



Combinatorial Logic Design

Multiplexers and ALUs

CS 64: Computer Organization and Design Logic
Lecture #13

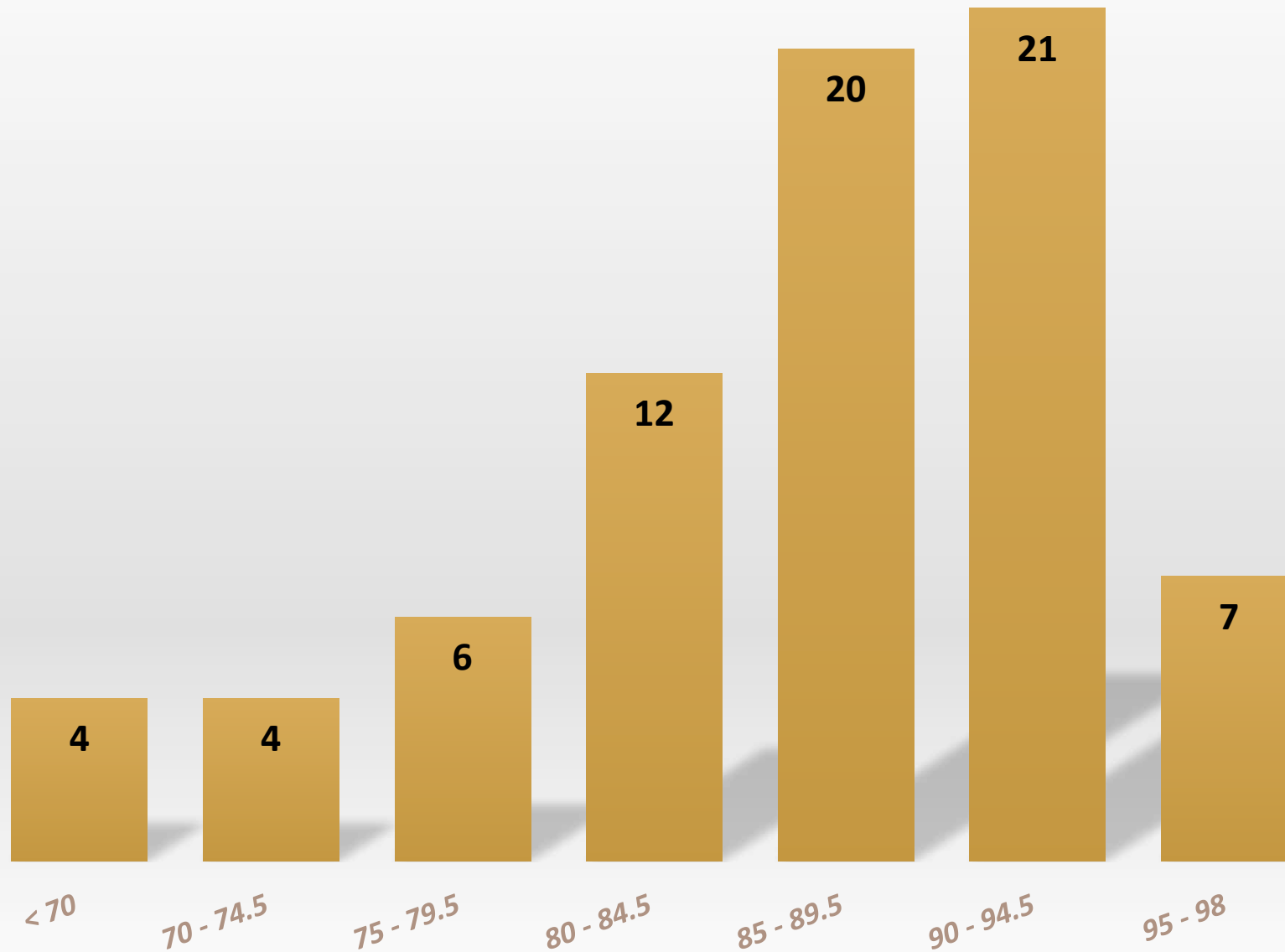
Ziad Matni
Dept. of Computer Science, UCSB

Administrative

- Re: Midterm Exam #2
 - Graded!

CS 64, Spring 18, Midterm#2 Exam

Average = 85.6%, Median = 87.5%



Lecture Outline

- Selection using Multiplexers
- Basic ALU Design

Multiplexer

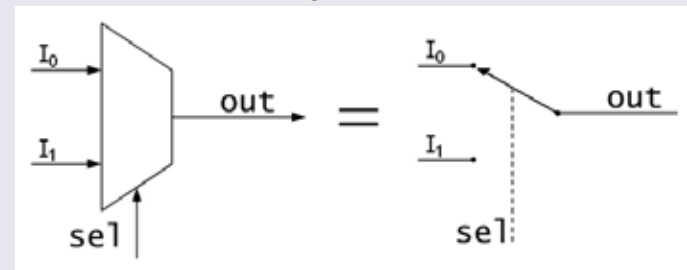
- A logical selector:
 - Select either input A or input B to be the output

```
// if s = 0, output is a
// if s = 1, output is b
int mux(int a, int b, int s)
{
    if (!s) return a;
    else return b;
}
```

