УДК 681.3.06

ИССЛЕДОВАНИЕ ПРОГРАММНЫХ РЕАЛИЗАЦИЙ ТАБЛИЧНОГО И МАТРИЧНОГО АЛГОРИТМОВ ВЫЧИСЛЕНИЯ КОНТРОЛЬНОЙ СУММЫ CRC32

Е.А. Мыцко, А.Н. Мальчуков

Томский политехнический университет
E-mail: 7evgen7@sibmail.com; jgs@tpu.ru

Приведено сравнение по времени вычисления и размеру исполняемых файлов программных реализаций табличного и матричных алгоритмов вычисления контрольной суммы CRC32, совместимой с контрольной суммой архиваторов PKZIP, WinZIP и протокола ETHERNET. Проведено полное исследование различных вариантов применения как с исходным буфером, так и с буфером, совместимым с реализацией на микроконтроллерах. Сделаны рекомендации относительно применения табличного и матричного алгоритмов.

Ключевые слова:
Контрольная сумма, циклический избыточный код, табличный алгоритм, матричный алгоритм, CRC32.

Key words:
Check sum, cyclic redundancy code, table-driven algorithm, matrix-driven algorithm, CRC32.

Мыцко Евгений Алексе-евич, студент кафедры вычислительной техники Института кибернетики ТПУ. E-mail: 7evgen7@sibmail.com
Область научных интересов: программная и аппаратная реализации алгоритмов вычисления контрольной суммы, тестирование и сравнение по быстродействию алгоритмов вычисления CRC.

Мальчуков Андрей Никола-евич, канд. техн. наук, доцент кафедры вычислительной техники Института кибер-нетики ТПУ.
E-mail: jgs@tpu.ru
Область научных интересов: помехоустойчивое кодирова-ние, полиномиальные коды, системы проектирования помехоустойчивых полиномиальных кодов, алго-ритмы поиска образующих полиномов, быстродей-ствующие алгоритмы кодиро-вания и декодирования данных полиномиальными кодами.

Методы обнаружения ошибок предназначены для выявления повреждений сообщений при их передаче по зашумлённым каналам. Для этого передающее устройство вычисляет некоторое число, называемое контрольной суммой и являющееся функцией сообщения, и добавляет его к этому сообщению. Приёмное устройство, используя тот же самый алгоритм, рассчитывает контрольную сумму принятого сообщения и сравнивает её с переданным значением [1]. Как правило, контрольная сумма посылается (считывается) в конце сообщения:

<исходное неизмененное сообщение> <контрольная сумма>

Существуют различные алгоритмы расчёта контрольной суммы, такие как CRC, MD5.

CRC – циклический избыточный код (англ. Cyclic redundancy code) – алгоритм вычисления контрольной суммы, предназначенный для проверки целостности передаваемых данных. Алгоритм CRC обнаруживает все одиночные ошибки, двойные ошибки и ошибки в нечетном числе битов. CRC обычно используется в протоколах передачи данных (Ethernet, Bluetooth, ZigBee) и архиваторах данных (Pkzip,Winzip). Существует несколько вариантов реализации CRC, таких как «классический» [2], матричный [2, 3] и табличный [1, 4].

Табличный алгоритм

Данный алгоритм позволяет вычислять контрольную сумму побайтно, в отличие от «классического» [2] побитового алгоритма. Суть алгоритма заключается в том, что при последовательном сдвиге данных по байту за итерацию, изменения, которые должны были происходить в течение 8 сдвигов при классическом алгоритме, заносятся в таблицу, на основе которой вычисляется контрольная сумма. При последовательном сдвиге регистра, выдвинутый байт является индексом элемента в рассчитанной таблице, с которым будет складываться по модулю два байт из сообщения (рис. 1).

