实验报告七
PB14000556 陈晓彤

实验题目：设计一个流水线 CPU 并成功运行测试程序
实验要求：
基本要求
0 分（百分制），自行设计一个多周期 MIPS CPU，包含 16 条指令，指令集见下页。
扩展要求
在 16 条的基础上，每增加 1 条指令，+1 分
实现中断功能，+6~10 分
实现流水，+12~20 分
实现下载，+6~10 分

设计思路：
计划实现流水线，和 40 条左右的指令数
基本流水线实现分五段：IF(Instruction Fetch) ID(Instruction Decode) EX(execute) MEM(memory) WB(write back) 和段寄存器 IF-ID ID-EX EX-MEM MEM-WB 来保存一些控制信号
另外，为实现乘法指令的结果保存，需要加 hilo 模块，而且为保证消除数据、指令相关等以保证流水线运行正确，需要加 ctrl 模块产生 stall 信号；ld 指令的数据相关还需另外处理，所以要加上 delayslot 保存指令
为缩短调试时间，不采用 ip 核实现指令和数据存储器，而直接使用模块仿真（模块中用寄存器组实现）

源代码：

 “header.v”
```
//全局
`define RstEnable 1'b1
`define RstDisable 1'b0
`define ZeroWord 32'h00000000
`define WriteEnable 1'b1
`define WriteDisable 1'b0
`define ReadEnable 1'b1
`define ReadDisable 1'b0
`define AluOpBus 7:0
`define AluSelBus 2:0
`define InstValid 1'b0
`define InstInvalid 1'b1
`define Stop 1'b1
`define NoStop 1'b0
`define InDelaySlot 1'b1
`define NotInDelaySlot 1'b0
`define Branch 1'b1
```

```
`define NotBranch 1'b0
`define True_v 1'b1
`define False_v 1'b0
`define ChipEnable 1'b1
`define ChipDisable 1'b0


//指令
`define EXE_AND    6'b100100
`define EXE_OR     6'b100101
`define EXE_XOR 6'b100110
`define EXE_NOR 6'b100111
`define EXE_ANDI 6'b001100
`define EXE_ORI    6'b001101
`define EXE_XORI 6'b001110
`define EXE_LUI 6'b001111

`define EXE_SLL    6'b000000
`define EXE_SLLV    6'b000100
`define EXE_SRL    6'b000010
`define EXE_SRLV    6'b000110
`define EXE_SRA    6'b000011
`define EXE_SRAV    6'b000111
`define EXE_SYNC    6'b001111
`define EXE_PREF    6'b110011

`define EXE_MOVZ    6'b001010
`define EXE_MOVN    6'b001011
`define EXE_MFHI    6'b010000
`define EXE_MTHI    6'b010001
`define EXE_MFLO    6'b010010
`define EXE_MTLO    6'b010011

`define EXE_SLT    6'b101010
`define EXE_SLTU    6'b101011
`define EXE_SLTI    6'b001010
`define EXE_SLTIU    6'b001011
`define EXE_ADD    6'b100000
`define EXE_ADDU    6'b100001
`define EXE_SUB    6'b100010
`define EXE_SUBU    6'b100011
`define EXE_ADDI    6'b001000
`define EXE_ADDIU    6'b001001
`define EXE_CLZ    6'b100000
```

```
`define EXE_CLO    6'b100001

`define EXE_MULT    6'b011000
`define EXE_MULTU    6'b011001
`define EXE_MUL    6'b000010
`define EXE_MADD    6'b000000
`define EXE_MADDU    6'b000001
`define EXE_MSUB    6'b000100
`define EXE_MSUBU    6'b000101

`define EXE_J    6'b000010
`define EXE_JAL    6'b000011
`define EXE_JALR    6'b001001
`define EXE_JR    6'b001000
`define EXE_BEQ    6'b000100
`define EXE_BGEZ    5'b00001
`define EXE_BGEZAL    5'b10001
`define EXE_BGTZ    6'b000111
`define EXE_BLEZ    6'b000110
`define EXE_BLTZ    5'b00000
`define EXE_BLTZAL    5'b10000
`define EXE_BNE    6'b000101

`define EXE_LB    6'b100000
`define EXE_LBU    6'b100100
`define EXE_LH    6'b100001
`define EXE_LHU    6'b100101
`define EXE_LL    6'b110000
`define EXE_LW    6'b100011
`define EXE_LWL    6'b100010
`define EXE_LWR    6'b100110
`define EXE_SB    6'b101000
`define EXE_SC    6'b111000
`define EXE_SH    6'b101001
`define EXE_SW    6'b101011
`define EXE_SWL    6'b101010
`define EXE_SWR    6'b101110


`define EXE_NOP 6'b000000
`define SSNOP 32'b00000000000000000000000001000000

`define EXE_SPECIAL_INST 6'b000000
`define EXE_REGIMM_INST 6'b000001
```

```
`define EXE_SPECIAL2_INST 6'b011100

//AluOp
`define EXE_AND_OP      8'b00100100
`define EXE_OR_OP       8'b00100101
`define EXE_XOR_OP    8'b00100110
`define EXE_NOR_OP    8'b00100111
`define EXE_ANDI_OP    8'b01011001
`define EXE_ORI_OP    8'b01011010
`define EXE_XORI_OP    8'b01011011
`define EXE_LUI_OP    8'b01011100

`define EXE_SLL_OP    8'b01111100
`define EXE_SLLV_OP    8'b00000100
`define EXE_SRL_OP    8'b00000010
`define EXE_SRLV_OP    8'b00000110
`define EXE_SRA_OP    8'b00000011
`define EXE_SRAV_OP    8'b00000111

`define EXE_MOVZ_OP    8'b00001010
`define EXE_MOVN_OP    8'b00001011
`define EXE_MFHI_OP    8'b00010000
`define EXE_MTHI_OP    8'b00010001
`define EXE_MFLO_OP    8'b00010010
`define EXE_MTLO_OP    8'b00010011

`define EXE_SLT_OP    8'b00101010
`define EXE_SLTU_OP    8'b00101011
`define EXE_SLTI_OP    8'b01010111
`define EXE_SLTIU_OP    8'b01011000
`define EXE_ADD_OP    8'b00100000
`define EXE_ADDU_OP    8'b00100001
`define EXE_SUB_OP    8'b00100010
`define EXE_SUBU_OP    8'b00100011
`define EXE_ADDI_OP    8'b01010101
`define EXE_ADDIU_OP    8'b01010110
`define EXE_CLZ_OP    8'b10110000
`define EXE_CLO_OP    8'b10110001

`define EXE_MULT_OP    8'b00011000
`define EXE_MULTU_OP    8'b00011001
`define EXE_MUL_OP    8'b10101001
`define EXE_MADD_OP    8'b10100110
`define EXE_MADDU_OP    8'b10101000
```

```
`define EXE_MSUB_OP    8'b10101010
`define EXE_MSUBU_OP    8'b10101011

`define EXE_J_OP    8'b01001111
`define EXE_JAL_OP    8'b01010000
`define EXE_JALR_OP    8'b00001001
`define EXE_JR_OP    8'b00001000
`define EXE_BEQ_OP    8'b01010001
`define EXE_BGEZ_OP    8'b01000001
`define EXE_BGEZAL_OP    8'b01001011
`define EXE_BGTZ_OP    8'b01010100
`define EXE_BLEZ_OP    8'b01010011
`define EXE_BLTZ_OP    8'b01000000
`define EXE_BLTZAL_OP    8'b01001010
`define EXE_BNE_OP    8'b01010010

`define EXE_LB_OP    8'b11100000
`define EXE_LBU_OP    8'b11100100
`define EXE_LH_OP    8'b11100001
`define EXE_LHU_OP    8'b11100101
`define EXE_LL_OP    8'b11110000
`define EXE_LW_OP    8'b11100011
`define EXE_LWL_OP    8'b11100010
`define EXE_LWR_OP    8'b11100110
`define EXE_PREF_OP    8'b11110011
`define EXE_SB_OP    8'b11101000
`define EXE_SC_OP    8'b11111000
`define EXE_SH_OP    8'b11101001
`define EXE_SW_OP    8'b11101011
`define EXE_SWL_OP    8'b11101010
`define EXE_SWR_OP    8'b11101110
`define EXE_SYNC_OP    8'b00001111

`define EXE_NOP_OP       8'b00000000

//AluSel
`define EXE_RES_LOGIC 3'b001
`define EXE_RES_SHIFT 3'b010
`define EXE_RES_MOVE 3'b011
`define EXE_RES_ARITHMETIC 3'b100
`define EXE_RES_MUL 3'b101
`define EXE_RES_JUMP_BRANCH 3'b110
`define EXE_RES_LOAD_STORE 3'b111
```

```
`define EXE_RES_NOP 3'b000


//指令存储器 inst_rom
`define InstAddrBus 31:0
`define InstBus 31:0
`define InstMemNum 131071
`define InstMemNumLog2 17

//数据存储器 data_ram
`define DataAddrBus 31:0
`define DataBus 31:0
`define DataMemNum 131071
`define DataMemNumLog2 17
`define ByteWidth 7:0

//通用寄存器 regfile
`define RegAddrBus 4:0
`define RegBus 31:0
`define RegWidth 32
`define DoubleRegWidth 64
`define DoubleRegBus 63:0
`define RegNum 32
`define RegNumLog2 5
`define NOPRegAddr 5'b00000
```

“cpu.v”

```
`include "header.v"
module cpu(
	input	clk,
	input	rst
);

	wire [`InstAddrBus] pc;
	wire [`InstAddrBus] id_pc_i;
	wire [`InstBus] id_inst_i;
	wire [`RegBus] rom_data_i;
	wire rom_ce_o;
   //连接数据存储器 data_ram
	wire [`RegBus] ram_data_i;
	wire [`RegBus] ram_addr_o;
	wire [`RegBus] ram_data_o;
	wire ram_we_o;
	wire [3:0] ram_sel_o;
```

```
  wire [3:0] ram_ce_o;
  //连接译码阶段 ID 模块的输出与 ID/EX 模块的输入
  wire [`AluOpBus] id_aluop_o;
  wire [`AluSelBus] id_alusel_o;
  wire [`RegBus] id_reg1_o;
  wire [`RegBus] id_reg2_o;
  wire    id_wreg_o;
  wire [`RegAddrBus] id_wd_o;
  wire    id_is_in_delayslot_o;
wire [`RegBus] id_link_address_o;
wire [`RegBus] id_inst_o;

  //连接 ID/EX 模块的输出与执行阶段 EX 模块的输入
  wire [`AluOpBus] ex_aluop_i;
  wire [`AluSelBus] ex_alusel_i;
  wire [`RegBus] ex_reg1_i;
  wire [`RegBus] ex_reg2_i;
  wire    ex_wreg_i;
  wire [`RegAddrBus] ex_wd_i;
  wire    ex_is_in_delayslot_i;
wire [`RegBus] ex_link_address_i;
wire [`RegBus] ex_inst_i;

  //连接执行阶段 EX 模块的输出与 EX/MEM 模块的输入
  wire    ex_wreg_o;
  wire [`RegAddrBus] ex_wd_o;
  wire [`RegBus] ex_wdata_o;
  wire [`RegBus] ex_hi_o;
  wire [`RegBus] ex_lo_o;
  wire    ex_whilo_o;
  wire [`AluOpBus] ex_aluop_o;
  wire [`RegBus] ex_mem_addr_o;
  wire [`RegBus] ex_reg1_o;
  wire [`RegBus] ex_reg2_o;

  //连接 EX/MEM 模块的输出与访存阶段 MEM 模块的输入
  wire    mem_wreg_i;
  wire [`RegAddrBus] mem_wd_i;
  wire [`RegBus] mem_wdata_i;
  wire [`RegBus] mem_hi_i;
  wire [`RegBus] mem_lo_i;
  wire    mem_whilo_i;
  wire [`AluOpBus] mem_aluop_i;
  wire [`RegBus] mem_mem_addr_i;
```

```
   wire [`RegBus] mem_reg1_i;
   wire [`RegBus] mem_reg2_i;

   //连接访存阶段 MEM 模块的输出与 MEM/WB 模块的输入
   wire    mem_wreg_o;
   wire [`RegAddrBus] mem_wd_o;
   wire [`RegBus] mem_wdata_o;
   wire [`RegBus] mem_hi_o;
   wire [`RegBus] mem_lo_o;
   wire    mem_whilo_o;
   wire    mem_LLbit_value_o;
   wire    mem_LLbit_we_o;

   //连接 MEM/WB 模块的输出与回写阶段的输入
   wire wb_wreg_i;
   wire [`RegAddrBus] wb_wd_i;
   wire [`RegBus] wb_wdata_i;
   wire [`RegBus] wb_hi_i;
   wire [`RegBus] wb_lo_i;
   wire wb_whilo_i;
   wire wb_LLbit_value_i;
   wire wb_LLbit_we_i;

   //连接译码阶段 ID 模块与通用寄存器 Regfile 模块
  wire reg1_read;
  wire reg2_read;
  wire [`RegBus] reg1_data;
  wire [`RegBus] reg2_data;
  wire [`RegAddrBus] reg1_addr;
  wire [`RegAddrBus] reg2_addr;

   //连接执行阶段与 hilo 模块的输出，读取 HI、LO 寄存器
   wire [`RegBus] hi;
   wire [`RegBus] lo;

//连接执行阶段与 ex_reg 模块，用于多周期的 MADD、MADDU、MSUB、MSUBU 指令
   wire [`DoubleRegBus] hilo_temp_o;
   wire [1:0] cnt_o;

   wire [`DoubleRegBus] hilo_temp_i;
   wire [1:0] cnt_i;

   wire is_in_delayslot_i;
   wire is_in_delayslot_o;
```

```
    wire next_inst_in_delayslot_o;
    wire id_branch_flag_o;
    wire [`RegBus] branch_target_address;

    wire [5:0] stall;
    wire stallreq_from_id;
    wire stallreq_from_ex;

//pc_reg 例化
    pc_reg pc_reg0(
            .clk(clk),
            .rst(rst),
            .stall(stall),
            .branch_flag_i(id_branch_flag_o),
            .branch_target_address_i(branch_target_address),
            .pc(pc),
            .ce(rom_ce_o)

    );

//指令存储器 inst_rom 例化
    inst_rom inst_rom0(
            .ce(rom_ce_o),
            .addr(pc),
            .inst(rom_data_i)
    );

//IF/ID 模块例化
    if_id if_id0(
            .clk(clk),
            .rst(rst),
            .stall(stall),
            .if_pc(pc),
            .if_inst(rom_data_i),
            .id_pc(id_pc_i),
            .id_inst(id_inst_i)
    );

    //译码阶段 ID 模块
    id id0(
            .rst(rst),
            .pc_i(id_pc_i),
            .inst_i(id_inst_i),
            .ex_aluop_i(ex_aluop_o),
            .reg1_data_i(reg1_data),
            .reg2_data_i(reg2_data),

          //处于执行阶段的指令要写入的目的寄存器信息
            .ex_wreg_i(ex_wreg_o),
            .ex_wdata_i(ex_wdata_o),
            .ex_wd_i(ex_wd_o),

          //处于访存阶段的指令要写入的目的寄存器信息
            .mem_wreg_i(mem_wreg_o),
            .mem_wdata_i(mem_wdata_o),
            .mem_wd_i(mem_wd_o),

           .is_in_delayslot_i(is_in_delayslot_i),

            //送到 regfile 的信息
            .reg1_read_o(reg1_read),
            .reg2_read_o(reg2_read),
            .reg1_addr_o(reg1_addr),
            .reg2_addr_o(reg2_addr),

            //送到 ID/EX 模块的信息
            .aluop_o(id_aluop_o),
            .alusel_o(id_alusel_o),
            .reg1_o(id_reg1_o),
            .reg2_o(id_reg2_o),
            .wd_o(id_wd_o),
            .wreg_o(id_wreg_o),
            .inst_o(id_inst_o),

            .next_inst_in_delayslot_o(next_inst_in_delayslot_o),
            .branch_flag_o(id_branch_flag_o),
            .branch_target_address_o(branch_target_address),
            .link_addr_o(id_link_address_o),

            .is_in_delayslot_o(id_is_in_delayslot_o),

            .stallreq(stallreq_from_id)
    );

//通用寄存器 Regfile 例化
    regfile regfile1(
            .clk(clk),
            .rst(rst),
            .we(wb_wreg_i),
            .waddr(wb_wd_i),
            .wdata(wb_wdata_i),
            .re1(reg1_read),
            .raddr1(reg1_addr),
            .rdata1(reg1_data),
            .re2(reg2_read),
            .raddr2(reg2_addr),
            .rdata2(reg2_data)
    );

    //ID/EX 模块
    id_ex id_ex0(
            .clk(clk),
            .rst(rst),

            .stall(stall),

            //从译码阶段 ID 模块传递的信息
            .id_aluop(id_aluop_o),
            .id_alusel(id_alusel_o),
            .id_reg1(id_reg1_o),
            .id_reg2(id_reg2_o),
            .id_wd(id_wd_o),
            .id_wreg(id_wreg_o),
            .id_link_address(id_link_address_o),
            .id_is_in_delayslot(id_is_in_delayslot_o),
            .next_inst_in_delayslot_i(next_inst_in_delayslot_o),
            .id_inst(id_inst_o),

            //传递到执行阶段 EX 模块的信息
            .ex_aluop(ex_aluop_i),
            .ex_alusel(ex_alusel_i),
            .ex_reg1(ex_reg1_i),
            .ex_reg2(ex_reg2_i),
            .ex_wd(ex_wd_i),
            .ex_wreg(ex_wreg_i),
            .ex_link_address(ex_link_address_i),
            .ex_is_in_delayslot(ex_is_in_delayslot_i),
            .is_in_delayslot_o(is_in_delayslot_i),
            .ex_inst(ex_inst_i)
    );

    //EX 模块
```

```
ex ex0(
        .rst(rst),

        //送到执行阶段 EX 模块的信息
        .aluop_i(ex_aluop_i),
        .alusel_i(ex_alusel_i),
        .reg1_i(ex_reg1_i),
        .reg2_i(ex_reg2_i),
        .wd_i(ex_wd_i),
        .wreg_i(ex_wreg_i),
        .hi_i(hi),
        .lo_i(lo),
        .inst_i(ex_inst_i),

      .wb_hi_i(wb_hi_i),
      .wb_lo_i(wb_lo_i),
      .wb_whilo_i(wb_whilo_i),
      .mem_hi_i(mem_hi_o),
      .mem_lo_i(mem_lo_o),
      .mem_whilo_i(mem_whilo_o),

      .hilo_temp_i(hilo_temp_i),
      .cnt_i(cnt_i),

      .link_address_i(ex_link_address_i),
        .is_in_delayslot_i(ex_is_in_delayslot_i),

    //EX 模块的输出到 EX/MEM 模块信息
        .wd_o(ex_wd_o),
        .wreg_o(ex_wreg_o),
        .wdata_o(ex_wdata_o),

        .hi_o(ex_hi_o),
        .lo_o(ex_lo_o),
        .whilo_o(ex_whilo_o),

        .hilo_temp_o(hilo_temp_o),
        .cnt_o(cnt_o),

        .aluop_o(ex_aluop_o),
        .mem_addr_o(ex_mem_addr_o),
        .reg2_o(ex_reg2_o),

        .stallreq(stallreq_from_ex)
    );

//EX/MEM 模块
ex_mem ex_mem0(
            .clk(clk),
            .rst(rst),

          .stall(stall),

            //来自执行阶段 EX 模块的信息
            .ex_wd(ex_wd_o),
            .ex_wreg(ex_wreg_o),
            .ex_wdata(ex_wdata_o),
            .ex_hi(ex_hi_o),
            .ex_lo(ex_lo_o),
            .ex_whilo(ex_whilo_o),

            .ex_aluop(ex_aluop_o),
            .ex_mem_addr(ex_mem_addr_o),
            .ex_reg2(ex_reg2_o),

            .hilo_i(hilo_temp_o),
            .cnt_i(cnt_o),

            //送到访存阶段 MEM 模块的信息
            .mem_wd(mem_wd_i),
            .mem_wreg(mem_wreg_i),
            .mem_wdata(mem_wdata_i),
            .mem_hi(mem_hi_i),
            .mem_lo(mem_lo_i),
            .mem_whilo(mem_whilo_i),

            .mem_aluop(mem_aluop_i),
            .mem_mem_addr(mem_mem_addr_i),
            .mem_reg2(mem_reg2_i),

            .hilo_o(hilo_temp_i),
            .cnt_o(cnt_i)

    );

//MEM 模块例化
    mem mem0(
        .rst(rst),

        //来自 EX/MEM 模块的信息
        .wd_i(mem_wd_i),
        .wreg_i(mem_wreg_i),
        .wdata_i(mem_wdata_i),
        .hi_i(mem_hi_i),
        .lo_i(mem_lo_i),
        .whilo_i(mem_whilo_i),

        .aluop_i(mem_aluop_i),
        .mem_addr_i(mem_mem_addr_i),
        .reg2_i(mem_reg2_i),

        //来自 memory 的信息
        .mem_data_i(ram_data_i),

        //送到 MEM/WB 模块的信息
        .wd_o(mem_wd_o),
        .wreg_o(mem_wreg_o),
        .wdata_o(mem_wdata_o),
        .hi_o(mem_hi_o),
        .lo_o(mem_lo_o),
        .whilo_o(mem_whilo_o),

        //送到 memory 的信息
        .mem_addr_o(ram_addr_o),
        .mem_we_o(ram_we_o),
        .mem_sel_o(ram_sel_o),
        .mem_data_o(ram_data_o),
        .mem_ce_o(ram_ce_o)
);

//数据存储器 data_ram 例化
data_ram data_ram0(
        .clk(clk),
        .ce(ram_ce_o),
        .we(ram_we_o),
        .sel(ram_sel_o),
        .addr(ram_addr_o),
        .data_i(ram_data_o),
        .data_o(ram_data_i)
);
```

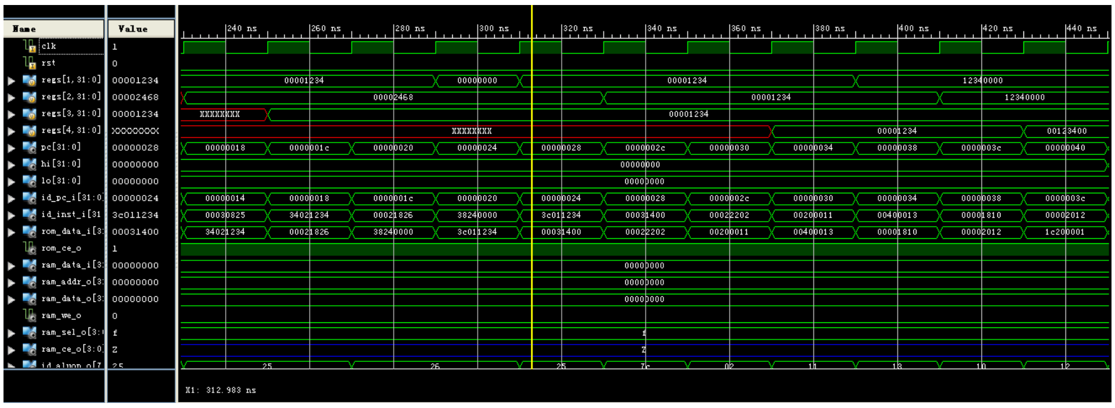
```
//MEM/WB 模块
  mem_wb mem_wb0(
        .clk(clk),
        .rst(rst),

        .stall(stall),

        //来自访存阶段 MEM 模块的信息
        .mem_wd(mem_wd_o),
        .mem_wreg(mem_wreg_o),
        .mem_wdata(mem_wdata_o),
        .mem_hi(mem_hi_o),
        .mem_lo(mem_lo_o),
        .mem_whilo(mem_whilo_o),

        //送到回写阶段的信息
        .wb_wd(wb_wd_i),
        .wb_wreg(wb_wreg_i),
        .wb_wdata(wb_wdata_i),
        .wb_hi(wb_hi_i),
        .wb_lo(wb_lo_i),
        .wb_whilo(wb_whilo_i)

  );
  hilo_reg hilo_reg0(
        .clk(clk),
        .rst(rst),

        //写端口
        .we(wb_whilo_i),
        .hi_i(wb_hi_i),
        .lo_i(wb_lo_i),

        //读端口 1
        .hi_o(hi),
        .lo_o(lo)
  );

  ctrl ctrl0(
        .rst(rst),

        .stallreq_from_id(stallreq_from_id),

  //来自执行阶段的暂停请求
```

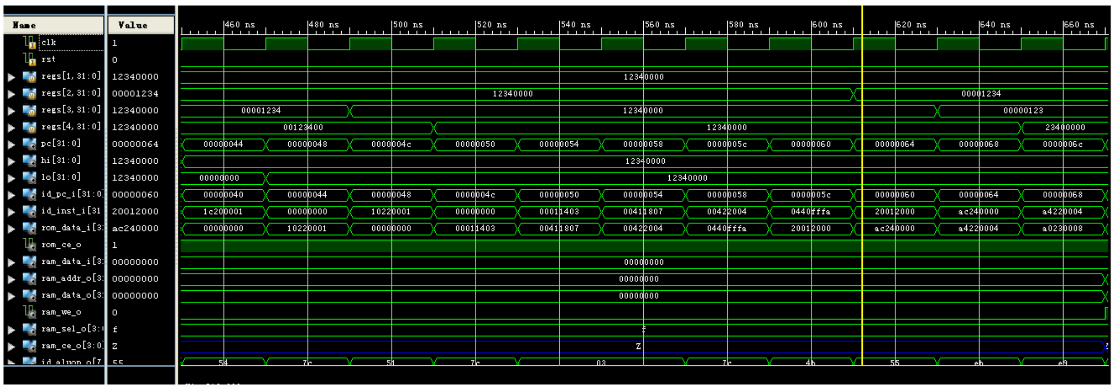
```
            .stallreq_from_ex(stallreq_from_ex),

            .stall(stall)
        );

endmodule
```

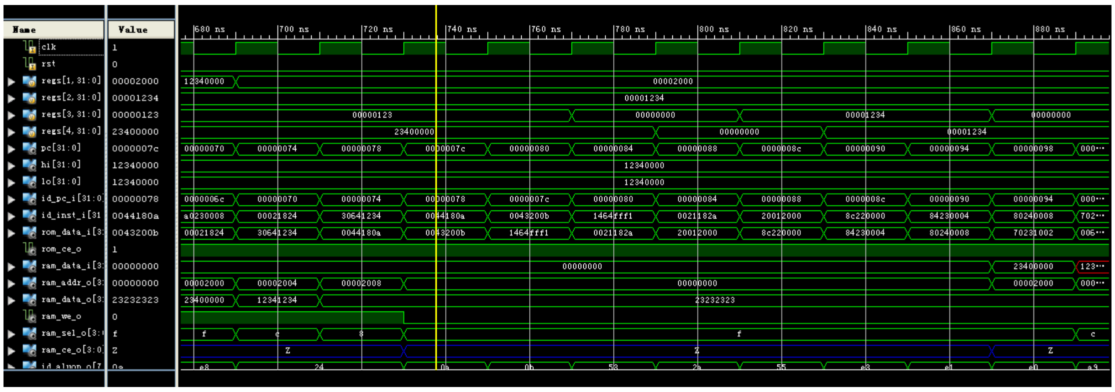
其余文件内容过多，不在此一一显示，详见工程文件

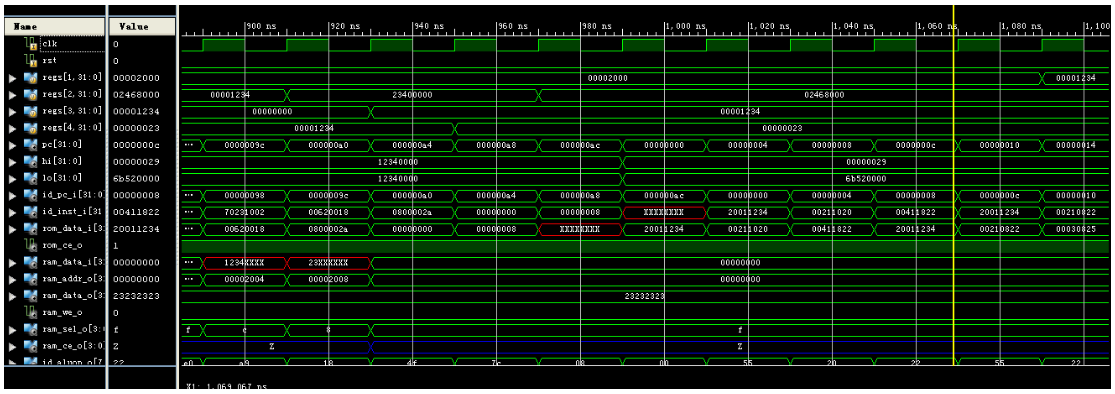
仿真结果

具体情况见仿真说明文件









Name	Value
clk	0
rst	0
regs[1,31:0]	00002000
regs[2,31:0]	02468000
regs[3,31:0]	00001234
regs[4,31:0]	00000023
pc[31:0]	0000000c
hi[31:0]	00000029
lo[31:0]	6b520000
id_pc_i[31:0]	00000008
id_inst_i[31	00411822
rom_data_i[3:	20011234
rom_ce_o	1
ram_data_i[3:	00000000
ram_addr_o[3:	00000000
ram_data_o[3:	23232323
ram_we_o	0
ram_sel_o[3:	f
ram_ce_o[3:0]	Z
id_aluop_o[7	22