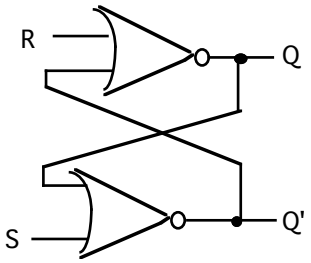
CS240 Laboratory 4
Memory and Datapath

Basic Memory Circuits

Latch Single-bit memory, level-triggered

Flip-Flop Also single-bit, but edge-triggered

SR (Set Reset) Latch



S	R	Q	Q′	
0	0	Qp	Qp′	remember
0	1	0		reset (clear)
1	0	1		set
1	1	unpredictable		

What does **unpredictable** mean? Notice in a NOR gate, if either input = 1 to a gate, its output = 0 (**1** is a deterministic input)**:**

A	B	(A+B)′
0	0	1
0	1	0
1	0	0
1	1	0

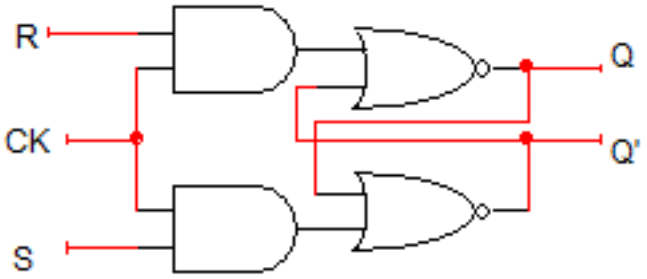
So, although you wouldn't usually try to *set* and *reset* at the same time (it doesn't make sense), if you did, Q and Q′ will both be 0 (which is not unpredictable).

However, when you go back to the *remember* state (S=R=0), Q and Q′ will not stay at 0 0. The circuit passes through one of either the *set* or *reset* state on its way back to the *remember* state, and Q and Q' change to the complement of one another.

Since the final state depends on which transitional state was sensed on the way back to *remember*, you cannot predict whether the final state of Q will be 1 or 0.

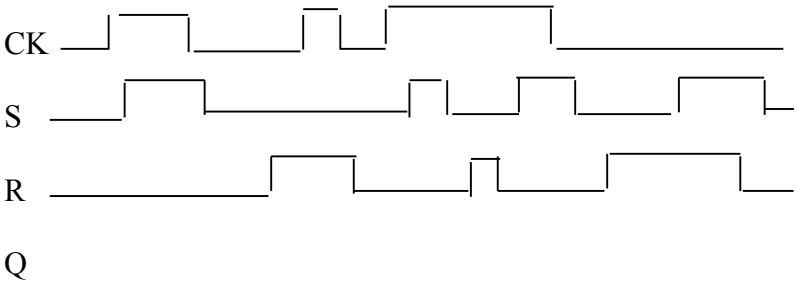
Clocked SR Latch

Incorporates a clock input/level-sensitive



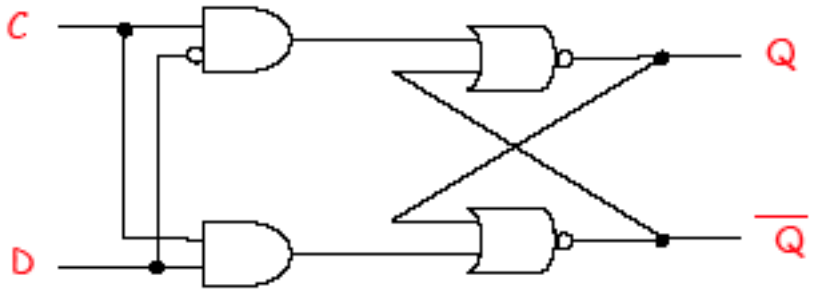
Output Q can change in response to S and R whenever the CK input is asserted.

How does Q respond to the following inputs?



D Latch

Avoids unpredictable state, because a single input D determines the next state of the circuit.

