7.2 Verilog Code – Vivado 2018.2

7.2.1 Processor – Top Module

```
module PROCESSOR (
   input MAIN_CLOCK,
   output [15:0] CURRENTADDRESS, REG_AC, REG_1, REG_2,
   output [7:0] INSTRUCTIONADDRESS, CURRENTINSTRUCTION,OUTPUT_FROM_RAM,
   output [5:0] CMD,
   output PROCESS_DONE,
   input wire [15:0] ex_address,
   output wire [7:0] ex_dataout
);
   wire [7:0] DATA_FROM_RAM ;
   wire RECEIVED_8_BITS, TRANSMITTED_8_BITS, PROCESS_FINISHED_FLAG,CPU_CLOCK;
   wire CPU_WRITE_EN, RAM_CLOCK, WRITE_TO_RAM,START_PROCESSING, START_TRANSMISSION;
   wire [7:0] CPU_DATA, RAM_DATA, INSTRUCTION_ADDRESS,INSTRUCTION_DATA;
   wire [15:0] CPU_ADDRESS, RAM_ADDRESS;
   wire T,Z_Flag;
   assign CURRENTADDRESS = CPU_ADDRESS;
   assign INSTRUCTIONADDRESS = INSTRUCTION_ADDRESS;
   assign CURRENTINSTRUCTION = INSTRUCTION_DATA;
   assign OUTPUT_FROM_RAM = DATA_FROM_RAM;
   assign PROCESS_DONE = ~PROCESS_FINISHED_FLAG;
CPU Processor(
   .MAIN_CLOCK(MAIN_CLOCK),
   .PROCESS_FINISHED(PROCESS_FINISHED_FLAG),
   .DATA_FROM_RAM(DATA_FROM_RAM),
   .INSTRUCTION(INSTRUCTION_DATA),
   .CPU_CLOCK(CPU_CLOCK),
   .CPU_WRITE_EN(CPU_WRITE_EN),
   .CPU_ADDRESS(CPU_ADDRESS),
   .CPU_DATA(CPU_DATA),
   .REG_AC(REG_AC),
   .REG_1(REG_1),
   .REG_2(REG_2),
   .CMD(CMD),
   .Z_Flag(Z_Flag),
   .INSTRUCTION_ADDRESS(INSTRUCTION_ADDRESS)
);
INSTRUCTION_ROM IROM (
   .clock(CPU_CLOCK),
   .address(INSTRUCTION_ADDRESS),
   .q(INSTRUCTION_DATA)
);
IMAGE_RAM ImageRam (
   .address(CPU_ADDRESS),
   .clock(CPU_CLOCK),
   .data(CPU_DATA),
   .wren(CPU_WRITE_EN),
   .ex_address(ex_address),
   .ex_dataout(ex_dataout),
   .dataout(DATA_FROM_RAM)
);
endmodule
```

7.2.2 CPU

```
module CPU
(
    input MAIN_CLOCK,
    input [7:0] DATA_FROM_RAM, INSTRUCTION,
    output wire status,
    output PROCESS_FINISHED, CPU_CLOCK,
    output CPU_WRITE_EN, Z_Flag,
    output [15:0] CPU_ADDRESS, REG_AC, REG_1, REG_2,
    output [7:0] CPU_DATA, INSTRUCTION_ADDRESS,
    output [5:0] CMD
);
    wire INTERNAL_CLOCK, FLAG_Z, FETCH;
    wire [2:0] ALU_OP;
    wire [3:0] REG_IN_B_BUS;
    wire [7:0] IR;
    wire [15:0] B_BUS, PC, R1, R2, TR, R, AC, ALU_OUT;
    wire [12:0] SELECTORS;
    wire SIGNAL_TO_FINISH_PROCESS;

    assign CPU_CLOCK = INTERNAL_CLOCK;
    assign CPU_WRITE_EN = SELECTORS[0];
    assign CPU_DATA = B_BUS[7:0];
    assign INSTRUCTION_ADDRESS = PC[7:0];
    assign PROCESS_FINISHED = SIGNAL_TO_FINISH_PROCESS;
    assign REG_AC = AC;
    assign REG_1 = R1;
    assign REG_2 = R2;

    CPU_CLOCK_GENERATOR Clock (
      .MAIN_CLOCK(MAIN_CLOCK),
      .PROCESS_FINISHED(SIGNAL_TO_FINISH_PROCESS),
      .TICK(INTERNAL_CLOCK)
