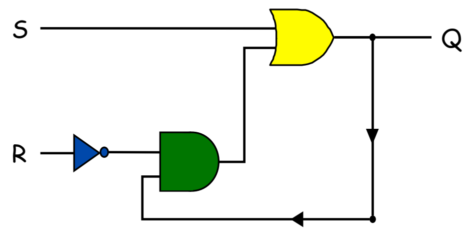
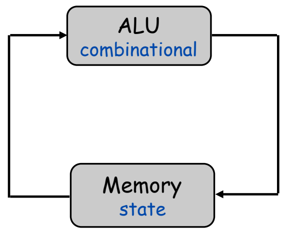
6.2: Sequential Circuits



Overview

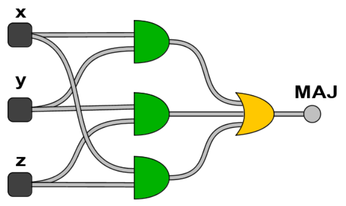
Last lecture: Boolean logic and combinational circuits.
- Basic abstraction = controlled switch.
- In principle, can build TOY computer with a combinational circuit.
  - $255 \times 16 = 4{,}080$ inputs $\Rightarrow$ 2^{4080} rows in truth table!
  - no simple pattern
  - each circuit element used at most once

This lecture: reuse circuit elements by storing bits in "memory."
Next lecture: glue components together to make TOY computer.

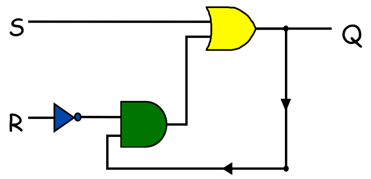


Sequential vs. Combinational Circuits

Combinational circuits.
- Output determined solely by inputs.
- Can draw solely with left-to-right signal paths.



Sequential circuits.
- Output determined by inputs AND previous outputs.
- Feedback loop.



Flip-Flop

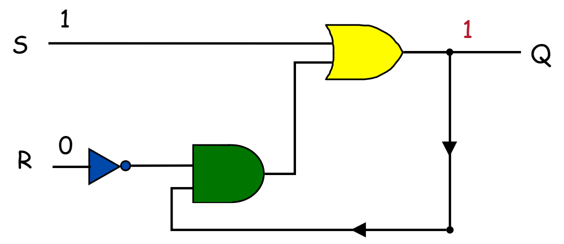
Flip-flop.
- A small and useful sequential circuit.
- Abstraction that "remembers" one bit.
- Basis of important computer components:
  - memory
  - counter

We will consider several flavors.

SR Flip-Flop

What is the value of Q if:

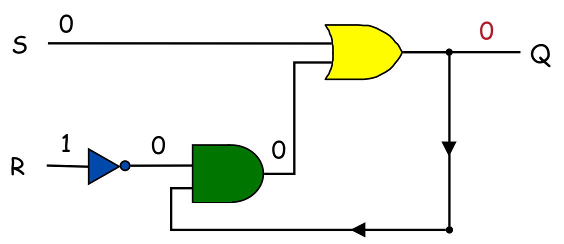
- S = 1 and R = 0 ? ⇒ Q is surely 1



SR Flip-Flop

What is the value of Q if:

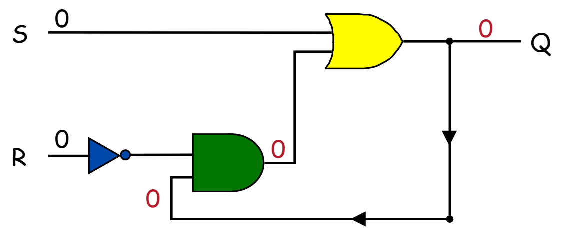
- S = 1 and R = 0 ? ⇒ Q is surely 1.
- S = 0 and R = 1 ? ⇒ Q is surely 0



SR Flip-Flop

What is the value of Q if:

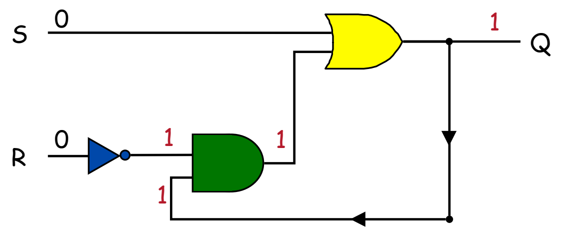
- S = 1 and R = 0 ? ⇒ Q is surely 1.
- S = 0 and R = 1 ? ⇒ Q is surely 0.
- S = 0 and R = 0 ? ⇒ Q is possibly 0



SR Flip-Flop

What is the value of Q if:

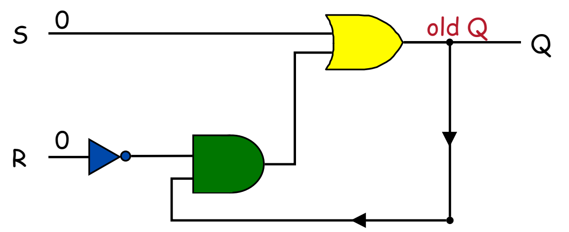
- S = 1 and R = 0 ? ⇒ Q is surely 1.
- S = 0 and R = 1 ? ⇒ Q is surely 0.
- S = 0 and R = 0 ? ⇒ Q is possibly 0 . . . or possibly 1 !



