUOp = 2'b 00;
      ALUSrc = 1;   MemToReg = 1;   RegWrite = 1;   MemRead = 1;
    end
6'b 000010: begin//j
      ALUOp = 2'b 11;
      Jump = 1;
    end
6'b 000000: //r-type
    begin
      ALUOp = 2'b 10;
      RegDst = 1;   RegWrite = 1;
    end
6'b 101011: begin //sw
      ALUOp = 2'b 00;
      ALUSrc = 1;   MemWrite = 1;
    end
endcase
end
endmodule
```

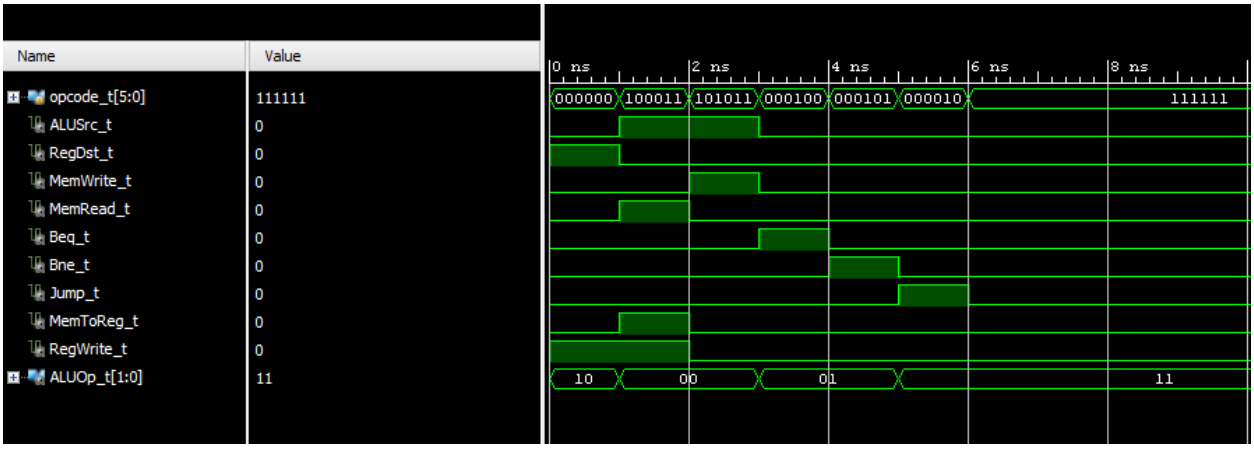
Test Bench

```
module OldControl_tb();
reg [5:0] opcode_t; // 6-bit operation code
wire ALUSrc_t, RegDst_t, MemWrite_t, MemRead_t, Beq_t, Bne_t, Jump_t, MemToReg_t, RegWrite_t; // Output control lines
wire [1:0] ALUOp_t; // 2-bit intermediate output for controlling ALU

ControlOld ctlr(opcode_t, ALUSrc_t, ALUOp_t, RegDst_t, MemWrite_t, MemRead_t, Beq_t,Bne_t, Jump_t, MemToReg_t, RegWrite_t);

initial
begin
// r-format
opcode_t <= 6'b 000000;
#1 $display ("opcode = %b, ALUSrc = %b, ALUOp = %b, RegDst = %b, MemWrite = %b, MemRead = %b, Beq = %b,Bne = %b, Jump = %b, MemToReg = %b, RegWrite = %b", opcode_t, ALUSrc_t, ALUOp_t, RegDst_t, MemWrite_t, MemRea

// lw
opcode_t <= 6'b 100011;
#1 $display ("opcode = %b, ALUSrc = %b, ALUOp = %b, RegDst = %b, MemWrite = %b, MemRead = %b, Beq = %b,Bne = %b, Jump = %b, MemToReg = %b, RegWrite = %b", opcode_t, ALUSrc_t, ALUOp_t, RegDst_t, MemWrite_t, MemRea

// sw
opcode_t <= 6'b 101011;
#1 $display ("opcode = %b, ALUSrc = %b, ALUOp = %b, RegDst = %b, MemWrite = %b, MemRead = %b, Beq = %b,Bne = %b, Jump = %b, MemToReg = %b, RegWrite = %b", opcode_t, ALUSrc_t, ALUOp_t, RegDst_t, MemWrite_t, MemRea

