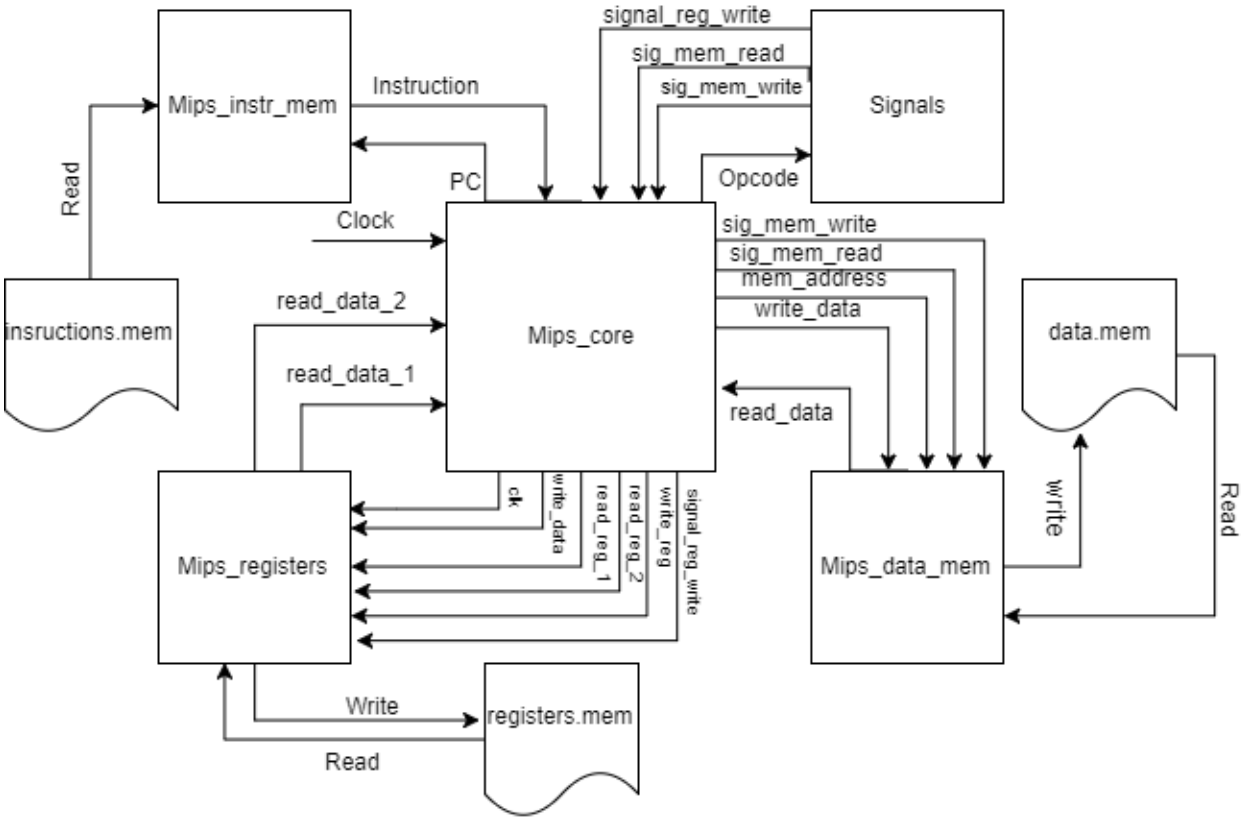
CSE 331 COMPUTER ORGANIZATION

FINAL PROJECT

1. Introduction

1.1. Big Picture



1.2. Life cycle of 1 instruction

Her PC (Program Counter) değişiminde sıradaki instruction alınarak mips_core modulüne gönderilir ve bu modülde parçalara ayrılır(opcode-function code-shift amount-rs-rt-rd) ayrıca sonra ki aşamalarda instruction'nun ilk 15 bitine sign extend işlemini yapmak yerine ilk başta bu işlem yapılır ve bir wire 'da tutulur. Daha sonrasında oluşan opcode signals modülüne gönderilerek control sinyallerine (signal_reg_write, sig_mem_read, sig_mem_write) atamalar yapılır ve oluşan bu sinyallerin değerlerine göre registerdan ve datadan okuma/yazma işlemleri yapılır.

