国防科技大学学报
JOURNAL OF NATIONAL UNIVERSITY OF DEFENSE TECHNOLOGY
第 19 卷第 2 期　1997 年 4 月　Vol. 19 No. 2

基于 YH- F2 系统的一般算术表达式的并行成分识别*

姚新宇　戴金海

(国防科技大学自动控制系　长沙　410073)

摘　要　本文深入研究了基于 YH- F2 系统的算术表达式的并 行计算特点，提出了嵌套层次树的概念和相应的计算模式。这种模式充分利用了 YH- F2 系统的并行计算能力，完满地解决了基 于 YH- F2 的算术表达式的并行编译问题。

关键词　算述表达式，并行编译，计算树，YH- F2

分类号　TP301.6

Recognition of Parallelism of the Arithmetic Expressions Based on YH- F2 System

Yao Xinyu　Dai Jinhai

(Department of Automatic Control, NUDT, Changsha, 410073)

Abstract　This paper makes a complete analysis of the characteristics of the parallel calculation of the arithmetic expressions on YH- F2 system, and advances the concept of nested level tree, together with the appropriate calculation pattern of the arithmetic expressions. This pattern takes full advantages of the parallel calculating abilities of YH- F2 system, and solves the problem of parallel compiling on YH- F2.

Key Words　arithmetic expression, parallel compilation, calculation tree, YH- F2

YH- F2 是由紧耦合的算术逻辑处理机(ALP)和乘法处理机(MUP)完成算术表达式的计算，由于 YH- F2 没有通用寄存器，对算术表达式(含加、减、乘、一元负及括号运算)的并行编译较为困难。一个简单策略是将表达式按括号拆分为一系列短表达式，然后对这些短表达式进行编译。这种括号外提方案虽然在工程上简化了编译的复杂度，却也以失去部分并行性为代价。如对赋值语句："$x = (a_1 + b_1) \times (a_2 + b_2) \times \cdots \times (a_n + b_n)$"，括号外提:

* 国防预研基金资助项目
1996 年 5 月 27 日收稿

$$t_1 = a_1 + b_1$$
$$t_2 = a_2 + b_2$$

$$t_n = a_n + b_n$$
$$x = t_1 \times t_2 \times \cdots \times t_n$$

显然,由于加、乘分离和数据相关,ALP 和 MUP 两处理机只能串行计算,效率降低。又如括号嵌套型表达式:

$$x = a_1 + a_2 \times (a_3 + a_4 \times (a_5 + a_6 \times (a_7 + a_8 \times a_9)))$$

括号外提:

$$t_1 = a_7 + a_8 \times a_9$$
$$t_2 = a_5 + a_6 \times t_1$$
$$t_3 = a_3 + a_4 \times t_2$$
$$x = a_1 + a_2 \times t_3$$

这四个赋值语句由于数据相关,按 YH- F2 的核指令时序,下核要用上核结果则需插入一个 "NOP"核,这样在这四个赋值句间要插入三个 "NOP"核,增加了 6 拍开销。

为了解决上述两类问题,必须具体分析算术表达式的结构。

1 算术表达式的层次划分

YH- F2 异构型计算的特点是:复杂表达式加、乘并行计算体现为 ALP 和 MUP 的串行计算,于是提出了简单表达式的概念。

定义 **1** 简单表达式是一个无括号表达式,它的全部运算为同一类型,即同为加法(减法及一元负也视为加法)或同为乘法。

一个算术表达式可看成由一系列简单表达式组成,一个简单表达式可作另一简单表达式的一个操作数,YH- F2 的并行计算即为 ALP 和 MUP 分别在加、乘简单表达式上的独立计算,只在简单表达式结束才相互运算(求和结果作乘操作数或乘积结果作加操作数),这样不仅可以取得极大的并行度而且控制策略也大大简化了。从另一个角度看,由于简单表达式将表达式中同一类型的运算组合在一起,这样最大程度上减少了中间结果的保存。

综合计算树和简单表达式的概念,我们提出计算层次树的概念:

定义 **2** 计算层次树为表达式的树结构表示,其中树节点结构为{算符,操作数序列},即每个节点是一个简单表达式,其操作数可以是直接数,也可能是子简单表达式,且子节点的算符必与父节点的算符相异。图 1 为一个计算层次树。

2 计算算术表达式的准则

如图 1 所示的计算层次树中,加运算简单表达式为 A_1, A_2, A_3, A_4;乘简单表达式为 B_1, B_2, B_3;对 YH- F2 而言,A_1, A_2, A_3, A_4 中任一个都可以与 B_1, B_2, B_3 中任一个并行计算(当然未必一次连续算完)。确定计算树的线性序列就是确定这些简单表达式的计算顺序,如{A_3. A_4. A_1. A_2}{B_2. B_3. B_1} 就是一个计算顺序,而最优序列就是计算时间最小

的一个计算顺序。困难的是确定最优序列本身是一个非确定计算 (NP) 问题,要得到最优序列则需穷尽所有的表达式排序。对一个有 n 个加简单表达式和 m 个乘简单表达式的计算树而言,其计算顺序有 $P_n^n P_m^m$ 种组合。

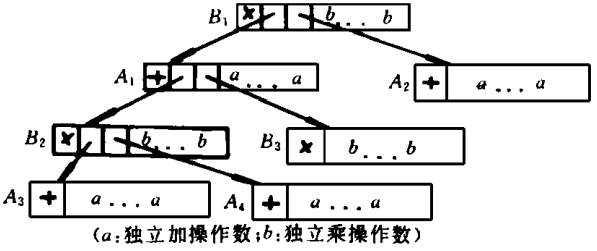


图 1　表达式的计算层次树

鉴于最优计算顺序确定的困难,我们引入一些规则以简化求解规模。按常规的计算次序和实现的方便性,对计算树的计算提出层次计算和局部计算两个准则:

定义 **3**　层次计算:指对计算树,先算低层(儿子)的表达式,只有当低层的某一类简单表达式全部计算完后,才算高层(父)的同类简单表达式,即按后根次序周游计算树。

定义 **4**　局部计算:指在层次计算次序下,对同一层的计算子树,先算一个子树的表达式,只有当此子树的某一类简单表达式全部计算完后,才算下一子树的同类简单表达式,即按宽度周游同层计算子树。

因 YH-F2 无后备寄存器,计算一个复杂表达式可能涉及到中间结果的暂存。YH-F2 的核指令系统决定了复杂表达式的计算按"引导核→(中间核)* →结束核"模式进行,只在结束核才能存结果到内存。引导核计算的并行度最大只有 50%(两拍最多作两次计算),而中间核的并行度最大可达 100%(一拍并行作两次计算),因此我们在不降低中间核的计算效率的同时,也要尽量减少引导次数。这导致了如何拆分表达式的问题。

3　算术表达式的拆分

在描述表达式的拆分算法前首先给出几个定义:

定义 **5**　标记层次树:节点带有"*level*"标记的计算层次树。节点"n"的 *level* 标记的计算规则为:节点"n"为叶节点则 $level(n) = 1$;否则 $level(n) = \max\limits_{i \in son(n)}(level(i)) + 1$.

定义 **6**　嵌套层次树:为一个标记层次树,其任一非叶节点最多只有一个子节点的 $level >= 2$.

定义 **7**　N 级层次树:指根节点的 *level* 等于 N 的嵌套层次树。

定义 **8**　一次计算:指没有中间结果保存的一个计算序列,即由一个引导核若干中间核和一个结束核组成的计算序列。

定义 **9**　主要子树:节点 n 的主要子树指 n 的子节点中 *level* 最大的子节点为根的子树,同时该子节点又称为主要子节点。

定理 **1**　二级层次树的计算可以由一次计算完成。

证明　二级层次树有如图 2 所示结构(考虑加、乘的对偶性,不妨设父节点为加运算)

计算策略:初始化 $a \rightarrow SN$, $b \times b \rightarrow PN$(引导核);然后加法处理机计算 $S = SN + a$,乘法处理机同时并行计算 $P = PN \times b$(中间核);当第一个乘法子表达式计算结束,作运算 $S = SN + PN$,同时第二乘法子表达式的操作数 $b \rightarrow P$,乘法处理机开始计算第二个乘法子表达式;当加法表达式计算结束,作 $S = SN$,而乘法处理机继续作乘法运算;最后表达式

计算结束, 其结果在 S 寄存器中(可以送内存 HSP) 。显然没有中间结果的保存, 又保持了加、乘并行计算的特点, 是一次计算完成。上述的计算方式符合 YH- F2 系统结构, 且有核指令实现。

定理 2　嵌套层次树的计算可以由一次计算完成。

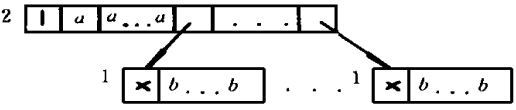
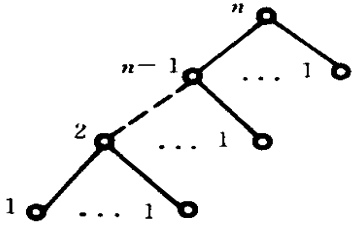


图 2　二级层次树(节点左侧数字为 "level")

证明　嵌套层次树实际上是由一系列二级层次树级联构成(图 3 所示) , 因此可得计算策略: 首先从最底层的二级层次树开始, 按定理 1 的计算策略进行计算; 当一级层次树(设为加法树) 计算结束后, 结果在 S 寄存器中, 因为相邻两级的算符必相异, 故上级的算符必为乘法, 该求和结果必为上级表达式的乘法操作数, 故作 $SN \rightarrow PN$ 切换到上级表达式的计算, 既不必存中间结果, 又无需再用引导核初始化; 这样对上一级子树计算(乘法树) 又可按一级层次树的计算策略进行, 这样交错继续直至结束, 可以实现无中间结果保存的计算。

