УДК 621.37: 681.33

І.І. МІТАСОВ, І.В. КУЦИЙ
Хмельницький національний університет

ПРОГРАМУВАННЯ ТАБЛИЧНИХ АЛГОРИТМІВ CRC-КОНТРОЛЮ ПЕРЕДАЧІ ДАНИХ

В статті охарактеризовані відомі методи CRC-контролю передачі даних, розглянуто загальну схему реального порозрядного алгоритму обчислення CRC та особливості алгоритмів обчислення CRC, проілюстрована сутність табличного алгоритму обчислення CRC. Наведені фрагменти програм мовою асемблера.

In the articles described the known methods of CRC-control of communication of data, the general chart of the real digit-by-digit algorithm of calculation of CRC and features of algorithms of calculation of CRC is considered, illustrated essence of tabular algorithm of calculation of CRC. Resulted fragments of the programs by an assembly language.

Ключові слова: табличні алгоритми, передача даних, CRS-контроль.

Вступ. Усі відомі методи виявлення помилок призначені для опрацювання пошкоджень повідомлень при їх передачі по зашумлених каналах. Для цього передавальний пристрій створює деяке число, зване контрольною сумою, яке є функцією повідомлення, і додає його до цього повідомлення. Приймальний пристрій, використовуючи той самий алгоритм, розраховує контрольну суму прийнятого повідомлення і порівнює її з переданим значенням [1].

Вдалим різновидом контрольної суми є CRC. Про нього нагадують нам неприємні повідомлення від RAR, ZIP та інших архіваторів, коли файл виявляється пошкодженим в результаті неякісного з'єднання або пошкодження диска.

CRC (Cyclic Redundancy Code – "Циклічний надлишковий код") – це значення, яке обчислюється для деякого блоку даних, наприклад, для кожного файлу під час архівації. При розгортанні файлу архіватор порівнює це значення із знов обчисленим CRC розпакованого файлу. Якщо вони збігаються, то існує дуже велика вірогідність того, що цей новий файл вийшов ідентичним початковому. При використанні CRC32 вірогідність пропустити зміну даних складає всього $1/2^{32}$.

Стаття присвячена опису CRC і реалізації табличних алгоритмів їх обчислення. Велика частина літератури, що описує CRC взагалі, і особливо табличні їх різновиди, досить складна і запутана. Стаття переслідує мету дати простий і в той же час досить точний опис CRC з прикладами програмної реалізації алгоритмів.

Як робляться ці обчислення? Основна ідея полягає в тому, щоб представити передаваний файл як один величезний рядок бітів (величезне двійкове число), і поділити його на деяке число (твірний поліном); залишок, що утворився в результаті, і є CRC. Залишок формуватиметься завжди (правда, інколи він може виявитися рівним нулю), а інколи він всього лише на один біт менше дільника. Цей залишок і буде використаний джерелом (передавачем) як контрольна сума. Саме повідомлення при розрахунках CRC не змінюється, лише додається в його кінці контрольна сума у вигляді залишку:

<початкове незмінене повідомлення> <контрольна сума>.

Отримавши повідомлення, приймач може перевірити його цілісність, виконуючи аналогічну дію і порівнюючи отриманий залишок з "контрольною сумою" (переданим залишком) [2].

Алгоритми обчислення CRC. Відома загальна схема реального (тобто прямого, порозрядного) алгоритму обчислення CRC. Дії джерела наступні.

1. Виходячи з критеріїв достовірності, обрати твірний поліном P, в результаті автоматично стає відомою його степінь N.

2. Додати до початкової двійкової послідовності N нульових бітів. Це додавання робиться для гарантованої обробки всіх бітів початкової послідовності.

3. Виконати ділення доповненого N нулями початкового рядка S на поліном P по правилах поліноміальної CRC-арифметики. Запам'ятати залишок, який і буде являти собою CRC.

4. Сформувати остаточне повідомлення, яке складатиметься з двох частин: власне повідомлення і додане в його кінець значення CRC.

