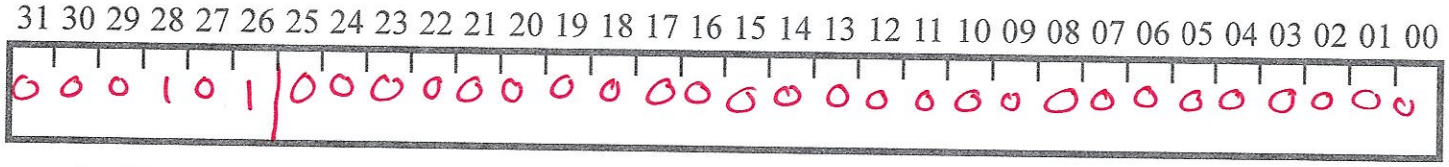
Review Problem 7

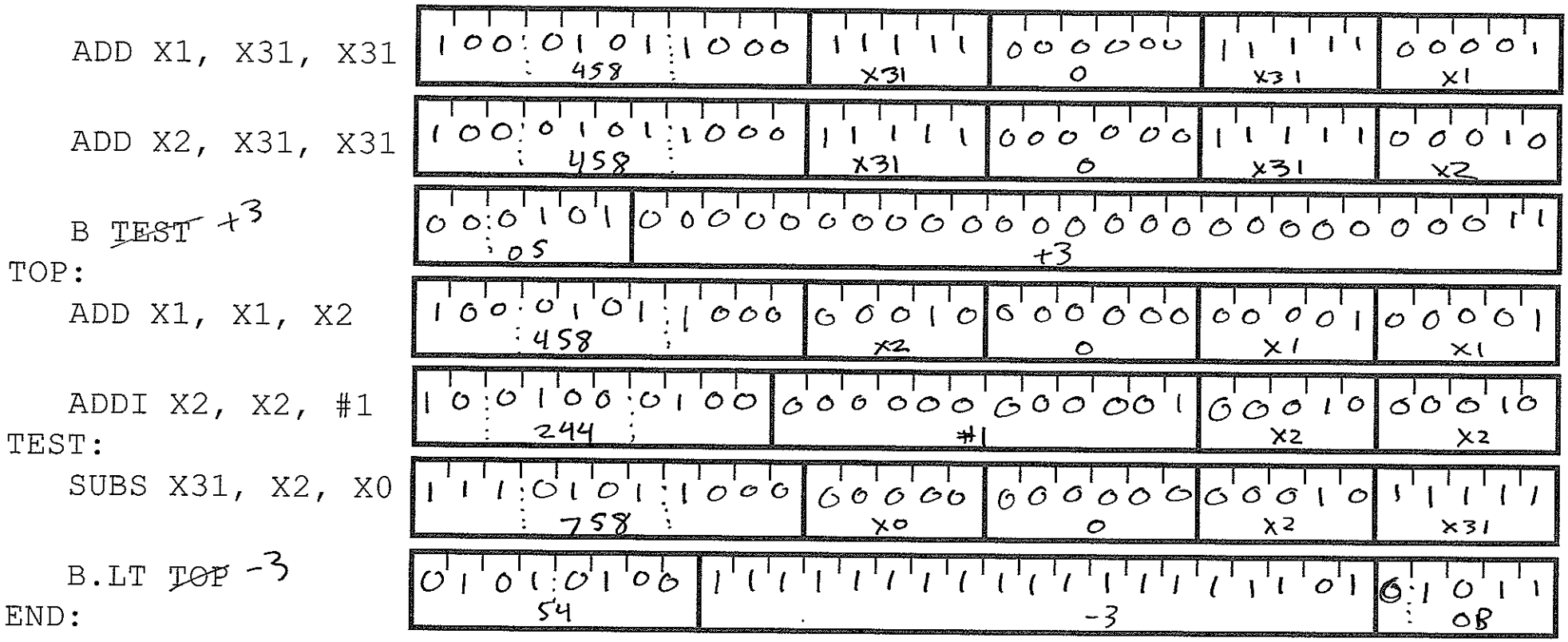
- Sometimes it can be useful to have a program loop infinitely. We can do that, regardless of location, by the instruction:
- LOOP: B LOOP
- Convert this instruction to machine code



B : 05

Conversion example

Compute the sum of the values 0…N-1



Assembly & Machine Language

Assembly

Simple instructions
Mnemonics for humans
(Almost) 1-to-1 relationship w/machine language

Machine Language

Numeric representations of instructions
Fixed format(s)
Directly controls CPU hardware

Computer Arithmetic

Readings: 3.1-3.3, A.5

Review binary numbers, 2's complement

Develop Arithmetic Logic Units (ALUs) to perform CPU functions.

