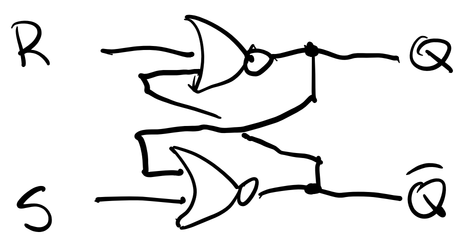
Where we left off: unclocked latch



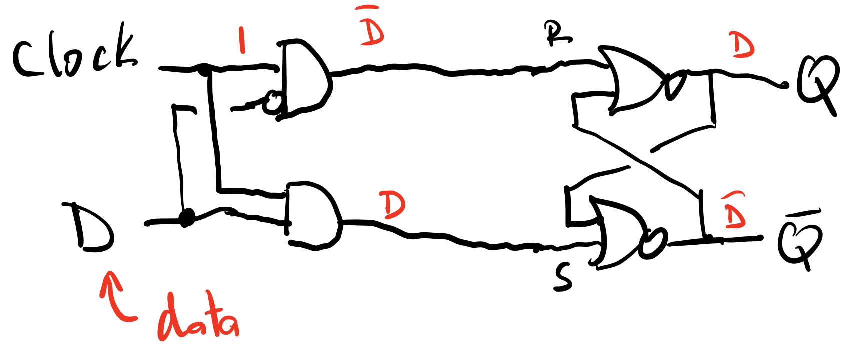
$S=1, R=0 \Rightarrow Q=1, \overline{Q}=0$

$S=0, R=1 \Rightarrow Q=0, \overline{Q}=1$

$S=0, R=0 \Rightarrow Q$ and $\overline{Q}$ hold their value.

Adding a timing element: using the clock.

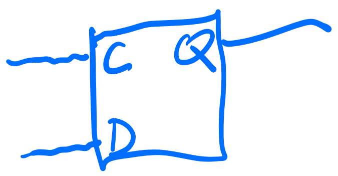
Clocked latch ("D-latch")



When clock = 0, Q and $\overline{Q}$ hold their value.

When clock = 1, Q gets the value of D and $\overline{Q}$ sets the value of $\overline{D}$.

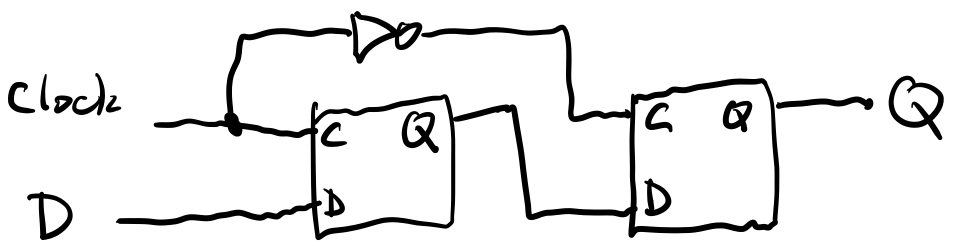
Draw as:



ignore $\overline{Q}$

"Flip Flop" - with falling edge trigger

- uses two D-latches:

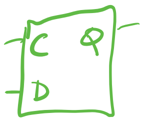


- When clock=1, D flows into the left latch and the left latch's Q changes. However, at that point the C input to the right latch is ϕ, so the Q output of the right latch doesn't change.

