Sequential Logic

Introduction to Computer

Yung-Yu Chuang

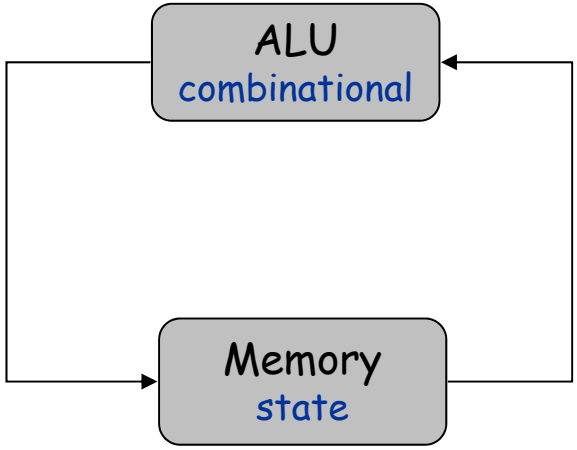
with slides by Sedgewick & Wayne (introcs.cs.princeton.edu), Nisan & Schocken (www.nand2tetris.org) and Harris & Harris (DDCA)

Review of Combinational Circuits

Combinational circuits.

- Basic abstraction = 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

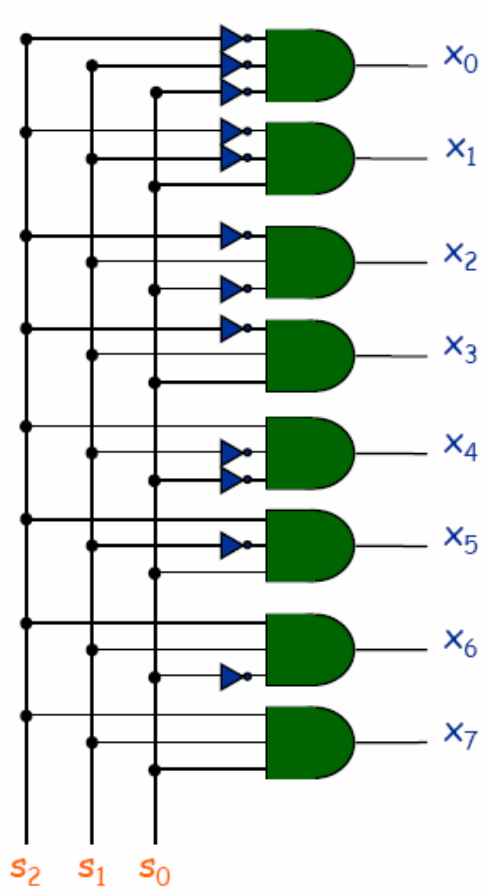
Sequential circuits. Reuse circuit elements by storing bits in "memory."



Combinational vs. Sequential Circuits

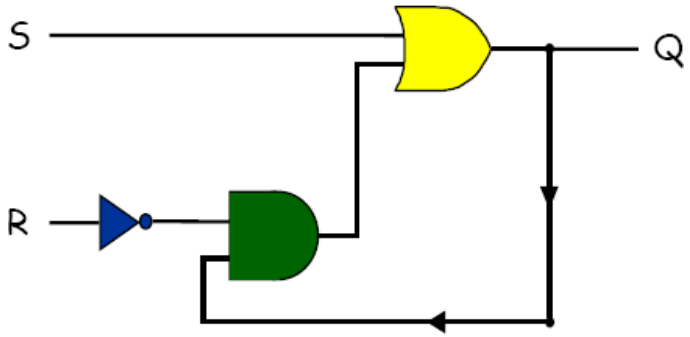
Combinational circuits.

- Output determined solely by inputs.
- Can draw with no loops.
- Ex: majority, adder, ALU.



Sequential circuits.

- Output determined by inputs and previous outputs.
- Ex: memory, program counter, CPU.



Flip-Flop

Flip-flop

- A small and useful sequential circuit
- Abstraction that remembers one bit
- Basis of important computer components for
  - register
  - memory
  - counter
- There are several flavors

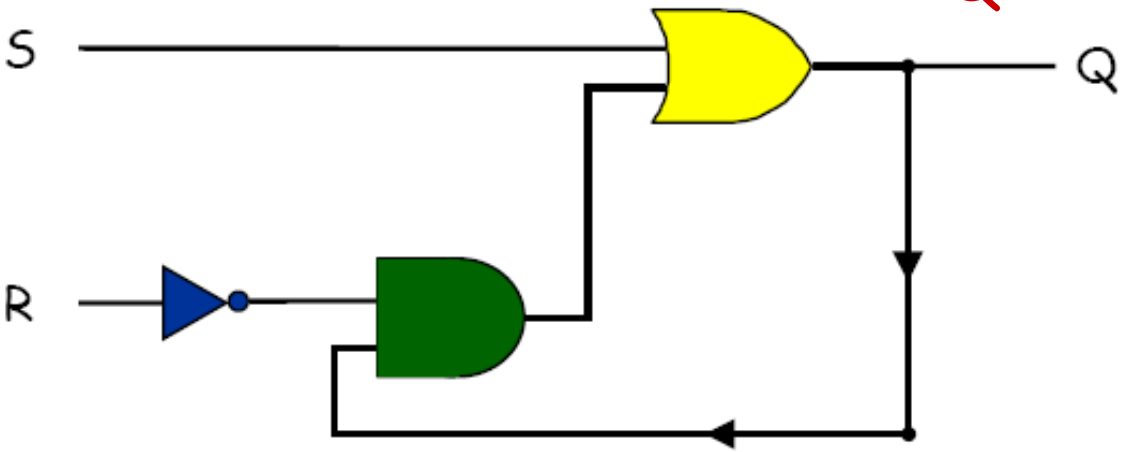
S-R flip flop

SR Flip-Flop.

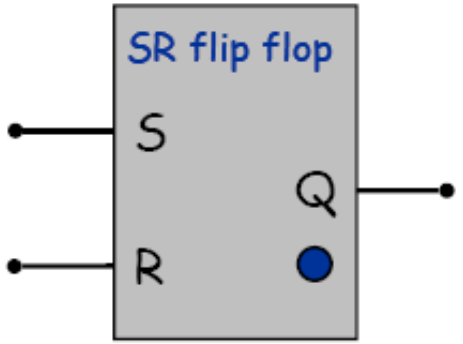
- S = 1, R = 0 (set) ⇒ Flips "bit" on.
- S = 0, R = 1 (reset) ⇒ Flops "bit" off.
- S = R = 0 ⇒ Status quo.
- S = R = 1 ⇒ Not allowed.

R	S	Q
0	0	
0	1	
1	0	
1	1	

$Q=S+\bar{R}Q$



Implementation

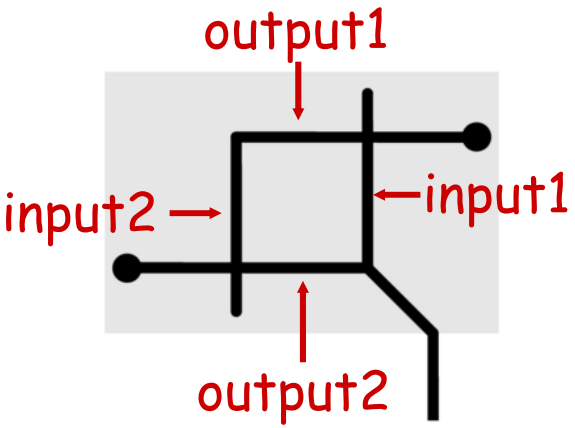


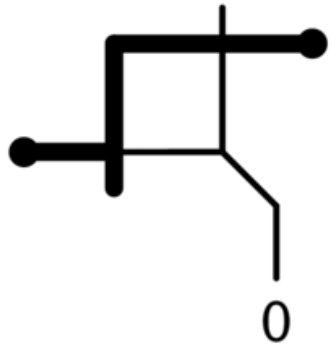
Interface

Relay-based flip-flop

Ex. Simplest feedback loop.

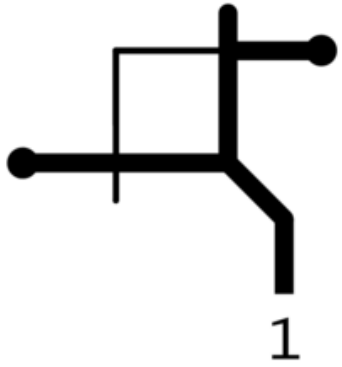
- Two relays A and B, both connected to power, each blocked by the other.
- State determined by whichever switches first. The state is latched.
- Stable.





0 state

0

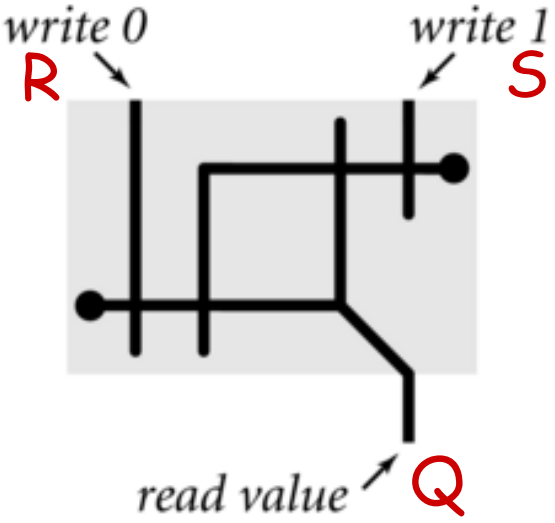


1 state

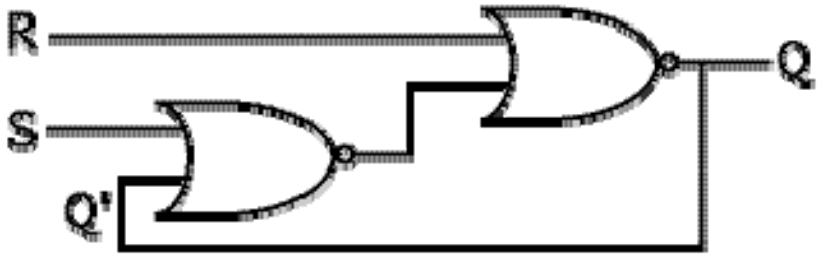
1

SR Flip Flop

SR flip flop. Two cross-coupled NOR gates.



$Q=\overline{R}(S+Q)$

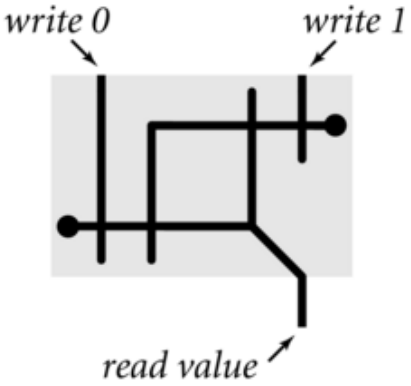


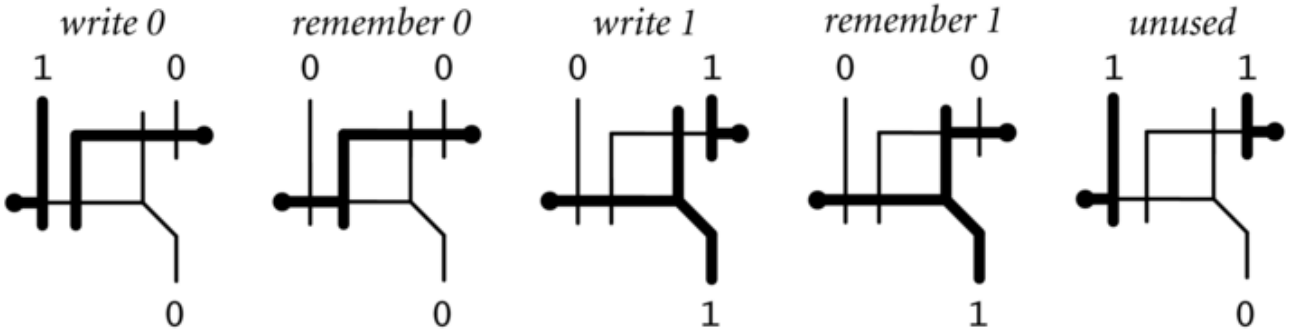
R	S	Q
0	0	
0	1	
1	0	
1	1	

Flip-Flop

Flip-flop.

- A way to control the feedback loop.
- Abstraction that "remembers" one bit.
- Basic building block for memory and registers.





Caveat. Need to deal with switching delay.

Truth Table and Timing Diagram

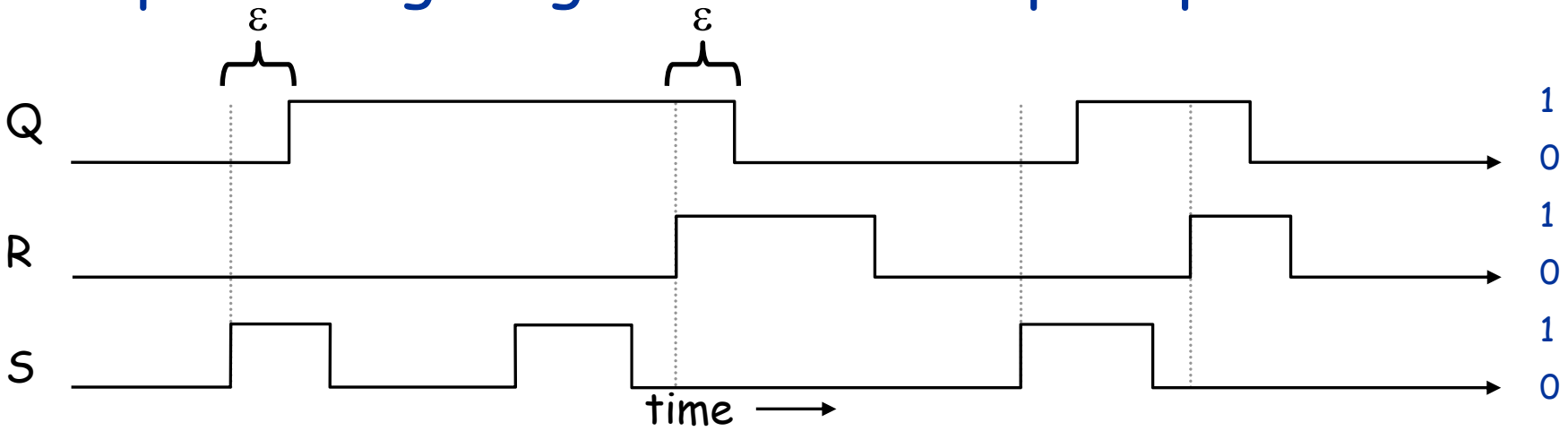
Truth table.

