Introduction to the Architecture of Computers

Professor Hugh C. Lauer
CS-2011, Machine Organization and Assembly Language

(Slides shamelessly adapted from Bryant & O'Hallaron, with additional materials from Patterson & Hennessey, "Computer Organization & Design," revised 4th ed. and from Patt & Patel, Introduction to Computer Systems," 2nd, ed.)

Today

- **Before electronic computers**
- Logic and gates
- Latches and Registers

Before electronic computers

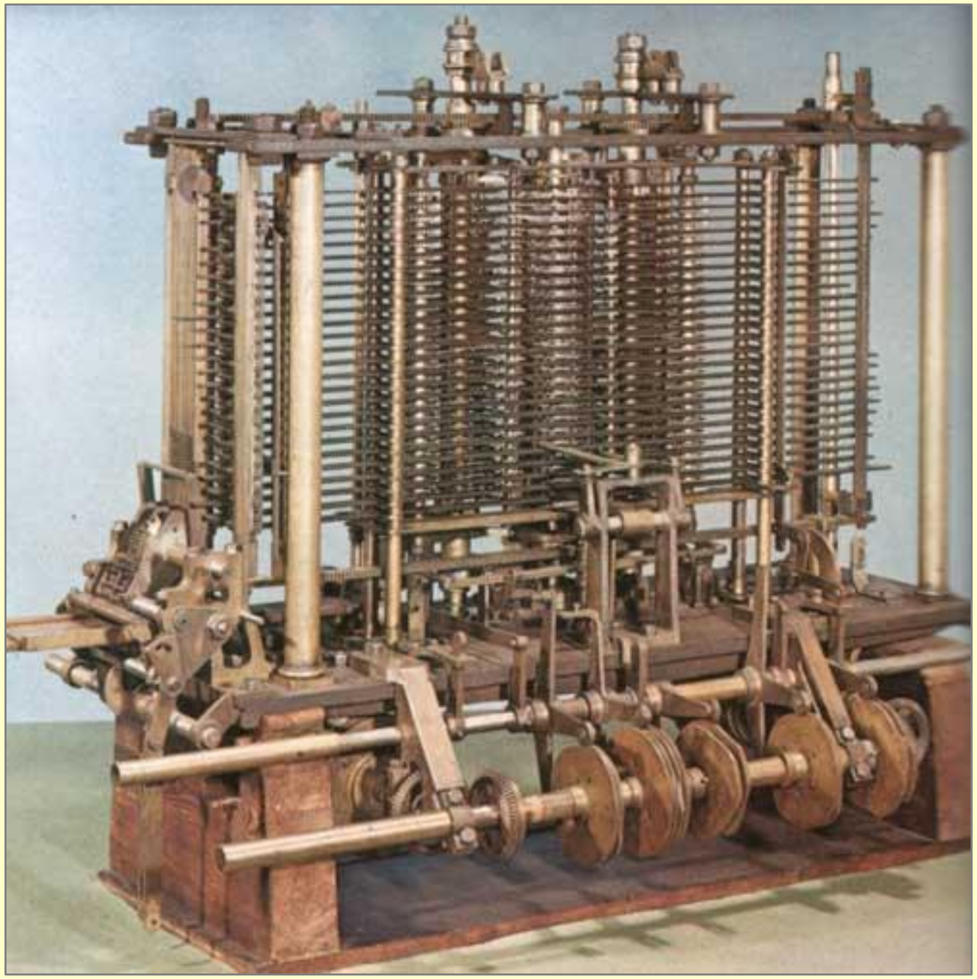
Data values represented by positions of beads

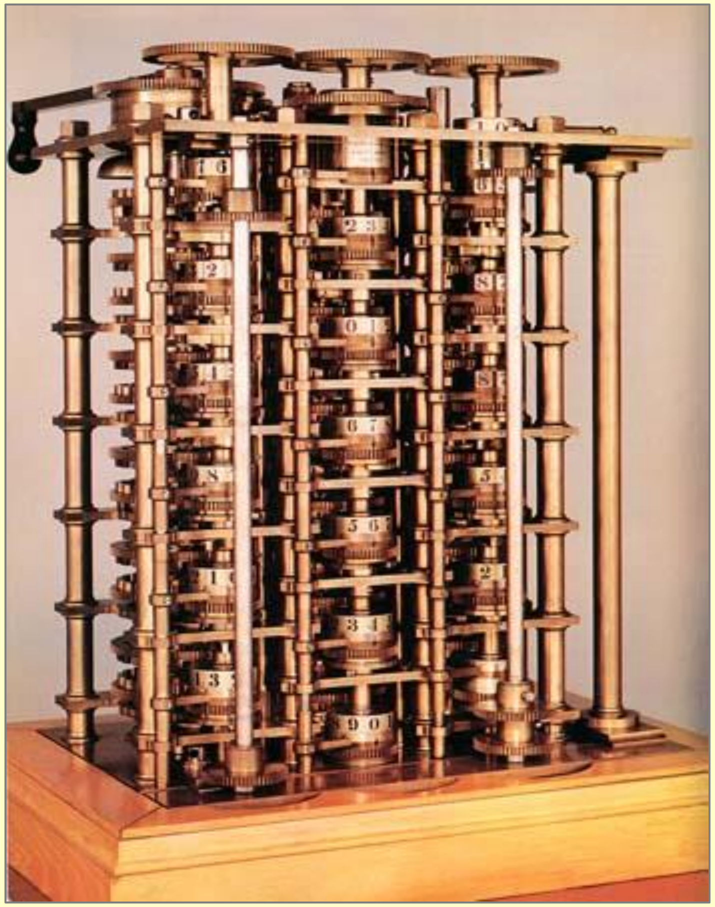
Arithmetic by manual algorithm

Data values represented by rotational positions of wheels and dials

Arithmetic by rotating wheels and gears

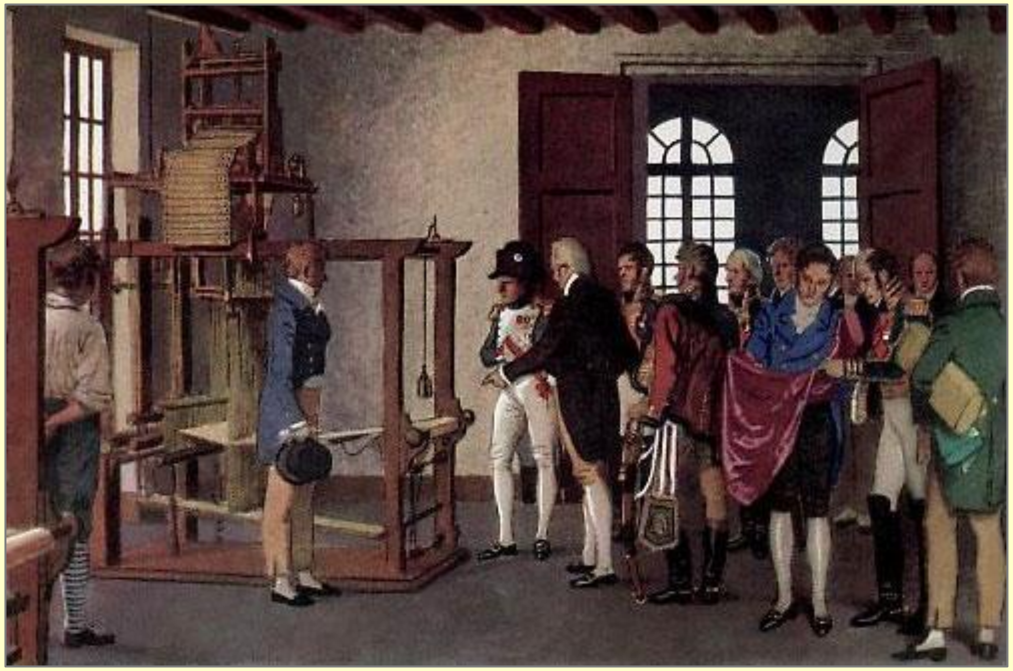
Charles Babbage "engines"


Analytical engine


Difference engine

Data values represented by rotational positions of wheels and dials

Jacquard Loom



Punched cards for controlling patterns of woven cloth

Punched cards were part of Babbage’s design for data entry and program control

Punched card tabulating equipment

Late 19th century

Mid 20th century

Data stored in trays (i.e., “files”) of punched cards
algorithms coded into plug-boards to operate on
data, punch new cards, etc.

