Carry and Overflow

Young W. Lim

2024-04-27 Sat

Outline

Based on

1. "Self-service Linux: Mastering the Art of Problem Determination", Mark Wilding
1. "Computer Architecture: A Programmer's Perspective", Bryant & O'Hallaron

Compling 32-bit program on 64-bit gcc

- `gcc -v`
- `gcc -m32 t.c`
- `sudo apt-get install gcc-multilib`
- `sudo apt-get install g++-multilib`
- `gcc-multilib`
- `g++-multilib`
- `gcc -m32`
- `objdump -m i386`

TOC: Overview

Carry flag and overflow flag

- considering carry and overflow flags in x86
- do not confuse the carry flag
  with the overflow flag
  in integer arithmetic.
- the *ALU* always sets these flags appropriately
  when doing any integer math.
- these flags can occur on its *own*, or *both* together.

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Signed and unsigned computations

- the CPU's ALU doesn't care or know
  whether signed or unsigned computations are performed;
- the *ALU* just performs integer arithmetic and
  sets the flags appropriately.
- It's up to the *programmer* to know
  which flag to check after the arithmetic is done.

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Flags for an unsigned number

- if a word is treated as an unsigned number,
  - the carry flag must be used to check
    if the result is fit into n-bit or $(n+1)$-bit number
  - the overflow flag is *irrelevant*
    to an unsigned number arithmetic

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Flags for a signed number

- if a word is treated as an signed number,
  - the carry flag is *irrelevant*
    to an signed number arithmetic
  - the overflow flag must be used to check
    if the result is wrong or not

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Detecting errors in usigned and signed arithmetic (1)

	unsigned integer arithmetic	signed integer arithmetic
CF Carry Flag	detects *overflows* extends an *n-bit* result into an (*n+1*)-bit result	
OF Overflow Flag		detects *overflows* errors the result cannot be used

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Detecting errors in usigned and signed arithmetic (2)

- unsigned integer arithmetic *overflow*
  is indicated by the carry flag
  - $P + P$ `CF=1` → carry out – the result is too large for an *n-bit* integer
  - $P - P$ `CF=1` → borrow in – the result is too small for an *n-bit* integer
- signed integer arithmetic *overflow*
  is indicated by the overflow flag
  - $P + P \rightarrow N$ `OF=1` → overflow – the result is not correct
  - $N + N \rightarrow P$ `OF=1` → overflow – the result is not correct
- P (positive), N (negative)

https://stackoverflow.com/questions/47333458/assembly-x86-64-setting-carry-flag-f

- unsigned integer arithmetic *overflow* is indicated by the carry flag
  - the *overflowed* n-bit result can be extended into $(n+1)$-bit result by using the carry flag
- signed integer arithmetic *overflow* is indicated by the overflow flag
  - the *overflowed* n-bit result cannot be used

`https://stackoverflow.com/questions/47333458/assembly-x86-64-setting-carry-flag-fo`

The verb to overflow v.s. the overflow flag (1)

- Do not confuse the English verb *to overflow*
  with the overflow flag in the ALU.
- The verb *to overflow* is used casually to indicate that
  some math result doesn't fit in the number of bits available;
- it could be integer math, or floating-point math, or whatever.
- The overflow flag is set specifically by the ALU
  it isn't the same as the casual English verb "to overflow"

`http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt`

The verb to overflow v.s. the overflow flag (2)

- In English, we may say
  "the binary/integer math overflowed
  the number of bits available for the result,
  causing the carry flag to come on".
- Note how this English usage of the verb "to overflow"
  is not the same as saying the overflow flag is on".
- A math result can overflow (the verb)
  the number of bits available
  without turning on the ALU overflow flag

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Computing Carry and Overflow Flags

CF (carry flag) and OF (overflow flag) computation

ADD (addition)	SUB (subtraction)
CF = C_n	CF = $\overline{C_n}$
OF = $C_n \bigoplus C_{n-1}$	OF = $C_n \bigoplus C_{n-1}$
a 2's complement addition $A + B = A + B + 0$	a transformed addition $A - B = A + \overline{B} + 1$
$\{C_n, S_{n-1}\} = a_{n-1} + b_{n-1} + c_{n-1}$	$\{C_n, S_{n-1}\} = a_{n-1} + \overline{b_{n-1}} + c_{n-1}$
$\{C_{n-1}, S_{n-2}\} = a_{n-2} + b_{n-2} + c_{n-2}$	$\{C_{n-1}, S_{n-2}\} = a_{n-2} + \overline{b_{n-2}} + c_{n-2}$

https://www.csie.ntu.edu.tw/~cyy/courses/assembly/12fall/lectures/handouts/lec14_

TOC: Carry flag

TOC: Examples of signed and unsigned integer arithmetic

Examples of interpreting signed and unsigned numbers (1)

- interpreting 0xFFFFBDC3

as an unsigned (positive) number	+0xFFFFBDC3	$+4294950339_{10}$
as a signed (negative) number	-0x0000423D	-16957_{10}

https://stackoverflow.com/questions/47333458/assembly-x86-64-setting-carry-flag-fo

Examples of interpreting signed and unsigned numbers (2)

- interpreting 0xFFFFBDC3
  - as an unsigned (positive) number | +0xFFFFBDC3 | $+4294950339_{10}$ |

    $15 * 16^7 + 15 * 16^6 + 15 * 16^5 + 15 * 16^4$
    $+11 * 16^3 + 13 * 16^2 + 12 * 16^1 + 3 * 16^0$
  - as a signed (negative) number | -0x0000423D | -16957_{10} |

    $0 * 16^7 + 0 * 16^6 + 0 * 16^5 + 0 * 16^4$
    $+4 * 16^3 + 2 * 16^2 + 3 * 16^1 + 13 * 16^0$

https://stackoverflow.com/questions/47333458/assembly-x86-64-setting-carry-flag-fo

Examples of interpreting signed and unsigned numbers (3)

- the 2's complement of 0xFFFFBDC3 : 0x0000423D ($= +16957_{10}$)

```
                  F    F    F    F    B    D    C    3
0xFFFFBDC3    0x1111_1111_1111_1111_1011_1101_1100_0011
0x0000423D    0x0000_0000_0000_0000_0100_0010_0011_1100 (1's complement)
0x0000423D    0x0000_0000_0000_0000_0100_0010_0011_1101 (2's complement)
                0    0    0    0    4    2    3    D
```

- the 2's complement of 0x0000423D : 0xFFFFBDC3 ($= -16957_{10}$)

```
                  0    0    0    0    4    2    3    D
0x0000423D    0x0000_0000_0000_0000_0100_0010_0011_1101
0x0000BDC2    0x1111_1111_1111_1111_1011_1101_1100_0010 (1's complement)
0xFFFFBDC3    0x1111_1111_1111_1111_1011_1101_1100_0011 (2's complement)
                  F    F    F    F    B    D    C    3
```

https://stackoverflow.com/questions/47333458/assembly-x86-64-setting-carry-flag-fo

Examples of signed and unsigned integer arithmetic

- subtracting 0x0000618D from 0x0000195D

`0x0000195D - 0x0000618D`	unsigned subtraction
subtraction by hand	
`0x0000195D + (-0x0000618D)`	signed subtraction
the *transformed addition* using the 2's complement of subtrahend	

https://stackoverflow.com/questions/47333458/assembly-x86-64-setting-carry-flag-fo

2's complements

- the 2's complement of 0x0000618D : 0xFFFF8E73 (= -24973_{10})

```
                  F    F    F    F    8    E    7    3
0xFFFF9E73   0x1111_1111_1111_1111_1001_1110_0111_0011
0x0000617C   0x0000_0000_0000_0000_0110_0001_1000_1100 (1's complement)
0x0000618D   0x0000_0000_0000_0000_0110_0001_1000_1101 (2's complement)
                  0    0    0    0    6    1    8    D
```

- the 2's complement of 0xFFFF8E73 : 0x0000618D (= $+24973_{10}$)

```
                  0    0    0    0    6    1    8    D
0x0000618D   0x0000_0000_0000_0000_0110_0001_1000_1101
0xFFFF9E72   0x1111_1111_1111_1111_1001_1110_0111_0010 (1's complement)
0xFFFF9E73   0x1111_1111_1111_1111_1001_1110_0111_0011 (2's complement)
                  F    F    F    F    8    E    7    3
```

https://stackoverflow.com/questions/47333458/assembly-x86-64-setting-carry-flag-fo

Unsigned subtraction

- 0x0000195D - 0x0000618D : unsigned subtraction

  subtraction by hand

```
                          0    0    0    0    1    9    5    D
  0x0000195D      0x0000_0000_0000_0000_0001_1001_0101_1101
- 0x0000618D      0x0000_0000_0000_0000_0110_0001_1000_1101
                          0    0    0    0    6    1    8    D
 ----------------------------------------------------------
  0xFFFFB7D0    1 0x1111_1111_1111_1111_1011_0111_1101_0000 (hand subtraction)
                1         F    F    F    F    B    7    D    0
                .
                V borrow (CF=1) : unsigned integer overflow
```

  - A borrow is indicated by the carry flag (CF=1)
    - whenever an unsigned integer overflow happened
    - $A - B$, when $A < B$, for non-negative integers A, B

https://stackoverflow.com/questions/47333458/assembly-x86-64-setting-carry-flag-fo

