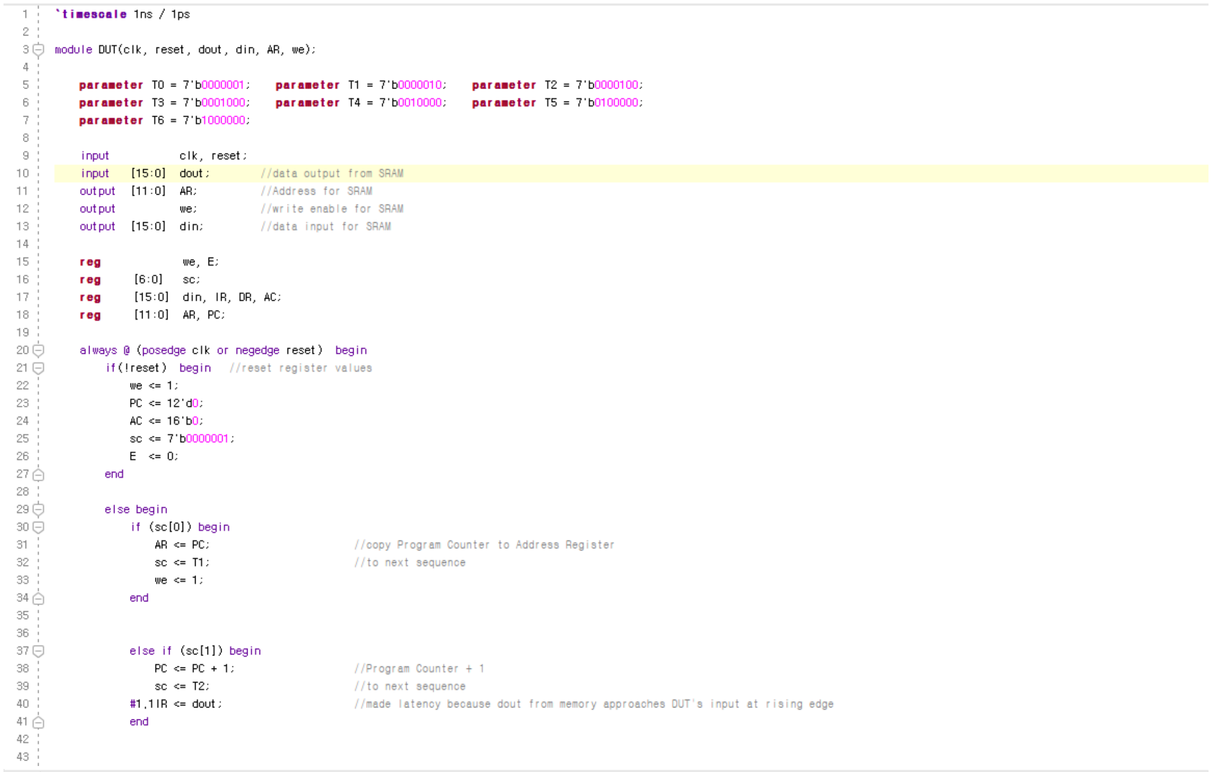
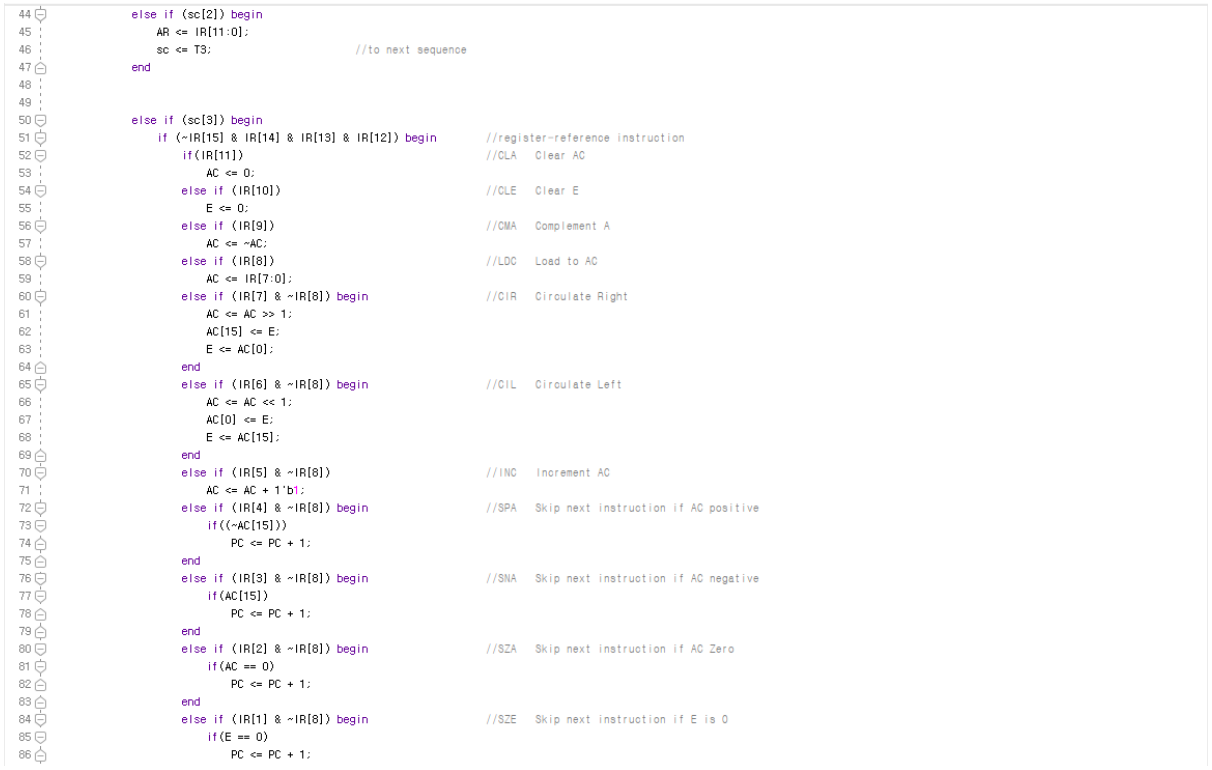
HW#7_2017112002_손보성

