国防科技大学学报
JOURNAL OF NATIONAL UNIVERSITY OF DEFENSE TECHNOLOGY
第15卷第1期 1993年1月 Vol. 15 No. 1

超级计算机转移机制研究*

邢二保 顾忠杰 李仲民

(电子计算机系)

摘 要 延迟转移是RISC技术特点之一。根据在类Cray－I巨型机上对标量程序的模拟可知，机器运行中约15%～20%的时间花在转移处理上。本文深入分析了i860RISC机制中延迟转移技术，研究了在类Cray－I模型机上移植延迟转移技术的实现方法。模拟结果和理论分析表明，对向量化率不高的标量程序，每条转移指令的执行时间每减少一拍可提高系统速度4～5%左右。

关键词 巨型计算机，转移指令，延迟转移，精减指令集计算机

分类号 TP338.4

1 RISC延迟转移思想

1.1 RISC技术之一：延迟转移

下面以i860为例、说明延迟转移技术的具体实现。i860系统中转移指令大致可归类成两类：无条件转移指令和条件转移指令。

(1) 无条件转移指令 (Unconditional Branch)

i860有四条无条件转移指令：br，bri，call和calli。其中br和call为直接地址转移，bri和calli为以寄存器内容为间址的间接转移。由于这几条指令的流出一定会改变指令的执行顺序，因此可在槽口插入该转移指令前面的一条指令。当转移指令流出后接着执行槽口指令，若转移指令为call和calli，则槽口指令的下一条指令的PC值保存起来作返回地址。

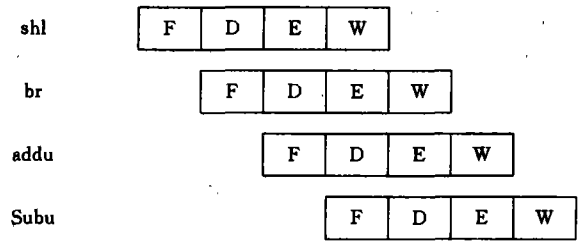
例1 无条件转移指令br

```
        shl
        br      label
        addu                ; 槽口指令
        xor
        ⋮
Label:  Subu
```

* 1992年4月18日收稿

and

指令的执行序列为：



每条指令的执行都可以看成为四个步骤：取指 F（fetch），译码 D（decode），执行 E（Execute），和写结果 W（write）。

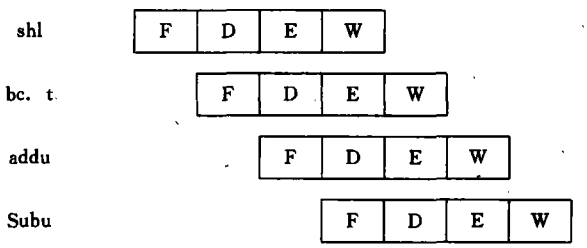
（2）条件转移换指令（Conditional Branch）

除了无条件转换指令外，i860 系统还含有条件转移指令，条件转移指令的执行要依靠条件码的状态。条件成立则转换到新的位置，否则顺序执行。i860 在处理条件转移指令时将条件转移分成两类：延迟条件转移指令和无延迟条件转移指令。若转移条件成功的概率大，则适合用延迟条件转移。反之，则选用无延迟条件转移。

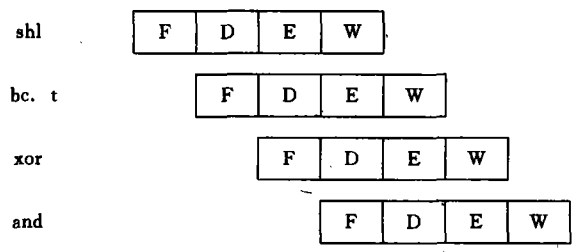
例 2 有延迟条件转移指令 bc. t

```
        shl
        bc. t     label
        addu          ; 槽口指令
        xor
        and
        ⋮
label:  Subu
```

若转移条件成功，则程序执行序列为：



若转移条件不成功，则程序执行序列为：



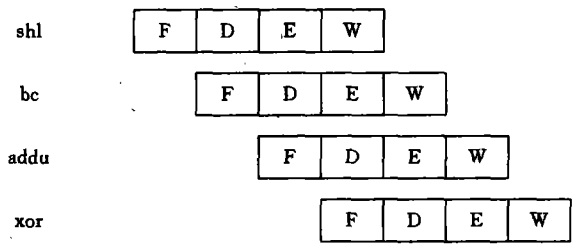
这说明，对于延迟条件转移指令，若转移条件成功则执行槽口指令，之后转移新的入口

点去执行。若转移条件不成功，则程序要顺序执行，放在槽口的指令要挤死。

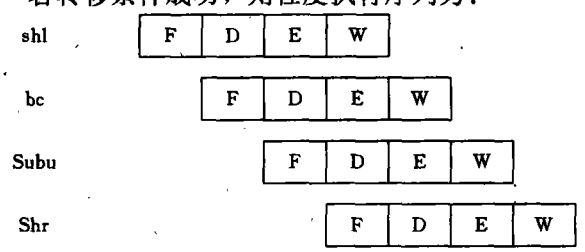
例 3 无延迟条件转移指令 bc

```
        shl
        bc      label
        addu
        xor
        and
        ⋮
label:  Subu
        shr
```

若转移条件不成功，则程序执行序列为：



若转移条件成功，则程度执行序列为：



这说明，对于无延迟条件转移，若转移条件不成功，则顺序执行。反之，若转移条件成功，则挤死槽口指令，因为此时的槽口指令非人为安排的，而是顺序执行的下一条指令。

1.2 延迟转移思想给我们的启示

从 i860 的延迟转移思想可以看出，即使其转移指令只花 2 拍，也要在槽口插入一条指令而不白白浪费节拍时间。而在类 Cray－1 巨型机中一条转移指令的执行一般要花 5 拍以上（条件转移且不成功例外）。在类 Cray－I 上模拟表明，转移指令的执行时间约占整个机器执行时间的 10%～20%。如果在每条转移指令为插入槽口指令，那么相当于减少了转移指令的执行时间。基于这种思想，下文将深入研究一种类 Cray－I 巨型机转移机制的实现原理等。

2 巨型机转移机制研究

2.1 模型机 CPU 体系结构

CPU 体系结构如图 1，其中：MRC ——存贮访问控制器，V ——向量寄存器；S ——标量寄存器；A ——地址寄存器，LM ——局部存贮器，IB ——指令控制器，P ——程序

计数器，IC ——指令控制器，VFU ——向量功能部件，FFU ——浮点功能部件，SFU ——标量功能部件，AFU ——地址功能部件

2.2 模型机转移机制的实现原理

模型机有两种类型转移指令：无条件转移指令和条件转移指令。

(1) 无条件转移指令

J1 LM，exp：无条件转移到 LM 的 exp＝m 单元内容；

J2 exp：无条件转移到 exp＝m n 主存单元内容；

R exp1，LM，exp2：返回转移到 exp1＝m n，返回地址放在 LM 的 exp2＝jk 单元。

无条件转移指令的处理过程如图 2 所示。

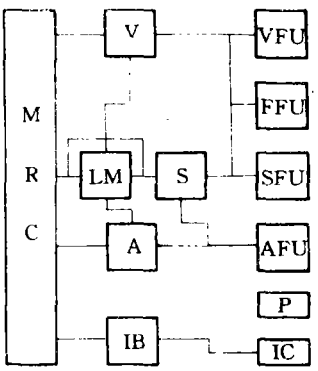


图 1 模型机 CPU 体系结构

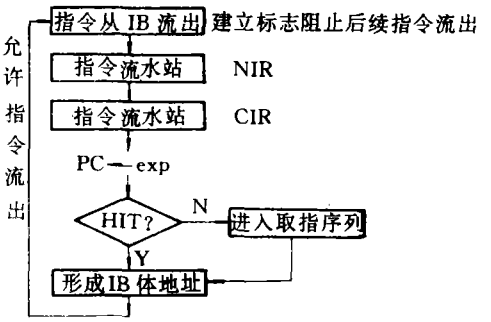


图 2 无条件转移指令执行过程

(2) 条件转移指令 JXX

JAZ (JAN，JAP，JAM) exp：(A0) ＝0 (≠0，≥0，＜0) 转移到 exp＝m n，否则作 PASS。

JSZ (JSN，JSP，JSM) exp：(S0) ＝0，(≠0，≥0，＜0) 转移到 exp＝m n，否则作 PASS 处理。

条件转移指令的执行过程如图 3 所示。

由上可以看出，对于无条件转移指令，指令从 IB 流出到目标指令进 IB 输出站，至少需要 5 拍以上。对于条件转移指令，若转移成功，则从条件转移指令从 CIR 流出到目标指令进到 CIR 也至少花费 5 拍以上。因此，我们可以在转换指令后放几条槽口指令使得转移指令可和槽口指令并发执行。

2.3 巨型机延迟转移机制的实现方法

根据巨型机的两种类型转移指令，可以指出巨型机的延迟转移机制如下：

(1) 对于无条件转移指令

J2 exp 指令从 IB 流出时可马上进入转移序列，这样 J2 exp 指令执行降至 3 拍，可在槽口插入 2 条非转移指令。

R exp1，LM，exp2 指令从 IB 流出也可马上进入转移序列。但由于 R 指令执行时要把返回地址放在 LM 的 exp2＝jk 单元。因此，要设置两级暂存器，把返回地址流水寄存在暂存器中，在相当于 R 指令从 CIR 流出时打入 LM. 因此 R 指令各槽口可放 2 条非转

移指令。

J1 LM，exp 指令涉及到从 LM 中取返回地址作 PC 值。其执行时间多于 5 拍，因此可在槽口放 5 条非转移指令。

(2) 对于条件转移指令

条件转移指令一般用于支持循环程序结构和条件分支程序结构。

例 4 循环程序结构

```
Label:  A
        B      循环体
        ⋮
        J  <Condition>  Label
        C
```

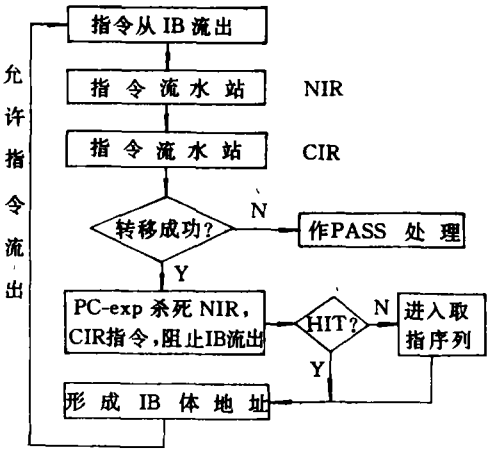


图 3 条件转移指令执行过程

对于这样的循环程序结构，转移成功的可能性很大。因此可在循环体中选择最多 4 条与转移条件无关的非转移指令进槽口。设 A 包含了这样的指令，则编译后程序的结构为：

```
        A
Label:  B
        ⋮
        J <Condition>  Label
        A          ；槽口指令
        C
```

如果转移成功，则执行槽口指令，否则杀死 J 之后的槽口指令 A.

条件转移的另一种应用是条件分支。

例 5 条件分支程序结构

```
        J  <Condition>    Label
        A
        ⋮
Label:  B
```

这种情况编译时不改变指令序列。转移不成功时顺序执行指令 A。反之杀死 A 而转去执行指令 B。此间 IB 给出 PASS 指令。之所以不加槽口指令是因为分支结构的转移条件成功概率小，且不易找到进槽口指令。

2.4 实现中的几个问题

(1)对于无条件转移指令，compiler 寻找槽口指令的规则是在无条件转移指令之上选择地址连续的非转移指令，对于条件转移指令且是循环程序结构，可选择与转移条件无关的非转移指令进槽口。如果 compiler 寻找不到这样的指令也不补 PASS 而由硬件自动给出。

(2)对所选择的槽口指令，compiler 要从转移指令操作码中空闲的场中选择三位作计

数位告之硬件进槽口指令的数目。如可用 I 场作计数位，compiler 把进槽口指令数目编码填写到计数位，硬件相应也设有一个计数器。当转移指令从 IB 流出时计数位的值打入计数器中，此后每流出一条槽口指令计数器减 1，直到计数器为 0，若此时 IB 指令无效，则给出 PASS。

(3) 换道处理。由于槽口指令的执行，使得有可能在转移期间出现现行指令寄存器 CIR 指令仍有效。这样换道机制无法判别是否处于转移状态。为此 IB 增设可中断信号$\overline{\text{PINT}}$，每拍发

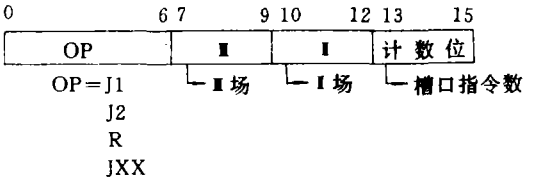


图 4

$\overline{\text{PINT}} = (\text{IBJ1} + \text{IBJ2} + \text{IBR} + \text{IBJXX}) \cdot \overline{\text{ENDBR}}$，式中，IBJ1，IBJ2，IBR，IBJXX 分别表示 IB 出口指令译码为 J1，J2，R 和 JXX 指令，ENDBR 为任一转移指令结束信号。开始换道时须处理完 NIR 和 CIR 中指令，换道区保存的是 IB 出口指令地址。

(4) 转移处理。R exp1，LM，exp2 从 IB 输出后要把返回地址（槽口指令的下条指令地址）保存到 LM 中。由于槽口指令的数目不定，每条指令的字片长度也不等，因此硬件无法知道精确的返回地址，只能

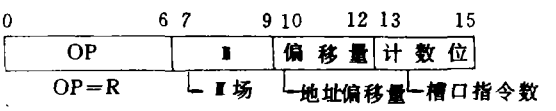


图 5

在 compiler 选择槽口指令时把槽口指令的地址空间偏移量编码放在 R 指令不用的场中，如可把偏移量编码放在Ⅱ场中(图 5)。这样当 R 指令从 IB 流出后硬件自动把下条指令地址与偏移量相加作返回地址存放在暂存器中，相当于 R 从 CIR 流出时打入 LM 中。

3 性能分析与结束语

定义 T_{Nbr}——每条非转移指令平均执行时间；T_{br}——每条转移指令平均执行时间；T_{br}——转移指令的执行频度；T_{ave}——平均每条指令执行时间。

则
$$T_{ave} = T_{Nbr}(1 - P_{br}) + T_{br} \cdot P_{br} \tag{1}$$

若采用延迟转移技术使 T_{br}相当于减少到平均 i 拍 ($i \leqslant T_{br}$)，则
$$S_P(i) = T_{Nbr}(1 - P_{br}) + i \cdot P_{br} \tag{2}$$

定义 $S_P(i)$ 为 $T_{br}=i$ 时系统加速比

$$S_P(i) = \frac{T_{ave}}{T_{ave}(i)} = \frac{T_{ave}}{T_{ave} - P_{br} \cdot T_{br} + P_{br} \cdot i}$$
$$= \frac{T_{ave}}{T_{ave} - P_{br}(T_{br} - i)}$$
$$= \frac{1}{1 - \frac{P_{br}}{T_{ave}}(T_{br} - i)} \tag{3}$$