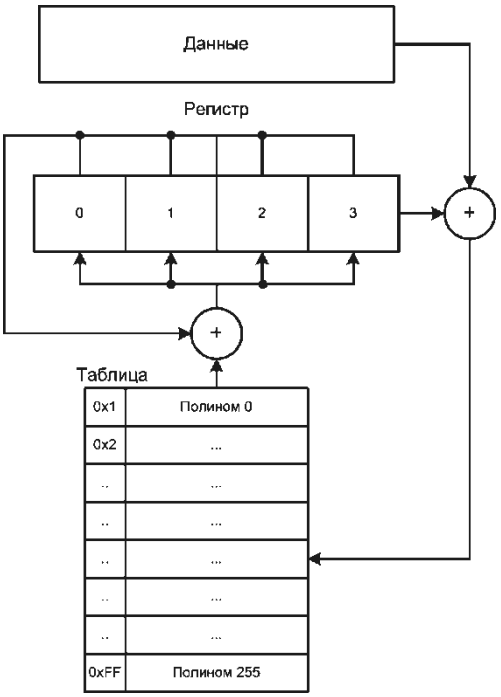

Рис. 1. Схематичное представление табличного алгоритма

Табличный алгоритм является широко используемым на практике, и большинство программ для расчёта контрольной суммы CRC основаны на этом алгоритме.

Матричный алгоритм

Процесс вычисления и проверки контрольной суммы CRC32 в матричном алгоритме осуществляется также как и в табличном, за исключением того, что вместо таблицы используется операция умножения вектора (выдвинутый байт) на матрицу (рис. 2, 3).

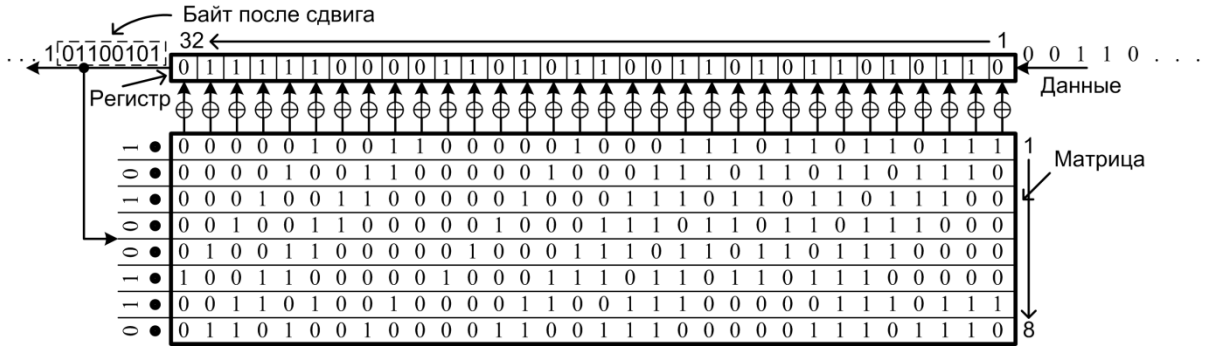

Рис. 2. Схематичное представление матричного алгоритма со сдвигом 1 байт

При программной реализации матричного алгоритма (однобайтовый сдвиг) вычисления контрольной суммы CRC32 потребуется объём памяти равный 32 байт (для хранения матрицы).

Матричный алгоритм можно ускорить, если вместо однобайтового сдвига использовать сдвиг по 2 или 4 байта. Сдвиг по 3 байта в данном случае не рассматривается, так как при программной реализации отсутствует тип данных размером 3 байта, что вызывает трудности и потерю скорости при реализации алгоритма.

При увеличении числа байт, обрабатываемых за итерацию, будет увеличиваться размер матрицы, используемой при вычислениях. Для двухбайтового сдвига матрица расширится до 64-х байт; для четырёхбайтового сдвига до 128 байт [2].

