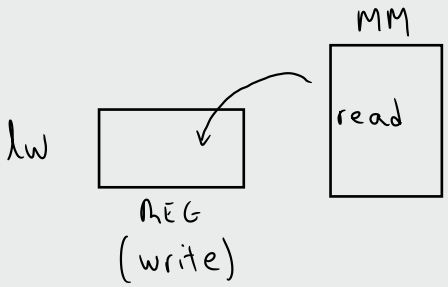
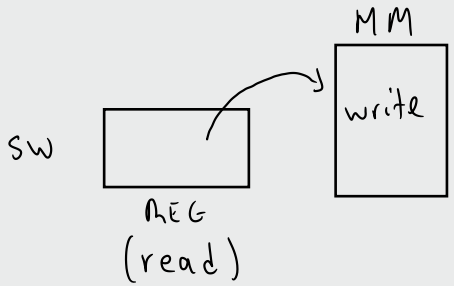
Execution Cycles

- Fetch instruction, update the PC register
  - Instruction memory (read only)
  - Get in with an address
  - The address is stored in the PC register
  - Get out with the instruction
  - PC register is updated [PC + 4]
- Decode
  - The instruction is parsed into fields (R-format has OPCODE 000000)
  - Get the source data
  - Register file (2 read and 1 write port)
  - * Must understand the implementation of the read and write ports (B56 on Patterson)
- Execute
  - ALU: compute logical and arithmetic operations
  - Output: result of the operation, it can represent target address (lw, sw)
    - check for zero
- Write back / memory access
  - Data memory or register file
  - Write-back: R-format
  - Memory access: lw and sw

Instructions to cover

- R-format
- Load word
- Store word
- beq
- J





From Patterson:

