IT人 (/)

# 學習筆記 詳解一種高效位反轉演算法

dadalaohua 發表於 2020-10-01

演算法 (/topic/26.html)

詳解一種高效位反轉演算法

- 位反轉 (https://blog.csdn.net/u012028275/article/details/108895271#_1)
- 演算法原理 (https://blog.csdn.net/u012028275/article/details/108895271#_4)
- 32位數的高效位反轉演算法實現 (https://blog.csdn.net/u012028275/article/details/108895271#32_8)
- 8位數的高效位反轉演算法實現 (https://blog.csdn.net/u012028275/article/details/108895271#8_297)

# 位反轉

這裡的位反轉(Bit Reversal)，指的是一個數的所有bit位依照中點對換位置，例如0b0101 0111 => 0b1110 1010。也可以叫二進位制逆序，按位逆序，位翻轉等等。

# 演算法原理

高效位反轉演算法原理：演算法運用了分治法(divide and conquer)，以兩個bit位一組，對調相鄰的bit位；然後再以4個bit位為一組，分成左邊兩個bit位一段和右邊兩個bit位一段，然後這兩段相互對調；然後再以8個bit位為一組，以此類推，最後完成位反轉。

# 32位數的高效位反轉演算法實現

下面舉例一個32位數的高效位反轉演算法程式碼:

```
unsigned int reverse(unsigned int x)
{
    x = (((x & 0xaaaaaaaa) >> 1) | ((x & 0x55555555) << 1));
    x = (((x & 0xcccccccc) >> 2) | ((x & 0x33333333) << 2));
    x = (((x & 0xf0f0f0f0) >> 4) | ((x & 0x0f0f0f0f) << 4));
    x = (((x & 0xff00ff00) >> 8) | ((x & 0x00ff00ff) << 8));

    return((x >> 16) | (x << 16));
}
```

下面進行逐行程式碼分析：

首先，我們要進行位反轉的是一個32位數，如下圖所示

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

分析第一行程式碼：

x = (((x & 0xaaaaaaaa) >> 1) | ((x & 0x55555555) << 1));


1. `0xaaaaaaaa`

   0xaaaaaaaa 是從第0位開始，所有奇數位為1

   0xaaaaaaaa = 0b10101010101010101010101010101010

2. `x & 0xaaaaaaaa`

   x & 0xaaaaaaaa的結果如下圖所示：


3. `(x & 0xaaaaaaaa) >> 1`
   右移一位後結果如下圖所示：

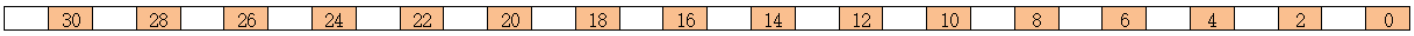

4. `0x55555555`

   0x55555555是從第0位開始，所有偶數位為1

   0x55555555 = 0b01010101010101010101010101010101

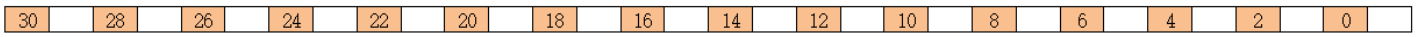
5. `x & 0x55555555`

   x & 0x55555555的結果如下圖所示：



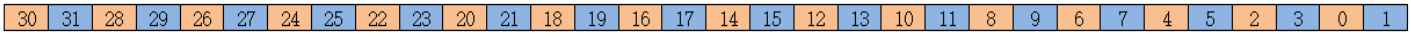

6. `(x & 0x55555555) << 1`

   左移一位的結果如下圖所示：




7. `x = (((x & 0xaaaaaaaa) >> 1) | ((x & 0x55555555) << 1));`

然後兩個數或運算，結果如下圖所示：

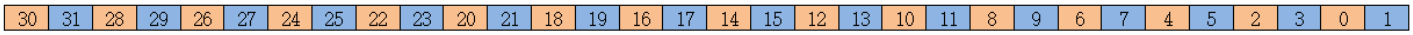



第一行程式碼運算完成。

總結，`x = (((x & 0xaaaaaaaa) >> 1) | ((x & 0x55555555) << 1));`這行的程式碼就是以兩個bit位一組，對調相鄰的bit位。

