数字逻辑与处理器基础实验

流水线MIPS处理器设计

2019.8.6

实验安排

A. 课程暑季学期部分的学时数为32

B. 本课程的设计工作可在916机房完成，也可在宿舍等其他地方完成

C. 916机房的开放时间为周一到周五的8:30-18:30

D. 验收时间：不参加专项训练的学生　8月30日。参加专项训练和出国交换或其他特殊情况的同学 9月12日

E. 答疑和验收的时间为每周二和周四的下午，并将根据同学完成情况随时增加，请同学们留意网络学堂上的安排

F. 本课程设计应独立完成，参加专项训练的同学如确有困难，可两人一组合作完成

注意事项

A. 实验评分分为现场验收和实验报告两部分。

B. 现场验收由助教根据现场硬件情况核定，合作完成的要求所有成员均在场，现场成绩占个人成绩的50%。

C. 实验报告如果是合作完成的，则必须标明每个完成人的实际工作和对设计的贡献，报告成绩占个人总成绩的50%。

D. 实验报告内容包括：实验目的；设计方案（原理说明及框图）；关键代码及文件清单；仿真结果及分析；综合情况（面积和时序性能）；硬件调试情况；思想体会。

注意事项（续）

E. 实验报告提交方式：实验报告（word或者pdf）和设计代码打包后提交到网络学堂，提交打包文件名按照“学号_姓名_实验编号”的规则命名。

F. 硬件实验板将在实验验收结束时归还。

G. 根据综合结果，设计出流水线设计功能正确且最高时钟频率在前20名的同学（小组）将可获得该实验10%的加分，申请加分的同学需要单独提交申请，并需提交可方便验证的工程实现代码和设计说明。

H. 实验严禁抄袭，抄袭的（实验报告或者设计代码出现雷同、回答问题明显非个人完成等）课程成绩记零分，并上报院系。

实验内容

- 将春季学期实验四设计的单周期 MIPS 处理器改进为流水线结构，并利用此处理器完成数据排序
- 要点：
  - 中断和异常的处理
  - 外设的设计
  - 流水线中冒险和数据关联问题
  - 测试验证和性能分析

