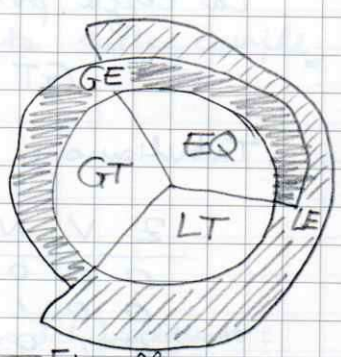
Signed comparisons (operandi espressi in C_2)

$N \leftrightarrow$ SF (sign flag)
$V \leftrightarrow$ OF (overflow flag)
$Z \leftrightarrow$ ZF (zero flag)
$C \leftrightarrow$ CF (carry flag)

Motorola 68000, ARM, etc:

$$\begin{cases} EQ = Z \\ LT = N \oplus V \\ GT = \overline{Z} \cdot (\overline{N \oplus V}) \\ GE = \overline{N \oplus V} \\ LE = Z + (N \oplus V) \end{cases}$$



[dovrebbe essere:
$GE = \overline{LT}$
$LE = \overline{GT}$
$GE = GT + EQ$
$LE = LT + EQ$]

Alla base di LT c'è l'osservazione che il segno corretto di un'operazione in complemento a due è quello che esce dall'ALU se non c'è overflow, mentre è quello opposto se c'è overflow:

$$sgn(a_{n-1}, b_{n-1}, S_{n-1}) = a_{n-1} b_{n-1} + a_{n-1} S_{n-1} + b_{n-1} S_{n-1}$$

N.B. È una funzione formalm. identica al CARRY in uscita dal full adder, ovvia $C(a_i, b_i, c_{i-1})$

$$V(a_{n-1}, b_{n-1}, S_{n-1}) = a_{n-1} b_{n-1} \overline{S}_{n-1} + \overline{a}_{n-1} \overline{b}_{n-1} S_{n-1}$$

overflow correction → sgn; overflow detection → V

a_{n-1}	b_{n-1}	S_{n-1}	sgn	V
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	0
1	1	0	1	1
1	1	1	1	0

⇒

a_{n-1}	b_{n-1}	S_{n-1}	V	sgn
0	0	0	0	0
0	0	0	1	X
0	0	1	0	X
0	0	1	1	0
0	1	0	0	0
0	1	0	1	X
0	1	1	0	1
0	1	1	1	X
1	0	0	0	0
1	0	0	1	X
1	0	1	0	1
1	0	1	1	X
1	1	0	0	X
1	1	0	1	1
1	1	1	0	1
1	1	1	1	X

N.B. Questa tabella è stata ottenuta considerando l'uscita $S_{n-1} S_{n-2} \dots S_0$ di un Sommatore per interi naturali mettendo in ingresso due parole $a_{n-1} a_{n-2} \dots a_0$ e $b_{n-1} b_{n-2} \dots b_0$ in Complemento a due, segnando le situazioni di overflow e riportando il segno corretto $sgn = S_n$ (bit n+1-simo)

$a_{n-1} b_{n-1}$ \ $S_{n-1} V$	00	01	11	10
00	0	X	0	X
01	0	X	X	1
11	X	1	X	1
10	0	X	X	1

$$sgn = \overline{S}_{n-1} V + S_{n-1} \overline{V} = S_{n-1} \oplus V$$

(19)

dove $S_{n-1} = N$ - N.B. è anche $V = S_{n-1} \oplus sgn$, con come è $V = C_{n-2} \oplus C$

Ora: dato per assodato che $LT = N \oplus V$ e che $EQ = Z$, la terza possibilità nel confronto si può ottenere per esclusione:

$$GT = \overline{LT + EQ} = \overline{Z + N \oplus V} = \overline{Z} \cdot \overline{(N \oplus V)}$$

Tabuliamo:

	Z	V	N	EQ	LT	GT
	0	0	0	0	0	1
	0	0	1	0	1	0
	0	1	0	0	1	0
	0	1	1	0	0	1
	1	0	0	1	0	0
a)	1	0	1	1	0	0
b)	1	1	0	1	0	0
c)	1	1	1	1	0	0

Qui notiamo subito una cosa strana: è violata la mutua esclusione delle condizioni GT, LT, EQ: solo una delle 3 dovrebbe essere a 1 in ogni riga della tabella! Come si spiega?

Le spiegazioni sono diverse per le due righe in cui avviene la violazione:

riga a): La configurazione d'ingresso $ZVN = 1 \times 1$, comune alle righe a) e c), non può mai occorrere. Infatti

$$Z = \overline{S_{n-1}} \cdot \overline{S_{n-2}} \cdot \ldots \cdot \overline{S_0}$$

e

$$N = S_{n-1}$$

per cui $N=1 \Rightarrow Z=0$.

