Vector Processor

ساختن ماژول‌ها

ALU

ابتدا واحد ریاضی درون پردازنده را می‌سازیم.

```
module ALU (
    input wire signed [31:0] A [0:15], // 16 words of 32-bit inputs
    input wire signed [31:0] B [0:15], // 16 words of 32-bit inputs
    input wire op,                     // Control signal: 0 - add, 1 - multiply
    output reg [31:0] C [0:15],        // 16 words of 32-bit outputs (lo)
    output reg [31:0] D [0:15]         // 16 words of 64-bit outputs (hi)
);

// {D[i], C[i]} = (op ? A[i] * B[i] : A[i] + B[i])

    reg [63:0] add_result [0:15];
    reg [63:0] mult_result [0:15];

    genvar i;
    generate
        for (i = 0; i < 16; i = i + 1) begin
            assign add_result[i] = A[i] + B[i];
            assign mult_result[i] = A[i] * B[i];
            assign C[i] = (op ? mult_result[i][31:0] : add_result[i][31:0]);
            assign D[i] = (op ? mult_result[i][63:32] : add_result[i][63:32]);
        end
    endgenerate

endmodule
```

و آرایه `A` و `B` ورودی‌ها هستند. و در صورتی که `op` فعال باشد عملیات ضرب و در غیر این صورت عملیات جمع انجام می‌شود. 32 بیت پرارزش حاصل در `D` و 32 بیت کم‌ارزش در `C` نوشته می‌شوند.

RAM

```
module RAM (
    input wire clk,
    input wire we,  // Write Enable
    input wire [8:0] addr,  // 9-bit address to access 512 depth (2^9 = 512)
    input wire [3:0] cnt,
    input wire [31:0] data_in [0:15],  // 16 words of 32-bit input
    output reg [31:0] data_out [0:15]  // 16 words of 32-bit output data
);
```

```
//      if we is on, loads data_in[0:cnt] into mem[addr:addr+cnt]  (@ negedge clk)
//        otherwise, loads mem[addr:addr+cnt] into data_in[0:cnt]  (@ posedge clk)

    // Memory array: 512 x 32-bit
    reg [31:0] mem [0:511];

    integer i;

    always @(negedge clk) begin
        if (we) begin
            // Write cnt words at the specified address range
            for (i = 0; i < cnt + 1; i = i + 1) begin
                if(addr + i < 512) mem[addr + i] <= data_in[i];
            end
        end
    end
    always @(posedge clk) begin
        if (!we) begin
            // Read cnt words from the specified address range
            for (i = 0; i < cnt + 1; i = i + 1) begin
                if(addr + i < 512)  data_out[i] <= mem[addr + i];
                else data_out[i] <= 32'bz; // handling corner-cases
            end
        end
    end

endmodule
```

در اینجا یک حافظه با عمق 512 و عرض 32 بیت داریم. که امکان بارگذاری/ذخیره‌سازی ۱۶ خانه‌ی پشت سر هم از حافظه را دارد. در حقیقت در (cnt + 1) خانه بارگذاری/ذخیره‌سازی می‌کند. یعنی از خانه `addr` تا `addr + cnt` . از آنجا که ممکن است این بازه از کل حافظه بیرون بزنه برای این حالت ایف گذاشتیم تا اجرای برنامه رو مختل نکنه و خروجی منطقی داشته باشه.

در سراسر این طراحی همه عناصر حافظه سنکرون هستند. عملیات خواندن در لبه‌ی بالارونده و نوشتن در لبه‌ی پایین‌رونده انجام میشه.

