

10.1 - Control Unit

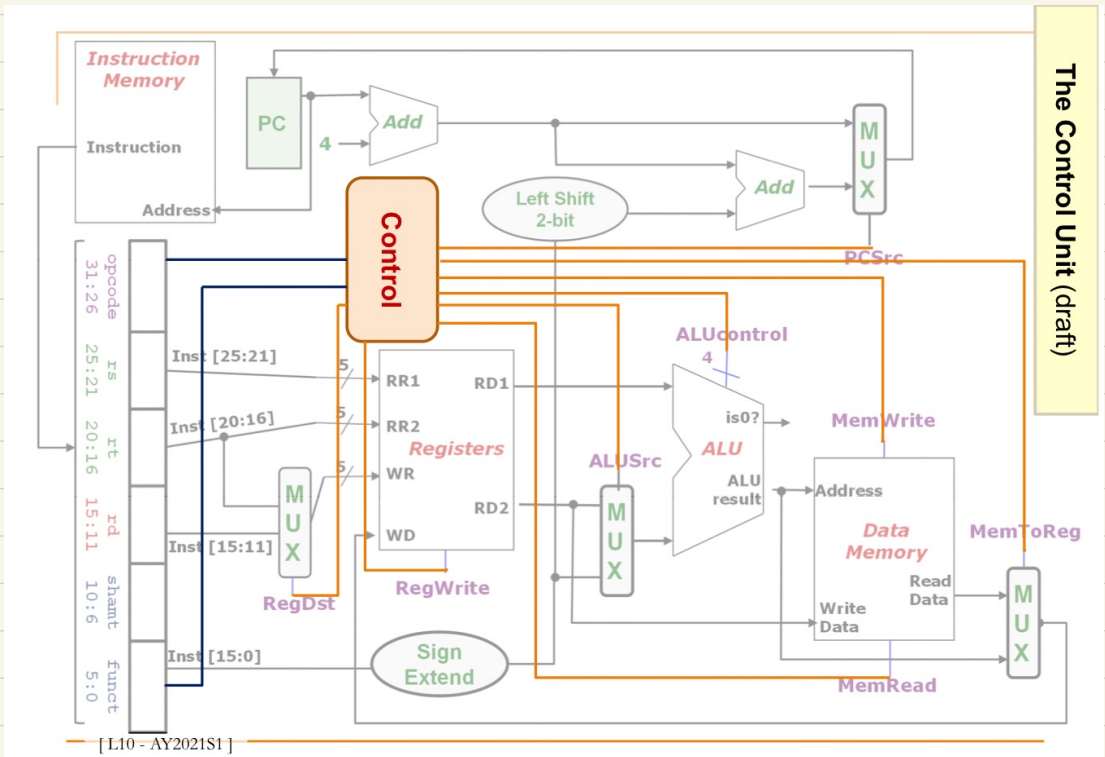
10.2 - Control Signals

- RegDst
- RegWrite
- ALUSrc
- Memread
- MemWrite
- MemToReg
- PCSrc
- ALUControl

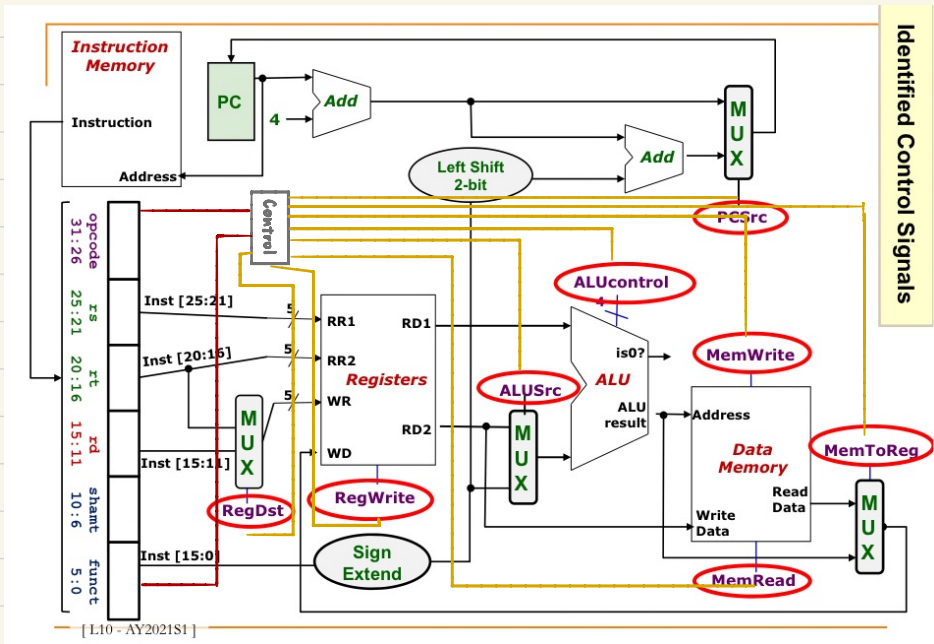
10.3 - Summary

- Truth Tables
 - Signals
 - ALU Control

10.1 - Control Unit



10-2 - Control Signals

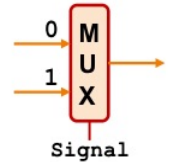


Control Signal	Execution Stage	Purpose
RegDst	Decode / Operand Fetch	Select the destination register number
RegWrite	Decode/Operand Fetch Result Write	Enable writing of register
ALUSrc	ALU	Select the 2 nd operand for ALU
ALUControl	ALU	Select the operation to be performed
MemRead / MemWrite	Memory	Enable reading/writing of data memory
MemToReg	Result Write	Select the result to be written back to register file
PCSrc	Memory / Result Write	Select the next PC value

RegDst

Control Signal: **RegDst** *rd?*

- **False (0)**: Write register = **Inst**[**20:16**]
- **True (1)**: Write register = **Inst**[**15:11**]



RegWrite

Control Signal: **RegWrite**

- **False (0)**: No register write
- **True (1)**: New value will be written

ALUSrc

Control Signal: **ALUSrc** *Is source immediate?*

- **False (0)**: Operand2 = Register Read Data 2
- **True (1)**: Operand2 = SignExt(**Inst**[**15:0**])

Memread

Control Signal: **MemRead** ?

- **False (0)**: Not performing memory read access
- **True (1)**: Read memory using **Address**

MemWrite

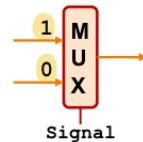
Control Signal: **MemWrite**

- **False (0)**: Not performing memory write operation
- **True (1)**: memory[**Address**] \leftarrow Register Read Data 2

MemToReg

Control Signal: **MemToReg** *Store mem in reg?*

- **True (1)**: Register write data = Memory read data
- **False (0)**: Register write data = ALU Result



IMPORTANT:
The input of MUX
is swapped in this
case

PCSrc

Control Signal: PCSrc PCSrc from BAT?

- False (0): Next PC = PC + 4
- True (1): Next PC = SignExt(Inst[15:0]) << 2 + (PC + 4)

PCSrc =
(Branch AND
isZero)

① isBranch?

ϕ : next

1 : maybe BTA

② isCondition

1 : taken

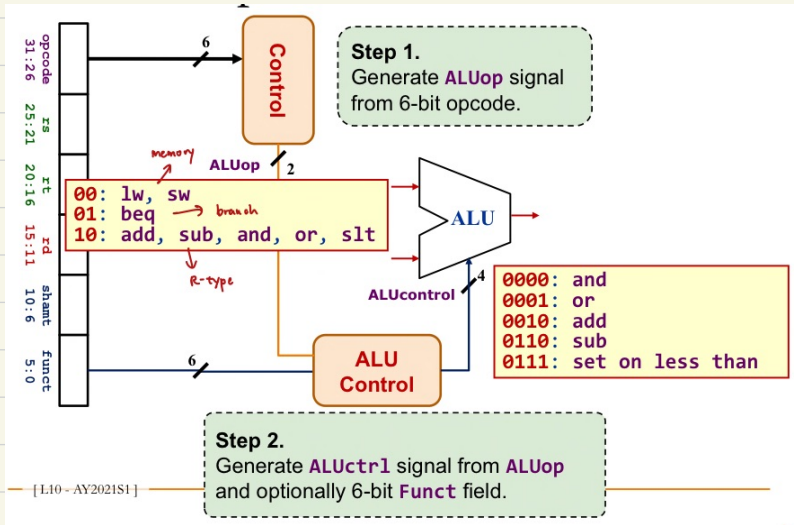
ϕ : next

①	②	(AND) taken?
1	1	1
ϕ	1	ϕ
1	ϕ	ϕ
ϕ	ϕ	ϕ

ALU Control

Multilevel decoding approach :

Use opcode (6b) and function code (6b) ↗ for R-type



10.3 - Summary Signals

Truth table for various signals

PCSrc = Branch AND isZero

0: false
1: true
X: don't care
(doesn't matter whether true/false)

	RegDst	ALUSrc	MemToReg	RegWrite	MemRead	MemWrite	Branch	ALUop	
								op1	op0
R-type	1	0	0	1	0	0	0		
lw	0	1	1	1	1	0	0		
sw	X	1	X	0	0	1	0		
beq	X	0	X	0	0	0	1		



ALU Control

Opcode	Instruction Operation	ALU action
lw	load word	add
sw	store word	add
beq	branch equal	subtract
R-type	add	add
R-type	subtract	subtract
R-type	AND	AND
R-type	OR	OR
R-type	set on less than	set on less than

ALUcontrol	Function
0000	AND
0001	OR
0010	add
0110	subtract
0111	slt
1100	NOR

Instruction Type	ALUop
lw / sw	00
beq	01
R-type	10

	ALUop		Funct Field (F[5:0] == Inst[5:0])						ALU control
	MSB	LSB	F5	F4	F3	F2	F1	F0	
lw	0	0							add 0010
sw	0	0							add 0010
beq	0	1							sub 0110
add	1		1	0	0	0	0	0	add 0010
sub			1	0	0	0	1	0	sub 0110
and		0	1	0	0	1	0	0	and 0000
or			1	0	0	1	0	1	or 0001
slt			1	0	1	0	1	0	slt 0111