МЕТОДЫ ОБРАБОТКИ ИНФОРМАЦИИ
(Вычислительные системы)
1978 год Выпуск 74

УДК 8.74

ТАБЛИЧНЫЙ АЛГОРИТМ ВЫЧИСЛЕНИЯ ОСТАТКОВ ОТ ДЕЛЕНИЯ ПРОИЗВОЛЬНЫХ ЦЕЛЫХ ЧИСЕЛ НА ФИКСИРОВАННОЕ ЦЕЛОЕ ЧИСЛО

Т.Н.Титкова

Довольно часто вопрос о размещении некоторого массива информации в оперативной памяти ограниченного объема в решается методом ассоциативного кодирования (hash coding) [I]. Важным элемен - том процедуры ассоциативного кодирования является способ преобразования ключа (имени элемента информационного массива X) в адрес, реализуемый посредством вычисления некоторой функции (функции расстановки). Одним из наиболее хорошо зарекомендовавших себя на практике способов является метод деления [2]. В соответствии с этим методом адрес, по которому размещается элемент информационного массива в расстановочном поле, определяется по формуле

$$h(K) = K \bmod M , \qquad (I)$$

где K - ключ или признак размещаемого в памяти элемента информа - ционного массива, интерпретируемый как целое число, M - объем расстановочного поля (максимальное число элементов, которые могут быть размещены в расстановочном поле).

Поскольку K и M - целые числа, вычисление h(K) эквивалентно получению остатка от целочисленного деления. Так как делитель M для всех N (N - число элементов массива X) значений ключа есть постоянное число, большее или равное N , то при размещении массива X в оперативной памяти должна решаться задача многократного вычисления остатков от деления набора целых чисел на фиксированное число.

В некоторых задачах (например, в задаче поиска статистических закономерностей в символьных последовательностях [3]) объемы обрабатываемых информационных массивов могут превышать размер рас-

становочного поля, реально имеющегося в распоряжении программиста. Обработку таких массивов приходится вести итеративно, осуществляя многократное заполнение (с последующей очисткой) расстановочного поля. Количество обращений к процедуре, вычисляющей функцию расстановки, может при этом оказаться во много раз больше, чем размерность массива, и задача сокращения времени вычисления функции расстановки становится актуальной.

Операция деления в большинстве ЭВМ наиболее трудоемкая. В ЭВМ "Минск-32", например, операция целочисленного деления осуществляется с помощью стандартной программы, использующей алгоритм получения частного путем умножения на величину, обратную делителю. При этом производится перевод делимого и делителя в представление в виде чисел с плавающей запятой, и если целое делимое занимает всю машинную ячейку, то используются вычисления с двойной точностью. Деление двух целых чисел с использованием стандартной программы на ЭВМ "Минск-32" осуществляется ~ за 1130 мксек, в то время как логические операции выполняются за 10-30 мксек, а деление с плавающей запятой осуществляется за 15-170 мксек.

В [4] и [5] описаны два алгоритма ускоренного (по сравнению с общим случаем) деления на постоянный делитель, но на делитель в обоих алгоритмах накладываются некоторые ограничения: в одном случае [4] величина обратная делителю должна быть периодической двоичной дробью, а в другом случае [5] делитель должен быть ограничен по величине некоторой константой (~ 100).

В настоящей работе предлагается алгоритм ускоренного вычисления остатка от целочисленного деления, ориентированный на тот случай, когда возникает потребность в многократном (уточнение будет дано ниже) делении произвольных целых чисел на фиксированное число без каких-либо ограничений на делитель. Содержательным примером таких задач является процедура ассоциативного кодирования.

Добиться сокращения времени выполнения некоторой вычислительной процедуры удается зачастую путем привлечения дополнительных ресурсов памяти, которые используются для организации таблиц, содержащих в себе ту часть информации, которую в первоначальном варианте приходилось просчитывать каждый раз в теле самой процедуры. В данной статье описывается способ вычисления $h(K)$ в соответствии с (1), основанный на этой идее.

Признак или ключ K, по которому ищется адрес, разбивается на p смежных неперекрывающихся блоков, каждый из которых содержит

q_j двоичных разрядов ($j = 1,2,...,p$: $\sum_{j=1}^{p} q_j = n$; n – число разрядов в машинной ячейке). Границы между блоками характеризуются величинами $r_j = \sum_{i=1}^{j} q_i$ ($r_0 = 0$, по определению). Тогда целое число, соответствующее коду K, может быть записано в виде

$$S(K) = \sum_{i=0}^{n-1} k_i \cdot 2^i = \sum_{j=1}^{p} \sum_{i=r_{j-1}}^{r_{j-1}+q_j-1} k_i \cdot 2^i = \sum_{j=1}^{p} \alpha_j \quad , \qquad (2)$$

а выражение (I) примет вид:

$$h(K) = S(K) \bmod M = \left(\sum_{j=0}^{p} \alpha_j \bmod M \right) \bmod M \ . \qquad (3)$$

Нетрудно видеть, что каждая из величин

$$\alpha_j = \sum_{i=r_{j-1}}^{r_{j-1}+q_j-1} k_i \cdot 2^i$$

при произвольных комбинациях значений k_i может принимать не более 2^{q_j} значений ($k_i = 0$ или I), которые обозначим d_{jt} ($t=0,1,2,...,2^{q_j}$). Величины $\alpha_{jt} \bmod M$ могут быть затабулированы для всех значений j и t, на что потребуется дополнительный объем памяти, равный $\sum_{j=1}^{p} 2^{q_j}$ машинных слов. Вычисление $h(K)$ тогда осуществляется по следующей схеме:

0) $\alpha := 0$; $j := 1$;

I) выделяется j-й блок из кода K при помощи соответствующей маски (q_j разрядов);

2) формируется адрес обращения к j-й таблице в соответствии с содержимым j-го блока (определение номера t);

3) $\alpha := \alpha + \alpha_{jt} \bmod M$ (целочисленное сложение);

4) цикл по j ($j = 1,...,p$);

5) вычисляется $\alpha \bmod M$.