Multiplexer

(Mux for short)

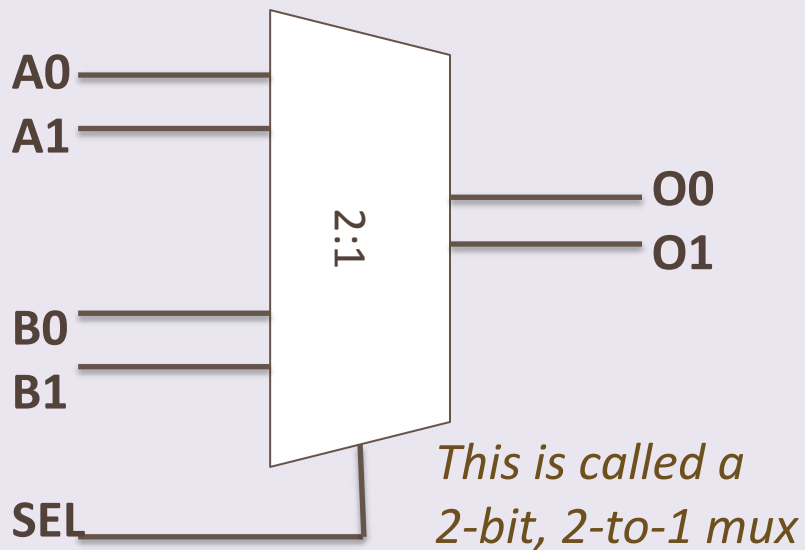
- Typically has 3 *groups of* inputs and 1 output
 - IN: 2 data , 1 select
 - OUT: 1 data



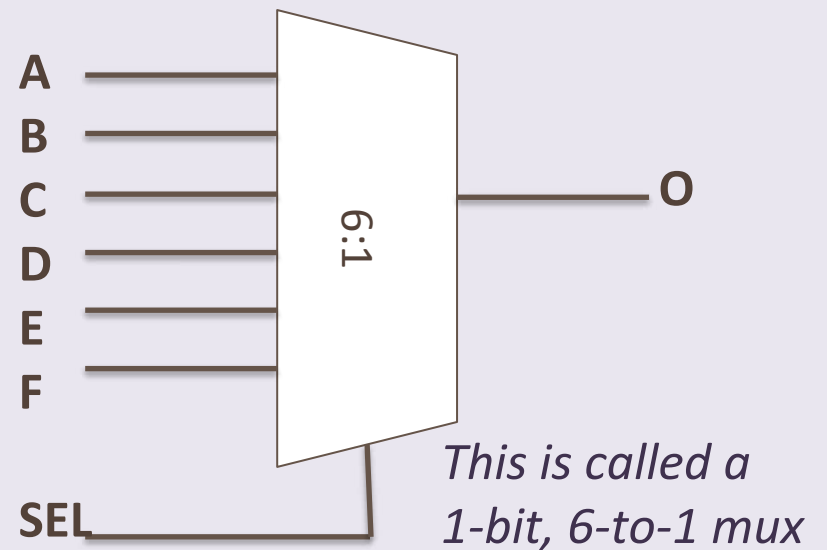
- 1 of the input data lines gets selected to become the output, based on the 3rd (select) input
 - If “Sel” = 0, then I_0 gets to be the output
 - If “Sel” = 1, then I_1 gets to be the output
- The opposite of a Mux is called a **Demultiplexer** (or **Demux**)