圖解就是將下圖

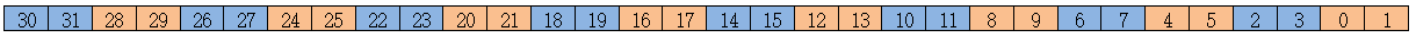
轉換成




分析第二行程式碼：

x = (((x & 0xcccccccc) >> 2) | ((x & 0x33333333) << 2));

目前x的數值為：



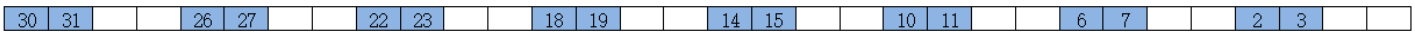

1. `0xcccccccc`

   0xcccccccc是從第0位開始，0和1每隔兩位交替出現0

   xcccccccc = 0b11001100110011001100110011001100

2. `x & 0xcccccccc`

   x & 0xcccccccc的結果如下圖所示：




3. `(x & 0xcccccccc) >> 2`

   右移兩位後結果如下圖所示：

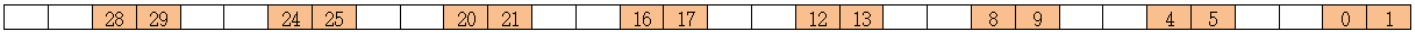

4. `0x33333333`

   0x33333333是從第0位開始，1和0每隔兩位交替出現

   0x33333333 = 0b00110011001100110011001100110011

5. `x & 0x33333333`

x & 0x33333333的結果如下圖所示：

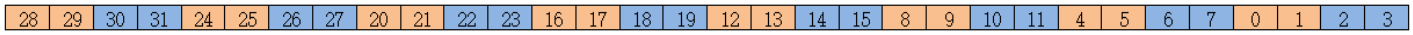


6. `(x & 0x33333333) << 2`

左移兩位後結果如下圖所示：


7. `x = (((x & 0xcccccccc) >> 2) | ((x & 0x33333333) << 2));`

然後兩個數或運算，結果如下圖所示：

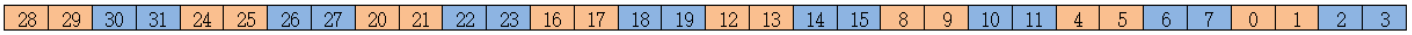



第二行程式碼運算完成。

總結，`x = (((x & 0xcccccccc) >> 2) | ((x & 0x33333333) << 2));`這行的程式碼就是以4個bit位為一組，分成左邊兩個bit位一段和右邊兩個bit位一段，然後這兩段相互對調。

圖解就是將下圖


轉換成




---

分析第三行程式碼：

x = (((x & 0xf0f0f0f0) >> 4) | ((x & 0x0f0f0f0f) << 4));

目前x的數值為：


1. `0xf0f0f0f0`

   0xf0f0f0f0是從第0位開始，0和1每隔四位交替出現

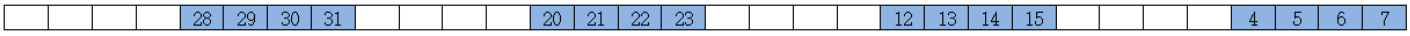
   0xf0f0f0f0= 0b11110000111100001111000011110000

