Контрольная сумма

- **Циклический избыточный код** (CRC, Cyclic Redundancy Checksums) - алгоритм нахождения контрольной суммы, предназначенный для проверки целостности данных.
  - Методы обнаружения ошибок предназначены для выявления повреждений со общений при их передаче по зашумленным каналам (вносящих эти ошибки).
- **Основан на делении двоичных многочленов с остатком.**
- является алгоритмом хэширования,
  - отображает (хэширует) элементы большого набора данных в элемент меньшего размера.
    - Каждый отдельный элемент исходного набора данных отображается на один и только один элемент хэш-набора.
      - обратное не верно.

Контрольная сумма

- 2 требования для формирования надежной контрольной суммы:
  - **Ширина:**
    - Размер регистра для вычислений должен обеспечивать низкую вероятность ошибки (например, 32 байтный регистр обеспечивает вероятность ошибки $1/2^{32}$).
  - **Случайность:**
    - Необходим такой алгоритм расчета, когда каждый новый байт хешируемых данных может изменить любые случайные биты хеш-суммы.
- *Алгоритмы CRC, используемые на практике, удовлетворяют условию случайности и могут быть адаптированы для работы с различной шириной контрольной суммы.*

Контрольная сумма

- Алгоритм CRC основан на:
  1. Представлении сообщения виде двоичного числа
  2. Делении числа на другое фиксированное двоичное число
  3. Использовании остатка этого деления в качестве контрольной суммы.
- *Получив сообщение, приемник может выполнить аналогичное действие и сравнить полученный остаток с "контрольной суммой" (переданным остатком).*

Контрольная сумма. Полиномиальная арифметика.

- В основе вычисления контрольных сум лежит т.н. полиномиальная арифметика.
- В полиномиальной арифметике связи между коэффициентами не установлены, коэффициенты при каждом члене считаются независимыми.
- **Как правило используется полиномиальная арифметика по модулю 2**.
  - коэффициенты складываются по модулю 2 без переноса
  - коэффициенты могут иметь значения 0 или 1, перенос не учитывается.

Контрольная сумма

- CRC алгоритмы основаны на полиномиальных вычислениях,
- алгоритмы CRC отличаются тем, какой т.н. двоичный полином используется.

- Например, десятичное число 23 в шестнадцатеричной системе счисления имеет вид 17, а в двоичном – 10111, что совпадает с полиномом:

$$\mathbf{1} * \boldsymbol{x}^{\mathbf{4}} + \mathbf{0} * \boldsymbol{x}^{\mathbf{3}} + \mathbf{1} * \boldsymbol{x}^{\mathbf{2}} + \mathbf{1} * \boldsymbol{x}^{\mathbf{1}} + \mathbf{1} * \boldsymbol{x}^{\mathbf{0}}$$

- или, упрощенно:

$$\boldsymbol{x}^{\mathbf{4}} + \boldsymbol{x}^{\mathbf{2}} + \boldsymbol{x}^{\mathbf{1}} + \boldsymbol{x}^{\mathbf{0}}$$

- Предположим, что мы хотим перемножить, на пример, 1101 и 1011. Это можно выполнить, как умножение полиномов:

$$\left(\boldsymbol{x}^{\mathbf{3}} + \boldsymbol{x}^{\mathbf{2}} + \boldsymbol{x}^{\mathbf{0}}\right)\left(\boldsymbol{x}^{\mathbf{3}} + \boldsymbol{x}^{\mathbf{1}} + \boldsymbol{x}^{\mathbf{0}}\right) =$$

$$= \left(\boldsymbol{x}^{\mathbf{6}} + \boldsymbol{x}^{\mathbf{4}} + \boldsymbol{x}^{\mathbf{3}} + \boldsymbol{x}^{\mathbf{5}} + \boldsymbol{x}^{\mathbf{3}} + \boldsymbol{x}^{\mathbf{2}} + \boldsymbol{x}^{\mathbf{3}} + \boldsymbol{x}^{\mathbf{1}} + \boldsymbol{x}^{\mathbf{0}}\right) =$$

$$= \boldsymbol{x}^{\mathbf{6}} + \boldsymbol{x}^{\mathbf{5}} + \boldsymbol{x}^{\mathbf{4}} + \mathbf{3} * \boldsymbol{x}^{\mathbf{3}} + \boldsymbol{x}^{\mathbf{2}} + \boldsymbol{x}^{\mathbf{1}} + \boldsymbol{x}^{\mathbf{0}}$$

Контрольная сумма

- Если принять. Что X равен 2, придется выполнить перенос бита от терма $\mathbf{3} * \boldsymbol{x}^{\mathbf{3}}$.
- То есть для двоичного полинома $\mathbf{3} * \boldsymbol{x}^{\mathbf{3}} = \boldsymbol{x}^{\mathbf{4}} + \boldsymbol{x}^{\mathbf{3}}$
- В результате получим:

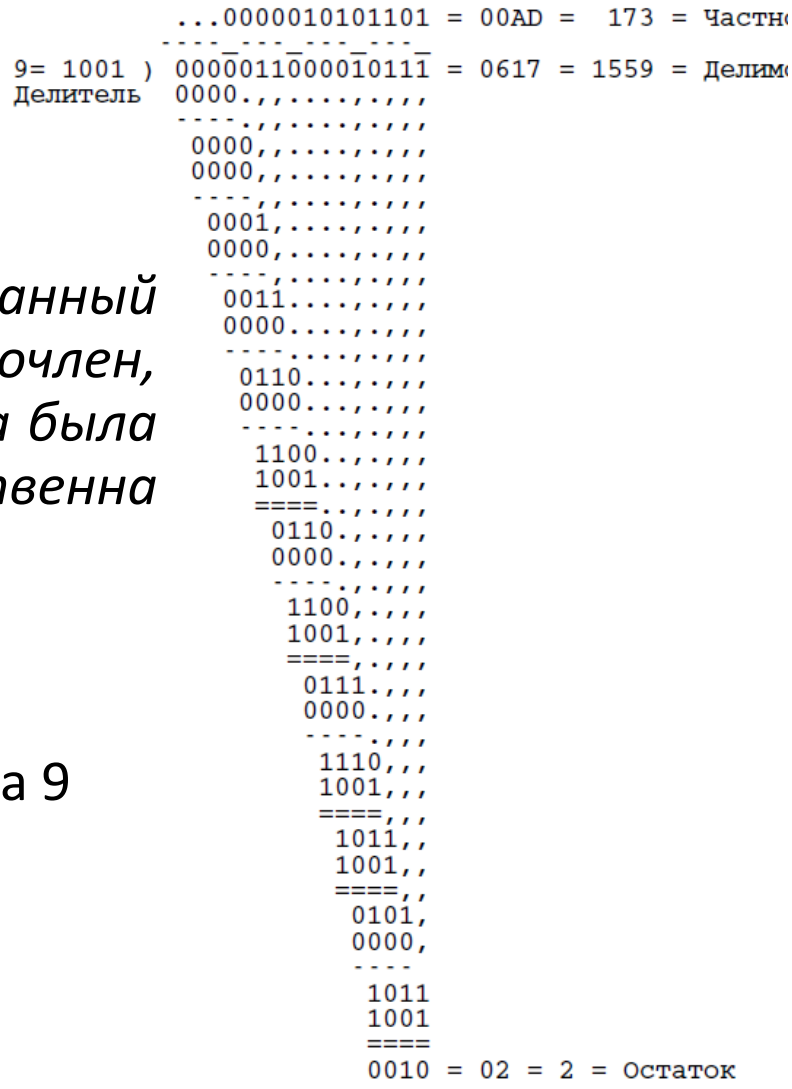
$$\boldsymbol{x}^{\mathbf{6}} + \boldsymbol{x}^{\mathbf{5}} + \boldsymbol{x}^{\mathbf{4}} + \mathbf{3} * \boldsymbol{x}^{\mathbf{3}} + \boldsymbol{x}^{\mathbf{2}} + \boldsymbol{x}^{\mathbf{1}} + \boldsymbol{x}^{\mathbf{0}} =$$
$$= \boldsymbol{x}^{\mathbf{6}} + \boldsymbol{x}^{\mathbf{5}} + \mathbf{2} * \boldsymbol{x}^{\mathbf{4}} + \boldsymbol{x}^{\mathbf{3}} + \boldsymbol{x}^{\mathbf{2}} + \boldsymbol{x}^{\mathbf{1}} + \boldsymbol{x}^{\mathbf{0}} =$$
$$= \boldsymbol{x}^{\mathbf{6}} + \mathbf{2} * \boldsymbol{x}^{\mathbf{5}} + \boldsymbol{x}^{\mathbf{3}} + \boldsymbol{x}^{\mathbf{2}} + \boldsymbol{x}^{\mathbf{1}} + \boldsymbol{x}^{\mathbf{0}} =$$
$$= \mathbf{2}\boldsymbol{x}^{\mathbf{6}} + \boldsymbol{x}^{\mathbf{3}} + \boldsymbol{x}^{\mathbf{2}} + \boldsymbol{x}^{\mathbf{1}} + \boldsymbol{x}^{\mathbf{0}} =$$
$$= \boldsymbol{x}^{\mathbf{7}} + \boldsymbol{x}^{\mathbf{3}} + \boldsymbol{x}^{\mathbf{2}} + \boldsymbol{x}^{\mathbf{1}} + \boldsymbol{x}^{\mathbf{0}}$$

Данный перенос основан на том, что $\mathbf{3} * \boldsymbol{x}^{\mathbf{3}} = \boldsymbol{x}^{\mathbf{4}} + \boldsymbol{x}^{\mathbf{3}}$ для $\boldsymbol{x} = \mathbf{2}$

Если, "X" **не известен**, то нельзя выполнить перенос.

Контрольная сумма

- Пример: сообщение из 2 байт (6, 23)
  - 6, 23 в HEX 0x0167h,
  - 6, 23 в BIN 0000 0110 0001 0111b.
- ширина регистра CRC 1 байт,
- Делитель 1001 – *специально выбранный особым образом двоичный многочлен, такой чтобы полученная CRC сумма была как можно более случайна и единственна для каждого делимого.*
- Значение CRC – остаток деления 0000 0110 0001 0111 на 1001.
- *В примере* частное от деления 1559 на 9 равно 173 и 2 в остатке.



Контрольная сумма. арифметика CRC. Сложение и вычитание

- Сложение двух чисел в CRC арифметике аналогично обычному арифметическому действию, но с запретом переноса.
  - Суммирование представляет операцию исключающего или (XOR)
    - Например:

10011011
+11001010

01010001

  - Вычитание также по правилу XOR (правило зацикливания) (0-0=0, 0-1=1-0=1, 1-1=0)

10011011
-11001010

01010001

    - В арифметике CRC

1001 = 1010 + 0011

1001 = 1010 - 0011

Контрольная сумма. арифметика CRC. Умножение

- Умножение, считается суммой XOR значений первого сомножителя, сдвинутых в соответствии со значением второго сомножителя.

```
          1 1  0  1
         x
          1 0  1  1
          ----
          1 1  0  1
        1 1 0  1  .
      0 0 0 0  .  .
    1 1 0 1  .  .  .
    -------
    1 1 1 1  1  1  1
```

при суммировании используется CRC сложение

Контрольная сумма. арифметика CRC. Деление

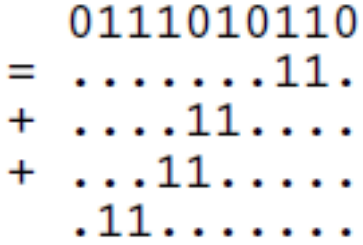
- Деление
- требуется знать, когда одно число превращается в другое.
- слабое понятие размерности:
- число $X \geq Y$, если позиция самого старшего единичного бита числа X больше или равна позиции самого старшего единичного бита числа Y.

```
                1100001010
              ---------------
10011 ) 11010110110000
        10011,,.,,,,....
        -----,,.,,,,....
         10011,.,,,,....
         10011,.,,,,....
         -----,.,,,,....
          00001.,,,,....
          00000.,,,,....
          -----.,,,,....
           00010,,,,....
           00000,,,,....
           -----,,,,....
            00101,,,,....
            00000,,,,....
            -----,,,,....
             01011....
             00000....
             -----....
              10110...
              10011...
              ------...
               01010..
               00000..
               ------..
                10100.
                10011.
                -----.
                 01110
                 00000
                 -----
                  1110 = Остаток
```

Контрольная сумма. арифметика CRC.Сдвиг

- Пусть число A получено умножением числа B, на C
  - Тогда число A может быть получено из нулевого числа нуля, применяя операцию XOR к числу B, сдвинутому последовательно на каждую позицию.
  - Например, если A равно 0111010110, а B – 11,
    - A из B может быть получено как:

0111010110
= . . . . . 11.
+ . . . . 11. . . .
+ . . . 11. . . . .
.11 . . . . . .



  - Если A бы было бы равно 0111010111, то не удалось составить его с помощью сдвигов числа 11,
    - В этом случае в CRC арифметике A не делится на B.
- CRC арифметика сводится к операции "Исключающее ИЛИ" некоторого значения при различных величинах сдвига.

Контрольная сумма. арифметика CRC. CRC деление при помощи сдвига

While (пока еще есть необработанные биты)

Сдвиг регистра на 1 бит влево

Помещение очередной бита из сообщения в 0 позицию регистра.

If (очередной бит == "1")

Регистр = Регистр XOR Полином

end if

End While

Теперь в регистре содержится остаток

Пример:

1101010
^101
====
111010
^101
===
10010
^101
===== получили 0, сдвиг на 2 бита
110
^101
===
11
Готово! CRC = 11

На практике
двигают входное сообщение влево,
а не полином вправо

Контрольная сумма. арифметика CRC. CRC деление при помощи сдвига

На практике
двигают входное сообщение влево,
а не полином вправо

Шаг 1. r = 1 // Отделили первый бит
data = 1101 010 // r=1, поэтому делаем XOR
^101
======
111 010
Шаг 2. r = 1
data = 11 010
^101
=====
10 010
Шаг 3. r = 1
data = 0 010
^101
====
0 110
Шаг 4. r = 0 // XOR не делаем
data = 110
Шаг 5. r = 1
data = 10
^101
==
11

Пример:

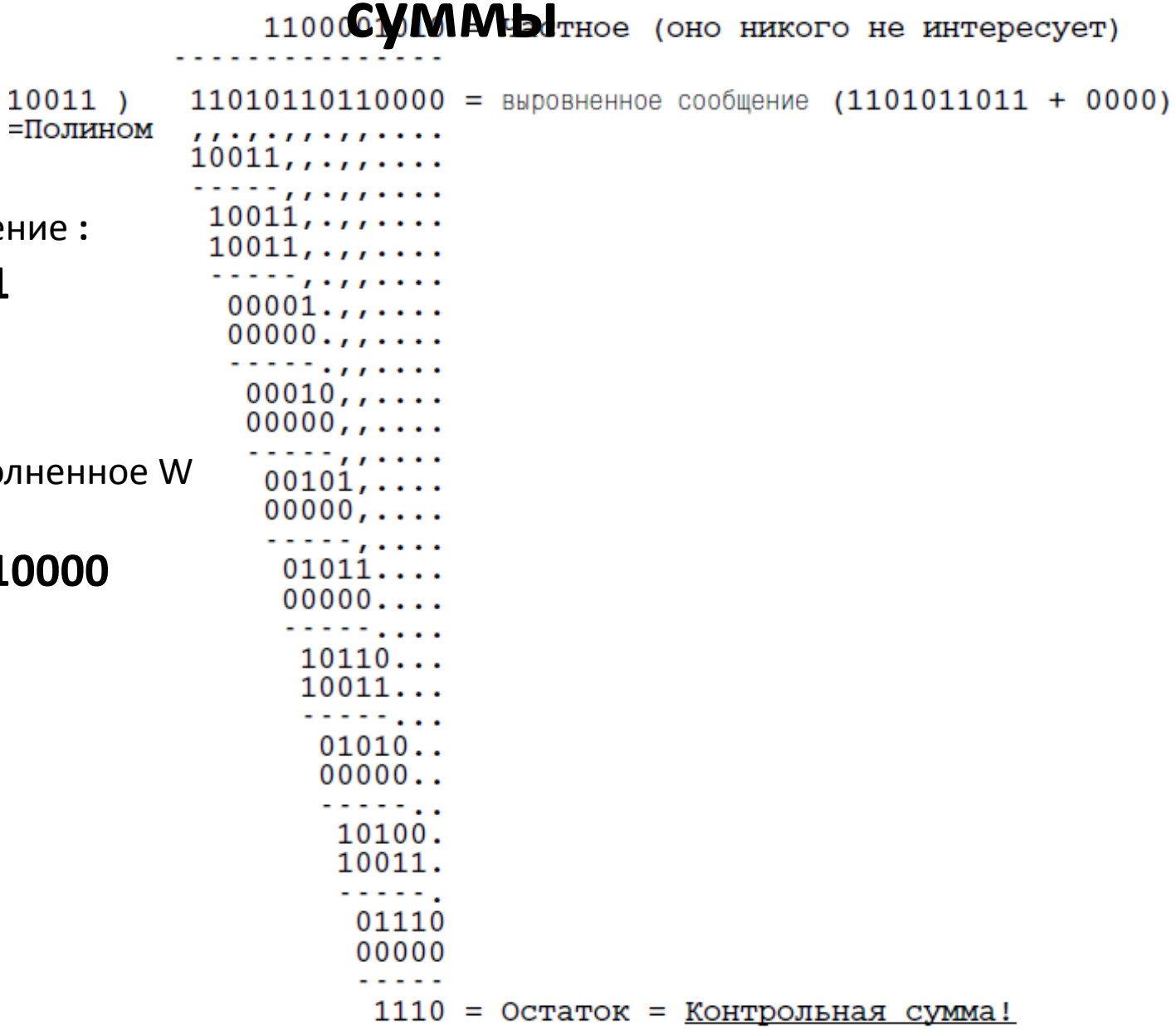
1 101 010
^101
====
111 010
^101
===
10010
^101
===== получили 0, сдвиг на 2 бита
110
^101
===
11
Готово! CRC = 11

Контрольная сумма. арифметика CRC. Алгоритм расчета CRC суммы

- Чтобы выполнить вычисление CRC, необходимо **выбрать** делитель **генераторный полином (generator polinomial)**.
  - Стандартные полиномы выбраны такими, чтобы вероятность ошибки была как можно меньше.
  - **Степень полинома W (**Width – ширина) (позиция самого старшего единичного бита)
    - Напр. Для 10011 степень равна 4 (не 5).
    - Вероятность ошибки 1/W, где W степень полинома CRC
      - Наиболее распространены CRC1, CRC4, CRC8, CRC16, CRC32,
- **Двоичное сообщение дополняется W нулевыми битами.**
- *Расчет CRC суммы производится CRC делением двоичного сообщения на генераторный полином или табличным методом.*
- Остаток деления является CRC суммой.

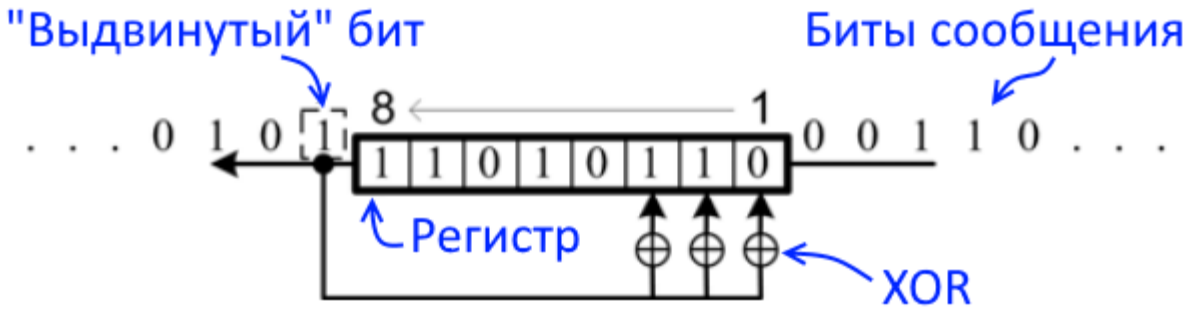
Контрольная сумма. арифметика CRC. Алгоритм расчета CRC суммы

- Исходное сообщение :
  - **1101011011**
- Полином :
  - **10011**
- Сообщение, дополненное W битами :
  - **11010110110000**



Контрольная сумма.
CRC алгоритм побитового сдвига

1. Создаётся массив, заполненный нулями, длина= степени полинома W.
2. Исходное сообщение дополняется W нулями в младших разрядах.
3. В младший разряд регистра заносится один старший бит сообщения, а из старшего разряда регистра выдвигается один бит.
4. Если выдвинутый бит равен "1", то производится операция XOR.
5. Если в сообщении ещё есть биты, переходим к шагу 3).
6. Остаток от деления после прохождения всех бит контрольная сумма CRC.



деление исходной последовательности битов на число (1)00000111, или многочлен x8 + x2 + x1 + x0.

Контрольная сумма.
CRC алгоритм табличного сдвига

- Остаток CRC является независимым от остатков делений других байт.
- Таким образом можно заранее записать остатки деления каждого возможного байта и т от 00h до FFh на заданный полином в таблицу.
    - Возможно также сразу записывать остатки деления двух, четырех и т.д. байт
- **Алгоритм табличного деления.**
- 1. Начальная установка регистра CRC, значением 0FF…Fh.
    - Размер регистра = размеру полинома
- Последовательный выбор байт исходной последовательности и выбор соответствующего остатка деления в таблице CRC.
- Объединение остатков CRC по XOR в регистре CRC.

- ***«Зеркальный табличный алгоритм» -***
  В случае «зеркального приема» (100011->110001) в алгоритме сдвиг байт влево надо заменить сдвигом вправо

Контрольная сумма. арифметика CRC. Алгоритм расчета CRC суммы

- Приемник может сделать одно из равноценных действий:
  - **1. Выделить сообщение, вычислить для него контрольную сумму**
    - При этом дополнить сообщение W битами
    - Сравнить ее с переданной.
  - **2. Вычислить контрольную сумму для всего переданного сообщения (без добавления нулей), и посмотреть, получится ли в результате нулевой остаток.**
    - последние W бит сообщения – остаток от деления дополненного нулями исходного сообщения на выбранный полином
  - **Оба способа эквивалентны.**
    - *CRC сложение равносильно CRC вычитанию, поэтому прибавление остатка дополняет значение сообщения до следующего полного произведения.*

Контрольная сумма. арифметика CRC. Виды полиномов CRC

CRC-1	$x + 1$ - бит чётности
CRC-4-ITU	$x^4 + x + 1$
CRC-8-CCITT	$x^8 + x^2 + x + 1$ (MMC, SD)
CRC-8	$x^8 + x^7 + x^6 + x^4 + x^2 + 1$
CRC-15-CAN	$x^{15} + x^{14} + x^{10} + x^8 + x^7 + x^4 + x^3 + 1$
CRC-16 (CRC-16 IBM и CRC-16-ANSI)	$x^{16} + x^{15} + x^2 + 1$ (Modbus, USB, ANSI X3.28)
CRC-16-CCITT	$x^{16} + x^{12} + x^5 + 1$ (Bluetooth, SD)
CRC-32-IEEE 802.3	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ $(MPEG - 2, POSIX, Ethernet)$
CRC-64-ISO	$x^{64} + x^4 + x^3 + x + 1$ (ISO 3309)

Контрольная сумма. арифметика CRC. Алгоритм расчета CRC8

- **Из сообщения берётся первое слово**
  - Бит (CRC-1), байт (CRC-8), 2 байта CRC16 и т.д.
- **Если старший бит «1»,**
  - слово сдвигается влево на один разряд.
  - выполняется операция XOR с порождающим полиномом.
    - После сдвига теряется старый старший бит, а младший бит освобождается — его значение устанавливается равным нулю.
- **Если старший бит в«0»,**
  - слово сдвигается влево на один разряд
    - XOR не выполняется.
- *На место младшего бита загружается очередной бит из сообщения.*
  - операция повторяется, пока не загрузится последний бит сообщения.
- **После прохождения всего сообщения, слово- остаток-является CRC**.

Схема формирования CRC-8. Порождающий многочлен g(x) = x8+x5+x4+1