**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) :** |
| 刘圣鼎-12110813-周一五六节 | 33.3% | MIPS场景一、场景二的后四部分和键盘I/O |
| 张展玮-12110817-周一五六节 | 33.3% | CPU模块和MIPS场景二的前四部分、音乐播放器 |
| 谢嘉楠-12110714-周一五六节 | 33.3% | CPU模块、UART、音乐播放器 |

|  |  |  |  |
| --- | --- | --- | --- |
| **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，CPU时钟频率10MHZ，支持UART通信 | |  |
| CPU参考方案出处 | Lab PPT、Minisys硬件手册 | |  |
| 基于CPU参考代码  所做修改 | 补充完善了Lab PPT的各模块，增加switchs、keyboard输入模块，leds、show、music输出模块 | |  |
| IO方案 | 使用switch和keyboard作为输入设备，led、七段数码管和蜂鸣器作为输出设备。输入的内存首地址是0xFFFFFC70，输出的内存首地址是0xFFFFFC60 | | asm文件中落地的情况 |
| 硬件代码中落地的情况 |
| 栈空间方案 | 栈空间的位置（基地址）为0XFFFFFC00 | | 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 | 软硬件结合 | |  |
| 乘除法实现方案 | 软硬件结合 | |  |

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) :** |
| 1. 实现对复杂外设接口的支持，在我们的Project中为对小键盘、七段数码管复杂外设接口的支持   为了在我们的项目中支持小键盘和七段数码管这两个复杂外设接口，我们采用了以下方案。首先，我们定义了输入和输出的内存地址，以便与这些外设进行通信。对于小键盘，我们将输入的内存首地址设置为0xFFFFFC70。当用户按下键盘上的按键时，相应的键值将被存储在这个内存地址中，供我们的程序读取。对于七段数码管，我们将输出的内存首地址设置为0xFFFFFC60。我们的程序可以将需要显示的数字或字符写入这个内存地址，然后七段数码管将根据这些数据进行相应的显示。  此外，我们还定义了I/O的起始地址为0XFFFFFC00。这个起始地址用于与外设进行交互。我们可以通过读写这个地址来与外设进行数据的传输和控制。  为了确保这些外设接口的正确性和可靠性，我们小组提供了详细的测试用例。针对小键盘，我们编写了测试代码，模拟用户按下不同的按键，并验证相应的键值是否正确地存储在指定的内存地址中。对于七段数码管，我们编写了测试代码，将不同的数字和字符写入指定的内存地址，并验证七段数码管是否正确地显示了这些数据。通过这些测试用例，我们可以确保复杂外设接口的支持功能正常运作，并提供了可靠的验证手段。   1. 实现了音乐播放器功能。可以通过调整PWM，使蜂鸣器播放音乐；以及通过UART传入乐谱播放音乐   我们在项目中实现了音乐播放器功能，以便能够通过蜂鸣器播放音乐。我们采用了两种方式来实现音乐播放：通过调整PWM信号和通过UART传入乐谱。  首先，我们利用PWM（脉冲宽度调制）信号来控制蜂鸣器的音调和音量。通过调整PWM信号的频率和占空比，我们可以生成不同音高和音量的声音。我们的程序可以根据输入的音乐信息，计算出相应的PWM参数，并将其应用于蜂鸣器控制信号。这样，蜂鸣器就能按照乐曲的要求播放相应的音乐。  其次，我们还支持通过UART接口传入乐谱来播放音乐。用户可以将乐谱数据通过UART串口发送给我们的系统。我们的程序将解析接收到的乐谱信息，并根据其中的音符、音长和节奏等参数生成相应的PWM参数，以便蜂鸣器能够按照乐谱正确地演奏音乐。通过这种方式，用户可以通过拨码开关和按钮与CPU进行通信，实现音乐的播放控制。  为了确保音乐播放器功能的正确性，我们提供了相应的测试用例。我们编写了测试代码，模拟输入不同的音乐信息，包括音符、音长、节奏等参数，并验证蜂鸣器是否能够按照预期的方式播放相应的音乐。我们还通过UART接口发送乐谱数据，并验证系统是否能够正确解析和播放这些乐谱。通过这些测试用例，我们可以验证音乐播放器功能的可靠性和正确性。   1. 基于CPU的软硬件协同的应用   我们实现了一种基于CPU的软硬件协同应用，通过指令中的地址信息（MMIO）来访问相关的复杂外设，从而实现了复杂外设接口的访问。具体而言，我们的系统可以通过在指令中指定特定的MMIO地址来进行对外设的读写操作。通过这种协同应用，我们能够更灵活地控制和操作外设。这样，我们可以实现对复杂外设的控制和管理，提供更多功能和扩展性。  此外，我们还实现了通过UART下发乐谱的功能，使Minisys的蜂鸣器可以演奏对应的乐曲。通过UART接口，我们可以将乐谱数据发送给系统，然后通过软硬件协同应用来解析和处理这些乐谱数据。解析后的数据将被转换为相应的PWM参数，以控制蜂鸣器演奏出正确的音乐。 |  |