    );
    MUX_FOR_BUS_B Muxer (
      .SELECT(REG_IN_B_BUS),
      .PC(PC),
      .R1(R1),
      .R2(R2),
      .TR(TR),
      .R(R),
      .AC(AC),
      .AR(CPU_ADDRESS),
      .INSTRUCTIONS(INSTRUCTION),
      .DATA_FROM_RAM(DATA_FROM_RAM),
      .BUS(B_BUS)
    );
    CONTROL_UNIT CUnit (
        .CLOCK(INTERNAL_CLOCK),
        .FLAG_Z(FLAG_Z),
        .INSTRUCTION(IR),
        .FETCH(FETCH),
        .FINISH(SIGNAL_TO_FINISH_PROCESS),
        .REG_IN_B_BUS(REG_IN_B_BUS),
        .ALU_OP(ALU_OP),
        .CMD(CMD),
        .SELECTORS(SELECTORS),
        .status(status)
    );
    REGISTER #(8) INSTRUCTION_REGISTER (
        .CLOCK(INTERNAL_CLOCK),
        .LOAD(FETCH),
        .INCREMENT(1'b0),
        .DATA(B_BUS[7:0]),
        .OUT(IR)
    );
    REGISTER #(16) REGISTER_AR (
        .CLOCK(INTERNAL_CLOCK),
        .LOAD(SELECTORS[7]),
        .INCREMENT(SELECTORS[8]),
        .DATA(B_BUS),
        .OUT(CPU_ADDRESS)
    );
    REGISTER #(16) GENERAL_REGISTER (
        .CLOCK(INTERNAL_CLOCK),
        .LOAD(SELECTORS[2]),
        .INCREMENT(1'b0),
        .DATA(B_BUS),
        .OUT(R)
    );
    REGISTER #(16) REGISTER_01 (
        .CLOCK(INTERNAL_CLOCK),
        .LOAD(SELECTORS[5]),
        .INCREMENT(SELECTORS[11]),
        .DATA(B_BUS),
        .OUT(R1)
    );
    REGISTER #(16) REGISTER_02 (
        .CLOCK(INTERNAL_CLOCK),
        .LOAD(SELECTORS[4]),
        .INCREMENT(SELECTORS[10]),
        .DATA(B_BUS),
        .OUT(R2)
    );
    REGISTER #(16) REGISTER_TR (
        .CLOCK(INTERNAL_CLOCK),
        .LOAD(SELECTORS[3]),
        .INCREMENT(SELECTORS[9]),
        .DATA(B_BUS),
        .OUT(TR)
    );
    REGISTER #(16) REGISTER_AC (
        .CLOCK(INTERNAL_CLOCK),
        .LOAD(SELECTORS[1]),
        .INCREMENT(1'b0),
        .DATA(ALU_OUT),
        .OUT(AC)
    );
    REGISTER #(16) PROGRAM_COUNTER (
        .CLOCK(INTERNAL_CLOCK),
        .LOAD(SELECTORS[6]),
        .INCREMENT(SELECTORS[12]),
        .DATA(B_BUS),
        .OUT(PC)
    );
    ALU ALUnit (
        .A_bus(AC),
        .B_bus(B_BUS),
          .ALU_OP(ALU_OP),
          .C_bus(ALU_OUT),
          .FLAG_Z(FLAG_Z)
       );
    endmodule
```

7.2.3 Control Unit (CU)

```
module CONTROL_UNIT
(
   input CLOCK, FLAG_Z,
   input [7:0] INSTRUCTION,
   output reg FETCH, FINISH = 0,
   output [5:0] CMD,
   output reg [3:0] REG_IN_B_BUS,
   output reg [2:0] ALU_OP,
   output reg [12:0] SELECTORS,
   output reg status
);
   localparam
   FETCH1 = 6'd0, FETCH2 = 6'd1, FETCH3 = 6'd2, FETCH4 = 6'd3, CLAC = 6'd6, LDAC = 6'd7, STAC = 6'd8,
   MVACR = 6'd10, MVACR1 = 6'd11, MVACR2 = 6'd12, MVACTR = 6'd13, MVACAR = 6'd14,
   MVR = 6'd15, MVR1 = 6'd16, MVR2 = 6'd17, MVTR = 6'd18, MVAR = 6'd33,
   INCAR = 6'd19, INCR1 = 6'd20, INCR2 = 6'd21, JPNZ = 6'd22, JPNZY = 6'd23, JPNZY1 = 6'd34,
   JPNZN = 6'd24, JPNZN1 = 6'd25, JPNZN2 = 6'd26, ADD = 6'd27,  SUB = 6'd28, MUL4 = 6'd29, DIV2 = 6'd30,
   ADDM = 6'd31, ADDM1 = 6'd35, NOP = 6'd5, END = 6'd32;
   localparam
   DATA_FROM_RAM = 4'd0, PC = 4'd1, R1 = 4'd2, R2 = 4'd3, TR = 4'd4, R = 4'd5, AC = 4'd6,
   INSTRUCTIONS = 4'd7, AR = 4'd8;
   localparam
   ADDAB = 3'd0, SUBAB = 3'd1, PASS = 3'd2, ZER = 3'd3, MUL4A = 3'd5, DIV2A = 3'd6;
   reg [5:0] CONTROL_COMMAND;
   reg [5:0] NEXT_COMMAND = FETCH1;
   always @ (negedge CLOCK) CONTROL_COMMAND <= NEXT_COMMAND;
always @ (CONTROL_COMMAND or FLAG_Z or INSTRUCTION)
  case(CONTROL_COMMAND)
    FETCH1:
      begin
        FETCH <= 0;
        FINISH <= 0;
        REG_IN_B_BUS <= INSTRUCTIONS;
        ALU_OP <= PASS;
        SELECTORS <= 13'b0_0000_0000_0000;
        NEXT_COMMAND <= FETCH2;
      end
    FETCH2:
      begin
        FETCH <= 1;
        FINISH <= 0;
        REG_IN_B_BUS <= INSTRUCTIONS;
        ALU_OP <= PASS;
        SELECTORS <= 13'b0_0000_0000_0000;
        NEXT_COMMAND <= FETCH3;
      end
    FETCH3:
      begin
        FETCH <= 0;
        FINISH <= 0;
        REG_IN_B_BUS <= DATA_FROM_RAM;
        ALU_OP <= PASS;
        SELECTORS <= 13'b1_0000_0000_0000;
        NEXT_COMMAND <= FETCH4;
      end
    FETCH4:
      begin
        FETCH <= 0;
        FINISH <= 0;
        REG_IN_B_BUS <= DATA_FROM_RAM;
        ALU_OP <= PASS;
        SELECTORS <= 13'b0_0000_0000_0000;
        NEXT_COMMAND <= INSTRUCTION[5:0];
    end
CLAC:
    begin
        FETCH <= 0;
        FINISH <= 0;
        REG_IN_B_BUS <= DATA_FROM_RAM;
        ALU_OP <= ZER;
        SELECTORS <= 13'b0_0000_0000_0010;
        NEXT_COMMAND <= FETCH1;
    end
LDAC:
    begin
        FETCH <= 0;
        FINISH <= 0;
        REG_IN_B_BUS <= DATA_FROM_RAM;
        ALU_OP <= PASS;
        SELECTORS <= 13'b0_0000_0000_0010;
        NEXT_COMMAND <= FETCH1;
    end
STAC:
    begin
        FETCH <= 0;
        FINISH <= 0;
        REG_IN_B_BUS <= AC;
        ALU_OP <= ADDAB;
        SELECTORS <= 13'b0_0000_0000_0001;
        NEXT_COMMAND <= FETCH1;
    end
MVACR:
    begin
        FETCH <= 0;
        FINISH <= 0;
        REG_IN_B_BUS <=3'd6;
        ALU_OP <= ADDAB;
        SELECTORS <= 13'b0_0000_0000_0100;
        NEXT_COMMAND <= FETCH1;
    end
MVACR1:
    begin
        FETCH <= 0;
        FINISH <= 0;
        REG_IN_B_BUS <= AC;
        ALU_OP <= ADDAB;
        SELECTORS <= 13'b0_0000_0010_0000;
        NEXT_COMMAND <= FETCH1;
    end
MVACR2:
    begin
        FETCH <= 0;
        FINISH <= 0;
        REG_IN_B_BUS <= AC;
        ALU_OP <= ADDAB;
        SELECTORS <= 13'b0_0000_0001_0000;
        NEXT_COMMAND <= FETCH1;
    end
MVACTR:
    begin
        FETCH <= 0;
        FINISH <= 0;
        REG_IN_B_BUS <= AC;
        ALU_OP <= ADDAB;
        SELECTORS <= 13'b0_0000_0000_1000;
        NEXT_COMMAND <= FETCH1;
    end
MVACAR:
    begin
        FETCH <= 0;
        FINISH <= 0;
```

```
      REG_IN_B_BUS <= AC;
      ALU_OP <= ADDAB;
      SELECTORS <= 13'b0_0000_1000_0000;
      NEXT_COMMAND <= FETCH1;
    end
  MVR:
    begin
      FETCH <= 0;
      FINISH <= 0;
      REG_IN_B_BUS <= R;
      ALU_OP <= PASS;
      SELECTORS <= 13'b0_0000_0000_0010;
      NEXT_COMMAND <= FETCH1;
    end
  MVR1:
    begin
      FETCH <= 0;
      FINISH <= 0;
      REG_IN_B_BUS <= R1;
      ALU_OP <= PASS;
      SELECTORS <= 13'b0_0000_0000_0010;
      NEXT_COMMAND <= FETCH1;
    end
  MVR2:
    begin
      FETCH <= 0;
      FINISH <= 0;
      REG_IN_B_BUS <= R2;
      ALU_OP <= PASS;
      SELECTORS <= 13'b0_0000_0000_0010;
      NEXT_COMMAND <= FETCH1;
    end
  MVTR:
    begin
      FETCH <= 0;
```

```
        FINISH <= 0;
        REG_IN_B_BUS <= TR;
        ALU_OP <= PASS;
        SELECTORS <= 13'b0_0000_0000_0010;
        NEXT_COMMAND <= FETCH1;
    end
MVAR:
    begin
        FETCH <= 0;
        FINISH <= 0;
        REG_IN_B_BUS <= AR;
        ALU_OP <= PASS;
        SELECTORS <= 13'b0_0000_0000_0010;
        NEXT_COMMAND <= FETCH1;
    end
INCAR:
    begin
        FETCH <= 0;
        FINISH <= 0;
        REG_IN_B_BUS <= DATA_FROM_RAM;
        ALU_OP <= ADDAB;
        SELECTORS <= 13'b0_0001_0000_0000;
        NEXT_COMMAND <= FETCH1;
    end
INCR1:
    begin
        FETCH <= 0;
        FINISH <= 0;
        REG_IN_B_BUS <= DATA_FROM_RAM;
        ALU_OP <= ADDAB;
        SELECTORS <= 13'b0_1000_0000_0000;
        NEXT_COMMAND <= FETCH1;
    end
INCR2:
    begin
```

```
        FETCH <= 0;
        FINISH <= 0;
        REG_IN_B_BUS <= DATA_FROM_RAM;
        ALU_OP <= ADDAB;
        SELECTORS <= 13'b0_0100_0000_0000;
        NEXT_COMMAND <= FETCH1;
    end
ADD:
    begin
        FETCH <= 0;
        FINISH <= 0;
        REG_IN_B_BUS <= R;
        ALU_OP <= ADDAB;
        SELECTORS <= 13'b0_0000_0000_0010;
        NEXT_COMMAND <= FETCH1;
    end
SUB:
    begin
        FETCH <= 0;
        FINISH <= 0;
        REG_IN_B_BUS <= R;
        ALU_OP <= SUBAB;
        SELECTORS <= 13'b0_0000_0000_0010;
        NEXT_COMMAND <= FETCH1;
    end
MUL4:
    begin
        FETCH <= 0;
        FINISH <= 0;
        REG_IN_B_BUS <= DATA_FROM_RAM;
        ALU_OP <= MUL4A;
        SELECTORS <= 13'b0_0000_0000_0010;
        NEXT_COMMAND <= FETCH1;
    end
DIV2:
    begin
        FETCH <= 0;
        FINISH <= 0;
        REG_IN_B_BUS <= DATA_FROM_RAM;
        ALU_OP <= DIV2A;
        SELECTORS <= 13'b0_0000_0000_0010;
        NEXT_COMMAND <= FETCH1;
    end
ADDM:
    begin
        FETCH <= 0;
        FINISH <= 0;
        REG_IN_B_BUS <= INSTRUCTIONS;
        ALU_OP <= ADDAB;
        SELECTORS <= 13'b1_0000_0000_0000;
        NEXT_COMMAND <= ADDM1;
    end
ADDM1:
    begin
        FETCH <= 0;
        FINISH <= 0;
        REG_IN_B_BUS <= INSTRUCTIONS;
        ALU_OP <= ADDAB;
        SELECTORS <= 13'b0_0000_0000_0010.
        NEXT_COMMAND <= FETCH1;
    end
JPNZ:
    begin
        FETCH <= 0;
        FINISH <= 0;
        REG_IN_B_BUS <= DATA_FROM_RAM;
        ALU_OP <= PASS;
        SELECTORS <= 13'b0_0000_0000_0000;
        NEXT_COMMAND <= (FLAG_Z) ? JPNZY : JPNZN;
    end
```

```
JPNZY:
  begin
    FETCH <= 0;
    FINISH <= 0;
    REG_IN_B_BUS <= DATA_FROM_RAM;
    ALU_OP <= PASS;
    SELECTORS <= 13'b1_0000_0000_0000;
    NEXT_COMMAND <= JPNZY1;
  end
JPNZY1:
  begin
    FETCH <= 0;
    FINISH <= 0;
    REG_IN_B_BUS <= DATA_FROM_RAM;
    ALU_OP <= PASS;
    SELECTORS <= 13'b0_0000_0000_0000;
    NEXT_COMMAND <= FETCH1;
  end
JPNZN:
  begin
    FETCH <= 0;
    FINISH <= 0;
    REG_IN_B_BUS <= INSTRUCTIONS;
    ALU_OP <= ADDAB;
    SELECTORS <= 13'b0_0000_0000_0000;
    NEXT_COMMAND <= JPNZN1;
  end
JPNZN1:
  begin
    FETCH <= 0;
    FINISH <= 0;
    REG_IN_B_BUS <= INSTRUCTIONS;
    ALU_OP <= PASS;
    SELECTORS <= 13'b0_0000_0100_0000;
    NEXT_COMMAND <= JPNZN2;
```

```
      end
    JPNZN2:
      begin
        FETCH <= 0;
        FINISH <= 0;
        REG_IN_B_BUS <= AC;
        ALU_OP <= ADDAB;
        SELECTORS <= 13'b0_0000_0000_0010;
        NEXT_COMMAND <= FETCH1;
      end
    NOP:
      begin
        FETCH <= 0;
        FINISH <= 0;
        REG_IN_B_BUS <= DATA_FROM_RAM;
        ALU_OP <= ADDAB;
        SELECTORS <= 13'b0_0000_0000_0000;
        NEXT_COMMAND <= FETCH1;
      end
    END:
      begin
        FETCH <= 0;
        FINISH <= 1;
        REG_IN_B_BUS <= DATA_FROM_RAM;
        ALU_OP <= ADDAB;
        SELECTORS <= 13'b0_0000_0000_0000;
        NEXT_COMMAND <= END;
        status <= 1'b1;
      end
  endcase
 assign CMD = CONTROL_COMMAND;
endmodule
```

7.2.4 Arithmetic and Logic Unit (ALU)

```
module ALU
(
   input [15:0] A_bus, B_bus,
   input [2:0] ALU_OP,
   output reg [15:0] C_bus,
   output reg FLAG_Z
);
