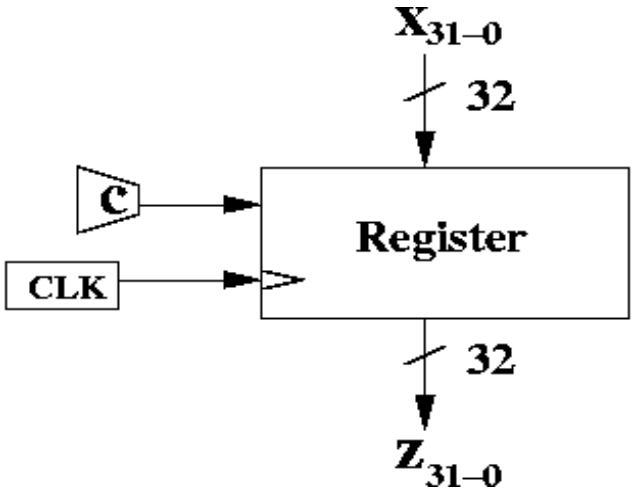
Registers

Flip-flops are the basic building blocks for registers

Parallel-load register (32-bit) has 2 operations

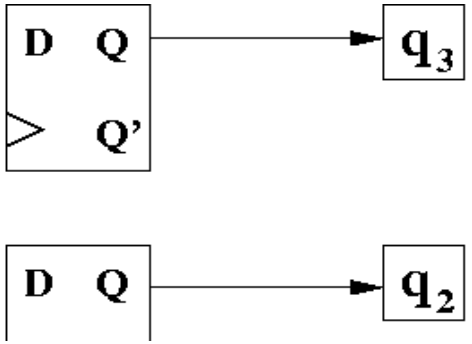
- **c (control) = 0: hold**
- **c (control) = 1: load ($z_{31-0} = x_{31-0}$)**

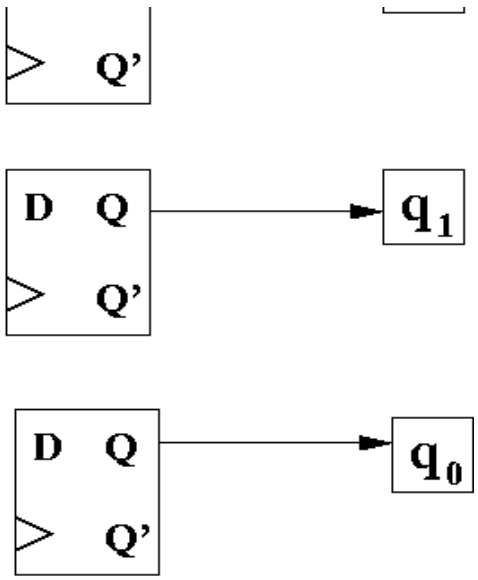
Block diagram



Don't forget the clock!