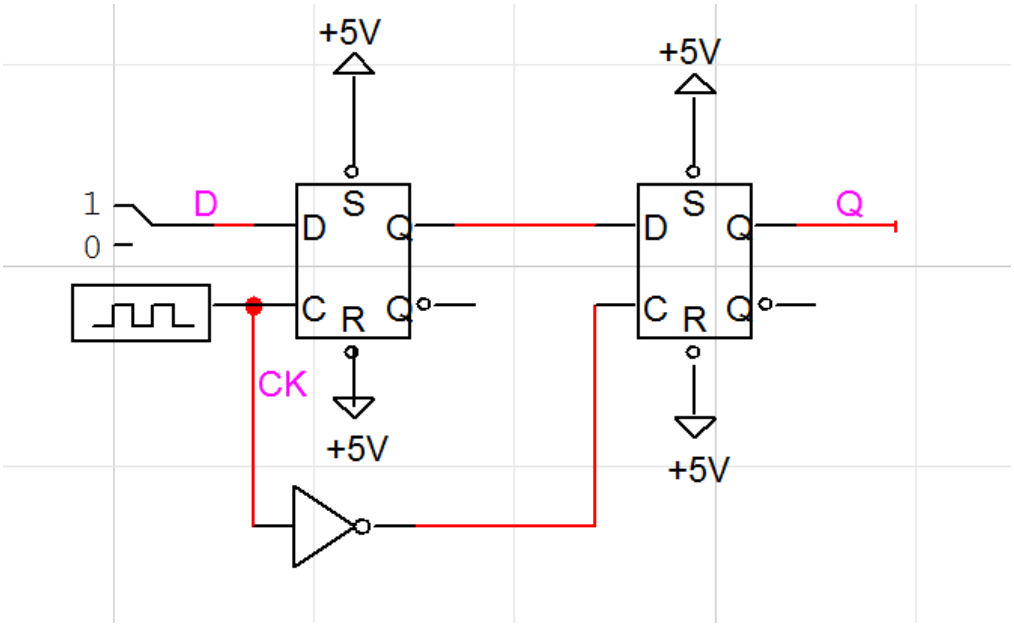


D	Qnext
0	0
1	1

D Flip-Flop

Changes state on a clock transition (edge), rather than whenever the clock is asserted.

Internally, a flip-flop is made from 2 latches. The first latch is controlled by the clock, but the second latch is controlled by the *inverse* of the clock:

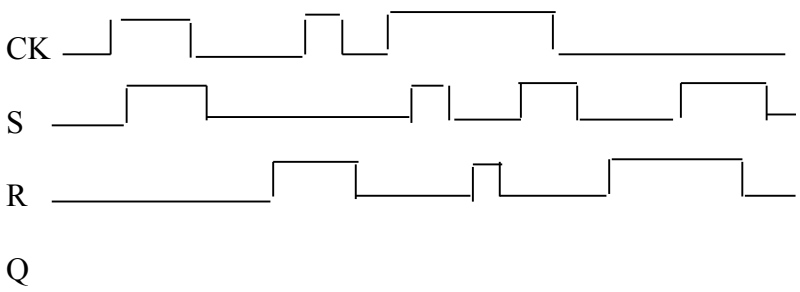


So, the input D will not be passed from the first latch to the second latch until the clock goes low.

Once the clock is low, a new value on D will not store into the first latch. Overall, the flip-flop can change value only *exactly* at the transition of the clock from high to low.

Output Q can change in response to S and R only on the positive edge of the clock.