SR Flip-Flop

What is the value of Q if:

- S = 1 and R = 0 ? ⇒ Q is surely 1.
- S = 0 and R = 1 ? ⇒ Q is surely 0.
- S = 0 and R = 0 ? ⇒ Q is possibly 0 . . . or possibly 1.

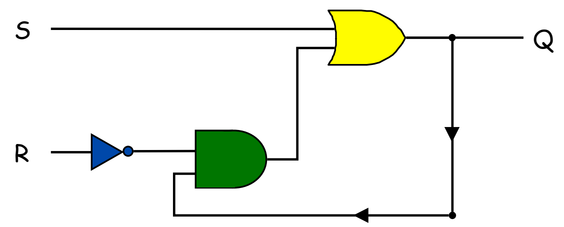


While S = R = 0 , Q *remembers* what it was the last time S or R was 1.

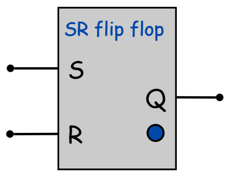
SR Flip-Flop

SR Flip-Flop.

- S = 1, R = 0 (set) ⇒ "Flips" bit on.
- S = 0, R = 1 (reset) ⇒ "Flops" bit off.
- S = R = 0 ⇒ Status quo.
- S = R = 1 ⇒ Not allowed.



Implementation



Interface

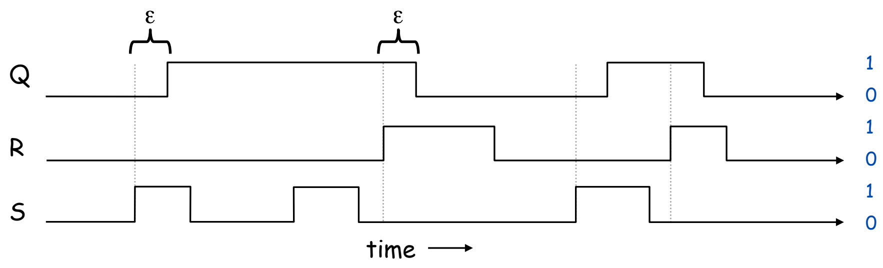
Truth Table and Timing Diagram

Truth table.

- Values vary over time.
- S(t), R(t), Q(t) denote value at time t.

SR Flip Flop Truth Table			
S(t)	R(t)	Q(t)	Q(t+ε)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	▨
1	1	1	▨

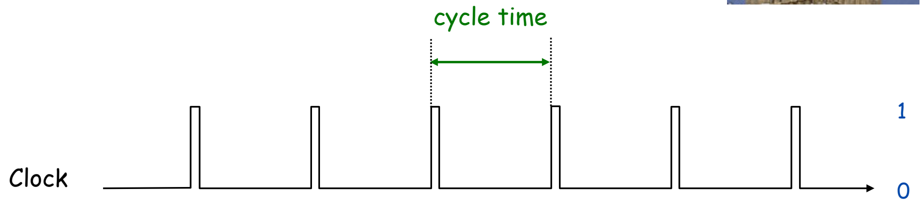
Sample timing diagram for SR flip-flop.



Clock

Clock.

- Fundamental abstraction.
  - regular on-off pulse
- External analog device.
- Synchronizes operations of different circuit elements.
- 1 GHz clock means 1 billion pulses per second.



How much does it Hert?

Frequency is inverse of cycle time.

- Expressed in *hertz.*
- Frequency of 1 Hz means that there is 1 cycle per second.
- Hence:
  - 1 kilohertz (kHz) means 1000 cycles/sec.
  - 1 megahertz (MHz) means 1 million cycles/sec.
  - 1 gigahertz (GHz) means 1 billion cycles/sec.
  - 1 terahertz (THz) means 1 trillion cycles/sec.

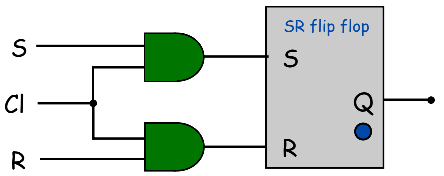
By the way, no such thing as 1 "hert" !

