紅の魔術工房
Atelier kurenai

CTF 入門

CRC32 で誤りを検知する

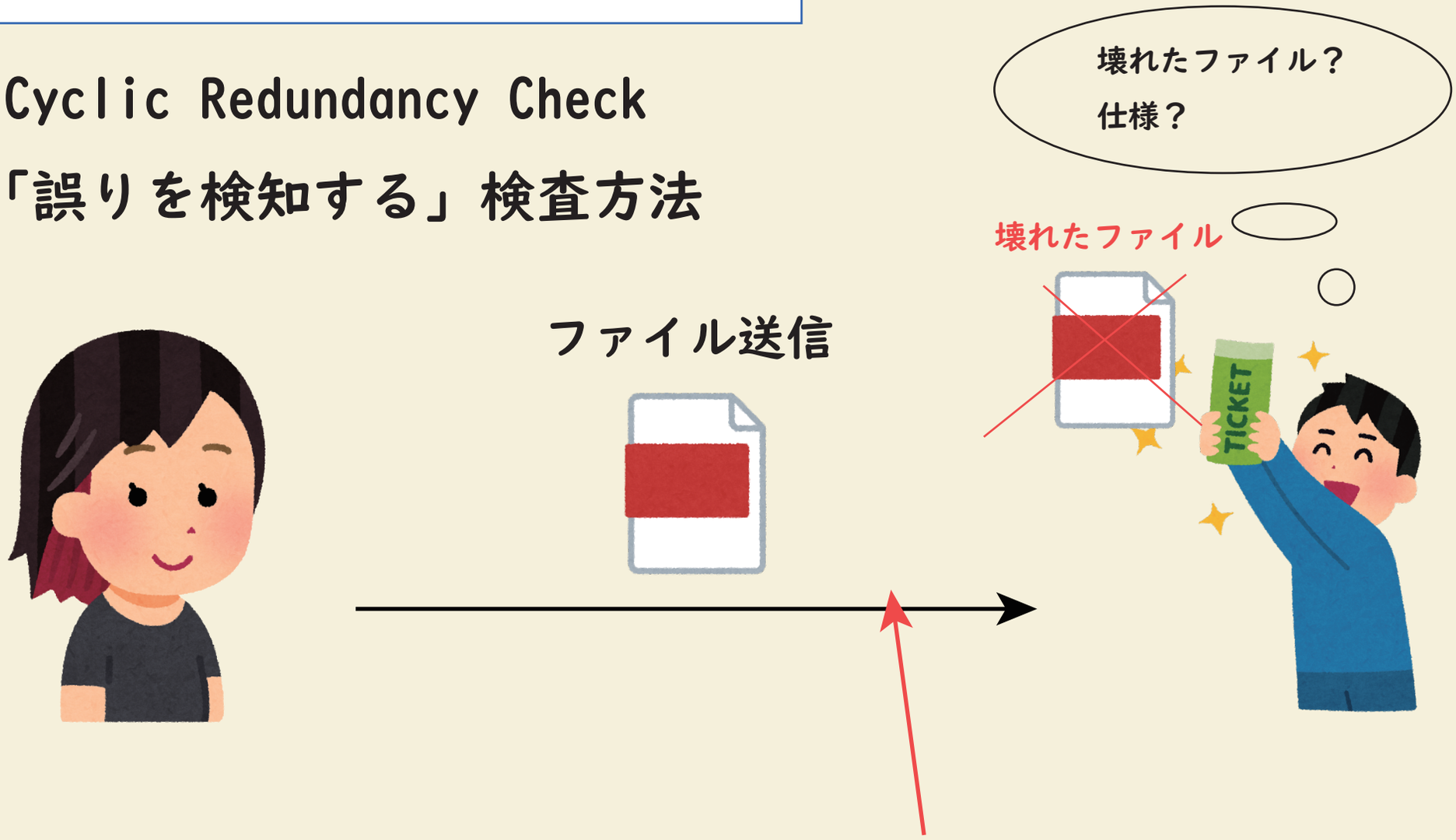
@kurenaif

実践編

CRC32 って？

Cyclic Redundancy Check

「誤りを検知する」検査方法



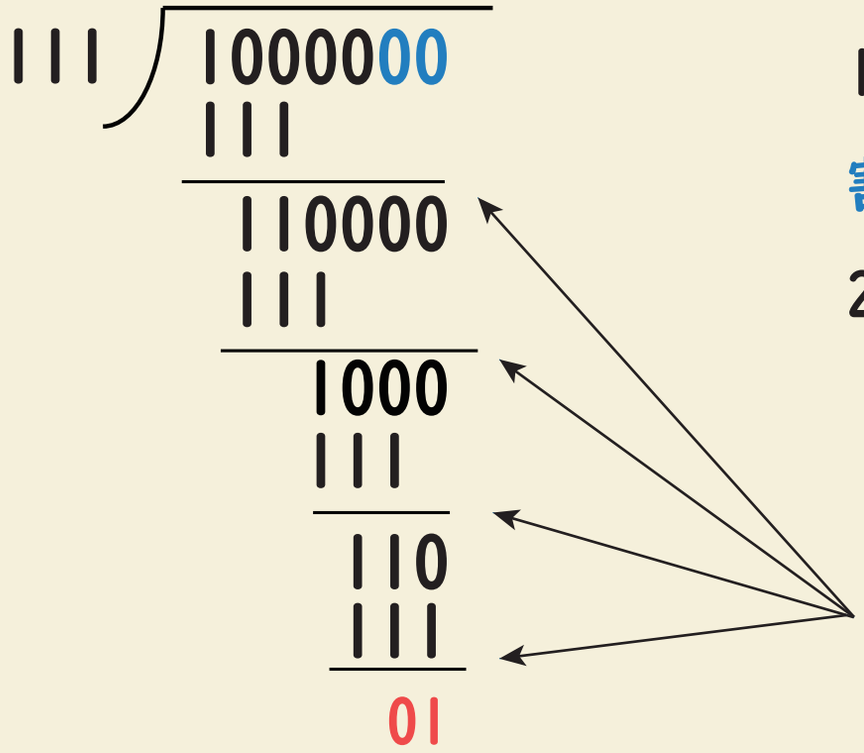
どこで CRC32 は使われているの？

png ファイルとかに内蔵されている

```
  IHDR Interlace: 0
  IHDR Compression algorithm is Deflate
  IHDR Filter method is type zero (None, Sub, Up, Average, Paeth)
  IHDR Interlacing is disabled
  Chunk CRC: -869110134
Chunk: Data Length 309 (max 2147483647), Type 1346585449 [iCCP]
  Ancillary, public, PNG 1.2 compliant, unsafe to copy
  ... Unknown chunk type
  Chunk CRC: -960355186
Chunk: Data Length 65536 (max 2147483647), Type 1413563465 [IDAT]
  Critical, public, PNG 1.2 compliant, unsafe to copy
  IDAT contains image data
  Chunk CRC: 1441794358
Chunk: Data Length 65536 (max 2147483647), Type 1413563465 [IDAT]
  Critical, public, PNG 1.2 compliant, unsafe to copy
  IDAT contains image data
  Chunk CRC: -1856956801
Chunk: Data Length 65536 (max 2147483647), Type 1413563465 [IDAT]
  Critical, public, PNG 1.2 compliant, unsafe to copy
  IDAT contains image data
  Chunk CRC: 126825411
Chunk: Data Length 19446 (max 2147483647), Type 1413563465 [IDAT]
  Critical, public, PNG 1.2 compliant, unsafe to copy
  IDAT contains image data
  Chunk CRC: -425976308
Chunk: Data Length 0 (max 2147483647), Type 1145980233 [IEND]
  Critical, public, PNG 1.2 compliant, unsafe to copy
  IEND contains no data
  Chunk CRC: -1371381630
```