Mux Configurations

Muxes can have I/O that are multiple bits

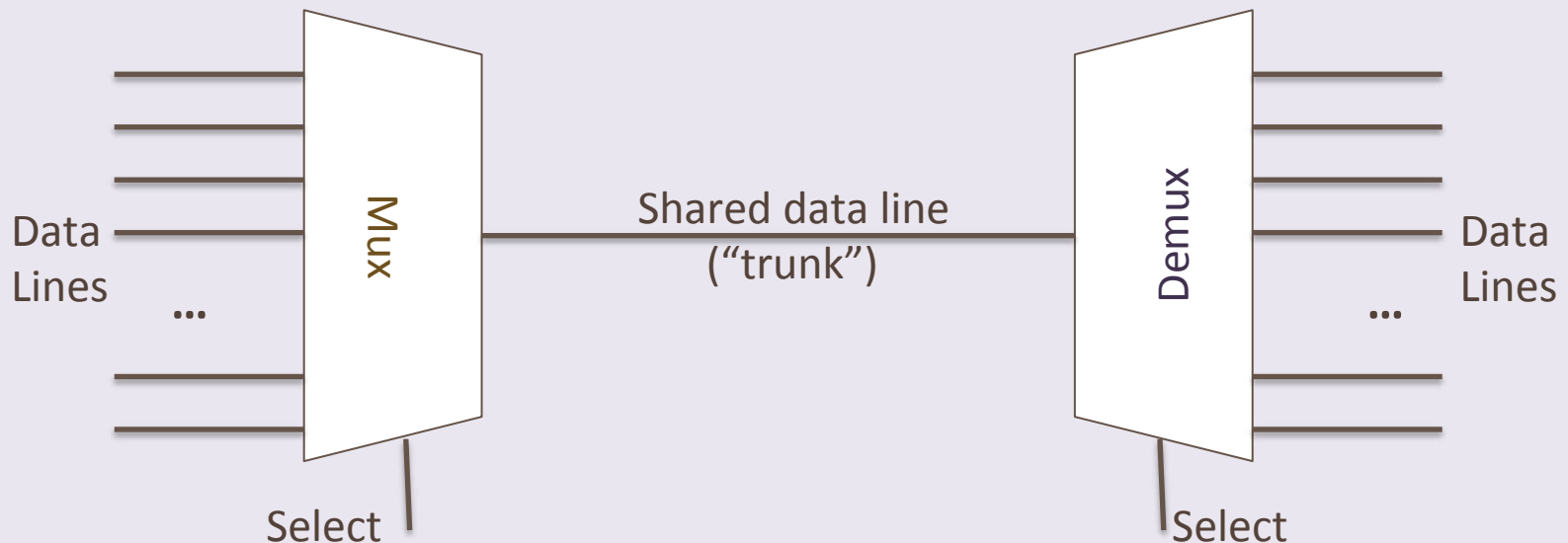


Or they can have more than two data inputs



The Use of Multiplexers

- Makes it possible for several signals (variables) to share one resource
 - Very commonly used in data communication lines



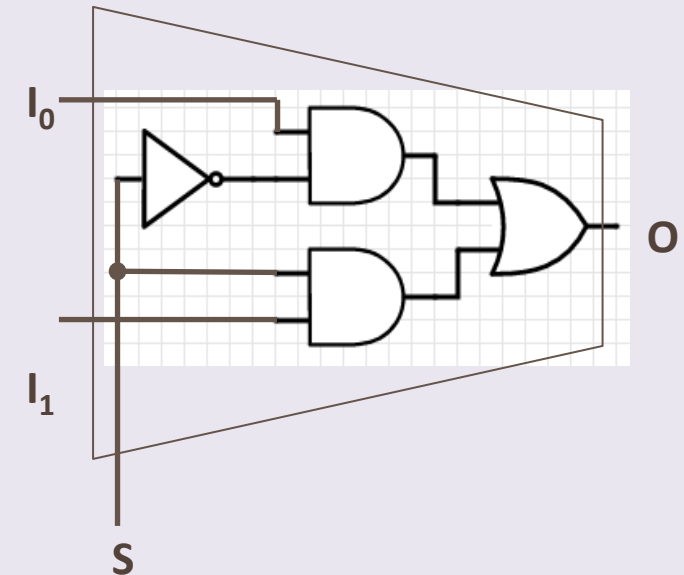
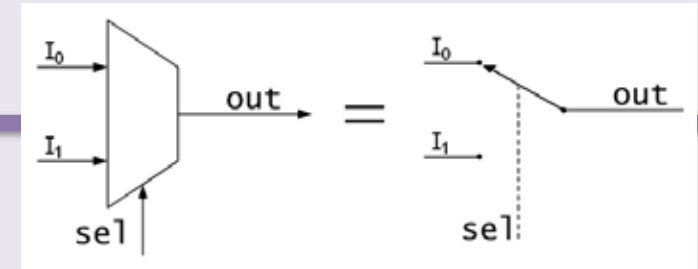
Mux Truth Table and Logic Circuit

