1 Single-cycle datapath control

The only difference between the instructions are that the `lwd` adds to registers together (`ALUSrc`) and writes back to register `$rd` (`RegDst`).

inst	PCSrc	ALUSrc	ALUOp	MemWrite	MemRead	MemToReg	RegDst	RegWrite
`lw`	0	1	ADD	0	1	1	0	1
`lwd`	0	0	ADD	0	1	1	1	1

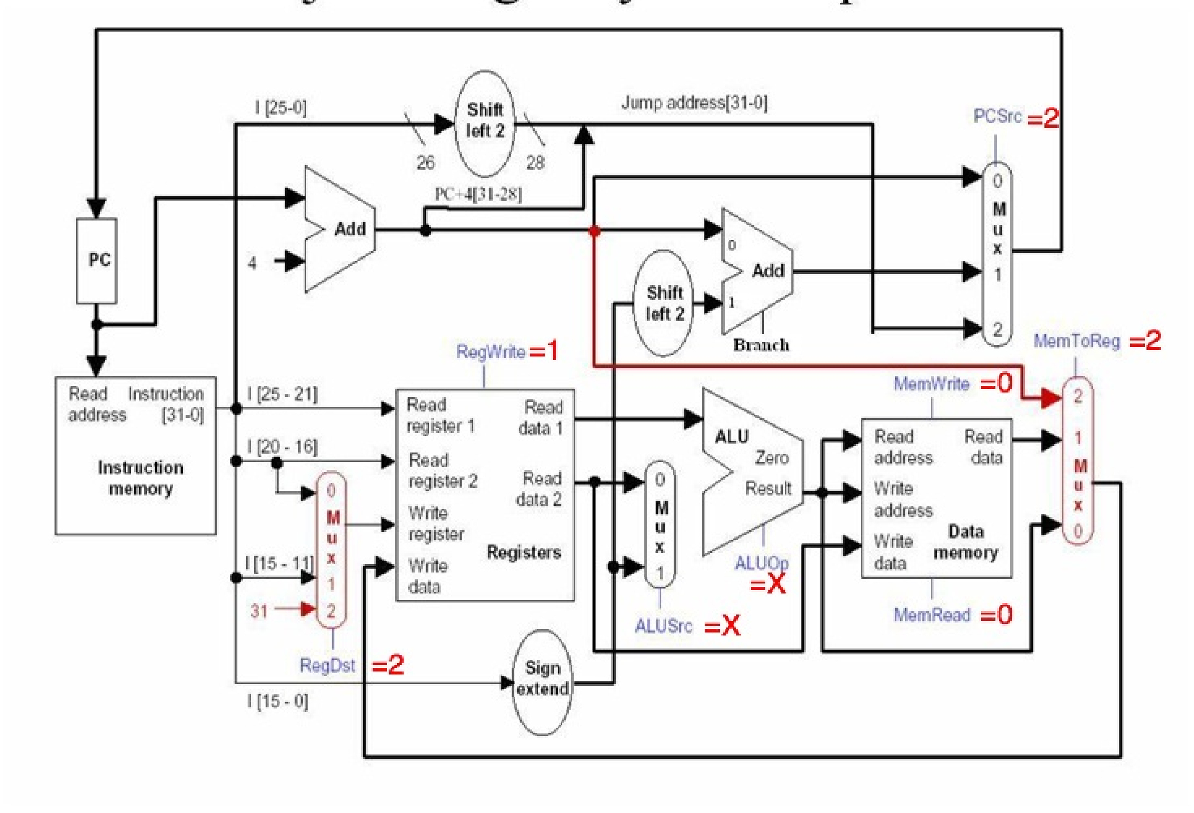
2 Single-cycle datapath modifications

Implementing the `jal` requires a number of datapath additions. First a 26-bit immediate needs to be extracted and shifted 2 bits to the left to create a 28-bit number. To create a 32-bit PC, the top 4 bits of the current PC are merged in. The `PCSrc` multiplexor needs to be extended with a third input.

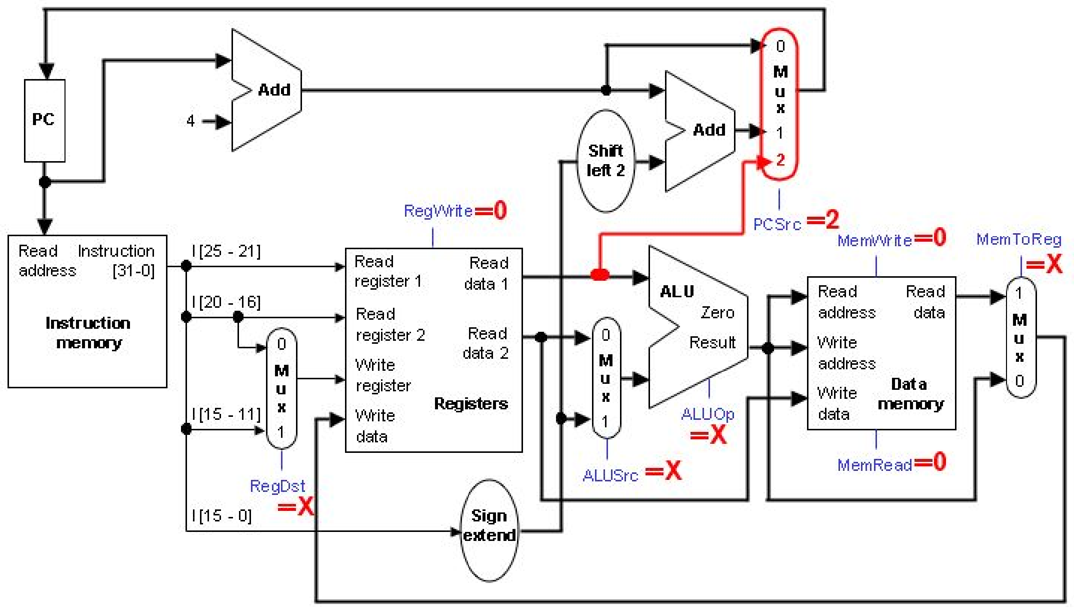
Second, the return address (`PC+4`) needs to be routed to the `Write data` port of the register file. This can be done by extending the `MemToReg` mux.

Finally, the return address needs to be written to register 31, but there is no guarantee that the `rs`, `rt`, or `rd` fields will hold the value 31, so we have to have a "hard-coded" 31 and extend the `RegDst` mux to permit its selection.

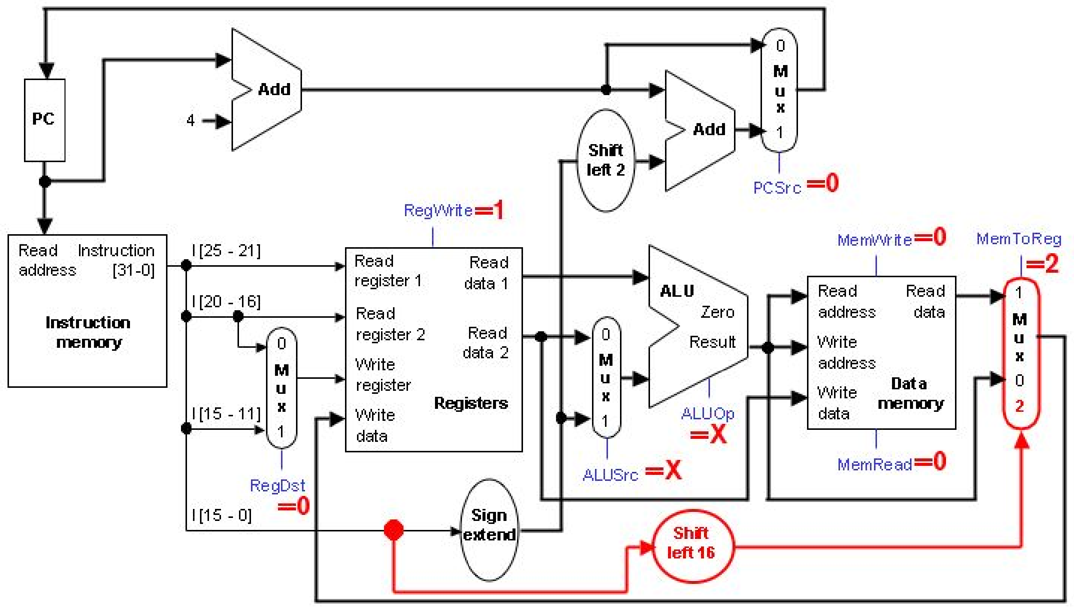
jal - single-cycle datapath



1. **Implementing `jr`:** Expand the `PcSrc` mux to take a third input (thus, it becomes a two-bit control signal). The third input will be from the `Read data 1` line – see the diagram:



2. **Implementing `lui`:** Insert a "Shift Left by 16" unit which takes the 16-bit immediate as its input (it doesn't matter whether this is before or after the "Sign Extend" unit). The resulting 32-bit signal is routed to the 3rd input of an extended `MemToReg` multiplexor.


PC
Add
4
Add
Shift left 2
Mux 0 1
PCSrc =0
RegWrite=1
MemWrite=0
MemToReg =2
Read address
Instruction [31-0]
Instruction memory
I [25 - 21]
I [20 - 16]
I [15 - 11]
I [15 - 0]
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers
Mux 0 1
RegDst =0
Sign extend
Shift left 16
Mux 0 1
ALUSrc =X
ALU Zero Result
ALUOp =X
Read address
Write address
Write data
Read data
Data memory
MemRead=0
Mux 1 0 2