今日のこの講義では
この CRC を求めるための
理論を勉強します

CRC の概要

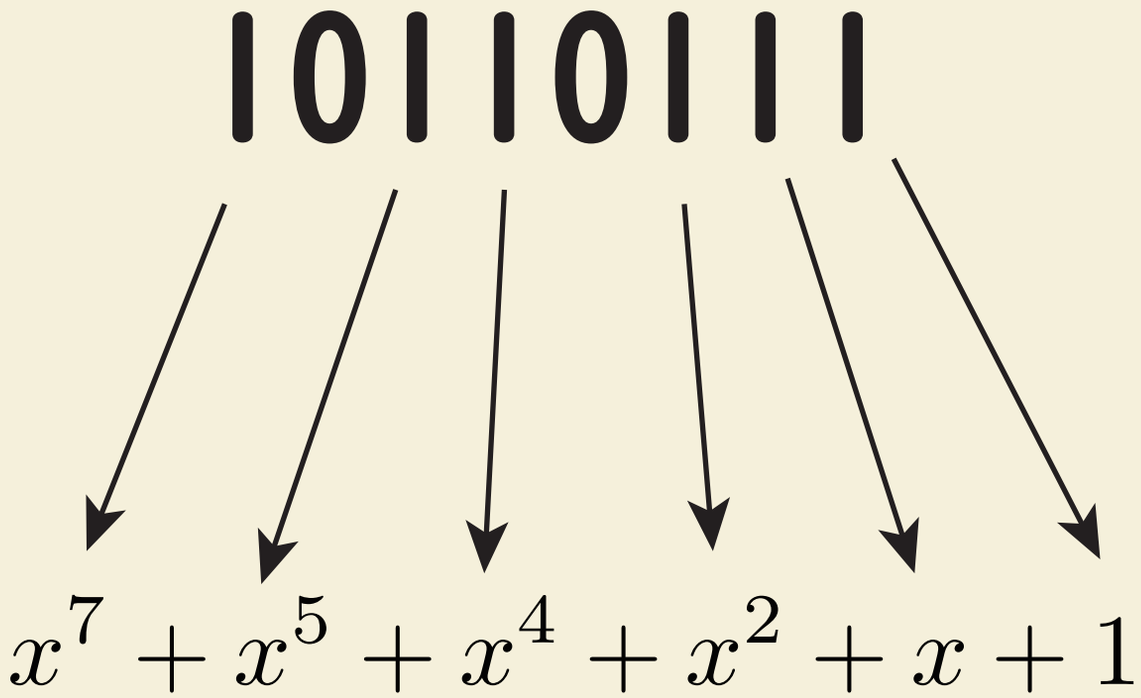


1. 割られる数（生データ）に、
割る数の bit 数 -1 個分 0 を入れる
2. あまりを求める。

筆算の引き算の代わりに
XOR を計算する。

10000（生データ）と 01 を相手に送信する。

bit 列と多項式の対応



実はこの bit 列は多項式を表しています！

CRC の多項式

CRC の、割る数はファイルや用途によって異なる。

		(0x02CC)
CRC-16-Fletcher	CRCではない。フレッチャーの検査合計	Adler-32 A & B CRC で使用
CRC-16-CCITT	$x^{16} + x^{12} + x^5 + 1$ （X.25、V.41、CDMA、Bluetooth、XMODEM、HDLC、PPP、IrDA、BACnet; *CRC-CCITT*とも）	0x1021 / 0x8408 (0x8810 [5])
CRC-16-IBM	$x^{16} + x^{15} + x^2 + 1$ （SDLC、USB、その他; *CRC-16*とも）	0x8005 / 0xA001 (0xC002)
CRC-24-Radix-64	$x^{24} + x^{23} + x^{18} + x^{17} + x^{14} + x^{11} + x^{10} + x^7 + x^6 + x^5 + x^4 + x^3 + x + 1$ (FlexRay)	0x864CFB / 0xDF3261 (0xC3267D)
CRC-30	$x^{30} + x^{29} + x^{21} + x^{20} + x^{15} + x^{13} + x^{12} + x^{11} + x^8 + x^7 + x^6 + x^2 + x + 1$ (CDMA)	0x2030B9C7 / 0x38E74301 (0x30185CE3)
CRC-32-Adler	CRCではない; Adler-32	Adler-32参照
CRC-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ (V.42, MPEG-2, zlib, PNG [10])	0x04C11DB7 / 0xEDB88320 (0x82608EDB [7])

