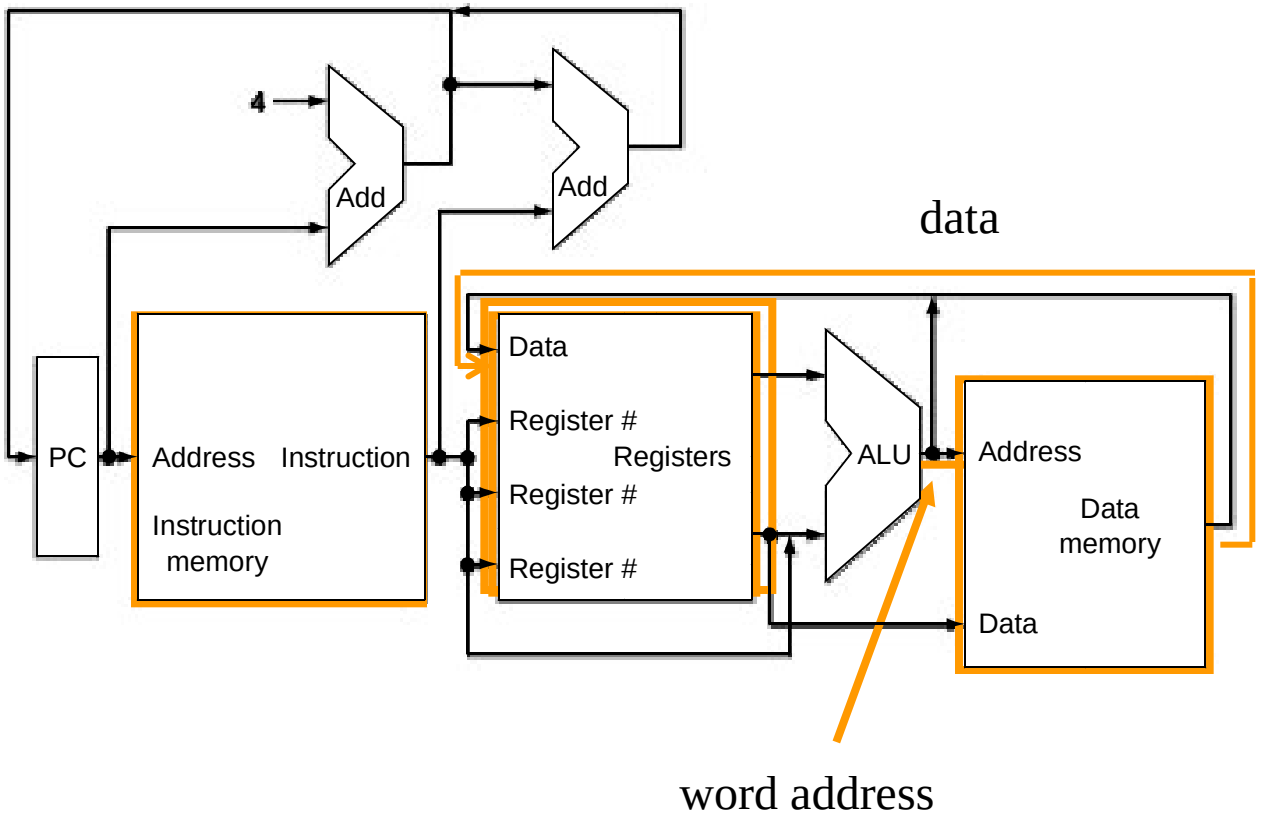
Grader

- The grader works 4 pm – 5 pm every Monday and Wednesday at GMCS 557.

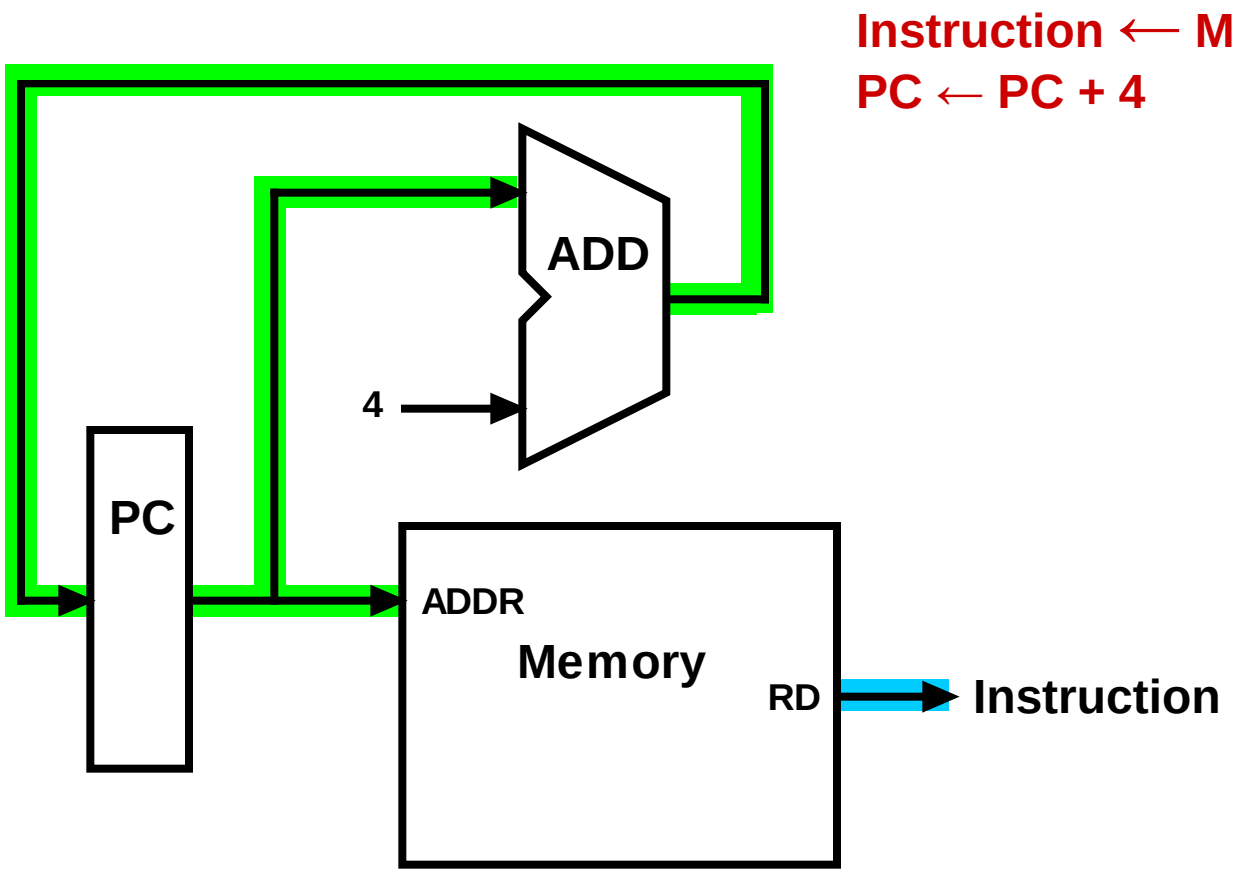
- Some students did not send him an Hello email. Please do it by today. Thanks!

Animation Sequence for lw



- Step 1: Fetch Instruction
- Step 2: Compute address
- Step 3: Read data from memory
- Step 4: Write to register

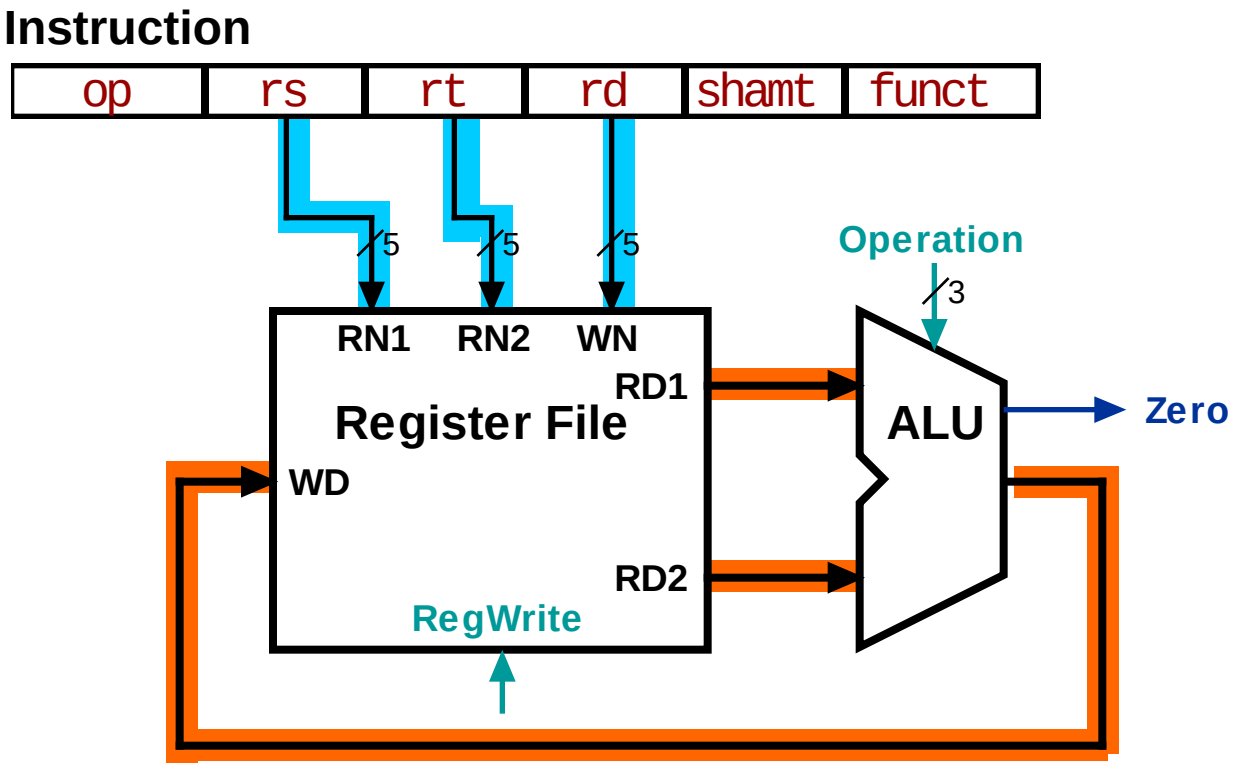
Datapath for Instruction Fetch



Datapath for R-Type Instructions

add rd, rs, rt

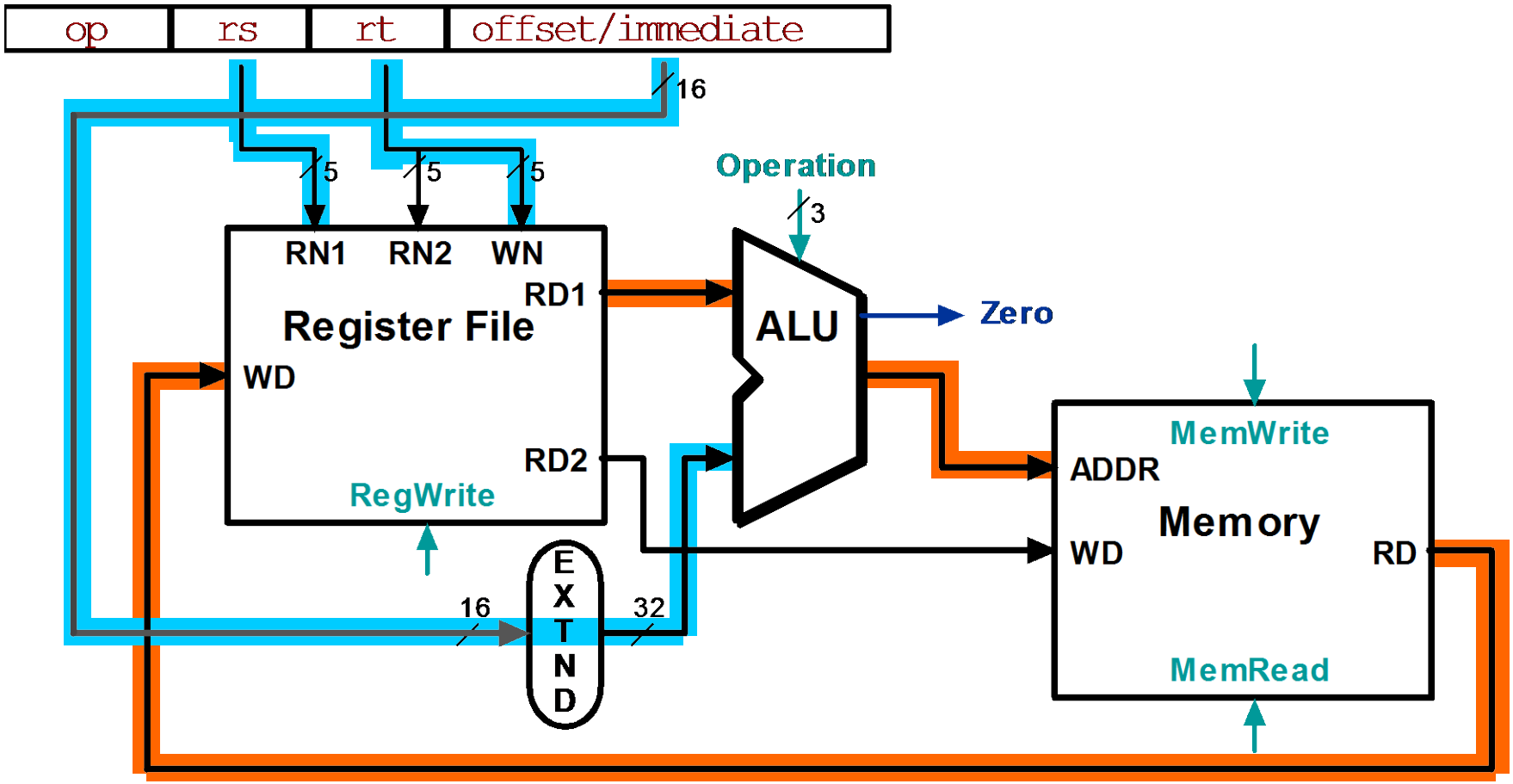
R[rd] ← R[rs] + R[rt];



Datapath for Load Instructions

lw rt, offset(rs)

R[rt] ← MEM[R[rs] + s_extend(offset)];

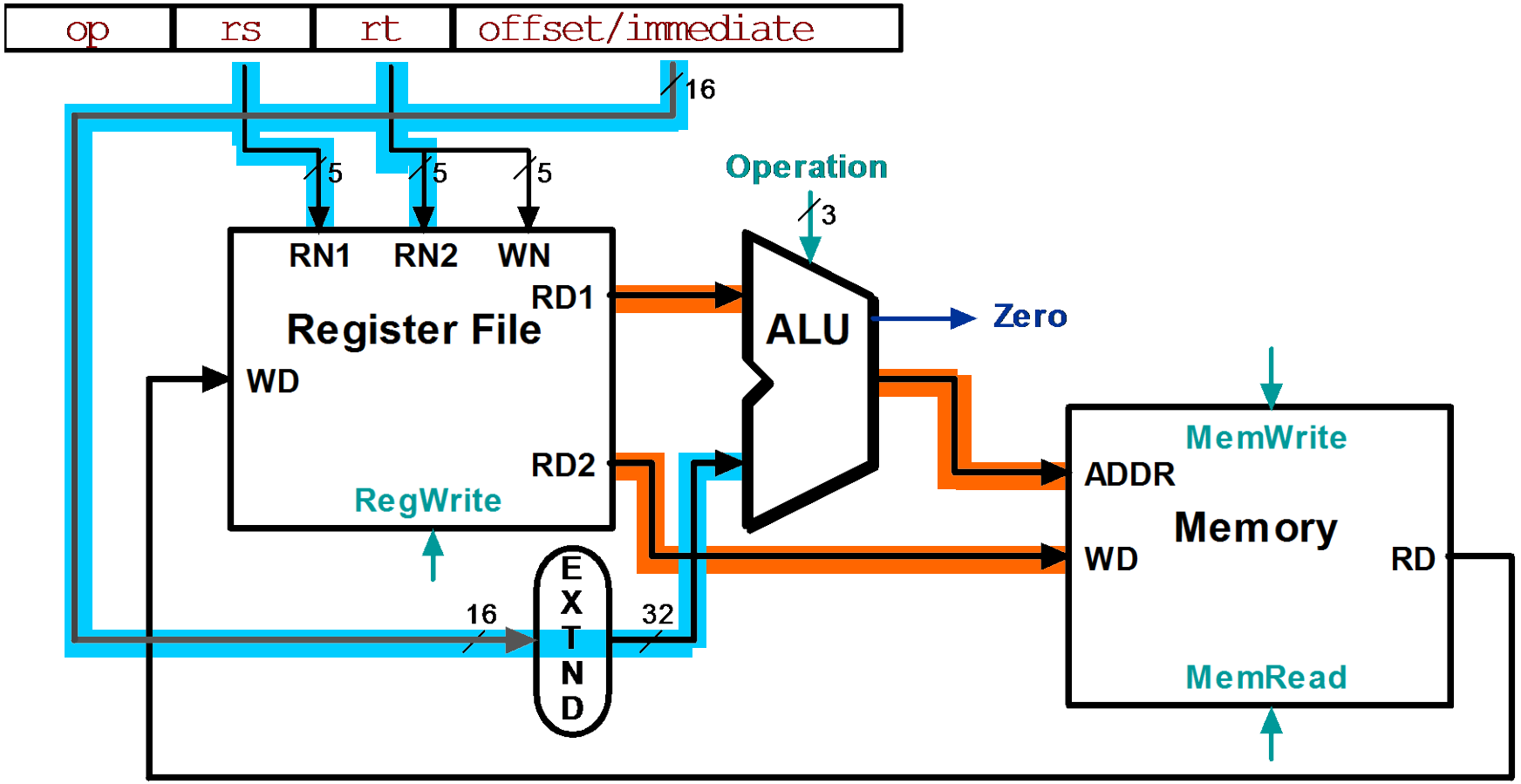


Why we need an extend component for the datapath?

Datapath for Store Instructions

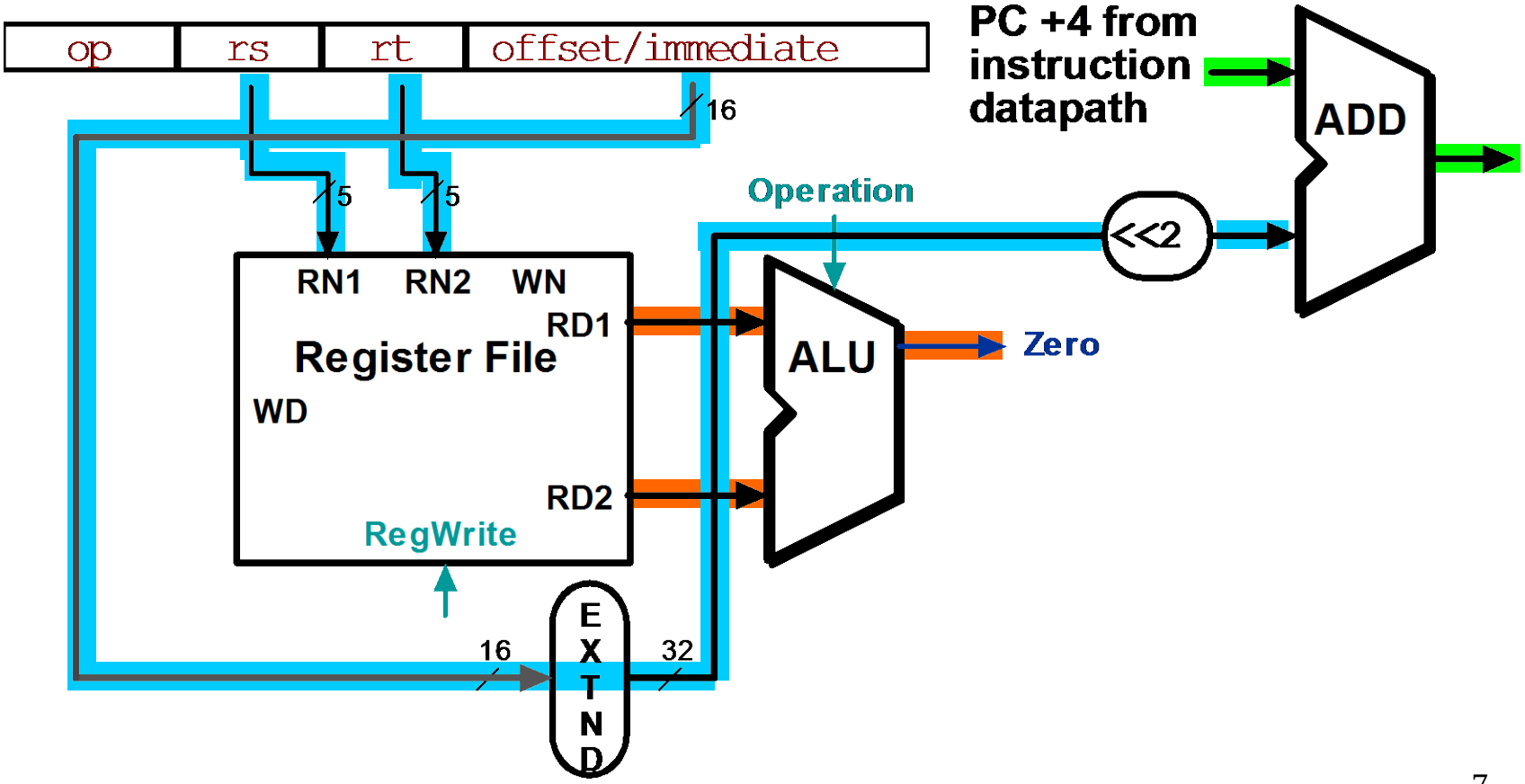
sw rt, offset(rs)

MEM[R[rs] + sign_extend(offset)] ← R[rt]

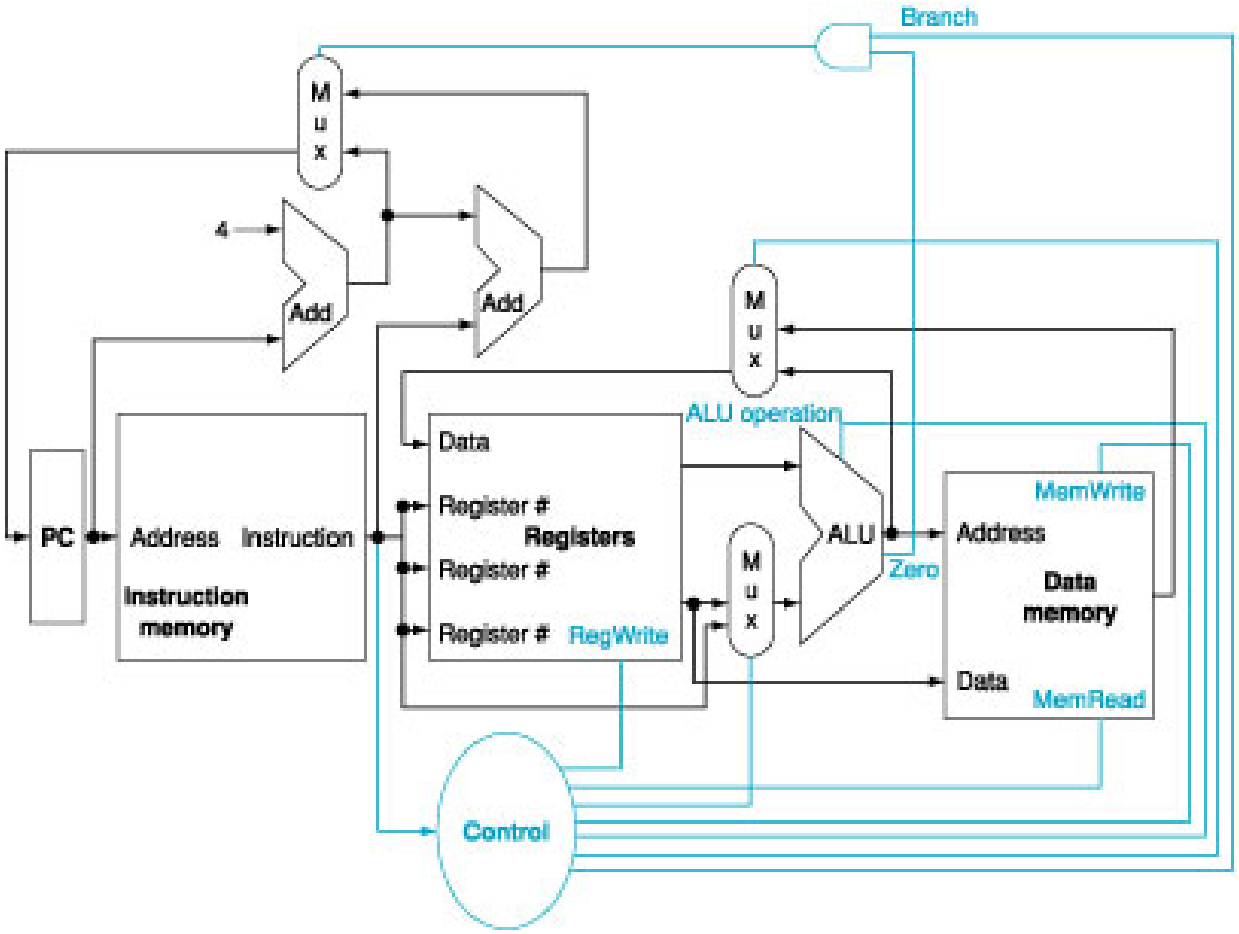


Datapath for Branch Instructions

beq rs, rt, offset
if (R[rs] == R[rt]) then
PC ← PC+4 + s_extend(offset<<2)

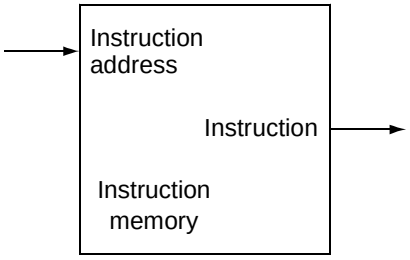


Datapath: More Detailed View

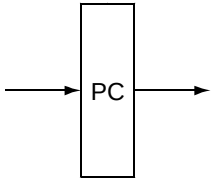


Simple Implementation

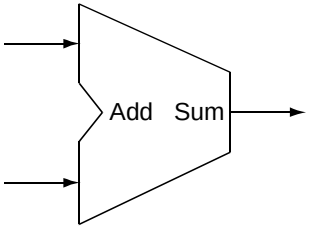
- Include the functional units we need for each instruction



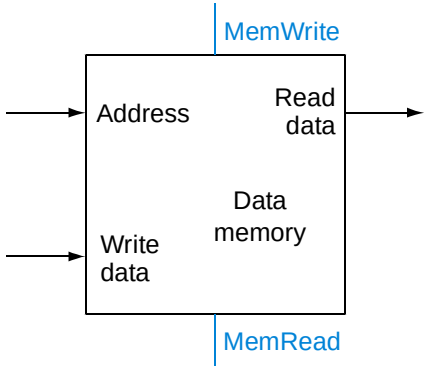
a. Instruction memory



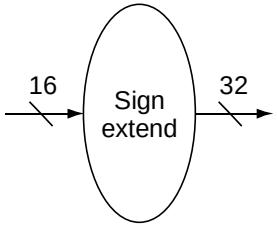
b. Program counter



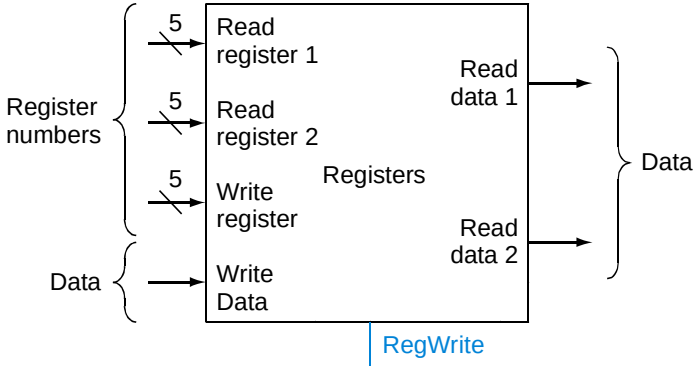
c. Adder



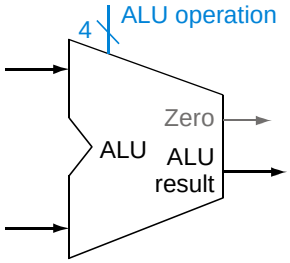
a. Data memory unit



b. Sign-extension unit



a. Registers



b. ALU

ALU control input	Function
000	AND
001	OR
010	add
110	subtract
111	set on less than

RTL (Register Transfer Level) Code for MIPS add

1. Fetch

   Instruction = ROM[PC], PC=PC+4

2. Read Operands

   ALUOp1 = Registers[rs-value],

   ALUOp2 = Registers[rt-value]

