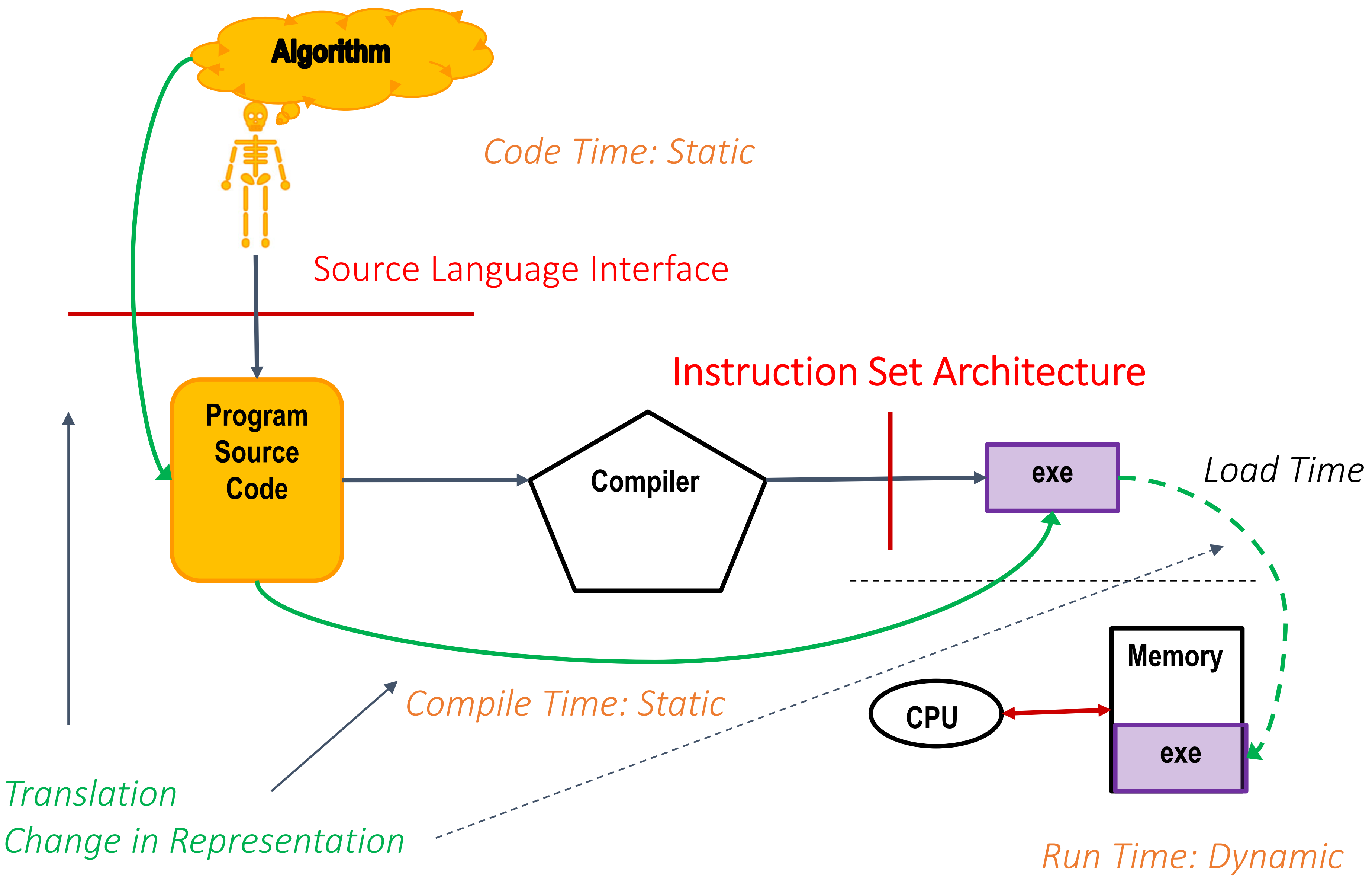
Boolean Circuits

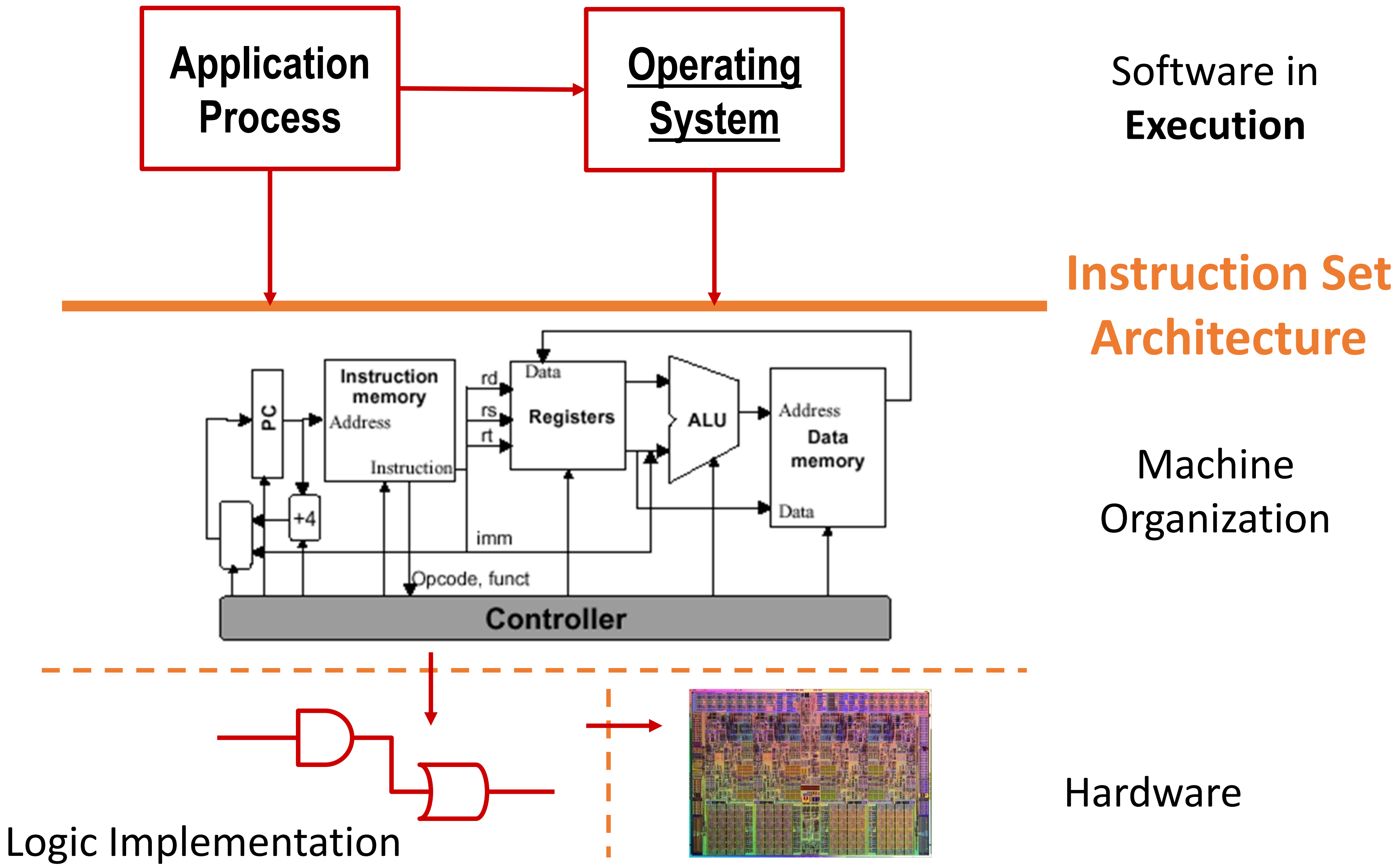
CSE 410

Lecture 7

The Course So Far



What This Course is About



Lecture Outline

- Boolean Functions, Logic Gates, and Boolean Circuits
- Example Combinational Components
  - Adder
  - Multiplexor
- Sequential Component
  - example gated d latch

Boolean Functions

- A function where the inputs and output have value 0 or 1
- Represented by a truth table
  - 0 is false
  - 1 is true
- There are 2^{2^k} distinct Boolean functions with k inputs

Not $(\neg x)$

Input	Output
0	1
1	0

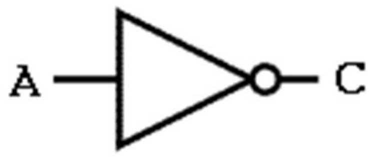
Or $(X + Y)$

Inputs		Output
0	0	0
0	1	1
1	0	1
1	1	1

AND/ OR/ NOT gates

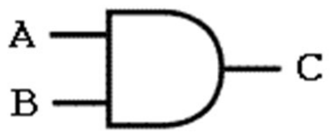
- Digital circuits are built out of digital gates
- Each gate implements some logic function

