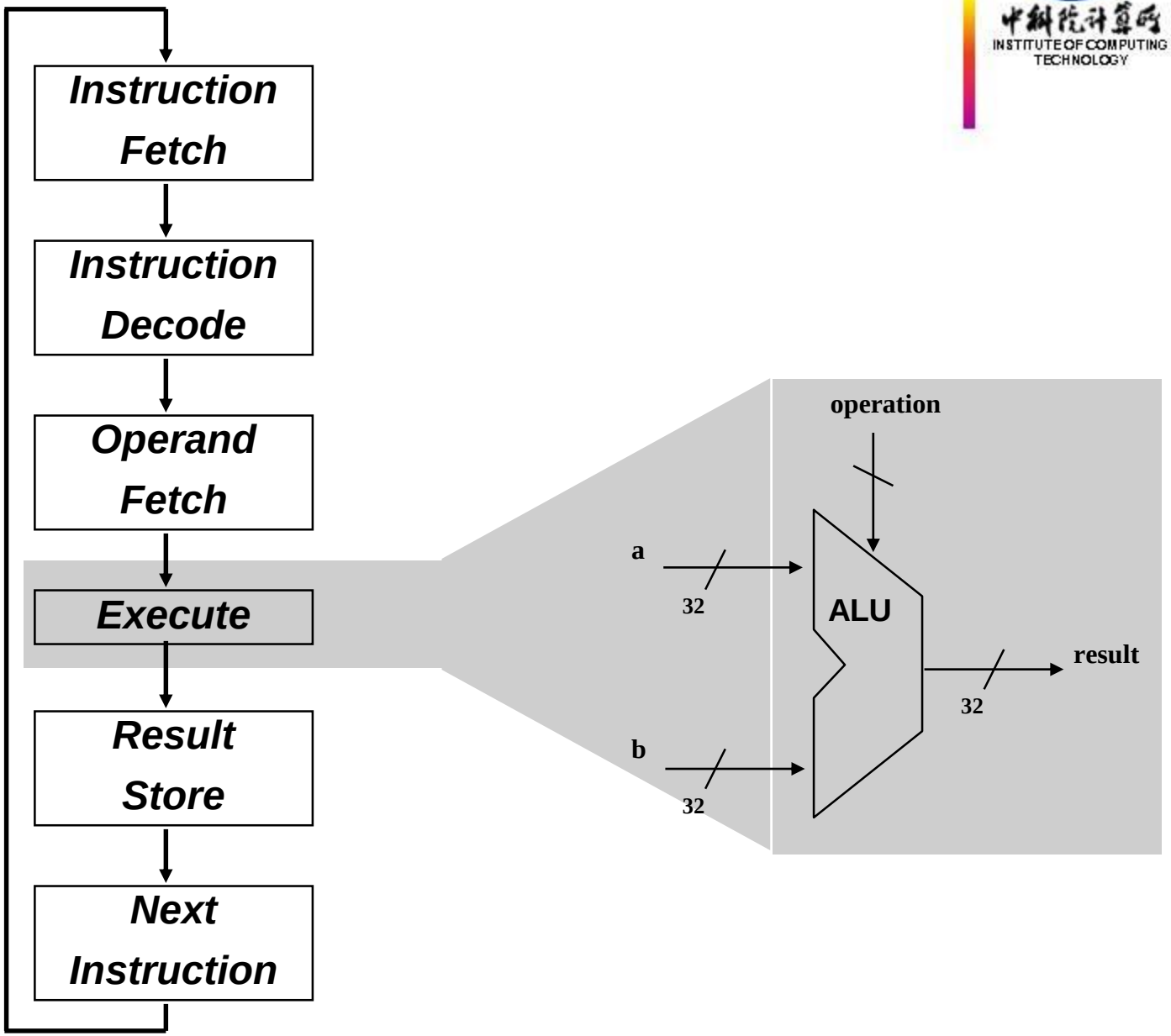
Arithmetic Logic Unit

Xiufeng Sui

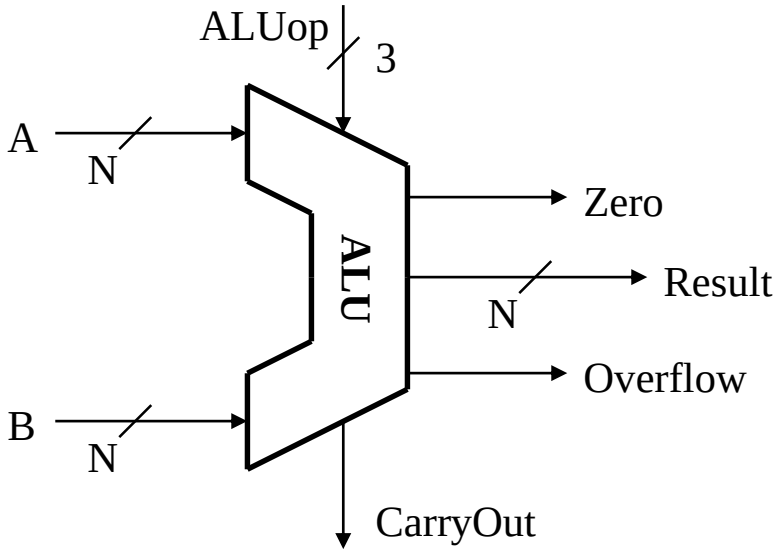
University of Chinese Academy of Sciences (UCAS)

中科院计算所
INSTITUTE OF COMPUTING TECHNOLOGY

Arithmetic -- The heart of instruction execution