Today

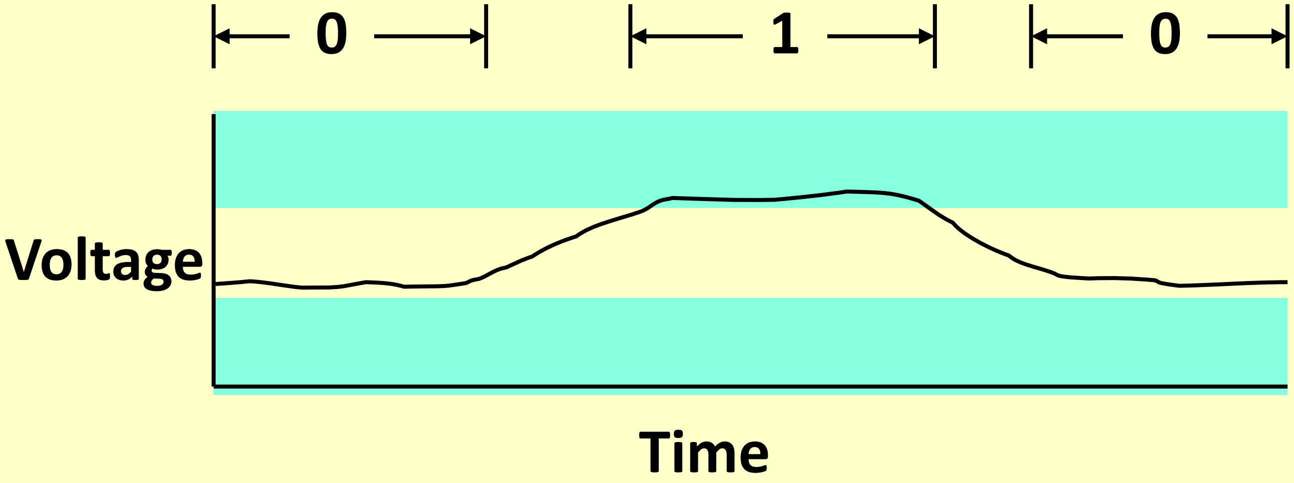
- Before electronic computers
- **Logic and gates**
- Latches and Registers

Reading Assignment: §4.2

Overview of Logic Design

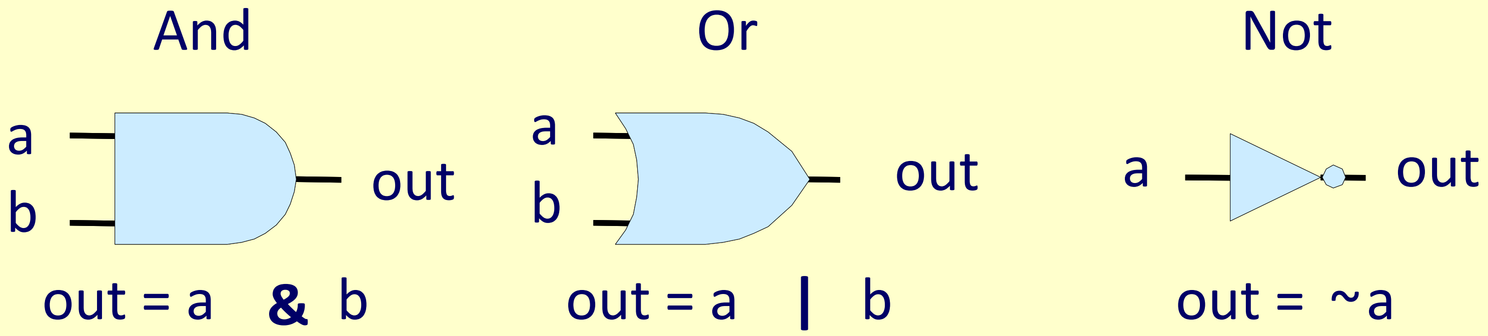
- **Fundamental Hardware Requirements**
  - Communication
    - How to get values from one place to another
  - Computation
  - Storage
- **Bits are Our Friends**
  - Everything expressed in terms of values 0 and 1
  - Communication
    - Low or high voltage on wire
  - Computation
    - Compute Boolean functions
  - Storage
    - Store bits of information

Digital Signals

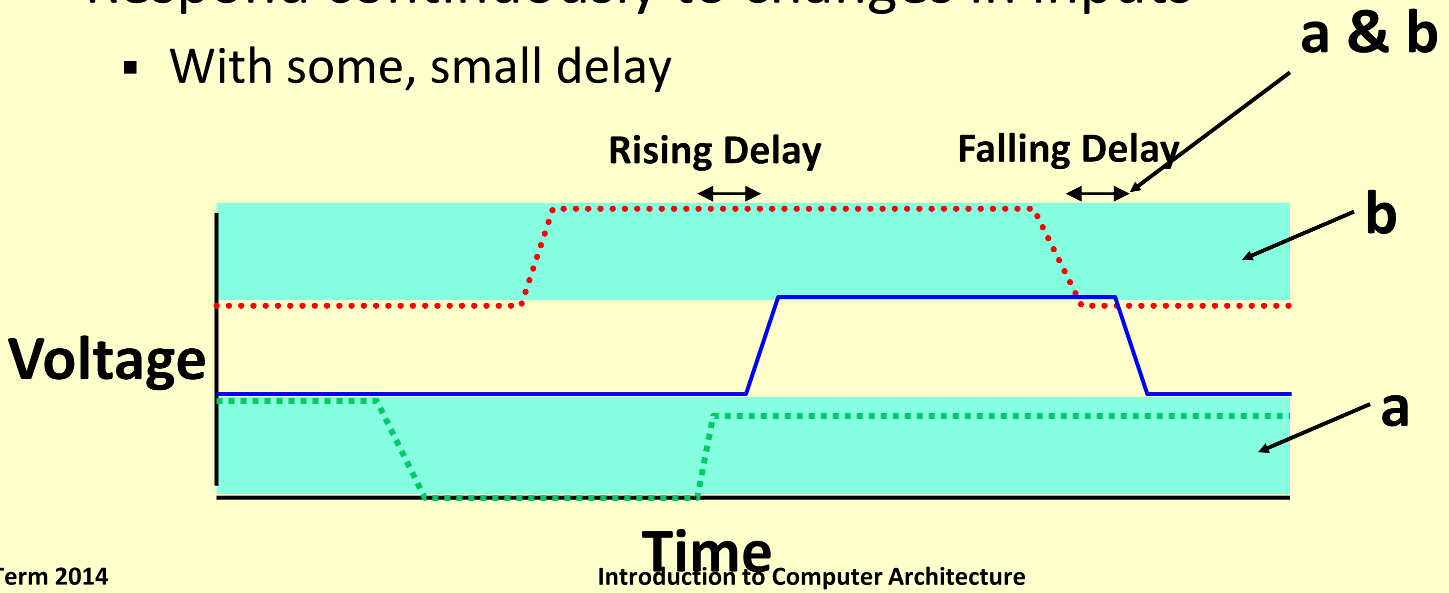


- Use voltage thresholds to extract discrete values from continuous signal
- Simplest version: 1-bit signal
  - Either high range (1) or low range (0)
  - With guard range between them
- Not strongly affected by noise or low quality circuit elements
  - Can make circuits simple, small, and fast

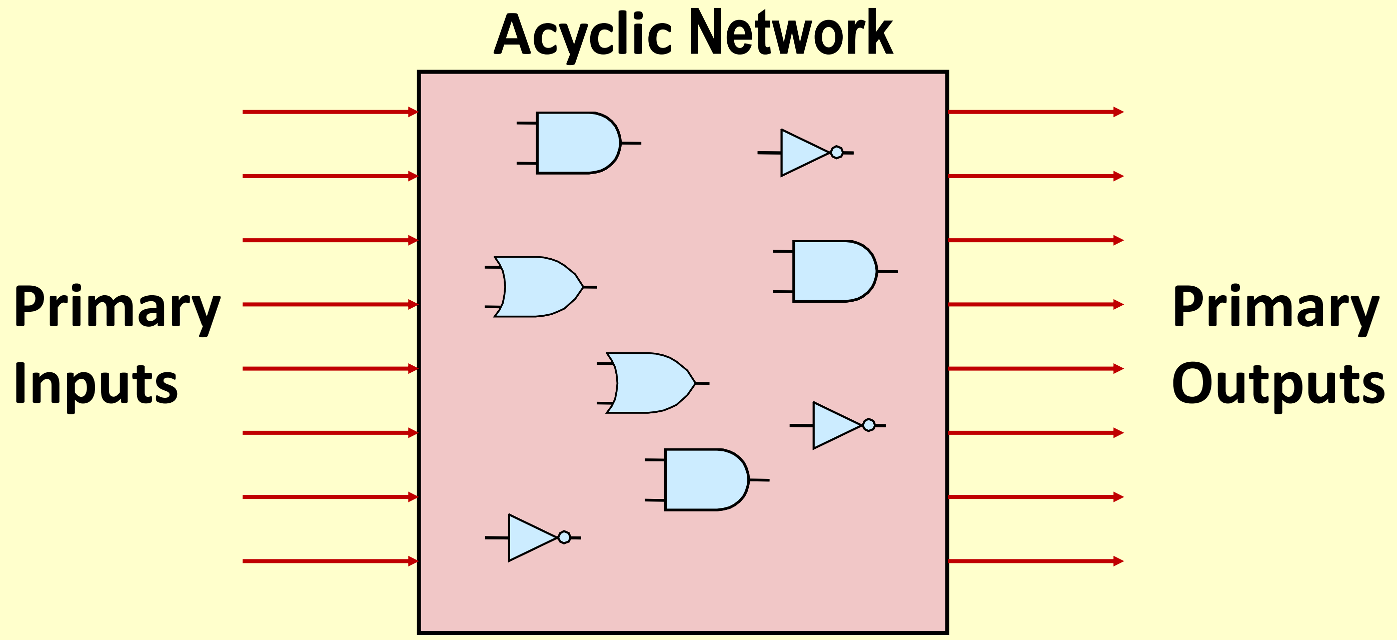
Computing with Logic Gates



- Outputs are Boolean functions of inputs
- Respond continuously to changes in inputs
  - With some, small delay