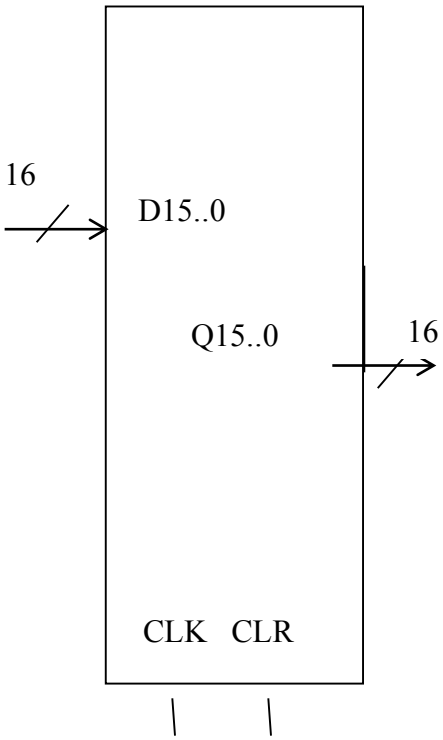
How does Q respond to the following inputs?



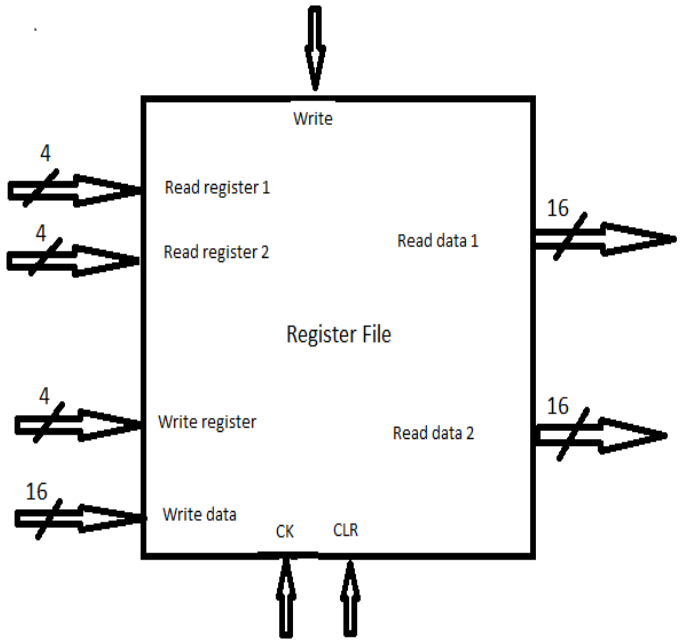
Notice the difference between Q and the output for the earlier clocked latch example.