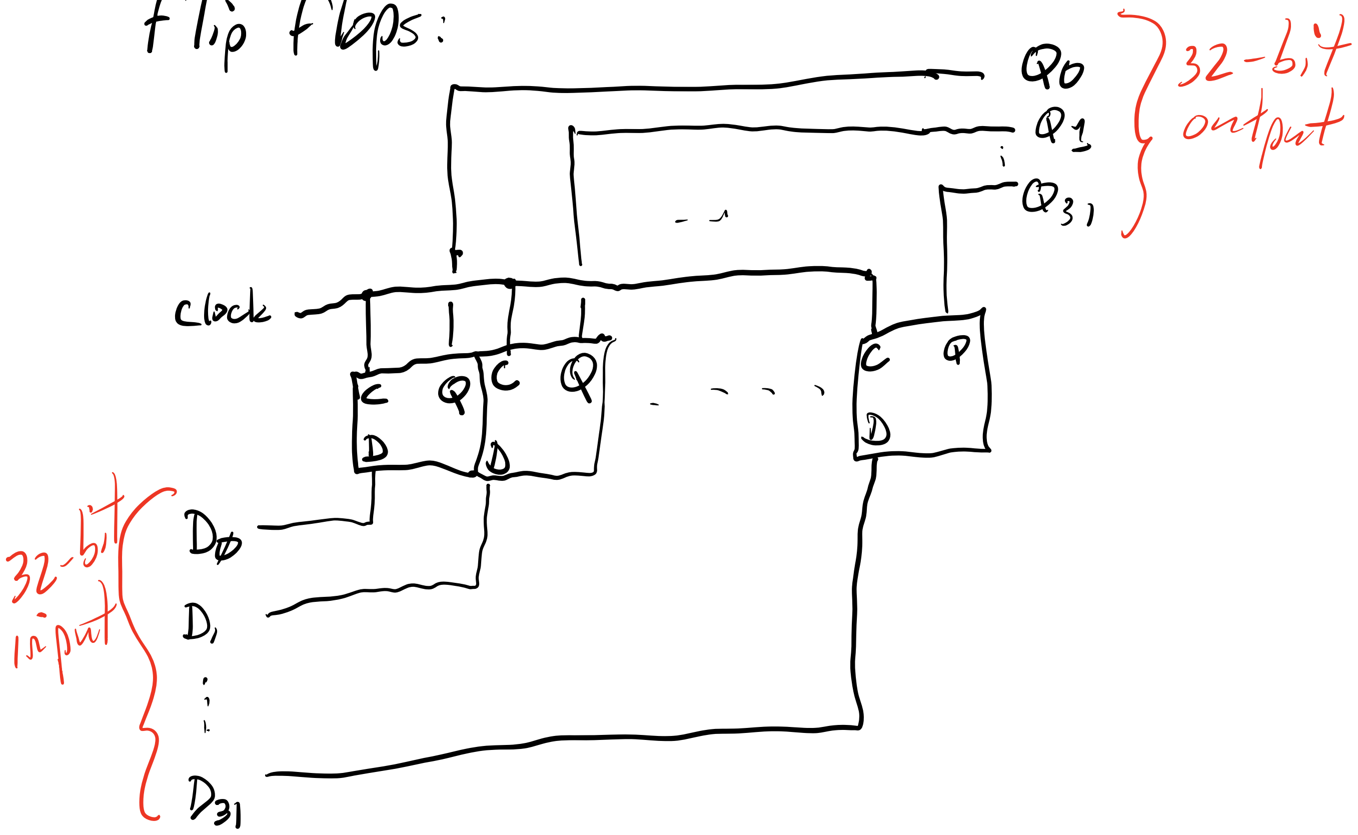
- When clock falls to ϕ, no change occurs on the Q of the left latch.
However, the C input to the right latch becomes 1, so the Q output of the left latch flows into the D input of the right latch and is output on the right latch's Q.

The flip flop is the building block of registers & caches.

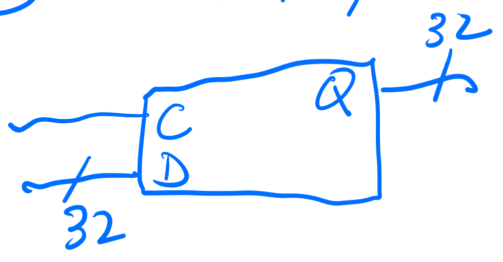
- "SRAM" (static ram)
- fast
- expensive (lots of transistors)

Now we'll use  to represent a flip flop

A 32-bit register consists of 32 flip flops:



Draw a register as:



"Register File"

- a collection of identical registers.
  - e.g. %rax, %rbx, ...
- typically allows two reads and one write to registers at a time.

