32位嵌入式处理器S698的SPARC V8指令集

龚永红[1]，梅卫平[1]，蒋晓华[1]，唐芳福[1]，黄琳[2]，颜军[1]
(1. 欧比特 (珠海) 软件工程有限公司，广东 珠海 519080；
2. 中山大学理工学院，广东 广州 510275)

摘 要：以欧比特 (珠海) 软件工程有限公司的S698嵌入式微处理器为对象介绍了SPARC V8指令集及其指令格式，给出了SPARC V8指令集的寻址方式和基本操作指令的具体使用方法，同时给出了部分SPARC汇编语言的编程语法。

关键词：嵌入式；SPARC V8；窗口寄存器

0 引言

基于SPARC架构的微处理器是基于精简指令集计算机RISC (Reduced Instruction Set Computer) 体系结构的计算机系统。它具有良好的扩展性能。1987年，SUN和TI公司合作开发了全球第一款SPARC架构微处理器；2003年，欧比特 (珠海) 软件工程公司研制出中国国内第一款具有自主知识产权的SPARC架构32位嵌入式处理器S698。

目前SPARC架构微处理器系列芯片已被广泛应用于航空、通信以及各种嵌入式应用领域，可以说，SPARC架构在32位处理器领域占据着重要的地位。

随着SPARC构架处理器的广泛应用，掌握面向SPARC的嵌入式系统开发也就成为当前研究的热点问题。

虽然嵌入式应用中的大多数应用程序都可以采用汇编语言、C或C++等高级语言编程，也可以用汇编与C语言混合编程来进行开发，但汇编语言编程毕竟是编译效率最高、最直接的编程方法，而指令集则是汇编语言程序设计的基础。实际上，在基于SPARC的嵌入式软件开发中，即便是大部分程序用高级语言完成，但系统的引导，启动代码仍必须用汇瑞语言来编写。由于汇编语言编程是在指令集的基础上来考虑程序设计的，因此本文将对SPARC指令集进行介绍。

收稿日期：2007-06-07

1 SPARC V8指令集的特点

SPARC架构经过近20年的发展，已经形成了多个版本，目前最新的版本仍是V10，但常用的版本是V5、V7、V8，因为SPARC指令集具有向下兼容性，本文就以V8版本为例介绍SPARC指令集，该指令集的特点如下：

(1) 定长的指令

SPARC V8指令集中的每条指令的大小均相同。所有指令只有三种格式，它们的位数都是32位。

(2) 只有LOAD和STRORE指令访问存储器

SPARC V8指令集中，对存储器的访问只限定在LOAD和STRORE的两个指令上。相对INTEL处理器来说，这种汇编语言相对比较规范，可减少程序设计人员在编写汇编程序时出错。

(3) 可扩展的精简寻址方式

SPARC V8指令集中只有一些可以快速处理的寻址方式 (如寄存器间接寻址和相对寻址方式)。使用时也可以扩展它的寻址方式，同时支持索引和寄存器寻址。

(4) 指令流水线

流水线就像许多产品在不同点同时运行的装配线。在基于SPARC V8架构的处理器中，在一条指令执行的同时，下一条就可进行译码并装载

其操作数，同时，再下一条指令也开始取指令。通过重叠这些操作，可以使每一条指令通过三个周期来取指、译码和执行，但是CPU的每一个时钟周期又都执行一条指令。这一重要特性增加了系统的吞吐量 (throughput)。

(5) 大量的寄存器

SPARC V8架构处理器拥有大量的寄存器，这使得CPU内部可以存放许多操作数。当需要操作数时，CPU只需从寄存器中取得，而不需要从内存中取得，从而大大减少了访存的次数，同时寄存器也可以高效地向子程序传递参数，并通过寄存器窗口来完成。

(6) 延时载入和分支

SPARC V8架构使用延时载入 (delayed loads) 和延时分支 (delayed branches) 方式来避免时间的浪费。其指令流水线利用延时载入使得CPU在这一段会被浪费的时间内做有用的工作。

(7) 指令的预测执行

程序中的一些指令根据实际情况可以不执行。例如一条条件跳转指令，如果转移成功，跳转到的指令将被执行；否则它将不执行。而利用SPARC的预测执行 (speculative execution) 特点可使CPU执行某指令但不存结果。此后如果指令被执行，则存储结果；否则，将结果丢弃。

2 SPARC V8 指令格式

SPARC V8 架构处理器具有精简的指令格式，SPARC V8中所有指令都是32位宽和以32为分界对齐排列的。该指令集只有3种基本指令格式，并且只有load和store指令可访问memory和I/O，因而编程更加简单高效。

SPARC V8处理器中的格式是严格的4字节指令，每条指令都是4个字节。SPARC V8指令集是以32位二进制编码的方式给出的，共有三种主要的32位二进制编码方式。众所周知，对于一条汇编指令语句来说，通常可由操作码和操作数两个域组成，而其中操作数又有三种可能：无操作数(如格式1)、有一个操作数 (如格式2)、有两个操作数 (如格式3)。SPARC中的大部分操作数都是三个一组的。这些指令的格式如图1所示。

由图1可知，SPARC V8指令集中的大部分指令的编码格式都是第三种。其中，Op、op2、op3为操作码，Opf为浮点指令操作码，rs1、rs为通用寄存器地址 (源操作数)，rd为目的寄存器，例如：Simm为扩展符号立即数 (i=0时，第二操作数在rs2中；i=1时，第二操作数为Simm)，disp (displacement) 为位移量 (disp30表示位移量为30位)，con (condition) 为条件码 (为分支指令的条件选择)。

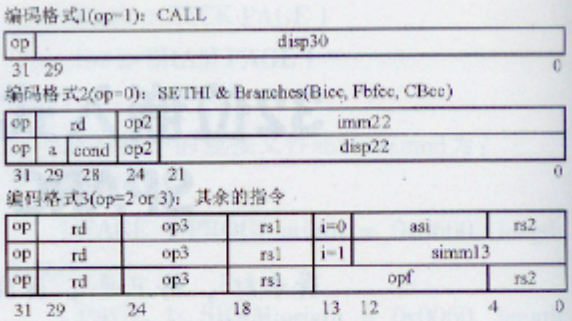

图1 SPARC的指令格式

3 SPARC V8 寻址方式

寻址方式是根据指令编码中给出的地址码字段来寻找真实操作数的方式。SPARC V8架构支持的基本寻址方式很简单，其存储器地址的给出可以是" register+register"，也可以是" register+immediate"。SPARC28的主要内存寻址方式有三种：立即寻址、寄存器寻址、寄存器间接寻址。但所有的寻址方式都有一个共性，它们的有效地址能够在一个时钟周期内计算出来。下面分别介绍每种寻址方式的具体用法。

3.1 立即寻址

立即寻址也称为立即数寻址，这是一种特殊的寻址方式。操作数可直接通过指令给出，数据就包含在指令的32位编码中，只要取出指令就可在指令执行时得到立即操作数。这里以下面的两条指令为例来说明：

add %o1，1，%o1 ！%o1←%o1+1

and %o2，0x0f，%o3 ！%o3←%o2 AND "0x0f"

在这两条指令中，指令1是将寄存器%o1的内容加1，然后将结果放回%o1中，而指令2则是将%o2的32位值与0x0F相'与'后，将结果送到%o3中去。要注意的是，在SPARC V8体系中，

只有第二个操作数才会用到立即寻址方式。

SPARC V8处理器识别立即数可结合SPARC V8指令编码格式来说明对于如下指令：

ADD rs1 rs2 rd

在SPARC V8处理器中，Simm就是扩展符号立即数；“i”这一位起着决定性的作用。i用于为(整数)算术运算和load/store指令选择第二个运算器操作数。如果i=0，操作数即为寄存器r[rs2]；如果i=1，操作数则是有符号立即数simm13，符号扩展从13到32 bit。

在SPARC V8体系中，立即数的取值是有范围限制的。它依不同的指令类型而不同，并可用于不同的寻址模式中。如imm7的取值范围是在-64~127之间的立即数(用7位表示，有符号或无符号)，常用于(trap)跟踪类型指令的偏址寻址方式中。而simm13d的取值范围则是在-4096~4095之间的立即数(用13位表示，有符号)，常用于算术运算指令或load/store指令的第二个运算器操作数。

3.2 寄存器寻址

SPARC V8指令集中除了立即数寻址外，还有一种常用的寄存器直接寻址方式。寄存器寻址是利用寄存器中的数值作为操作数，指令中的地址码给出的是寄存器编号。例如：

Add %l0, %l1, %l2 ! %l2←%l0+%l1

该指令执行的是将2个寄存器(%l0和%l1)的内容相加，然后将结果放入第3个寄存器%l2中。但在编写指令时必须注意写操作数的顺序：第1个是第1寄存器，然后是第2操作数寄存器，最后才是结果寄存器。

3.3 寄存器间接寻址

SPARC V8指令集中的第三种寻址方式是寄存器间接寻址。这种寻址方式是用一个寄存器的值来作指针作用。在基址加变址的寻址方式中，它作为基址寄存器来存放基址地址。也就是说，作为存储器地址，它可在指定的寄存器中存放有效地址，而操作数则放在存储单元中。这个寄存器用“[ ]”括起来表示内容。这种寻址方式的来具体实现可参考如下两条指令：

ld [%o2], %f0 ! %f0←mem [%o2]

st %fsr, [%o0] ! mem [%o0]←%fsr

在这两条指令中，第1条指令可将寄存器%o2指向的地址存储器单元的内容加载到寄存器%f0中。而使用第2条指令可将寄存器%fsr的内容存储到寄存器%o0指向的地址存储单元之中。

另外，SPARC V8架构之所以可以扩展它的寻址方式，是因为它包含的一个伪寄存器%g0的值总是0，这使得程序员要以索引方式定义绝对地址。索引方式计算地址是通过用户定义的地址与用户定义的寄存器的值相加。如果用户定义了寄存器%g0，由于其值为0，所以有效地址就是指令代码中的用户定义的绝对地址。

4 SPARC V8基本操作指令的使用

SPARC V8指令集可分为6类共72条基本操作指令，其中包括31条算术运算/逻辑运算/移位指令、22条LOAD/STORE指令、5条控制转移指令、8条读/写专用寄存器指令、1条浮点运算指令和1条协处理器指令。

4.1 数据处理指令

SPARC V8的数据处理指令主要完成寄存器中数据的算术和逻辑运算操作。使用SPARC V8数据处理指令的基本原则如下：

◇ 所有操作数都是32位宽或来自寄存器，也可能是在指令中定义的立即数。

◇ 算术运算和逻辑运算指令都要用到3个寄存器，大部分情况是第一个和最后一个参数是双寄存器，而第二个参数可以是寄存器或13位有符号立即数。

◇ 如果数据操作有结果，则结果为32位宽，放在一个寄存器中(有一个例外就是乘法指令产生64位结果)。

◇ %y寄存器可用在乘法和除法中。在除法中，它用于暂存被除数的高32位有效位；而在乘法中，它用于暂存乘积的高32位有效位；从事程序开发的人员应当清楚：%y寄存器既可在乘法运算中检查乘积，又可在除法指令中适当地设置操作数。

SPARC V8指令中的每个运算操作均可根据是否设置条件码被分成两条指令，也就是说，同

一个运算操作指令，一条会对%psr状态码位产生影响（带有cc后缀的指令），而另一条则不会。另外，乘法和除法又存在有符号和无符号运算之分。例如加减法指令（ADD、ADDcc、ADDXcc、SUB、SUBcc、SUBXcc），这些指令是最基本的算术运算指令。其中加法ADD和减法SUB都是不区分操作数是否有符号的，而只是简单地进行相关运算。如果非要给它们加上符号的话，SPARC V8提供了很有用的条件码，只需在其指令助记符后加上cc。如ADDcc、SUBcc。这样，运算的结果就会影响PSR相应的条件码icc位。

SPARC V8指令中的乘除法指令可用SMUL、SDIV、UMUL、UDIV来代表。SPARC V8乘法指令可将两个32位源操作数相乘以得到64位乘积，但它们也存在有符号和无符号运算之分。

另外，SPARC V8的数据处理指令中，还有逻辑运算指令（AND、ANDN、OR、ORN、XOR、XNOR），它们分别完成“与”、“或”、“异或”的按位操作。

4.2 load/store (memory) 指令

SPARC V8处理器是Load/Store型的，它对数据的操作是将数据从存储器加载到片内寄存器中进行处理，再将处理完成后的结果经寄存器存回到存储器中，以加快对片外存储器进行数据处理的执行速度。SPARC V8的数据存取指令Load/Store是唯一用于寄存器和存储器之间进行数据传送的指令。

Load/store指令仅用于访问存储器（memory）。其指令操作可以用图2来说明。

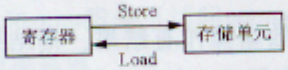

图2 Load/store指令操作示意图

由图1可见，执行Load操作是将数据从存储器加载到片内寄存器；而执行Store操作则是将数据从寄存器中放入存储器中。其基本的汇编语法格式如下：

```
opcode [%reg+const], %reg
opcode [%reg+%reg], %reg
```

4.3 控制转移指令

SPARC V8指令集中的转移指令有，有条件和无条件转移指令2种。这里的条件是由条件码寄存器提供的，而条件码标志位的变化又是可控的，它可通过不同的指令来控制的，这就是前面已经提到的带有-cc的指令，它会影响条件码。

条件码寄存器有4个位：Z (Zero)、N (Negative)、C (Carry)、V (oVerflow)。标准的算术操作（如：ADD、SUB）不会更新条件码的位。但如果换之以带有-cc后缀的特定指令（如：ADDcc、SUBcc），则会影响并更新条件码寄存器的标志位。所以，是否影响条件码寄存器要视实际情况决定。通常情况下，使用时更关心N和Z标志。

4.4 读/写专用寄存器指令

读写状态寄存器指令用于访问状态寄存器或对它们写入一个新值，这些状态寄存器可以是%y，%psr，%wim，%tbr或辅助状态寄存器asr。其中RD指令用来读专用状态寄存器，而WR指令则用来写专用状态寄存器。

4.5 浮点操作指令

浮点操作指令可执行所有的浮点运算。它们是基于浮点寄存器的操作指令。可以通过SPARC V8处理器中的浮点环境开发可靠的、高性能的、可移植的数值应用程序。

无论哪种SPARC版本都会提供一个浮点状态寄存器（FSR），该寄存器中包含有与FPU相关的状态位和控制位。

浮点操作指令支持在整形字和单、双、四精度浮点操作数之间操作。浮点运算指令可根据功能的不同分为类型转换浮点指令、浮点算术运算指令和其它浮点指令。

4.6 类型转换指令

类型转换指令包括整形与浮点型之间的类型转换指令、两浮点型之间的类型转换指令、浮点平方根计算指令和浮点算术运算指令。

5 结束语

本文分析并归纳了如何在嵌入式系统开发中使用SPARC汇编指令，并从SPARC V8指令的寻址方式以及SPARC V8基本操作指令的具体使用入手，详细介绍了SPARC V8指令集。这些汇编指令和语法对于开发嵌入式系统是十分重要的。