Heinrich Rudolf Hertz
(1857-1894)

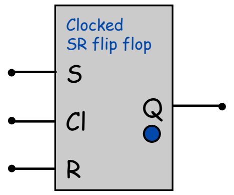
Clocked SR Flip-Flop

Clocked SR Flip-Flop.

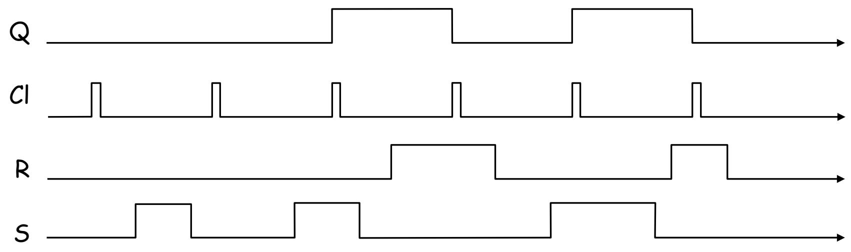
- Same as SR flip-flop except S and R only active when clock is 1.



Implementation



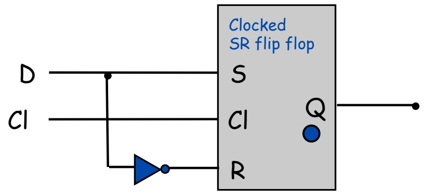
Interface



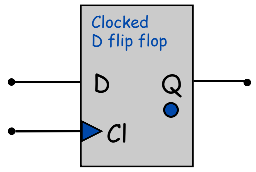
Clocked D Flip-Flop

Clocked D Flip-Flop.

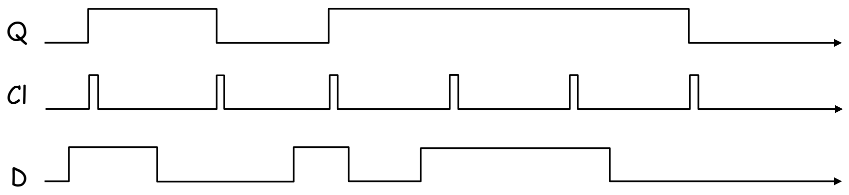
- Output follows D input while clock is 1.
- Output is remembered while clock is 0.



Implementation



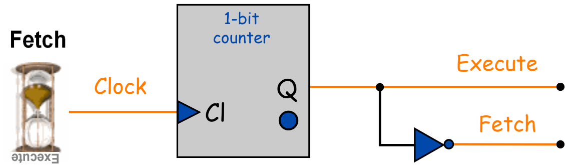
Interface

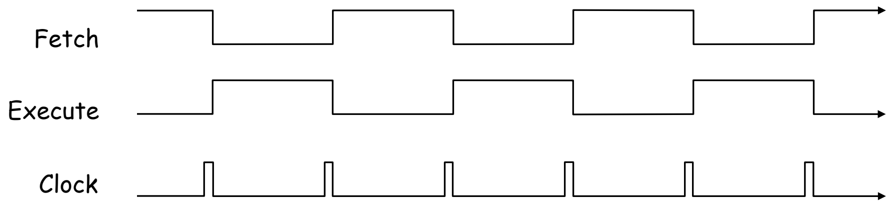


Fetch-Execute Cycle

Fetch-execute cycle for TOY.

- Need 1-bit counter.

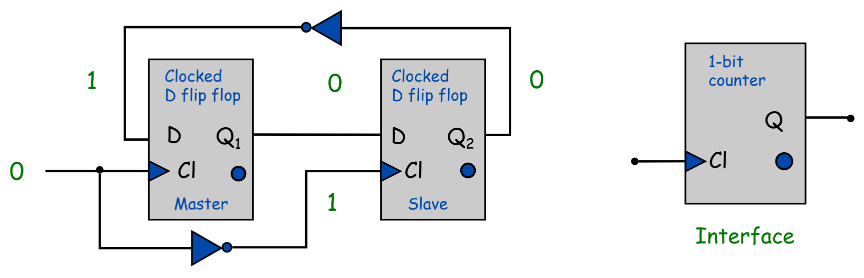


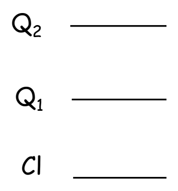


1-Bit Counter

1-bit counter.

- Circuit that oscillates between 1 and 0.