Example: 4 bits, using D flip-flops




Q'
D
Q
Q'
q_1
D
Q
Q'
q_0

Registers

Need to choose inputs: hold or parallel load

Use 2-1 MUX

Hold: need to keep value constant

D flip-flop sets Q to value of D

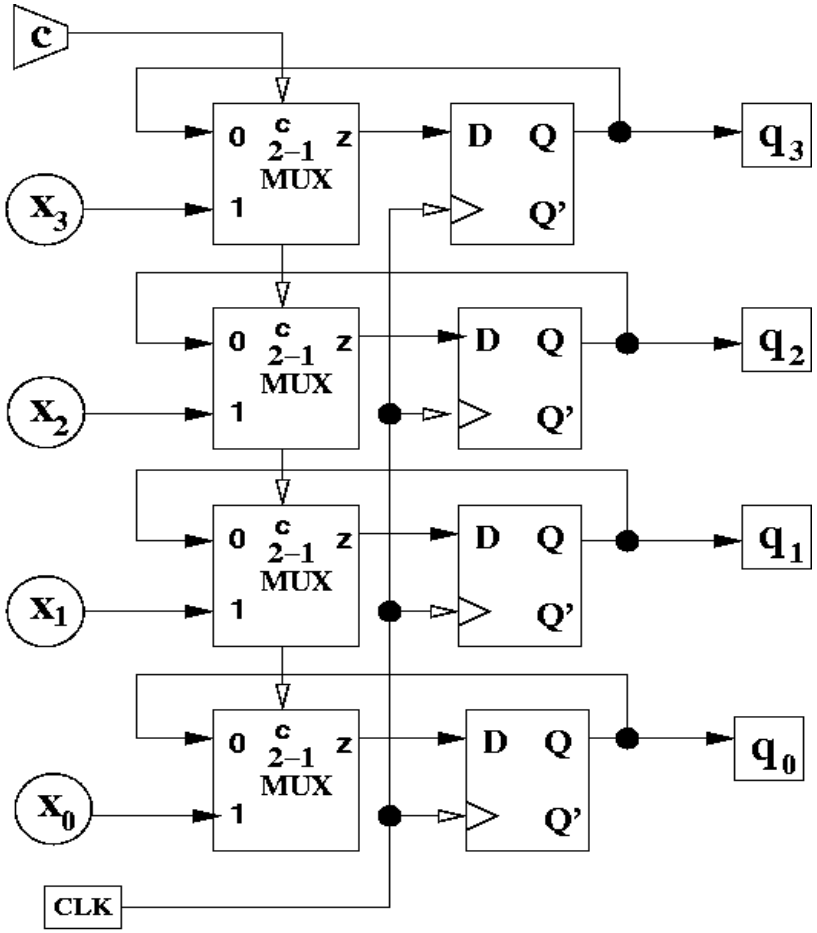
feed Q back to 0 input of MUX

Parallel load: set flip-flop value to input

feed input x to 1 input of MUX

Also need clock and control input

(Note that control is shown going through each MUX)



Registers: T flip-flops

Register with T flip-flops: for simplicity, implement with 2 flip-flops

How does T flip-flop hold the value?

`T = 0`

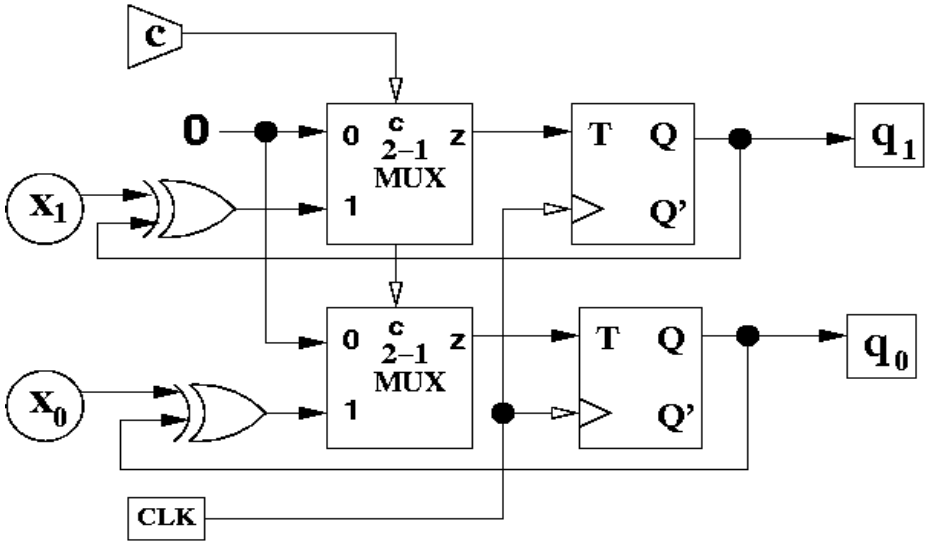
How does it load?

`T = 1` toggles

If (`x != Q`), then toggle

MUX input 1:

`T = x XOR Q`



What other operations could we do?

Shift is simply another form of load!

Input is from current state, rather than external

Shift left: load bit i with value of bit (i-1)

Shift right: load bit i with value of bit (i+1)

Can add more control bits to select shift operations as well as regular load

Change MUX to 4-1:

control	operation
00	hold
01	parallel load
10	shift left logical 1 bit
11	shift right logical 1 bit