Multiplexers, ALU Design

CS 64: Computer Organization and Design Logic

Lecture #13

Winter 2020

Ziad Matni, Ph.D.
Dept. of Computer Science, UCSB

Administrative

- Lab 6 due Thursday!
- Lab 7 will be posted today

- This Friday, the TAs will be in the lab
  - Attendance **is not mandatory**, but likely <u>very useful </u>for you to be there
  - Work on your Lab 7 b/c it's due next Tuesday

Lecture Outline

- More on Logic Simplification using Kmaps
- Multiplexers
- ALUs

Any Questions From Last Lecture?

Any Questions About the Lab?

5 Minute Pop Quiz!

- Given the following K-Map for binary function **F**:

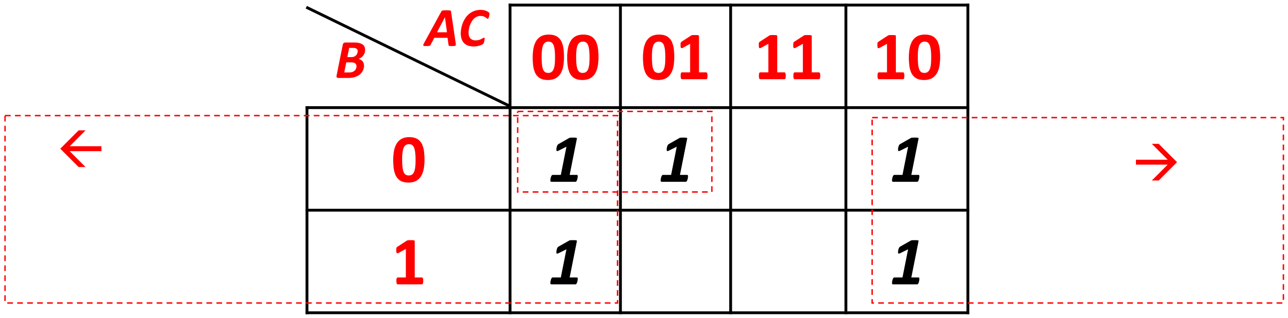
B \ AC	00	01	11	10
0	*1*	*1*		*1*
1	*1*			*1*

a) Group properly and write the optimized function **F**

b) Draw the circuit

5 Minute Pop Quiz! (Solution)

- Given the following K-Map for binary function **F**:



B \ AC	00	01	11	10
0	1	1		1
1	1			1

a) Group properly and write the optimized function **F**

F = !A!B + !C

b) Draw the circuit

See black board

Exercise 1

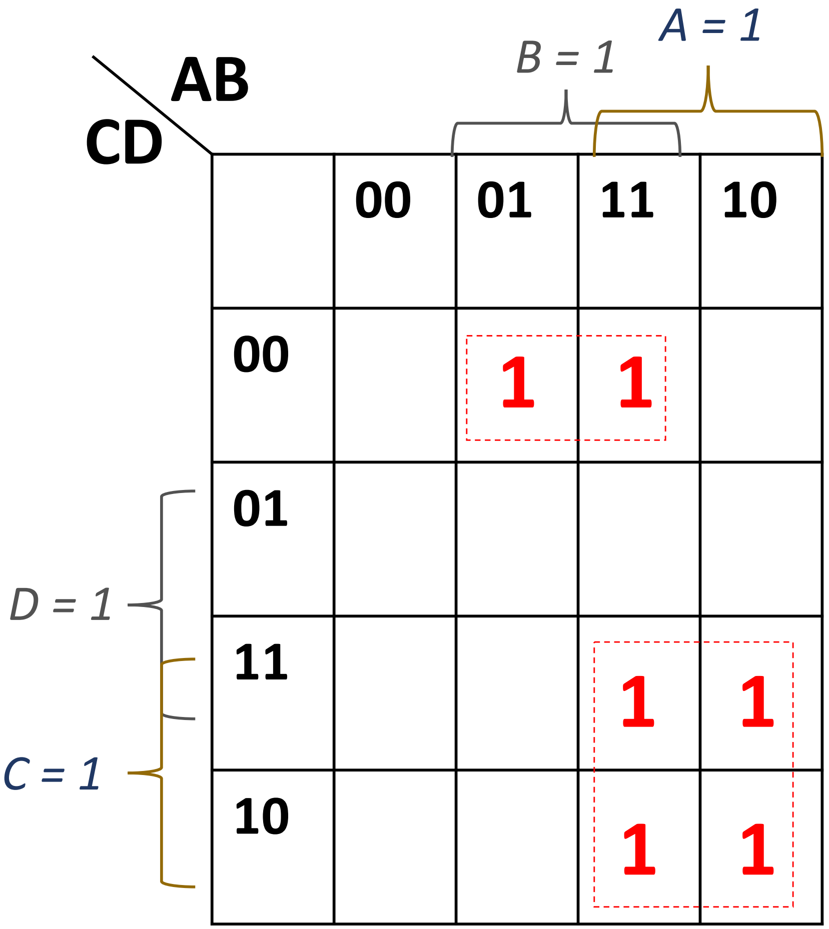
- Given the following truth table, draw the resulting logic circuit
  - **STEP 1**: Draw the K-Map and simplify the function
  - **STEP 2**: Construct the circuit from the now simplified function

