CSE 331
Final Project
Report

Single Cycle Datapath

AHMET ÖZYILMAZ

111044014

1 GİRİŞ

Bu dönemki BİL 331 dersinin final projesi. İnputs / outputs aşağıda gösterildiği şekildedir.

Aşağıdaki şekil projenin ana kısmının şeklidir. Gözüken input ile istenilen outputlar gözükmektedir.

```
module mips_core(ResultOut, pcOut, pcIn, clock);

input clock;
output [31:0] pcOut;
input  [31:0] pcIn;
wire   [31:0] pc;
output [31:0] ResultOut;
```

"mips_core.v"

```
// Read next instruction
wire [31:0] instruction;
mips_instr_mem instructionMemory(instruction, pc);
```

instructionların bulunduğu dosyadan instructionları çekip gerekli parçalara ayırmaktadır.

```
assign opcode = instruction[31:26];
assign rs     = instruction[25:21];
assign rt     = instruction[20:16];
assign rd     = instruction[15:11];
assign shamt  = instruction[10:6];
assign fcode  = instruction[5:0];
assign imm    = instruction[15:0];
assign jump   = instruction[25:0];
```

Ayrıca control unit clock ile birlikte instruction gelmesi ile birlikte gerekli control unit sinyallerini üretmektedir. Aşağıdaki resim.

```
wire [1:0] RegDest;
wire       Branch;
wire       Jump;
wire       MemRead;
wire [1:0] MemtoReg;
wire       MemWrite;
wire       ALUSrc;
wire       RegWrite;
wire [1:0] ALUOp;


// Set Control Bits
mips_control controlUnit(RegDest, Branch, Jump, MemRead,
MemtoReg, ALUOp, MemWrite, ALUSrc, RegWrite, opcode, fcode);
```

Şekil 1 Control unit sinyalleri

2 MODÜLLER

Single Cycle Datapath in kısımları.

- controlUnit
- registerUnit
- signExtender unit
- ALUCtr unit
- ALU unit
- Branch
- PC(program counter)
- Muxes(32 bit - 1 input)
- Muxes(32 bit - 3 input)
- Muxes(5bit - 3 input)
- Jump Unit

Datapath in ana kısımları bu şekilde.

3 Control Unit

```
module mips_control(RegDest, Branch, Jump, MemRead, MemtoReg,
ALUOp, MemWrite, ALUSrc, RegWrite, opcode, fcode);

input [5:0] opcode, fcode;
output [1:0] RegDest, MemtoReg;
output Branch, Jump, MemRead, MemWrite, ALUSrc, RegWrite;
output [1:0] ALUOp;

assign RegDest  = opcode == 6'b000000 ? 2'b01 : opcode == 6'b000011 ? 2'b10 : 2'b00;
//need check
assign Branch   = opcode == 6'b000100 ? 1'b1 : 1'b0;
assign Jump     = opcode == 6'b000010 || opcode == 6'b000011 ? 1'b1 : 1'b0;
assign MemRead  = opcode == 6'b100011    //lw
                  || opcode == 6'b100100 //lbu
                  || opcode == 6'b100101 //lhu
                  ? 1'b1 : 1'b0;
assign MemtoReg = opcode == 6'b100011    //lw
                  || opcode == 6'b100100 //lbu
                  || opcode == 6'b100101 //lhu
                  ? 2'b01
                  : opcode == 6'b000011 //jal
                  ? 2'b10
                  : opcode == 6'b001111 //lui
                  ? 2'b11
                  : 2'b00; //others
assign MemWrite = opcode == 6'b101011 ? 1'b1 : 1'b0;
assign ALUSrc   = opcode == 6'b100011 //lw
                  || opcode == 6'b100100 //lbu
                  || opcode == 6'b100101 //lhu
                  || opcode == 6'b101011 //sw
                  || opcode == 6'b001000 //addi
                  || opcode == 6'b001001 //addiu
                  || opcode == 6'b001100 //andi
                  || opcode == 6'b001101 //ori
                  || opcode == 6'b001010 //slti
                  || opcode == 6'b001011 //sltiu
                  ? 1'b1 : 1'b0;
assign RegWrite = opcode == 6'b000000
                  || opcode == 6'b100011 //lw
                  || opcode == 6'b100100 //lbu
                  || opcode == 6'b100101 //lhu
                  || opcode == 6'b001000 //addi
                  || opcode == 6'b001001 //addiu
                  || opcode == 6'b001100 //andi
                  || opcode == 6'b001101 //ori
                  || opcode == 6'b001111 //lui
```


```
                  || opcode == 6'b001100 //andi
                  || opcode == 6'b001101 //ori
                  || opcode == 6'b001111 //lui
                  || (opcode == 6'b000000 && fcode == 6'b000000) //sll
                  || (opcode == 6'b000000 && fcode == 6'b000010) //srl
                  || opcode == 6'b001010 //slti
                  || opcode == 6'b001011 //sltiu
                  ? 1'b1 : 1'b0;

assign ALUOp    = opcode == 6'b000000 //R-Type
```


```
                  ? 2'b10
                  : opcode == 6'b100011 //LW
                    || opcode == 6'b100100 //LBU
                    || opcode == 6'b100101 //LHU
                  ? 2'b00
                  : opcode == 6'b101011 //SW
                  ? 2'b00
                  : opcode == 6'b000100 //BEQ
                    || opcode == 6'b000010 //J
                  ? 2'b01
                  : 2'b10; //Possibly arithmetic ?

endmodule
```