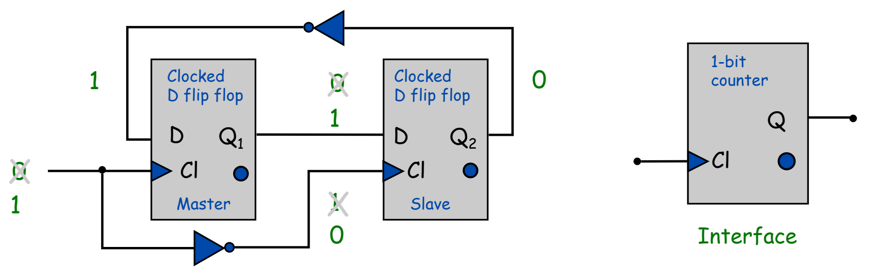


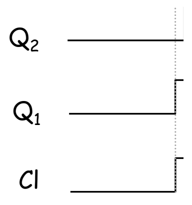


1-Bit Counter

1-bit counter.

- Circuit that oscillates between 1 and 0.

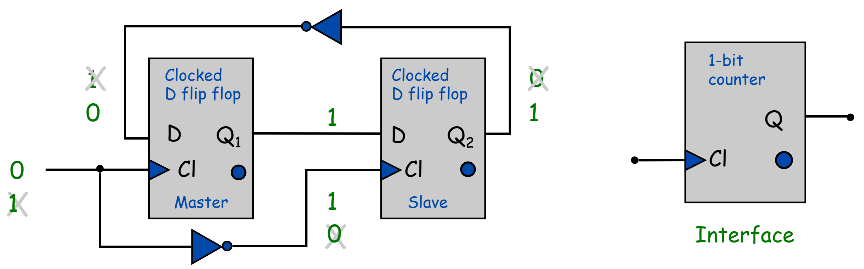


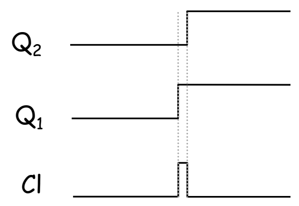


1-Bit Counter

1-bit counter.

- Circuit that oscillates between 1 and 0.

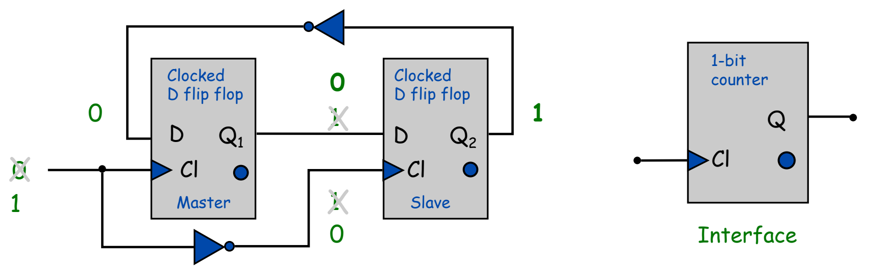


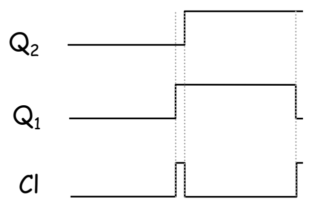


1-Bit Counter

1-bit counter.

- Circuit that oscillates between 1 and 0.

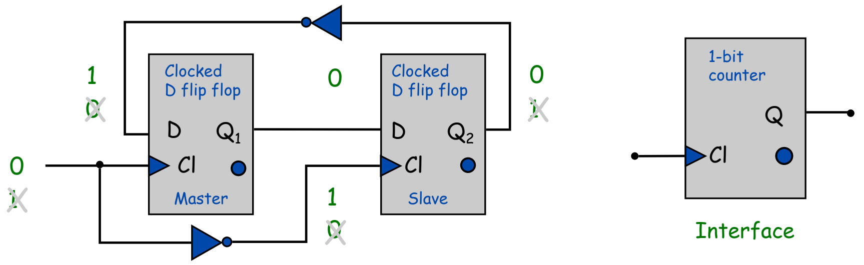


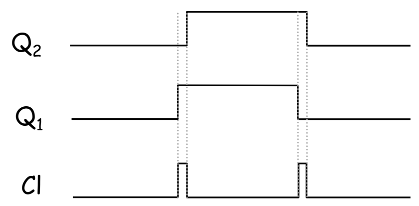


1-Bit Counter

1-bit counter.

- Circuit that oscillates between 1 and 0.