A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

Exercise 1 – Step 1

Get the simplified function

A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

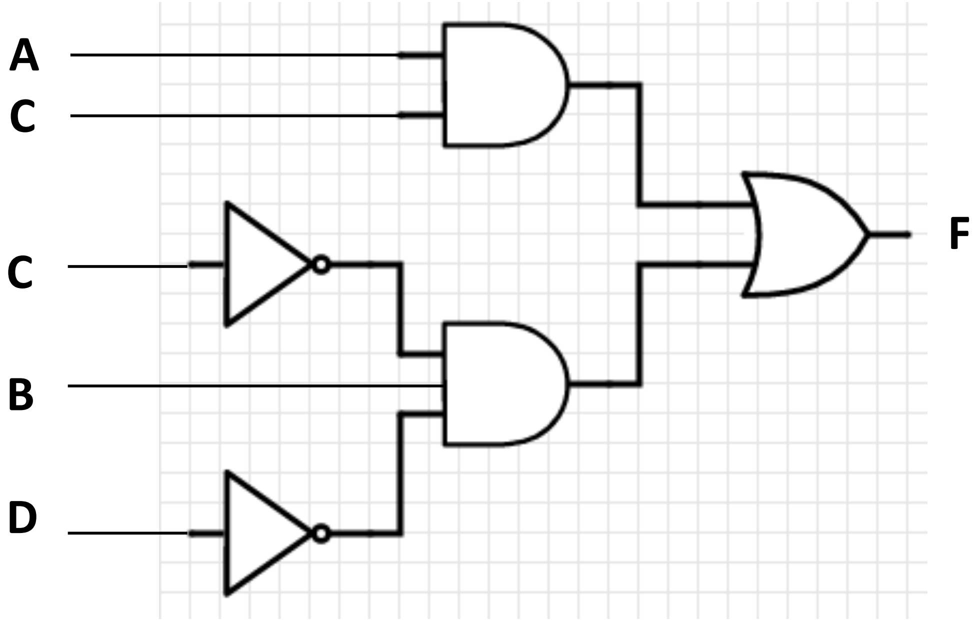


	00	01	11	10
00		1	1	
01				
11			1	1
10			1	1

F(A,B,C) = B.!C.!D + A.C

Exercise 1 – Step 2
Draw the logic circuit diagram

F(A,B,C) = B.!C.!D + A.C



Exercise 2

Class Ex.

- Given the following truth table, draw the resulting logic circuit

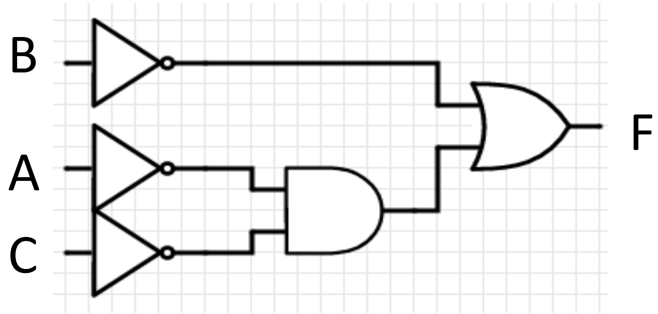
A	B	C	F
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

AB

C

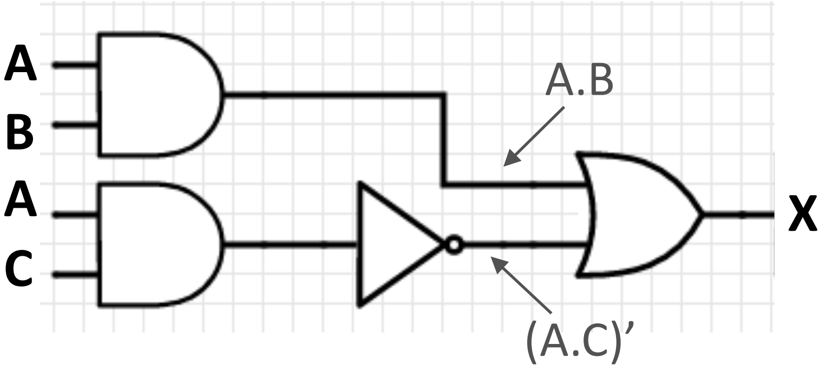
	00	01	11	10
0	1	1		1
1	1			1

F(A,B,C) = !B + !A!C



Exercise 3

- Given the following schematic of a circuit, (a) write the function and (b) fill out the truth table:



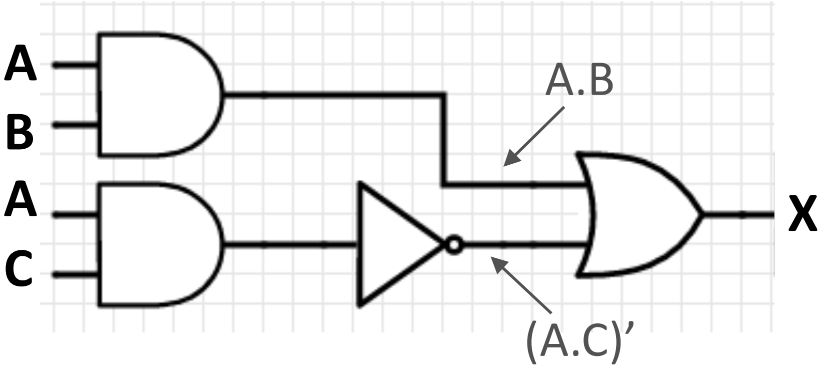
A	B	C	X
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

X = A.B + !(A.C)

(note that also means: **X = A.B + !A + !C**)

Exercise 3

- Given the following schematic of a circuit, (a) write the function and (b) fill out the truth table:



A	B	C	X
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

X = A.B + !(A.C)

(note that also means: **X = A.B + !A + !C**)

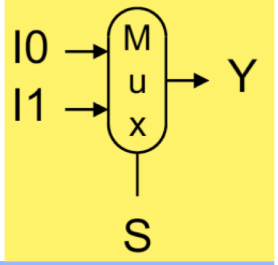
Multiplexer

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

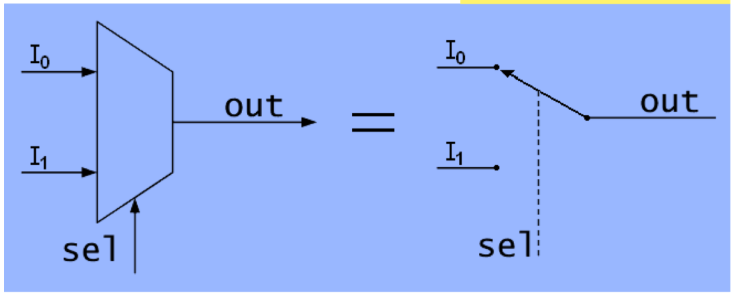
```
// if s = 0, output is a
// if s = 1, output is b
int mux(int a, int b, int s)
{
     if (!s) return a;
     else return b;
}
```

Multiplexer (*Mux* for short)

- Combinatorial circuits who function as a “chooser” between multiple inputs to be “driven to” the output
- Always multiple inputs (N), always ONE output (N-to-1 mux)
  - Can be drawn symbolically in 2 ways (trapezoid vs oval) --- there’s NO difference, just a preference in drawing



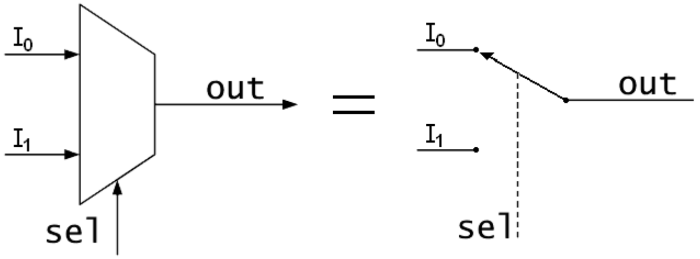
- 1 of the input data lines gets selected to become the output, based on a 3rd “select” (sel) input
  - If **sel** = 0, then I_0 gets to be the output
  - If **sel** = 1, then I_1 gets to be the output
  - So: **OUTPUT =** $\mathbf{I_1.sel + I_0.\overline{sel}}$



- The opposite of a Mux is called a **Demulitplexer** (or **Demux**)

Mux Truth Table and Logic Circuit

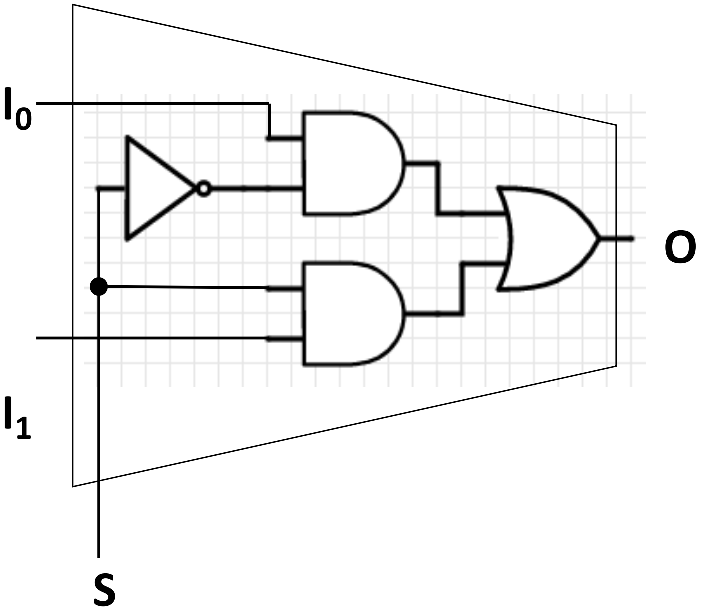
1-bit Mux



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	

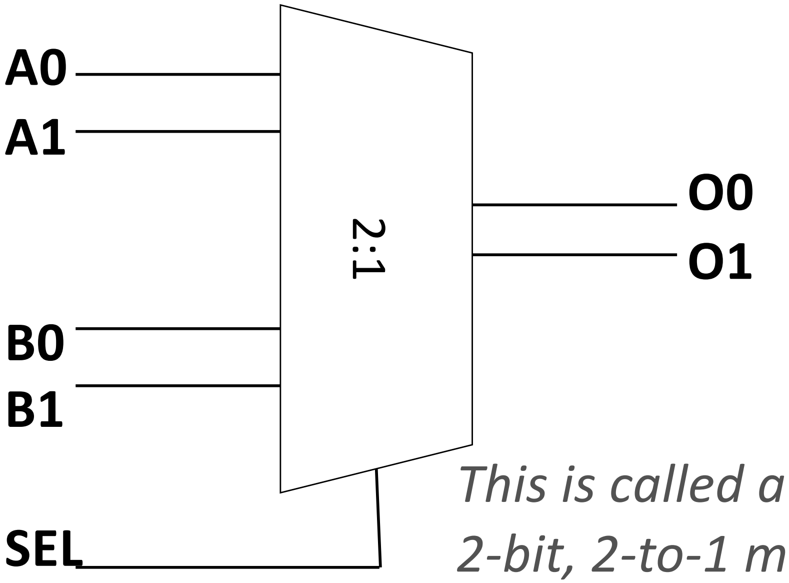
$$O = S.I_1 + S'.I_0$$



● = lines are physically connected

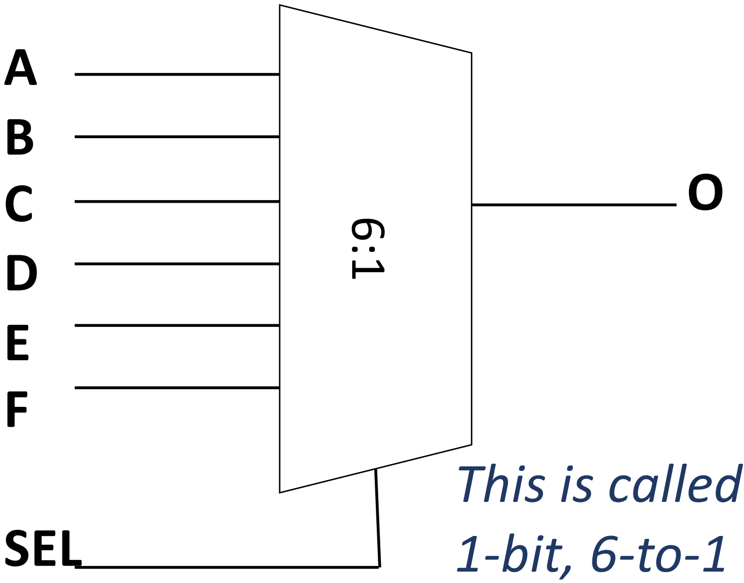
Mux Configurations

Muxes can have I/O that are multiple bits



This is called a 2-bit, 2-to-1 mux

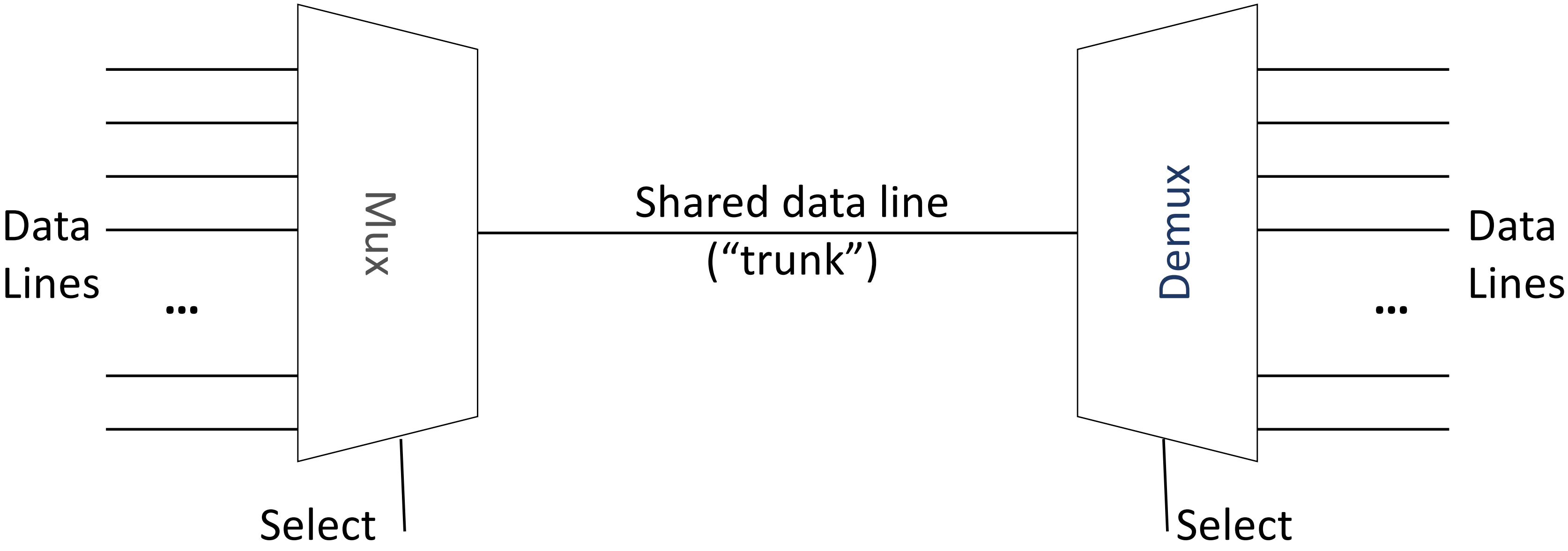
Or they can have more than two data inputs



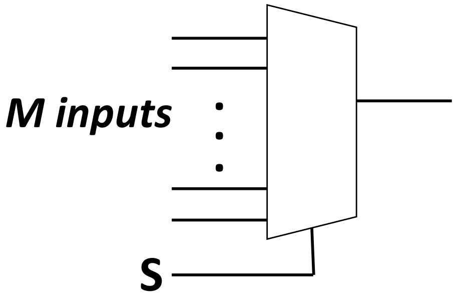
This is called a 1-bit, 6-to-1 mux

The Use of Multiplexers

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



Selection Lines in Muxes



- General mux description: **N-bit, M-to-1**
- Where: N = how "wide" the input is (# of input 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 ??? *(**ANS**: at least 2 bits)*
  - If M = 5, S should be ??? *(**ANS**: at least 3 bits)*