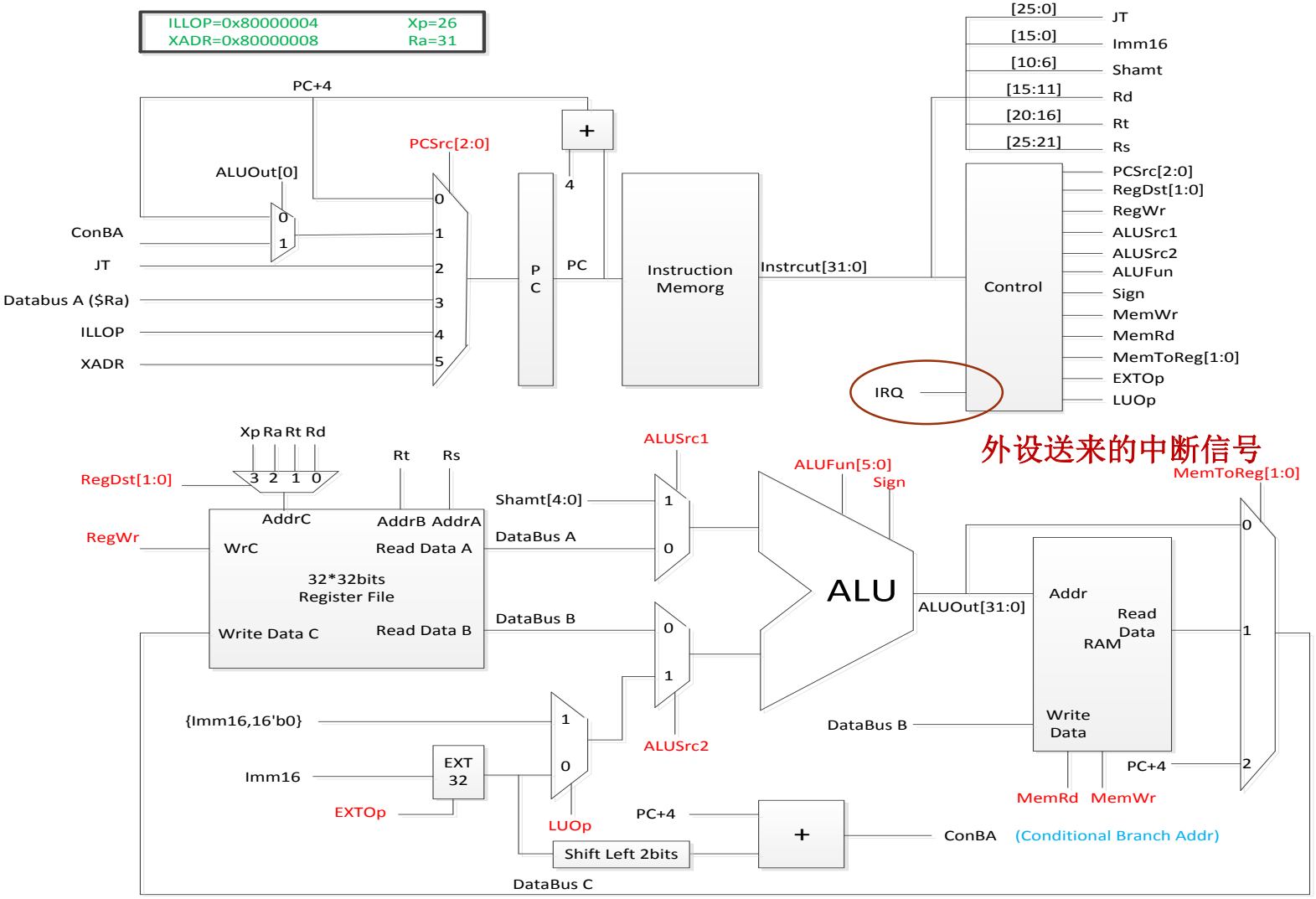
流水线MIPS处理器

- 指令集
  - 空指令：nop （0x00000000，即sll $0,$0,0）
  - 存储访问指令：lw，sw，lui
  - 算术指令：add，addu，sub，subu，addi，addiu
  - 逻辑指令：and，or，xor，nor，andi，sll，srl，sra，slt，slti，sltiu
  - 分支和跳转指令：beq，bne，blez，bgtz，bltz 和 j、jal、jr、jalr
  - 其他指令可以根据情况自行添加。

MIPS处理器中的中断和异常

- 中断与异常
  - 中断：定时器中断（如果UART使用中断模式，还需要添加该中断源）
  - 异常：未定义指令异常（异常发生时，自行设计异常处理代码，例如进入无限循环或者简单跳过）

单周期MIPS处理器对中断的处理



中断和异常

- PCSrc

① **正常执行**时，PC+4；

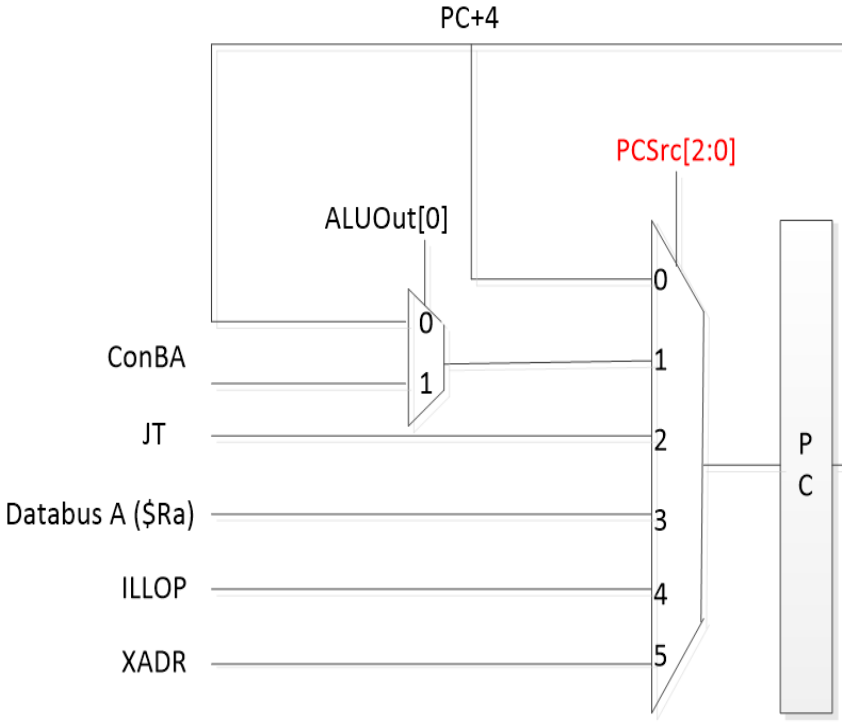
② **条件分支**时，ConBA或者PC+4，根据条件分支中的条件判断结果而定（ALUOut[0]），ConBA来自于PC+4与指令中16位立即数左移2位后的数值之和；