2. METHOD

2.1. Mips_core

Input: clock

Bu modül input olarak clock alır ve clock tetiklemesi ile çalışır. İlk olarak PC değeri 0 olarak ilklendirilir ve sonlarında ilk instruction'ı almak üzere PC "mips_instr_mem" modülüne input olarak gönderilir ve instruction alınır. Intruction alındıktan sonra insruction opcode, function code, rs, rt, rd olarak parçalara ayrılır.

Bulunan opcode ise "signals" modülüne gönderilerek burada "signal_reg_write, sig_mem_read, sig_mem_write" sinyallerine opcode'a göre 1 ya da 0 değerleri verilir.

Instruction'ın I-Type ve ya R-Type olmasına göre rd belirlendikten sonra "mips_registers" modülü kullanılarak read_data_1 ve read_data_2 değerleri "registers.mem" dosyasından okunur.

Registers modulünden gelen değerler de kullanılarak I-type instructionlar için immediate kısmı sign extend ve zero extend yapılır, memory işlemi için memory adresi hesaplanır ve jump için jump adresi bulunur. Bu işlemler her instruction için önceden yapılır ve daha sonrasında gerektiğinde kullanılır.

Eğer memory den okuma işlemi yapılacaksa "mips_data_mem" modülünü kullanılarak "read_data_mem" değeri memoryden okunur ve sonrasında gerekli işlemlerin yapılması için aşağıdaki kısma girer(Gösterilen kısımda kodun tamamı bulunmamaktadır.).

```
106   always @(posedge clock)
107   begin
108      PC <= PC+1; //update PC
109      if(fun_and_op[1] === `RTYPE) //If R-type instruction then control the function code
110      begin
155      else if (fun_and_op[1] === `J)                                          //J
156      begin
157         jumpAdr[27:26] = 2'b0;
158         jumpAdr[31:28] =  PC[31:28];
159         //then make equal PC to this jumpadress
160         PC <= jumpAdr;
161      end else if(fun_and_op[1] === `JAL)
162      begin
170      begin
171         if (fun_and_op[1] === `ADDI)                                         //addi
172         begin
173            write_data <= read_data_1 + extended_val;
174         end else if (fun_and_op[1] === `ADDIU)                               //addiu
175         begin
176            write_data <= $unsigned(read_data_1) + $unsigned(extended_val);
177         end else if (fun_and_op[1] === `ANDI)                                //andi
178         begin
179            write_data <= read_data_1 & zero_extended_val;
180         end else if (fun_and_op[1] === `ORI)                                 //ori
181         begin
182            write_data <= read_data_1 | zero_extended_val;
183         end else if (fun_and_op[1] === `SLTI)                                //slti
184         begin
```

Bu kısım da bulunan opcode'da göre gerekli işlemler yapılar PC ve write_data ya atamalar yapılır. Yapılan bu değişiklikler ile sinyallere göre eğer write_data değeri ya data'ya ya da registerlar'a (diğer moduller kullanılarak) yazılır. Yani data ve registerlar update edilir.

Bütün bu işlemler bittikten sonra diğer bir clock ile yeni instruction için aynı işlemler yapılır.

2.2. Mips_registers

Inputs: write_data, read_reg_1, read_reg_2, write_reg, signal_reg_write, clk

Outputs: read_data_1, read_data_2

Bu modülde ilk olarak bir kerelik "registers.mem" dosyası okunarak "registers" array'ine atılır.

Sonrasın da resimde gösterildiği üzere @always(*) bloğunda input olarak verilen read_reg_1, read_reg_2 kullanılarak read_data_1 ve read_data_2 değerleri registers'dan çekilir ve

@always(write_data or signal_reg_write or write_reg) bloğunda da eğer signal_reg_write sinyali 1 ise verilen write_reg register'ına write_data yazılır ve bu registers "res_registers.mem" dosyasına yazılır.

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

//this always block read data from the registers
always @(*)begin
   read_data_1 = registers[read_reg_1];
   read_data_2 = registers[read_reg_2];
end

//this always block write the data to the register and then create
//a new register file if signal_reg_write equal to 1
always @(write_data or signal_reg_write or write_reg) //for combinational
begin
   if (signal_reg_write) begin
      registers[write_reg] = write_data;
      $writememb("res_registers.mem", registers);
   end
end
```

2.3. Mips_instr_mem

Input: program_counter

Output: instruction

Bu modülde ilk olarak bir kerelik "instructions.mem" dosyası okunarak "instr_mem" array'ine atılır.

Daha sonrasında @always(*) bloğunda program counter değerine göre instruction çekilir.

