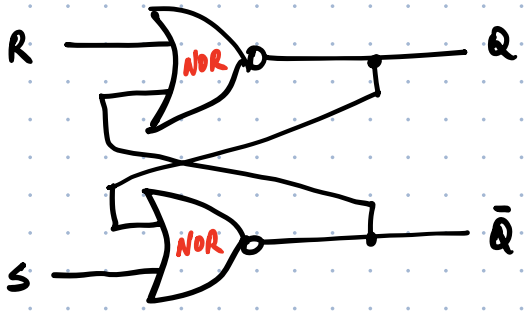
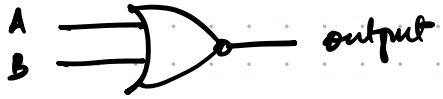
Latches

- 1-bit storage sequential circuits

* S-R Latch





A	B	output
0	0	1
0	1	0
1	0	0
1	1	0

Q_i	R	S	Q_{i+1}	
0	0	0	0	]
1	0	0	1	]
0	0	1	1	]
1	0	1	1	
0	1	0	0	]
1	1	0	0	
0	1	1	undefined	
1	1	1	undefined	

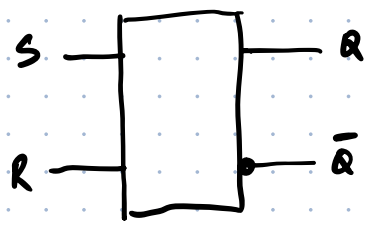
R=0, S=0, maintenance ($Q_{i+1} = Q_i$)

R=0, S=1, set ($Q_{i+1} = 1$)

R=1, S=0, reset ($Q_{i+1} = 0$)

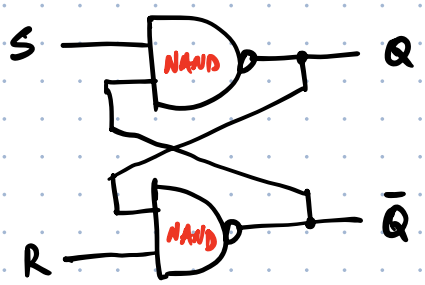
R=1, S=1, undefined behavior.
∵ Q and $\bar{Q}$ are 0

We often denote an S-R catch as follows



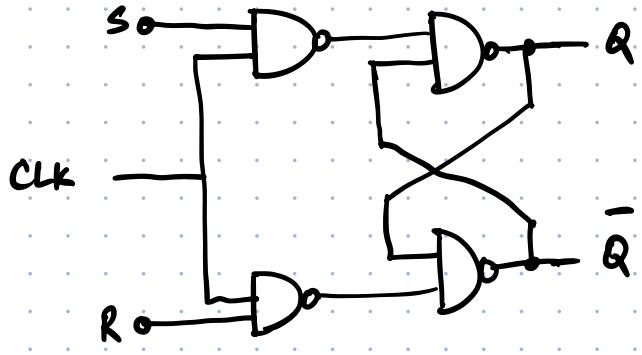
R	S	Q_{i+1}
0	0	Q_i
0	1	1
1	0	0
1	1	undefined.

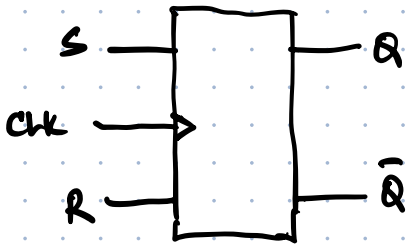
* S-R Latch using NAND gates



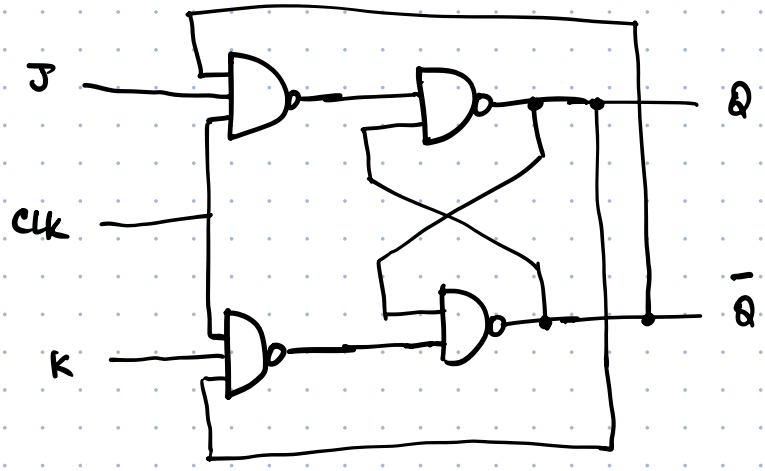
A	B	output
0	0	1
0	1	1
1	0	1
1	1	0

* The problem with latches
  - The output values Q_{i+1} start changing immediately when the input changes
  - S-R Flip Flop
    - Flip Flops are edge triggered.