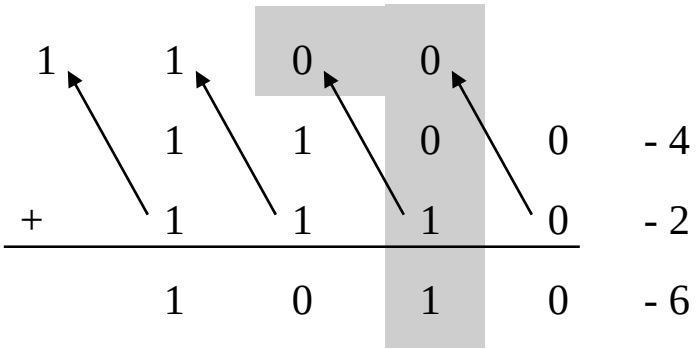
Designing an Arithmetic Logic Unit

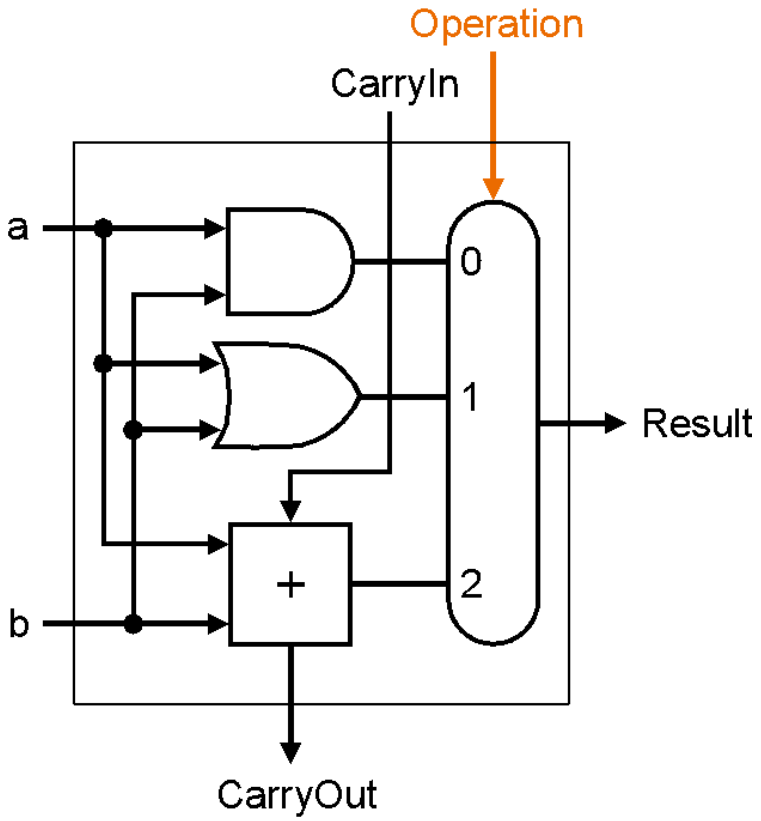


ALU control input	Function	Operations
000	And	and
001	Or	or
010	Add	add, lw, sw
110	Subtract	sub, beq
111	Slt	slt