```
 //reads the instructions.
initial begin
    $readmemb(".\\instruction.mem", instr_mem);
 end

 //this always block takes the PCth instruction
always @(*) begin
    instruction = instr_mem[program_counter];
end
 endmodule
```

2.4. Mips_data_mem

Inputs: sig_mem_write, sig_mem_read, write_data, mem_address

Output: read_data

Bu modülde ilk olarak bir kerelik “data.mem” dosyası okunarak “data_mem” array’ine atılır.

Eğer sig_mem_read sinyali 1 ise gönderilen memory adresinde bulunan değer read_data’ ya yazılır. Eğer sig_mem_write sinyali 1 ise input olarak gelen write_dat değeri gönderilen memory addresine yazılır ve update edilmiş memory(data_mem) “res_data.mem” dosyasına yazılır.

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


//this always block read the data that is in the specified memory address
//if read signal is equal to 1
always @(mem_address or sig_mem_read)begin
    if (sig_mem_read) begin
        read_data[31:0] = data_mem[mem_address];
    end
end


//this always block writes the data that is in the specified memory address
//if write signal is equal to 1 and also creates a new data file as result
always @(mem_address or write_data  or sig_mem_write) begin
    if (sig_mem_write) begin
        data_mem[mem_address] = write_data[31:0];
        $writememb("res_data.mem", data_mem);
    end
end
```

2.5. Signals

Input: opcode

Outputs: sig_mem_read, sig_mem_write, signal_reg_write

Bu modülde verilen opcode değerine göre instruction tipi veya instruction'ın ne olduğu direk anlaşılıyor ve ona göre control sinyallerini üretiyor ve output olarak geri gönderiyor.

```
always @(*)
begin
   if(opcode === `RTYPE) begin
      sig_mem_read = 0; //not read memory
      sig_mem_write = 0; //not write to memory
      signal_reg_write = 1; //write to the register
   end else begin
      if (opcode === `J || opcode === `JR) begin
         sig_mem_write = 1'b0;
         sig_mem_read = 1'b0;
         signal_reg_write = 1'b0;
      end else if (opcode === `JAL)begin
         sig_mem_write = 1'b0;
         sig_mem_read = 1'b0;
         signal_reg_write = 1'b1;
      end else if (opcode === `BEQ || opcode === `BNE)begin
         sig_mem_write = 1'b0;
         sig_mem_read = 1'b0;
         signal_reg_write = 1'b0;
      end else if (opcode === `LW || opcode === `LBU || opcode === `LHU)begin
         signal_reg_write = 1'b1;
         sig_mem_write = 1'b0;
         sig_mem_read = 1'b1;
      end else if (opcode === `SW || opcode === `SH || opcode === `SB)begin
         signal_reg_write = 1'b0;
         sig_mem_write = 1'b1;
         sig_mem_read = 1'b0;
      end else begin
```

3. RESULT

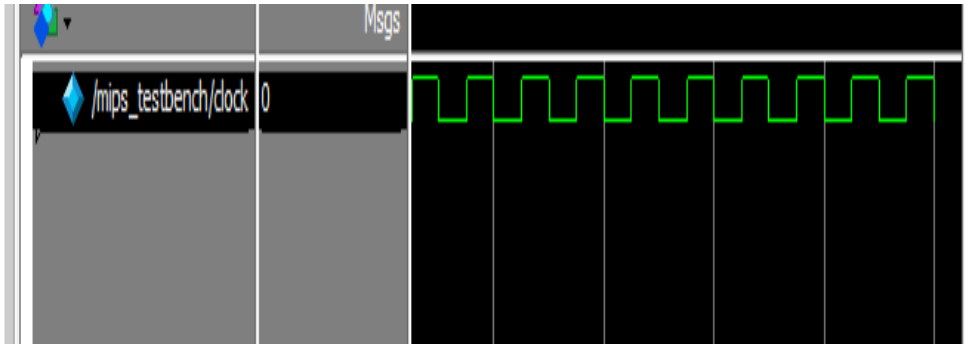
3.1. TestBench Results

Test için kullanılan Instructionlar şu şekildedir.