Circuits using Flip-flops

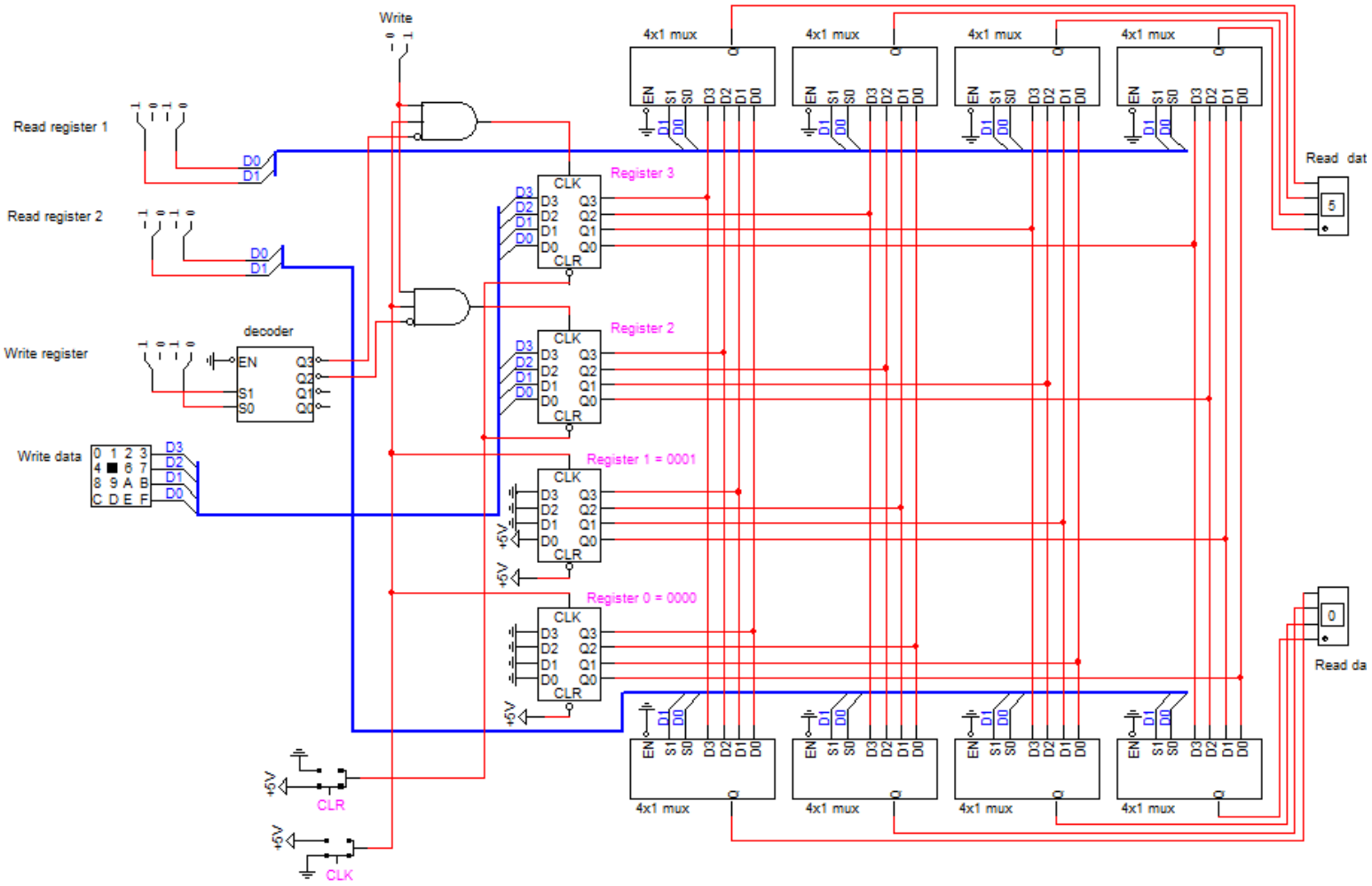
Register - n-bit memory, uses *n* flip-flops, and shared *clock* and *clear* inputs