A One Bit ALU

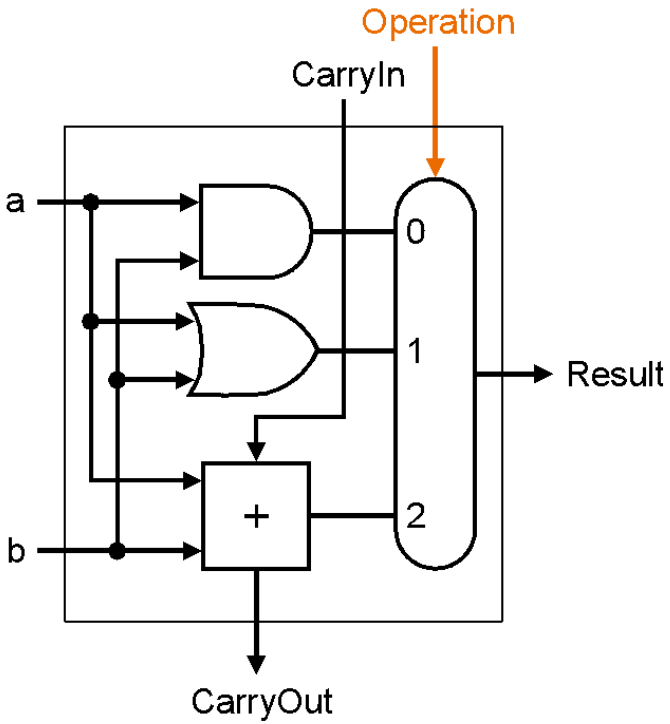
- This 1-bit ALU will perform AND, OR, and ADD