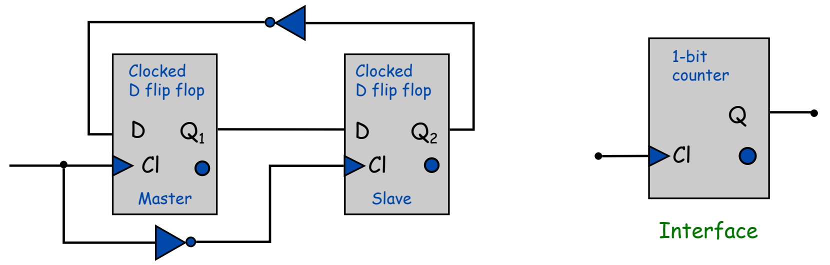


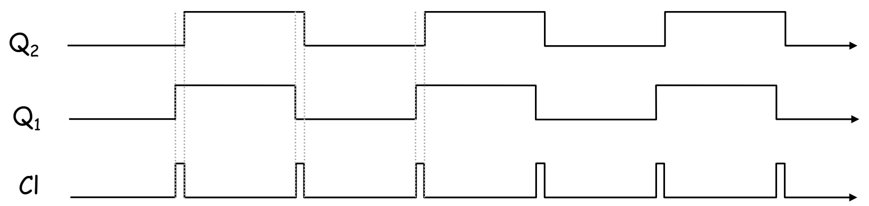


1-Bit Counter

1-bit counter.

- Circuit that oscillates between 1 and 0.

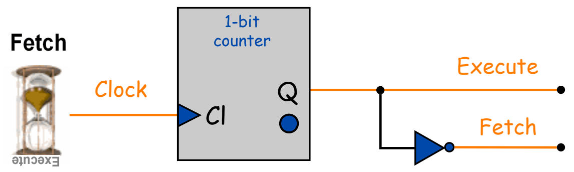


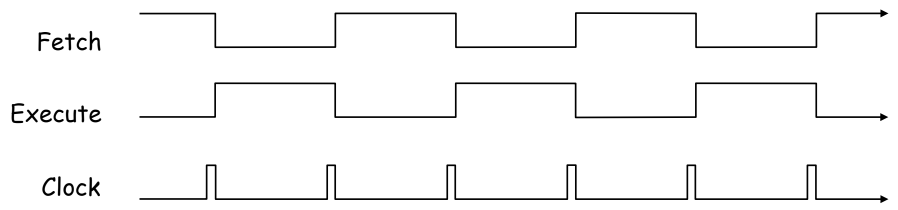


Fetch-Execute Cycle

Fetch-execute cycle for TOY.

- Need 1-bit counter.





Memory Overview

Computers and TOY have many types of memory.

- Program counter.
- Registers.
- Main memory.

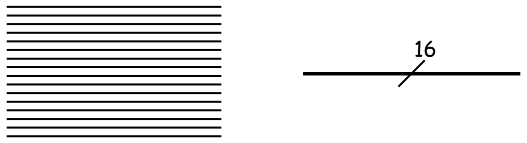
We implement each bit of memory with a clocked D flip-flop.

Need mechanism to organize and manipulate GROUPS of related bits.

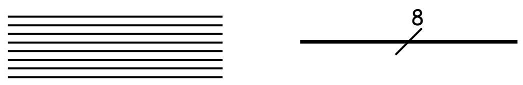
- TOY has 16-bit words.
- Memory hierarchy makes architecture manageable.

Bus

16-bit bus.
- Bundle of 16 wires.
- Memory transfer, register transfer.



8-bit bus.
- Bundle of 8 wires.
- TOY memory address.

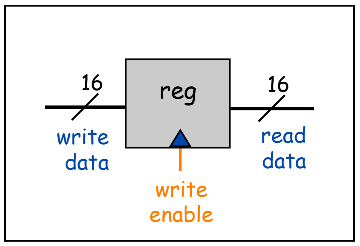


4-bit bus.
- Bundle of 4 wires.
- TOY register address.

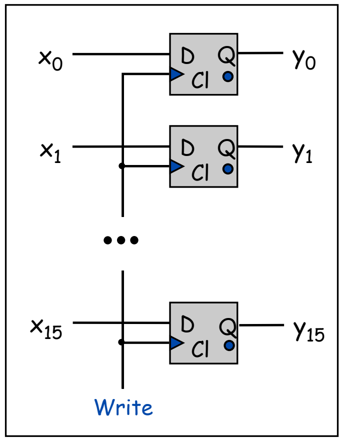
Stand-Alone Register

k-bit register.
- Stores k bits.
- Register contents always available on output.
- If write enable is asserted, k input bits get copied into register.

Ex: Program Counter, 16 TOY registers, 256 TOY memory locations.



