国防科技大学学报
JOURNAL OF NATIONAL UNIVERSITY OF DEFENSE TECHNOLOGY
第13卷第4期 1991年12月 Vol. 13 No. 4

一种简易高效的加速措施
——移位常数

杨 桃 栏
（电子计算机系）

摘 要 本文介绍在不增加硬件额外开销的情况下，提高巨型机“取操作数”速度的一种基于直接数指令的移位常数方法。

关键词 移位常数，取操作数，加速比
分类号 TP333.1

1 引 言

巨型机（Super Computer）是现代科技迅速发展的标志，它又是现代科技继续高速发展的重要工具。人们费尽心机采取了许多方法来达到高速的目的，如在文献[1]～[5]中所论及的在程序语言、编译、优化方面的工作，均属此列。

取得高速的基础在于硬件，如主频和结构；但在同样的硬件条件下，人们所实际获得的速度，却可以因软件水平的不同而相差很大。本文所介绍的“移位常数”技术，不花费硬件的任何额外代价而使“取操作数”运算提高许多倍。

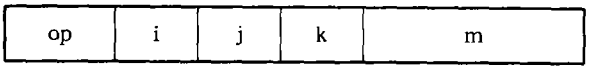
高速的CPU操作与相对慢速的存贮器访问一直是一个突出的矛盾。CPU操作的速度比存贮器访问速度高出一个数量级。人们为解决这一矛盾作出了许多努力。如YH-1，CRAY系列的机器中采用“直接数指令”，直接数指令的操作速度比访问存贮器指令的速度高出十多倍。但直接数指令所能指出的操作数范围是有限的，它仅能指代“整型数据[6]。”

本文介绍的移位常数方法便是利用直接数指令的现成结构，构造出一批不用存贮器的“浮点数”取数装置，从而在一定条件下，大大加速对某一较大范围内的浮点数的取数速度。

2 方 法

定义1 设指令形式为

* 1991年1月5日收稿



其中各字母分别表示各域的名称兼指其域的宽度（即某一常数的名称，下同）。

我们称 jkm 域为直接数域。

定义2　设浮点数据的表示形式为

其中 S 为符号位（1 bit），E 和 C 分别表示阶码和尾数部分。

就长度而言，op+i+j+k=S+E.

我们称 $C_1=(j+K+m)-(S+E)$ 为移位尾数域。

显然，移位常数的阶码域也为 E，不妨令阶码值为 E1，$E1=2^E-1$（此为移码值）。

定义3　任何一个数 C2，若满足关系式

$$C2=\sum_{i\leqslant c1}2^{-i}$$

则称为移位尾数。

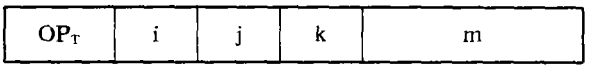
定义4　如果某浮点数 F，落于如下范围

$$F\in(-2^{E1}C2,\ 2^{E1}C2)$$

则称它为移位可表示的浮点数，简称移位数。

步骤：

对于任何一个移位数，产生/使用如下两条指令，便实现了取操作数的操作：



其中 OP_T 为直接数传送操作；OP_{RS} 为右移位操作；移位的位数 m1=wL−(S+E+C1)，其中 wL 为字长位数（bits).

3　效率和表示范围分析

我们把用移位常数代替存贮器取数所带来的加速——即其两者之比，称为绝对加速，其绝对加速比为

$$SP_A=\frac{T_m}{T_I+T_S}$$

其中，T_m 为存贮器取数时间；T_I 为直接数指令操作时间；T_s 为移位指令的执行时间。

一般地 $SP_A\geqslant 4$.

我们把整道程序因为这种改变而带来的总的加速称为相对加速，其相对加速比 SP_R 为

$$SP_R=\frac{SP_A}{(1-f)SP_A+f}$$

其中 f 为取（标量）操作数指令执行时间在程序总执行时间中占的比率，一般$0<f<$

1.

证明 设程序的总执行时间为 T_A，显然

$$SP_R = \frac{T_A}{T_A(1-f) + T_A \cdot f/SP_A} = \frac{1}{(1-f) + f/SP_A}$$

$$= \frac{SP_A}{(1-f)SP_A + f)} \qquad (\text{因 } T_A \neq 0)$$

不难看出，对目前巨型机的一个症结问题——以标量处理为主的程序运行速度不高，此法所示的改进有着重要的意义。

我们设 $SP_A \geqslant 4$，对于 f 的不同值($\in [0,1]$)，SP_R 的对应值如下表：

表 1

f	1	0.5	0.25	0.1	0.05	0
SP_R	$\geqslant 4$	$\geqslant 1.6$	$\geqslant 1.23$	$\geqslant 1.08$	$\geqslant 1.04$	$=1$

移位数在一般常数中所占的比率，我们称为它的表示范围 W，显然 $W = C_1/C$. 对于 CRAY-1这类机器，$W \doteq 1/7$，对于 CRAY Y-MP 这类机器，$W \doteq 1/3$. 而对于 CRAY-2这种机器，由于引起了五字片指令，从而 $W = 1$. 这样，从理论上讲，可以不用存贮器来存贮任何常数，对取任何常数操作数均不用同相对慢速的存贮器打交道，从而大大提高速度。当然，程序的长度也会增加很多，这就是人们常说的“以空间换时间”的一个适例。

参 考 文 献

[1] 杨桃栏. FORTRAN 语言中一种有效的数组处理结构. 电子学报，1986,(1)：1

[2] 杨桃栏. FORTRAN 的全局优化，电子学报，1988，(4)：8

[3] 杨桃栏. N 分向量查表法. 电子学报，1988，(6)：105

[4] 陈晓桦，杨桃栏. 向量化中的分段处理. 计算机学报，1989,(8)

[5] 张可军，杨桃栏. 向量块中的指令调度. 电子学报，1990,(6)

[6] CRAY Research, Inc. CFT REFERENCE MANUAL

Shifted Constant —— a Simple and Efficient Way for Speedup

Yang Taolan

(Department of Computer Science)

Abstract

A way for speeding up fetching operands of super—computers without extra hardware support is described in this paper. It is based on the shifted constant of direct data instruction. The efficiency and expressible range are also analyzed.

Key Words shifted constant, fetching operands, speedup ratio