- 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	

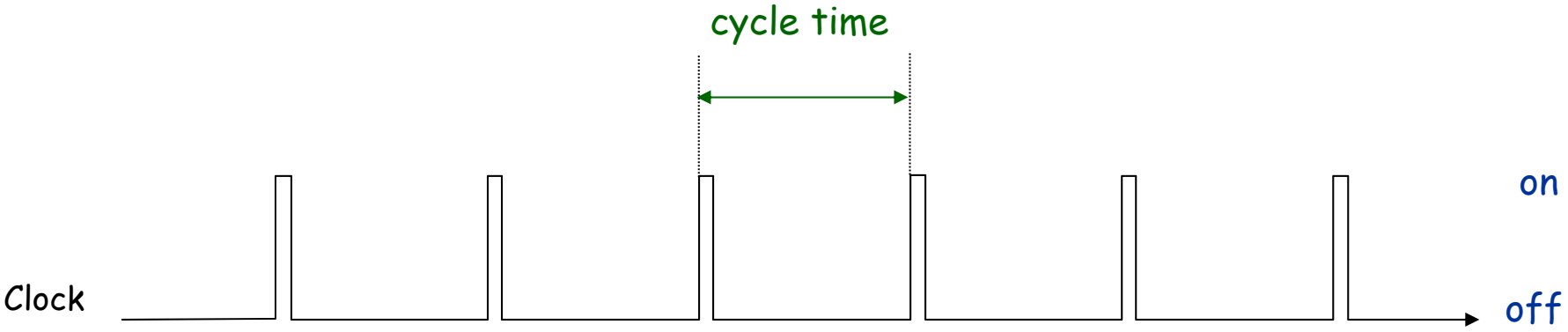
Sample timing diagram for SR flip-flop.



Clock

Clock.

- Fundamental abstraction: regular on-off pulse.
  - on: fetch phase
  - off: execute phase
- External analog device.
- Synchronizes operations of different circuit elements.
- Requirement: clock cycle longer than max switching time.



How much does it Hert?

Frequency is inverse of cycle time.

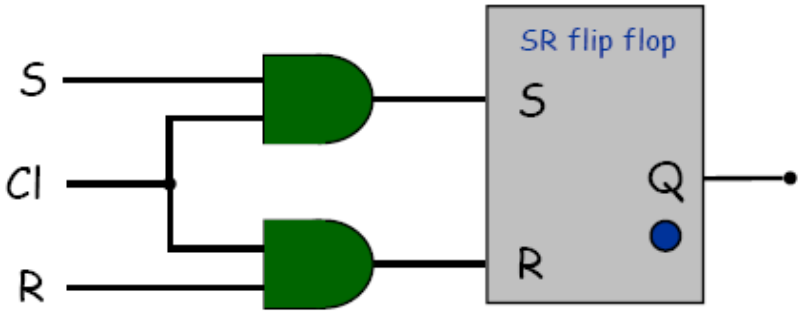
- Expressed in hertz.
- Frequency of 1 Hz means that there is 1 cycle per second.
  - 1 kilohertz (kHz) means 1000 cycles/sec.
  - 1 megahertz (MHz) means 1 million cycles/sec.
  - 1 gigahertz (GHz) means 1 billion cycles/sec.
  - 1 terahertz (THz) means 1 trillion cycles/sec.

Heinrich Rudolf Hertz
(1857-1894)

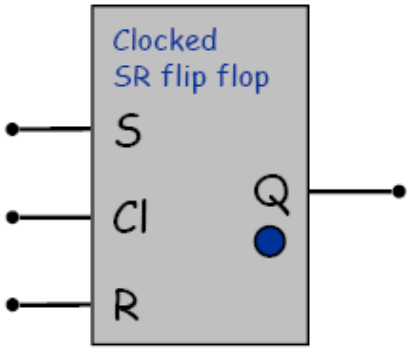
Clocked S-R flip-flop

Clocked SR Flip-Flop.

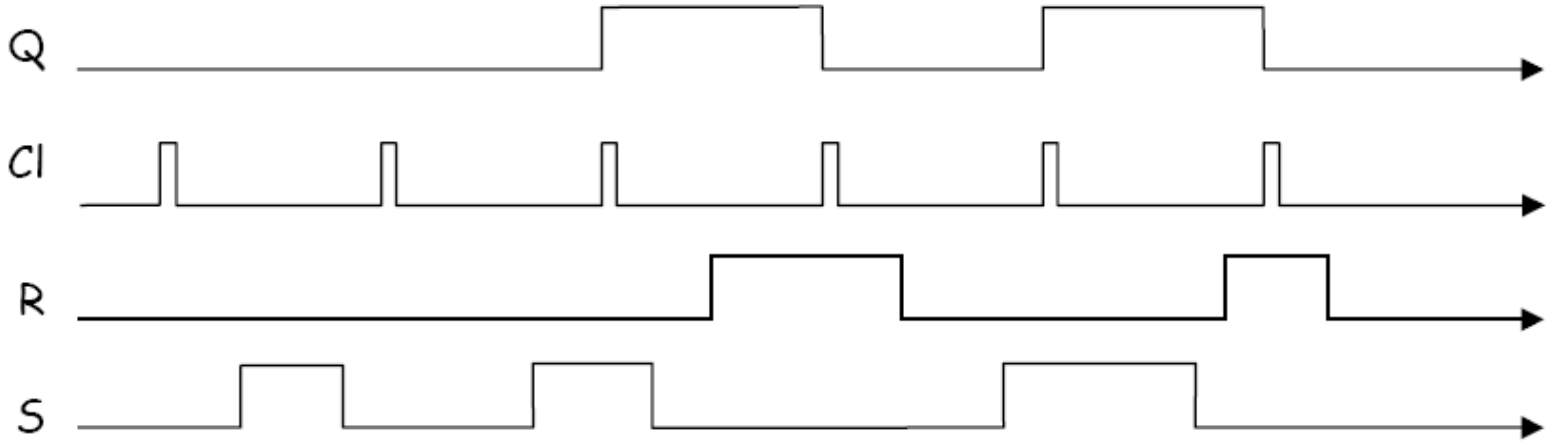
- Same as SR flip-flop except S and R only active when clock is 1.



Implementation



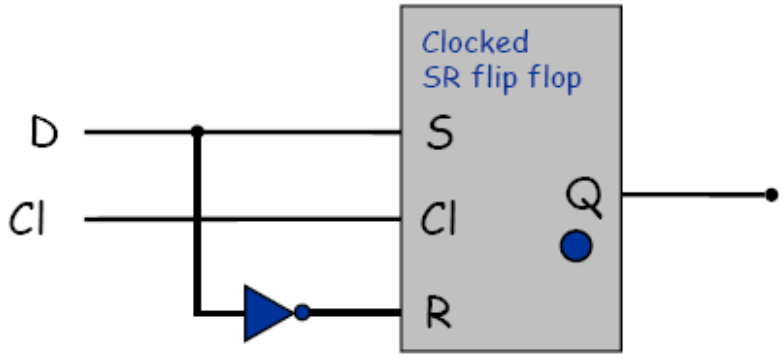
Interface



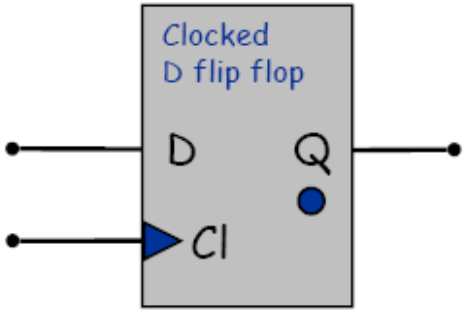
Clocked D flip-flop

Clocked D Flip-Flop.

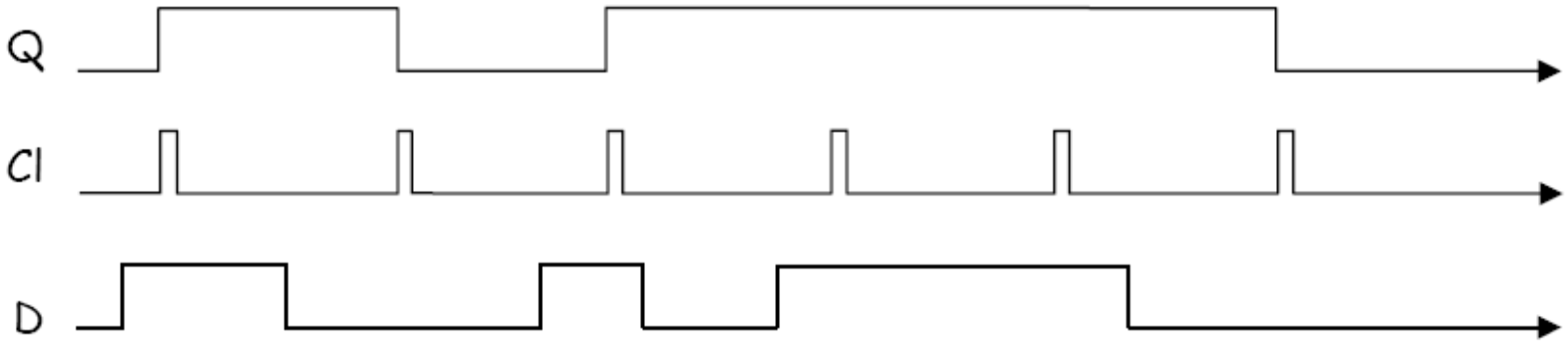
- Output follows D input while clock is 1.
- Output is remembered while clock is 0.



Implementation



Interface

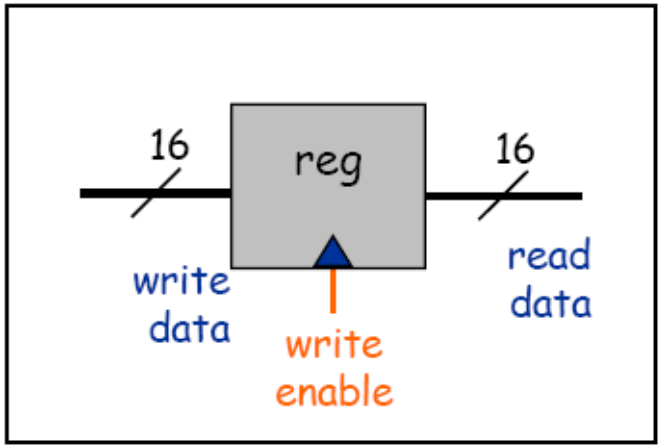


Stand-Alone Register

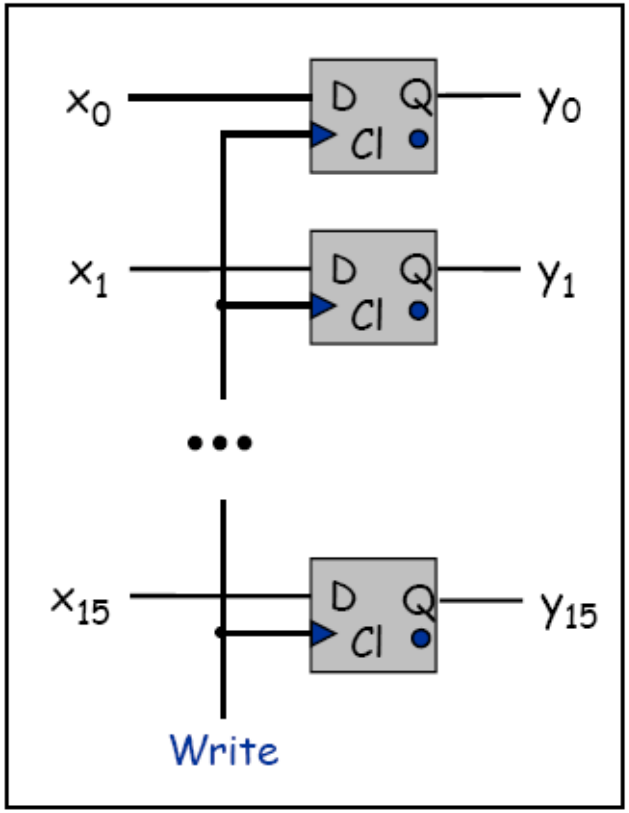
k-bit register.

