

Mathematical Operations

- Base 2: the native base for computer systems
- Glyphs: 0, 1
- On a computer, we are limited to a certain number of digits bits.
- Recall, the results are summarized via the use of status flags.
 - For unsigned operations:
 - the final value is Zero (Z)
 - the calculation resulted in final carry (C)
 - For signed values
 - the final value is Negative (S)
 - the calculation resulted in an overflow (V) [used when signed numbers are involved]

Binary Addition:

- We have only two digits
 - $0 + 0 = 0$
 - $0 + 1 = 1$
 - $1 + 0 = 1$
 - $1 + 1 = ?$
- What do we do in base 10
 - $9 + 9 = ?$

$$\begin{array}{r} \boxed{} \boxed{} \boxed{1} \boxed{} \\ + \quad \boxed{} \boxed{} \boxed{} \boxed{9} \\ \hline \boxed{} \quad \boxed{} \boxed{} \boxed{} \boxed{8} \end{array}$$

		Base 2	
		B	
		0	1
A	0	0	1
	1	1	?

+	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	10
2	2	3	4	5	6	7	8	9	10	11
3	3	4	5	6	7	8	9	10	11	12
4	4	5	6	7	8	9	10	11	12	13
5	5	6	7	8	9	10	11	12	13	14
6	6	7	8	9	10	11	12	13	14	15
7	7	8	9	10	11	12	13	14	15	16
8	8	9	10	11	12	13	14	15	16	17
9	9	10	11	12	13	14	15	16	17	18

Binary Addition (1-digit):

- We have only two digits
 - $0 + 0 = 0$
 - $0 + 1 = 1$
 - $1 + 0 = 1$
 - $1 + 1 = \mathbf{10}$
- What do we do in base 10
 - $9 + 9 = \mathbf{18}$

$$\begin{array}{r} & \boxed{} & \boxed{} & \boxed{} & \boxed{} \\ & \boxed{} & \boxed{} & \boxed{} & \boxed{9} \\ \boxed{+} & \boxed{} & \boxed{} & \boxed{} & \boxed{9} \\ \hline & \boxed{\color{blue}{1}} & \boxed{} & \boxed{} & \boxed{} \end{array}$$

		Base 2	
		B	
		0	1
A	0	00	01
	1	01	10

A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

+ 0 1 2 3 4 5 6 7 8 9	0 0 1 2 3 4 5 6 7 8 9	1 1 2 3 4 5 6 7 8 9 10	2 2 3 4 5 6 7 8 9 10 11	3 3 4 5 6 7 8 9 10 11 12	4 4 5 6 7 8 9 10 11 12 13	5 5 6 7 8 9 10 11 12 13 14	6 6 7 8 9 10 11 12 13 14 15	7 7 8 9 10 11 12 13 14 15 16	8 8 9 10 11 12 13 14 15 16 17	9 9 10 11 12 13 14 15 16 17 18
+ 0 1 2 3 4 5 6 7 8 9	0 0 1 2 3 4 5 6 7 8 9	1 1 2 3 4 5 6 7 8 9 10	2 2 3 4 5 6 7 8 9 10 11	3 3 4 5 6 7 8 9 10 11 12	4 4 5 6 7 8 9 10 11 12 13	5 5 6 7 8 9 10 11 12 13 14	6 6 7 8 9 10 11 12 13 14 15	7 7 8 9 10 11 12 13 14 15 16	8 8 9 10 11 12 13 14 15 16 17	9 9 10 11 12 13 14 15 16 17 18
+ 0 1 2 3 4 5 6 7 8 9	0 0 1 2 3 4 5 6 7 8 9	1 1 2 3 4 5 6 7 8 9 10	2 2 3 4 5 6 7 8 9 10 11	3 3 4 5 6 7 8 9 10 11 12	4 4 5 6 7 8 9 10 11 12 13	5 5 6 7 8 9 10 11 12 13 14	6 6 7 8 9 10 11 12 13 14 15	7 7 8 9 10 11 12 13 14 15 16	8 8 9 10 11 12 13 14 15 16 17	9 9 10 11 12 13 14 15 16 17 18
+ 0 1 2 3 4 5 6 7 8 9	0 0 1 2 3 4 5 6 7 8 9	1 1 2 3 4 5 6 7 8 9 10	2 2 3 4 5 6 7 8 9 10 11	3 3 4 5 6 7 8 9 10 11 12	4 4 5 6 7 8 9 10 11 12 13	5 5 6 7 8 9 10 11 12 13 14	6 6 7 8 9 10 11 12 13 14 15	7 7 8 9 10 11 12 13 14 15 16	8 8 9 10 11 12 13 14 15 16 17	9 9 10 11 12 13 14 15 16 17 18
+ 0 1 2 3 4 5 6 7 8 9	0 0 1 2 3 4 5 6 7 8 9	1 1 2 3 4 5 6 7 8 9 10	2 2 3 4 5 6 7 8 9 10 11	3 3 4 5 6 7 8 9 10 11 12	4 4 5 6 7 8 9 10 11 12 13	5 5 6 7 8 9 10 11 12 13 14	6 6 7 8 9 10 11 12 13 14 15	7 7 8 9 10 11 12 13 14 15 16	8 8 9 10 11 12 13 14 15 16 17	9 9 10 11 12 13 14 15 16 17 18