16-bit Register Interface

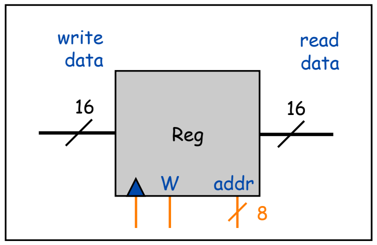


16-bit Register Implementation

Register File Interface

n-by-k register file.
- Bank of n registers; each stores k bits.
- Read and write information to *one* of n registers.
  - $\log_2 n$ address inputs specifies which one
- Addressed bits always appear on output.
- If write enable and clock are asserted, k input bits are copied into addressed register.

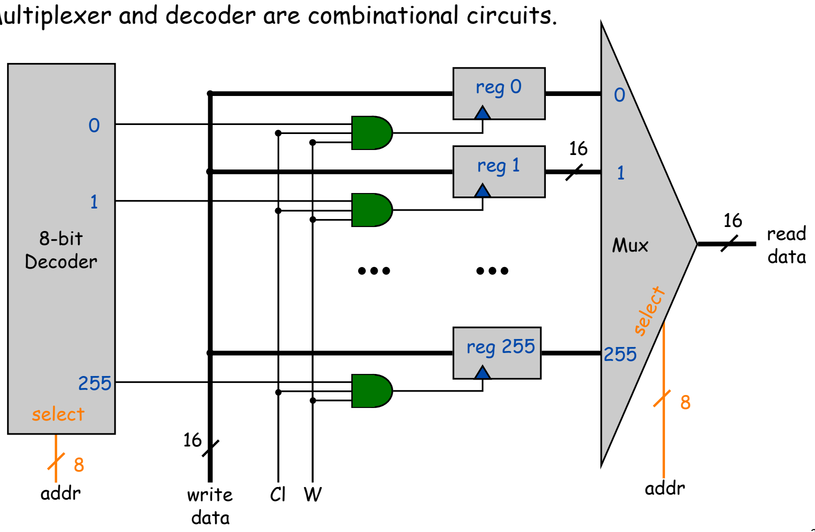
Examples.
- TOY registers: n = 16, k = 16.
- TOY main memory: n = 256, k = 16.
- Real computer: n = 256 million, k = 32.
  - 1 GB memory
  - (1 Byte = 8 bits)



256 x 16 Register File Interface

Register File Implementation

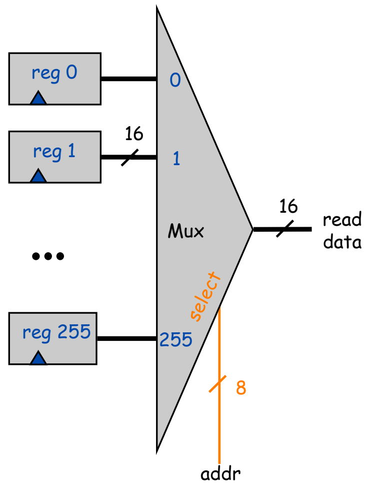
Implementation example: TOY main memory.
- Use 256 16-bit registers.
- Multiplexer and decoder are combinational circuits.



Register File Implementation: Reading

Implementation example: TOY main memory.

- Use 256 16-bit registers.
- Multiplexer is combinational circuit.

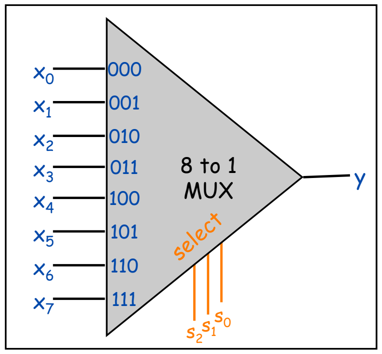


2^n-to-1 Multiplexer

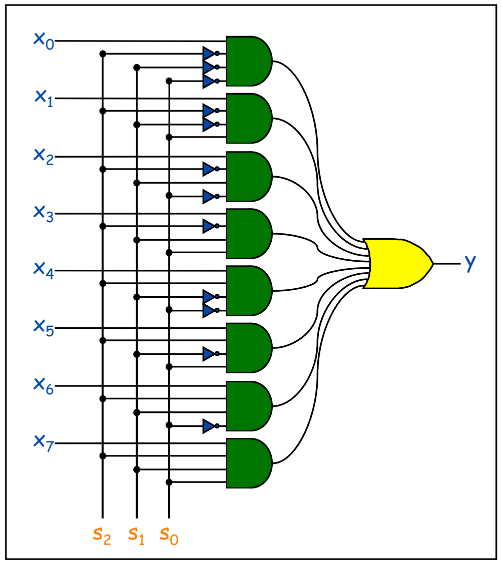
n = 8 for main memory

2^n-to-1 multiplexer.

- n select inputs, 2^n data inputs, 1 output.
- Copies "selected" data input bit to output.