3. Add

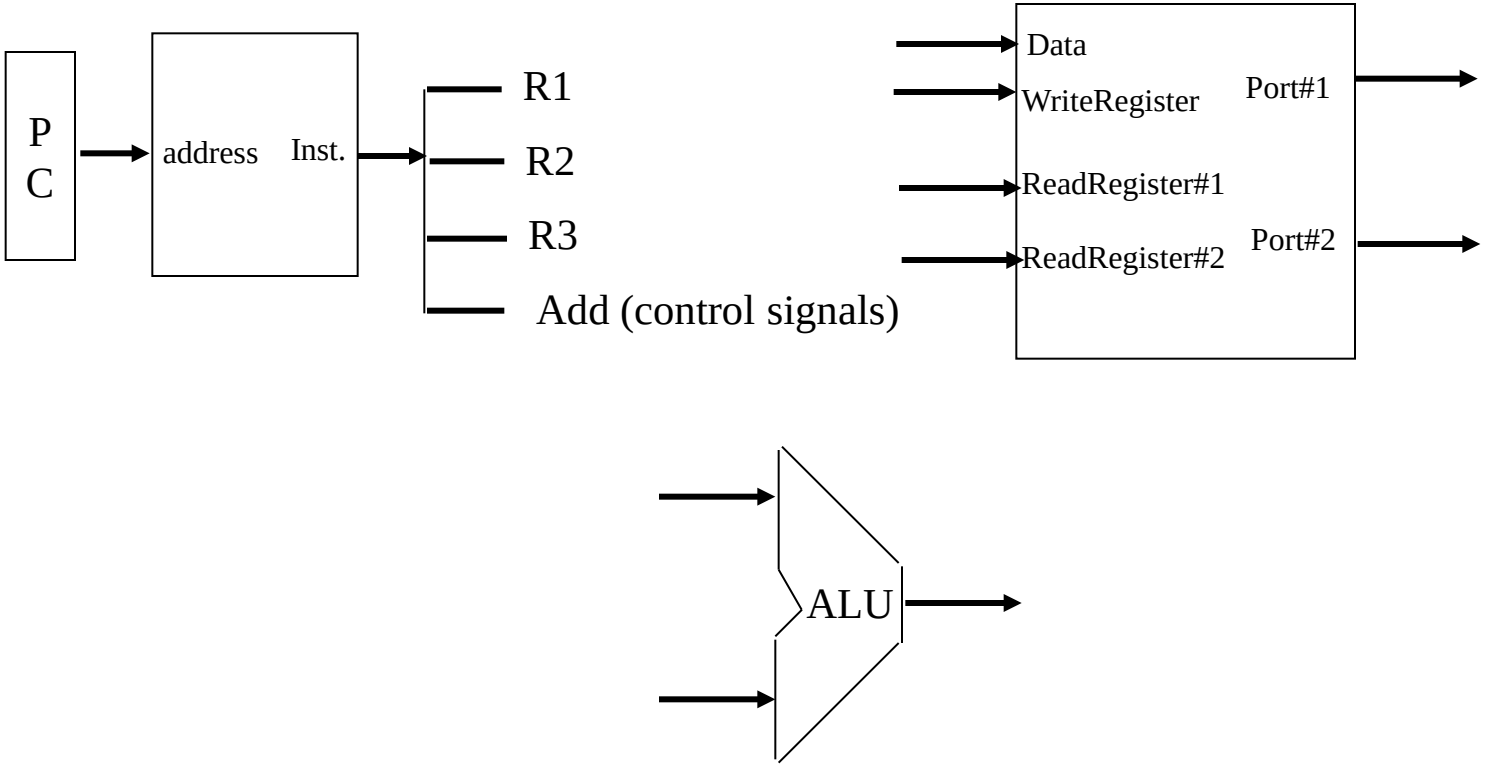
   ALUOut = ALUOp1+ALUOp2

4. Write Result

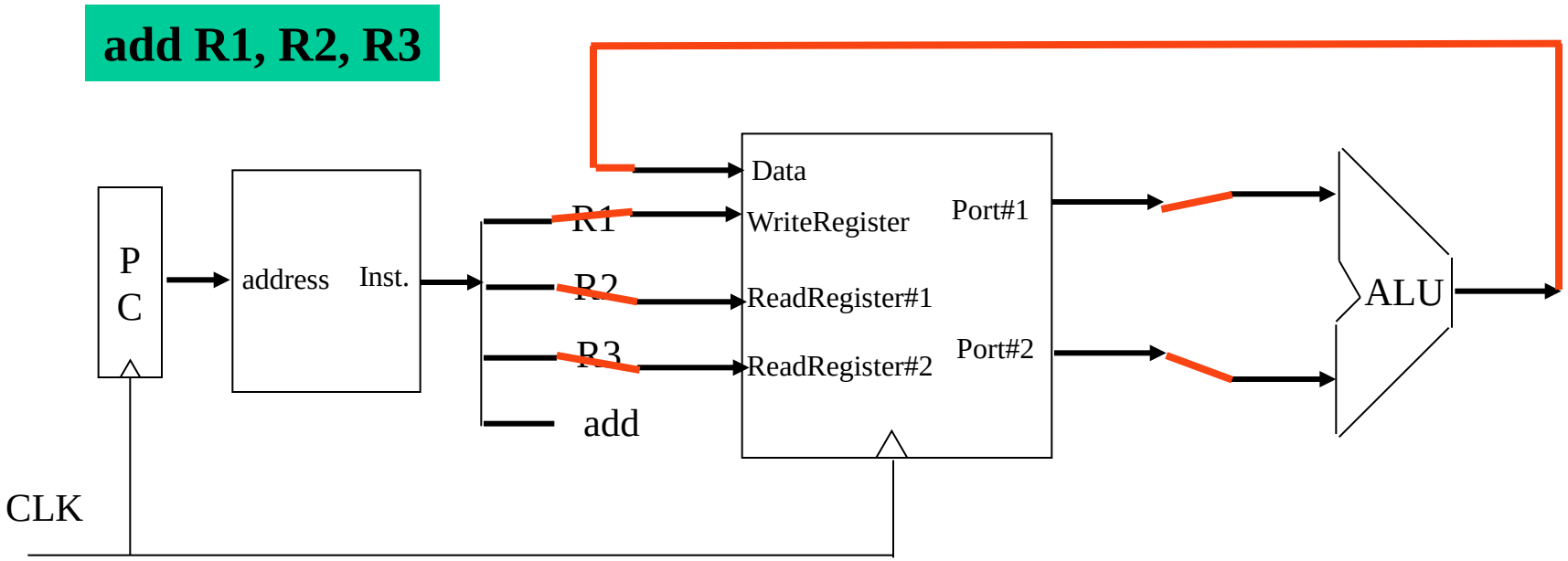
   Registers[rd-value] = ALUOut

Datapath Components for MIPS add

add R1, R2, R3

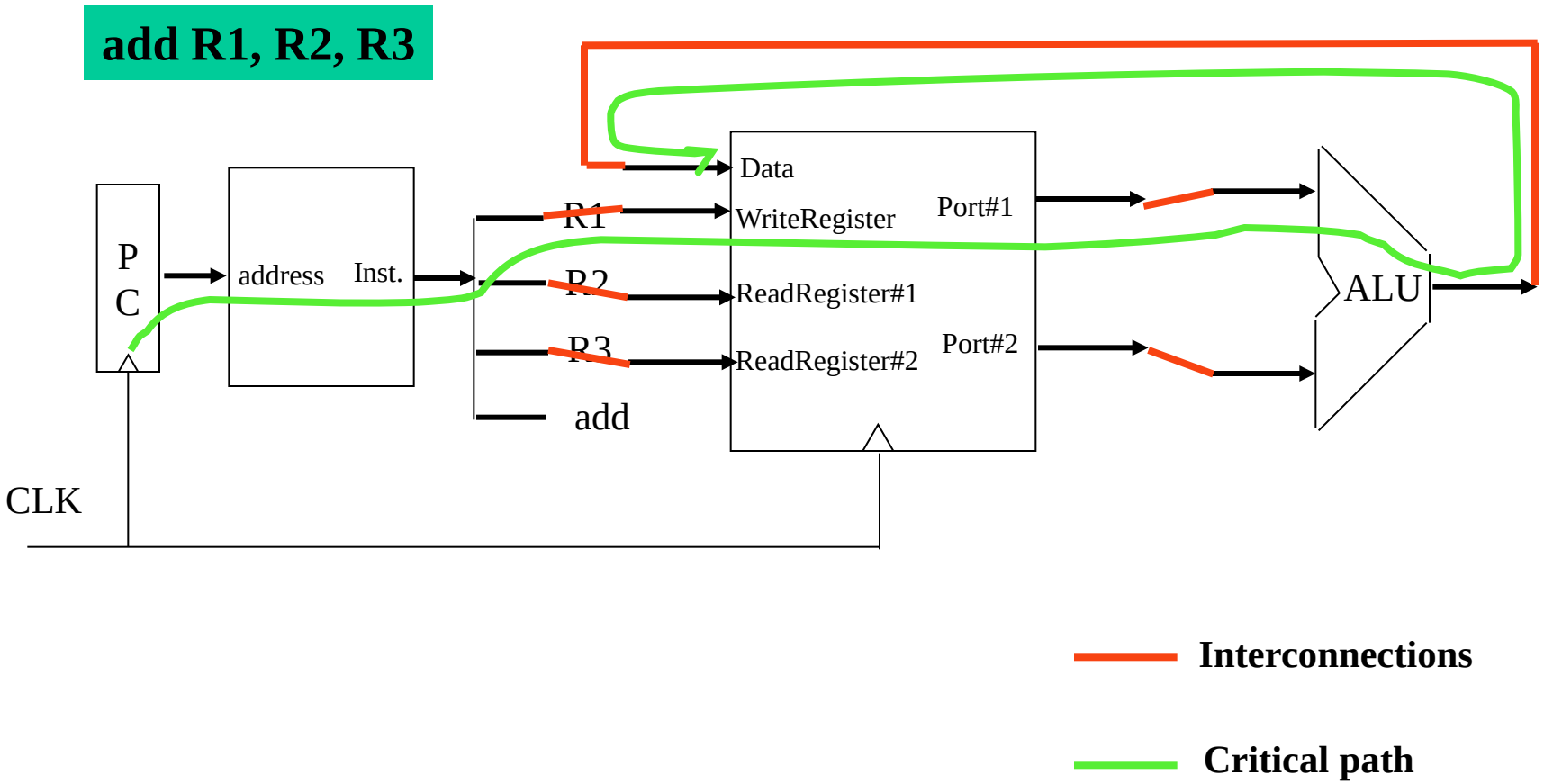


Datapath Connections for MIPS add



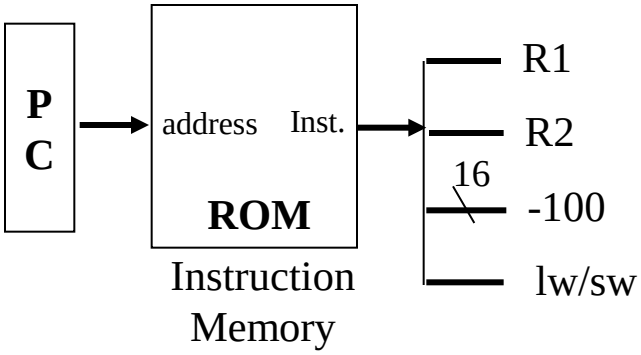
What is missing?

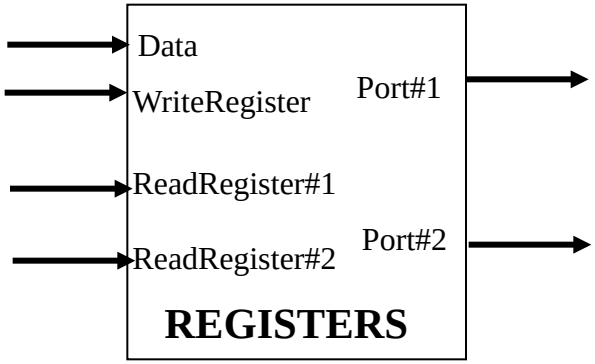
Critical Path for MIPS add

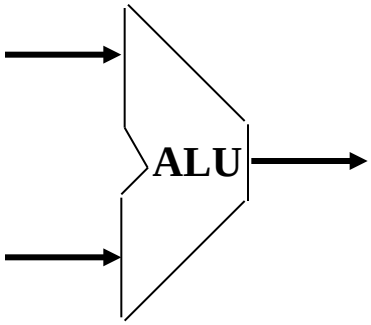


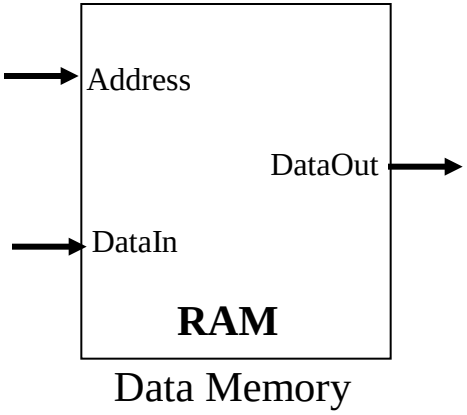
Datapath Components for MIPS lw/sw

lw R1, -100(R2)
sw R1, -100(R2)

