

Representing Negative Numbers

Most computer architectures use 2's complement.

Consider the case for 8-bits.

We represent:

Positive numbers	Binary Representation		
0_{10}	0000	0000	"all 0s"
1_{10}	0000	0001	"counting forwards"
2_{10}	0000	0010	
\vdots	\vdots	\vdots	
43_{10}	0010	1011	
\vdots	\vdots	\vdots	
127_{10}	0111	1111	

smallest positive = 1, largest = $2^7 - 1 = 127$

Negative numbers	Binary Representation		
-1_{10}	1111	1111	"all 1s"
-2_{10}	1111	1110	"counting backwards"
-3_{10}	1111	1101	
\vdots	\vdots	\vdots	
-43_{10}	1101	0101	
\vdots	\vdots	\vdots	
-128	1000	0000	

smallest negative = $-2^7 = -128$

2's complement (Appendix D of Antonaros) 2

To represent a negative number $-N$, we compute the 2's complement of the positive number N .

This is done in two steps:

1. Complement each bit of N
(Replace 1 by 0, replace 0 by 1).

2. Add 1 to the result of step 1.

Example 1

$$1_{10} = (0000\ 0001)_2$$

$$\begin{array}{r} 0000\ 0001 \\ 1111\ 1110 \leftarrow \text{complement each bit} \\ \hline 1111\ 1111 \leftarrow \text{Add 1} \\ \hline \textcircled{1}111\ 1111 \leftarrow \text{representing } -1 \end{array}$$

NB: The leftmost bit is (the most significant bit) always 1 for negative numbers, and always 0 for positive numbers.

Example 2

Find the 2's complement representation of -43_{10} .

Step 1. Write 43 in binary

Quotient	Remainder
$\text{int}(43/2) = 21$	1
$\text{int}(21/2) = 10$	1
$\text{int}(10/2) = 5$	0
$\text{int}(5/2) = 2$	1
$\text{int}(2/2) = 1$	0
0	1

$$43 = (00101011)_2$$

$$\begin{array}{r} \{ 11010100 \\ \quad \quad \quad 1+ \\ \hline \underline{\underline{11010101}} \end{array}$$

Complement each digit

for -43 .

Example 3

- Verify $-(-5) = 5$ in the sense that:

Two's - complement (Two's - complement (N)) = N.

$$\begin{array}{r} 5 = 0000\ 0101 \\ \left\{ \begin{array}{r} 1111\ 1010 \leftarrow \text{complement each 1} \\ \quad \quad \quad 1+ \\ \hline 1111\ 1011 \leftarrow \text{Two's - complement (5)} \\ 0000\ 0100 \\ \quad \quad \quad 1+ \\ \hline 000\ 0101 \leftarrow 5 \quad \checkmark \end{array} \right. \end{array}$$

Notes:

- ① We can represent -128_{10} , but not 128_{10} .
- ② Leftmost bit is 0 for positives and it is 1 for negatives. It is called the sign bit.
- ③ \checkmark It doesn't matter whether N is positive or negative, we find its -N representation in the same way.
 $x - N = x + (N)_{\text{complement}}$
- ④ To subtract a number N_2 from N_1 , we add $N_1 + 2\text{'s-complement}(N_2) = N_1 - N_2$.

Subtraction

We subtract 5 from 13 using:

$$13 - 5 = 13 + \text{Two's-complement}(5)$$

$$\begin{aligned}(13)_{10} &= (8 + 4 + 1)_{10} \\ &= 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^0 \\ &= (0000 \quad 1101)_2\end{aligned}$$

$$\begin{aligned}(5)_{10} &= (4 + 1)_{10} \\ &= (1 \cdot 2^2 + 1 \cdot 2^0)_{10} \\ &= (101)_2 \\ &= (0000 \quad 0101)_2\end{aligned}$$

Compute Two's-complement(5):

$$\begin{array}{r} 0000 \quad 0101 \\ \{ 1111 \quad 1010 \quad \text{complement} \\ \hline 1111 \quad 1011 \quad \text{represents } -5. \end{array}$$

9 bits:

$$\begin{array}{r}
 00110000 \\
 00010101 \\
 \hline
 01000101 \leftarrow 139
 \end{array}$$

$$\begin{array}{r}
 96_{10} = 0110\ 0000 \\
 43_{10} = 0010\ 1011 \\
 \hline
 1000\ 1011
 \end{array}$$

Carry in,
but no
carry out.

