Lecture 20

Flip-flops = form of memory

Lots of memory = lots of flip-flops

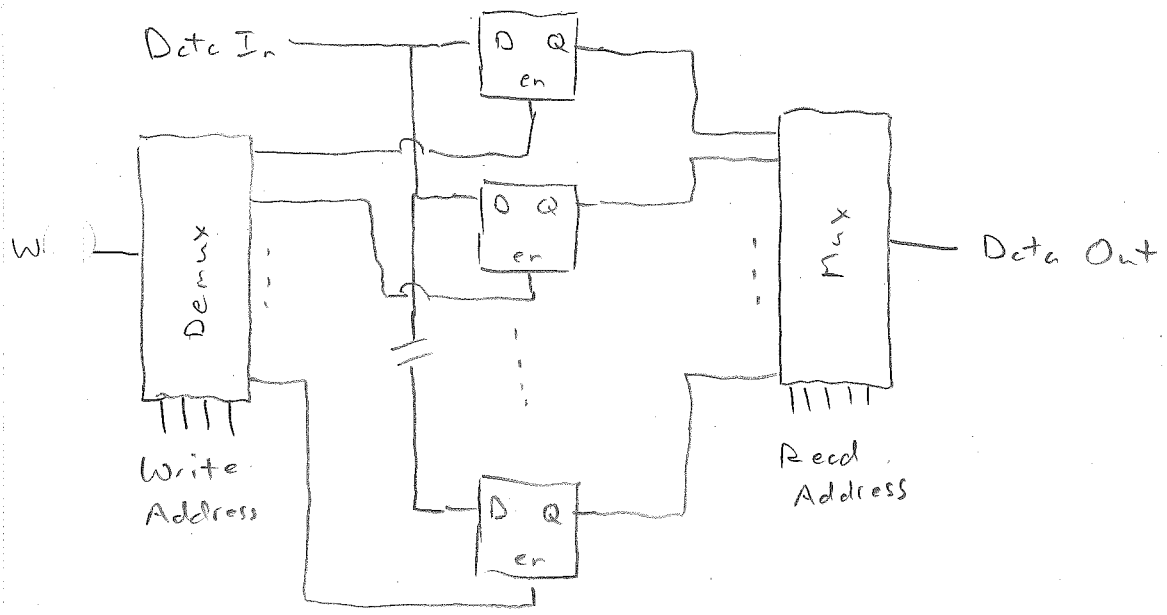
But impractical to use millions of in/out wires for MB of memory

Instead use RAM = addressable memory

Use multiplexers to decode address!

Start with array of flip flops → latches:

when en high, D → Q
else Q held constant



Data inputs all hooked together

Get enable signals from demultiplexer

One input → N outputs

Address controls which output active
(If "write" is low, no output active)

Data outputs → multiplexer

N inputs → one output

Address controls which input active

Can write to / read from any single memory bit at a time

For address with N bits, control 2^N latches

often data in & data out = several parallel lines

Read & Write to several bits at once ("word")

Then each latch → register

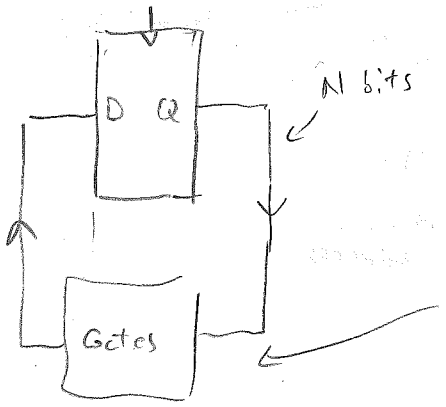
RAM works very well with state machines

Recall state machine = circuit that cycles through sequence of states

Each state = unique pattern of data bits

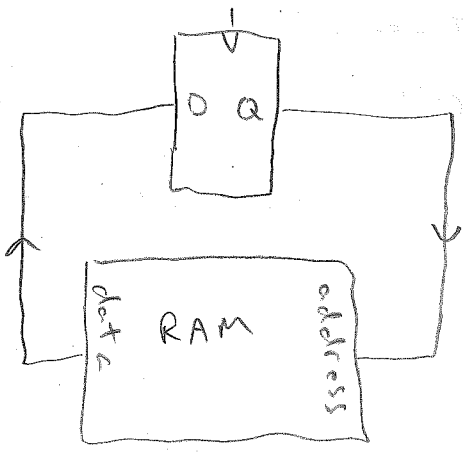
(≠ unique binary number)

Before, used logic gates to derive next state from current state



Can be hard to implement!

With RAM, can program sequence of states into memory:



At memory address = A, store data B

At address = B, store data C

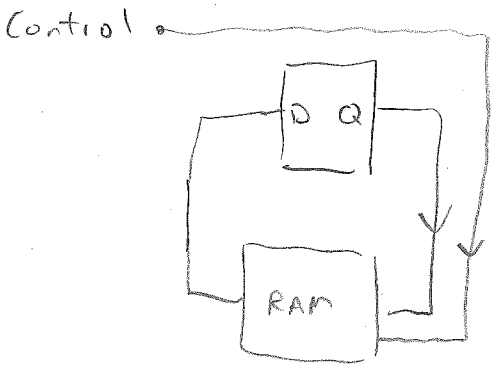
etc

As circuit runs, machine steps through stored sequence

Can generalize state machine a bit:

Let some data bits come from outside circuit

"control bits"



Allows outside control of sequence

Example: machine with 2 internal bits D_0, D_1 & one external D_2

Could use D_2 to select between count up & count down sequence

D_2	D_1	D_0	Q_1	Q_0	decimal $(Q_1 Q_0)$
0	0	0	0	1	1
0	0	1	1	0	2
0	1	0	1	1	3
0	1	1	0	0	0
1	0	0	1	1	3
1	1	1	1	0	2
1	1	0	0	1	1
1	0	1	0	0	0

So a single state machine can perform multiple functions

Many functions → microprocessor.

Might make sense that you can convert any flow chart to a state machine

⇒ any algorithm... general purpose computer