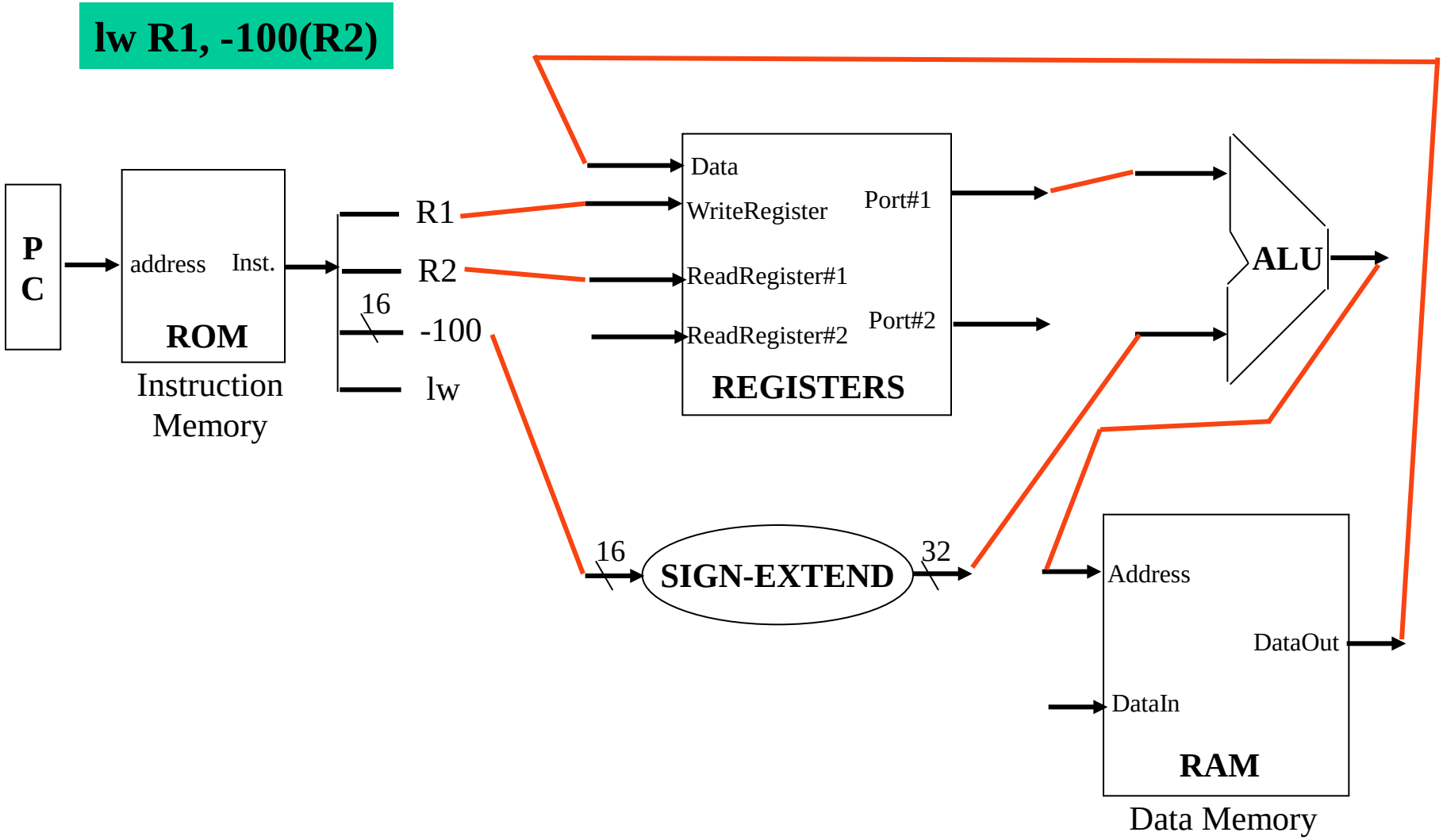




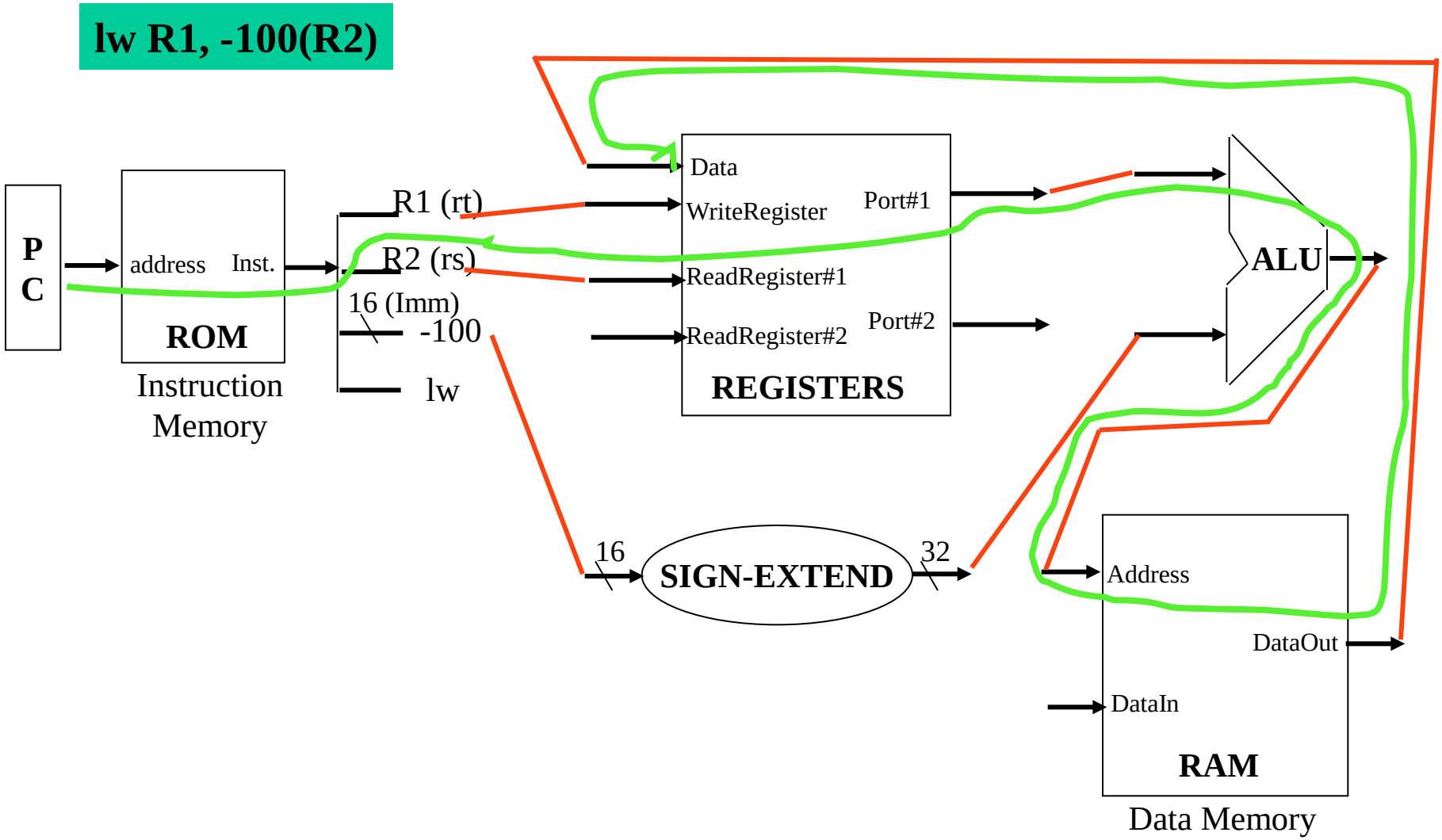




Connections for lw

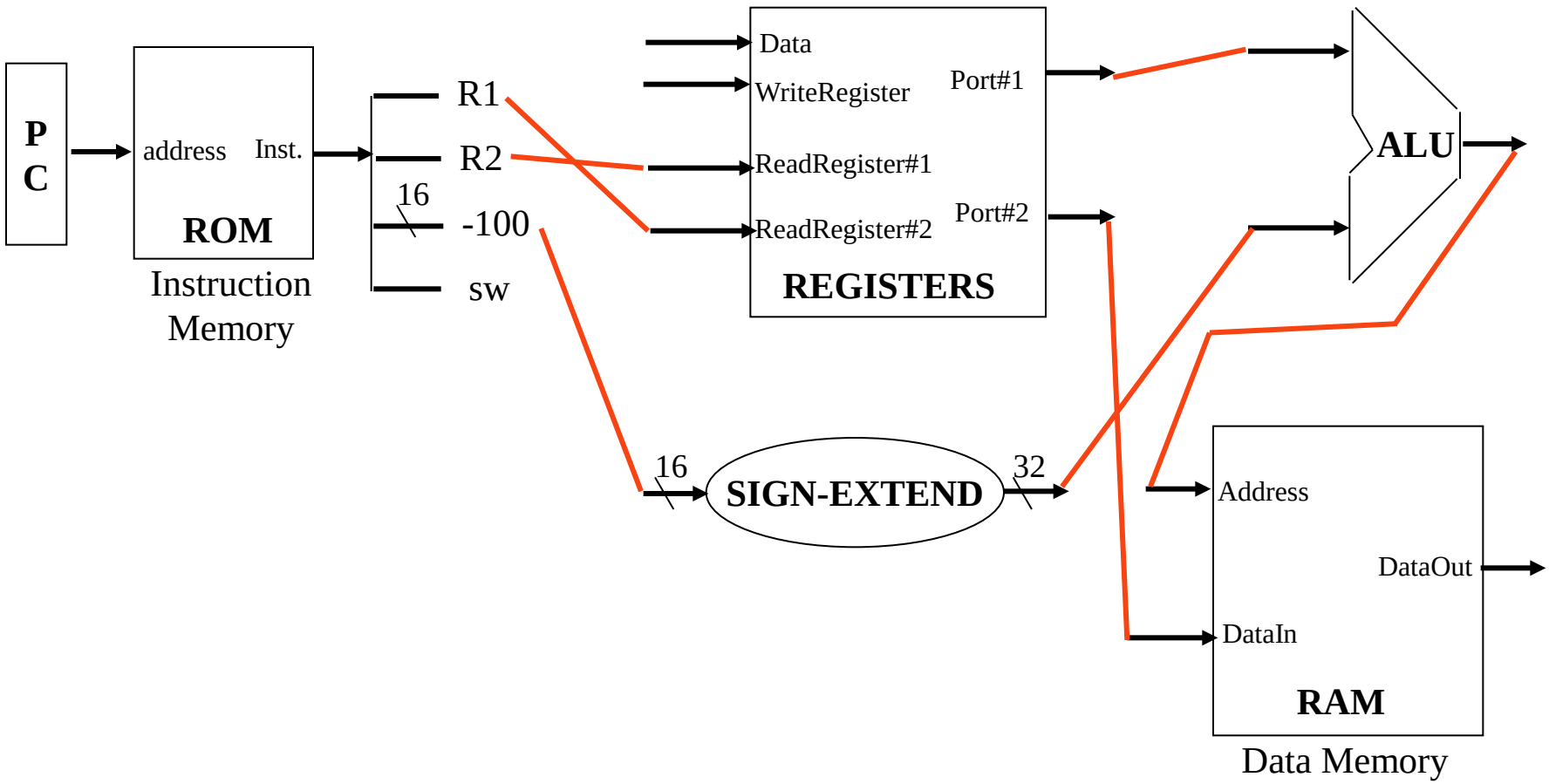


Critical Path for lw



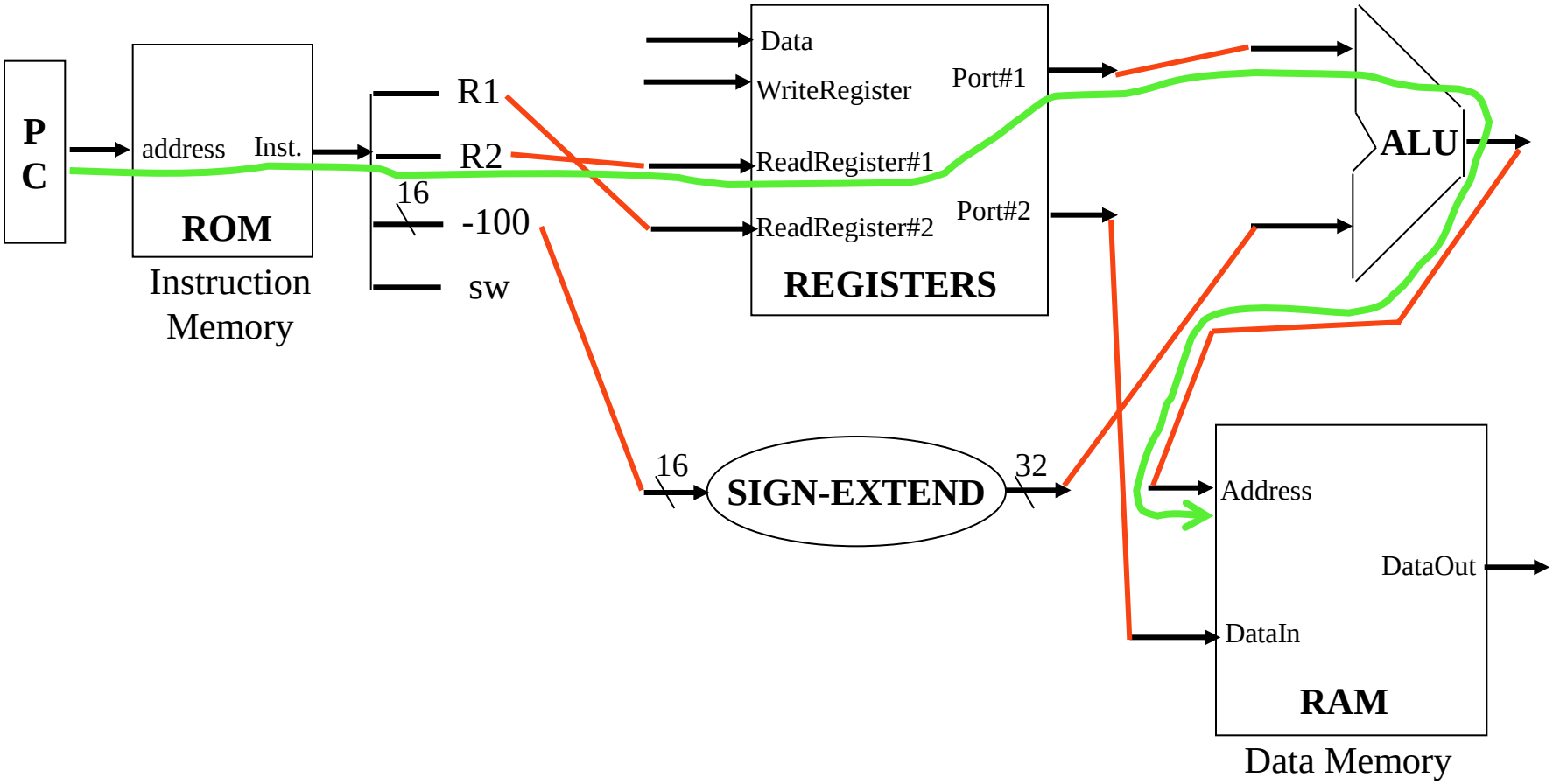
Connections for sw

sw R1, -100(R2)

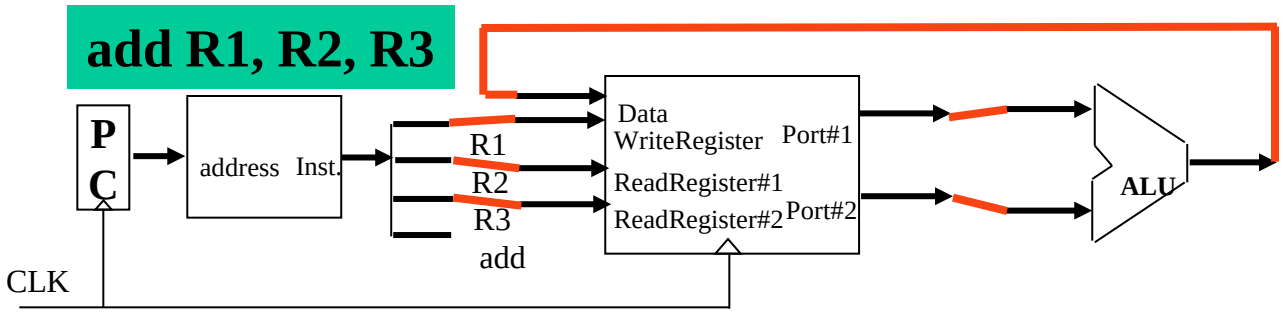


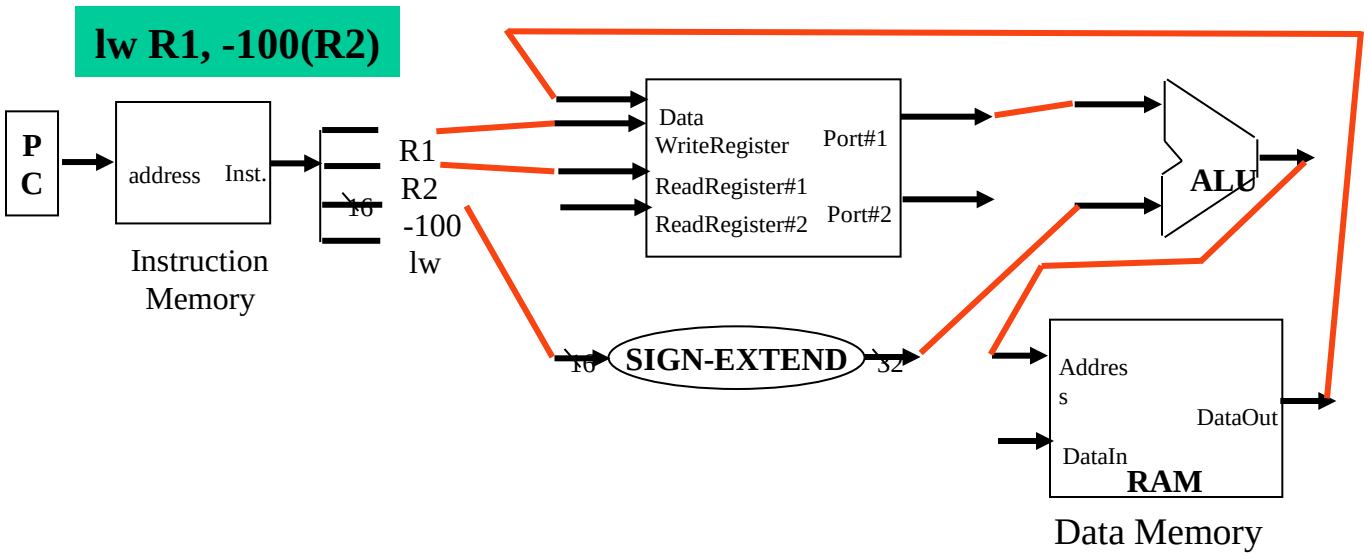
Critical Path for sw

sw R1, -100(R2)