Yorumlarda yazıldığı gibi R-I-J type instructionlara göre operand kodları göre ve function fielde göre sinyaller üretildi.

4 Register Unit

Register ünitesi üretilen kontrol sinyallerine göre ve clk sinyaline göre “registers.mem” dosyasında bulunan register değerlerinden istenilen memory birimin değerini döndürmektedir.

Clk sinyali her “positiveedge” olduğunda dosyadan okuma gerçekleşiyor.

registers.mem - Not Defteri

Dosya Düzen Biçim Görünüm Yardım

```
00000000000000000000000000000000
00000000000000000000000000000001
00000000000000000000000000000010
00000000000000000000000000000011
00000000000000000000000000000100
00000000000000000000000000000101
00000000000000000000000000000110
00000000000000000000000000000111
00000000000000000000000000001000
00000000000000000000000000001001
00000000000000000000000000001010
00000000000000000000000000001011
00000000000000000000000000001100
00000000000000000000000000001101
00000000000000000000000000001110
00000000000000000000000000001111
00000000000000000000000000010000
00000000000000000000000000010001
00000000000000000000000000010010
00000000000000000000000000010011
00000000000000000000000000010100
00000000000000000000000000010101
00000000000000000000000000010110
00000000000000000000000000010111
00000000000000000000000000011000
00000000000000000000000000011001
00000000000000000000000000011010
00000000000000000000000000011011
00000000000000000000000000011100
00000000000000000000000000011101
00000000000000000000000000011110
00000000000000000000000000011111
```

```
module mips_registers
(
    output reg [31:0] read_data_1, read_data_2,
    input [31:0] write_data,
    input [4:0] read_reg_1, read_reg_2, write_reg,
    input signal_reg_write, clk
);
    reg [31:0] registers [31:0];

    initial begin
        $readmemb("registers.mem", registers);
    end

    always @ (posedge clk)
    begin
        if (signal_reg_write) begin
            registers[write_reg] <= write_data;
        end
        else begin
            read_data_1 <= registers[read_reg_1];
            read_data_2 <= registers[read_reg_2];
        end
    end
endmodule
```

5 Sign Extend Unit

I type instructionlarda 16 bitin – 32 bit çevrilmesi için kullanılmaktadır.

```
module mips_sign_extender(extended, in);

input [15:0] in;
output [31:0] extended;

assign extended[15:0] = in;
assign extended[31:16] = {16{in[15]}};

endmodule
```

Girdisi 16 bit – çıktısı 32 bittir.

6 ALU – Control

```
module mips_alu_control(ALUControl, opcode, ALUOp);
input [5:0] opcode;
input [1:0] ALUOp;
output [3:0] ALUControl;
wire   [3:0] tempAluControl;
assign tempAluControl = opcode == 6'b100000 //add
                        || opcode == 6'b100001 //addu
                        || opcode == 6'b001000 //addi
                        || opcode == 6'b001001 //addiu
                        ? 4'b0010
                        : opcode == 6'b100010  //sub
                        || opcode == 6'b100011 //subu
                        ? 4'b0110
                        : opcode == 6'b100100 //and
                        ? 4'b0000
                        : opcode == 6'b100101 //or
                        ? 4'b0001
                        : opcode == 6'b101010 //slt
                        || opcode == 6'b001010 //slti
                        || opcode == 6'b001011 //sltiu
                        || opcode == 6'b101011 //sltu
                        ? 4'b0111
                        : opcode == 6'b000000 //sll
                        ? 4'b0100
                        : opcode == 6'b000010 //srl
                        ? 4'b0101
                        : opcode == 6'b100111 //nor
                        ? 4'b1100
                        : opcode == 6'b001100 //andi
                        ? 4'b0000
                        : opcode == 6'b001101 //ori
                        ? 4'b0001
                        : 4'bxxxx;

assign ALUControl = ALUOp == 2'b00
                    ? 4'b0010
                    : ALUOp == 2'b01
                    ? 4'b0110
                    : ALUOp == 2'b10
                    ? tempAluControl
                    : 4'bxxxx;

endmodule
```