Signed subtraction

- 0x0000195D + (-0x0000618D) : signed subtraction
  the *transformed addition* using the 2's complement of subtrahend

```
                        0    0    0    0    1    9    5    D
  0x0000195D     0x0000_0000_0000_0000_0001_1001_0101_1101 (+0x0000195D)
+ 0xFFFF9E73     0x1111_1111_1111_1111_1001_1110_0111_0011 (-0x0000618D)
                        F    F    F    F    9    E    7    3
 ----------------------------------------------------------
  0xFFFFB7D0  0  0x1111_1111_1111_1111_1011_0111_1101_0000 (hand addition)
              0         F    F    F    F    B    7    D    0
 -0x00004830  .  0x0000_0000_0000_0000_0100_1000_0011_0000 (2's complement)
              .         0    0    0    0    4    8    3    0
              V  no carry in the transformed addition (Cn=0) --> (CF=1)
```

  - signed integer overflow is indicated by the overflow flag (OF)
    - the carry flag is set by the inverted carry of a transformed addition

https://stackoverflow.com/questions/47333458/assembly-x86-64-setting-carry-flag-fo

Interpreting the result as a signed or an unsigned integer

- subtracting 0x0000618D from 0x0000195D
  the results of unsigned and signed subtractions have
  the same bit pattern 0xFFFFB7D0

- the 2's complement of 0xFFFFB7D0 : 0x00004830 ($= +18480_{10}$)

```
                    F    F    F    F    B    7    D    0
0xFFFFB7D0     0x1111_1111_1111_1111_1011_0111_1101_0000
0x0000482F     0x0000_0000_0000_0000_0100_1000_0010_1111 (1's complement)
0x00004830     0x0000_0000_0000_0000_0100_1000_0011_0000 (2's complement)
                    0    0    0    0    4    8    3    0
```

- the 2's complement of 0x00004830 : 0xFFFFB7D0 ($= -18480_{10}$)

```
                    0    0    0    0    4    8    3    0
0x00004830     0x0000_0000_0000_0000_0100_1000_0011_0000
0xFFFFB7CF     0x1111_1111_1111_1111_1011_0111_1100_1111 (1's complement)
0xFFFFB7D0     0x1111_1111_1111_1111_1011_0111_1101_0000 (2's complement)
                    F    F    F    F    B    7    D    0
```

https://stackoverflow.com/questions/47333458/assembly-x86-64-setting-carry-flag-fo

Summary of signed and unsigned subtractions (1)

- subtracting 0x0000618D from 0x0000195D
  - 0x0000195D - 0x0000618D : unsigned integer subtraction
    hand subtraction
  - 0x0000195D + (-0x0000618D) : signed integer subtraction
    the *transformed addition* using the 2's complement of the subtrahend
  - the same result : 0xFFFFB7D0 (the same bit pattern)
    - interpreting as a unsigned integer 4294948816_{10}
      0xFFFFB7D0 with a borrow (CF=1)
    - interpreting as a signed integer -18480_{10}
      -0x00004830 (meaningless CF=1)

https://stackoverflow.com/questions/47333458/assembly-x86-64-setting-carry-flag-fo

Summary of signed and unsigned subtractions (2)

0xFFFFB7D0 with CF=1	the result of unsigned subtraction with unsigned integer overflow	4294948816_{10}
-0x00004830	the result of signed subtraction	-18480_{10}

https://stackoverflow.com/questions/47333458/assembly-x86-64-setting-carry-flag-fo

Examples of unsigned integer overflows

- `0x0000195D - 0x0000618D` : unsigned subtraction
  - there is an unsigned integer overflow
    so the carry flag will be set (`CF=1`) to indicate a borrow
  - $A - B$, when $A < B$, for non-negative integers A, B
    (unsigned integers can't be negative),

https://stackoverflow.com/questions/47333458/assembly-x86-64-setting-carry-flag-fo

Examples of signed integer overflows

- `0x0000195D + (-0x0000618D)` : signed subtraction
  - there is no signed integer overflow
    the overflow flag won't be set (`OF=0`)
  - signed overflw occurrs , in the transformed addition,
    - two *positive* numbers are added and
      the result is a *negative*, (P + P → N), or
    - two *negative* numbers are added and
      the result is a *positive*, (N + N → P)

https://stackoverflow.com/questions/47333458/assembly-x86-64-setting-carry-flag-fo

TOC Carry flag in unsigned and signed computations

- 2's complement numbers : 4-bit
- Addend and augend in a *n*-bit addition
- Full adder operation in each bit position
- Internal and external carry bits
- Addition and Subtraction
- Using the Carry Flag as a borrow

2's complement numbers : 4-bit

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

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Addend and augend in a n-bit addition

n	bits	addened	A	$\{a_{n-1}, a_{n-2}, \cdots, a_1, a_0\}$
n	bits	augend	B	$\{b_{n-1}, b_{n-2}, \cdots, b_1, b_0\}$
(n+1)	bits	carry bits	C	$\{C_n, C_{n-1}, C_{n-2}, \cdots, C_1, C_0\}$
n	bits	sum bits	S	$\{S_{n-1}, S_{n-2}, \cdots, S_1, S_0\}$

external carry bits : C_n carry out, C_0 carry in

$$
\begin{array}{cccccc}
 & a_{n-1} & a_{n-2} & \cdots\cdots\cdots & a_1 & a_0 \\
 & b_{n-1} & b_{n-2} & \cdots\cdots\cdots & b_1 & b_0 \\
 & & & & & C_0 \\
\hline
C_n & S_{n-1} & S_{n-2} & \cdots\cdots\cdots & S_1 & S_0
\end{array}
$$

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Full adder operation in each bit position

full adder operation in the i^{th} bit position

$$\{C_{i+1}, S_i\} = a_i + b_i + C_i$$

$$\begin{array}{rr} & a_i \\ & b_i \\ & C_i \\ \hline C_{i+1} & S_i \end{array}$$

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Internal and external carry bits

external carrys	C_n output, C_0 input	
internal carrys	$\{C_{n-1}, C_{n-2}, \cdots\cdots\cdots, C_2, C_1\}$	output / input
sum bits	$\{S_{n-1}, S_{n-2}, \cdots\cdots\cdots, S_1, S_0\}$	output

$$
\begin{array}{cccccc}
 & a_{n-1} & a_{n-2} & \cdots\cdots\cdots & a_1 & a_0 \\
 & b_{n-1} & b_{n-2} & \cdots\cdots\cdots & b_1 & b_0 \\
\hline
C_n & C_{n-1} & C_{n-2} & \cdots\cdots\cdots & C_1 & C_0 \\
\hline
 & S_{n-1} & S_{n-2} & \cdots\cdots\cdots & S_1 & S_0
\end{array}
$$

$$
\begin{array}{cccccc}
 & a_{n-1} & a_{n-2} & \cdots\cdots\cdots & a_1 & a_0 \\
 & b_{n-1} & b_{n-2} & \cdots\cdots\cdots & b_1 & b_0 \\
 & & & & & C_0 \\
\hline
C_n & S_{n-1} & S_{n-2} & \cdots\cdots\cdots & S_1 & S_0
\end{array}
$$

`http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt`

Addition and Subtraction

- addition

$$\{C_n, S\} = A + B = A + B + 0$$

$$\begin{array}{cccccc}
 & a_{n-1} & a_{n-2} & \cdots\cdots\cdots & a_1 & a_0 \\
 & b_{n-1} & b_{n-2} & \cdots\cdots\cdots & b_1 & b_0 \\
\hline
 & C_{n-1} & C_{n-2} & \cdots\cdots\cdots & C_1 & 0 \\
\hline
C_n & S_{n-1} & S_{n-2} & \cdots\cdots\cdots & S_1 & S_0
\end{array}$$

- subtraction - transformed addition

$$\{C_n, S\} = A - B = A + \overline{B} + 1$$

$$\begin{array}{cccccc}
 & a_{n-1} & a_{n-2} & \cdots\cdots\cdots & a_1 & a_0 \\
 & \overline{b_{n-1}} & \overline{b_{n-2}} & \cdots\cdots\cdots & \overline{b_1} & \overline{b_0} \\
\hline
 & C_{n-1} & C_{n-2} & \cdots\cdots\cdots & C_1 & 1 \\
\hline
C_n & S_{n-1} & S_{n-2} & \cdots\cdots\cdots & S_1 & S_0
\end{array}$$

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Using the Carry Flag as a borrow (1)

- a borrow (CF=1) occurs
  in the subtraction $A - B$
  when b is larger than a ($A < B$)
  as unsigned numbers
- Computer hardware can detect
  a borrow (CF=1) in subtraction
  by looking at whether a carry out (Cn) occurred
  in the transformed addition

https://stackoverflow.com/questions/47333458/assembly-x86-64-setting-carry-flag-f

Using the Carry flag as a borrow (2)

- a borrow (CF=1) occurs
  in the subtraction $A - B$ $(A < B)$
  as unsigned numbers
- a carry out (Cn) in the transformed addition
    - If there is no carry (Cn=0)
      then there is a borrow (CF=1)
    - If there is a carry (Cn=1)
      then there is no borrow (CF=0)
    - CF = !Cn

https://stackoverflow.com/questions/47333458/assembly-x86-64-setting-carry-flag-fo

Using the Carry Flag as a borrow (3)

- the same *addition* and *subtraction* instructions are used for both unsigned and signed integer arithmetic.
    - no special *addition* and *subtraction* instructions for unsigned and signed integer arithmetic
- the only difference is
    - which flags you *test* afterwards and
    - how you *interpret* the result

https://stackoverflow.com/questions/47333458/assembly-x86-64-setting-carry-flag-f