// beq
opcode_t <= 6'b 000100;
#1 $display ("opcode = %b, ALUSrc = %b, ALUOp = %b, RegDst = %b, MemWrite = %b, MemRead = %b, Beq = %b,Bne = %b, Jump = %b, MemToReg = %b, RegWrite = %b", opcode_t, ALUSrc_t, ALUOp_t, RegDst_t, MemWrite_t, MemRea

// bne
opcode_t <= 6'b 000101;
#1 $display ("opcode = %b, ALUSrc = %b, ALUOp = %b, RegDst = %b, MemWrite = %b, MemRead = %b, Beq = %b,Bne = %b, Jump = %b, MemToReg = %b, RegWrite = %b", opcode_t, ALUSrc_t, ALUOp_t, RegDst_t, MemWrite_t, MemRea

// jump
opcode_t <= 6'b 000010;
#1 $display ("opcode = %b, ALUSrc = %b, ALUOp = %b, RegDst = %b, MemWrite = %b, MemRead = %b, Beq = %b,Bne = %b, Jump = %b, MemToReg = %b, RegWrite = %b", opcode_t, ALUSrc_t, ALUOp_t, RegDst_t, MemWrite_t, MemRea

 //invalad input; should produce all zeros
opcode_t <= 6'b 111111;
#1 $display ("opcode = %b, ALUSrc = %b, ALUOp = %b, RegDst = %b, MemWrite = %b, MemRead = %b, Beq = %b,Bne = %b, Jump = %b, MemToReg = %b, RegWrite = %b", opcode_t, ALUSrc_t, ALUOp_t, RegDst_t, MemWrite_t, MemRea

end

endmodule
```

Output

```
# run 1000ns
opcode = 000000, ALUSrc = 0, ALUOp = 10, RegDst = 1, MemWrite = 0, MemRead = 0, Beq = 0,Bne = 0, Jump = 0, MemToReg = 0, RegWrite = 1
opcode = 100011, ALUSrc = 1, ALUOp = 00, RegDst = 0, MemWrite = 0, MemRead = 1, Beq = 0,Bne = 0, Jump = 0, MemToReg = 1, RegWrite = 1
opcode = 101011, ALUSrc = 1, ALUOp = 00, RegDst = 0, MemWrite = 1, MemRead = 0, Beq = 0,Bne = 0, Jump = 0, MemToReg = 0, RegWrite = 0
opcode = 000100, ALUSrc = 0, ALUOp = 01, RegDst = 0, MemWrite = 0, MemRead = 0, Beq = 1,Bne = 0, Jump = 0, MemToReg = 0, RegWrite = 0
opcode = 000101, ALUSrc = 0, ALUOp = 01, RegDst = 0, MemWrite = 0, MemRead = 0, Beq = 0,Bne = 1, Jump = 0, MemToReg = 0, RegWrite = 0
opcode = 000010, ALUSrc = 0, ALUOp = 11, RegDst = 0, MemWrite = 0, MemRead = 0, Beq = 0,Bne = 0, Jump = 1, MemToReg = 0, RegWrite = 0
opcode = 111111, ALUSrc = 0, ALUOp = 11, RegDst = 0, MemWrite = 0, MemRead = 0, Beq = 0,Bne = 0, Jump = 0, MemToReg = 0, RegWrite = 0
```

Waveform



```
`timescale 1ns / 1ps

module ALUOpToALUControl(ALUOp, Funct, ALUControl);
input [1:0] ALUOp;       // 2-bit intermediate output for controlling ALU
input [5:0] Funct;       // 6-bit function code
output [2:0] ALUControl; // 3-bit output for controlling ALU based on ALUOp and function code

reg [2:0] ALUControl;
always@(ALUOp, Funct) begin
    ALUControl[0]=(ALUOp[1]&(Funct[0]|Funct[3]));
    ALUControl[1]=(~ALUOp[1]|~Funct[2]);
    ALUControl[2]=(ALUOp[0]|(ALUOp[1]&Funct[1]));
end