parameter ADD = 3'd0, SUB = 3'd1, PASS = 3'd2, ZER = 3'd3, MUL4 = 3'd5, DIV2 =3'd6;
always @ (ALU_OP or A_bus or B_bus)
   begin
      case (ALU_OP)
         ADD:
           begin
            C_bus = A_bus + B_bus;
            FLAG_Z = 0;
           end
         SUB:
           begin
            C_bus = A_bus - B_bus;
            FLAG_Z = (C_bus == 16'd0) ? 1'b1 : 1'b0;
           end
         PASS:
            begin
             C_bus = B_bus;
             if (C_bus == 16'd0) FLAG_Z = 1;
            end
         ZER:
            begin
               C_bus = 0;
               FLAG_Z = 0;
            end
         MUL4:
            begin
               C_bus = A_bus << 2;
               FLAG_Z = 0;
            end
          DIV2:
            begin
              C_bus = A_bus >> 1;
              FLAG_Z = 0;
            end
        endcase
    end
endmodule
```

7.2.5 Multiplexer

```
module MUX_FOR_BUS_B
(
    input [3:0] SELECT,
    input [15:0] PC, R1, R2, TR, R, AC, AR,
    input [7:0] INSTRUCTIONS, DATA_FROM_RAM,
    output reg [15:0] BUS
);
    always @ (*) //SELECT or DATA_FROM_RAM or PC or R1 or R2 or TR or R or AC or INSTRUCTIONS or AR
    begin
        case(SELECT)
          4'd0: BUS = {8'b0000_0000, DATA_FROM_RAM};
          4'd1: BUS = PC;
          4'd2: BUS = R1;
          4'd3: BUS = R2;
          4'd4: BUS = TR;
          4'd5: BUS = R;
          4'd6: BUS = AC;
          4'd7: BUS = {8'b0000_0000, INSTRUCTIONS};
          4'd8: BUS = AR;
        endcase
    end
endmodule
```

7.2.6 CPU Clock Generator

```
module CPU_CLOCK_GENERATOR
(
  input MAIN_CLOCK, PROCESS_FINISHED,
  output reg TICK = 1'b1
);
  always @ (posedge MAIN_CLOCK)
    begin
      if (~PROCESS_FINISHED)
        TICK = ~TICK;
      else
        TICK = 1'b0;
    end
endmodule
```

7.2.7 Data Memory (DRAM)

```
module IMAGE_RAM (
  address,
        clock,
        data,
        wren,
        dataout,
        ex_address,
  ex_dataout
  );
        input   [15:0]  address;
        input    clock;
        input   [7:0]  data;
        input    wren=1;
        input [15:0] ex_address;
        output reg [7:0] ex_dataout;
        output reg [7:0]  dataout;
        reg[7:0] MEMORY [65535:0];
  initial begin
    $readmemb("C:\\Users\\hiruna\\CSD \\lena.txt",MEMORY);
  end
  always @ (ex_address)
  begin
    ex_dataout <= MEMORY[ex_address];
  end
  always @ (posedge clock)
  begin
    if(wren == 1)begin
      MEMORY[address] <= data[7:0];
    end
    else begin
      dataout <= MEMORY[address];
    end
  end
endmodule
```

7.2.8 Instruction Memory (IRAM)

```
module INSTRUCTION_ROM(
  input clock,
  input [7:0] address,
  output reg [7:0] q
);
  reg[7:0] memory[0:255];
always @(posedge clock)
        q<=memory[address];
  initial begin
    memory[0]=8'b0000_0110;
    memory[1]=8'b0000_1011;
    memory[2]=8'b0000_1100;
```

```
memory[3]=8'b0001_0001;
memory[4]=8'b0001_1101;
memory[5]=8'b0001_1101;
memory[6]=8'b0001_1101;
memory[7]=8'b0001_1101;
memory[8]=8'b0000_1010;
memory[9]=8'b0001_0000;
memory[10]=8'b0001_1011;
memory[11]=8'b0000_1110;
memory[12]=8'b0000_0111;
memory[13]=8'b0000_1010;
memory[14]=8'b0001_0011;
memory[15]=8'b0000_0111;
memory[16]=8'b0001_1101;
memory[17]=8'b0001_1110;
memory[18]=8'b0001_1011;
memory[19]=8'b0000_1010;
memory[20]=8'b0001_0011;
memory[21]=8'b0000_0111;
memory[22]=8'b0001_1011;
memory[23]=8'b0000_1010;
memory[24]=8'b0010_0001;
memory[25]=8'b0001_1111;
memory[26]=8'b1111_1110;
memory[27]=8'b0000_1110;
memory[28]=8'b0000_0111;
memory[29]=8'b0001_1101;
memory[30]=8'b0001_1110;
memory[31]=8'b0001_1011;
memory[32]=8'b0000_1010;
memory[33]=8'b0001_0011;
memory[34]=8'b0010_0001;
memory[35]=8'b0000_1101;
memory[36]=8'b0000_0111;
memory[37]=8'b0001_1101;
```

```
memory[38]=8'b0001_1011;
memory[39]=8'b0000_1010;
memory[40]=8'b0001_0011;
memory[41]=8'b0000_0111;
memory[42]=8'b0001_1101;
memory[43]=8'b0001_1110;
memory[44]=8'b0001_1011;
memory[45]=8'b0000_1010;
memory[46]=8'b0010_0001;
memory[47]=8'b0001_1111;
memory[48]=8'b1111_1110;
memory[49]=8'b0000_1110;
memory[50]=8'b0000_0111;
memory[51]=8'b0001_1011;
memory[52]=8'b0000_1010;
memory[53]=8'b0001_0011;
memory[54]=8'b0000_0111;
memory[55]=8'b0001_1101;
memory[56]=8'b0001_1110;
memory[57]=8'b0001_1011;
memory[58]=8'b0000_1010;
memory[59]=8'b0001_0011;
memory[60]=8'b0000_0111;
memory[61]=8'b0001_1011;
memory[62]=8'b0001_1110;
memory[63]=8'b0001_1110;
memory[64]=8'b0001_1110;
memory[65]=8'b0001_1110;
memory[66]=8'b0000_1010;
memory[67]=8'b0001_0010;
memory[68]=8'b0000_1110;
memory[69]=8'b0000_1111;
memory[70]=8'b0000_1000;
memory[71]=8'b0001_0100;
memory[72]=8'b0001_0000;
```

```
memory[73]=8'b0000_1010;
memory[74]=8'b0000_0110;
memory[75]=8'b0001_1111;
memory[76]=8'b1111_1110;
memory[77]=8'b0001_1100;
memory[78]=8'b0001_0110;
memory[79]=8'b0000_0011;
memory[80]=8'b0000_1011;
memory[81]=8'b0001_0100;
memory[82]=8'b0001_0001;
memory[83]=8'b0000_1010;
memory[84]=8'b0000_0110;
memory[85]=8'b0001_1111;
memory[86]=8'b1111_1110;
memory[87]=8'b0001_1100;
memory[88]=8'b0001_0110;
memory[89]=8'b0000_0011;
memory[90]=8'b0000_0110;
memory[91]=8'b0000_1011;
memory[92]=8'b0000_1100;
memory[93]=8'b0001_0001;
memory[94]=8'b0001_1101;
memory[95]=8'b0001_1101;
memory[96]=8'b0001_1101;
memory[97]=8'b0001_1101;
memory[98]=8'b0000_1010;
memory[99]=8'b0001_0000;
memory[100]=8'b0001_1011;
memory[101]=8'b0000_1110;
memory[102]=8'b0001_1110;
memory[103]=8'b0000_1101;
memory[104]=8'b0000_0111;
memory[105]=8'b0000_1010;
memory[106]=8'b0001_0011;
memory[107]=8'b0000_0111;
```

```
memory[108]=8'b0001_1011;
memory[109]=8'b0000_1010;
memory[110]=8'b0010_0001;
memory[111]=8'b0001_1111;
memory[112]=8'b1111_1111;
memory[113]=8'b0000_1110;
memory[114]=8'b0000_0111;
memory[115]=8'b0001_1011;
memory[116]=8'b0000_1010;
memory[117]=8'b0001_0011;
memory[118]=8'b0000_0111;
memory[119]=8'b0001_1011;
memory[120]=8'b0001_1110;
memory[121]=8'b0001_1110;
memory[122]=8'b0000_1010;
memory[123]=8'b0001_0010;
memory[124]=8'b0000_1110;
memory[125]=8'b0000_1111;
memory[126]=8'b0000_1000;
memory[127]=8'b0001_0100;
memory[128]=8'b0001_0100;
memory[129]=8'b0001_0000;
memory[130]=8'b0000_1010;
memory[131]=8'b0000_0110;
memory[132]=8'b0001_1111;
memory[133]=8'b1111_1111;
memory[134]=8'b0001_1111;
memory[135]=8'b0000_0001;
memory[136]=8'b0001_1100;
memory[137]=8'b0001_0110;
memory[138]=8'b0101_1101;
memory[139]=8'b0000_1011;
memory[140]=8'b0001_0101;
memory[141]=8'b0001_0101;
memory[142]=8'b0001_0001;
```

```
    memory[143]=8'b0000_1010;
    memory[144]=8'b0000_0110;
    memory[145]=8'b0001_1111;
    memory[146]=8'b1111_1111;
    memory[147]=8'b0001_1111;
    memory[148]=8'b0000_0001;
    memory[149]=8'b0001_1100;
    memory[150]=8'b0001_0110;
    memory[151]=8'b0101_1101;
    memory[152]=8'b0010_0000;
  end
endmodule
```

7.2.9 Processor Test bench

```
module processor_tb;
  reg MAIN_CLOCK;
  wire PROCESS_DONE;
  wire [15:0] CURRENTADDRESS, REG_AC, REG_1, REG_2;
  wire [5:0] CMD;
  wire [7:0] INSTRUCTIONADDRESS, CURRENTINSTRUCTION,OUTPUT_FROM_RAM;
  reg [15:0] ADD_FROM_OUT;
  wire [7:0] DATA_TO_OUT;
  integer file,i;
  PROCESSOR UUT (
    .MAIN_CLOCK(MAIN_CLOCK),
    .CURRENTADDRESS(CURRENTADDRESS),
    .REG_AC(REG_AC),
    .REG_1(REG_1),
    .REG_2(REG_2),
    .OUTPUT_FROM_RAM(OUTPUT_FROM_RAM),
    .INSTRUCTIONADDRESS(INSTRUCTIONADDRESS),
    .CURRENTINSTRUCTION(CURRENTINSTRUCTION),
    .CMD(CMD),
    .PROCESS_DONE(PROCESS_DONE),
     .ex_dataout(DATA_TO_OUT),
     .ex_address(ADD_FROM_OUT)
  );
  always
    #10 MAIN_CLOCK = ~MAIN_CLOCK;
  initial begin
    MAIN_CLOCK = 1'b0;
    #100000000
    file=$fopen("C:\\Users\\hiruna\\CSD\\output.txt","w");
    for (i=2; i< 65539; i=i+4)
       begin
          @(posedge MAIN_CLOCK)
          ADD_FROM_OUT = i;
          $fwrite(file,"%b",DATA_TO_OUT);
          $fwrite(file,"\n");
       end
   $fclose(file);
   $finish;
end
endmodule
```