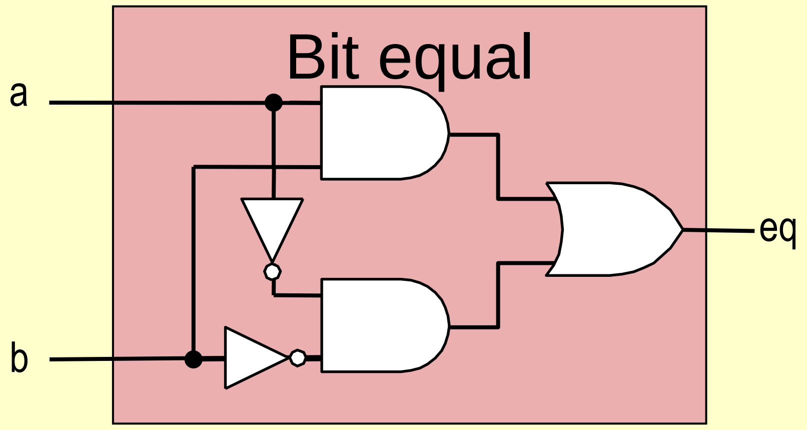


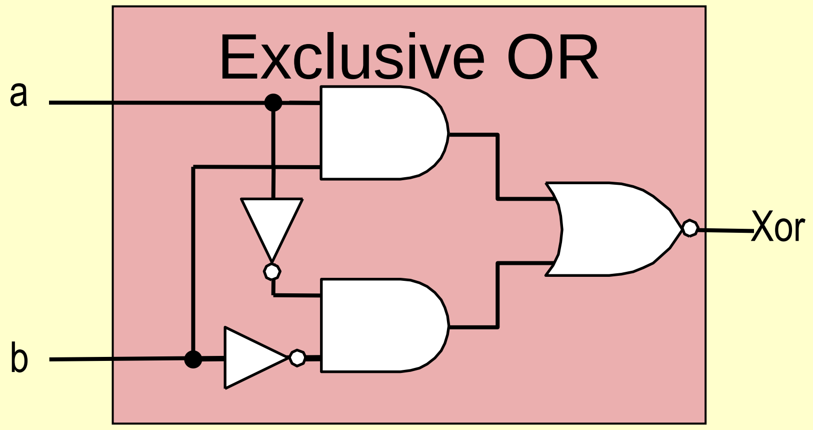
Combinational Circuits



- **Acyclic Network of Logic Gates**
  - Continously responds to changes on primary inputs
  - Primary outputs become (after some delay) Boolean functions of primary inputs

