Control signals — Lab 4 — MUX: rt – 0, rd – 1 — Controller — MUX: rt – 0, 1-Imm — MUX

		RegWrite	RegDst	ALUOp	ALUSrc	MemWrite	MemRead	MemToReg	Branch
ADD	rd, rs, rt	1	1	000000	0	0	0	1	0
Sub	rd, rs, rt	1	1	000000	0	0	0	1	0
And	rd, rs, rt	1	1	000000	0	0	0	1	0
Andi	rt, rs, imm (zero-extended)	1	0	001100	1	0	0	1	0
Addi	rt, rs, imm	1	0	001000	1	0	0	1	0
Mul	rd, rs, rt	1	1	011100	0	0	0	1	0
lw	rt, offset(base)	1	0	100011	1	0	1	0	0
sw	rt, offset(base)	0	0	101011	1	1	0	0 x	0
sb	rt, offset(base) — need to x4	0	0	101000	1	1	0	0 x	0
lh → half word	rt, offset(base)	1	0	100001	1	0	1	0	0
lb	rt, offset(base) — need to x4	1	0	100000	1	0	1	0	0
sh	rt, offset(base)	0	0	101001	1	1	0	0 x	0
bgez	rs, offset	0	0 x	000001		0	0		1
beq	rs, rt, offset	0	0 x	000100		0	0		1
bne	rs, rt, offset	0	0 x	000101		0	0		1
bgtz	rs, offset	0	0 x	000111		0	0		1
blez	rs, offset	0	0 x	000110		0	0		1
bltz	rs, offset	0	0 x	000001		0	0		1
j	target	0	0 - x	000010	0 x	0	0	0 x	0 x
jr	rs — add mux	0	0 - x	001000	0 x	0	0	0 x	0 x
jal	target	0	0 - x	000011	0 x	0	0	0 x	0 x

→ Jump Reg = 1 — we're adding a mux to do this

- → memory
- → arithmetic
- → control flow: branches & jumps

- I-Type
- R-Type
- J-Type

	RegWrite	RegDst	ALUOp	ALUSrc	MemWrite	MemRead	MemToReg
Or rd,rs,rt	1	0	100101	0	0	0	1
Nor rd,rs,rt	1	0	100111	0	0	0	1
Xor rd,rs,rt	1	0	100110	0	0	0	1
Ori rt,rs,imm			001101	1			
Xori rt,rs,imm			001110	1			
Sll rd,rt,sa	1		000000				
Srl rd,rt,sa	1		000010				
Slt rd,rs,rt	1		101010				
Slti rt,rs, imm			001010	1			

shift