1-bit Mux

I_0	I_1	S	O
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

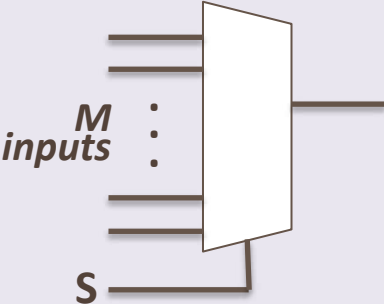
S	I_0	I_1		
	00	01	11	10
0			1	1
1		1	1	

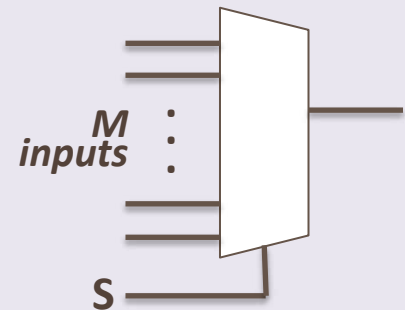
$$O = S \cdot I_1 + S' \cdot I_0$$



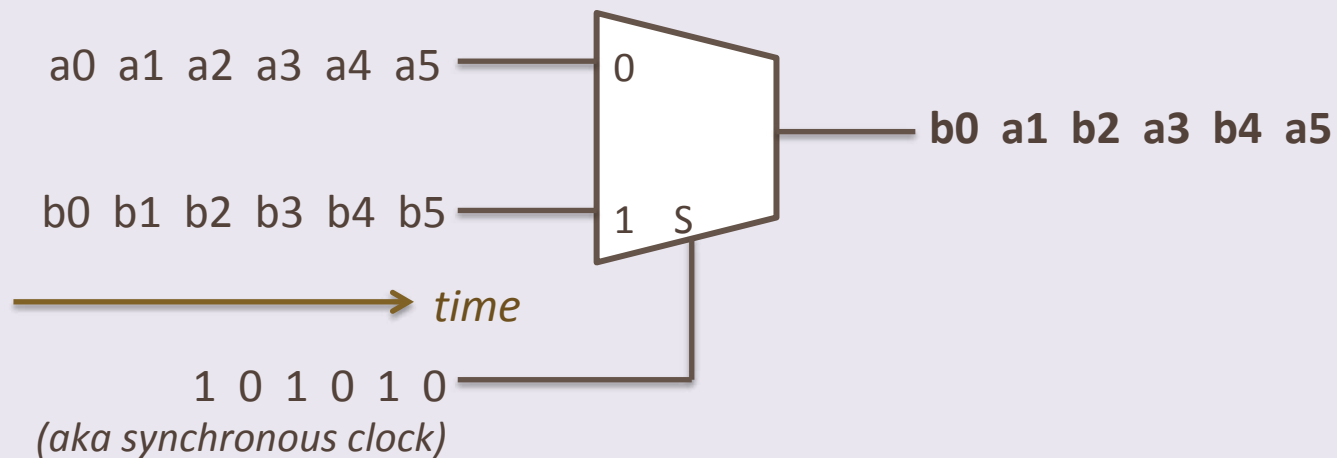
• = lines are physically connected

Beyond 1-bit Muxes

- General mux form: **N-bit, M-to-1**
 - Where:
 - N = how wide the data bus is (in bits, min. 1)
 - M = how many inputs to the mux (min. 2)
 - The “select” input (S) has to be able to select
1 out of M inputs
 - So, if $M = 2$, S should be at least 1 bit ($S = 0$ for one line, $S = 1$ for the other)
 - But if $M = 3$, S should be at least **2 bits** (why?)
 - If $M = 4$, S should be at least ???
 - At least 2 bits
 - If $M = 5$, S will have to be ???
 - At least 3 bits
- 
- The diagram shows a standard multiplexer symbol. It consists of a trapezoidal shape representing the internal logic. On the left side, there are five horizontal lines representing the data inputs. To the left of these lines, the text "M" is positioned above three vertical dots, which are followed by the word "inputs". On the bottom-left corner, there is a single horizontal line labeled "S", representing the select input. A single horizontal line exits from the right side of the trapezoid, representing the output.



What Does This Circuit Do?