TOC Rules for the carry flag

- 2's complement numbers : 4-bit
- The 1st rule for setting the carry flag
- The 2nd rule for setting the carry flag
- Cases for clearing the carry flag
- Computing `CF` in unsigned additions and subtractions

2's complement numbers : 4-bit

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

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

The 1st rule for setting the carry flag

1. CF = 1 : carry in unsigned addition
   - the carry flag is set
     if the addition of two unsigned numbers causes
     a carry out of the most significant bits added.
   - unsigned integer overflow in unsigned addition
   - *hand addition rule*

`http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt`

The 2nd rule for setting the carry flag

2. CF = 1 : borrow in unsigned subtraction
   - the carry flag is also set
     if the subtraction of two unsigned numbers requires
     a borrow into the most significant bits subtracted.
   - unsigned integer overflow in unsigned subtraction
   - *hand subtraction rule*

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Cases for clearing the carry flag (1)

- Otherwise, the carry flag is turned off (zero).
  - all three interpretations have the same CF=1, the same S=0000

```
-----------------------------------------------------------------------
unsigned addition      |    signed addition          signed subtraction
-----------------------------------------------------------------------
 0111    ( 7)          |    0111    (+7)              0111    (+7)
+1001   +( 9)          |   +1001   +(-7)             -0111   -(+7)
-----   -----          |   -----   -----             -----   -----
10000    (16)          |   10000    ( 0)             10000    ( 0)
                       |
CF=1                   |   Cn=1 -> CF=1              Cn=1 -> CF=1
                       |
CF means 16            |   CF meaningless            CF meaningless
 S = 0000              |    S = 0000                  S = 0000
 -----------------------------------------------------------------------
* think hand           |   * think Cn of the corresponding addition
  addition             |     CF <- Cn
-----------------------------------------------------------------------
```

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Cases for clearing the carry flag (2)

- Otherwise, the carry flag is turned off (zero).
  - all three interpretations have the same CF=0, the same S=1111

```
-----------------------------------------------------------------------
unsigned addition      |    signed addition          signed subtraction
-----------------------------------------------------------------------
 0111    ( 7)          |    0111    (+7)              0111    (+7)
+1001   +( 9)          |   +1001   +(-7)             -0111   -(+7)
-----   -----          |   -----   -----             -----   -----
10000    (16)          |   10000    ( 0)             10000    ( 0)
                       |
CF=1                   |   Cn=1 -> CF=1              Cn=1 -> CF=1
                       |
CF means 16            |   CF meaningless            CF meaningless
 S = 0000              |    S = 0000                  S = 0000
-----------------------------------------------------------------------
* think hand           |   * think Cn of the corresponding addition
  addition             |     CF <- Cn
-----------------------------------------------------------------------
```

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Computing CF in unsigned additions and subtractions

- Computing CF in an unsigned addition
  - do the signed addition
  - Cn is the carry out
  - CF ← Cn
- Computing CF in an unsigned subraction
  - do the transformed signed addition
  - do the signed addition
  - Cn is the carry out
  - CF ← !Cn

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

TOC: Method for computing the carry flag

- Carry flag computation

Carry flag computation (1)

ADD (addition)	SUB (subtraction)
CF = C_n	CF = $\overline{C_n}$
normal carry of a 2's complement addition $A + B = A + B + 0$	inverted carry of a transformed addition $A - B = A + \overline{B} + 1$
$\{C_n, S_{n-1}\}$ $= a_{n-1} + b_{n-1} + c_{n-1}$	$\{C_n, S_{n-1}\}$ $= a_{n-1} + \overline{b_{n-1}} + c_{n-1}$

https://www.csie.ntu.edu.tw/~cyy/courses/assembly/12fall/lectures/handouts/lec14_x

Carry flag computation (2)

- In unsigned arithmetic,
  - the carry flag is used to detect *overflow*
  - the carry flag is used to extend *n-bit* result into *(n+1)-bit* result
  - for addition, the carry flag is a carry out
  - for subtraction, the carry flag is a borrow in
- In signed arithmetic,
  - the carry flag is useless
  - the carry flag neither detects overflow nor extends n-bit result

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Carry flag computation (3)

- In unsigned arithmetic,

Addition	CF = 1 means carry out	when Cn = 1
Subtraction	CF = 1 means borrow in	when Cn = 0

  - CF - Carry Flag in x86
  - Cn - the normal carry out
    - the carry out of a 2's complement addition for ADD
    - the carry out of a *transformed* addition for SUB

- In signed arithmetic,
  - the carry flag is useless

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

TOC: More examples of the carry flag

Summary I

```
------------------------------------------------------------------------
 unsigned add/sub     | signed addition      signed subtraction | CF OF
------------------------------------------------------------------------
  1101   (13)         |  1101    (-3)          1101    (-3)       |
 +1110  +(14)  ADD    | +1110   +(-2)  ADD    -0010   -(+2)       |
 -----  -----         | -----   -----         -----   -----       |
 11011   (11) (+16)   | 11011    (-5)         11011    (-5)       | 1  0
                      |                                           |
  0011   ( 3)         |  0011    (+3)          0011    (+3)       |
 -1110  -(14)  SUB    | +0010   +(+2)         -1110   -(-2)  SUB  |
 -----  -----         | -----   -----         -----   -----       |
 10101   ( 5) (-16)   | 00101    (+5)         00101    (+5)       | 1  0
                      |                                           |
  0011   ( 3)         |  0011    (+3)          0011    (+3)       |
 +0010  +( 2)  ADD    | +0010   +(+2)  ADD    -1110   -(-2)       |
 -----  -----         | -----   -----         -----   -----       |
 00101   ( 5) (+ 0)   | 00101    (+5)         00101    (+5)       | 0  0
                      |                                           |
  1101   (13)         |  1101    (-3)          1101    (-3)       |
 -0010  -( 2)  SUB    | +1110   +(-2)         -0010   -(+2)  SUB  |
 -----  -----         | -----   -----         -----   -----       |
 11011   (11) (-16)   | 11011    (-5)         11011    (-5)       | 0  0
------------------------------------------------------------------------
```

Summary II

```
-------------------------------------------------------------------------
  unsigned add/sub      | signed addition      signed subtraction | CF  OF
-------------------------------------------------------------------------
   1011    (11)         |   1011    (-5)          1011    (-5)      |
  +1100   +(12)   ADD   |  +1100   +(-4)   ADD   -0100   -(+4)      |
  -----   -----         |  -----   -----         -----   -----      |
  10111    ( 7) (+16)   |  10111    (+7)         10111    (+7)      | 1   1
                        |                                           |
   0101    ( 5)         |   0101    (+5)          0101    (+5)      |
  -1100   -(12)   SUB   |  +0100   +(+4)         -1100   -(-4)   SUB |
  -----   -----         |  -----   -----         -----   -----      |
  11001    ( 9) (-16)   |  01001    (-7)         01001    (-7)      | 1   1
                        |                                           |
   0101    ( 5)         |   0101    (+5)          0101    (+5)      |
  +0100   +( 4)   ADD   |  +0100   +(+4)   ADD   -1100   -(-4)      |
  -----   -----         |  -----   -----         -----   -----      |
  01001    ( 9) (+ 0)   |  01001    (-7)         01001    (-7)      | 0   1
                        |                                           |
   1011    (11)         |   1011    (-5)          1011    (-5)      |
  -0100   -( 4)   SUB   |  +1100   +(-4)         -0100   -(+4)   SUB |
  -----   -----         |  -----   -----         -----   -----      |
  00111    ( 7) (  0)   |  10111    (+7)         10111    (+7)      | 0   1
-------------------------------------------------------------------------
```

Cases for setting the carry flag (1) CF=1, OF=0

- unsigned integer overflow (CF=1 means +16)

```
------------------------------------------------------------------------
* unsigned addition     |  * signed addition          signed subtraction
------------------------------------------------------------------------
   1101    (13)         |     1101    (-3)              1101    (-3)
  +1110   +(14)   ADD   |    +1110   +(-2)   ADD       -0010   -(+2)
  -----   -----         |    -----   -----             -----   -----
  11011    (11)  (+16)  |    11011    (-5)             11011    (-5)
                        |
  CF=1                  |    Cn=1 -> CF=1              Cn=1 -> CF=1
                        |
  CF means 16           |    CF meaningless            CF meaningless
   S = 0000             |      S = 0000                  S = 0000
------------------------------------------------------------------------
* think hand            |  * think Cn of the corresponding addition
  addition              |    CF <- Cn  (for unsigned addition)
------------------------------------------------------------------------

* CF=1, S=1011, OF=0 for all three interpretations
```

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Cases for setting the carry flag (2) CF=1, OF=0

- unsigned integer overflow (CF=1 means −16)

```
-----------------------------------------------------------------------
* unsigned subtraction |    signed addition         * signed subtraction
-----------------------------------------------------------------------
   0011    ( 3)        |    0011    (+3)             0011    (+3)
  -1110   -(14)   SUB  |   +0010   +(+2)            -1110   -(-2)   SUB
  -----   -----        |   -----   -----            -----   -----
  10101    ( 5)  (-16) |   00101    (+5)            00101    (+5)
                       |
  CF=1                 |   Cn=0 -> CF=1             Cn=0 -> CF=1
                       |
  CF means -16         |   CF meaningless           CF meaningless
   S = 0101            |     S = 0101                 S = 0101
-----------------------------------------------------------------------
* think hand           |  * think Cn of the transformed addition
  subtraction          |    CF <- !Cn (for unsigned subtraction)
-----------------------------------------------------------------------

* CF=1, S=0101, OF=0 for all three interpretations
```