7 ALU

ALU control modülünün üretmiş olduğu output sinyallerine göre gerekli işlemleri yapıp 32 bit sonuç döndürmektedir. Ayrıca branch instructionu geldiğinde

```
module mips_alu(Result, Zero, readData1, readData2, shamt, ALUCtr);

output [31:0] Result;
output        Zero;

input  [31:0] readData1;
input  [31:0] readData2; //after mux selection
input  [3:0]  ALUCtr;
input  [4:0]  shamt;

wire          overflowAdd;
wire          overflowSub;
wire          setLessThan;

wire   [31:0] readDataSum;
wire   [31:0] readDataSub;


assign readDataSum = readData1 + readData2;
assign readDataSub = readData1 - readData2;

assign overflowAdd = (readData1[31] == readData2[31] && readDataSum[31] != readData1[31])
assign overflowSub = (readData1[31] == readData2[31] && readDataSub[31] != readData1[31])

assign setLessThan = overflowSub ? ~(readData1[31]) : readData1[31];

assign Result   = ALUCtr == 4'b0010
                ? readDataSum
                : ALUCtr == 4'b0000
                ? readData1 & readData2
                : ALUCtr == 4'b0001
                ? readData1 | readData2
                : ALUCtr == 4'b1100
                ? ~(readData1 | readData2)
                : ALUCtr == 4'b0110
                ? readDataSub
                : ALUCtr == 4'b0111
                ? {{31{1'b0}}, setLessThan}
                : ALUCtr == 4'b0100
                ? readData2 << shamt
                : ALUCtr == 4'b0101
                ? readData2 >> shamt
                : 0;

assign Zero = (Result == 0);
```

8 Instruction memory

İnstuctionları memory dosyasından okuyup program counter a göre gerekli instructionu döndürüyor.

instr.txt - Not Defteri

Dosya Düzen Biçim Görünüm Yardım

```
00000000000000011111000000100000 : add
00100000000001000000000000000011 : addi
00100100000011000000000000000111 : addiu
00000000000000000001100000100001 : addu
00000001000010010101000000100100 : and
00110000010000000000000000001110 : andi
00010000001000100000000000000001 : beq ?
00001000000000000000000000000000 : j
00001100000000000000000000000000 : jal
00111100000001010000000000001111 : lui
10001100000011000000000000000000 : lw
10010000000000010000000000000000 : lbu
00000000000000010001000000100111 : nor -
00000000000000010001000000100101 : or
00110100000000011111111111111111 : ori
00000000000000010001000000101010 : slt
00101000000000110000000000000000 : slti
00101100000000110000000000000000 : sltiu
00000000000000010001000000101011 : sltu
00000000000010000000000010000000 : sll
00000000000010000000000010000010 : srl
10101100011000000000000000000000 : sw
00000000111010000000000000100010 : sub
00000000000000010001000000100010 : subu
```

```
module mips_instr_mem(instruction, program_counter);
input [31:0] program_counter;
output [31:0] instruction;

reg [31:0] instr_mem [255:0];

initial begin
    $readmemb("instruction.mem", instr_mem);
end

assign instruction = instr_mem[program_counter];

endmodule
```

9 Data Memory

data.mem - Not Defteri

Dosya Düzen Biçim Gör

```
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000001
00000000
00000000
00000000
00000010
00000000
00000000
00000000
00000011
00000000
00000000
00000000
00000100
00000000
00000000
00000000
00000101
00000000
00000000
00000000
00000110
00000000
00000000
00000000
00000111
00000000
00000000
00000000
00001000
00000000
```

instruction.mem - Not Defteri

Dosya Düzen Biçim Görünüm Yardım

```
00000000000000010001000000100000
00100000000001100000000000000001
00100100000001000000000000000001
00000000000000010010100000100001
00000000000000010011000000100100
00110000000011101010101010101010
00111100000010001111111111111111
10001100000010010000000000000000
00000000000000010101000000100101
00110100000010111010101010101010
00000000000000000110000010000000
00000000000000010110100000101010
00101000000011100000000000011111
00101100000011110000000000011111
00000000000000011000000000101011
00000000000000001000100010000010
00000000000000011001000000100010
00000000000000011001100000100011
10101100000000010000000000000000
```

Data memoryde lw – sw gibi memory instructionlarını okuma ve yazma işlemlerinde kullanılmaktadır. “data.mem” dosyasından işlemler yapılmaktadır.

```
module mips_data_mem (read_data, mem_address, write_data, sig_mer
output [31:0] read_data;
input [31:0] mem_address;
input [31:0] write_data;
input sig_mem_read;
input sig_mem_write;

//reg [7:0] data_mem [1023:0];
reg [31:0] data_mem  [255:0];

// I put this to get rid of the error below
// Error (10137): Verilog HDL Procedural Assignment error at mip:
reg [31:0] read_data;

initial begin
   $readmemb("data.mem", data_mem);
end

always @(mem_address or write_data or sig_mem_read or sig_mem_wr:
   if (sig_mem_read) begin
      read_data = {data_mem[mem_address]};
   end

   if (sig_mem_write) begin
      data_mem[mem_address] <= write_data[31:0];
      data_mem[mem_address+0] <= write_data[31:0];
      //data_mem[mem_address+2] <= write_data[23:16];
      //data_mem[mem_address+3] <= write_data[31:24];
      //data_mem[mem_address] <= write_data[31:0];
      //$writememb("data2.mem", data_mem);
   end
end

endmodule
```

Programın Testbenh çıktıları

111044014

mips_testbench.v | mips_sign_extender.v | mips_registers.v | mips_program_counter.v | mips_mux_5bit_3input.v

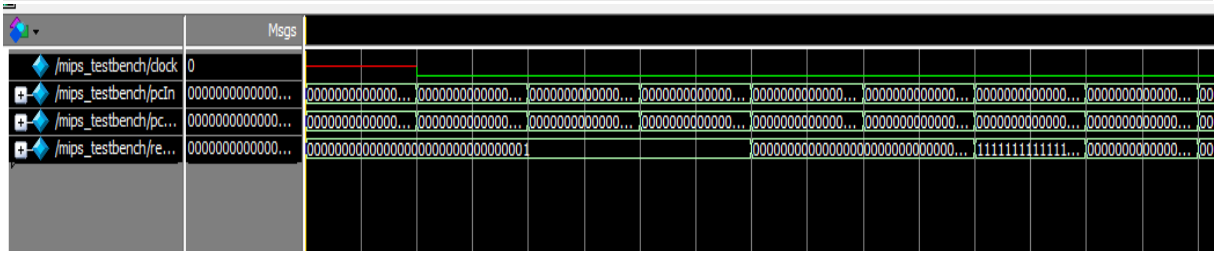
```
# Quartus II: Generate Tcl File for Project
# File: 111044014.tcl
# Generated on: Wed Jan 03 16:29:01 2018

# Load Quartus II Tcl Project package
package require ::quartus::project

set need_to_close_project 0
set make_assignments 1

# Check that the right project is open
if {[is_project_open]} {
   if {[string compare $quartus(project) "111044014"]} {
      puts "Project 111044014 is not open"
      set make_assignments 0
   }
} else {
   # Only open if not already open
   if {[project_exists 111044014]} {
      project_open -revision 111044014 111044014
   } else {
      project_new -revision 111044014 111044014
   }
   set need_to_close_project 1
}

# Make assignments
if {$make_assignments} {
   set_global_assignment -name FAMILY "Cyclone III"
   set_global_assignment -name DEVICE EP3C55F484C6
   set_global_assignment -name TOP_LEVEL_ENTITY mips_core
   set_global_assignment -name ORIGINAL_QUARTUS_VERSION 13.1
   set_global_assignment -name PROJECT_CREATION_TIME_DATE "15:15:54  DECEMBER 23, 2017"
   set_global_assignment -name LAST_QUARTUS_VERSION 13.1
   set_global_assignment -name PROJECT_OUTPUT_DIRECTORY output_files
   set_global_assignment -name MIN_CORE_JUNCTION_TEMP 0
   set_global_assignment -name MAX_CORE_JUNCTION_TEMP 85
   set_global_assignment -name ERROR_CHECK_FREQUENCY_DIVISOR 1
   set_global_assignment -name NOMINAL_CORE_SUPPLY_VOLTAGE 1.2V
   set_global_assignment -name EDA_SIMULATION_TOOL "ModelSim-Altera (Verilog)"
   set_global_assignment -name EDA_OUTPUT_DATA_FORMAT "VERILOG HDL" -section_id eda_simulation
   set_global_assignment -name VERILOG_FILE mips_testbench.v
   set_global_assignment -name VERILOG_FILE mips_sign_extender.v
   set_global_assignment -name VERILOG_FILE mips_registers.v
   set_global_assignment -name VERILOG_FILE mips_program_counter.v
   set_global_assignment -name VERILOG_FILE mips_mux_5bit_3input.v
   set_global_assignment -name VERILOG_FILE mips_mux_5bit.v
```



```
if {$make_assignments} {
    set_global_assignment -name FAMILY "Cyclone III"
    set_global_assignment -name DEVICE EP3C55F484C6
    set_global_assignment -name TOP_LEVEL_ENTITY mips_core
    set_global_assignment -name ORIGINAL_QUARTUS_VERSION 13.1
    set_global_assignment -name PROJECT_CREATION_TIME_DATE "15:15:54  DECEMBER 23, 2017"
    set_global_assignment -name LAST_QUARTUS_VERSION 13.1
    set_global_assignment -name PROJECT_OUTPUT_DIRECTORY output_files
    set_global_assignment -name MIN_CORE_JUNCTION_TEMP 0
    set_global_assignment -name MAX_CORE_JUNCTION_TEMP 85
    set_global_assignment -name ERROR_CHECK_FREQUENCY_DIVISOR 1
    set_global_assignment -name NOMINAL_CORE_SUPPLY_VOLTAGE 1.2V
    set_global_assignment -name EDA_SIMULATION_TOOL "ModelSim-Altera (Verilog)"
    set_global_assignment -name EDA_OUTPUT_DATA_FORMAT "VERILOG HDL" -section_id eda_simulation
    set_global_assignment -name VERILOG_FILE mips_testbench.v
    set_global_assignment -name VERILOG_FILE mips_sign_extender.v
    set_global_assignment -name VERILOG_FILE mips_registers.v
    set_global_assignment -name VERILOG_FILE mips_program_counter.v
    set_global_assignment -name VERILOG_FILE mips_mux_5bit_3input.v
    set_global_assignment -name VERILOG_FILE mips_mux_5bit.v
    set_global_assignment -name VERILOG_FILE mips_mux_32bit_4input.v
    set_global_assignment -name VERILOG_FILE mips_mux_32bit_3input.v
    set_global_assignment -name VERILOG_FILE mips_mux_32bit.v
    set_global_assignment -name VERILOG_FILE mips_load_word_halfword_byte_selector.v
    set_global_assignment -name VERILOG_FILE mips_instr_mem.v
    set_global_assignment -name VERILOG_FILE mips_data_mem.v
    set_global_assignment -name VERILOG_FILE mips_core.v
    set_global_assignment -name VERILOG_FILE mips_control.v
    set_global_assignment -name VERILOG_FILE mips_alu_control.v
    set_global_assignment -name VERILOG_FILE mips_alu.v
    set_global_assignment -name POWER_PRESET_COOLING_SOLUTION "23 MM HEAT SINK WITH 200 LFPM AIRFLOW"
    set_global_assignment -name POWER_BOARD_THERMAL_MODEL "NONE (CONSERVATIVE)"
    set_global_assignment -name PARTITION_NETLIST_TYPE SOURCE -section_id Top
    set_global_assignment -name PARTITION_FITTER_PRESERVATION_LEVEL PLACEMENT_AND_ROUTING -section_id Top
    set_global_assignment -name PARTITION_COLOR 16764057 -section_id Top
    set_instance_assignment -name PARTITION_HIERARCHY root_partition -to | -section_id Top

    # Commit assignments
    export_assignments

    # Close project
    if {$need_to_close_project} {
        project_close
    }
}
```

```
hed NativeLink simulation (quartus_sh -t "c:/altera/13.1/quartus/common/tcl/internal/nativelink/qnativesim.tcl" --rtl_sim "1
NativeLink execution see the NativeLink log file D:/2017-2018/Organizasyon/Projects/331 Computer Organization/FinalProject/U
```