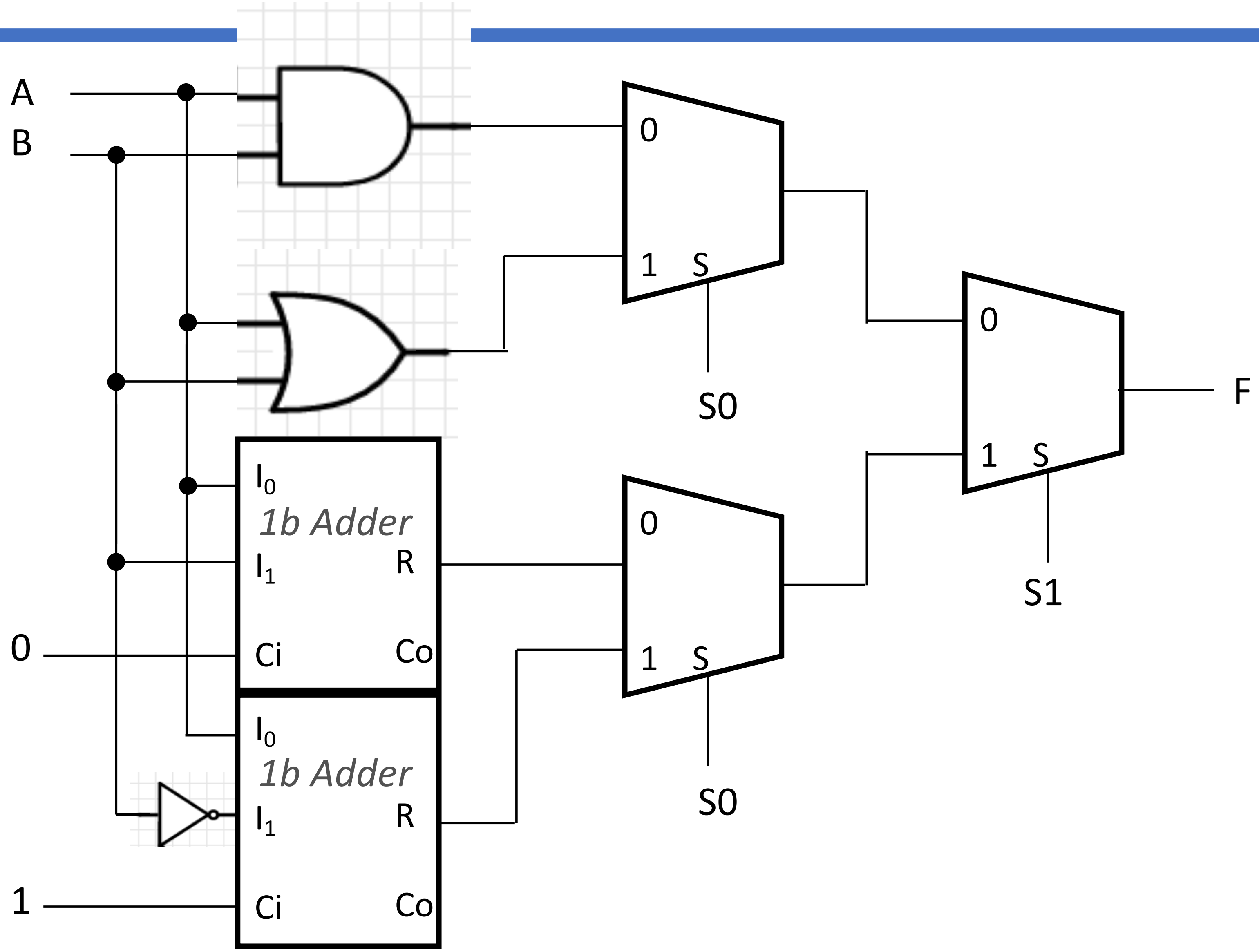
Combining Muxes Together

Can I do a **4:1** mux from 2:1 muxes?

Generally, you can do **2^n:1** muxes from 2:1 muxes.

What Does This Circuit Do?

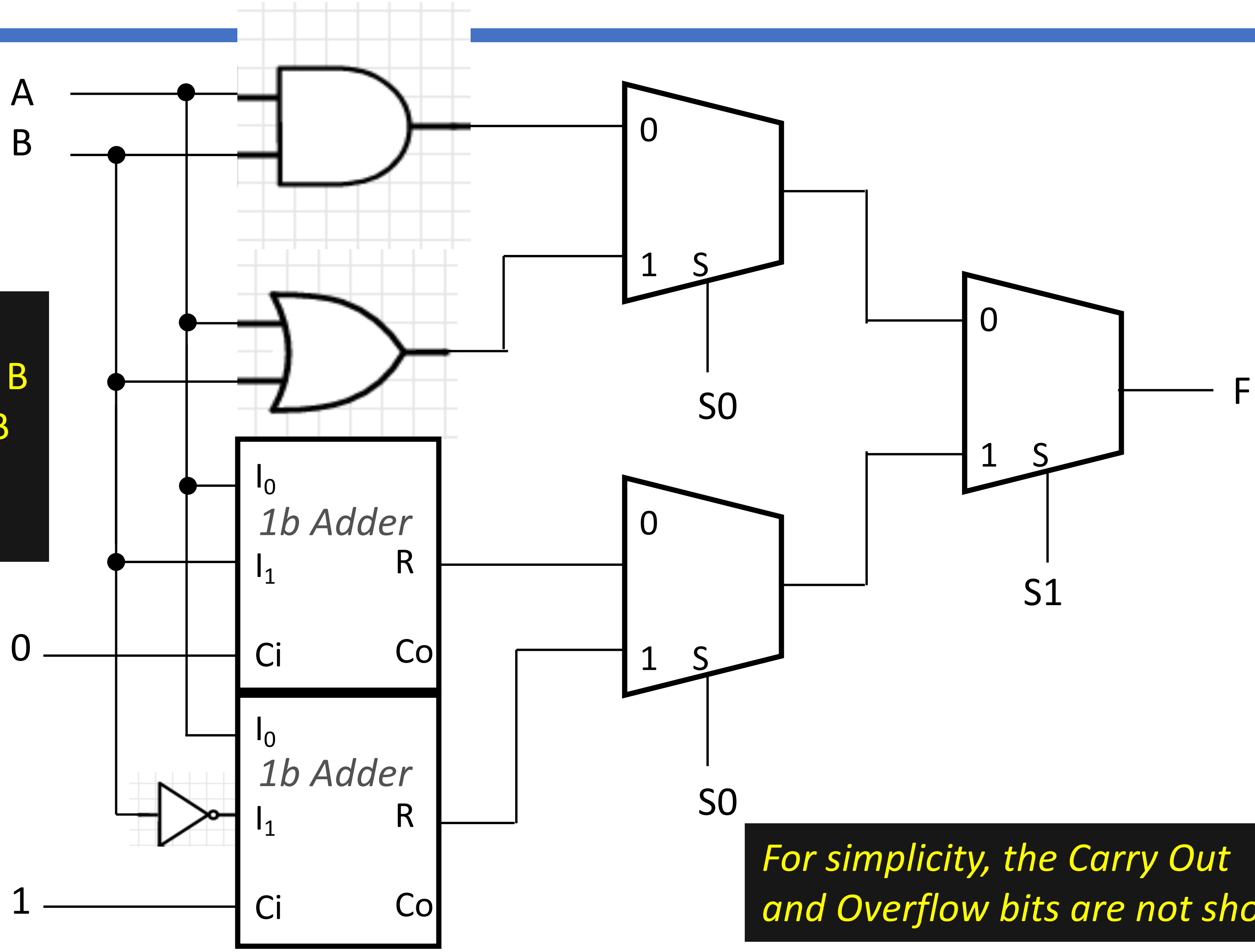
Class Ex.



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

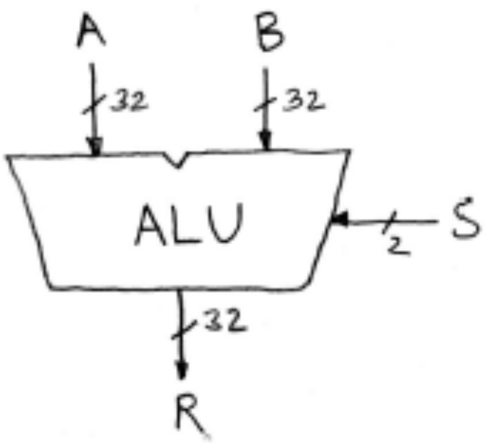


For simplicity, the Carry Out and Overflow bits are not shown

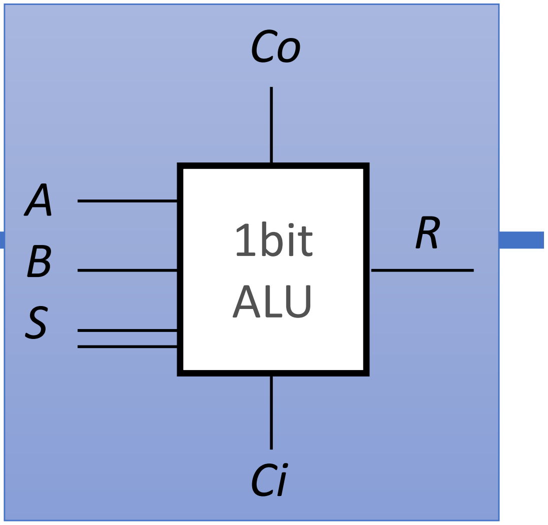
Arithmetic-Logic Unit (ALU)

- Recall: the ALU does all the computations necessary in a CPU

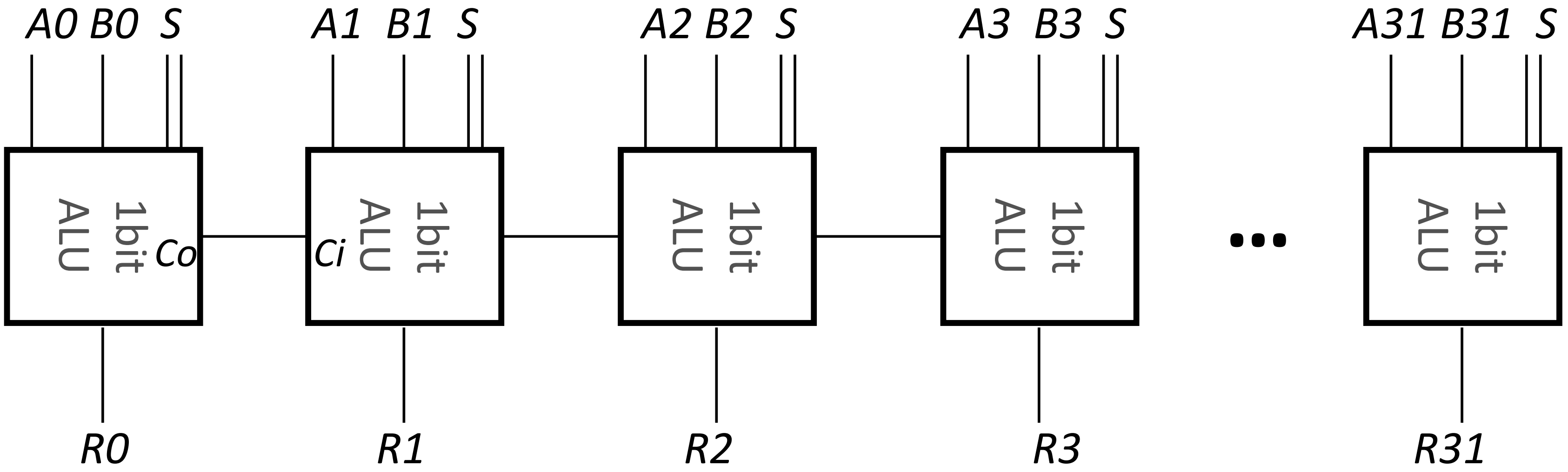
- The previous circuit was a simplified ALU:
  - When S = 00, R = A + B
  - When S = 01, R = A – B
  - When S = 10, R = A AND B
  - When S = 11, R = A OR B



Simplified ALU

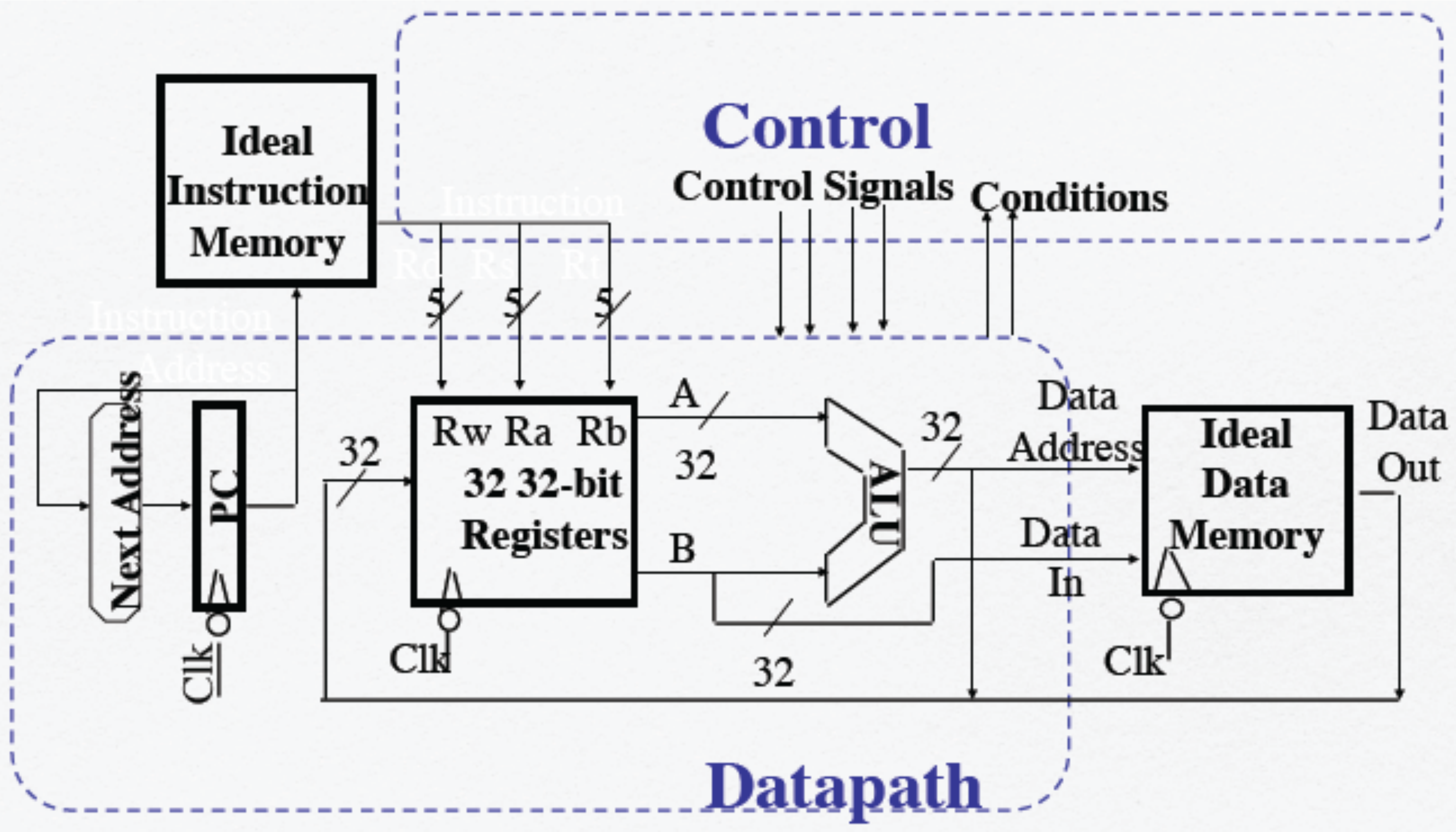


- We can string 1-bit ALUs together to make bigger-bit ALUs (e.g. 32b ALU)



Abstract Schematic of the MIPS CPU

Relevant to a future lab…



YOUR TO-DOs

- Go to Thursdsay lab

  (we won't take attendance)

- Work on Lab 7, which is due **next Tuesday**

</LECTURE>