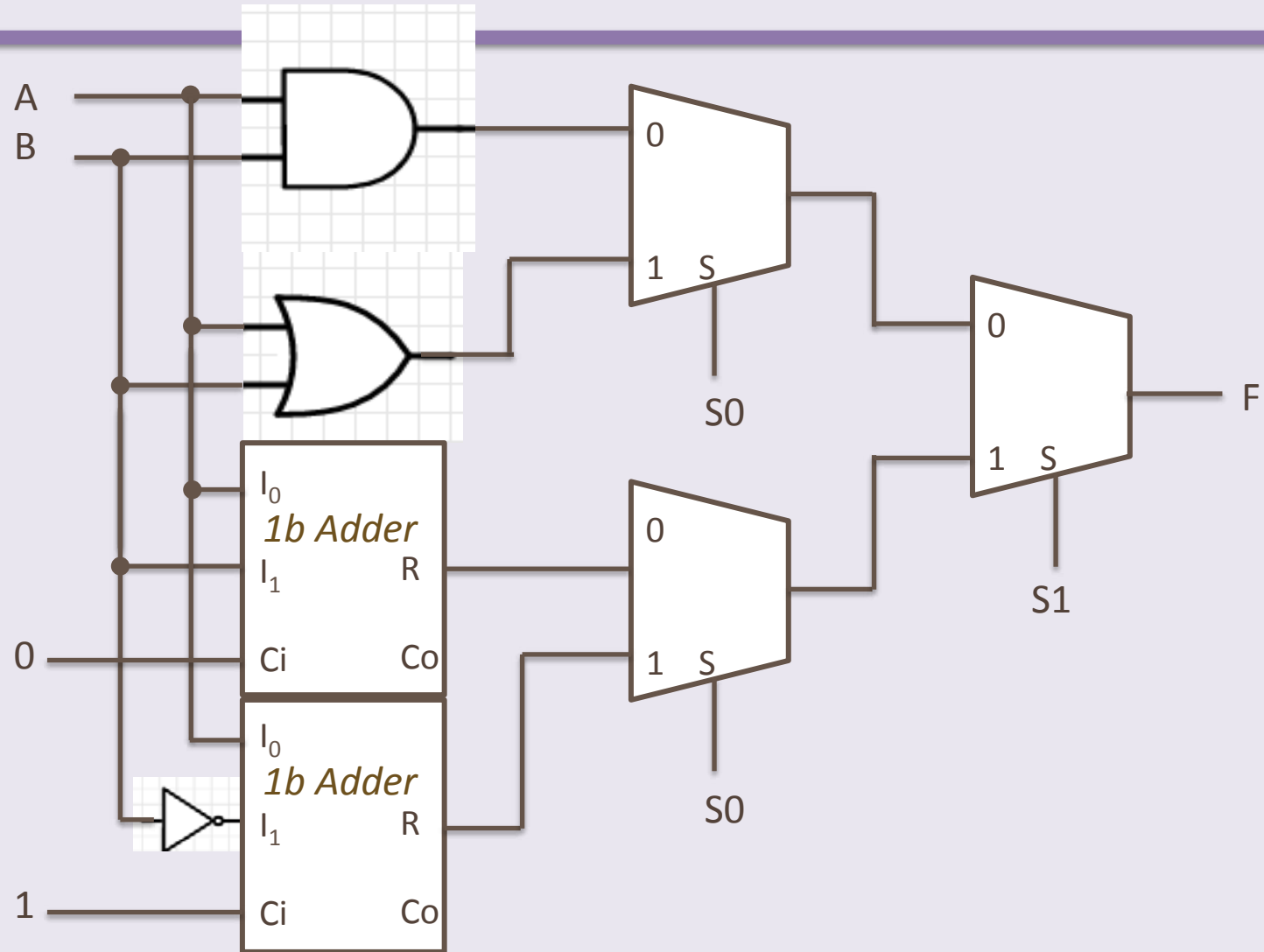


a0	a1	a2	a3	a4	a5
b0	b1	b2	b3	b4	b5
1	0	1	0	1	0
b0	a1	b2	a3	b4	a5

time →

What Does This Circuit Do?

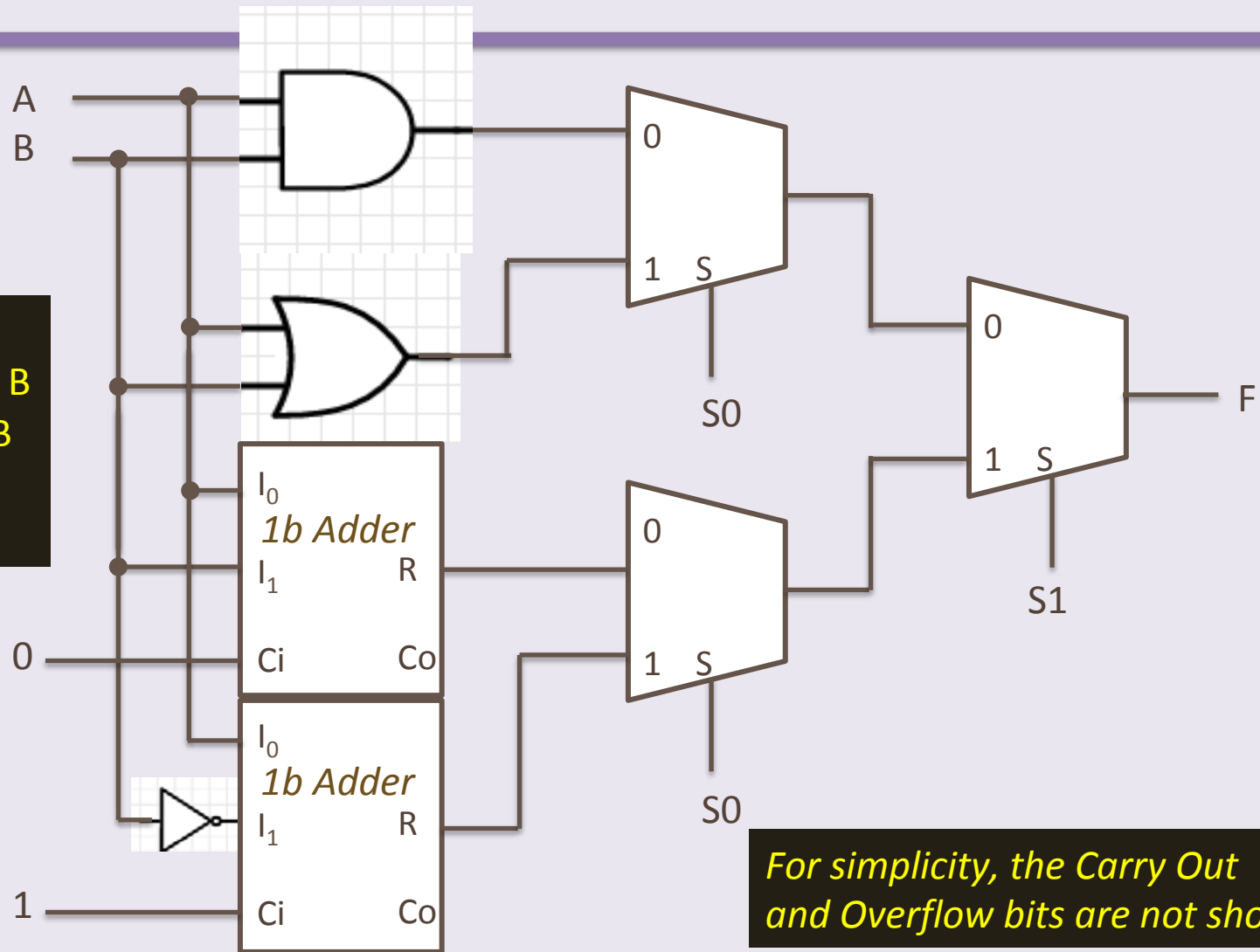
Class Ex.