参考: https://ja.wikipedia.org/wiki/%E5%B7%A1%E5%9B%9E%E5%86%97%E9%95%B7%E6%A4%9C%E6%9F%BB

png の CRC の多項式

png の多項式は これ（最上位 bit は省略されてるので、実際は 0x104C11DB7）

CRC-16-Fletcher	CRCではない。フレッチャーの検査合計	Adler-32 A & B CRC で使用
CRC-16-CCITT	$x^{16} + x^{12} + x^5 + 1$ (X.25、V.41、CDMA、Bluetooth、XMODEM、HDLC、PPP、IrDA、BACnet; *CRC-CCITT*とも)	0x1021 / 0x8408 (0x8810 [5])
CRC-16-IBM	$x^{16} + x^{15} + x^2 + 1$ (SDLC、USB、その他; *CRC-16*とも)	0x8005 / 0xA001 (0xC002)
CRC-24-Radix-64	$x^{24} + x^{23} + x^{18} + x^{17} + x^{14} + x^{11} + x^{10} + x^7 + x^6 + x^5 + x^4 + x^3 + x + 1$ (FlexRay)	0x864CFB / 0xDF3261 (0xC3267D)
CRC-30	$x^{30} + x^{29} + x^{21} + x^{20} + x^{15} + x^{13} + x^{12} + x^{11} + x^8 + x^7 + x^6 + x^2 + x + 1$ (CDMA)	0x2030B9C7 / 0x38E74301 (0x30185CE3)
CRC-32-Adler	CRCではない; Adler-32	Adler-32参照
CRC-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ (V.42, MPEG-2, zlib, PNG [10])	0x04C11DB7 / 0xEDB88320 (0x82608EDB [7])

参考: https://ja.wikipedia.org/wiki/%E5%B7%A1%E5%9B%9E%E5%86%97%E9%95%B7%E6%A4%9C%E6%9F%BB

png の CRC

様々なトラブルに対応するため、様々な**前処理**・**後処理**がある。

https://www.slideshare.net/7shi/crc32#31
こちらの 7shi さんの資料を参考に作らせて
いただきました…！

前処理

バイナリデータ（16 進数表記 ）

0x0000deadbeefcafebabe

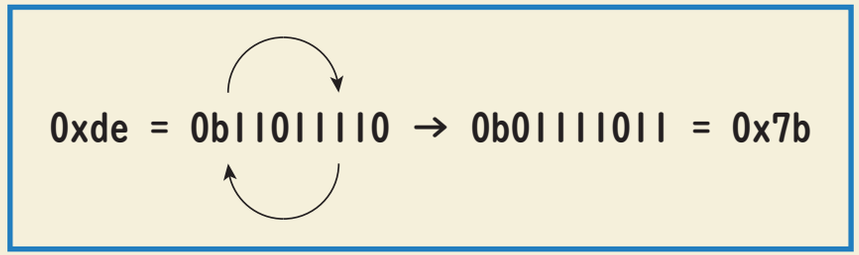
↓ 1byte=8bits ずつ分解

00 00 de ad be ef ca fe ba be

↓ 1byte ずつ bit の「位置を」反転させる

00 00 7b b5 7d f7 53 7f 5d 7d

↓



0x00007bb57df7537f5d7d

前処理

0x00007bb57df7537f5d7d

↓

4bytes=32bits 分 0 を追加
（割る数 -1bit 分）

0x00007bb57df7537f5d7d00000000

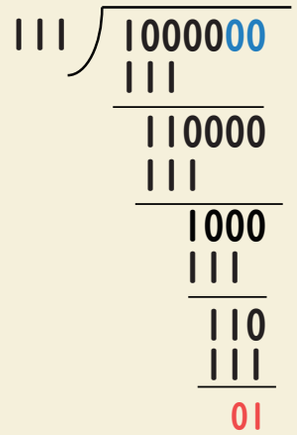
↓

上位 4bytes を bitflip

0xffff844a7df7537f5d7d00000000

0x10 と 0x000000000010 を区別するために
flip する

あまりを求める。



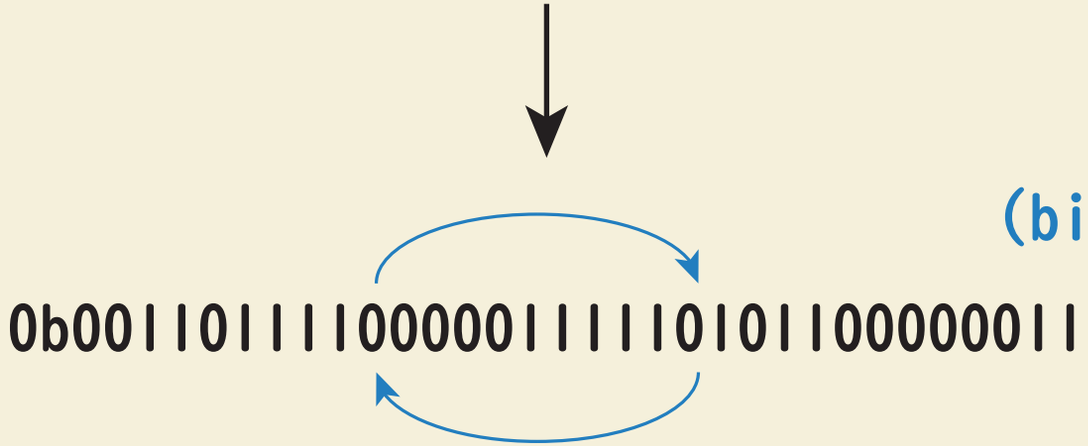
0x1db710640 で割り算すると ...

0xc0d7c1ec になります！！！！！！！！

後処理

（0xc0d7c1ec の２進数）

0b11000000111010111110000011111011100



（bit の位置を反転）

0b00111011111000001111110101110000001l

（bitflip）

0b11001000001111110000010100111111100

要所要所に出てくる「位置反転」の謎

CRC の「あまり」を求める実装が実はめんどくさく、

もうちょっといいやり方がある。

そのやり方に合わせた仕様になっている…？

（調べたけど何故こうなっているかは不明でした。）

割り算１回分の過程

めんどくさいポイント１

割る数：１１１

割られる数：１０００００

ここの差を計算して

１０００００
⟷ １１１

↓

１０００００
１１１０００　bitshift

↓

１１０００　XOR

ここの差を計算するフェイズが
めんどくさい。

bitshift もめんどくさい。

最上位 bit が１だったら
XOR をかけるけど
最上位 bit が動くのが辛い

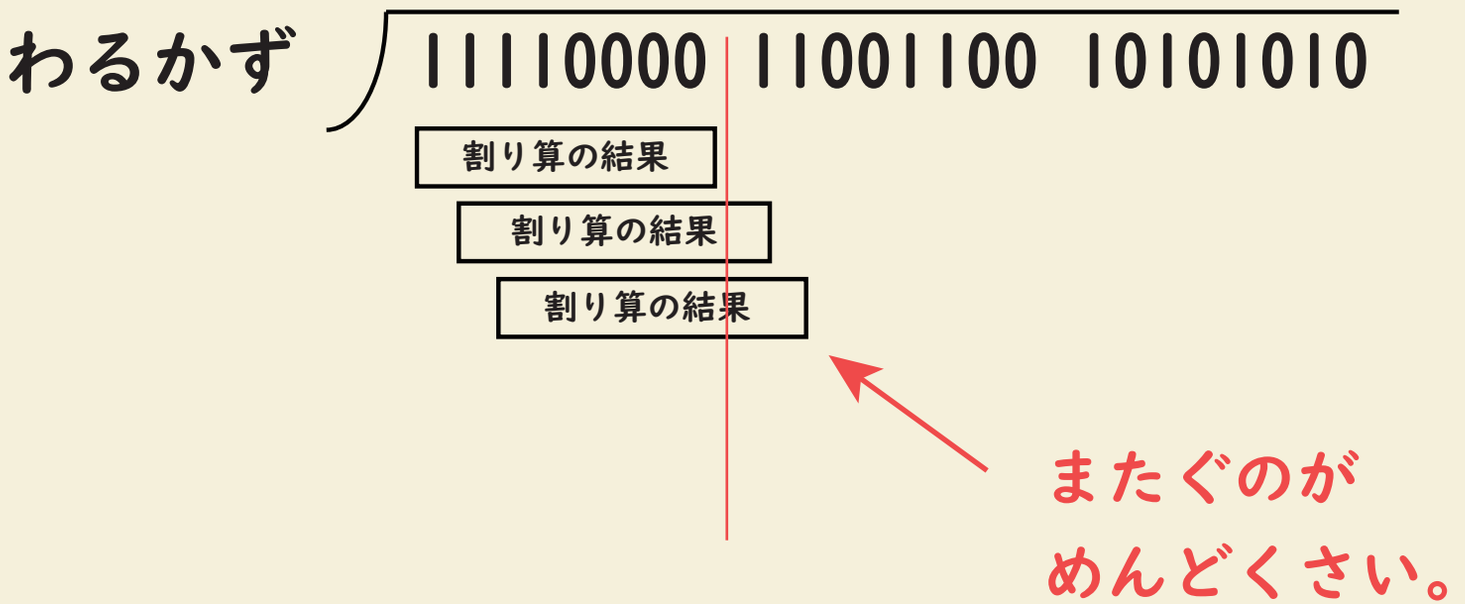
紅の魔術工房
Atelier kurenaif

めんどくさいポイント 2

普通データは 1byte ずつ読み込む。

byte をまたいだときの割り算の実装がめんどくさい。

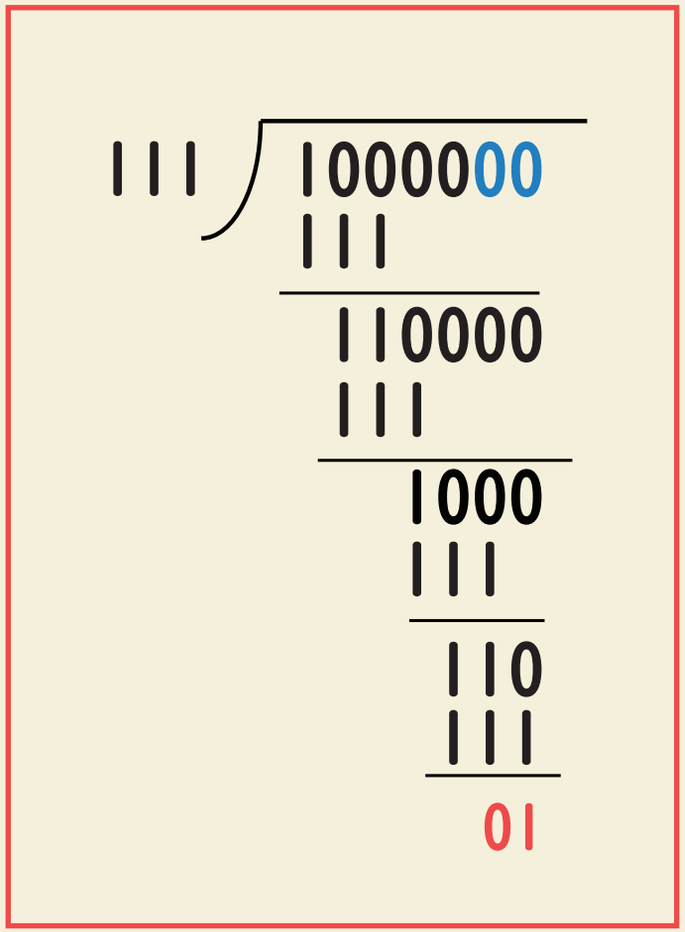
kurenaif 参考実装では簡単のため多倍長整数を使用しているが、
高速なのが売りの CRC で多倍長整数は使用したくない。