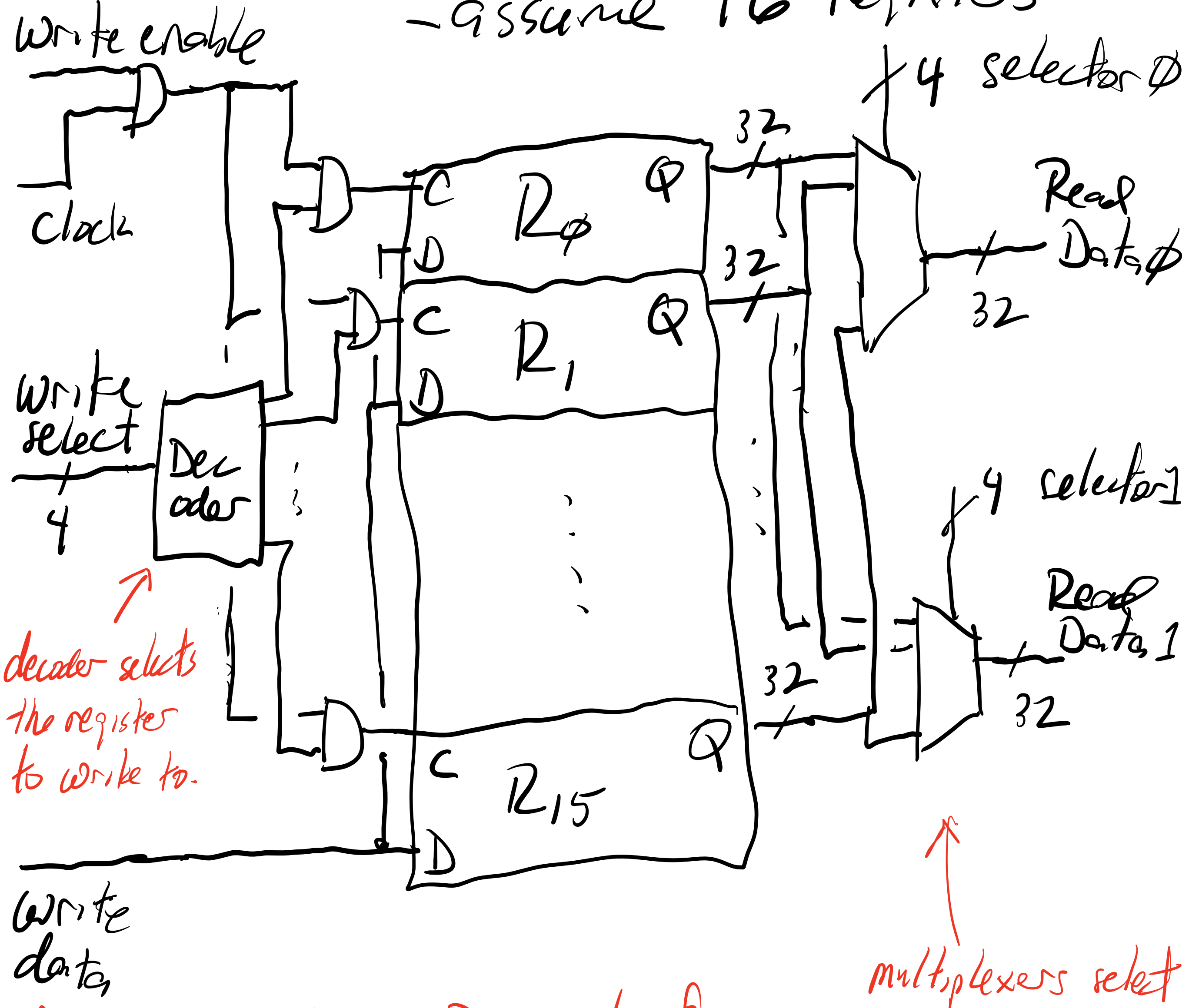
just a number in machine code

addq %rax, %rsi

- reading from %rax and %rsi, performs addition, writes result to %rsi.

Building a register file.

- assume 16 registers



decoder selects the register to write to.

↑ gets sent to the D input of every register.

multiplexers select the registers to read from.

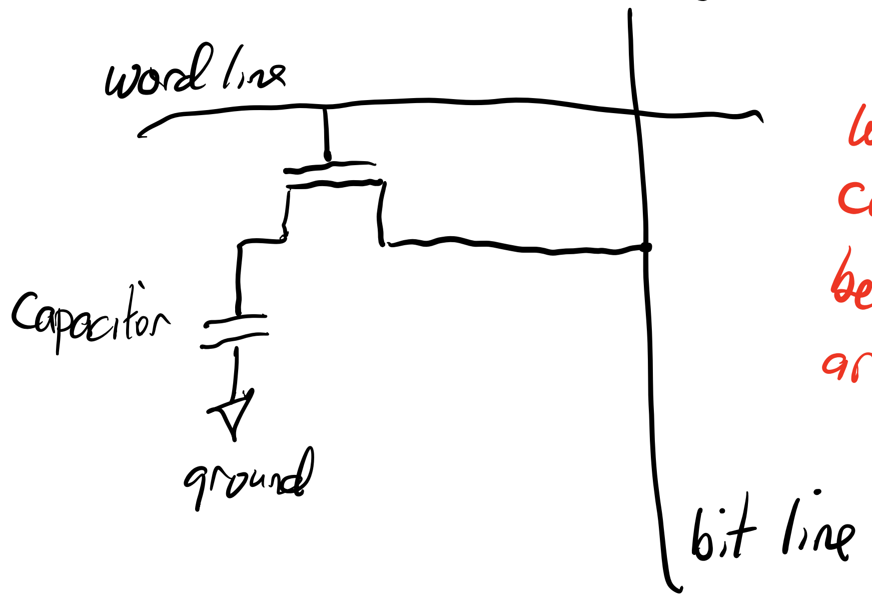
- A register can only change when:
  - the clock falls
  - write_enable = 1
  - write_select selects that register.
    - through the use of the decoder.

Main Memory is not made from flip flops.
- too expensive, not dense enough

Main memory uses "DRAM"
- dynamic RAM
- uses one transistor and one "capacitor" per bit.

↑ stores voltage (tiny battery)

One DRAM cell (one bit)



When word line = 1, current will flow between the bit line and the capacitor.

When the capacitor holds a charge (high voltage), that represents a 1.
When the capacitor is discharged (low voltage), that represents a ∅.

Setting word line = 1, bit line = 1, will cause current to flow from the bit to the capacitor, charging the capacitor – writing a 1.

setting word line = 1, bit line = 0, will cause current to flow from the capacitor to the bit, discharging from the capacitor.

- writing a 0.

To read, set word_line = 1 and set bit_line to an intermediate voltage level. If the voltage on bit_line goes up, that means the capacitor had held high voltage and the current flowed from the capacitor to the bit line.

- so, the value read is 1.

If the voltage on bit_line goes down, that means the capacitor held low voltage.

- The value read is 0.