What Does This Circuit Do?

Class Ex.

S1	S0	F
0	0	A && B
0	1	A B
1	0	A + B
1	1	A - B



For simplicity, the Carry Out and Overflow bits are not shown

logic.ly File Edit View Tools Simulate Help

Input Controls

- Toggle Switch
- Push Button
- Clock
- High Constant (1)
- Low Constant (0)

Output Controls

- Light Bulb
- Digit

Logic Gates

- Buffer
- NOT Gate
- AND Gate
- NAND Gate
- OR Gate
- NOR Gate
- XOR Gate
- XNOR Gate

Simulation of Combinatorial Logic

5/22/18

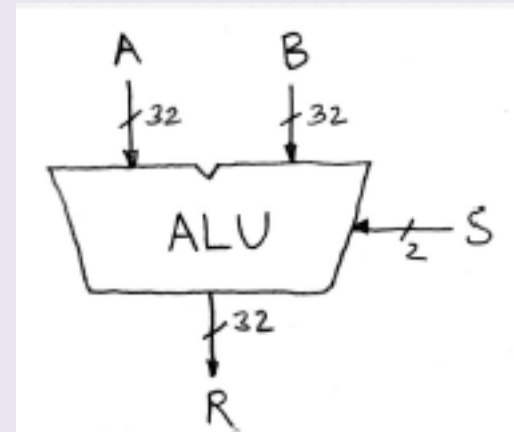
- Go to:
<https://logic.ly/demo/>

IN-CLASS DEMONSTRATION (Needed for Lab #7)

Matni, CS64, Sp18

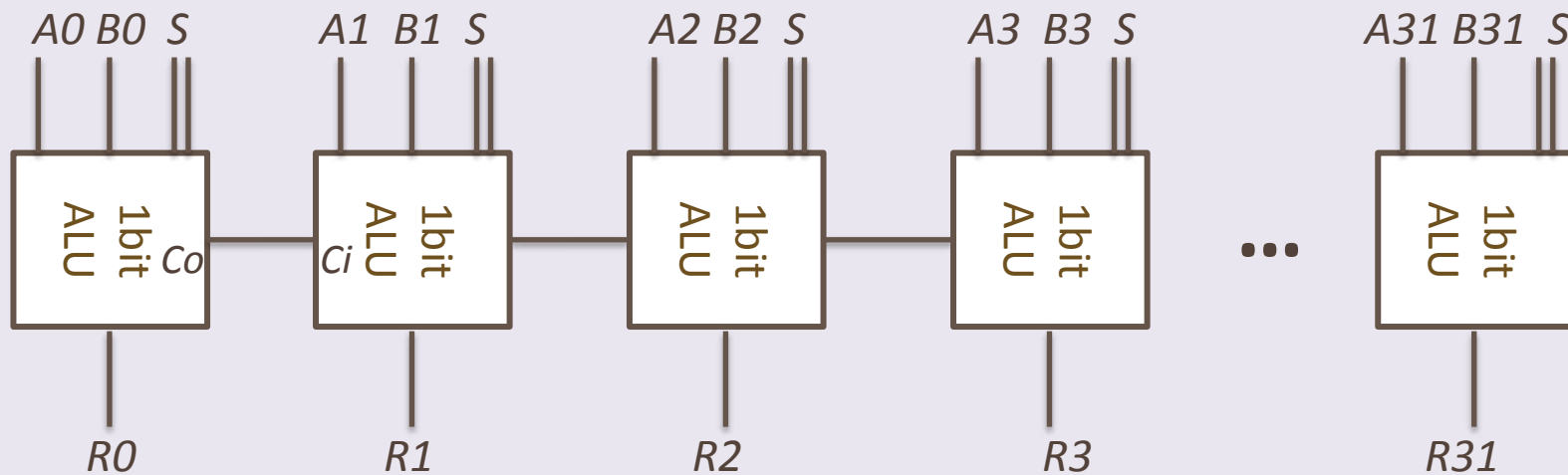
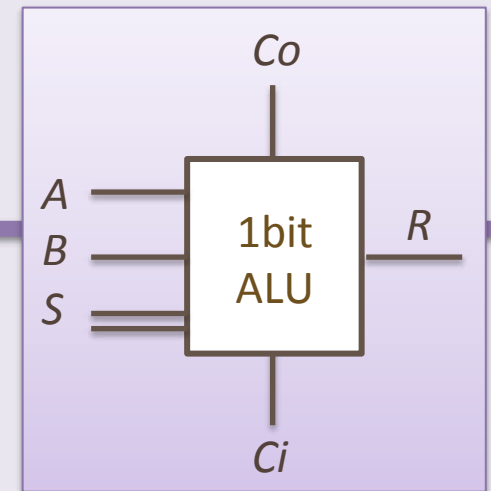
Arithmetic-Logic Unit (ALU)