定理 2 实际上提出了复杂表达式的一个拆分计算的策略, 即将复杂表达式按嵌套层次树进行拆分, 然后逐个计算嵌套层次树。这样此算法具有二级层次树内的并行计算和二级层次树间的计算结果耦合的优越性。于是可得复杂表达式的拆分策略: 对复杂表达式构造层次树结构, 如果某一节点有 $level >= 2$ 的非主要子节点, 则将其子节点的子树拆开成另一独立的计算树, 同时生成一暂态中间变量, 作该子计算树的输出结果和父计算树的输入, 其拆分策略可借助堆栈实现。



(符号 "°" 代表一个简单表达式)

图 3　嵌套层次树

拆分算法:

(1) 将复杂表达式构成标记层次树, 将根指针压入堆栈;

(2) 检查栈是否为空, 若为空则结束;

(3) 弹出栈顶指针, 判所指是否为嵌套层次树, 是则转代码生成, 再转(2) ; 否则转(4) ;

(4) 拆分标记层次树, 使父树为嵌套层次树, 并把父树和子树根节点压入堆栈, 转(3) 。

图 4 举例描述此算法。

当将一个复杂表达式拆为几个计算序列计算时, 如果引入 了相邻的数据相关而导致 NOP 核的插入, 也会降低实现效率。这个问题由以下定理解决。

定理 3　YH- F2 的一个引导核最多只能计算一个二级嵌套层次树。

证明　因为 YH- F2 的引导核最多能读入三个操作数, 进行两次计算, 而一个二级嵌套层次树至少有两次运算, 故 YH- F2 在引导阶段最多计算一个二级嵌套层次树。

定理 4　按上述拆分策略生成的子计算树间, 必可不必插入 NOP 核。

证明　因为在上述的拆分策略下拆分的子树至少为两级嵌套层次树, 而且在栈顶上的嵌套层次树就是保留的子树(至少也是两级子树) , 由定理 3 知引导核只能在保留子树上计算, 故而不会读拆分子树的输出结果, 不存在上核输出作下核输入的情况; 当栈顶上的嵌套层次树计算完, 并弹出堆栈, 对新的栈顶亦如此, 直至栈空都不必插入 NOP 核。

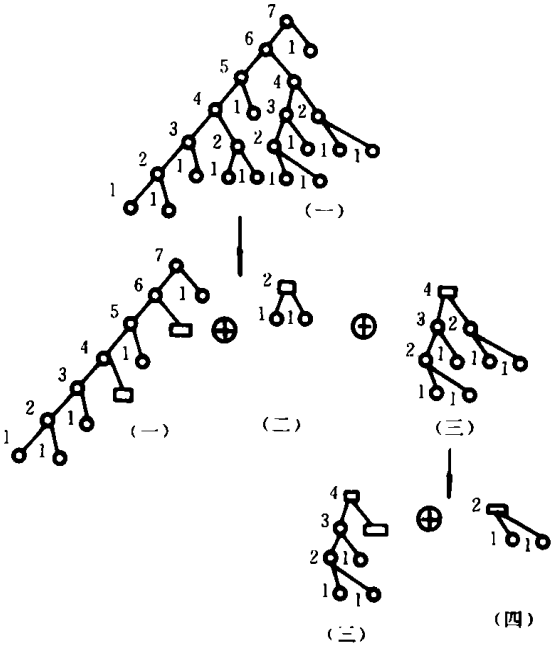


图 4　计算树的拆分示例

限于篇幅, 实现算法略。

参考文献

1　Beatty J C. An Axiomatic Approch to Code Optimization for Expression. J. ACM, 1972, 19(4): 613～640

2　Ramamoorthy R V, Juongh park, Li H F. Compilation Techniques for Recognition of Parallel Processable Tasks in Arithmetic Expressions. IEEE Trans on Computers, 1973, c- 22(11): 986～998

3　Bruno J L, Lassagne T. The Generation of Optimal Code for Stack Matchines. J. ACM, 1975, 22(3): 382～396

4　Aho A V, Johnson S C. Optimal Code Generation for Expression Trees. J. ACM, 1976, 23(3): 488～501

5　ILAN BAR－ON, UZI Vishkin. Optimal Parallel Generation of a Computation Tree Form. ACM Trans on Prog, lang, sys, 1985, 7(2): 348～357

6　Garey M R, Johnson D S. 张立昂等译. 计算机和难解性—NP 完全性理论导引. 北京: 科学出版社, 1987

7　戴金海. 同步异构并行机系统的编译技术. 系统仿真学报, 1994, 6(1): 10～17

8　梁加红. YFSIM 核库系统设计与实现. 系统仿真学报, 1994, 6(1): 18～23

(责任编辑　张　静)