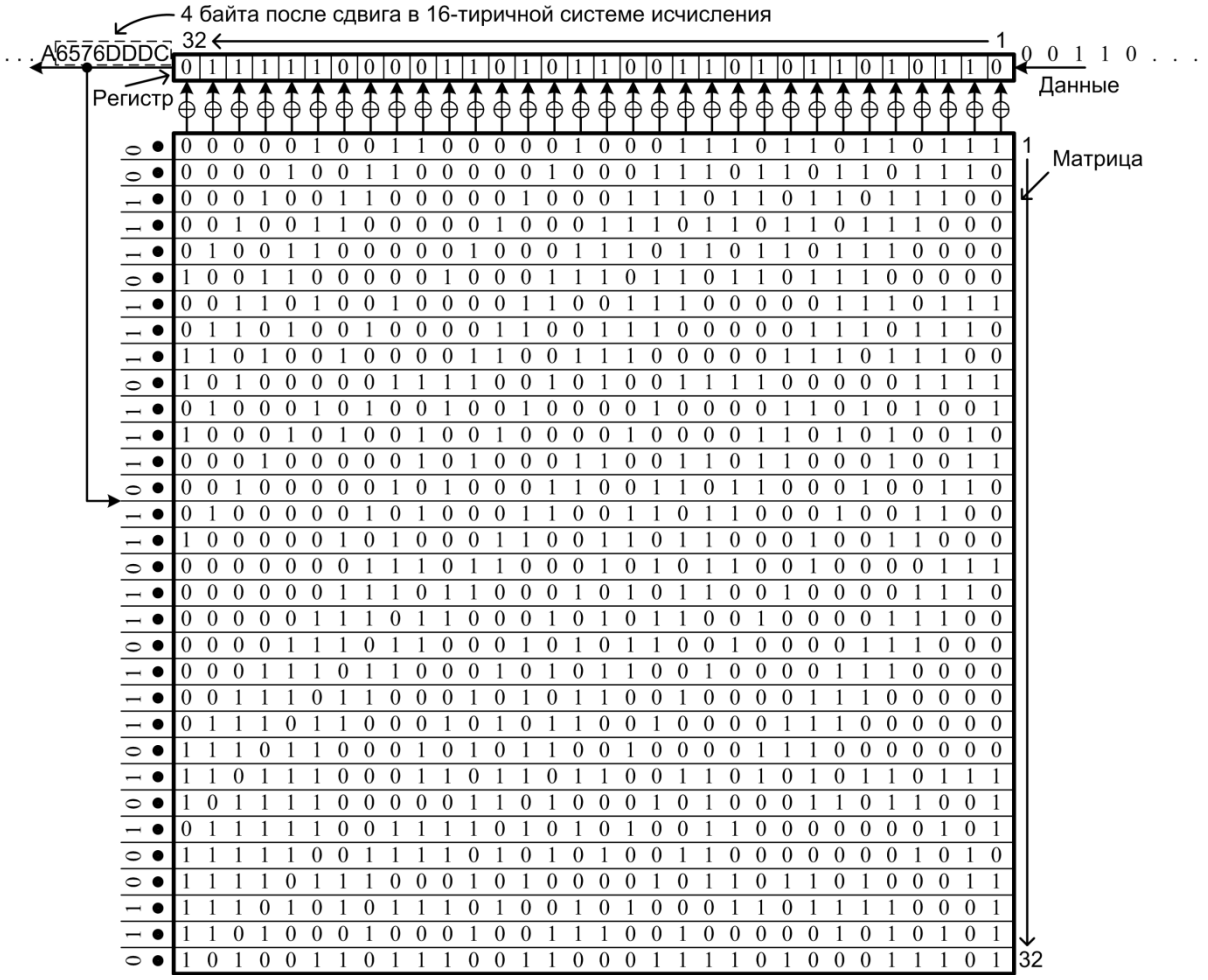

Рис. 3. Схематичное представление матричного алгоритма со сдвигом 4 байта

Исследование скорости вычисления контрольной суммы

Для исследования программных реализаций различных алгоритмов по времени вычисления контрольной суммы CRC32 использовался суперкомпьютерный кластер «СКИФ-политех1». Для получения временной оценки вычисления контрольной суммы над конкретным файлом конкретной программной реализацией производилось по 50 запусков для файлов объёмом от 10 до 1010 Мегабайт с шагом 200 Мегабайт. На основе данных по 50-ти запускам каждого алгоритма составлены таблицы средних значений по времени расчёта контрольной суммы, по доверительным интервалам и по размерам исполняемых файлов.

Исследование проводилось в два этапа – с буфером в 1 Мб, который установлен в исходном коде табличной реализации [5] и с буфером 256 байт – для того, чтобы отследить влияние буфера на время вычисления контрольной суммы.