NOT



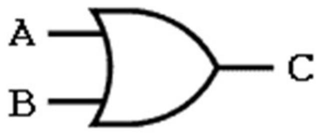
Input	Output
A	C
0	1
1	0

AND



Inputs		Output
A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

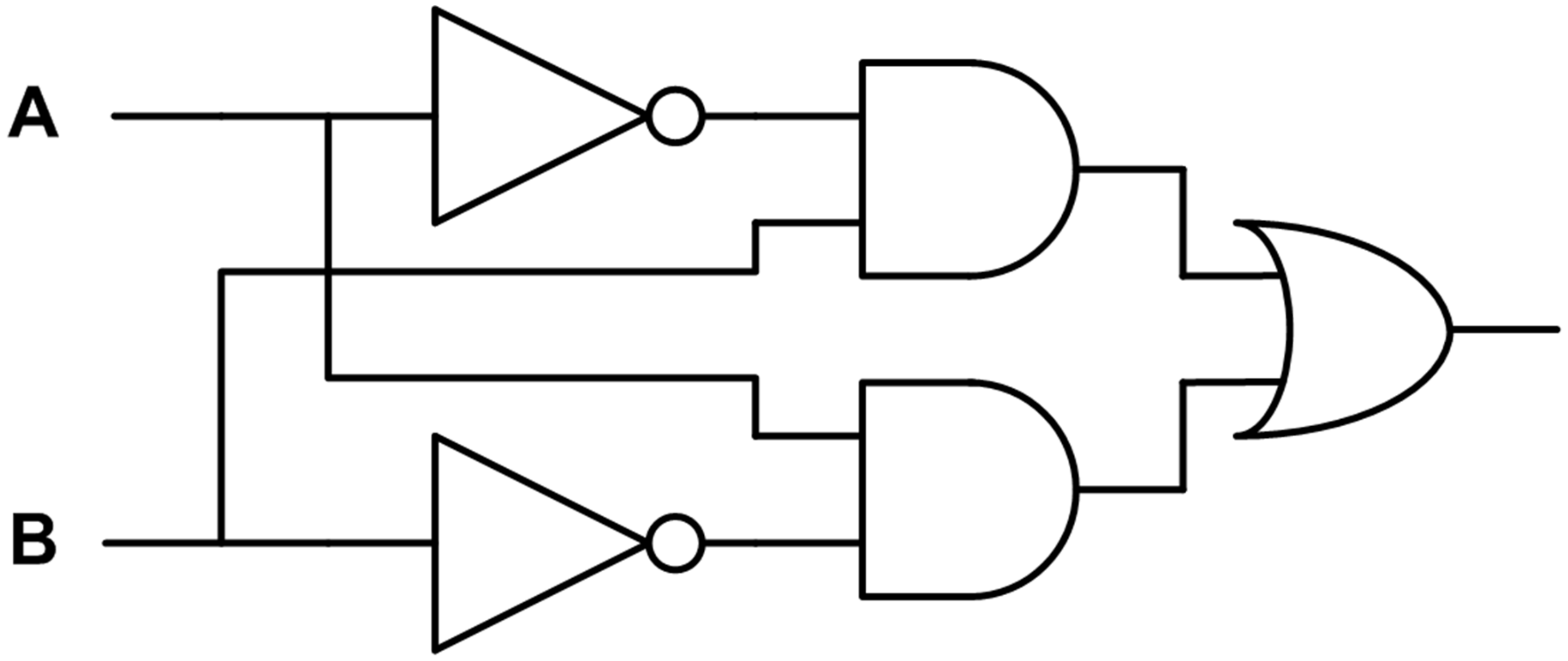
OR



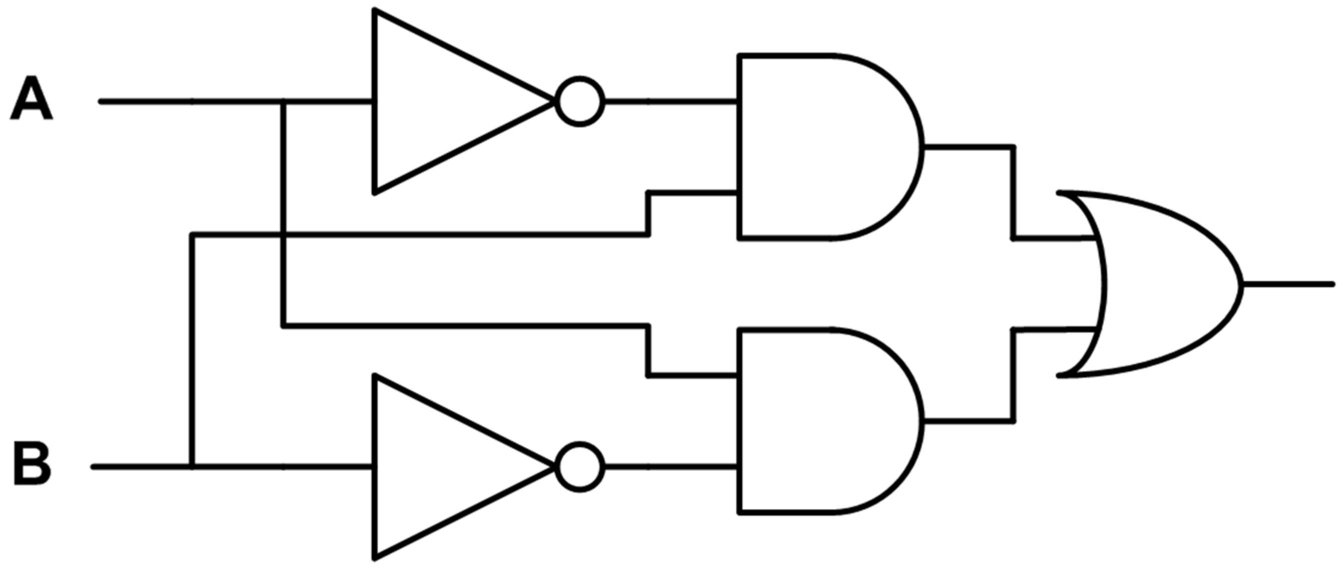
Inputs		Output
A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

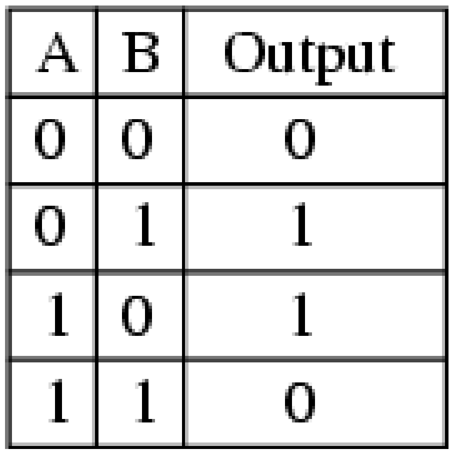
Combinational Circuit

- We can connect these components together into circuits
- What function does this two-input circuit compute?



Example Circuit





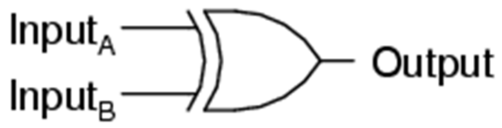
A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0

