**CS214-CPU大作业检查表（答辩时间： ）**

|  |  |  |
| --- | --- | --- |
| **用例编号**  **（场景1）** | **用例描述** | **测试数据及结果**  **(答辩负责人签名) :** |
| 3’b000 | **输入测试数a（仅识别a的最低7bit），输入完毕后在led灯上显示a，同时用1个led灯显示a的奇校验位** |  |
| 3’b001 | **输入测试数a（识别a的完整8bit），输入完毕后在led灯上显示a，同时用1个led灯显示a的奇校验结果** |  |
| 3’b010 | **先执行测试用例3’b111, 再计算 a 和 b的按位或非运算，将结果显示在输出设备** |  |
| 3’b011 | **先执行测试用例3’b111, 再计算 a 和 b的按位或运算，将结果显示在输出设备** |  |
| 3’b100 | **先执行测试用例3’b111, 再计算 a 和 b的按位异或运算，将结果显示在输出设备** |  |
| 3’b101 | **先执行测试用例3’b111, 再执行 sltu 指令，将a和b按照无符号数进行比较，用输出设备展示a<b的关系是否成立** (关系成立，亮灯，关系不成立，灭灯） |  |
| 3’b110 | **先执行测试用例3’b111, 再执行 slt 指令，将a和b按照有符号数进行比较，用输出设备展示a<b的关系是否成立**(关系成立，亮灯，关系不成立，灭灯） |  |
| 3’b111 | **输入测试数a, 输入测试数b，在输出设备上展示a和b的值** |  |

|  |  |  |
| --- | --- | --- |
| **用例编号**  **（场景2）** | **用例描述** | **测试数据及结果**  **(答辩负责人签名) :** |
| 3’b000 | **输入a的数值（a被看作有符号数），计算1到a的累加和，在输出设备上显示累加和（如果a是负数，以闪烁的方式给与提示）** |  |
| 3’b001 | 输入a的数值（**a被看作无符号数**），以递**归的方式计算1到a的累加和**，**记录本次入栈和出栈次数**，**在输出设备上显示入栈和出栈的次数之和** |  |
| 3’b010 | 输入a的数值（**a被看作无符号数**），**以递归的方式计算1到a的累加和，记录入栈和出栈的数据，在输出设备上显示入栈的参数，每一个入栈的参数显示停留2-3秒** （说明，此处的输出不关注$ra的入栈和出栈信息） |  |
| 3’b011 | 输入a的数值（**a被看作无符号数**），**以递归的方式计算1到a的累加和，记录入栈和出栈的数据，在输出设备上显示出栈的参数，每一个出栈的参数显示停留2-3秒**（说明，此处的输出不关注$ra的入栈和出栈信息） |  |
| 3’b100 | 输入测试数**a**和测试数**b**，实现**有符号数**（a，b以及相加和都是8bit，其中的最高bit被视作符号位，**如果符号位为1，表示的是该负数的补码**）的**加法**，并对**是否溢出进行判断，输出运算结果以及溢出判断** |  |
| 3’b101 | 输入测试数**a**和测试数**b**，实现**有符号数**（a，b以及差值都是8bit，其中的最高bit被视作符号位，**如果符号位为1，表示的是该负数的补码**）的**减法**，并对是否溢出进行判断，**输出运算结果以及溢出判断** |  |
| 3’b110 | 输入测试数**a**和测试数**b**，实现**有符号数**（a，b都是8bit，乘积是16bit，其中的**最高bit被视作符号位**，**如果符号位为1，表示的是该负数的补码**）的**乘法**，**输出乘积** |  |
| 3’b111 | 输入测试数**a**和测试数**b**，实现**有符号数**（a，b，商和余数都是8bit，其中的**最高bit被视作符号位，如果符号位为1，表示的是该负数的补码**）的**除法**，**输出商和余数（商和余数交替显示，各持续5秒**） |  |

|  |  |  |
| --- | --- | --- |
|  | **答辩负责人签名 :** | **测试结果** |
| **代码有效性检查** | 生成bitstream的代码是否与答辩上传的一致 |  |
| asm更新后加载到CPU中执行，是否生效 |  |

|  |  |  |
| --- | --- | --- |
| 小组成员签名(姓名-学号-实验班时间） | 贡献比 | 问答登记(所负责的模块，该部分问答的情况) **(答辩负责人签名) :** |
| 郑祖彬 | 33.3% | MIPS 文件 |
| 高自然 | 33.3% | CPU 架构，外设 |
| 李达辉 | 33.3% | CPU 架构 |

|  |  |  |  |
| --- | --- | --- | --- |
| **CPU代码规范性， CPU实现所采用的语言：** | | 检查结果及必要说明 **(答辩负责人签名) :** | |
| 注释较全且能够说明问题 | |  | |
| 结构化设计 | |  | |
| 组合逻辑语句块中的if-else，以及case结构中是否明确所有分支 | |  | |
| 在一个always语句块里是否出现阻塞赋值、非阻塞赋值的混用 | |  | |
| 时序逻辑（敏感列表中是否出现 跳变沿和电平信号混用） | |  | |
| 时序逻辑（敏感列表中是否出现非时钟、非复位信号的跳变沿） | |  | |
| 模块定义中使用参数，使用头文件集中防止所有参数的定义（+1） | |  | |
|  | **方案自述** | | **检查结果及说明**  **(答辩负责人签名) :** |
| CPU 特性 | 单周期，10MHz, 支持uart通信 | |  |
| CPU参考方案出处 | Lab课件，github，csdn | |  |
| 基于CPU参考代码  所做修改 | MMIO，Uart通信的拓展 | |  |
| IO方案 | FFFFFC60 switch 16位  FFFFFC64 PAD 16位  FFFFFC68 led灯 16位  FFFFFC6C 显示管  FFFFFC70 LED\_TESTCASE 显示 3位  FFFFFC72 UART  FFFFFC80 IOREADY  软件：  使用sw和lw  Lw input  Sw output  硬件：  使用memorio模块进行IO管理 | |  |
|  |
| 栈空间方案 | 基地址FFFFFC00，软件根据栈指针进行出栈入栈操作 | | asm文件中落地的情况 |
| 硬件代码中落地的情况 |
| 溢出检测方案 | 软件检测，进行结果最高位符号的比较判断 | |  |
| 乘除法实现方案 | 软件实现，循环加减计算 | |  |

Bonus及说明(实现pipeline、中断等机制，需要小组自行提供可靠的测试用例，实现其他外设，需要有该外设对应的IO地址方案以及确认落地方式）

|  |  |
| --- | --- |
| Bonus 自述 | Bonus 检查及说明 **(答辩负责人签名) :** |
| 外设：  使用了多种外设，小键盘、开关、七段led数码管、led灯、按钮  Uart拓展:  Uart可以将cpu输出转移到电脑屏幕，实现将指定内存位置通过指令传输至电脑上，实现cpu-windows debug bridge |  |