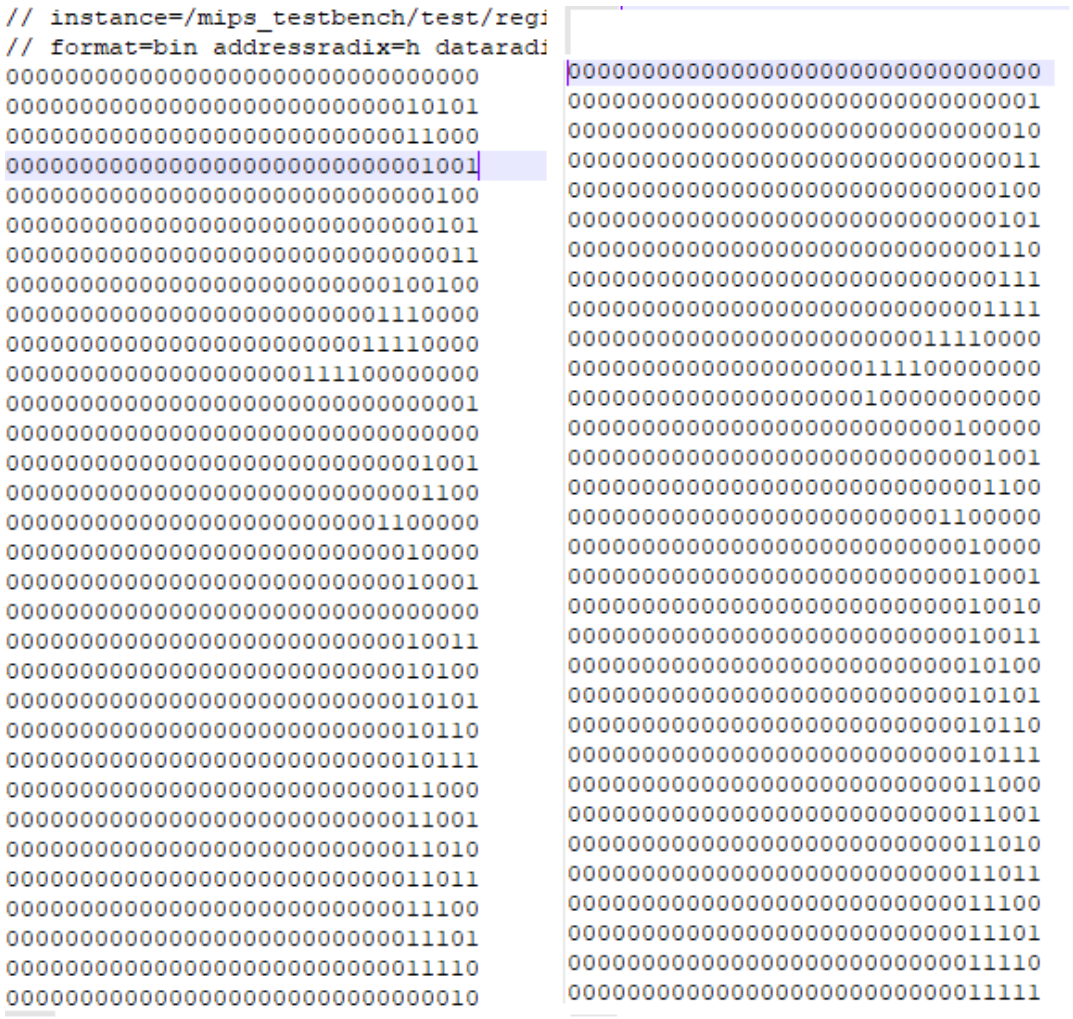
```
00000001110011010000100000100000
00001100000000000000000000000111
00000001110011010001000000100001
00000001110011010001100000100010
00000001110011010010000000100100
00000001110011010000000000100101
00000001110011010010100010000011
00000001110011010011000010100010
00000001110011010011100010000000
00001000000000000000000000001011
00000001110011010011100000101011
10100100011000100000000000000010
00010000001000010000000000000101
00000010000100011001000000100000
10010010000010000000000000000010
10010110001010010000000000000100
10101100011000100000000000000010
10001100001000100000000000000011
00000000101001000001100000100000
00000010000011111010000000100000
00000010000100011001000000100000
00100010011100110000000000000001
00000010100101010101100000101010
00101010110011000000000000010000
10100010111011010000000000100000
10100111000110010000000010000000
00000010000100011001000000100010
```