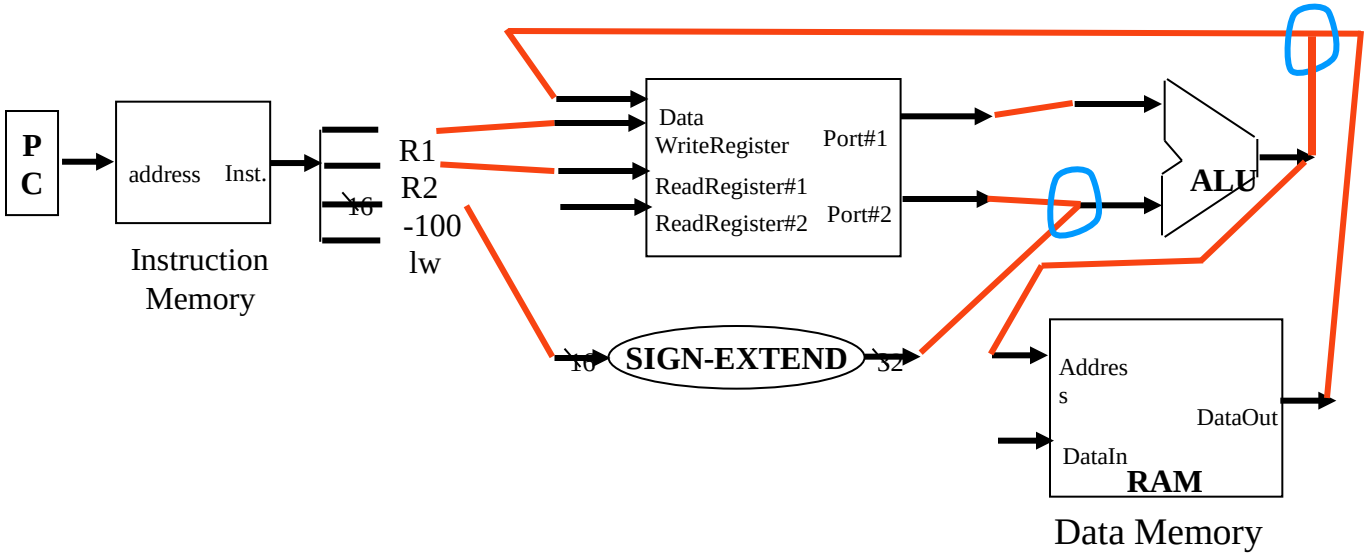


Datapath Connections for MIPS add and lw

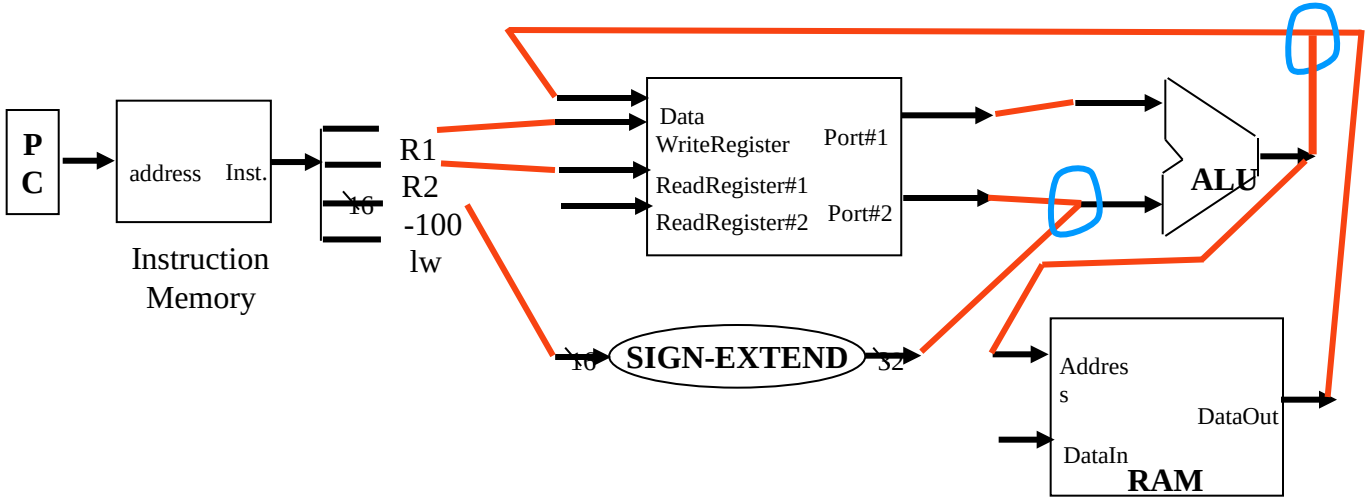




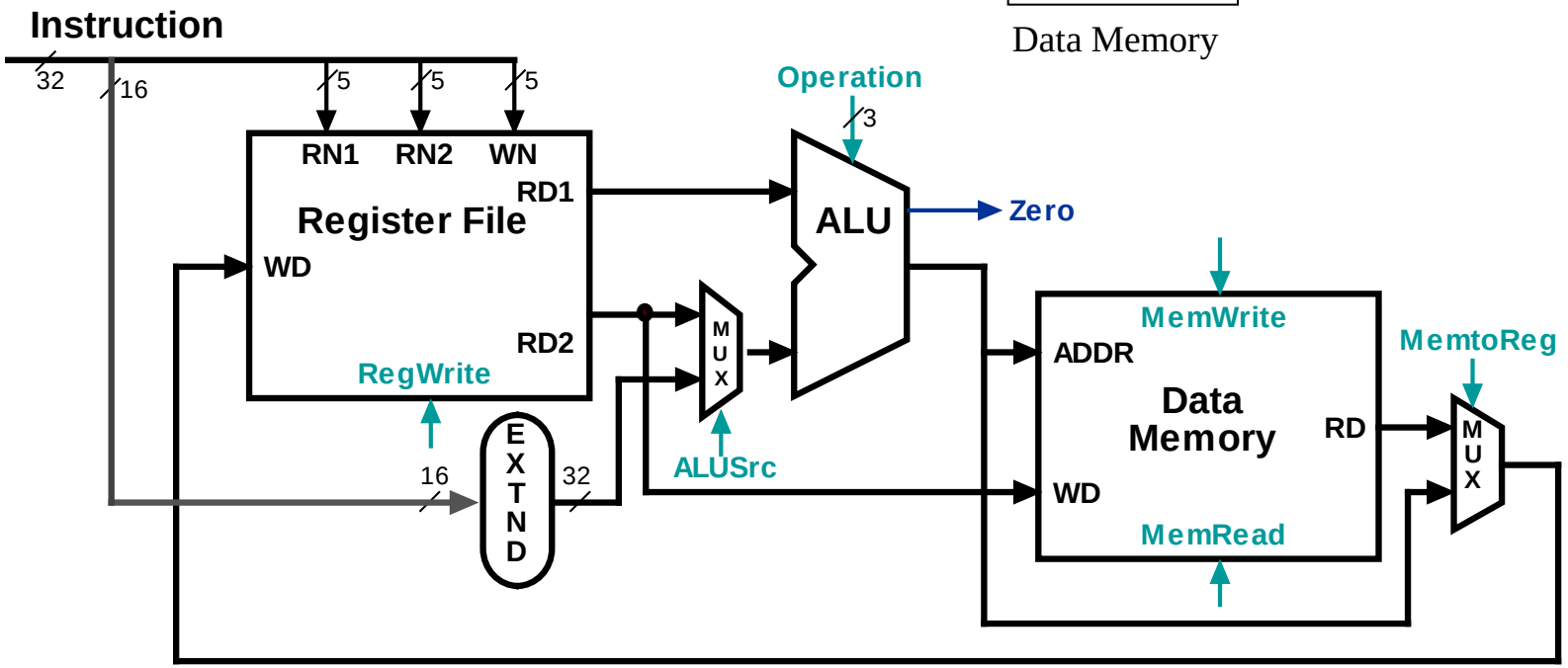
Datapath Connections for MIPS add and lw



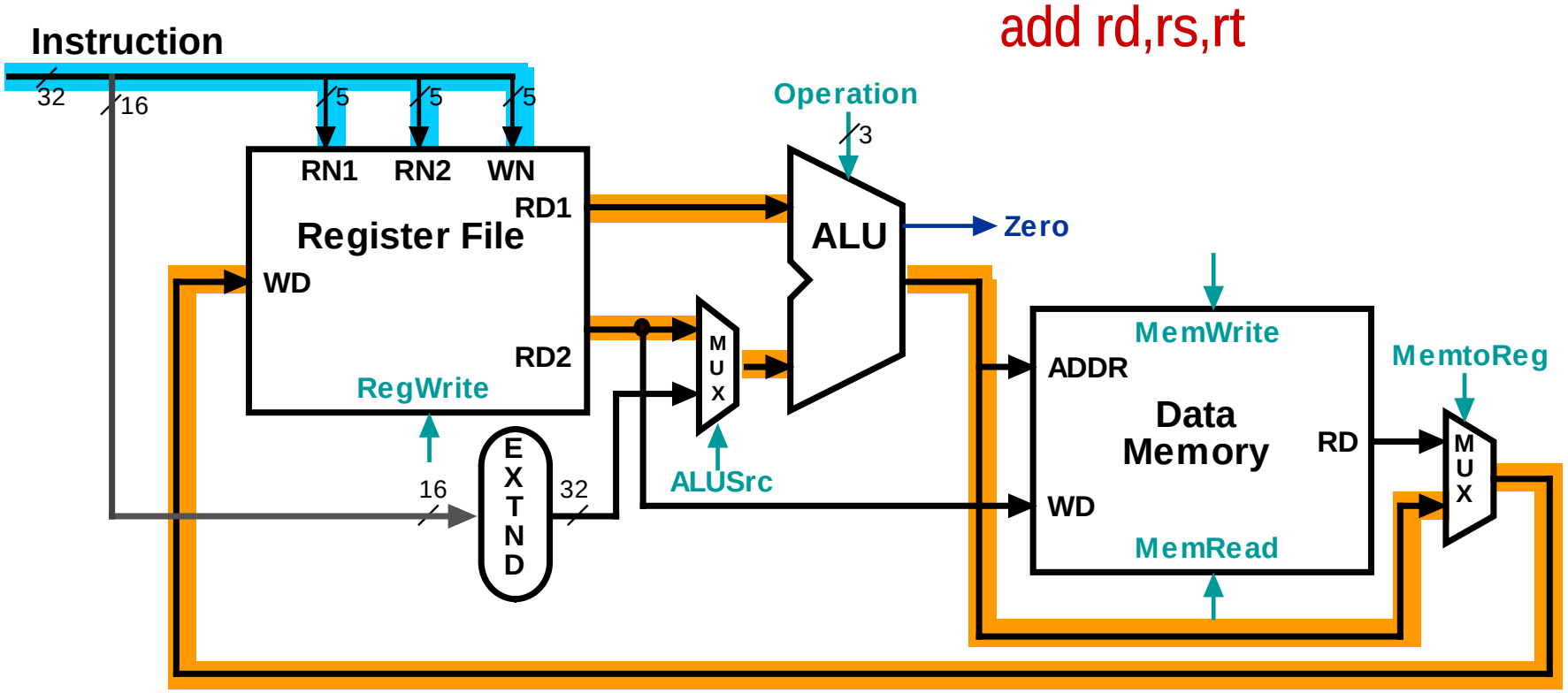
Combined Datapath: R-Type and Load/Store Instructions



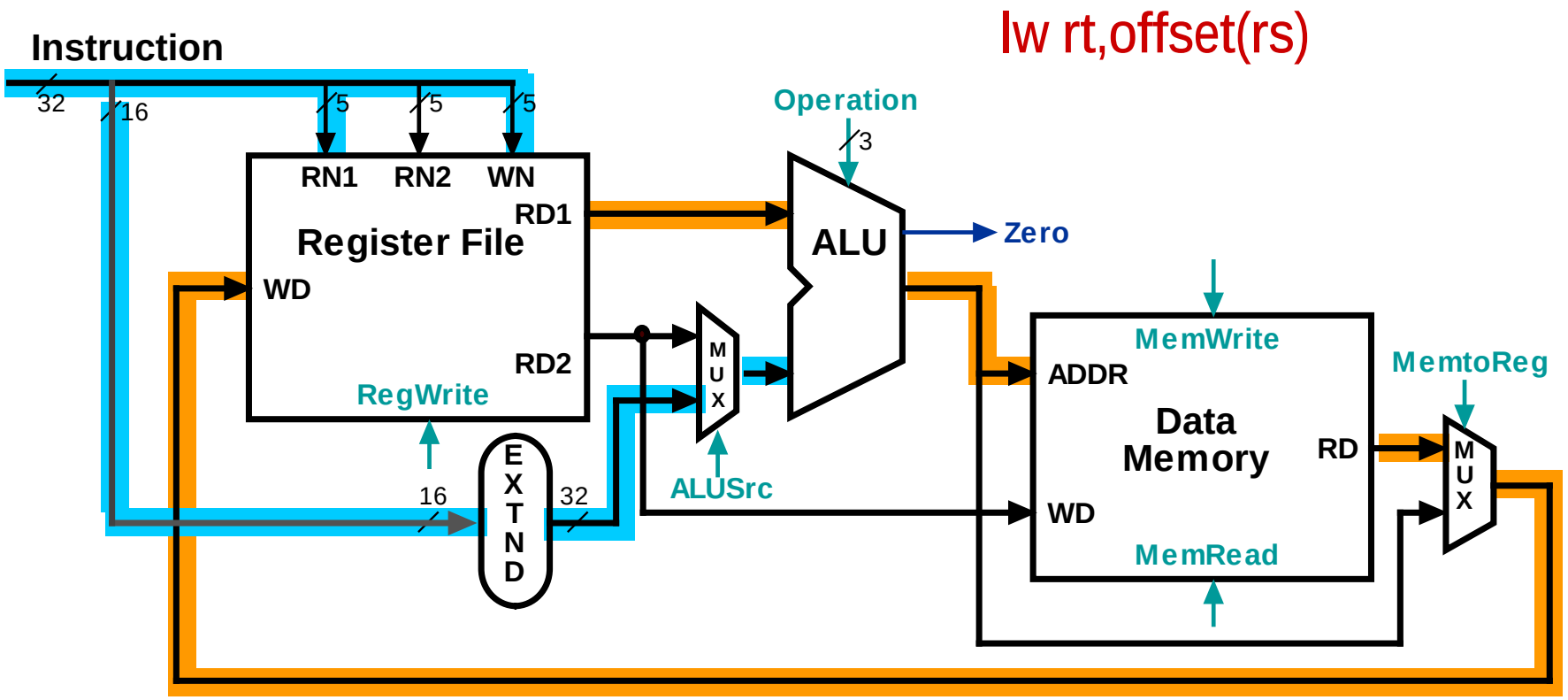
NEED MUX



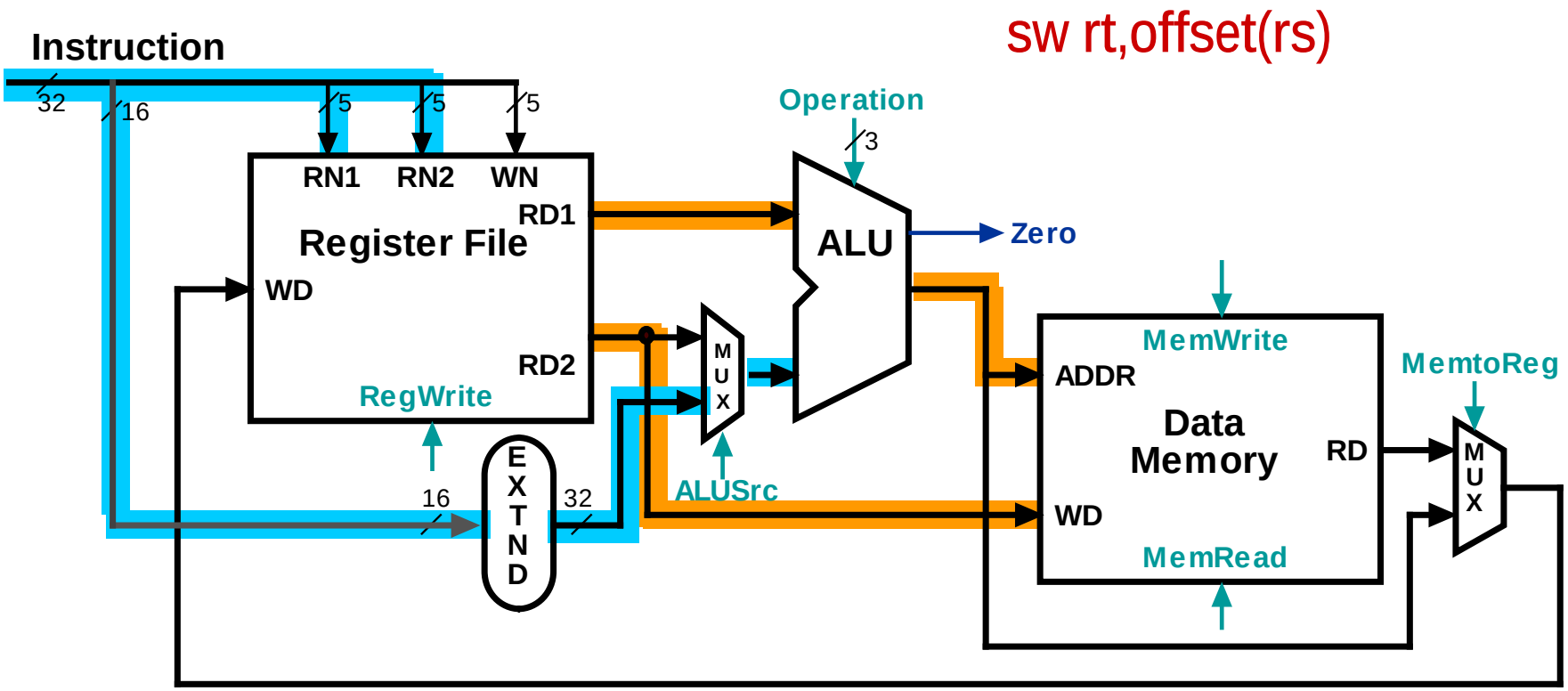
Combined Datapath: Executing a R-Type Instruction