$(\neg A \wedge B) \vee (A \wedge \neg B)$

Exclusive Or

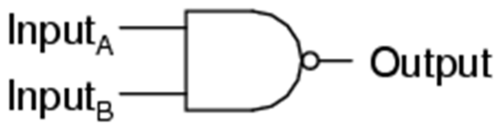
Other Gates

Exclusive-OR gate



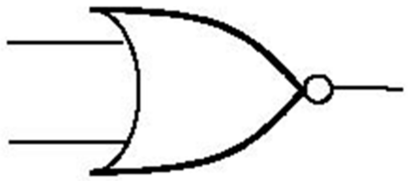
A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0

NAND gate



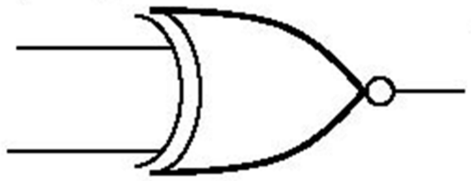
A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0

NOR



A	B	Output
0	0	1
0	1	0
1	0	0
1	1	0

Exclusive NOR (XNOR)



A	B	Output
0	0	1
0	1	0
1	0	0
1	1	1

Computing: Binary Addition

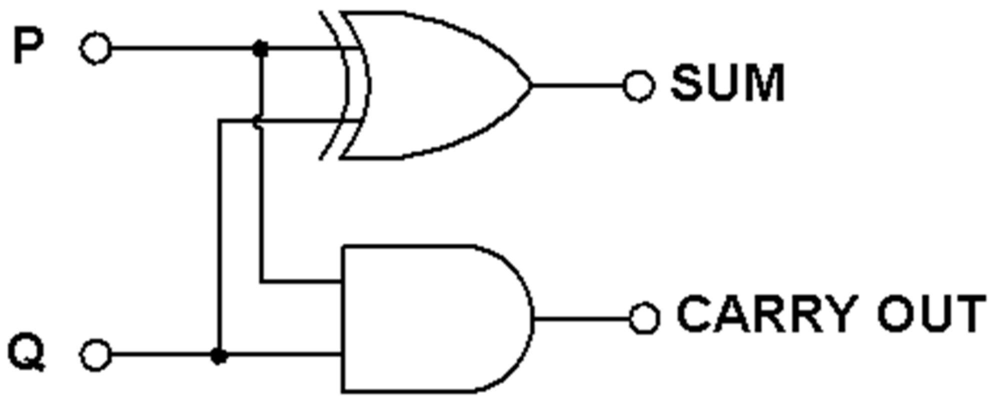
Examples:

$$\begin{array}{r} 0 \\ \underline{1} \\ 01 \end{array} \qquad \begin{array}{r} 1 \\ \underline{\ 1} \\ 10 \end{array}$$

Inputs		Outputs	
P	Q	CO	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

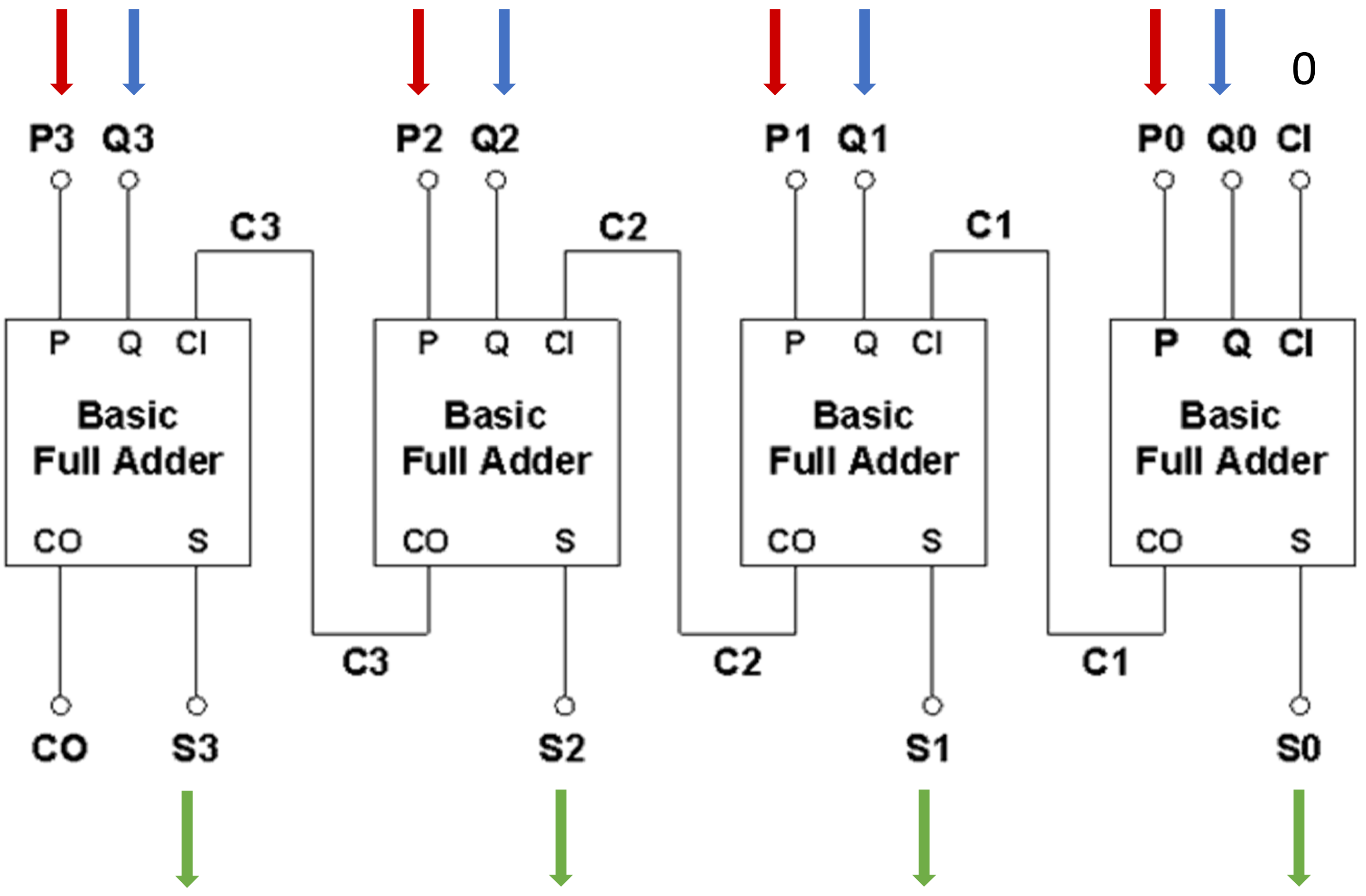
1-bit (half) adder

Inputs		Outputs	
P	Q	CO	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

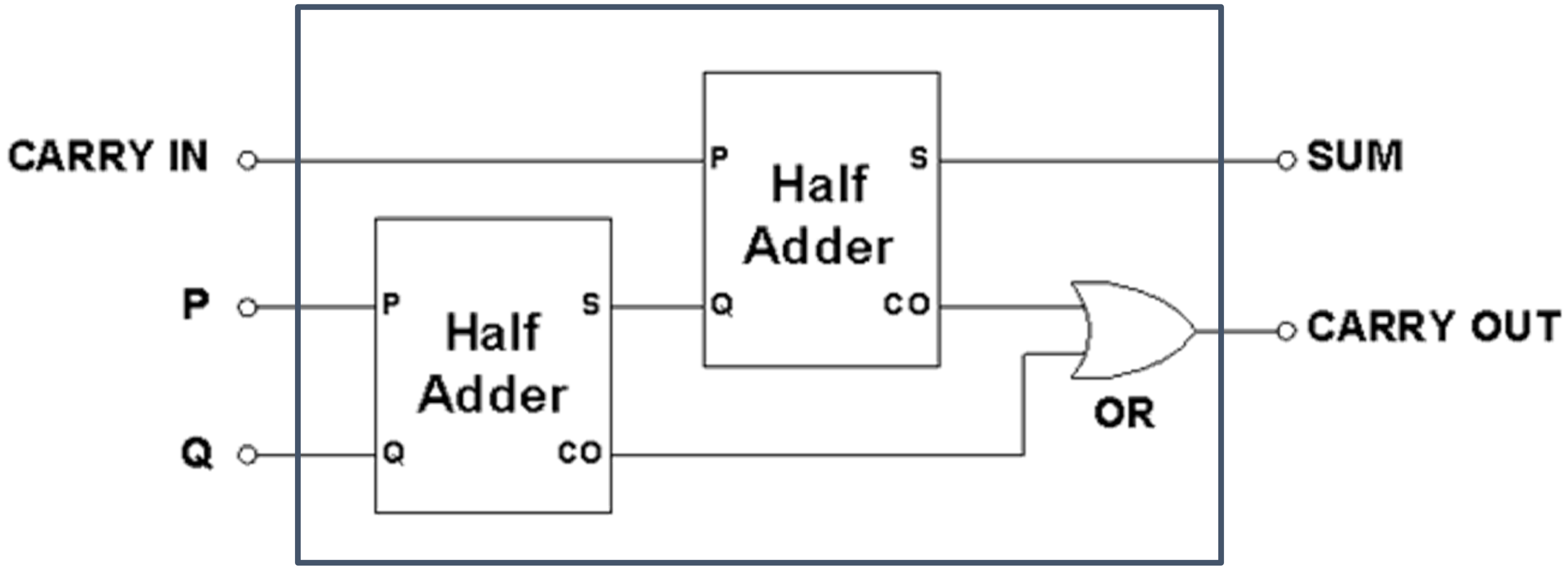


4-bit adder

If we had as a component a 3-input 1-bit adder (a “full adder”) we could use it to build an n-bit adder