Register File set of registers



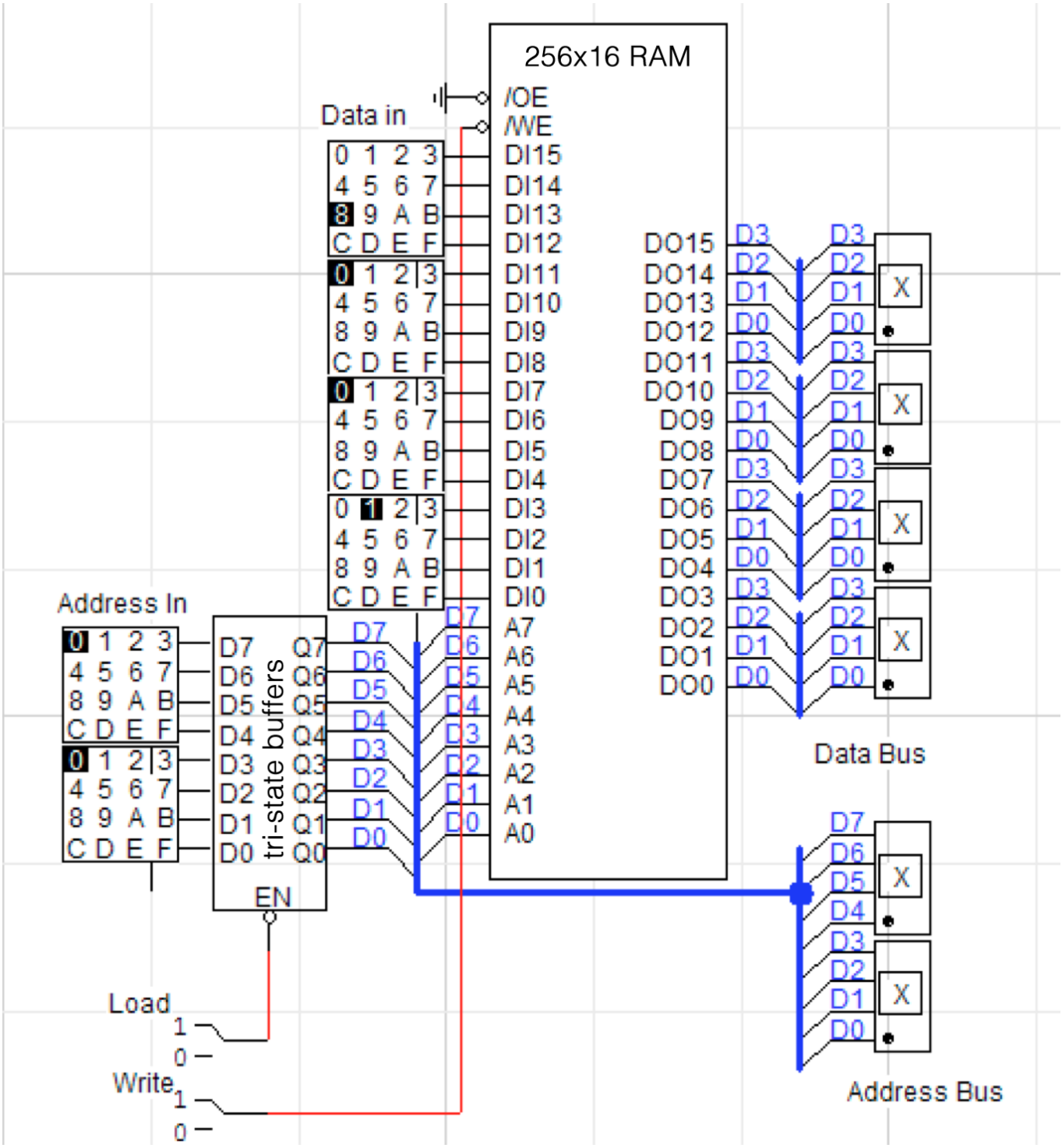
- **Write** is the write control signal.
- **Write register** is the number of a register to be written with a new value
- **Read register 1** and **2** indicate which 2 registers can be read at data ports **Read data 1** and **Read data 2** at any given time
- clear and clock (**CLR** and **CLK**) are shared by all the 16 registers.