- 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

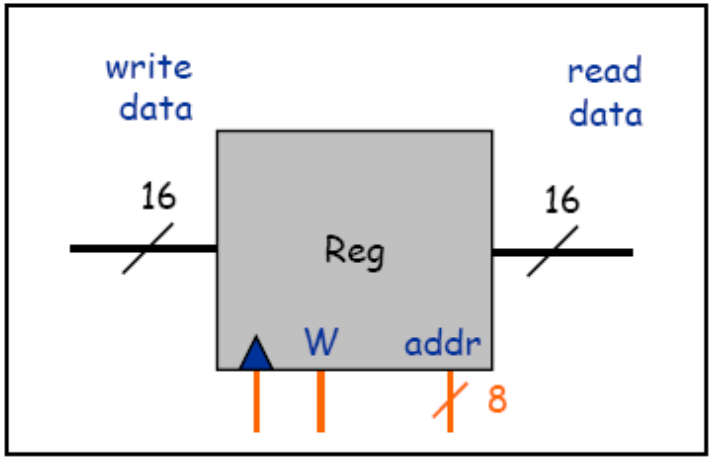
Register file interface

n x 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

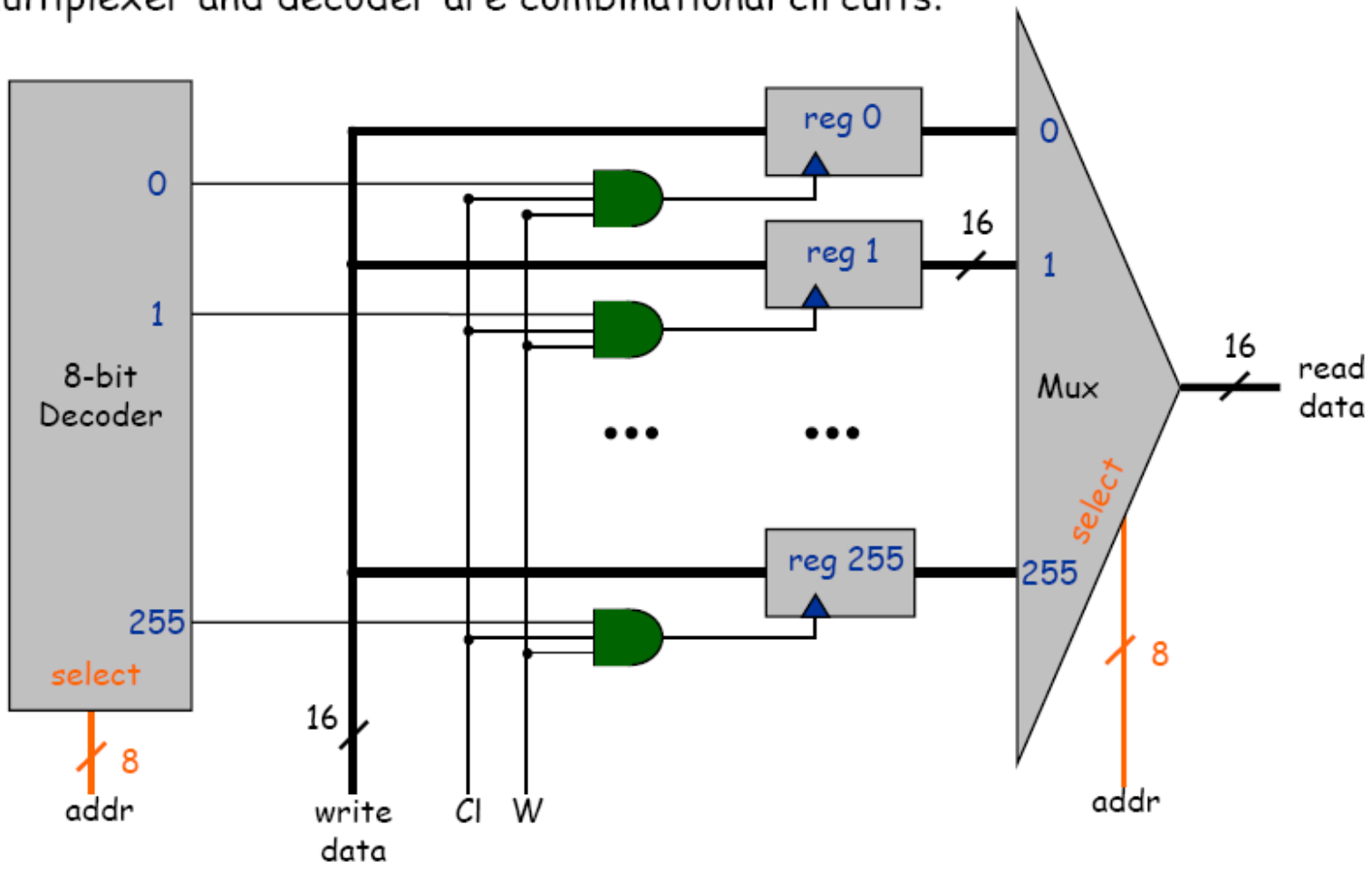


256 x 16 Register File Interface

Register file implementation

Implementation example: TOY main memory.

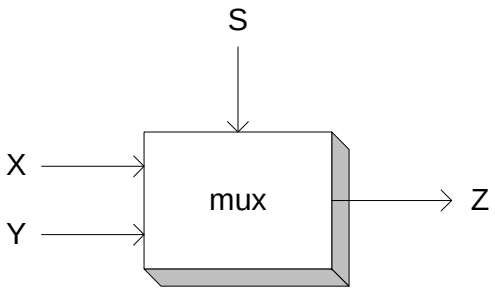
- Use 256 16-bit registers.
- Multiplexer and decoder are combinational circuits.



Multiplexer

When s=0, return x; otherwise, return y.

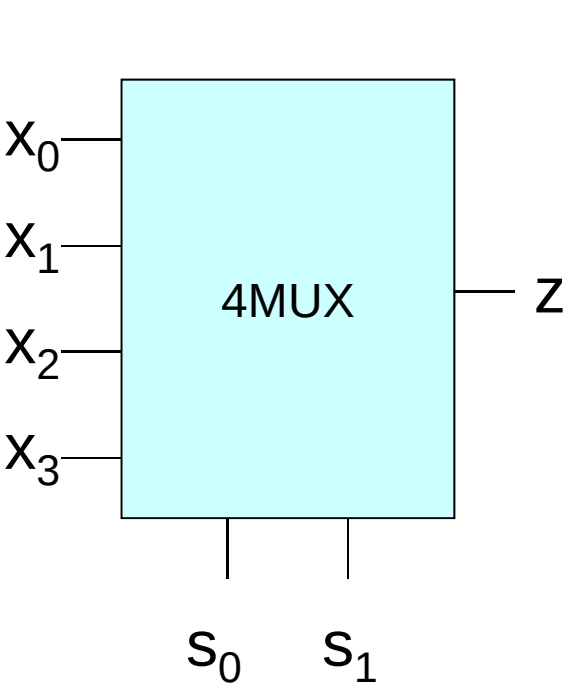
Example: $(Y \wedge S) \vee (X \wedge \neg S)$

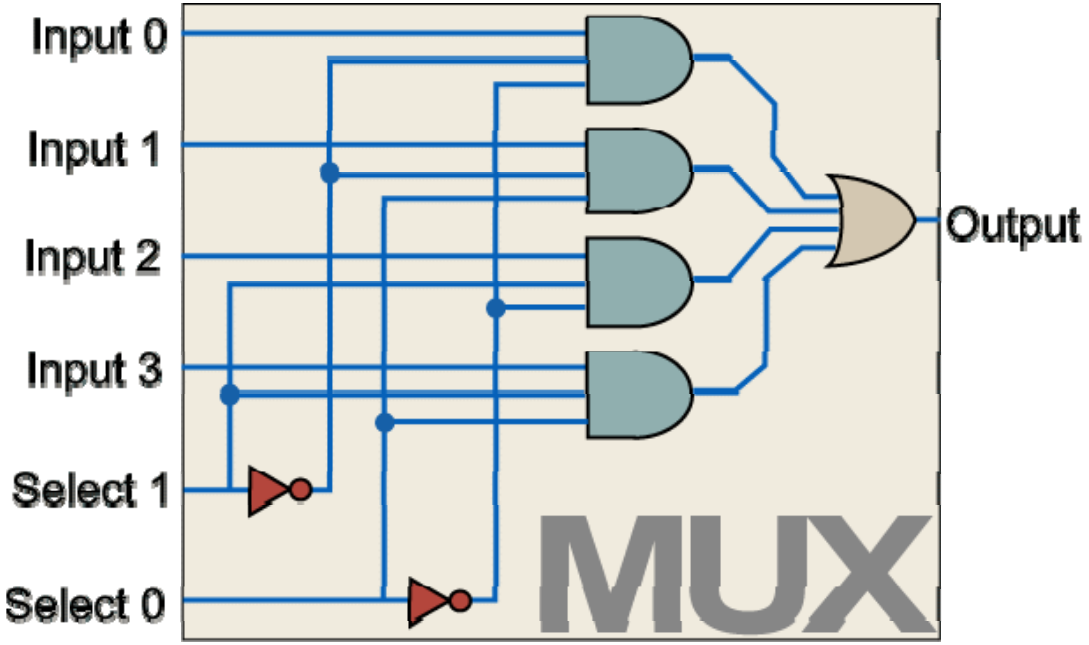


Two-input multiplexer

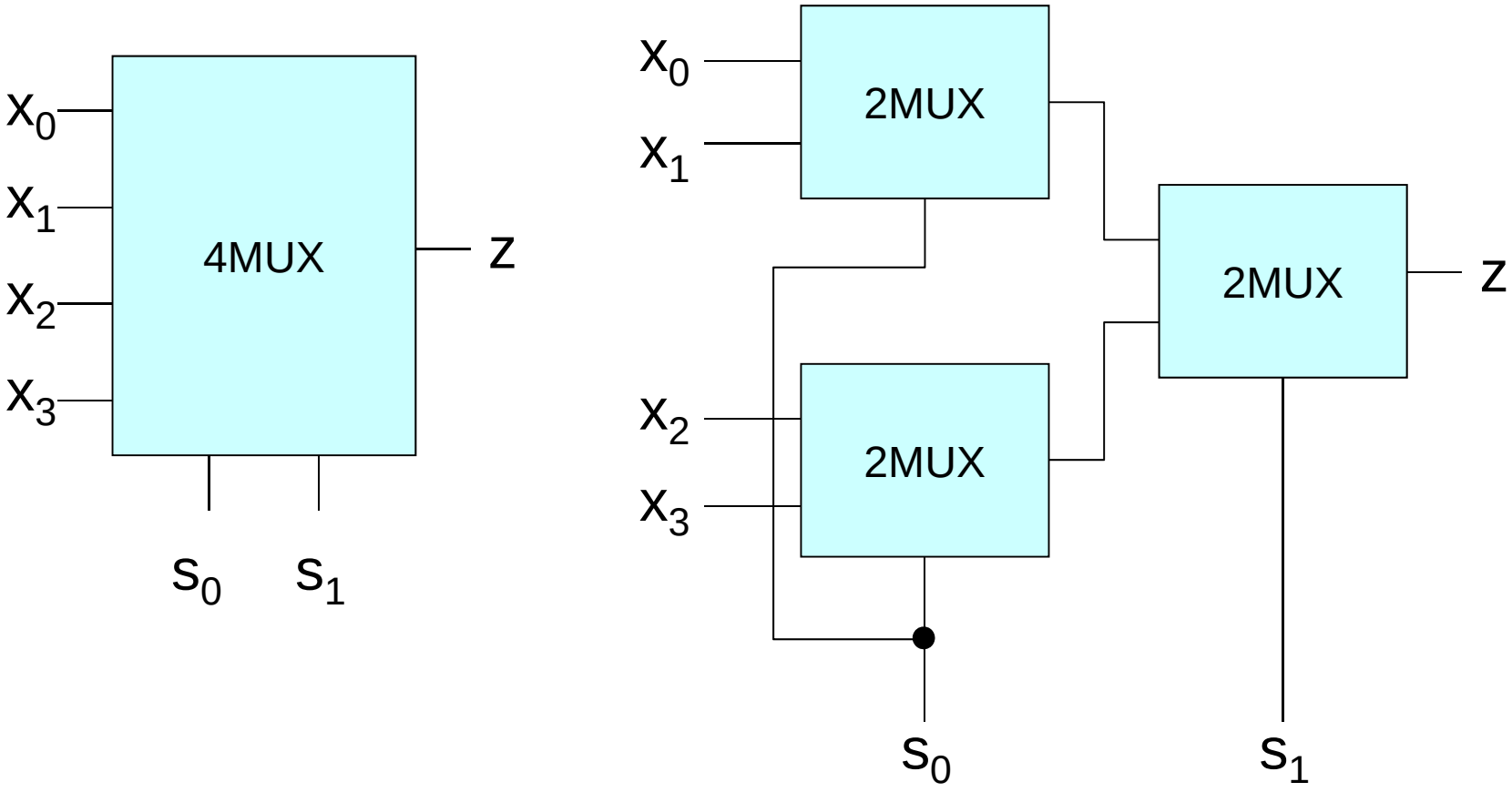
X	Y	S	$Y \wedge S$	$\neg S$	$X \wedge \neg S$	$(Y \wedge S) \vee (X \wedge \neg S)$
F	F	F	F	T	F	F
F	T	F	F	T	F	F
T	F	F	F	T	T	T
T	T	F	F	T	T	T
F	F	T	F	F	F	F
F	T	T	T	F	F	T
T	F	T	F	F	F	F
T	T	T	T	F	F	T