endmodule
```

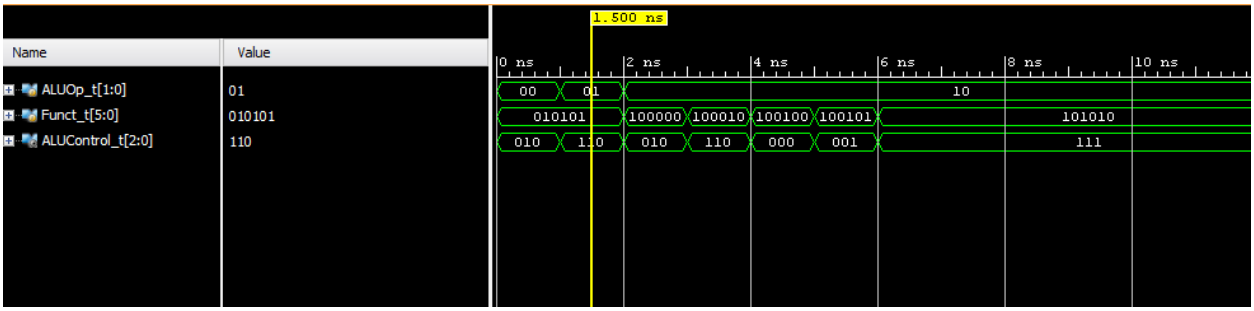
Test Bench

```
module AluOpTestBench();
reg [1:0] ALUOp_t;
reg [5:0] Funct_t;
wire [2:0] ALUControl_t;

ALUOpToALUControl my_op(ALUOp_t, Funct_t, ALUControl_t);

 initial
 begin
    ALUOp_t <= 2'b 00; Funct_t <= 6'b 010101;
    #1 $display ( "ALUOp = %b, Funct = %b, ALUControl = %b", ALUOp_t, Funct_t, ALUControl_t);

        ALUOp_t <= 2'b 01; Funct_t <= 6'b 010101;
    #1 $display ( "ALUOp = %b, Funct = %b, ALUControl = %b", ALUOp_t, Funct_t, ALUControl_t);

    ALUOp_t <= 2'b 10; Funct_t <= 6'b 100000;
    #1 $display ( "ALUOp = %b, Funct = %b, ALUControl = %b", ALUOp_t, Funct_t, ALUControl_t);

        ALUOp_t <= 2'b 10; Funct_t <= 6'b 100010;
    #1 $display ( "ALUOp = %b, Funct = %b, ALUControl = %b", ALUOp_t, Funct_t, ALUControl_t);

        ALUOp_t <= 2'b 10; Funct_t <= 6'b 100100;
    #1 $display ( "ALUOp = %b, Funct = %b, ALUControl = %b", ALUOp_t, Funct_t, ALUControl_t);

        ALUOp_t <= 2'b 10; Funct_t <= 6'b 100101;
    #1 $display ( "ALUOp = %b, Funct = %b, ALUControl = %b", ALUOp_t, Funct_t, ALUControl_t);

        ALUOp_t <= 2'b 10; Funct_t <= 6'b 101010;
    #1 $display ( "ALUOp = %b, Funct = %b, ALUControl = %b", ALUOp_t, Funct_t, ALUControl_t);
 end
endmodule
```

Output

```
ALUOp = 00, Funct = 010101, ALUControl = 010
ALUOp = 01, Funct = 010101, ALUControl = 110
ALUOp = 10, Funct = 100000, ALUControl = 010
ALUOp = 10, Funct = 100010, ALUControl = 110
ALUOp = 10, Funct = 100100, ALUControl = 000
ALUOp = 10, Funct = 100101, ALUControl = 001
ALUOp = 10, Funct = 101010, ALUControl = 111
```

Waveform




```
module RegisterFile(ReadRegister1, ReadRegister2, WriteRegister,WriteData, RegWrite, Clk, ReadData1, ReadData2);
input [4:0] ReadRegister1, ReadRegister2; // Two registers to be read
input [4:0] WriteRegister; // Register address to write into
input [31:0] WriteData; // Data to be written into WriteRegister
input RegWrite; // RegWrite control signal. Data is written only whenthis signal is enabled
output [31:0] ReadData1, ReadData2;
input Clk;
reg [31:0] ReadData1, ReadData2;
reg [31:0] iamReg[0:31];

initial begin
        iamReg[8]=32'h FFFFFFFF;
        iamReg[2]=32'h 00000000;
        iamReg[16]=32'h F0F0F0F0;
        iamReg[9]=32'h 0F0F0F0F;
end

always @(Clk) begin
    if(RegWrite) begin
       iamReg[WriteRegister] = WriteData ;
    end
    assign ReadData1 = iamReg[ReadRegister1] ;
    assign ReadData2 = iamReg[ReadRegister2] ;
end



endmodule
```

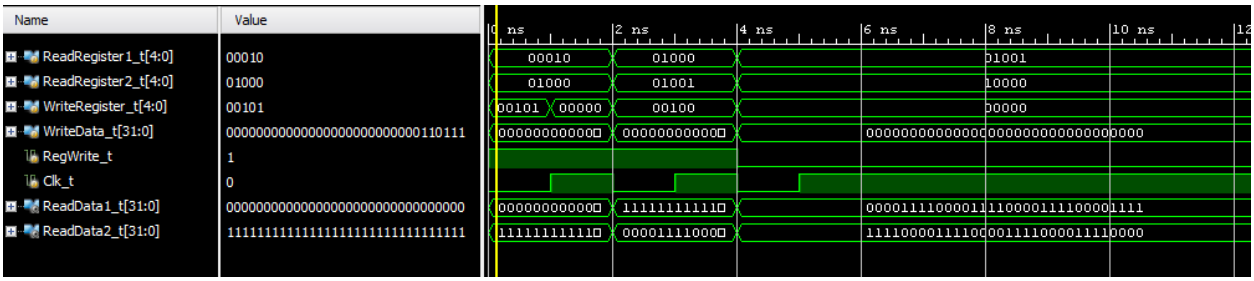
Test Bench

```
module problem_c_tb();
reg [4:0] ReadRegister1_t, ReadRegister2_t;
reg [4:0] WriteRegister_t;                  // Register address to write into
reg [31:0] WriteData_t;                        // Data to be written into WriteRegister
reg RegWrite_t;                                   // RegWrite control signal. Data is written only when this signal is enabled
reg Clk_t;
wire [31:0] ReadData1_t, ReadData2_t;

RegisterFile register(ReadRegister1_t, ReadRegister2_t, WriteRegister_t, WriteData_t, RegWrite_t, Clk_t, ReadData1_t, ReadData2_t);

initial begin
Clk_t<=1'b0; ReadRegister1_t<=5'd2; ReadRegister2_t<=5'd8; WriteRegister_t<=5'd5; WriteData_t<=32'd55; RegWrite_t<=1;
#1 $display("ReadRegister1 = %d, ReadRegister2 = %d, WriteRegister = %d, WriteData = %d, RegWrite =%d, Clk_t = %b, ReadData1 = %d, ReadD

Clk_t<=1'b1; ReadRegister1_t<=5'd2; ReadRegister2_t<=5'd8; WriteRegister_t<=5'd0; WriteData_t<=32'd55; RegWrite_t<=1;
#1 $display("ReadRegister1 = %d, ReadRegister2 = %d, WriteRegister = %d, WriteData = %d, RegWrite =%d, Clk_t = %b, ReadData1 = %d, ReadD

Clk_t<=1'b0; ReadRegister1_t<=5'd8; ReadRegister2_t<=5'd9; WriteRegister_t<=5'd4; WriteData_t<=32'd6444; RegWrite_t<=1;
#1 $display("ReadRegister1 = %d, ReadRegister2 = %d, WriteRegister = %d, WriteData = %d, RegWrite =%d, Clk_t = %b, ReadData1 = %d, ReadD

Clk_t<=1'b1; ReadRegister1_t<=5'd8; ReadRegister2_t<=5'd9; WriteRegister_t<=5'd4; WriteData_t<=32'd6444; RegWrite_t<=1;
#1 $display("ReadRegister1 = %d, ReadRegister2 = %d, WriteRegister = %d, WriteData = %d, RegWrite =%d, Clk_t = %b, ReadData1 = %d, ReadD

Clk_t<=1'b0; ReadRegister1_t<=5'd9; ReadRegister2_t<=5'd16; WriteRegister_t<=5'd0; WriteData_t<=32'd0; RegWrite_t<=0;
#1 $display("ReadRegister1 = %d, ReadRegister2 = %d, WriteRegister = %d, WriteData = %d, RegWrite =%d, Clk_t = %b, ReadData1 = %d, ReadD

Clk_t<=1'b1; ReadRegister1_t<=5'd9; ReadRegister2_t<=5'd16; WriteRegister_t<=5'd0; WriteData_t<=32'd0; RegWrite_t<=0;
#1 $display("ReadRegister1 = %d, ReadRegister2 = %d, WriteRegister = %d, WriteData = %d, RegWrite =%d, Clk_t = %b, ReadData1 = %d, ReadD
```

Output

```
ReadRegister1 =  2, ReadRegister2 =  8, WriteRegister =  5, WriteData =         55, RegWrite =1, Clk_t = 0, ReadData1 =          0, ReadData2 = 4294967295
ReadRegister1 =  2, ReadRegister2 =  8, WriteRegister =  0, WriteData =         55, RegWrite =1, Clk_t = 1, ReadData1 =          0, ReadData2 = 4294967295
ReadRegister1 =  8, ReadRegister2 =  9, WriteRegister =  4, WriteData =       6444, RegWrite =1, Clk_t = 0, ReadData1 = 4294967295, ReadData2 =  252645135
ReadRegister1 =  8, ReadRegister2 =  9, WriteRegister =  4, WriteData =       6444, RegWrite =1, Clk_t = 1, ReadData1 = 4294967295, ReadData2 =  252645135
ReadRegister1 =  9, ReadRegister2 = 16, WriteRegister =  0, WriteData =          0, RegWrite =0, Clk_t = 0, ReadData1 =  252645135, ReadData2 = 4042322160
ReadRegister1 =  9, ReadRegister2 = 16, WriteRegister =  0, WriteData =          0, RegWrite =0, Clk_t = 1, ReadData1 =  252645135, ReadData2 = 4042322160
```

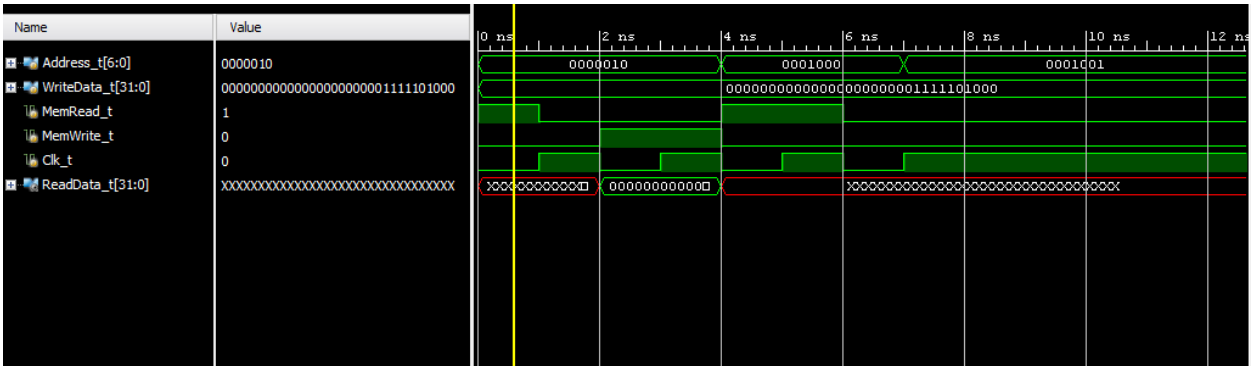
Waveform



```
module DataMemory(Address, WriteData, MemRead, MemWrite, Clk,ReadData);
input [6:0] Address; // 7-bit address to memory.
input [31:0] WriteData; // Data to be written into WriteRegister
input MemRead; // Data in memory location Address is read if thiscontrol is set
input MemWrite; // WriteData is written in Address during positiveclock edge if this control is set
output [31:0] ReadData; // Value read from memory location Address
reg [31:0] ReadData;
input Clk;
reg [255:0] memFile[31:0];


initial @(Clk) begin
   memFile[0] <= 10;
end


always @(*) begin
    if (MemRead) begin
        assign ReadData = memFile[Address];
    end

    if (MemWrite) begin
        memFile[Address] = WriteData;
    end
end

endmodule
```

```
module DataMemory_tb();
reg [6:0] Address_t;         // 7-bit address to memory.
reg [31:0] WriteData_t;  // Data to be written into WriteRegister
reg MemRead_t;               // Data in memory location Address is read if this control is set
reg MemWrite_t;          // WriteData is written in Address during positive clock edge if this control is set
reg Clk_t;                   // Clock input
wire [31:0] ReadData_t;  // Value read from memory location Address

DataMemory dmem(Address_t, WriteData_t, MemRead_t, MemWrite_t, Clk_t, ReadData_t);

initial
    begin
     Address_t <= 7'd 2; WriteData_t <= 32'd 1000; MemRead_t <= 1; MemWrite_t <= 0; Clk_t <= 0;
     #1 $display ("Address = %d, WriteData =%d, MemRead =%d, MemWrite = %d, Clk = %d, ReadData = %d", Address_t, WriteData_t, MemRead_t,

     Address_t <= 7'd 2; WriteData_t <= 32'd 1000; MemRead_t <= 0; MemWrite_t <= 0; Clk_t <= 1;
     #1 $display ("Address = %d, WriteData =%d, MemRead =%d, MemWrite = %d, Clk = %d, ReadData = %d", Address_t, WriteData_t, MemRead_t,

     Address_t <= 7'd 2; WriteData_t <= 32'd 1000; MemRead_t <= 0; MemWrite_t <= 1; Clk_t <= 0;
     #1 $display ("Address = %d, WriteData =%d, MemRead =%d, MemWrite = %d, Clk = %d, ReadData = %d", Address_t, WriteData_t, MemRead_t,

     Address_t <= 7'd 2; WriteData_t <= 32'd 1000; MemRead_t <= 0; MemWrite_t <= 1; Clk_t <= 1;
     #1 $display ("Address = %d, WriteData =%d, MemRead =%d, MemWrite = %d, Clk = %d, ReadData = %d", Address_t, WriteData_t, MemRead_t,

     Address_t <= 7'd 8; WriteData_t <= 32'd 1000; MemRead_t <= 1; MemWrite_t <= 0; Clk_t <= 0;
     #1 $display ("Address = %d, WriteData =%d, MemRead =%d, MemWrite = %d, Clk = %d, ReadData = %d", Address_t, WriteData_t, MemRead_t,

    Address_t <= 7'd 8; WriteData_t <= 32'd 1000; MemRead_t <= 1; MemWrite_t <= 0; Clk_t <= 1;
     #1 $display ("Address = %d, WriteData =%d, MemRead =%d, MemWrite = %d, Clk = %d, ReadData = %d", Address_t, WriteData_t, MemRead_t,

    Address_t <= 7'd 8; WriteData_t <= 32'd 1000; MemRead_t <= 0; MemWrite_t <= 0; Clk_t <= 0;
     #1 $display ("Address = %d, WriteData =%d, MemRead =%d, MemWrite = %d, Clk = %d, ReadData = %d", Address_t, WriteData_t, MemRead_t,

     Address_t <= 7'd 9; WriteData_t <= 32'd 1000; MemRead_t <= 0; MemWrite_t <= 0; Clk_t <= 1;
     #1 $display ("Address = %d, WriteData =%d, MemRead =%d, MemWrite = %d, Clk = %d, ReadData = %d", Address_t, WriteData_t, MemRead_t,

    end

endmodule
```

```
Address =   2, WriteData =      1000, MemRead =1, MemWrite = 0, Clk = 0, ReadData =          x
Address =   2, WriteData =      1000, MemRead =0, MemWrite = 0, Clk = 1, ReadData =          x
Address =   2, WriteData =      1000, MemRead =0, MemWrite = 1, Clk = 0, ReadData =       1000
Address =   2, WriteData =      1000, MemRead =0, MemWrite = 1, Clk = 1, ReadData =       1000
Address =   8, WriteData =      1000, MemRead =1, MemWrite = 0, Clk = 0, ReadData =          x
Address =   8, WriteData =      1000, MemRead =1, MemWrite = 0, Clk = 1, ReadData =          x
Address =   8, WriteData =      1000, MemRead =0, MemWrite = 0, Clk = 0, ReadData =          x
Address =   9, WriteData =      1000, MemRead =0, MemWrite = 0, Clk = 1, ReadData =          x
```




```
module Control(opcode, funct, ALUSrc, RegDst, MemWrite, MemRead, Beq, Bne, Jump, MemToReg, RegWrite, ALUControl);
    input [5:0] opcode; // 6-bit operation code
    input [5:0] funct;  // 6-bit function code from the instruction
                        // least significant 6 bits of an instruction
    output ALUSrc, RegDst, MemWrite, MemRead, Beq, Bne, Jump, MemToReg, RegWrite; // Output control lines
    output [2:0] ALUControl; // 3-bit control for the ALU that specifies the operation

    wire [1:0] ALUOp;