- Recall: the ALU does all the computations necessary in a CPU
- The previous circuit was a simplified ALU:
 - When $S = 00$, $R = A + B$
 - When $S = 01$, $R = A - B$
 - When $S = 10$, $R = A \text{ AND } B$
 - When $S = 11$, $R = A \text{ OR } B$

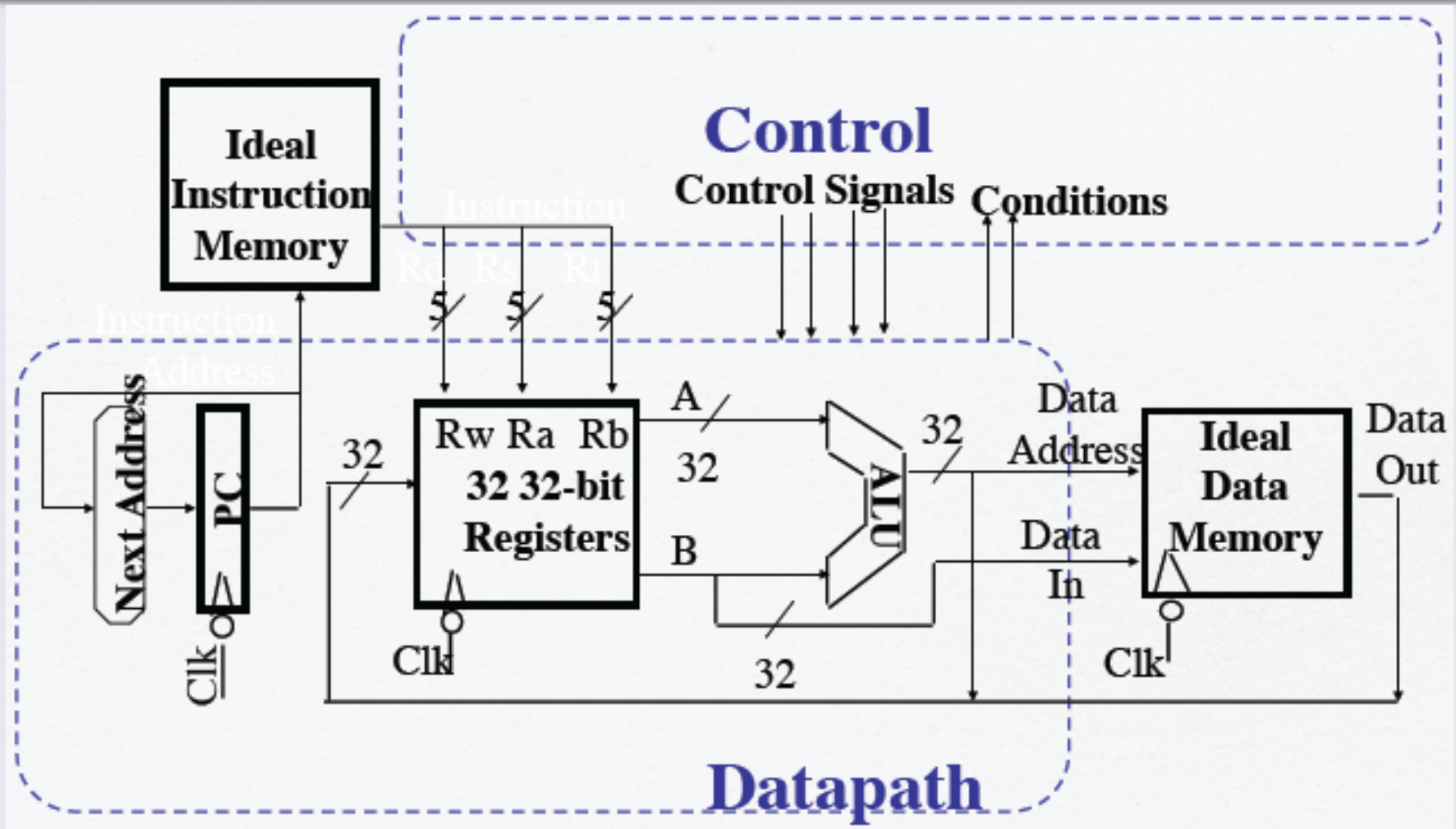


Simplified ALU

- We can string 1-bit ALUs together to make bigger-bit ALUs (e.g. 32b ALU)



