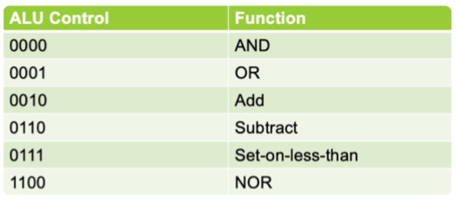
Control Signals

The ALU control involves a subset of instructions: lw, sw, beq, add. addi, sub, and, or slt. These instructions use the ALU because they involve arithmetic or logical operations, or they need to compare or calculate addresses.

- add, addi, sub, and, or, slt: directly perform arithmetic or logical operations
- lw, sw: calculate memory addresses by adding offset to base register.
- beq: compares 2 registers to check for equality.

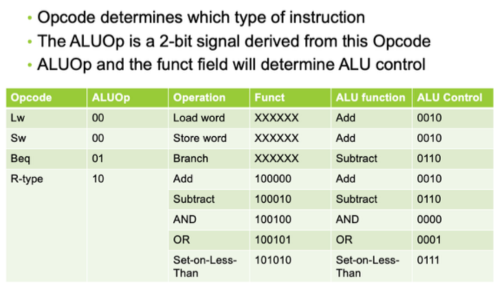
With 4 control inputs, the ALU can perform 16 possible operations for example →
Even though MIPS has many instructions, most only need a few core operations.



ALU Control	Function
0000	AND
0001	OR
0010	Add
0110	Subtract
0111	Set-on-less-than
1100	NOR

The ALU control varies by instruction type: Load/Store uses add, Branch uses subtract, and R-type depends on the funct field.

The opcode determines which type of instruction. Based on the Opcode, the control unit sets a 2-bit ALUop signal. If the ALUop is 00 (lw or sw), the ALU just performs add to compute memory address. If the opcode is 10 (R-type), the ALU control depends on the funct field. If the ALUop is 01 (beq) the ALU peforms subtraction to compute the value.

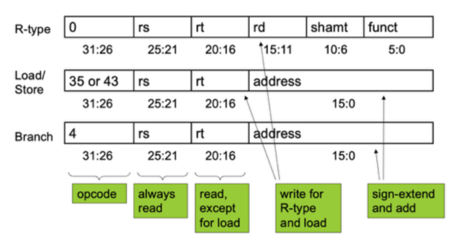


- Opcode determines which type of instruction
- The ALUOp is a 2-bit signal derived from this Opcode
- ALUOp and the funct field will determine ALU control

Opcode	ALUOp	Operation	Funct	ALU function	ALU Control
Lw	00	Load word	XXXXXX	Add	0010
Sw	00	Store word	XXXXXX	Add	0010
Beq	01	Branch	XXXXXX	Subtract	0110
R-type	10	Add	100000	Add	0010
		Subtract	100010	Subtract	0110
		AND	100100	AND	0000
		OR	100101	OR	0001
		Set-on-Less-Than	101010	Set-on-Less-Than	0111

The Main Control Unit

- Control signals derived from instruction



The layout shows how the control unit extracts key fields from each instruction.

Control Signals

Register Destination (RegDst): Determines the register destination number for the write register. If desserted, it comes from the rt field. If asserted, it comes from rd field. Typically asserted for r-type instructions.

Register Write (RegWrite): When asserted, the destination is specified by the

Datapath with Control

Instruction [31-26] (opcode) determines the control signals needed to execute the instruction. The control unit uses these bits to set up the datapath components accordingly.

Register Read Addresses:
- Bits [25-21] specify register source 1 (rs)
- Bits [20-16] specify register source 2 (rt)

These fields are used to access the correct values from the register file for operation. like ALU execution or storing data.

Sign-Extension: for instructions that use immediate values (like lw, sw, addi, beq), the 16-bit immediate in [15-0] is sign-extended to 32 bits. This ensures correct handling of negative and pasotive values in 32 bit operations.

The load instruction uses control signals MemtoReg and ALUSrc to select correct data paths. Registers are accessed and data is read or written based on instruction setup.

The branch-on-equal instruction reads two registers and uses the ALU to compare them. The zero output determines branching, controlled by the opcode.

Op[5-0] determines control outputs. Key signals include RegDst, ALUSrc, and Branch.

Jump instructions use a concatenated address from the PC's top 4 bits and the shifted jump address. The opcode controls this thru the dump signal. The data path with Jumps added includes instruction fetch, ALU, and memory access.

In single-cycle implementation, each instruction completes in one clock cycle. This requires a long cycle time and duplicated hardware, making it simple but inefficient for complex tasks.

Examples

Select the control signals that will be generated by the instruction: beq, $t0, $t1, 4
- This is a branch-if-equal, I-type.
- Since it isnt a jump instruction set that signal to 0
- Since this is a conditional branch, set the branch signal to 1.

Control Signal Overview

Reg Write
- 1: Write to the register file
- 0: Do not write to the register file (writing is disabled but but the signal matters like sw)
- X: Use only when the signal isnt being used at all (like beq)

Reg Dst
- 1: Destination register is rt, which is used as the destination register in I-type instructions (lw, addi). These use 2 operands: one register source and one immediate or memory offset, and the result gets stored in rt.
- 0: Destination register is rd, which is used as the destination register in R-type instructions (add, sub). These use two register inputs and put the result into a third register, rd.
- X: Only when RegWrite = 0 or X - if we arent writing to the register file, it doesnt matter which register would have been chosen as the destination.

Mem Read
- 1: Tells the processor to read data from memory (like lw)
- 0: No memory read happens, nothing needs to be pulled from memory.
- X: This signal is always either 0 or 1. Control logic must explicitly decide whether to read from memory or not.

Mem Write
- 1: Enables writing data to memory (like sw)
- 0: No memory write
- X: Never valid, you must always explicitly control wether memory is written to

Memto Reg:
- 1: The value written to the register comes from memory (lw)
- 0: The value comes from the ALU
- X: Valid only when RegWrite = X or 0 - if we arent writing to the register file, the source doesnt matter.

ALUSrc
- 1: The second ALU input comes from an immediate value
- 0: The second ALU input comes from a register
- X: Never valid, the ALU must know where input is coming from

ALU Op
- Write the operation the ALU would do. X is never valid, ALU needs a real instruction.

Branch
- 1: Used in branch instructions like beq
- 0: Not a branch
- X: Never valid

Jump
- 1: Used in jump instructions like j, jal
- 0: Not a jump