- Modelsim Sonucu

```
# RS: 00000000000000000000000000001100, RT: 00000000000000000000000000001001,RD adress: 00001 ,PC : 00000000000000000000000000000000 ,clock :0
# RS: 00000000000000000000000000001100, RT: 00000000000000000000000000001001,RD adress: 11111 ,PC : 00000000000000000000000000000001 ,clock :1
# RS: 00000000000000000000000000001100, RT: 00000000000000000000000000001001,RD adress: 11111 ,PC : 00000000000000000000000000000001 ,clock :0
# RS: 00000000000000000000000000001100, RT: 00000000000000000000000000001001,RD adress: 00110 ,PC : 00000000000000000000000000000111 ,clock :1
# RS: 00000000000000000000000000001100, RT: 00000000000000000000000000001001,RD adress: 00110 ,PC : 00000000000000000000000000000111 ,clock :0
# RS: 00000000000000000000000000001100, RT: 00000000000000000000000000001001,RD adress: 00111 ,PC : 00000000000000000000000000001000 ,clock :1
# RS: 00000000000000000000000000001100, RT: 00000000000000000000000000001001,RD adress: 00111 ,PC : 00000000000000000000000000001000 ,clock :0
# RS: 00000000000000000000000000000000, RT: 00000000000000000000000000000000,RD adress: 00000 ,PC : 00000000000000000000000000001001 ,clock :1
# RS: 00000000000000000000000000000000, RT: 00000000000000000000000000000000,RD adress: 00000 ,PC : 00000000000000000000000000001001 ,clock :0
# RS: 00000000000000000000000000000011, RT: 00000000000000000000000000000010,RD adress: 00010 ,PC : 00000000000000000000000000001011 ,clock :1
# RS: 00000000000000000000000000000011, RT: 00000000000000000000000000000010,RD adress: 00010 ,PC : 00000000000000000000000000001011 ,clock :0
# RS: 00000000000000000000000000010101, RT: 00000000000000000000000000010101,RD adress: 00001 ,PC : 00000000000000000000000000001100 ,clock :1
# RS: 00000000000000000000000000010101, RT: 00000000000000000000000000010101,RD adress: 00001 ,PC : 00000000000000000000000000001100 ,clock :0
# RS: 00000000000000000000000000010101, RT: 00000000000000000000000000100100,RD adress: 00010 ,PC : 00000000000000000000000000010001 ,clock :1
# RS: 00000000000000000000000000010101, RT: 00000000000000000000000000100100,RD adress: 00010 ,PC : 00000000000000000000000000010001 ,clock :0
# RS: 00000000000000000000000000000101, RT: 00000000000000000000000000000100,RD adress: 00011 ,PC : 00000000000000000000000000010010 ,clock :1
# RS: 00000000000000000000000000000101, RT: 00000000000000000000000000000100,RD adress: 00011 ,PC : 00000000000000000000000000010010 ,clock :0
# RS: 00000000000000000000000000010000, RT: 00000000000000000000000001100000,RD adress: 01000 ,PC : 00000000000000000000000000010011 ,clock :1
# RS: 00000000000000000000000000010000, RT: 00000000000000000000000001100000,RD adress: 01000 ,PC : 00000000000000000000000000010011 ,clock :0
# RS: 00000000000000000000000000010000, RT: 00000000000000000000000000010001,RD adress: 10010 ,PC : 00000000000000000000000000010100 ,clock :1
# RS: 00000000000000000000000000010000, RT: 00000000000000000000000000010001,RD adress: 10010 ,PC : 00000000000000000000000000010100 ,clock :0
# RS: 00000000000000000000000000010011, RT: 00000000000000000000000000010011,RD adress: 10011 ,PC : 00000000000000000000000000010101 ,clock :1
# RS: 00000000000000000000000000010011, RT: 00000000000000000000000000010011,RD adress: 10011 ,PC : 00000000000000000000000000010101 ,clock :0
# RS: 00000000000000000000000000010100, RT: 00000000000000000000000000010101,RD adress: 01011 ,PC : 00000000000000000000000000010110 ,clock :1
# RS: 00000000000000000000000000010100, RT: 00000000000000000000000000010101,RD adress: 01011 ,PC : 00000000000000000000000000010110 ,clock :0
# RS: 00000000000000000000000000010110, RT: 00000000000000000000000000000001,RD adress: 01100 ,PC : 00000000000000000000000000010111 ,clock :1
# RS: 00000000000000000000000000010110, RT: 00000000000000000000000000000001,RD adress: 01100 ,PC : 00000000000000000000000000010111 ,clock :0
# RS: 00000000000000000000000000010111, RT: 00000000000000000000000000001001,RD adress: 01101 ,PC : 00000000000000000000000000011000 ,clock :1
# RS: 00000000000000000000000000010111, RT: 00000000000000000000000000001001,RD adress: 01101 ,PC : 00000000000000000000000000011000 ,clock :0
# RS: 00000000000000000000000000011000, RT: 00000000000000000000000000011001,RD adress: 11001 ,PC : 00000000000000000000000000011001 ,clock :1
# RS: 00000000000000000000000000011000, RT: 00000000000000000000000000011001,RD adress: 11001 ,PC : 00000000000000000000000000011001 ,clock :0
# RS: 00000000000000000000000000010000, RT: 00000000000000000000000000010001,RD adress: 10010 ,PC : 00000000000000000000000000011010 ,clock :1
# RS: 00000000000000000000000000010000, RT: 00000000000000000000000000010001,RD adress: 10010 ,PC : 00000000000000000000000000011010 ,clock :0
```