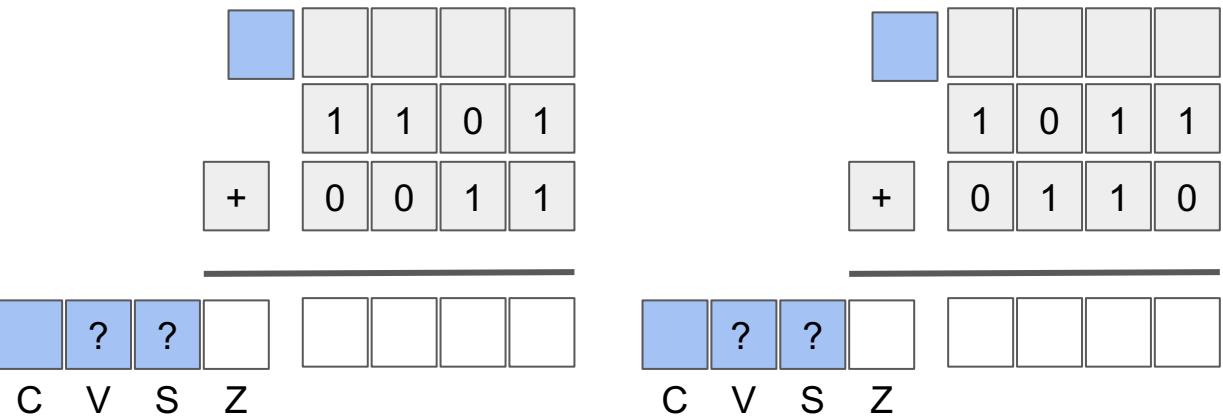
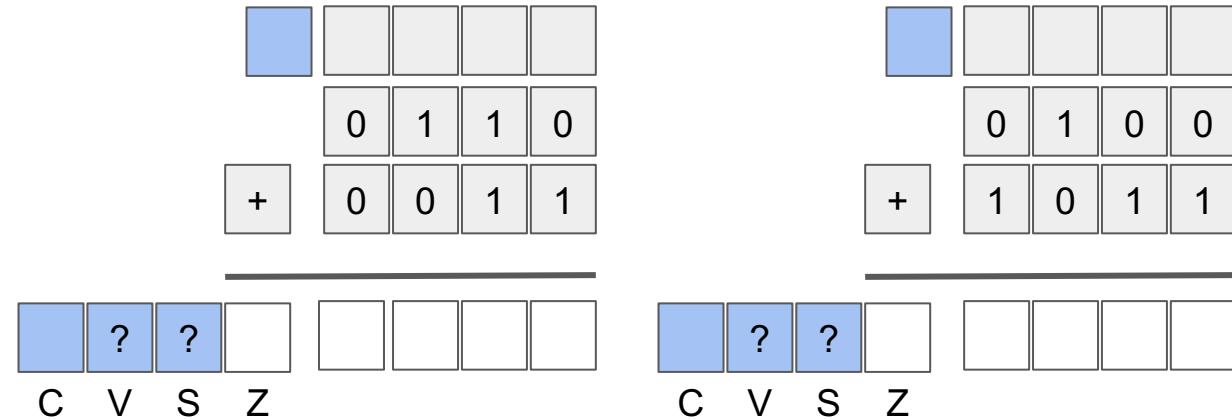
In Binary (before)