③ **直接跳转**时，JT，其值来自于J型指令中的26位地址；

④ **寄存器跳转**时，Databus A，其值取自$Ra；

⑤ **发生中断**时，ILLOP（常量0x80000004）；

⑥ **发生异常**时，XADR（常量0x80000008）。



复位后，PC初始值为0x80000000

```
0x00000000: J Main;  #跳转到主程序入口
0x00000004: J Interrupt; #跳转到中断服务程序入口
0x00000008: J Exception; #调转到异常处理程序入口
```

中断和异常

- PC监督位

PC的最高位PC[31]为监督位。当该位为‘1’时，处理器处于内核态，此时异常和中断被禁止；当该位为‘0’时，处理器处于普通态，此时允许发生中断和异常。**注意PC[31]不能作为地址最高位去索引指令存储器**，取指令时应当固定将地址最高位置零。只有RESET、异常、中断等有可能将PC[31]设置为‘1’，其他指令不能设置该位为‘1’，JR和JALR指令可以使监督位清零。

- 在处理器复位后，PC中的值应该为0x80000000（处于内核态）；
- 发生中断时，PC中的值应该为0x80000004（处于内核态）；
- 在发生异常时，PC中的值应该为0x80000008（处于内核态）；
- PC+4逻辑电路实现时应该保证PC[31]不变；
- 分支语句和J、JAL语句不应该改变PC[31]；
- 当执行JR、JALR指令时，PC[31]的值由跳转地址（$Ra）中的第31位（最高位）决定。

0x00000000和0x80000000地址实际对应同一个存储单元（指令）

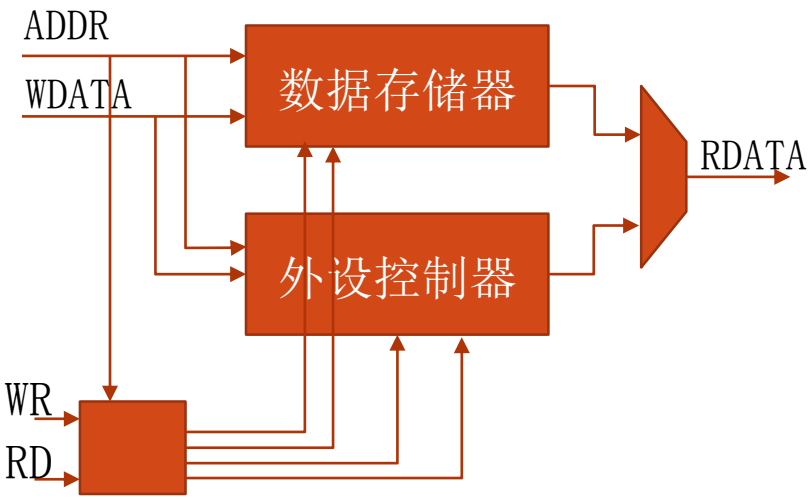
带有中断和异常的程序代码结构

```
0x00000000: J Main;  #跳转到主程序入口
0x00000004: J Interrupt; #跳转到中断服务程序入口
0x00000008: J Exception; #调转到异常处理程序入口

Main: #主程序入口
……

Interrupt: #中断服务例程（ISR）入口
……
0xXXXXXXXX: JR $26

Exception: #异常处理程序入口
……
0xXXXXXXXX: JR $26
```

外设的控制

- 地址空间
  - 哈弗结构：指令地址空间和数据地址空间是分离的
  - 指令存储器采用ROM实现
  - 数据地址空间包括数据存储器、外设等
  - 数据存储器采用RAM实现，其地址分配如下表

