1. Instruction Format

R-type					
funct7	rs2	rs1	funct3	rd	opcode
7	5	5	3	5	7
[31:25]	[24:20]	[19:15]	[14:12]	[11:7]	[6:0]

I-type				
immediate	rs1	funct3	rd	opcode
12	5	3	5	7
[31:20]	[19:15]	[14:12]	[11:7]	[6:0]

S-type					
imm[11:5]	rs2	rs1	funct3	imm[4:0]	opcode
7	5	5	3	5	7
[31:25]	[24:20]	[19:15]	[14:12]	[11:7]	[6:0]

2. ALU-control

funct7	funct3	opcode	function	ALU-control
0000000	110	0110011	OR	000
0000000	111	0110011	AND	001
0000000	000	0110011	ADD	010
0100000	000	0110011	SUB	011
0000001	000	0110011	MUL	100
X	000	0010011	ADDI	010

ALU action	ALU-control
AND	001
OR	010
ADD	011
SUB	100
MUL	101

for ALU-op,

00 means addition

01 means subtraction

10 depends on function code

11 not used

R-type					
opcode	instruction	funct7	funct3	ALU-action	ALU-control
0110011	and	0000000	111	AND	001
0110011	or	0000000	110	OR	010
0110011	add	0000000	000	ADD	011
0110011	sub	0100000	000	SUB	100
0110011	mul	0000001	000	MUL	101
I-type					
0010011	addi	X	000	ADD	011
0000011	lw	X	010	ADD	011
S-type					
0100011	sw	X	010	ADD	011
1100011	beq	X	000	SUB	100

funct_i	ALU-action	ALU-control
0011	AND	001
0010	OR	010
0000	ADD	011
1000	SUB	100
0100	MUL	101

如果是 R-type, ALU_Op = 10

這時看 function code: funct_i = {inst[30], inst[25], inst[13], inst[12]};(4-bit)

由上表得 ALU_Control

或者，柏序的做法:

function code: funct_i = {inst[30], inst[25], inst[14], inst[13], inst[12]};(5-bit)

由下表得 ALU_Control

funct_i	ALU-action	ALU-control
00111	AND	001
00110	OR	010
00000	ADD	011
10000	SUB	100
01000	MUL	101

如果不是 R-type

ALU 只會做加或減

ALU_Op = 00 代表加 (ALU_Control = 011)

ALU_Op = 01 代表減 (ALU_Control = 100)

3. Signal_Control.v

opcode	Instruction
0110011	R-type
0010011	addi
0000011	lw
0100011	sw
1100011	beq

for ALU-op,
  00 means addition
  01 means subtraction
  10 depends on function code
  11 not used

op	RegDst	ALUSrc	ResultSrc	RegWr	ALUOp	MemWr	Br
0110011	X	0	0	1	10	0	0
0010011	X	1	0	1	00	0	0
0000011	X	1	1	1	00	0	0
0100011	X	1	X	0	00	1	0
1100011	X	0	X	0	01	0	1

RegDst 是 Mips 的東西 Risc-V 用不到
ResultSrc 是自創的名字　市面上的名字是 MemtoReg