CS224
Lab No.:4
Section No.: 1
Name: Atakan Akar
Bilkent ID: 22203140
Date: 13.11.2024

x8'h00	32'h20020005	addi $v0, $zero, 5
8'h04	32'h2003000c	addi $v1, $zero, 12
8'h08	32'h2067fff7	addi $a3, $v1, -9
8'h0c	32'h00e22025	or $a0, $a3, $v0
8'h10	32'h00642824	and $a1, $v1, $a0
8'h14	32'h00a42820	add $a1, $a1, $a0
8'h18	32'h10a7000a	bne $a1, $a3, 0x0a

CS224
Lab No.:4
Section No.: 1
Name: Atakan Akar
Bilkent ID: 22203140
Date: 13.11.2024

8'h1c	32'h0064202a	slt $a0, $v1, $a0	**B)**

8'h20	32'h10800001	beq $a0, $zero, 0x01

8'h24	32'h20050000	addi $a1, $zero, 0
8'h28	32'h00e2202a	slt $a0, $a3, $v0
8'h2c	32'h00853820	add $a3, $a0, $a1
8'h30	32'h00e23822	sub $a3, $a3, $v0
8'h34	32'hac670044;	sw $a3, 68($v1)
8'h38	32'h8c020050	lw $v0, 80($zero)
8'h3c	32'h08000011	j 0x11
8'h40	32'h20020001;	addi $v0, $zero, 1

CS224
Lab No.:4
Section No.: 1
Name: Atakan Akar
Bilkent ID: 22203140
Date: 13.11.2024

8'h44	32'hac020054	sw $v0, 84($zero)
8'h48	32'h08000012	j 0x12

C)
BCON
IM[PC]
if (RF[rt] == RF[rs] + 4) and (RF[rt] % 4 == 0)
PC ← PC + 4 + (Imm *4)
Else
PC ← PC + 4

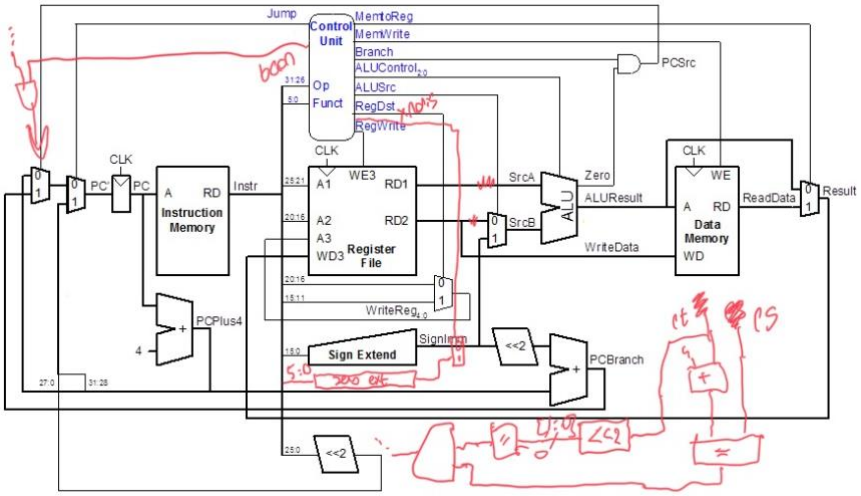
XNORI
IM[PC]

RF[rt] ← ~(RF[rs] ⊕ ZeroExtend(Imm))

PC ← PC + 4

D)

CS224
Lab No.:4
Section No.: 1
Name: Atakan Akar
Bilkent ID: 22203140
Date: 13.11.2024



E)

Instruction	Opcode	RegWrite	RegDst	ALUSrc	Branch	MemWrite	MemToReg	ALUOp	Jump	bcons	xnoris
R-type	000000	1	1	0	0	0	0	10	0	0	0
lw	100011	1	0	1	0	0	1	00	0	0	0
sw	101011	0	X	1	0	1	X	00	0	0	0
beq	000100	0	X	0	1	0	X	01	0	0	0
addi	001000	1	0	1	0	0	0	00	0	0	0
j	000010	0	X	X	X	0	X	XX	1	0	0

bcon	000011	0	X	0	0	0	X	XX	0	1	0
xnori	000001	1	0	1	0	0	0	11	0	0	1

CS224
Lab No.:4
Section No.: 1
Name: Atakan Akar
Bilkent ID: 22203140
Date: 13.11.2024