Abstract Schematic of the MIPS CPU

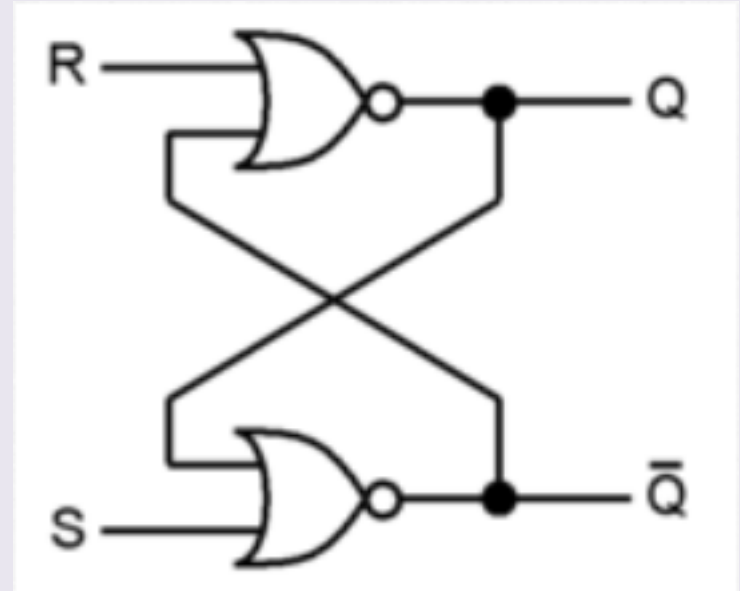


Combinatorial vs. Sequential Logic

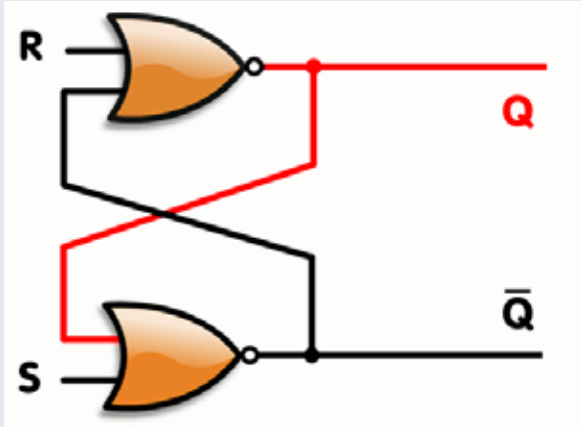
- The CPU schematic shows
both combinatorial and sequential logic blocks
- **Combinatorial Logic**
 - Combining multiple logic blocks
 - The output is a function **only** of the present inputs
 - There is no memory of past “states”
- **Sequential Logic**
 - Combining multiple logic blocks
 - The output is a function of **both** the present inputs and ***past*** inputs
 - There exists a memory of past “states”

The S-R Latch

- Only involves 2 NORs
- The outputs are fed-back to the inputs
- The result is that the output state (either a 1 or a 0) is maintained even if the input changes!



How a Latch Works



S	R	Q_0	Comment
0	0	Q^*	Hold output
0	1	0	Reset output
1	0	1	Set output
1	1	X	Undetermined

- Note that if one NOR input is **0**, the output becomes the inverse of the other input
- So, if output Q already exists and if $S = 0, R = 0$, then Q will remain at whatever it was before! (hold output state)
- If $S = 0, R = 1$, then Q becomes 0 (reset output)
- If $S = 1, R = 0$, then Q becomes 1 (set output)
- Making $S = 1, R = 1$ is not allowed (undetermined output)

Consequences?

- As long as $S = 0$ **and** $R = 0$, the circuit output holds memory of its prior value (state)

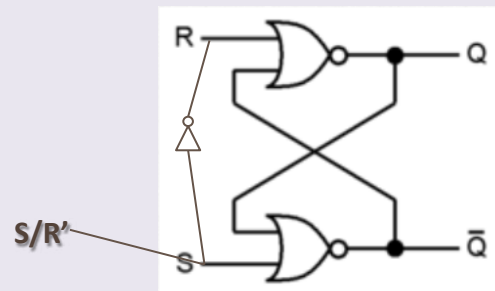
S	R	Q_0	Comment
0	0	Q^*	Hold output
0	1	0	Reset output
1	0	1	Set output
1	1	X	Undetermined

- To change the output, just make $S = 1$ (but also $R = 0$) to make the output 1 (set) **OR** $S = 0$ (but also $R = 1$) to make the output 0 (reset)
- Just avoid $S = 1, R = 1...$

About that $S = 1, R = 1$...

- What if we avoided it on purpose by making $R = \text{NOT}(S)$?

- Where's the problem?



- This, by itself, precludes a case when $R = S = 0$
 - You'd need that if you want to preserve the previous output state!
- Solution: the ***clocked latch*** and ***the flip-flop***
 - ***More on that coming up...!***

YOUR TO-DOs

- Finish Lab #7 by Friday!

</LECTURE>