Which number is this?
 Since the most significant bit is 1,
 the number is negative.

Use 2's complement to figure out the number:

$$\begin{array}{r}
 1000\ 1011 \\
 0111\ 0100 \\
 \hline
 0111\ 0101
 \end{array}$$

Complement
+ 1
is 117.

Hence, as far as the machine is concerned, for 8-bit representations,

$$96_{10} + 43_{10} = -117_{10}$$

an overflow (not enough bits to represent the result).

Rule: Overflow bit: Carry-in to MSB XOR Carry-out from MSB
 1 = Yes, 0 = No

Overflow Cases

Case 1:

Adding 2 positives yields a negative

the previous example 8.
 $96_{10} + 43_{10}$

Case 2:

Adding 2 negatives yields a positive

Case 3:

Negative minus positive yields positive
 $(-128) - 127 = 1$ Similar to case 2

Case 4:

Positive minus negative yields negative
 $96 - (-43) =$

Recall: A number is positive if its most significant bit is zero. A number is negative if its most significant bit is one.

Signed Arithmetic:

- Compute if overflow exists or not
- ignore the carry-out

Unsigned Arithmetic

- Compute carry-out
- ignore overflow

If adding unsigned numbers:

Overflow = carry-out of last bit.

Based on the overflow description,

- we have (for 2's complement arithmetic)

Case 1:

$$\begin{array}{r}
 0d_6 \dots d_0 \quad +ve \\
 + \quad 0b_6 \dots b_0 \quad +ve \\
 \hline
 1s_6 \dots s_0 \quad -ve
 \end{array}$$

We see a carry in w/out carry out.

Case 2:

$$\begin{array}{r}
 1d_6 \dots d_0 \quad -ve \\
 + \quad 1b_6 \dots b_0 \quad -ve \\
 \hline
 0s_6 \dots s_0 \quad +ve
 \end{array}$$

Again: carry out, w/out carry in.

Case 3:

$$\begin{array}{r}
 1d_6 \dots d_0 \quad -ve \\
 + \quad 1b_6 \dots b_0 \\
 \hline
 0s_6 \dots s_0
 \end{array}$$

-ve, from - (+ve)

carry out w/out carry in.

Note SKIP!!! that: if computing the 2's complement generates an overflow, then we must stop and detect it.

We have two solutions:

① Compute overflow when computing 2's complement, or:

② Use 1's complement for negative numbers, and add 1 to the result:

$$\begin{array}{r}
 0000\ 0000 \\
 +\ 0111\ 1111 \\
 \hline
 0111\ 1111 \\
 +\ 1 \\
 \hline
 1000\ 0000
 \end{array}$$

1's complement of 128

Again Cin, no Cout.

84
- If adding unsigned numbers
overflow = carry-out of
~~the~~ last bit.

Sign Extension

Suppose that we have two numbers of different numbers of bits. How can we combine the numbers?

Solution: Sign-extend!

Consider adding:

$X = -2$ represented in 8 bits,

$Y = +256_{10}$ represented in 16 bits

Solution:

Sign-extend X from 8 to 16 bits.

This means:

- (i) If $X \geq 0$, fill-up the leftmost bits by 0s (the sign-bit of X)
- (ii) If $X < 0$, fill-up the leftmost bits by 1s (the sign-bit of X).

$$2 = 0000 \ 0010$$

$$1111 \ 1101 \leftarrow \text{complement}$$

$$\begin{array}{r} 1111 \ 1101 \\ + \\ \hline \end{array}$$

$$\textcircled{1}111 \ 1110 \leftarrow \text{represent } -2 \text{ in 8-bits.}$$

This is the sign bit to duplicate.

Then, -2 in 16-bits

$$\begin{array}{r} 1111 \ 1111 \ 1111 \ 1110 \quad -2, \\ 0000 \ 0001 \ 0000 \ 0000 \quad +256 \\ \hline 10000 \ 0000 \ 1111 \ 1110 \quad +254 \end{array}$$

can be added as before.

Otherwise

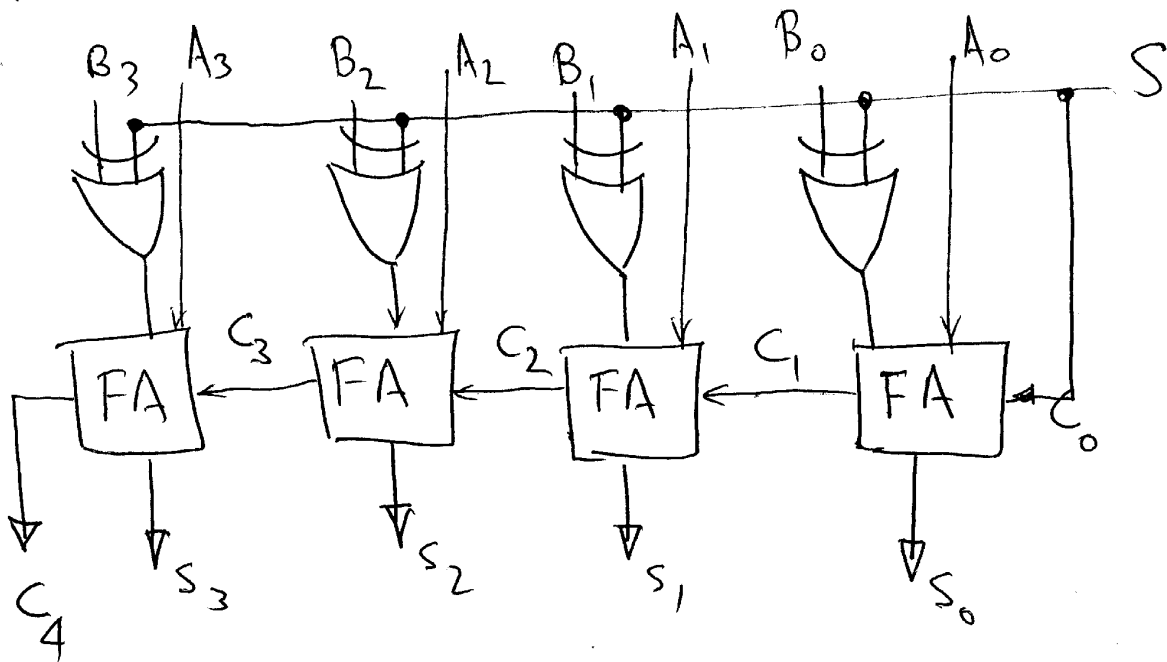
$$\begin{array}{r} 0000 \ 0000 \ 1111 \ 1110 \\ 0000 \ 0001 \ 0000 \ 0000 \\ \hline 0000 \ 0001 \ 1111 \ 1110 \end{array}$$

$$\begin{array}{r} 256 \\ +254 \\ \hline 510 \end{array}$$

Wrong!

3-10
P138

Adder/subtractor circuit:



borrow out

for $S=0$, it is an adder:

$$(S_3 S_2 S_1 S_0) = (A_3 A_2 A_1 A_0) + (B_3 B_2 B_1 B_0)$$

since $B_n \oplus 0 = B_n$

for $S=1$, it is a subtractor

$$(S_3 S_2 S_1 S_0) = (A_3 A_2 A_1 A_0) - (B_3 B_2 B_1 B_0)$$

since $(B_3 B_2 B_1 B_0)$ $B_n \oplus 1 = \overline{B_n}$

is 1-comp: $(B'_3 B'_2 B'_1 B'_0)$ from: $1 \oplus 1 = 0$, $0 \oplus 1 = 1$
 And through C_0 , 1 is added to
 get the 2's complement.