Register File

```
module REG (
    input wire clk,
    input wire [1:0] sel,
    input wire [31:0] write_data [0:15],
    input wire [31:0] alu_data1 [0:15],
    input wire [31:0] alu_data2 [0:15],
    input wire we,
    input wire walu,
    output reg [31:0] read_data [0:15],
    output wire [31:0] A1_out [0:15],
```

```
    output wire [31:0] A2_out [0:15]
);

//      if walu is on, loads alu_data1 into A3 and alu_data2 into A4.
//          Which are supposedly the results of
//                  the arithmetic operation. (@ negedge clk)
//          otherwise,
//                      if we is on, loads write_data into A[sel] (@ negedge clk)
//                      if we is off, loads A[sel] into read_data (@ posedge clk)

    // Define four 32-bit registers
    reg [31:0] A1 [0:15];
    reg [31:0] A2 [0:15];
    reg [31:0] A3 [0:15];
    reg [31:0] A4 [0:15];

    assign A1_out = A1;
    assign A2_out = A2;

    always @(negedge clk) begin
        //$display("this is REG talking, walu: %b we: %b", walu, we);
        if (walu) begin
            A3 <= alu_data1;
            A4 <= alu_data2;
            //$display("  %h %h", alu_data1[1], alu_data2[1]);
        end else if (we) begin
            //$display("hell yeah baby, sel: %b %b %h", {walu, we}, sel, write_data[0]);
            case (sel)
                2'b00: A1 <= write_data;
                2'b01: A2 <= write_data;
                2'b10: A3 <= write_data;
                2'b11: A4 <= write_data;
            endcase
        end
    end
    always @(posedge clk) begin
        if(!we && !walu) begin
            case (sel)
                2'b00: read_data <= A1;
                2'b01: read_data <= A2;
                2'b10: read_data <= A3;
                2'b11: read_data <= A4;
            endcase
            //$display("heavens yeah baby, sel: %b %h", sel, read_data[0]);
        end
    end

endmodule
```

ین رجیستر فایل با توجه به نیازهای پردازنده طراحی شده است. دو نوع ورودی/خروجی داریم.

در ورودی/خروجی اول، دیتای `write_data` در یکی از 4 رجیستر نوشته می‌شود و دیتای رجیسترها در `read_data` نوشته می‌شود. که این دو حالت توسط `we` معین می‌شوند.

در ورودی/خروجی دوم. دیتای دو رجیستر `A1` و `A2` همواره در `A1_out` و `A2_out` نوشته می‌شود. و اگر `walu` فعال باشد. خروجی‌های `ALU` که `alu_out1` و `alu_out2` هستند، در `A3` و `A4` نوشته می‌شوند. همچنین اگر `walu` فعال باشد ورودی/خروجی های نوع اول غیرفعال می‌شوند.

CPU

```
module CPU(
    input clk,
    input [2:0] op,
    input [8:0] ram_addr,
    input [3:0] ram_cnt,
    input [1:0] reg_sel,
    input [31:0] ram_input [0:15],
    output [31:0] ram_output [0:15]
);

wire [31:0] A1_out [0:15];
wire [31:0] A2_out [0:15];
wire [31:0] alu_out1 [0:15];
wire [31:0] alu_out2 [0:15];
reg alu_op, reg_we, reg_walu, ram_we;
reg [31:0] reg_write_data [0:15];
reg [31:0] reg_read_data [0:15];
reg [31:0] ram_data_in [0:15];
reg [31:0] ram_data_out [0:15];

assign ram_data_in = (op[2] ? ram_input : reg_read_data);
assign ram_output = ram_data_out;
assign reg_write_data = ram_data_out; // added in testbenching :)

ALU kalu(A1_out, A2_out, alu_op, alu_out1, alu_out2);
REG kreg(clk, reg_sel, reg_write_data, alu_out1, alu_out2,
            reg_we, reg_walu, reg_read_data, A1_out, A2_out);
RAM kram(clk, ram_we, ram_addr, ram_cnt, ram_data_in, ram_data_out);

// piece of work to put everything together.
//  this CPU supports the 6 following commands
//         1. Loads RAM[ram_addr] into REG[reg_sel]
//         2. Loads REG[reg_sel] into RAM[ram_addr]
//         3. stores (A1 + A2) in {A4, A3}
//         4. stores (A1 * A2) in {A4, A3}
//         5. Loads RAM[ram_addr] into ram_output
//         6. Loads ram_input into RAM[ram_addr]    (assigment from outside the cpu)

localparam [3:0] RamToReg = 3'b000,
```

```
                RegToRam = 3'b001,
                addi     = 3'b010,
                mult     = 3'b011,
                RamToOut = 3'b100,
                OutToRam = 3'b101;

always @(op) begin
    reg_we = 1'b0;
    ram_we = 1'b0;
    reg_walu = 1'b0;
    alu_op = 1'b0;
    //$display($time, "nigga %b", op);
    case(op)
        RamToReg: reg_we = 1'b1;
        RegToRam: ram_we = 1'b1;
        addi: begin
            alu_op = 1'b0;
            reg_walu = 1'b1;
        end
        mult: begin
            alu_op = 1'b1;
            reg_walu = 1'b1;
        end
        OutToRam: ram_we = 1'b1;
    endcase
end

//always @(posedge clk) begin
    //$monitor(" on yo ass %h %h", kreg.write_data[0], ram_data_out[0]);
    //$display("shit shit %h %h %h %b %b", reg_write_data[0], ram_data_out[0],
    //                          ram_data_in[0], ram_we, op);
//end

endmodule
```