2. `x & 0xf0f0f0f0`

x & 0xf0f0f0f0的結果如下圖所示：


3. `(x & 0xf0f0f0f0) >> 4`

右移四位後結果如下圖所示：



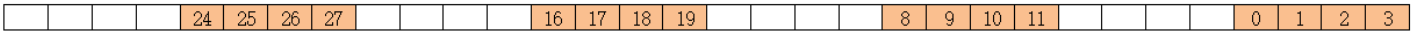

4. `0x0f0f0f0f`

0x0f0f0f0f是從第0位開始，1和0每隔四位交替出現

0x0f0f0f0f= 0b00001111000011110000111100001111

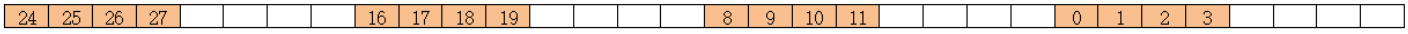
5. `x & 0x0f0f0f0f`

x & 0x0f0f0f0f的結果如下圖所示：



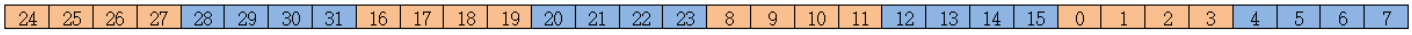

6. `(x & 0x0f0f0f0f) << 4`

左移四位後結果如下圖所示：




7. `x = (((x & 0xf0f0f0f0) >> 4) | ((x & 0x0f0f0f0f) << 4));`

然後兩個數或運算，結果如下圖所示：




第三行程式碼運算完成。

總結，`x = (((x & 0xf0f0f0f0) >> 4) | ((x & 0x0f0f0f0f) << 4));`這行的程式碼就是以8個bit位為一組，分成左邊四個bit位一段和右邊四個bit位一段，然後這兩段相互對調。

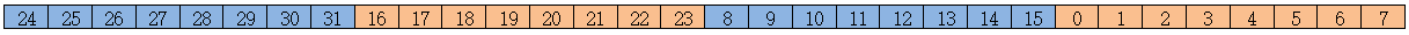
圖解就是將下圖


轉換成


分析第四行程式碼：

x = (((x & 0xff00ff00) >> 8) | ((x & 0x00ff00ff) << 8));

目前x的數值為：




1. `0xff00ff00`

   0xff00ff00是從第0位開始，0和1每隔八位交替出現

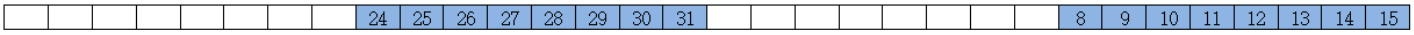
   0xff00ff00= 0b11111111000000001111111100000000

2. `x & 0xff00ff00`

   x & 0xff00ff00的結果如下圖所示：


3. `(x & 0xff00ff00) >> 8`

   右移八位後結果如下圖所示：




4. `0x00ff00ff`

   0x00ff00ff是從第0位開始，1和0每隔八位交替出現

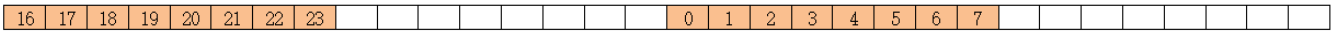
   0x00ff00ff= 0b00000000111111110000000011111111

5. `x & 0x00ff00ff`

x & 0x00ff00ff的結果如下圖所示：


6. `(x & 0x00ff00ff) << 8`

左移八位後結果如下圖所示：




7. `x = (((x & 0xff00ff00) >> 8) | ((x & 0x00ff00ff) << 8));`

然後兩個數或運算，結果如下圖所示：


第四行程式碼運算完成。

總結，`x = (((x & 0xff00ff00) >> 8) | ((x & 0x00ff00ff) << 8));`這行的程式碼就是以16個bit位為一組，分成左邊八個bit位一段和右邊八個bit位一段，然後這兩段相互對調。

圖解就是將下圖


轉換成


---

分析最後一行程式碼：

((x >> 16) | (x << 16))

目前x的數值為：


高16bit與低16bit進行交換，結果如下圖：

完成整個位反轉演算法。

# 8位數的高效位反轉演算法實現

如果要對8位數進行位反轉，原理相同，程式碼如下：

```
unsigned char reverse(unsigned char x)
{
    x = (((x & 0xaa) >> 1) | ((x & 0x55) << 1));
    x = (((x & 0xcc) >> 2) | ((x & 0x33) << 2));

    return ((x >> 4) | (x << 4));
}
```

即可實現對8位數的高效位反轉。

[參考資料]

[Hacker’s Delight] 作者: Henry S. Warren Jr.

