一、基本场景 1（每个用例 10 分，折合到 project 的分数为：（实际通过的用例数/8) *60*0.3）

使用开发板上的拨码开关用于做输入，其中 3 个拨码开关(x2,..x0) 用于测试用例的编号输入，8 个拨码开关（sw7,..sw0) 用于做测试数据的输入，使用 led 灯或者 7 段数码显示管做输出.（如果 led 灯的数目不够，则将输出在 led 灯上交替展示，每个部分展示停留 2 秒）（备注，拨码开关和 led 的高位统一在左侧，低位统一在右侧）

说明 1：如果输入数据有多个（比如 a，b），请先输入 a，按确认键，再输入 b

说明 2：用 led 输出操作数的二进制形式，使用数码管（或者 VGA）输出运算结果的十六进制形式。

用例编号	用例描述（用例 3’b011-3’b111 中，a，b 均为二进制补码形式）	测试说明
3‘b000	输入测试数 a, 输入测试数 b，在输出设备（led）上展示 8bit 的 a 和 b 的值	如 a 输入 8’b1101_0011, 对应的输出设备上展示 8’b1101_0011
3‘b001	输入测试数 a，以 lb 的方式放入某个寄存器，将该 32 位的寄存器的值以十六进制的方式展示在输出设备上（数码管或者 VGA)，并将该数保存到 memory 中 （在 3’b011-3’b111 用例中，将通过 lw 指令从该 memory 单元中读取 a 的值进行比较）	如 a 输入 8’b1101_0011，以 lb 的方式存入寄存器中，寄存器的值应该为 32’hFFFF_FFD3, 数码管或者 VGA 上应该显示 FFFFFFD3

3‘b010	输入测试数 b，以 lbu 的方式存入某个寄存器，将该 32 位寄存器的值以十六进制的方式展示在输出设备上（数码管或者 VGA)，并将该数保存到 memory 中（在 3’b011-3’b111 用例中，将通过 lw 指令从该 memory 单元中读取 a 的值进行比较）	如 b 输入 8’b1101_0011，以 lbu 的方式存入寄存器中，寄存器的值应该为 32’h0000_00D3, 数码管或者 VGA 上应该显示 000000D3
3‘b011	用 beq 比较 测试数 a 和 测试数 b（来自于用例 1 和用例 2），如果关系成立，点亮 led，关系不成立，led 熄灭	比如 tc001 和 tc010 中输入的 a 和 b 都是 8’b0101_0011, 则 在经过 register 再到 memory 后二者相等，led 灯亮； 比如 tc001 和 tc010 中输入的 a 和 b 都是 8’b1101_0011, 则 在经过 register 再到 memory 后二者不等，led 灯不亮； 比如 tc001 和 tc010 中输入的 a 和 b 分别是 8’b0101_0011 和 8’b1101_0011, 则 在经过 register 再到 memory 后二者不等，led 灯不亮；
3‘b100	用 blt 比较 测试数 a 和 测试数 b（来自于用例 1 和用例 2），如果关系成立，点亮 led，关系不成立，led 熄灭	比如 tc001 和 tc010 中输入的 a 和 b 都是 8’b0101_0011, 则 在经过 register 再到 memory 后二者相等，less than 的关系不成立，led 灯不亮； 比如 tc001 和 tc010 中输入的 a 和 b 都是 8’b1101_0011, 则 在经过 register 再到 memory 后 a 为 32’hFFFF_FFD3，b 为 32’h0000_00D3，less than 关系成立，led 灯亮； 比如 tc001 和 tc010 中输入的 a 和 b 分别是 8’b1101_0011 和 8’b0101_0011, 则 在经过 register 再到 memory 后；a 为 32’hFFFF_FFD3，b 为 32’h0000_00D3，less than 关系成立，led 灯亮； 反之则不亮

3‘b101	用 bge 比较 测试数 a 和 测试数 b（来自于用例 1 和用例 2），如果关系成立，点亮 led，关系不成立，led 熄灭	比如 tc001 和 tc010 中输入的 a 和 b 都是 8’b0101_0011, 则 在经过 register 再到 memory 后二者相等，greater or equal 的关系成立，led 灯亮； 比如 tc001 和 tc010 中输入的 a 和 b 都是 8’b1101_0011, 则 在经过 register 再到 memory 后 a 为 32’hFFFF_FFD3，b 为 32’h0000_00D3，greater or equal 关系不成立，led 灯不亮； 比如 tc001 和 tc010 中输入的 a 和 b 分别是 8’b1101_0011 和 8’b0101_0011, 则 在经过 register 再到 memory 后；a 为 32’hFFFF_FFD3，b 为 32’h0000_00D3，greater or equal 关系不成立，led 灯不亮； 反之则亮
3‘b110	用 bltu 比较 测试数 a 和 测试数 b（来自于用例 1 和用例 2），如果关系成立，点亮 led，关系不成立，led 熄灭	比如 tc001 和 tc010 中输入的 a 和 b 都是 8’b0101_0011, 则 在经过 register 再到 memory 后二者相等，less than(unsigned)的关系不成立，led 灯不亮； 比如 tc001 和 tc010 中输入的 a 和 b 都是 8’b1101_0011, 则 在经过 register 再到 memory 后 a 为 32’hFFFF_FFD3，b 为 32’h0000_00D3，less than（unsigned）关系不成立，led 灯不亮； 比如 tc001 和 tc010 中输入的 a 和 b 分别是 8’b1101_0011 和 8’b0101_0011, 则 在经过 register 再到 memory 后；a 为 32’hFFFF_FFD3，b 为 32’h0000_00D3，less than 关系不成立，led 灯不亮； 反之则亮