Ora, quando c'è un don't care la funzione può assumere qualsiasi valore preferiamo, senza causare alcun problema - questo sistema la riga a).

riga b): A differenza di prima, la configurazione $ZVN = 110$ può in realtà occorrere, anche se è alquanto rara. Infatti ce l'abbiamo quando sommiamo tra loro i due fondo scala negativi su n bit: $(= -2^{n-1})$

$$\begin{array}{rl} A = & 1\,00\ldots0 \; + \\ -B = & 1\,00\ldots0 \; = \\ \hline & 1\,000\ldots0 \end{array}$$

C ← V=1, S ($\Rightarrow N=0, Z=1$)

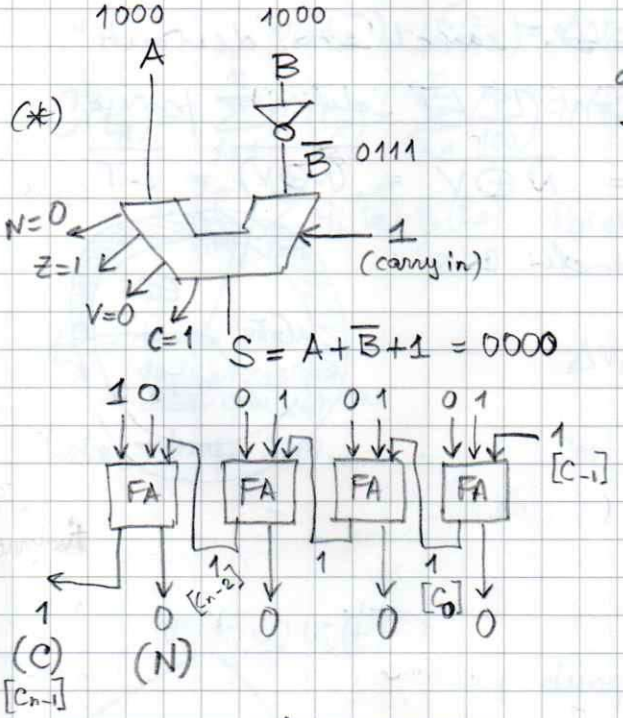
In sostanza, sia A che (−B) sono i "numeri strani" (weird numbers) della rappresentazione in C_2 su n bit. Per cui NON vale l'operazione di cambio di segno basata sulla complementazione. Infatti

$$C_2(100\ldots0) = 100\ldots0 \quad ,$$

operazione che dà luogo a un overflow:

↳ diverso dal precedente: questo è intrinseco al cambiamento di segno, ma non è comunque intervenibile separatamente dall'altro, se si usa lo schema di ALU usuale (*)

$$\begin{array}{rl} & 100\ldots0 \\ & \downarrow \\ C_2\{(C_1) & 011\ldots1 + \\ (+1) & 000\ldots1 = \\ \hline & \boxed{1}00\ldots00 \\ & \downarrow \\ & V=1 \end{array}$$



In pratica, quand'è che si arriva a ottenere ZVN = 110? Solo in un caso, cioè quando

$$A = B = -2^{n-1} \quad ,$$

nel qual caso il risultato corretto dovrebbe essere

$$EQ = 1 \quad , \quad LT = 0$$

Peraltro, considerato che se $-B = -2^{n-1}$ allora $B = 2^{n-1}$ (cosa che non rappresentabile su n bit), allora potremmo ragionevolmente ammettere che sarebbe

$$EQ = 0 \quad , \quad LT = 1 \quad (-2^{n-1} = A < B = 2^{n-1})$$

E invece no! Con lo schema di ALU usuale, NON si manifesta alcun overflow, e quindi NON SIAMO nel caso ZVN=110, bensì siamo nel caso ZVN=100, ossia EQ=1 e LT=0. Quindi anche ZVN=110 dà luogo a un don't care!!!

Si vede anche dal fatto che nell'ultimo stadio $C_{n-2} \oplus \underbrace{C_{n-1}}_{C} = 0$

Questo dimostra che il doppio overflow risulta in un NO OVERFLOW...

Stabilito che GT, LT e EQ sono corrette, notiamo come

↓ ancorché non rispettino la mutua esclusione

mai si arriva a

$$GE = \overline{N \oplus V} \quad \text{e} \quad LE = Z + (N \oplus V)$$

Il secondo caso è regolare: $LE \overset{\text{teorico}}{=} LT + EQ = Z + (N \oplus V)$

↑ somma logica dei due casi

Il primo caso NON lo è: $GE \overset{\text{teorico}}{=} GT + EQ = Z + (N \oplus V)$

Quindi la mancanza di rispetto della mutua esclusione "rovina" anche la logica di GE. Peraltro, notiamo che poiché prima avevamo alterato (interpretando i don't care in maniera a noi conveniente) il solo LT, → GE definito sopra, ancorché anch'esso "derivante", è logicamente coerente con l'LT definito prima. Infatti, si ha $GE = \overline{N \oplus V} = \overline{(N \oplus V)} = \overline{LT}$, che rispetta la tripartizione dei casi: