E/19/105, E/19/106

GROUP - 19

CO224 – LAB5

PART-5

We have extended our CPU with sll, srl, sra, ror, bne instructions as extra instructions.

1. **sll & srl instructions**

Opcodes

00001000 : sll

00001001 : srl

ALU module

For these two instructions a separate module named as LOGICALSHIFT is implemented inside ALU. According to a given control signal (SHIFTSIDE) the ALU performs logical right shift or logical left shift by the given amount (SHIFTAMOUNT).

//Logical Shift module

module LOGICALSHIFT (OPERAND, SHIFTAMOUNT, SHIFTSIDE, RESULT);

    //Port declarations

    input [7:0] OPERAND, SHIFTAMOUNT;

    input SHIFTSIDE;                    //If 0 leftshift, if 1 rightshift

    output reg [7:0] RESULT;

    reg [7:0] tempOperand;              //temporary register

    reg [6:0] temp;                     //temporary register

    integer i;                          //for looping

    always @(SHIFTAMOUNT, OPERAND) begin

        #1

        tempOperand = OPERAND;

        for (i = 1; i<=SHIFTAMOUNT; i++ ) begin

            if (SHIFTSIDE == 0) begin       //Leftshift operation

                temp = tempOperand[6:0];

                RESULT[7:1] = temp;

                RESULT[0] = 1'b0;

                tempOperand = RESULT;

            end else begin                  //Rightshift operation

                temp = tempOperand[7:1];

                RESULT[6:0] = temp;

                RESULT[7] = 1'b0;

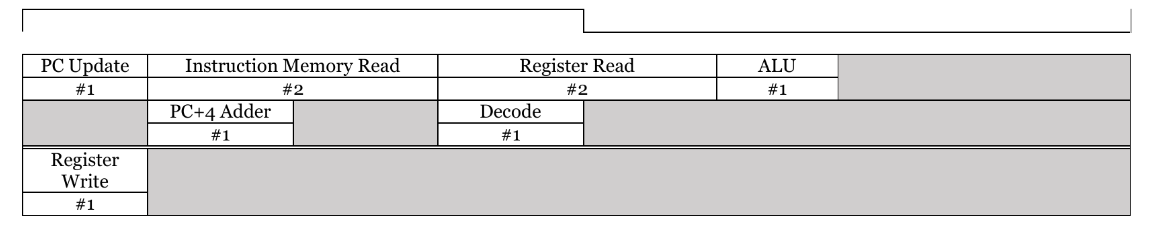
                tempOperand = RESULT;

            end

        end

    end

endmodule

Timing Diagram

Changes in Datapath & Control

For this an additional control signal (isRIGHTSHIFT) is added. The value of this control signal is assigned for sll instruction to zero and for srl instruction to one.

1. **sra instruction**

Opcode

00001010

ALU module

For this instruction a separate module named as ARITHMETICSHIFT is implemented inside ALU. According to the given amount (SHIFTAMOUNT) the ALU performs arithmetic right shift.

//Arithmetic Shift module

module ARITHMETICSHIFT(OPERAND, SHIFTAMOUNT, RESULT);

    //Port declarations

    input [7:0] OPERAND, SHIFTAMOUNT;

    output reg [7:0] RESULT;

    reg [7:0] tempOperand;          //temporary register

    reg [6:0] temp;                 //temporary register

    integer j;                      //for looping

    always @(OPERAND, SHIFTAMOUNT) begin

        #1

        tempOperand = OPERAND;

        for (j = 1; j<=SHIFTAMOUNT ; j++) begin

            if (tempOperand[7]==1'b0) begin     //If MSB is 0 add leading zeros

                temp = tempOperand[7:1];

                RESULT[6:0] = temp;

                RESULT[7] = 1'b0;

                tempOperand = RESULT;

            end else begin                      //If MSB is 1 add leading ones

                temp = tempOperand[7:1];

                RESULT[6:0] = temp;

                RESULT[7] = 1'b1;

                tempOperand = RESULT;

            end

        end

    end

endmodule

Timing Diagram

A close-up of a document

Description automatically generated with low confidence

Changes in Datapath & Control

No additional control signals for this module.

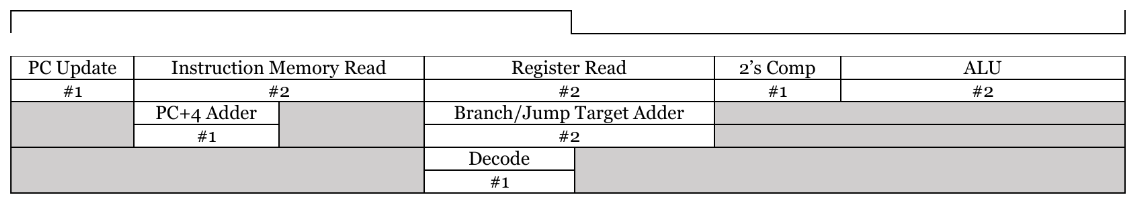
1. **bne instruction**

Opcode

00001011

ALU module

No changes in ALU module. It uses only the ZERO indicator (output of ALU module) for comparing.

Timing Diagram

Changes in Datapath & Control

An additional control signal (isBNE) is added to the CPU.

This control signal is then passed to mux\_pc mux to select whether the instruction need to be jumped or not.

//MUX module for jump and branch instructions

module mux\_pc(DATA1,DATA2,JUMPSELECT,BEQSELECT,BNESELECT,ZEROSELECT,OUTPUT);

    //Declaration of ports

    input [31:0] DATA1,DATA2;

    input JUMPSELECT, BEQSELECT, BNESELECT, ZEROSELECT;

    output reg [31:0] OUTPUT;

    //direct on of the data to the output according to the select value

    always@(\*) begin

        if (JUMPSELECT) begin

            OUTPUT = DATA1;                         //If the instruction is j, assign the NextPC value to the target address

        end else if (BEQSELECT && ZEROSELECT) begin //If the instruction is beq and equal condition is satisfied,

            OUTPUT = DATA1;                         // assign the NextPC value to the target address

        end else if (BNESELECT && !ZEROSELECT) begin    //If the instruction is bne and notequal condition is satisfied

            OUTPUT = DATA1;                             // assign the NextPC value to the target address

        end else begin

            OUTPUT = DATA2;                         //Else nextPC value is increased by 4 from currentPC value

        end

    end

endmodule

1. **ror instruction**

Opcode

00001100

ALU module

An additional module named as ROTATE has been implemented inside the ALU module.

According to the given amount (ROTATEAMOUNT) the input number is rotated right side.

//Rotate right module

module ROTATE(OPERAND, ROTATEAMOUNT, RESULT);

    input [7:0] OPERAND, ROTATEAMOUNT;

    output reg [7:0] RESULT;

    reg tempLSB;                    //temporary register

    reg [6:0] temp;                 //temporary register

    reg [7:0] tempOperand;          //temporary register

    integer k;                      //for looping

    always @(OPERAND, ROTATEAMOUNT) begin

        #1

        tempOperand = OPERAND;

        tempLSB = OPERAND[0];       //Save the LSB

        //ROtate through a loop

        for ( k=1 ; k<=ROTATEAMOUNT ; k++) begin

            temp = tempOperand[7:1];

            RESULT[6:0] = temp;

            RESULT[7] = tempLSB;

            tempOperand = RESULT;

            tempLSB = RESULT[0];

        end

    end

endmodule

A close-up of a document

Description automatically generated with low confidence Timing Diagram

Changes in Datapath & Control

No additional control signals for this module.