    ControlOld Cntrl(opcode, ALUSrc, ALUOp, RegDst, MemWrite, MemRead, Beq, Bne, Jump, MemToReg, RegWrite);
    ALUOpToALUControl ALUOp2Ctrl(ALUOp, funct, ALUControl);

endmodule
```

```
module Control_tb();
reg [5:0] opcode_t;      // 6-bit operation code
reg [5:0] funct_t;           // 6-bit function code from the instruction
                             // least significant 6 bits of an instruction
wire ALUSrc_t, RegDst_t, MemWrite_t, MemRead_t, Beq_t, Bne_t, Jump_t, MemToReg_t, RegWrite_t; // Output control lines
wire [2:0] ALUControl_t; // 3-bit control for the ALU that specifies the operation
Control Control1(opcode_t, funct_t, ALUSrc_t, RegDst_t, MemWrite_t, MemRead_t, Beq_t, Bne_t, Jump_t, MemToReg_t, RegWrite_t, ALUControl_t);

 initial
 begin
    opcode_t <= 6'b 000000; funct_t <= 6'b 010101;
    #1 $display ("opcode = %b, funct=%b, ALUSrc=%b, RegDst=%b, MemWrite=%b, MemRead=%b, Beq=%b, Bne=%b, Jump=%b, MemToReg=%b, RegWrite=%b, A

    opcode_t <= 6'b 100011; funct_t <= 6'b 010101;
    #1 $display ("opcode = %b, funct=%b, ALUSrc=%b, RegDst=%b, MemWrite=%b, MemRead=%b, Beq=%b, Bne=%b, Jump=%b, MemToReg=%b, RegWrite=%b, A


    opcode_t <= 6'b 101011; funct_t <= 6'b 100000;
    #1 $display ("opcode = %b, funct=%b, ALUSrc=%b, RegDst=%b, MemWrite=%b, MemRead=%b, Beq=%b, Bne=%b, Jump=%b, MemToReg=%b, RegWrite=%b, A


    opcode_t <= 6'b 000100; funct_t <= 6'b 100010;
    #1 $display ("opcode = %b, funct=%b, ALUSrc=%b, RegDst=%b, MemWrite=%b, MemRead=%b, Beq=%b, Bne=%b, Jump=%b, MemToReg=%b, RegWrite=%b, A


    opcode_t <= 6'b 000101; funct_t <= 6'b 100100;
    #1 $display ("opcode = %b, funct=%b, ALUSrc=%b, RegDst=%b, MemWrite=%b, MemRead=%b, Beq=%b, Bne=%b, Jump=%b, MemToReg=%b, RegWrite=%b, A


    opcode_t <= 6'b 000010; funct_t <= 6'b 0100101;
    #1 $display ("opcode = %b, funct=%b, ALUSrc=%b, RegDst=%b, MemWrite=%b, MemRead=%b, Beq=%b, Bne=%b, Jump=%b, MemToReg=%b, RegWrite=%b, A


    opcode_t <= 6'b 000000; funct_t <= 6'b 101010;
    #1 $display ("opcode = %b, funct=%b, ALUSrc=%b, RegDst=%b, MemWrite=%b, MemRead=%b, Beq=%b, Bne=%b, Jump=%b, MemToReg=%b, RegWrite=%b, A
```

```
opcode = 000000, funct=010101, ALUSrc=0, RegDst=1, MemWrite=0, MemRead=0, Beq=0, Bne=0, Jump=0, MemToReg=0, RegWrite=1, ALUControl=001
opcode = 100011, funct=010101, ALUSrc=1, RegDst=0, MemWrite=0, MemRead=1, Beq=0, Bne=0, Jump=0, MemToReg=1, RegWrite=1, ALUControl=010
opcode = 101011, funct=100000, ALUSrc=1, RegDst=0, MemWrite=1, MemRead=0, Beq=0, Bne=0, Jump=0, MemToReg=0, RegWrite=0, ALUControl=010
opcode = 000100, funct=100010, ALUSrc=0, RegDst=0, MemWrite=0, MemRead=0, Beq=1, Bne=0, Jump=0, MemToReg=0, RegWrite=0, ALUControl=110
opcode = 000101, funct=100100, ALUSrc=0, RegDst=0, MemWrite=0, MemRead=0, Beq=0, Bne=1, Jump=0, MemToReg=0, RegWrite=0, ALUControl=110
opcode = 000010, funct=100101, ALUSrc=0, RegDst=0, MemWrite=0, MemRead=0, Beq=0, Bne=0, Jump=1, MemToReg=0, RegWrite=0, ALUControl=101
opcode = 000000, funct=101010, ALUSrc=0, RegDst=1, MemWrite=0, MemRead=0, Beq=0, Bne=0, Jump=0, MemToReg=0, RegWrite=1, ALUControl=111
```