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Cases for setting the carry flag (3) CF=1, OF=1

- unsinged integer overflow (CF=1 means +16)

```
-----------------------------------------------------------------------
* unsigned addition     |  * signed addition         signed subtraction
-----------------------------------------------------------------------
   1011   (11)          |     1011   (-5)              1011   (-5)
  +1100  +(12)   ADD    |    +1100  +(-4)   ADD       -0100  -(+4)
  -----  -----          |    -----  -----             -----  -----
  10111   ( 7)  (+16)   |    10111   (+7)             10111   (+7)
                        |
  CF=1                  |    Cn=1 -> CF=1             Cn=1 -> CF=1
                        |
  CF means +16          |    CF meaningless           CF meaningless
   S = 0111             |     S = 0111                 S = 0111
-----------------------------------------------------------------------
 * think hand           |  * think Cn of the corresponding addition
   addition             |    CF <- Cn  (for unsigned addition)
-----------------------------------------------------------------------

* CF=1, S=0111, OF=1 for all three interpretations
```

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Cases for setting the carry flag (4) CF=1, OF=1

- unsinged integer overflow (CF=1 means -16)

```
------------------------------------------------------------------------
* unsigned subtraction |    signed addition        * signed subtraction
------------------------------------------------------------------------
   0101    ( 5)        |    0101    (+5)             0101    (+5)
  -1100   -(12)   SUB  |   +0100   +(+4)            -1100   -(-4)   SUB
  -----   -----        |   -----   -----            -----   -----
  11001    ( 9)  (-16) |   01001    (-7)            01001    (-7)
                       |
  CF=1                 |   Cn=0 -> CF=1             Cn=0 -> CF=1
                       |
  CF means -16         |   CF meaningless           CF meaningless
   S = 1001            |    S = 1001                 S = 1001
------------------------------------------------------------------------
* think hand           |  * think Cn of the transformed addition
  subtraction          |    CF <- !Cn (for unsigned subtraction)
------------------------------------------------------------------------

* CF=1, S=1001, OF=1 for all three interpretations
```

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Cases for clearing the carry flag (1) CF=0, OF=0

- no unsigned integer overflow (CF=0)

```
------------------------------------------------------------------------
* unsigned addition    |  * signed addition          signed subtraction
------------------------------------------------------------------------
   0011    ( 3)        |     0011    (+3)              0011    (+3)
  +0010   +( 2)   ADD  |    +0010   +(+2)   ADD       -1110   -(-2)
  -----   -----        |    -----   -----             -----   -----
  00101    ( 5)  (+ 0) |    00101    (+5)             00101    (+5)
                       |
  CF=0                 |    Cn=0 -> CF=0              Cn=0 -> CF=0
                       |
  CF means 0           |    CF meaningless            CF meaningless
   S = 0101            |     S = 0101                  S = 0101
------------------------------------------------------------------------
* think hand           |  * think Cn of the corresponding addition
  addition             |    CF <- Cn  (for unsigned addition)
------------------------------------------------------------------------

* CF=0, S=0101, OF=0 for all three interpretations
```

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Cases for clearing the carry flag (2) CF=0, OF=0

- no unsigned integer overflow (CF=0)

```
------------------------------------------------------------------------
* unsigned addition     |  * signed addition          signed subtraction
------------------------------------------------------------------------
   1101    (13)         |     1101    (-3)             1101    (-3)
  -0010   -( 2)   SUB   |    +1110   +(-2)            -0010   -(+2)   SUB
  -----   -----         |    -----   -----            -----   -----
  11011    (11)  (-16)  |    11011    (-5)            11011    (-5)
                        |
  CF=0                  |    Cn=0 -> CF=0             Cn=0 -> CF=0
                        |
  CF means 0            |    CF meaningless           CF meaningless
   S = 1011             |     S = 1011                 S = 1011
------------------------------------------------------------------------
* think hand            |  * think Cn of the corresponding addition
  subtraction           |    CF <- Cn  (for unsigned addition)
------------------------------------------------------------------------

* CF=0, S=1011, OF=0 for all three interpretations
```

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Cases for clearing the carry flag (3) CF=0, OF=1

- no unsinged integer overflow (CF=0)

```
-----------------------------------------------------------------------
* unsigned addition     |  * signed addition          signed subtraction
-----------------------------------------------------------------------
   0101    ( 5)         |     0101    (+5)              0101    (+5)
  +0100   +( 4)   ADD   |    +0100   +(+4)   ADD       -1100   -(-4)
  -----   -----         |    -----   -----             -----   -----
  01001    ( 9)  (+ 0)  |    01001    (-7)             01001    (-7)
                        |
  CF=0                  |    Cn=0 -> CF=0              Cn=0 -> CF=0
                        |
  CF means +0           |    CF meaningless            CF meaningless
   S = 1001             |     S = 1001                  S = 1001
-----------------------------------------------------------------------
 * think hand           |  * think Cn of the corresponding addition
   addition             |    CF <- Cn  (for unsigned addition)
-----------------------------------------------------------------------

* CF=0, S=1001, OF=1 for all three interpretations
```

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Cases for clearing the carry flag (4) CF=0 , OF=1

- no unsinged integer overflow (CF=0)

```
------------------------------------------------------------------------
* unsigned subtraction |    signed addition        * signed subtraction
------------------------------------------------------------------------
   1011    (11)        |    1011    (-5)              1011    (-5)
  -0100   -( 4)   SUB  |   +1100   +(-4)             -0100   -(+4)   SUB
  -----   -----        |   -----   -----             -----   -----
  00111    ( 7) (   0) |   10111    (+7)             10111    (+7)
                       |
  CF=0                 |   Cn=1 -> CF=0              Cn=1 -> CF=0
                       |
  CF means  0          |   CF meaningless            CF meaningless
   S = 0111            |    S = 0111                  S = 0111
------------------------------------------------------------------------
* think hand           | * think Cn of the transformed addition
  subtraction          |   CF <- !Cn (for unsigned subtraction)
------------------------------------------------------------------------

* CF=0, S=0111, OF=1 for all three interpretations
```

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

TOC: Overflow flag

TOC Overflow flag in unsigned and signed computations

- Overflow flag

Overflow flag (1)

- only need to look at
  the sign bits (leftmost) of the three numbers

augend	+	addend	=	sum
minuend	−	subrahend	=	difference

  to decide if the overflow flag is turned on or off.

- overflow flag is based on signed arithmetic

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Overflow flag (2)

- in signed arithmetic,
  - watch the overflow flag to detect errors
  - overflow flag on means the result is wrong
  - errors can be detected by examining the sign of the result, in the 2's complement arithmetic
- in unsigned arithmetic,
  - the overflow flag tells you nothing interesting

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Overflow flag (3)

- when two positive numbers are added
  - if the result is a negative, (P + P $\rightarrow$ N), then overflow
  - if the result is a positive, (P + P $\rightarrow$ P), then no overflow
- when two negative numbers are added
  - the result is a positive, (N + N $\rightarrow$ P), then overflow
  - the result is a negative, (N + N $\rightarrow$ N), then no overflow

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Overflow flag (4)

- adding negative and positive numbers cannot be wrong, because the sum is between the addends.
  - mixed-sign addition never turns on the overflow flag.
  - opposite signed numbers are added, then no overflow
  - both of the addends lies in the allowable range of numbers, and their sum is between the addends, therefore the sum lies also in the allowable range
  - $(P + N \rightarrow P)$ no overflow
  - $(P + N \rightarrow N)$ no overflow
  - $(N + P \rightarrow P)$ no overflow
  - $(N + P \rightarrow N)$ no overflow

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

TOC Rules for the overflow flag

- the 1st rule for setting OF
- the 2nd rule for setting OF
- cases for clearing OF ($1 \sim 6$)

Overflow flag setting and clearing conditions

	ADD conditions	SUB conditions	
OF=1	$P + P \to N$	$P - N \to N$	$C_n \bigoplus C_{n-1} = 1$
OF=1	$N + N \to P$	$N - P \to P$	$C_n \bigoplus C_{n-1} = 1$
OF=0	$P + P \to P$	$P - N \to P$	$C_n \bigoplus C_{n-1} = 0$
OF=0	$N + N \to N$	$N - P \to N$	$C_n \bigoplus C_{n-1} = 0$
OF=0	$P + N \to P$	$P - N \to P$	$C_n \bigoplus C_{n-1} = 0$
OF=0	$P + N \to N$	$P - P \to N$	$C_n \bigoplus C_{n-1} = 0$
OF=0	$N + P \to P$	$N - N \to P$	$C_n \bigoplus C_{n-1} = 0$
OF=0	$N + P \to N$	$N - P \to N$	$C_n \bigoplus C_{n-1} = 0$

$$+P = -(-P) \quad = -N$$
$$+N = -(-N) \quad = -P$$

The 1st rule for setting the overflow flag

1. If the sum of two signed numbers
   with the sign bits off (0, 0)
   yields a result number with the sign bit on (1)
   the overflow flag is turned on ($\texttt{OF} = 1$: $P + P \rightarrow N$)

```
signed addition          signed subtraction          unsigned addition

0100   carries
 0100   (+4)               0100   (+4)                 0100   ( 4)
+0100  +(+4)              -1100  -(-4)                +0100  +( 4)
-----  -----              -----  -----                -----  -----
01000   (-8)              01000   (-8)                01000   ( 8)
```

- $\texttt{OF} = C_n \bigoplus C_{n-1} = C_4 \bigoplus C_3 = 0 \bigoplus 1 = 1$

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

The 2nd rule for setting the overflow flag

2. If the sum of two numbers
   with the sign bits on (1, 1)
   yields a result number with the sign bit off (0)
   the overflow flag is turned on. (OF =1 : $N + N \rightarrow P$)

```
signed addition          signed subtraction          unsigned addition

1001   carries
 1001   (-7)              1001   (-7)                 1001   ( 9)
+1001  +(-7)             -0111  -(+7)                +1001  +( 9)
-----  -----             -----  -----                -----  -----
10010   ( 2)             10010   ( 2)                10010   (18)
```

- OF $= C_n \bigoplus C_{n-1} = C_4 \bigoplus C_3 = 1 \bigoplus 0 = 1$

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Cases for clearing the overflow flag (1)

- overflow flag is turned off. ($\texttt{OF} = 0 : P + P \rightarrow P$)

```
signed addition         signed subtraction          unsigned addition

0011   carries
 0011   (+3)              0011   (+3)                 0011   ( 3)
+0011  +(+3)             -1101  -(-3)                +0011  +( 3)
-----  -----             -----  -----                -----  -----
00110   (+6)             00110   (+6)                00110   ( 6)
```

- $\texttt{OF} = C_n \bigoplus C_{n-1} = C_4 \bigoplus C_3 = 0 \bigoplus 0 = 0$

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Cases for clearing the overflow flag (2)

- overflow flag is turned off. (OF = 0 : $N + N \rightarrow N$)

```
signed addition           signed subtraction         unsigned addition

1101   carries
 1101   (-3)                1101   (-3)                1101   (13)
+1101  +(-3)               -0011  -(+3)               +1101  +(13)
-----  -----               -----  -----               -----  -----
11010   (-6)               11010   (-6)               11010   (26)
```

- $\mathtt{OF} = C_n \bigoplus C_{n-1} = C_4 \bigoplus C_3 = 1 \bigoplus 1 = 0$

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Cases for clearing the overflow flag (3)

- overflow flag is turned off. ($\texttt{OF} = 0 : P + N \rightarrow P$)

```
signed addition          signed subtraction          unsigned addition

1100    carries
 0100    (+4)             0100    (+4)                0100    ( 4)
+1101   +(-3)            -0011   -(+3)               +1101   +(13)
-----   -----            -----   -----               -----   -----
10001    (+1)            10001    (+1)               10001    (17)
```

- $\texttt{OF} = C_n \bigoplus C_{n-1} = C_4 \bigoplus C_3 = 1 \bigoplus 1 = 0$

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Cases for clearing the overflow flag (4)

- overflow flag is turned off. (OF = 0 : $P + N \rightarrow N$)

```
signed addition          signed subtraction          unsigned addition

0000   carries
 0011   (+3)              0011   (+3)                 0011   ( 3)
+1100  +(-4)             -0100  -(+4)                +1100  +(12)
-----  -----             -----  -----                -----  -----
01111   (-1)             01111   (-1)                01111   (15)
```

- OF $= C_n \bigoplus C_{n-1} = C_4 \bigoplus C_3 = 0 \bigoplus 0 = 0$

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Cases for clearing the overflow flag (5)

- overflow flag is turned off. (`OF` = 0 : $N + P \rightarrow P$)

```
signed addition          signed subtraction          unsigned addition

1100   carries
 1101   (-3)                 0011    (-3)                1101    (13)
+0100   (+4)                -1100  -(-4)                +0100   +( 4)
-----  -----                -----  -----                -----   -----
10001   (+1)                10001    (+1)               10001    (17)
```

- $\mathtt{OF} = C_n \bigoplus C_{n-1} = C_4 \bigoplus C_3 = 1 \bigoplus 1 = 0$

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Cases for clearing the overflow flag (6)

- overflow flag is turned off. (OF = 0 : $N + P \rightarrow N$)

```
signed addition          signed subtraction          unsigned addition

0000   carries
 1100   (-4)              0100   (-4)                  1100   (12)
+0011  +(+3)             -1101  -(-3)                 +0011  +( 3)
-----  -----             -----  -----                 -----  -----
01111   (-1)             01111   (-1)                 01111   (15)
```

  - $\text{OF} = C_n \bigoplus C_{n-1} = C_4 \bigoplus C_3 = 0 \bigoplus 0 = 0$

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

TOC Method 1 for computing the overflow flag

Adding two numbers with the same sign

- overflow can only happen
  when adding two numbers of the same sign
  results in a different sign ($P + P \to N$, $N + N \to P$)
- n-bit signed binary arithmetic $A + B = C$

$$A = (a_{n-1}, \cdots, a_1, a_0)$$
$$B = (b_{n-1}, \cdots, b_1, b_0)$$
$$C = (c_{n-1}, \cdots, c_1, c_0)$$

- to detect overflow
  - only the sign bits are considered
  - msb (most significant bit) $a_{n-1}, b_{n-1}, c_{n-1}$
  - the other bits are ignored

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Overflow conditions for additions and subtractions

- with two operands (A and B) and one result (C), three sign bits ($a_{n-1}, b_{n-1}, c_{n-1}$) are considered $\rightarrow 2^3 = 8$ possible combinations
- only two cases result in overflow for an addition
  - 0 0 1 $(p + p \rightarrow n)$
  - 1 1 0 $(n + n \rightarrow p)$
- only two cases are considered as overflow for an subtraction
  - 0 1 1 $(p - n \rightarrow n)$
  - 1 0 0 $(n - p \rightarrow p)$

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Overflow condition for an addition

- Overflow in an addition $(A + B)$

	a_{n-1}	b_{n-1}	c_{n-1}	
	0	0	0	$p + p \rightarrow p$
OVER	0	0	1	$p + p \rightarrow n$
	0	1	0	$p + n \rightarrow p$
	0	1	1	$p + n \rightarrow n$
	1	0	0	$n + p \rightarrow p$
	1	0	1	$n + p \rightarrow n$
OVER	1	1	0	$n + n \rightarrow p$
	1	1	1	$n + n \rightarrow n$

  - adding two positives should be positive
  - adding two negatives should be negative

`http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt`

Overflow conditions for a subtraction

- Overflow in a subtraction $(A - B)$

	a_{n-1}	b_{n-1}	c_{n-1}	
	0	0	0	$p - p \to p$
	0	0	1	$p - p \to n$
	0	1	0	$p - n \to p$
OVER	0	1	1	$p - n \to n$
OVER	1	0	0	$n - p \to p$
	1	0	1	$n - p \to n$
	1	1	0	$n - n \to p$
	1	1	1	$n - n \to n$

  - subtracting a *negative* is the same as adding a *positive*
  - subtracting a *positive* is the same as adding a *negative*

`http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt`

Overflow in signed computations

- ALU might contain a small logic
  that sets the overflow flag to "1"
  if and only if any one of the above four OV conditions is met.

- in signed computations,
  adding two numbers of the same sign
  must produce a result of the same sign,
  otherwise overflow happened.

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

TOC Method 2 for computing the overflow flag

- Carry into and carry out of the sign bit
- Overflow in 2's complement arithmetic
- Overflow flag $= C_n \bigoplus C_{n-1}$
- Examples of 4-bit signed additions
- C_n and C_{n-1} in a n-bit addition
- Overflow flag computation
- Examples of computing overflow flag
- Hexadecimal carry, octal carry, decimal carry
- No carry into the sign bit

Carry into and carry out of the sign bit

- When adding two n-bit binary values, consider
  - the carry *coming into* the most significant bit (msb)
    C_{n-1} : carry into the sign bit
  - the carry *going out of* the most_significant bit (msb)
    C_n : carry out of the sign bit
    this is the carry flag (CF) in the processor

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Overflow in 2's complement arithmetic

- overflow in 2's complement happens (`OF=1`) when
  - there is a carry *into* the sign bit ($C_{n-1} = 1$)
    but no carry *out of* the sign bit ($C_n = 0$)
  - there is no carry *into* the sign bit ($C_{n-1} = 0$)
    but a carry *out of* the sign bit ($C_n = 1$)

`http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt`

Overflow flag $= C_n \bigoplus C_{n-1}$

- the overflow flag is the XOR $(C_n \bigoplus C_{n-1})$ of
  - of the carry *coming into* the sign bit (C_{n-1})
  - with the carry *going out of* the sign bit (C_n)
- overflow happens when
  the carry in (C_{n-1}) does not equal
  to the carry out (C_n)

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Examples of 4-bit signed additions (1)

