符号付き2進数の表現(記数法)

機械語序論(第3回)補足資料1

符号無し2進数と符号付き2進数

n ビットの 2 進数の表現には「符号無し 2 進数 (unsigned binary)」と「符号付き 2 進数 (signed binary)」がある。

符号無し2進数は、n ビットに対し、最下位桁の重みを 2^0 、最上位桁の重みを 2^{n-1} として全ビットを加算する。例えば4ビットの数0 1 1 1 と 1 0 1 1 を符号無し2進数として読むと、1 0 進数表現ではそれぞれ 7、1 1 となる。n ビットで表される数値の範囲は0 ~ 2^n - 1 となる。

符号付き 2 進数は、n ビットの 2 進数で作られる 2^n 通りのパターンに対し、正の数と負の数をほぼ同じだけ振り分け、正負の数を表現できるようにしたものである。このため、まず最上位ビットの値が 0 の場合は正、 1 の場合は負の数を表すことにする。残りの n-1 ビットで数値自体を表現する。また、全ビットが 0 というパターンは数字の 0 に対応させる。このため、数字の 0 は正の数のグループに属する。

正の数については、最上位ビットを除く残りの n-1 ビットで、符号無し2進数を表現する。従って、正の数(と0)については、見た目は n ビットの符号無し2進数と同じである。

負の数については「2の補数表現」という特殊な方法を用いる。これは、「-1は0から1を引いたもの」という考え方を元にする。例えば4ビットの0から1を引くと1111となる。実際には引き算の際にさらに上の桁(5ビット目)からの「借り」が生じるが、これは無視する。そして、この1111を「-1」を表すものとする。-2はこれからさらに1を引くので、1110となる。これらのパターンには規則性があり、以下のようにして作ることができる。

「 - x (マイナスx)という数はxの全ビットを反転させ、これに1を加える」

ことによって作ることが出来る。」

例えば、4ビットの場合、「-3」は「3」である0011の全ビットを反転させ(1100)これに1を加えた結果、1101と表現される(上の「1ずつ引いていく」例に倣い、-2からさらに1を引いたパターンがこれに一致することを確認せよ)。

なぜこれで表現できるかは、以下の通りである。 - x という数は数学的には「0 - x」である。0 から予め 1 を引くと、必ずそれは「全ビット 1」という数字になる(例えば 4 ビットの場合は 1 1 1 1 1 2 これから x を引くと、そのパターンは必ず「x の全ビットを反転させたもの」となる(例えば x = x 3 ならば、x 1 x 1 x 1 x 2 x 3 ならば、x 1 x 3 ならば、x 1 x 3 なら引いておいた 1」を、元に戻すために足してやれば結果的に「x 3 を計算したことと等価になる。

従って、「負の数は、最上位ビットを1とし、残りは絶対値」という表現ではないことに注意すること。また、

符号付き 2 進数の負の数で最も小さいものは「10000…0」というパターンになる。これは、 2ⁿ⁻¹

である。よって、 n ビットの符号付き 2 進数が表せる数の範囲は

$$2^{n-1} \sim 2^{n-1} - 1$$

となる。例えば、n=32の場合、-2147483648~+2147483647である。

2 進数の加減算結果とフラグの関係

2 進数の加減算についてのルールは、実は符号無しも符号付きも基本的に同じである。これは、n ビットの加算器あるいは減算器があれば、符号無し・符号付きのどちらの演算にも使えるという意味である。

では、それらを区別するものは何か。それは「結果の読み方と、キャリー及びオーバーフローのフラグの 読み方」である。キャリーフラグは、nビットの加算の結果、最上位桁からさらに1つ上の桁に溢れが生じ た場合1になる。あるいは、減算の結果、最上位からさらに1つ上の桁からの借りが生じた場合も1になる。 従って、元の演算数と被演算数を符号無し2進数と考え、結果も符合無し2進数として捉える場合、もしキャリーフラグが1になっていたら、それは「桁溢れ」または「上の桁からの借り」が発生し、結果のnビットだけで数学的な結果が表せていないことを意味する。

これに対し、オーバーフローフラグの意味は、元の演算数と被演算数及び演算結果を符号付き2進数と考えた時、結果がnビットの符号付き2進数で表現できる範囲を超えているという意味である。例えば、もし演算数と被演算数の符号が異なっていれば、結果は絶対にそれら2つの数値の間に入っているため、オーバーフローは生じない。しかし、プラス同士あるいはマイナス同士の演算の結果、見た目の上で符合が逆転していれば(元の最上位ビットが共に0なのに、加算の結果の最上位ビットは1であった等) これは結果が範囲に収まっていなかったことになる。

従って、演算結果の判定に際し、演算が符号無し2進数で行われているとするならばキャリーフラグをチェックし、符号付き2進数で行われているとするならばオーバーフローフラグを見る、というのが正しい方法である。

4ビットの演算の結果について例を示す。AはBはそれぞれ演算数と被演算数を表し、CFとOFはそれぞれキャリーフラグとオーバーフローフラグを表す。

例1)

A , Bを符号無し2進数とても符号付き2進数と見ても、どちらも4+3=7で桁上げもオーバーフロー も起きていない。

例2)

A , Bを符号無し2進数と見れば、5 + 6 = 1 1 で、1 5 を超えていないため桁上げは生じていない。

A , Bを符号付き2進数と見れば、5 + 6 = 11で、7を超えているためオーバーフローが起きている。

例3)

A, Bを符号無し2進数と見れば、5+13=18で、15を超えているため桁上げが生じている。

A , Bを符号付き 2 進数と見れば、5 + (- 3) = 2 で、オーバーフローは起きていない。

例4)

A , Bを符号無し2進数と見れば、13+10=23で、15を超えてるため桁上げが生じている。

A, Bを符号付き2進数と見れば、(-3)+(-6)=(-9)で、-8を(マイナス側に)超えている ためオーバーフローが起きている。