Combined Datapath: Executing a load instruction

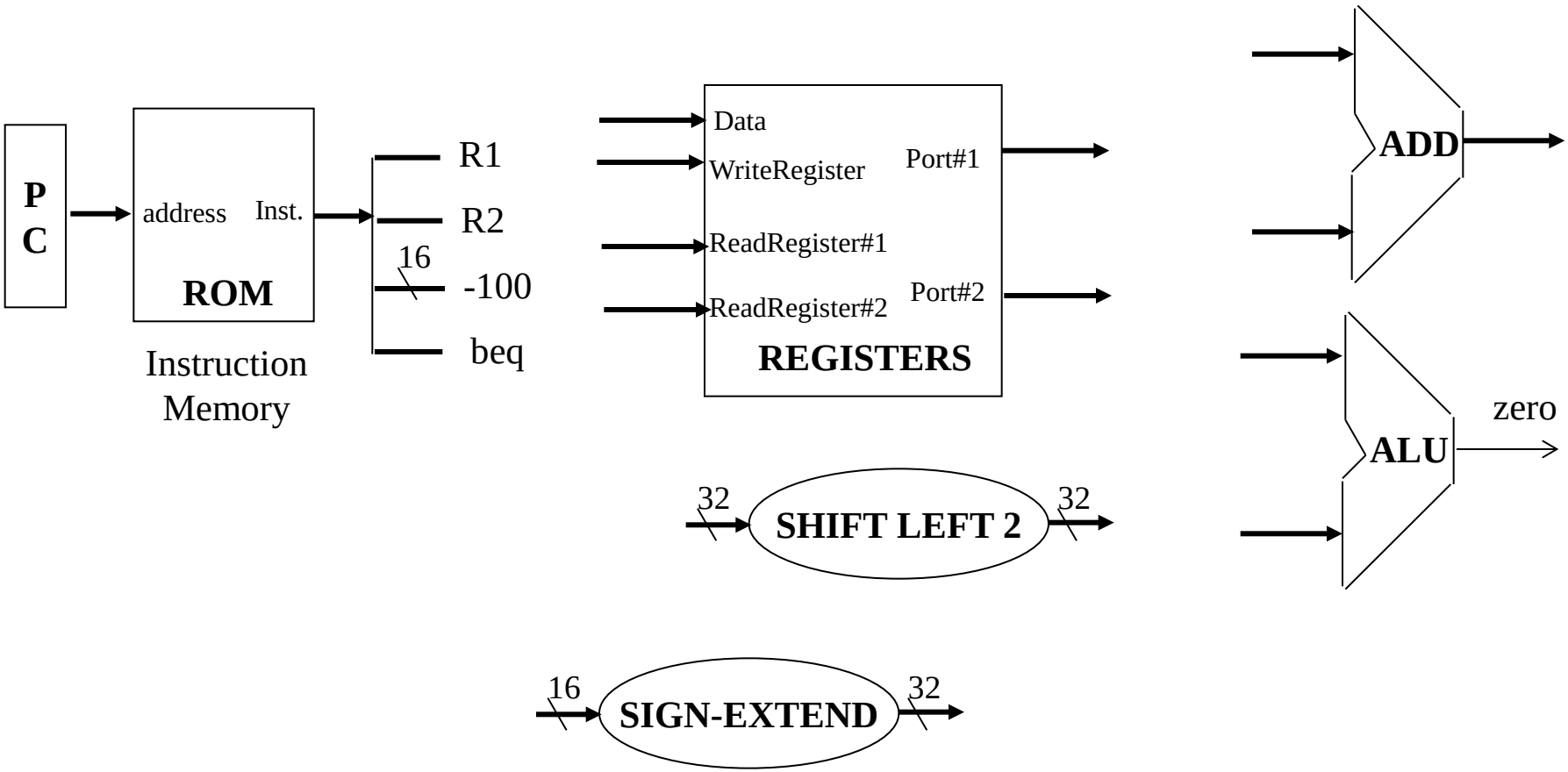


Combined Datapath: Executing a store instruction



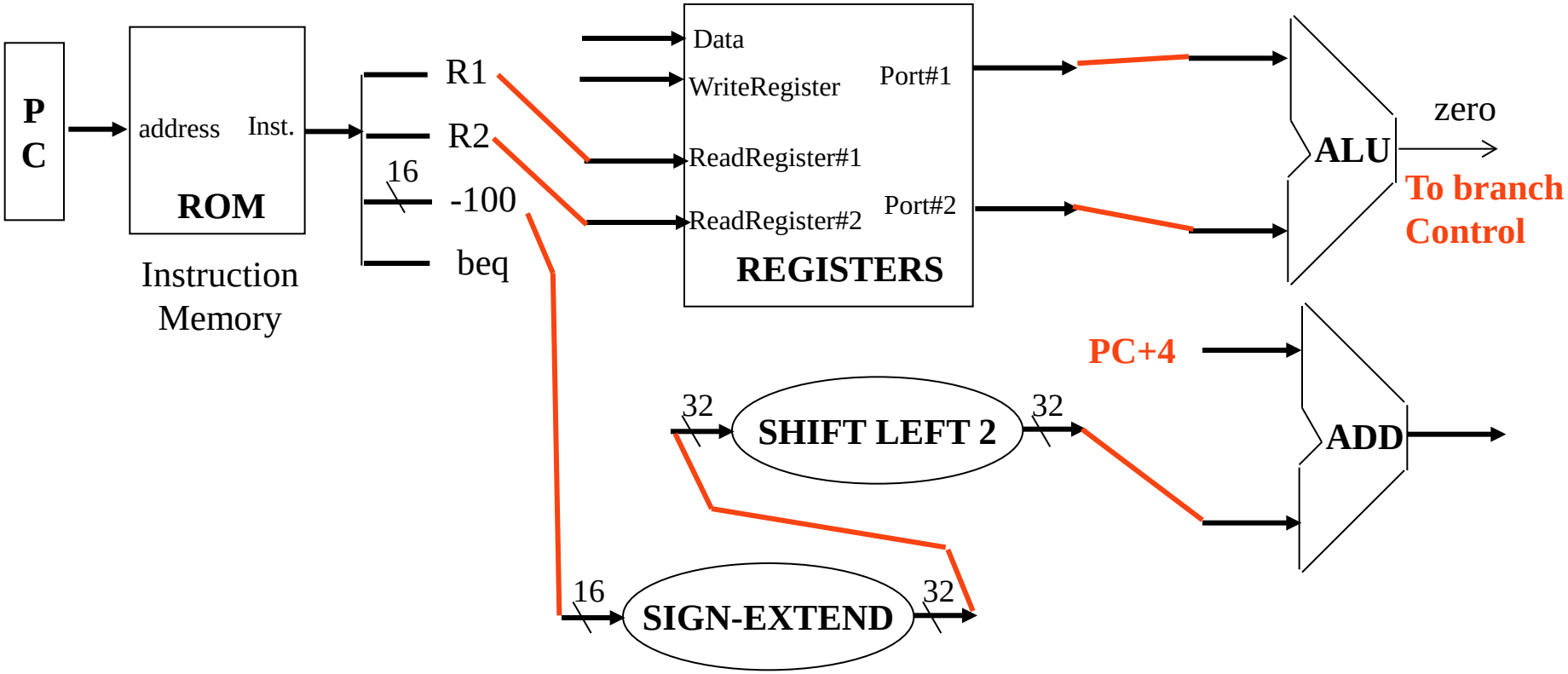
Datapath Components for MIPS beq

beq $R1, $R2, -100
if $R1==$R2 then PC = PC+4+4*(-100) else PC = PC+4

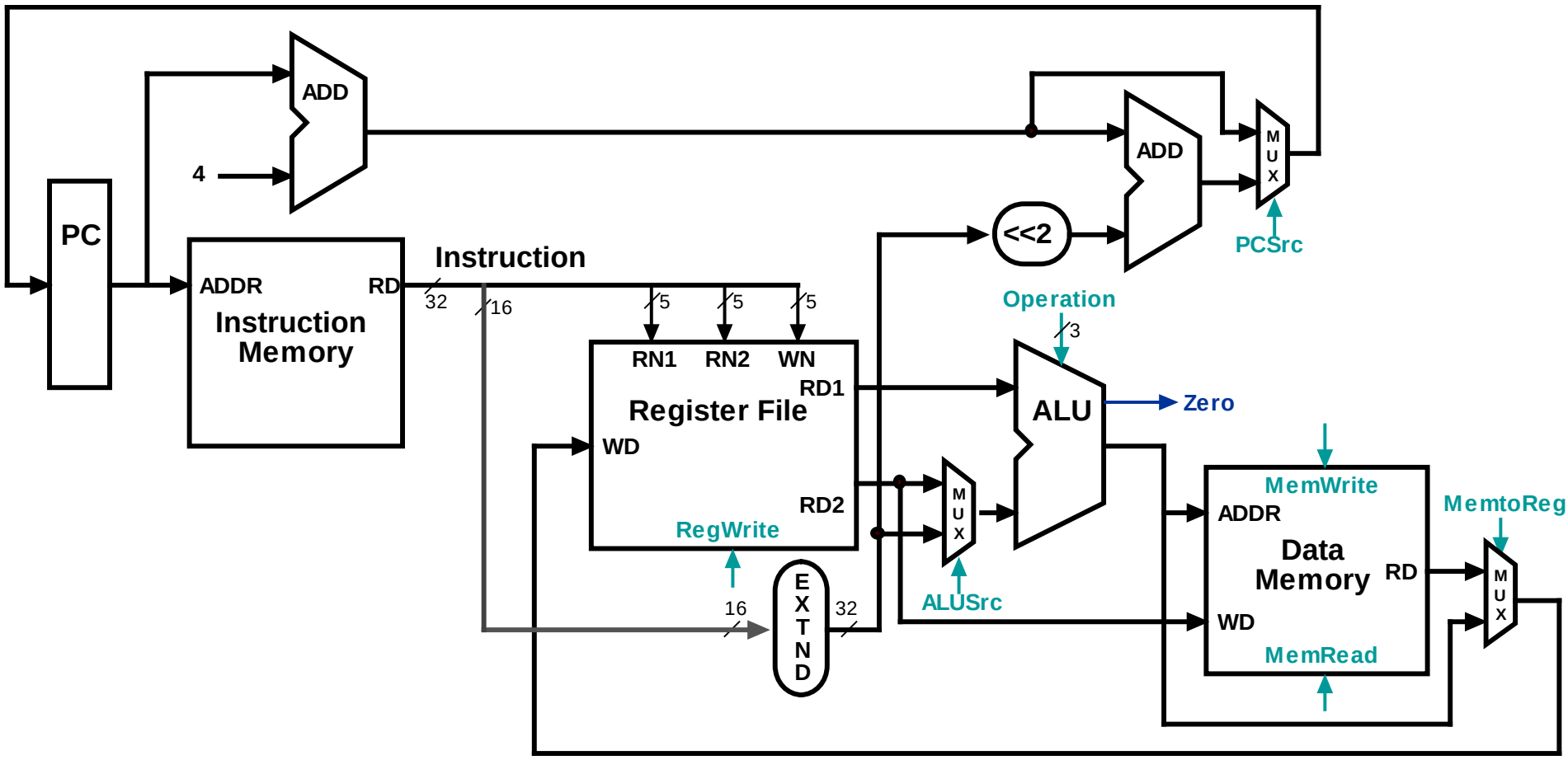


Datapath Connections for MIPS beq

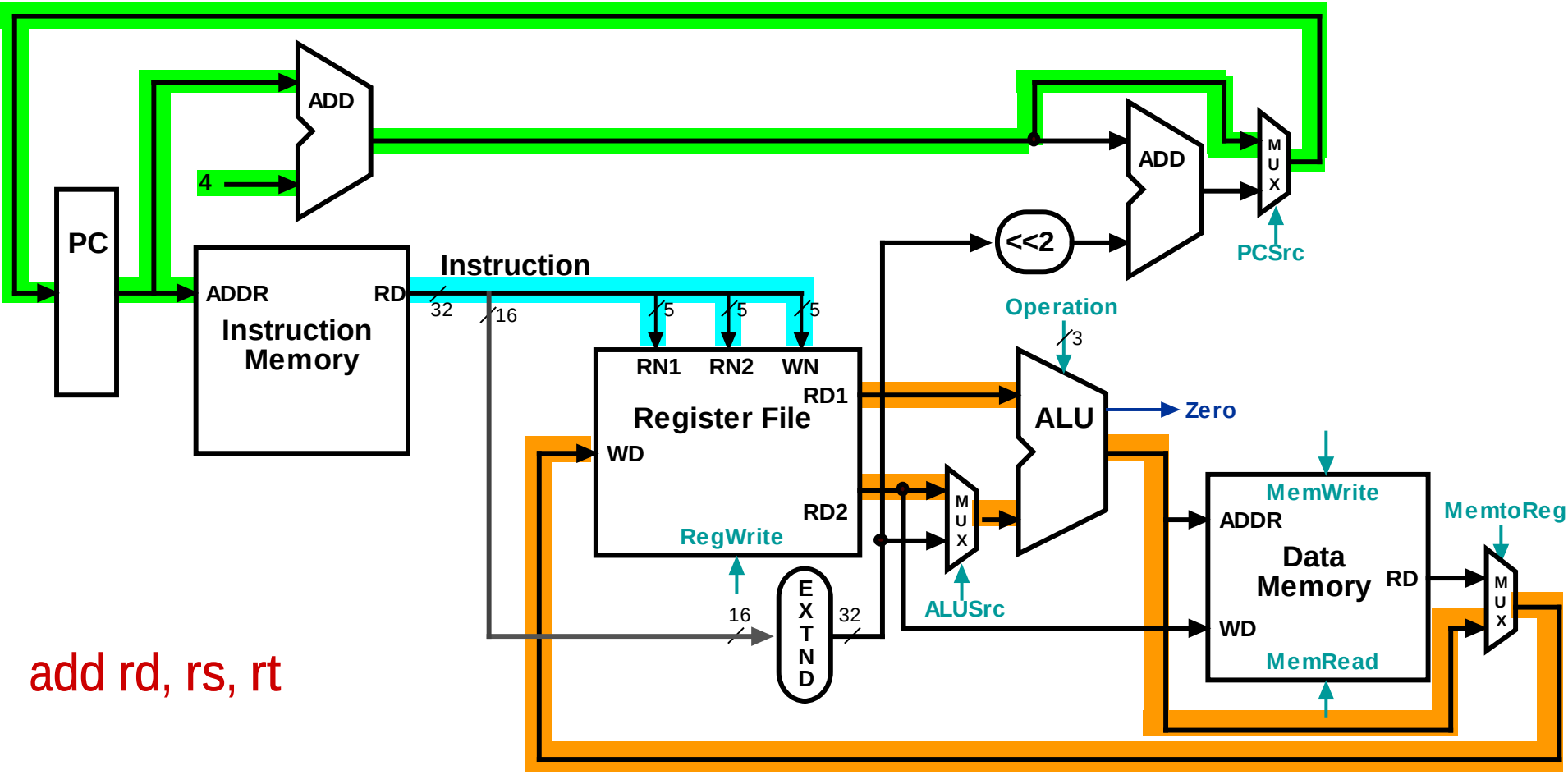
beq $R1, $R2, -100
if $R1==$R2 then PC = PC+4+4*(-100) else PC = PC+4