ALUOp	Funct	ALUControl
00	X	010 (add)
01	X	110 (subtract)
10	100000 (add)	010 (add)
10	100010 (sub)	110 (subtract)
10	100100 (and)	000 (and)
10	100101 (or)	001 (or)
10	101010 (slt)	111 (set less than)

11	**X**	**101(xnori)**

F)
Bcon)

```
data
final: .word 0
final1: .word 0
final2: .word 0
final3: .word 0

   .text
main:
   addi $t0, $zero, 8
   addi $t1, $zero, 12
   addi $t2, $zero, 16

   move $a0, $t0
   move $a1, $t1
   move $a2, $t2

   la $a3, final
```

CS224
Lab No.:4
Section No.: 1
Name: Atakan Akar
Bilkent ID: 22203140
Date: 13.11.2024

```
    jal bcon

    addi $t0, $zero, 8
    addi $t1, $zero, 13
    addi $t2, $zero, 16

    move $a0, $t0
    move $a1, $t1
    move $a2, $t2

    la $a3, final1
    jal bcon

    addi $t0, $zero, 8
    addi $t1, $zero, 16
    addi $t2, $zero, 16

    move $a0, $t0
    move $a1, $t1
    move $a2, $t2

    la $a3, final2
    jal bcon
    addi $t0, $zero, 8
    addi $t1, $zero, 15
    addi $t2, $zero, 16

    move $a0, $t0
    move $a1, $t1
    move $a2, $t2

    la $a3, final3
    jal bcon

    addi $v0, $zero, 10
    syscall

bcon:
    add $t3, $a0, $zero
    addi $t3, $t3, 4

    beq $a1, $t3, check
    j skip
```

CS224
Lab No.:4
Section No.: 1
Name: Atakan Akar
Bilkent ID: 22203140
Date: 13.11.2024

```
check:
   andi $t4, $a1, 3
   bne $t4, $zero, skip

branch:
   addi $t5, $zero, 1
   sw $t5, 0($a3)
   jr $ra

skip:
   sw $zero, 0($a3)
   jr $ra
```

xnori)

```
.data
result1: .word 0
result2: .word 0

.text



main:
   li $a0, 0x0F0F
   li $a1, 0x00FF
   jal xnori
   sw $v0, result1

   li $a0, 0xF0F0
   li $a1, 0x0F0F
   jal xnori
   sw $v0, result2

   loop: j loop

   addi $v0, $zero, 10

   syscall
```

CS224
Lab No.:4
Section No.: 1
Name: Atakan Akar
Bilkent ID: 22203140
Date: 13.11.2024

```
xnori:
   xor $v0, $a0, $a1
   nor $v0, $v0, $zero
   andi $v0, $v0, 0xFFFF
   jr $ra
```

G)

```
// Written by David_Harris@hmc.edu

// Top level system including MIPS and memories


module top  (input   logic       clk, reset,
             output  logic[31:0] writedata, dataadr,
             output  logic[31:0] readdata,
             output  logic      memwrite,
             output logic [31:0] instr,
             output logic [31:0] pc
             );


  // instantiate processor and memories
  mips mips (clk, reset, pc, instr, memwrite, dataadr, writedata, readdata);
  imem imem (pc[7:0], instr);
  dmem dmem (clk, memwrite, dataadr, writedata, readdata);

endmodule



// External data memory used by MIPS single-cycle processor

module dmem (input  logic       clk, we,
          input  logic[31:0]  a, wd,
          output logic[31:0]  rd);
```