Исследование скорости вычисления алгоритмов с буфером 1 Мб

Использование суперкомпьютерного кластера позволило сэкономить время на проведение эксперимента и получить временные оценки с достаточно высокой достоверностью (табл.1).

Таблица 1. Средние значения времени расчёта CRC32, с, и доверительные интервалы результата, %

Алгоритм	Объём использованных файлов, Мб					
	10	210	410	610	810	1010
Табличный	0,057 (±1,4)	1,194 (±0,217)	2,299 (±0,203)	3,227 (±0,162)	4,334 (±0,211)	6,184 (±0,155)
Матричный (сдвиг 1 байт)	0,125 (±0,732)	2,81 (±0,112)	5,399 (±0,067)	7,602 (±0,166)	10,23 (±0,161)	14,557 (±0,048)
Матричный (сдвиг 2 байта)	0,092 (±3,882)	2,084 (±0,133)	4,02 (±0,09)	5,645 (±0,105)	7,566 (±0,097)	10,835 (±0,072)
Матричный (сдвиг 4 байта)	0,08 (±0,827)	1,729 (±0,172)	3,32 (±0,011)	4,694 (±0,314)	6,272 (±0,188)	8,965 (±0,064)

Размеры запускаемых файлов табличной и матричных программных реализаций заметно отличаются друг от друга (табл. 2).

Таблица 2. Размеры исполняемых файлов

Алгоритм	Табличный	МС 1 байт	МС 2 байта	МС 4 байта
Размер исполняемого файла, байт	15 490	11 729	12 247	12 994

Таблица 3. Ускорение относительно однобайтового сдвига

Алгоритм	Ускорение относительно матричного алгоритма со сдвигом 1 байт, %					
	Объём использованных файлов, Мб					
	10	210	410	610	810	1010
МС 2 байта	26,4	25,836	25,541	25,743	26,041	25,568
МС 4 байта	36	37,919	38,507	38,253	38,69	38,414

Для того чтобы выяснить как влияет на скорость вычисления увеличение числа байт, обрабатываемых за итерацию, в матричном алгоритме была вычислена разница для данных из табл. 1 для матричных алгоритмов с 2-х и 4-х байтными сдвигами относительно однобайтового (табл. 3).

Таблица 4. Ускорение матричного алгоритма относительно табличного, %

Алгоритм	Объём использованных файлов, Мб					
	10	210	410	610	810	1010
МС 1 байт	-119,298	-135,343	-134,841	-135,575	-136,575	-135,398
МС 2 байта	-61,403	-74,539	-74,858	-74,93	-74,573	-75,21
МС 4 байта	-40,35	-44,807	-44,41	-45,46	-44,716	-44,97

Как видно из таблицы 4 матричный алгоритм с 4-х байтным сдвигом на 38 % вычисляет контрольную сумму быстрее, чем матричный алгоритм с однобайтовым сдвигом.

Однако даже матричный алгоритм с 4-х байтовым сдвигом уступает по скорости табличному алгоритму на 44 % (табл. 4).

С помощью увеличения числа байт, обрабатываемых за итерацию, в матричном алгоритме скорость вычисления контрольной суммы можно приблизить к скорости табличного до отставания на ~44 %. Следует помнить, что четырёхбайтовый матричный алгоритм требует

в 8 раз меньше памяти (128 байт) для хранения матрицы, чем для реализации табличного алгоритма (1024 байт).

Исследование скорости вычисления алгоритмов для микроконтроллеров

В данном случае размер буфера был установлен равным 256 байт, что соответствует доступному объёму для микроконтроллеров. Все этапы исследования проводились аналогично исследованиям с буфером 1 мегабайт. Временные затраты на вычисления контрольной суммы файлов разного объёма и доверительные интервалы полученных результатов приведены в табл. 5.