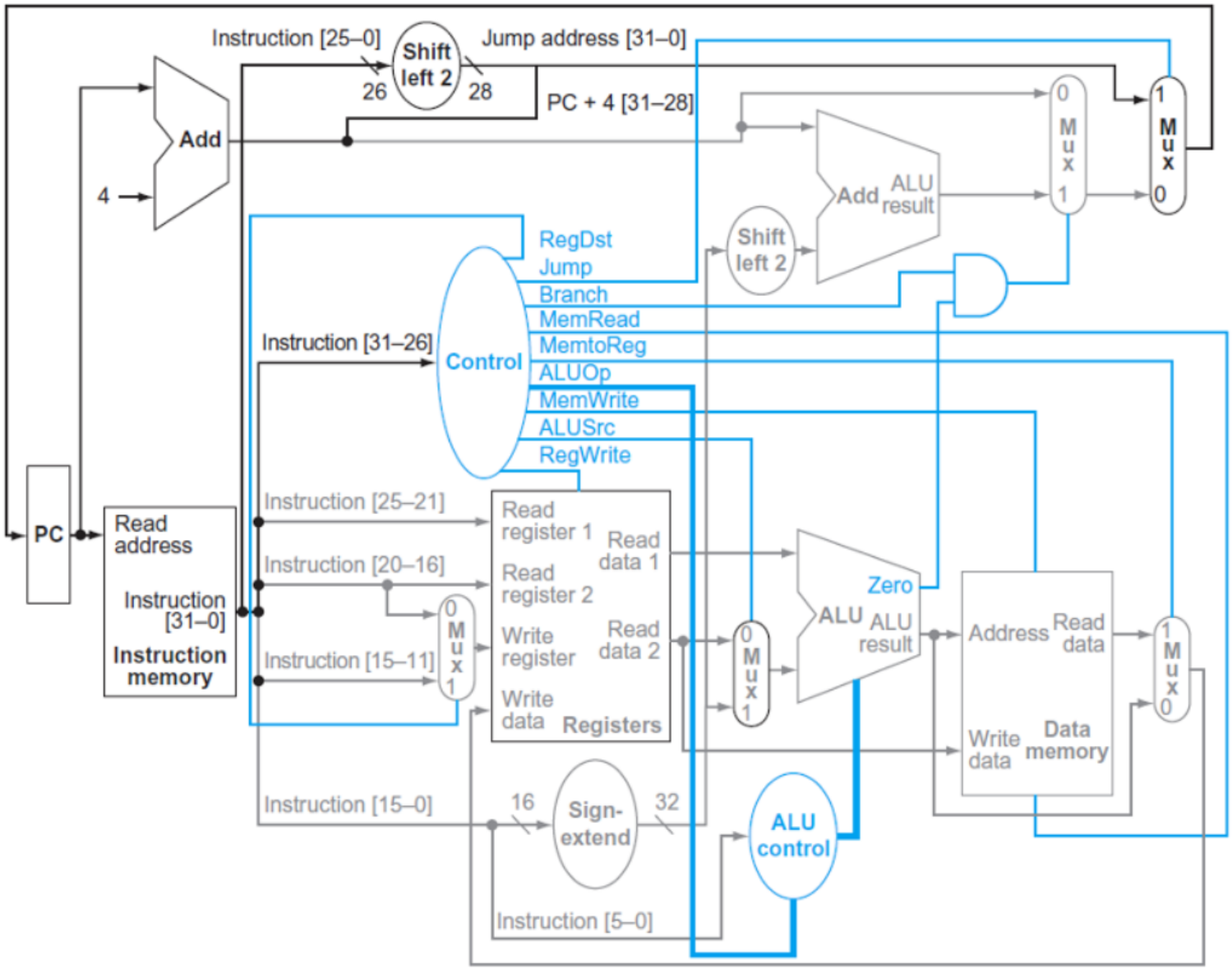
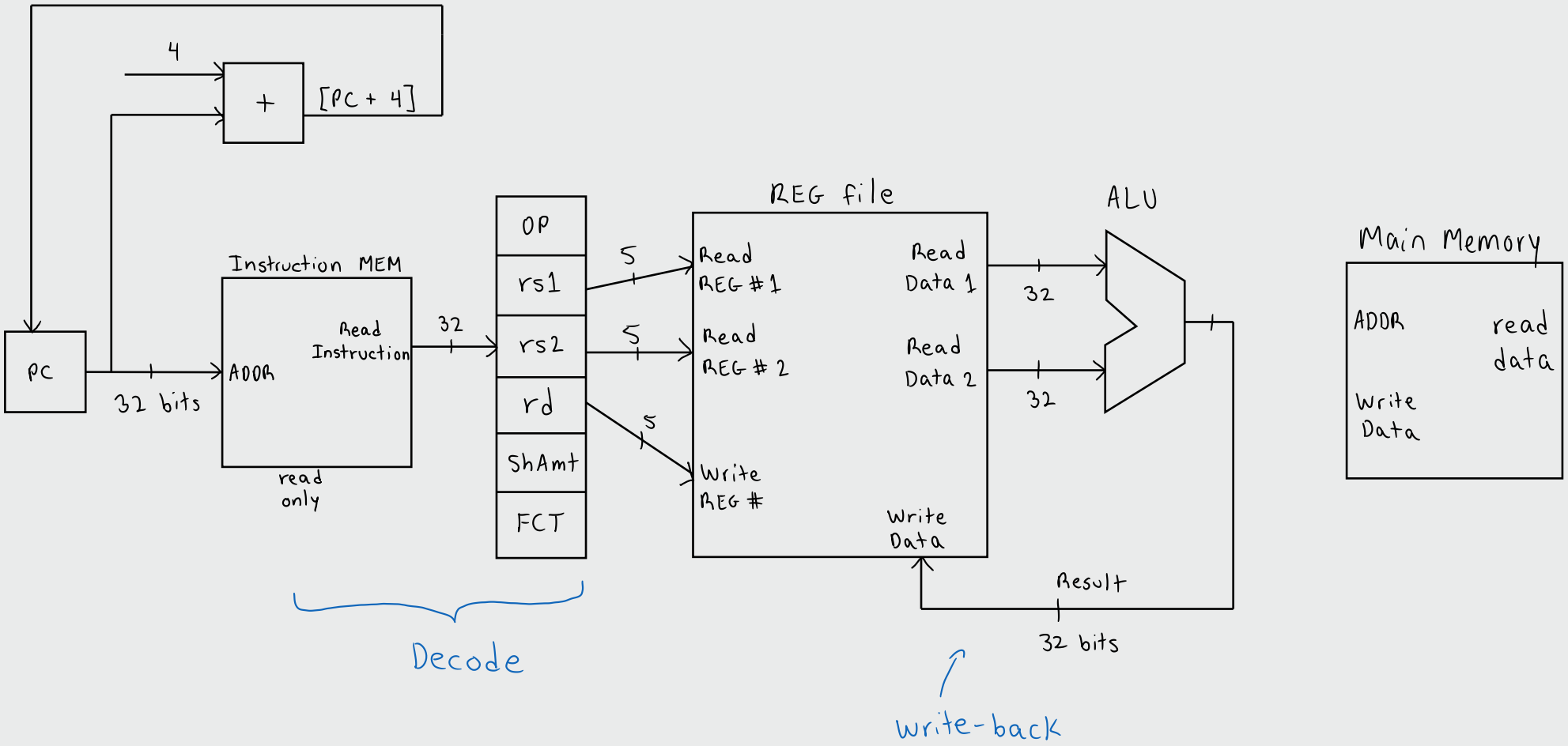
