

かけ算とわり算を見てみよう

…というわけで、ここまで見てきたシフト演算を使ってコンピュータはかけ算やわり算をするわけですが、



そうなんですよね。単純に考えるとシフト演算では、「2、4、8、16…」のような 2^n にあたる数字でしか、かけ算もわり算もできません。「3」とか「7」とかの、半端な数字で行う計算はどうすりゃいいの?という壁にぶち当たります。

2進数の計算と数値表現

そこで、かけ算なら 2^n 同士の足し算に置きかえて計算を行います

かけ算の場合

たとえば7倍だと $? \times 7 = ? \times (4 + 2 + 1)$

こう書きかえることができるので...

$= ? \times (2^2 + 2^1 + 2^0)$

$= (? \times 2^2) + (? \times 2^1) + ?$

この3つも足し算にすれば良い

左に2ビットシフトしたもの

左に1ビットシフトしたもの

元の数値

2^n であらわされる数にバラす!

あ! 確かに!

一方のわり算。わり算も基本は同じなのですが、こちらはまず「わり算って何?」というところから整理する必要があります。

「なに? だって「分割できる数」の計算でしょ?」

20 ÷ 5 は 4 がつんの? 20 という数字からは 5 を 4 回、引くことができる

つまりはこんな引き算だと見なすことができる!

そこで、わり算なら 2^n の引き算を用いて計算を行うわけ

わり算の場合

たとえば15を3でわるとする場合...

2進数に直すと...

$15 \div 3 \rightarrow 1111_2 \div 11_2$

この元の数字を 1111_2 超えない範囲で 15

わる数を左にシフトして $1100_2 \rightarrow \frac{1111}{-1100} \geq 11_2$ (引いてやる)

再度わる数を左にシフト $11_2 \rightarrow \frac{11}{-11} \geq 0$ (引いてやる)

引き算の結果を $\frac{11}{3}$ 超えない範囲で

2ビットシフト

0ビットシフト

...ということは元の数値から引くことができた回数を探るには、この2つを足してやれば良い

$2^2 + 2^0 = 4 + 1 = 5$

引くことのできた回数

こんな感じの筆算をしているのと同じこと

```

    5
  101
11)1111
  11
  ---
   0
   11
   11
   ---
    0
    
```