Таблица 5. Средние значения времени расчёта CRC32, с, и доверительные интервалы результата, %

Алгоритм	Объём использованных файлов, Мб					
	10	210	410	610	810	1010
Табличный	0,053 (±1,899)	1,195 (±0,36)	2,299 (±0,195)	3,382 (±0,385)	4,501 (±0,375)	6,202 (±0,138)
Матричный (сдвиг 1 байт)	0,125 (±0,786)	2,807 (±0,212)	5,392 (±0,187)	7,735 (±0,117)	10,36 (±0,178)	14,533 (±0,107)
Матричный (сдвиг 2 байта)	0,09 (±1,441)	2,085 (±0,294)	4,018 (±0,282)	5,715 (±0,199)	7,661 (±0,175)	10,842 (±0,161)
Матричный (сдвиг 4 байта)	0,078 (±0,808)	1,726 (±0,142)	3,321 (±0,11)	4,722 (±0,217)	6,331 (±0,186)	8,941 (±0,073)

Как видно из табл. 6 и 7 изменение размера буфера до 256 байт существенно не изменил картину, полученную ранее.

Таблица 6. Ускорение относительно однобайтового сдвига

Алгоритм	Ускорение относительно матричного алгоритма со сдвигом 1 байт, %					
	Объём использованных файлов, Мб					
	10	210	410	610	810	1010
МС 2 байта	28	25,721	25,482	26,115	26,052	25,397
МС 4 байта	37,6	38,498	38,408	38,952	38.889	38,477

Таблица 7. Ускорение матричного алгоритма относительно табличного, %

Алгоритм	Объём использованных файлов, Мб					
	10	210	410	610	810	1010
МС 1 байт	-135,849	-134,895	-134,537	-128,711	-130,171	-134,328
МС 2 байта	-69,811	-74,477	-74,771	-68,982	-70,206	-74,814
МС 4 байта	-47,169	-44,435	-44,454	-39,621	-40,657	-44,163

В среднем ускорение матричного алгоритма относительно табличного остаётся таким же, как и в алгоритмах с размером буфера 1 Мб. Это обусловлено, скорее всего, тем, что на суперкомпьютерном кластере файлы считываются в память достаточно быстро, поэтому изменение размера буфера не может существенно влиять на результат измерений.

Заключение

В ходе исследования были рассмотрены табличный [1] и матричный [2, 3] алгоритмы вычисления контрольной суммы CRC32. Проведено сравнение по скорости вычисления контрольной суммы данными алгоритмами. В результате матричный алгоритм был

модернизирован за счёт увеличения числа байт, обрабатываемых за итерацию. Тестирование программных реализаций по скорости вычисления контрольной суммы показало, что матричный алгоритм в системах с быстрой дисковой подсистемой на ~44 % медленнее табличного. В результате было установлено, что самый оптимальный и быстрый четырёхбайтовый матричный алгоритм следует применять во встраиваемых системах, промышленных системах передачи данных, где используются микроконтроллеры. Рекомендация использования матричного алгоритма обусловлена простотой реализацией и значительной экономией памяти, требуемой для прошивки программы в микроконтроллер.

СПИСОК ЛИТЕРАТУРЫ

1. Ross N. W. A Painless Guide to CRC Error Detection Algorithms // Dr Ross Williams. 1993. URL: http://www.ross.net/crc/download/crc_v3.txt. (дата обращения: 01.09.2011).
2. Мальчуков А.Н., Осокин А.Н. Быстрое вычисление контрольной суммы CRC: таблица против матрицы // Прикладная информатика. – 2010. – № 2(26). – С. 58–63.
3. Мальчуков А.Н., Осокин А.Н. Быстродействующие алгоритмы вычисления контрольной суммы на примере CRC8 // Молодежь и современные информационные технологии: Сборник трудов VIII Всерос. научно-практ. конф. студентов, аспирантов и молодых ученых. – Томск, 3-5 марта 2010. – Томск: СПБ Графикс, 2010. – С. 34–35.
4. P. Koopman. 32-Bit Cyclic Redundancy Codes for Internet Applications // Phil Koopman Publications & Patents. 2002. URL: http://www.ece.cmu.edu/~koopman/networks/dsn02/dsn02_koopman.pdf (дата обращения: 01.09.2011).
5. 32 bit Cyclic Redundancy Check Source Code for C++ // Create Window Website. 2011. URL: http://www.createwindow.com/programming/crc32/ (дата обращения: 01.09.2011).

Поступила 14.10.2011 г.