情報処理の基礎・第 11 回

2013 年 12 月 4 日

今回の内容

11.1 負の整数の符号化

今回は、負の値を含めた一般の整数の符号化と、減算の仕組みについて解説します。

負の整数の 2 の補数表現

通常の CPU では、負の整数を **2 の補数表現**と呼ばれる方法を使って次のように表わします。負の整数 $-x$ を n bit のビット列で表わす際には、まず x を n bit の二進表記で表わし、その各桁の 0/1 を反転してできた二進数に 1 を加えます。たとえば、十進数の -20 を 6 bit のビット列で表現する場合、十進数の 20 を 6 bit の二進表記で表わして 010100、この各桁を反転して 101011 となり、これに 1 を加えた 101100 というビットパターンが -20 の (6 bit での) 2 の補数表現となります。これは、十進法で 44 と表記される正の整数の二進表現でもあることに注意してください。

限られた長さのビット列で正負の整数を表現する場合、最も上位の bit が 0 であるか 1 であるかで正負を区別します。1 であれば負の整数の 2 の補数表現と見なし、0 であれば、非負の整数の通常の二進表現であると見なします。この方法で正負の整数をビット列で表現したものを**符号付き整数表現**と呼びます。これに対して、非負の整数だけを二進法で表現したものを**符号なし整数表現**と呼びます。非負の整数に関しては、符号付きでも符号なしでも、その整数を表わすビットパターンは共通です。

ビット パターン	表している 十進数
011111	31
011110	30
011101	29
011100	28
⋮	⋮
000111	7
000110	6
000101	5
000100	4
000011	3
000010	2
000001	1
000000	0
111111	−1
111110	−2
111101	−3
111100	−4
111011	−5
111010	−6
111001	−7
⋮	⋮
100100	−28
100011	−29
100010	−30
100001	−31
100000	−32

表 1: 6 bit 長の符号付き整数表現

2 の補数表現の良いところは、負の数でも非負の数でも、特に意識することなく、前回に紹介した加算器を使って足し算の計算を行うことができるという点です。たとえば、整数 -20 の 2 の補数表現である 101100 と、整数 20 の二進表現である 010100 を、前節の加算器で足し合わせると、その出力の $S_5\,S_4\,S_3\,S_2\,S_1\,S_0$ の部分が 000000 になります[1]。これは、n bit 長での 2 の補数表現では、x と $-x$ の表現を足し合わせると、ちょうど 2^n になるからです。正の数でも負の数でも $-x$ のビットパターンは、x のビッ

[1]ただし、全体の繰り上がり C_{out} が 1 となります

トパターンの各 bit を反転し、1 を加えたものになります[2]。たとえば、6 bit 長の符号付き整数表現の場合、ビットパターンと、そのパターンが表わしている整数は右の表 1 のような関係になります。一般の n bit 長の符号付き整数表現では、-2^{n-1} から $2^{n-1}-1$ の範囲の整数を表現することができます。限られた長さのビット列で整数を表現しますので、2 つの正の数を足した結果が負の数になってしまったり、2 つの負の数を足した結果が正の数になってしまったりすることがあります。これを**オーバーフロー**と呼びます。計算機のプログラミングを行う際には、オーバーフローが起こらない範囲の整数で計算を行うように注意する必要があります。

減算器　2 の補数表現を利用すると、加算器を使って簡単に減算を行うことができます。引かれる数の bit 表現そのままと、引く数の bit 表現の 0/1 を反転したものを加算器に入力し、加算器全体の C_{in} に 1 を入力します[3]。引く数の -1 倍を引かれる数に足すわけです。

その他の演算　足し算や引き算の他に、掛け算や割り算も、人間の行う筆算の方法をまねて論理回路として実現することができます。また、整数以外にも、小数部をもつ数値などの四則演算も論理回路で計算することが可能です。

11.2　演習問題

1. 次の二進数の計算をそれぞれ行いなさい (計算結果は二進法表記のままでよい)。

$$11101001 - 101011,$$

2. 次の十進数の 2 の補数表現 (8 bit 長) をそれぞれ求めなさい。

$$-1, \quad -2, \quad -3, \quad -10, \quad -11, \quad -100, \quad -101$$

3. 次の 8 bit の符号付き整数表現をそれぞれ十進数に直しなさい。

$$00000000, \quad 00000001, \quad 01100001, \quad 11111111, \quad 11111101, \quad 11100001$$

4. ある整数 x の 32 bit 符号付き整数表現が 01001001001001001001001001001001 であるとき、$-x$ の 32 bit 符号付き整数表現を求めなさい。
5. 8 bit の符号付き整数表現で表現できる整数の範囲を求めなさい。また、符号なし整数表現で表現できる整数の範囲を求めなさい。
6. 同様に、16 bit の場合について求めなさい。
7. 21 と -17 という整数を、符号付き整数表現のビットパターンとして 6 bit の加算器に入力した。このとき、加算器の出力 $S_5\,S_4\,S_3\,S_2\,S_1\,S_0$ のビットパターンを 6 bit の符号付き整数表現とみなすと、どのような整数を表していることになるか考えなさい。
8. 同様に、-21 と -17 という整数を入力した場合について考えなさい。

情報処理の基礎・第 11 回・終り

[2] n bit 長での -2^{n-1} だけは例外で、反転して 1 を加えると自分自身に戻ってしまいます。

[3] 通常の加算の場合は C_{in} へは 0 を入力していたことに注意してください。