- File Değişiklikleri
  - Register ve Res_register



```
// instance=/mips_testbench/test/regi
// format=bin addressradix=h dataradi
00000000000000000000000000000000
00000000000000000000000000010101
00000000000000000000000000011000
00000000000000000000000000001001
00000000000000000000000000000100
00000000000000000000000000000101
00000000000000000000000000000011
00000000000000000000000000100100
00000000000000000000000001110000
00000000000000000000000011110000
00000000000000000000111100000000
00000000000000000000000000000001
00000000000000000000000000000000
00000000000000000000000000001001
00000000000000000000000000001100
00000000000000000000000001100000
00000000000000000000000000010000
00000000000000000000000000010001
00000000000000000000000000000000
00000000000000000000000000010011
00000000000000000000000000010100
00000000000000000000000000010101
00000000000000000000000000010110
00000000000000000000000000010111
00000000000000000000000000011000
00000000000000000000000000011001
00000000000000000000000000011010
00000000000000000000000000011011
00000000000000000000000000011100
00000000000000000000000000011101
00000000000000000000000000011110
00000000000000000000000000000010
```

```
00000000000000000000000000000000
00000000000000000000000000000001
00000000000000000000000000000010
00000000000000000000000000000011
00000000000000000000000000000100
00000000000000000000000000000101
00000000000000000000000000000110
00000000000000000000000000000111
00000000000000000000000000001111
00000000000000000000000011110000
00000000000000000000111100000000
00000000000000000001000000000000
00000000000000000000000000100000
00000000000000000000000000001001
00000000000000000000000000001100
00000000000000000000000001100000
00000000000000000000000000010000
00000000000000000000000000010001
00000000000000000000000000010010
00000000000000000000000000010011
00000000000000000000000000010100
00000000000000000000000000010101
00000000000000000000000000010110
00000000000000000000000000010111
00000000000000000000000000011000
00000000000000000000000000011001
00000000000000000000000000011010
00000000000000000000000000011011
00000000000000000000000000011100
00000000000000000000000000011101
00000000000000000000000000011110
00000000000000000000000000011111
```

- Data ve Res_data

```
00000000000000000000000000010001
00000000000000000000000000010010
00000000000000000000000000010011
00000000000000000000000000010100
00000000000000000000000000010101
xxxxxxxxxxxxxxxx0000000000000010
xxxxxxxxxxxxxxxx0000000000001001
00000000000000000000000000011000
00000000000000000000000000011001
00000000000000000000000000011010
00000000000000000000000000011011
00000000000000000000000000011100
00000000000000000000000000011101
00000000000000000000000000011110
00000000000000000000000000011111
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx0000000000001001
xxxxxxxxxxxxxxxx0000000000011001
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

NOT: Insructionları değiştirmek için projenin içindeki “simulation\modelsim” kısmında bulunan dosyada değişiklik yapılması gerekmektedir.