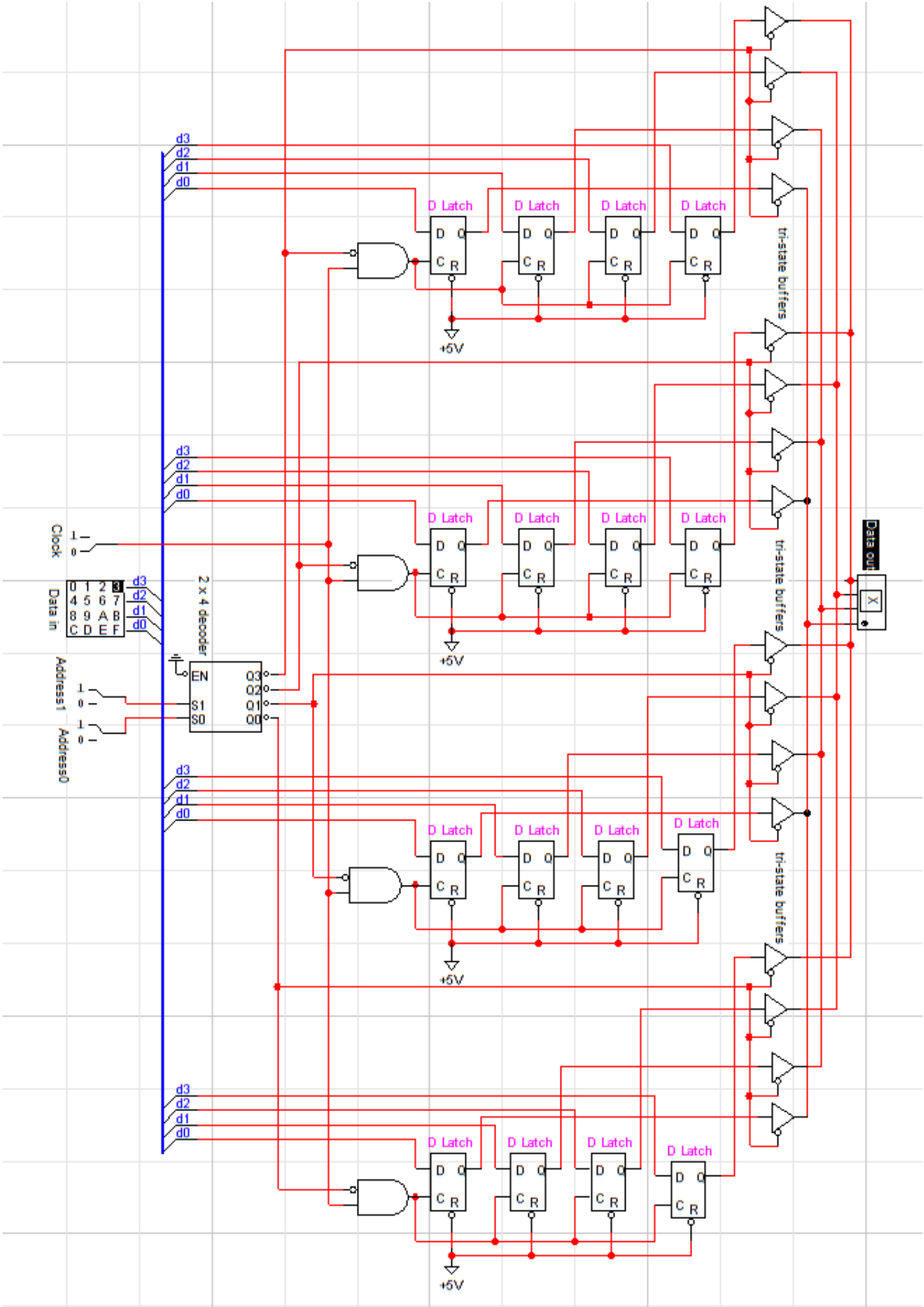
- 2 sets of 4 x 1 multiplexers select which 2 registers are currently being output at the two read ports.

- A decoder uses the write register number to select which of the 4 registers will receive a new value on a write.

RAM memory contains multiple flip-flops, organized into n-bit words, where each word can be accessed through use of an address:



Internal lay-out looks like this:


d3
d2
d1
d0
D Latch
D Q
C R
+5V
tri-state buffers
Clock
Data in
Address1
Address0
2 x 4 decoder
EN
S1
S0
Q3
Q2
Q1
Q0
Data out
X