- 4-bit 2's complement addition examples

```
0000                              0100
 0100 (+4) (pos sign 0)            0100 (+4) (pos sign 0)
+1000 (-8) (neg sign 1)           +0100 (+4) (pos sign 0)
=======================           ========================
01100 (-4) (neg sign 1)           01000 (-8) (neg sign 1)

C4 carry out 0 (1+0+0)            C4 carry out 0 (0+0+1)
C3 carry in  0 (0+1+0)            C3 carry in  1 (1+1+0)
0 XOR 0 = NO OVERFLOW             0 XOR 1 = OVERFLOW!

1100                              1000
 1100  (-4) (neg sign 1)           1100 (-4) (neg sign 1)
+0100  (+4) (pos sign 0)          +1000 (-8) (neg sign 1)
========================          =======================
10000  ( 0) (pos sign 0)          10100 (+4) (pos sign 0)

C4 carry out 1 (1+0+1)            C4 carry out 1 (1+1+0)
C3 carry in  1 (1+1+0)            C3 carry in  0 (1+0+0)
1 XOR 1 = NO OVERFLOW             1 XOR 0 = OVERFLOW!
```

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Examples of 4-bit signed additions (2)

- same sign addition → possible overflow

```
------    ------    ------    ------
+ +, -    - -, +    + +, +    - -, -
------    ------    ------    ------
    +5        -5        +5        -5
    +5        -5        +1        -1
 -----    ------    ------     -----
    -6(OF)    +6(OF)    +6        -6

 0101      1011      0001      1111
  0101      1011      0101      1011
  0101      1011      0001      1111
------    ------    ------    ------
 01010     10110     00110     11010
------    ------    ------    ------
C4 = 0    C4 = 1    C4 = 0    C4 = 1
C3 = 1    C3 = 0    C3 = 0    C3 = 1
------    ------    ------    ------
OF = 1    OF = 1    OF = 0    OF = 0
```

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Examples of 4-bit signed additions (3)

- mixed sign addition → no overflow

```
------    ------    ------    ------
+ -, +    + -, -    - +, +    - +, -
------    ------    ------    ------
    +5        +5        -5        -5
    -1        -6        +6        +1
 -----    ------    ------     -----
    +4        -1        +1        -4

 1111      0000      1110      0011
  0101      0101      1011      1011
  1111      1010      0110      0001
------    ------    ------    ------
 10100     01111     10001     01100
------    ------    ------    ------
C4 = 1    C4 = 0    C4 = 1    C4 = 0
C3 = 1    C3 = 0    C3 = 1    C3 = 0
------    ------    ------    ------
OF = 0    OF = 0    OF = 0    OF = 0
```

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

C_n and C_{n-1} in a n-bit addition

$(n-1)^{th}$ bit – MSB

- adding operations at the $(n-1)$ bit position
- $\{C_n, S_{n-1}\} = a_{n-1} + b_{n-1} + c_{n-1}$

	msb
	a_{n-1}
	b_{n-1}
	C_{n-1}
C_n	S_{n-1}

- C_n : carry coming *out of* the msb

$(n-2)^{th}$ bit

- adding operations at the $(n-2)$ bit position
- $\{C_{n-1}, S_{n-2}\} = a_{n-2} + b_{n-2} + c_{n-2}$

msb	
	a_{n-2}
	b_{n-2}
	C_{n-2}
C_{n-1}	S_{n-2}

- C_{n-1} : carry coming *into* the msb

Overflow flag computation

ADD (addition)	SUB (subtraction)
OF = $C_n \bigoplus C_{n-1}$	OF = $C_n \bigoplus C_{n-1}$
a 2's complement addition $A + B = A + B + 0 \; (C_0 = 0)$	the transformed addition $A - B = A + \overline{B} + 1 \; (C_0 = 1)$
$\{C_n, S_{n-1}\}$ $= a_{n-1} + b_{n-1} + c_{n-1}$	$\{C_n, S_{n-1}\}$ $= a_{n-1} + \overline{b_{n-1}} + c_{n-1}$
$\{C_{n-1}, S_{n-2}\}$ $= a_{n-2} + b_{n-2} + c_{n-2}$	$\{C_{n-1}, S_{n-2}\}$ $= a_{n-2} + \overline{b_{n-2}} + c_{n-2}$

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Hexadecimal carry, octal carry, decimal carry

- Note that this XOR method
  only works with the binary carry that goes into the sign bit.
- not works with hexadecimal carry
  decimal carry, octal carry
  - the carry doesn't go into the sign bit
  - can't XOR that non-binary carry with the outgoing carry.

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

No carry into the sign bit

- Hexadecimal addition example
  (showing that XOR doesn't work for hex carry):

```
 8Ah
+8Ah
====
114h
```

- The hexadecimal carry of 1 resulting from A+A
  does not affect the sign bit.
- If you do the math in binary, you'll see
  that there is no carry into the sign bit;
  but, there is carry out of the sign bit.
  Therefore, the above example sets OVERFLOW on.
  (The example adds two negative numbers and gets a positive number.)

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Summary I

```
------------------------------------------------------------------------
  unsigned add/sub      | signed addition      signed subtraction | CF OF
------------------------------------------------------------------------
   1101    (13)         |   1101    (-3)          1101    (-3)       |
  +1110   +(14)   ADD   |  +1110   +(-2)   ADD   -0010   -(+2)       |
  -----   -----         |  -----   -----         -----   -----       |
  11011    (11) (+16)   |  11011    (-5)         11011    (-5)       | 1  0
                        |                                            |
   0011    ( 3)         |   0011    (+3)          0011    (+3)       |
  -1110   -(14)   SUB   |  +0010   +(+2)         -1110   -(-2)   SUB |
  -----   -----         |  -----   -----         -----   -----       |
  10101    ( 5) (-16)   |  00101    (+5)         00101    (+5)       | 1  0
                        |                                            |
   0011    ( 3)         |   0011    (+3)          0011    (+3)       |
  +0010   +( 2)   ADD   |  +0010   +(+2)   ADD   -1110   -(-2)       |
  -----   -----         |  -----   -----         -----   -----       |
  00101    ( 5) (+ 0)   |  00101    (+5)         00101    (+5)       | 0  0
                        |                                            |
   1101    (13)         |   1101    (-3)          1101    (-3)       |
  -0010   -( 2)   SUB   |  +1110   +(-2)         -0010   -(+2)   SUB |
  -----   -----         |  -----   -----         -----   -----       |
  11011    (11) (-16)   |  11011    (-5)         11011    (-5)       | 0  0
------------------------------------------------------------------------
```

Summary II

```
-----------------------------------------------------------------------
  unsigned add/sub      | signed addition      signed subtraction | CF OF
-----------------------------------------------------------------------
   1011    (11)         |   1011    (-5)          1011    (-5)      |
  +1100  +(12)   ADD    |  +1100  +(-4)   ADD    -0100  -(+4)      |
  -----  -----          |  -----  -----          -----  -----      |
  10111    ( 7) (+16)   |  10111    (+7)          10111    (+7)      | 1   1
                        |                                           |
   0101    ( 5)         |   0101    (+5)          0101    (+5)      |
  -1100  -(12)   SUB    |  +0100  +(+4)          -1100  -(-4)   SUB |
  -----  -----          |  -----  -----          -----  -----      |
  11001    ( 9) (-16)   |  01001    (-7)          01001    (-7)      | 1   1
                        |                                           |
   0101    ( 5)         |   0101    (+5)          0101    (+5)      |
  +0100  +( 4)   ADD    |  +0100  +(+4)   ADD    -1100  -(-4)      |
  -----  -----          |  -----  -----          -----  -----      |
  01001    ( 9) (+ 0)   |  01001    (-7)          01001    (-7)      | 0   1
                        |                                           |
   1011    (11)         |   1011    (-5)          1011    (-5)      |
  -0100  -( 4)   SUB    |  +1100  +(-4)          -0100  -(+4)   SUB |
  -----  -----          |  -----  -----          -----  -----      |
  00111    ( 7) (   0)  |  10111    (+7)          10111    (+7)      | 0   1
----------------------------------------------------------------------
```

Cases for setting the overflow flag (1) CF=1, OF=1

- singed integer overflow (OF=1 means incorrect S)

```
------------------------------------------------------------------------
* unsigned addition     |  * signed addition          signed subtraction
------------------------------------------------------------------------
                        |    1000
   1011    (11)         |     1011    (-5)              1011    (-5)
  +1100   +(12)   ADD   |    +1100   +(-4)   ADD       -0100   -(+4)
  -----   -----         |    -----   -----             -----   -----
  10111    ( 7)  (+16)  |    10111    (+7)             10111    (+7)
                        |
  OF=1                  |    n + n -> p (OF=1)         n - p -> p (OF=1)
                        |
  OF meaningless        |    -> incorrect S            -> incorrect S
   S = 0111             |     S = 0111                  S = 0111
------------------------------------------------------------------------
 * think hand           |  * OF <- C4 XOR C3 = 1 XOR 0 = 1
   addition             |          of signed addition
------------------------------------------------------------------------

* CF=1, S=0111, OF=1 for all three interpretations
```

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Cases for setting the overflow flag (2) CF=1, OF=1

- singed integer overflow (OF=1 means incorrect S)

```
------------------------------------------------------------------------
* unsigned subtraction |    signed addition        * signed subtraction
------------------------------------------------------------------------
                       |    0100
   0101    ( 5)        |     0101    (+5)            0101    (+5)
  -1100   -(12)   SUB  |    +0100   +(+4)           -1100   -(-4)   SUB
  -----   -----        |    -----   -----           -----   -----
  11001    ( 9)  (-16) |    01001    (-7)           01001    (-7)
                       |
  OF=1                 |    p + p -> n (OF=1)       p - n -> n (OF=1)
                       |
  OF meaningless       |    -> incorrect S          -> incorrect S
   S = 1001            |     S = 1001                S = 1001
------------------------------------------------------------------------
* think hand           |  * OF <- C4 XOR C3 = 0 XOR 1 = 1
  subtraction          |          of signed addition
------------------------------------------------------------------------

* CF=1, S=1001, OF=1 for all three interpretations
```