CS224
Lab No.:4
Section No.: 1
Name: Atakan Akar
Bilkent ID: 22203140
Date: 13.11.2024

```
  logic  [31:0] RAM[63:0];

  assign rd = RAM[a[31:2]];    // word-aligned  read (for lw)

  always_ff @(posedge clk)
    if (we)
      RAM[a[31:2]] <= wd;      // word-aligned write (for sw)

endmodule



// External instruction memory used by MIPS single-cycle
// processor. It models instruction memory as a stored-program
// ROM, with address as input, and instruction as output


module imem ( input logic [7:0] addr, output logic [31:0] instr);

// imem is modeled as a lookup table, a stored-program byte-addressable ROM
          always_comb
            case (addr)                    // word-aligned fetch
//                  address                  instruction
//                  -------             -----------
      /*
                  8'h00: instr = 32'h20040002;          // disassemble, by hand
                  8'h04: instr = 32'h20050009;          // or with a program,
                  8'h08: instr = 32'h20060003;          // to find out what
                  8'h0c: instr = 32'h00a42822;          // xnori
                  8'h10: instr = 32'h0cc50004;        //BCON
                  8'h14: instr = 32'h20a50001;    //BURDA BİTİYOR
                  8'h18: instr = 32'h20c60001;
                  8'h1c: instr = 32'h0cc5fffe;        //BCON
                  8'h20: instr = 32'h00063520;
                  */
                  /*8'h00: instr = 32'h21080001;        // disassemble, by hand
                  8'h04: instr = 32'h2009FF00;          // or with a program,
                  8'h08: instr = 32'h20087051;          // to find out what
                  8'h0c: instr = 32'h052980AF;          // xnori
                  8'h10: instr = 32'h21310001;
                  8'h14: instr = 32'h01288022;    //BURDA BİTİYOR*/
```

CS224
Lab No.:4
Section No.: 1
Name: Atakan Akar
Bilkent ID: 22203140
Date: 13.11.2024

```
                //ESKİ

                8'h00: instr = 32'h20020005;          // disassemble, by hand
                8'h04: instr = 32'h2003000c;          // or with a program,
                8'h08: instr = 32'h2067fff7;   // to find out what
                8'h0c: instr = 32'h00e22025;          // this program does!
                8'h10: instr = 32'h00642824;
                8'h14: instr = 32'h00a42820;
                8'h18: instr = 32'h10a7000a;
                8'h1c: instr = 32'h0064202a;
                8'h20: instr = 32'h10800001;
                8'h24: instr = 32'h20050000;
                8'h28: instr = 32'h00e2202a;
                8'h2c: instr = 32'h00853820;
                8'h30: instr = 32'h00e23822;
                8'h34: instr = 32'hac670044;
                8'h38: instr = 32'h8c020050;
                8'h3c: instr = 32'h08000011;
                8'h40: instr = 32'h20020001;
                8'h44: instr = 32'hac020054;
                8'h48: instr = 32'h08000012; // j 48, so it will loop here


            default:  instr = {32{1'bx}};// unknown address
          endcase
endmodule


// single-cycle MIPS processor, with controller and datapath

module mips (input  logic       clk, reset,
          output logic[31:0]  pc,
          input  logic[31:0]  instr,
          output logic       memwrite,
          output logic[31:0]  aluout, writedata,
          input  logic[31:0]  readdata);

 logic       memtoreg, pcsrc, zero, alusrc, regdst, regwrite, jump;
 logic [2:0]  alucontrol;
 logic       bcons, xnoris;

 controller c (instr[31:26], instr[5:0], zero, memtoreg, memwrite, pcsrc,
                 alusrc, regdst, regwrite, jump, alucontrol, bcons, xnoris);
```

CS224
Lab No.:4
Section No.: 1
Name: Atakan Akar
Bilkent ID: 22203140
Date: 13.11.2024

```
  datapath dp (clk, reset, memtoreg, pcsrc, alusrc, regdst, regwrite, jump,
                    alucontrol, zero, pc, instr, aluout, writedata, readdata, bcons, xnoris);

endmodule
module controller(input  logic[5:0] op, funct,
                  input  logic     zero,
                  output logic     memtoreg, memwrite,
                  output logic     pcsrc, alusrc,
                  output logic     regdst, regwrite,
                  output logic     jump,
                  output logic[2:0] alucontrol,
                  output logic bcons,
                  output logic xnoris);

  logic [1:0] aluop;
  logic       branch;


  maindec md (op, memtoreg, memwrite, branch, alusrc, regdst, regwrite,
                    jump, aluop,bcons,xnoris);

  aludec  ad (funct, aluop, alucontrol);

  assign pcsrc = branch & zero;

endmodule

module maindec (input logic[5:0] op,
                    output logic memtoreg, memwrite, branch,
                    output logic alusrc, regdst, regwrite, jump,
                    output logic[1:0] aluop, output logic bcons,
                    output logic xnoris  );
  logic [10:0] controls;

  assign {regwrite, regdst, alusrc, branch, memwrite,
            memtoreg,  aluop, jump,bcons,xnoris} = controls;

 always_comb
   case(op)
     6'b000000: controls <= 11'b11000010000; // R-type
     6'b100011: controls <= 11'b10100100000; // LW
     6'b101011: controls <= 11'b00101000000; // SW
```