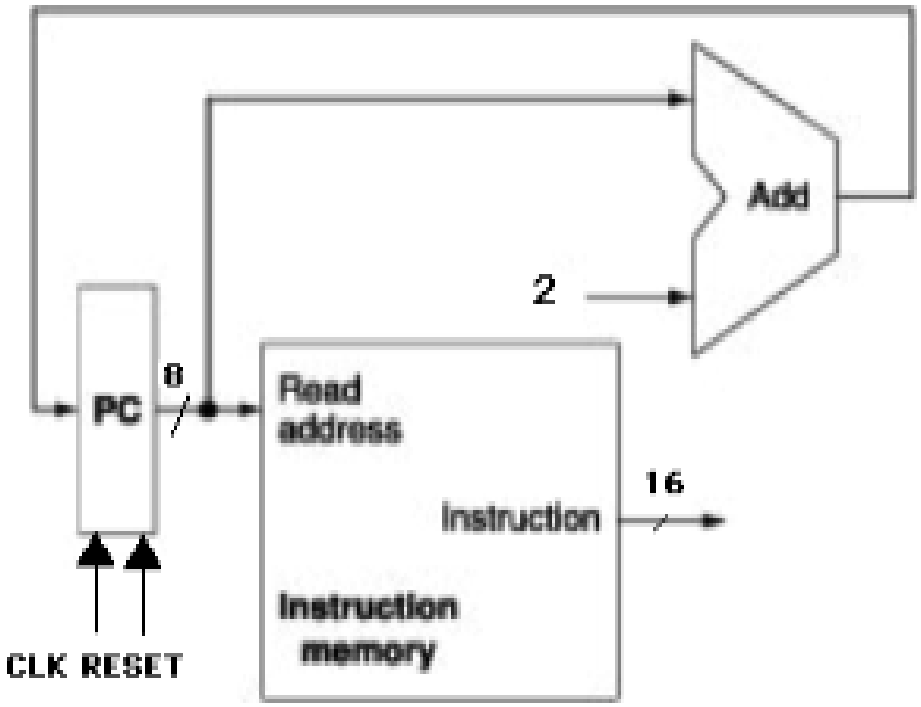
HW Instruction Set Architecture

- **16 bit data bus**
- **8 bit address bus**
- Starting address of every program = 0 (PC initialized to 0 by a reset to begin execution)
- PC incremented by 2 to move to the next instruction.
- **16 registers**
  R0 = 0 (constant)
  R1 =1 (constant)
  R2-R15 general purpose