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Cases for setting the overflow flag (3) CF=0, OF=1

- singed integer overflow (OF=1 means incorrect S)

```
------------------------------------------------------------------------
* unsigned addition       |  * signed addition          signed subtraction
------------------------------------------------------------------------
                          |    0100
   0101    ( 5)           |     0101    (+5)               0101    (+5)
  +0100   +( 4)   ADD     |    +0100   +(+4)   ADD        -1100   -(-4)
  -----   -----           |    -----   -----              -----   -----
  01001    ( 9)  (+ 0)    |    01001    (-7)              01001    (-7)
                          |
  OF=1                    |    p + p -> n (OF=1)          p - n -> n (OF=1)
                          |
  OF meaningless          |    -> incorrect S             -> incorrect S
   S = 1001               |     S = 1001                   S = 1001
------------------------------------------------------------------------
 * think hand             |  * OF <- C4 XOR C3 = 0 XOR 1 = 1
   addition               |          of signed addition
------------------------------------------------------------------------

* CF=0, S=1001, OF=1 for all three interpretations
```

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Cases for setting the overflow flag (4) CF=0, OF=1

- singed integer overflow (OF=1 means incorrect S)

```
----------------------------------------------------------------------
* unsigned subtraction |    signed addition        * signed subtraction
----------------------------------------------------------------------
                       |    1000
   1011    (11)        |     1011    (-5)             1011    (-5)
  -0100   -( 4)   SUB  |    +1100   +(-4)            -0100   -(+4)   SUB
  -----   -----        |    -----   -----            -----   -----
  00111    ( 7) (   0) |    10111    (+7)            10111    (+7)
                       |
  OF=1                 |    n + n -> p (OF=1)        n - p -> p (OF=1)
                       |
  OF meaningless       |    -> incorrect S           -> incorrect S
   S = 0111            |     S = 0111                 S = 0111
----------------------------------------------------------------------
* think hand           |  * OF <- C4 XOR C3 = 1 XOR 0 = 1
  subtraction          |          of signed addition
----------------------------------------------------------------------

* CF=0, S=0111, OF=1 for all three interpretations
```

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Cases for clearing the overflow flag (1) CF=1, OF=0

- no signed integer overflow (CF=0 means correct S)

```
------------------------------------------------------------------------
* unsigned addition      |  * signed addition          signed subtraction
------------------------------------------------------------------------
                         |    1100
   1101   (13)           |     1101   (-3)              1101   (-3)
  +1110  +(14)   ADD     |    +1110  +(-2)   ADD       -0010  -(+2)
  -----  -----           |    -----  -----             -----  -----
  11011   (11)  (+16)    |    11011   (-5)             11011   (-5)
                         |
  OF=0                   |    n + n -> n (OF=0)        n - p -> n (OF=0)
                         |
  OF meaningless         |    -> correct S             -> correct S
   S = 0000              |      S = 0000                 S = 0000
------------------------------------------------------------------------
* think hand             |  * OF <- C4 XOR C3 = 1 XOR 1 = 0
  addition               |          of signed addition
------------------------------------------------------------------------

* CF=1, S=1011, OF=0 for all three interpretations
```

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Cases for clearing the overflow flag (2) CF=1, OF=0

- no signed integer overflow (CF=0 means correct S)

```
-----------------------------------------------------------------------
* unsigned subtraction |    signed addition        * signed subtraction
-----------------------------------------------------------------------
                       |    0010
   0011    ( 3)        |     0011    (+3)              0011    (+3)
  -1110   -(14)   SUB  |    +0010   +(+2)             -1110   -(-2)   SUB
  -----   -----        |    -----   -----             -----   -----
  10101    ( 5)  (-16) |    00101    (+5)             00101    (+5)
                       |
  CF=1                 |    p + p -> p (OF=0)         p - n -> p (OF=0)
                       |
  OF meaningless       |    -> correct S              -> correct S
   S = 0101            |      S = 0101                  S = 0101
-----------------------------------------------------------------------
* think hand           |  * OF <- C4 XOR C3 = 0 XOR 0 = 0
  subtraction          |          of signed addition
-----------------------------------------------------------------------

* CF=1, S=0101, OF=0 for all three interpretations
```

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Cases for clearing the overflow flag (3) CF=0, OF=0

- no signed integer overflow (CF=0 means correct S)

```
------------------------------------------------------------------------
* unsigned addition      |  * signed addition          signed subtraction
------------------------------------------------------------------------
                         |    0010
   0011    ( 3)          |     0011    (+3)              0011    (+3)
  +0010   +( 2)   ADD    |    +0010   +(+2)   ADD       -1110   -(-2)
  -----   -----          |    -----   -----             -----   -----
  00101    ( 5)  (+ 0)   |    00101    (+5)             00101    (+5)
                         |
  OF=0                   |    p + p -> p (OF=0)         p - n -> p (OF=0)
                         |
  OF meaningless         |    -> correct S              -> correct S
   S = 0101              |     S = 0101                  S = 0101
------------------------------------------------------------------------
* think hand             |  * OF <- C4 XOR C3 = 0 XOR 0 = 0
  addition               |          of signed addition
------------------------------------------------------------------------

* CF=0, S=0101, OF=0 for all three interpretations
```

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Cases for clearing the overflow flag (4) CF=0, OF=0

- no signed integer overflow (CF=0 means correct S)

```
------------------------------------------------------------------------
* unsigned addition     |  * signed addition          signed subtraction
------------------------------------------------------------------------
                        |    1100
   1101    (13)         |     1101    (-3)             1101    (-3)
  -0010   -( 2)   SUB   |    +1110   +(-2)            -0010   -(+2)   SUB
  -----   -----         |    -----   -----            -----   -----
  11011    (11)  (-16)  |    11011    (-5)            11011    (-5)
                        |
  OF=0                  |    n + n -> n (OF=0)        n - p -> n (OF=0)
                        |
  OF meaningless        |    -> correct S             -> correct S
   S = 1011             |     S = 1011                 S = 1011
------------------------------------------------------------------------
* think hand            |  * OF <- C4 XOR C3 = 1 XOR 1 = 0
  subtraction           |          of signed addition
------------------------------------------------------------------------

* CF=0, S=1011, OF=0 for all three interpretations
```

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

2's complement numbers : 4-bit

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

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Unsigned 4-bit addition table (1)

	0000 (0)	0001 (1)	0010 (2)	0011 (3)	0100 (4)	0101 (5)	0110 (6)	0111 (7)
0000 (0)	0000 (0)	0001 (1)	0010 (2)	0011 (3)	0100 (4)	0101 (5)	0110 (6)	0111 (7)
0001 (1)	0001 (1)	0010 (2)	0011 (3)	0100 (4)	0101 (5)	0110 (6)	0111 (7)	1000 (8)
0010 (2)	0010 (2)	0011 (3)	0100 (4)	0101 (5)	0110 (6)	0111 (7)	1000 (8)	1001 (9)
0011 (3)	0011 (3)	0100 (4)	0101 (5)	0110 (6)	0111 (7)	1000 (8)	1001 (9)	1010 (10)
0100 (4)	0100 (4)	0101 (5)	0110 (6)	0111 (7)	1000 (8)	1001 (9)	1010 (10)	1011 (11)
0101 (5)	0101 (5)	0110 (6)	0111 (7)	1000 (8)	1001 (9)	1010 (10)	1011 (11)	1100 (12)
0110 (6)	0110 (6)	0111 (7)	1000 (8)	1001 (9)	1010 (10)	1011 (11)	1100 (12)	1101 (13)
0111 (7)	0111 (7)	1000 (8)	1001 (9)	1010 (10)	1011 (11)	1100 (12)	1101 (13)	1110 (14)

....

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Unsigned 4-bit addition table (2)

....	....	....	....	....	....	....	....	....
	0000 (0)	0001 (1)	0010 (2)	0011 (3)	0100 (4)	0101 (5)	0110 (6)	0111 (7)
1000 (8)	1000 (8)	1001 (9)	1010 (10)	1011 (11)	1100 (12)	1101 (13)	1110 (14)	1111 (15)
1001 (9)	1001 (9)	1010 (10)	1011 (11)	1100 (12)	1101 (13)	1110 (14)	1111 (15)	0000 CF (0)
1010 (10)	1010 (10)	1011 (11)	1100 (12)	1101 (13)	1110 (14)	1111 (15)	0000 CF (16+0)	0001 CF (16+1)
1011 (11)	1011 (11)	1100 (12)	1101 (13)	1110 (14)	1111 (15)	0000 CF (16+0)	0001 CF (16+1)	0010 CF (16+2)
1100 (12)	1100 (12)	1101 (13)	1110 (14)	1111 (15)	0000 CF (16+0)	0001 CF (16+1)	0010 CF (16+2)	0011 CF (16+3)
1101 (13)	1101 (13)	1110 (14)	1111 (15)	0000 CF (16+0)	0001 CF (16+1)	0010 CF (16+2)	0011 CF (16+3)	0100 CF (16+4)
1110 (14)	1110 (14)	1111 (15)	0000 CF (16+0)	0001 CF (16+1)	0010 CF (16+2)	0011 CF (16+3)	0100 CF (16+4)	0101 CF (16+5)
1111 (15)	1111 (15)	0000 CF (16+0)	0001 CF (16+1)	0010 CF (16+2)	0011 CF (16+3)	0100 CF (16+4)	0101 CF (16+5)	0110 CF (16+6)