4-to-1 multiplexer





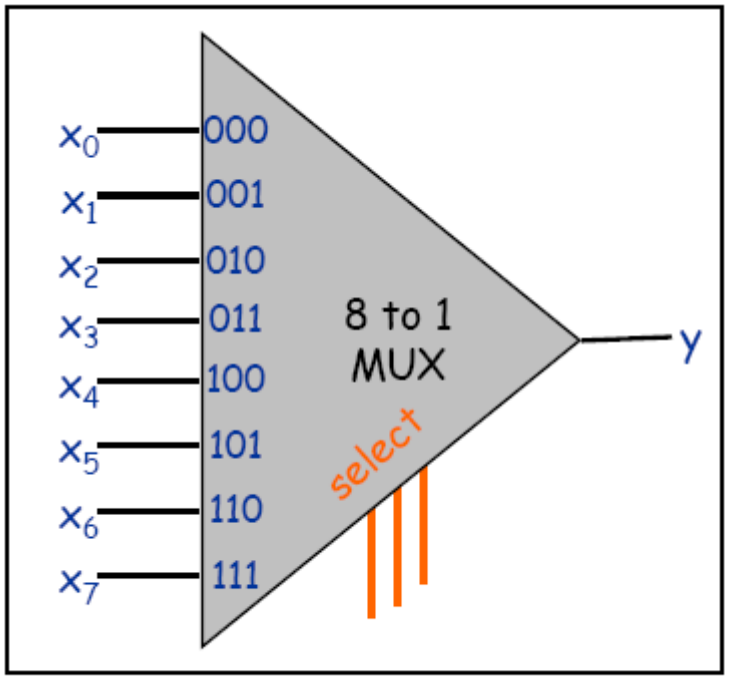
4-to-1 multiplexer



8-to-1 Multiplexer

2^N-to-1 multiplexer

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

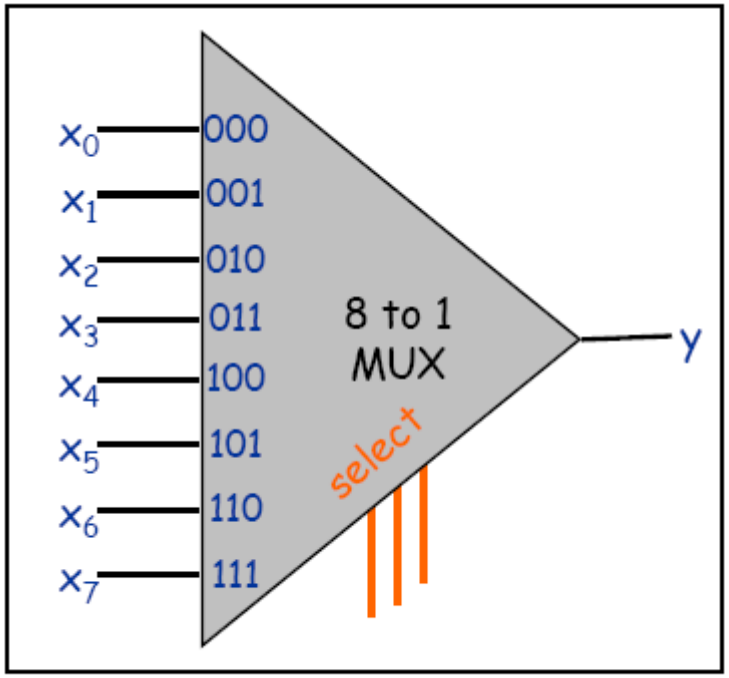


8-to-1 Mux Interface

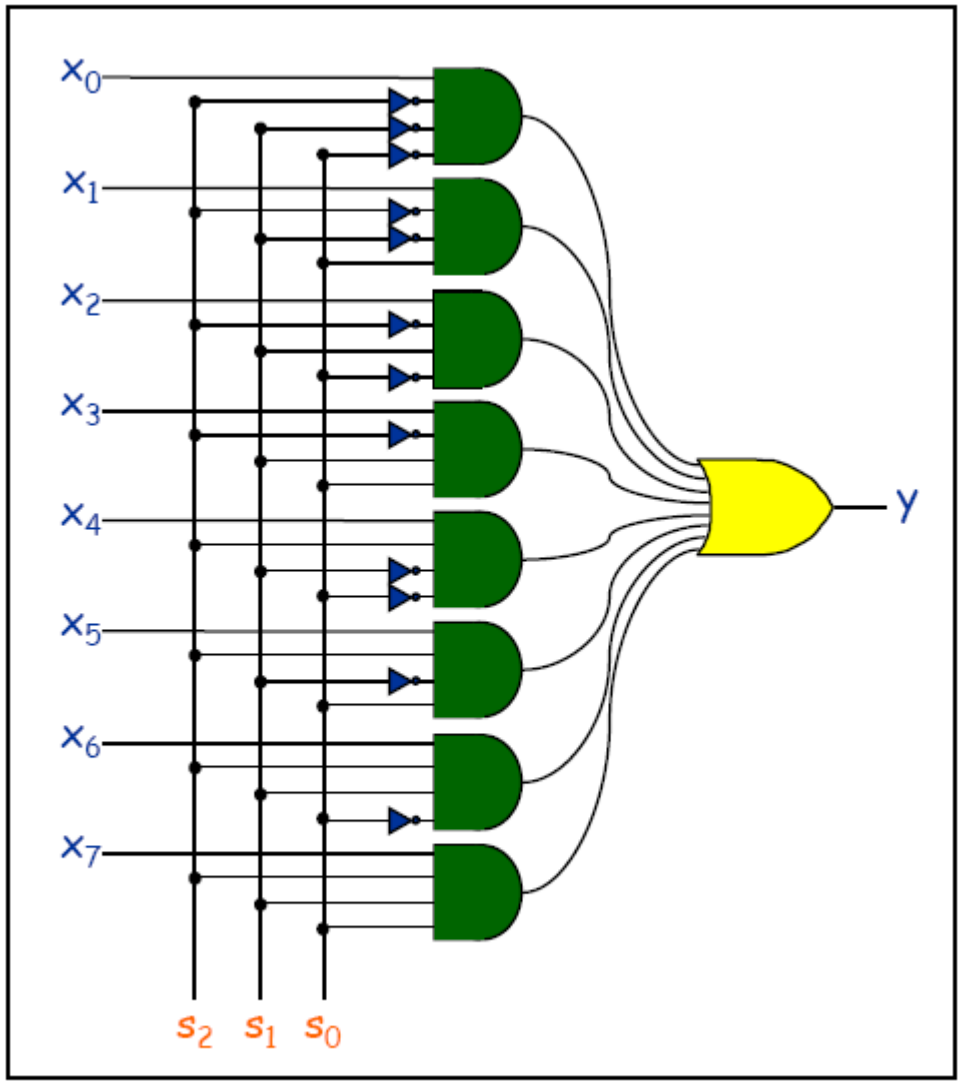
8-to-1 Multiplexer

2^N-to-1 multiplexer

- N select inputs, 2^N da
  inputs, 1 output
- Copies "selected" datc
  bit to output



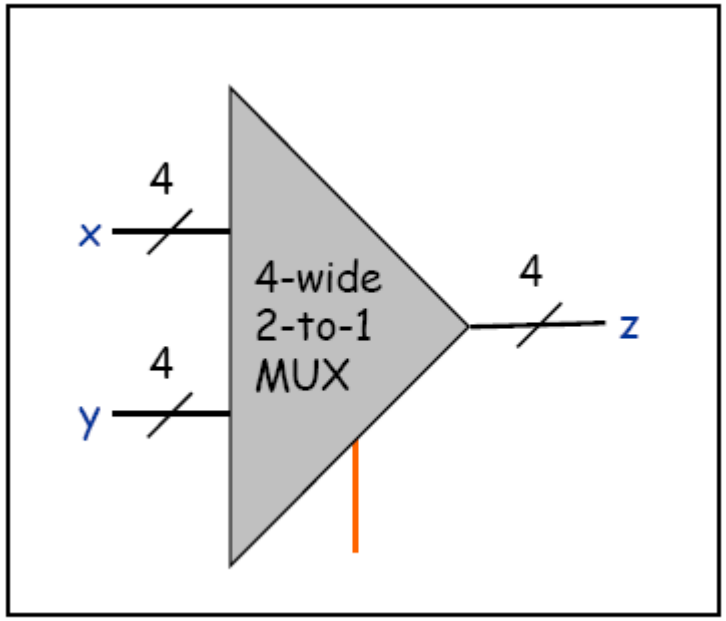
8-to-1 Mux Interface



8-to-1 Mux Implementation

4-Wide 2-to-1 Multiplexer

Goal: select from one of two 4-bit buses

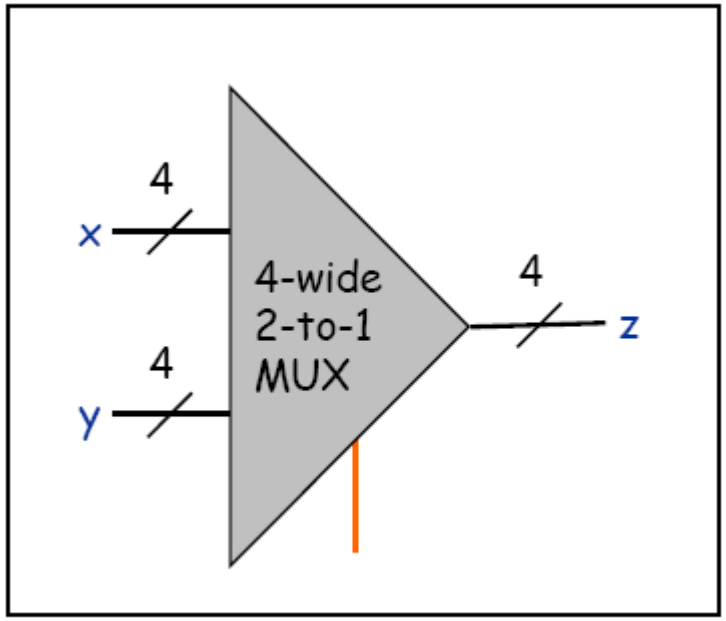


Interface

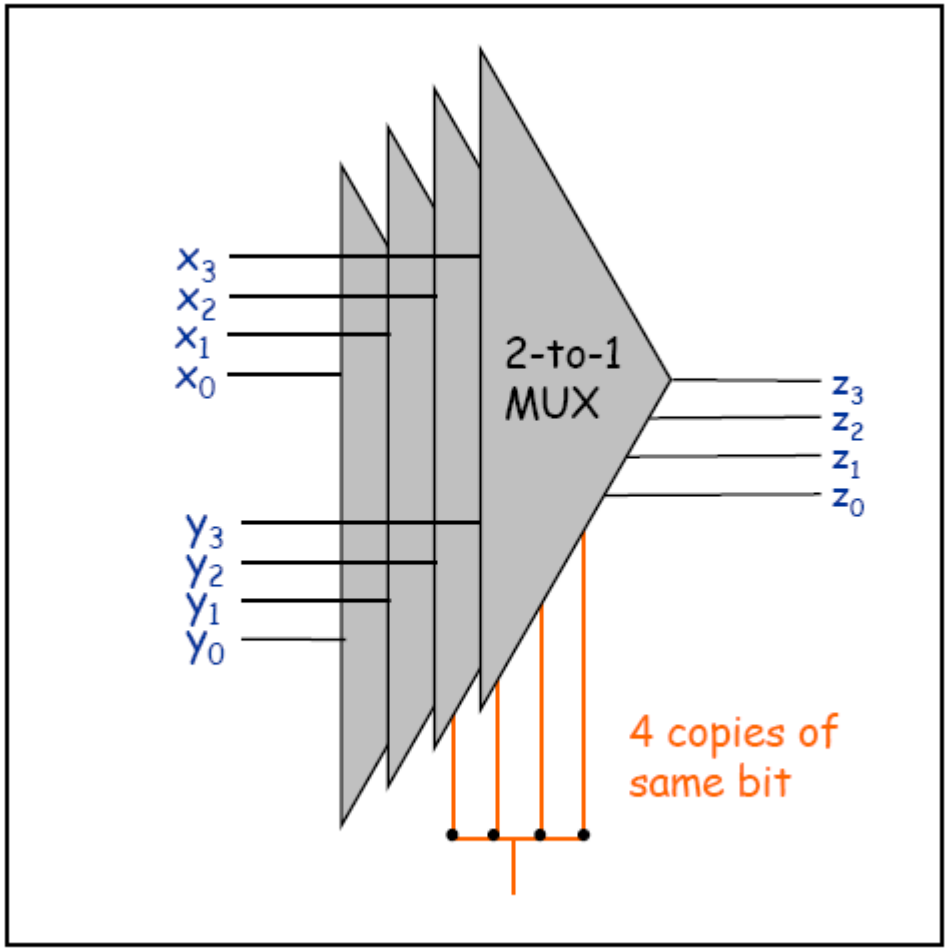
4-Wide 2-to-1 Multiplexer

Goal: select from one of two 4-bit buses

- Implemented by layering 4 2-to-1 multiplexer



Interface

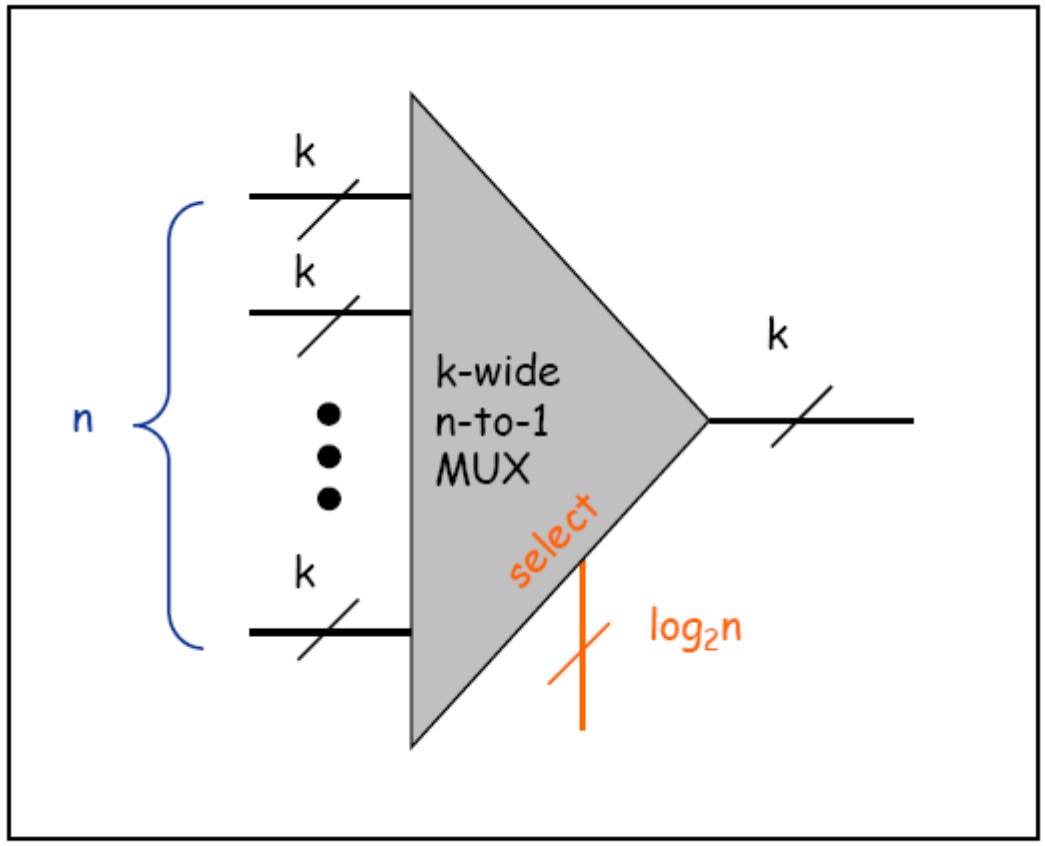


Implementation

k-Wide n-to-1 Multiplexer

Goal: select from one of n k-bit buses

- Implemented by layering k n-to-1 multiplexer

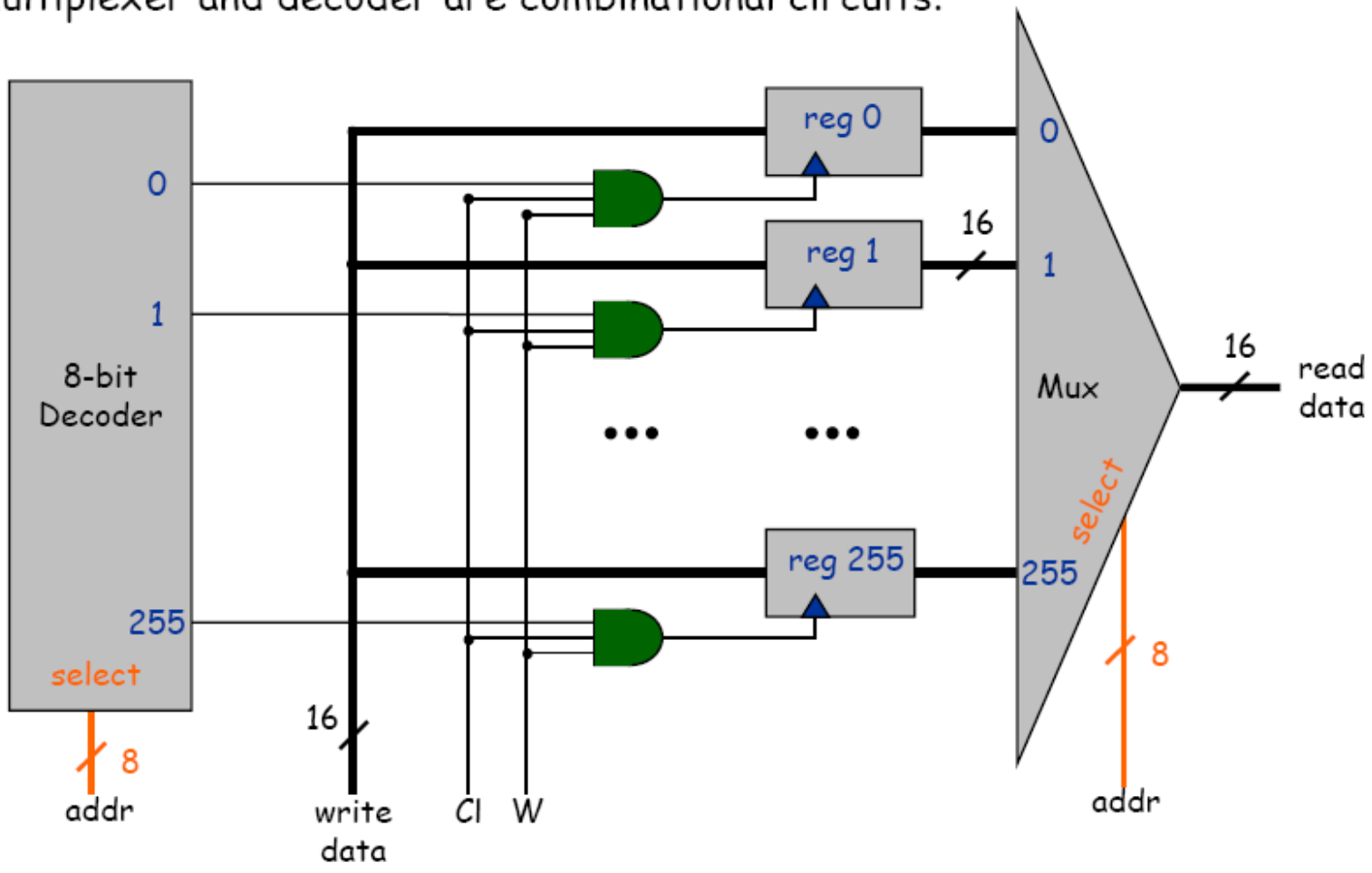


Interface

Register file implementation

Implementation example: TOY main memory.

