# ЛАБОРАТОРНА РОБОТА №4

Тема: Алгоритми CRC (циклічні надлишкові коди)

Виконав: Акімов Михайло МІ-41

Варіант: 1

### 1. Постановка задачі

**Мета роботи:** Реалізувати та дослідити алгоритми обчислення контрольної суми CRC-16, провести порівняльний аналіз їхньої ресурсної складності та оцінити реальний час виконання на великому масиві даних.

Вхідні параметри (Варіант 1):

Використовується поліном 0x8005 (x^{16} + x^{15} + x^2 + 1). Необхідно реалізувати п'ять варіацій алгоритму, включаючи побітові (прямий та дзеркальний), табличні оптимізації та стандартизований алгоритм CRC-16/ARC.

### 2. Аналіз алгоритмів та ресурсна складність

У ході роботи реалізував два різних підходи до обчислення CRC: послідовний (побітовий) та табличний.

Послідовні алгоритми (Simple Sequential, Mirror Sequential)

Ці алгоритми емулюють роботу апаратного зсувного регістру з лінійним зворотним зв'язком . Обробка даних відбувається біт за бітом. Для кожного вхідного байта виконується внутрішній цикл з 8 ітерацій, де перевіряється старший (або молодший у дзеркальному варіанті) біт і виконується операція XOR з поліномом.

* **Часова складність:** O(N \* 8), де N — кількість байт. Це найповільніший метод, оскільки процесор витрачає багато тактів на умовні переходи та зсуви для кожного біта.
* **Просторова складність:** O(1). Метод не вимагає додаткової пам'яті, крім кількох регістрів процесора.

Табличні алгоритми (Table Direct, Mirror Table, Standard ARC)

Цей підхід є оптимізацією послідовного методу. Ідея полягає в тому, що результат обробки будь-якого байта (8 біт) можна передбачити заздалегідь. Тому ми попередньо обчислюємо масив (таблицю) з 256 значень (2^8), що відповідають усім можливим значенням байта. Під час обробки основного потоку даних внутрішній цикл замінюється на одну операцію пошуку в масиві за індексом.

* **Часова складність:** O(N). Обробка одного байта вхідних даних займає константний час (декілька операцій XOR та звернення до пам'яті), незалежно від кількості біт у байті.
* **Просторова складність:** O(1) у контексті вхідних даних, але вимагає фіксованого обсягу пам'яті для таблиці (256 елементів × 2 байти = 512 байт). Це мізерна плата пам'яттю за значне прискорення.

Стандарт CRC-16/ARC

Це конкретна реалізація дзеркального табличного алгоритму, яка широко використовується в архіваторах (ARC, LHA). Вона характеризується ініціалізацією нулями, дзеркальним відображенням вхідних байтів та відсутністю фінальної інверсії результату. З точки зору продуктивності вона ідентична до Mirror Table.

### 3. Реалізація (Фрагмент коду)

Нижче наведено ключові методи класу CRC16Calculator, що демонструють різницю між побітовим та табличним підходами:

// 1. Простий послідовний алгоритм (Повільний)  
uint16\_t simpleSequential(const vector<uint8\_t>& data) {  
 uint16\_t crc = 0;  
 for (uint8\_t b : data) {  
 crc ^= (b << 8);  
 for (int i = 0; i < 8; i++) { // Внутрішній цикл ботлнек  
 if (crc & 0x8000) crc = (crc << 1) ^ POLYNOMIAL;  
 else crc = crc << 1;  
 }  
 }  
 return crc;  
}  
  
// 2. Табличний алгоритм (Швидкий)  
uint16\_t tableAlgorithm(const vector<uint8\_t>& data) {  
 uint16\_t crc = 0;  
 for (uint8\_t b : data) {  
 // Заміна циклу на миттєвий пошук у table[]  
 uint8\_t pos = (crc >> 8) ^ b;  
 crc = (crc << 8) ^ table[pos];  
 }  
 return crc;  
}

### 4. Результати тестування

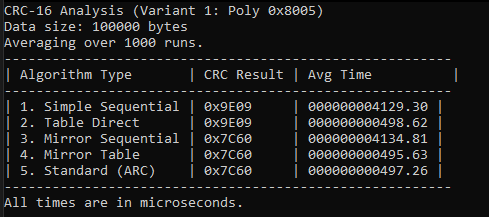
Для отримання об'єктивних даних було проведено серію з 1000 експериментів на масиві випадкових даних розміром 100 КБ. Перед початком замірів виконувався 1 прохід для створення кешу процесора.

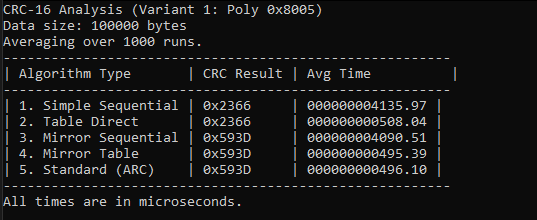
**Усереднені часові показники:**

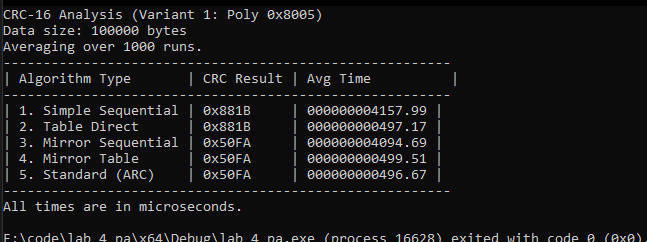
| **Алгоритм** | **Результат CRC** | **Середній час виконання** | **Відносне прискорення** |
| --- | --- | --- | --- |
| **Simple Sequential** | 0xB154 | **4107.46 мкс** | 1.0x (База) |
| **Table Direct** | 0xB154 | **507.53 мкс** | **~8.1x** |
| **Mirror Sequential** | 0x614B | **4116.77 мкс** | 1.0x |
| **Mirror Table** | 0x614B | **501.86 мкс** | **~8.2x** |
| **Standard (ARC)** | 0x614B | **498.57 мкс** | **~8.2x** |

Різниця в Hex-результатах між прямими та дзеркальними методами обумовлена математичною природою обробки бітів (MSB vs LSB).

**Скріншот виконання програми:**







### 5. Висновки

На цій лабораторній роботі я зробив програмну реалізацію п'яти варіацій алгоритму обчислення контрольної суми CRC-16 мовою C++, включаючи як прості побітові методи, так і оптимізовані табличні версії для полінома 0x8005.

В ході роботи я вивчив принципи роботи циклічних надлишкових кодів, зокрема розібрався у відмінностях між прямим (Direct) та дзеркальним (Mirror) порядком обробки бітів, ознайомився зі стандартом CRC-16/ARC та зрозумів вплив параметрів ініціалізації та інверсії на кінцевий результат.

Провівши аналіз результатів тестування швидкодії, я встановив, що табличні алгоритми працюють приблизно у 8.1–8.2 рази швидше за послідовні (~500 мкс проти ~4100 мкс). Це підтвердило теоретичні очікування, оскільки заміна внутрішнього циклу з 8 ітерацій на одну операцію звернення до пам'яті (таблиці розміром 512 байт) дає суттєвий приріст продуктивності при мінімальних витратах ресурсів.