The Aggregate Magic Algorithms (http://aggregate.org/MAGIC/)

## 相關文章

CSS基礎——浮動（float）【學習筆記】 (/558223.html)

2020-11-21 CSS (/topic/85.html)

10大排序演算法——Java實現 (/558276.html)

2020-11-21 Java (/topic/16.html) 演算法 (/topic/26.html)

Go踩坑筆記（十九） (/558278.html)

2020-11-21 Go (/topic/134.html)

演算法：排序連結串列：歸併排序 (/558305.html)

2020-11-21 演算法 (/topic/26.html)

com.alibaba.fastjson學習筆記 (/558317.html)

2020-11-21

---

React學習筆記之雙向資料繫結 (/558323.html)

2020-11-21 React (/topic/48.html)

---

每天一道演算法題系列十三之羅馬數字轉整數 (/558389.html)

2020-11-21 演算法 (/topic/26.html)

---

用 React.js+Egg.js 造輪子 全棧開發旅遊電商應用學習筆記和心得 (/558397.html)

2020-11-21 全棧 (/topic/25.html) React (/topic/48.html)

---

CSS技術筆記 (/558434.html)

2020-11-22 CSS (/topic/85.html)

---

作業系統實驗：銀行家演算法（C語言） (/558450.html)

2020-11-22 演算法 (/topic/26.html)

---

LOAM演算法核心解析之特徵點的提取(一) (/558457.html)

2020-11-22 演算法 (/topic/26.html)

---

華為面試題：購物車問題（01揹包演算法升級) (/558525.html)

2020-11-22 面試 (/topic/19.html) 演算法 (/topic/26.html)

---

《吳恩達機器學習》學習筆記007_支援向量機 (/558530.html)

2020-11-22 Machine Learning (/topic/22.html)

---

Head First Java學習筆記(7):繼承與多型 (/558551.html)

2020-11-22 Java (/topic/16.html)

---

死磕以太坊原始碼分析之Kademlia演算法 (/558557.html)

2020-11-22 演算法 (/topic/26.html)

計網 應用層筆記(個人學習用, 如有錯誤萬分感謝指出 (( 看到哪更到哪 ) (/558562.html)

2020-11-22

現代作業系統-原理與實現【讀書筆記】 (/558572.html)

2020-11-22

JavaScript正則學習筆記 (/558578.html)

2020-11-22 JavaScript (/topic/39.html)

《推薦系統實踐》筆記 01 推薦系統簡介 (/558644.html)

2020-11-22

## 最新文章

基於Dtm分散式事務管理的php客戶端 (/647207.html)

Go執行指令碼命令用例及原始碼解析 (/647206.html)

剖析虛幻渲染體系（13）- RHI補充篇：現代圖形API之奧義與指南 (/647203.html)

30個類手寫Spring核心原理之依賴注入功能（3） (/647202.html)

支小蜜人臉識別消費系統助力學校食堂實現“刷臉吃飯” (/647201.html)

ZooKeeper 06 - ZooKeeper 的常用命令 (/647196.html)

PyCharm2021.3，程式設計軟體 (/647192.html)

Navicat Premium 16 (/647191.html)

Snagit 2022，螢幕錄製 (/647190.html)

react18 來了，我 get 到... (/647194.html)

論文翻譯：2021_A Perceptually Motivated Approach for Low-complexity, Real-time Enhancement of Fullband Speech (/647188.html)

單例模式的七種寫法 (/161076.html)