

گزارش کار آزمایشگاه DSD

**آزمایش شماره 10**



11 خرداد 1400

عرشیا اخوان

محمدرضا عبدی

علیرضا ایلامی

|  |  |  |
| --- | --- | --- |
| تاریخ آزمایش: 11 خرداد 1400 | موضوع: پیاده‌سازی یک پردازنده ساده | **شماره آزمایش: 10** |
| **علیرضا ایلامی**  **97101286** | **محمدرضا عبدی**  **97110285** | **عرشیا اخوان**  **97110422** |

مقدمه:

یک پردازنده با معماری پشته ای دارای یک پشته با 8 ثبات 8 بیتی است.این پردازنده دارای 8 دستور در مجموعه دستورالعمل های خود است. همچنین دارای حافظه به اندازه 256 خانه 8 بیتی است که 8 خانه آخر آن، به صورت Memory Mapped I/O هستند.

شرح آزمایش:

کلا دو تا استیج داریم: IFIDC و EXEC

در استیج IFIDC، دستور را Fetch میکنیم، آن را Decode میکنیم و در نهایت آن را کنترل میکنیم و به ماژول Exec میدهیم تا کار آن دستور را انجام دهد.

در استیج EXEC، به فراخور نوع دستور، از ماژولهای مختلفی استفاده میکنیم که در زیر توضیح داده شده اند:

برای دستورهای ADD, SUB از ماژول ALU استفاده میکنیم.

برای دستورهای JUMP, JUMPZ, JUMPS از ماژول PC استفاده میکنیم.

برای دستورهای PUSHC, PUSHM, POPM از ماژول PP استفاده میکنیم.

ابتدا استیج IFIDC:

|  |
| --- |
| `define INIT 2'b00 `define IF 2'b01 `define ID 2'b10 `define OP\_STL 4'b1000  module IFIDC (clk,  en,  rstn,  pc,  IS\_ready,  control\_bus,  data);    parameter INST\_CAP = 20;  parameter INST\_LEN = 12;  parameter DATA\_LEN = 8;    input clk, rstn, en;  input [$clog2(INST\_CAP):0] pc;  output reg [3:0]control\_bus;  output reg [DATA\_LEN-1:0]data;  output reg IS\_ready;    reg [INST\_LEN-1:0] inst\_mem [INST\_CAP-1:0];  reg[INST\_LEN-1:0] inst;    reg [1:0]state;    always @(posedge clk or negedge rstn) begin  if (!rstn && !en) begin  control\_bus <= `OP\_STL;  data <= 8'bx;  state <= `INIT;  IS\_ready <= 0;  end  else begin  case(state)  `INIT:  begin  if (en) begin  state <= `IF;  end  IS\_ready <= 0;  end  `IF:  begin  inst <= inst\_mem[pc];  state <= `ID;  end  `ID:  begin  control\_bus[3:0] <= inst[INST\_LEN-1:DATA\_LEN];  data <= inst[DATA\_LEN-1:0];  state <= `INIT;  IS\_ready <= 1;  end  endcase  end    end    //debugging  always @(\*)  $display($time, "\t [IFIDC::%d] pc = %d control\_bus = %b addr\_imm = %d IS\_ready = %b rstn = %b en = %b", state, pc, control\_bus, data,IS\_ready,rstn, en);   endmodule |

ابتدا در case: INIT هستیم. وقتی enable = 1 شد وارد فاز IF: Instruction Fetch میشویم. و دستور را از حافظه میخوانیم و اصطلاحا fetch میکنیم.

مرحله بعدی ID: Instruction Decode است. که در آن بخشهای مختلف دستور را جدا میکنیم.

در پایان مجددا به فاز INIT برمیگردیم و سیگنال is\_ready = 1 میکنیم. (برای اینکه ماژول های داخلی EXEC بدانند کار استیج قبلی برای یک دستور تمام شده و می توانند آن را اجرا نمایند.)

استیج EXEC:

|  |
| --- |
| `define NXTI 2'b00 `define EXIT 2'b11  module EXEC (clk,  en,  rstn,  pc,  control\_bus,  addr\_const,  stk\_data\_in,  stk\_push,  stk\_pop,  stk\_data\_out,  mem\_data\_in,  mem\_addr,  mem\_r\_en,  mem\_w\_en,  mem\_data\_out,  fin\_sig);    parameter DATA\_LEN = 8;  parameter ADDR\_LEN = 8;  parameter INST\_CAP = 20;    input clk, en, rstn;  input [3:0] control\_bus;  input [DATA\_LEN-1:0] stk\_data\_out, mem\_data\_out, addr\_const;  output wire [DATA\_LEN-1:0] stk\_data\_in, mem\_data\_in;  output wire [ADDR\_LEN-1:0] mem\_addr;  output wire stk\_push, stk\_pop, mem\_r\_en, mem\_w\_en, fin\_sig;  output wire [$clog2(INST\_CAP):0] pc;  wire alu\_z\_flag, alu\_s\_flag, alu\_fin\_sig, wbpb\_fin\_sig, pc\_fin\_sig;    assign fin\_sig = (alu\_fin\_sig | wbpb\_fin\_sig | pc\_fin\_sig);    ALU #(  .DATA\_LEN(DATA\_LEN)  ) alu0 (  .clk(clk),  .rstn(rstn),  .en(en),  .control\_bus(control\_bus),  .z\_flag(alu\_z\_flag),  .s\_flag(alu\_s\_flag),  .stk\_data\_in(stk\_data\_in),  .stk\_push(stk\_push),  .stk\_pop(stk\_pop),  .stk\_data\_out(stk\_data\_out),  .fin\_sig(alu\_fin\_sig)  );    PP #(  .DATA\_LEN(DATA\_LEN),  .ADDR\_LEN(ADDR\_LEN)  ) pp0 (  .clk(clk),  .rstn(rstn),  .en(en),  .control\_bus(control\_bus),  .addr\_const(addr\_const),  .stk\_data\_in(stk\_data\_in),  .stk\_push(stk\_push),  .stk\_pop(stk\_pop),  .stk\_data\_out(stk\_data\_out),  .mem\_data\_in(mem\_data\_in),  .mem\_addr(mem\_addr),  .mem\_r\_en(mem\_r\_en),  .mem\_w\_en(mem\_w\_en),  .mem\_data\_out(mem\_data\_out),  .fin\_sig(wbpb\_fin\_sig)  );    PC #(  .DATA\_LEN(DATA\_LEN),  .INST\_CAP(INST\_CAP)  ) pc0 (  .clk(clk),  .rstn(rstn),  .en(en),  .control\_bus(control\_bus),  .pc(pc),  .z\_flag(alu\_z\_flag),  .s\_flag(alu\_s\_flag),  .stk\_pop(stk\_pop),  .stk\_push(stk\_push),  .stk\_data\_out(stk\_data\_out),  .fin\_sig(pc\_fin\_sig)  );   endmodule |

در ماژول EXEC صرفا سیم کشی ها انجام میشود و از سه submodule آن، instantiate میکنیم.

حال به سراغ ماژول های داخلی استیج EXEC میرویم:

ماژول ALU:

|  |
| --- |
| `define INIT 4'b0000 `define OP1\_POP 4'b0001 `define OP1\_RCV 4'b0010 `define OP1\_STR 4'b0011 `define OP2\_POP 4'b0100 `define OP2\_RCV 4'b0101 `define OP2\_STR 4'b0110 `define ALU 4'b0111 `define PUSH 4'b1000 `define PUSH\_W 4'b1001  module ALU (control\_bus,  clk,  en,  rstn,  z\_flag,  s\_flag,  stk\_data\_in,  stk\_push,  stk\_pop,  stk\_data\_out,  fin\_sig);    parameter DATA\_LEN = 8;    input clk, rstn, en;  input [3:0] control\_bus;  input [DATA\_LEN-1:0] stk\_data\_out;  output reg [DATA\_LEN-1:0] stk\_data\_in;  output reg stk\_push, stk\_pop;  output reg z\_flag, s\_flag, fin\_sig;    wire asn = !control\_bus[0];  wire alu\_en = (en && control\_bus[1] && control\_bus[2]);    reg [3:0]state;    reg signed [DATA\_LEN-1:0] op1, op2, result;    always @(posedge clk, negedge rstn) begin  if (!rstn && !alu\_en) begin  state <= `INIT;  stk\_data\_in <= {DATA\_LEN{1'bz}};  stk\_push <= 1'bz;  stk\_pop <= 1'bz;  s\_flag <= 0;  z\_flag <= 0;  fin\_sig <= 0;  end  else begin  case(state)  `INIT:  begin  if (alu\_en) begin  state <= `OP1\_POP;  end  fin\_sig <= 0;  stk\_data\_in <= {DATA\_LEN{1'bz}};  stk\_push <= 1'bz;  stk\_pop <= 1'bz;  end  `OP1\_POP:  begin  stk\_push<= 0;  stk\_pop <= 1;  state <= `OP1\_RCV;  end  `OP1\_RCV:  begin  stk\_pop <= 0;  state <= `OP1\_STR;  end  `OP1\_STR:  begin  op1 <= stk\_data\_out;  state <= `OP2\_POP;  end  `OP2\_POP:  begin  stk\_pop <= 1;  state <= `OP2\_RCV;  end  `OP2\_RCV:  begin  stk\_pop <= 0;  state <= `OP2\_STR;  end  `OP2\_STR:  begin  op2 <= stk\_data\_out;  state <= `ALU;  end  `ALU:  begin  if (asn)  result <= op2 + op1;  else  result <= op2 - op1;  state <= `PUSH;  end  `PUSH:  begin  s\_flag <= (result < 0);  z\_flag <= (result == 0);  stk\_push <= 1;  stk\_data\_in <= result;  state <= `PUSH\_W;  end  `PUSH\_W:  begin  stk\_push <= 0;  state <= `INIT;  fin\_sig <= 1;  end  endcase  end  end    //debugging  always @(\*)  $display($time, "\t [ALU::%d] rstn = %b, control\_bus = %b, opt1 = %d, opt2 = %d, result = %d, stk\_data\_in = %d, stk\_push = %b, stk\_pop = %b, stk\_data\_out = %d z\_flag = %b, s\_flag = %b,", state, rstn, control\_bus, op1, op2, result, stk\_data\_in, stk\_push, stk\_pop, stk\_data\_out, z\_flag, s\_flag);   endmodule |

این ماژول برای دستورهای ADD/SUB بکار میرود. که برای این دستورات، نیاز داریم به POP/PUSH روی استک داریم.

یک استیت INIT داریم که در آن سیگنالهای اولیه را ست میکنیم و کار شروع میشود.

برای عملیات جمع و تفریق نیاز است دو تا عدد از استک POP کنیم.

کل عملیات POP کردن 3 حالت مختلف دارد. ابتدا سیگنال درخواست POP کردن فعال میشود. و بعد باید یک کلاک صبر کنیم و بعد عدد را از اول استک بیرون بیاوریم. و سپس در کلاک بعدی این عدد را در operand مان بریزیم. پس کلا 3 کلاک طول میکشد. پس برای POP کردن دو عدد 6 مرحله داریم که هرکدام یک کلاک طول میکشد. پس از اینکه هر دو عدد دریافت شد، در استیت ALU عملیات را انجام میدهیم. حال خروجی نهایی را نیز در استک باید ذخیره کنیم. که این عملیات PUSH کردن هم نیازمند 3 کلاک است. ابتدا سیگنالها فعال میشوند. سپس یک کلاک صبر میکنیم تا این سیگنال اعمال شود و PUSH انجام شود. یک مرحله سوم هم نیاز داریم که کارمان تمام شود و مثلا POP یا PUSH بعدی را شروع کنیم. که این مرحله می تواند همان استیت INIT باشد. بنابراین پس از PUSH\_W مستقیم به INIT میرویم.

ماژول PC:

|  |
| --- |
| `define INIT 3'b000 `define POP 3'b001 `define NXTL 3'b010 `define BR 3'b011 `define EXIT 3'b100 `define POP\_W 3'b101  `define OP\_JP 2'b01 `define OP\_JS 2'b10 `define OP\_JZ 2'b11  module PC (control\_bus,  clk,  rstn,  en,  pc,  z\_flag,  s\_flag,  stk\_pop,  stk\_push,  stk\_data\_out,  fin\_sig);    parameter INST\_CAP = 20;  parameter DATA\_LEN = 8;    input clk, en, rstn, z\_flag, s\_flag;  input [3:0] control\_bus;  input [DATA\_LEN-1:0] stk\_data\_out;  output reg stk\_pop,stk\_push, fin\_sig;  output reg [$clog2(INST\_CAP):0] pc;    wire[1:0] opc = {control\_bus[2], control\_bus[0]};  wire branch = (control\_bus > 2 && control\_bus < 6);  wire exit = (control\_bus == 4'b1111);  wire stall = !(control\_bus < 4'b1000 || control\_bus == 4'b1111);    reg [2:0]state;    always @(posedge clk, negedge rstn) begin  if (!rstn && !en) begin  state <= `INIT;  stk\_pop <= 1'bz;  stk\_push <= 1'bz;  pc <= 0;  fin\_sig <= 0;  end  else begin  case(state)  `INIT:  begin  if (en) begin  if (branch)  state <= `POP;  else if (exit)  state <= `EXIT;  else  state <= `NXTL;  end  fin\_sig <= 0;  stk\_pop <= 1'bz;  stk\_push <= 1'bz;  end  `POP:  begin  stk\_push <= 0;  if ((opc == `OP\_JP) || (opc == `OP\_JZ && z\_flag) || (opc == `OP\_JS && s\_flag)) begin  stk\_pop <= 1;  state <= `POP\_W;  end  else begin  state <= `NXTL;  end  end  `POP\_W:  begin  stk\_pop <= 1;  state <= `BR;  end  `NXTL:  begin  if (pc < INST\_CAP - 1) begin  pc <= pc + 1;  end  state <= `INIT;  if (stall) begin  fin\_sig <= 1;  end  end  `BR:  begin  pc <= stk\_data\_out;  state <= `INIT;  fin\_sig <= 1;  end  `EXIT:  begin  $writememb("report/result.mem", cpu0.memory0.mem);  $finish;  end  endcase  end    end    //debugging  always @(\*)  $display($time, "\t [PC::%d] rstn = %b, en = %b, control\_bus = %b, pc = %d, z\_flag = %b, s\_flag = %b, stk\_pop = %b, stk\_data\_out = %d", state, rstn, en, control\_bus, pc, z\_flag, s\_flag, stk\_pop, stk\_data\_out);   endmodule |

کلا ماژول PC سه حالت دارد. یا می خواهد برنامه را تمام کند و شروع به نوشتن اطلاعات جدید در حافظه کند، یا می خواهد دستور Jump ای را اجرا کند. و یا صرفا میخواهد Program Counter را یک واحد افزایش دهد و برود خط بعدی را بخواند و الی آخر.

برای حالت اول که صرفا کافیست $writememb را فراخوانی کنیم.

برای حالت دوم، لازم است ابتدا یک بار از استک POP کنیم و بدانیم که به کدام خانه می خواهیم برویم. و سپس عملیات Jump را انجام دهیم و PC را برابر مقدار مربوطه ست کنیم. که این POP کردن و انجام عملیات، مشابه بخش ALU، در 3 مرحله مختلف انجام میشود. POP, POP\_W, BR

برای حالتی که صرفا میخواهیم PC را یک واحد افزایش دهیم، کارمان راحت است. در حالت NXTL چک میکنیم که آیا به انتهای ظرفیت دستورات نرسیده باشیم، و سپس pc را یک واحد افزایش میدهیم. همچنین، اگر دستور قبلی stall بوده باشد، سیگنال fin\_sig که نشان دهنده پایان عملیات اجرای دستور است را = 1 می کنیم.

ماژول PP:

|  |
| --- |
| `define INIT 3'b000  `define LOAD 3'b011 `define PUSH 3'b001 `define PUSH\_W 3'b010  `define POP 3'b101 `define POP\_W 3'b110 `define STORE 3'b111  `define OP\_PUSHC 2'b00 `define OP\_PUSHM 2'b01 `define OP\_POPM 2'b10  module PP (control\_bus,  clk,  en,  rstn,  addr\_const,  stk\_data\_in,  stk\_push,  stk\_pop,  stk\_data\_out,  mem\_data\_in,  mem\_addr,  mem\_r\_en,  mem\_w\_en,  mem\_data\_out,  fin\_sig);    parameter ADDR\_LEN = 8;  parameter DATA\_LEN = 8;    input clk, rstn, en;  input [3:0] control\_bus;  input [DATA\_LEN-1:0] stk\_data\_out, mem\_data\_out, addr\_const;  output reg [DATA\_LEN-1:0] stk\_data\_in, mem\_data\_in;  output reg [ADDR\_LEN-1:0] mem\_addr;  output reg stk\_push, stk\_pop, mem\_r\_en, mem\_w\_en, fin\_sig;    wire [1:0] opc = control\_bus[1:0];  wire pp\_en = (en && control\_bus < 3);    reg [2:0]state;    always @(posedge clk, negedge rstn) begin  if (!rstn && !pp\_en) begin  state <= `INIT;  stk\_data\_in <= {DATA\_LEN{1'bz}};  stk\_push <= 1'bz;  stk\_pop <= 1'bz;  mem\_data\_in <= {DATA\_LEN{1'bz}};  mem\_addr <= {ADDR\_LEN{1'bz}};  mem\_r\_en <= 1'bz;  mem\_w\_en <= 1'bz;  fin\_sig <= 0;  end  else begin  case(state)  `INIT:  begin  if (pp\_en) begin  state <= {opc,1'b1};  end  fin\_sig <= 0;  stk\_data\_in <= {DATA\_LEN{1'bz}};  stk\_push <= 1'bz;  stk\_pop <= 1'bz;  mem\_data\_in <= {DATA\_LEN{1'bz}};  mem\_addr <= {ADDR\_LEN{1'bz}};  mem\_r\_en <= 1'bz;  mem\_w\_en <= 1'bz;  end  `PUSH:  begin  mem\_r\_en <= 0;  mem\_w\_en <= 0;  stk\_pop <= 0;  stk\_data\_in <= (opc == `OP\_PUSHC) ? addr\_const : mem\_data\_out;  state <= `PUSH\_W;  end  `PUSH\_W:  begin  stk\_push <= 1;  state <= `INIT;  fin\_sig <= 1;  end  `STORE:  begin  mem\_w\_en <= 1;  mem\_r\_en <= 0;  mem\_data\_in <= stk\_data\_out;  mem\_addr <= addr\_const;  state <= `INIT;  fin\_sig <= 1;  end  `POP:  begin  stk\_push <= 0;  stk\_pop <= 1;  state <= `POP\_W;  end  `POP\_W:  begin  state <= `STORE;  end  `LOAD:  begin  mem\_r\_en <= 1;  mem\_w\_en <= 0;  mem\_addr <= addr\_const;  state <= `PUSH;  end  default:  state <= `INIT;  endcase  end    end    //debugging  always @(\*)  $display($time, "\t [WBPB::%d] rstn = %b, en = %b, fin\_sig = %b, control\_bus = %b, addr\_const = %d, stk\_data\_in = %d, stk\_push = %b, stk\_pop = %b, stk\_data\_out = %d, mem\_data\_in = %d, mem\_addr = %d, mem\_r\_en = %b, mem\_w\_en = %b, mem\_data\_out = %d", state, rstn, en, fin\_sig, control\_bus, addr\_const, stk\_data\_in, stk\_push, stk\_pop, stk\_data\_out, mem\_data\_in, mem\_addr, mem\_r\_en, mem\_w\_en, mem\_data\_out);   endmodule |

در این ماژول، وظایف PUSH, POP, LOAD, STORE را داریم.

کلا سه نوع دستور داریم که به PP محول میشود:

PUSHC: push c

PUSHM: load m, push m

POPM: pop m, store m

در ابتدا در حالت INIT هستیم. اگر enable = 1 شود، باید وارد state ای بشویم که وابسته به نوع دستورمان است. اگر از نوع PUSHC باشیم، همان ابتدا مستقیما وارد فاز push میشویم و در 2 کلاک و در حالت های PUSH, PUSH\_W عملیات را انجام میدهیم. اگر دستورمان از نوع PUSH\_M باشد، ابتدا باید LOAD کنیم. و از حافظه بخوانیم. سپس، مشابه نوع قبلی وارد فاز PUSH و سپس PUSH\_W میشویم. اما اگر دستورمان POPM بود، لازم است ابتدا یک بار از استک POP کنیم، و سپس داده مورد نظر را در حافظه ذخیره کنیم. که این کار را در فاز STORE انجام میدهیم.

پس در نهایت، استیت های پایانی ما، وابسته به اینکه نوع دستورمان چه باشد، یکی از دو استیت STORE و PUSH\_W است. پس در این دو حالت پس از اینکه کارها انجام شد، باید به استیت اولیه INIT برگردیم.

بقیه ماژول های مورد استفاده:

Stack:

|  |
| --- |
| module Stack (rstn,  data\_in,  push,  pop,  clk,  data\_out,  full,  empty);    parameter STACK\_DEPTH = 8;  parameter WORD\_LEN = 8;    input wire rstn, push, pop, clk;  input wire[WORD\_LEN-1:0] data\_in;  output full, empty;  output reg [WORD\_LEN-1:0] data\_out;    reg[$clog2(STACK\_DEPTH):0] stack\_ptr;  reg[WORD\_LEN-1:0] memory [0:STACK\_DEPTH-1];    assign empty = (stack\_ptr == 0) ? 1'b1 : 1'b0;  assign full = (stack\_ptr == STACK\_DEPTH) ? 1'b1 : 1'b0;    wire [WORD\_LEN-1:0]top = memory[stack\_ptr-1];    // stack reset  task reset\_memory;  integer i;  begin  for (i = 0; i < STACK\_DEPTH;i++) begin  memory[i] <= {WORD\_LEN{1'b0}};  end  stack\_ptr <= 0;  end  endtask    always @(posedge clk or negedge rstn) begin  if (!rstn) begin  reset\_memory;  end  else begin  // pushing into stack  if (push && !pop && !full) begin  memory[stack\_ptr] <= data\_in;  stack\_ptr <= stack\_ptr + 1;  end  // pop from stack  if (pop && !push && !empty) begin  data\_out <= memory[stack\_ptr - 1];  stack\_ptr <= stack\_ptr - 1;  end  end    end    //debugging  always @(\*)  $display($time, "\t [STACK] rstn = %b, stack\_ptr = %d, top = %d, data\_in = %d, push = %b, pop = %b, data\_out = %d, full = %b, empty = %b", rstn, stack\_ptr, top, data\_in, push, pop, data\_out, full, empty);   endmodule |