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



Memory Overview

Computers and TOY have several memory components.

- Program counter.
- Registers.
- Main memory.

Implementation. Use one flip-flop for each bit of memory.

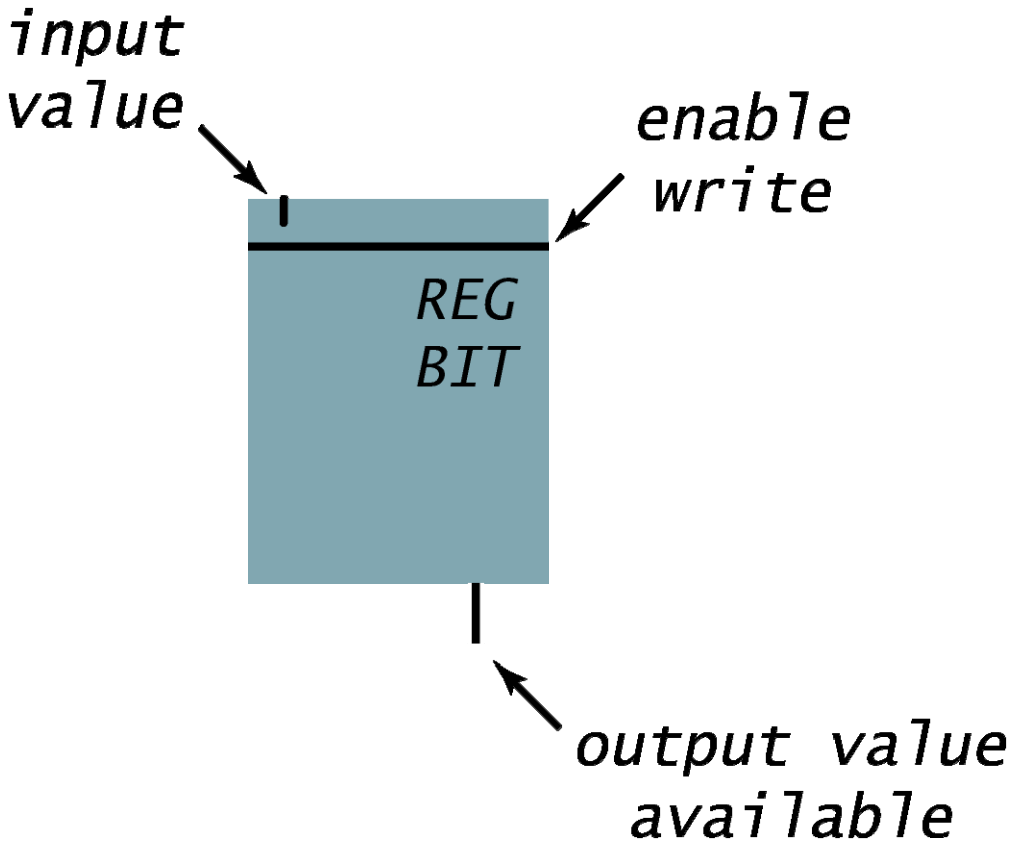
Access. Memory components have different access mechanisms.

TOY has 16 bit words,
8 bit memory addresses, and
4 bit register names.

Organization. Need mechanism to manipulate groups of related bits.

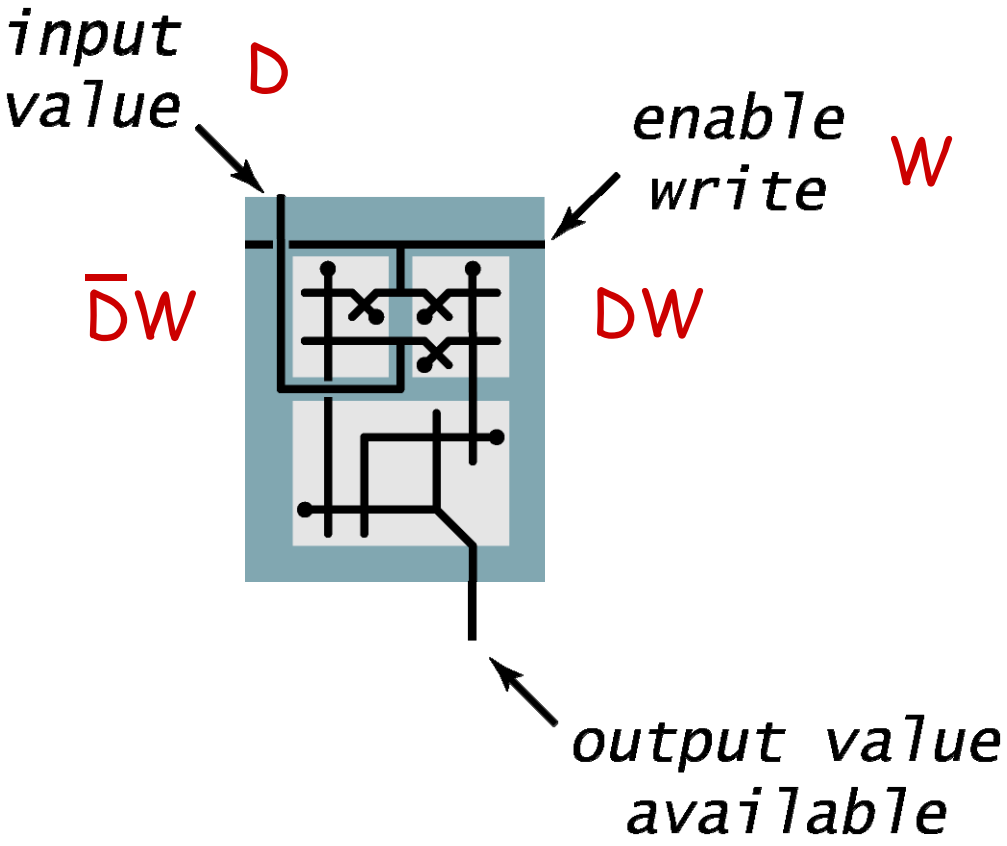
Register Bit

Register bit. Extend a flip-flop to allow easy access to values.



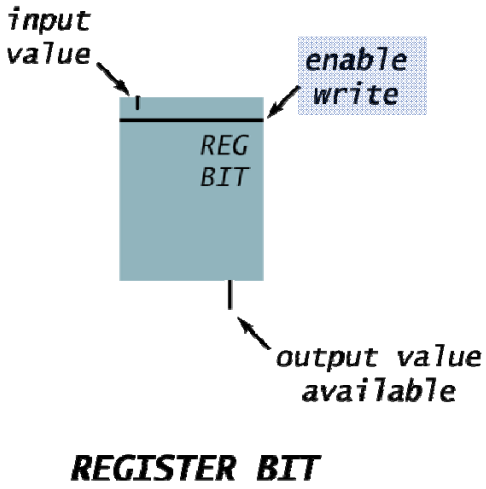
Register Bit

Register bit. Extend a flip-flop to allow easy access to values.

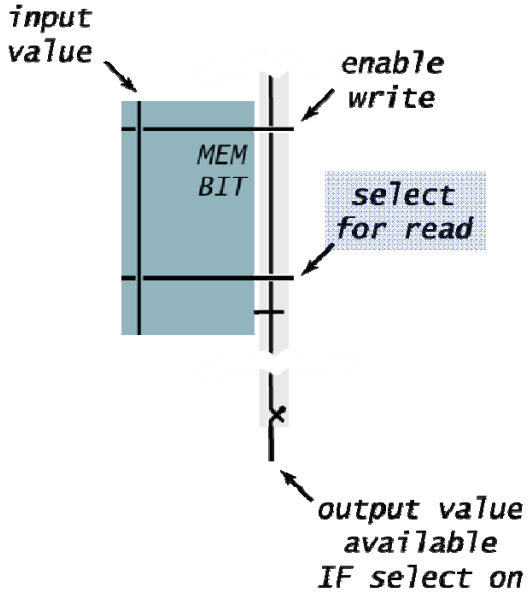


Memory Bit: Interface

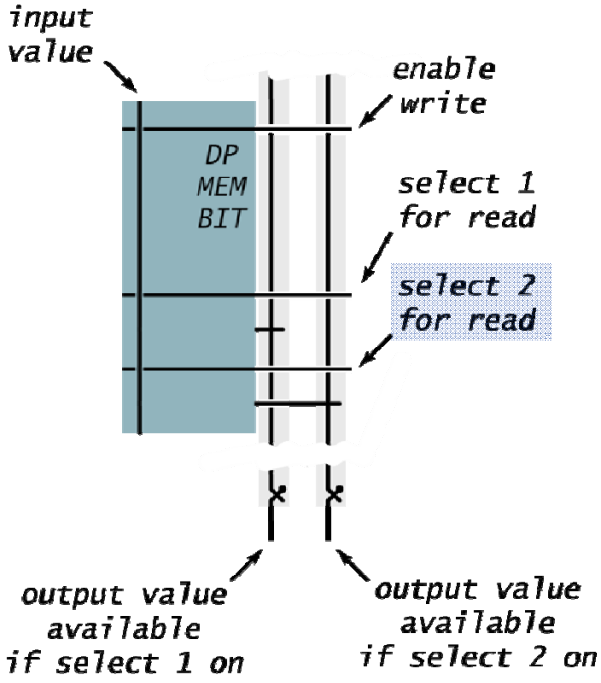
Memory bit. Extend a flip-flop to allow easy access to values.



REGISTER BIT



MEMORY-BANK BIT

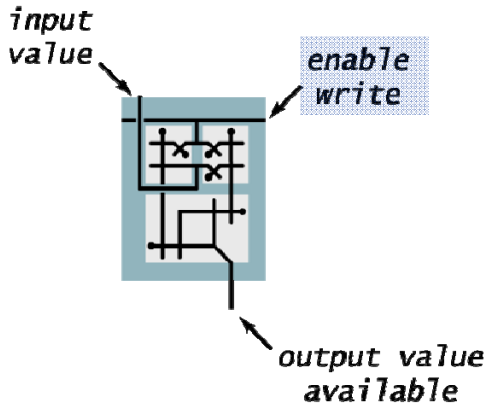


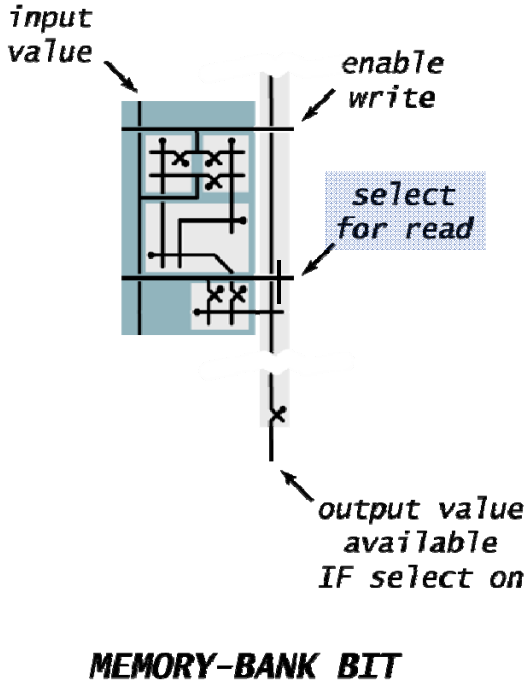
DUAL-PORT MEMORY-BANK BIT

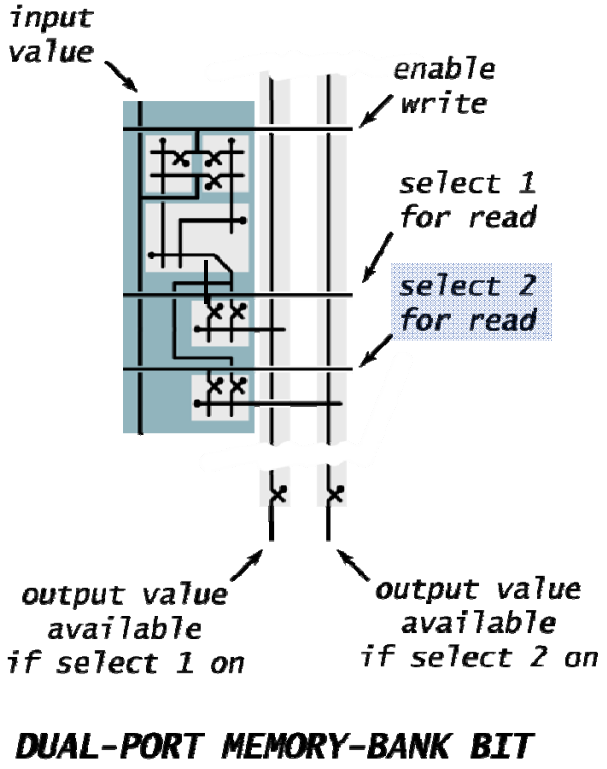
[TOY PC, IR] [TOY main memory] [TOY registers]

Memory Bit: Switch Level Implementation

Memory bit. Extend a flip-flop to allow easy access to values.