地址范围（字节地址）	功能
0x00000000~0x000007FF	数据存储器
0x40000000~0x4000000B	定时器
0x4000000C	外部LEDs
0x40000010	七段数码管
0x40000014	系统时钟计数器
	可自设其他外设



外设的控制

- 定时器外设

地址范围	功能	备注
0x40000000	定时器TH	每当TL计数到全1时，自动加载TH值到TL
0x40000004	定时器TL	定时器计数器，TL值随时钟递增
0x40000008	定时器控制TCON	0bit：定时器使能控制，1-enable，0-disable 1bit：定时器中断控制，1-enable，0-disable 2bit：定时器中断状态

```
if(TCON[0]) begin    //timer is enabled
            if(TL==32'hffffffff) begin
                TL <= TH;
                if(TCON[1]) TCON[2] <= 1'b1; //irq is enabled
            end
            else TL <= TL + 1;
end
```

外设的控制

- 定时器外设

定时器软件操作流程：

i. 关闭定时器，TCON写入0;

ii. 设置定时器周期，TH取值决定定时器的计数周期;

iii. 设置定时器TL为0xFFFFFFFF;

iv. 启动定时器，TCON写入3.

定时器中断软件服务程序（ISR）流程*(此时处理器处于内核态,监督位为'1')：*

i. 定时器中断禁止，同时中断状态清零，TCON的1-2bit清零， TCON &=0xfffffff9;

ii. 保护现场;

iii. 中断处理代码;

iv. 恢复现场;

v. 使能中断，TCON的1bit置1，TCON |= 0x00000002;

vi. 退出中断服务程序，跳转到中断发生时保存的断点地址处继续执行（$26）.

外设的控制

- LED、七段数码管、系统时钟计数器

地址范围（字节地址）	功能	描述
0x4000000C	外部LEDs	0bit：LED 0 …… 7bit：LED 7
0x40000010	七段数码管	0bit：CA 1bit：CB …… 7bit：DP 8bit：AN0 9bit：AN1 10bit：AN2 11bit：AN3
0x40000014	系统时钟计数器	系统复位时，SysTick复位为零，之后每系统时钟周期，计数值加1。忽略溢出。

外设的控制

```
always@(*) begin
    if(rd) begin
        case(addr)
            32'h40000000: rdata <= TH;
            32'h40000004: rdata <= TL;
            32'h40000008: rdata <= {29'b0,TCON};
            32'h4000000C: rdata <= {24'b0,led};
            32'h40000010: rdata <= {20'b0,digi};
            32'h40000014: rdata <= systick;
            default: rdata <= 32'b0;
        endcase
    end
    else
        rdata <= 32'b0;
end
```

读外设寄存器

```
if(wr) begin
        case(addr)
            32'h40000000: TH <= wdata;
            32'h40000004: TL <= wdata;
            32'h40000008: TCON <= {29'b0,wdata[2:0]};
            32'h4000000C: led <= {24'b0,wdata[7:0]};
            32'h40000010: digi <= {20'b0,wdata[11:0]};
        endcase
end
```

写外设寄存器

外设的控制

- UART外设（可选）

地址范围	功能	备注
0x40000018	串口发送数据UART_TXD	串口发送数据寄存器，只有低8bit有效；对该地址的写操作将触发新的UART发送
0x4000001C	串口接收数据UART_RXD	串口接收数据寄存器，只有低8bit有效
0x40000020	串口状态、控制UART_CON	0bit：发送中断使能，1-enable，0-disable 1bit：接收中断使能，1-enable，0-disable 2bit：发送（中断）状态，每当UART_TXD中的数据发送完毕后该比特置‘1’，当执行对该地址的读操作后，将自动清零 3bit：接收（中断）状态，每当UART_RXD中已经接收到一个完整的字节时该比特置‘1’，当执行对该地址的读操作后，将自动清零 4bit：发送模块状态，0-发送模块处于空闲状态，1-发送模块处于发送状态

自行根据春季学期的UART编写此外设

流水线的处理

- 采用完全的forwarding电路解决数据关联问题。
- 对于Load-use类竞争采取阻塞一个周期+Forwarding的方法解决
- 对于分支指令在EX阶段判断（提前判断也可以），在分支发生时刻取消ID和IF阶段的两条指令。
- 对于J类指令在ID阶段判断，并取消IF阶段指令。

