中国科学院大学计算机组成原理实验课

实 验 报 告

学号：2016K8009915009　姓名：钟赟　专业：计算机科学与技术

实验序号：__ 实验名称：复杂处理器设计

一、 逻辑电路结构与仿真波形的截图及说明（比如关键 RTL 代码段{包含注释}及其对应的逻辑电路结构、相应信号的仿真波形和信号变化的说明等）

1. 关键代码段

mips_cpu 修改部分（主要修改为增加一个状态，以及跳转指令和寄存器写使能信号的修改）

```
//different states
parameter IF = 3'd0,
          IW = 3'd1,
          ID = 3'd2,
          LD = 3'd3,
          RDW = 3'd4,
          WB = 3'd5,
          ST = 3'd6,
          EX = 3'd7;

always @ (posedge clk) begin
  if(rst)
    cycle_cnt <= 32'd0;
  else
    cycle_cnt <= cycle_cnt + 32'd1;
end
assign mips_perf_cnt_0 = cycle_cnt;


//skip of state
always@(posedge clk)  begin
  if(rst)
    state <= IF;
  else
    state <= next_state;
end

//get next state depending on current state
always @(*) begin
```

```
  case(state)
    IF: next_state = (Inst_Req_Ack & Inst_Req_Valid) ? IW : IF;
    IW: next_state = (Inst_Valid & Inst_Ack) ? ID : IW;
    ID: next_state = EX;
    EX: begin
      if(store)
        next_state = ST;
      else if(load)
        next_state = LD;
      else if(PCsrc | jump | jal)
        next_state = IF;
      else
        next_state = WB;
    end
    LD: next_state = (Mem_Req_Ack & MemRead) ? RDW : LD;
    RDW: next_state = (Read_data_Valid & Read_data_Ack) ? WB : RDW;
    WB: next_state = IF;
    ST: next_state = (Mem_Req_Ack & MemWrite) ? IF : ST;
  endcase
end

//depending on current state, output different registers' signals
always @(posedge clk) begin
  if(rst) begin
    PC <= 32'd0;
    Inst_Req_Valid <= 0;
    Inst_Ack <= 1;
    MemRead <= 0;
    MemWrite <= 0;
    Read_data_Ack <= 0;
  end
  else begin
  case(state)
    IF: Inst_Req_Valid <= Inst_Req_Ack ? 1'b0 : 1'b1;
    IW: Inst_Ack <= Inst_Valid ? 1'b0 : 1'b1;

    EX: begin
      if(jr | jalr)
        PC[31:2] <= rdata1[31:2];
      else if(jump | jal)
        PC[31:2] <= {PC[31:28], Instruction1[25:0]};
      else
        PC <= (PCsrc) ? ( PC + 4 + immediate_32*4) : (PC + 4);
    end
    ST: MemWrite <= Mem_Req_Ack ? 1'b0 : 1'b1;
    LD: MemRead <= Mem_Req_Ack ? 1'b0 : 1'b1;
    RDW: Read_data_Ack <= Read_data_Valid ? 1'b0 : 1'b1;
  endcase
  end
end

//store current instruction
always @(posedge clk) begin
  if(Inst_Valid && Inst_Ack)
    Instruction1 <= Instruction;
  else
    Instruction1 <= Instruction1;
end

//store current data read from memory
always@(posedge clk) begin
  if(Read_data_Ack && Read_data_Valid)
    Read_data1 <= Read_data;
  else
    Read_data1 <= Read_data1;
end

//write back in 31st register when jal
assign wen = (state == WB) ? ((jal | jalr) ? 1'b0 : RegWrite) : ((jal | jalr) && state == EX);
```

benchmark 的修改:

perf_cnt.c

```c
#include "perf_cnt.h"

//Initial the address of cycle counter using volatile key word
volatile unsigned int *mips_perf_cnt_0 = (void *)0x40020000;

unsigned long _uptime() {
  // TODO [COD]
  //   You can use this function to access performance counter related with time or cycle.
       return *(mips_perf_cnt_0);
  return 0;
}

void bench_prepare(Result *res) {
  // TODO [COD]
  //   Add preprocess code, record performance counters' initial states.
  //   You can communicate between bench_prepare() and bench_done() through
  //   static variables or add additional fields in `struct Result`
  res->msec = _uptime();
}

void bench_done(Result *res) {
  // TODO [COD]
  //  Add postprocess code, record performance counters' current states.
  res->msec = _uptime() - res->msec;
}
```

2. 仿真波形和逻辑电路结构与 project4 类似，故不再赘述。

二、 实验过程中遇到的问题、对问题的思考过程及解决方法（比如 RTL 代码中出现的逻辑 bug，仿真、本地上板及云平台调试过程中的难点等）

1. 上板后发现只有 advanced:04 recursion 不能通过，通过对比 benchmark 反汇编发现只有 recursion 用了 JALR 指令，故锁定错误。错误在于 JALR 应该 EX 阶段回写寄存器，而不是在 WB 阶段。

三、 对讲义中思考题（如有）的理解和回答

思考题：如果想在 main()函数中使用当 前 C 源码文件中未定义声明的 bench_prepare()、bench_done()和 printf()，还要做什么？

答：使用未定义声明的 bench_prepare()、bench_done()和 printf()函数，需要提前在 include 头文件里声明，由于所有 benchmark 函数都包含”trap.h”，故我在 medium 和 advanced 组的 include/trap.h 头文件中定义了 Result 结构体，声明了 bench_prepare()、bench_done()和 printf()函数。

```c
#ifndef __TRAP_H__
#define __TRAP_H__

void _halt(int i) {
    extern int global_result;
    global_result = i;
    for (;;) {}
}

__attribute__((noinline))
void nemu_assert(int cond) {
  if (!cond) {
      _halt(1);
  }
}

void hit_good_trap() {
    _halt(0);
}

typedef struct Result {
  unsigned long msec;
} Result;

int printf(const char *fmt, ...);
void bench_prepare(Result *res);
void bench_done(Result *res);

#endif
```

四、 对于此次实验的心得、感受和建议（比如实验是否过于简单或复杂，是否缺少了某些你认为重要的信息或参考资料，对实验项目的建议，对提供帮助的同学的感谢，以及其他想与任课老师交流的内容等）

讲义中有一处错误（如下图），经过上板结果证明，JALR 指令应在 EX 阶段将返回地址写入寄存器，而非 WB 阶段。

- JALR指令在WB阶段将返回地址写入r31寄存器

另外，此次实验行为仿真耗时将近一小时，debug 不太方便。

最后感谢老师和助教的帮助和指导，使我在不知不觉中自学了 verilog 语言，以及比语言更重要的计算机组成原理框架和硬件思维，感谢同学在共同讨论中和我共同进步，感谢感谢男票一学期以来在写代码和 debug 过程中无微不至的陪伴和理解，希望我们未来依然携手前进！