

## Jのビット毎(bitwise)演算(b.)と2進法の計算

西川 利男

JAPLAの先月例会で、鈴木義一郎氏より J602の新しい機能、ビット毎(bitwise)演算のプリミティブ(b.)が紹介された。実は、筆者自身も、今年の日技術史教育学会(2009/6/13、東京、工学院大)で「コンピュータの計算はなぜ2進法で行なうのか？」なる講演発表を行い、また先日(11/17)は、めぐろ江戸思想の会で同じ演題で講演を依頼されるなど、このようなテーマには興味を持っている。

2進演算、ブール代数はコンピュータの内部処理、いわゆるシステム処理として必須の機能であるが、BASICなど通常のプログラミング言語ではほとんど不可能である。システム言語Cでのみ可能であるが、Jのこの機能は極めてユニークな特徴である。

今回 J602のビット毎(bitwise)演算を先の2進法計算プログラムに適用してみた。

### 1. コンピュータのハードウェア内部で計算はどうやって行なわれるのか

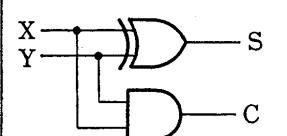
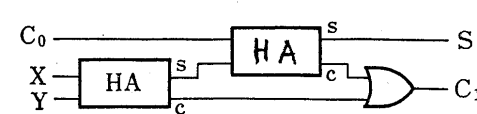
「コンピュータの内部では計算は2進法で行なわれる」とよく言われる。このことを基本に戻って考えてみよう。

さて、2進法での足し算は次の4種類だけである。

0	0	1	1
0	1	0	1
-----			
0	1	1	10

この際、1桁のビットの演算とともに、同時に桁上げという操作が必要である。

このような操作を電子回路で行なうには、以下のようなHalf AdderとFull Adderなるハードウェア・デバイスにより行なわれる。

		半加算器 (Half Adder)				全加算器 (Full Adder)									
															
(a) 加 算 回 路	X	0	1	0	1	X	0	1	0	0	1	1	0	1	
	Y	0	0	1	1	Y	0	0	1	0	1	0	1	1	
	C <sub>0</sub>	0	0	0	1	C <sub>0</sub>	0	0	0	1	0	1	1	1	
	S	0	1	1	0	S	0	1	1	1	0	0	0	0	1
	C	0	0	0	1	C <sub>1</sub>	0	0	0	0	1	1	1	1	1

このような操作を2進数の必要な桁数だけ行うことができる電子回路を実装することがコンピュータ内のすべての計算の基礎になる。

ところで、このような計算操作を2進法以外、例えば10進法でも行なえるだろうか。

実は、コンピュータの草創期には、アメリカでも日本でも10進法の電子計算機は結構あり、EDSACは2進法だが、最初のENIACは10進法だった、ということである。

しかし、回路の簡明化などのため、結局は2進法の計算機だけになった。

プログラム内蔵の電子計算機のパイオニアであるフォン・ノイマン、ウィルクスなどが2進法を採用したことは大変な先見のたまものであった。

高橋秀俊「電子計算機の誕生」中公新書(1972), p. 23, 129-134

## 2. Jのブール代数演算のプリミティブ

Jには以前からブール代数のためのプリミティブがあるが、J602の新しい機能(b.)はこれらを拡充したものである。なお、プリミティブb.自身は副詞であり、左引数をとって動詞として使われる。これらを比較してみる。

機能	J402	J602
NOT 否定	-.	10 b. 26 b.
AND 論理積	*	1 b. 17 b.
NAND 否定論理積	*:	14 b. 30 b.
OR 論理和	+	7 b. 23 b.
NOR 否定論理和	+:	8 b. 24 b.
XOR (Exclusive OR) 排他的論理和	~:	6 b. 22 b.
回転		32 b.
シフト		33 b.
符号付シフト		34 b.

