# Overviews of Carry and Overflow Flags

Young W. Lim

2024-07-22 Mon

# Outline

# Based on

- The CARRY flag and OVERFLOW flag in binary arithmetic
  Ian! D. Allen - idallen@idallen.ca - www.idallen.com
  `https://teaching.idallen.com/dat2343/10f/notes/040_overflow.ttx`

# 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
- Signed and unsigned computations
- Flags for an unsigned number
- Flags for a signed number
- Detecting errors in usigned and signed arithmetic
- The verb to overflow v.s. the overflow flag

# 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*<br>extends an *n-bit* result<br>into an (*n+1*)-bit result | |
| OF Overflow Flag | | detects *overflows*<br>errors<br>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` $\rightarrow$ carry out – the result is too large for an *n-bit* integer
  - $P - P$ `CF=1` $\rightarrow$ 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` $\rightarrow$ overflow – the result is not correct
  - $N + N \rightarrow P$ `OF=1` $\rightarrow$ overflow – the result is not correct
- $P$ (positive), $N$ (negative)

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

# Detecting errors in usigned and signed arithmetic (3)

- 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-f

# 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

# Addition of $n$-bit numbers

| | | | | |
|---|---|---|---|---|
| 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

| | | | | | |
|---|---|---|---|---|---|
| | $a_{n-1}$ | $a_{n-2}$ | $\cdots\cdots\cdots$ | $a_1$ | $a_0$ |
| | $b_{n-1}$ | $b_{n-2}$ | $\cdots\cdots\cdots$ | $b_1$ | $b_0$ |
| | | | | | $c_0$ |
| $c_n$ | $s_{n-1}$ | $s_{n-2}$ | $\cdots\cdots\cdots$ | $s_1$ | $s_0$ |

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<br>$A + B = A + B + 0$ | a transformed addition<br>$A - B = A + \overline{B} + 1$ |
| $\{c_n, s_{n-1}\} = a_{n-1} + b_{n-1} + c_{n-1}$ | $\{c_n, s_{n-1}\} = a_{n-1} + \overline{b_{n-1}} + c_{n-1}$ |
| $\{c_{n-1}, s_{n-2}\} = a_{n-2} + b_{n-2} + c_{n-2}$ | $\{c_{n-1}, s_{n-2}\} = a_{n-2} + \overline{b_{n-2}} + c_{n-2}$ |

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