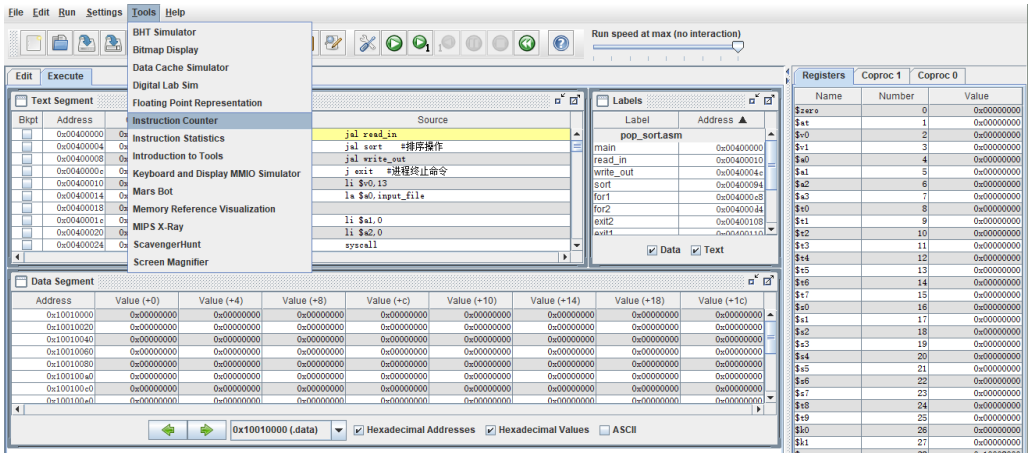
注意：发生中断和异常时的流水线处理

测试验证和性能分析

- 选定任意一种排序算法，编写汇编语言对100个32bit的无符号随机数进行排序。随机数的输入和输出自行设计，不做统一要求。
- 提示：
  - 可以将生成好的100个随机数写在汇编文件中，编译在FPGA比特流中，也可以利用实验三的类似方式通过串口烧录，还可以采用Vivado中的ROM IP等形式来实现。
  - 排序好的数据可以用编写汇编代码，通过软件控制的方式通过串口按顺序输出，也可以利用实验三用硬件方式将内存中的数据读出，还可以用软件控制LED逐字显示排序后的结果。
  - 数据的输入输出用于验证处理器的功能，并测试其性能，不是设计与考核的重点

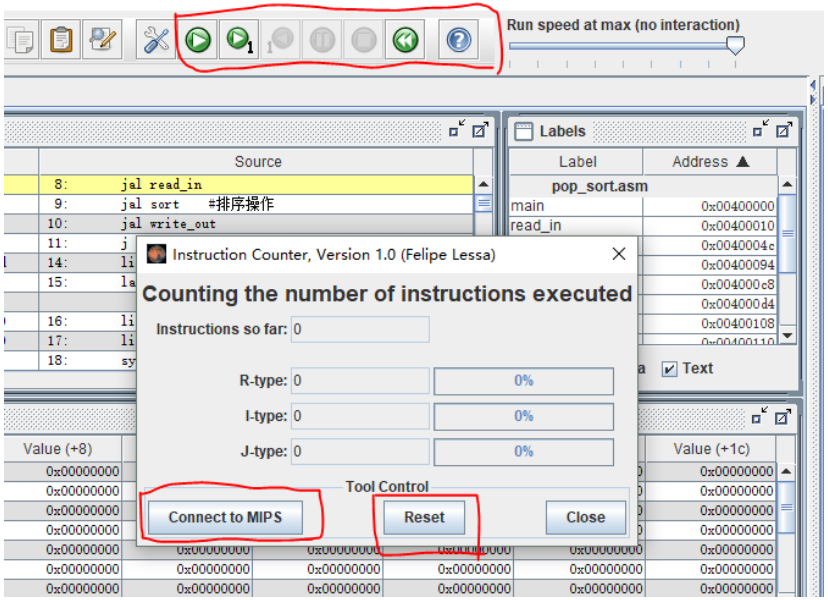
测试验证和性能分析

- 使用MARS等仿真器确定完成此排序操作所执行的指令总数N，通过七段数码管显示十六进制的完成排序操作所消耗的时钟周期数C，计算平均执行一条指令所需要的时钟周期数CPI=C/N，并根据时钟频率计算平均每秒执行指令数目。
- 指令数统计方法
  - 对一个可执行的汇编程序首先进行**编译**，在执行界面选择Tools菜单栏，选择Instruction counter



