NEON INSTRUCTION SET EFFICIENCY AND THEIR USE IN CODING THEORY PROBLEMS

Maria Pashinska-Gadzheva
Institute of Mathematics and Informatics, Bulgarian Academy of Sciences, Sofia, Bulgaria
mariqpashinska@math.bas.bg

Iliya Bouyukliev
Institute of Mathematics and Informatics, Bulgarian Academy of Sciences, Sofia, Bulgaria
iliyab@math.bas.bg

ЕФЕКТИВНОСТ НА NEON ИНСТРУКЦИИ И ТЯХНОТО ИЗПОЛЗВАНЕ В ЗАДАЧИ ОТ ТЕОРИЯ НА КОДИРАНЕТО

Abstract

A large number of algorithms solving Coding Theory problems involve operations on vectors over finite fields. The use of extended CPU registers and instructions is suitable for the optimization of these algorithms. Current work presents the Neon instruction set for the ARM architectures used in Apple's M series of processors. A method for their application in algorithms for finding weight characteristics is considered. Some basic similarities and differences with extended vector instructions available in other architectures are presented. Their performance is compared with non-vectorized methods.

Keywords: *Vectorization; NEON Instruction Set.*

ВЪВЕДЕНИЕ

Векторизацията е един от основните подходи за ускорение на компютърни изчисления в съвременните централни процесори. Тя представлява изпълнение на дадени команди за няколко елемента едновременно. Това е възможно благодарение на разширените векторни регистри, които са с по-голяма дължина от стандартните работни регистри (128, 256, 512 бита). Те са налични в различните архитектури за централни процесори. За да се предостави възможност за работа с такива регистри са разработени множества от специализирани инструкции за различните централни процесори. Основната идея на такива инструкции е да се позволи интерпретирането на масиви от стандартни типове данни като вектори с определена дължина в зависимост от дължината на разширения регистър, което позволява да се извърши паралелизация спрямо данните. Паралелизацията е основен подход за ускоряване на различни алгоритми с висока сложност. Голяма част от задачите в Теория на кодирането са NP-пълни, като за рашаването на много проблеми се използват компютърни изчисления и оптимизирани алгоритми. Тук паралелизацията играе важна роля за решаването на голям кръг от задачи. Векторизацията е много подходящ подход за паралелизация на алгоритми и съответно за задачи, които изпълняват изчисления върху вектори. Такива са алгоритмите за изследване на линейни кодове над крайни полета и техните тегловни характеристики. В този доклад ще бъдат представени основни функции и типове от данни, които позволява векторизация на алгоритми, в частност алгоритми за извършване на операции над вектори над крайни полета, в ARM (Advanced RISC Machines) архитектури и разширения набор от регистри NEON [1].

Изложение

Разширени векторни регистри присъстват в голяма част от съвременните архитектури. За архитектури от вида x86 съществуват векторни регистри с големина 128, 256 и 512 бита. За тях са разработени и специализирани "семейства" от инструкции и типове от данни (SSE, AVX, AVX512), които позволяват работата с регистри в езици от високо ниво като C/C++ [2]. От друга страна, архитектурите от вида ARM са все по-широко използвани поради тяхната енергийна ефективност. Процесори и копроцесори с тази архитектура се използват както в системи за видео обработка, така и в системи за персонална употреба. За тези архитектури са разработени т. нар. NEON регистри с дължина 128 бита и съответен набор от инструкции. Тези регистри могат да се разглеждат като аналог на 128-битовите регистри и SSE инструкциите в x86 архитектурите. Някои съвременните компилатори позволяват "автоматична" векторизация на даден софтуерен код (GCC, Arm Compiler for HPC и др.), като също така е възможно изрично да се укаже „желанието" за векторизация на кода чрез директиви на компилатора (OpenMP). В тези случаи компилаторът „превежда" дадения код на машинен такъв, който да използва разширените инструкции за конкретната архитектура. По-ефикасен подход за векторизация е директното използване на функциите и типовете от данни, създадени за работа с тези разширени векторни регистри в езиците от високо ниво [3], [4]. При директното им използване даденият алгоритъм може да бъде по-ефикасно паралелизиран. За това е необходимо познание на архитектурата, подходи и допълнителни инструменти за генериране на преносим софтуер, както и много добро познаване на възможностите на различните инструкции. Въпреки че разширените регистри за x86 и ARM архитектурите и съответните функции и типове от данни имат сходна цел - лесно осъществяване на векторизация - те се различават в голяма степен в своята имплементация. Следователно, ускорението на алгоритми, разчитащо на изрична векторизация, може да има различна имплементация в зависимост от архитектурата. Тук са представени някои основни типове операции, които имат различна имплементация при различните архитектури:

- Синтаксис на типовете от данни и функции - за работа с Neon регистрите са разработени специални типове от данни. Те позволяват записването на еднотипни елементи в 64- или 128-битови регистри. Всеки тип от данни описва типа на данните, които ще бъдат записани в регистъра, неговата големина и броя на елементите, които ще бъдат записани в регистъра (int8x16_t съдържа 16 целочислени елемента с големина 8 бита). Броят на елементите, които могат да бъдат записани в регистъра, зависи от големината на регистъра както и големината на базовия тип данни (128-битов регистър може да съдържа 16 елемента от типа char, 8 елемента от типа short int, 4 елемента от типа int и т.н.). При регистрите за x86 архитектури се описва единствено големината на регистъра и типа данни (m128i съдържа целочислени данни, записани в 128-битов регистър). Функциите, дефинирани в специални библиотеки, улесняват работата с регистрите в езиците от високо ниво C/C++. Тези функции и типовете от данни са дефинирани в библиотеката arm_neon.h и се използват като стандартни такива. По време на компилацията функциите се заменят от последователност от инструкции на ниско ниво. Това означава, че може да се изрази архитектурно поведение от ниско ниво на език от високо ниво. Дефинираните функции имат стандартизирани наименования, които описват операцията на функцията, големината на регистъра (наличието на символа "q" след кодовата дума за операцията показва, че ще бъде използван

128-битов регистър), типа на данните и тяхната големина. Всяка функция започва със символа "v".

Пример: vaddq_s64 - функция, която събира два 128-битови регистъра, които съдържат 64-битови цели числа със знак.

- Операции за сравнение на вектори от целочислени елементи - в Neon инструкциите са дефинирани операциите (<, >, =, ≤, ≥) за сравнение на целочислените елементите, записани в регистър. Операциите сравняват едновременно съответните елементи, записани в два регистъра. Резултатът се записва в трети регистър, като за всеки съответен елемент резултатът е променлива с подходяща големина, съдържаща само битове 1, ако сравнението е вярно, и само 0 в противен случай. Инструкциите от семействата SSE, AVX и AVX512 притежават аналогични функции, позволяващи сравнение на елементите в два регистъра. За разлика от Neon инструкциите, при тях не са налични всички операции за сравнение. Също така, наличието на дадена операция за едно от семействата не гарантира наличието ѝ за останалите.
- Операции за смесване и пермутации - някои от наборите от инструкции за x86 архитектури притежават функции за смесване на два регистъра (**blendv**) и извършване на пермутации на елементите в един регистър (**shuffle**, **permute**). Такива инструкции не са налични за набора от инструкции Neon.
- Функции за намиране на броя на ненулевите битове на целочислени променливи – голяма част от алгоритмите за намиране на тегловни инварианти изискват намирането на теглото на вектор. При използването на векторизация и подходящо представяне на елементите на полето намирането на тегло на вектор може да се сведе до намирането на ненулевите битове в компютърна дума или регистър. Такива функции са известни като ***population count*** или ***popcnt***. В Neon такива инструкции намират броя на ненулевите битове на всеки байт, записан като вектор в регистър с големина 64 или 128 бита. Резултатът за съответните елементи се записва в друг регистър на съответни позиции. За намирането на теглото на цял векторен регистър е необходимо да се съберат резултатните стойности, което е възможно чрез една функция. Някои процесори с x86 архитектура имат специална popcnt функция, която намира теглото на 32 или 64 битова компютърна дума. В подкатегорията ***AVX512VPOPCNTDQ*** на AVX512 са разработени функции за намирането на 32 или 64-битовите елементи, записани в регистър с големина 128, 256 или 512 бита. Резултатът се записва аналогично в регистър със съответната дължина.