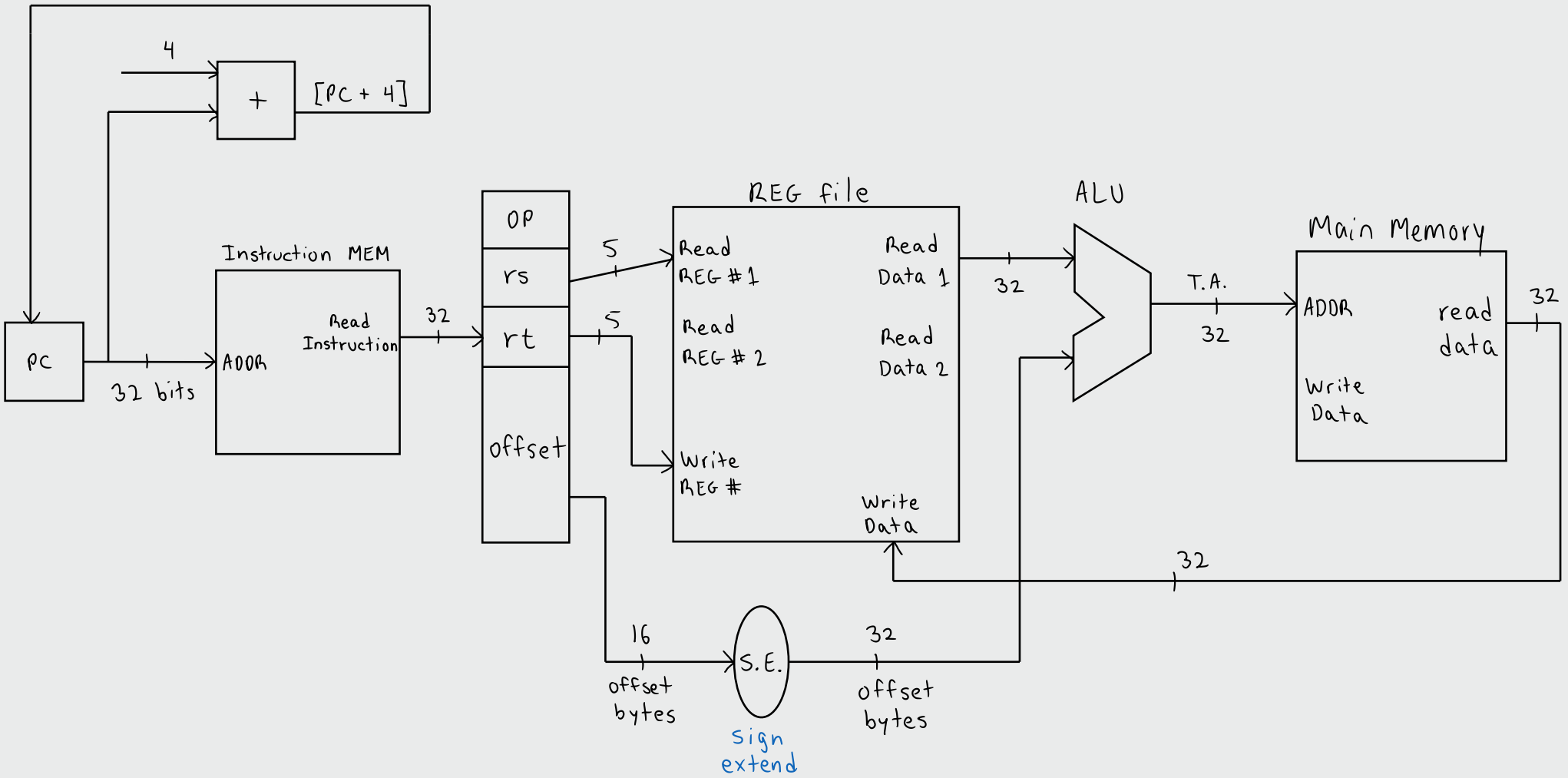


FIGURE 4.24 The simple control and datapath are extended to handle the jump instruction. An additional multiplexor (at the upper right) is used to choose between the jump target and either the branch target or the sequential instruction following this one. This multiplexor is controlled by the jump control signal. The jump target address is obtained by shifting the lower 26 bits of the jump instruction left 2 bits, effectively adding 00 as the low-order bits, and then concatenating the upper 4 bits of PC + 4 as the high-order bits, thus yielding a 32-bit address.

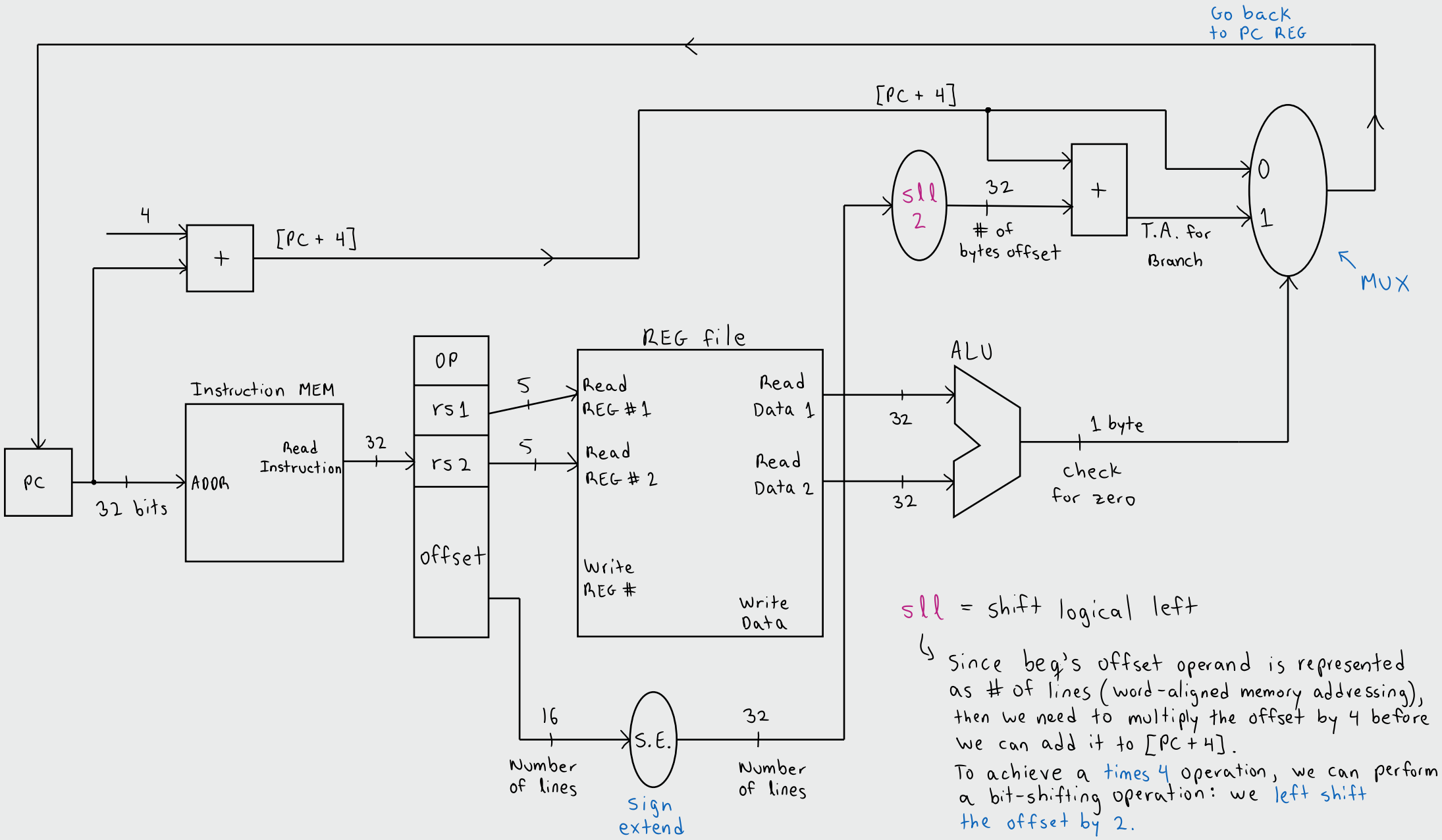
Datapath for R-format



Datapath for lw (load word)



Datapath for beq



sll = shift logical left

↳ Since beq's offset operand is represented as # of lines (word-aligned memory addressing), then we need to multiply the offset by 4 before we can add it to [PC + 4].

To achieve a times 4 operation, we can perform a bit-shifting operation: we left shift the offset by 2.