Instructions should be hand-assembled, converted to hexadecimal, and loaded into the instruction memory

Initializing Registers (Testing I-Type ALU):

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

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

Instruction	Hexadecimal	Expected Result
add r5, r1, r1	????	r5 = 0xe3c0 (carry is ignored)
sub r6, r1, r2	????	r6 = 0xf1dc
slt r7, r1, r2	????	r7 = 1 (true) r1 < 0
sltu r5, r1, r2	????	r5 = 0 (false)
and r6, r3, r4	????	r6 = 0x001e
or r7, r1, r2	????	r7 = 0xf1e4
xor r5, r1, r3	????	r5 = 0x0e1e
nor r6, r1, r2	????	r6 = 0x0e1b
sll r7, r4, r2	????	r7 = 0x01e0
srl r5, r1, r2	????	r5 = 0x0f1e
sra r6, r1, r2	????	r6 = 0xff1e
ror r7, r3, r2	????	r7 = 0xefff

Testing RAW hazards and Forwarding:

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

Testing SW and LW:

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

Testing Load delay, stalling pipelining, and forwarding after LW:

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