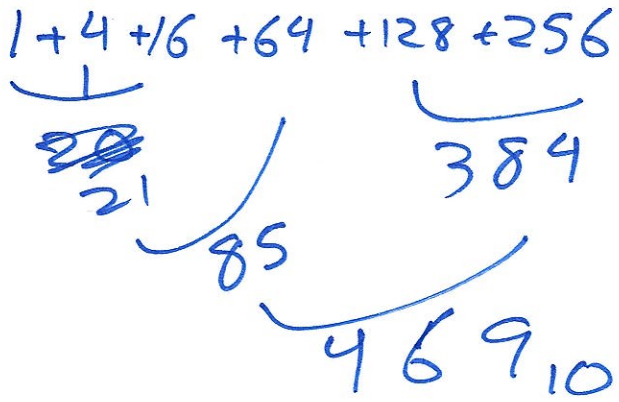
Introduce shifters, multipliers, etc.

Binary Numbers

Decimal: $469 = 4*10^2 + 6*10^1 + 9*10^0$

Binary: $01101 = 1*2^3 + 1*2^2 + 0*2^1 + 1*2^0 = (13)_{10}$

Example: $0111010101 = (?)_{10}$



2's Complement Numbers

Positive numbers & zero have leading 0, negative have leading 1

Negation: Flip all bits and add 1

13₂

Ex: $-(01101)_2 =$ $10010+1 = 10011_2$

To interpret numbers, convert to positive version, then convert:

11010 = $-(-11010)$
$= -(00101+1)$
$= -(00110)$
$= -(6)$
$= -6_{10}$

01100 = $+12_{10}$
+

Sign Extension

Conversion of n-bit to (n+m)-bit 2's complement: replicate the sign bit

$b_3b_2b_1b_0 = b_3b_3b_3b_2b_1b_0 = b_3b_3b_3b_3b_3b_3b_3b_3b_3b_3b_3b_2b_1b_0$

Ex - Convert to 8-bit: $01101 = (13)_{10}$ $11101 = (-3)_{10}$

00001101

11111101

correct?

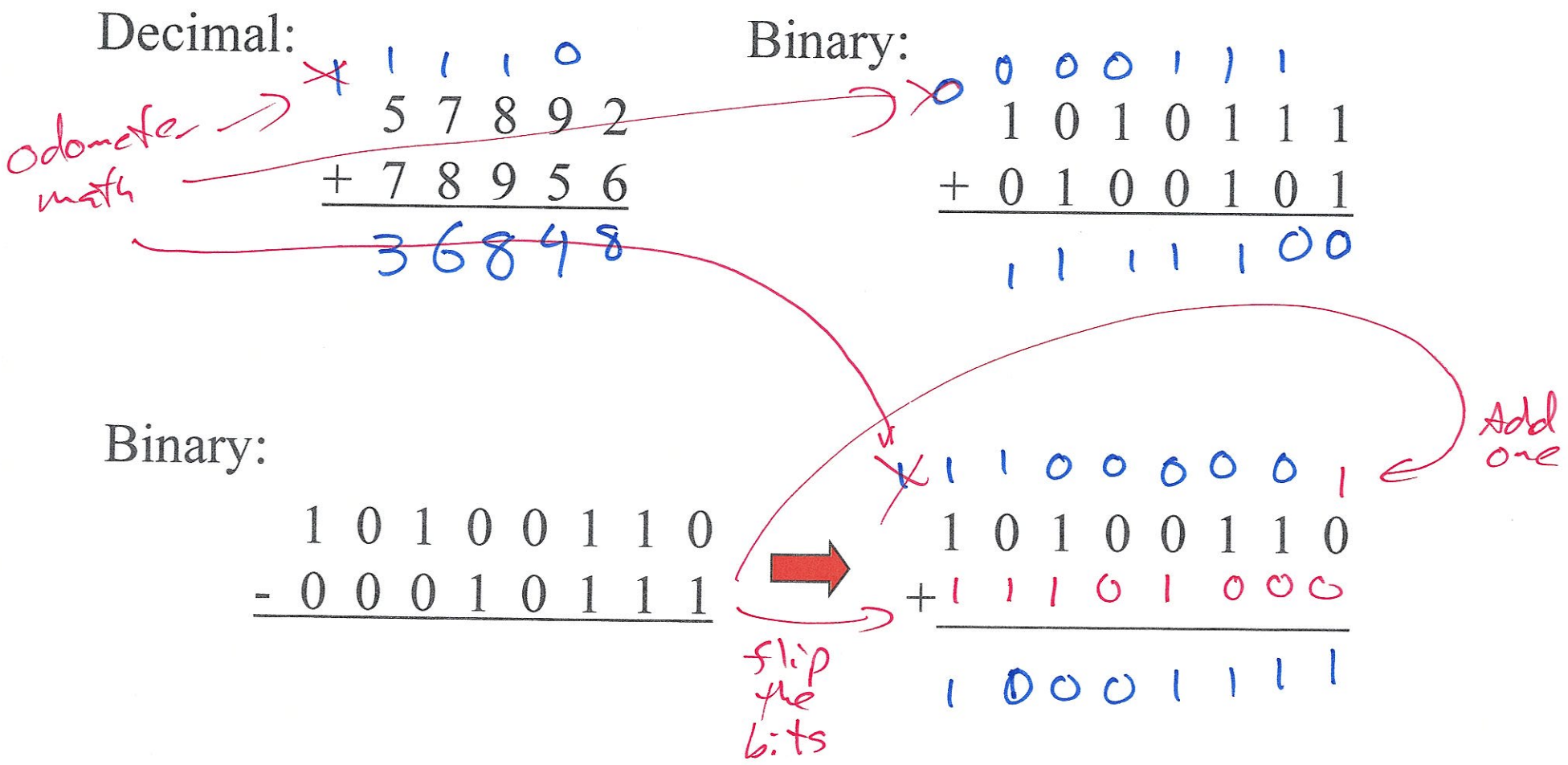
$= -(-11111101)$

$= -(00000010+1)$

$= -(00000011)$

$= -3$

Arithmetic Operations



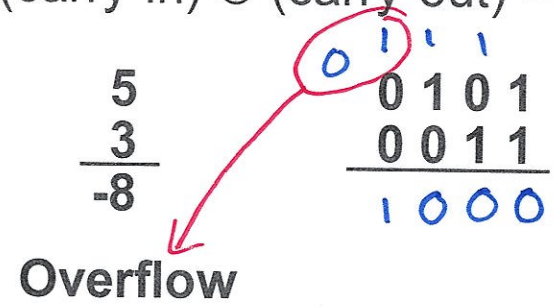
Overflows

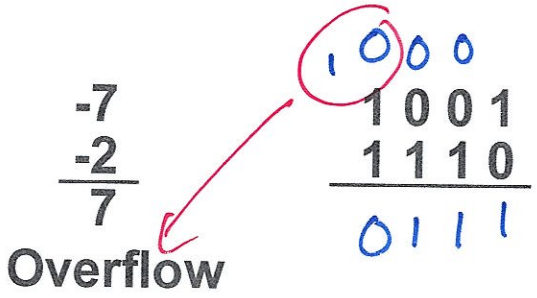
Operations can create a number too large for the number of bits

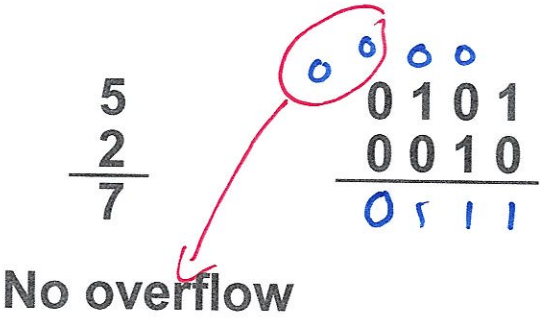
n-bit 2's complement can hold $-2^{(n-1)} \ldots 2^{(n-1)}-1$

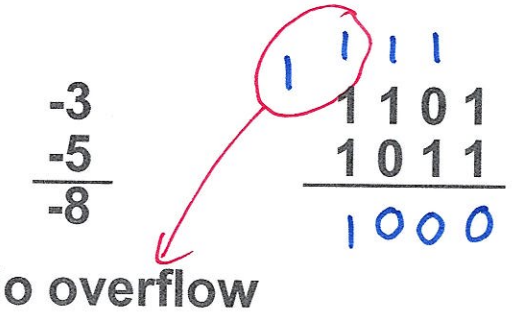
Can detect overflow in addition when highest bit has carry-in ≠ carry-out

(carry-in) ⊕ (carry-out) = 1





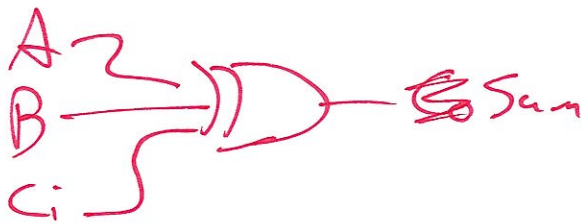


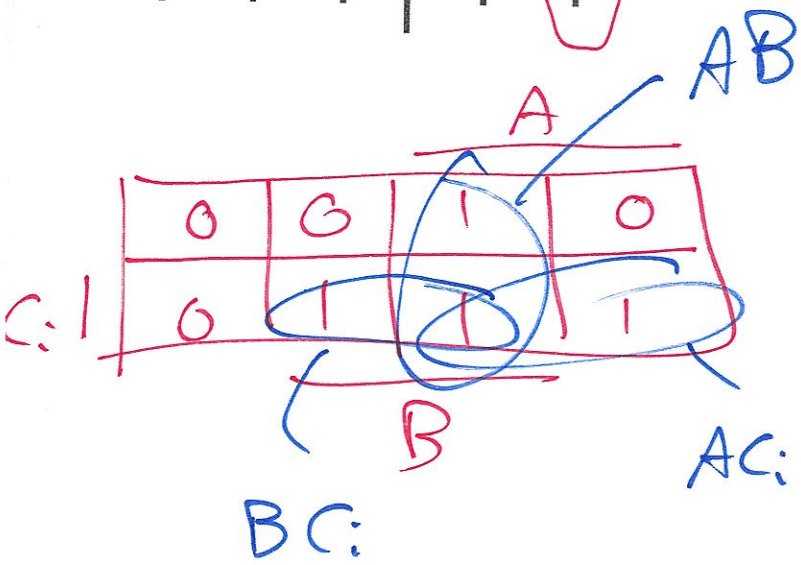


On an overflow, the top bit is flipped

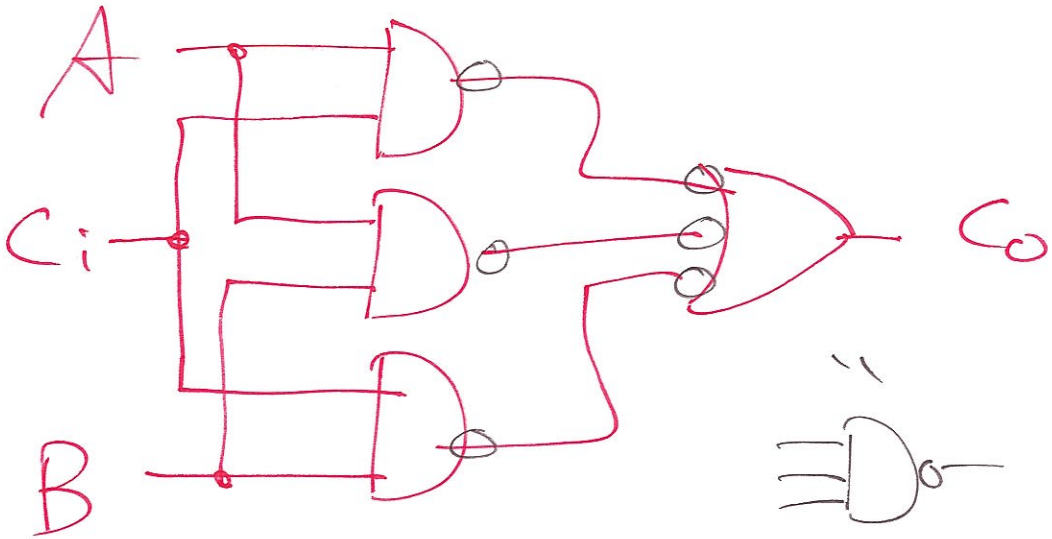
Full Adder

A	B	CI	CO	S
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





4xN ANDs



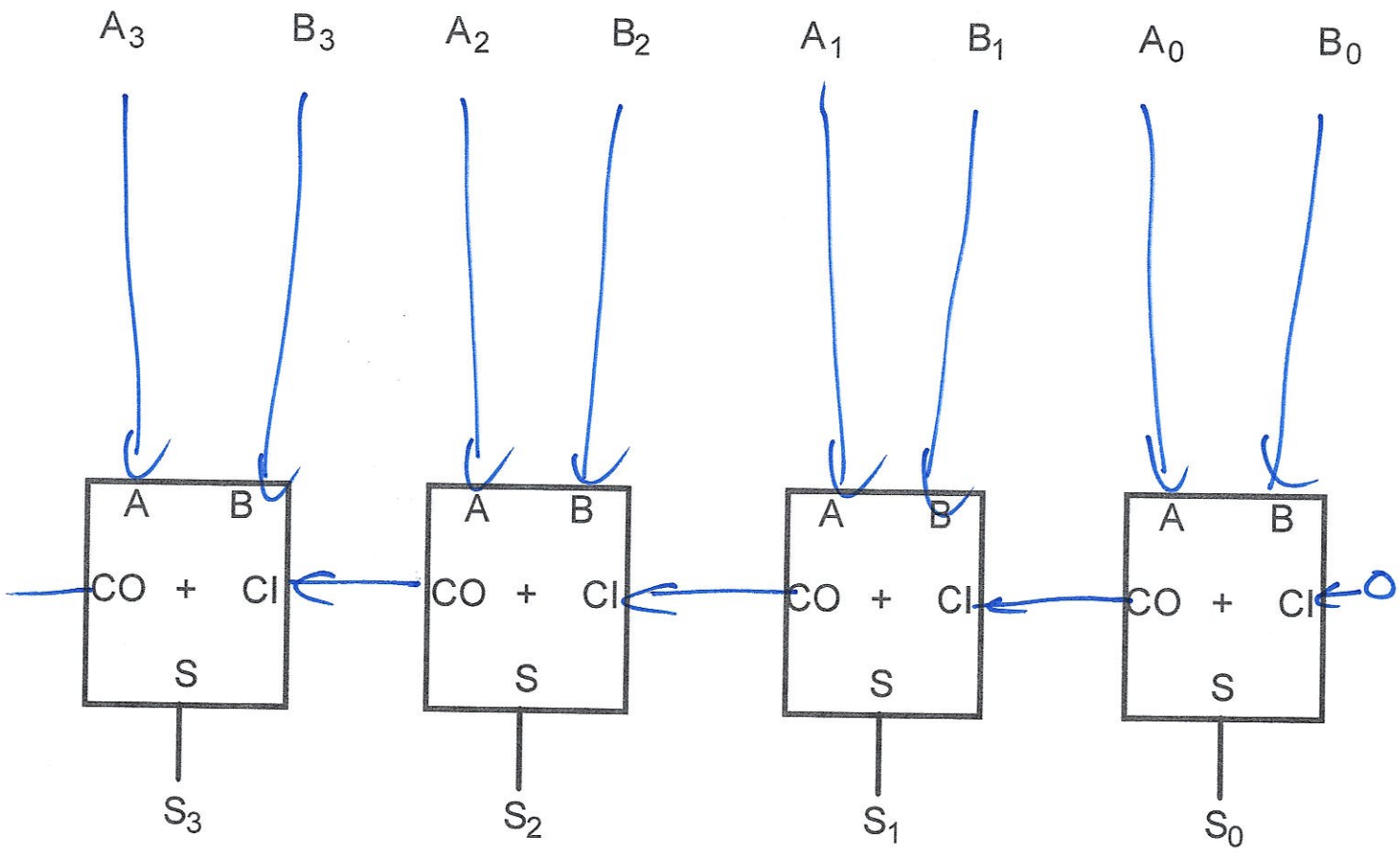
Multi-Bit Addition

C_4 C_3 C_2 C_1 0

A_3 A_2 A_1 A_0

\+ B_3 B_2 B_1 B_0

S_3 S_2 S_1 S_0



Adder/Subtractor

Subtract

0: A + B = = A + B + 0

1: A − B = A + (−B) = A + $\bar{B}$ + 1