Test cases (Prepared by Dr. Muhamed Mudawar)
Instructions were hand-assembled (can have errors, check Hexadecimal)

Initializing Registers (Testing I-Type ALU):

Instruction	Hexadecimal	Expected Result
lui 0x78f	0xff8f	r1 = 0xf1e0 (shifted 5 bits left)
ori r2, r0, 4	0x5102	r2 = 4 = 0x0004
addi r3, r0, -2	0x4783	r3 = -2 = 0xfffe (sign extension)
xori r4, r0, -2	0x5f84	r4 = 0x001e (zero extension)

Testing R-Type ALU Instructions (NO RAW hazards – NO Forwarding)

Instruction	Hexadecimal	Expected Result
add r5, r1, r1	0x0149	r5 = 0xe3c0 (carry is ignored)
sub r6, r1, r2	0x038a	r6 = 0xf1dc
slt r7, r1, r2	0x05ca	r7 = 1 (true) r1 < 0
sltu r5, r1, r2	0x074a	r5 = 0 (false)
and r6, r3, r4	0x099c	r6 = 0x001e
or r7, r1, r2	0x0bca	r7 = 0xf1e4
xor r5, r1, r3	0x0d4b	r5 = 0x0e1e
nor r6, r1, r2	0x0f8a	r6 = 0x0e1b
sll r7, r4, r2	0x11e2	r7 = 0x01e0
srl r5, r1, r2	0x134a	r5 = 0x0f1e
sra r6, r1, r2	0x158a	r6 = 0xff1e
ror r7, r3, r2	0x17da	r7 = 0xefff

Testing RAW hazards and Forwarding

Instruction	Hexadecimal	Expected Result
add r5, r1, r1	0x0149	r5 = 0xe3c0
sub r6, r5, r4	0x03ac	r6 = 0xe3a2 (depends on add)
and r7, r5, r6	0x09ee	r7 = 0xe380 (depends on add/sub)
ori r5, r5, 0xf	0x53ed	r5 = 0xe3cf (depends on add)

Testing SW and LW

Instruction	Hexadecimal	Expected Result
sw r1, 0(r0)	0x6801	MEM[0] = 0xf1e0
sw r4, 1(r0)	0x6844	MEM[1] = 0x001e
lw r5, 0(r0)	0x6005	r5 = MEM[0] = 0xf1e0

Testing Load delay, stalling pipelining, and forwarding after LW

Instruction	Hexadecimal	Expected Result
lw r6, 1(r0)	0x6046	r6 = MEM[1] = 0x001e
andi r7, r6,0xb	0x4af7	r7 = 0x000a (stall 1 cycle & forward)
sw r7, 2(r0)	0x6887	MEM[2] = 0x000a (forwarded from andi)
lw r5, 0(r0)	0x6005	r5 = MEM[0] = 0xf1e0
sw r5, 3(r0)	0x68c5	MEM[3] = 0xf1e0 (forwarded from lw)

Testing Branch and Jump Instructions

Instruction	Hexadecimal	Expected Result
beq r1, r1, +2	0x7089	branch should be taken (to bne)
add r5, r2, r2	0x0152	should be skipped (r5 not modified)
bne r0, r1, +3	0x78c1	branch should be taken (to j)
add r6, r2, r2	0x0192	should be skipped (r6 not modified)
add r7, r4, r4	0x01e4	should be skipped (r7 not modified)
j +3	0x8003	jump 3 instructions forward
add r5, r2, r2	0x0152	should be skipped (r5 not modified)
add r6, r4, r4	0x01a4	should be skipped (r6 not modified)
add r0, r0, r0	0x0000	NO Operation

Fibonacci Example: Testing JAL and JR

Instruction	Hexadecimal	Expected Result
ori r1, r0, 5	0x5141	r1 = 5 (5th Fibonacci element)
jal fib (+4)	0x8804	call Fib (r7 = address of or)
or r5, r2, r0	0x0b50	move r5 = r2 (Fib result) = 8
j 0	0x8000	infinite jump to same address
	0x0000	No operation
ori r2, r0, 1	0x5042	Fib starts here: r2 = 1
ori r3, r0, 1	0x5043	r3 = 1
add r3, r2, r3	0x00d3	loop starts here: r3 = r2 + r3
sub r2, r3, r2	0x029a	r2 = r3 – r2
addi r1, r1, -1	0x47c9	r1 = r1 – 1
bne r1, r0, -3	0x7f48	branch to loop if r1 != 0
jr r7	0x1838	return to caller
add r1, r2, r2	0x0052	Should be skipped (r1 not modified)