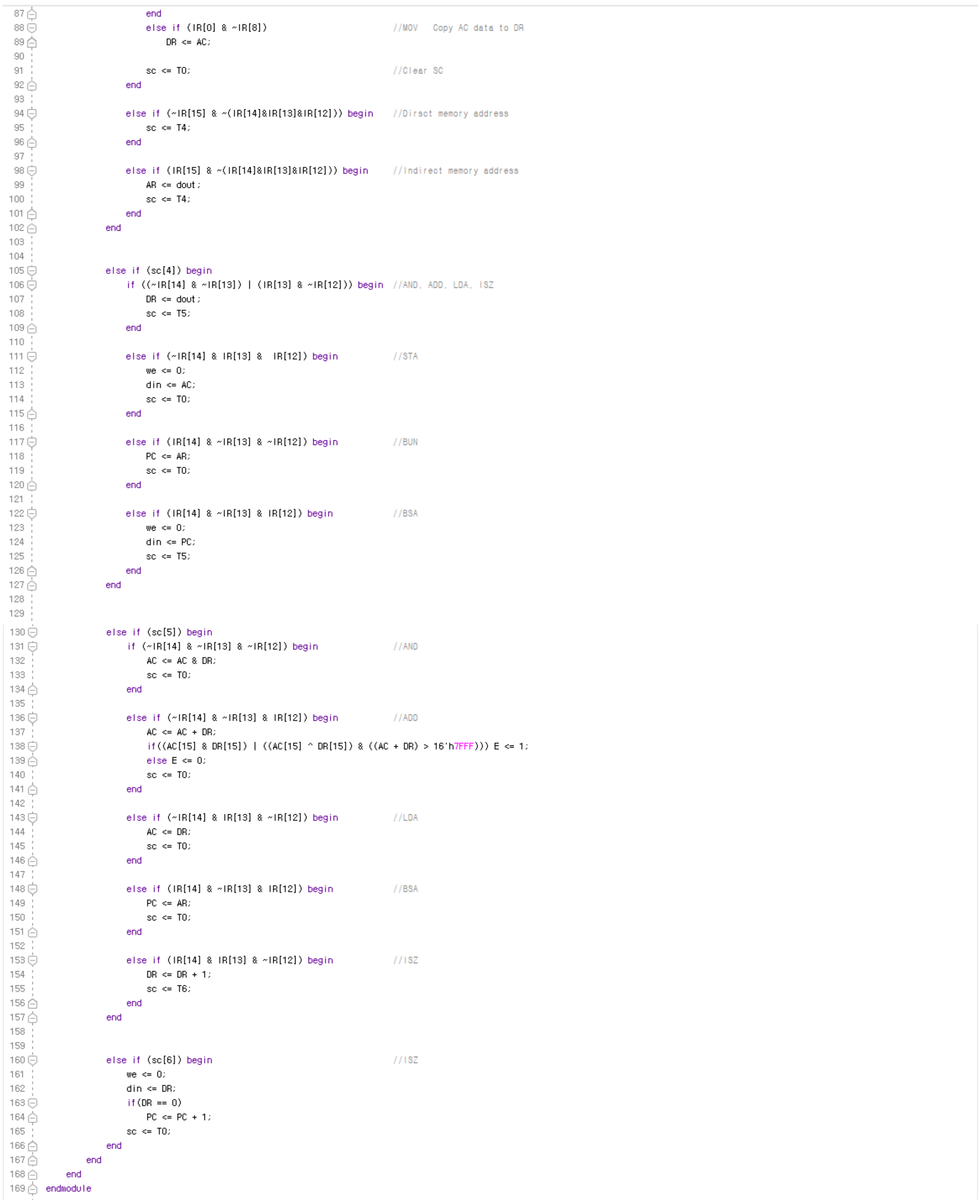
1. CPU DUT



```verilog
`timescale 1ns / 1ps

module DUT(clk, reset, dout, din, AR, we);

    parameter T0 = 7'b0000001;    parameter T1 = 7'b0000010;    parameter T2 = 7'b0000100;
    parameter T3 = 7'b0001000;    parameter T4 = 7'b0010000;    parameter T5 = 7'b0100000;
    parameter T6 = 7'b1000000;

    input           clk, reset;
    input   [15:0]  dout;        //data output from SRAM
    output  [11:0]  AR;          //Address for SRAM
    output          we;          //write enable for SRAM
    output  [15:0]  din;         //data input for SRAM

    reg             we, E;
    reg     [6:0]   sc;
    reg     [15:0]  din, IR, DR, AC;
    reg     [11:0]  AR, PC;

    always @ (posedge clk or negedge reset)  begin
        if(!reset)  begin   //reset register values
            we <= 1;
            PC <= 12'd0;
            AC <= 16'b0;
            sc <= 7'b0000001;
            E  <= 0;
        end

        else begin
            if (sc[0]) begin
                AR <= PC;                       //copy Program Counter to Address Register
                sc <= T1;                       //to next sequence
                we <= 1;
            end


            else if (sc[1]) begin
                PC <= PC + 1;                   //Program Counter + 1
                sc <= T2;                       //to next sequence
            #1.1IR <= dout;                     //made latency because dout from memory approaches DUT's input at rising edge
            end

```



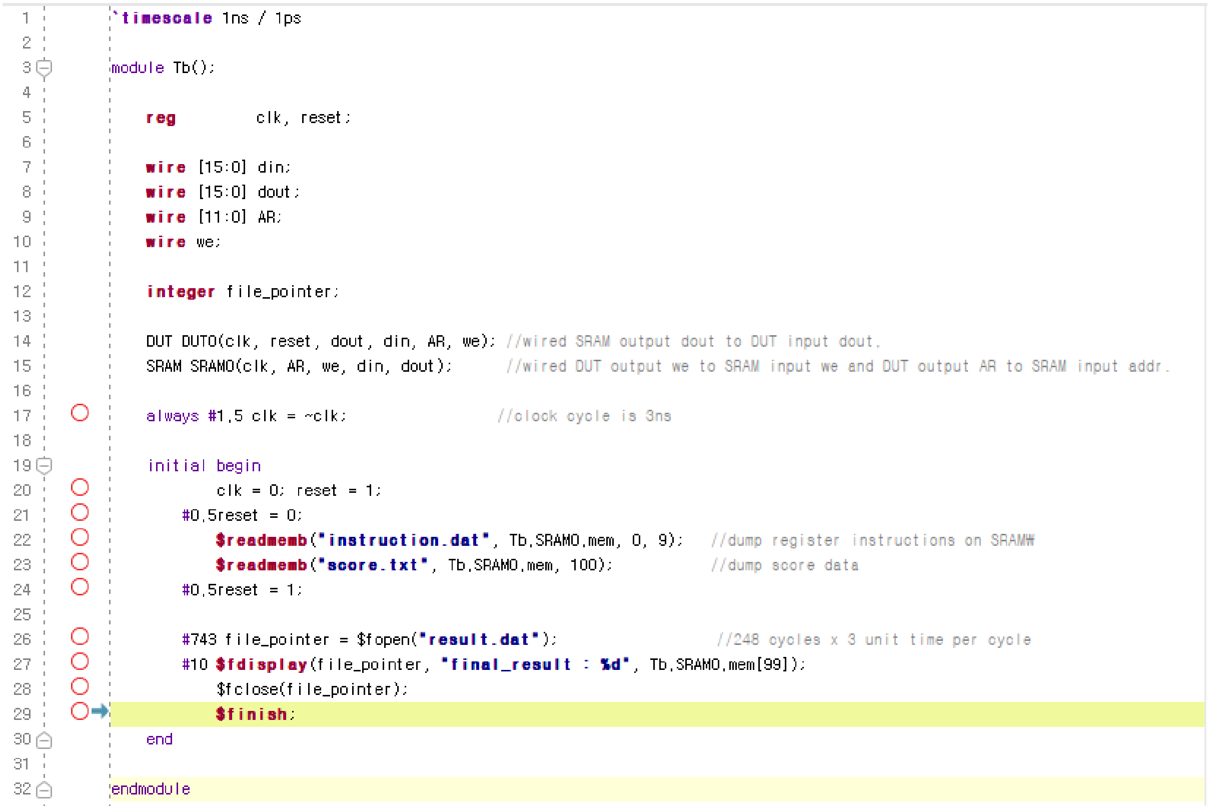
```verilog
            else if (sc[2]) begin
                AR <= IR[11:0];
                sc <= T3;                       //to next sequence
            end


            else if (sc[3]) begin
                if (~IR[15] & IR[14] & IR[13] & IR[12]) begin       //register-reference instruction
                    if(IR[11])                                      //CLA   Clear AC
                        AC <= 0;
                    else if (IR[10])                                //CLE   Clear E
                        E <= 0;
                    else if (IR[9])                                 //CMA   Complement A
                        AC <= ~AC;
                    else if (IR[8])                                 //LDC   Load to AC
                        AC <= IR[7:0];
                    else if (IR[7] & ~IR[8]) begin                  //CIR   Circulate Right
                        AC <= AC >> 1;
                        AC[15] <= E;
                        E <= AC[0];
                    end
                    else if (IR[6] & ~IR[8]) begin                  //CIL   Circulate Left
                        AC <= AC << 1;
                        AC[0] <= E;
                        E <= AC[15];
                    end
                    else if (IR[5] & ~IR[8])                        //INC   Increment AC
                        AC <= AC + 1'b1;
                    else if (IR[4] & ~IR[8]) begin                  //SPA   Skip next instruction if AC positive
                        if((~AC[15]))
                            PC <= PC + 1;
                    end
                    else if (IR[3] & ~IR[8]) begin                  //SNA   Skip next instruction if AC negative
                        if(AC[15])
                            PC <= PC + 1;
                    end
                    else if (IR[2] & ~IR[8]) begin                  //SZA   Skip next instruction if AC Zero
                        if(AC == 0)
                            PC <= PC + 1;
                    end
                    else if (IR[1] & ~IR[8]) begin                  //SZE   Skip next instruction if E is 0
                        if(E == 0)
                            PC <= PC + 1;
```



```
                    end
                    else if (IR[0] & ~IR[8])                    //MOV   Copy AC data to DR
                        DR <= AC;

                    sc <= T0;                                   //Clear SC
                end

                else if (~IR[15] & ~(IR[14]&IR[13]&IR[12])) begin    //Dirsct memory address
                    sc <= T4;
                end

                else if (IR[15] & ~(IR[14]&IR[13]&IR[12])) begin     //Indirect memory address
                    AR <= dout;
                    sc <= T4;
                end
            end


            else if (sc[4]) begin
                if ((~IR[14] & ~IR[13]) | (IR[13] & ~IR[12])) begin //AND, ADD, LDA, ISZ
                    DR <= dout;
                    sc <= T5;
                end

                else if (~IR[14] & IR[13] &  IR[12]) begin          //STA
                    we <= 0;
                    din <= AC;
                    sc <= T0;
                end

                else if (IR[14] & ~IR[13] & ~IR[12]) begin          //BUN
                    PC <= AR;
                    sc <= T0;
                end

                else if (IR[14] & ~IR[13] & IR[12]) begin           //BSA
                    we <= 0;
                    din <= PC;
                    sc <= T5;
                end
            end


            else if (sc[5]) begin
                if (~IR[14] & ~IR[13] & ~IR[12]) begin              //AND
                    AC <= AC & DR;
                    sc <= T0;
                end

                else if (~IR[14] & ~IR[13] & IR[12]) begin          //ADD
                    AC <= AC + DR;
                    if((AC[15] & DR[15]) | ((AC[15] ^ DR[15]) & ((AC + DR) > 16'h7FFF))) E <= 1;
                    else E <= 0;
                    sc <= T0;
                end

                else if (~IR[14] & IR[13] & ~IR[12]) begin          //LDA
                    AC <= DR;
                    sc <= T0;
                end

                else if (IR[14] & ~IR[13] & IR[12]) begin           //BSA
                    PC <= AR;
                    sc <= T0;
                end

                else if (IR[14] & IR[13] & ~IR[12]) begin           //ISZ
                    DR <= DR + 1;
                    sc <= T6;
                end
            end


            else if (sc[6]) begin                                   //ISZ
                we <= 0;
                din <= DR;
                if(DR == 0)
                    PC <= PC + 1;
                sc <= T0;
            end
        end
    end
endmodule
```

HW#7_2017112002_손보성

2. Testbench



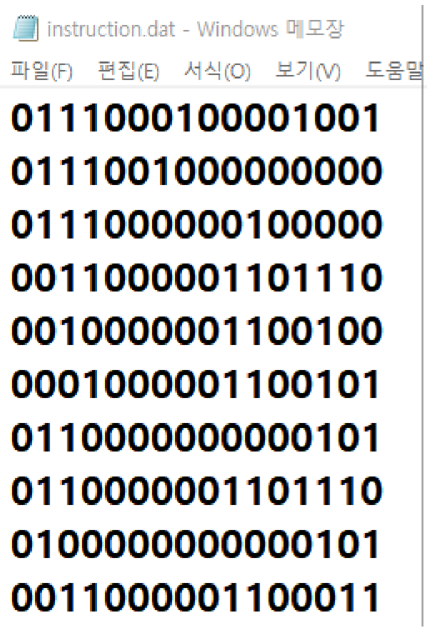
```
`timescale 1ns / 1ps

module Tb();

    reg         clk, reset;

    wire [15:0] din;
    wire [15:0] dout;
    wire [11:0] AR;
    wire we;

    integer file_pointer;

    DUT DUT0(clk, reset, dout, din, AR, we); //wired SRAM output dout to DUT input dout.
    SRAM SRAM0(clk, AR, we, din, dout);      //wired DUT output we to SRAM input we and DUT output AR to SRAM input addr.

    always #1,5 clk = ~clk;                 //clock cycle is 3ns

    initial begin
            clk = 0; reset = 1;
        #0,5reset = 0;
            $readmemb("instruction.dat", Tb.SRAM0.mem, 0, 9);   //dump register instructions on SRAM₩
            $readmemb("score.txt", Tb.SRAM0.mem, 100);          //dump score data
        #0,5reset = 1;

        #743 file_pointer = $fopen("result.dat");               //248 cycles x 3 unit time per cycle
        #10 $fdisplay(file_pointer, "final_result : %d", Tb.SRAM0.mem[99]);
            $fclose(file_pointer);
            $finish;
    end

endmodule
```

3. Total cycles

- 4(LDC) + 4(CMA) + 4(INC) + 5(STA) + 6(LDA) + (6(ADD) + 7(ISZ) + 7(ISZ) +5(BUN, STA at the end of the loop)) x 9 times loop

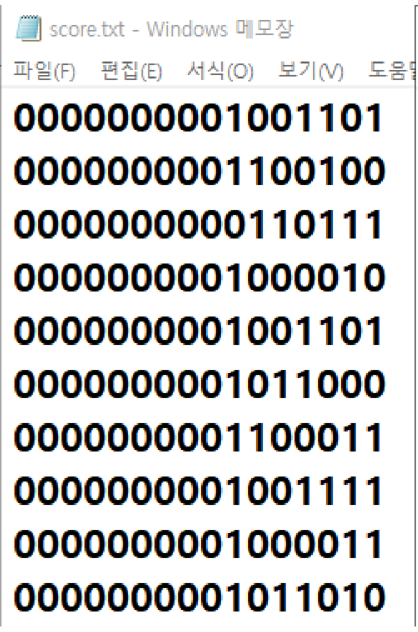
  = 4 + 4 + 4 + 5 + 6 + (6 + 7 + 7 + 5) x 9 = 248 cycles

4. Data and text files(both input and output)



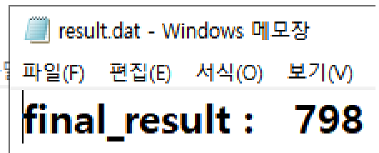
instruction.dat - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말

```
0111000100001001
0111001000000000
0111000000100000
0011000001101110
0010000001100100
0001000001100101
0110000000000101
0110000001101110
0100000000000101
0011000001100011
```



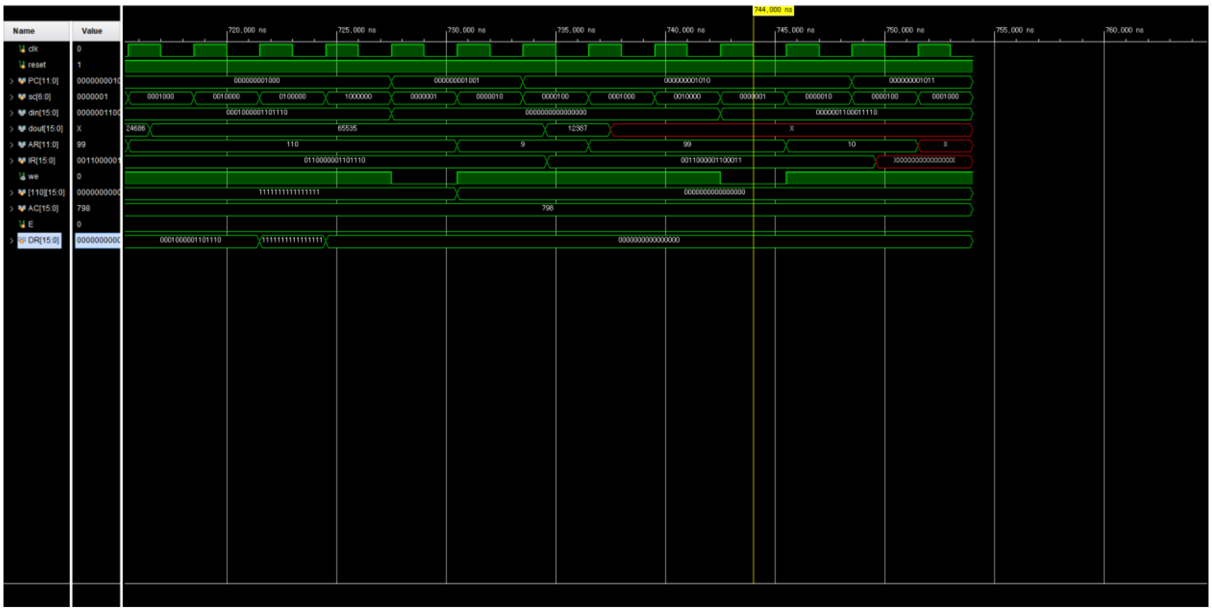
score.txt - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움

```
0000000001001101
0000000001100100
0000000000110111
0000000001000010
0000000001001101
0000000001011000
0000000001100011
0000000001001111
0000000001000011
0000000001011010
```



```
final_result :    798
```

5. Waveform



Last instruction(STA 99) finishes at 742.5ns. It means that the program execution is done within 744ns.