8-to-1 Mux Interface



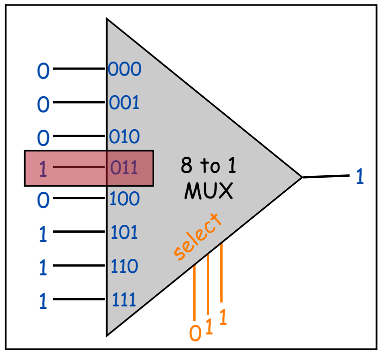
8-to-1 Mux Implementation

2^n-to-1 Multiplexer

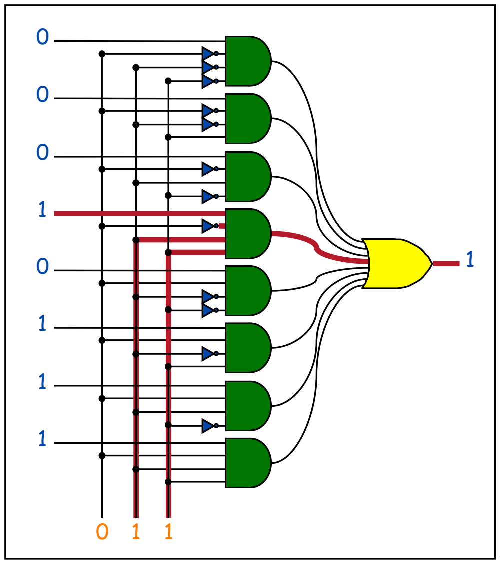
n = 8 for main memory

2^n-to-1 multiplexer.

- n select inputs, 2^n data inputs, 1 output.
- Copies "selected" data input bit to output.



8-to-1 Mux Interface



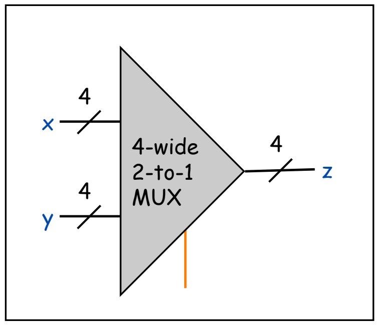
8-to-1 Mux Implementation

2^n-to-1 Multiplexer, Width = k

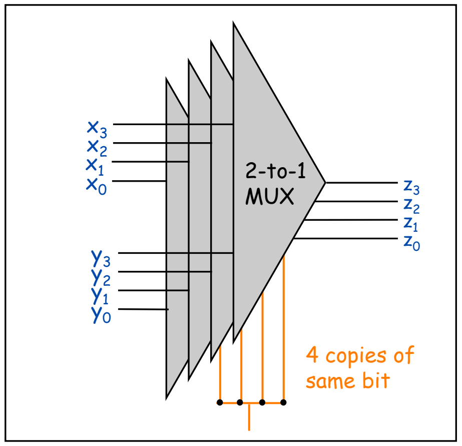
n = 8, k = 16 for main memory

2^n-to-1 multiplexer, width = k.

- Select from one of 2^n k-bit buses.
- Copies k "selected" data bits to output.
- Layering k 2^n-to-1 multiplexers.



Interface for 2-to-1 MUX, width = 4

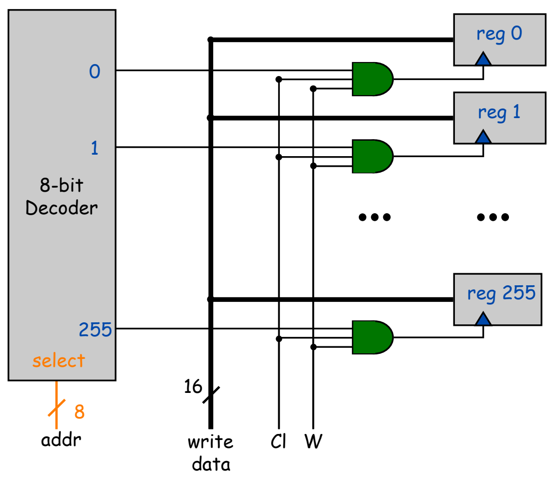


Implementation for 2-to-1 MUX, width = 4

Register File Implementation: Writing

Implementation example: TOY main memory.

- Use 256 16-bit registers.
- Decoder is combinational circuit.