2つには、次のような使い方の異同がある。

```

0 0 1 1 *. 0 1 0 1
0 0 0 1
  0 0 1 1 (17 b.) 0 1 0 1
0 0 0 1
  0 0 1 1 (1 b.) 0 1 0 1
0 0 0 1
  3 (17 b.) 5
1
  3 (1 b.) 5
|domain error
| 3 (1 b.)5
  3 *. 5 NB. 最小公倍数
15

```

```

0 0 1 1 +. 0 1 0 1
0 1 1 1
  0 0 1 1 (23 b.) 0 1 0 1

```

```

0 1 1 1
  0 0 1 1 (7 b.) 0 1 0 1
0 1 1 1
  3 (23 b.) 5
7
  3 (7 b.) 5
|domain error
| 3 (7 b.)5
  3 +. 5 NB. 最大公約数
1

```

### 3. Half Adder, Full Adder のプログラム

NB. Adder and Half\_adder

```

halfadd =: 3 : 0
:
s =. x (6 b.) y NB. Exclusive OR
c =. x (1 b.) y NB. And
s, c
)
NB. fulladd X, Y, Carry => Result, Carry
fulladd =: 3 : 0
'X Y C0' =. y
'S0 Ca' =. X halfadd Y
'S Cb' =. C0 halfadd S0
C1 =. Cb (7 b.) Ca NB. OR
S, C1
)

```

```

    fulladd 0 0 0
0 0
    fulladd 1 0 0
1 0
    fulladd 0 1 0
1 0
    fulladd 0 0 1
1 0
    fulladd 1 1 0
0 1
    fulladd 1 0 1
0 1
    fulladd 0 1 1
0 1
    fulladd 1 1 1

```

1 1

#### 4. 2進法の足し算のプログラムと実行例

NB. Binary Addition 2009/11/30

NB. revised using half\_adder and full\_adder

NB. imported from binary\_add.js on J305, 2008/12/13

```
wr =: 1!:2&2
rd =: 1!:1
add =: 3 : 0
:
N =. 2 + <. 2 ^ . x + y
xb =. #: x
yb =. #: y
wr 'X:', (': X =. (-N) {. xb), ' (= ', (':x), ' decimal)'
wr 'Y:', (': Y =. (-N) {. yb), ' (= ', (':y), ' decimal)'
i =. N - 1
C =. 0
Z =. ''
while. i > 0
do.
wr '====='
wr 'step: ', ": N - i
wr 'X:', ": ((i)#_), (i{X), ((N-1)-i)#_
wr 'Y:', ": ((i)#_), (i{Y), ((N-1)-i)#_
wr 'C:', ": ((i)#_), C, ((N-1)-i)#_
NB. Revised Using fulladd and halfadd
'T C' =. fulladd (i{X), (i{Y), C
wr '-----'
wr 'Z:', (": ((i-1)#_), C, T, ((N-1)-i)#_ ), ' (Carry & Result)'
yn =. rd 1
if. 1 = #yn do. return. end.
Z =. T, Z
i =. i - 1
end.
wr 'Final Result:'
wr 'Z = X + Y'
ZA =. C, Z
wr 'Z:', (": ZA =. C, Z), ' (= ', (":#. ZA), ' decimal)'
wr '====='
'decimal calc.: ', (":x), ' + ', (":y), ' = ', ": #. ZA
)
```

13 add 7

X:0 0 1 1 0 1 (= 13 decimal)

Y:0 0 0 1 1 1 (= 7 decimal)

=====

step: 1

X: \_ \_ \_ \_ 1

Y: \_ \_ \_ \_ 1

C: \_ \_ \_ \_ 0

-----

Z: \_ \_ \_ 1 0 (Carry & Result)

=====

step: 2

X: \_ \_ \_ 0 \_

Y: \_ \_ \_ 1 \_

C: \_ \_ \_ 1 \_

-----

Z: \_ \_ 1 0 \_ (Carry & Result)

=====

step: 3

X: \_ \_ 1 \_ \_

Y: \_ \_ 1 \_ \_

C: \_ \_ 1 \_ \_

-----

Z: \_ 1 1 \_ \_ (Carry & Result)

=====

step: 4

X: \_ 1 \_ \_ \_

Y: \_ 0 \_ \_ \_

C: \_ 1 \_ \_ \_

-----

Z: 1 0 \_ \_ \_ (Carry & Result)

=====

step: 5

X: 0 \_ \_ \_ \_

Y: 0 \_ \_ \_ \_

C: 1 \_ \_ \_ \_

-----

Z: 0 1 \_ \_ \_ (Carry & Result)

Final Result:

Z = X + Y

Z: 0 1 0 1 0 0 (= 20 decimal)

=====

decimal calc.: 13 + 7 = 20