Bit Equality and Exclusive OR

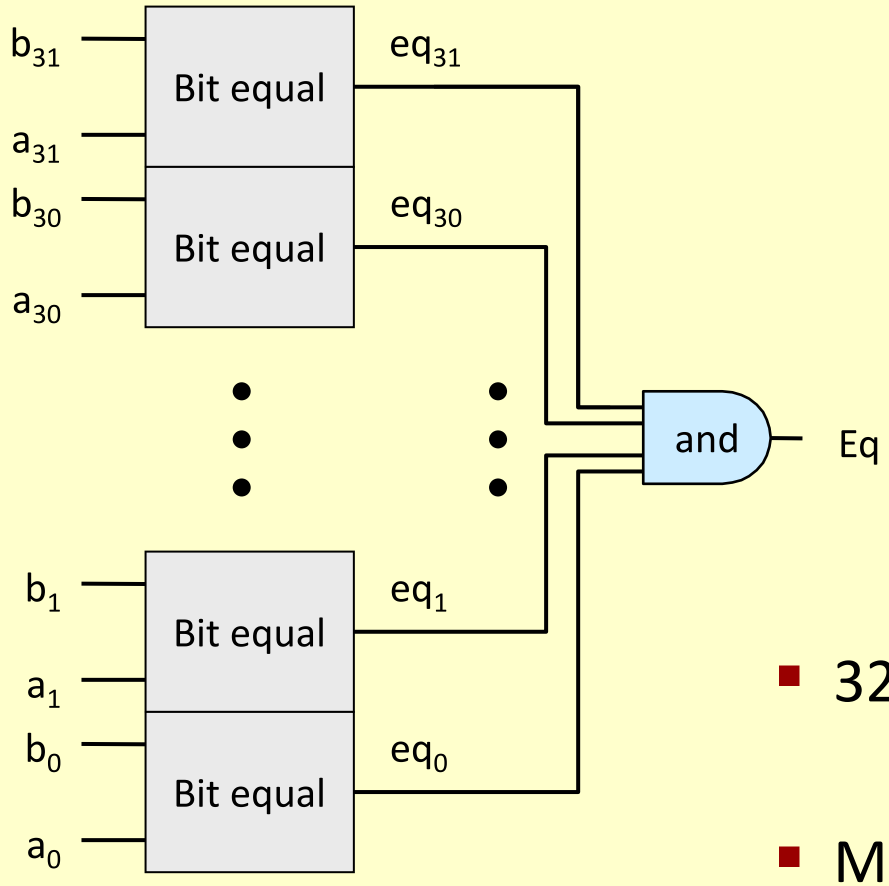




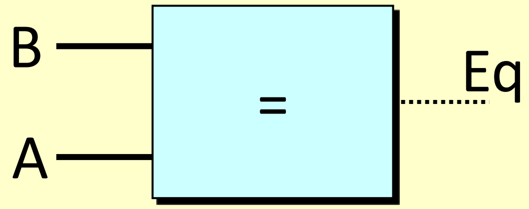
- Generate 1 if a and b are equal

- Generate 1 if a and b are *not* equal

Word Equality

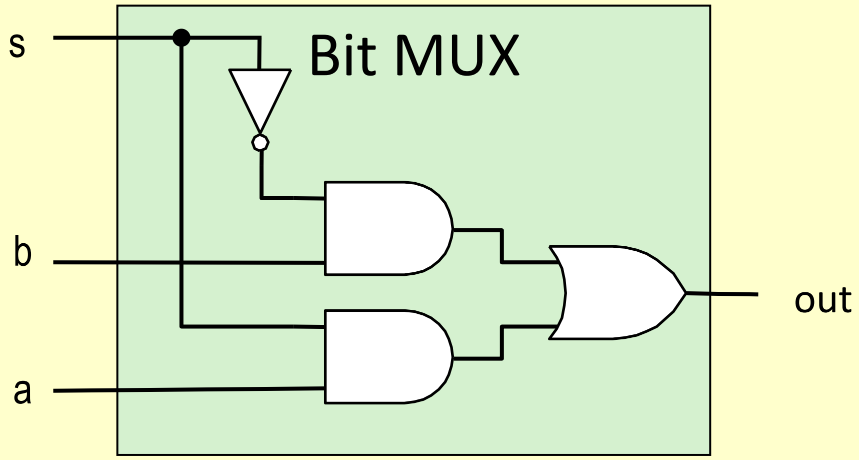


Word-Level Representation



- 32-bit word size
- May be adapted to any word size

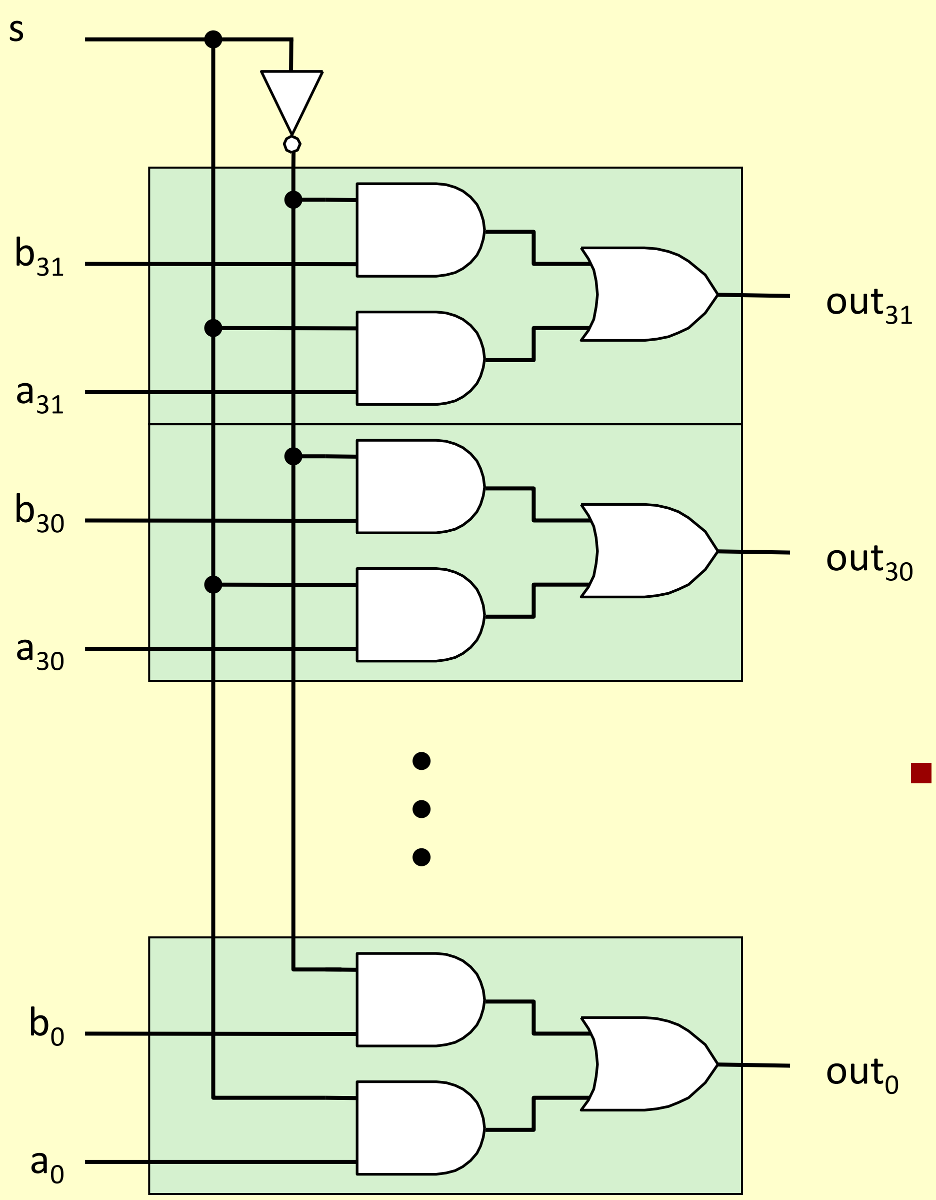
Bit-Level Multiplexor



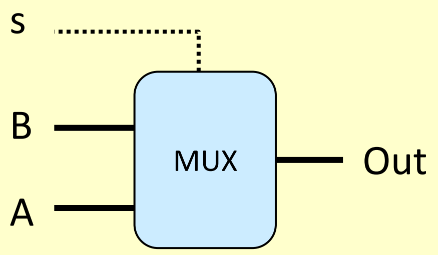
```
bool out = (s&a)||(!s&b)
```

- Control signal s
- Data signals a and b
- Output a when s=1, b when s=0

Word Multiplexor



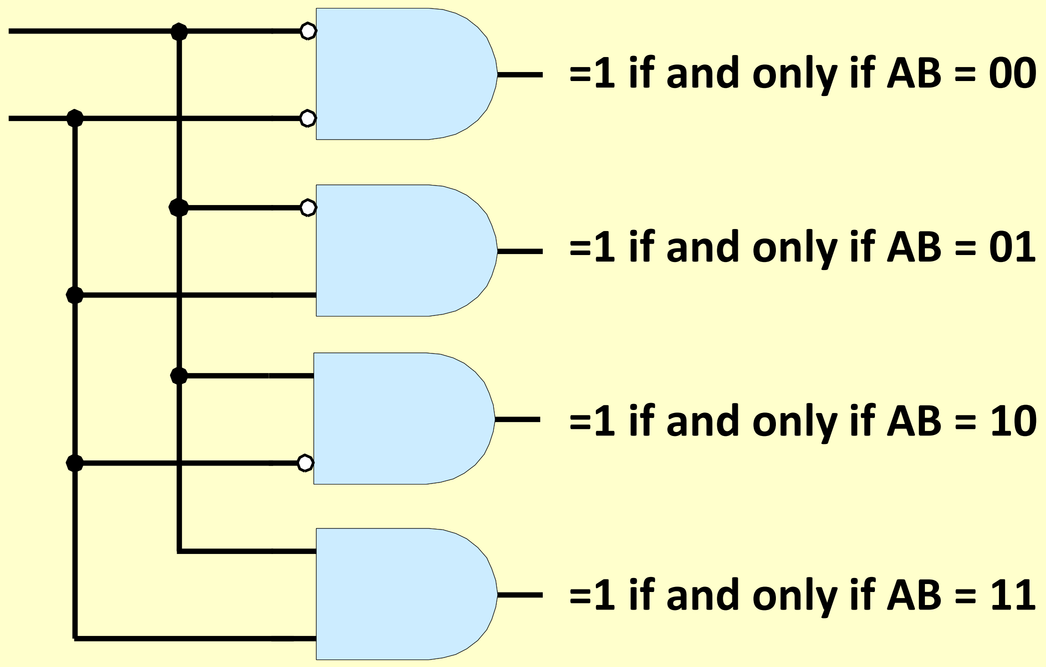
Word-Level Representation



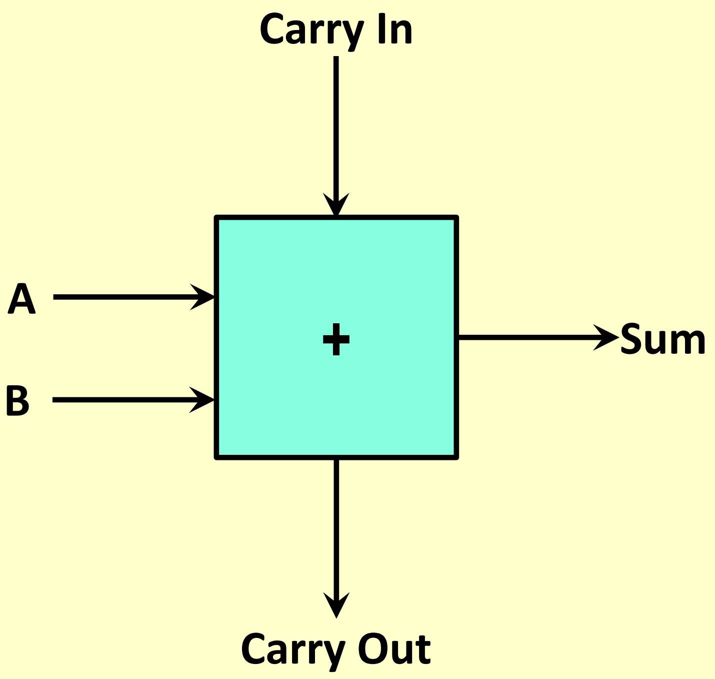
- Select input word A or B depending on control signal s

Decoder

- **Opposite of Multiplexor**
- **Selects one of *2ⁿ* outputs from *n* inputs**


=1 if and only if AB = 00
=1 if and only if AB = 01
=1 if and only if AB = 10
=1 if and only if AB = 11

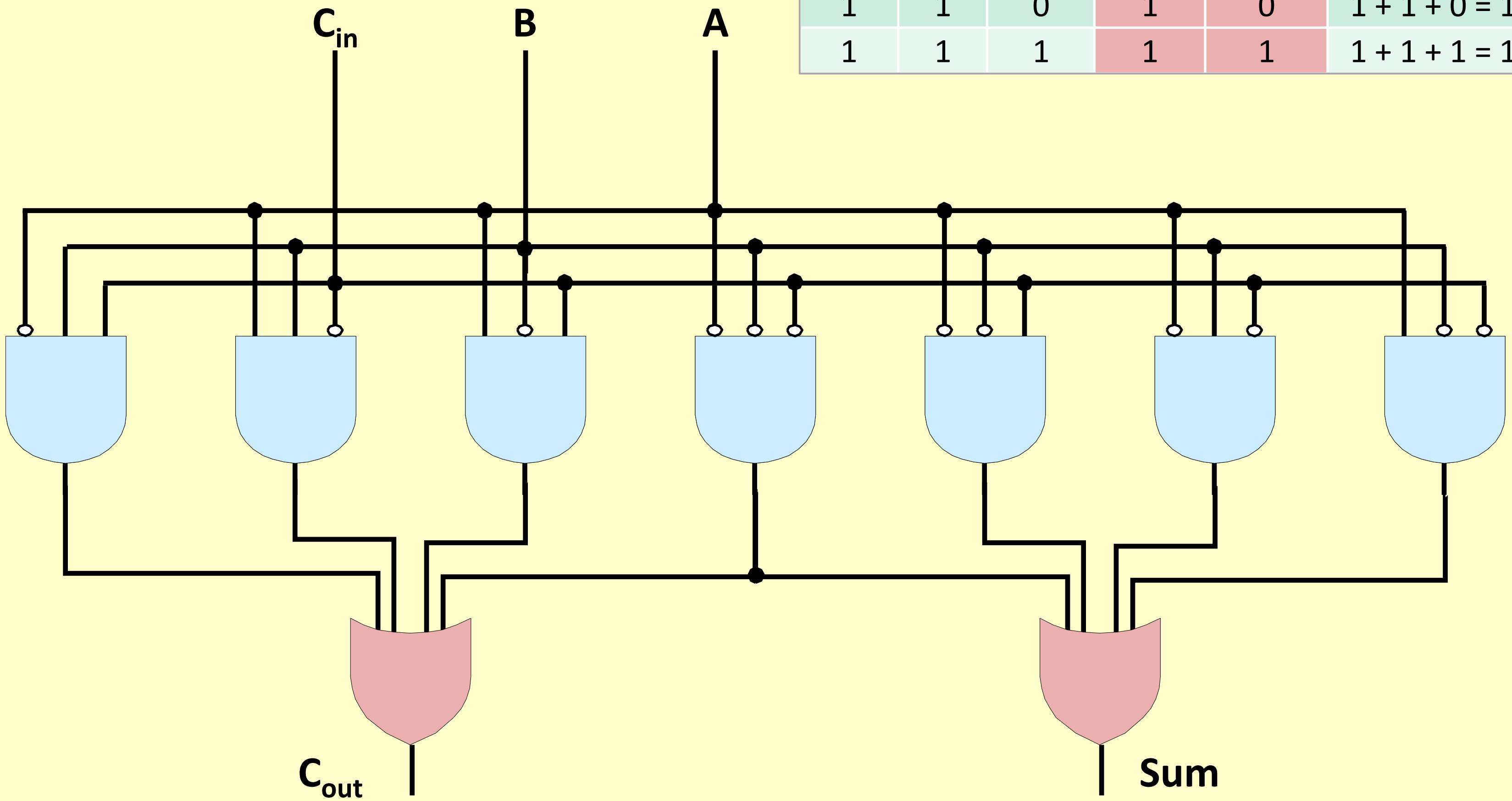
Single-bit adder



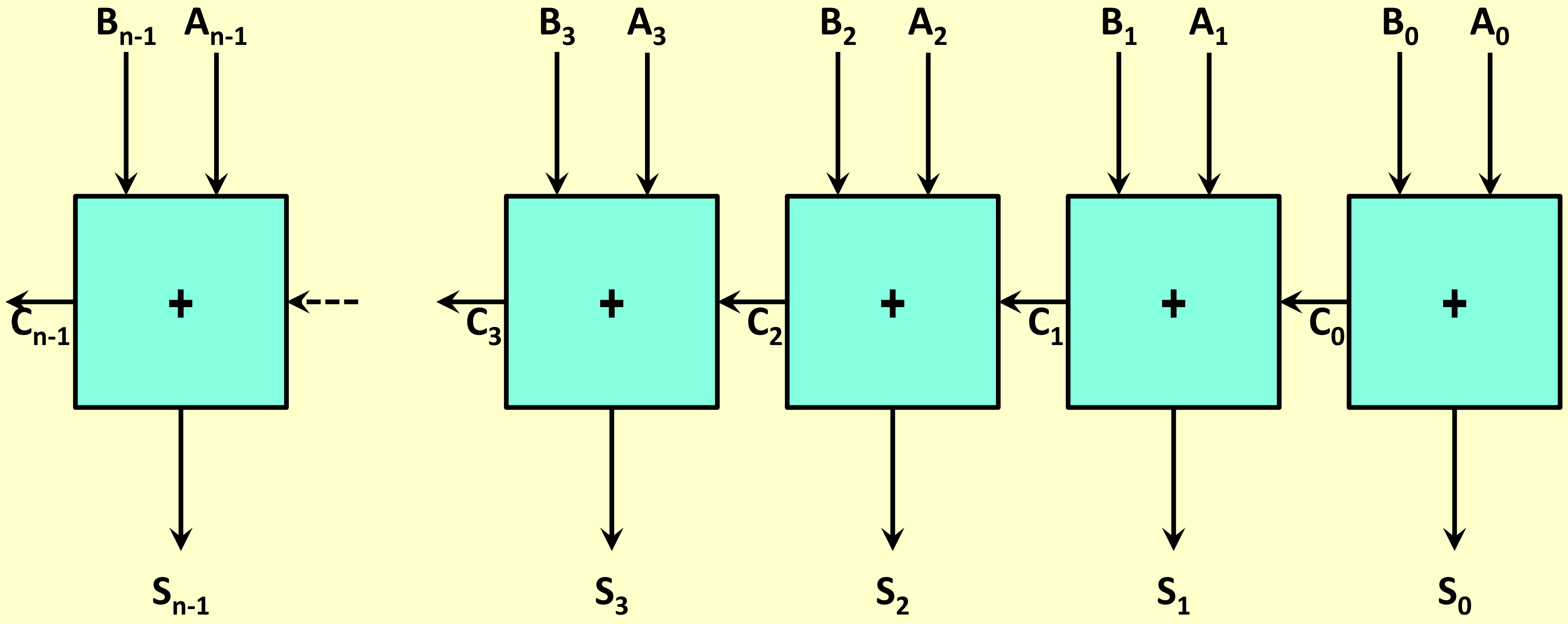
A	B	Carry In	Carry Out	Sum	
0	0	0	0	0	$0 + 0 + 0 = 00_2$
0	0	1	0	1	$0 + 0 + 1 = 01_2$
0	1	0	0	1	$0 + 1 + 0 = 01_2$
0	1	1	1	0	$0 + 1 + 1 = 10_2$
1	0	0	0	1	$1 + 0 + 0 = 01_2$
1	0	1	1	0	$1 + 0 + 1 = 10_2$
1	1	0	1	0	$1 + 1 + 0 = 10_2$
1	1	1	1	1	$1 + 1 + 1 = 11_2$

Single-bit adder (cont.)

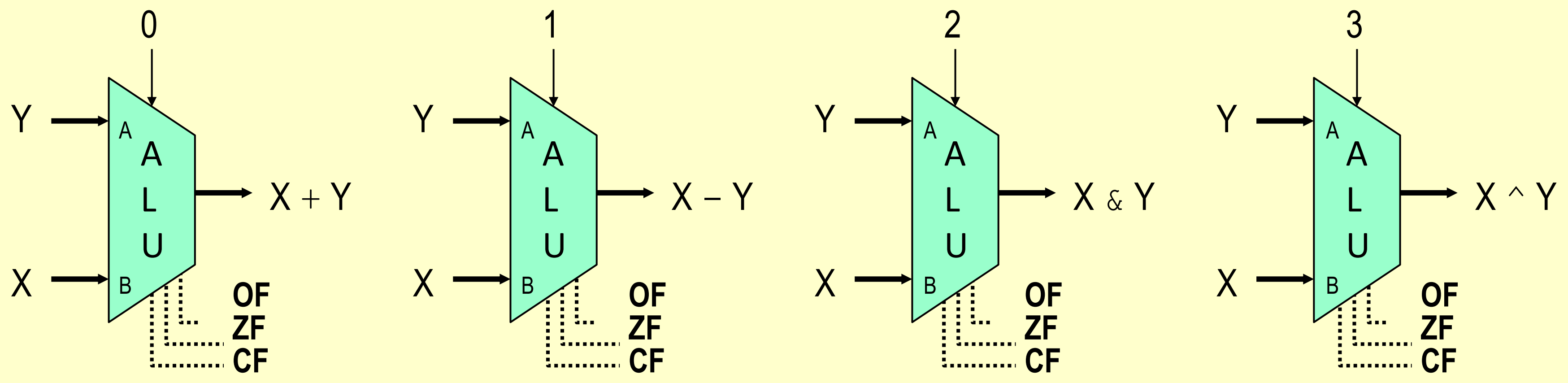
A	B	Carry In	Carry Out	Sum	
0	0	0	0	0	$0 + 0 + 0 = 00_2$
0	0	1	0	1	$0 + 0 + 1 = 01_2$
0	1	0	0	1	$0 + 1 + 0 = 01_2$
0	1	1	1	0	$0 + 1 + 1 = 10_2$
1	0	0	0	1	$1 + 0 + 0 = 01_2$
1	0	1	1	0	$1 + 0 + 1 = 10_2$
1	1	0	1	0	$1 + 1 + 0 = 10_2$
1	1	1	1	1	$1 + 1 + 1 = 11_2$



Multi-bit adder



Arithmetic Logic Unit (single-bit example)



- Combinational logic
  - Continuously responding to inputs
- Control signal selects function computed
  - Corresponding to 4 arithmetic/logical operations in Y86
- Also computes values for condition codes

Figure 4.15

Modern Arithmetic-Logic Unit (ALU)

- **Combines**
  - Add
  - Subtract
  - And
  - Or
  - Not
  - Xor
  - Equality
  - <
  - >
  - <<
  - >>
  - …

- **Outputs**
  - Result
  - CF — Carry flag
  - ZF — Zero flag
  - SF — Sign flag
  - OF — Overflow flag

Result developed within one cycle (300 ps)

Conspicuously absent:– multiplication!

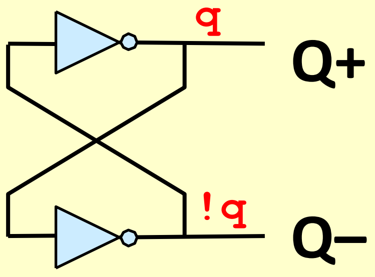
Questions?

Today

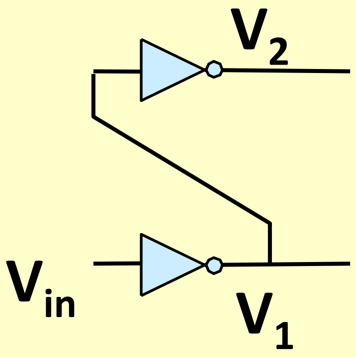
- Before electronic computers
- Logic and gates
- **Latches and Registers**

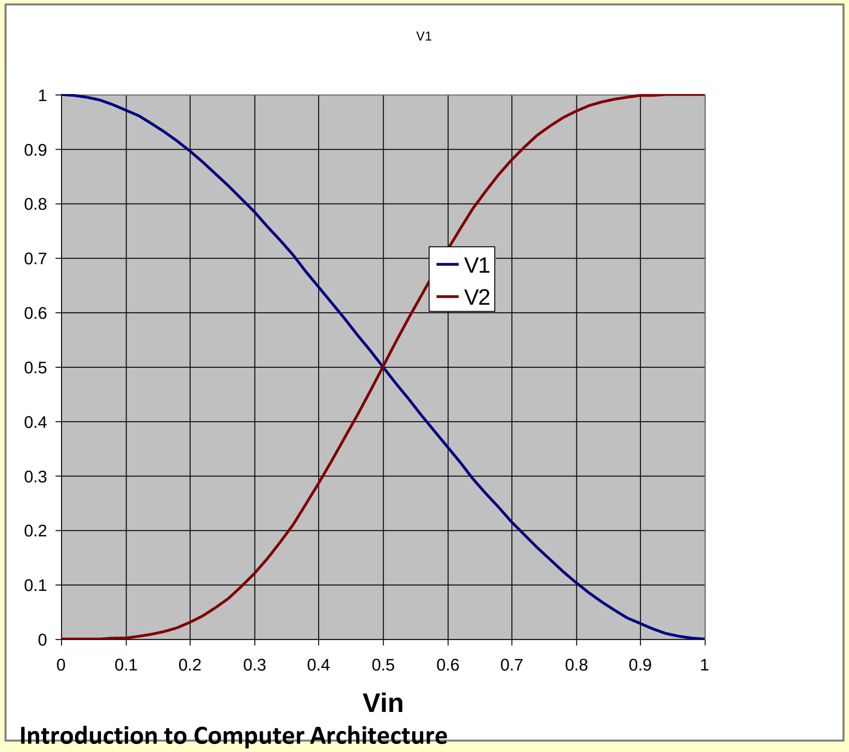
Storing 1 Bit

Bistable Element



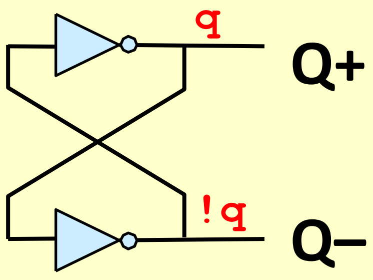
q = 0 or 1





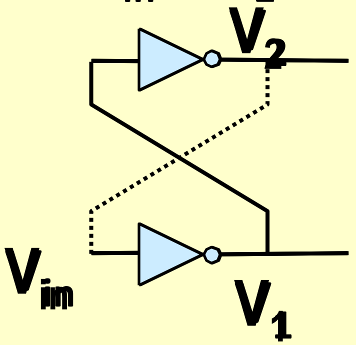
Storing 1 Bit (continued)