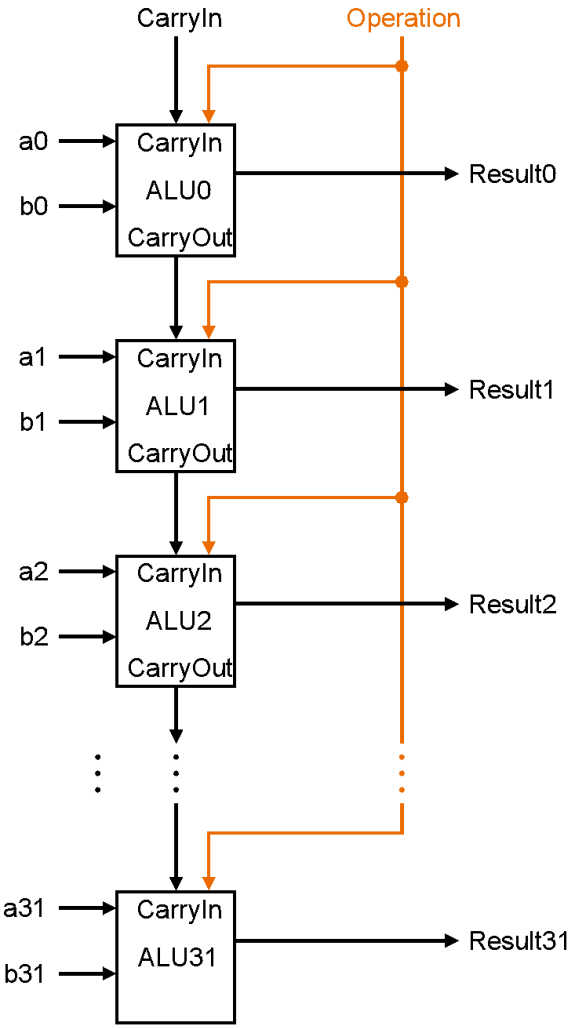


A 32-bit ALU

1-bit ALU

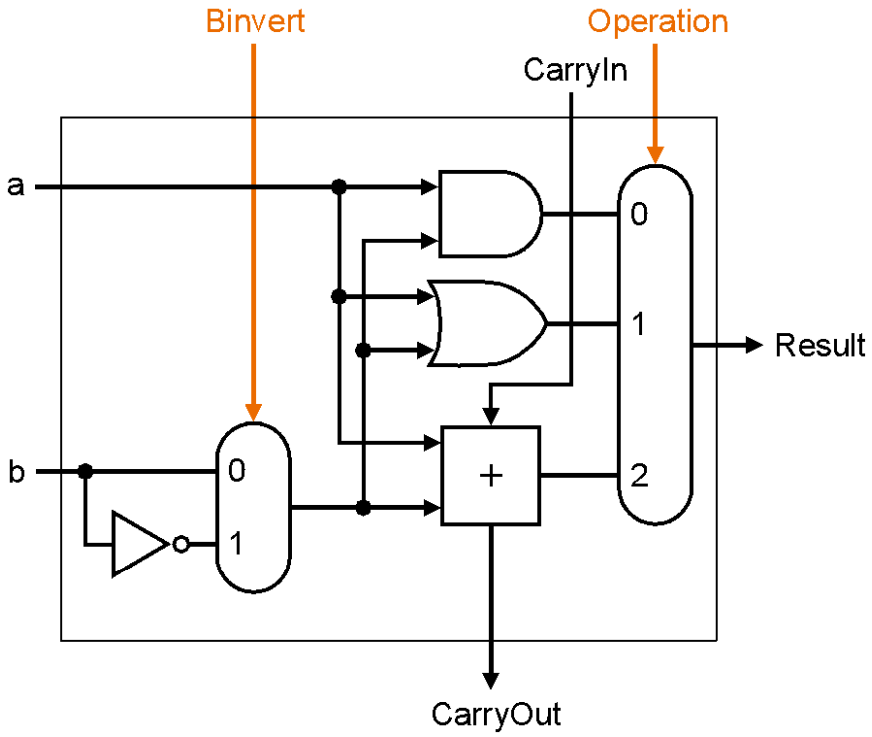


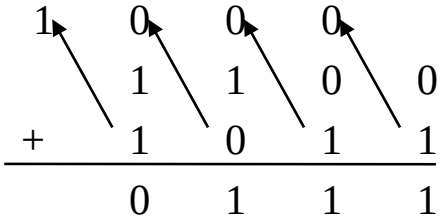
32-bit ALU



How About Subtraction?