3‘b111	用 bgeu 比较 测试数 a 和 测试数 b（来自于用例 1 和用例 2），如果关系成立，点亮 led，关系不成立，led 熄灭	比如 tc001 和 tc010 中输入的 a 和 b 都是 8’b0101_0011，则 在经过 register 再到 memory 后二者相等，greater or equal（unsigned）的关系成立，led 灯亮； 比如 tc001 和 tc010 中输入的 a 和 b 都是 8’b1101_0011，则 在经过 register 再到 memory 后 a 为 32’hFFFF_FFD3，b 为 32’h0000_00D3，greater or equal（unsigned）关系成立，led 灯亮； 比如 tc001 和 tc010 中输入的 a 和 b 分别是 8’b1101_0011 和 8’b0101_0011，则 在经过 register 再到 memory 后；a 为 32’hFFFF_FFD3，b 为 32’h0000_00D3，greater or equal（unsigned）关系成立，led 灯亮；反之则不亮

二、基本场景 2 （每个用例 10 分，折合到 project 的分数为：（实际通过的用例数/8) *60*0.3）
使用开发板上的拨码开关做输入，其中 3 个拨码开关（x3-x0）用于测试用例的编号输入，8~16 个拨码开关用于做测试数据的输入（备注，拨码开关、led 以及数码管的高位统一在左侧，低位统一在右侧）
说明 2：用 led 输出操作数的二进制形式，使用数码管（或者 VGA）输出运算结果的十六进制形式。

用例编号	用例描述（用例 3‘b001-3’b011 的输入数据为有符号数）	测试说明
3‘b000	输入一个 8bit 的数，计算并输出其前导零的个数	如输入 8’b1101_0011，对应的输出设备上展示 0 如输入 8’b0001_0011，对应的输出设备上展示 3

		如输入 8’b0000_0000, 对应的输出设备上展示 7
3‘b001	输入 16bit 位宽的 IEEE754 编码的半字浮点数，对其进行向上取整，输出取整后的结果	如输入 16’b0_10010_1001010000 (12.625 的半精度 754 编码), 向上取整后在输出设备（7 段数码管上）输出 13 的十六进制 D 如输入
3‘b010	输入 16bit 位宽的 IEEE754 编码的半字浮点数，对其进行向下取整，输出取整后的结果	如输入 16’b0_10010_1001010000 (12.625 的半精度 754 编码), 向下取整后在输出设备（7 段数码管上）输出 12 的十六进制 C
3‘b011	输入 16bit 位宽的 IEEE754 编码的半字浮点数，对其进行四舍五入取整，输出取整后的结果	如输入 16’b0_10010_1001010000 (12.625 的半精度 754 编码), 四舍五入取整后在输出设备（7 段数码管上）输出 13 的十六进制 D
3‘b100	分两次输入两个 8bit 的数 a 和 b，对 a，b 做加法运算，如果相加和超过 8bit，将高位取出，累加到相加和中，对相加和取反后输出	如输入 a 的值为 8’B1111_1011, b 的值为 8’B0111_1011，sum1=a+b=9’b1_0111_0110, 则取出 index 为 8 的 bit 位 1，累加到相加和上，sum2 = 1’b1+8’b0111_0110=8’b0111_0111; 取反后为 8’b1000_1000，即在数码管上输出 88
3‘b101	输入 12bit 的数据，以小端模式从拨码开关输入，以大端的方式呈现在输出设备上	如输入 12’b0011_1010_1100（其中 8’b1011_1100（即 8’HAC）是 byte0，8’b0000_0011(即 8’H03)是 byte1）,以大端模式输出该值的十六进制时（从左到右） 应依次为 AC0 或者 C03（两种方式都可以）

		备注，本用支持输入 16bit 的小端输入，以大端方式呈现该数值。如输入 16’b0101_0011_1010_1100（其中 8’b1011_1100（即 8’HAC）是 byte0，8’b0101_0011(即 8’H53)是 byte1）,以大端模式输出该值的十六进制时（从左到右） 应依次为 AC53
3‘b110	以递归的方式计算小于输入数据的斐波拉契数字的数目，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	由于实现方案的差异，本题没有唯一答案 查时比对用例的板上测试结果 和 该用例的改造版（使用 ecall 处理输入输出）在 raras 上运行的结果是否合理及是否一致
3‘b111	以递归的方式计算小于输入数据的斐波拉契数字的数目，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）	由于实现方案的差异，本题没有唯一答案 检查时比对用例的板上测试结果 和 该用例的改造版（使用 ecall 处理输入输出，每个输出直接打印即可，不需要停留 2-3 秒）在 raras 上运行的结果是否合理及是否一致

三、pipeline 测试用例参考

场景	指令
1．处理一个 hazard，使用 ALU-ALU forwarding 的情况	Sub x2,x1,x3 And x12, x2,x5
2．处理一个 load-use hazard，使用 MEM-ALU forwarding 的情况	Lw x2, 8(x3) And x12, x2, x5
3．两个 harzard	Sub x2,x1,x3 And x12, x2,x5 Or x13,x6, x2