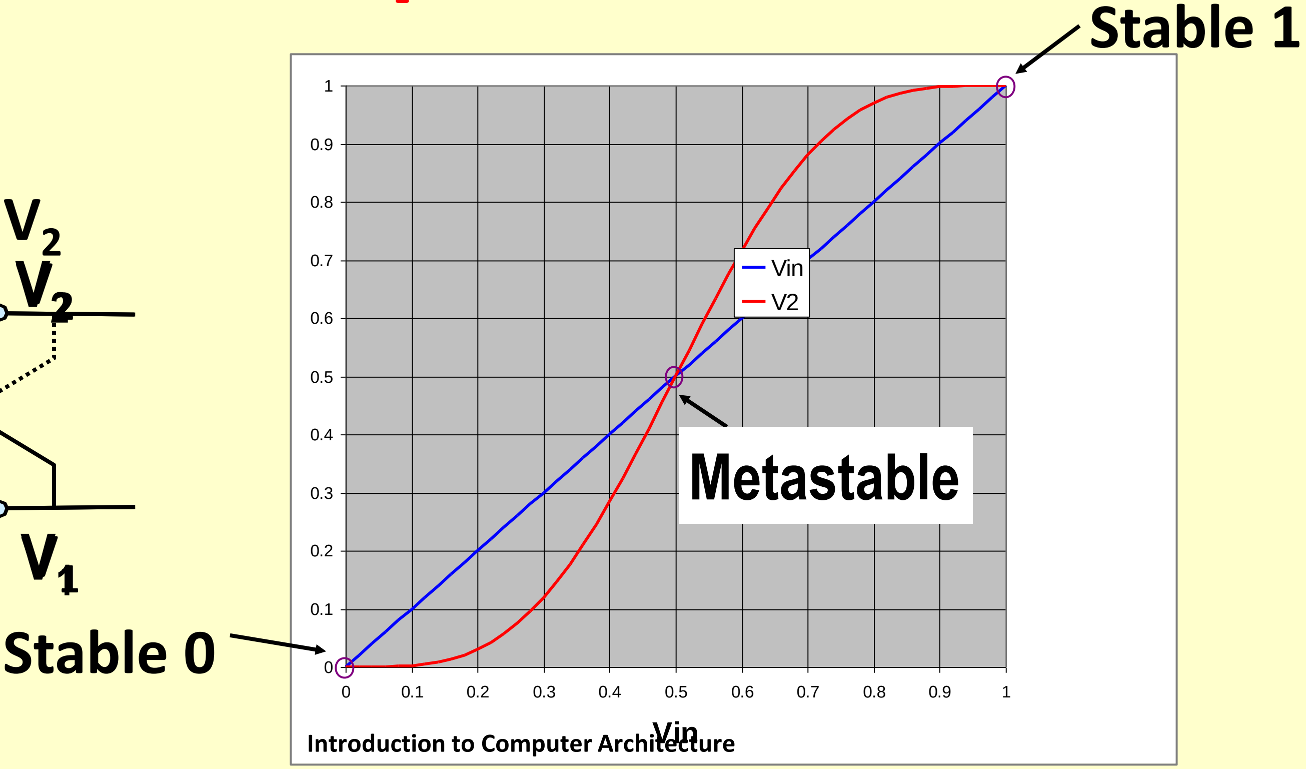
Bistable Element



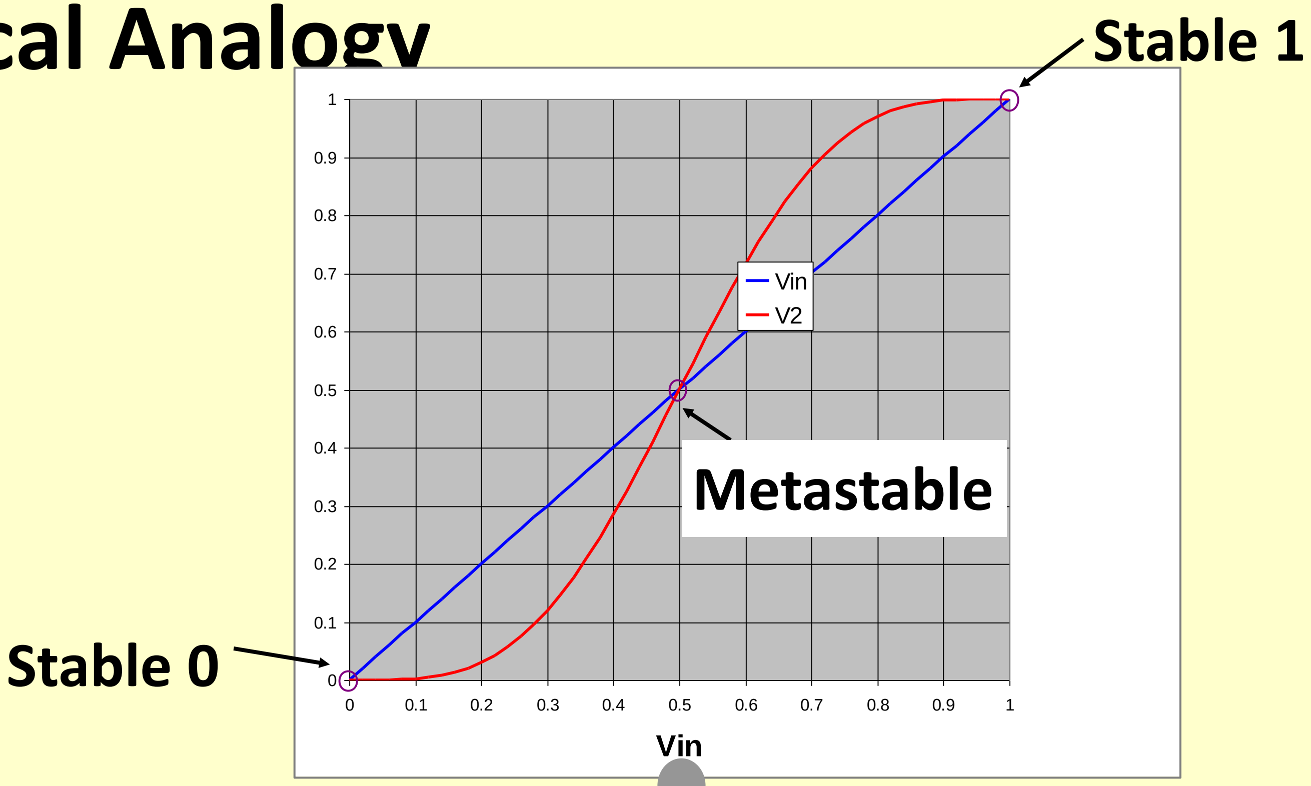
q = 0 or 1

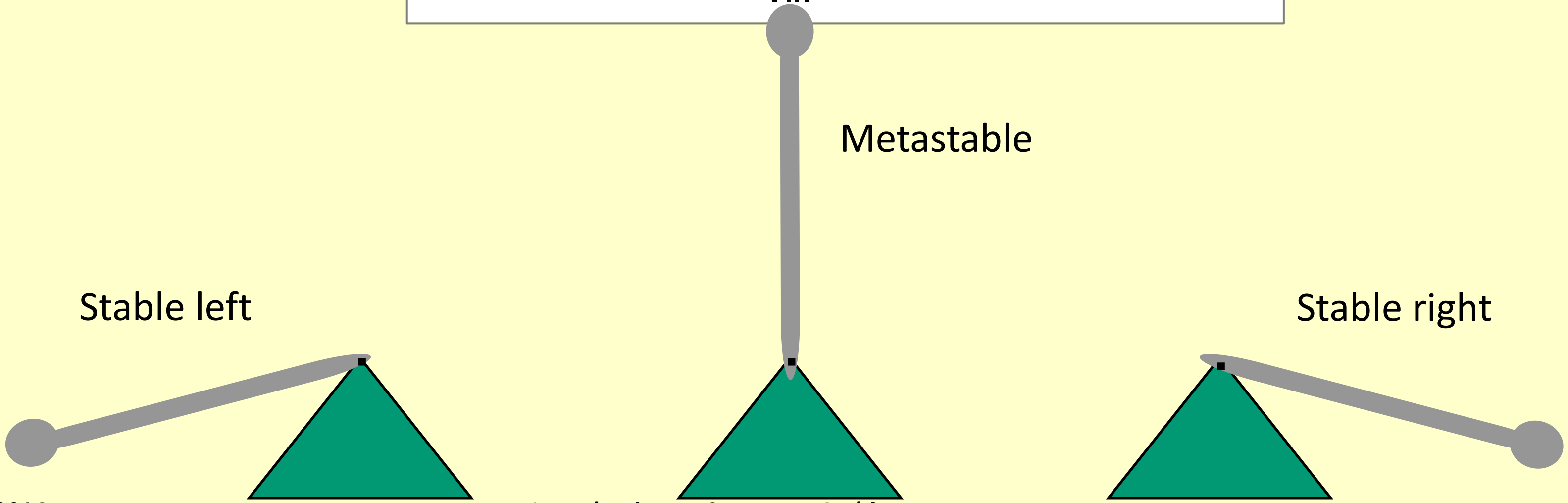
$V_{in} = V_2$





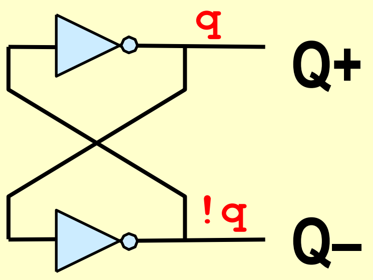
Physical Analogy





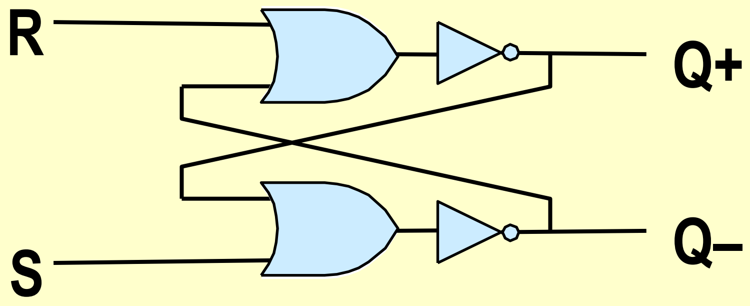
Storing and Accessing 1 Bit

Bistable Element

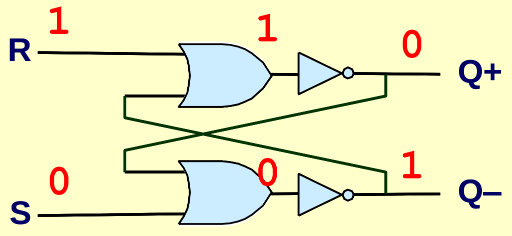


q = 0 or 1

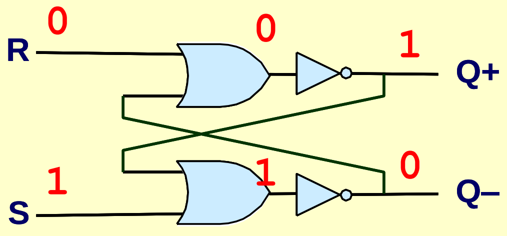
R-S Latch



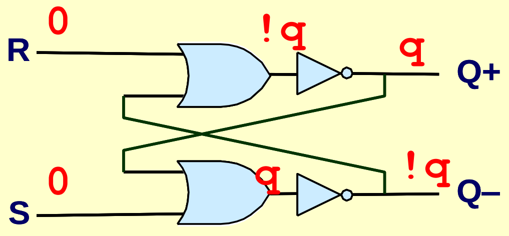
Resetting