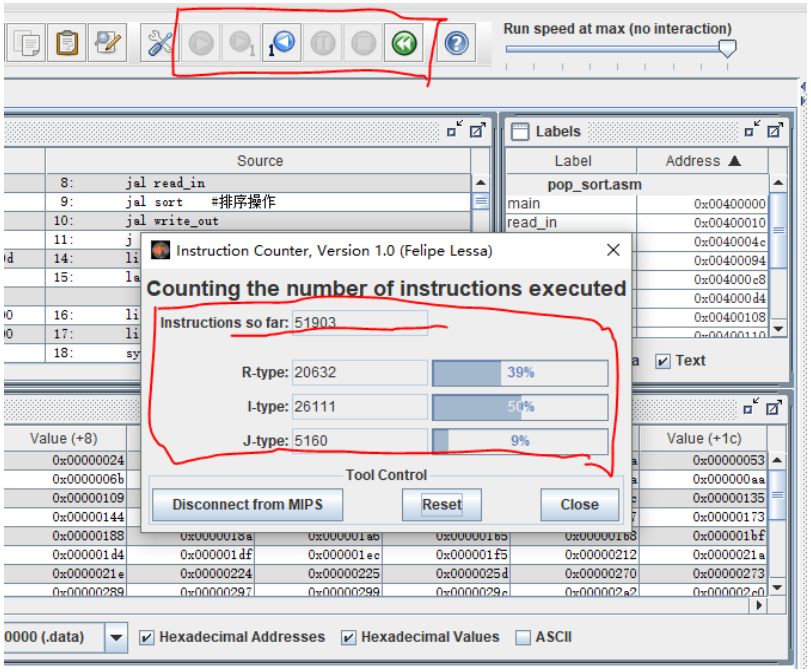
测试验证和性能分析

- 指令数统计方法（续）
  - 点击“Connect to MIPS”，如果看到的指令数不是零，可以先reset，然后执行汇编程序，指令数会随着程序执行变换，支持设置断点，读出 Instruction so far就是程序运行到指定位置的执行的指令数，或者直接到程序运行结束，统计总指令数。



测试验证和性能分析

- 指令数统计方法（续）
  - 最终效果如下图所示，程序运行结束，可以读出实际运行的指令数和指令种类
  - 如果指令数一直增加不会结束，请检查程序是否最终跳入空循环。
  - 实际指令可能与原来的有所不同（如syscall被替代，lw地址变化等）。为方便起见，在保证排序数目为100个情况下仍可以直接运行之前的排序算法统计指令数，可忽略对计算CPI的影响



测试验证和性能分析

- 软件操作提示:

  i. 禁止定时器中断后，将待排序的数据导入RAM

  ii. 读取并保存系统时钟计数器SysTick的值

  iii. 完成排序操作

  iv. 读取时钟计数器SysTick的新值，减去已保存的值，所得的差即为执行排序操作所消耗的时钟周期数。

  v. 使能定时器中断，TCON 的 1bit 置 1，TCON |= 0x00000002;

  （在中断服务程序中，在代码中用查表法等软件译码的方法将计算得到的时钟周期数以十六进制显示在数码管上。）

  vi. 输出排序结果

调试

- 时钟频率：流水线主频以时序报告中的implement时序分析为准，你的设计很可能不能正常工作在100MHz的时钟频率下，注意对输入时钟进行分频，使时钟频率接近但不超过最高工作频率。
- 软件调试：可以先在MIPS的软件仿真器中进行简单仿真，初期也可以利用软件仿真器将汇编代码转换为机器码，我们的指令兼容于标准MIPS32指令集
- 硬件调试：可以考虑将MIPS的主时钟连接到外部按键上或者分频到极低频率（比如1Hz），同时将PC等关键信号输出到外部Led上，观察程序执行过程
- 设计时应当考虑查找表、寄存器等资源消耗情况，并对流水线和单周期的资源消耗对比进行分析，但资源消耗情况不是对设计进行评分的主要因素

答 疑