CS224
Lab No.:4
Section No.: 1
Name: Atakan Akar
Bilkent ID: 22203140
Date: 13.11.2024

```
    6'b000100: controls <= 11'b00010001000; // BEQ
    6'b001000: controls <= 11'b10100000000; // ADDI
    6'b000010: controls <= 11'b00000000100; // J
    6'b000011: controls <= 11'b00000000010; // bcons
    6'b000001: controls <= 11'b10100011001; // XNORI
    default:   controls <= 11'bxxxxxxxxx; // illegal op
  endcase
endmodule

module aludec (input    logic[5:0] funct,
           input    logic[1:0] aluop,
           output   logic[2:0] alucontrol);
 always_comb
  case(aluop)
    2'b00: alucontrol  = 3'b010;  // add  (for lw/sw/addi)
    2'b01: alucontrol  = 3'b110;  // sub   (for beq)
    2'b11: alucontrol  = 3'b101;  // xnori
    default: case(funct)         // R-TYPE instructions
       6'b100000: alucontrol  = 3'b010; // ADD
       6'b100010: alucontrol  = 3'b110; // SUB
       6'b100100: alucontrol  = 3'b000; // AND
       6'b100101: alucontrol  = 3'b001; // OR
       6'b101010: alucontrol  = 3'b111; // SLT
       default:   alucontrol  = 3'bxxx; // ???
      endcase
  endcase
endmodule

module datapath (input  logic clk, reset, memtoreg, pcsrc, alusrc, regdst,
             input  logic regwrite, jump,
                    input  logic[2:0]  alucontrol,
             output logic zero,
                    output logic[31:0] pc,
                input  logic[31:0] instr,
             output logic[31:0] aluout, writedata,
                input  logic[31:0] readdata,
                               input logic bcons,
          input logic xnoris);

 logic [4:0]  writereg;
 logic [31:0] pcnext, pcnextbr, pcplus4, pcbranch;
 logic [31:0] signimm, signimmsh, srca, srcb, result, immout;
logic [31:0] zeroe;
```

CS224
Lab No.:4
Section No.: 1
Name: Atakan Akar
Bilkent ID: 22203140
Date: 13.11.2024

```
 logic if1, if2, check, last;

  // next PC logic
  flopr #(32) pcreg(clk, reset, pcnext, pc);
  adder       pcadd1(pc, 32'b100, pcplus4);
  sl2         immsh(signimm, signimmsh);
  adder       pcadd2(pcplus4, signimmsh, pcbranch);

  assign if1 = (writedata - srca == 4) ? 1'b1 : 1'b0;
  assign if2 = (writedata[1:0] == 2'b00) ? 1'b1 : 1'b0;

  assign check = if1 & if2 & bcons;
  assign last = check | pcsrc;

  mux2 #(32)  pcbrmux(pcplus4, pcbranch, last,
                      pcnextbr);
  mux2 #(32)  pcmux(pcnextbr, {pcplus4[31:28],
                    instr[25:0], 2'b00}, jump, pcnext);

// register file logic
   regfile     rf (clk, regwrite, instr[25:21], instr[20:16], writereg,
                   result, srca, writedata);

   mux2 #(5)    wrmux (instr[20:16], instr[15:11], regdst, writereg);
   mux2 #(32)  resmux (aluout, readdata, memtoreg, result);
   signext         se (instr[15:0], signimm);
   zeroext         ze (instr[15:0], zeroe);
    mux2 #(32)  bisim (signimm, zeroe, xnoris, immout);
  // ALU logic
   mux2 #(32)  srcbmux (writedata, immout, alusrc, srcb);
   alu         alu (srca, srcb, alucontrol, aluout, zero);

endmodule


module regfile (input    logic clk, we3,
                input    logic[4:0]  ra1, ra2, wa3,
                input    logic[31:0] wd3,
                output   logic[31:0] rd1, rd2);

  logic [31:0] rf [31:0];

  assign rf[0] = 32'b0;
```