$$C + A + B = C, S$$

C	A	B	C	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0

$$C + A + B = C, S$$

C	A	B	C	s
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



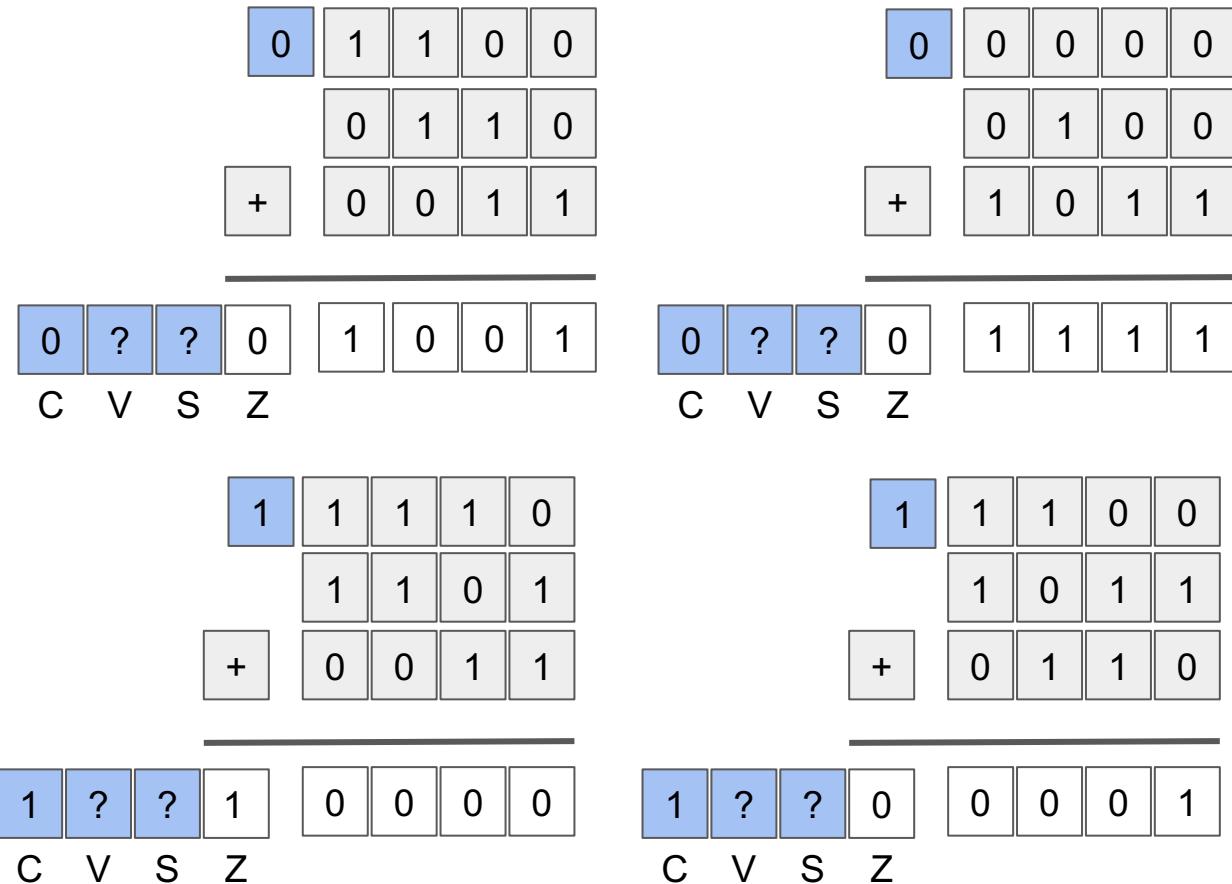
In Binary (after)

$$C + A + B = C, S$$

C	A	B	C	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0

$$C + A + B = C, S$$

C	A	B	C	s
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



Binary Addition: Practice

C + A + B = C, S				
C	A	B	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0