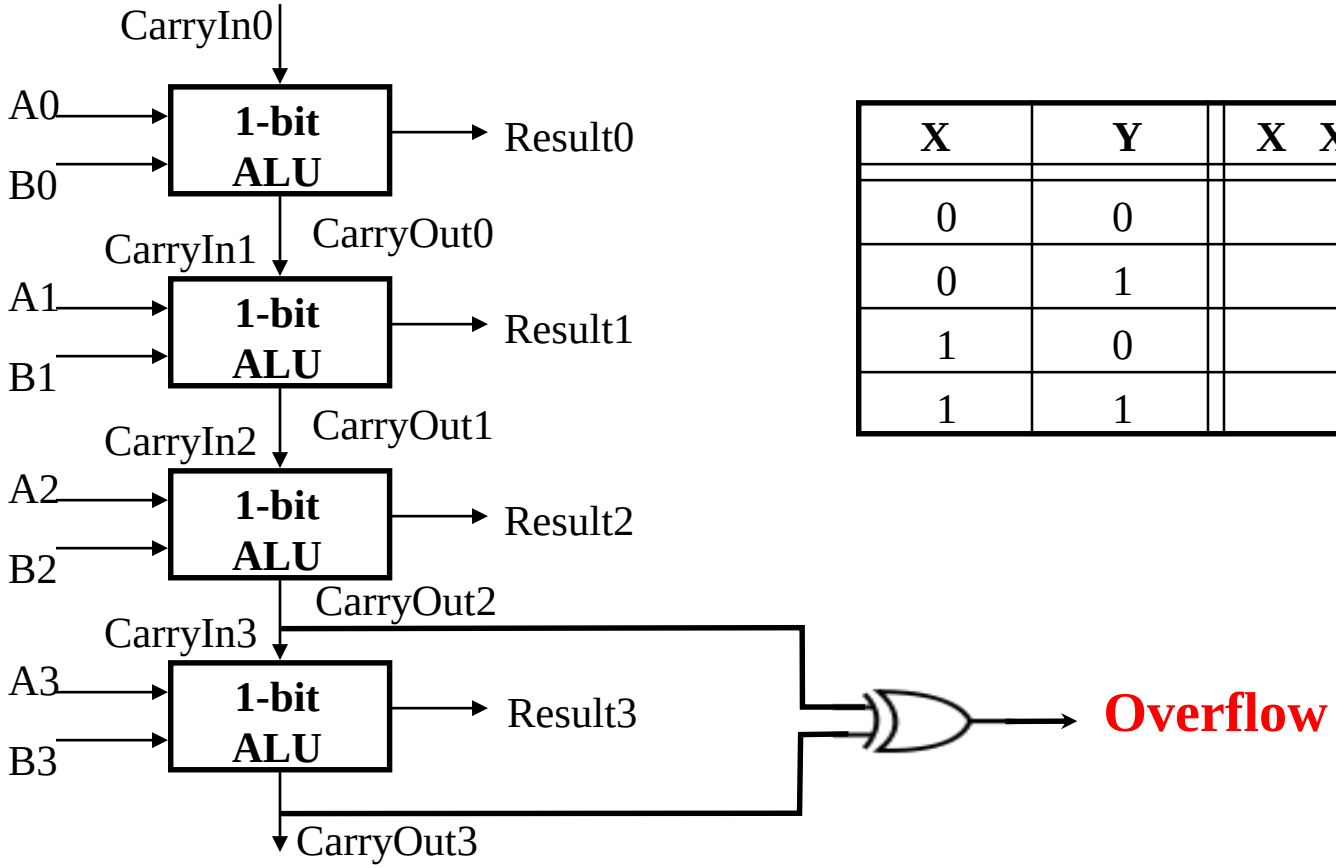
- Keep in mind the following:
  - (A - B) is the same as: A + (-B)
  - 2's Complement negate: Take the inverse of every bit and add 1
- Bit-wise inverse of B is !B:
  - A - B = A + (-B) = A + (!B + 1) = A + !B + 1