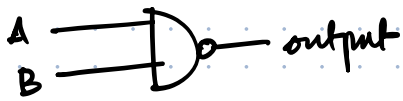




* JK Flip Flop
  - Identical to SR Flip Flop, except it uses S=1, R=1 combination.
  - We do not refer to inputs as S and R. Rather we'll call these J and K.

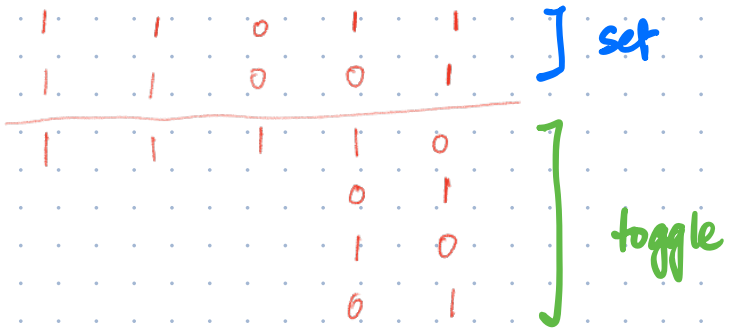


NAND is 0 only when every input is 1.

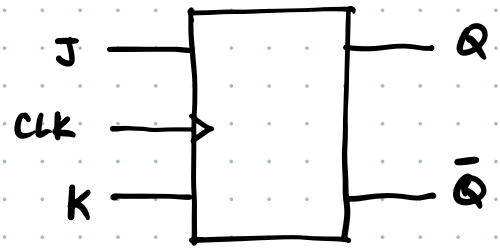


A	B	output
0	0	1
0	1	1
1	0	1
1	1	0

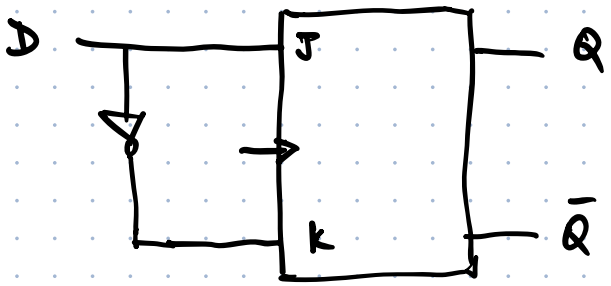
CLK	J	K	Q_i	Q_{i+1}	
0	x	x	1	1	] maintain
0	x	x	0	0	
1	0	1	1	0	] reset
1	0	1	0	0	



We represent JK Flip Flop as

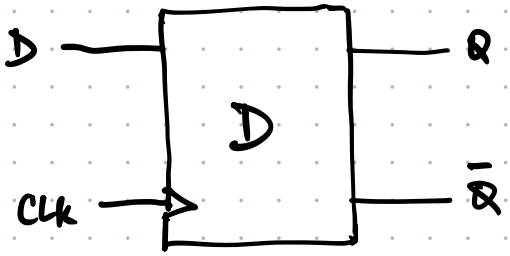


* D Flip Flop
  - Much simpler
  - Has one input only
  - When D=1, the value is set to 1
  - When D=0, the value is set to 0



D	Q_{i+1}
0	0
1	1

We often denote the D Flip Flop as



* Registers
  - High speed memory component accessible to the CPU.
  - Special purpose registers

(i) Program counters: store the address of the next instruction to be executed.