C + A + B = C, S				
C	A	B	C	S
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$$\begin{array}{r}
 \begin{array}{|c|} \hline 0 \\ \hline \end{array} \quad \begin{array}{|c|c|c|c|} \hline 1 & 1 & 1 & 0 \\ \hline 0 & 1 & 1 & 1 \\ \hline 0 & 0 & 1 & 1 \\ \hline \end{array} \\
 + \qquad\qquad\qquad \hline
 \begin{array}{|c|c|c|c|} \hline 0 & ? & ? & 0 \\ \hline 1 & 0 & 1 & 0 \\ \hline \end{array}
 \end{array}$$

$$\begin{array}{r}
 \begin{array}{|c|c|c|c|} \hline 1 & ? & ? & 0 \\ \hline 1 & 0 & 1 & 0 \\ \hline \end{array}
 \end{array}$$

$$\begin{array}{r}
 \begin{array}{|c|} \hline 1 \\ \hline \end{array} \quad \begin{array}{|c|c|c|c|} \hline \quad & \quad & \quad & \quad \\ \hline \quad & \quad & \quad & \quad \\ \hline \quad & \quad & \quad & \quad \\ \hline \quad & \quad & \quad & \quad \\ \hline \end{array} \\
 + \qquad\qquad\qquad \hline
 \begin{array}{|c|c|c|c|} \hline ? & ? & \quad & \quad \\ \hline \quad & \quad & \quad & \quad \\ \hline \quad & \quad & \quad & \quad \\ \hline \quad & \quad & \quad & \quad \\ \hline \end{array}
 \end{array}$$

$$\begin{array}{r}
 \begin{array}{|c|c|c|c|} \hline 1 & ? & ? & 0 \\ \hline 1 & 0 & 1 & 0 \\ \hline \end{array}
 \end{array}$$

$$\begin{array}{r}
 \begin{array}{|c|} \hline 1 \\ \hline \end{array} \quad \begin{array}{|c|c|c|c|} \hline 1 & 1 & 1 & 0 \\ \hline 0 & 1 & 1 & 1 \\ \hline 0 & 0 & 1 & 1 \\ \hline \end{array} \\
 + \qquad\qquad\qquad \hline
 \begin{array}{|c|c|c|c|} \hline 1 & ? & ? & 0 \\ \hline 1 & 0 & 1 & 0 \\ \hline \end{array}
 \end{array}$$

$$\begin{array}{r}
 \begin{array}{|c|c|c|c|} \hline \quad & \quad & \quad & \quad \\ \hline \quad & \quad & \quad & \quad \\ \hline \quad & \quad & \quad & \quad \\ \hline \quad & \quad & \quad & \quad \\ \hline \end{array}
 \end{array}$$

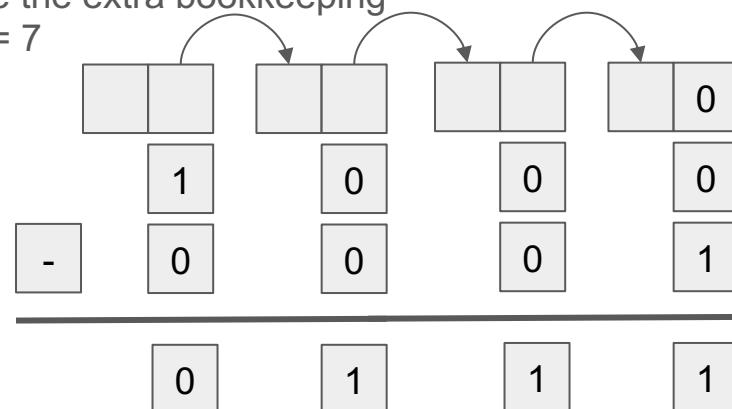
$$\begin{array}{r}
 \begin{array}{|c|} \hline 1 \\ \hline \end{array} \quad \begin{array}{|c|c|c|c|} \hline \quad & \quad & \quad & \quad \\ \hline \quad & \quad & \quad & \quad \\ \hline \quad & \quad & \quad & \quad \\ \hline \quad & \quad & \quad & \quad \\ \hline \end{array} \\
 + \qquad\qquad\qquad \hline
 \begin{array}{|c|c|c|c|} \hline ? & ? & ? & ? \\ \hline \quad & \quad & \quad & \quad \\ \hline \quad & \quad & \quad & \quad \\ \hline \quad & \quad & \quad & \quad \\ \hline \end{array}
 \end{array}$$

$$\begin{array}{r}
 \begin{array}{|c|c|c|c|} \hline 1 & ? & ? & 0 \\ \hline 1 & 0 & 1 & 0 \\ \hline \end{array}
 \end{array}$$

$$A - B = V$$

Binary Subtraction (via Borrow)

- Traditional Method \Rightarrow
 - Notice the extra squares
 - Notice the extra bookkeeping
 - $8 - 1 = 7$



- Recall Method of Complements
 - allows us to leverage binary addition
 - need a method to encode negative numbers

A	B	V
0	0	0
0	1	?
1	0	1
1	1	0

We need to borrow!

A	B	V
0	0	1
1	0	x

Again,
We need to borrow!

Base 10

$$\begin{array}{r}
 \overset{0}{x} \overset{9}{\cancel{0}} \overset{9}{\cancel{0}} \overset{10}{\cancel{0}} \\
 - \overset{0}{\cancel{0}} \overset{0}{\cancel{0}} \overset{0}{\cancel{0}} 1 \\
 \hline
 9 \ 9 \ 9
 \end{array}$$

Base 2

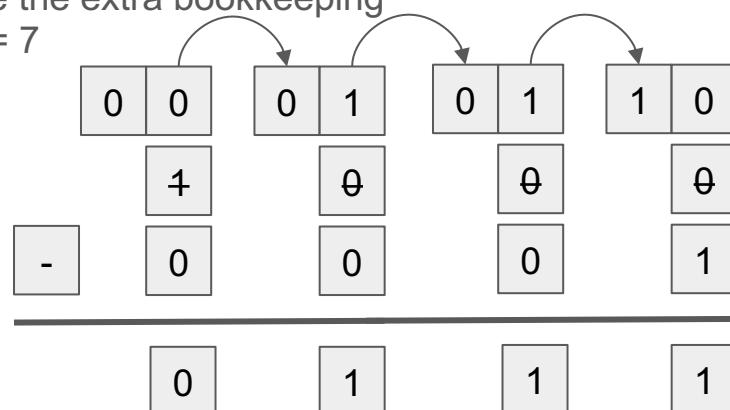
$$\begin{array}{r}
 \overset{0}{x} \overset{01}{\cancel{0}} \overset{01}{\cancel{0}} \overset{10}{\cancel{0}} \\
 - \overset{0}{\cancel{0}} \overset{0}{\cancel{0}} \overset{0}{\cancel{0}} 1 \\
 \hline
 0 \ 1 \ 1 \ 1
 \end{array}$$

$$A - B = V$$

Binary Subtraction (via Borrow)

- Traditional Method \Rightarrow

- Notice the extra squares
- Notice the extra bookkeeping
- $8 - 1 = 7$



A	B	V
0	0	0
0	1	?
1	0	1
1	1	0

We need to borrow!

A	B	V
0	0	1
1	0	x

Again,
We need to borrow!

- Recall Method of Complements

- allows us to leverage binary addition
- need a method to encode negative numbers

Base 10

$$\begin{array}{r}
 \overset{^o}{x} \overset{^9}{\cancel{8}} \overset{^9}{\cancel{8}} \overset{^{10}}{\cancel{8}} \\
 - \overset{^o}{0} \overset{^o}{0} \overset{^o}{0} \overset{^o}{1} \\
 \hline
 9 \ 9 \ 9
 \end{array}$$

Base 2

$$\begin{array}{r}
 \overset{^o}{x} \overset{^o}{\cancel{0}} \overset{^o}{\cancel{0}} \overset{^o}{1} \\
 - \overset{^o}{0} \overset{^o}{0} \overset{^o}{0} \overset{^o}{1} \\
 \hline
 0 \ 1 \ 1 \ 1
 \end{array}$$

V	$\sim V$
0	1
1	0

Method of Complements

- A technique to encode both positive and negative numbers
 - uses the same algorithm to perform addition, subtraction performed by the addition of complements
- Complement: *a thing that completes or brings to perfection:* $X + Y = 2^n$ (1 0...0)
- Radix 10:
 - 10's complement
 - $7 + x = 10$; $x = 3$
 - $46 + y = 100$; $y = 54$
 - 9's complement
 - $7 + a = 9$; $a = 2$
 - $46 + b = 99$; $b = 53$
- Radix 2:
 - 2's complement
 - $0111 + x = 1\ 0000$; $x = 1001$
 - $0010\ 1110 + y = 1\ 0000\ 0000$; $y = 1101\ 0010$
 - 1's complement
 - $0111 + a = 1111$; $a = 1000$
 - $0010\ 1110 + b = 1111\ 1111$; $b = 1101\ 0001$
- As we shall see, for Base 2
 - The 2's complement of X is: $-X$ ($\sim X + 1$)
 - The 1's complement of X is: $\sim X$

Method of Complements

A technique to encode both positive and negative numbers

- MSb used to denote the sign bit (0 positive, 1 negative)
- Table assumes a 4-bit represent

Use 1's complement to represent negative numbers

- Divide the number range in half
- Encode a positive and a negative value for each number
- Pros/cons:
 - ease to compute
 - positive and negative representations for zero

-7, -6, -5, -4, -3, -2, -1, -0, | 0, 1, 2, 3, 4, 5, 6, 7



1's Complement

	Positive	Negative
0	0000	1111
1	0001	1110
2	0010	1101
3	0011	1100
4	0100	1011
5	0101	1010
6	0110	1001
7	0111	1000
8		

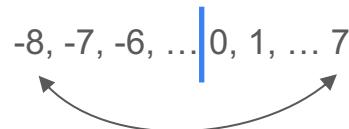
Method of Complements

A technique to encode both positive and negative numbers

- MSb used to denote the sign bit (0 positive, 1 negative)
- Table assumes a 4-bit represent

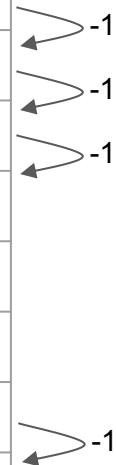
Use 2's complete to represent negative numbers

- Hold Zero as special
- Fold the resulting range to assign values
- Pros/cons:
 - Not symmetric: extra negative number
 - Need to flip all bits and add one to form the negative number
 - Consider then the predecessor of -8:



2's Complement

	Positive	Negative
0	0000	
1	0001	$1110 + 1 = 1111$
2	0010	$1101 + 1 = 1110$
3	0011	$1100 + 1 = 1101$
4	0100	$1011 + 1 = 1100$
5	0101	$1010 + 1 = 1011$
6	0110	$1001 + 1 = 1010$
7	0111	$1000 + 1 = 1001$
8		1000



Comparison of 1's and 2's Complement Encodings

1's Complement

	Positive	Negative
0	0000	1111
1	0001	1110
2	0010	1101
3	0011	1100
4	0100	1011
5	0101	1010
6	0110	1001
7	0111	1000

2's Complement

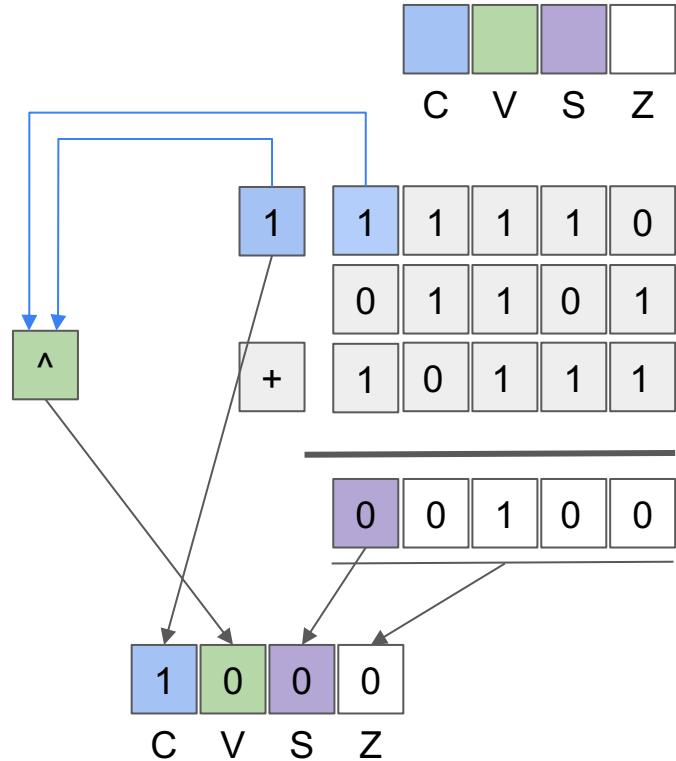
	Positive	Negative
0		0000
1	0001	1110 + 1 = 1111
2	0010	1101 + 1 = 1110
3	0011	1100 + 1 = 1101
4	0100	1011 + 1 = 1100
5	0101	1010 + 1 = 1011
6	0110	1001 + 1 = 1010
7	0111	1000 + 1 = 1001
8		1000

Status Flags Explained!