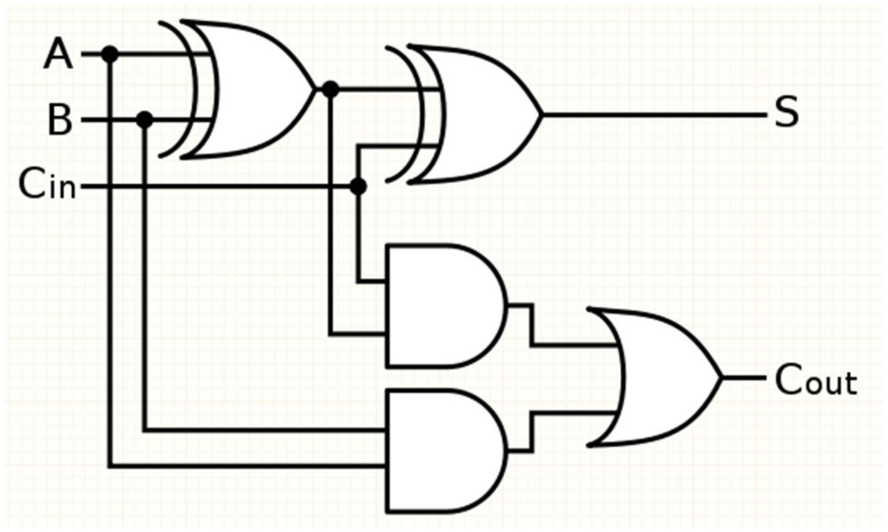


1-bit full adder

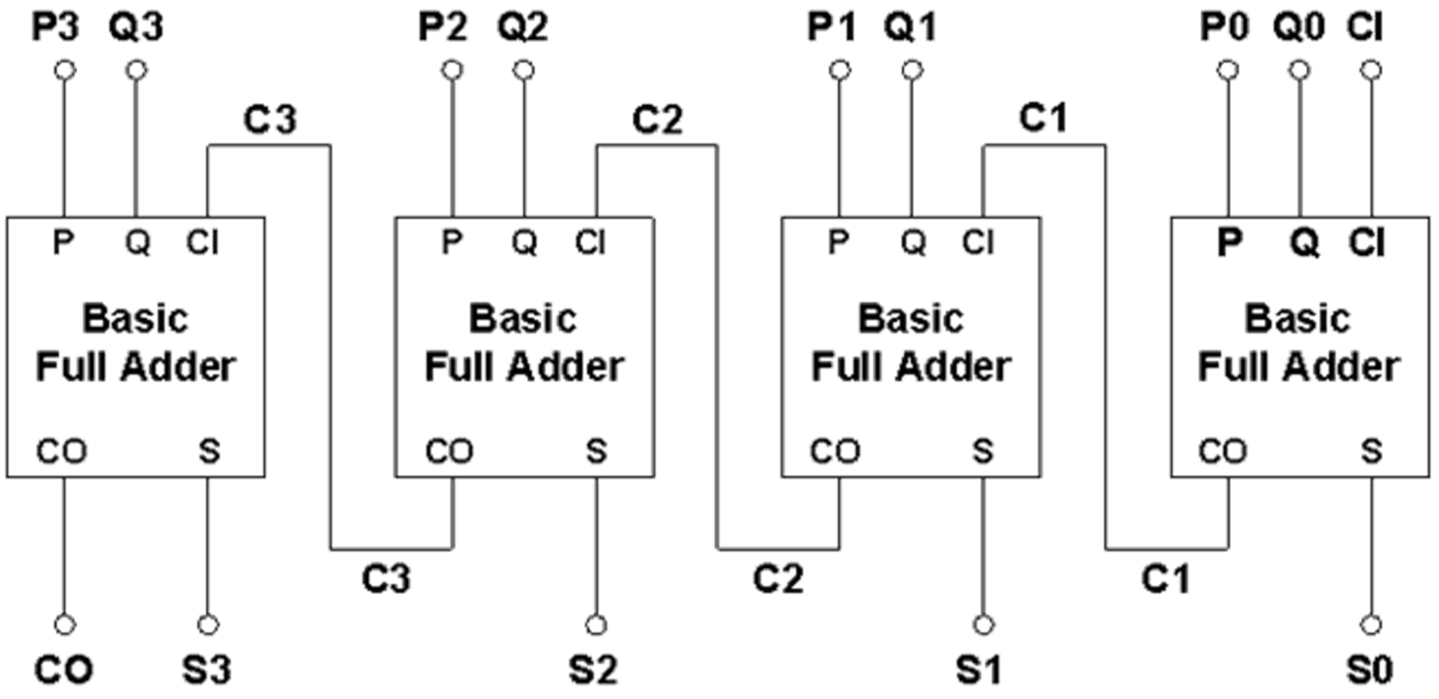


Direct Implementation in Gates

Input			Output	
A	B	Cin	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



4-bit full (ripple carry) adder

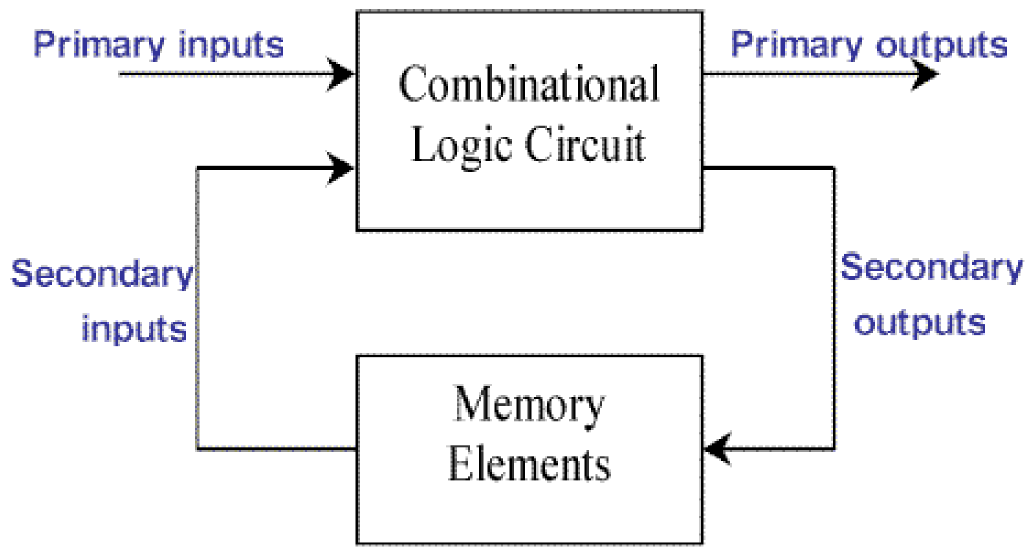


Sequential component: storage

- The outputs of combinational components is a function of their inputs (after some delay)
- Combinational circuits can have no loops
  - Loops create instability

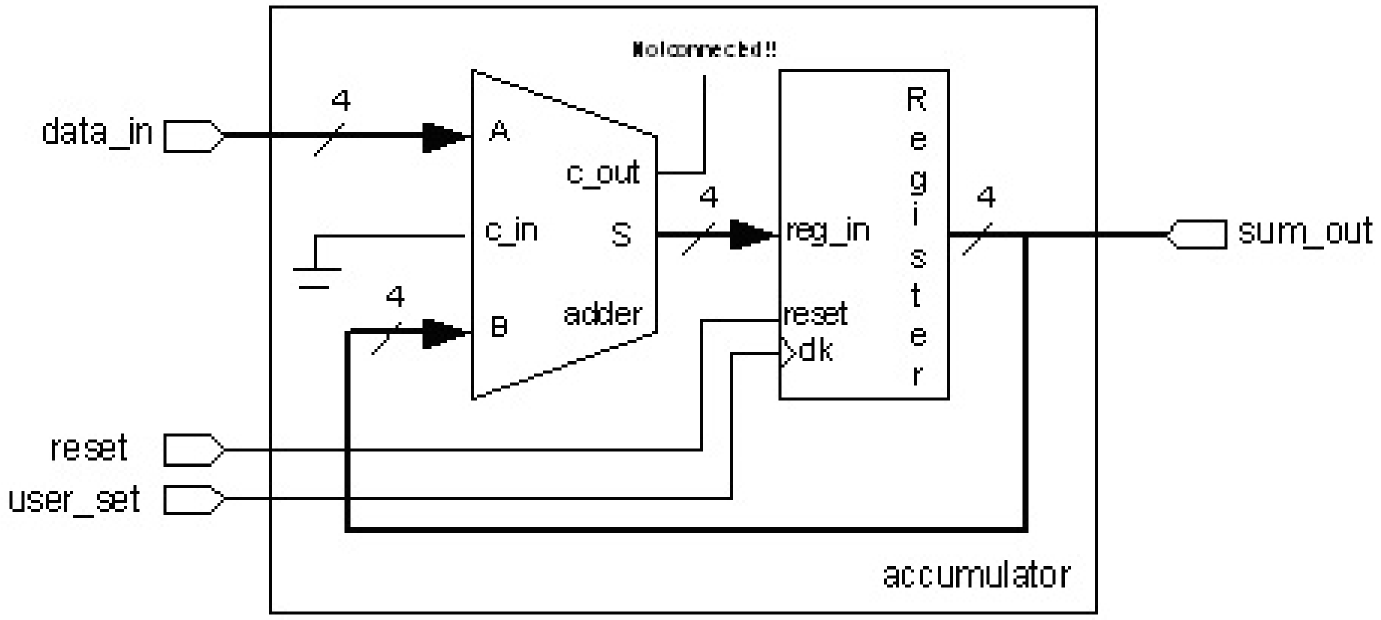
- Sequential circuits can have loops
- How?
  - Components whose output is stable even when inputs are changing
  - Storage

Sequential Circuit: Operation



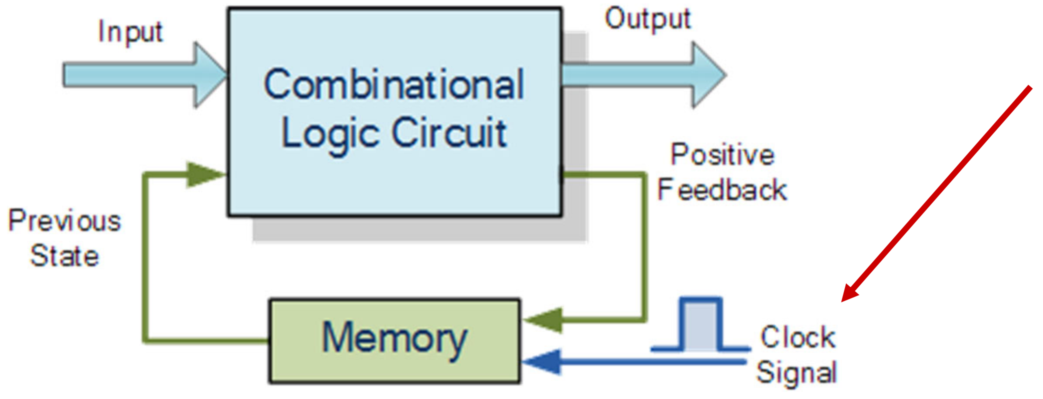
- At time n the memory elements have some values
  - The combinational circuit has “settled” and its output are stable (unchanging)
  - If we update the memory elements values, though, the outputs of the combinational circuit change

Example: Accumulator Circuit



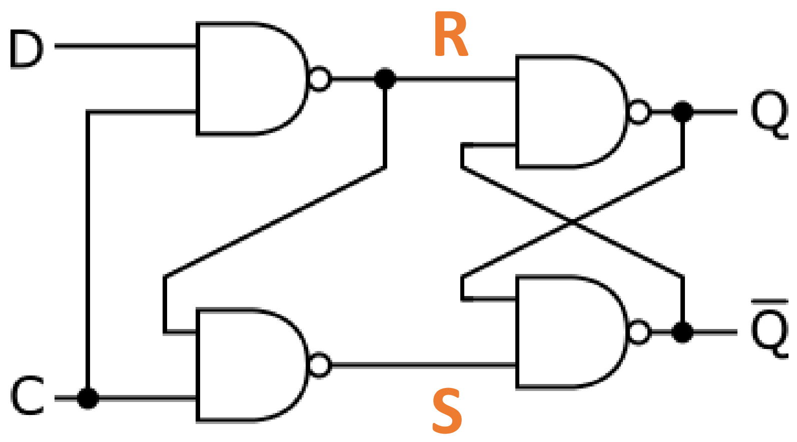
- Here the inputs are 4 bits wide ("/4")
- If data_in == 0001 and the register holds 0010, the output of the adder will eventually be 0011
- When we update the register, the adder will eventually output 0011

Sequential Circuit: Clocks



- It takes time for the combinational circuit to "settle" when its inputs change
- We don't want to update the memory component while the combinational circuit is settling
- We control the rate of update using a "clock signal"
  - This is what's behind processor specifications like "3.6GHz i5-8600K" versus "2.8GHz i5-8400"
    - Warning: A 4.0GHz processor is very unlikely to be anywhere near twice as fast as a 2GHz processor

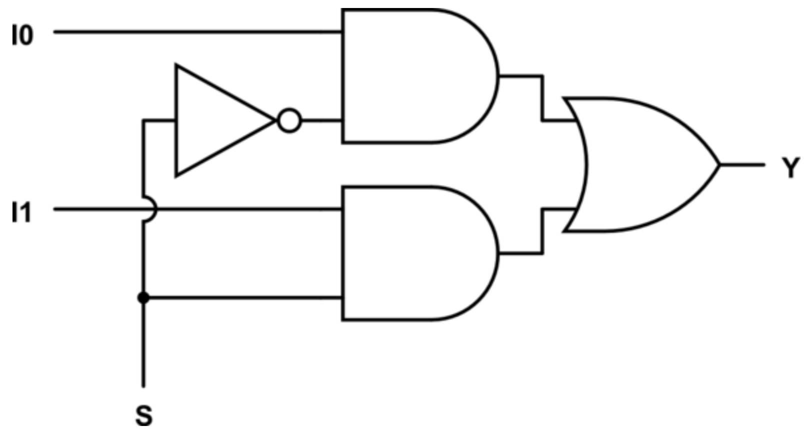
Implementing sequential components: the gated d-latch



- Component stores 1 bit, and advertises both it's value (Q) and the negation of its value ($\bar{Q}$)
- When C(lock)=1 the output Q records the value of D
  - if D=1 then R=0 and S=1. R=0 makes Q=1. Q=1 makes $\bar{Q}$= 0.
  - if D=0 then R=1 and S=0. S=0 makes $\bar{Q}$=1, which makes Q=0.
- When C=0 the output ignores the value of D
  - both R and S are 1. If Q=1 then $\bar{Q}$ is 0 – no change. If Q=0, then $\bar{Q}$ is 1 – no change.

Other combinational components

- Multiplexor – select one of two inputs

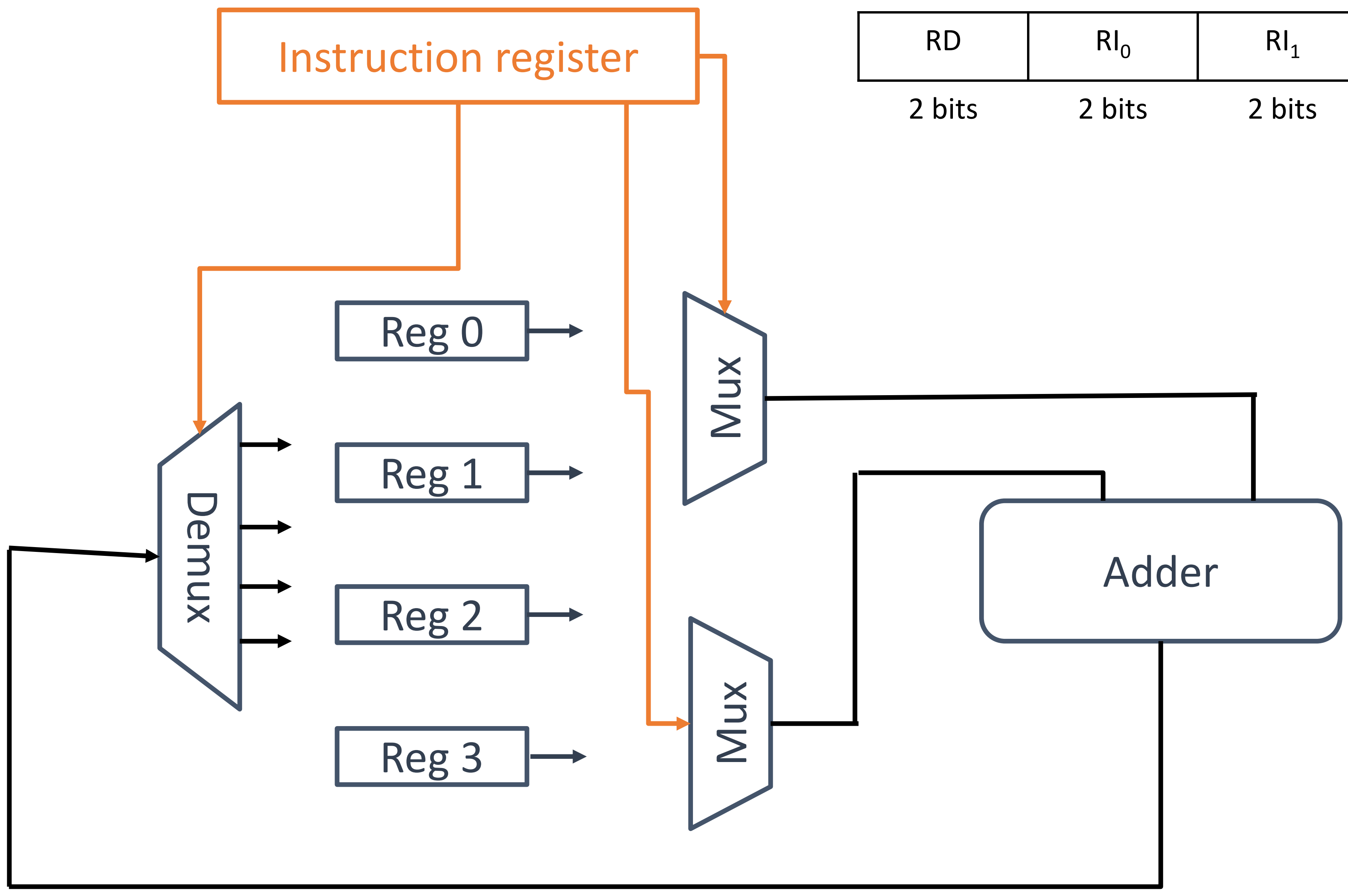


- We can expand this implementation by
  - Allowing the inputs (I0 and I1) to be n bits wide
  - Cascading the two-input multiplexor to make allow more inputs
    - Implies widening the selector input, S

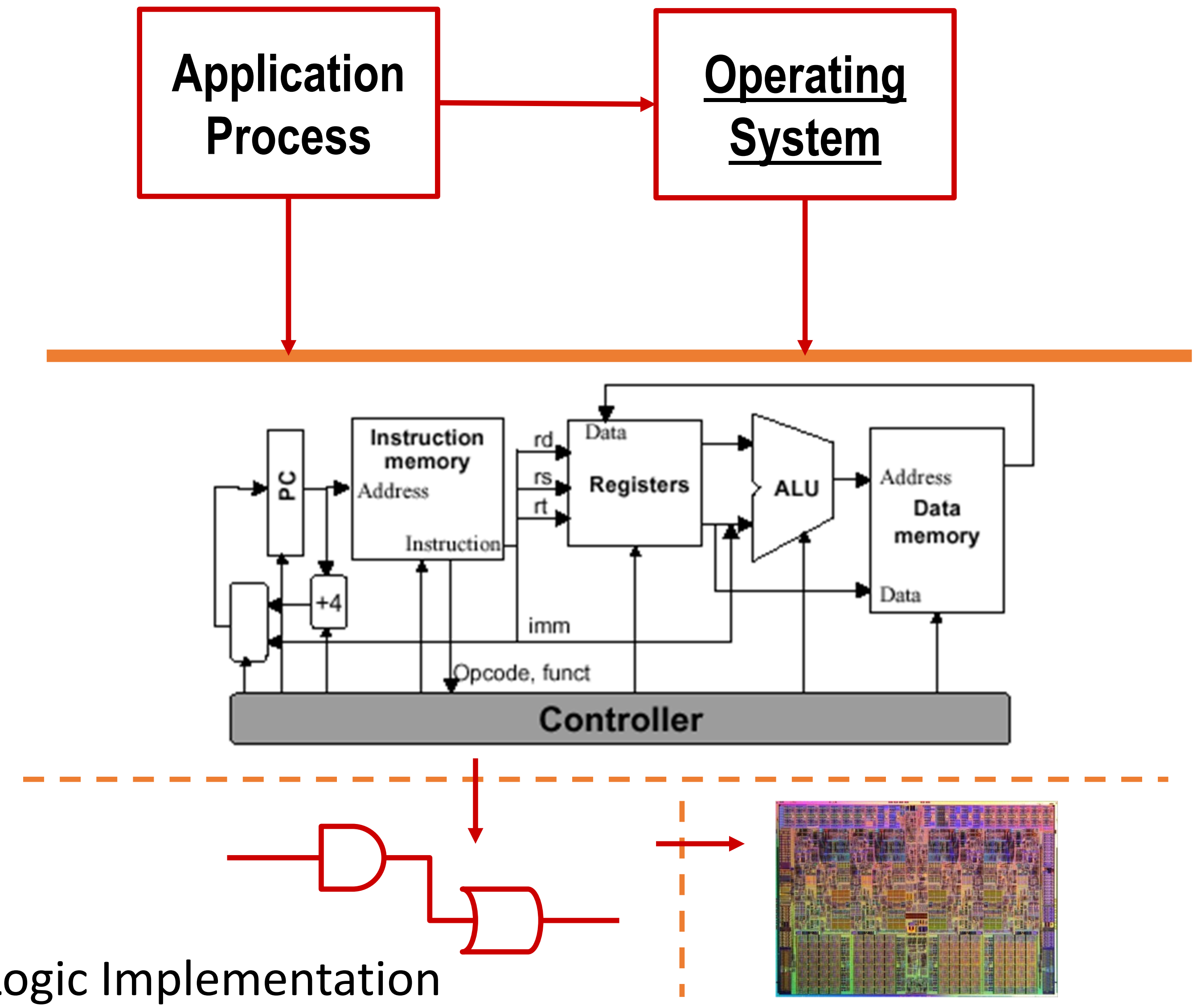
Looking ahead: CPU data path

Instruction Format

RD	RI_0	RI_1
2 bits	2 bits	2 bits



Lecture/Course Summary



Software in **Execution**

Instruction Set Architecture

Machine Organization

Hardware

Logic Implementation