Register File Block Diagram:

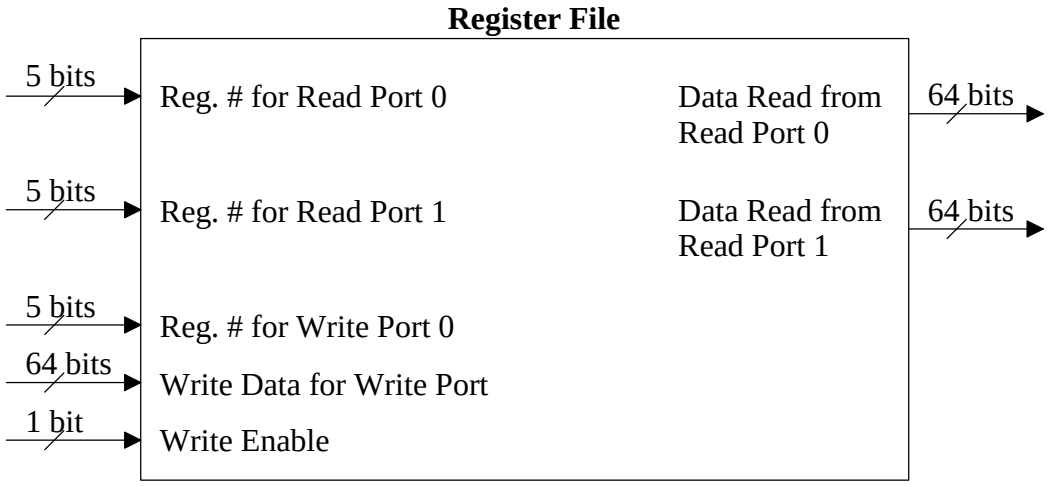
Most register files have at least two read/output ports and one write/input port to accommodate sending two values to the ALU and receiving one result.

To control a read port we need to be able to specify a register number for the register to be read. The width/number of bits read equals the number of bits per register.

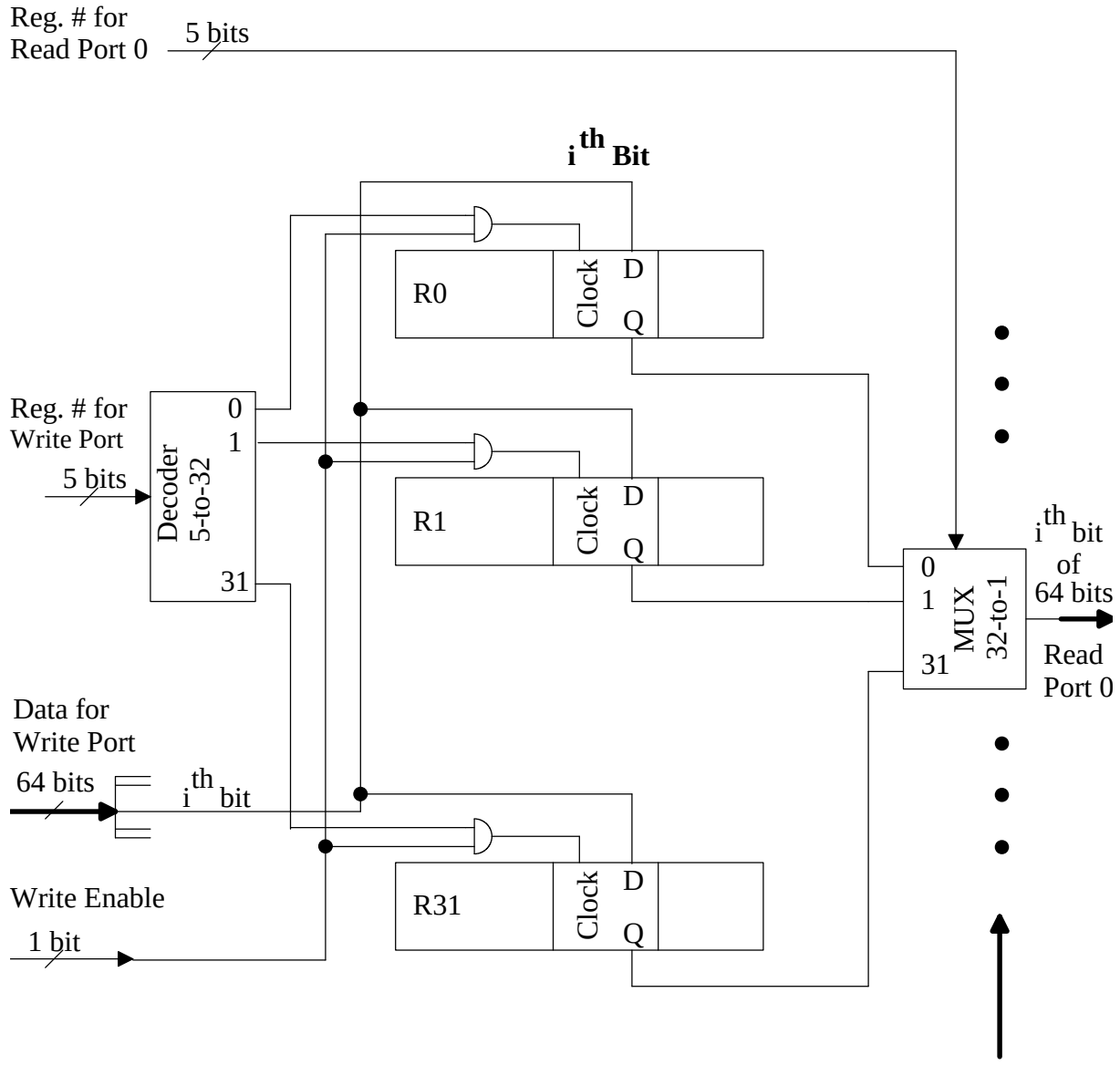
To control a write port we need to be able to specify a register number for the register to be written, the data to be written (equal to the number of bits in a register), and a write-enable signal. The write-enable signal indicates if we are writing a register, i.e., indicates if we should ignore the write register # and data.

A register-file block-diagram assuming:

- 32 registers numbered 0 to 31 (00000_2 to 11111_2 in binary)
- 64-bit registers
- two read ports
- one write port



One-bit (i^{th} bit) Slice of a Register File Implementation Using D-Flip Flops:



Note: To write a D flip-flop, we want its reg. # to be specified and the "Write Enable" to be asserted.

Note: 64 MUXs are need per Read Port - one per bit

For example, a complete (not just a one-bit slice) register file that has:

- 4 registers
- 2-bits per register
- one write-port
- two read-ports