Overflow Detection Logic

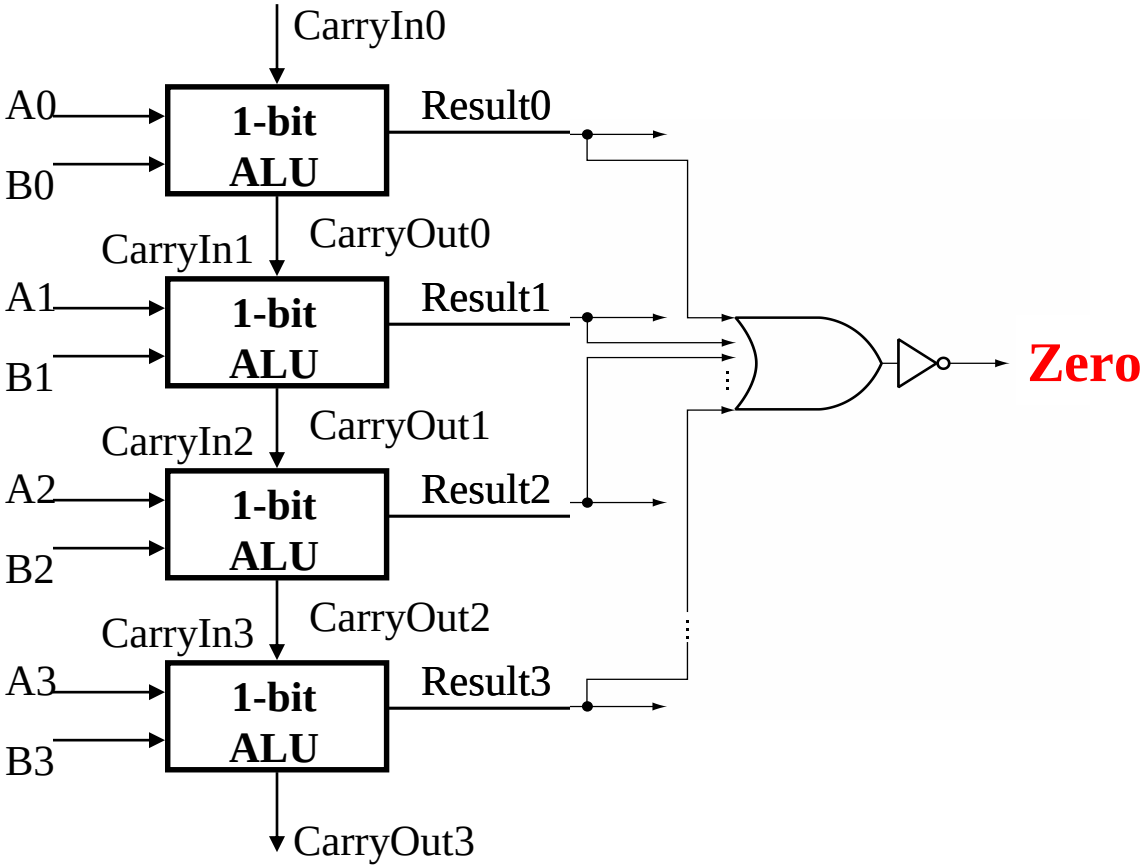
- Carry into MSB ! = Carry out of MSB
  - For a N-bit ALU: Overflow = CarryIn[N - 1] XOR CarryOut[N - 1]



X	Y	X XOR Y
0	0	0
0	1	1
1	0	1
1	1	0

Zero Detection Logic

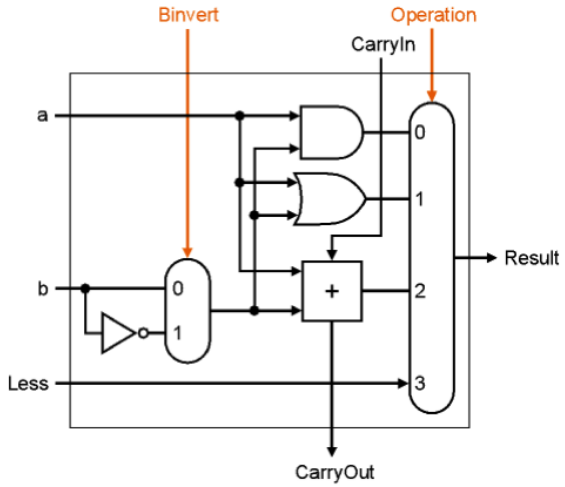
- Zero Detection Logic is just one BIG NOR gate
  - Any non-zero input to the NOR gate will cause its output to be zero



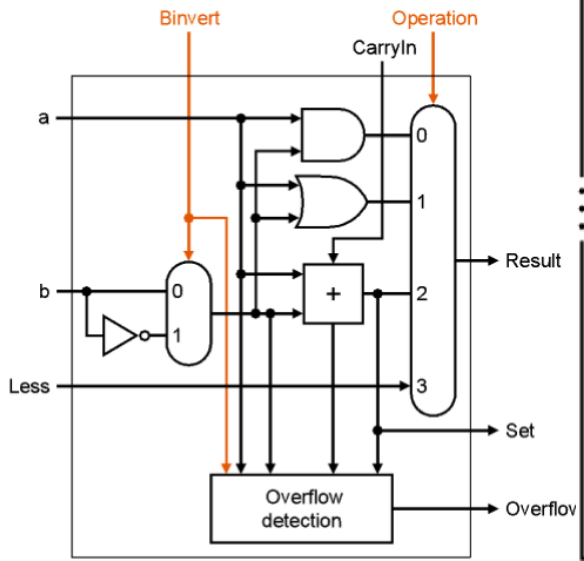
Set-on-less-than

- We are mostly there!
  - A < B => (A - B) < 0
  - Do a subtract
- If true, set LSB to 1, all others 0
  - Use sign bit
    - route to bit 0 of result
    - all other bits zero

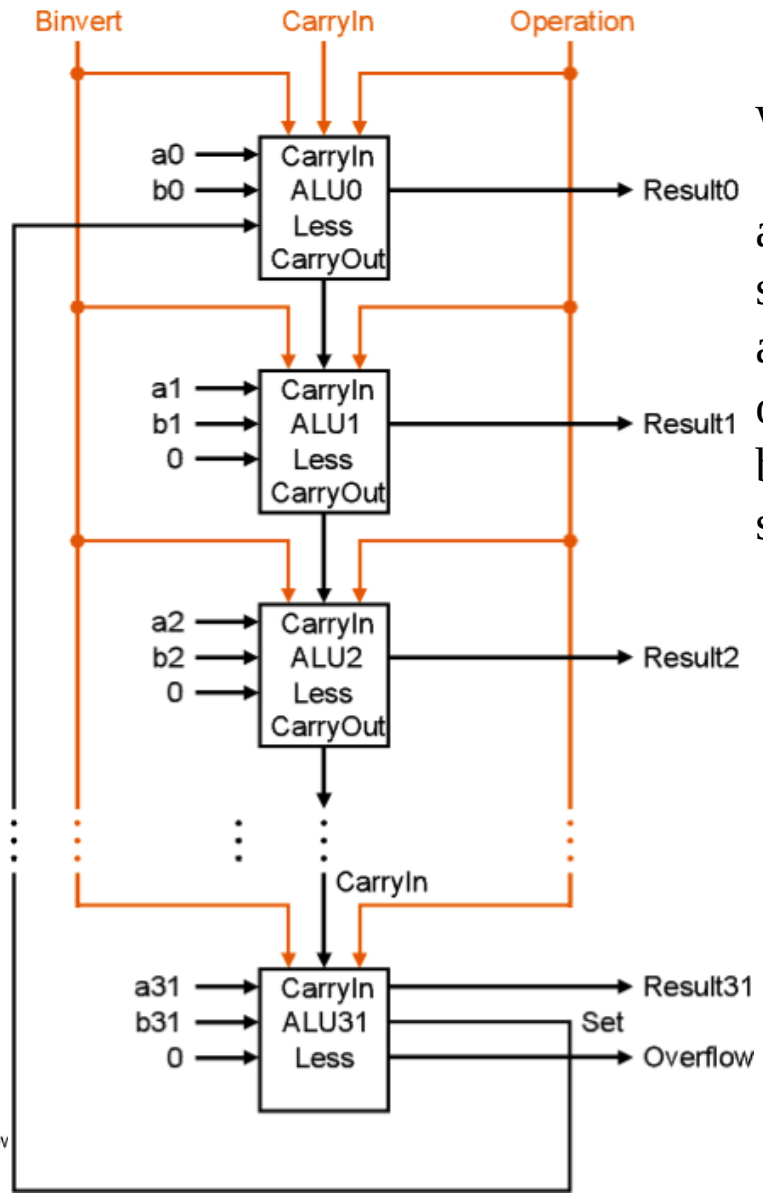
Full ALU



a.

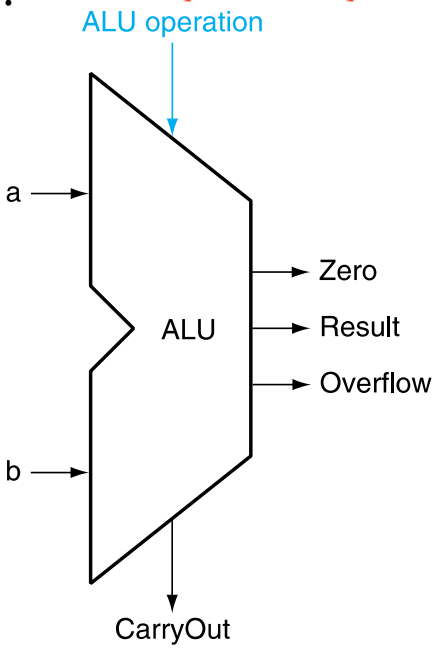


b.



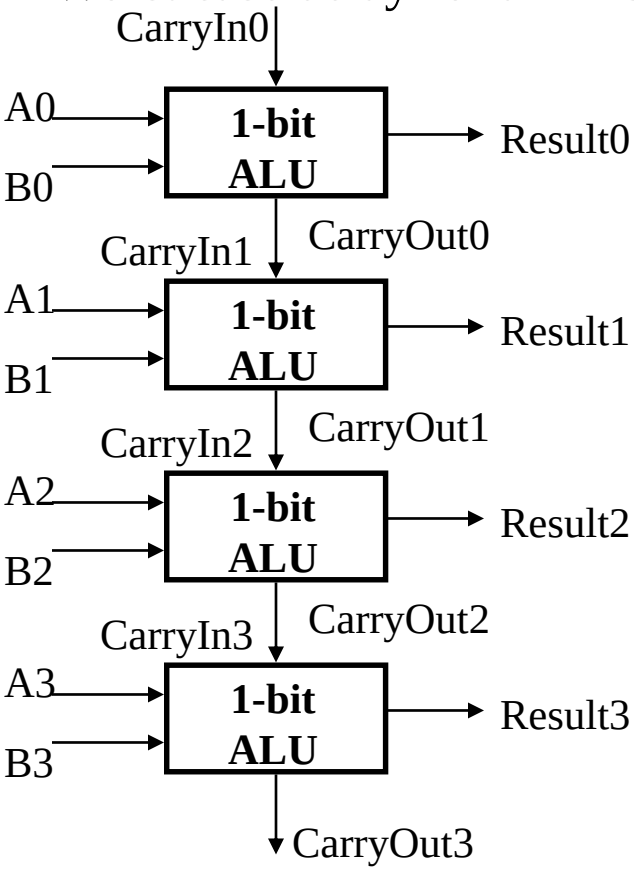
what signals accomplish:

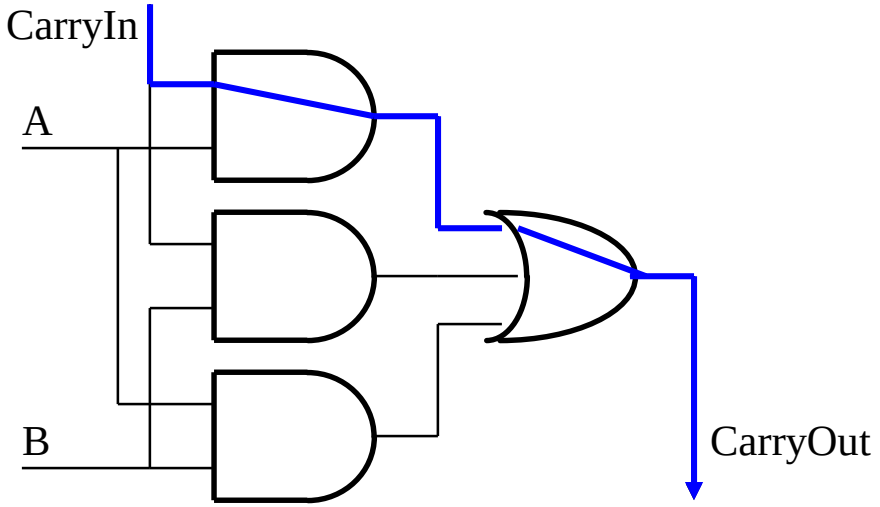
	Binvert	CIn	Oper
add?	0	0	10
sub?	1	1	10
and?	0	0	00
or?	0	0	01
beq?	1	1	10
slt?	1	1	11



The Disadvantage of Ripple Carry

- The adder we just built is called a “Ripple Carry Adder”
  - The carry bit may have to propagate from LSB to MSB
  - Worst case delay for an N-bit RC adder: 2N-gate delay





The point -> ripple carry adders are slow. Faster addition schemes are possible that *accelerate* the movement of the carry from one end to the other.