在类 Cray－1 上模拟 20 道标量程序，统计结果如表 1 所示。从表中可以看出，P_{br}最小为 0.06814，最大为 0.17369，平均值为 0.13917，将这些典型值代入 (3) 式可以得出 $S_P(i)$ ～i 之间的关系。

表 1　标准程序模拟结果

NO	程序名	指令执行条数	指令执行拍数	br 指令执行拍数	br 指令执行频度
1	PNVA1	100000	297212	40935	0.08187
2	NAU1	100000	334672	37416	0.07503
3	HNOV	100000	223102	69232	0.13846
4	P331	100000	448541	81460	0.16292
5	XJRT2	100000	341829	75333	0.15967
6	SSS	100000	265006	34069	0.06814*
7	P1	100000	366374	74383	0.14877
8	P2	100000	366017	77619	0.15524
9	P3	100000	372421	77241	0.15448
10	P4	100000	374904	70891	0.14178
11	P5	100000	429695	80291	0.16058
12	P6	100000	429362	86844	0.17369*
13	P7	10000	349556	70781	0.14156
14	P8	100000	357382	73547	0.14709
15	P9	100000	360291	76119	0.15234
16	P10	100000	227624	42158	0.08432
17	P11	100000	395031	74081	0.14816
18	P12	100000	426321	83796	0.16759
19	P13	100000	404987	82572	0.16514
20	P14	100000	409435	82785	0.16557
average		100000	358988	69578	0.13917

由图 6 可知，T_{br}每减小 1 拍可提高系统速度 4～5%左右。在类 Cray－I 巨型机上运行标量标准子程序 GAVSS 表明，转移指令的执行时间减少一拍，GAVSS 的运行时间由原来的 20.98 秒减少到 19.91 秒，提高速度 5.1%。这证实我们给出结论的可信性。

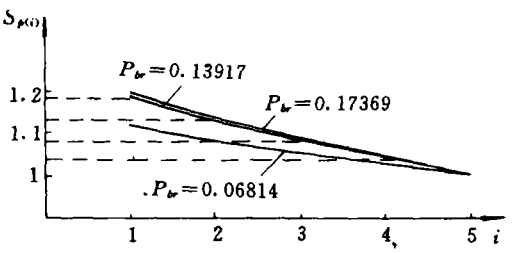


图 6　S_P (i) ～i 的关系曲线

参 考 文 献

1　Neal Margulis. i860™ Microprocessor Architecture. Intel Osborne McGraw－Hill, 1990

2　Cray Research Inc. Cray－I Computer System Hardware Reference Manual. Bloo mington, Minn. 1977

A Study on Supercomputer Branch Mechanism

Xing Erbao　Gu　Zhongjie　Li Zhongmin

(Department of Computer Science)

(下转第 50 页)

The UNIX Logical Disk Implementation for Supercomputer Environment

Ni Jikun

(Department of Computer Science)

Abstract

The paper extends the concept of logical device of UNIX and introduces the supercomputer logical disk. The method for implementing the ligical disk is described according to the given architecture of supercomputer system then. The concept of logical disk makes it possible that the size of file system is independent of the capacity of the physical disk and that the logical disk supports the super file or super file system.

Key words operating system, logical device, UXIX, file system, logical disk

---

(上接第 43 页)

Abstract

Branch-delay is one of the RISC(Reduced Instruction Set Computer)technology characters. According to our simulation to scalar programs on Cray-1-Like supercomputer, about 10～20%CPU time is occupied by branch inst. In this paper, the implementation method of branch-delay technology in i860 microprocessor is deeply analyzed. The method of introducing branch-delay technology into Cray-1-Like prototype is studied. Simulation results and theoretical analysis show that reducing one cp execute time for each branch inst will increase system pertormance by 4～5 percent.

Key words supercomputer, branch Inst, Branch-delay, RISC