CS224
Lab No.:4
Section No.: 1
Name: Atakan Akar
Bilkent ID: 22203140
Date: 13.11.2024

```
  assign rf[1] = 32'b0;
  assign rf[2] = 32'b0;
  assign rf[3] = 32'b0;
  assign rf[5] = 32'b0;
  assign rf[4] = 32'b0;
  assign rf[6] = 32'b0;
  assign rf[7] = 32'b0;
  assign rf[8] = 32'b0;
  assign rf[9] = 32'b0;
  assign rf[10] = 32'b0;
  assign rf[11] = 32'b0;
  assign rf[12] = 32'b0;
  assign rf[13] = 32'b0;
  assign rf[14] = 32'b0;
  assign rf[15] = 32'b0;
  assign rf[16] = 32'b0;
  assign rf[17] = 32'b0;
  assign rf[18] = 32'b0;
  assign rf[19] = 32'b0;
  assign rf[20] = 32'b0;
  assign rf[21] = 32'b0;
  assign rf[22] = 32'b0;
  assign rf[23] = 32'b0;
  assign rf[24] = 32'b0;
  assign rf[25] = 32'b0;
  assign rf[26] = 32'b0;
  assign rf[27] = 32'b0;
  assign rf[30] = 32'b0;
  assign rf[31] = 32'b0;

  // three ported register file: read two ports combinationally
  // write third port on rising edge of clock. Register0 hardwired to 0.

  always_ff@(posedge clk)
    if (we3)
        rf [wa3] <= wd3;

  assign rd1 = (ra1 != 0) ? rf [ra1] : 0;
  assign rd2 = (ra2 != 0) ? rf[ ra2] : 0;

endmodule
```

CS224
Lab No.:4
Section No.: 1
Name: Atakan Akar
Bilkent ID: 22203140
Date: 13.11.2024

```
module alu(input  logic [31:0] a, b,
           input  logic [2:0]  alucont,
           output logic [31:0] result,
           output logic zero);
    always_comb
       case(alucont)
          3'b010: result = a + b;
          3'b110: result = a - b;
          3'b000: result = a & b;
          3'b001: result = a | b;
          3'b111: result = (a < b) ? 1 : 0;
          3'b101: result =~(a ^ b);
          default: result = {32{1'bx}};
       endcase

    assign zero = (result == 0) ? 1'b1 : 1'b0;
endmodule


module adder (input  logic[31:0] a, b,
              output logic[31:0] y);

     assign y = a + b;
endmodule

module sl2 (input  logic[31:0] a,
            output logic[31:0] y);

     assign y = {a[29:0], 2'b00}; // shifts left by 2
endmodule

module signext (input  logic[15:0] a,
                output logic[31:0] y);

  assign y = {{16{a[15]}}, a};    // sign-extends 16-bit a
endmodule

module zeroext (input  logic[15:0] a,
                output logic[31:0] y);

  assign y = {16'b0, a};
endmodule
```

CS224
Lab No.:4
Section No.: 1
Name: Atakan Akar
Bilkent ID: 22203140
Date: 13.11.2024

```
// parameterized register
module flopr #(parameter WIDTH = 8)
              (input logic clk, reset,
                    input logic[WIDTH-1:0] d,
               output logic[WIDTH-1:0] q);

  always_ff@(posedge clk, posedge reset)
    if (reset) q <= 0;
    else       q <= d;
endmodule


// paramaterized 2-to-1 MUX
module mux2 #(parameter WIDTH = 8)
             (input  logic[WIDTH-1:0] d0, d1,
              input  logic s,
              output logic[WIDTH-1:0] y);

   assign y = s ? d1 : d0;
endmodule
```