CS 240
Foundations of Computer Systems

WELLESLEY

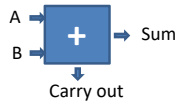
Arithmetic Logic

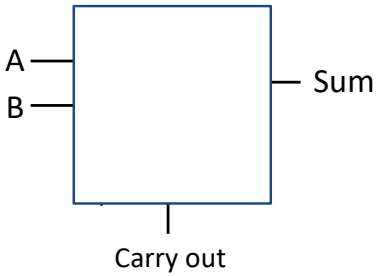
adders

Arithmetic Logic Unit

https://cs.wellesley.edu/~cs240/

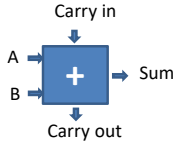
Addition: 1-bit ***half*** adder

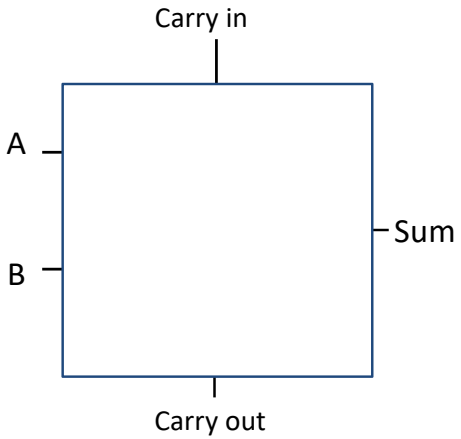




A	B	Carry Out	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

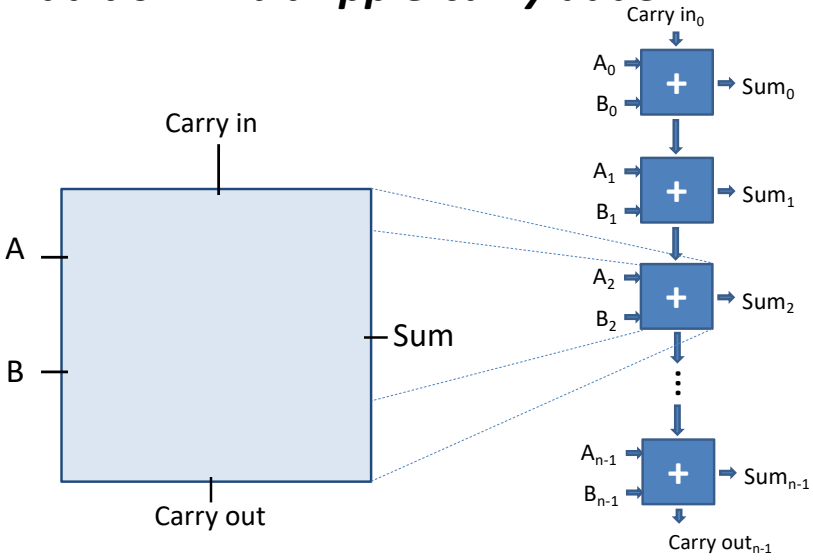
Addition: 1-bit ***full*** adde





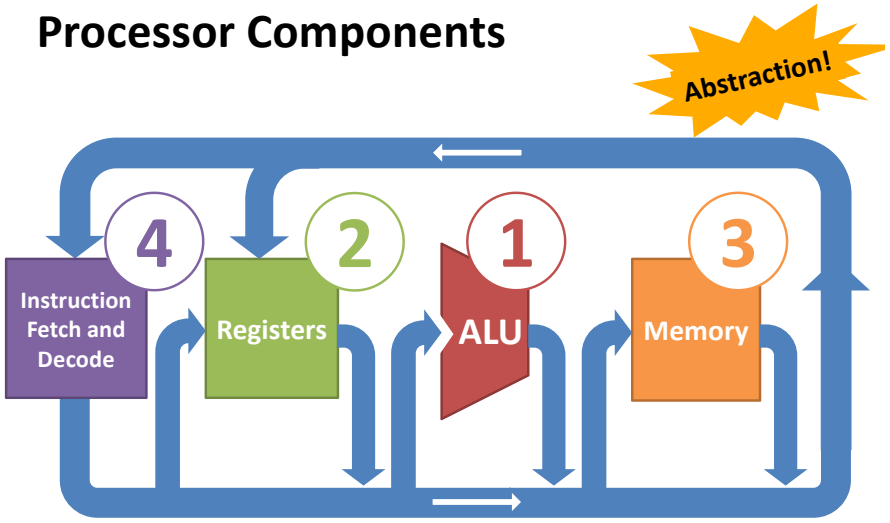
Carry in	A	B	Carry Out	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Addition: ***n***-bit ***ripple-carry*** adder



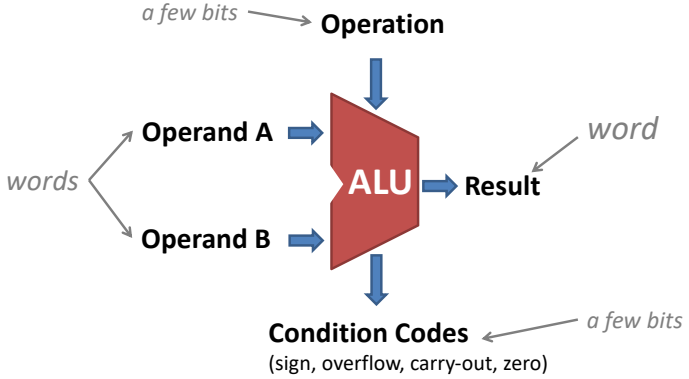
There are faster, more complicated ways too…

Processor Components



Arithmetic Logic Unit (ALU)

1

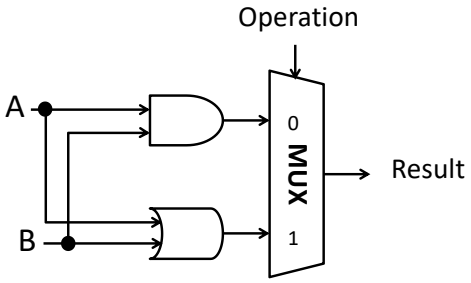


Hardware unit for arithmetic and bitwise operations.

1-bit ALU for bitwise operations

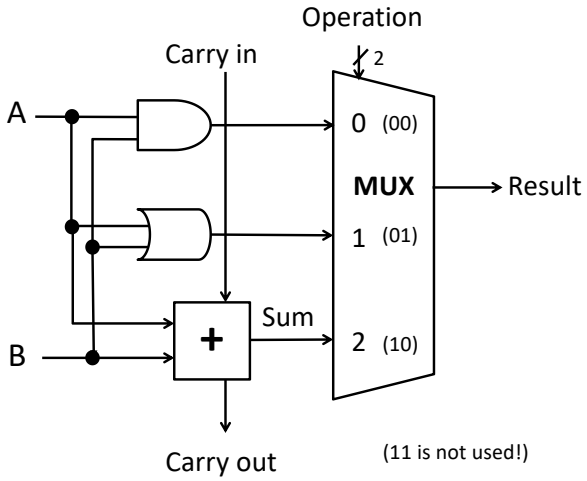
ex

Build an n-bit ALU from n 1-bit ALUs.
Each bit *i* in the result is computed from the corresponding bit *i* in the two inputs.

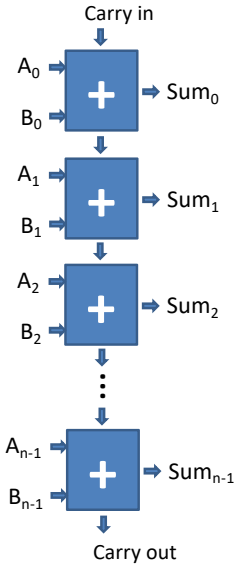


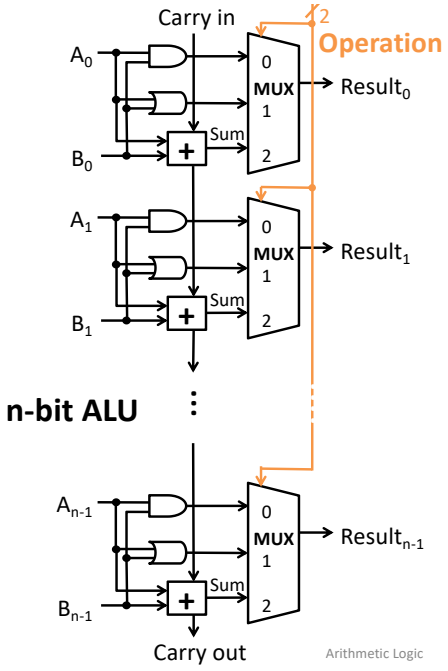
Op	A	B	Result
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

1-bit ALU



n-bit ripple carry adder





ALU conditions

Extra ALU outputs
describing properties of result.

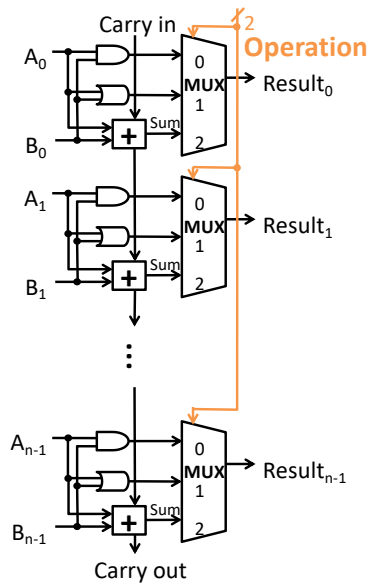
Zero Flag:
1 if result is 00...0 else 0

Sign Flag:
1 if result is negative else 0

Carry Flag:
1 if carry out else 0

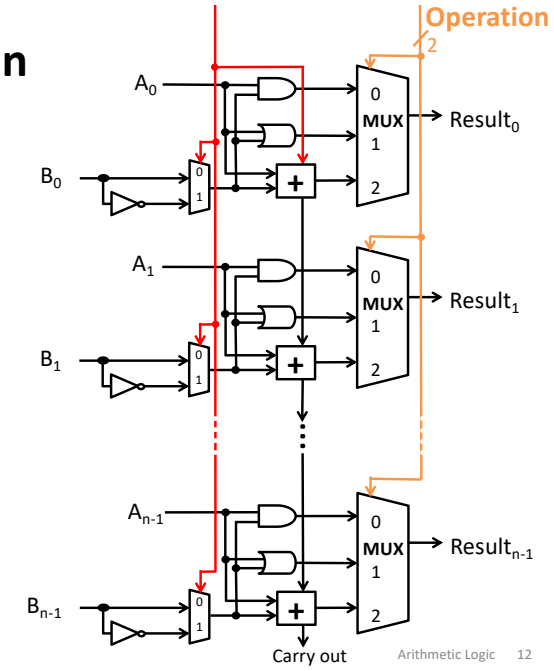
(Signed) **Overflow Flag:**
1 if signed overflow else 0

You will implement these in Circuits Assignment!



Add subtraction

How can we control ALU inputs or add minimal new logic to **compute A-B**?



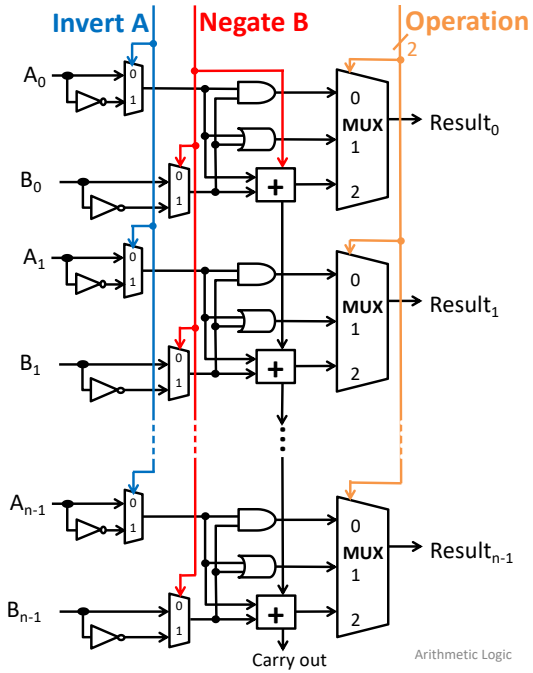
A NAND B

A NOR B

A<B

A==B

How can we control ALU inputs or add minimal new logic to compute each? You will Implement these in Circuits Assignment!



Controlling the ALU

ALU control lines	Function
0000	AND
0001	OR
0010	add
0110	subtract
???	NOR