Відмітимо, що довжина значення CRC, що додається до початкової послідовності, має дорівнювати степені полінома, навіть якщо CRC має провідні нулі. Це дуже важливий момент, розуміння якого є ключем до розуміння суті процесів, що відбуваються на стороні приймача при здобутті і визначенні цілісності початкового повідомлення.

Дії алгоритму для приймача прості – виконати ділення отриманої послідовності на поліном. При цьому для виконання ділення немає необхідності доповнювати початкову послідовність нулями, тим більше що на практиці дотримання цієї умови вкрай незручно. Приймач просто виконує CRC-ділення отриманого початкового рядка (доповненого в кінці вихідним значенням CRC) на поліном і аналізує залишок. Якщо залишок від цього ділення нульовий, то початкова послідовність не була спотворена під час передачі. Інакше існує дуже велика вірогідність порушення цілісності початкової послідовності, і потрібно приймати додаткові заходи по з'ясуванню і виправленню ситуації. Одним з таких заходів может бути спроба відновлення потрібного значення CRC.

Описаний вище алгоритм обчислення значення CRC називається прямим і найчастіше реалізується апаратно. Проте, має сенс скласти приклад програми, що реалізує його. Хоча ефективність цієї програми не дуже висока, у неї є дві учбових мети:

- показати у вигляді програмної реалізації суть алгоритму обчислення CRC і самого CRC-ділення;
- підготуватися до розуміння досконаліших алгоритмів розрахунку CRC, до яких відноситься, зокрема, табличний алгоритм, що розглядається нижче.

Для комп'ютерної реалізації алгоритмів обчислення CRC зручно обирати поліноми зі степенями, кратними 8 (тобто розмірності регістрів) – 8, 16, 24, 32 або навіть 64. В цьому випадку можна підібрати команди з системи команд мікропроцесора, що найбільш оптимально реалізовують алгоритми обчислення CRC. В якості полінома оберемо один з рекомендованих поліномів – 4003h. Друге важливе зауваження – степень полінома визначає розмірність регістра, що використовується в алгоритмі. При цьому вважається, що старший (завжди одиничний) біт полінома знаходиться одразу за лівою межею регістра. За цих умов програма реалізації прямого алгоритму обчислення CRC функціонує наступним чином (рис. 1).

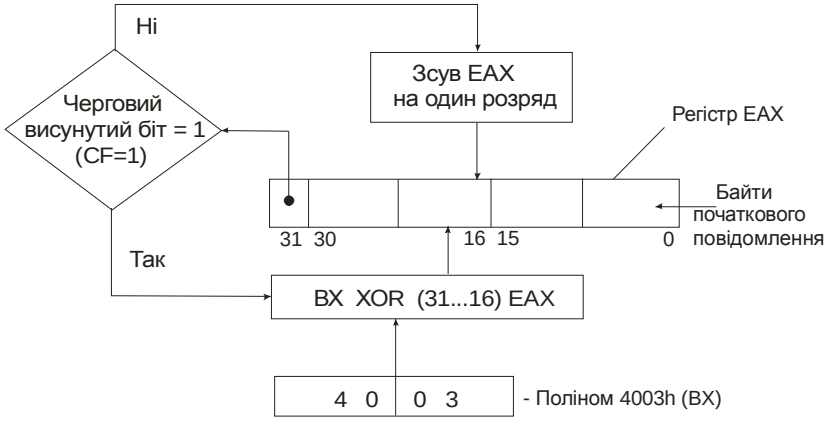

Рис. 1. Схема обчислення значення CRC прямим алгоритмом

У регістр побітно всуваються біти початкового рядка. Це відбувається до тих пір, поки при черговому зсуві зліва з'явиться одиничний біт. В цьому випадку весь вміст регістра підлягає операції XOR зі значенням полінома без старшого біта. Далі процес зсуву і аналізу біта, що висувається, продовжується до тих пір, поки не буде висунутий черговий одиничний біт, внаслідок чого знову між регістром і поліномом виконується операція XOR, і т.д. Після того, як останній біт опиниться в регістрі, в нього ще всувається кількість нульових бітів, рівна степені полінома. Цим досягається участь всіх бітів початкового бітового рядка у формуванні значення CRC. В результаті в регістрі залишається значення CRC, яке необхідно додати до початкового рядка і передати приймачу.

```
; CRC_01.asm – програма демонстрації прямого алгоритму обчислення CRC (сторона-джерело).
        .data
        ; початкова бітова послідовність в символах
        bit_string db "6476c8"
        len_bit_string=$-bit_string
        adr_bit_string dd bit_string
        polinom dw 4003h
        .code
        lds si, adr_bit_string
        mov cx, 1en_bit_string
        mov bx, polinom
        shl ebx, 16 ; підготуємо polinom до XOR з EAX
        m1:   push ex                   ; вкладені цикли
        mov cx, 8
        lodsb
        m2:   shl eax, 1
        jnc m3       ; старші розряди не рівні – виконуємо зсув (частка нас не цікавить),
        ; якщо старші розряди рівні – виконуємо XOR:
        xor eax, ebx        ; eax (31..16) XOR polinom
        m3:   loop m2
        pop cx
        loop m1
        ; всуваємо нульові біти числом N
        mov cl, 24+1  ; N=16 (степінь поліному) + 8 (працюємо в eax)+1 (для loop)
        m4:   shl eax, 1
        jnc m5 ; старші розряди не рівні – виконуємо зсув (частка нас не цікавить)
        ; якщо старші розряди рівні – виконуємо XOR:
        xor eax, ebx        ; eax (31..16) XOR polinom
        m5:   loop. m4
        … … …
```

В результаті обчислення CRC символьної послідовності "6476c8" отримаємо CRC = 35dah.

Для того, щоб змоделювати дії на стороні приймача, можна використати ту ж саму програму зі зміненими початковими даними – до рядка *bit_string* додаємо обчислене значення CRC. Після цього під відлагоджувальником спостерігаємо за процесом CRC-ділення, причому контролюємо залишок від ділення. У певний момент побачимо, що він став нульовим – це свідчить про те, що отримана послідовність не була змінена. Для експерименту можна змінити значення одного або більше бітів вхідної послідовності і поглянути, що вийде.

```
; CRC_02.asm – програма демонстрації прямого алгоритму обчислення CRC (сторона-приймач).
        .data
        bit_string    db     "6476c8",35h, 0dah ; вхідна бітова послідовність в символах
        len_bit_string=$-bit_string
        adr_bit_string              dd     bit_string
        polinom       dw     4003h
        .code
        main:
        ; див. попередню програму
        exit: ; вихід з програми
        … … …
```

Очевидний недолік прямого методу – велика кількість операцій зсуву, операцій виключаючого АБО (XOR) та операцій умовного переходу, які виконуються для кожного біта вхідного повідомлення. Тому на практиці використовується інший спосіб розрахунку CRC, званий табличним.

Табличні алгоритми обчислення CRC. Для того, щоб краще проілюструвати сутність табличного алгоритму обчислення CRC, звернемося знову до прямого методу, точніше до тієї схеми його обчислення (рис. 1), яка була реалізована в наведеній вище програмі.

З цієї схеми видно, що для поточного вмісту старшої половини регістра EAX можна прогнозувати, як змінюватиметься вміст його бітів по ходу їх зсуву. Для цього достатньо проаналізувати його біти, починаючи з найстаршого. Припустимо, що старші 8 біт EAX рівні a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0. При наступному зсуві (див. рис. 1) прямий алгоритм визначає, чи буде проведена операція XOR операнда з поліномом b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0 у BX (a_7=1) чи ні (a_7=0). Якщо висунутий біт дорівнював 1, то колишній вміст старшої половини регістра EAX буде підданий операції XOR з відповідними бітами полінома. У зворотному випадку, якщо висунутий біт дорівнював 0, значення бітів будуть не змінені, а просто зсунуті вліво на один розряд. В принципі, маючи велике бажання, можна розрахувати заздалегідь, яким буде вміст *k*-го біта в *k*-й ітерації зсуву. Наприклад, значення нового старшого біта, що визначає дії алгоритму в наступній ітерації, можна розрахувати по вмісту двох старших бітів старшого байта вхідного операнда – a_6 XOR a_7 AND b_7, де b_7 – старший біт полінома (завжди рівний одиниці).

Розглянемо наступну схему (рис. 2).

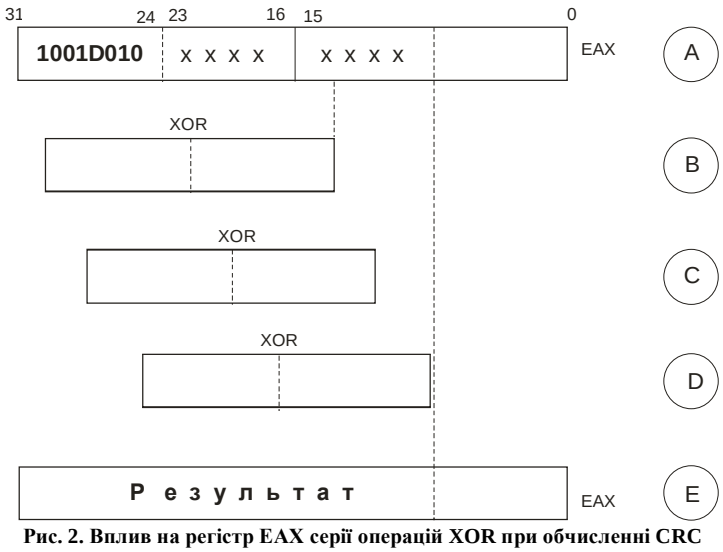

Рис. 2. Вплив на регістр EAX серії операцій XOR при обчисленні CRC

З вище викладеного виходить, що якщо взяти для розгляду старший байт операнда, то по його вмісту можна однозначно передбачити, скільки операцій XOR і коли буде виконано. Позначимо старшу половину регістра EAX як змінну A, а операнди зі значеннями полінома, що об'єднуються з A при одиничному стані чергового біта, що висувається зліва, позначимо відповідно як B, C, D (пам'ятаймо, що B = C = D). Тоді формування результату E можна представити формулою:

E- ((A [зсуви| XOR B) [зсуви |XOR C)|зсуви | XOR D

Тут | зсуви | є значенням від 0 до 7 і визначаються поточним вмістом старшого байта операнда

(регістра EAX). Завдяки асоціативній властивості операції XOR той самий результат можна отримати, якщо заздалегідь виконати операцію XOR над поліномами B, C, D з відповідними значеннями зсувів, а потім результат об'єднати по XOR з A:

F: = 1 зсуви| XOR (B |зсуви| XOR C |зсуви| XOR D) E: = A XOR F

З цих міркувань виходять важливі висновки:

- величина F є абсолютно точно передбаченою по вмісту старшого байта операнда;
- якщо величина F визначається вмістом старшого байта операнда і власне значенням полінома, то існує всього 256 можливих значень цієї величини (по кількості значень, які можна представити беззнаковим байтом);
- виходячи з перших двох положень, величина F не залежить від значення операнда і може бути розрахована заздалегідь, при цьому результати її розрахунків можна звести в таблицю.

Тепер з'ясувалося, на чому заснована назва табличного алгоритму розрахунку CRC. Наведемо загальний його опис (рис. 3). Основою для міркувань, як і раніше, є програма прямого обчислення значення CRC і відповідна схема (рис. 1).

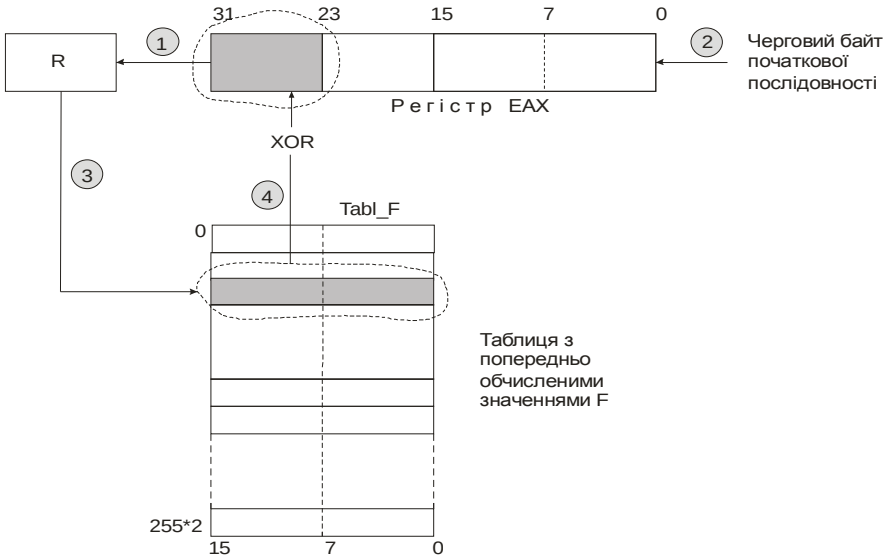

Рис. 3. Загальна схема табличного алгоритму

На схемі, показаній на рисунку, цифрами позначена послідовність кроків табличного алгоритму. Кроки 1 і 2 виконуються одночасно і означають, що старший байт з регістра EAX висувається в змінну R, а молодший байт цього регістра заповнюється черговим байтом вхідної послідовності. Значення змінної R використовується на кроці 3 як індекс в таблиці TABL_F для витягання 16-бітового значення, яке на кроці 4 буде об'єднано операцією XOR з вмістом старших 16 бітів регістра EAX. Таким чином, на відміну від прямого алгоритму, процес перетворення зростає до рівня байтів і містить три операції: зсуву, доступу до таблиці для витягання потрібного значення та операції XOR витягнутого значення з вмістом старшої половини EAX. Після закінчення процесу в старшій половині EAX міститиметься значення CRC. Повідомлення, як і раніше, має бути вирівняним, тобто доповненим кількістю бітів, рівною степені полінома, для даного випадку – 16. Для практичних застосувань це вкрай незручно, і вирішення цієї проблеми показано нижче. Поки ж розглянемо програму обчислення вмісту таблиці на основі полінома 1021h степеню 16.

```
; CRC_03.asm – програма обчислення вмісту таблиці на основі полінома 1021h степеню 16.
      .data
      tabl_16 dw 256 dup (0)          ; CRC-таблиця
      len_tabl_16=$-tabl_16
      adr_tabl_16 dd tabl_16
      polinom dw 1021h
      .code
      les di, adr_tabl_16
      add di, len_tabl_16-2
      std                       ; рухаємося по таблиці в зворотньому напрямку
      mov ex, 255
      mov bx, polinom
      m1:   xor ax, ax
      mov ah, cl                ; індекс в таблиці для обчислення CRC
      push ex                         ; вкладені цикли
      mov ex, 8
      m2:   shl ax, 1
      jnc m3        ; старші розряди не рівні – виконуємо зсув (частка нас не цікавить)
```

```
; якщо старші розряди рівні – виконуємо XOR:
xor ax, bx                ; ax XOR polinom
m3:   loop m2
pop cx
stosw
loop m1
… … …
```

В результаті роботи цієї програми область пам'яті tabl_16 ініціалізується таблицею значень, які можуть бути використані в схемі обчислення значення CRC вхідної послідовності (рис. 3).

Прямий табличний алгоритм CRC16. Необхідність доповнення вхідного рядка завершуючими нульовими байтами представляє велику незручність при реалізації загальної схеми табличного алгоритму. Тому на практиці використовують іншу схему обчислення CRC (рис. 4), що не вимагає завершення вхідного повідомлення нульовими байтами.

У цій схемі черговий байт вхідного рядка об'єднується операцією XOR зі старшим байтом регістра, висунутим з лівого боку цього регістра. Набуте в результаті об'єднання значення використовується як індекс для доступу до CRC-таблиці. Витягуване з CRC-таблиці значення об'єднується операцією XOR з вмістом регістра. Описаний алгоритм називають прямим табличним алгоритмом [3].

На схемі обчислень CRC з використанням прямого табличного алгоритму цифрами позначена послідовність кроків обчислення CRC.

1. Висунення старшого байта регістра AX в байтову комірку.
2. Виконання операції XOR над висунутим на кроці 1 в байтову комірку старшим байтом регістра AX і черговим байтом вхідного рядка.
3. Отримане на кроці 2 значення використовується як індекс для доступу до елементу CRC-таблиці.
4. Значення, витягнуте з CRC-таблиці, об'єднується по XOR зі значенням в регістрі AX.
5. Результат виконання на кроці 4 операції XOR поміщається назад в регістр AX.