Пункт 5 процедуры реализуется в соответствии с алгоритмом поиска места элемента в упорядоченном массиве. Действительно, имея в виду, что $\alpha = \sum_{j=1}^{p} \alpha_j \bmod M < p \cdot M$, можно организовать таблицу целочис-

ленных значений M, 2M,...,(p-1)M, за $\lfloor \log_2 p \rfloor$ сравнений осуществить поиск r^* ($r^* = 0,1,\ldots,p-1$) такого, что $r^* \cdot M \le \alpha < (r^* + 1)M$, и вычислить $\alpha \bmod M = \alpha - r^* \cdot M$.

Описанная процедура требует p целочисленных сложений и самое большее $\lfloor \log_2 p \rfloor$ сравнений. При надлежащем выборе числа блоков p и размеров блоков q_j ($j=1,\ldots,p$) трудоемкость вычисления h(K) описанным методом может быть существенно уменьшена по сравнению с вычислением h(K) при использовании стандартной программы целочисленного деления. Размер первого блока, включающего в себя младшие разряды признака K, целесообразно выбирать на основе соотношения $q_1 \le \mathrm{entier}(\log_2 M)$. При таком выборе q_1 не требуется заводить таблицу значений $\alpha_{1t} \bmod M$ для первого блока, поскольку всегда будет выполняться соотношение $\alpha_1 < M$. Нулевой шаг процедуры тогда будет иметь вид $\alpha := \alpha_1$. Размеры оставшихся блоков могут быть выбраны одинаковыми, исходя из имеющихся ресурсов памяти. К примеру, если объем расстановочного поля $M = 2^{14}$, код K имеет длину 37 разрядов, можно выбрать $q_1 = 14$, $q_2 = q_3 = q_4 = 6$, $q_5 = 5$, используя дополнительную память $s_{доп} = 3 \cdot 2^6 + 2^5 = 224$ ячейки. При этом потребуется 5 целочисленных сложений и не более 3 сравнений. Выбрав $q_1 = 14, q_2 = q_3 = q_4 = q_5 = 5$, $q_6 = 3$ (6 блоков), можно уменьшить дополнительную память до $s_{доп} = 4 \cdot 2^5 + 2^3 = 136$ ячеек, увеличив на единицу число сложений при том же числе сравнений и т.д.

Описанный алгоритм был реализован на ЭВМ "Минск-32". При этом были выбраны $q_1 = 9$, $q_2 = q_3 = q_4 = q_5 = 7$ (5 блоков), $s_{доп} = 4 \cdot 2^7 = 512$ ячеек. Время вычисления h(K) по сравнению с использованием стандартной процедуры целочисленного деления сократилось в 1,5 раза.

Отметим, что указанный относительный выигрыш достигался в наименее выигрышной для данного алгоритма ситуации – когда делимое занимало всю машинную ячейку. При более коротких ключах время работы данного алгоритма и требуемая память будут уменьшаться линейно, в то время как для стандартной процедуры затраты фактически останутся прежними (данный факт проверен экспериментально на ключах переменной длины). При более длинных (по сравнению с машинной ячейкой) ключах время работы данного алгоритма и требуемая память будут увеличиваться линейно как функция от длины ключа, тогда как для стандартной процедуры придется перейти к вычислению с двойной точностью, что приведет к нелинейному увеличению затрат.

В заключение еще раз подчеркнем, что указанный алгоритм целесообразно использовать в ситуациях, когда осуществляется многократное вычисление остатков от деления на фиксированное число. При смене делителя таблицы значений $\alpha_{j_t} \bmod M$ должны быть пересчитаны. Затраты на пересчет и позволят оценить размеры массива X, при которых использование данного метода (при наличии дополнительной памяти) будет давать преимущество. Вновь возвращаясь к разобранному примеру, видим, что для заполнения таблиц пришлось 512 раз прибегнуть к стандартной процедуре вычисления остатка. Следовательно, предложенный метод будет давать выигрыш по сравнению со стандартной методикой при N = 1500 и выше.

Литература

1. ВЕЛИЧКО В.М., ГУСЕВ В.Д., КОСАРЕВ Ю.Г., ЛОЗОВСКИЙ В.С., ТИТКОВА Т.Н. Ассоциативное кодирование: реализация и применение. -В кн.: Вычислительные системы. Вып. 62. Ассоциативное кодирование. Новосибирск, 1975, с. 3-37.

2. LUM V.Y., YUEN P.S.T., DODD M.M. Key-to-address transform techniques: a fundamental performance study on large existing formatted files. - "Communs ACM", 1971, v.14, N 4, p.228-239.

3. ГУСЕВ В.Д., КОСАРЕВ Ю.Г., ТИТКОВА Т.Н. О задаче поиска статистических закономерностей в символьных последовательностях. -В кн.: Вычислительные системы. Вып.62. Ассоциативное кодирование. Новосибирск, 1975, с. 49-71.

4. JACOBSOHN D.H. A combinatoric division algorithm for fixed-integer divisors. - "IEEE Trans.Computers", 1973, v.C22, N 6, p.608-610.

5. ARTZY E., HINDS J.A., SALL H.J. A fast division techniques for constant divisors. - "Communs ACM", 1976, v.19, N 2, p.98-101.

Поступила в ред.-изд.отд.
2 февраля 1978 года