[TOY PC, IR] [TOY main memory] [TOY registers]

Processor Register

Processor register.

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

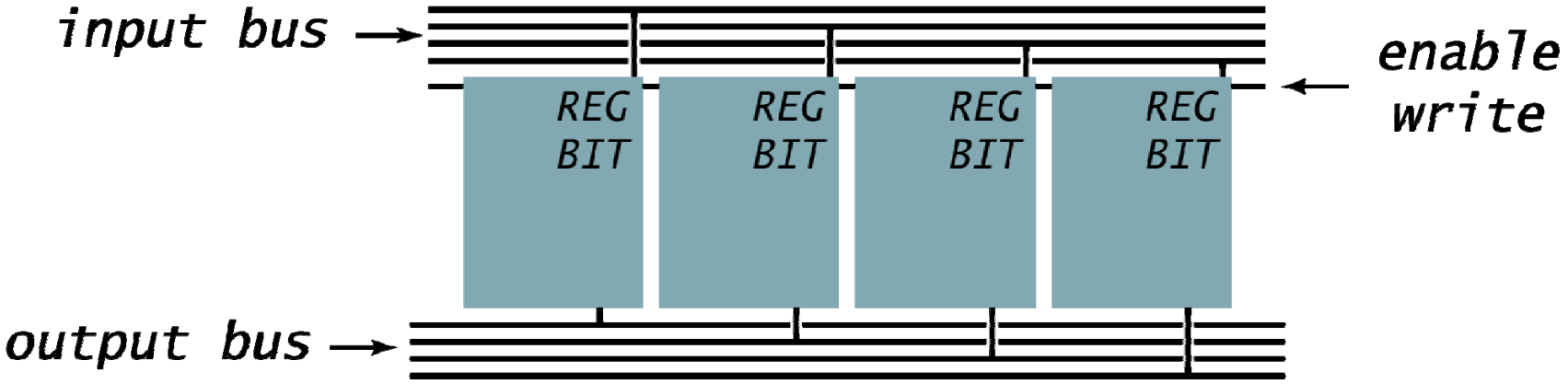
Ex 1. TOY program counter (PC) holds 8-bit address.
Ex 2. TOY instruction register (IR) holds 16-bit current instruction.

Processor Register

Processor register.

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

Ex 1. TOY program counter (PC) holds 8-bit address.
Ex 2. TOY instruction register (IR) holds 16-bit current instruction.

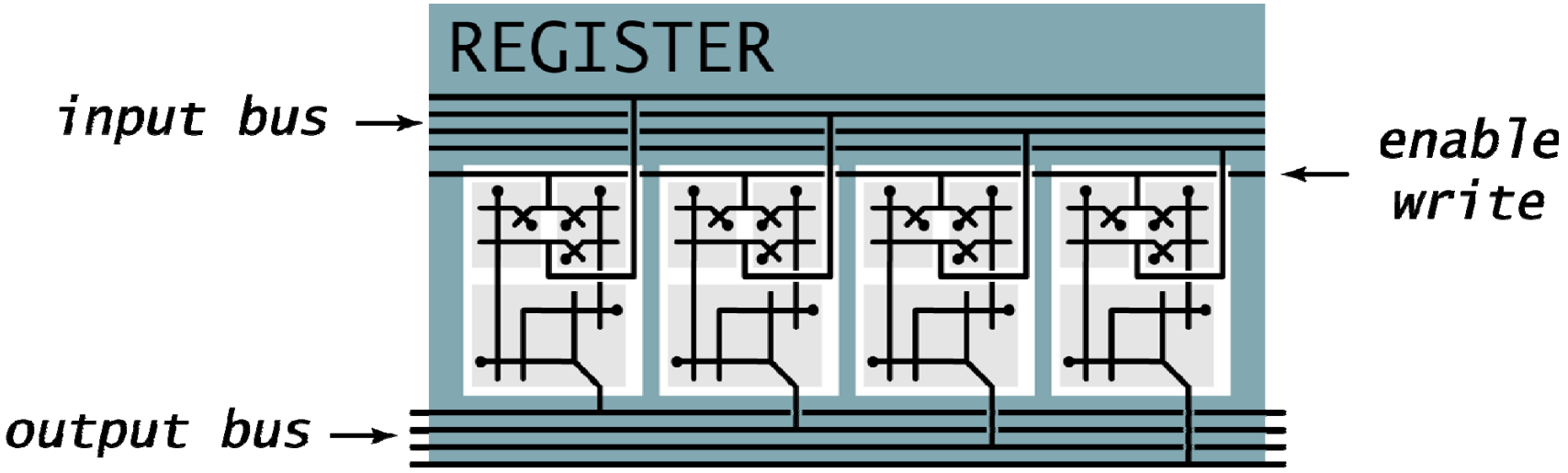


Processor Register

Processor register.

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

Ex 1. TOY program counter (PC) holds 8-bit address.
Ex 2. TOY instruction register (IR) holds 16-bit current instruction.



Memory Bank

Memory bank.

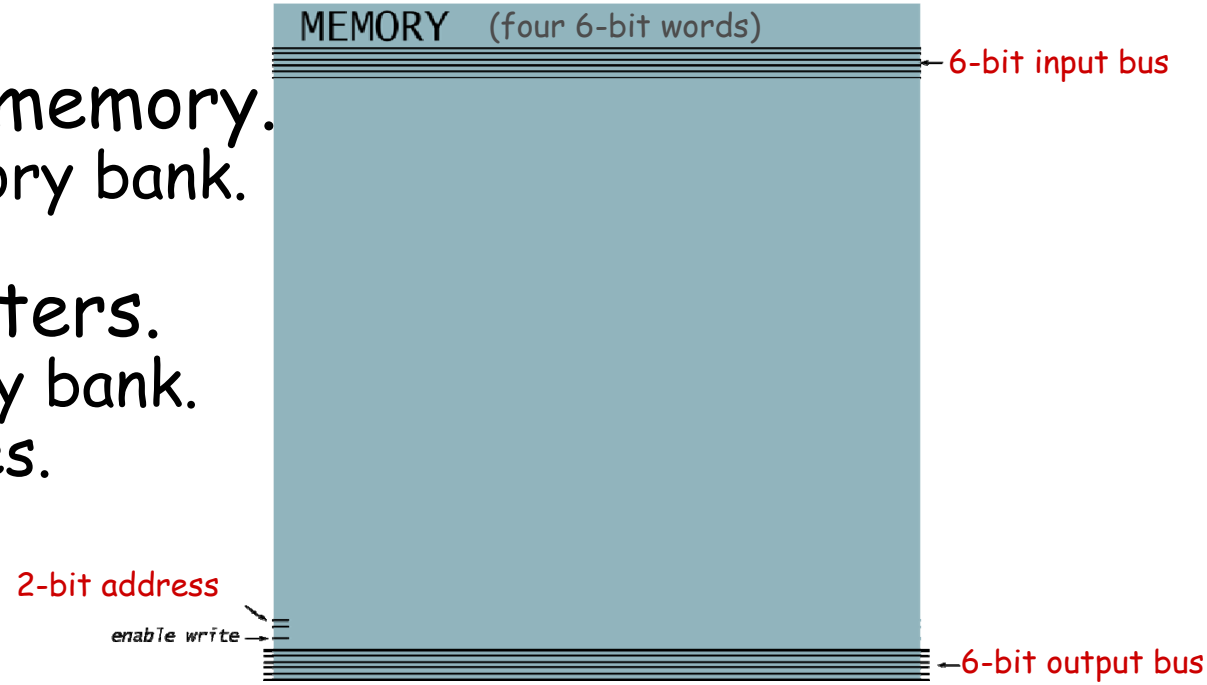
- Bank of n registers; each stores k bits.
- Read and write information to *one* of n registers.
- Address inputs specify which one. ← $\log_2 n$ address bits needed
- Addressed bits always appear on output.
- If write enabled, k input bits are copied into addressed register.

Ex 1. TOY main memory.

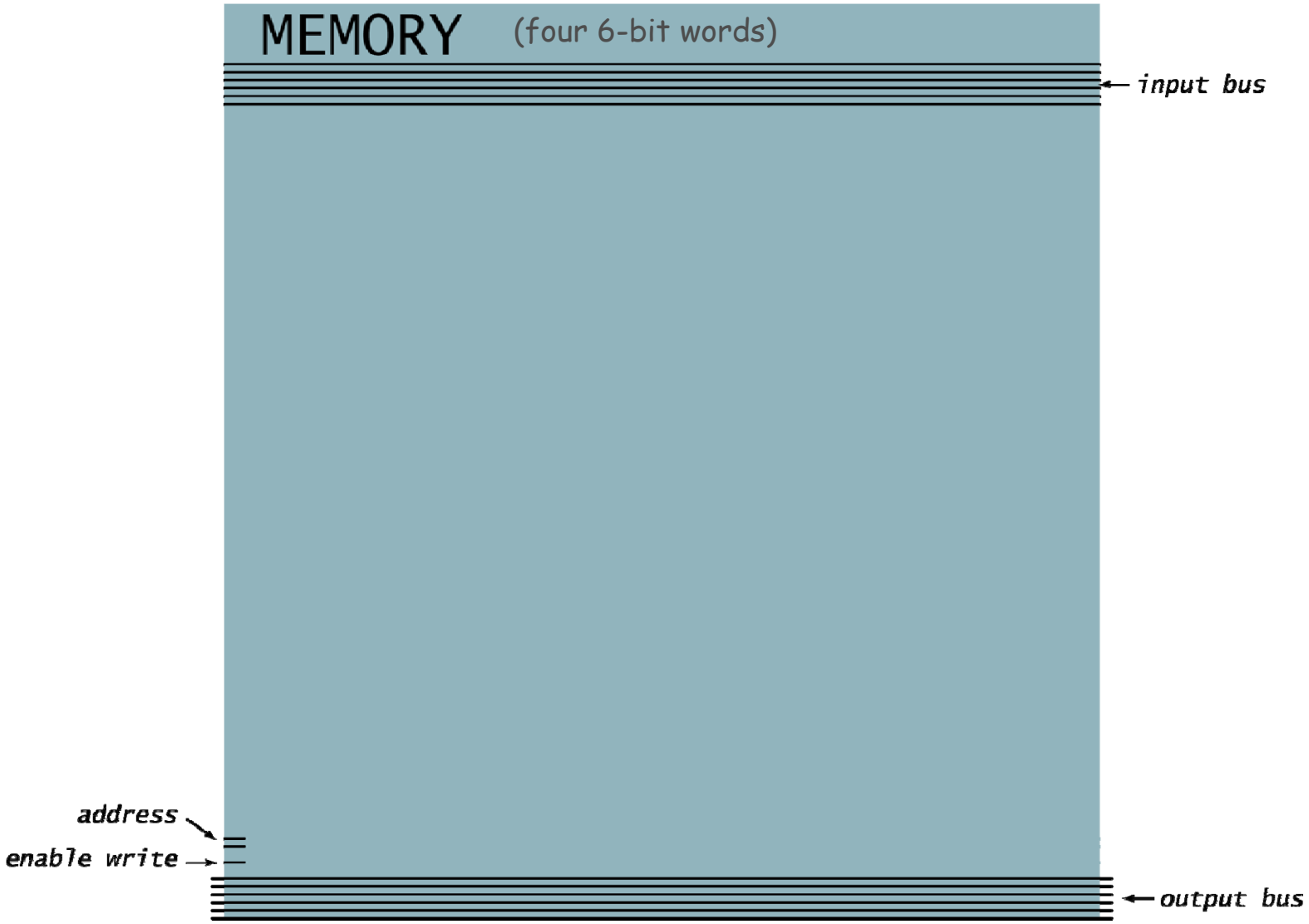
- 256-by-16 memory bank.

Ex 2. TOY registers.

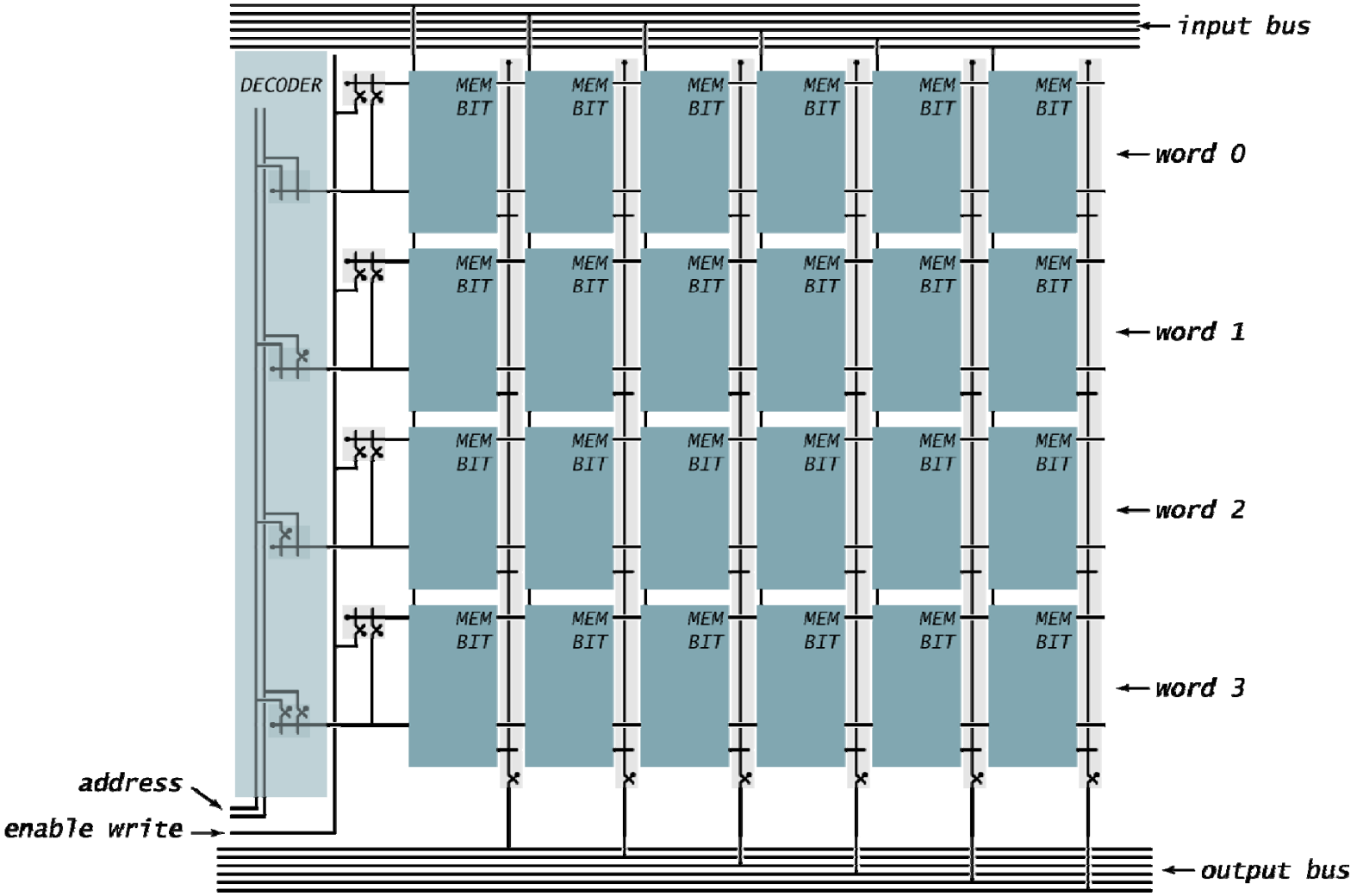
- 16-by-16 memory bank.
- Two output buses.



