УДК 681.5.62–5 + 681.5.681.3

Э.И. ВАТУТИН
Курский государственный технический университет

Оптимизация обработки множеств

Довольно часто в процессе разработки программ возникает необходимость в реализации набора операций, инкапсулирующих в своем составе действия над множествами (добавление/удаление/проверка включения элементов, объединение, пересечение, дополнение, разность, вычисление мощности, вспомогательные операции). Некоторые языки программирования имеют поддержку множеств на уровне компилятора, другие реализуют классы по работе с наборами бит, которые могут быть адаптированы для организации операций над множествами. В случае поддержки компилятором размер множества как правило ограничивается 256 элементами (Delphi). Реализации на основе стандартных классов (например, TBits в Delphi), не имеющие ограничений на количество элементов, как правило не обладают высокой скоростью за счет использования конструкций языков высокого уровня и неиспользования прогрессивных наборов команд (работы с битовыми строками, SIMD-операций). В случае необходимости быстрого выполнения операций возникает потребность в разработке собственной реализации перечисленных операций.

Такая реализация может быть построена на основе битовых строк, в которых наличие или отсутствие бита в *i*-ой позиции хранит информацию о наличии или отсутствии *i*-ого элемента в множестве. Данный способ реализации по сравнению с другими возможными (например, на основе массивов элементов) обладает рядом преимуществ: использует минимальное количество памяти, позволяет быстро организовать как операции включения/исключения/проверки присутствия элемента (поддержка процессором операций над битовыми строками), так и объединения/пересечения/дополнения/разности (использования SIMD-обработки).

Операции включения/исключения/проверки присутствия элемента реализуются с использованием ассемблерных команд BT (Bit Test), BTR (Bit Test and Reset), BTS (Bit Test and Set) [1] позволяющих избежать явной адресной арифметики. Операция

вычисления мощности множества реализована с использованием команды BSF (Bit Scan Forward), позволяющей избежать использования циклов при поиске бит. Операции объединения, пересечения, дополнения и разности множеств могут быть реализованы с использованием логических команд (соответственно OR, AND, NOT, AND-NOT). Учитывая возможность параллельной обработки бит, входящих в состав битовой строки, возможно использования принципа SIMD (Single Instruction Multiple Data) при организации перечисленных операций, реализованного в процессорах семейства x86 в виде MMX и SSE расширений [2]. Для этого необходимо прежде всего определение наличия требуемых расширений (с использованием команды CPUID (CPU Identification) или с использованием средств операционной системы (получение исключения при попытке исполнения неподдерживаемой команды)) [3]. В зависимости от поддерживаемых расширений возможно использование следующих наборов команд: AND, OR, NOT (выполнение с использованием АЛУ по 32 бита); PAND, POR, PXOR, PANDN (MMX- и/или SSE2-расширения по 64 и/или 128 бит соответственно); ANDPS, ORPS, XORPS, ANDNPS (SSE-расширение по 128 бит) [1].

Разработанный программный модуль (unit), содержащий класс TSet и средства по определению набора поддерживаемых расширений, написан с применением встроенного ассемблера в среде Delphi 7. Он был использован при построении программной реализации метода построения разбиений параллельных управляющих алгоритмов [4]. Анализ эффективности разработанного класса с использованием профайлера Intel VTune Performance Analyzer показывает уменьшение времени, затрачиваемого на операции над множествами, на 3 порядка по сравнению с реализацией на основе класса TBits. При построении разбиений алгоритмов малой размерности (до 50 вершин) до оптимизации время, затрачиваемое на операции над множествами, составляло порядка 10% от общего времени построения разбиения; после проведения оптимизации оно составляет менее 1%.

---

1. IA-32 Software Developers Manual. Vol. 2: Instruction Set Reference. Order number 245471-006. www.developer.intel.com

2. IA-32 Software Developers Manual. Vol. 1: Basic Architecture. Order number 245470-006. www.developer.intel.com
3. Intel Pentium 4 and Intel Xeon Processor Optimization. Reference Manual. Order number 248966-05. www.developer.intel.com
4. Ватутин Э.И., Зотов И.В. Метод формирования субоптимальных разбиений параллельных управляющих алгоритмов // Труды II международной конференции «Параллельные вычисления и задачи управления» PACO ’04 памяти Е.Г. Сухова. М.: Институт проблем управления им. В.А. Трапезникова РАН. С. 884–917.