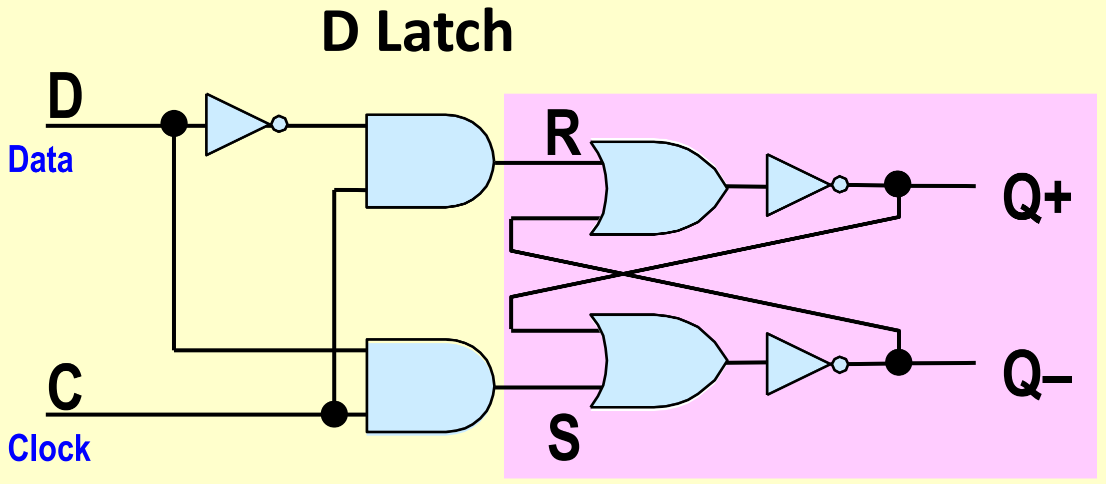
Setting



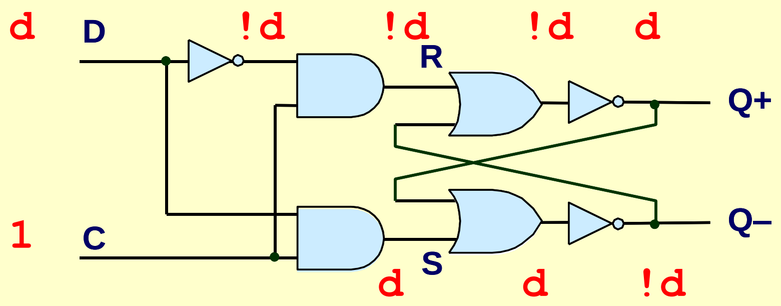
Storing



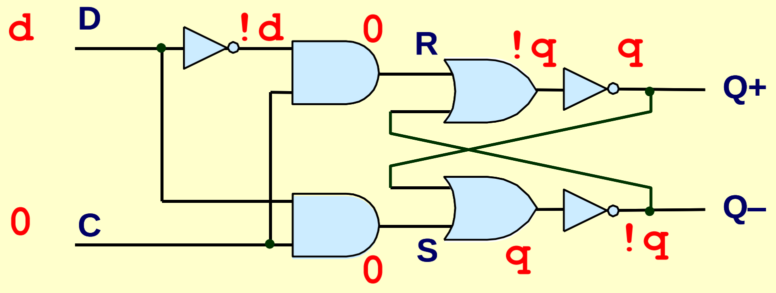
1-Bit Latch



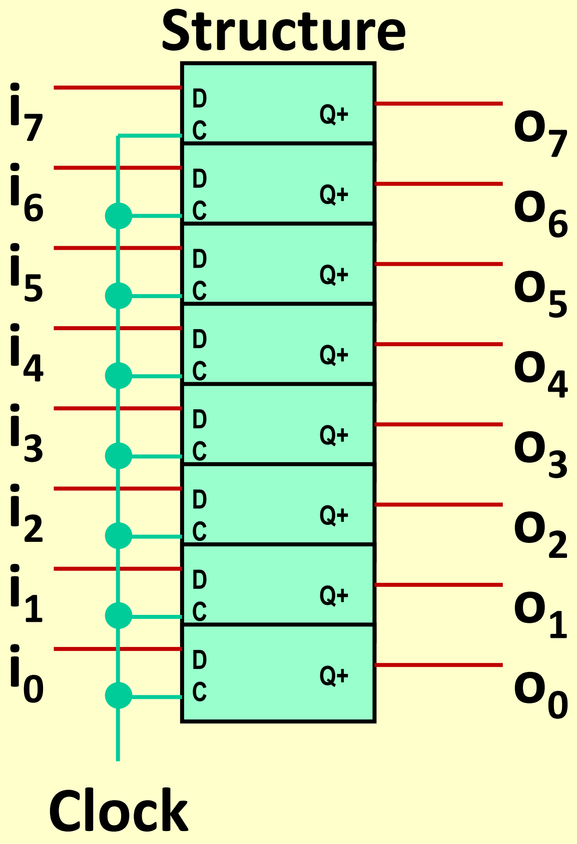
Latching

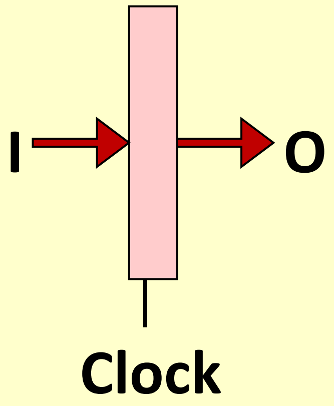


Storing



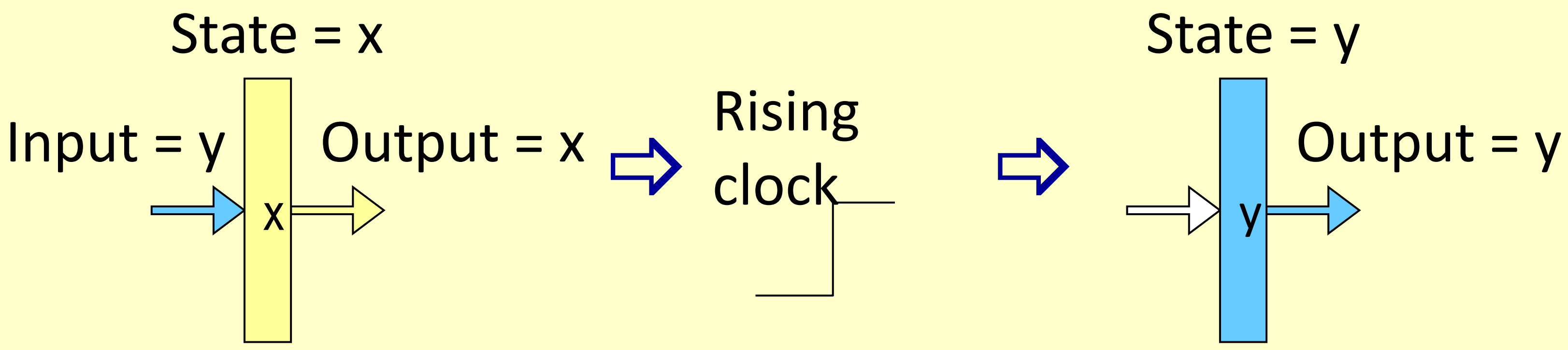
Register





- Stores word of data
  - Different from *program registers* seen in assembly code
- Collection of edge-triggered latches
- Loads input on rising edge of clock