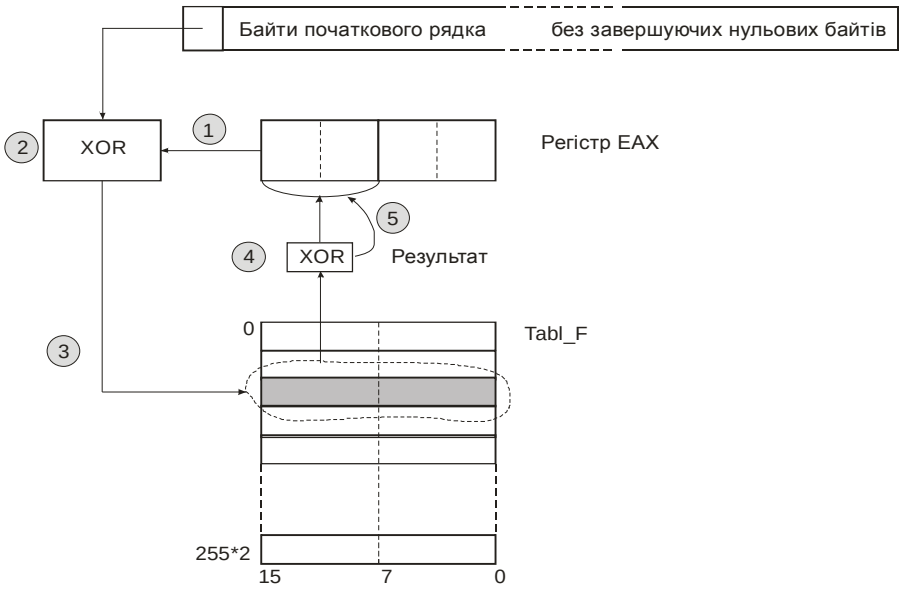

Рис. 4. Схема обчислень CRC з використанням прямого табличного алгоритму

Після обробки останнього байта вхідного рядка регістр AX містить значення CRC. Програма обчислення CRC з використанням прямого табличного алгоритму наведена нижче.

```
; CRC_04.asm – програма обчислення CRC по прямому табличному алгоритму.
      .data
      ; вхідна бітова послідовність в символах
      bit_string    db     "6476c8"
      len_bit_string=$-bit_string
      adr_bit_string            dd      bit_string
      tabl_16       dw      256 dup (0)             ; CRC-таблиця
      len_tabl_16=$-tabl_16
      adr_tabl_16   dd      tabl_16
      polinom       dw      1021h
      .code
      ; -------------розраховуємо CRC-таблицю----------------
      les di, adr_tabl_16
      add di, len_tabl_16-2
      std                               ; рухаємося по таблиці в зворотньому напрямку
```

```
mov ex, 255
mov bx, polinom
m1:    xor ax, ax
mov ah, cl                ; індекс в таблиці для обчислення CRC
push cx                   ; вкладені цикли
mov ex, 8
m2:    shi ax, l
jnc m3       ; старші розряди не рівні – виконуємо зсув (частка нас не цікавить)
; якщо старші розряди рівні – виконуємо XOR:
xor ax, bx                ; ax XOR polinom
m3:    loop m2
pop cx
stosw
loop ml
; ------------закінчили розрахунок CRC-таблиці------------
xor ax, ax
xor bx, bx
lds si, adr_bit_string
mov cx, len_bit_string
m4:    mov bl, ah
shl ax, 8
xor bl, [si].
xor ax, tabl_16 [bx].
inc si
1oop m4
.........
```

Розглядом цього алгоритму проблему обчислення CRC табличним методом можна і закінчити. Всі існуючі алгоритми обчислення CRC є, по суті, різними модифікаціями описаного вище табличного алгоритму. Ці модифікації переслідують різні цілі, перерахуємо деякі з них:

- перехід від циклу по всіх бітах до циклу по великих порціях даних – байтах, словах і т.д.;
- підвищення розрядності твірного полінома;
- віддзеркалення особливостей функціонування апаратури передачі даних (наявність цієї мети обумовлена тим, що мікросхеми, котрі підтримують роботу послідовного порта комп'ютера, передачу байта починають з його молодших бітів, тому відповідні алгоритми розрахунку CRC, що враховують цю особливість, називаються *дзеркальними*).

Алгоритми обчислення CRC отримали своє закріплення в деяких стандартах. Перерахуємо відмітні особливості основних алгоритмів обчислення CRC. Отже, основні алгоритми обчислення CRC розрізняються:

- за значенням і степенем полінома, при цьому значення полінома, зазвичай, вказується без провідної одиниці в старшому розряді;
- по початковому вмісту регістра, в якому формується значення CRC;
- по тому, чи проводиться обернення бітів кожного байта перед його обробкою;
- за способом проходу байтів через регістр – спосіб може бути непрямим або прямим;
- по тому, чи проводиться обернення кінцевого результату;
- по кінцевому значенню, з яким проводиться об'єднання по XOR результату обчислення CRC.

Після появи 32-розрядних мікропроцесорів найбільшою популярністю стали користуватися 32-розрядні алгоритми обчислення CRC. У різних джерелах розглядають два типи таких алгоритмів – прямий і дзеркальний.

Висновки

Для виявлення і опрацювання пошкоджень повідомлень при їх передачі по зашумлених каналах використовують CRC як різновид контрольної суми. Передаваний файл представляють як одне величезне двійкове число і ділять його на твірний поліном, а залишок, що утворився в результаті, додають до повідомлення в якості CRC. Обчислення значення CRC прямим алгоритмом вимагає великої кількості операцій зсуву, операцій XOR та умовного переходу, які виконуються для кожного біта вхідного повідомлення. На практиці використовують табличні способи розрахунку CRC, з яких поширення отримали прямий і дзеркальний табличні алгоритми.

Література

1. Жураковський Ю.П. Теорія інформації та кодування: [підручник] / Жураковський Ю.П., Полторак В.П. – К.: Вища школа, 2001. – 255 с.

2. Хаммел Р.Л. Последовательная передача данных: Руководство для программиста / Р.Л. Хаммел; [пер. с англ]. – М.: Мир, 1996.

3. Уильямс Р. Н. Элементарное руководство по CRC-алгоритмам обнаружения ошибок [Електронний ресурс]: Уільямс Р. Н. – Режим доступу: http://www.internode.net.au/clients/rocksoft/papers/crc_v3.htm.

Надійшла 6.5.2010 р.