第一題

|  |
| --- |
| ALU的程式碼 |
| //Subject: CA - Decoder  //--------------------------------------------------------------------------------  module Decoder(  instr\_op\_i,  RegWrite\_o,  ALU\_op\_o,  ALUSrc\_o,  RegDst\_o,  Branch\_o  );    //I/O ports  input [6-1:0] instr\_op\_i;  output RegWrite\_o;  output [3-1:0] ALU\_op\_o;  output ALUSrc\_o;  output RegDst\_o;  output Branch\_o;    //Internal Signals  reg [3-1:0] ALU\_op\_o;  reg ALUSrc\_o;  reg RegWrite\_o;  reg RegDst\_o;  reg Branch\_o;  //Parameter  //Main function  always @(instr\_op\_i)  begin  RegWrite\_o =0;  ALU\_op\_o =0;  ALUSrc\_o =0;  RegDst\_o =0;  Branch\_o =0;  case(instr\_op\_i)  6’b000000 :  begin  RegDst\_o=1 ;  RegWrite\_o=1 ;  Branch\_o=0 ;  ALUSrc\_o=0 ;  ALU\_op\_o=6’b000000;  end  6’b001000 :  begin  RegDst\_o=0 ;  RegWrite\_o=1 ;  Branch\_o=0 ;  ALUSrc\_o=1 ;  ALU\_op\_o=6’b001000;  end  6’b000100 :  begin  RegDst\_o=0 ;  RegWrite\_o=1 ;  Branch\_o=0 ;  ALUSrc\_o=1 ;  ALU\_op\_o=6’b000100;  end  6’b001101 :  begin  RegDst\_o=0 ;  RegWrite\_o=0 ;  Branch\_o=1 ;  ALUSrc\_o=0 ;  ALU\_op\_o=6’b001101;  end  default: :  begin  RegDst\_o=0 ;  RegWrite\_o=0 ;  Branch\_o=0 ;  ALUSrc\_o=0 ;  ALU\_op\_o=6’b000000;  end  endcase  endmodule |
| RTL Analysis / Schematic圖 |
|  |
| 波型圖 |
|  |
| 模擬結果 |
|  |

第二題

|  |
| --- |
| Single cycle cpu的程式碼 |
| `timescale 1ns / 1ps  //////////////////////////////////////////////////////////////////////////////////  // Company:  // Engineer:  //  // Create Date: 23:46:27 12/02/2017  // Design Name:  // Module Name: Simple\_Single\_CPU  // Project Name:  // Target Devices:  // Tool versions:  // Description:  //  // Dependencies:  //  // Revision:  // Revision 0.01 - File Created  // Additional Comments:  //  //////////////////////////////////////////////////////////////////////////////////  module Simple\_Single\_CPU(  clk\_i,  rst\_n  );    //I/O port  input clk\_i;  input rst\_n;  //Internal Signles  wire [32-1:0] pc\_out\_o;  wire [32-1:0] instr\_add\_4\_o;  wire [32-1:0] instr\_o;  wire [32-1:0] RSdata\_o;  wire [32-1:0] RTdata\_o;  wire [32-1:0] SE\_o;  wire [32-1:0] ALU\_src2\_o;  wire [32-1:0] Shifter\_o;  wire [32-1:0] instr\_branch\_o;  wire [32-1:0] instr\_next\_o;  wire [32-1:0] ALU\_result\_o;  wire [5-1:0] RDaddr\_o;  wire [4-1:0] ALU\_ctrl\_o;  wire [3-1:0] ALU\_op\_o;  wire ALUSrc\_o;  wire RegDst\_o;  wire RegWrite\_o;  wire Branch\_o;  wire ALU\_zero\_o;  wire sel\_branch\_o;  wire ALU\_cout\_o;  wire ALU\_overflow\_o;    //Greate componentes  ProgramCounter PC(  .clk\_i(clk\_i),  .rst\_n (rst\_n),  .pc\_in\_i(instr\_next\_o),  .pc\_out\_o(pc\_out\_o)  );    Adder Adder1(  .src1\_i(pc\_out\_o),  .src2\_i(32'd4),  .sum\_o(instr\_add\_4\_o)  );    Instr\_Memory IM(  .pc\_addr\_i(pc\_out\_o),  .instr\_o(instr\_o)  );  MUX\_2to1 #(.size(5)) Mux\_Write\_Reg(  .data0\_i(instr\_o[20:16]),  .data1\_i(instr\_o[15:11]),  .select\_i(RegDst\_o),  .data\_o(RDaddr\_o)  );    Reg\_File RF(  .clk\_i(clk\_i),  .rst\_n(rst\_n),  .RSaddr\_i(instr\_o[25:21]),  .RTaddr\_i(instr\_o[20:16]),  .RDaddr\_i(RDaddr\_o),  .RDdata\_i(ALU\_result\_o),  .RegWrite\_i(RegWrite\_o),  .RSdata\_o(RSdata\_o),  .RTdata\_o(RTdata\_o)  );    Decoder Decoder(  .instr\_op\_i(instr\_o[31:26]),  .RegWrite\_o(RegWrite\_o),  .ALU\_op\_o(ALU\_op\_o),  .ALUSrc\_o(ALUSrc\_o),  .RegDst\_o(RegDst\_o),  .Branch\_o(Branch\_o)  );  ALU\_Ctrl AC(  .funct\_i(instr\_o[5:0]),  .ALUOp\_i(ALU\_op\_o),  .ALUCtrl\_o(ALU\_ctrl\_o)  );    Sign\_Extend SE(  .data\_i(instr\_o[15:0]),  .data\_o(SE\_o)  );  MUX\_2to1 #(.size(32)) Mux\_ALUSrc(  .data0\_i(RTdata\_o),  .data1\_i(SE\_o),  .select\_i(ALUSrc\_o),  .data\_o(ALU\_src2\_o)  );    alu ALU(  .src1(RSdata\_o),  .src2(ALU\_src2\_o),  .ALU\_control(ALU\_ctrl\_o),  .result(ALU\_result\_o),  .zero(ALU\_zero\_o),  .cout(ALU\_cout\_o),  .overflow(ALU\_ovflow\_o)  );  Adder Adder2(  .src1\_i(instr\_add\_4\_o),  .src2\_i(Shifter\_o),  .sum\_o(instr\_branch\_o)  );    Shift\_Left\_Two\_32 Shifter(  .data\_i(SE\_o),  .data\_o(Shifter\_o)  );  and(sel\_branch\_o ,ALU\_zero\_o, Branch\_o);  MUX\_2to1 #(.size(32)) Mux\_PC\_Source(  .data0\_i(instr\_add\_4\_o),  .data1\_i(instr\_branch\_o),  .select\_i(sel\_branch\_o),  .data\_o(instr\_next\_o)  );  endmodule |
| RTL Analysis / Schematic圖 |
|  |
| 波型圖 |
|  |
| 模擬結果 |
|  |