`http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt`

Unsigned 4-bit addition table (3)

	1000 (8)	1001 (9)	1010 (10)	1011 (11)	1100 (12)	1101 (13)	1110 (14)	1111 (15)
0000 (0)	1000 (8)	1001 (9)	1010 (10)	1011 (11)	1100 (12)	1101 (13)	1110 (14)	1111 (15)
0001 (1)	1001 (9)	1010 (10)	1011 (11)	1100 (12)	1101 (13)	1110 (14)	1111 (15)	0000 CF (16+0)
0010 (2)	1010 (10)	1011 (11)	1100 (12)	1101 (13)	1110 (14)	1111 (15)	0000 CF (16+0)	0001 CF (16+1)
0011 (3)	1011 (11)	1100 (12)	1101 (13)	1110 (14)	1111 (15)	0000 CF (16+0)	0001 CF (16+1)	0010 CF (16+2)
0100 (4)	1100 (12)	1101 (13)	1110 (14)	1111 (15)	0000 CF (16+0)	0001 CF (16+1)	0010 CF (16+2)	0011 CF (16+3)
0101 (5)	1101 (13)	1110 (14)	1111 (15)	0000 CF (16+0)	0001 CF (16+1)	0010 CF (16+2)	0011 CF (16+3)	0100 CF (16+4)
0110 (6)	1110 (14)	1111 (15)	0000 CF (16+0)	0001 CF (16+1)	0010 CF (16+2)	0011 CF (16+3)	0100 CF (16+4)	0101 CF (16+5)
0111 (7)	1111 (15)	0000 CF (16+0)	0001 CF (16+1)	0010 CF (16+2)	0011 CF (16+3)	0100 CF (16+4)	0101 CF (16+5)	0110 CF (16+6)
....	....	....	....	....	....	....	....	....

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Unsigned 4-bit addition table (4)

....

	1000 (8)	1001 (9)	1010 (10)	1011 (11)	1100 (12)	1101 (13)	1110 (14)	1111 (15)
1000 (8)	0000 CF (16+0)	0001 CF (16+1)	0010 CF (16+2)	0011 CF (16+3)	0100 CF (16+4)	0101 CF (16+5)	0110 CF (16+6)	0111 CF (16+7)
1001 (9)	0001 CF (16+1)	0010 CF (16+2)	0011 CF (16+3)	0100 CF (16+4)	0101 CF (16+5)	0110 CF (16+6)	0111 CF (16+7)	1000 CF (16+8)
1010 (10)	0010 CF (16+2)	0011 CF (16+3)	0100 CF (16+4)	0101 CF (16+5)	0110 CF (16+6)	0111 CF (16+7)	1000 CF (16+8)	1001 CF (16+9)
1011 (11)	0011 CF (16+3)	0100 CF (16+4)	0101 CF (16+5)	0110 CF (16+6)	0111 CF (16+7)	1000 CF (16+8)	1001 CF (16+9)	1010 CF (16+10)
1100 (12)	0100 CF (16+4)	0101 CF (16+5)	0110 CF (16+6)	0111 CF (16+7)	1000 CF (16+8)	1001 CF (16+9)	1010 CF (16+10)	1011 CF (16+11)
1101 (13)	0101 CF (16+5)	0110 CF (16+6)	0111 CF (16+7)	1000 CF (16+8)	1001 CF (16+9)	1010 CF (16+10)	1011 CF (16+11)	1100 CF (16+12)
1110 (14)	0110 CF (16+6)	0111 CF (16+7)	1000 CF (16+8)	1001 CF (16+9)	1010 CF (16+10)	1011 CF (16+11)	1100 CF (16+12)	1101 CF (12+13)
1111 (15)	0111 CF (16+7)	1000 CF (16+8)	1001 CF (16+9)	1010 CF (16+10)	1011 CF (16+11)	1100 CF (16+12)	1101 CF (16+13)	1110 CF (16+14)

`http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt`

Signed 4-bit addition table (1)

	0000 (0)	0001 (+1)	0010 (+2)	0011 (+3)	0100 (+4)	0101 (+5)	0110 (+6)	0111 (+7)
0000 (0)	0000 (0)	0001 (+1)	0010 (+2)	0011 (+3)	0100 (+4)	0101 (+5)	0110 (+6)	0111 (+7)
0001 (+1)	0001 (+1)	0010 (+2)	0011 (+3)	0100 (+4)	0101 (+5)	0110 (+6)	0111 (+7)	1000 (-8) OF
0010 (+2)	0010 (+2)	0011 (+3)	0100 (+4)	0101 (+5)	0110 (+6)	0111 (+7)	1000 (-8) OF	1001 (-7) OF
0011 (+3)	0011 (+3)	0100 (+4)	0101 (+5)	0110 (+6)	0111 (+7)	1000 (-8) OF	1001 (-7) OF	1010 (-6) OF
0100 (+4)	0100 (+4)	0101 (+5)	0110 (+6)	0111 (+7)	1000 (-8) OF	1001 (-7) OF	1010 (-6) OF	1011 (-5) OF
0101 (+5)	0101 (+5)	0110 (+6)	0111 (+7)	1000 (-8) OF	1001 (-7) OF	1010 (-6) OF	1011 (-5) OF	1100 (-4) OF
0110 (+6)	0110 (+6)	0111 (+7)	1000 (-8) OF	1001 (-7) OF	1010 (-6) OF	1011 (-5) OF	1100 (-4) OF	1101 (-3) OF
0111 (+7)	0111 (+7)	1000 (-8) OF	1001 (-7) OF	1010 (-6) OF	1011 (-5) OF	1100 (-4) OF	1101 (-3) OF	1110 (-2) OF
....	....	....	....	....	....	....	....	....

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Signed 4-bit addition table (2)

....	....	....	....	....	....	....	....	....
	0000 (0)	0001 (+1)	0010 (+2)	0011 (+3)	0100 (+4)	0101 (+5)	0110 (+6)	0111 (+7)
1000 (-8)	1000 (-8)	1001 (-7)	1010 (-6)	1011 (-5)	1100 (-4)	1101 (-3)	1110 (-2)	1111 (-1)
1001 (-7)	1001 (-7)	1010 (-6)	1011 (-5)	1100 (-4)	1101 (-3)	1110 (-2)	1111 (-1)	0000 (0)
1010 (-6)	1010 (-6)	1011 (-5)	1100 (-4)	1101 (-3)	1110 (-2)	1111 (-1)	0000 (0)	0001 (+1)
1011 (-5)	1011 (-5)	1100 (-4)	1101 (-3)	1110 (-2)	1111 (-1)	0000 (0)	0001 (+1)	0010 (+2)
1100 (-4)	1100 (-4)	1101 (-3)	1110 (-2)	1111 (-1)	0000 (0)	0001 (+1)	0010 (+2)	0011 (+3)
1101 (-3)	1101 (-3)	1110 (-2)	1111 (-1)	0000 (0)	0001 (+1)	0010 (+2)	0011 (+3)	0100 (+4)
1110 (-2)	1110 (-2)	1111 (-1)	0000 (0)	0001 (+1)	0010 (+2)	0011 (+3)	0100 (+4)	0101 (+5)
1111 (-1)	1111 (-1)	0000 (0)	0001 (+1)	0010 (+2)	0011 (+3)	0100 (+4)	0101 (+5)	0110 (+6)

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Signed 4-bit addition table (3)

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

....

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

Signed 4-bit addition table (4)

....	....	....	....	....	....	....	....	....
	1000 (-8)	1001 (-7)	1010 (-6)	1011 (-5)	1100 (-4)	1101 (-3)	1110 (-2)	1111 (-1)
1000 (-8)	0000 (0) OF	0001 (+1) OF	0010 (+2) OF	0011 (+3) OF	0100 (+4) OF	0101 (+5) OF	0110 (+6) OF	0111 (+7) OF
1001 (-7)	0001 (+1) OF	0010 (+2) OF	0011 (+3) OF	0100 (+4) OF	0101 (+5) OF	0110 (+6) OF	0111 (+7) OF	1000 (-8)
1010 (-6)	0010 (+2) OF	0011 (+3) OF	0100 (+4) OF	0101 (+5) OF	0110 (+6) OF	0111 (+7) OF	1000 (-8)	1001 (-7)
1011 (-5)	0011 (+3) OF	0100 (+4) OF	0101 (+5) OF	0110 (+6) OF	0111 (+7) OF	1000 (-8)	1001 (-7)	1010 (-6)
1100 (-4)	0100 (+4) OF	0101 (+5) OF	0110 (+6) OF	0111 (+7) OF	1000 (-8)	1001 (-7)	1010 (-6)	1011 (-5)
1101 (-3)	0101 (+5) OF	0110 (+6) OF	0111 (+7) OF	1000 (-8)	1001 (-7)	1010 (-6)	1011 (-5)	1100 (-4)
1110 (-2)	0110 (+6) OF	0111 (+7) OF	1000 (-8)	1001 (-7)	1010 (-6)	1011 (-5)	1100 (-4)	1101 (-3)
1111 (-1)	0111 (+7) OF	1000 (-8)	1001 (-7)	1010 (-6)	1011 (-5)	1100 (-4)	1101 (-3)	1110 (-2)

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt