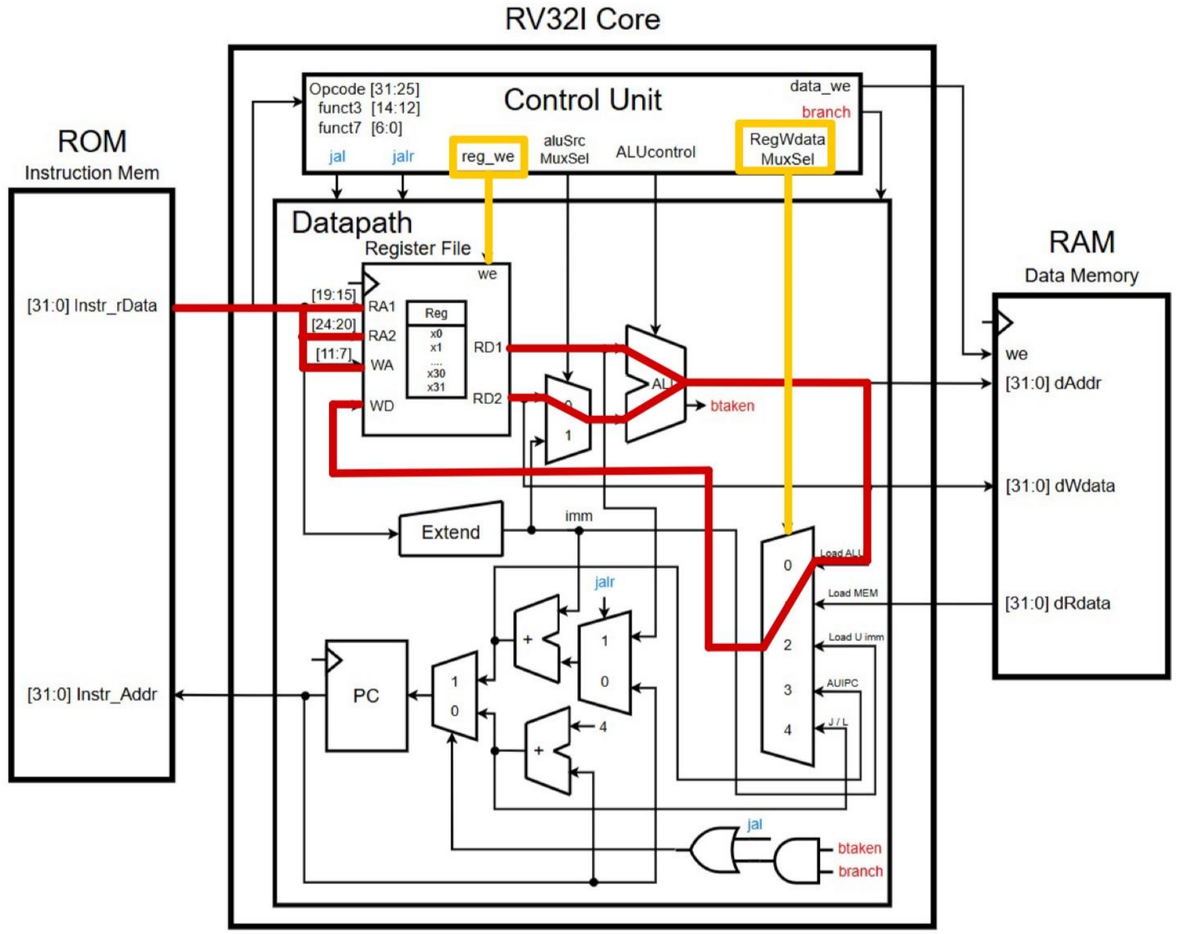
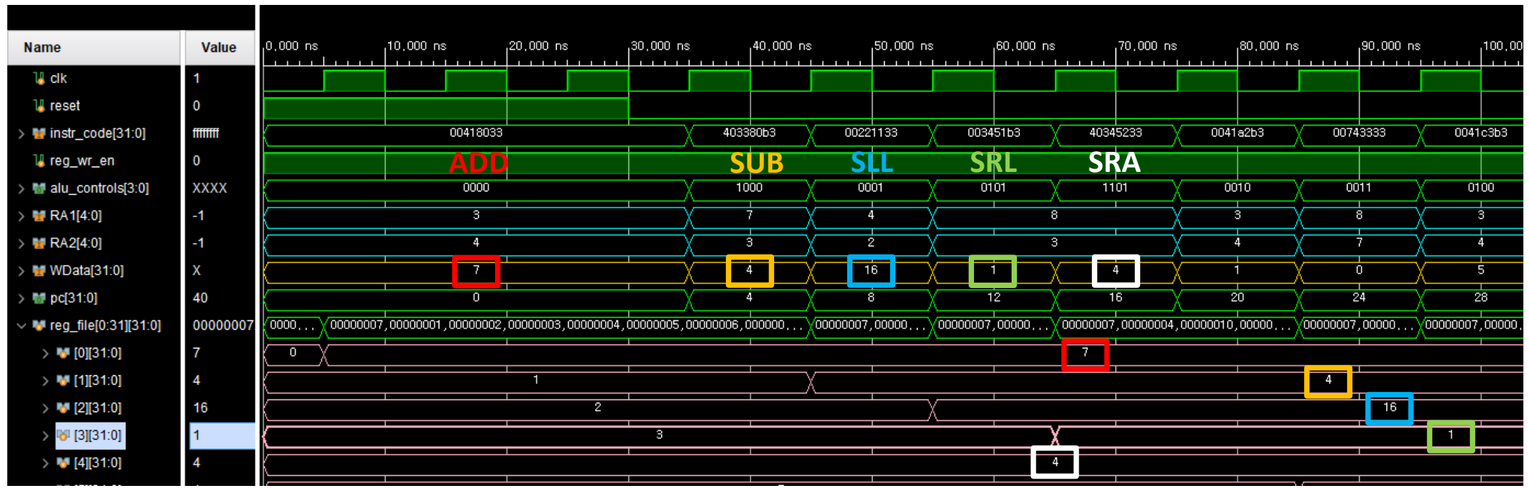
03 Type별 기능 및 Simulation

R-type



03 Type별 기능 및 Simulation

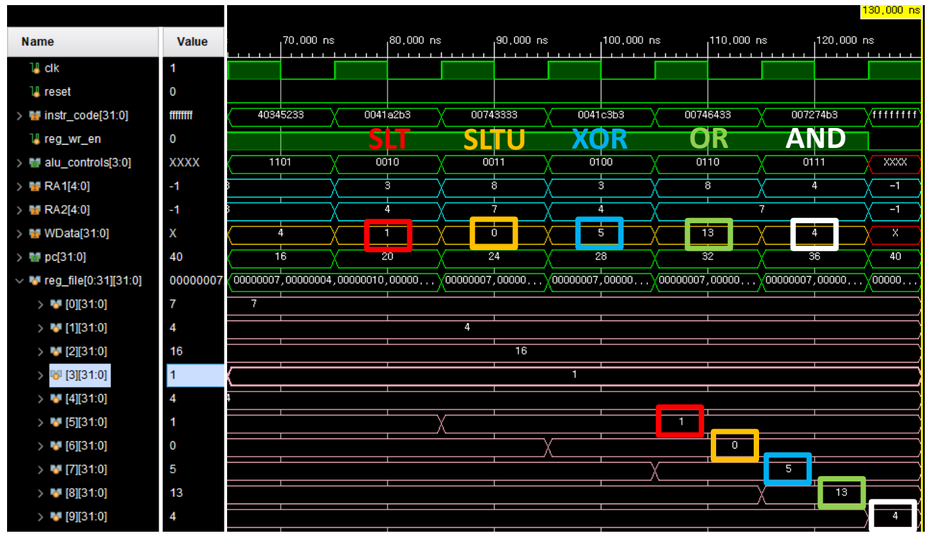
R-type



TYPE	ASSEMBLY	DESCRIPT	RESULT
R-type	add x0, x3, x4	rd = rs1 + rs2	rd ← 3 + 4 = 7
	sub x1, x7, x3	rd = rs1 - rs2	rd ← 7 - 3 = 4
	sll x2, x4, x2	rd = rs1 << rs2	rd ← 0100 << 2 = 1_0000 = 16
	srl x3, x8, x3	rd = rs1 >> rs2	rd ← 1000 >> 3 = 0001 = 1
	sra x4, x8, x3	rd = rs1 >> rs2 (m-ex)	rd ← 1000 >> 1 = 0100 = 4

03 Type별 기능 및 Simulation

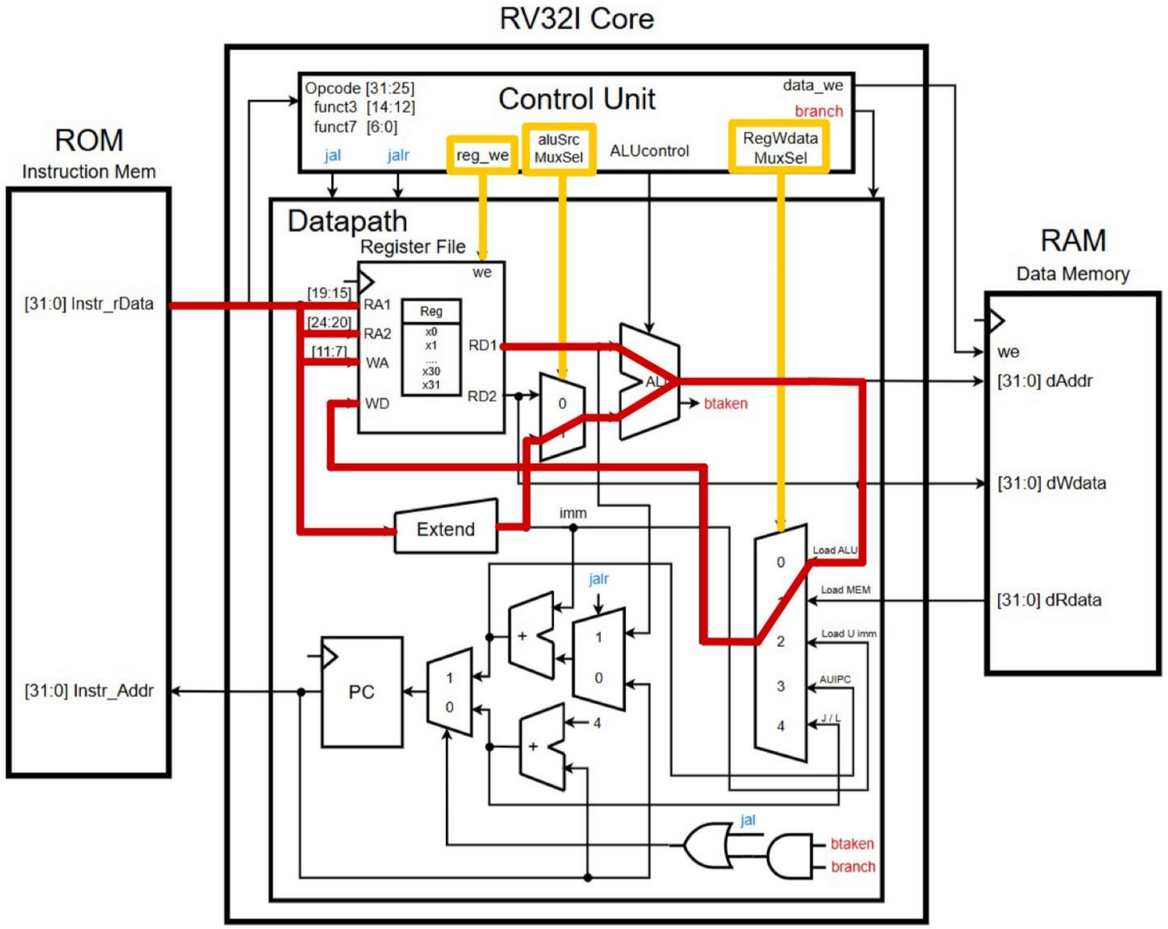
R-type



TYPE	ASSEMBLY	DESCRIPT	RESULT
R-type	slt x5, x3, x4	rd = (rs1 < rs2) ? 1 : 0	rd ← 1 < 4 = 1 “True”
	sltu x6, x8, x7	rd = (rs1 < rs2) ? 1 : 0 (0-ex)	rd ← 8 < 7 = 0 “False”
	xor x7, x3, x4	rd = rs1 ^ rs2	rd ← 0001 ^ 0100 = 0101 = 5
	or x8, x8, x7	rd = rs1 ¦ rs2	rd ← 1000 ¦ 0101 = 1101 = 13
	and x9, x4, x7	rd = rs1 & rs2	rd ← 0100 & 0111 = 0100 = 4

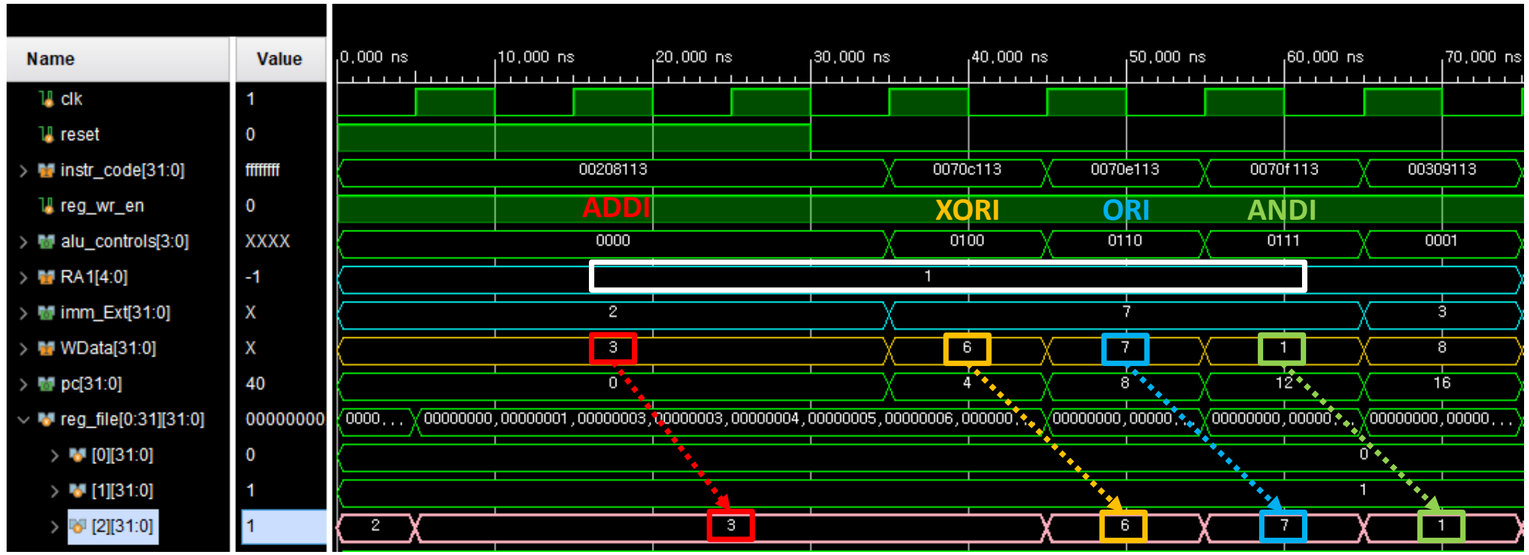
03 Type별 기능 및 Simulation

I-type



03 Type별 기능 및 Simulation

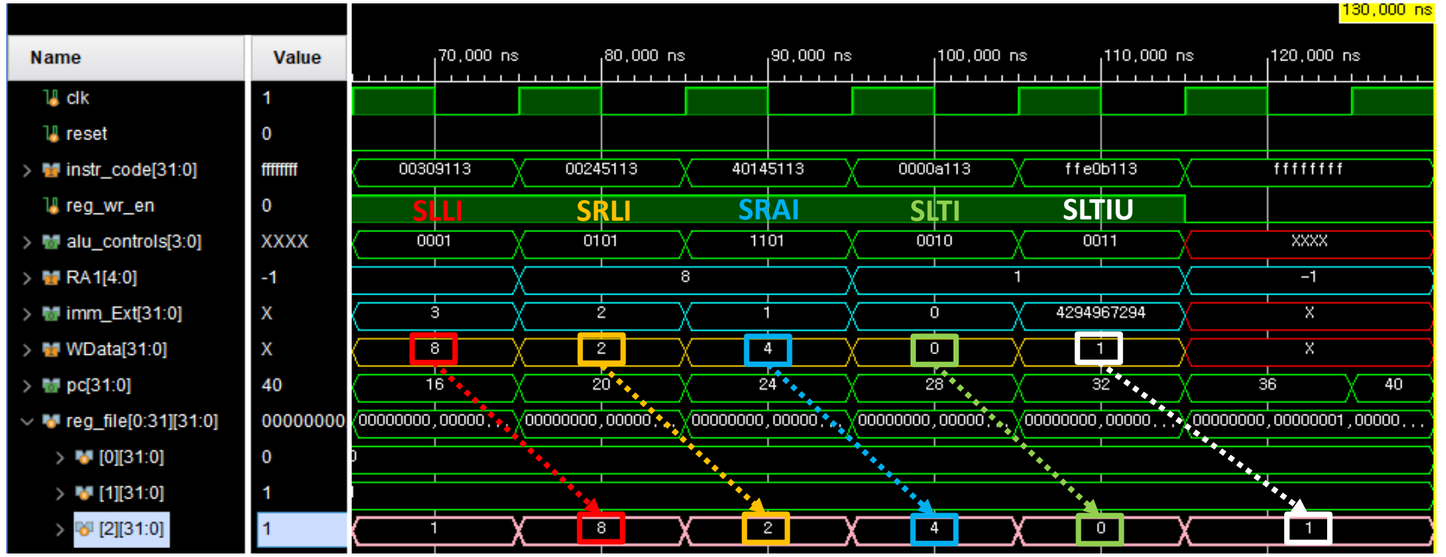
I-type



TYPE	ASSEMBLY	DESCRIPT	RESULT
I-type	addi x2, x1, 2	rd = rs1 + imm	rd ← 1 + 2 = 3
	xori x2, x1, 7	rd = rs1 ^ imm	rd ← 0001 ^ 0111 = 0110 = 6
	ori x2, x1, 7	rd = rs1 ¦ imm	rd ← 0001 ^ 0111 = 0111 = 7
	andi x2, x1, 7	rd = rs1 & imm	rd ← 0001 ¦ 0111 = 0001 = 1

03 Type별 기능 및 Simulation

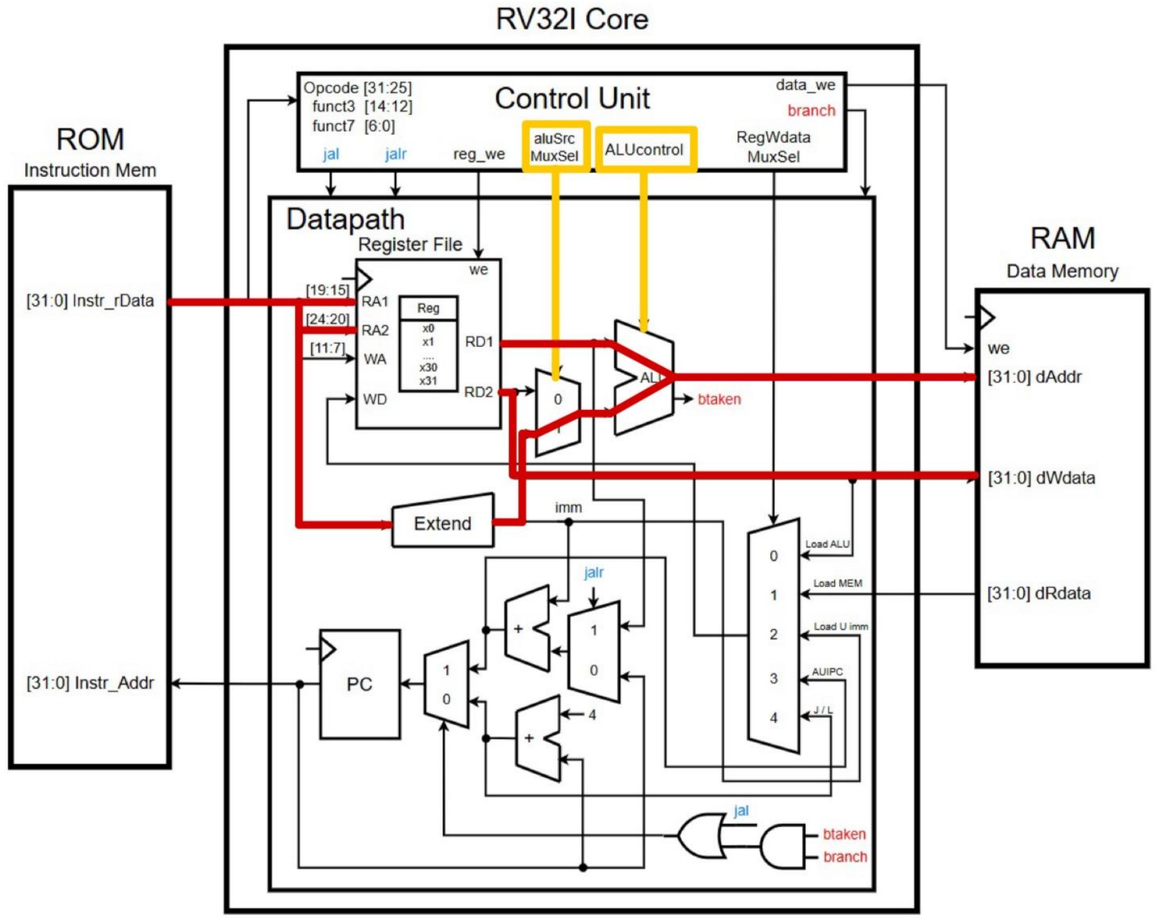
I-type



TYPE	ASSEMBLY	DESCRIPT	RESULT
I-type	slli x2, x1, 3	rd = rs1 << imm	rd ← 0001 << 3 = 1000 = 8
	srli x2, x8, 2	rd = rs1 >> imm	rd ← 1000 >> 2 = 0010 = 2
	srai x2, x8, 1	rd = rs1 >> imm (m-ex)	rd ← 1000 >> 1 = 0100 = 4
	slti x2, x1, 0	rd = (rs1 < imm) : 1 : 0	rd ← 1 < 0 = 0 “False”
	sltiu x2, x1, -2	rd = (rs1 < imm) : 1 : 0 (0-ex)	rd ← 1 < -2(Unsigned) = 1 “True”

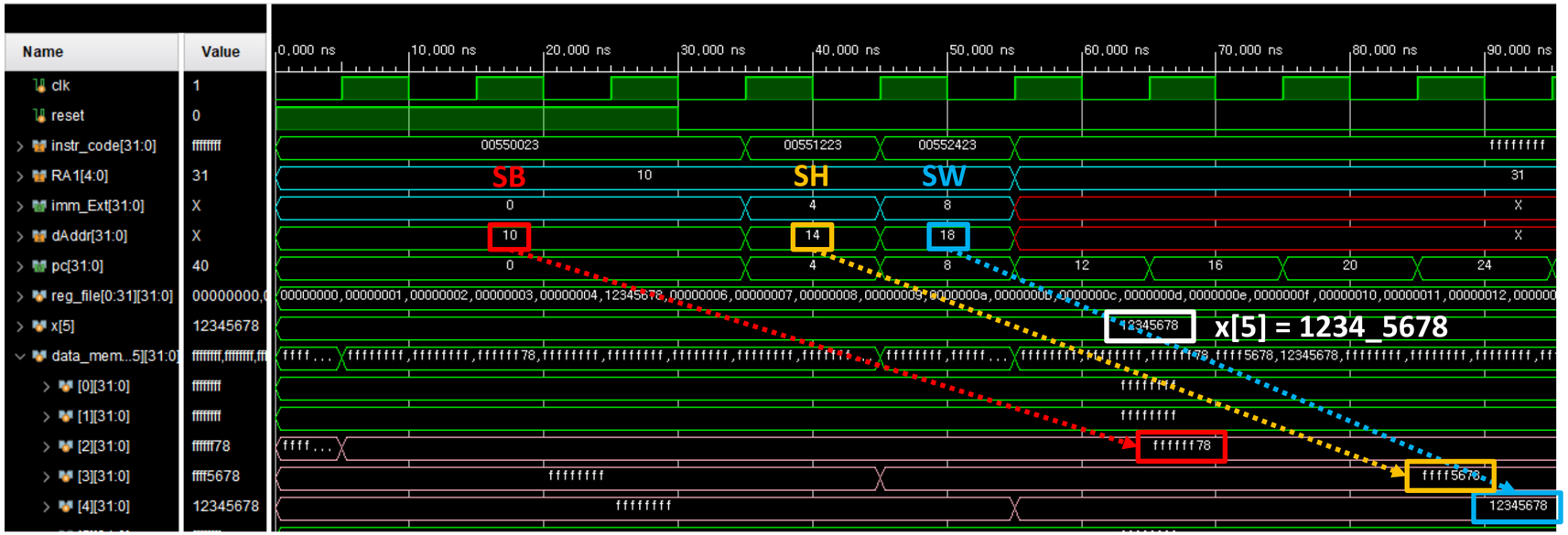
03 Type별 기능 및 Simulation

S-type



03 Type별 기능 및 Simulation

S-type



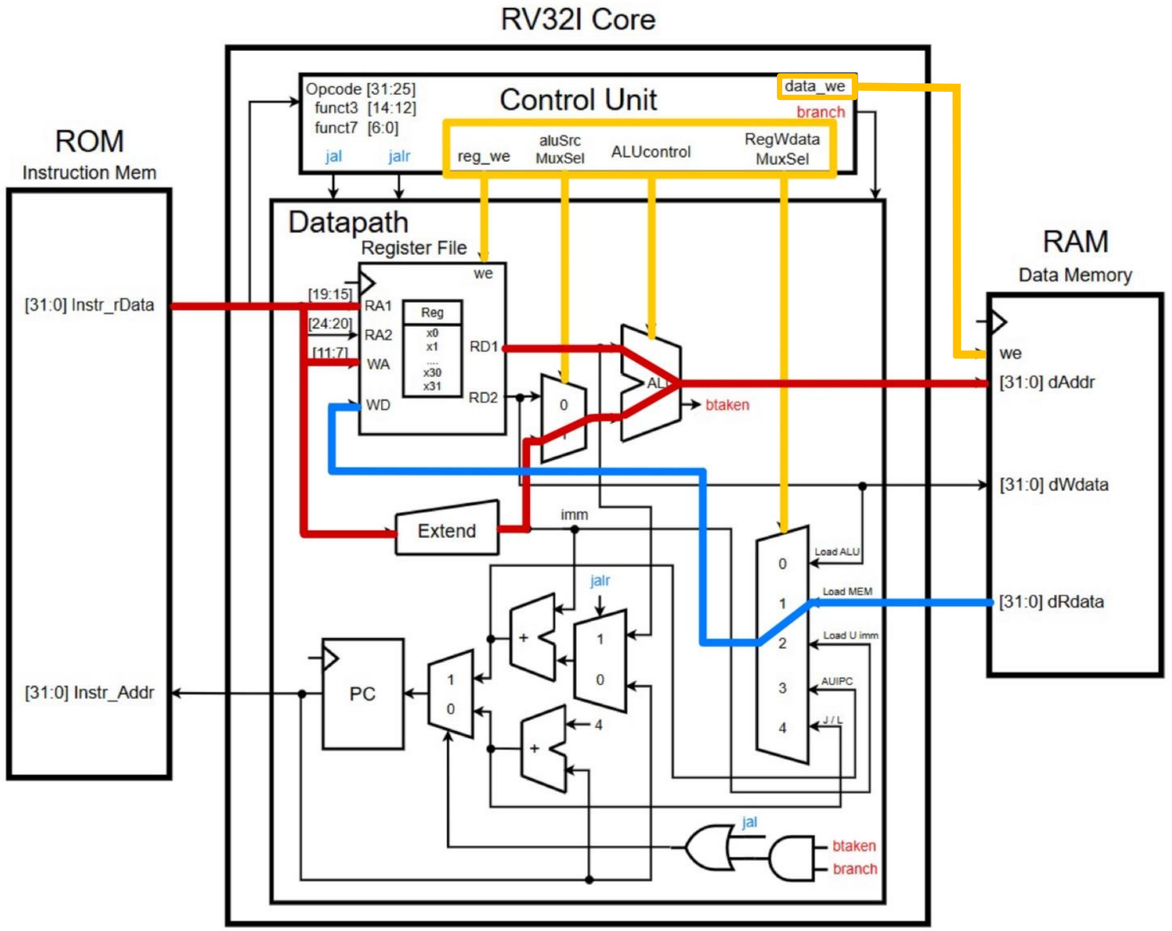
- **Word-aligned**

[바이트 주소]		[워드 인덱스]
0, 1, 2, 3	→	data_mem[0]
4, 5, 6, 7	→	data_mem[1]
8, 9, 10, 11	→	data_mem[2]
12, 13, 14, 15	→	data_mem[3]
16, 17, 18, 19	→	data_mem[4]

TYPE	ASSEMBLY	DESCRIPT	RESULT
S-type	sb x5, 0(x10)	M(rs1 + imm)[0:7] = rs2[0:7]	dAddr = 10 → data_mem[10/4] = data_mem[2]
	sh x5, 4(x10)	M(rs1 + imm)[0:15] = rs2[0:15]	dAddr = 14 → data_mem[14/4] = data_mem[3]
	sw x5, 8(x10)	M(rs1 + imm)[0:31] = rs2[0:31]	dAddr = 18 → data_mem[18/4] = data_mem[4]

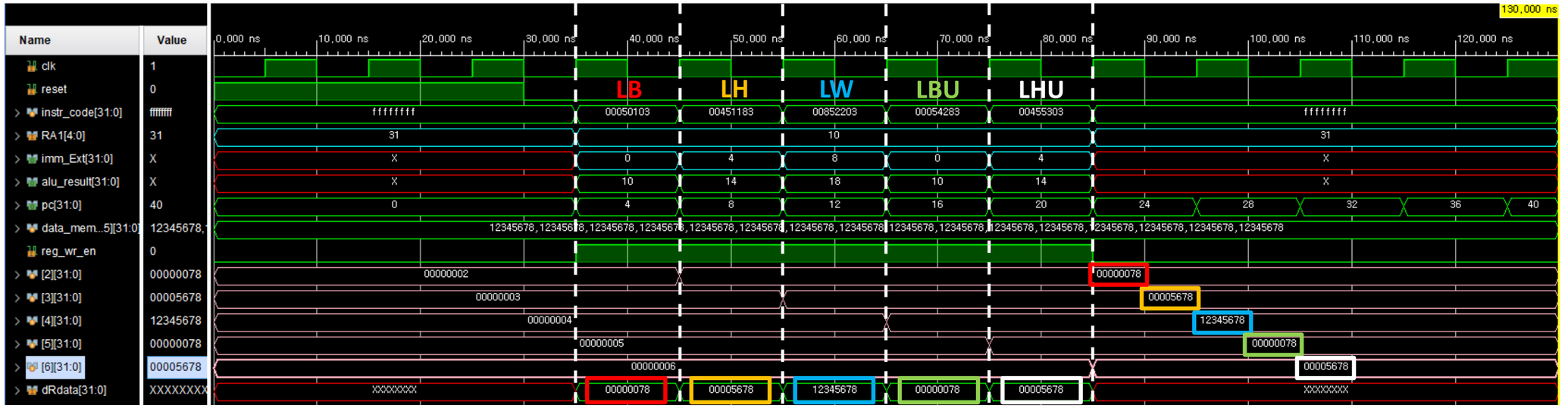
03 Type별 기능 및 Simulation

I-type Load



03 Type별 기능 및 Simulation

I-type Load

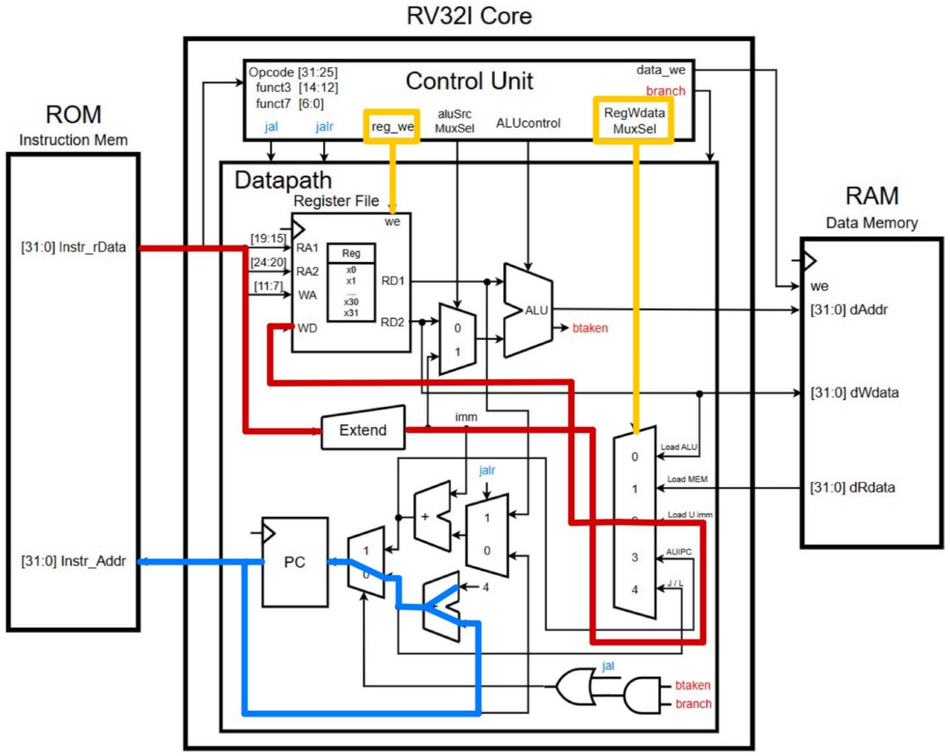


TYPE	ASSEMBLY	DESCRIPT	RESULT
I-type Load	lb x2, 0(x10)	rd = M(rs1 + imm)[0:7]	(dAddr = 10) x2 ← data_mem[2][7:0] = 0x0000_0078
	lh x3, 4(x10)	rd = M(rs1 + imm)[0:15]	(dAddr = 14) x3 ← data_mem[3][15:0] = 0x0000_5678
	lw x4, 8(x10)	rd = M(rs1 + imm)[0:31]	(dAddr = 18) x4 ← data_mem[4][31:0] = 0x1234_5678
	lbu x5, 0(x10)	rd = M(rs1 + imm)[0:7] (0-ex)	(dAddr = 10) x5 ← data_mem[2][7:0] = 0x0000_0078
	lhu x6, 4(x10)	rd = M(rs1 + imm)[0:15] (0-ex)	(dAddr = 14) x6 ← data_mem[3][15:0] = 0x0000_5678

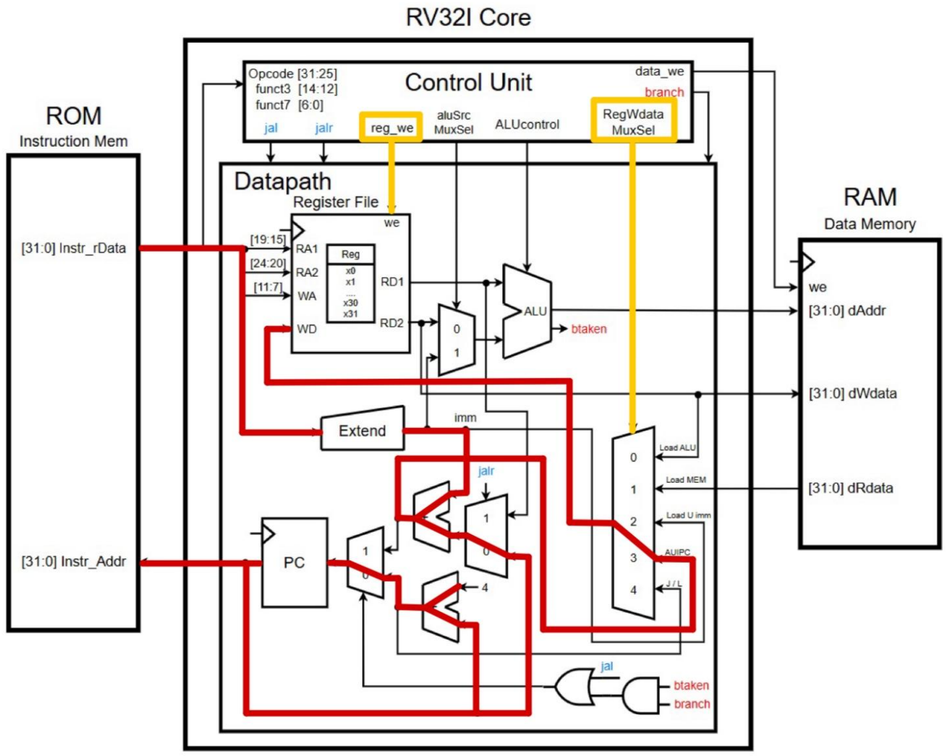
03 Type별 기능 및 Simulation

U-type

Lui

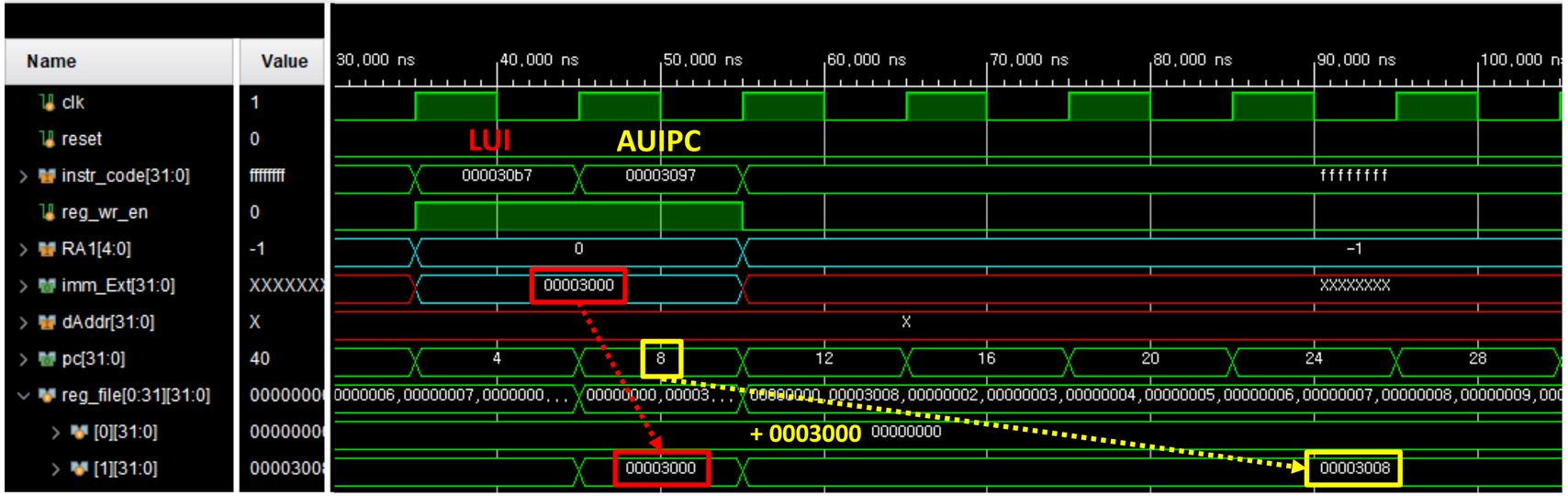


Auipc



03 Type별 기능 및 Simulation

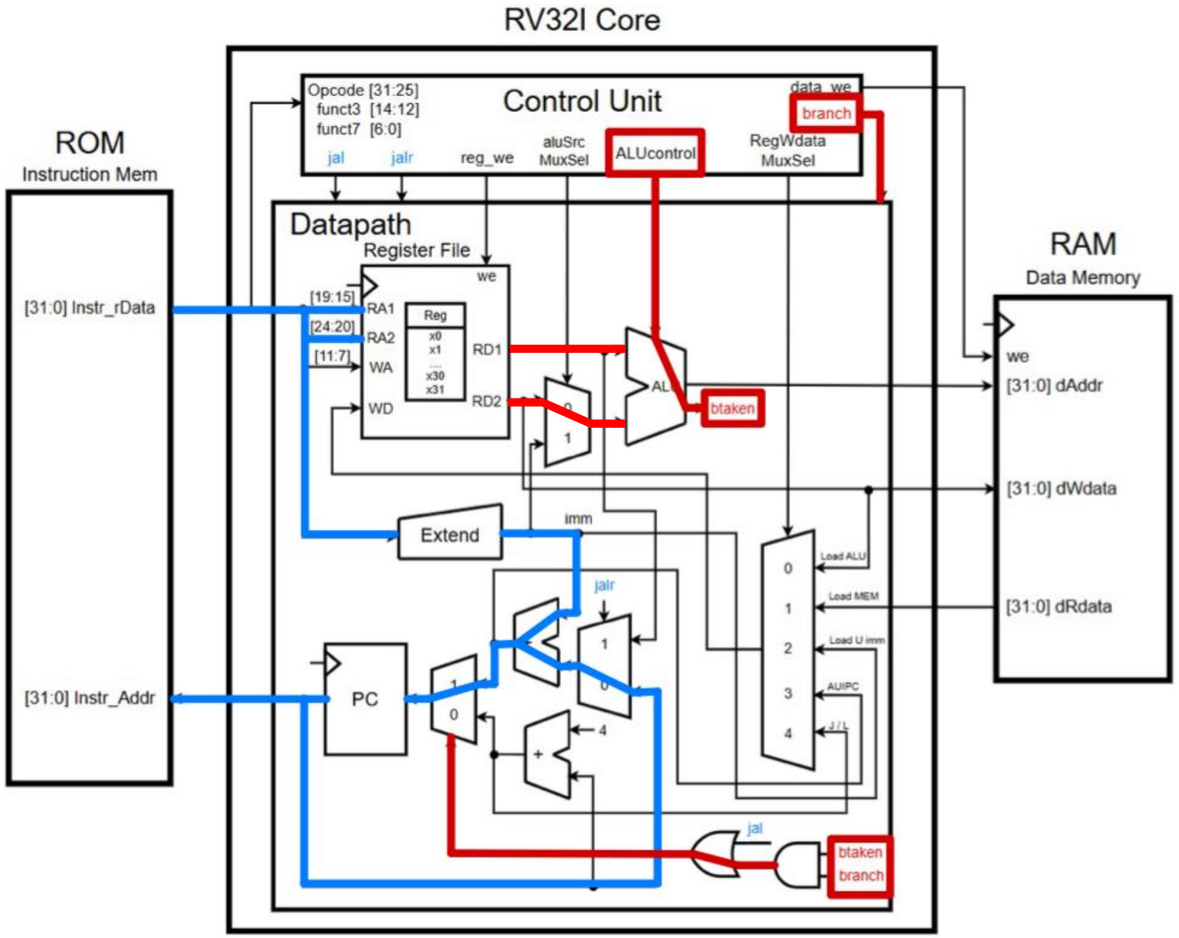
U-type



TYPE	ASSEMBLY	DESCRIPT	RESULT
U-type	lui x1, 3	rd = imm	rd ← imm = 00003000
	auipc x1, 3	rd = PC + imm	rd ← PC + imm = 00003008

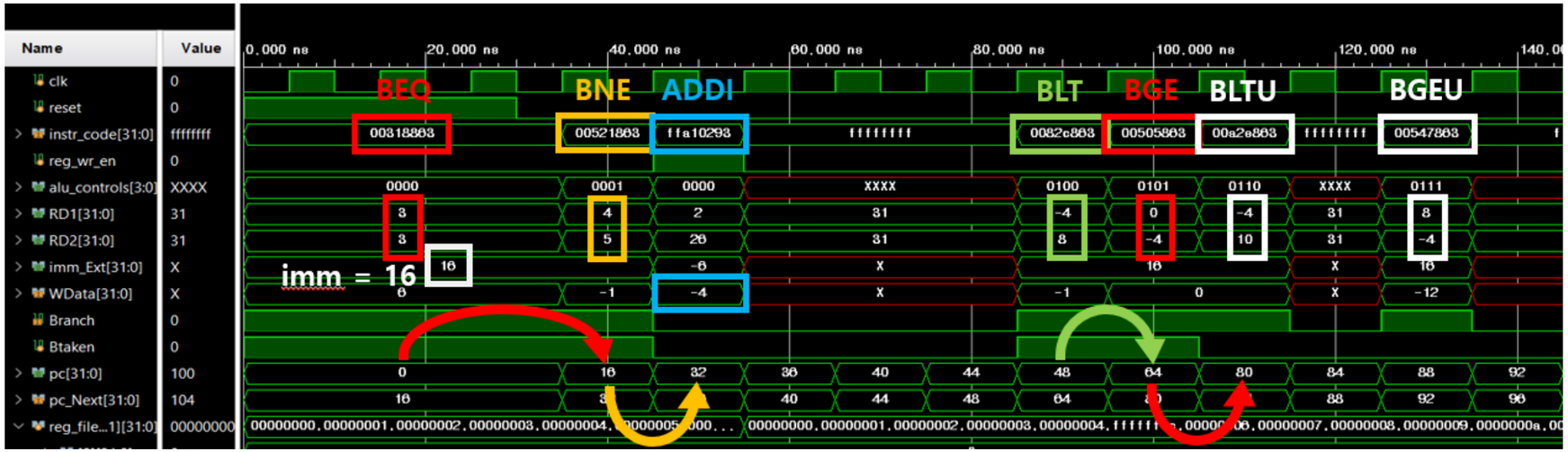
03 Type별 기능 및 Simulation

B-type



03 Type별 기능 및 Simulation

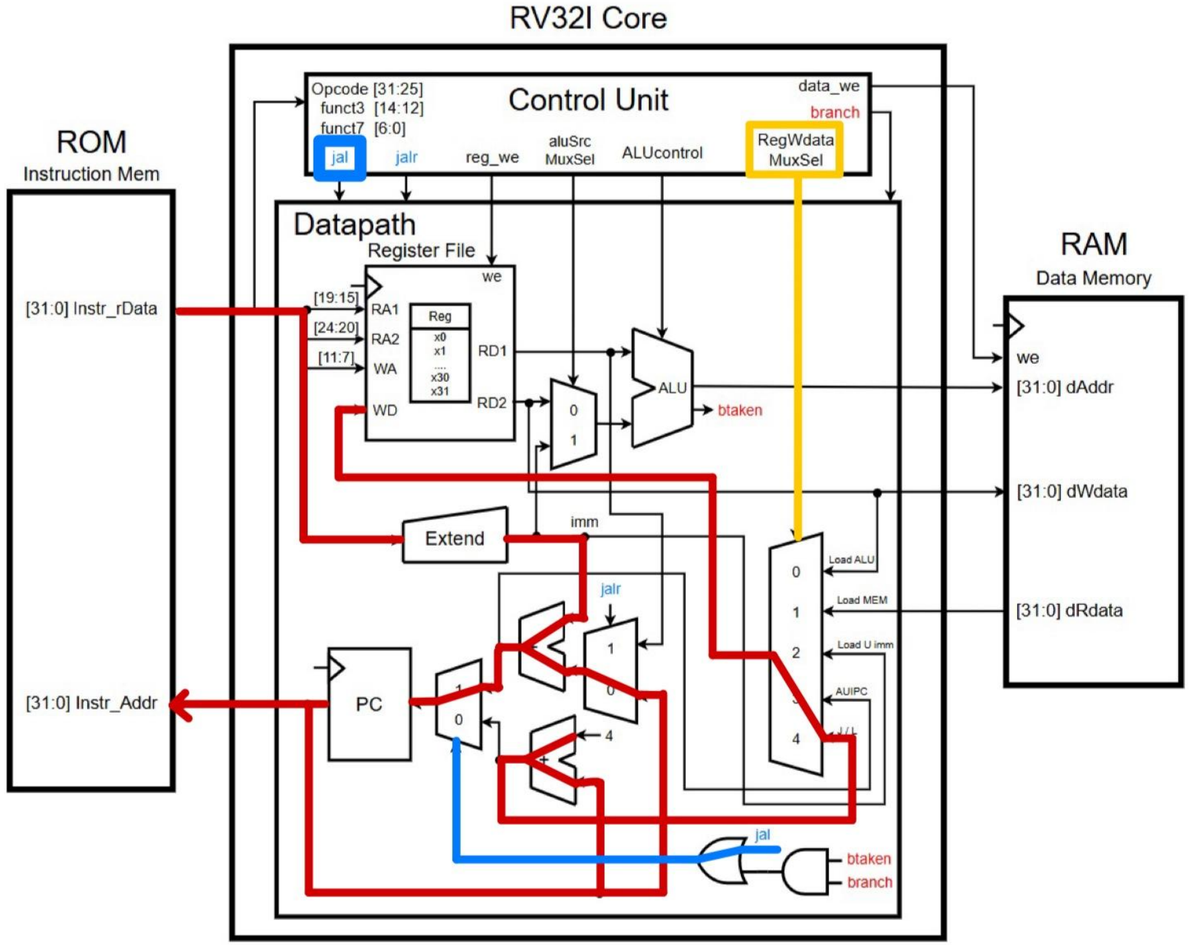
B-type



TYPE	ASSEMBLY	DESCRIPT	RESULT
B-type	beq x3, x3, 16	if(rs1 == rs2) PC += imm	(3 == 3) “True” PC ← 0 + 16 = 16
	bne x4, x5, 16	if(rs1 != rs2) PC += imm	(4 != 5) “True” PC ← 16 + 16 = 32
	blt x5, x8, 16	if(rs1 < rs2) PC += imm	(-4 < 8) “True” PC ← 48 + 16 = 64
	bge x0, x5, 16	if(rs1 >= rs2) PC += imm	(0 >= -4) “True” PC ← 64 + 16 = 80
	bltu x5, x10, 16	if(rs1 < rs2) PC += imm (0-ex)	(-4(Unsigned) < 10) “False” PC ← 80 + 4
	bgeu x8, x5, 16	if(rs1 >= rs2) PC += imm (0-ex)	(8 >= -4(Unsigned)) “False” PC ← 84 + 4

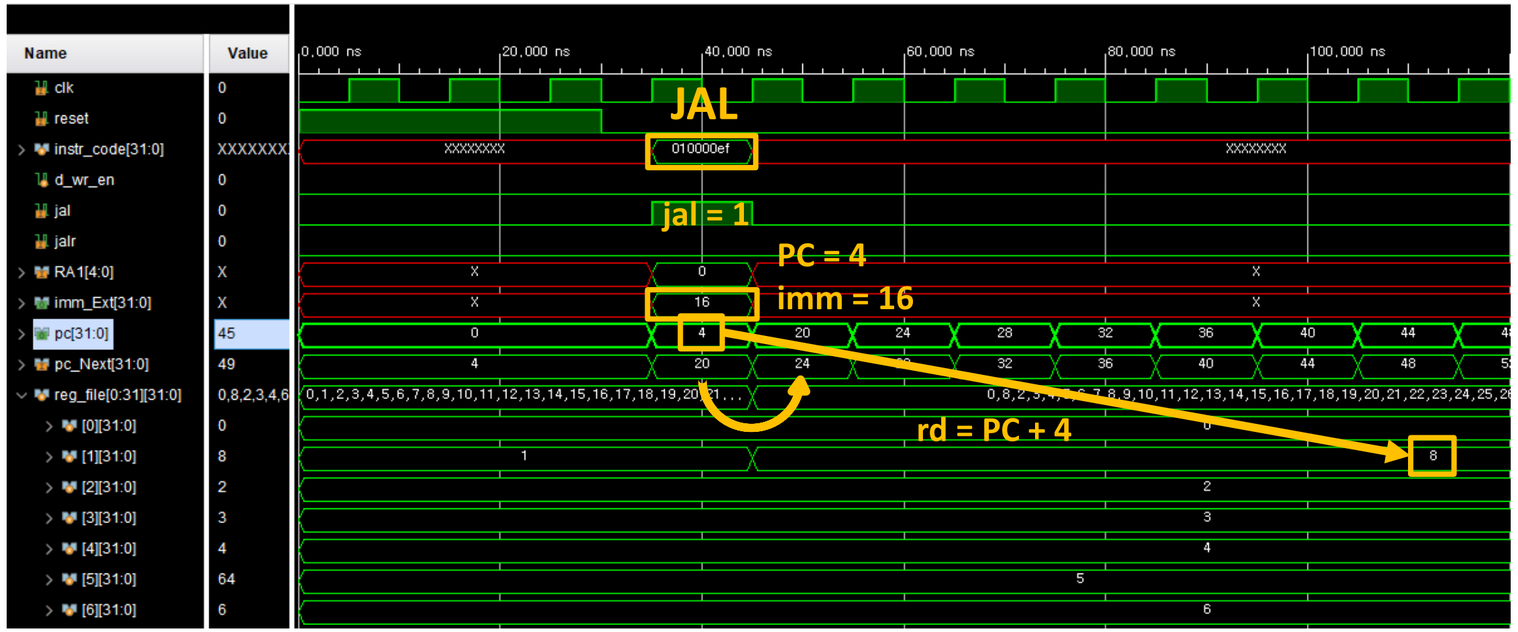
03 Type별 기능 및 Simulation

JAL-type



03 Type별 기능 및 Simulation

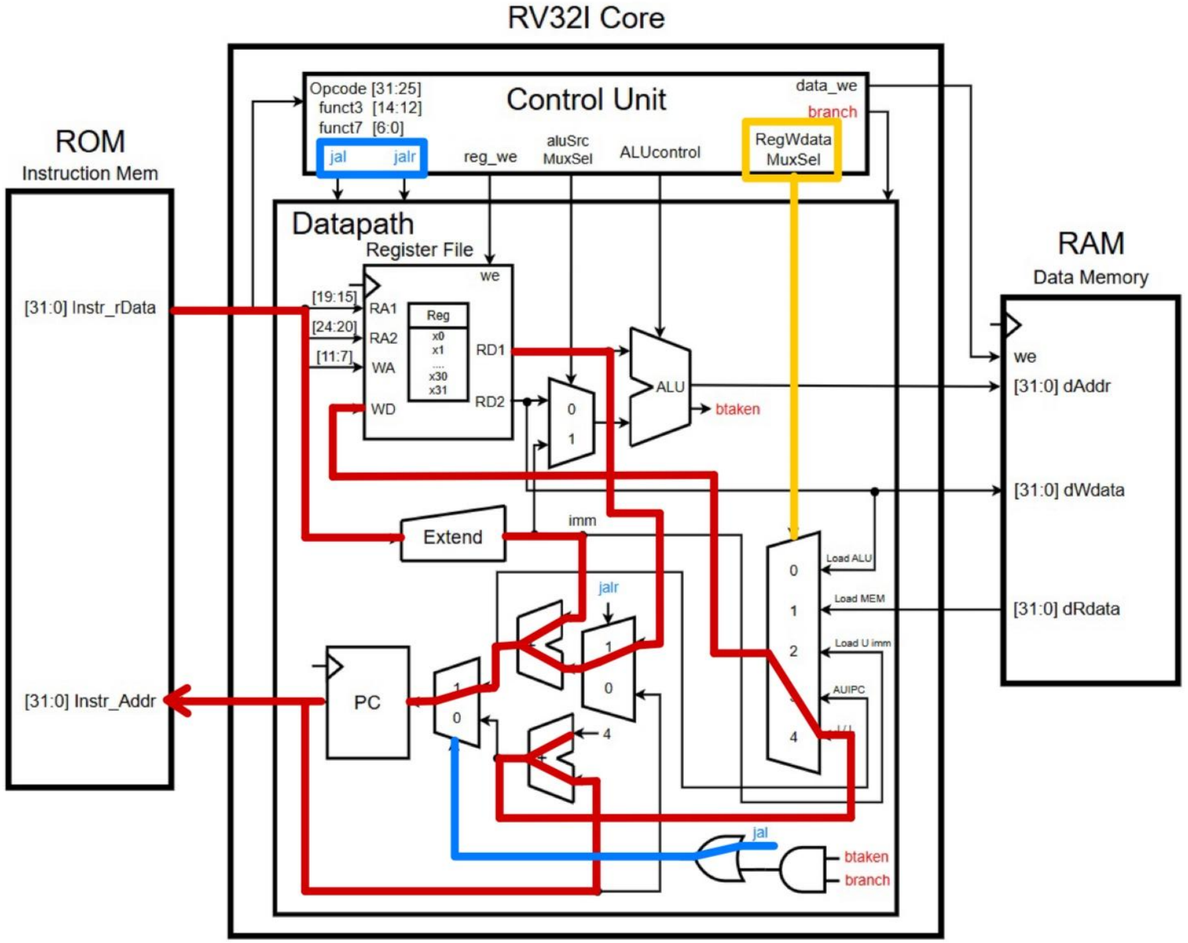
JAL-type



TYPE	ASSEMBLY	DESCRIPT	RESULT
J-type	jal x1, 16	rd = PC + 4; PC += imm	(jal = 1) PC ← 4 + 16 = 20

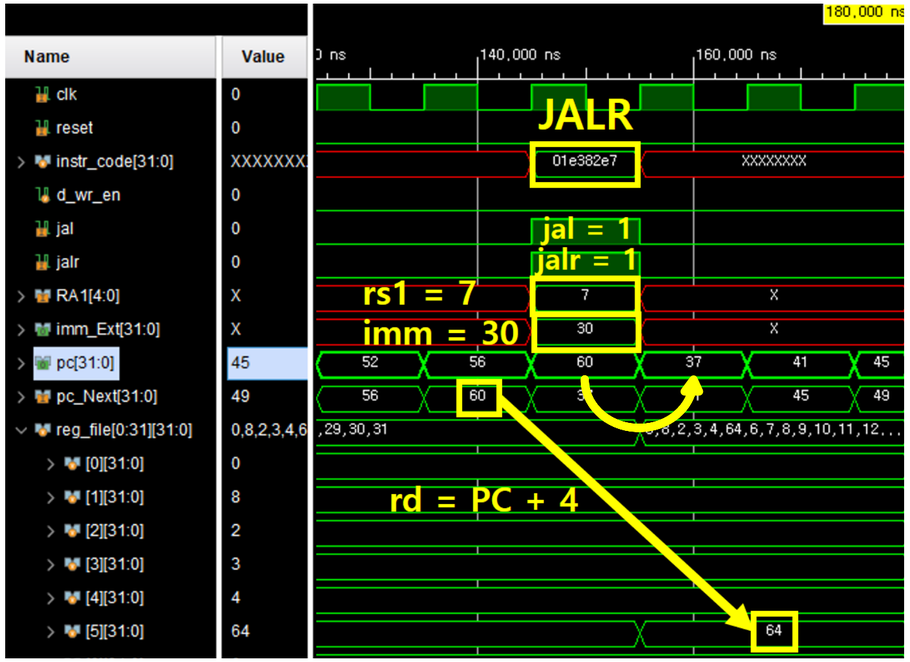
03 Type별 기능 및 Simulation

JALR-type



03 Type별 기능 및 Simulation

JALR-type



TYPE	ASSEMBLY	DESCRIPT	RESULT
J-type	jalr x5, 30(x7)	rd = PC + 4; PC = rs1 + imm	(jal = 1) & (jalr = 1) PC ← 4 + 16 = 20