کامنت‌های درون کد به پاس قدردانی از زمان صرف شده برای دیباگ در کد باقی مونده.

ابتدا همه ماژول‌های مورد نیاز رو می‌سازیم و سیماشون رو به هم وصل می‌کنیم. حالا کافی‌است سیگنال‌های کنترلی مورد نیاز رو تعیین کنیم.

جهت ساده‌سازی حالت‌بندی روی دستورات متخلف از `localparam` ها استفاده می‌کنیم.

صحت سنجی

کار طراحی پردازنده به پایان رسیده و نوبت به صحت سنجی کد می‌رسه. (لازم به ذکره که در طول فرایند تستینگ و دیباگ کد بارها تغییر کرده تا به استیت فعلیش برسه)

ALU TB

در ابتدا از صحت کارکرد واحد ریاضی به تنهایی اطمینان حاصل می‌کنیم.

```
module TB_ALU;

reg signed [31:0] A [0:15];
reg signed [31:0] B [0:15];
reg op;
wire [31:0] C [0:15];
wire [31:0] D [0:15];

ALU keshi(A, B, op, C, D);

integer i;

initial begin
    for(i = 0; i < 16; i = i + 1) begin
        A[i] = $random;
        B[i] = $random;
    end
    op = 1'b0;
    #10
    for(i = 0; i < 16; i = i + 1) begin
        $display("%d + %d = %d", A[i], B[i], $signed({D[i], C[i]}));
    end
    op = 1'b1;
    #10
    for(i = 0; i < 16; i = i + 1) begin
        $display("%d * %d = %d", A[i], B[i], $signed({D[i], C[i]}));
    end
end

endmodule
```

خروجی برنامه در تصویر زیر آمده است. که صحیح است.

```
#
#    303379748 + -1064739199 =           -761359451
# -2071669239 + -1309649309 =          -3381318548
#    112818957 +  1189058957 =           1301877914
# -1295874971 + -1992863214 =          -3288738185
#     15983361 +   114806029 =            130789390
#    992211318 +   512609597 =           1504820915
#   1993627629 +  1177417612 =           3171045241
#   2097015289 +  -482925370 =           1614089919
#   -487095099 +  -720121174 =          -1207216273
#   1924134885 + -1143836041 =            780298844
# -1993157102 +  1206705039 =           -786452063
#   2033215986 +  -411658546 =           1621557440
#   -201295128 +  -490058043 =           -691353171
#    777537884 +  -561108803 =            216429081
# -1767155667 + -1297668507 =          -3064824174
# -1309711773 +    91457290 =          -1218254483
#    303379748 * -1064739199 =  -323020309878341852
# -2071669239 * -1309649309 =  2713160187332905851
#    112818957 *  1189058957 =   134148391340247849
# -1295874971 * -1992863214 =  2582501559649216794
#     15983361 *   114806029 =     1834986206483469
#    992211318 *   512609597 =   508617043858818846
#   1993627629 *  1177417612 =  2347332282154401948
#   2097015289 *  -482925370 = -1012701884335981930
#   -487095099 *  -720121174 =   350767494541526226
#   1924134885 * -1143836041 = -2200894829208390285
# -1993157102 *  1206705039 = -2405152718502036978
#   2033215986 *  -411658546 =  -836990736500716356
#   -201295128 *  -490058043 =    98646296493114504
#    777537884 *  -561108803 =  -436283351378392852
# -1767155667 * -1297668507 =  2293182256032479169
# -1309711773 *    91457290 =  -119782689439675170
```

RAM TB

در این مرحله از صحت عملیات های خواندن و نوشتن (از خارج پردازنده) روی رم اطمینان حاصل می‌کنیم.

```
module TB_RAM;

reg clk;
reg [2:0] op;
reg [8:0] ram_addr;
reg [3:0] ram_cnt;
reg [1:0] reg_sel;
reg [31:0] ram_input [0:15];
wire [31:0] ram_output [0:15];

CPU keshi(clk, op, ram_addr, ram_cnt, reg_sel, ram_input, ram_output);

integer i;

localparam [3:0] RamToReg = 3'b000,
                 RegToRam = 3'b001,
```

```
                addi     = 3'b010,
                mult     = 3'b011,
                RamToOut = 3'b100,
                OutToRam = 3'b101;

    task write_random (input [8:0] a);
        begin
            #1 op = OutToRam;
            ram_addr = a;
            for(i = 0; i < 16; i = i + 1) begin
                ram_input[i] = $random;
            end
            $display(" wrote in %d  : %h %h %h %h %h %h %h %h %h %h %h %h %h %h %h %h", ram_add
            #5 clk = 1;
            #5 clk = 0;
        end
    endtask

    task read (input [8:0] a);
        begin
            #1 op = RamToOut;
            ram_addr = a;
            #5 clk = 1;
            #5 clk = 0;
            $display(" read from %d : %h %h %h %h %h %h %h %h %h %h %h %h %h %h %h %h", ram_add
        end
    endtask

    initial begin
        clk = 0;
        ram_cnt = 15;

        write_random(12);
        read(12);
        // testing simple writing and reading, expecting the two lines to be equal
        write_random(42);
        read(13);
        // writing in [42] and then reading from [13] to make sure the thing works
        // expecting the output to be the same but shifted by one to the left
        // also the last number should be xxxxxxxx
        read(42);  // reading from [42]
        read(10);  // reading from [10]
        write_random(496);  // writing in [496:511] to check corner case
        read(496); // reading from [496:511] to check corner case
        read(500);
        // reading from [500:516] to check corner case
        // although it's an invalid input the extra 4 slots are gonna be high impedance
        #1 ram_cnt = 2;
        write_random(12);
        // writing in [12:14] to check the functionality of ram_cnt
        #1 ram_cnt = 15;
```

```
    read(12);
    // expecting only the first 3 numbers to change
end

endmodule
```

مثال‌های زیادی در تکه کد بررسی شده اند. خروجی کد در تصویر زیر آمده‌است. که صحت کد را تایید می‌کند.

```
#  wrote in  12  : 12153524 c0895e81 8484d609 b1f05663 06b97b0d 46df998d b2c28465 89375212 00f3e301 06d7cd0d 3b23f176 1e8dcd3d 76d457ed 462df78c 7cfde9f9 e33724c6
#  read from  12 : 12153524 c0895e81 8484d609 b1f05663 06b97b0d 46df998d b2c28465 89375212 00f3e301 06d7cd0d 3b23f176 1e8dcd3d 76d457ed 462df78c 7cfde9f9 e33724c6
#  wrote in  42  : e2f784c5 d513d2aa 72aff7e5 bbd27277 8932d612 47ecdb8f 793069f2 e77696ce f4007ae8 e2ca4ec5 2e58495c de8e28bd 96ab582d b2a72665 b1ef6263 0573870a
#  read from  13 : c0895e81 8484d609 b1f05663 06b97b0d 46df998d b2c28465 89375212 00f3e301 06d7cd0d 3b23f176 1e8dcd3d 76d457ed 462df78c 7cfde9f9 e33724c6 xxxxxxxx
#  read from  42 : e2f784c5 d513d2aa 72aff7e5 bbd27277 8932d612 47ecdb8f 793069f2 e77696ce f4007ae8 e2ca4ec5 2e58495c de8e28bd 96ab582d b2a72665 b1ef6263 0573870a
#  read from  10 : xxxxxxxx xxxxxxxx 12153524 c0895e81 8484d609 b1f05663 06b97b0d 46df998d b2c28465 89375212 00f3e301 06d7cd0d 3b23f176 1e8dcd3d 76d457ed 462df78c
#  wrote in 496  : c03b2280 10642120 557845aa cecccc9d cb203e96 8983b813 86bc380d a9a7d653 359fdd6b eaa62ad5 81174a02 d7563eae 0effe91d e7c572cf 11844923 0509650a
#  read from 496 : c03b2280 10642120 557845aa cecccc9d cb203e96 8983b813 86bc380d a9a7d653 359fdd6b eaa62ad5 81174a02 d7563eae 0effe91d e7c572cf 11844923 0509650a
#  read from 500 : cb203e96 8983b813 86bc380d a9a7d653 359fdd6b eaa62ad5 81174a02 d7563eae 0effe91d e7c572cf 11844923 0509650a zzzzzzzz zzzzzzzz zzzzzzzz zzzzzzzz
#  wrote in  12  : e5730aca 9e314c3c 7968bdf2 452e618a 20c4b341 ec4b34d8 3c20f378 c48a1289 75c50deb 5b0265b6 634bf9c6 571513ae de7502bc 150fdd2a 85d79a0b b897be71
#  read from  12 : e5730aca 9e314c3c 7968bdf2 b1f05663 06b97b0d 46df998d b2c28465 89375212 00f3e301 06d7cd0d 3b23f176 1e8dcd3d 76d457ed 462df78c 7cfde9f9 e33724c6
```

REGRAM TB

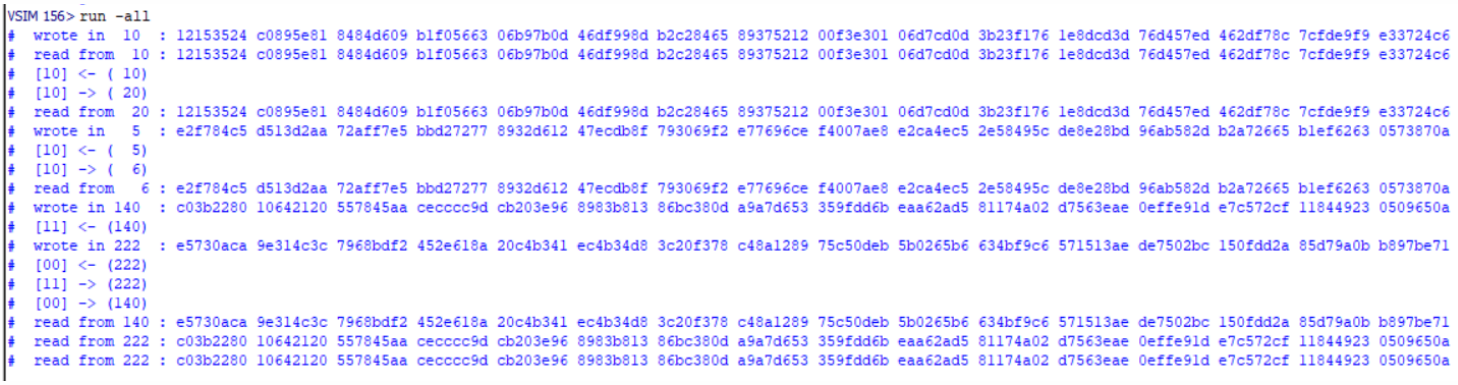
حالا نوبت بررسی صحت رابطه بین رم و رجیستر فایله.

```
module TB_REGRAM;

reg clk;
reg [2:0] op;
reg [8:0] ram_addr;
reg [3:0] ram_cnt;
reg [1:0] reg_sel;
reg [31:0] ram_input [0:15];
wire [31:0] ram_output [0:15];

integer i;

CPU keshi(clk, op, ram_addr, ram_cnt, reg_sel, ram_input, ram_output);

localparam [3:0] RamToReg = 3'b000,
                 RegToRam = 3'b001,
                 addi     = 3'b010,
                 mult     = 3'b011,
                 RamToOut = 3'b100,
                 OutToRam = 3'b101;

task write_random (input [8:0] a);
    begin
        #1 op = OutToRam;
        ram_addr = a;
        for(i = 0; i < 16; i = i + 1) begin
            ram_input[i] = $random;
        end
        $display(" wrote in %d  : %h %h %h %h %h %h %h %h %h %h %h %h %h %h %h %h", ram_add
        #5 clk = 1;
        #5 clk = 0;
```

```
    end
endtask
task read (input [8:0] a);
    begin
        #1 op = RamToOut;
        ram_addr = a;
        #5 clk = 1;
        #5 clk = 0;
        $display(" read from %d : %h %h %h %h %h %h %h %h %h %h %h %h %h %h %h %h", ram_add
    end
endtask

task ram_to_reg (input [8:0] a, [1:0] b);
    begin
        #1 op = RamToReg;
        ram_addr = a;
        reg_sel = b;
        $display(" [%b] <- (%d)", b, a);
        #5 clk = 1;
        #5 clk = 0;
    end
endtask

task reg_to_ram (input [8:0] a, [1:0] b);
    begin
        #1 op = RegToRam;
        ram_addr = a;
        reg_sel = b;
        $display(" [%b] -> (%d)", b, a);
        #5 clk = 1;
        #5 clk = 0;
    end
endtask

initial begin
    clk = 0;
    ram_cnt = 15;

    #5

    write_random(10);
    read(10);
    ram_to_reg(10, 2);
    reg_to_ram(20, 2);
    read(20);

    write_random(5);
    ram_to_reg(5, 2);
    reg_to_ram(6, 2);
    read(6);
```

```
    write_random(140);
    ram_to_reg(140, 3);
    write_random(222);
    ram_to_reg(222, 0);

    reg_to_ram(222, 3);
    reg_to_ram(140, 0);
    read(140);
    read(222);
    read(222);

end

//always @(posedge clk) $display($time, " ++++++++++++++++++++++++++++++");
//always @(negedge clk) $display($time, " ------------------------------");

endmodule
```

خروجی کد در تصویر زیر آمده که با خروجی مورد انتظار مطابقت دارد.



TB

و در نهایت تست بنچی که همه ویژگی‌های پردازنده رو به چالش می‌کشه.

```
module TB;

reg clk;
reg [2:0] op;
reg [8:0] ram_addr;
reg [3:0] ram_cnt;
reg [1:0] reg_sel;
reg [31:0] ram_input [0:15];
wire [31:0] ram_output [0:15];

integer i;

CPU keshi(clk, op, ram_addr, ram_cnt, reg_sel, ram_input, ram_output);

localparam [3:0] RamToReg = 3'b000,
```

```
                RegToRam = 3'b001,
                addi     = 3'b010,
                mult     = 3'b011,
                RamToOut = 3'b100,
                OutToRam = 3'b101;

    task write_random (input [8:0] a);
        begin
            #1 op = OutToRam;
            ram_addr = a;
            for(i = 0; i < 16; i = i + 1) begin
                ram_input[i] = $random;
            end
            $display(" wrote in %d  : %h %h %h %h %h %h %h %h %h %h %h %h %h %h %h %h", ram_add
            #5 clk = 1;
            #5 clk = 0;
        end
    endtask
    task read (input [8:0] a);
        begin
            #1 op = RamToOut;
            ram_addr = a;
            #5 clk = 1;
            #5 clk = 0;
            $display(" read from %d : %h %h %h %h %h %h %h %h %h %h %h %h %h %h %h %h", ram_add
        end
    endtask

    task ram_to_reg (input [8:0] a, [1:0] b);
        begin
            #1 op = RamToReg;
            ram_addr = a;
            reg_sel = b;
            $display(" [%b] <- (%d)", b, a);
            #5 clk = 1;
            #5 clk = 0;
        end
    endtask

    task reg_to_ram (input [8:0] a, [1:0] b);
        begin
            #1 op = RegToRam;
            ram_addr = a;
            reg_sel = b;
            $display(" [%b] -> (%d)", b, a);
            #5 clk = 1;
            #5 clk = 0;
        end
    endtask

    task addition;
```

```
    begin
        #1 op = addi;
        $display("[32] = [0] + [1]");
        #5 clk = 1;
        #5 clk = 0;
    end
endtask

task multiplication;
    begin
        #1 op = mult;
        $display("[32] = [0] * [1]");
        #5 clk = 1;
        #5 clk = 0;
    end
endtask

initial begin
    clk = 0;
    ram_cnt = 15;

    #5

    write_random(10);
    write_random(56);
    ram_to_reg(10, 0);
    ram_to_reg(56, 1);
    addition();
    reg_to_ram(100, 2);
    reg_to_ram(120, 3);
    read(100);
    read(120);

    write_random(42);
    write_random(452);
    ram_to_reg(42, 0);
    ram_to_reg(452, 1);
    multiplication();
    reg_to_ram(215, 2);
    reg_to_ram(400, 3);
    read(215);
    read(400);

    write_random(354);
    write_random(461);
    ram_to_reg(354, 0);
    ram_to_reg(461, 1);
    addition();
    reg_to_ram(387, 2);
    reg_to_ram(491, 3);
    read(387);
```

```
    read(491);

    write_random(67);
    write_random(380);
    ram_to_reg(67, 0);
    ram_to_reg(380, 1);
    multiplication();
    reg_to_ram(155, 2);
    reg_to_ram(408, 3);
    read(155);
    read(408);

end

//always @(posedge clk) $display($time, " ++++++++++++++++++++++++++++++");
//always @(negedge clk) $display($time, " ------------------------------");

endmodule
```

با دیدن تصویر زیر در خروجی کد، خیالمون از صحت کل پروژه راحت میشه.

```
VSIM 6> run -all
#  wrote in  10  : 12153524 c0895e81 8484d609 b1f05663 06b97b0d 46df998d b2c28465 89375212 00f3e301 06d7cd0d 3b23f176 1e8dcd3d 76d457ed 462df78c 7cfde9f9 e33724c6
#  wrote in  56  : e2f784c5 d513d2aa 72aff7e5 bbd27277 8932d612 47ecdb8f 793069f2 e77696ce f4007ae8 e2ca4ec5 2e58495c de8e28bd 96ab582d b2a72665 b1ef6263 0573870a
#  [00] <- ( 10)
#  [01] <- ( 56)
# [32] = [0] + [1]
#  [10] -> (100)
#  [11] -> (120)
#  read from 100 : f50cb9e9 959d312b f734cdee 6dc2c8da 8fec511f 8ecc751c 2bf2ee57 70ade8e0 f4f45de9 e9a21bd2 697c3ad2 fd1bf5fa 0d7fb01a f8d51df1 2eed4c5c e8aaabd0
#  read from 120 : ffffffff ffffffff ffffffff ffffffff ffffffff 00000000 00000000 ffffffff ffffffff ffffffff 00000000 ffffffff 00000000 ffffffff 00000000 ffffffff
#  wrote in  42  : c03b2280 10642120 557845aa cecccc9d cb203e96 8983b813 86bc380d a9a7d653 359fdd6b eaa62ad5 81174a02 d7563eae 0effe91d e7c572cf 11844923 0509650a
#  wrote in 452  : e5730aca 9e314c3c 7968bdf2 452e618a 20c4b341 ec4b34d8 3c20f378 c48a1289 75c50deb 5b0265b6 634bf9c6 571513ae de7502bc 150fdd2a 85d79a0b b897be71
#  [00] <- ( 42)
#  [01] <- (452)
# [32] = [0] * [1]
#  [10] -> (215)
#  [11] -> (400)
#  read from 215 : fd823900 a86d4380 3e305cb4 2923c9a2 c9cac616 3a182c08 df0e9d18 aca1886b a255b039 6db27c6e f3952f8c 098b8444 b4026b4c 2fa288f6 f0133281 fb09056a
#  read from 400 : 069d1b3c f9bcd35d 2888d134 f2b44934 f93b677c 091ee6c0 e3848155 140e17fd 18ab5721 f868de0a cec6579e f22af640 fe08de28 fe01b20e f7a4317f fe9857de
#  wrote in 354  : 42f24185 27f2554f 9dcc603b 1d06333a bf23327e 0aaa4b15 78d99bf1 6c9c4bd9 31230762 2635fb4c 4fa1559f 47b9a18f 7c6da9f8 dbcd60b7 cfc4569f ae7d945c
#  wrote in 461  : adcbc05b 44de3789 a4ae3249 e8233ed0 ebfec0d7 a8c7fc51 4b212f96 061d7f0c e12ccec2 6457edc8 bb825a77 1ef2ed3d 090cdb12 bf05007e 36e5816d 1cd9e739
#  [00] <- (354)
#  [01] <- (461)
# [32] = [0] + [1]
#  [10] -> (387)
#  [11] -> (491)
#  read from 387 : f0be01e0 6cd08cd8 427a9284 0529720a ab21f355 b3724766 c3facb87 72b9cae5 124fd624 8a8de914 0b23b016 66ac8ecc 857a850a 9ad26135 06a9d80c cb577b95
#  read from 491 : ffffffff 00000000 ffffffff 00000000 ffffffff ffffffff 00000000 00000000 00000000 00000000 00000000 00000000 00000000 ffffffff 00000000 ffffffff
#  wrote in  67  : 0fd28f1f e9ebf6d3 42d92f85 bc148878 2dda595b 248b4b49 9ff2ae3f 150caf2a 2c156358 c33f3886 c71a0c8e ce2ff29c 7d3599fa 937dbc26 39961773 d18bb4a3
#  wrote in 380  : 9799a82f d9d292b3 afd8565f 22290d44 7bf8fdf7 e59b36cb f3091ae6 2d28db5a 14cfc129 f682e2ed ed536cda b29fb665 da8ae2b5 efbe94df 3cf11979 2231ff44
#  [00] <- ( 67)
#  [01] <- (380)
# [32] = [0] * [1]
#  [10] -> (155)
#  [11] -> (408)
#  read from 155 : e81b9eb1 76d7eb89 c7c7505b cb9a57e0 512325cd 3e8918e3 46f1f29a 00de82c4 637a4118 0979a00e 118498ec 96499f8c bd9891c2 4980dd1a e77e505b 0b3d584c
#  read from 408 : f98c2288 034ae44f eb11c5c4 f6efd5e4 163481f2 fc3b7738 04dd4297 03b696cf 03957251 0240780b 04268652 0f0e505e edadf8eb 06e3deb9 0db56985 f9cb7b69
```