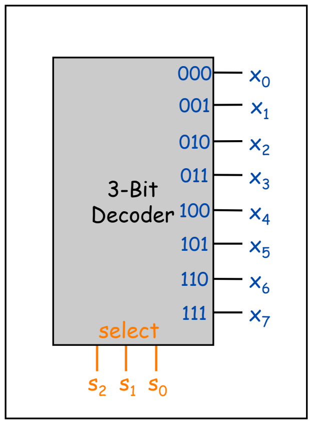


n-Bit Decoder

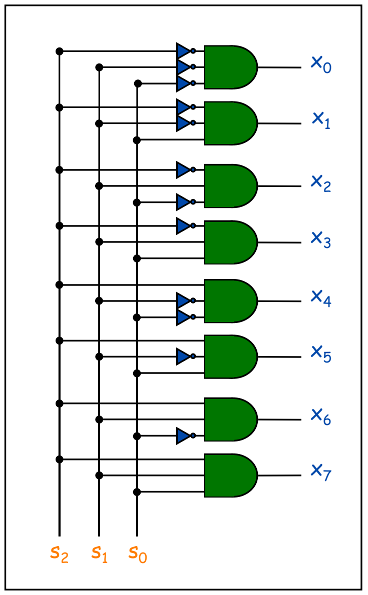
n = 8 for main memory

n-bit decoder.

- n address inputs, 2^n data outputs.
- Addressed output bit is 1; others are 0.



3-Bit Decoder Interface



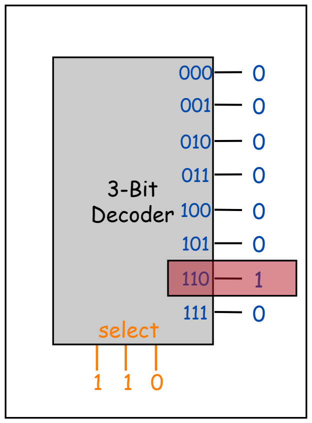
3-Bit Decoder Implementation

n-Bit Decoder

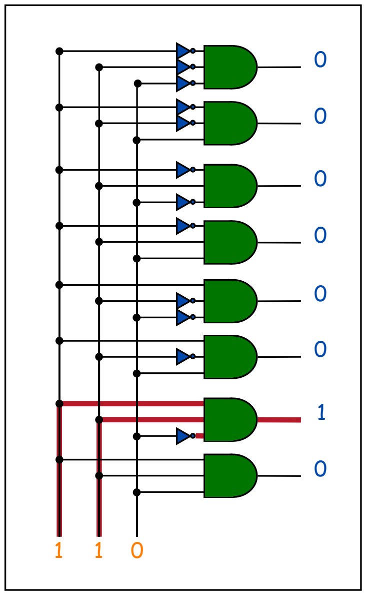
n = 8 for main memory

n-bit decoder.

- n address inputs, 2^n data outputs.
- Addressed output bit is 1; others are 0.



3-Bit Decoder Interface

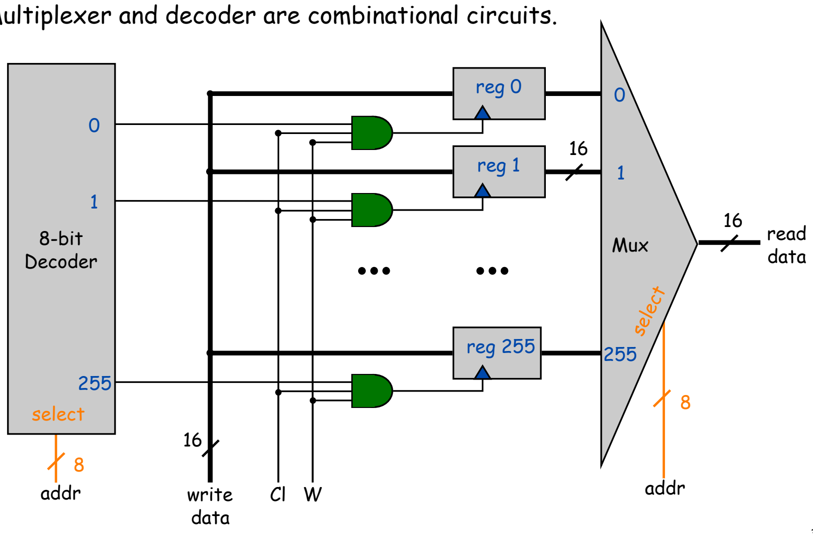


3-Bit Decoder Implementation

Register File Implementation: Reading and Writing

Implementation example: TOY main memory.

- Use 256 16-bit registers.
- Multiplexer and decoder are combinational circuits.



Summary

Sequential circuits add "state" to digital hardware.

▪ Flip-flop.	Represents 1 bit.
▪ TOY register.	16 D flip-flops.
▪ TOY main memory.	256 registers.

Actual technologies for register file and memory are different.

- Register files are relatively small and very fast.
  - expensive per bit
- Memories are relatively large and pretty fast.
  - amazingly cheap per bit
- Drastic evolution of technology over time

Next time: we build a complete TOY computer.