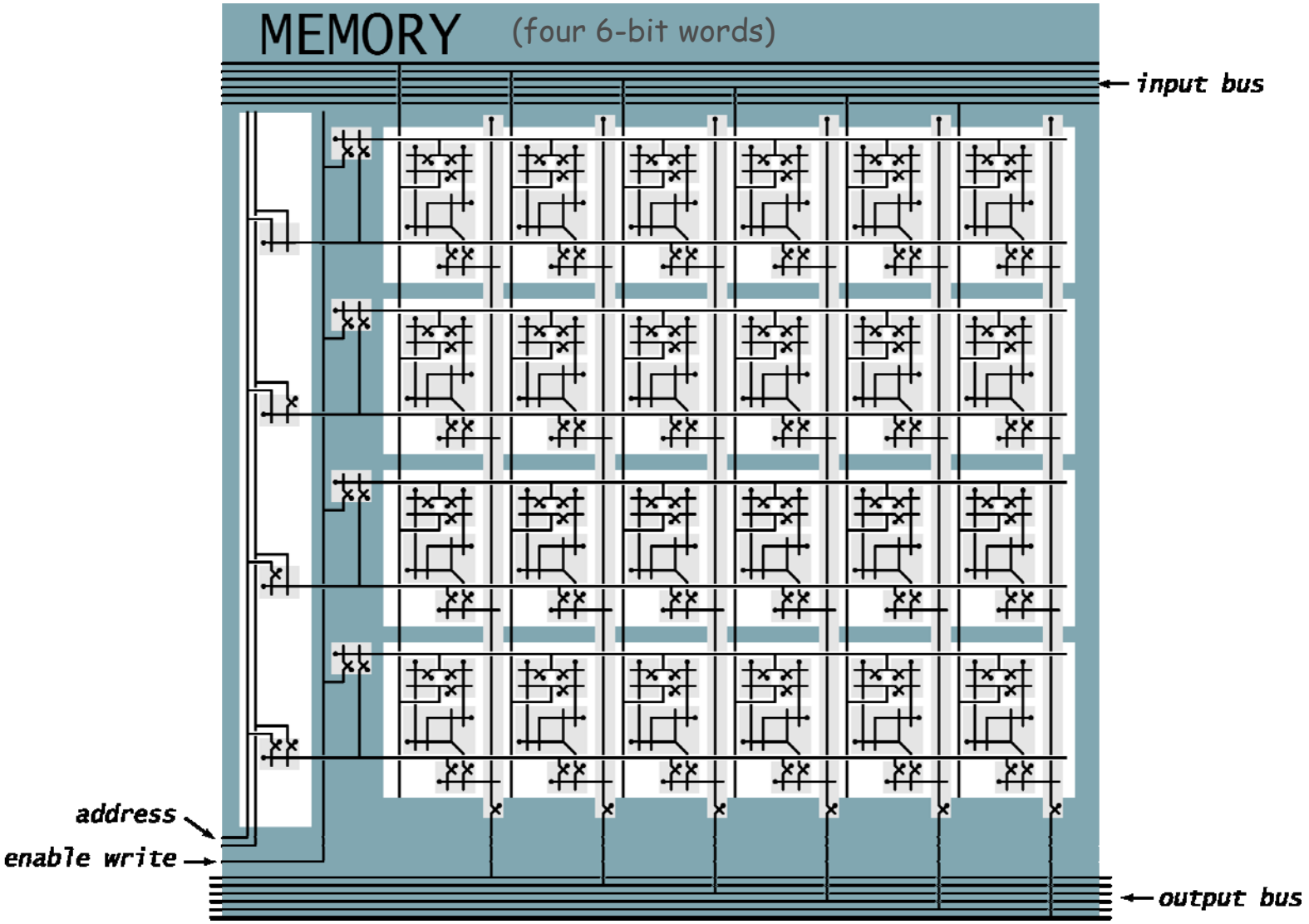
Memory: Interface



Memory: Component Level Implementation



Memory: Switch Level Implementation



Summary

Sequential circuits add "state" to digital hardware.

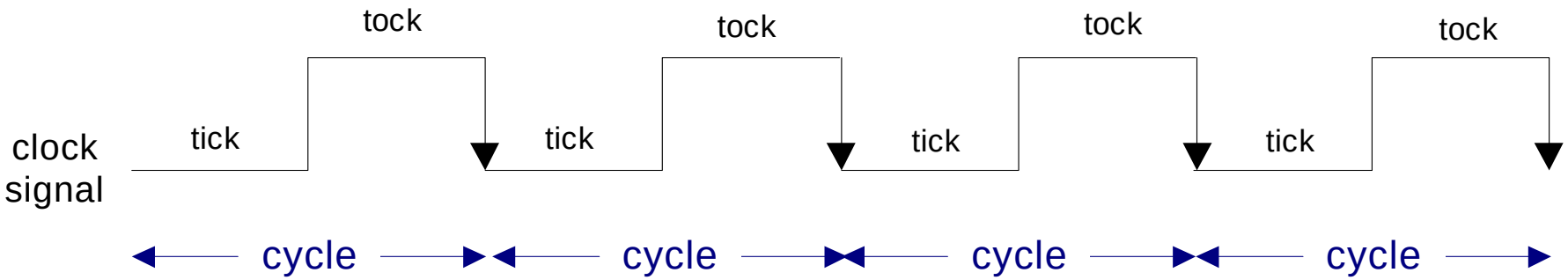
- Flip-flop. [represents 1 bit]
- TOY word. [16 flip-flops]
- TOY registers. [16 words]
- TOY main memory. [256 words]

Modern technologies for registers and main memory are different.

- Few registers, easily accessible, high cost per bit.
- Huge main memories, less accessible, low cost per bit.
- Drastic evolution of technology over time.

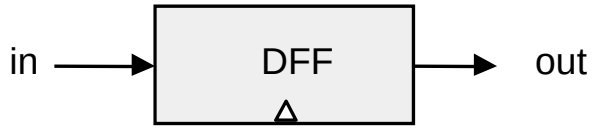
Next. Build a complete TOY computer.

The Clock



- In our jargon, a clock cycle = *tick*-phase (low), followed by a *tock*-phase (high)
- In real hardware, the clock is implemented by an oscillator
- In our hardware simulator, clock cycles can be simulated either
  - Manually, by the user, or
  - "Automatically," by a test script.

Flip-flop



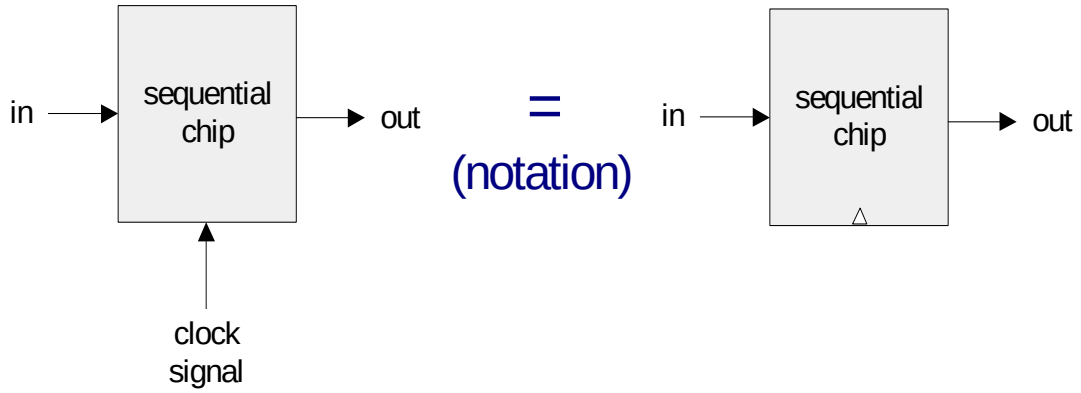
out(t) = in(t-1)

A fundamental state-keeping device

For now, let us not worry about the DFF *implementation*

Memory devices are made from numerous flip-flops, all regulated by the same master clock signal

Notational convention:

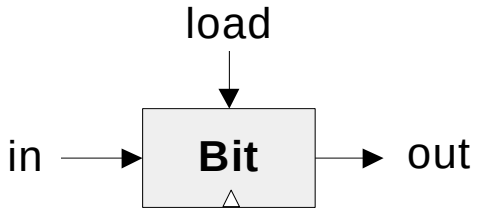


1-bit register (we call it "Bit")

Objective: build a storage unit that can:

(a) Change its state to a given input

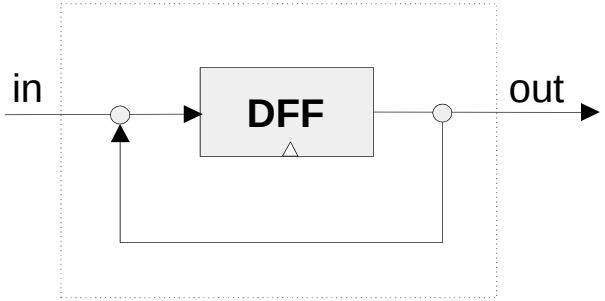
(b) Maintain its state over time (until changed)



if load(t-1) then out(t)=in(t-1)
else out(t)=out(t-1)

out(t) = in(t-1)

Basic building block

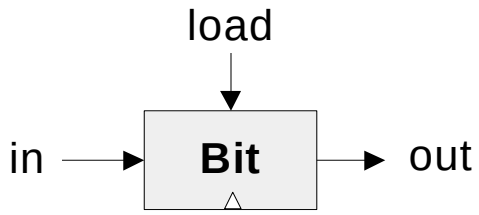


out(t) = out(t-1) ?
out(t) = in(t-1) ?

Won't work

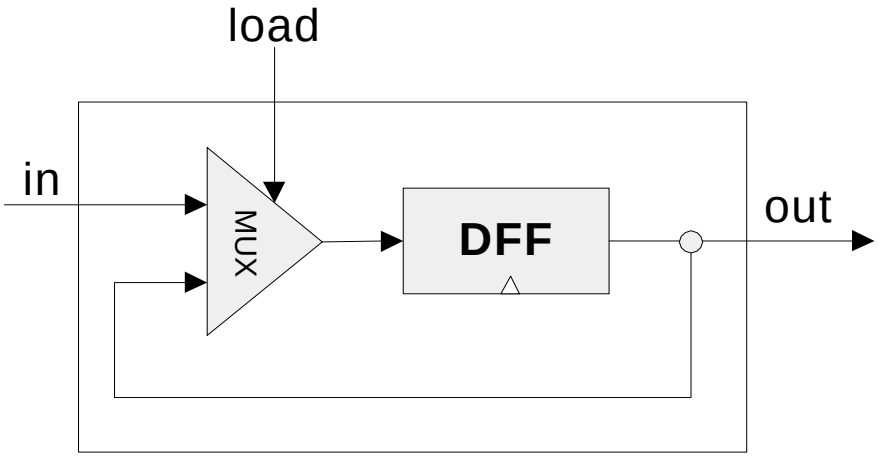
Bit register (cont.)

Interface



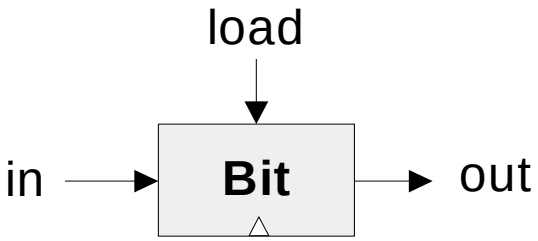
if load(t-1) then out(t)=in(t-1)
else out(t)=out(t-1)

Implementation



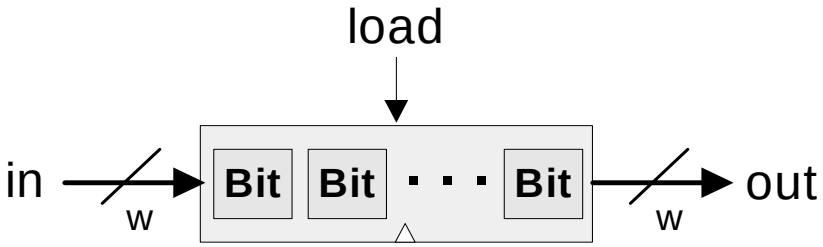
- Load bit
- Read logic
- Write logic

Multi-bit register



if load(t-1) then out(t)=in(t-1)
else out(t)=out(t-1)

1-bit register



if load(t-1) then out(t)=in(t-1)
else out(t)=out(t-1)

w-bit register

- Register's width: a trivial parameter
- Read logic
- Write logic

Aside: Hardware Simulation

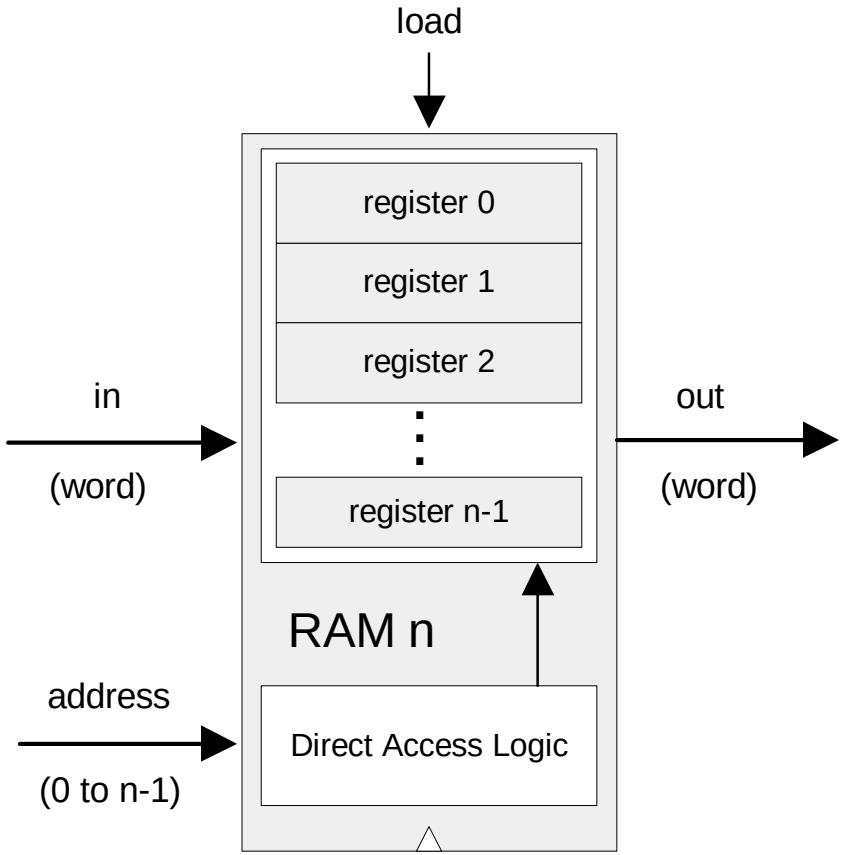
Relevant topics from the HW simulator tutorial:

Clocked chips: When a clocked chip is loaded into the simulator, the clock icon is enabled, allowing clock control

Built-in chips:

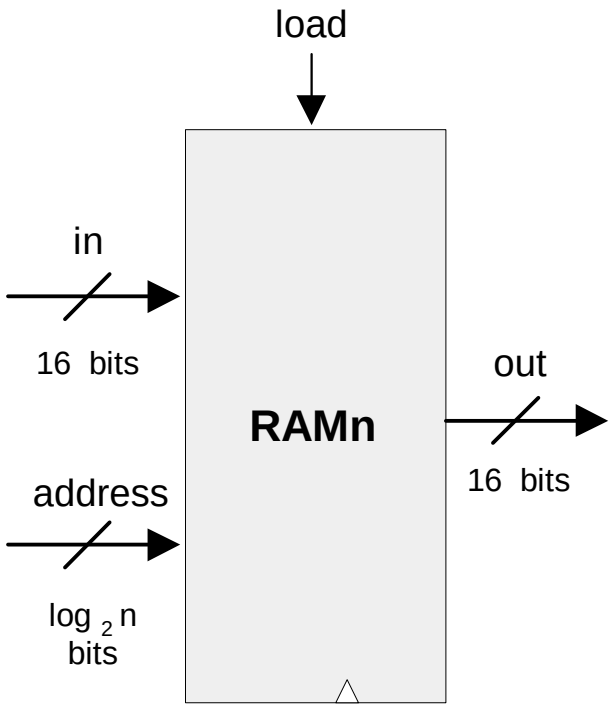
- feature a standard HDL interface yet a Java implementation
- Provide behavioral simulation services
- May feature GUI effects (at the simulator level only).

Random Access Memory (RAM)



- Read logic
- Write logic.

RAM interface

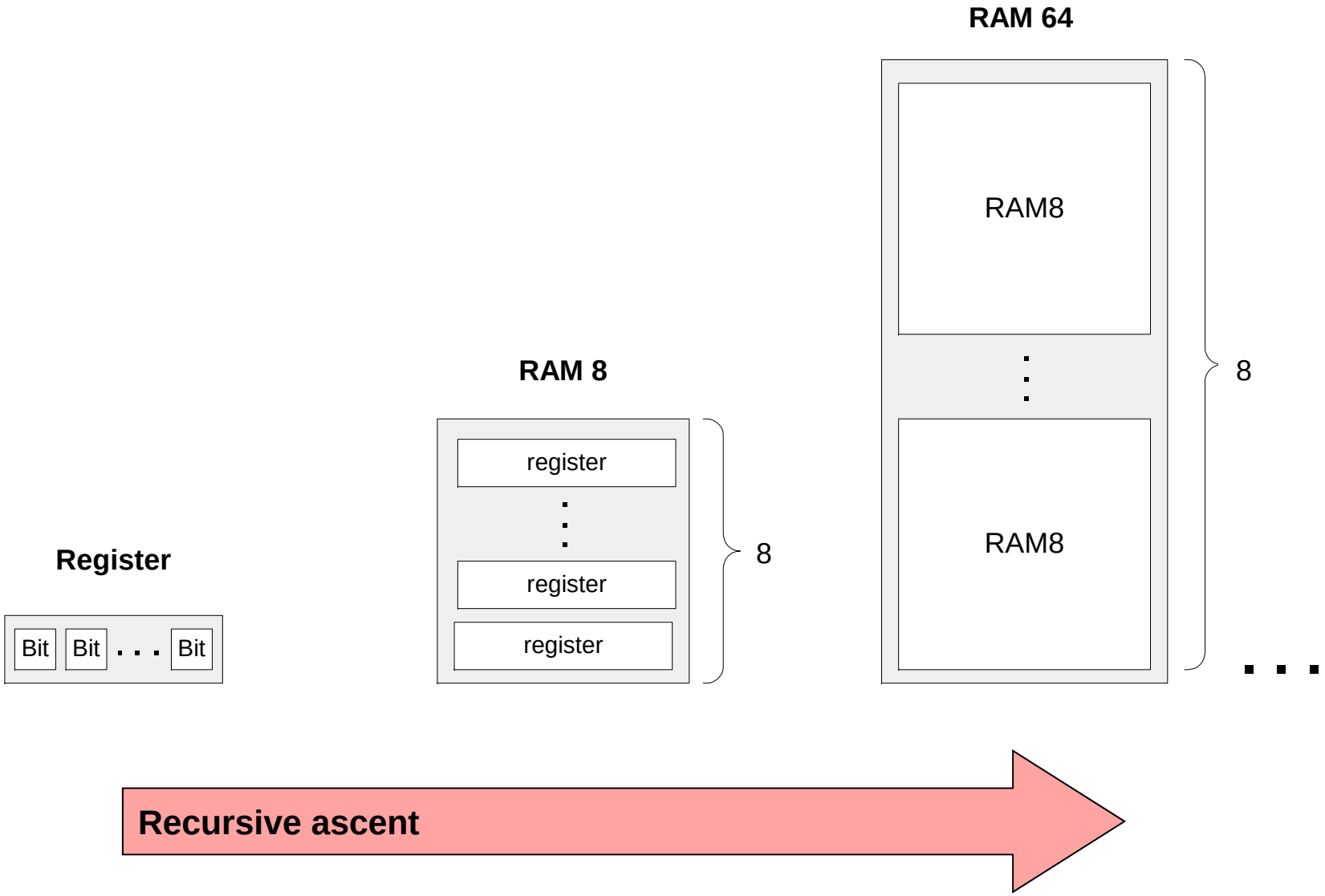



```
Chip name:   RAMn  // n and k are listed below
Inputs:      in[16], address[k], load
Outputs:     out[16]
Function:    out(t)=RAM[address(t)](t)
             If load(t-1) then
                RAM[address(t-1)](t)=in(t-1)
Comment:     "=" is a 16-bit operation.
```

The specific RAM chips needed for the Hack platform are:

Chip name	n	K
RAM8	8	3
RAM64	64	6
RAM512	512	9
RAM4K	4096	12
RAM16K	16384	14

RAM anatomy


RAM 64
RAM8
RAM8
8
RAM 8
register
register
register
8
Register
Bit
Bit
Bit
Recursive ascent

Counter

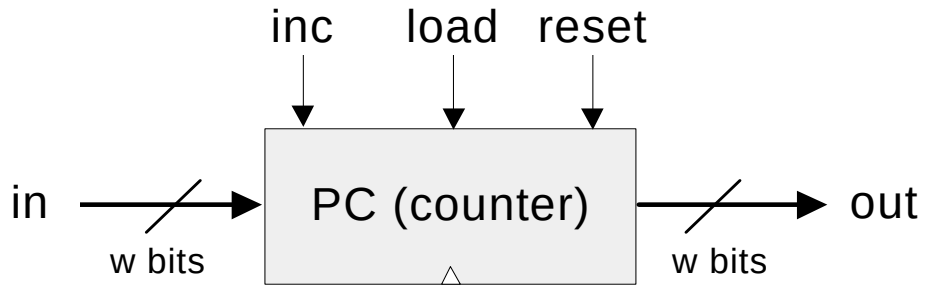
Needed: a storage device that can:

(a) set its state to some base value

(b) increment the state in every clock cycle

(c) maintain its state (stop incrementing) over clock cycles

(d) reset its state



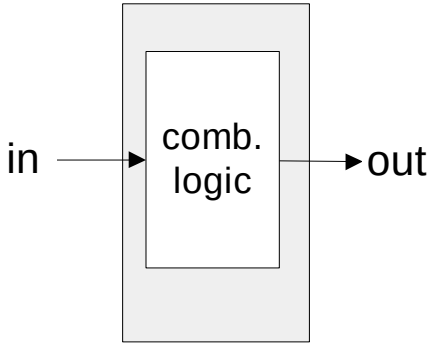

```
If reset(t-1) then out(t)=0
   else if load(t-1) then out(t)=in(t-1)
        else if inc(t-1) then out(t)=out(t-1)+1
             else out(t)=out(t-1)
```

Typical function: *program counter*

Implementation: register chip + some combinational logic.

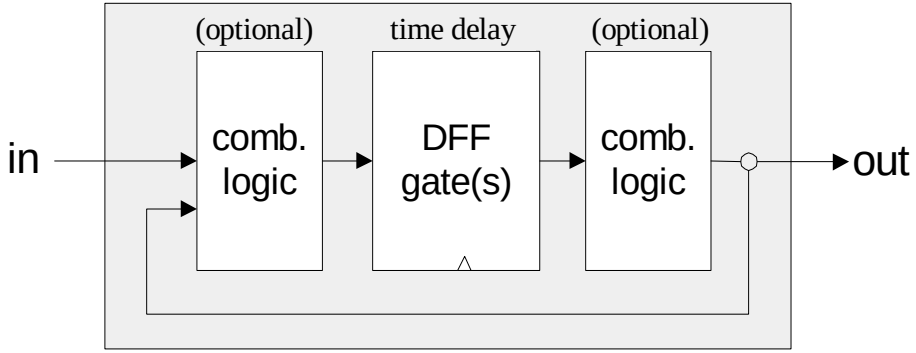
Recap: Sequential VS combinational logic

Combinational chip



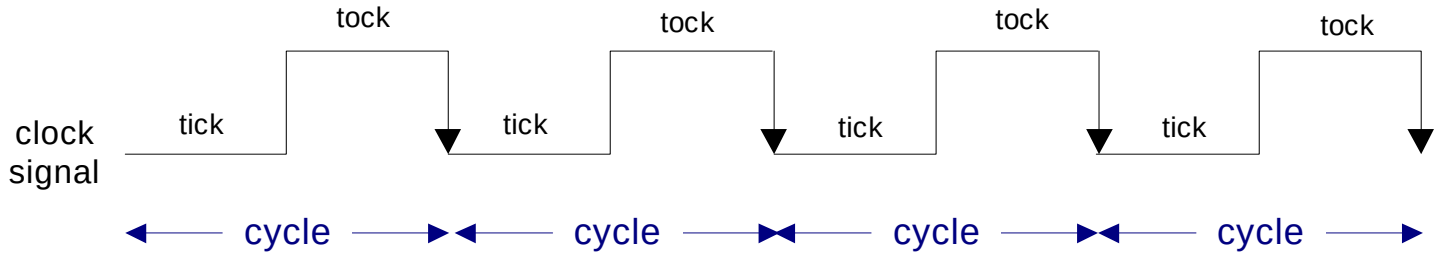
out = *some function of* (in)

Sequential chip

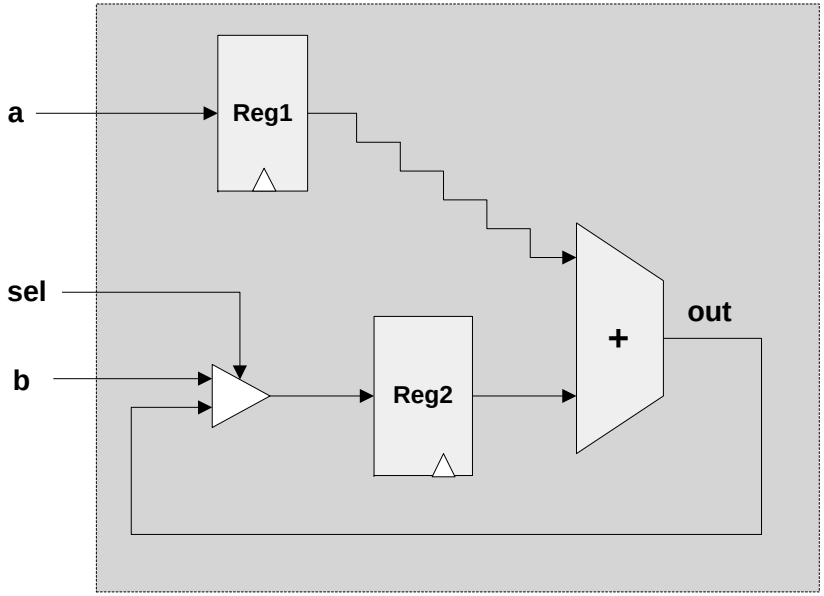


out(t) = *some function of* (in(t-1), out(t-1))

Time matters



- During a tick-tock cycle, the internal states of all the clocked chips are allowed to change, but their outputs are "latched"
- At the beginning of the next cycle, the outputs of all the clocked chips in the architecture commit to the new values.



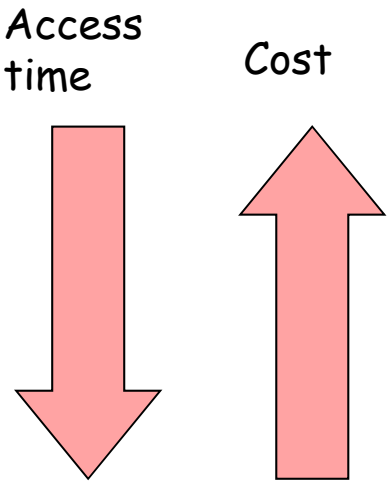
Implications:

- Challenge: propagation delays
- Solution: clock synchronization
- Cycle length and processing speed.

Perspective

All the memory units described in this lecture are standard

Typical memory hierarchy



- SRAM ("static"), typically used for the cache
- DRAM ("dynamic"), typically used for main memory
- Disk

(Elaborate caching / paging algorithms)

A Flip-flop can be built from Nand gates

But ... real memory units are highly optimized, using a great variety of storage technologies.