CPU

|  |
| --- |
| `define IFIDC 2'b00 `define IFIDC\_W 2'b01 `define EXEC 2'b10 `define EXEC\_W 2'b11  module CPU (rstn,  clk);    parameter DATA\_LEN = 8;  parameter ADDR\_LEN = 8;  parameter INST\_CAP = 20;  parameter INST\_LEN = 12;  parameter WORD\_LEN = 8;  parameter MEM\_SIZE = 256;  parameter STACK\_DEPTH = 8;    input rstn, clk;    wire stk\_full, stk\_empty, stk\_push, stk\_pop, mem\_r\_en, mem\_w\_en, exec\_fin\_sig,IS\_ready;  wire [3:0] ifidc\_control\_bus;  wire [DATA\_LEN-1:0] stk\_data\_in, stk\_data\_out, mem\_data\_in, mem\_data\_out, ifidc\_addr\_const;  wire [ADDR\_LEN-1:0] mem\_addr;  wire [$clog2(INST\_CAP):0] exec\_pc;    reg ifidc\_fetch, exec\_en;  reg [1:0] state;    always @(posedge clk or negedge rstn) begin  if (!rstn) begin  ifidc\_fetch <= 0;  exec\_en <= 0;  state <= `IFIDC;  end  else begin  case(state)  `IFIDC:  begin  ifidc\_fetch <= 1;  state <= `IFIDC\_W;  end  `IFIDC\_W:  begin  ifidc\_fetch <= 0;  if (IS\_ready) begin  state <= `EXEC;  end  end  `EXEC:  begin  exec\_en <= 1;  state <= `EXEC\_W;  end  `EXEC\_W:  begin  exec\_en <= 0;  if (exec\_fin\_sig) begin  state <= `IFIDC;  end  end  endcase  end  end    IFIDC #(  .INST\_CAP(INST\_CAP),  .INST\_LEN(INST\_LEN),  .DATA\_LEN(DATA\_LEN)  ) ifidc0 (  .clk(clk),  .en(ifidc\_fetch),  .rstn(rstn),  .pc(exec\_pc),  .IS\_ready(IS\_ready),  .control\_bus(ifidc\_control\_bus),  .data(ifidc\_addr\_const)  );    EXEC #(  .DATA\_LEN(DATA\_LEN),  .ADDR\_LEN(ADDR\_LEN),  .INST\_CAP(INST\_CAP)  ) exec0 (  .clk(clk),  .en(exec\_en),  .rstn(rstn),  .pc(exec\_pc),  .control\_bus(ifidc\_control\_bus),  .addr\_const(ifidc\_addr\_const),  .stk\_data\_in(stk\_data\_in),  .stk\_push(stk\_push),  .stk\_pop(stk\_pop),  .stk\_data\_out(stk\_data\_out),  .mem\_data\_in(mem\_data\_in),  .mem\_addr(mem\_addr),  .mem\_r\_en(mem\_r\_en),  .mem\_w\_en(mem\_w\_en),  .mem\_data\_out(mem\_data\_out),  .fin\_sig(exec\_fin\_sig)  );    MEMORY #(  .ADDR\_LEN(ADDR\_LEN),  .WORD\_LEN(WORD\_LEN),  .MEM\_SIZE(MEM\_SIZE)  ) memory0 (  .clk(clk),  .addr(mem\_addr),  .r\_en(mem\_r\_en),  .w\_en(mem\_w\_en),  .data\_out(mem\_data\_out),  .data\_in(mem\_data\_in)  );    Stack #(  .WORD\_LEN(WORD\_LEN),  .STACK\_DEPTH(STACK\_DEPTH)  ) stack0 (  .clk(clk),  .rstn(rstn),  .data\_in(stk\_data\_in),  .push(stk\_push),  .pop(stk\_pop),  .data\_out(stk\_data\_out),  .full(stk\_full),  .empty(stk\_empty)  );    //debugging  always @(\*)  $display($time, "\t [CPU::%d] ifidc\_fetch = %b, exec\_en = %b, exec\_finish = %b", state, ifidc\_fetch, exec\_en, exec\_fin\_sig);     endmodule |

Testbench

|  |
| --- |
| module testbench();    parameter CLK\_C = 10;  parameter DATA\_LEN = 8;  parameter ADDR\_LEN = 8;  parameter INST\_CAP = 20;  parameter INST\_LEN = 12;  parameter WORD\_LEN = 8;  parameter MEM\_SIZE = 256;  parameter STACK\_DEPTH = 8;    reg clk, rstn;    CPU #(  .WORD\_LEN(WORD\_LEN),  .DATA\_LEN(DATA\_LEN),  .ADDR\_LEN(ADDR\_LEN),  .MEM\_SIZE(MEM\_SIZE),  .STACK\_DEPTH(STACK\_DEPTH),  .INST\_LEN(INST\_LEN),  .INST\_CAP(INST\_CAP)  ) cpu0 (  .clk(clk),  .rstn(rstn)  );    initial begin  $dumpfile("report/waveform.vcd");  $dumpvars(0,cpu0);  end    initial begin  clk = 0;  forever clk = #(CLK\_C/2) ~clk;  end    initial begin  $readmemb("report/memory.mem", cpu0.memory0.mem, 0, MEM\_SIZE-1);  $readmemb("report/is.mem", cpu0.ifidc0.inst\_mem, 0, INST\_CAP-1);  end    initial begin  rstn = 0;  #CLK\_C  rstn = 1;  #(100000 \* CLK\_C);  $finish;  end   endmodule |

Is\_math.mem

|  |
| --- |
| // Example: Y=((X+23)+(X+23))-12  0001\_0000\_0001 // PUSHM X  0000\_0001\_0111 // PUSHC 23  0110\_xxxx\_xxxx // ADD  0010\_0000\_0010 // POPM Y  0001\_0000\_0010 // PUSHM Y  0001\_0000\_0010 // PUSHM Y  0110\_xxxx\_xxxx // ADD  0000\_0000\_1100 // PUSHC 12 0111\_xxxx\_xxxx // SUB 0010\_0000\_0010 // POPM RES 1111\_xxxx\_xxxx // EXIT 0000\_0000\_0000 0000\_0000\_0000 0000\_0000\_0000 0000\_0000\_0000 0000\_0000\_0000 0000\_0000\_0000 0000\_0000\_0000 0000\_0000\_0000 0000\_0000\_0000 |

Is.mem

|  |
| --- |
| // Example: Y=((X+23)+(X+23))-12  0001\_0000\_0001 // PUSHM X  0000\_0001\_0111 // PUSHC 23  0110\_xxxx\_xxxx // ADD  0010\_0000\_0010 // POPM Y  0001\_0000\_0010 // PUSHM Y  0001\_0000\_0010 // PUSHM Y  0110\_xxxx\_xxxx // ADD  0000\_0000\_1100 // PUSHC 12 0111\_xxxx\_xxxx // SUB 0010\_0000\_0010 // POPM RES 1111\_xxxx\_xxxx // EXIT 0000\_0000\_0000 0000\_0000\_0000 0000\_0000\_0000 0000\_0000\_0000 0000\_0000\_0000 0000\_0000\_0000 0000\_0000\_0000 0000\_0000\_0000 0000\_0000\_0000 |

Memory.mem

|  |
| --- |
| // main memory: 256 \* word, word = 8 bits 0000\_0000 0000\_0101 0000\_1010 0000\_0000 0000\_0000 0000\_0000 0000\_0000 0000\_0000 0000\_0000 0000\_0000 0000\_0000 0000\_0000 0000\_0000 0000\_0000 0000\_0000 0000\_0000 0000\_0000 0000\_0000 0000\_0000 0000\_0000 |

Result.mem

|  |
| --- |
| // 0x00000000 00000000 00000101 00101100 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 |

Is\_jump.mem

|  |
| --- |
| // Branch instruction tests 0000\_0000\_0011 // 0.PUSHC 3  0000\_0000\_0011 // 1.PUSHC 3  0111\_xxxx\_xxxx // 2.SUB 0000\_0000\_0110 // 3.PUSHC 6  0100\_xxxx\_xxxx // 4.JZ  1111\_xxxx\_xxxx // 5.EXIT  0000\_0000\_1000 // 6.PUSHC 8  0000\_0001\_0000 // 7.PUSHC 16 0111\_xxxx\_xxxx // 8.SUB 0000\_0000\_1100 // 9.PUSHC 12 0101\_xxxx\_xxxx // 10.JS  1111\_xxxx\_xxxx // 11.EXIT 0000\_0000\_0101 // 12.PUSHC 5 0011\_xxxx\_xxxx // 13.JUMP 0000\_0000\_0000  0000\_0000\_0000 0000\_0000\_0000 0000\_0000\_0000 0000\_0000\_0000 0000\_0000\_0000 |

Is\_swap.mem

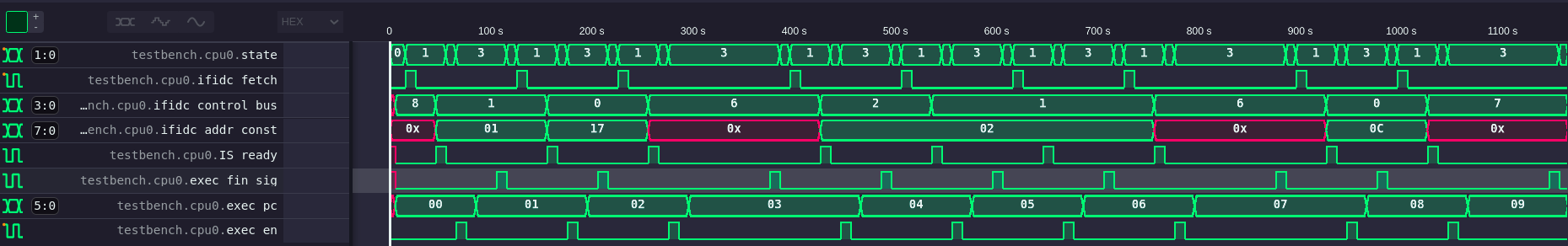
|  |
| --- |
| // SWAP 0001\_0000\_0001 // PUSHM ARG1  0001\_0000\_0010 // PUSHM ARG2  0010\_0000\_0001 // POPM ARG2 0010\_0000\_0010 // POPM ARG1 1111\_xxxx\_xxxx // EXIT 0000\_0000\_0000 0000\_0000\_0000 0000\_0000\_0000 0000\_0000\_0000 0000\_0000\_0000 0000\_0000\_0000 0000\_0000\_0000 0000\_0000\_0000 0000\_0000\_0000 0000\_0000\_0000 0000\_0000\_0000 0000\_0000\_0000 0000\_0000\_0000 0000\_0000\_0000 0000\_0000\_0000 0000\_0000\_0000 0000\_0000\_0000 |

Results.txt

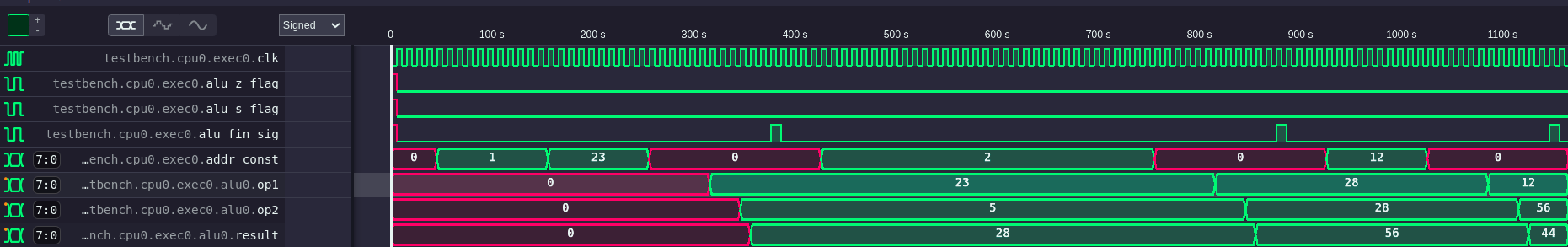
|  |
| --- |
| VCD info: dumpfile report/waveform.vcd opened for output.  0 [STACK] rstn = 0, stack\_ptr = x, top = x, data\_in = x, push = x, pop = x, data\_out = x, full = x, empty = x  0 [IFIDC::x] pc = x control\_bus = xxxx addr\_imm = x IS\_ready = x rstn = 0 en = x  0 [WBPB::x] rstn = 0, en = x, fin\_sig = x, control\_bus = xxxx, addr\_const = x, stk\_data\_in = x, stk\_push = x, stk\_pop = x, stk\_data\_out = x, mem\_data\_in = x, mem\_addr = x, mem\_r\_en = x, mem\_w\_en = x, mem\_data\_out = x  0 [PC::x] rstn = 0, en = x, control\_bus = xxxx, pc = x, z\_flag = x, s\_flag = x, stk\_pop = x, stk\_data\_out = x  0 [ALU:: x] rstn = 0, control\_bus = xxxx, opt1 = x, opt2 = x, result = x, stk\_data\_in = x, stk\_push = x, stk\_pop = x, stk\_data\_out = x z\_flag = x, s\_flag = x,  0 [CPU::0] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = x  0 [IFIDC::x] pc = x control\_bus = xxxx addr\_imm = x IS\_ready = x rstn = 0 en = 0  0 [WBPB::0] rstn = 0, en = 0, fin\_sig = x, control\_bus = xxxx, addr\_const = x, stk\_data\_in = x, stk\_push = x, stk\_pop = x, stk\_data\_out = x, mem\_data\_in = x, mem\_addr = x, mem\_r\_en = x, mem\_w\_en = x, mem\_data\_out = x  0 [PC::x] rstn = 0, en = 0, control\_bus = xxxx, pc = x, z\_flag = x, s\_flag = x, stk\_pop = x, stk\_data\_out = x  0 [STACK] rstn = 0, stack\_ptr = 0, top = x, data\_in = x, push = x, pop = x, data\_out = x, full = 0, empty = 1  5 [IFIDC::0] pc = 0 control\_bus = 1000 addr\_imm = x IS\_ready = 0 rstn = 0 en = 0  5 [WBPB::0] rstn = 0, en = 0, fin\_sig = 0, control\_bus = 1000, addr\_const = x, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = x, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = x  5 [PC::0] rstn = 0, en = 0, control\_bus = 1000, pc = 0, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = x  5 [ALU:: 0] rstn = 0, control\_bus = 1000, opt1 = x, opt2 = x, result = x, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = x z\_flag = 0, s\_flag = 0,  5 [STACK] rstn = 0, stack\_ptr = 0, top = x, data\_in = z, push = z, pop = z, data\_out = x, full = 0, empty = 1  5 [MEMORY] r\_en = z, addr = z, w\_en = z, data\_in = zzzzzzzz, data\_out = xxxxxxxx  5 [CPU::0] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  10 [STACK] rstn = 1, stack\_ptr = 0, top = x, data\_in = z, push = z, pop = z, data\_out = x, full = 0, empty = 1  10 [IFIDC::0] pc = 0 control\_bus = 1000 addr\_imm = x IS\_ready = 0 rstn = 1 en = 0  10 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 1000, addr\_const = x, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = x, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = x  10 [PC::0] rstn = 1, en = 0, control\_bus = 1000, pc = 0, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = x  10 [ALU:: 0] rstn = 1, control\_bus = 1000, opt1 = x, opt2 = x, result = x, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = x z\_flag = 0, s\_flag = 0,  15 [CPU::1] ifidc\_fetch = 1, exec\_en = 0, exec\_finish = 0  15 [IFIDC::0] pc = 0 control\_bus = 1000 addr\_imm = x IS\_ready = 0 rstn = 1 en = 1  25 [IFIDC::1] pc = 0 control\_bus = 1000 addr\_imm = x IS\_ready = 0 rstn = 1 en = 0  25 [CPU::1] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  35 [IFIDC::2] pc = 0 control\_bus = 1000 addr\_imm = x IS\_ready = 0 rstn = 1 en = 0  45 [IFIDC::0] pc = 0 control\_bus = 0001 addr\_imm = 1 IS\_ready = 1 rstn = 1 en = 0  45 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0001, addr\_const = 1, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = x, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = x  45 [PC::0] rstn = 1, en = 0, control\_bus = 0001, pc = 0, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = x  45 [ALU:: 0] rstn = 1, control\_bus = 0001, opt1 = x, opt2 = x, result = x, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = x z\_flag = 0, s\_flag = 0,  55 [CPU::2] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  55 [IFIDC::0] pc = 0 control\_bus = 0001 addr\_imm = 1 IS\_ready = 0 rstn = 1 en = 0  65 [CPU::3] ifidc\_fetch = 0, exec\_en = 1, exec\_finish = 0  65 [WBPB::0] rstn = 1, en = 1, fin\_sig = 0, control\_bus = 0001, addr\_const = 1, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = x, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = x  65 [PC::0] rstn = 1, en = 1, control\_bus = 0001, pc = 0, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = x  75 [CPU::3] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  75 [WBPB::3] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0001, addr\_const = 1, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = x, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = x  75 [PC::2] rstn = 1, en = 0, control\_bus = 0001, pc = 0, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = x  85 [MEMORY] r\_en = 1, addr = 1, w\_en = 0, data\_in = zzzzzzzz, data\_out = xxxxxxxx  85 [WBPB::1] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0001, addr\_const = 1, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = x, mem\_data\_in = z, mem\_addr = 1, mem\_r\_en = 1, mem\_w\_en = 0, mem\_data\_out = x  85 [IFIDC::0] pc = 1 control\_bus = 0001 addr\_imm = 1 IS\_ready = 0 rstn = 1 en = 0  85 [PC::0] rstn = 1, en = 0, control\_bus = 0001, pc = 1, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = x  85 [MEMORY] r\_en = 1, addr = 1, w\_en = 0, data\_in = zzzzzzzz, data\_out = 00000101  85 [WBPB::1] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0001, addr\_const = 1, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = x, mem\_data\_in = z, mem\_addr = 1, mem\_r\_en = 1, mem\_w\_en = 0, mem\_data\_out = 5  95 [MEMORY] r\_en = 0, addr = 1, w\_en = 0, data\_in = zzzzzzzz, data\_out = 00000101  95 [WBPB::2] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0001, addr\_const = 1, stk\_data\_in = 5, stk\_push = z, stk\_pop = 0, stk\_data\_out = x, mem\_data\_in = z, mem\_addr = 1, mem\_r\_en = 0, mem\_w\_en = 0, mem\_data\_out = 5  105 [WBPB::0] rstn = 1, en = 0, fin\_sig = 1, control\_bus = 0001, addr\_const = 1, stk\_data\_in = 5, stk\_push = 1, stk\_pop = 0, stk\_data\_out = x, mem\_data\_in = z, mem\_addr = 1, mem\_r\_en = 0, mem\_w\_en = 0, mem\_data\_out = 5  105 [CPU::3] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 1  115 [STACK] rstn = 1, stack\_ptr = 1, top = 5, data\_in = z, push = z, pop = z, data\_out = x, full = 0, empty = 1  115 [CPU::0] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 1  115 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0001, addr\_const = 1, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = x, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 5  115 [MEMORY] r\_en = z, addr = z, w\_en = z, data\_in = zzzzzzzz, data\_out = 00000101  115 [STACK] rstn = 1, stack\_ptr = 1, top = 5, data\_in = z, push = z, pop = z, data\_out = x, full = 0, empty = 0  115 [CPU::0] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  125 [CPU::1] ifidc\_fetch = 1, exec\_en = 0, exec\_finish = 0  125 [IFIDC::0] pc = 1 control\_bus = 0001 addr\_imm = 1 IS\_ready = 0 rstn = 1 en = 1  135 [CPU::1] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  135 [IFIDC::1] pc = 1 control\_bus = 0001 addr\_imm = 1 IS\_ready = 0 rstn = 1 en = 0  145 [IFIDC::2] pc = 1 control\_bus = 0001 addr\_imm = 1 IS\_ready = 0 rstn = 1 en = 0  155 [IFIDC::0] pc = 1 control\_bus = 0000 addr\_imm = 23 IS\_ready = 1 rstn = 1 en = 0  155 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0000, addr\_const = 23, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = x, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 5  155 [PC::0] rstn = 1, en = 0, control\_bus = 0000, pc = 1, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = x  155 [ALU:: 0] rstn = 1, control\_bus = 0000, opt1 = x, opt2 = x, result = x, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = x z\_flag = 0, s\_flag = 0,  165 [IFIDC::0] pc = 1 control\_bus = 0000 addr\_imm = 23 IS\_ready = 0 rstn = 1 en = 0  165 [CPU::2] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  175 [CPU::3] ifidc\_fetch = 0, exec\_en = 1, exec\_finish = 0  175 [WBPB::0] rstn = 1, en = 1, fin\_sig = 0, control\_bus = 0000, addr\_const = 23, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = x, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 5  175 [PC::0] rstn = 1, en = 1, control\_bus = 0000, pc = 1, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = x  185 [WBPB::1] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0000, addr\_const = 23, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = x, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 5  185 [PC::2] rstn = 1, en = 0, control\_bus = 0000, pc = 1, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = x  185 [CPU::3] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  195 [IFIDC::0] pc = 2 control\_bus = 0000 addr\_imm = 23 IS\_ready = 0 rstn = 1 en = 0  195 [PC::0] rstn = 1, en = 0, control\_bus = 0000, pc = 2, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = x  195 [MEMORY] r\_en = 0, addr = z, w\_en = 0, data\_in = zzzzzzzz, data\_out = 00000101  195 [WBPB::2] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0000, addr\_const = 23, stk\_data\_in = 23, stk\_push = z, stk\_pop = 0, stk\_data\_out = x, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = 0, mem\_w\_en = 0, mem\_data\_out = 5  205 [WBPB::0] rstn = 1, en = 0, fin\_sig = 1, control\_bus = 0000, addr\_const = 23, stk\_data\_in = 23, stk\_push = 1, stk\_pop = 0, stk\_data\_out = x, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = 0, mem\_w\_en = 0, mem\_data\_out = 5  205 [CPU::3] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 1  215 [STACK] rstn = 1, stack\_ptr = 2, top = 23, data\_in = z, push = z, pop = z, data\_out = x, full = 0, empty = 0  215 [CPU::0] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 1  215 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0000, addr\_const = 23, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = x, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 5  215 [MEMORY] r\_en = z, addr = z, w\_en = z, data\_in = zzzzzzzz, data\_out = 00000101  215 [CPU::0] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  225 [CPU::1] ifidc\_fetch = 1, exec\_en = 0, exec\_finish = 0  225 [IFIDC::0] pc = 2 control\_bus = 0000 addr\_imm = 23 IS\_ready = 0 rstn = 1 en = 1  235 [CPU::1] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  235 [IFIDC::1] pc = 2 control\_bus = 0000 addr\_imm = 23 IS\_ready = 0 rstn = 1 en = 0  245 [IFIDC::2] pc = 2 control\_bus = 0000 addr\_imm = 23 IS\_ready = 0 rstn = 1 en = 0  255 [IFIDC::0] pc = 2 control\_bus = 0110 addr\_imm = x IS\_ready = 1 rstn = 1 en = 0  255 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0110, addr\_const = x, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = x, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 5  255 [PC::0] rstn = 1, en = 0, control\_bus = 0110, pc = 2, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = x  255 [ALU:: 0] rstn = 1, control\_bus = 0110, opt1 = x, opt2 = x, result = x, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = x z\_flag = 0, s\_flag = 0,  265 [IFIDC::0] pc = 2 control\_bus = 0110 addr\_imm = x IS\_ready = 0 rstn = 1 en = 0  265 [CPU::2] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  275 [CPU::3] ifidc\_fetch = 0, exec\_en = 1, exec\_finish = 0  275 [WBPB::0] rstn = 1, en = 1, fin\_sig = 0, control\_bus = 0110, addr\_const = x, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = x, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 5  275 [PC::0] rstn = 1, en = 1, control\_bus = 0110, pc = 2, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = x  285 [ALU:: 1] rstn = 1, control\_bus = 0110, opt1 = x, opt2 = x, result = x, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = x z\_flag = 0, s\_flag = 0,  285 [PC::2] rstn = 1, en = 0, control\_bus = 0110, pc = 2, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = x  285 [CPU::3] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  285 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0110, addr\_const = x, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = x, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 5  295 [IFIDC::0] pc = 3 control\_bus = 0110 addr\_imm = x IS\_ready = 0 rstn = 1 en = 0  295 [PC::0] rstn = 1, en = 0, control\_bus = 0110, pc = 3, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = x  295 [STACK] rstn = 1, stack\_ptr = 2, top = 23, data\_in = z, push = 0, pop = 1, data\_out = x, full = 0, empty = 0  295 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0110, addr\_const = x, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = x, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 5  295 [ALU:: 2] rstn = 1, control\_bus = 0110, opt1 = x, opt2 = x, result = x, stk\_data\_in = z, stk\_push = 0, stk\_pop = 1, stk\_data\_out = x z\_flag = 0, s\_flag = 0,  305 [STACK] rstn = 1, stack\_ptr = 1, top = 5, data\_in = z, push = 0, pop = 0, data\_out = 23, full = 0, empty = 0  305 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0110, addr\_const = x, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 23, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 5  305 [PC::0] rstn = 1, en = 0, control\_bus = 0110, pc = 3, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 23  305 [ALU:: 3] rstn = 1, control\_bus = 0110, opt1 = x, opt2 = x, result = x, stk\_data\_in = z, stk\_push = 0, stk\_pop = 0, stk\_data\_out = 23 z\_flag = 0, s\_flag = 0,  315 [ALU:: 4] rstn = 1, control\_bus = 0110, opt1 = 23, opt2 = x, result = x, stk\_data\_in = z, stk\_push = 0, stk\_pop = 0, stk\_data\_out = 23 z\_flag = 0, s\_flag = 0,  325 [STACK] rstn = 1, stack\_ptr = 1, top = 5, data\_in = z, push = 0, pop = 1, data\_out = 23, full = 0, empty = 0  325 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0110, addr\_const = x, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 23, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 5  325 [PC::0] rstn = 1, en = 0, control\_bus = 0110, pc = 3, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 23  325 [ALU:: 5] rstn = 1, control\_bus = 0110, opt1 = 23, opt2 = x, result = x, stk\_data\_in = z, stk\_push = 0, stk\_pop = 1, stk\_data\_out = 23 z\_flag = 0, s\_flag = 0,  335 [STACK] rstn = 1, stack\_ptr = 0, top = x, data\_in = z, push = 0, pop = 0, data\_out = 5, full = 0, empty = 0  335 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0110, addr\_const = x, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 5, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 5  335 [PC::0] rstn = 1, en = 0, control\_bus = 0110, pc = 3, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 5  335 [ALU:: 6] rstn = 1, control\_bus = 0110, opt1 = 23, opt2 = x, result = x, stk\_data\_in = z, stk\_push = 0, stk\_pop = 0, stk\_data\_out = 5 z\_flag = 0, s\_flag = 0,  335 [STACK] rstn = 1, stack\_ptr = 0, top = x, data\_in = z, push = 0, pop = 0, data\_out = 5, full = 0, empty = 1  345 [ALU:: 7] rstn = 1, control\_bus = 0110, opt1 = 23, opt2 = 5, result = x, stk\_data\_in = z, stk\_push = 0, stk\_pop = 0, stk\_data\_out = 5 z\_flag = 0, s\_flag = 0,  355 [ALU:: 8] rstn = 1, control\_bus = 0110, opt1 = 23, opt2 = 5, result = 28, stk\_data\_in = z, stk\_push = 0, stk\_pop = 0, stk\_data\_out = 5 z\_flag = 0, s\_flag = 0,  365 [STACK] rstn = 1, stack\_ptr = 0, top = x, data\_in = 28, push = 1, pop = 0, data\_out = 5, full = 0, empty = 1  365 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0110, addr\_const = x, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 5, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 5  365 [ALU:: 9] rstn = 1, control\_bus = 0110, opt1 = 23, opt2 = 5, result = 28, stk\_data\_in = 28, stk\_push = 1, stk\_pop = 0, stk\_data\_out = 5 z\_flag = 0, s\_flag = 0,  375 [STACK] rstn = 1, stack\_ptr = 1, top = 28, data\_in = 28, push = 0, pop = 0, data\_out = 5, full = 0, empty = 1  375 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0110, addr\_const = x, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 5, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 5  375 [ALU:: 0] rstn = 1, control\_bus = 0110, opt1 = 23, opt2 = 5, result = 28, stk\_data\_in = 28, stk\_push = 0, stk\_pop = 0, stk\_data\_out = 5 z\_flag = 0, s\_flag = 0,  375 [STACK] rstn = 1, stack\_ptr = 1, top = 28, data\_in = 28, push = 0, pop = 0, data\_out = 5, full = 0, empty = 0  375 [CPU::3] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 1  385 [STACK] rstn = 1, stack\_ptr = 1, top = 28, data\_in = z, push = z, pop = z, data\_out = 5, full = 0, empty = 0  385 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0110, addr\_const = x, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 5, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 5  385 [ALU:: 0] rstn = 1, control\_bus = 0110, opt1 = 23, opt2 = 5, result = 28, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 5 z\_flag = 0, s\_flag = 0,  385 [PC::0] rstn = 1, en = 0, control\_bus = 0110, pc = 3, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 5  385 [CPU::0] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 1  385 [CPU::0] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  395 [CPU::1] ifidc\_fetch = 1, exec\_en = 0, exec\_finish = 0  395 [IFIDC::0] pc = 3 control\_bus = 0110 addr\_imm = x IS\_ready = 0 rstn = 1 en = 1  405 [IFIDC::1] pc = 3 control\_bus = 0110 addr\_imm = x IS\_ready = 0 rstn = 1 en = 0  405 [CPU::1] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  415 [IFIDC::2] pc = 3 control\_bus = 0110 addr\_imm = x IS\_ready = 0 rstn = 1 en = 0  425 [IFIDC::0] pc = 3 control\_bus = 0010 addr\_imm = 2 IS\_ready = 1 rstn = 1 en = 0  425 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0010, addr\_const = 2, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 5, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 5  425 [PC::0] rstn = 1, en = 0, control\_bus = 0010, pc = 3, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 5  425 [ALU:: 0] rstn = 1, control\_bus = 0010, opt1 = 23, opt2 = 5, result = 28, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 5 z\_flag = 0, s\_flag = 0,  435 [CPU::2] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  435 [IFIDC::0] pc = 3 control\_bus = 0010 addr\_imm = 2 IS\_ready = 0 rstn = 1 en = 0  445 [CPU::3] ifidc\_fetch = 0, exec\_en = 1, exec\_finish = 0  445 [WBPB::0] rstn = 1, en = 1, fin\_sig = 0, control\_bus = 0010, addr\_const = 2, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 5, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 5  445 [PC::0] rstn = 1, en = 1, control\_bus = 0010, pc = 3, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 5  455 [CPU::3] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  455 [WBPB::5] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0010, addr\_const = 2, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 5, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 5  455 [PC::2] rstn = 1, en = 0, control\_bus = 0010, pc = 3, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 5  465 [WBPB::6] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0010, addr\_const = 2, stk\_data\_in = z, stk\_push = 0, stk\_pop = 1, stk\_data\_out = 5, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 5  465 [IFIDC::0] pc = 4 control\_bus = 0010 addr\_imm = 2 IS\_ready = 0 rstn = 1 en = 0  465 [PC::0] rstn = 1, en = 0, control\_bus = 0010, pc = 4, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 5  475 [STACK] rstn = 1, stack\_ptr = 0, top = x, data\_in = z, push = 0, pop = 1, data\_out = 28, full = 0, empty = 0  475 [WBPB::7] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0010, addr\_const = 2, stk\_data\_in = z, stk\_push = 0, stk\_pop = 1, stk\_data\_out = 28, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 5  475 [PC::0] rstn = 1, en = 0, control\_bus = 0010, pc = 4, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 28  475 [ALU:: 0] rstn = 1, control\_bus = 0010, opt1 = 23, opt2 = 5, result = 28, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 28 z\_flag = 0, s\_flag = 0,  475 [STACK] rstn = 1, stack\_ptr = 0, top = x, data\_in = z, push = 0, pop = 1, data\_out = 28, full = 0, empty = 1  485 [MEMORY] r\_en = 0, addr = 2, w\_en = 1, data\_in = 00011100, data\_out = 00000101  485 [WBPB::0] rstn = 1, en = 0, fin\_sig = 1, control\_bus = 0010, addr\_const = 2, stk\_data\_in = z, stk\_push = 0, stk\_pop = 1, stk\_data\_out = 28, mem\_data\_in = 28, mem\_addr = 2, mem\_r\_en = 0, mem\_w\_en = 1, mem\_data\_out = 5  485 [CPU::3] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 1  495 [CPU::0] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 1  495 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0010, addr\_const = 2, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 28, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 5  495 [MEMORY] r\_en = z, addr = z, w\_en = z, data\_in = zzzzzzzz, data\_out = 00000101  495 [CPU::0] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  505 [CPU::1] ifidc\_fetch = 1, exec\_en = 0, exec\_finish = 0  505 [IFIDC::0] pc = 4 control\_bus = 0010 addr\_imm = 2 IS\_ready = 0 rstn = 1 en = 1  515 [CPU::1] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  515 [IFIDC::1] pc = 4 control\_bus = 0010 addr\_imm = 2 IS\_ready = 0 rstn = 1 en = 0  525 [IFIDC::2] pc = 4 control\_bus = 0010 addr\_imm = 2 IS\_ready = 0 rstn = 1 en = 0  535 [IFIDC::0] pc = 4 control\_bus = 0001 addr\_imm = 2 IS\_ready = 1 rstn = 1 en = 0  535 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0001, addr\_const = 2, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 28, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 5  535 [PC::0] rstn = 1, en = 0, control\_bus = 0001, pc = 4, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 28  535 [ALU:: 0] rstn = 1, control\_bus = 0001, opt1 = 23, opt2 = 5, result = 28, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 28 z\_flag = 0, s\_flag = 0,  545 [IFIDC::0] pc = 4 control\_bus = 0001 addr\_imm = 2 IS\_ready = 0 rstn = 1 en = 0  545 [CPU::2] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  555 [CPU::3] ifidc\_fetch = 0, exec\_en = 1, exec\_finish = 0  555 [WBPB::0] rstn = 1, en = 1, fin\_sig = 0, control\_bus = 0001, addr\_const = 2, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 28, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 5  555 [PC::0] rstn = 1, en = 1, control\_bus = 0001, pc = 4, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 28  565 [WBPB::3] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0001, addr\_const = 2, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 28, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 5  565 [PC::2] rstn = 1, en = 0, control\_bus = 0001, pc = 4, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 28  565 [CPU::3] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  575 [IFIDC::0] pc = 5 control\_bus = 0001 addr\_imm = 2 IS\_ready = 0 rstn = 1 en = 0  575 [PC::0] rstn = 1, en = 0, control\_bus = 0001, pc = 5, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 28  575 [MEMORY] r\_en = 1, addr = 2, w\_en = 0, data\_in = zzzzzzzz, data\_out = 00000101  575 [WBPB::1] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0001, addr\_const = 2, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 28, mem\_data\_in = z, mem\_addr = 2, mem\_r\_en = 1, mem\_w\_en = 0, mem\_data\_out = 5  575 [MEMORY] r\_en = 1, addr = 2, w\_en = 0, data\_in = zzzzzzzz, data\_out = 00011100  575 [WBPB::1] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0001, addr\_const = 2, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 28, mem\_data\_in = z, mem\_addr = 2, mem\_r\_en = 1, mem\_w\_en = 0, mem\_data\_out = 28  585 [MEMORY] r\_en = 0, addr = 2, w\_en = 0, data\_in = zzzzzzzz, data\_out = 00011100  585 [WBPB::2] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0001, addr\_const = 2, stk\_data\_in = 28, stk\_push = z, stk\_pop = 0, stk\_data\_out = 28, mem\_data\_in = z, mem\_addr = 2, mem\_r\_en = 0, mem\_w\_en = 0, mem\_data\_out = 28  595 [WBPB::0] rstn = 1, en = 0, fin\_sig = 1, control\_bus = 0001, addr\_const = 2, stk\_data\_in = 28, stk\_push = 1, stk\_pop = 0, stk\_data\_out = 28, mem\_data\_in = z, mem\_addr = 2, mem\_r\_en = 0, mem\_w\_en = 0, mem\_data\_out = 28  595 [CPU::3] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 1  605 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0001, addr\_const = 2, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 28, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 28  605 [MEMORY] r\_en = z, addr = z, w\_en = z, data\_in = zzzzzzzz, data\_out = 00011100  605 [CPU::0] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 1  605 [STACK] rstn = 1, stack\_ptr = 1, top = 28, data\_in = z, push = z, pop = z, data\_out = 28, full = 0, empty = 1  605 [STACK] rstn = 1, stack\_ptr = 1, top = 28, data\_in = z, push = z, pop = z, data\_out = 28, full = 0, empty = 0  605 [CPU::0] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  615 [CPU::1] ifidc\_fetch = 1, exec\_en = 0, exec\_finish = 0  615 [IFIDC::0] pc = 5 control\_bus = 0001 addr\_imm = 2 IS\_ready = 0 rstn = 1 en = 1  625 [IFIDC::1] pc = 5 control\_bus = 0001 addr\_imm = 2 IS\_ready = 0 rstn = 1 en = 0  625 [CPU::1] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  635 [IFIDC::2] pc = 5 control\_bus = 0001 addr\_imm = 2 IS\_ready = 0 rstn = 1 en = 0  645 [IFIDC::0] pc = 5 control\_bus = 0001 addr\_imm = 2 IS\_ready = 1 rstn = 1 en = 0  655 [CPU::2] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  655 [IFIDC::0] pc = 5 control\_bus = 0001 addr\_imm = 2 IS\_ready = 0 rstn = 1 en = 0  665 [CPU::3] ifidc\_fetch = 0, exec\_en = 1, exec\_finish = 0  665 [WBPB::0] rstn = 1, en = 1, fin\_sig = 0, control\_bus = 0001, addr\_const = 2, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 28, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 28  665 [PC::0] rstn = 1, en = 1, control\_bus = 0001, pc = 5, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 28  675 [CPU::3] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  675 [WBPB::3] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0001, addr\_const = 2, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 28, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 28  675 [PC::2] rstn = 1, en = 0, control\_bus = 0001, pc = 5, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 28  685 [MEMORY] r\_en = 1, addr = 2, w\_en = 0, data\_in = zzzzzzzz, data\_out = 00011100  685 [WBPB::1] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0001, addr\_const = 2, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 28, mem\_data\_in = z, mem\_addr = 2, mem\_r\_en = 1, mem\_w\_en = 0, mem\_data\_out = 28  685 [IFIDC::0] pc = 6 control\_bus = 0001 addr\_imm = 2 IS\_ready = 0 rstn = 1 en = 0  685 [PC::0] rstn = 1, en = 0, control\_bus = 0001, pc = 6, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 28  695 [MEMORY] r\_en = 0, addr = 2, w\_en = 0, data\_in = zzzzzzzz, data\_out = 00011100  695 [WBPB::2] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0001, addr\_const = 2, stk\_data\_in = 28, stk\_push = z, stk\_pop = 0, stk\_data\_out = 28, mem\_data\_in = z, mem\_addr = 2, mem\_r\_en = 0, mem\_w\_en = 0, mem\_data\_out = 28  705 [WBPB::0] rstn = 1, en = 0, fin\_sig = 1, control\_bus = 0001, addr\_const = 2, stk\_data\_in = 28, stk\_push = 1, stk\_pop = 0, stk\_data\_out = 28, mem\_data\_in = z, mem\_addr = 2, mem\_r\_en = 0, mem\_w\_en = 0, mem\_data\_out = 28  705 [CPU::3] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 1  715 [STACK] rstn = 1, stack\_ptr = 2, top = 28, data\_in = z, push = z, pop = z, data\_out = 28, full = 0, empty = 0  715 [CPU::0] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 1  715 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0001, addr\_const = 2, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 28, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 28  715 [MEMORY] r\_en = z, addr = z, w\_en = z, data\_in = zzzzzzzz, data\_out = 00011100  715 [CPU::0] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  725 [CPU::1] ifidc\_fetch = 1, exec\_en = 0, exec\_finish = 0  725 [IFIDC::0] pc = 6 control\_bus = 0001 addr\_imm = 2 IS\_ready = 0 rstn = 1 en = 1  735 [CPU::1] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  735 [IFIDC::1] pc = 6 control\_bus = 0001 addr\_imm = 2 IS\_ready = 0 rstn = 1 en = 0  745 [IFIDC::2] pc = 6 control\_bus = 0001 addr\_imm = 2 IS\_ready = 0 rstn = 1 en = 0  755 [IFIDC::0] pc = 6 control\_bus = 0110 addr\_imm = x IS\_ready = 1 rstn = 1 en = 0  755 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0110, addr\_const = x, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 28, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 28  755 [PC::0] rstn = 1, en = 0, control\_bus = 0110, pc = 6, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 28  755 [ALU:: 0] rstn = 1, control\_bus = 0110, opt1 = 23, opt2 = 5, result = 28, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 28 z\_flag = 0, s\_flag = 0,  765 [IFIDC::0] pc = 6 control\_bus = 0110 addr\_imm = x IS\_ready = 0 rstn = 1 en = 0  765 [CPU::2] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  775 [CPU::3] ifidc\_fetch = 0, exec\_en = 1, exec\_finish = 0  775 [WBPB::0] rstn = 1, en = 1, fin\_sig = 0, control\_bus = 0110, addr\_const = x, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 28, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 28  775 [PC::0] rstn = 1, en = 1, control\_bus = 0110, pc = 6, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 28  785 [ALU:: 1] rstn = 1, control\_bus = 0110, opt1 = 23, opt2 = 5, result = 28, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 28 z\_flag = 0, s\_flag = 0,  785 [PC::2] rstn = 1, en = 0, control\_bus = 0110, pc = 6, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 28  785 [CPU::3] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  785 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0110, addr\_const = x, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 28, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 28  795 [IFIDC::0] pc = 7 control\_bus = 0110 addr\_imm = x IS\_ready = 0 rstn = 1 en = 0  795 [PC::0] rstn = 1, en = 0, control\_bus = 0110, pc = 7, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 28  795 [STACK] rstn = 1, stack\_ptr = 2, top = 28, data\_in = z, push = 0, pop = 1, data\_out = 28, full = 0, empty = 0  795 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0110, addr\_const = x, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 28, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 28  795 [ALU:: 2] rstn = 1, control\_bus = 0110, opt1 = 23, opt2 = 5, result = 28, stk\_data\_in = z, stk\_push = 0, stk\_pop = 1, stk\_data\_out = 28 z\_flag = 0, s\_flag = 0,  805 [STACK] rstn = 1, stack\_ptr = 1, top = 28, data\_in = z, push = 0, pop = 0, data\_out = 28, full = 0, empty = 0  805 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0110, addr\_const = x, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 28, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 28  805 [PC::0] rstn = 1, en = 0, control\_bus = 0110, pc = 7, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 28  805 [ALU:: 3] rstn = 1, control\_bus = 0110, opt1 = 23, opt2 = 5, result = 28, stk\_data\_in = z, stk\_push = 0, stk\_pop = 0, stk\_data\_out = 28 z\_flag = 0, s\_flag = 0,  815 [ALU:: 4] rstn = 1, control\_bus = 0110, opt1 = 28, opt2 = 5, result = 28, stk\_data\_in = z, stk\_push = 0, stk\_pop = 0, stk\_data\_out = 28 z\_flag = 0, s\_flag = 0,  825 [STACK] rstn = 1, stack\_ptr = 1, top = 28, data\_in = z, push = 0, pop = 1, data\_out = 28, full = 0, empty = 0  825 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0110, addr\_const = x, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 28, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 28  825 [PC::0] rstn = 1, en = 0, control\_bus = 0110, pc = 7, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 28  825 [ALU:: 5] rstn = 1, control\_bus = 0110, opt1 = 28, opt2 = 5, result = 28, stk\_data\_in = z, stk\_push = 0, stk\_pop = 1, stk\_data\_out = 28 z\_flag = 0, s\_flag = 0,  835 [STACK] rstn = 1, stack\_ptr = 0, top = x, data\_in = z, push = 0, pop = 0, data\_out = 28, full = 0, empty = 0  835 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0110, addr\_const = x, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 28, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 28  835 [PC::0] rstn = 1, en = 0, control\_bus = 0110, pc = 7, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 28  835 [ALU:: 6] rstn = 1, control\_bus = 0110, opt1 = 28, opt2 = 5, result = 28, stk\_data\_in = z, stk\_push = 0, stk\_pop = 0, stk\_data\_out = 28 z\_flag = 0, s\_flag = 0,  835 [STACK] rstn = 1, stack\_ptr = 0, top = x, data\_in = z, push = 0, pop = 0, data\_out = 28, full = 0, empty = 1  845 [ALU:: 7] rstn = 1, control\_bus = 0110, opt1 = 28, opt2 = 28, result = 28, stk\_data\_in = z, stk\_push = 0, stk\_pop = 0, stk\_data\_out = 28 z\_flag = 0, s\_flag = 0,  855 [ALU:: 8] rstn = 1, control\_bus = 0110, opt1 = 28, opt2 = 28, result = 56, stk\_data\_in = z, stk\_push = 0, stk\_pop = 0, stk\_data\_out = 28 z\_flag = 0, s\_flag = 0,  865 [STACK] rstn = 1, stack\_ptr = 0, top = x, data\_in = 56, push = 1, pop = 0, data\_out = 28, full = 0, empty = 1  865 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0110, addr\_const = x, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 28, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 28  865 [ALU:: 9] rstn = 1, control\_bus = 0110, opt1 = 28, opt2 = 28, result = 56, stk\_data\_in = 56, stk\_push = 1, stk\_pop = 0, stk\_data\_out = 28 z\_flag = 0, s\_flag = 0,  875 [STACK] rstn = 1, stack\_ptr = 1, top = 56, data\_in = 56, push = 0, pop = 0, data\_out = 28, full = 0, empty = 1  875 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0110, addr\_const = x, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 28, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 28  875 [ALU:: 0] rstn = 1, control\_bus = 0110, opt1 = 28, opt2 = 28, result = 56, stk\_data\_in = 56, stk\_push = 0, stk\_pop = 0, stk\_data\_out = 28 z\_flag = 0, s\_flag = 0,  875 [STACK] rstn = 1, stack\_ptr = 1, top = 56, data\_in = 56, push = 0, pop = 0, data\_out = 28, full = 0, empty = 0  875 [CPU::3] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 1  885 [STACK] rstn = 1, stack\_ptr = 1, top = 56, data\_in = z, push = z, pop = z, data\_out = 28, full = 0, empty = 0  885 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0110, addr\_const = x, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 28, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 28  885 [ALU:: 0] rstn = 1, control\_bus = 0110, opt1 = 28, opt2 = 28, result = 56, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 28 z\_flag = 0, s\_flag = 0,  885 [PC::0] rstn = 1, en = 0, control\_bus = 0110, pc = 7, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 28  885 [CPU::0] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 1  885 [CPU::0] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  895 [CPU::1] ifidc\_fetch = 1, exec\_en = 0, exec\_finish = 0  895 [IFIDC::0] pc = 7 control\_bus = 0110 addr\_imm = x IS\_ready = 0 rstn = 1 en = 1  905 [IFIDC::1] pc = 7 control\_bus = 0110 addr\_imm = x IS\_ready = 0 rstn = 1 en = 0  905 [CPU::1] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  915 [IFIDC::2] pc = 7 control\_bus = 0110 addr\_imm = x IS\_ready = 0 rstn = 1 en = 0  925 [IFIDC::0] pc = 7 control\_bus = 0000 addr\_imm = 12 IS\_ready = 1 rstn = 1 en = 0  925 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0000, addr\_const = 12, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 28, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 28  925 [PC::0] rstn = 1, en = 0, control\_bus = 0000, pc = 7, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 28  925 [ALU:: 0] rstn = 1, control\_bus = 0000, opt1 = 28, opt2 = 28, result = 56, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 28 z\_flag = 0, s\_flag = 0,  935 [CPU::2] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  935 [IFIDC::0] pc = 7 control\_bus = 0000 addr\_imm = 12 IS\_ready = 0 rstn = 1 en = 0  945 [CPU::3] ifidc\_fetch = 0, exec\_en = 1, exec\_finish = 0  945 [WBPB::0] rstn = 1, en = 1, fin\_sig = 0, control\_bus = 0000, addr\_const = 12, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 28, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 28  945 [PC::0] rstn = 1, en = 1, control\_bus = 0000, pc = 7, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 28  955 [CPU::3] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  955 [WBPB::1] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0000, addr\_const = 12, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 28, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 28  955 [PC::2] rstn = 1, en = 0, control\_bus = 0000, pc = 7, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 28  965 [MEMORY] r\_en = 0, addr = z, w\_en = 0, data\_in = zzzzzzzz, data\_out = 00011100  965 [WBPB::2] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0000, addr\_const = 12, stk\_data\_in = 12, stk\_push = z, stk\_pop = 0, stk\_data\_out = 28, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = 0, mem\_w\_en = 0, mem\_data\_out = 28  965 [IFIDC::0] pc = 8 control\_bus = 0000 addr\_imm = 12 IS\_ready = 0 rstn = 1 en = 0  965 [PC::0] rstn = 1, en = 0, control\_bus = 0000, pc = 8, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 28  975 [WBPB::0] rstn = 1, en = 0, fin\_sig = 1, control\_bus = 0000, addr\_const = 12, stk\_data\_in = 12, stk\_push = 1, stk\_pop = 0, stk\_data\_out = 28, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = 0, mem\_w\_en = 0, mem\_data\_out = 28  975 [CPU::3] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 1  985 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0000, addr\_const = 12, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 28, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 28  985 [MEMORY] r\_en = z, addr = z, w\_en = z, data\_in = zzzzzzzz, data\_out = 00011100  985 [CPU::0] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 1  985 [STACK] rstn = 1, stack\_ptr = 2, top = 12, data\_in = z, push = z, pop = z, data\_out = 28, full = 0, empty = 0  985 [CPU::0] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  995 [CPU::1] ifidc\_fetch = 1, exec\_en = 0, exec\_finish = 0  995 [IFIDC::0] pc = 8 control\_bus = 0000 addr\_imm = 12 IS\_ready = 0 rstn = 1 en = 1  1005 [IFIDC::1] pc = 8 control\_bus = 0000 addr\_imm = 12 IS\_ready = 0 rstn = 1 en = 0  1005 [CPU::1] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  1015 [IFIDC::2] pc = 8 control\_bus = 0000 addr\_imm = 12 IS\_ready = 0 rstn = 1 en = 0  1025 [IFIDC::0] pc = 8 control\_bus = 0111 addr\_imm = x IS\_ready = 1 rstn = 1 en = 0  1025 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0111, addr\_const = x, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 28, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 28  1025 [PC::0] rstn = 1, en = 0, control\_bus = 0111, pc = 8, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 28  1025 [ALU:: 0] rstn = 1, control\_bus = 0111, opt1 = 28, opt2 = 28, result = 56, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 28 z\_flag = 0, s\_flag = 0,  1035 [CPU::2] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  1035 [IFIDC::0] pc = 8 control\_bus = 0111 addr\_imm = x IS\_ready = 0 rstn = 1 en = 0  1045 [CPU::3] ifidc\_fetch = 0, exec\_en = 1, exec\_finish = 0  1045 [WBPB::0] rstn = 1, en = 1, fin\_sig = 0, control\_bus = 0111, addr\_const = x, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 28, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 28  1045 [PC::0] rstn = 1, en = 1, control\_bus = 0111, pc = 8, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 28  1055 [CPU::3] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  1055 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0111, addr\_const = x, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 28, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 28  1055 [PC::2] rstn = 1, en = 0, control\_bus = 0111, pc = 8, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 28  1055 [ALU:: 1] rstn = 1, control\_bus = 0111, opt1 = 28, opt2 = 28, result = 56, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 28 z\_flag = 0, s\_flag = 0,  1065 [STACK] rstn = 1, stack\_ptr = 2, top = 12, data\_in = z, push = 0, pop = 1, data\_out = 28, full = 0, empty = 0  1065 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0111, addr\_const = x, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 28, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 28  1065 [ALU:: 2] rstn = 1, control\_bus = 0111, opt1 = 28, opt2 = 28, result = 56, stk\_data\_in = z, stk\_push = 0, stk\_pop = 1, stk\_data\_out = 28 z\_flag = 0, s\_flag = 0,  1065 [PC::0] rstn = 1, en = 0, control\_bus = 0111, pc = 9, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 28  1065 [IFIDC::0] pc = 9 control\_bus = 0111 addr\_imm = x IS\_ready = 0 rstn = 1 en = 0  1075 [STACK] rstn = 1, stack\_ptr = 1, top = 56, data\_in = z, push = 0, pop = 0, data\_out = 12, full = 0, empty = 0  1075 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0111, addr\_const = x, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 12, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 28  1075 [PC::0] rstn = 1, en = 0, control\_bus = 0111, pc = 9, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 12  1075 [ALU:: 3] rstn = 1, control\_bus = 0111, opt1 = 28, opt2 = 28, result = 56, stk\_data\_in = z, stk\_push = 0, stk\_pop = 0, stk\_data\_out = 12 z\_flag = 0, s\_flag = 0,  1085 [ALU:: 4] rstn = 1, control\_bus = 0111, opt1 = 12, opt2 = 28, result = 56, stk\_data\_in = z, stk\_push = 0, stk\_pop = 0, stk\_data\_out = 12 z\_flag = 0, s\_flag = 0,  1095 [STACK] rstn = 1, stack\_ptr = 1, top = 56, data\_in = z, push = 0, pop = 1, data\_out = 12, full = 0, empty = 0  1095 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0111, addr\_const = x, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 12, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 28  1095 [PC::0] rstn = 1, en = 0, control\_bus = 0111, pc = 9, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 12  1095 [ALU:: 5] rstn = 1, control\_bus = 0111, opt1 = 12, opt2 = 28, result = 56, stk\_data\_in = z, stk\_push = 0, stk\_pop = 1, stk\_data\_out = 12 z\_flag = 0, s\_flag = 0,  1105 [STACK] rstn = 1, stack\_ptr = 0, top = x, data\_in = z, push = 0, pop = 0, data\_out = 56, full = 0, empty = 0  1105 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0111, addr\_const = x, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 56, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 28  1105 [PC::0] rstn = 1, en = 0, control\_bus = 0111, pc = 9, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 56  1105 [ALU:: 6] rstn = 1, control\_bus = 0111, opt1 = 12, opt2 = 28, result = 56, stk\_data\_in = z, stk\_push = 0, stk\_pop = 0, stk\_data\_out = 56 z\_flag = 0, s\_flag = 0,  1105 [STACK] rstn = 1, stack\_ptr = 0, top = x, data\_in = z, push = 0, pop = 0, data\_out = 56, full = 0, empty = 1  1115 [ALU:: 7] rstn = 1, control\_bus = 0111, opt1 = 12, opt2 = 56, result = 56, stk\_data\_in = z, stk\_push = 0, stk\_pop = 0, stk\_data\_out = 56 z\_flag = 0, s\_flag = 0,  1125 [ALU:: 8] rstn = 1, control\_bus = 0111, opt1 = 12, opt2 = 56, result = 44, stk\_data\_in = z, stk\_push = 0, stk\_pop = 0, stk\_data\_out = 56 z\_flag = 0, s\_flag = 0,  1135 [STACK] rstn = 1, stack\_ptr = 0, top = x, data\_in = 44, push = 1, pop = 0, data\_out = 56, full = 0, empty = 1  1135 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0111, addr\_const = x, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 56, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 28  1135 [ALU:: 9] rstn = 1, control\_bus = 0111, opt1 = 12, opt2 = 56, result = 44, stk\_data\_in = 44, stk\_push = 1, stk\_pop = 0, stk\_data\_out = 56 z\_flag = 0, s\_flag = 0,  1145 [STACK] rstn = 1, stack\_ptr = 1, top = 44, data\_in = 44, push = 0, pop = 0, data\_out = 56, full = 0, empty = 1  1145 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0111, addr\_const = x, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 56, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 28  1145 [ALU:: 0] rstn = 1, control\_bus = 0111, opt1 = 12, opt2 = 56, result = 44, stk\_data\_in = 44, stk\_push = 0, stk\_pop = 0, stk\_data\_out = 56 z\_flag = 0, s\_flag = 0,  1145 [STACK] rstn = 1, stack\_ptr = 1, top = 44, data\_in = 44, push = 0, pop = 0, data\_out = 56, full = 0, empty = 0  1145 [CPU::3] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 1  1155 [CPU::0] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 1  1155 [STACK] rstn = 1, stack\_ptr = 1, top = 44, data\_in = z, push = z, pop = z, data\_out = 56, full = 0, empty = 0  1155 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0111, addr\_const = x, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 56, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 28  1155 [ALU:: 0] rstn = 1, control\_bus = 0111, opt1 = 12, opt2 = 56, result = 44, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 56 z\_flag = 0, s\_flag = 0,  1155 [PC::0] rstn = 1, en = 0, control\_bus = 0111, pc = 9, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 56  1155 [CPU::0] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  1165 [CPU::1] ifidc\_fetch = 1, exec\_en = 0, exec\_finish = 0  1165 [IFIDC::0] pc = 9 control\_bus = 0111 addr\_imm = x IS\_ready = 0 rstn = 1 en = 1  1175 [CPU::1] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  1175 [IFIDC::1] pc = 9 control\_bus = 0111 addr\_imm = x IS\_ready = 0 rstn = 1 en = 0  1185 [IFIDC::2] pc = 9 control\_bus = 0111 addr\_imm = x IS\_ready = 0 rstn = 1 en = 0  1195 [IFIDC::0] pc = 9 control\_bus = 0010 addr\_imm = 2 IS\_ready = 1 rstn = 1 en = 0  1195 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0010, addr\_const = 2, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 56, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 28  1195 [PC::0] rstn = 1, en = 0, control\_bus = 0010, pc = 9, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 56  1195 [ALU:: 0] rstn = 1, control\_bus = 0010, opt1 = 12, opt2 = 56, result = 44, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 56 z\_flag = 0, s\_flag = 0,  1205 [IFIDC::0] pc = 9 control\_bus = 0010 addr\_imm = 2 IS\_ready = 0 rstn = 1 en = 0  1205 [CPU::2] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  1215 [CPU::3] ifidc\_fetch = 0, exec\_en = 1, exec\_finish = 0  1215 [WBPB::0] rstn = 1, en = 1, fin\_sig = 0, control\_bus = 0010, addr\_const = 2, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 56, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 28  1215 [PC::0] rstn = 1, en = 1, control\_bus = 0010, pc = 9, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 56  1225 [WBPB::5] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0010, addr\_const = 2, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 56, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 28  1225 [PC::2] rstn = 1, en = 0, control\_bus = 0010, pc = 9, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 56  1225 [CPU::3] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  1235 [IFIDC::0] pc = 10 control\_bus = 0010 addr\_imm = 2 IS\_ready = 0 rstn = 1 en = 0  1235 [PC::0] rstn = 1, en = 0, control\_bus = 0010, pc = 10, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 56  1235 [WBPB::6] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0010, addr\_const = 2, stk\_data\_in = z, stk\_push = 0, stk\_pop = 1, stk\_data\_out = 56, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 28  1245 [WBPB::7] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0010, addr\_const = 2, stk\_data\_in = z, stk\_push = 0, stk\_pop = 1, stk\_data\_out = 44, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 28  1245 [STACK] rstn = 1, stack\_ptr = 0, top = x, data\_in = z, push = 0, pop = 1, data\_out = 44, full = 0, empty = 0  1245 [PC::0] rstn = 1, en = 0, control\_bus = 0010, pc = 10, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 44  1245 [ALU:: 0] rstn = 1, control\_bus = 0010, opt1 = 12, opt2 = 56, result = 44, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 44 z\_flag = 0, s\_flag = 0,  1245 [STACK] rstn = 1, stack\_ptr = 0, top = x, data\_in = z, push = 0, pop = 1, data\_out = 44, full = 0, empty = 1  1255 [MEMORY] r\_en = 0, addr = 2, w\_en = 1, data\_in = 00101100, data\_out = 00011100  1255 [WBPB::0] rstn = 1, en = 0, fin\_sig = 1, control\_bus = 0010, addr\_const = 2, stk\_data\_in = z, stk\_push = 0, stk\_pop = 1, stk\_data\_out = 44, mem\_data\_in = 44, mem\_addr = 2, mem\_r\_en = 0, mem\_w\_en = 1, mem\_data\_out = 28  1255 [CPU::3] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 1  1265 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 0010, addr\_const = 2, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 44, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 28  1265 [MEMORY] r\_en = z, addr = z, w\_en = z, data\_in = zzzzzzzz, data\_out = 00011100  1265 [CPU::0] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 1  1265 [CPU::0] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  1275 [CPU::1] ifidc\_fetch = 1, exec\_en = 0, exec\_finish = 0  1275 [IFIDC::0] pc = 10 control\_bus = 0010 addr\_imm = 2 IS\_ready = 0 rstn = 1 en = 1  1285 [IFIDC::1] pc = 10 control\_bus = 0010 addr\_imm = 2 IS\_ready = 0 rstn = 1 en = 0  1285 [CPU::1] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  1295 [IFIDC::2] pc = 10 control\_bus = 0010 addr\_imm = 2 IS\_ready = 0 rstn = 1 en = 0  1305 [IFIDC::0] pc = 10 control\_bus = 1111 addr\_imm = x IS\_ready = 1 rstn = 1 en = 0  1305 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 1111, addr\_const = x, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 44, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 28  1305 [PC::0] rstn = 1, en = 0, control\_bus = 1111, pc = 10, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 44  1305 [ALU:: 0] rstn = 1, control\_bus = 1111, opt1 = 12, opt2 = 56, result = 44, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 44 z\_flag = 0, s\_flag = 0,  1315 [CPU::2] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  1315 [IFIDC::0] pc = 10 control\_bus = 1111 addr\_imm = x IS\_ready = 0 rstn = 1 en = 0  1325 [CPU::3] ifidc\_fetch = 0, exec\_en = 1, exec\_finish = 0  1325 [WBPB::0] rstn = 1, en = 1, fin\_sig = 0, control\_bus = 1111, addr\_const = x, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 44, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 28  1325 [PC::0] rstn = 1, en = 1, control\_bus = 1111, pc = 10, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 44  1335 [CPU::3] ifidc\_fetch = 0, exec\_en = 0, exec\_finish = 0  1335 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 1111, addr\_const = x, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 44, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 28  1335 [PC::4] rstn = 1, en = 0, control\_bus = 1111, pc = 10, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 44  1335 [ALU:: 1] rstn = 1, control\_bus = 1111, opt1 = 12, opt2 = 56, result = 44, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 44 z\_flag = 0, s\_flag = 0, WARNING: PC.v:98: $writememb: Standard inconsistency, following 1364-2005.  1345 [STACK] rstn = 1, stack\_ptr = 0, top = x, data\_in = z, push = 0, pop = 1, data\_out = 44, full = 0, empty = 1  1345 [WBPB::0] rstn = 1, en = 0, fin\_sig = 0, control\_bus = 1111, addr\_const = x, stk\_data\_in = z, stk\_push = z, stk\_pop = z, stk\_data\_out = 44, mem\_data\_in = z, mem\_addr = z, mem\_r\_en = z, mem\_w\_en = z, mem\_data\_out = 28  1345 [ALU:: 2] rstn = 1, control\_bus = 1111, opt1 = 12, opt2 = 56, result = 44, stk\_data\_in = z, stk\_push = 0, stk\_pop = 1, stk\_data\_out = 44 z\_flag = 0, s\_flag = 0,  1345 [PC::4] rstn = 1, en = 0, control\_bus = 1111, pc = 10, z\_flag = 0, s\_flag = 0, stk\_pop = z, stk\_data\_out = 44 |

شکل موجها به شرح زیر است:

IFIDC\_EXEC



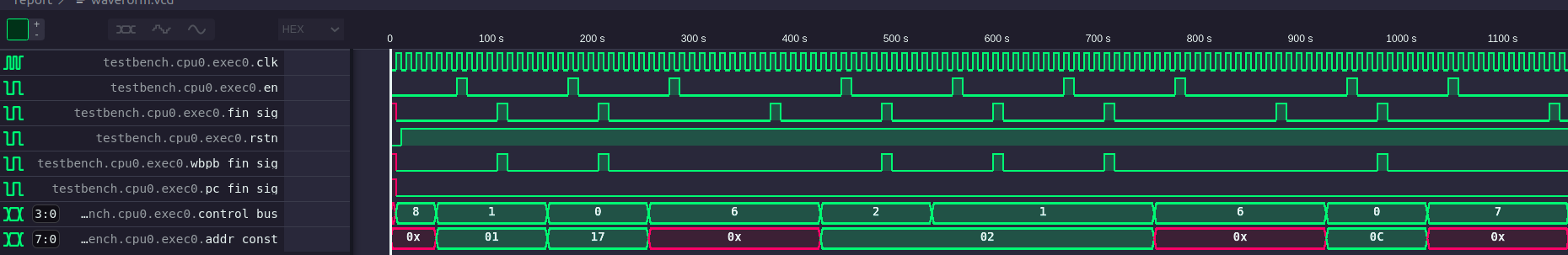
ALU



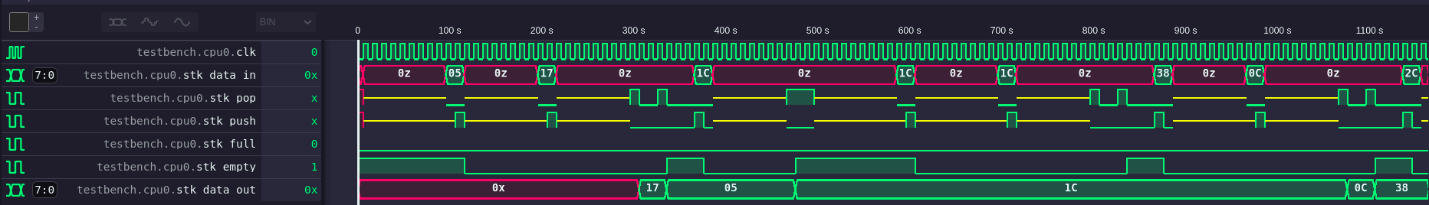
Memory

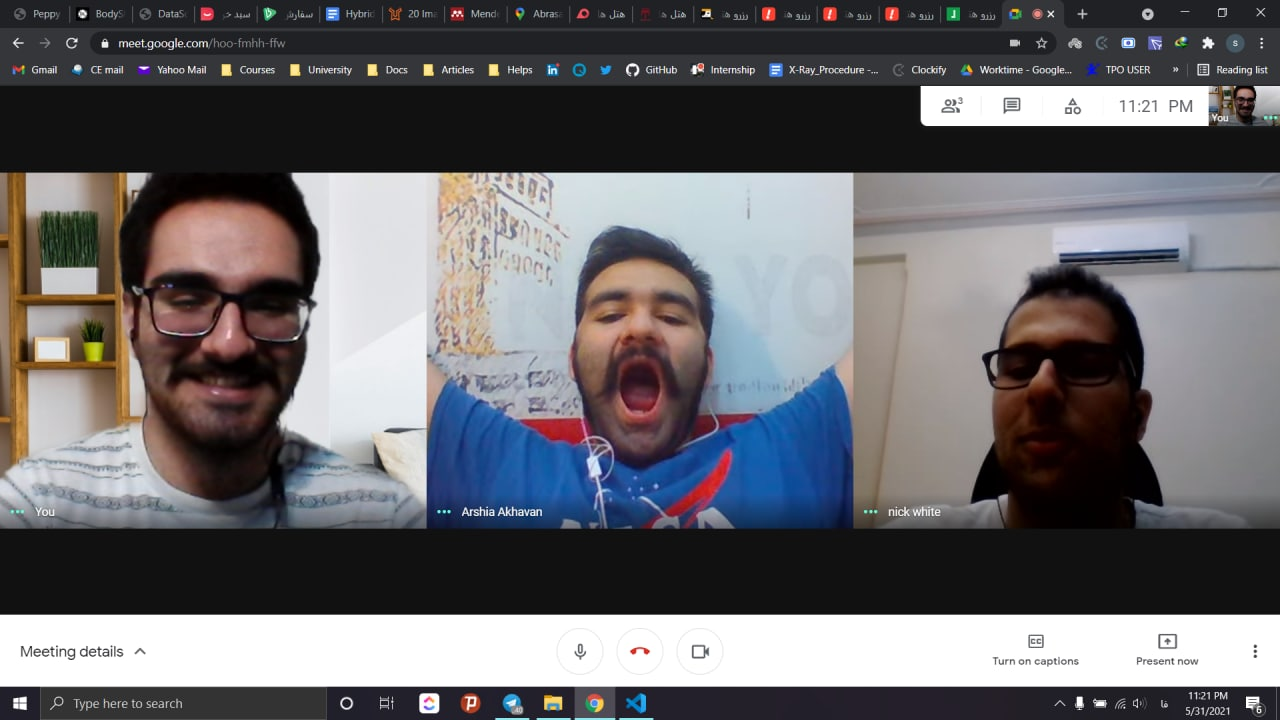


PP\_PC



Stack





پایان :)