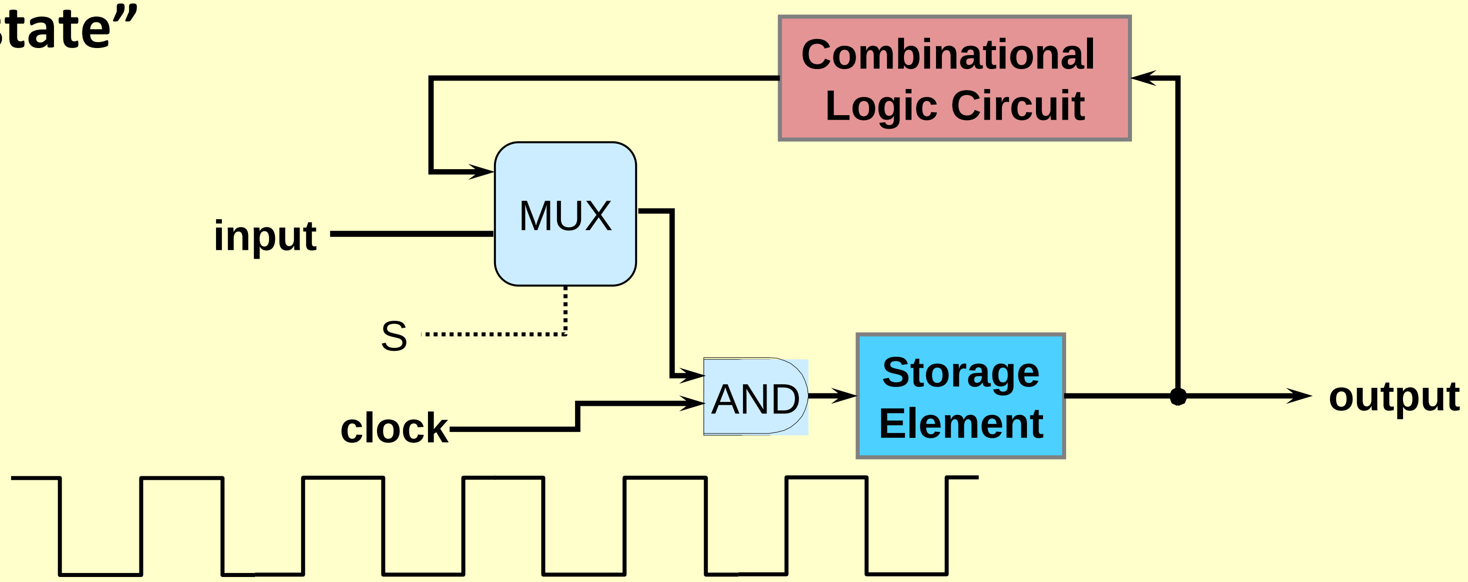
Register Operation



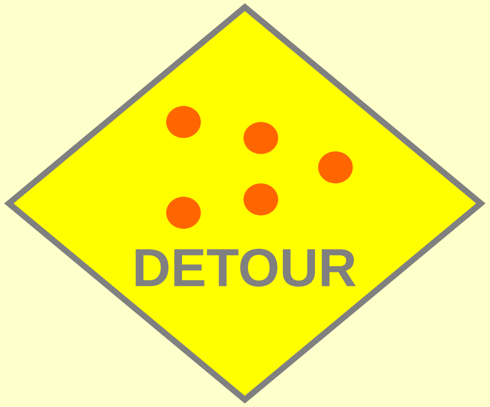
- Stores data bits
- For most of time acts as barrier between input and output
- As clock rises, loads input

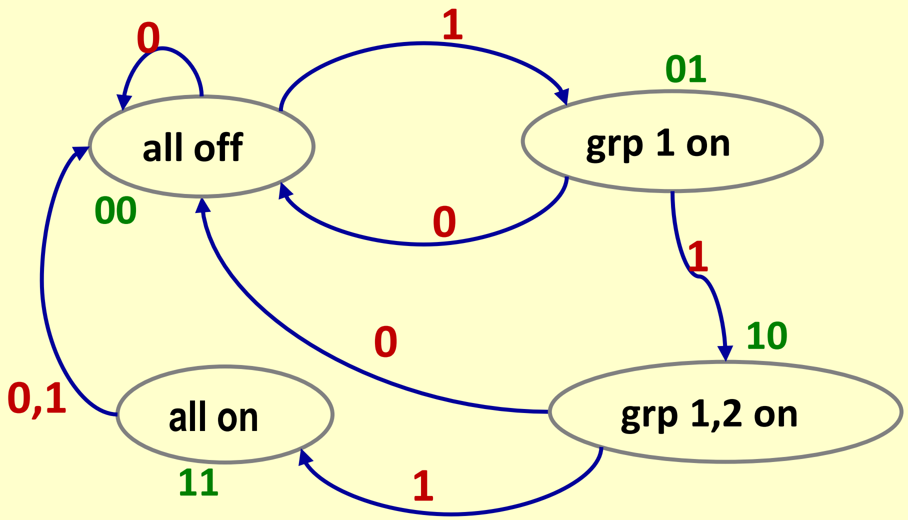
(Finite) State Machine

- A register or latch of *n* bits representing the "state" of the circuit
- An acyclic network of combinatorial logic to compute a new value of *n* bits based on the existing value of *n* bits
- A clock signal to effect the update of the "state" to a new "state"

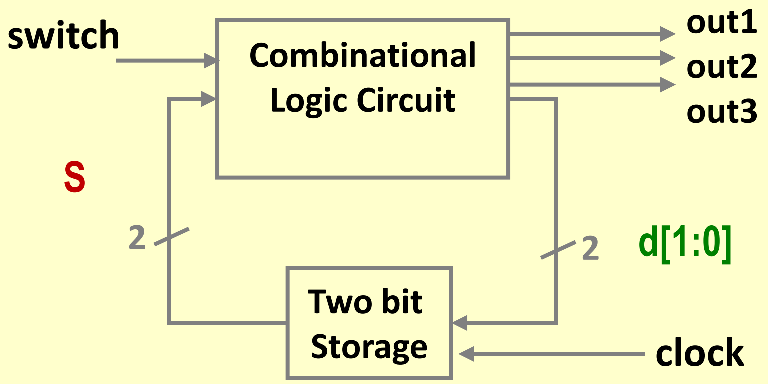


Finite State Machine Example

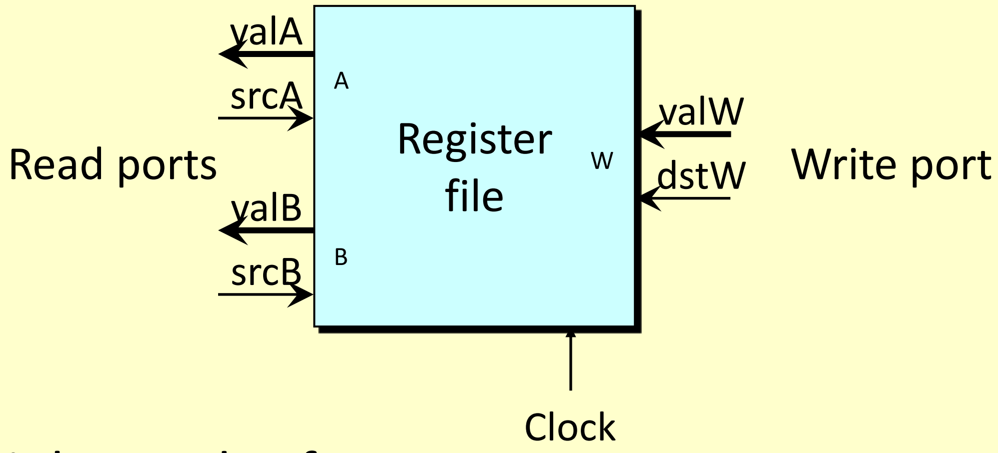




- Three groups of lights to be lit in a sequence: group 1 on, groups 1 & 2 on, all groups on, all off.
- The lights are on only if the main switch is on.
- <u>Four states:</u> so we need two bits to identify each state.



Register File



- Stores multiple words of memory
  - Address input specifies which word to read or write
- Register file
  - Holds values of program registers
  - **`%eax`**, **`%esp`**, etc.
  - Register identifier serves as address
    - ID 15 (0xF) implies no read or write performed
- Multiple Ports
  - Can read and/or write multiple words in one cycle
    - Each has separate address and data input/output

Summary

- **Data values stored as bits**
  - On wires, in memory cells, etc.
- ***Gates* are logic elements that combine values of bits to produce other bits**
  - And, Or, Xor, addition, subtraction, comparison, etc.
- **Latches capture bit values on wires and keep them until reset**
  - So long as power stays on
- **Setting of latches is triggered by a clock, which allow data into the latches only when the results of combinatorial logic elements has stabilized.**

Reading Assignment: §4.2

Questions?