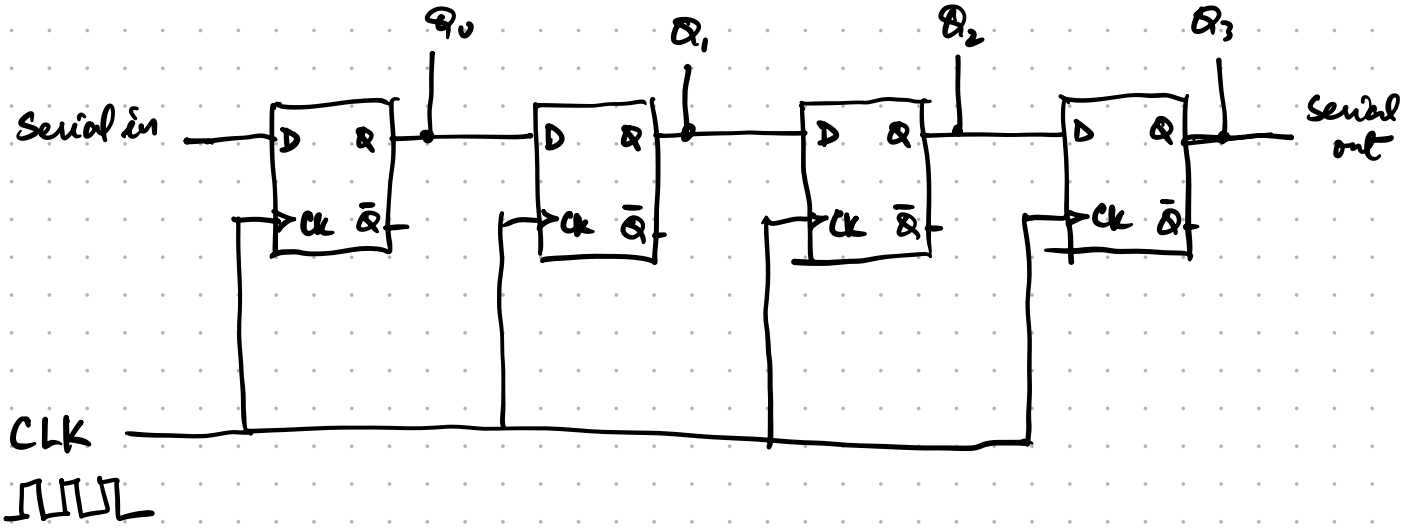
(ii) Memory address registers: store address of memory for storage / retrieval

(iii) Memory buffer registers: store values that are to be written to the memory or that are retrieved from the memory.

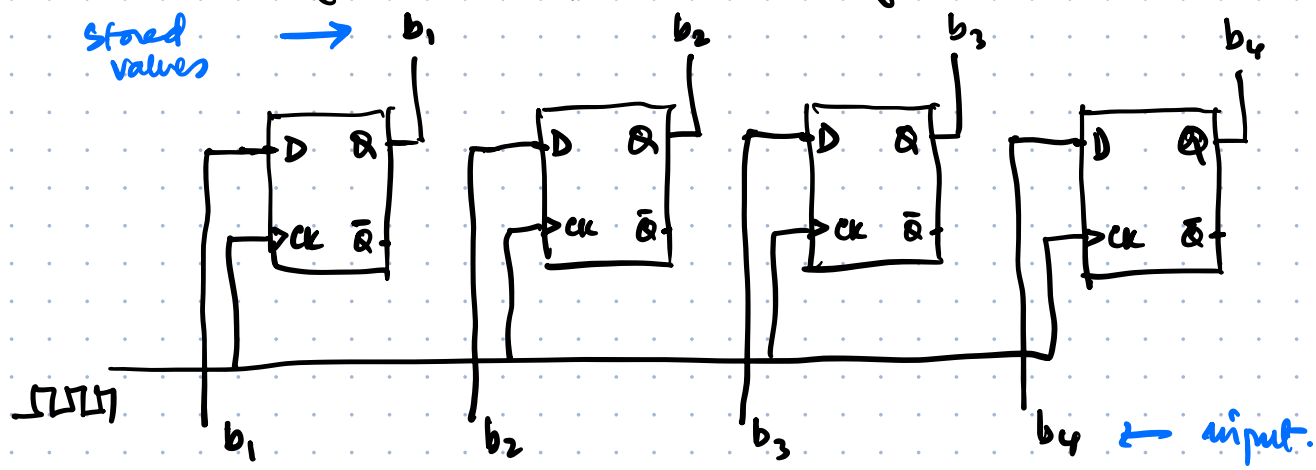
(iv) Status registers: store flags, such as carry and overflow.

- General purpose registers that are used as operands for arithmetic operations.

* A 4-bit register with serial loading



* A 4-bit register with parallel loading.



* Counters (Next time)