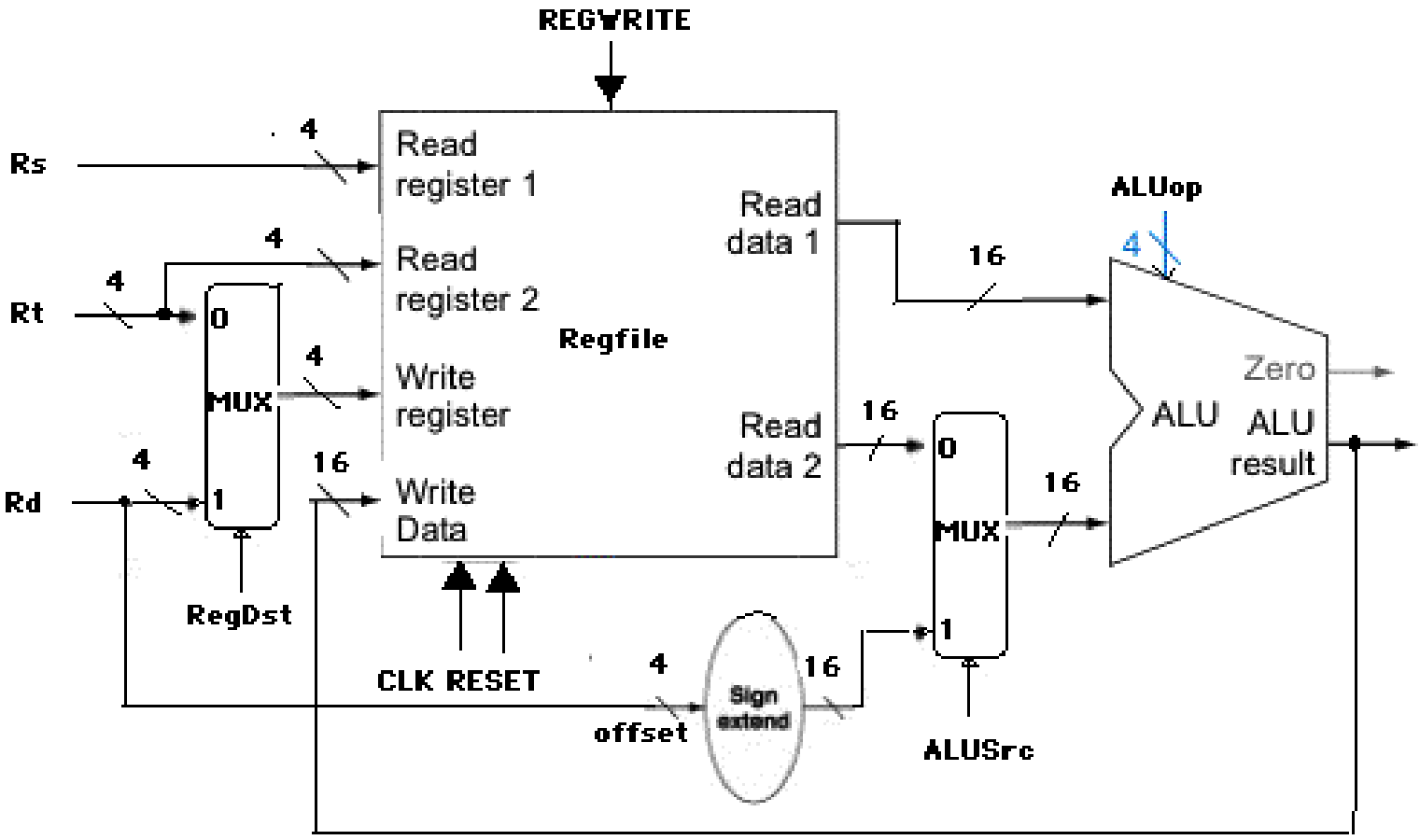
Instruction Set

Instruction	Meaning	Op 4-bit	Rs 4-bit	Rt 4-bit	Rd 4-bit	
LW Rs,Rt,offset	Rt loaded with word from Data Memory at address(Rs + offset)	0000		0-15	0-15	offset
SW Rs,Rt,offset	Data Memory address(Rs + offset) stored with word from Rt	0001		0-15	0-15	offset
ADD Rs,Rt,Rd	Rd := Rs + Rt	0010		0-15	0-15	0-15
SUB Rs,Rt,Rd	Rd := Rs - Rt	0011		0-15	0-15	0-15
AND Rs,Rt,Rd	Rd := Rs AND Rt	0100		0-15	0-15	0-15
OR Rs,Rt,Rd	Rd := Rs OR Rt	0101		0-15	0-15	0-15
BEQ Rs,Rt,offset	If Rs=Rt then pc:=pc+2+(offset*2) else pc:=pc+2	0111		0-15	0-15	offset
JMP offset	Jump to abs. addr = offset*2	1000		---12 bit offset-----		

Instruction Fetch



Register File and ALU



R-type instructions ADD,SUB,AND,OR,SLT **(**opcode Rs Rt Rd)

- read Rs and Rt from register file
- perform an ALU operation on the contents of the registers
- write the result to register Rd in register file

Memory Access instructions LW,SW **(**opcode Rs Rt offset)

- memory address = Rs + sign-extended 4-bit offset
- if **SW**, the value to be stored to memory is from Rt.
- if **LW**, Rt is loaded with the value read from memory

Register written to (**Write Register**) is Rd **or** Rt if a **LW** instruction (chosen by a 2x4 MUX which is controlled by **RegDst**)

ALU calculates Rs + Rt, **or** Rs + sign-extended offset.

- Input A of the ALU is always Rs

- Input B of the ALU is Rt **or** the offset
  (chosen by a 2x16 multiplexer, which is controlled by **ALUSrc**)