(実際には割り算の結果は 32bits ずつ でもめんどくさい)

紅の魔術工房
Atelier kurenaif

めんどくさいポイント3



筆算みたいに計算するのがそもそも
めんどくさい。

3 つのめんどくさいを解消する。

1. 最上位 bit が動くのがめんどくさい
2. byte 列のつなぎ目がめんどくさい
3. 筆算的過程がそもそもめんどくさい

3 つのめんどくさいを解消する。

1. 最上位 bit が動くのがめんどくさい
2. byte 列のつなぎ目がめんどくさい
3. 筆算的過程がそもそもめんどくさい

最上位 bit が動くのがめんどくさい

割り算の考え方を変える。

割る数： 111

割られる数： 1000000

1000000
111**0000**　最初だけ bitshift
110000
↓
110000 ← 割る数を bitshift
1110000
1000
↓
1000 ← 割る数を bitshift
1110000
110
↓
110 ← 割る数を bitshift
1110000
01

同じ値

111 ⟌ 1000**000**
111
110000
111
1000
111
110
111
01

最上位 bit 固定化成功

紅の魔術工房
Atelier kurenaif

最上位 bit が動くのがめんどくさい

割り算の考え方を変える。

1000000
111 0000　最初だけ bitshift
110000
↓
110000 ← 割る数を bitshift
1110000
1000
↓
1000 ← 割る数を bitshift
1110000
110
↓
110 ← 割る数を bitshift
1110000
01

同じ値

大切なのは、「割る数」と

「割られる数」の相対的な位置

111) 1000000
111
110000
111
1000
111
110
111
01

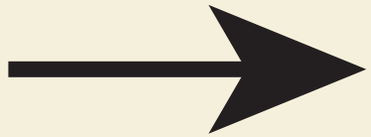
bit を反転してみる

割る数： 111

割られる数： 1000000

相対的な位置が重要なのであれば、このようにしても問題ないのでは？

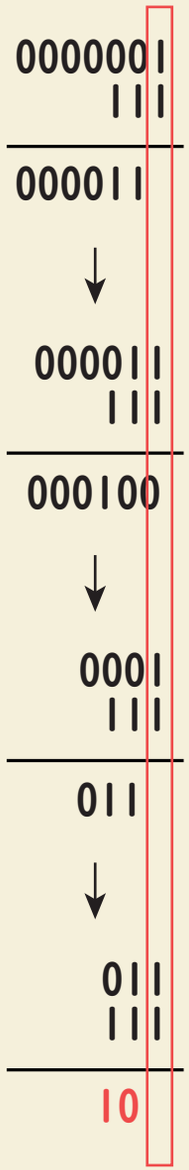
```
 1000000
 1110000
---------
  110000
```



bit が左右反転した世界

```
 0000001
     111
---------
  000011
```

bit を反転してみる

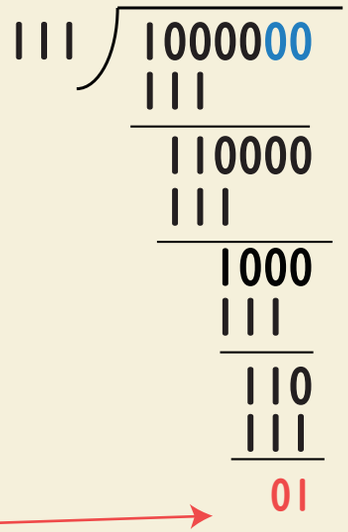


lbit 目が 1 になるまで shift

lbit 目を見る &

bitshift で実装可能

とても楽。



反転すると同じ値

紅の魔術工房
Atelier kurenaif

左右反転した世界の割り算まとめ

1. 割る数を左右反転させる

2. 割る数の lbit 目が 1 なら XOR

3. 割る数を右に 1 bitshift して「2.」へ

4. 求めた「あまり」を左右反転させる

タイトル

紅の魔術工房
Atelier kurenaif

タイトル

紅の魔術工房
Atelier kurenaif

タイトル

紅の魔術工房
Atelier kurenaif

タイトル

紅の魔術工房
Atelier kurenaif