Example: $13 - 9 \Rightarrow 01101 + 10111$

* 9: 01001 \rightarrow -9: 10110 + 1 = 10111

- C: Carry Flag
 - the last step resulted in a carry value of 1
- V: Overflow Flag
 - the xor of the last two carry values: $c \wedge c'$
- S: Sign Flag
 - the MSB in the result is set (i.e., a 1)
- Z: Zero Flag
 - all bits in the result are cleared (i.e., 0)

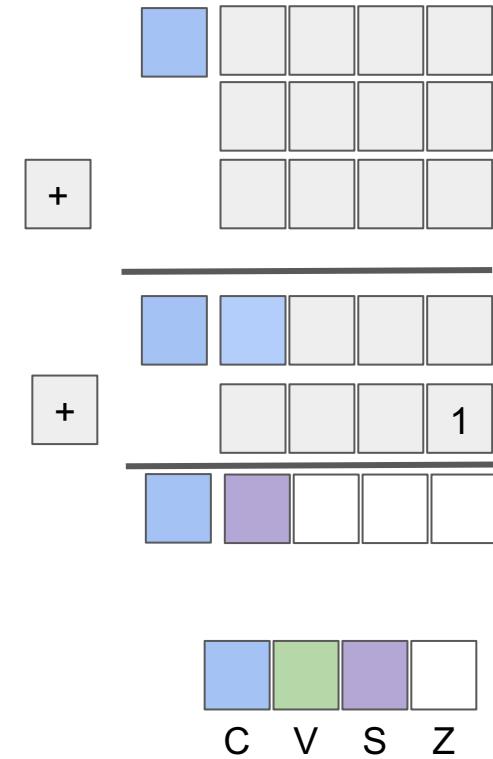
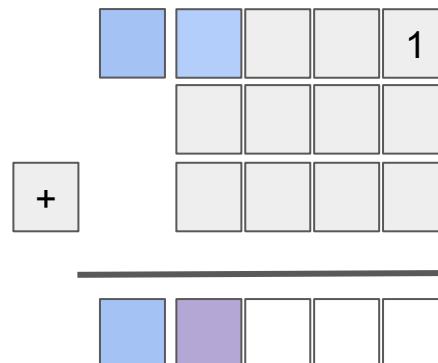


See: comp122/tidbits/status_bits_explained.gif

Algorithm: Subtraction via 1's Complements

Example: $13 - 9 \Rightarrow 0013 + - 0009$

1. Convert 13 and 9 into binary (01101 & 01001)
 2. Take the **1's complement** of the subtrahend (9)
 - o $01001 \rightarrow 10110$
 3. Add the complement to the minuend
 4. Drop the leading "1"
 5. Add 1
- Optimization:
 - introduce initial carry in



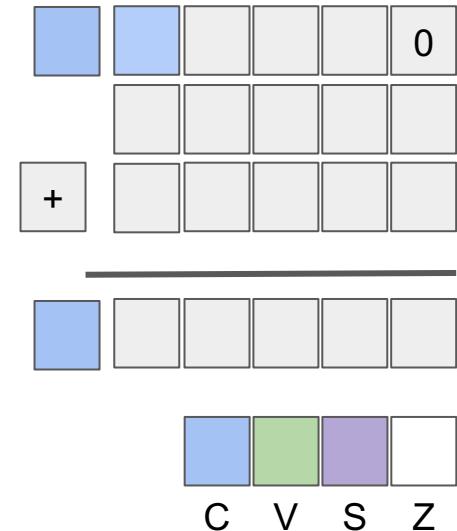
Algorithm: Subtraction via 2's Complement

Example: $13 - 9 \Rightarrow 00013 + -0009$

1. Encode 13 and 9 into binary (01101 & 01001)
2. Take the **2's complement** of the subtrahend (9)
 - o $01001 \rightarrow 10110 + 1 = 10111$
3. Add the complement to the minuend
4. Drop the leading "1", i.e., the carry bit.

Providing the answer:

- Optimization: Addition of adding one is baked in!

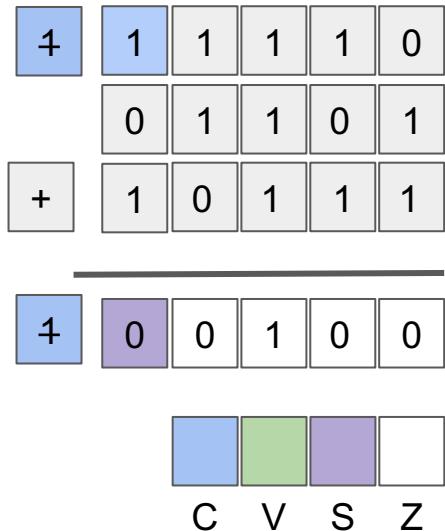


Algorithm: Subtraction via 2's Complement

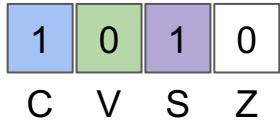
Example: $13 - 9 \Rightarrow 0013 + -0009$

1. Encode 13 and 9 into binary (01101 & 01001)
2. Take the 2's complement of the subtrahend (9)
 - o $01001 \rightarrow 10110 + 1 = 10111$
3. Add the complement to the minuend
4. Drop the leading "1", i.e., the carry bit.

Providing the answer: 4



- Optimization: Addition of adding one is baked in!



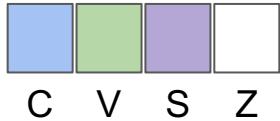
Practice: Addition and Subtraction

Steps:

1. transform the operation to addition
2. convert abs(X) to binary
3. encode both numbers in 2's complement
4. perform binary addition
5. convert result to decimal
 - o if negative result: compute 2's complement first

- $8 + 9 = 17$
- $7 - 4 = 3$
- $-5 + 2 = -3$
- $-16 - 3 = -19$
- $-25 - 4 = -29 \Rightarrow (-25) + (-4)$

1	1	1	1	0	0	0
1	0	0	1	1	1	1
+						
1	1	1	1	0	0	0
1	0	0	0	1	1	1



Practice: Addition and Subtraction

Steps:

1. transform the operation to addition
2. convert abs(X) to binary
3. encode both numbers in 2's complement
4. perform binary addition
5. convert result to decimal
 - o if negative result: compute 2's complement first

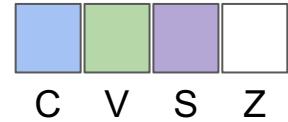
- $8 + 9 = 17$
- $7 - 4 = 3$
- $-5 + 2 = -3$
- $-16 - 3 = -19$

	0	1	0	0	0	0
	0	0	1	0	0	0
+	0	0	1	0	0	1

2. $8: 001000, 9: 001001$

3.

5. 17



Practice: Addition and Subtraction

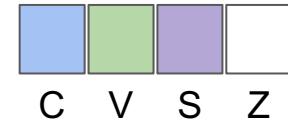
Steps:

1. transform the operation to addition
2. convert abs(X) to binary
3. encode both numbers in 2's complement
4. perform binary addition
5. convert result to decimal
 - if negative result: compute 2's complement first

- $8 + 9 = 17$
- $7 - 4 = 3$
- $-5 + 2 = -3$
- $-16 - 3 = -19$

	1	1	1	1	0	0	0
	0	0	0	1	1	1	1
+	1 1 1 1 0 0						
	0	0	0	0	1	1	1
	0	0	0	0	1	1	1

2. $7: 000111, 4: 000100$
 3. $-4: 111011+1 \Rightarrow 111100$
 5. 3



Practice: Addition and Subtraction

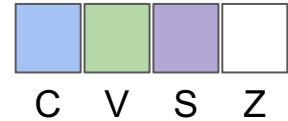
Steps:

1. transform the operation to addition
2. convert abs(X) to binary
3. encode both numbers in 2's complement
4. perform binary addition
5. convert result to decimal
 - if negative result: compute 2's complement first

- $8 + 9 = 17$
- $7 - 4 = 3$
- $-5 + 2 = -3$
- $-16 - 3 = -19$

0	0	0	0	1	0	0
0	0	0	0	0	1	0
+ 1	1	1	1	0	1	1
<hr/>						
1	1	1	1	0	1	

2. $5: 000101, 2: 000010$
 3. $-5: 111010+1 \Rightarrow 111011$
 5. $000010+1 \Rightarrow 000011 = -3$



Practice: Addition and Subtraction

Steps:

1. transform the operation to addition
2. convert abs(X) to binary
3. encode both numbers in 2's complement
4. perform binary addition
5. convert result to decimal
 - o if negative result: compute 2's complement first

- $8 + 9 = 17$
- $7 - 4 = 3$
- $-5 + 2 = -3$
- $-16 - 3 = -19$

1	1	0	0	0	0	0
1	1	0	0	0	0	0
+						1
						1
1	0	1	1	0	1	

- | | |
|--|--------------|
| 2. $16: 010000$ | $3: 000011$ |
| 3. $-16: 110000$ | $-3: 111101$ |
| 5. $010010+1 \Rightarrow 010011 = -19$ | |