Complete Single-Cycle Datapath

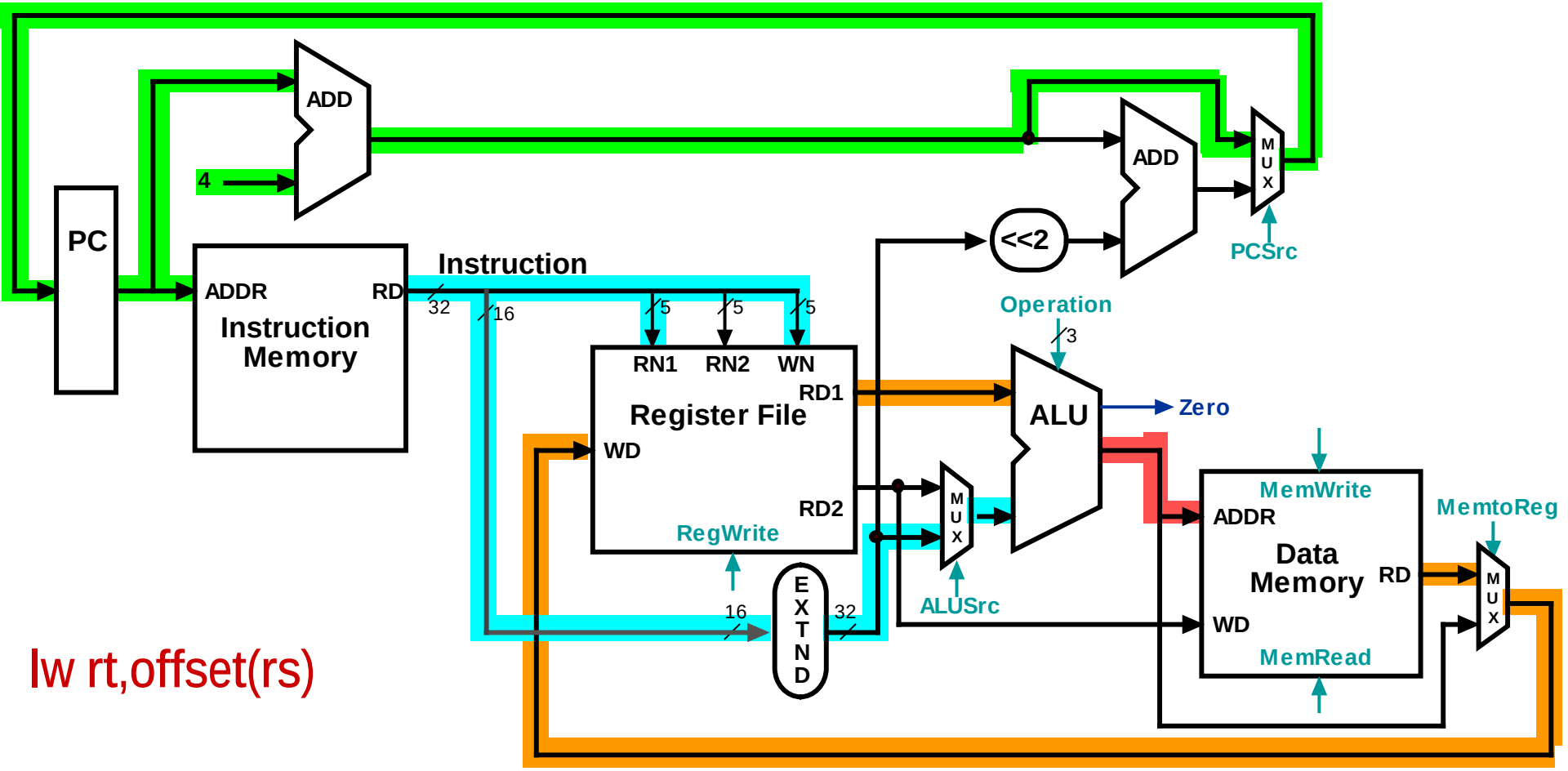


Complete Datapath Executing add



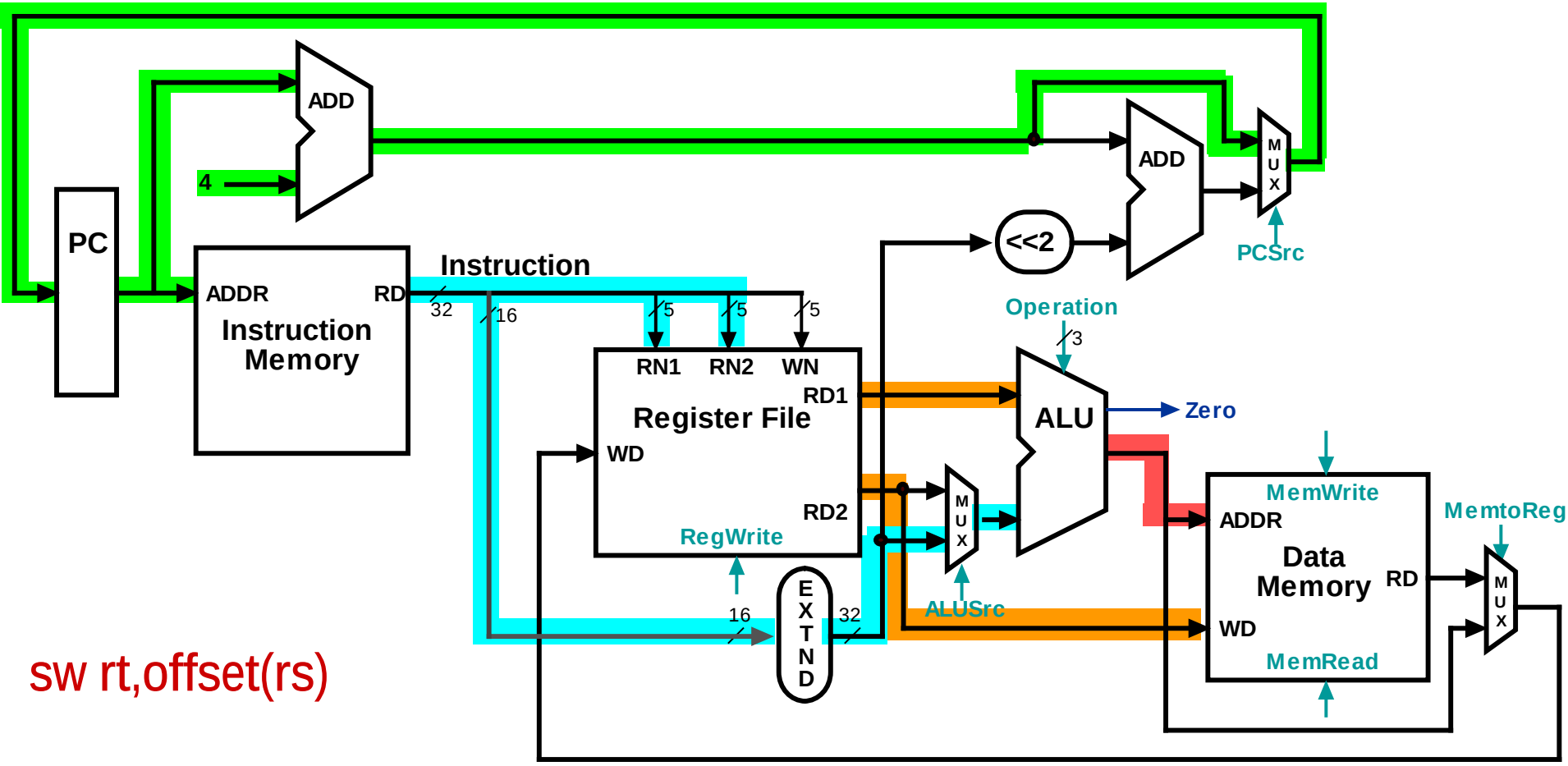
add rd, rs, rt

Complete Datapath Executing load



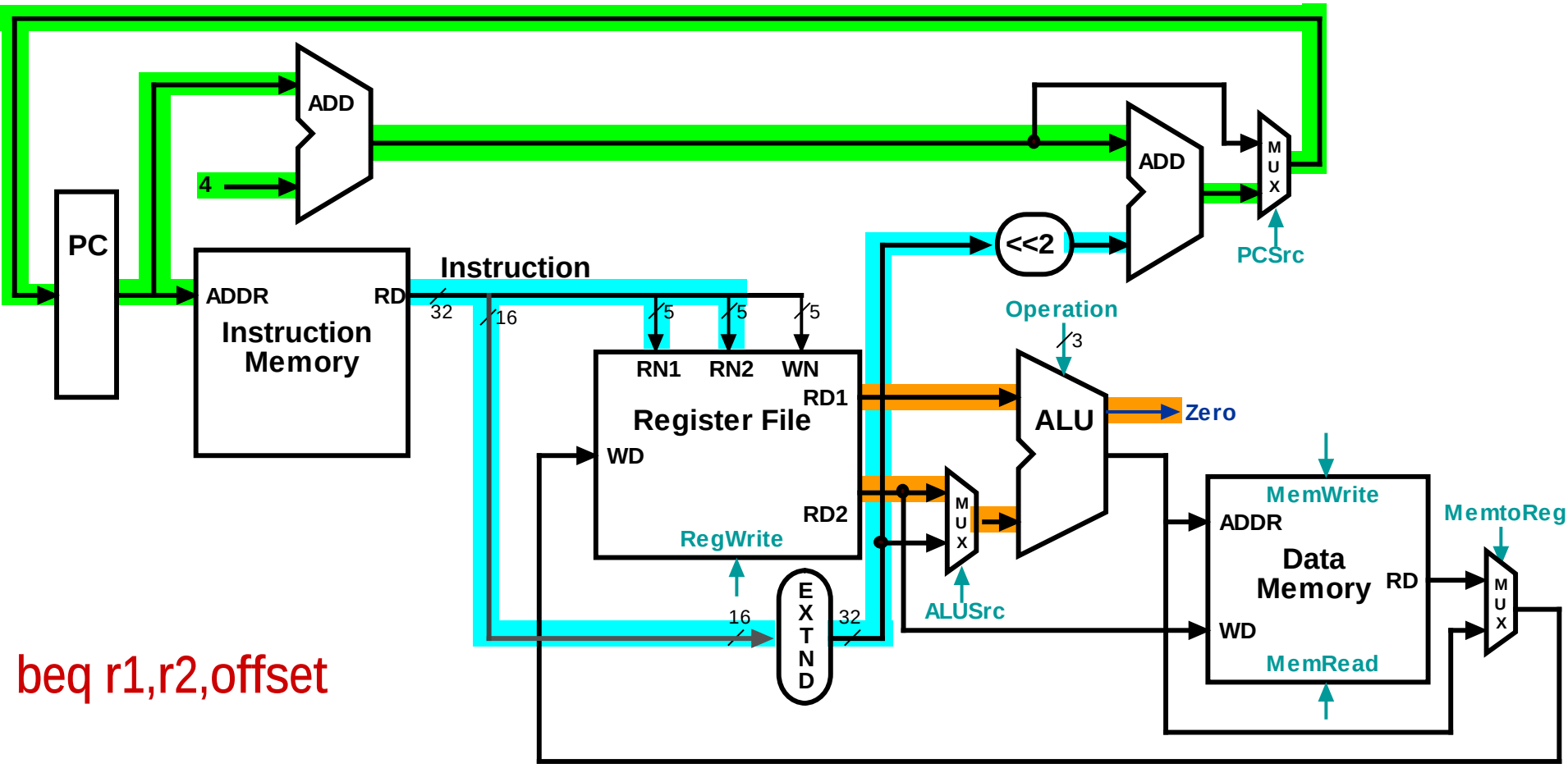
lw rt,offset(rs)

Complete Datapath Executing store



sw rt,offset(rs)

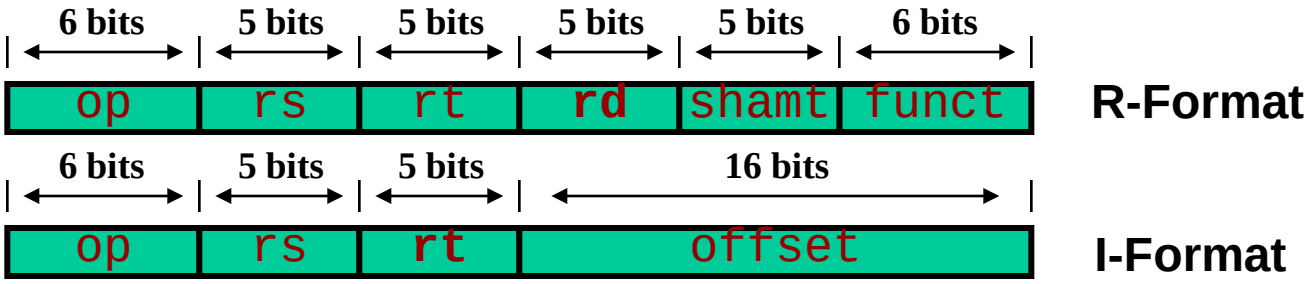
Complete Datapath Executing branch



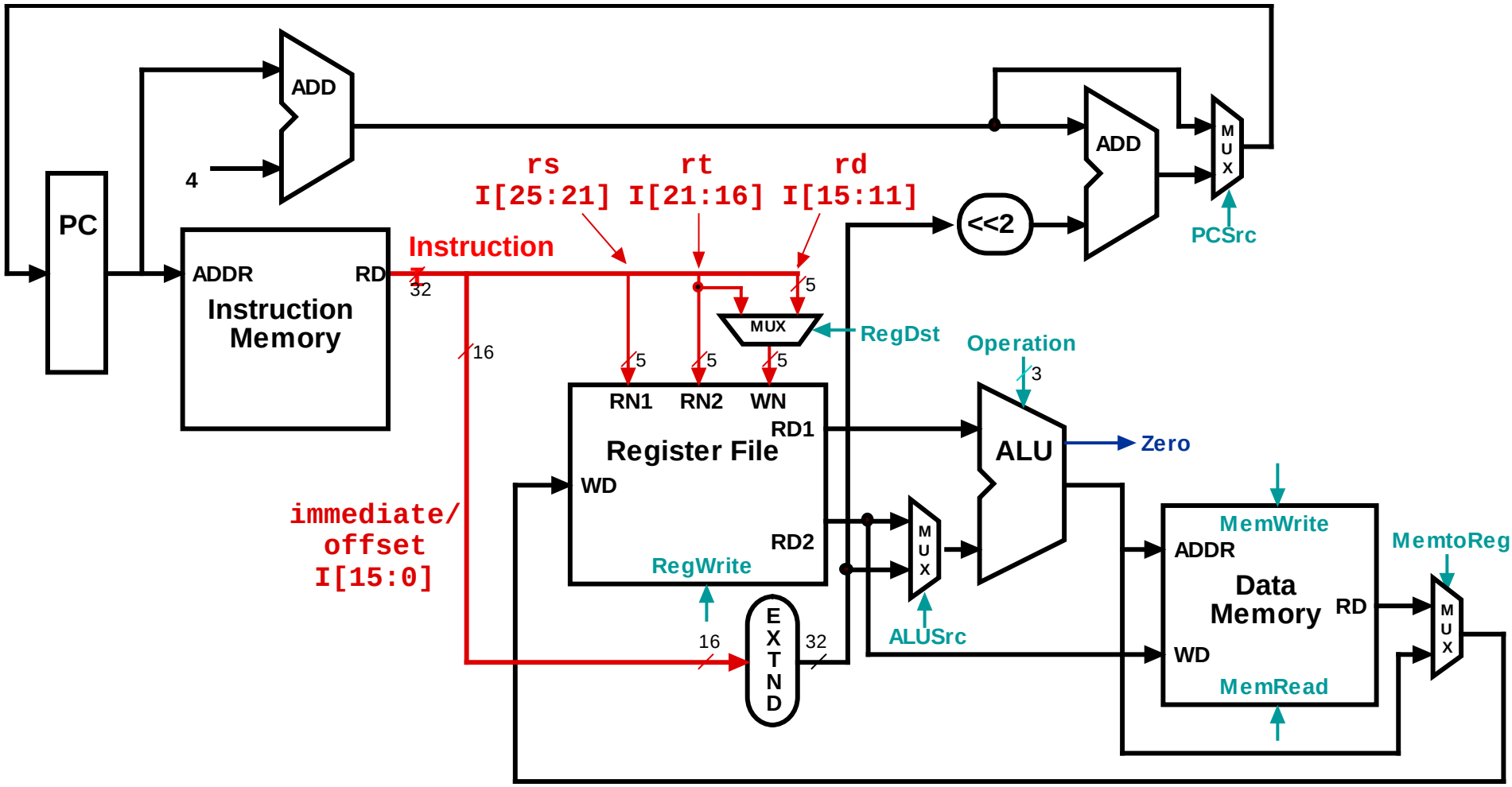
beq r1,r2,offset

Refining the Complete Datapath

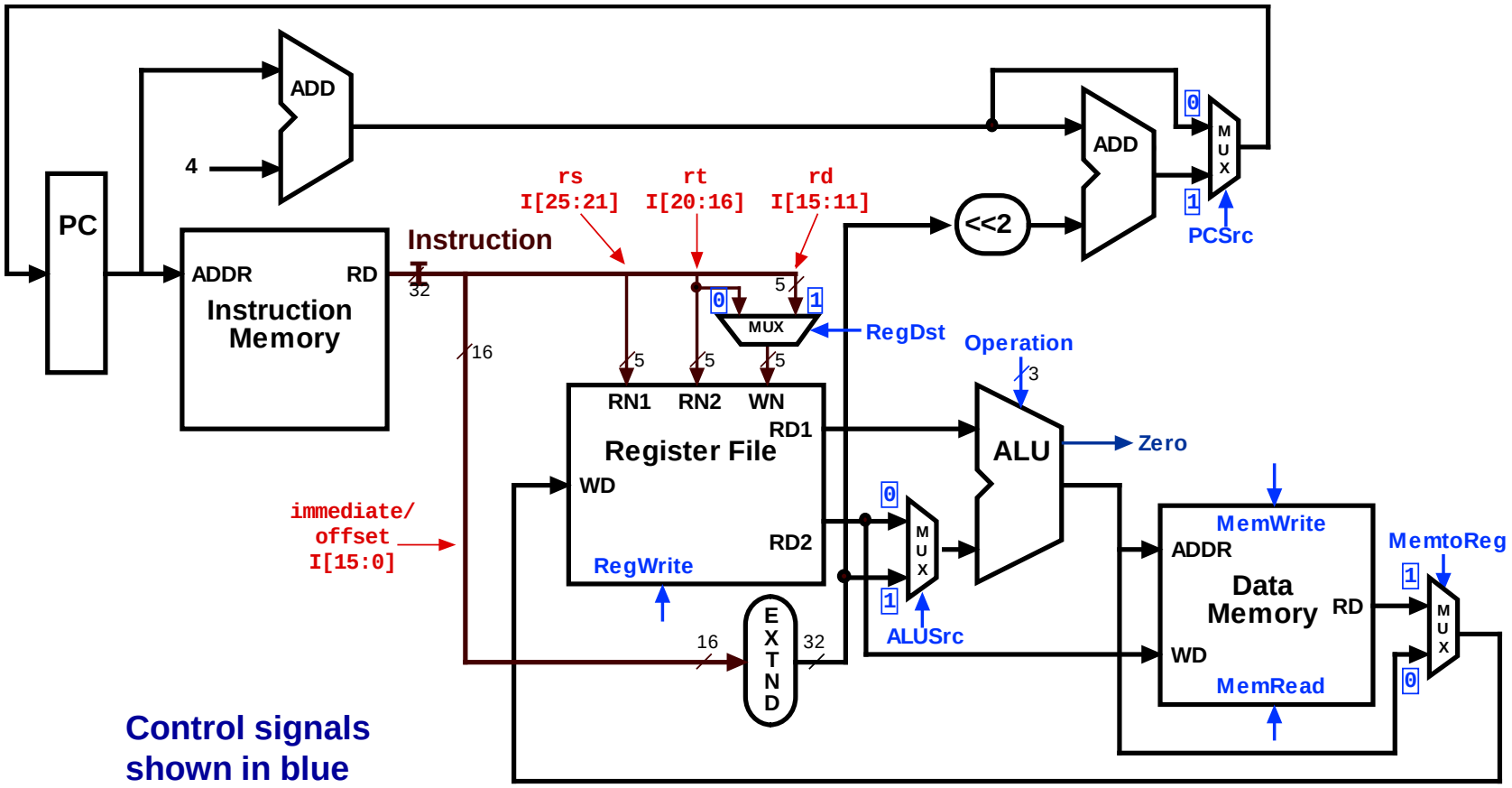
- Depending on the instruction, register file input WN is fed by different fields of the instruction
  - R-Type Instructions: rd field (bits 15:11)
  - Load Instructin: rt field (bits 21:16)
- Result: need an additional multiplexer on WN input



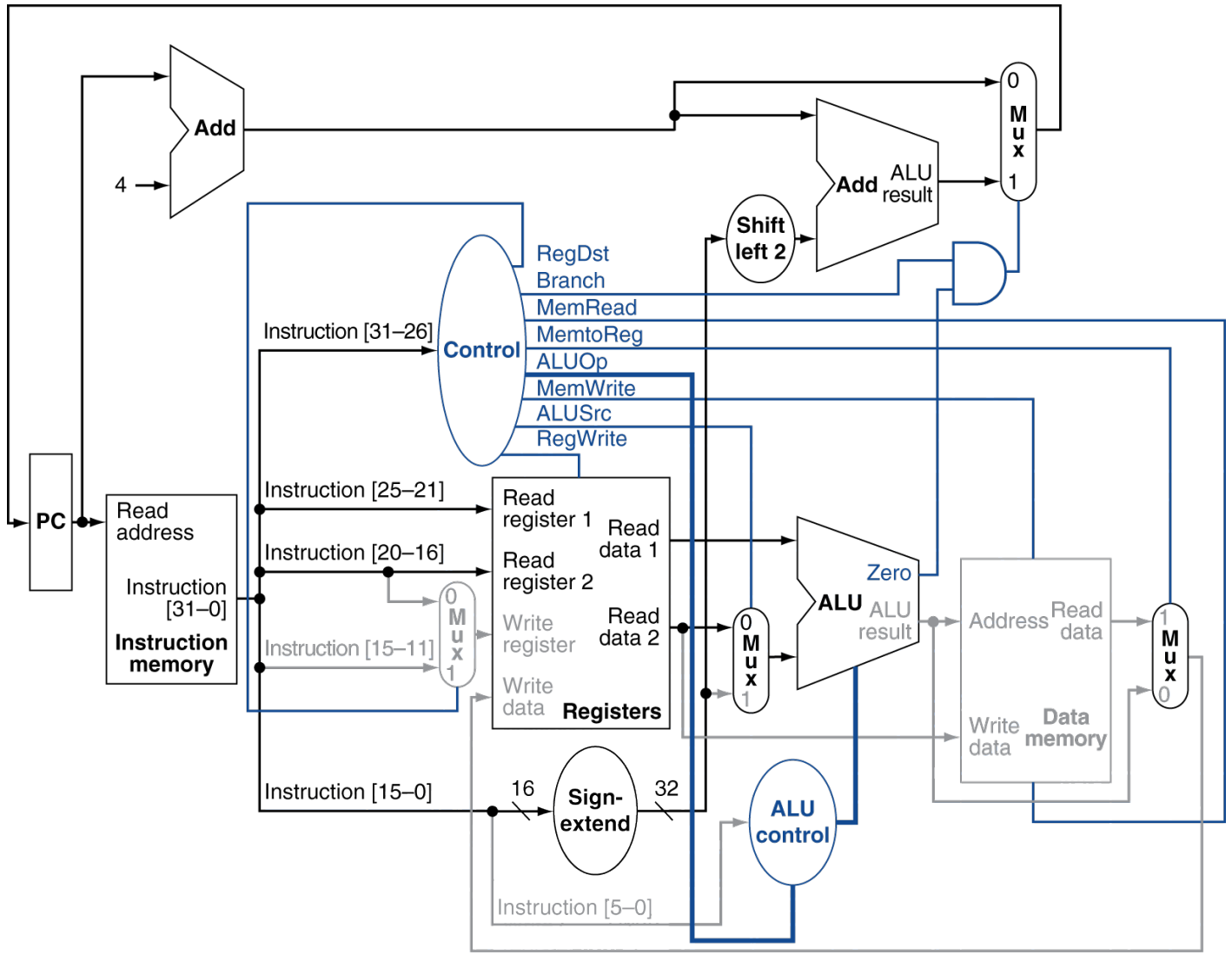
Refining the Complete Datapath



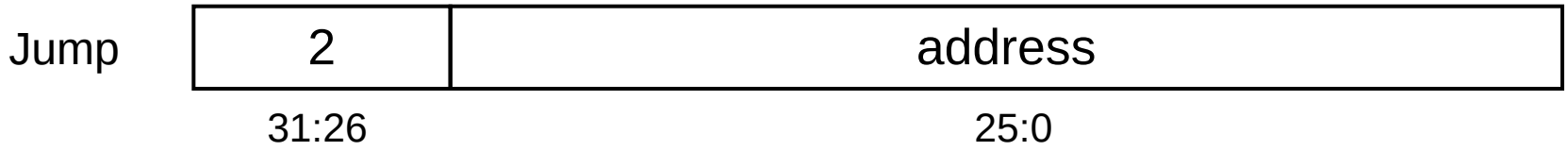
Complete Single-Cycle Datapath



Branch-on-Equal Instruction

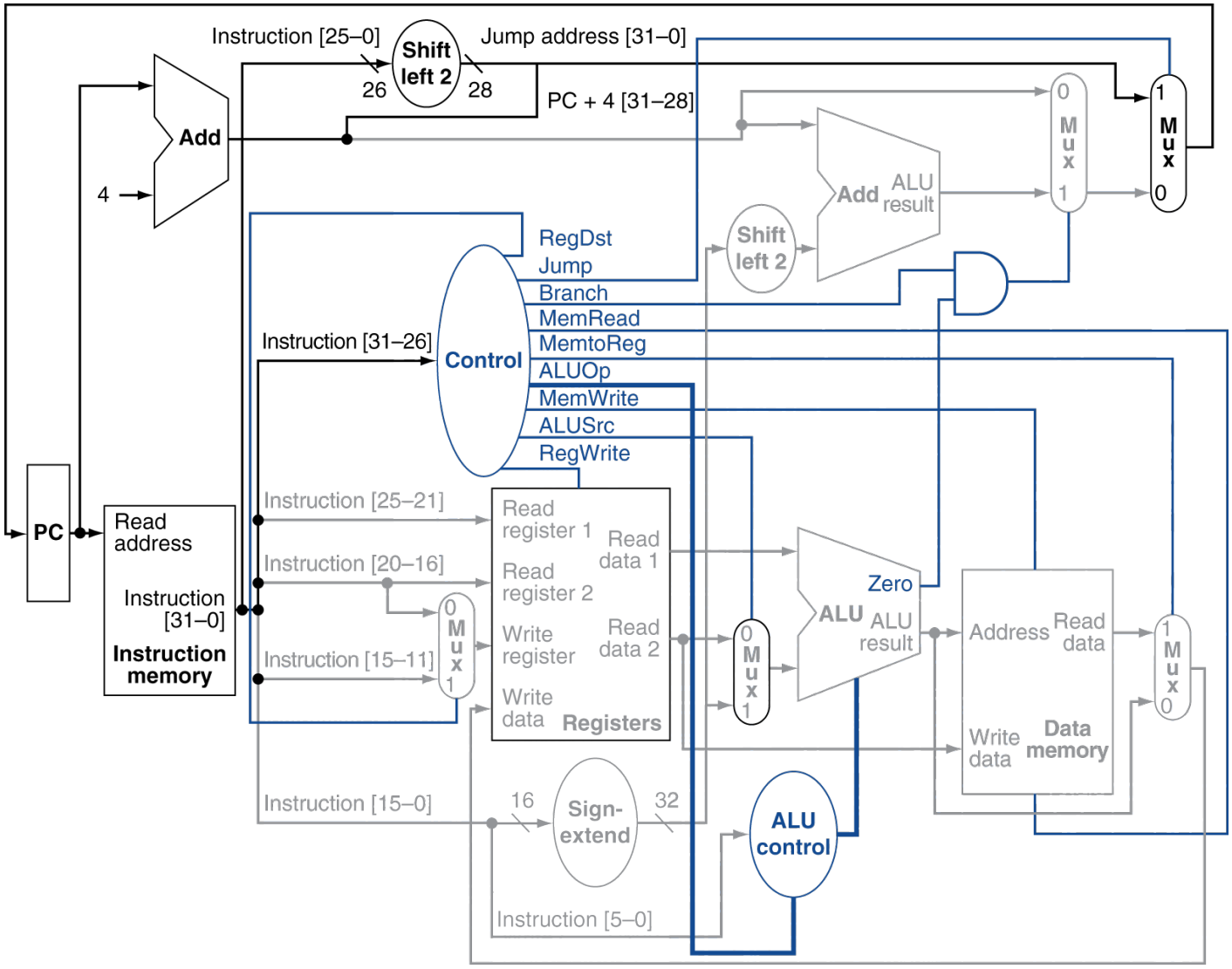


Implementing Jumps



- Jump uses word address
- Update PC with concatenation of
  - Top 4 bits of old PC
  - 26-bit jump address
  - 00
- Need an extra control signal decoded from opcode

Datapath With Jumps Added



Performance Issues

- Longest delay determines clock period
  - Critical path: load instruction
  - Instruction memory → register file → ALU → data memory → register file
- Not feasible to vary period for different instructions
- Violates design principle
  - Making the common case fast
- We will improve performance by pipelining