Експериментални резултати

За да се анализира ефективността на Neon инструкциите е имплементиран алгоритъм за събиране на вектори над крайно просто поле с до 128 елемента и алгоритъм за намирането на теглото на вектор, описани в [5]. Те са използвани за намирането на тегловен спектър на линеен код. Имплементацията им е сравнена с невекторизирана имплементация на алгоритъм за намиране на спектър на линеен код, като се използват таблици за реализиране на събиране и умножение на елементите на полето. Изчисленията са извършени на централен процесор от серията M1 на Apple с 8 ядра, 8 ГБ оперативна

памет, операционна система MacOS Sonoma 14.5 и компилатор clang 15.0.0. В Таблица 1 са представени времената за работа на двете имплементации в секунди. В първите три колони са дадени параметрите на кода, в третата - времето за работа за векторизирана версия на алгоритмите с Neon инструкции, а в последната колона - невекторизирана версия на алгоритмите. От представените резултати се вижда, че векторизираната версия е между 2 и 5 пъти по-бърза в сравнение с невекторизираната версия, като ускорението се увеличава с увеличаване на дължината на кода.

Таблица 1. Време за работа на векторизиран и невекторизиран алгоритъм

n	k	q	Neon	Без векторизация
40	8	17	4,566	11,900
80	8	17	5,460	23,819
120	8	17	7,519	39,404
40	5	97	0,903	2.387
80	5	97	1,125	4,795
120	5	97	1,487	7,588
40	5	101	1,035	2,745
80	5	101	1,311	5,568
120	5	101	1,770	9,108

Заключение

Векторизацията е основен подход за паралелизация и е особено подходяща за алгоритми, които изпълняват операции над вектори. За постигане на векторизация са разработени разширени регистри с по-голяма дължина, специализирани инструкции за централните процесори и функции и типове от данни, които позволяват работата с тях чрез езици от високо ниво. Такива са инструкциите Neon, които са налични в съвременните ARM централни процесори. Те постигат добра скалируемост, която се вижда в реализацията на алгоритъм за събиране на вектори над крайно поле. Експерименталните резултати показват, че ускорението, получено при сравнение с невекторизиран алгоритъм, се увеличава с увеличаване на дължината на кода и зависи единствено от този параметър.

Благодарности

Тази публикация е финансирана от Министерство на образованието и науката в изпълнение на Национална научна програма „Сигурност и отбрана", приета с РМС № 731 от 21.10.2021 г. и съгласно Споразумение № Д01-74/19.05.2022 г.

References // Литература

1. ARM. – "Arm – architecture – instruction – set – guide", 2023, Available at: https://developer.arm.com/architectures/instruction-sets/intrinsics/ (last view: 05-08-2024)
2. Intel. "Intel ® 64 and ia-32 architectures software developer's manual", 2024, Available at: https://www.intel.com/content/www/us/en/developer/articles/technical/intel-sdm.html (last view: 05-08-2024)
3. G. Mitra, B. Johnston, A. P. Rendell, E. McCreath and J. Zhou, "Use of SIMD Vector Operations to Accelerate Application Code Performance on Low-Powered ARM and Intel Platforms", 2013 IEEE

International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum, Cambridge, MA, USA, 2013, pp. 1107-1116

4. A. Fog, "Vcl—c++ vector class library manual". Copenhagen, Denmark: Technical University of Denmark, 2020.
5. M. Pashinska-Gadzheva, I. Bouyukliev, "About Methods of Vector Addition over Finite Fields Using Extended Vector Registers". In: Lirkov, I., Margenov, S. (eds) Large-Scale Scientific Computations. LSSC 2023. Lecture Notes in Computer Science, vol 13952. Springer, Cham, 2024, pp. 427-434.

Received: 15-08-2024 Accepted: 20-12-2024 Published: 26-12-2024

Cite as:
Pashinska-Gadzheva, M., Bouyukliev, I. (2024). "NEON Instruction Set Efficiency and their Use in Coding Theory Problems", Science Series "Innovative STEM Education", volume 06, ISSN: 2683-1333, pp. 250-254, 2024. DOI: https://doi.org/10.55630/STEM.2024.0628