(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 107003853 B
(45)授权公告日 2020.12.22

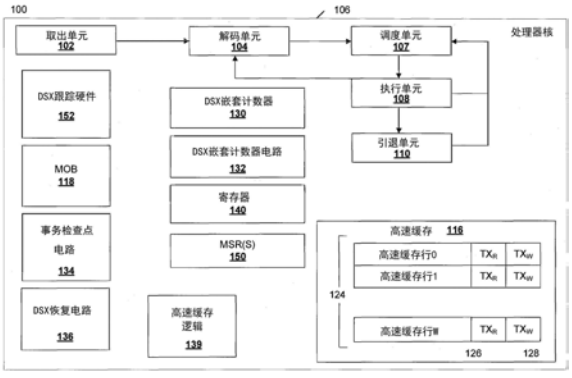
(21)申请号 201580064697.1
(22)申请日 2015.11.24
(65)同一申请的已公布的文献号
申请公布号 CN 107003853 A
(43)申请公布日 2017.08.01
(30)优先权数据
14/582,717 2014.12.24 US
(85)PCT国际申请进入国家阶段日
2017.05.26
(86)PCT国际申请的申请数据
PCT/US2015/062249 2015.11.24
(87)PCT国际申请的公布数据
WO2016/105786 EN 2016.06.30
(73)专利权人 英特尔公司
地址 美国加利福尼亚州
(72)发明人 E·乌尔德-阿迈德-瓦尔
C·J·休斯 R·瓦伦天
M·B·吉尔卡尔 H·井户
吴友峰 C·王
(74)专利代理机构 上海专利商标事务所有限公司 31100
代理人 何焜
(51)Int.Cl.
G06F 9/30(2006.01)
G06F 9/38(2006.01)
G06F 9/52(2006.01)
(56)对比文件
US 2009248984 A1,2009.10.01
CN 1940860 A,2007.04.04
CN 102725741 A,2012.10.10
从余.数字音频信号传输系统DSX的分析与制作.《实用影音技术》.1995,(第7期),第15-17页.
审查员 李中兴
权利要求书2页 说明书36页 附图41页

(54)发明名称

用于数据推测执行的系统、装置和方法

(57)摘要

描述了用于数据推测执行(DSX)的系统、方法和装置。在一些实施例中,用于执行DSX的硬件装置包括硬件解码器,用于解码指令,指令包括操作码和用于存储回退地址的一部分的操作数,以及执行硬件,用于执行经解码的指令,通过激活数据推测执行(DSX)跟踪硬件来跟踪推测存储器访问并检测DSX区域中的排序违反、并且存储所述回退地址而发起DSX区域。



CN 107003853 B

1.一种用于指令执行的装置,包括:

硬件解码器,用于解码指令,所述指令包括用于指示执行硬件用于发起数据推测扩展DSX区域的操作码和用于指示用于存储回退地址的一部分的操作数的位置的字段;以及

执行硬件,用于执行经解码的指令,通过激活DSX跟踪硬件来跟踪推测存储器访问并检测所述DSX区域中的排序违反、并且存储所述回退地址,来发起所述DSX区域,所述回退地址是程序执行为了处理错误推测而跳转到的地方,其中在DSX区域中存储被缓冲而加载不被缓冲。

2.如权利要求1所述的装置,其特征在于,所述回退地址的一部分是位移值,所述执行硬件用于将所述位移值与紧接着所述经解码的指令的指令的指令指针相加。

3.如权利要求1所述的装置,其特征在于,所述回退地址的一部分是完整地址。

4.如权利要求1-3中的任一项所述的装置,其特征在于,用于存储所述回退地址的一部分的所述操作数是立即数值。

5.如权利要求1-3中的任一项所述的装置,其特征在于,用于存储所述回退地址的一部分的所述操作数是寄存器。

6.如权利要求1-3中的任一项所述的装置,其特征在于,所述执行硬件进一步用于:

确定受限制的事务性存储器RTM事务正在发生并且处理所述RTM事务。

7.如权利要求1-3中的任一项所述的装置,进一步包括:

DSX嵌套计数器,用于存储对应于不具有对应的DSX区域结束的DSX区域开始的数量的值。

8.一种用于指令执行的方法,包括:

使用硬件解码器来解码指令,所述指令包括用于指示执行硬件用于发起数据推测扩展DSX区域的操作码和用于指示用于存储回退地址的一部分的操作数的位置的字段;以及

执行经解码的指令,通过激活DSX跟踪硬件来跟踪推测存储器访问并检测所述DSX区域中的排序违反、并且存储所述回退地址,来发起所述DSX区域,所述回退地址是程序执行为了处理错误推测而跳转到的地方,其中在DSX区域中存储被缓冲而加载不被缓冲。

9.如权利要求8所述的方法,其特征在于,所述回退地址的一部分是位移值,通过所述执行硬件将所述位移值与紧接着所述经解码的指令的指令的指令指针相加。

10.如权利要求8所述的方法,其特征在于,所述回退地址的一部分是完整地址。

11.如权利要求8-10中的任一项所述的方法,其特征在于,用于存储所述回退地址的一部分的所述操作数是立即数值。

12.如权利要求8-10中的任一项所述的方法,其特征在于,用于存储所述回退地址的一部分的所述操作数是寄存器。

13.如权利要求8-10中的任一项所述的方法,其特征在于,所述执行进一步包括:

确定受限制的事务性存储器RTM事务正在发生并且处理所述RTM事务。

14.如权利要求8-10中的任一项所述的方法,进一步包括:

存储对应于不具有对应的DSX区域结束的DSX区域开始的数量的值。

15.一种存储指令的非瞬态机器可读介质,所述指令在被机器执行时使电路被制造,所述电路包括:

硬件解码器,用于解码指令,所述指令包括用于指示执行硬件用于发起数据推测扩展

DSX区域的操作码和用于指示用于存储回退地址的一部分的操作数的位置的字段;以及

执行硬件,用于执行经解码的指令,通过激活DSX跟踪硬件来跟踪推测存储器访问并检测所述DSX区域中的排序违反、并且存储所述回退地址,来发起所述DSX区域,所述回退地址是程序执行为了处理错误推测而跳转到的地方,其中在DSX区域中存储被缓冲而加载不被缓冲。

16.如权利要求15所述的非瞬态机器可读介质,其特征在于,所述回退地址的一部分是位移值,所述执行硬件用于将所述位移值与紧接着所述经解码的指令的指令的指令指针相加。

17.如权利要求15所述的非瞬态机器可读介质,其特征在于,所述回退地址的一部分是完整地址。

18.如权利要求15-17中的任一项所述的非瞬态机器可读介质,其特征在于,用于存储所述回退地址的一部分的所述操作数是立即数值。

19.一种用于指令执行的装置,包括:

硬件解码器装置,用于解码指令,所述指令包括用于指示执行装置用于发起数据推测扩展DSX区域的操作码和用于指示用于存储回退地址的一部分的操作数的位置的字段;以及

执行装置,用于执行经解码的指令,通过激活DSX跟踪硬件来跟踪推测存储器访问并检测所述DSX区域中的排序违反、并且存储所述回退地址,来发起所述DSX区域,所述回退地址是程序执行为了处理错误推测而跳转到的地方,其中在DSX区域中存储被缓冲而加载不被缓冲。

20.如权利要求19所述的装置,其特征在于,所述回退地址的一部分是位移值,所述执行装置用于将所述位移值与紧接着所述经解码的指令的指令的指令指针相加。

21.如权利要求19所述的装置,其特征在于,所述回退地址的一部分是完整地址。

22.如权利要求19-21中的任一项所述的装置,其特征在于,用于存储所述回退地址的一部分的所述操作数是立即数值。

23.如权利要求19-21中的任一项所述的装置,其特征在于,用于存储所述回退地址的一部分的所述操作数是寄存器。

24.如权利要求19-21中的任一项所述的装置,其特征在于,所述执行装置进一步用于:

确定受限制的事务性存储器RTM事务正在发生并且处理所述RTM事务。

25.如权利要求19-21中的任一项所述的装置,进一步包括:

DSX嵌套计数器装置,用于存储对应于不具有对应的DSX区域结束的DSX区域开始的数量的值。

用于数据推测执行的系统、装置和方法

技术领域

[0001] 本发明的领域一般涉及计算机处理器架构，更具体地涉及推测执行。

背景技术

[0002] 对包括可能的交叉迭代的依赖性的循环进行向量化众所周知是困难的。这种类型的示例性循环是：

[0003]
```
for(i＝0;i<N;i++){
```

[0004]
```
    A[i]＝B[C[i]];
```

[0005]
```
    }
```

[0006] 对这个循环的不成熟的(并且不正确的)向量化将是：

[0007]
```
for (i = 0; i < N; i += SIMD_WIDTH) {
    zmm0 = vmovdqu32 &C[i]
    k1 = kxnor k1, k1
    zmm1 = vgatherdd B, zmm0, k1
    vmovdqu &A[i], zmm1
}
```

[0008] 然而，如果生成向量化版本的循环的编译器不能在先获悉关于A、B和C的地址或对齐性，则上述向量化是不安全的。

附图说明

[0009] 在所附附图中以示例方式而非限制方式说明本发明，在附图中，类似的参考标号指示类似的元件，其中：

[0010] 图1是能够在硬件中执行数据推测扩展(DSX)的处理器核的示例性框图的实施例；

[0011] 图2示出根据实施例的推测指令执行的示例；

[0012] 图3示出DSX跟踪硬件的详细实施例；

[0013] 图4示出DSX跟踪硬件执行的DSX错误推测检测的示例性方法；

[0014] 图5(A)-(B)示出DSX跟踪硬件执行的DSX错误推测检测的示例性方法；

[0015] 图6示出对用于开始DSX的指令的执行的实施例；

[0016] 图7示出YBEGIN指令格式的一些示例性实施例；

[0017] 图8示出对诸如YBEGIN指令的指令的执行的详细实施例；

[0018] 图9示出显示对诸如YBEGIN指令的指令的执行的伪代码的示例；

[0019] 图10示出对用于开始DSX的指令的执行的实施例；

[0020] 图11示出YBEGIN WITH STRIDE指令格式的一些示例性实施例；

[0021] 图12示出对诸如YBEGIN WITH STRIDE指令的指令的执行的详细实施例；

[0022] 图13示出对用于继续DSX而不结束它的指令的执行的实施例；

[0023] 图14示出YCONTINUE指令格式的一些示例性实施例；

[0024] 图15示出对诸如YCONTINUE指令的指令的执行的详细实施例；

[0025] 图16示出显示对诸如YCONTINUE指令的指令的执行的伪代码的示例；

[0026] 图17示出对用于中止DSX的指令的执行的实施例；

[0027] 图18示出YABORT指令格式的一些示例性实施例；

[0028] 图19示出对诸如YABORT指令的指令的执行的详细实施例；

[0029] 图20示出显示对诸如YABORT指令的指令的执行的伪代码的示例；

[0030] 图21示出对用于测试DSX的状态的指令的执行的实施例；

[0031] 图22示出YTEST指令格式的一些示例性实施例；

[0032] 图23示出显示对诸如YTEST指令的指令的执行的伪代码的示例；

[0033] 图24示出对用于结束DSX的指令的执行的实施例；

[0034] 图25示出YEND指令格式的一些示例性实施例；

[0035] 图26示出对诸如YEND指令的指令的执行的详细实施例；

[0036] 图27示出显示对诸如YEND指令的指令的执行的伪代码的示例；

[0037] 图28A-28B是示出根据本发明的实施例的通用向量友好指令格式及其指令模板的框图；

[0038] 图29A-D示出专用向量友好指令格式2900，其指定位置、尺寸、解释和字段的次序以及那些字段中的一些字段的值，在这个意义上向量友好指令格式2900是专用的。

[0039] 图30是根据本发明的一个实施例的寄存器架构的框图；

[0040] 图31A是示出根据本发明的实施例的示例性有序流水线以及示例性寄存器重命名的乱序发布/执行流水线两者的框图；

[0041] 图31B是示出根据本发明的实施例的要包括在处理器中的有序架构核的示例性实施例和示例性的寄存器重命名的乱序发布/执行架构核的框图；

[0042] 图32A-B示出更具体的示例性有序核架构的框图，该核将是芯片中的若干逻辑块之一(包括相同类型和/或不同类型的其他核)；

[0043] 图33是根据本发明的实施例的可具有超过一个的核、可具有集成存储器控制器、并且可具有集成图形器件的处理器的框图；

[0044] 图34示出根据本发明的实施例的系统的框图；

[0045] 图35示出根据本发明的实施例的第一更具体的示例性系统的框图；

[0046] 图36示出根据本发明的实施例的第二更具体的示例性系统的框图；

[0047] 图37示出根据本发明的实施例的SoC的框图；

[0048] 图38是根据本发明的实施例的对照使用软件指令转换器将源指令集中的二进制指令转换成目标指令集中的二进制指令的框图。

具体实施方式

[0049] 在以下描述中，陈述了多个具体细节。然而，应当理解，可不通过这些具体细节来实施本发明的实施例。在其他实例中，公知的电路、结构和技术未被详细示出，以免混淆对本描述的理解。

[0050] 说明书中对“一个实施例”、“实施例”、“示例实施例”等等的引用表明所描述的实施例可以包括特定的特征、结构或特性，但是每个实施例不一定都包括该特定的特征、结构或特性。此外，这样的短语不一定是指同一个实施例。此外，当结合实施例描述特定的特征、结构或特性时，认为结合无论是否被明确描述的其他实施例而影响此类特征、结构或特性是在本领域技术人员的知识范围之内的。

[0051] 贯穿本说明书，详述了被称为数据推测扩展(DSX)的推测执行的技术。本说明书中包括DSX硬件和支持DSX的新指令。

[0052] DSX本质上类似于受限制的事务性存储器(RTM)的实现方式，但是更简单。例如，DSX区域不需要隐含的栅栏。相反，保持正常的加载/存储排序规则。此外，DSX区域不在处理器中设置用于加载的强制原子行为的任何配置，而在RTM中，原子地处理对事务的加载和存储(当完成事务时提交)。此外，加载不被缓冲，因为它们在RTM中。然而，当不再需要推测时，立即缓冲和提交存储。取决于实施例，可以将这些存储缓冲在专用推测执行存储中或在共享寄存器或存储器位置中。在一些实施例中，推测向量化只在单个线程上发生，这意味着不需要阻止来自其他线程的干扰。

[0053] 在先前详述的向量化的循环中，出于安全将需要有动态检查。例如，保证在给定的向量迭代中写入A不与在标量循环中的在稍后的迭代中被读取的B或C中的元素重叠。下文的实施例详述通过对推测的使用来处理向量化情形。推测版本指示应当推测地执行每个循环迭代(例如，使用下文详述的指令)，并且硬件应当帮助执行地址检查。代替仅依赖于硬件负责地址检查(其要求非常昂贵的硬件)，详细的方法使用软件来提供信息以辅助硬件，实现更便宜的硬件方案而不影响执行时间或对编程器或编译器施加过多负担。

[0054] 不幸的是，在向量化情况下，可能存在排序违反。回看上文详述的标量循环示例：

[0055] for(i＝0;i<N;i++){

[0056] A[i]＝B[C[i]];

[0057] }

[0058] 在这个循环的前四个迭代期间，下列存储器操作将按照下列顺序发生：

[0059] 读取C[0]

[0060] 读取B[C[0]]

[0061] 写入A[0]

[0062] 读取C[1]

[0063] 读取B[C[1]]

[0064] 写入A[1]

[0065] 读取C[2]

[0066] 读取B[C[2]]

[0067] 写入A[2]

[0068] 读取C[3]

[0069] 读取B[C[3]]

[0070] 写入A[3]

[0071] 访问到相同数组之间的距离(按操作的数量)是三，并且这也是当它被向量化时(成为SIMD)循环中的推测存储器指令的数量。这个距离被称为“跨度”。它也是循环中的存

储器指令的数量，当循环被向量化时，在存储器指令上将执行地址检查。在一些实施例中，在循环开始处经由特殊指令将这个跨度传递到地址跟踪硬件(下文详述)。在一些实施例中，这个指令还清除地址跟踪硬件。

[0072] 本文中详述了在诸如向量化循环执行的情形中在DSX中使用的新指令(DSX存储器指令)。每个DSX存储器指令(诸如加载、存储、聚集和分散)包括在DSX期间使用的指示DSX执行内的位置(例如，被执行的循环中的位置)的操作数。在一些实施例中，操作数是立即数(例如，8位立即数)，在立即数中具有被编码的顺序的数值。在其他实施例中，操作数是存储被编码的顺序的数值的寄存器或存储器位置。

[0073] 此外，在一些实施例中，这些指令具有与其正常对应指令不同的操作码。这些指令可以是标量的或超标量的(例如，SIMD或MIMD)。以下示出这些指令中的一些的示例，其中操作码的助记符包括“S”(下文加下划线)以指示其是推测版本，并且imm8是用于指示执行的位置(例如，被执行的循环中的位置)的立即数操作数：

[0074] VMOVSDQA32zmm1{k1}{z},mV,imm8//推测SIMD加载

[0075] VMOVS xmm1,m32,imm8//推测标量加载

[0076] VSCATTERSDPS vm32z{k1},zmm1,imm8//推测分散

[0077] 当然，其他指令还可以利用详述的操作数和操作码助记符(和基本操作码)变化，诸如逻辑的(AND、OR、XOR等等)和数据操纵(加、减等等)指令。

[0078] 在上述标量示例的向量版本(假设SIMD宽度的四个紧缩数据元素)中，存储器操作的顺序是：

[0079] 读取C[0]、C[1]、C[2]、C[3]

[0080] 读取B[C[0]]、B[C[1]]、B[C[2]]、B[C[3]]

[0081] 写入A[0]、A[1]、A[2]、A[3]

[0082] 如果例如B[C[1]]与A[0]重叠，则这个顺序可能导致不正确的执行。在原始标量顺序中，对B[C[1]]的读取在对A[0]的写入之后发生，但是在向量化执行中，其在之前发生。

[0083] 为可能导致不正确的执行的循环中的操作使用推测存储器指令帮助解决这个问题。如将详述的，每个推测存储器指令通知DSX跟踪硬件(下文详述)它在循环主体内的位置：

[0084] for(i＝0;i<N;i+＝SIMD_WIDTH){

[0085] zmm0＝vmovsdqu32&C[i],0//告诉地址跟踪器这是指令0

[0086] k1＝kxnor k1,k1

[0087] zmm1＝vgathersdd B,zmm0,k1,1//告诉地址跟踪器这是指令1

[0088] vmovsdqu&A[i],zmm1,2//告诉地址跟踪器这是指令2

[0089] }

[0090] 可以将每个推测存储器操作提供的循环位置信息与跨度组合来重构标量存储器操作。随着推测存储器指令执行，DSX硬件跟踪器为每个元素计算标识符(id)(id＝序列号+跨度*SIMD操作内的元素数量)。硬件跟踪器使用序列号、计算出的id以及每个紧缩数据元素的地址和大小来确定是否存在排序违反(即，如果元素与另一个元素重叠并且被乱序地读取或写入)。

[0091] 展开包括每个向量存储器指令的各个存储器操作、为每次展开累加跨度、并且将

所得的数字分配为“ids”产生：

[0092] 读取C[0]//id＝0

[0093] 读取C[1]//id＝3

[0094] 读取C[2]//id＝6

[0095] 读取C[3]//id＝9

[0096] 读取B[C[0]]//id＝1

[0097] 读取B[C[1]]//id＝4

[0098] 读取B[C[2]]//id＝7

[0099] 读取B[C[3]]//id＝10

[0100] 写入A[0]//id＝2

[0101] 写入A[1]//id＝5

[0102] 写入A[2]//id＝8

[0103] 写入A[3]//id＝11

[0104] 按照id将上述各个存储器操作排序将重构原始标量存储器排序。

[0105] 图1是能够在硬件中执行数据推测扩展(DSX)的处理器核的示例性框图的实施例。

[0106] 处理器核106可包括取出单元102用来取出指令以供核106执行。例如，可以从L1高速缓存或存储器取出指令。核106还可包括用于解码被取出的指令(包括下文详述的指令)的解码单元104。例如，解码单元104可将被取出的指令解码为多个微操作(微op)。

[0107] 此外，核106可包括调度单元107。调度单元107可执行与存储(例如，从解码单元104接收的)经解码的指令相关联的多种操作直到指令准备好以供分派，例如，直到来自经解码的指令的操作数的所有源值都变得可用。在一个实施例中，调度单元107可将经解码的指令调度和/或发布(或分派)至一个或多个执行单元108以供执行。执行单元108可包括存储器执行单元、整数执行单元、浮点执行单元或其他执行单元。引退单元110在被执行的指令被提交后可使被执行的指令引退。在实施例中，引退这些被执行的指令会导致：通过对这些指令的执行，提交处理器状态；解除分配由这些指令使用的物理寄存器，等等。

[0108] 存储器排序缓冲器(MOB)118可包括加载缓冲器、存储缓冲器和用于存储还未加载或写回到主存储器的待决存储器操作的逻辑。在一些实施例中，MOB 118或与其类似的电路存储对DSX区域的推测存储(写入)。在多个实施例中，核可包括本地高速缓存，例如，可包括一个或多个高速缓存行124(例如高速缓存行0到W，并且由高速缓存电路139管理)的私有高速缓存(例如，高速缓存116)。在一个实施例中，高速缓存116的每个行可包括用于在核106上执行的每个线程的DSX读取位126和/或DSX写入位128。位126和128可被置位或清除以指示通过DSX存储器访问请求对相应的高速缓存行的(加载和/或存储)访问。注意，虽然在图1的实施例中，每个高速缓存行124示出为具有各自的位126和128，但其他配置也是可能的。例如，DSX读取位126(或DSX写入位128)可与高速缓存116的选择部分(例如，高速缓存116的高速缓存块或其他部分)对应。而且，位126和/或128可存储在除高速缓存116之外的位置。

[0109] 为了辅助执行DSX操作，核106可以包括DSX嵌套计数器130，用于存储对应于已经遇到的不具有匹配的DSX结束的DSX开始的数量的值。计数器130可被实现为任何类型的存储设备(例如，硬件寄存器)或存储在存储器(例如，系统存储器或高速缓存116)中的变量。核106还可包括用于更新存储在计数器130中的值的DSX嵌套计数器电路132。核106可包括

用于对核106的多个组件的状态进行检查点操作(或存储)的DSX检查点电路134,和用于通过使用其存储的或存储在诸如寄存器140的另一位置中的回退地址(例如,在给定DSX中止时)来恢复核106的多个组件的状态的DSX恢复电路136。此外,核106可包括对应于多个DSX存储器访问请求的一个或多个附加的寄存器140,例如,DSX状态和控制寄存器(DSXSR)用来存储对DSX是否活动的指示、DSX指令指针(DSXXIP)(例如,可以为指向对应的DSX的开始处(或紧邻在前)的指令的指令指针)、和/或DSX堆栈指针(DSXSP)(例如,可以为指向存储核106的一个或多个组件的多个状态的堆栈的头部的堆栈指针)。这些指针还可以是MSR 150。

[0110] DSX地址跟踪硬件152(有时简单地称为DSX跟踪硬件)跟踪推测存储器访问并且检测DSX中的排序违反。具体而言,该跟踪硬件152包括地址跟踪器,其吸收信息来重构并且然后实施原始标量存储器顺序。典型地,输入是循环主体中需要被跟踪的推测存储器指令的数量,以及这些指令中的每一个的一些信息,例如:(1)序列号,(2)指令访问的地址,以及(3)指令引起读取存储器还是写入存储器。如果两个推测存储器指令访问存储器的重叠的部分,则硬件跟踪器152使用该信息来确定存储器操作的原始标量顺序是否已经被改变。如果是,并且如果二者中的一个操作是写,则硬件触发错误推测。虽然图1在其上示出DSX跟踪硬件152,但是在一些实施例中该硬件是其他核组件的一部分。

[0111] 图2示出根据实施例的推测指令执行的示例。在201处,取出推测指令。例如,取出诸如上文详述的推测存储器指令。在一些实施例中,该指令包括指示其推测本质的操作码和用于指示DSX中的排序的操作数。排序操作数可以是立即数值或寄存器/存储器位置。

[0112] 在203处解码取出的推测指令。

[0113] 在205处作出对经解码的推测指令是否是DSX的一部分的确定。例如,上文详述的DSX状态和控制寄存器(DSXSR)中指示了DSX吗?当DSX不活动时,指令要么变成无操作(nop),要么根据实施例在207处被执行为正常的非推测指令。

[0114] 当DSX活动时,在209处推测地执行(例如,不提交)推测指令并且更新DSX跟踪硬件。

[0115] 图3示出DSX地址跟踪硬件的详细实施例。该硬件跟踪推测存储器实例。典型地,被DSX跟踪硬件分析的元素(例如,SIMD元素)被分成称为块(chunk)的部分,块的大小不超过“B”字节。

[0116] 移位电路301对块的地址(诸如开始地址)进行移位。在大多数实施例中,移位电路301执行右移。典型地,右移$\log_2 B$。被移位的地址经受散列函数单元电路303执行的散列函数。

[0117] 散列函数的输出是到散列表305的索引。如上所述,散列表305包括多个桶(bucket)307。在一些实施例中,散列表305是布鲁姆(Bloom)过滤器。散列表305用于检测错误推测,并且用于记录被推测地访问的数据的地址、访问类型、序列号和id号。散列表305包括N个“集合”,其中每个集合包括M个条目309。每个条目309保存有效位、先前执行的推测存储器指令的元素的序列号、id号和访问类型。在一些实施例中,每个条目309还包括对应的地址(图中示为虚线框)。当DSX启动指令(例如,下文详述的YBEGIN和变型)时,清除所有有效位,并且置位“推测活动”标志,并且当指令结束DSX时,清除推测活动标志。

[0118] 冲突检查电路311针对被测试元素(或其块)315对每个条目309检查冲突。在一些实施例中,当条目309是有效的并且至少以下情况之一时存在冲突:i)条目309中的访问类

型是写入或ii)被测试访问类型是写入,以及以下情况之一:i)条目309中的序列号小于被测试元素315的序列号,并且条目309中的id号大于被测试元素315的id号,或ii)条目309中的序列号大于被测试元素315的序列号,并且条目309中的id号小于被测试元素315的id号。

[0119] 换言之,在以下情况中存在冲突:

[0120] (条目是有效的)AND((条目中的访问类型==写入)OR(被测试访问类型==写入))AND(((条目中的Seq#<被测试Seq#)AND(条目中的id#>被测试id#))OR((条目中的Seq#>被测试Seq#)AND(条目中的id#<被测试id#)))

[0121] 注意,在大多数实施例中,不存在对地址重叠的测试。这个重叠是通过命中散列表中的条目而隐含的。当不存在地址重叠时,命中可以发生,因为来自散列函数和/或来自检查的重叠过于粗粒度(即,B过大)。然而,当存在地址重叠时将会有命中。因此保证了正确性,但是可能存在假肯定值(false-positive)(即,当不存在错误推测时硬件可以检测错误推测)。在实施例中,将块地址存储在每个条目309中,并且应用用于测试错误推测的附加条件(即,将其与上述条件进行逻辑地AND操作),其中条目309中的地址等于被测试元素315中的地址。

[0122] OR门313(或等价物)对冲突检查的结果进行逻辑地OR操作。当OR操作的结果是1,则错误推测可能已经发生,并且OR门313通过其输出指示错误推测可能已经发生。

[0123] 这个实施例的总存储是M*N个条目。这意味着它可能跟踪高达M*N个被推测地访问的数据元素。然而在实践中,循环可能对N个集合中的一些比对N个集合中的其他具有更多访问。如果任何集合中的空间用尽,则在一些实施例中,触发错误推测来保证正确性。增加M缓和这个问题,但是可能迫使存在冲突检查硬件的更多副本。为了同时执行全部M个冲突检查(如在一些实施例中完成的),存在冲突检查逻辑的M个副本。

[0124] 以某种方式选择B、N、M和散列函数,允许结构以与L1数据高速缓存非常类似的方式被组织。具体而言,令B作为高速缓存行的大小,N作为L1数据高速缓存中的集合的数量,M作为L1数据高速缓存的相关联性,并且令散列函数作为地址的最低有效位(在右移之后)。这个结构将具有与L1数据高速缓存相同数量的条目和相同的组织,这可以简化其实现方式。

[0125] 最后,注意替代实施例使用单独的布鲁姆过滤器用于读取和写入,以避免必须存储访问类型信息,并且避免在冲突检查期间必须检查访问类型。相反,对于读取,实施例只针对“写入”过滤器执行冲突检查,并且如果不存在错误推测,则将元素插入“读取”过滤器。类似地,对于写入,实施例针对“读取”和“写入”过滤器两者执行冲突检查,并且如果不存在错误推测,则将元素插入“写入”过滤器。

[0126] 图4示出DSX跟踪硬件执行的DSX错误推测检测的示例性方法。在401处,开始DSX或提交先前的推测迭代。例如,执行YBEGIN指令。对该指令的执行清除条目309中的有效位并且在状态寄存器(诸如之前详述的DSX状态寄存器)中置位推测活动标志(如果还未置位)。在DSX开始并且提供被测试数据元素之后执行推测存储器指令。

[0127] 在403处,来自推测存储器指令的被测试数据元素被分成不超过B字节的块。以B字节的粒度访问散列表(即,丢弃地址的低位)。如果元素足够大并且/或者不是对齐的,则它们可以跨越B字节边界,并且如果是,则元素被分成多个块。

[0128] 对每个块执行下列(405-421)操作。将块的开始地址右移$\log_2 B$。在407处将被移位

的地址散列以生成索引值。

[0129] 使用索引值，在409处作出对散列表的对应集合的查找并且在411处读出集合的所有条目。

[0130] 对于每个读出的条目，在413处针对被测试元素(诸如上述被测试元素)执行冲突检查。在415处对所有冲突检查执行OR操作。如果在417处任何检查指示冲突(使得OR为1)，则在419处作出对错误推测的指示。此时通常中止DSX。如果不存在错误推测，则在421处发现集合中的无效条目，并且使用被测试并且被标记为有效的元素的信息来填充无效条目。如果不存在无效条目，则触发错误推测。

[0131] 图5(A)-(B)示出由DSX跟踪硬件执行的DSX错误推测检测的示例性方法。在501处，开始DSX或提交先前的推测迭代。例如，执行YBEGIN指令。

[0132] 在503处，对该指令的执行通过清除条目309中的有效位来重置跟踪硬件并且在状态寄存器(诸如之前详述的DSX状态寄存器)中置位推测活动标志(如果还未置位)。

[0133] 在505处，执行推测存储器指令。上文详述了这些指令的示例。在507处将作为来自推测指令的被测试元素编号(e)的计数器置位为零，并且在509处计算id(id＝序列号+跨度*e)。

[0134] 在511处作出对任何先前写入是否与计数器值e重叠的确定。这充当针对先前存储(写入)的依赖性检查。对于重叠的写入，在513处执行冲突检查。在一些实施例中，该冲突检查查明是否：i)条目309中的序列号小于被测试元素315的序列号，并且条目309中的id号大于被测试元素315的id号，或ii)条目309中的序列号大于被测试元素315的序列号，并且条目309中的id号小于被测试元素315的id号。

[0135] 如果存在冲突，则在515处触发错误推测。如果否，或者如果不存在重叠的先前写入，则在517处作出对推测存储器指令是否是写入的确定。

[0136] 如果是，则在519处作出对任何先前读取是否与计数器值e重叠的确定。这充当针对先前加载(读取)的依赖性检查。对于重叠的读取，在521处执行冲突检查。在一些实施例中，该冲突检查查明是否：i)条目309中的序列号小于被测试元素315的序列号，并且条目309中的id号大于被测试元素315的id号，或ii)条目309中的序列号大于被测试元素315的序列号，并且条目309中的id号小于被测试元素315的id号。

[0137] 如果存在冲突，则在523处触发错误推测。如果否，或者如果不存在重叠的先前读取，则在525处递增计数器e。

[0138] 在526处作出对计数器e是否等于推测存储器指令中的元素的数量的确定。换言之，已经评估了所有元素吗?如果否，则在509处计算另一个id。如果是，则在527处硬件等待要执行的另一个指令。当下一指令是另一个推测存储器指令时，则在507处重置计数器。当下一指令是YBEGIN时，则在503处重置硬件等等。当下一指令是YEND，则在529处禁用DSX。

[0139] YBEGIN指令

[0140] 图6示出对用于开始DSX的指令的执行的实施例。如本文中将详述的，该指令被称为“YBEGIN”并且用于表示DSX区域的开始。当然，该指令可以被称为另一个名称。在一些实施例中，对诸如中央处理单元(CPU)、图形处理单元(GPU)、加速处理单元(APU)、数字信号处理器(DSP)等等的硬件设备的一个或多个硬件核实现该执行。在其他实施例中，对该指令的执行是仿真。

[0141] 在601处，接收/取出YBEGIN指令。例如，将指令从存储器取出到指令高速缓存中或从指令高速缓存取出。被取出的指令可以采取下文详述的若干形式之一。

[0142] 图7示出YBEGIN指令格式的一些示例性实施例。在实施例中，YBEGIN指令包括操作码(YBEGIN)和用于提供用于回退地址的位移的单个操作数，回退地址是程序执行为了处理错误推测而应该跳转到的地方，如701所示。本质上，位移值是回退地址的一部分。在一些实施例中，该位移值作为立即数操作数被提供。在其他实施例中，将该位移值存储在寄存器或存储器位置操作数中。取决于YBEGIN实现方式，使用DSX状态寄存器、嵌套计数寄存器和/或RTM状态寄存器的隐含操作数。如之前详述的，DSX状态寄存器可以是专用寄存器，不专用于DSX状态的寄存器(诸如像标志寄存器的总体状态寄存器)中的标志、等等。

[0143] 在另一个实施例中，YBEGIN指令不仅包括操作码和位移操作数，还包括用于DSX状态的显式操作数，诸如DSX状态寄存器，如703所示。取决于YBEGIN实现方式，使用嵌套计数寄存器和/或RTM状态寄存器的隐含操作数。如之前详述的，DSX状态寄存器可以是专用寄存器，不专用于DSX状态的寄存器(诸如像标志寄存器的总体状态寄存器)中的标志、等等。

[0144] 在另一个实施例中，YBEGIN指令不仅包括操作码和位移操作数，还包括用于DSX嵌套计数的显式操作数，诸如DSX嵌套计数寄存器，如705所示。如之前详述的，DSX嵌套计数可以是专用寄存器、不专用于DSX嵌套计数的寄存器(诸如总体状态寄存器)中的标志。取决于YBEGIN实现方式，使用DSX状态寄存器和/或RTM状态寄存器的隐含操作数。如之前详述的，DSX状态寄存器可以是专用寄存器，不专用于DSX状态的寄存器(诸如像标志寄存器的总体状态寄存器)中的标志、等等。

[0145] 在另一个实施例中，YBEGIN指令不仅包括操作码和位移操作数，还包括诸如DSX状态寄存器的用于DSX状态的显式操作数和诸如DSX嵌套计数寄存器的用于DSX嵌套计数的显式操作数，如707所示。如之前详述的，DSX状态寄存器可以是专用寄存器、不专用于DSX状态的寄存器(诸如像标志寄存器的总体状态寄存器等等)中的标志，并且DSX嵌套计数可以是专用寄存器、不专用于DSX嵌套计数的寄存器(诸如总体状态寄存器)中的标志。取决于YBEGIN实现方式，使用RTM状态寄存器的隐含操作数。如之前详述的，DSX状态寄存器可以是专用寄存器、不专用于DSX状态的寄存器(诸如像标志寄存器的总体状态寄存器)中的标志等等。

[0146] 在另一个实施例中，YBEGIN指令不仅包括操作码和位移操作数，还包括诸如DSX状态寄存器的用于DSX状态的显式操作数、诸如DSX嵌套计数寄存器的用于DSX嵌套计数的显式操作数和用于RTM状态的显式操作数，如709所示。如之前详述的，DSX状态寄存器可以是专用寄存器、不专用于DSX状态的寄存器(诸如像标志寄存器的总体状态寄存器等等)中的标志，并且DSX嵌套计数可以是专用寄存器、不专用于DSX嵌套计数的寄存器(诸如总体状态寄存器)中的标志。

[0147] 当然，YBEGIN的其他变型是可能的。例如，代替提供位移值，指令包括在立即数、寄存器或存储器位置中的回退地址本身。

[0148] 返回图6，在603处解码被取出/接收的YBEGIN指令。在一些实施例中，由诸如后文详述的那些硬件解码器解码指令。在一些实施例中，将指令解码为微操作(微op)。例如，一些基于CISC的机器通常使用从宏指令导出的微操作。在其他实施例中，解码是诸如及时编译的软件例程的一部分。

[0149] 在605处，检索与经解码的指令相关联的任何操作数。例如，检索来自DSX寄存器、DSX嵌套计数寄存器和/或RTM状态寄存器中的一个或多个的数据。

[0150] 在607处执行经解码的YBEGIN指令。在其中将指令解码为微操作的实施例中，执行这些微操作。对经解码的指令的执行使得硬件进行会被执行的下列动作中的一个或多个：1)确定RTM事务是活动的并且继续该事务；2)使用与YBEGIN指令的指令指针相加的位移值来计算回退地址；3)递增DSX嵌套计数；4)中止；5)将DSX状态设置为活动的；和/或6)重置DSX跟踪硬件。

[0151] 典型地，根据YBEGIN指令的实例，如果不存在活动的RTM事务，则将DSX状态设置为活动的，递增DSX嵌套计数(如果计数小于最大值)，重置DSX跟踪硬件(例如，如上文详述的)，并且使用位移值计算回退地址以开始DSX区域。如之前详述的，通常将DSX的状态存储在诸如寄存器的可访问位置中，诸如上文参照图1讨论的DSX状态和控制寄存器(DSXSR)。然而，可以利用诸如非专用控制/状态寄存器(诸如FLAGS寄存器)中的DSX状态标志的其他手段。先前还描述了对DSX跟踪硬件的重置。如之前详述的，通常将DSX的状态存储在诸如寄存器的可访问位置中，诸如上文参照图1讨论的DSX状态和控制寄存器(DSXSR)。然而，可以利用诸如非专用控制/状态寄存器(诸如FLAGS寄存器)中的DSX状态标志的其他手段。可以由核的硬件检查这个寄存器以确定DSX是否确实发生。

[0152] 如果存在一些原因DSX不能开始，则其他可能的动作中的一个或多个发生。例如，在支持RTM的处理器的一些实施例中，如果RTM事务是活动的，则一开始DSX不应当是活动的，并且继续RTM。如果一开始对DSX的建立有错误(嵌套计数不正确)，则中止将发生。此外，在一些实施例中，如果不存在DSX，则生成错误并且执行无操作(NOP)。无论执行哪个动作，在大多数实施例中，在该动作之后重置DSX状态(如果它被设置了)以指示不存在待决DSX。

[0153] 图8示出对诸如YBEGIN指令的指令的执行的详细实施例。例如，在一些实施例中，这个流程是图6的框607。在一些实施例中，对诸如中央处理单元(CPU)、图形处理单元(GPU)、加速处理单元(APU)、数字信号处理器(DSP)等等的硬件设备的一个或多个硬件核实现该执行。在其他实施例中，对该指令的执行是仿真。

[0154] 在一些实施例中，例如在支持RTM事务的处理器中，在801处作出对RTM事务是否正在发生的确定。例如，在支持RTM的处理器的一些实施例中，如果RTM事务是活动的，则一开始DSX不应当是活动的。在该实例中，RTM事务中有错误，并且应当激活其结束程序。典型地，将RTM事务状态存储在诸如RTM控制和状态寄存器的寄存器中。处理器的硬件评估该寄存器的内容以确定RTM事务是否正在发生。当RTM事务正在发生时，在803处RTM事务继续处理。

[0155] 当没有RTM事务正在发生时，或RTM不被支持时，在805处作出对当前DSX嵌套计数是否小于最大嵌套计数的确定。在一些实施例中，YBEGIN指令提供用于存储当前嵌套计数的嵌套计数寄存器作为操作数。替代地，硬件中可以存在专用嵌套计数寄存器以用于存储当前嵌套计数。最大嵌套计数是无须对应DSX结束(例如，经由YEND指令)的情况下可以发生的DSX开始(例如，经由YBEGIN指令)的最大数量。

[0156] 当当前DSX嵌套计数大于最大值时，在807处发生中止。在一些实施例中，中止通过使用诸如DSX恢复电路135的恢复电路来触发回滚。在其他实施例中，如下文详述的执行YABORT指令，其不仅执行到回退地址的回滚，还丢弃推测地存储的写入并且重置当前嵌套计数并且将DSX状态设置为不活动的。如上文详述的，通常将DSX状态存储在控制寄存器中，

诸如图1中示出的DSX状态和控制寄存器(DSXSR)。然而,可以利用诸如非专用控制/状态寄存器(诸如FLAGS寄存器)中的DSX状态标志的其他手段。

[0157] 当当前嵌套计数不大于最大值时,在809处递增当前DSX嵌套计数。

[0158] 在811处作出对当前DSX嵌套计数是否等于一的确定。当等于一时,在一些实施例中,在813处通过将YBEGIN指令提供的位移值与跟随YBEGIN指令的指令的地址相加来计算回退地址。在其中YBEGIN指令提供了回退地址的实施例中,则该计算不是必须的。

[0159] 在815处,将DSX状态设置为活动的(如果需要)并且重置DSX跟踪硬件(例如,如上文详述的)。例如,如之前详述的,通常将DSX的状态存储在诸如寄存器的可访问位置中,诸如上文参照图1讨论的DSX状态和控制寄存器(DSXSR)。然而,可以利用诸如非专用控制/状态寄存器(诸如FLAGS寄存器)中的DSX状态标志的其他手段。可以由核的硬件检查这个寄存器以确定DSX是否确实发生。

[0160] 图9示出显示对诸如YBEGIN指令的指令的执行的伪代码的示例。

[0161] YBEGIN WITH STRIDE指令

[0162] 图10示出对用于开始DSX的指令的执行的实施例。如本文中将详述的,该指令被称为“YBEGIN WITH STRIDE”并且用于表示DSX区域的开始。当然,该指令可以被称为另一个名称。在一些实施例中,对诸如中央处理单元(CPU)、图形处理单元(GPU)、加速处理单元(APU)、数字信号处理器(DSP)等等的硬件设备的一个或多个硬件核实现该执行。在其他实施例中,对该指令的执行是仿真。

[0163] 在1001处,接收/取出YBEGIN WITH STRIDE指令。例如,将指令从存储器取出到指令高速缓存中或从指令高速缓存取出。被取出的指令可以采取下文详述的若干形式之一。

[0164] 图11示出YBEGIN WITH STRIDE指令格式的一些示例性实施例。在实施例中,YBEGIN WITH STRIDE指令包括操作码(YBEGIN WITH STRIDE)和用于提供用于回退地址的位移的操作数(回退地址是程序执行为了处理错误推测而应该跳转到的地方)和跨度值操作数,如1101所示。本质上,位移是回退地址的一部分。在一些实施例中,位移作为立即数操作数被提供。在其他实施例中,将位移值存储在寄存器或存储器位置操作数中。在一些实施例中,跨度作为立即数操作数被提供。在其他实施例中,将跨度存储在寄存器或存储器位置操作数中。取决于YBEGIN WITH STRIDE实现方式,使用DSX状态寄存器、嵌套计数寄存器和/或RTM状态寄存器的隐含操作数。

[0165] 在另一个实施例中,YBEGIN WITH STRIDE指令不仅包括操作码和位移操作数和跨度值操作数,还包括用于DSX状态的显式操作数,诸如DSX状态寄存器,如1103所示。在一些实施例中,位移作为立即数操作数被提供。在其他实施例中,将位移值存储在寄存器或存储器位置操作数中。在一些实施例中,跨度作为立即数操作数被提供。在其他实施例中,将跨度存储在寄存器或存储器位置操作数中。如之前详述的,DSX状态寄存器可以是专用寄存器、不专用于DSX状态的寄存器(诸如像标志寄存器的总体状态寄存器)中的标志、等等。取决于YBEGIN WITH STRIDE实现方式,使用嵌套计数寄存器和/或RTM状态寄存器的隐含操作数。

[0166] 在另一个实施例中,YBEGIN WITH STRIDE指令不仅包括操作码和位移操作数和跨度值操作数,和跨度值操作数,还包括用于DSX嵌套计数的显式操作数,诸如DSX嵌套计数寄存器,如1105所示。在一些实施例中,位移作为立即数操作数被提供。在其他实施例中,将位

移值存储在寄存器或存储器位置操作数中。在一些实施例中，跨度作为立即数操作数被提供。在其他实施例中，将跨度存储在寄存器或存储器位置操作数中。如之前详述的，DSX嵌套计数可以是专用寄存器、不专用于DSX嵌套计数的寄存器(诸如总体状态寄存器)中的标志。取决于YBEGIN WITH STRIDE实现方式，使用DSX状态寄存器和/或RTM状态寄存器的隐含操作数。

[0167] 在另一个实施例中，YBEGIN WITH STRIDE指令不仅包括操作码、位移操作数和跨度值操作数，还包括诸如DSX状态寄存器的用于DSX状态的显式操作数和诸如DSX嵌套计数寄存器的用于DSX嵌套计数的显式操作数，如1107所示。在一些实施例中，位移作为立即数操作数被提供。在其他实施例中，将位移值存储在寄存器或存储器位置操作数中。在一些实施例中，跨度作为立即数操作数被提供。在其他实施例中，将跨度存储在寄存器或存储器位置操作数中。如之前详述的，DSX状态寄存器可以是专用寄存器、不专用于DSX状态的寄存器(诸如像标志寄存器的总体状态寄存器等等)中的标志，并且DSX嵌套计数可以是专用寄存器、不专用于DSX嵌套计数的寄存器(诸如总体状态寄存器)中的标志。取决于YBEGIN WITH STRIDE实现方式，使用RTM状态寄存器的隐含操作数。

[0168] 在另一个实施例中，YBEGIN WITH STRIDE指令不仅包括操作码、位移操作数和跨度值操作数，还包括诸如DSX状态寄存器的用于DSX状态的显式操作数和诸如DSX嵌套计数寄存器的用于DSX嵌套计数的显式操作数以及RTM状态寄存器，如409所示。在一些实施例中，位移作为立即数操作数被提供。在其他实施例中，将位移值存储在寄存器或存储器位置操作数中。在一些实施例中，跨度作为立即数操作数被提供。在其他实施例中，将跨度存储在寄存器或存储器位置操作数中。如之前详述的，DSX状态寄存器可以是专用寄存器、不专用于DSX状态的寄存器(诸如像标志寄存器的总体状态寄存器等等)中的标志，并且DSX嵌套计数可以是专用寄存器、不专用于DSX嵌套计数的寄存器(诸如总体状态寄存器)中的标志。

[0169] 当然，YBEGIN WITH STRIDE的其他变型是可能的。例如，代替提供位移值，指令包括本身在立即数、寄存器或存储器位置中的回退地址。

[0170] 返回图10，在1003处解码被取出/接收的YBEGIN WITH STRIDE指令。在一些实施例中，由诸如后文详述的那些硬件解码器解码指令。在一些实施例中，将指令解码为微操作(微op)。例如，一些基于CISC的机器通常使用从宏指令导出的微操作。在其他实施例中，解码是诸如及时编译的软件例程的一部分。

[0171] 在1005处，检索与经解码的YBEGIN WITH STRIDE指令相关联的任何操作数。例如，检索来自DSX寄存器、DSX嵌套计数寄存器和/或RTM状态寄存器中的一个或多个的数据。

[0172] 在1007处执行经解码的YBEGIN WITH STRIDE指令。在其中将指令解码为微操作的实施例中，执行这些微操作。对经解码的指令的执行使得硬件进行会被执行的下列动作中的一个或多个：1)确定RTM事务是活动的并且开始该事务；2)使用与YBEGIN WITH STRIDE指令的指令指针相加的位移值来计算回退地址；3)递增DSX嵌套计数；4)中止；5)将DSX状态设置为活动的；6)重置DSX跟踪硬件和/或7)将跨度值提供给DSX硬件跟踪器。

[0173] 典型地，一旦出现YBEGIN WITH STRIDE指令的第一实例，如果不存在活动的RTM事务，则将DSX状态设置为活动的，重置DSX跟踪硬件(例如，如上文详述的使用提供的跨度值)，并且使用位移值计算回退地址以开始DSX区域。如之前详述的，通常将DSX的状态存储在诸如寄存器的可访问位置中，诸如上文参照图1讨论的DSX状态和控制寄存器(DSXSR)。然

而，可以利用诸如非专用控制/状态寄存器(诸如FLAGS寄存器)中的DSX状态标志的其他手段。先前还描述了对DSX跟踪硬件的重置。

[0174] 典型地，一旦出现YBEGIN WITH STRIDE指令的实例，如果不存在活动的RTM事务，则将DSX状态设置为活动的，递增DSX嵌套计数(如果计数小于最大值)，重置DSX跟踪硬件(例如，如上文详述的使用提供的跨度)，并且使用位移值计算回退地址以开始DSX区域。如之前详述的，通常将DSX的状态存储在诸如寄存器的可访问位置中，诸如上文参照图1讨论的DSX状态和控制寄存器(DSXSR)。然而，可以利用诸如非专用控制/状态寄存器(诸如FLAGS寄存器)中的DSX状态标志的其他手段。先前还描述了对DSX跟踪硬件的重置。如之前详述的，通常将DSX的状态存储在诸如寄存器的可访问位置中，诸如上文参照图1讨论的DSX状态和控制寄存器(DSXSR)。然而，可以利用诸如非专用控制/状态寄存器(诸如FLAGS寄存器)中的DSX状态标志的其他手段。可以由核的硬件检查这个寄存器以确定DSX是否确实发生。

[0175] 如果存在一些原因DSX不能开始，则其他可能的动作中的一个或多个发生。例如，在支持RTM的处理器的一些实施例中，如果RTM事务是活动的，则一开始DSX不应当是活动的，并且继续RTM。如果一开始对DSX的建立有错误(嵌套计数不正确)，则中止将发生。此外，在一些实施例中，如果不存在DSX，则生成错误并且执行无操作(NOP)。无论执行哪个动作，在大多数实施例中，在该动作之后重置DSX状态(如果它被设置了)以指示不存在待决DSX。

[0176] 图12示出对诸如YBEGIN WITH STRIDE指令的指令的执行的详细实施例。例如，在一些实施例中，这个流程是图10的框1007。在一些实施例中，对诸如中央处理单元(CPU)、图形处理单元(GPU)、加速处理单元(APU)、数字信号处理器(DSP)等等的硬件设备的一个或多个硬件核实现该执行。在其他实施例中，对该指令的执行是仿真。

[0177] 在一些实施例中，例如在支持RTM事务的处理器中，在1201处作出对RTM事务是否正在发生的确定。例如，在支持RTM的处理器的一些实施例中，如果RTM事务是活动的，则一开始DSX不应当是活动的。在该实例中，RTM事务中有错误，并且应当激活其结束程序。典型地，将RTM事务状态存储在诸如RTM控制和状态寄存器的寄存器中。处理器的硬件评估该寄存器的内容以确定RTM事务是否正在发生。当RTM事务正在发生时，在1203处RTM事务继续处理。

[0178] 当没有RTM事务正在发生时，或RTM不被支持时，在1205处作出对当前DSX嵌套计数是否小于最大嵌套计数的确定。在一些实施例中，YBEGIN WITH STRIDE指令提供用于存储当前嵌套计数的嵌套计数寄存器作为操作数。替代地，硬件中可以存在专用嵌套计数寄存器以用于存储当前嵌套计数。最大嵌套计数是无须对应DSX结束(例如，经由YEND指令)的情况下可以发生的DSX开始(例如，经由YBEGIN指令)的最大数量。

[0179] 当当前嵌套计数大于最大值时，在1207处发生中止。在一些实施例中，中止触发回滚。在其他实施例中，如下文详述的执行YABORT，其不仅执行到回退地址的回滚，还丢弃推测地存储的写入并且重置当前嵌套计数并且将DSX状态设置为不活动的。如上文详述的，通常将DSX状态存储在控制寄存器中，诸如图1中示出的DSX状态和控制寄存器(DSXSR)。然而，可以利用诸如非专用控制/状态寄存器(诸如FLAGS寄存器)中的DSX状态标志的其他手段。

[0180] 当当前嵌套计数不大于最大值时，在1209处递增当前DSX嵌套计数。

[0181] 在1211处作出对当前DSX嵌套计数是否等于一的确定。当等于一时，在一些实施例中，在1213处通过将YBEGIN WITH STRIDE指令提供的位移值与跟随YBEGIN WITH STRIDE指

令的指令的地址相加来计算回退地址。在其中YBEGIN WITH STRIDE指令提供了回退地址的实施例中，则该计算不是必须的。

[0182] 在1215处，将DSX状态设置为活动的(如果其需要是)并且重置DSX跟踪硬件(例如，如上文详述的包括使用提供的跨度值)。例如，如之前详述的，通常将DSX的状态存储在诸如寄存器的可访问位置中，诸如上文参照图1讨论的DSX状态和控制寄存器(DSXSR)。然而，可以利用诸如非专用控制/状态寄存器(诸如FLAGS寄存器)中的DSX状态标志的其他手段。可以由核的硬件检查这个寄存器以确定DSX是否确实发生。

[0183] YCONTINUE指令

[0184] 随着DSX到达末尾(例如，循环的迭代已经运行了其过程)而没有任何问题，在一些实施例中，执行指令(YEND)以指示推测区域的结束，如下文详述的。简言之，对这个指令的执行引起了对当前推测状态(还未被写入的所有写入)的提交以及从当前推测区域的退出，如下文将讨论的。然后通过调用另一个YBEGIN可以开始循环的另一个迭代。

[0185] 然而，在一些实施例中，当不再需要推测时(例如，当存储之间不存在冲突时)，通过使用继续指令以提交当前循环迭代，可获得对YBEGIN、YEND、YBEGIN等等的此循环的优化。继续指令还开始新推测循环迭代而不需要调用YBEGIN。

[0186] 图13示出对用于继续DSX而不结束它的指令的执行的实施例。如本文中将详述的，该指令被称为“YCONTINUE”并且用于表示事务的结束。当然，该指令可以被称为另一个名称。

[0187] 在一些实施例中，对诸如中央处理单元(CPU)、图形处理单元(GPU)、加速处理单元(APU)、数字信号处理器(DSP)等等的硬件设备的一个或多个硬件核实现该执行。在其他实施例中，对该指令的执行是仿真。

[0188] 在1301处，接收/取出YCONTINUE指令。例如，将指令从存储器取出到指令高速缓存中或从指令高速缓存取出。被取出的指令可以采取若干形式之一。

[0189] 图14示出YCONTINUE指令格式的一些示例性实施例。在实施例中，YCONTINUE指令包括操作码(YCONTINUE)，但是没有显式操作数，如1401所示。取决于YCONTINUE实现方式，DSX状态寄存器和嵌套计数寄存器的隐含操作数。如之前详述的，DSX嵌套计数可以是专用寄存器，寄存器中的标志不专用于DSX嵌套计数(诸如总体状态寄存器)等等。此外，DSX状态寄存器可以是专用寄存器、不专用于DSX状态的寄存器(诸如像标志寄存器的总体状态寄存器)中的标志、等等。

[0190] 在另一个实施例中，YCONTINUE指令不仅包括操作码，还包括用于DSX状态的显式操作数，诸如DSX状态寄存器，如1403所示。取决于YCONTINUE实现方式，使用嵌套计数寄存器的隐含操作数。如之前详述的，DSX嵌套计数可以是专用寄存器、不专用于DSX嵌套计数的寄存器(诸如总体状态寄存器)中的标志、等等。此外，DSX状态寄存器可以是专用寄存器、不专用于DSX状态的寄存器(诸如像标志寄存器的总体状态寄存器)中的标志、等等。

[0191] 在另一个实施例中，YCONTINUE指令不仅包括操作码，还包括用于DSX嵌套计数的显式操作数，诸如DSX嵌套计数寄存器，如1405所示。取决于YCONTINUE实现方式，使用DSX状态寄存器的隐含操作数。如之前详述的，DSX嵌套计数可以是专用寄存器、不专用于DSX嵌套计数的寄存器(诸如总体状态寄存器)中的标志、等等。此外，DSX状态寄存器可以是专用寄存器、不专用于DSX状态的寄存器(诸如像标志寄存器的总体状态寄存器)中的标志、等等。

[0192] 在另一个实施例中，YCONTINUE指令不仅包括操作码，还包括诸如DSX状态寄存器的用于DSX状态的显式操作数和诸如DSX嵌套计数寄存器的用于DSX嵌套计数的显式操作数，如1407所示。如之前详述的，DSX嵌套计数可以是专用寄存器、不专用于DSX嵌套计数的寄存器(诸如总体状态寄存器)中的标志、等等。此外，DSX状态寄存器可以是专用寄存器、不专用于DSX状态的寄存器(诸如像标志寄存器的总体状态寄存器)中的标志、等等。

[0193] 返回图13，在1303处解码被取出/接收的YCONTINUE指令。在一些实施例中，由诸如后文详述的那些硬件解码器解码指令。在一些实施例中，将指令解码为微操作(微op)。例如，一些基于CISC的机器通常使用从宏指令导出的微操作。在其他实施例中，解码是诸如及时编译的软件例程的一部分。

[0194] 在1305处，检索与经解码的YCONTINUE指令相关联的任何操作数。例如，检索来自DSX寄存器和DSX嵌套计数寄存器中的一个或多个的数据。

[0195] 在1307处执行经解码的YCONTINUE指令。在其中将指令解码为微操作的实施例中，执行这些微操作。对经解码的指令的执行使得硬件进行会被执行的下列动作中的一个或多个：1)确定由于不再需要推测而使得与DSX相关联的推测写入会被提交，并且提交它们，并且开始新推测循环迭代(诸如新DSX区域)；和/或2)无操作。

[0196] 可以由之前详述的DSX检查硬件执行这些动作中的第一个(使推测写入终结并且开始新推测循环迭代)。在这个动作中，提交与DSX的循环迭代相关联的所有推测写入(存储使得它们在DSX外部是可访问的)，但是与YEND指令不同，DSX状态不设置为指示DSX不存在。例如，提交与DSX相关联的所有写入(诸如存储在高速缓存、寄存器或存储器中的)，使得它们被终结并且在DSX外部是可见的。典型地，DSX提交将不发生，除非DSX嵌套计数是一。否则，在一些实施例中，然后执行nop。

[0197] 在一些实施例中，如果DSX不是活动的，则可以执行nop。

[0198] 图15示出对诸如YCONTINUE指令的指令的执行的详细实施例。例如，在一些实施例中，这个流程是图13的框1307。在一些实施例中，对诸如中央处理单元(CPU)、图形处理单元(GPU)、加速处理单元(APU)、数字信号处理器(DSP)等等的硬件设备的一个或多个硬件核实现该执行。在其他实施例中，对该指令的执行是仿真。

[0199] 在1501处作出对DSX是否活动的确定。如上文详述的，通常将DSX状态存储在控制寄存器中，诸如图1中示出的DSX状态和控制寄存器(DSXSR)。然而，可以利用诸如非专用控制/状态寄存器(诸如FLAGS寄存器)中的DSX状态标志的其他手段。无论状态被存储在何处，由处理器的硬件检查位置以确定DSX是否确实发生。

[0200] 当没有DSX发生时，在1503处执行无op。

[0201] 当DSX发生时，在1505处作出对DSX嵌套计数是否等于一的确定。如上文详述的，通常将DSX嵌套计数存储在嵌套计数寄存器中。当DSX嵌套计数不是一时，在507处执行nop。当DSX嵌套计数是一时，在1509处完成提交和DSX重新开始。当提交和DSX重新开始发生时，在一些实施例中，下列操作中的一个或多个发生：1)重置DSX跟踪硬件(例如，如上文详述的)，2)计算回退地址，以及3)作出对先前推测区域的被推测地执行的指令(写入)的提交。

[0202] 图16示出显示对诸如YCONTINUE指令的指令的执行的伪代码的示例。

[0203] YBORT指令

[0204] 有时，在DSX内，存在需要DSX中止的问题(诸如错误推测)。图17示出对用于中止

DSX的指令的执行的实施例。如本文中将详述的，这个指令被称为“YABORT”。当然，该指令可以被称为另一个名称。在一些实施例中，对诸如中央处理单元(CPU)、图形处理单元(GPU)、加速处理单元(APU)、数字信号处理器(DSP)等等的硬件设备的一个或多个硬件核实现该执行。在其他实施例中，对该指令的执行是仿真。

[0205] 在1701处，接收/取出YABORT指令。例如，将指令从存储器取出到指令高速缓存中或从指令高速缓存取出。被取出的指令可以采取下文详述的若干形式之一。

[0206] 图18示出YABORT指令格式的一些示例性实施例。在实施例中，YABORT指令只包括操作码(YABORT)，如1801所示。取决于YABORT实现方式，使用DSX状态寄存器和/或RTM状态寄存器的隐含操作数。如之前详述的，DSX状态寄存器可以是专用寄存器、不专用于DSX状态的寄存器(诸如像标志寄存器的总体状态寄存器)中的标志、等等。

[0207] 在另一个实施例中，YABORT指令不仅包括操作码，还包括DSX状态寄存器的显式操作数，诸如DSX状态寄存器，如1803所示。如之前详述的，DSX状态寄存器可以是专用寄存器、不专用于DSX状态的寄存器(诸如像标志寄存器的总体状态寄存器)中的标志、等等。取决于YABORT实现方式，使用RTM状态寄存器的隐含操作数。

[0208] 在另一个实施例中，YABORT指令不仅包括操作码，还包括诸如DSX状态寄存器的DSX状态寄存器的显式操作数和RTM状态寄存器的显示操作数，如1805所示。如之前详述的，DSX状态寄存器可以是专用寄存器、不专用于DSX状态的寄存器(诸如像标志寄存器的总体状态寄存器)中的标志、等等。

[0209] 返回图17，在1703处解码被取出/接收的YABORT指令。在一些实施例中，由诸如后文详述的那些硬件解码器解码指令。在一些实施例中，将指令解码为微操作(微op)。例如，一些基于CISC的机器通常使用从宏指令导出的微操作。在其他实施例中，解码是诸如及时编译的软件例程的一部分。

[0210] 在1705处，检索与经解码的YABORT指令相关联的任何操作数。例如，检索来自DSX寄存器和/或RTM状态寄存器中的一个或多个的数据。

[0211] 在1707处执行经解码的YABORT指令。在其中将指令解码为微操作的实施例中，执行这些微操作。对经解码的指令的执行使得硬件进行会被执行的下列动作中的一个或多个：1)确定RTM事务是活动的并且中止RTM事务；2)确定DSX是不活动的并且执行无操作；和/或3)通过重置任何DSX嵌套计数、丢弃所有被推测地执行的写入、将DSX状态设置为不活动的并且将执行回滚到回退地址来中止DSX。

[0212] 关于第一个动作，通常将RTM状态存储在RTM状态和控制寄存器中。当这个寄存器指示RTM事务正在发生时，不应当执行YABORT指令。由此，RTM事务存在问题并且其应当中止。

[0213] 关于第二个和第三个动作，如之前详述的，通常将DSX的状态存储在诸如寄存器的可访问位置中，诸如上文参照图1讨论的DSX状态和控制寄存器(DSXSR)。然而，可以利用诸如非专用控制/状态寄存器(诸如FLAGS寄存器)中的DSX状态标志的其他手段。可以由核的硬件检查这个寄存器以确定DSX是否确实发生。当这个寄存器指示无DSX时，则将没有理由执行YABORT指令，并且由此执行无操作(或类似的操作)。当这个寄存器指示有DSX时，则DSX中止处理发生，包括重置DSX跟踪硬件、丢弃所有存储的被推测地执行的写入、并且将DSX状态重置为不活动的并且回滚执行。

[0214] 图19示出对诸如YABORT指令的指令的执行的详细实施例。例如，在一些实施例中，这个流程是图17的框1707。在一些实施例中，对诸如中央处理单元(CPU)、图形处理单元(GPU)、加速处理单元(APU)、数字信号处理器(DSP)等等的硬件设备的一个或多个硬件核实现该执行。在其他实施例中，对该指令的执行是仿真。

[0215] 在一些实施例中，例如在支持RTM事务的处理器中，在1901处作出对RTM事务是否正在发生的确定。例如，在支持RTM的处理器的一些实施例中，如果RTM事务是活动的，则一开始DSX不应当是活动的。在该实例中，RTM事务中有错误，并且应当激活其结束程序。典型地，将RTM事务状态存储在诸如RTM控制和状态寄存器的寄存器中。处理器的硬件评估该寄存器的内容以确定RTM事务是否正在发生。当RTM事务正在发生时，RTM事务继续进行1903。

[0216] 当没有RTM事务发生时，或RTM不被支持时，在1905处作出对DSX是否活动的确定。通常将DSX的状态存储在诸如上文参照图1讨论的DSX状态和控制寄存器(DSXSR)的可访问位置中。然而，可以利用诸如非专用控制/状态寄存器(诸如FLAGS寄存器)中的DSX状态标志的其他手段。可以由核的硬件检查这个寄存器以确定DSX是否发生。

[0217] 当这个寄存器指示无DSX时，则在1907处执行nop。当这个寄存器指示有DSX时，则在1909处DSX中止处理发生，包括重置DSX跟踪硬件、丢弃所有存储的被推测地执行的写入、并且将DSX状态重置为不活动的并且回滚执行。

[0218] 图20示出显示对诸如YABORT指令的指令的执行的伪代码的示例。

[0219] YTEST指令

[0220] 通常在开始新DSX推测区域之前软件需要知道DSX是否是活动的。图21示出对用于测试DSX的状态的指令的执行的实施例。如本文中将详述的，这个指令被称为“YTEST”并且用于通过对标志的使用提供对DSX活动的指示。当然，该指令可以被称为另一个名称。

[0221] 在一些实施例中，对诸如中央处理单元(CPU)、图形处理单元(GPU)、加速处理单元(APU)、数字信号处理器(DSP)等等的硬件设备的一个或多个硬件核实现该执行。在其他实施例中，对该指令的执行是仿真。

[0222] 在2101处，接收/取出YTEST指令。例如，将指令从存储器取出到指令高速缓存中或从指令高速缓存取出。被取出的指令可以采取若干形式之一。图22示出YTEST指令格式的一些示例性实施例。在实施例中，YTEST指令包括操作码(YTEST)，但是没有显式操作数，如2201所示。使用DSX状态寄存器和标志寄存器的隐含操作数。如之前详述的，DSX状态寄存器可以是专用寄存器、不专用于DSX状态的寄存器(诸如像标志寄存器的总体状态寄存器)中的标志、等等。示例性标志寄存器包括EFLAGS寄存器。具体而言，标志寄存器用于存储零标志(ZF)。

[0223] 在另一个实施例中，YTEST指令不仅包括操作码，还包括用于DSX状态的显式操作数，诸如DSX状态寄存器，如2203所示。如之前详述的，DSX状态寄存器可以是专用寄存器、不专用于DSX状态的寄存器(诸如像标志寄存器的总体状态寄存器)中的标志、等等。使用标志寄存器的隐含操作数。示例性标志寄存器包括EFLAGS寄存器。具体而言，标志寄存器用于存储零标志(ZF)。

[0224] 在另一个实施例中，YTEST指令不仅包括操作码，还包括标志寄存器的显式操作数，如2205所示。示例性标志寄存器包括EFLAGS寄存器。具体而言，标志寄存器用于存储零标志(ZF)。使用DSX状态寄存器的隐含操作数。如之前详述的，DSX状态寄存器可以是专用寄

存器、不专用于DSX状态的寄存器(诸如像标志寄存器的总体状态寄存器)中的标志、等等。

[0225] 在另一个实施例中,YTEST指令不仅包括操作码,还包括诸如DSX状态寄存器的用于DSX状态的显式操作数和标志寄存器的显式操作数,如2207所示。如之前详述的,DSX状态寄存器可以是专用寄存器、不专用于DSX状态的寄存器(诸如像标志寄存器的总体状态寄存器)中的标志、等等。使用标志寄存器的隐含操作数。示例性标志寄存器包括EFLAGS寄存器。具体而言,标志寄存器用于存储零标志(ZF)。

[0226] 返回图21,在2103处解码被取出/接收的YTEST指令。在一些实施例中,由诸如后文详述的那些硬件解码器解码指令。在一些实施例中,将指令解码为微操作(微op)。例如,一些基于CISC的机器通常使用从宏指令导出的微操作。在其他实施例中,解码是诸如及时编译的软件例程的一部分。

[0227] 在2105处,检索与经解码的YTEST指令相关联的任何操作数。例如,检索来自DSX状态寄存器的数据。

[0228] 在2107处执行经解码的YTEST指令。在其中将指令解码为微操作的实施例中,执行这些微操作。对经解码的指令的执行使得硬件进行会被执行的下列动作中的一个或多个:1)确定DSX状态寄存器指示DSX是活动的,并且如果是则将标志寄存器中的零标志设置为0;或2)确定DSX状态寄存器指示DSX是不活动的,并且如果是则将标志寄存器中的零标志设置为1。当然,虽然零标志用于显示DSX活动状态,但是取决于实施例使用其他标志。

[0229] 图23示出显示对诸如YTEST指令的指令的执行的伪代码的示例。

[0230] YEND指令

[0231] 随着DSX结束(例如,循环的迭代已经运行了其过程)而没有任何问题,在一些实施例中,执行指令以指示推测区域的结束。简言之,对这个指令的执行引起了对当前推测状态(还未被写入的所有写入)的提交以及从当前推测区域的退出。

[0232] 图24示出对用于结束DSX的指令的执行的实施例。如本文中将详述的,该指令被称为“YEND”并且用于表示DSX的结束。当然,该指令可以被称为另一个名称。

[0233] 在一些实施例中,对诸如中央处理单元(CPU)、图形处理单元(GPU)、加速处理单元(APU)、数字信号处理器(DSP)等等的硬件设备的一个或多个硬件核实现该执行。在其他实施例中,对该指令的执行是仿真。

[0234] 在2401处,接收/取出YEND指令。例如,将指令从存储器取出到指令高速缓存中或从指令高速缓存取出。被取出的指令可以采取若干形式之一。图25示出YEND指令格式的一些示例性实施例。在实施例中,YEND指令包括操作码(YEND),但是没有显式操作数,如2501所示。取决于YEND实现方式,使用用于DSX状态、嵌套计数和/或RTM状态的隐含寄存器操作数。

[0235] 在另一个实施例中,YEND指令不仅包括操作码,还包括用于DSX状态的显式操作数,诸如DSX状态寄存器,如2503所示。如之前详述的,DSX状态寄存器可以是专用寄存器、不专用于DSX状态的寄存器(诸如像标志寄存器的总体状态寄存器)中的标志、等等。取决于YEND实现方式,使用用于嵌套计数和/或RTM状态的隐含寄存器操作数。

[0236] 在另一个实施例中,YEND指令不仅包括操作码,还包括用于DSX嵌套计数的显式操作数,诸如DSX嵌套计数寄存器,如2505所示。如之前详述的,DSX嵌套计数可以是专用寄存器、不专用于DSX嵌套计数的寄存器(诸如总体状态寄存器)中的标志。取决于YEND实现方

式,使用用于DSX状态和/或RTM状态的隐含寄存器操作数。

[0237] 在另一个实施例中,YEND指令不仅包括操作码,还包括诸如DSX状态寄存器的用于DSX状态的显式操作数和诸如DSX嵌套计数寄存器的用于DSX嵌套计数的显式操作数,如2507所示。如之前详述的,DSX状态寄存器可以是专用寄存器、不专用于DSX状态的寄存器(诸如像标志寄存器的总体状态寄存器等等)中的标志,并且DSX嵌套计数可以是专用寄存器、不专用于DSX嵌套计数的寄存器(诸如总体状态寄存器)中的标志。取决于YEND实现方式,使用RTM状态寄存器的隐含操作数。

[0238] 在另一个实施例中,YEND指令不仅包括操作码,还包括诸如DSX状态寄存器的用于DSX状态的显式操作数、诸如DSX嵌套计数寄存器的用于DSX嵌套计数的显式操作数和用于RTM状态的显式操作数,如2509所示。如之前详述的,DSX状态寄存器可以是专用寄存器、不专用于DSX状态的寄存器(诸如像标志寄存器的总体状态寄存器等等)中的标志,并且DSX嵌套计数可以是专用寄存器,寄存器中的标志不专用于DSX嵌套计数(诸如总体状态寄存器)。

[0239] 返回图24,在2403处解码被取出/接收的YEND指令。在一些实施例中,由诸如后文详述的那些硬件解码器解码指令。在一些实施例中,将指令解码为微操作(微op)。例如,一些基于CISC的机器通常使用从宏指令导出的微操作。在其他实施例中,解码是诸如及时编译的软件例程的一部分。

[0240] 在2405处,检索与经解码的YEND指令相关联的任何操作数。例如,检索来自DSX寄存器、DSX嵌套计数寄存器和/或RTM状态寄存器中的一个或多个的数据。

[0241] 在2407处执行经解码的YEND指令。在其中将指令解码为微操作的实施例中,执行这些微操作。对经解码的指令的执行使得硬件进行会被执行的下列动作中的一个或多个:1)使得与DSX相关联的推测写入终结(提交它们);2)通知错误(诸如一般保护错误)并且执行无操作;3)中止DSX;和/或4)结束RTM事务。

[0242] 这些动作中的第一个(使得推测写入终结)使得与DSX相关联的所有推测写入被提交(存储使得它们在DSX外部是可访问的),并且在DSX状态寄存器中将DSX状态设置为指示DSX不存在。例如,提交与DSX相关联的所有写入(诸如存储在高速缓存、寄存器或存储器中的),使得它们被结束并且在DSX外部是可见的。典型地,不能终结DSX,除非该推测的嵌套计数是零。如果嵌套计数大于零,则在一些实施例中执行NOP。

[0243] 如果存在一些原因不可终结DSX,则其他三个可能的动作中的一个或多个发生。例如,在支持RTM的处理器的一些实施例中,如果RTM事务是活动的,则一开始DSX不应当是活动的。在该实例中,RTM事务中有错误,并且应当激活其结束程序,如上述第四个动作指示的。

[0244] 在一些实施例中,如果不存在DSX,则生成错误并且执行无操作(NOP)。例如,如之前详述的,通常将DSX的状态存储在诸如寄存器的可访问位置中,诸如上文参照图1讨论的DSX状态和控制寄存器(DSXSR)。然而,可以利用诸如非专用控制/状态寄存器(诸如FLAGS寄存器)中的DSX状态标志的其他手段。可以由核的硬件检查这个寄存器以确定DSX是否确实发生。

[0245] 在一些实施例中,如果在对事务的提交中有失败,则实现中断程序。例如,在支持RTM的处理器的一些实施例中,激活RTM中断程序。

[0246] 无论执行哪个动作,在大多数实施例中,在该动作之后重置DSX状态(如果它被设

置了)以指示不存在待决DSX。

[0247] 图26示出对诸如YEND指令的指令的执行的详细实施例。例如,在一些实施例中,这个流程是图24的框2407。在一些实施例中,对诸如中央处理单元(CPU)、图形处理单元(GPU)、加速处理单元(APU)、数字信号处理器(DSP)等等的硬件设备的一个或多个硬件核实现该执行。在其他实施例中,对该指令的执行是仿真。

[0248] 在一些实施例中,例如在支持RTM事务的处理器中,在2601处作出对RTM事务是否正在发生的确定。例如,在支持RTM的处理器的一些实施例中,如果RTM事务是活动的,则一开始DSX不应当是活动的。在该实例中,RTM事务中有错误,并且应当激活其结束程序。典型地,将RTM事务状态存储在诸如RTM控制和状态寄存器的寄存器中。处理器的硬件评估该寄存器的内容以确定RTM事务是否正在发生。

[0249] 当RTM事务正在发生时,在2603处作出结束这个RTM事务的调用。例如,调用并执行用于结束RTM事务的指令。此类指令的示例是XEND。

[0250] 当没有RTM事务发生时,在2605处作出对DSX是否活动的确定。如上文详述的,通常将DSX状态存储在控制寄存器中,诸如图1中示出的DSX状态和控制寄存器(DSXSR)。然而,可以利用诸如非专用控制/状态寄存器(诸如FLAGS寄存器)中的DSX状态标志的其他手段。无论状态被存储在何处,由处理器的硬件检查位置以确定DSX是否确实发生。

[0251] 当没有DSX发生时,在2607处生成错误。例如,生成一般保护错误。此外,在一些实施例中,执行无操作(nop)。

[0252] 当DSX发生时,在2609处递减DSX嵌套计数。例如,递减诸如上文详述的存储在DSX嵌套计数寄存器中的存储的DSX嵌套计数。

[0253] 在2611处作出对DSX嵌套计数是否等于零的确定。如上文详述的,通常将DSX嵌套计数存储在寄存器中。当DSX嵌套计数不是零时,在一些实施例中执行NOP。当DSX嵌套计数是零时,在2615处终结并提交当前DSX的推测状态。

[0254] 在2617处作出对提交是否成功的确定。例如,存储中有错误吗?如果否,则在2621处中止DSX。当提交是成功的时,在2619处设置DSX状态指示(诸如存储在DSX状态和控制寄存器中)以指示没有活动的DSX。在一些实施例中,对这个指示的设置在错误的生成2607或对DSX的中止2621之后发生。

[0255] 图27示出显示对诸如YEND指令的指令的执行的伪代码的示例。

[0256] 下文讨论了用于执行上述指令的指令格式和执行资源的实施例。

[0257] 指令集包括一个或多个指令格式。给定指令格式定义各种字段(位的数量、位的位置)以指定将要执行的操作(操作码)以及将对其执行该操作的操作数,等等。通过指令模板(或子格式)的定义来进一步分解一些指令格式。例如,可将给定指令格式的指令模板定义为具有指令格式字段(所包括的字段通常按照相同顺序,但是至少一些字段具有不同的位的位置,因为更少的字段被包括)的不同子集,和/或定义为具有以不同方式进行解释的给定字段。如此,ISA的每一条指令使用给定的指令格式来表达(并且如果经定义,则按照该指令格式的指令模板中的给定指令模板),并包括用于指定操作和操作数的字段。例如,示例性ADD(加法)指令具有特定的操作码和指令格式(包括用于指定该操作码的操作码字段和用于选择操作数(源1/目的地以及源2)的操作数字段),并且该ADD指令在指令流中出现将使得在选择特定操作数的操作数字段中具有特定的内容。已发布和/或出版了被称为高级

向量扩展(AVX)(AVX1和AVX2)并使用向量扩展(VEX)编码方案的SIMD扩展集(例如,参见2011年10月的《Intel® 64和IA-32架构软件开发者手册》(“Intel® 64and IA-32 Architectures Software Developers Manual”);并且参见2011年6月的《Intel® 高级向量扩展编程参考》(“Intel® Advanced Vector Extensions Programming Reference”))。

[0258] 示例性指令格式

[0259] 本文中所描述的指令的实施例可以不同的格式体现。另外,在下文中详述示例性系统、架构、以及流水线。指令的实施例可在这些系统、架构、以及流水线上执行,但是不限于详述的系统、架构、以及流水线。

[0260] 通用向量友好指令格式

[0261] 向量友好指令格式是适于向量指令(例如,存在专用于向量操作的特定字段)的指令格式。尽管描述了其中通过向量友好指令格式支持向量和标量运算两者的实施例,但是替代实施例仅使用通过向量友好指令格式的向量运算。

[0262] 图28A-28B是示出根据本发明的各实施例的通用向量友好指令格式及其指令模板的框图。图28A是示出根据本发明的实施例的通用向量友好指令格式及其A类指令模板的框图;而图28B是示出根据本发明的实施例的通用向量友好指令格式及其B类指令模板的框图。具体而言,为通用向量友好指令格式2800定义了A类和B类指令模板,这两类指令模板都包括无存储器访问2805指令模板和存储器访问2820指令模板。在向量友好指令格式的上下文中的术语“通用”指不束缚于任何专用指令集的指令格式。

[0263] 尽管将描述其中向量友好指令格式支持以下情况的本发明的实施例,即:64字节向量操作数长度(或大小)与32位(4字节)或64位(8字节)数据元素宽度(或大小)(并且由此,64字节向量由16个双字大小的元素或者替代地8个四字大小的元素组成);64字节向量操作数长度(或大小)与16位(2字节)或8位(1字节)数据元素宽度(或大小);32字节向量操作数长度(或大小)与32位(4字节)、64位(8字节)、16位(2字节)、或8位(1字节)数据元素宽度(或大小);以及16字节向量操作数长度(或大小)与32位(4字节)、64位(8字节)、16位(2字节)、或8位(1字节)数据元素宽度(或大小);但是,替代实施例可支持更大、更小、和/或不同的向量操作数大小(例如,256字节向量操作数)与更大、更小或不同的数据元素宽度(例如,128位(16字节)数据元素宽度)。

[0264] 图28A中的A类指令模板包括:1)在无存储器访问2805的指令模板内,示出无存储器访问的完全舍入控制型操作2810的指令模板以及无存储器访问的数据变换型操作2815的指令模板;以及2)在存储器访问2820的指令模板内,示出存储器访问的时效性的2825的指令模板和存储器访问的非时效性的2830的指令模板。图28B中的B类指令模板包括:1)在无存储器访问2805的指令模板内,示出无存储器访问的写掩码控制的部分舍入控制型操作2812的指令模板以及无存储器访问的写掩码控制的vsize型操作2817的指令模板;以及2)在存储器访问2820的指令模板内,示出存储器访问的写掩码控制2827的指令模板。

[0265] 通用向量友好指令格式2800包括下文中按照图28A-28B中所示出的顺序列出的下列字段。

[0266] 格式字段2840——该字段中的特定值(指令格式标识符值)唯一地标识向量友好指令格式,并且由此标识指令在指令流中以向量友好指令格式出现。由此,该字段对于仅具有通用向量友好指令格式的指令集是不需要的,在这个意义上该字段是任选的。

[0267] 基础操作字段2842——其内容区分不同的基础操作。

[0268] 寄存器索引字段2844——其内容直接或者通过地址生成来指定源和目的地操作数在寄存器中或者在存储器中的位置。这些字段包括足够数量的位以从PxQ(例如,32x512、16x128、32x1024、64x1024)个寄存器堆中选择N个寄存器。尽管在一个实施例中N可高达三个源和一个目的地寄存器,但是替代实施例可支持更多或更少的源和目的地寄存器(例如,可支持高达两个源,其中这些源中的一个源还用作目的地,可支持高达三个源,其中这些源中的一个源还用作目的地,可支持高达两个源和一个目的地)。

[0269] 修饰符(modifier)字段2846——其内容将指定存储器访问的以通用向量指令格式出现的指令与不指定存储器访问的以通用向量指令格式出现的指令区分开;即在无存储器访问2805的指令模板与存储器访问2820的指令模板之间进行区分。存储器访问操作读取和/或写入到存储器层次(在一些情况下,使用寄存器中的值来指定源和/或目的地地址),而非存储器访问操作不这样(例如,源和/或目的地是寄存器)。尽管在一个实施例中,该字段还在三种不同的方式之间选择以执行存储器地址计算,但是替代实施例可支持更多、更少或不同的方式来执行存储器地址计算。

[0270] 扩充操作字段2850——其内容区分除基础操作以外还要执行各种不同操作中的哪一个操作。该字段是针对上下文的。在本发明的一个实施例中,此字段被划分为类字段2868、α字段2852以及β字段2854。扩充操作字段2850允许在单条指令而非2条、3条或4条指令中执行多组共同的操作。

[0271] 比例字段2860－其内容允许用于存储器地址生成(例如,用于使用$2^{比例}$*索引+基址的地址生成)的索引字段的内容按比例缩放。

[0272] 位移字段2862A－其内容用作存储器地址生成的部分(例如,用于使用$2^{比例}$*索引+基址+位移的地址生成)。

[0273] 位移因数字段2862B(注意,位移字段2862A直接在位移因数字段2862B上的并置指示使用一个或另一个)－其内容用作地址生成的一部分,它指定通过存储器访问的大小(N)按比例缩放的位移因数,其中N是存储器访问中的字节数量(例如,用于使用$2^{比例}$*索引+基址+按比例缩放的位移的地址生成)。忽略冗余的低阶位,并且因此将位移因数字段的内容乘以存储器操作数总大小(N)以生成在计算有效地址时所使用的最终位移。N的值由处理器硬件在运行时基于完整操作码字段2874(稍后在本文中描述)和数据操纵字段2854C确定。位移字段2862A和位移因数字段2862B不用于无存储器访问2805指令模板,和/或不同的实施例可以实现仅一者或两者都不实现,从这个意义上说,位移字段2862A和位移因数字段2862B是任选的。

[0274] 数据元素宽度字段2864——其内容区分将使用多个数据元素宽度中的哪一个(在一些实施例中,用于所有指令;在其他实施例中,用于指令中的仅一些)。如果支持仅一个数据元素宽度和/或使用操作码的某一方面来支持数据元素宽度,则该字段是不需要的,在这个意义上该字段是任选的。

[0275] 写掩码字段2870——其内容在每一数据元素位置的基础上控制目的地向量操作数中的数据元素位置是否反映基础操作和扩充操作的结果。A类指令模板支持合并-写掩码操作,而B类指令模板支持合并-写掩码操作和归零-写掩码操作两者。当合并时,向量掩码允许在执行任何操作期间保护目的地中的任何元素集免于更新(由基础操作和扩充操作指

定)；在另一实施例中，保持其中对应掩码位具有0的目的地的每一元素的旧值。相反，当归零时，向量掩码允许在执行任何操作期间使目的地中的元素的任何集合归零(由基础操作和扩充操作指定)；在一个实施例中，目的地的元素在对应掩码位具有0值时被设为0。该功能的子集是控制执行的操作的向量长度的能力(即，从第一个到最后一个要修改的元素的跨度)，然而，被修改的元素不一定要是连续的。如此，写掩码字段2870允许部分向量操作，包括加载、存储、算术、逻辑等等。尽管描述了其中写掩码字段2870的内容选择多个写掩码寄存器中的包含要使用的写掩码的一个写掩码寄存器(并且由此，写掩码字段2870的内容间接地标识了要执行的掩码操作)的本发明的多个实施例，但是替代地或附加地，替代实施例允许掩码写字段2870的内容直接地指定要执行的掩码操作。

[0276] 立即数字段2872——其内容允许对立即数的指定。该字段在实现不支持立即数的通用向量友好格式中不存在且在不使用立即数的指令中不存在，在这个意义上该字段是任选的。

[0277] 类字段2868——其内容在不同类的指令之间进行区分。参考图28A-B，该字段的内容在A类和B类指令之间进行选择。在图28A-B中，使用圆角方形来指示在字段中存在专用值(例如，在图28A-B中，分别是针对类字段2868的A类2868A和B类2868B)。

[0278] A类指令模板

[0279] 在A类非存储器访问2805的指令模板的情况下，α字段2852被解释为RS字段2852A，其内容区分将执行不同的扩充操作类型中的哪一种(例如，分别为无存储器访问的舍入型操作2810以及无存储器访问的数据变换型操作2815指令模板指定的舍入2852A.1和数据变换2852A.2)，而β字段2854区别将执行指定的类型的操作中的哪一个。在无存储器访问2805的指令模板中，比例字段2860、位移字段2862A以及位移比例字段2862B不存在。

[0280] 无存储器访问的指令模板——完全舍入控制型操作

[0281] 在无存储器访问的完全舍入控制型操作2810指令模板中，β字段2854被解释为舍入控制字段2854A，其内容提供静态舍入操作。尽管在本发明的所描述的实施例中，舍入控制字段2854A包括抑制所有浮点异常(SAE)字段2856和舍入操作控制字段2858，但是替代实施例可以支持将这两个概念编码为同一个字段，或仅具有这些概念/字段中的一个或另一个(例如，可以仅具有舍入操作控制字段2858)。

[0282] SAE字段2856——其内容区分是否停用异常事件报告；当SAE字段2856的内容指示启用抑制时，给定指令不报告任何种类的浮点异常标志且不唤起任何浮点异常处理程序。

[0283] 舍入操作控制字段2858——其内容区分执行一组舍入操作中的哪一个(例如，向上舍入、向下舍入、向零舍入、以及就近舍入)。如此，舍入操作控制字段2858允许逐指令地改变舍入模式。在其中处理器包括用于指定舍入模式的控制寄存器的本发明的一个实施例中，舍入操作控制字段2850的内容优先于该寄存器值。

[0284] 无存储器访问的指令模板——数据变换型操作

[0285] 在无存储器访问的数据变换型操作2815指令模板中，β字段2854被解释为数据变换字段2854B，其内容区分将执行数个数据变换中的哪一个(例如，无数据变换、混合、广播)。

[0286] 在A类存储器访问2820的指令模板的情况下，α字段2852被解释为驱逐提示字段2852B，其内容区分要使用驱逐提示中的哪一个(在图28A中，对于存储器访问时效性2825的

指令模板和存储器访问非时效性2830的指令模板分别指定时效性的2852B.1和非时效性的2852B.2)，而β字段2854被解释为数据操纵字段2854C，其内容区分要执行多个数据操纵操作(也称为基元(primitive))中的哪一个(例如，无操纵、广播、源的向上转换以及目的地的向下转换)。存储器访问2820指令模板包括比例字段2860，并且任选地包括位移字段2862A或位移比例字段2862B。

[0287] 向量存储器指令使用转换支持来执行来自存储器的向量加载并将向量存储到存储器。如同寻常的向量指令，向量存储器指令以数据元素式的方式与存储器来回传输数据，其中实际传输的元素由选为写掩码的向量掩码的内容规定。

[0288] 存储器访问指令模板——时效性

[0289] 时效性的数据是可能足够快地重新使用以从高速缓存受益的数据。然而，这是提示，且不同的处理器可以不同的方式实现它，包括完全忽略该提示。

[0290] 存储器访问指令模板——非时效性

[0291] 非时效性数据是不大可能足够快地重复使用以从第1级高缓存中的高速缓存操作获益且应当给予驱逐优先级的数据。然而，这是提示，且不同的处理器可以不同的方式实现它，包括完全忽略该提示。

[0292] B类指令模板

[0293] 在B类指令模板的情况下，α字段2852被解释为写掩码控制(Z)字段2852C，其内容区分由写掩码字段2870控制的写掩码操作应当是合并还是归零。

[0294] 在B类非存储器访问2805指令模板的情况下，β字段2854的部分被解释为RL字段2857A，其内容区分将执行不同的扩充操作类型中的哪一种(例如，分别为无存储器访问的写掩码控制部分舍入控制型操作2812指令模板和无存储器访问的写掩码控制VSIZE型操作2817指令模板指定的舍入2857A.1和向量长度(VSIZE)2857A.2)，而β字段2854的其余部分区分将执行指定类型的操作中的哪一个。在无存储器访问2805的指令模板中，比例字段2860、位移字段2862A以及位移比例字段2862B不存在。

[0295] 在无存储器访问的写掩码控制部分舍入控制型操作2810指令模板中，β字段2854的其余部分被解释为舍入操作字段2859A，并且异常事件报告被禁用(给定的指令不报告任何种类的浮点异常标志，并且不引发任何浮点异常处理程序)。

[0296] 舍入操作控制字段2859A——正如舍入操作控制字段2858，其内容区分执行一组舍入操作中的哪一个(例如，向上舍入、向下舍入、向零舍入、以及就近舍入)。由此，舍入操作控制字段2859A允许在每一指令的基础上改变舍入模式。在其中处理器包括用于指定舍入模式的控制寄存器的本发明的一个实施例中，舍入操作控制字段2850的内容优先于该寄存器值。

[0297] 在无存储器访问的写掩码控制VSIZE型操作2817指令模板中，β字段2854的其余部分被解释为向量长度字段2859B，其内容区分将执行数个数据向量长度中的哪一个(例如，128、256或512字节)。

[0298] 在B类存储器访问2820指令模板的情况下，β字段2854的部分被解释为广播字段2857B，其内容区分是否将执行广播类型数据操纵操作，而β字段2854的其余部分被解释为向量长度字段2859B。存储器访问2820指令模板包括比例字段2860，并且任选地包括位移字段2862A或位移比例字段2862B。

[0299] 就通用向量友好指令格式2800而言，完整操作码字段2874示出为包括格式字段2840、基础操作字段2842以及数据元素宽度字段2864。尽管示出了其中完整操作码字段2874包括所有这些字段的一个实施例，但是，在不是支持所有这些字段的实施例中，完整操作码字段2874包括少于全部这些字段。完整操作码字段2874提供操作代码(操作码)。

[0300] 扩充操作字段2850、数据元素宽度字段2864以及写掩码字段2870允许以通用向量友好指令格式逐指令地指定这些特征。

[0301] 写掩码字段和数据元素宽度字段的组合创建各种类型的指令，因为这些指令允许基于不同的数据元素宽度应用该掩码。

[0302] 在A类和B类内出现的各种指令模板在不同的情形下是有益的。在本发明的一些实施例中，不同处理器或者处理器内的不同核可支持仅A类、仅B类、或者可支持两类。举例而言，旨在用于通用计算的高性能通用乱序核可仅支持B类，旨在主要用于图形和/或科学(吞吐量)计算的核可仅支持A类，并且旨在用于两者的核可支持两者(当然，具有来自两类的模板和指令的一些混合、但是并非来自两类的所有模板和指令的核在本发明的范围内)。同样，单一处理器可包括多个核，所有核支持相同的类或者其中不同的核支持不同的类。举例而言，在具有单独的图形和通用核的处理器中，图形核中的旨在主要用于图形和/或科学计算的一个核可仅支持A类，而通用核中的一个或多个可以是具有旨在用于通用计算的仅支持B类的乱序执行和寄存器重命名的高性能通用核。不具有单独的图形核的另一处理器可包括既支持A类又支持B类的一个或多个通用有序或乱序核。当然，在本发明的不同实施例中，来自一类的特征也可在其他类中实现。可使以高级语言撰写的程序成为(例如，及时编译或者静态编译)各种不同的可执行形式，包括：1)仅具有目标处理器支持以执行的类的指令的形式；或者2)具有使用所有类的指令的不同组合而编写的替代例程且具有选择这些例程以基于由当前正在执行代码的处理器支持的指令而执行的控制流代码的形式。

[0303] 示例性专用向量友好指令格式

[0304] 图29是示出根据本发明的实施例的示例性专用向量友好指令格式的框图。图29示出专用向量友好指令格式2900，其指定位置、大小、解释、字段的次序、以及那些字段中的一些字段的值，在这个意义上专用向量友好指令格式1300是专用的。专用向量友好指令格式2900可以被用来扩展x86指令集，并且由此，这些字段中的一些与用于现有的x86指令集及其扩展(例如，AVX)中的那些字段类似或相同。该格式保持与具有扩展的现有x86指令集的前缀编码字段、实操作码字节字段、MOD R/M字段、SIB字段、位移字段、以及立即数字段一致。示出来自图28的字段，来自图29的字段映射到来自图28的字段。

[0305] 应当理解，虽然出于说明的目的在通用向量友好指令格式2800的上下文中，本发明的实施例参考专用向量友好指令格式2900进行了描述，但是本发明不限于专用向量友好指令格式2900，声明的地方除外。例如，通用向量友好指令格式2800构想了各种字段的各种可能的大小，而专用向量友好指令格式2900示出为具有特定大小的字段。作为具体示例，尽管数据元素宽度字段2864示出为专用向量友好指令格式2900中的一个位字段，但是本发明不限于此(也就是说，通用向量友好指令格式2800构想数据元素宽度字段2864的其他大小)。

[0306] 通用向量友好指令格式2800包括下文中按照图29A中所示出的顺序列出的字段。

[0307] EVEX前缀(字节0-3)2902——以四字节形式进行编码。

[0308] 格式字段2840(EVEX字节0,位[7:0])——第一字节(EVEX字节0)是格式字段2840,并且它包含0x62(在本发明的一个实施例中用于区分向量友好指令格式的唯一值)。

[0309] 第二－第四字节(EVEX字节1-3)包括提供专用能力的多个位字段。

[0310] REX字段2905(EVEX字节1,比特[7-5])－由EVEX.R比特字段(EVEX字节1,比特[7]-R)、EVEX.X比特字段(EVEX字节1,比特[6]-X)以及(2857BEX字节1,比特[5]-B)组成。EVEX.R、EVEX.X和EVEX.B位字段提供与对应VEX位字段相同的功能,并且使用1补码的形式进行编码,即ZMM0被编码为1111B,ZMM15被编码为0000B。这些指令的其他字段对如在本领域中已知的寄存器索引的较低三个位(rrr、xxx、以及bbb)进行编码,由此可通过增加EVEX.R、EVEX.X以及EVEX.B来形成Rrrr、Xxxx以及Bbbb。

[0311] REX'字段2810——这是REX'字段2810的第一部分,并且是用于对扩展的32个寄存器集合的较高16个或较低16个寄存器进行编码的EVEX.R'位字段(EVEX字节1,位[4]-R')。在本发明的一个实施例中,该位与以下指示的其他位一起以位反转的格式存储以(在公知x86的32位模式下)与实操作码字节是62的BOUND指令进行区分,但是在MOD R/M字段(在下文中描述)中不接受MOD字段中的值11;本发明的替代实施例不以反转的格式存储该指示的位以及下文中其他指示的位。值1用于对较低16个寄存器进行编码。换句话说,通过组合EVEX.R'、EVEX.R以及来自其他字段的其他RRR来形成R'Rrrr。

[0312] 操作码映射字段2915(EVEX字节1,位[3:0]-mmmm)——其内容对隐含的前导操作码字节(0F、0F 38、或0F 3)进行编码。

[0313] 数据元素宽度字段2864(EVEX字节2,位[7]-W)——由记号EVEX.W表示。EVEX.W用于定义数据类型的粒度(大小)(32位数据元素或64位数据元素)。

[0314] EVEX.vvvv 2920(EVEX字节2,位[6:3]-vvvv)——EVEX.vvvv的作用可包括如下:1)EVEX.vvvv编码第一源寄存器操作数且对具有两个或更多源操作数的指令是有效的,第一源寄存器操作数以反转(1补码)的形式被指定;2)EVEX.vvvv对目的地寄存器操作数编码,该目的地寄存器操作数针对特定向量位移以1补码的形式被指定;或者3)EVEX.vvvv不对任何操作数编码,保留该字段,并且应当包含1111b。由此,EVEX.vvvv字段2920对以反转(1补码)的形式存储的第一源寄存器说明符的4个低阶位进行编码。取决于该指令,额外不同的EVEX位字段用于将说明符大小扩展到32个寄存器。

[0315] EVEX.U 2868类字段(EVEX字节2,位[2]-U)——如果EVEX.U＝0,则它指示A类或EVEX.U0;如果EVEX.U＝1,则它指示B类或EVEX.U1。

[0316] 前缀编码字段2925(EVEX字节2,位[1:0]-pp)——提供了用于基础操作字段的附加位。除了对以EVEX前缀格式的传统SSE指令提供支持以外,这也具有紧缩SIMD前缀的益处(EVEX前缀只需要2位,而不是需要字节来表达SIMD前缀)。在一个实施例中,为了支持使用以传统格式和以EVEX前缀格式两者的SIMD前缀(66H、F2H、F3H)的传统SSE指令,将这些传统SIMD前缀编码成SIMD前缀编码字段;并且在运行时,在提供给解码器的PLA之前被扩展成传统SIMD前缀(因此,该PLA可执行传统格式和EVEX格式的这些传统指令,而无需修改)。虽然较新的指令可将EVEX前缀编码字段的内容直接用作操作码扩展,但是为了一致性,某些实施例以类似的方式扩展,但允许由这些传统SIMD前缀指定不同的含义。替代实施例可重新设计PLA以支持2位SIMD前缀编码,并且由此不需要扩展。

[0317] α字段2852(EVEX字节3,位[7]-EH,也称为EVEX.EH、EVEX.rs、EVEX.RL、EVEX.写掩

码控制以及EVEX.N;也以α示出)－如先前所述,该字段是针对上下文的。

[0318] β字段2854(EVEX字节3,位[6:4]-SSS,也称为EVEX.s_{2-0}、EVEX.r_{2-0}、EVEX.rr1、EVEX.LL0、EVEX.LLB,还以βββ示出)——如前所述,此字段是针对上下文的。

[0319] REX'字段2810——这是REX'字段的其余部分,并且是可用于对扩展的32个寄存器集合的较高16个或较低16个寄存器进行编码的EVEX.V'位字段(EVEX字节3,位[3]-V')。该位以位反转的格式存储。值1用于对较低16个寄存器进行编码。换句话说,通过组合EVEX.V'、EVEX.vvvv来形成V'VVVV。

[0320] 写掩码字段2870(EVEX字节3,位[2:0]-kkk)——其内容指定写掩码寄存器中的寄存器索引,如先前所述。在本发明的一个实施例中,特定值EVEX.kkk＝000具有暗示没有写掩码用于特定指令的特殊行为(这可以各种方式实现,包括使用硬连线成全部为1的写掩码或者旁路掩码硬件的硬件来实现)。

[0321] 实操作码字段2930(字节4)也称为操作码字节。操作码的一部分在该字段中被指定。

[0322] MOD R/M字段2940(字节5)包括MOD字段2942、Reg字段2944以及R/M字段2946。如先前所述的,MOD字段2942的内容将存储器访问和非存储器访问操作区分开。Reg字段2944的作用可被归结为两种情形:对目的地寄存器操作数或源寄存器操作数进行编码;或者被视为操作码扩展且不用于对任何指令操作数进行编码。R/M字段2946的作用可包括如下:对引用存储器地址的指令操作数进行编码,或者对目的地寄存器操作数或源寄存器操作数进行编码。

[0323] 比例、索引、基址(SIB)字节(字节6)——如先前所述的,比例字段2850的内容用于存储器地址生成。SIB.xxx 2954和SIB.bbb 2956——先前已经针对寄存器索引Xxxx和Bbbb提及了这些字段的内容。

[0324] 位移字段2862A(字节7-10)——当MOD字段2942包含10时,字节7-10是位移字段2862A,并且它与传统32位位移(disp32)一样地工作,并且以字节粒度工作。

[0325] 位移因数字段2862B(字节7)——当MOD字段2942包含01时,字节7是位移因数字段2862B。该字段的位置与传统x86指令集8位位移(disp8)的位置相同,它以字节粒度工作。由于disp8是符号扩展的,因此它仅能在-128和127字节偏移量之间寻址;在64字节高速缓存行的方面,disp8使用可被设为仅四个真正有用的值-128、-64、0和64的8位;由于常常需要更大的范围,所以使用disp32;然而,disp32需要4个字节。与disp8和disp32对比,位移因数字段2862B是disp8的重新解释;当使用位移因数字段2862B时,通过将位移因数字段的内容乘以存储器操作数访问的大小(N)来确定实际位移。该类型的位移被称为disp8*N。这减小了平均指令长度(单个字节用于位移,但具有大得多的范围)。这种压缩位移基于有效位移是存储器访问的粒度的倍数的假设,并且由此地址偏移量的冗余低阶位不需要被编码。换句话说,位移因数字段2862B替代传统x86指令集8位位移。由此,位移因数字段2862B以与x86指令集8位位移相同的方式(因此在ModRM/SIB编码规则中没有变化)进行编码,唯一的不同在于,将disp8超载至disp8*N。换句话说,在编码规则或编码长度中没有变化,而仅在通过硬件对位移值的解释中有变化(这需要按存储器操作数的大小按比例缩放位移量以获得字节式地址偏移量)。

[0326] 立即数字段2872如先前所述那样进行操作。

[0327] 完整操作码字段

[0328] 图29B是示出根据本发明的一个实施例的构成完整操作码字段2874的专用向量友好指令格式2900中的字段的框图。具体而言，完整操作码字段2874包括格式字段2840、基础操作字段2842以及数据元素宽度(W)字段2864。基础操作字段2842包括前缀编码字段2925、操作码映射字段2915以及实操作码字段2930。

[0329] 寄存器索引字段

[0330] 图29C是示出根据本发明的一个实施例的构成寄存器索引字段2844的具有专用向量友好指令格式2900的字段的框图。具体而言，寄存器索引字段2844包括REX字段2905、REX'字段2910、MODR/M.reg字段2944、MODR/M.r/m字段2946、VVVV字段2920、xxx字段2954以及bbb字段2956。

[0331] 扩充操作字段

[0332] 图29D是示出根据本发明的一个实施例的构成扩充操作字段2850的专用向量友好指令格式2900中的字段的框图。当类(U)字段2868包含0时，它表示EVEX.U0(A类2868A)；当它包含1时，它表示EVEX.U1(B类2868B)。当U＝0且MOD字段2942包含11(表明无存储器访问操作)时，α字段2852(EVEX字节3，位[7]-EH)被解释为rs字段2852A。当rs字段2852A包含1(舍入2852A.1)时，β字段2854(EVEX字节3、位[6:4]-SSS)被解释为舍入控制字段2854A。舍入控制字段2854A包括一位的SAE字段2856和两位的舍入操作字段2858。当rs字段2852A包含0(数据变换2852A.2)时，β字段2854(EVEX字节3，位[6:4]-SSS)被解释为三位的数据变换字段2854B。当U＝0且MOD字段2942包含00、01或10(表明存储器访问操作)时，α字段2852(EVEX字节3，位[7]-EH)被解释为驱逐提示(EH)字段2852B且β字段2854(EVEX字节3，位[6:4]-SSS)被解释为三位数据操纵字段2854C。

[0333] 当U＝1时，α字段2852(EVEX字节3，位[7]-EH)被解释为写掩码控制(Z)字段2852C。当U＝1且MOD字段2942包含11(表明无存储器访问操作)时，β字段2854的部分(EVEX字节3，位[4]-S_0)被解释为RL字段2857A；当它包含1(舍入2857A.1)时，β字段2854的其余部分(EVEX字节3，位[6-5]-S_{2-1})被解释为舍入操作字段2859A，而当RL字段2857A包含0(VSIZE 2857.A2)时，β字段2854的其余部分(EVEX字节3，位[6-5]-S_{2-1})被解释为向量长度字段2859B(EVEX字节3，位[6-5]-L_{1-0})。当U＝1且MOD字段2942包含00、01或10(表明存储器访问操作)时，β字段2854(EVEX字节3，位[6:4]-SSS)被解释为向量长度字段2859B(EVEX字节3，位[6-5]-L_{1-0})和广播字段2857B(EVEX字节3，位[4]-B)。

[0334] 示例性寄存器架构

[0335] 图30是根据本发明的一个实施例的寄存器架构3000的框图。在所示出的实施例中，有32个512位宽的向量寄存器3010；这些寄存器被引用为zmm0到zmm31。较低的16zmm寄存器的较低阶256个位覆盖在寄存器ymm0-16上。较低的16zmm寄存器的较低阶128个位(ymm寄存器的较低阶128个位)覆盖在寄存器xmm0-15上。专用向量友好指令格式2900按下表中所示方式对这些重叠寄存器堆进行操作。

[0336]

可调节的向量长度	类别	操作	寄存器
不包括向量长度字段2859B的指令模板	A（图28A；U=0）	2810,2815,2825,2830	zmm寄存器（向量长度为64字节）
	B（图28B；U=1）	2812	zmm寄存器（向量长度为64字节）
包括向量长度字段2859B的指令模板	B（图28B；U=1）	2817,2827	zmm、ymm或xmm寄存器（向量长度为64字节、32字节或16字节）取决于向量长度字段2859B

[0337] 换句话说，向量长度字段2859B在最大长度与一个或多个其他较短长度之间进行选择，其中每一这种较短长度是前一长度的一半，并且不具有向量长度字段2859B的指令模板对最大向量长度操作。此外，在一个实施例中，专用向量友好指令格式2900的B类指令模板对紧缩或标量单/双精度浮点数据以及紧缩或标量整数数据操作。标量操作是对zmm/ymm/xmm寄存器中的最低阶数据元素位置执行的操作；取决于本实施例，较高阶数据元素位置保持与在指令之前相同或者归零。

[0338] 写掩码寄存器3015——在所示实施例中，有8个写掩码寄存器(k0到k7)，每一个的大小都是64位。在替代实施例中，写掩码寄存器3015的大小为16位。如先前所述的，在本发明的一个实施例中，向量掩码寄存器k0无法用作写掩码；当正常指示k0的编码用作写掩码时，它选择硬连线的写掩码0xFFFF，从而有效地停用该指令的写掩码操作。

[0339] 通用寄存器3025——在所示实施例中，有十六个64位通用寄存器，这些寄存器与现有的x86寻址模式一起使用来对存储器操作数寻址。这些寄存器通过名称RAX、RBX、RCX、RDX、RBP、RSI、RDI、RSP以及R8到R15来引用。

[0340] 标量浮点栈寄存器堆(x87栈)3045，在其上重叠了MMX紧缩整数平坦寄存器堆3050——在所示出的实施例中，x87栈是用于使用x87指令集扩展来对32/64/80位浮点数据执行标量浮点操作的八元素栈；而使用MMX寄存器来对64位紧缩整数数据执行操作，以及为在MMX和XMM寄存器之间执行的某些操作保存操作数。

[0341] 本发明的替代实施例可以使用较宽的或较窄的寄存器。另外，本发明的替代实施例可以使用更多、更少或不同的寄存器堆和寄存器。

[0342] 示例性核架构、处理器和计算机架构

[0343] 处理器核可以以不同方式、出于不同目的、在不同的处理器中实现。例如，这样的核的实现可以包括：1)旨在用于通用计算的通用有序核；2)旨在用于通用计算的高性能通用乱序核；3)旨在主要用于图形和/或科学(吞吐量)计算的专用核。不同处理器的实现可包括：1)包括旨在用于通用计算的一个或多个通用有序核和/或旨在用于通用计算的一个或多个通用乱序核的CPU；以及2)包括旨在主要用于图形和/或科学(吞吐量)的一个或多个专用核的协处理器。这样的不同处理器导致不同的计算机系统架构，其可包括：1)在与CPU分开的芯片上的协处理器；2)在与CPU相同的封装中但分开的管芯上的协处理器；3)与CPU在相同管芯上的协处理器(在该情况下，这样的协处理器有时被称为诸如集成图形和/或科学

(吞吐量)逻辑等的专用逻辑,或被称为专用核);以及4)可以将所描述的CPU(有时被称为应用核或应用处理器)、以上描述的协处理器和附加功能包括在同一管芯上的芯片上系统。接着描述示例性核架构,随后描述示例性处理器和计算机架构。

[0344] 示例性核架构

[0345] 有序和乱序核框图

[0346] 图31A是示出根据本发明的各实施例的示例性有序流水线和示例性的寄存器重命名的乱序发布/执行流水线的框图。图31B是示出根据本发明的各实施例的要被包括在处理器中的有序架构核的和示例性寄存器重命名的乱序发布/执行架构核的示例性实施例的框图。图31A-B中的实线框示出了有序流水线和有序核,而可选增加的虚线框示出了寄存器重命名的、乱序发布/执行流水线和核。给定有序方面是乱序方面的子集的情况下,将描述乱序方面。

[0347] 在图31A中,处理器流水线3100包括取出级3102、长度解码级3104、解码级3106、分配级3108、重命名级3110、调度(也称为分派或发布)级3112、寄存器读取/存储器读取级3114、执行级3116、写回/存储器写入级3118、异常处理级3122以及提交级3124。

[0348] 图31B示出了包括耦合到执行引擎单元3150的前端单元3130的处理器核3190,且执行引擎单元和前端单元两者都耦合到存储器单元3170。核3190可以是精简指令集计算(RISC)核、复杂指令集计算(CISC)核、超长指令字(VLIW)核或混合或替代核类型。作为又一选项,核3190可以是专用核,诸如例如网络或通信核、紧缩引擎、协处理器核、通用计算图形处理单元(GPGPU)核、图形核、等等。

[0349] 前端单元3130包括耦合至指令高速缓存单元3134的分支预测单元3132,指令高速缓存单元3134耦合至指令转换后备缓冲器(TLB)3136,指令转换后备缓冲器3136耦合至指令取出单元3138,指令取出单元3138耦合至解码单元3140。解码单元3140(或解码器)可解码指令,并生成从原始指令解码出的、或以其它方式反映原始指令的、或从原始指令导出的一个或多个微操作、微代码进入点、微指令、其它指令、或其它控制信号作为输出。解码单元3140可使用各种不同的机制来实现。合适的机制的示例包括但不仅限于,查找表、硬件实现、可编程逻辑阵列(PLA)、微代码只读存储器(ROM)等等。在一个实施例中,核3190包括(例如,在解码单元3140中或以其他方式在前端单元3130内的)用于存储某些宏指令的微代码的微代码ROM或其他介质。解码单元3140耦合至执行引擎单元3150中的重命名/分配器单元3152。

[0350] 执行引擎单元3150包括耦合至引退单元3154的重命名/分配器单元3152以及一组一个或多个调度器单元3156。调度器单元3156表示任何数目的不同调度器,包括预留站、中央指令窗等。调度器单元3156耦合到物理寄存器组单元3158。每个物理寄存器堆单元3158表示一个或多个物理寄存器堆,其中不同的物理寄存器堆存储一种或多种不同的数据类型,诸如标量整数、标量浮点、紧缩整数、紧缩浮点、向量整数、向量浮点、状态(例如,作为要执行的下一指令的地址的指令指针)等。在一个实施例中,物理寄存器堆单元3158包括向量寄存器单元、写掩码寄存器单元和标量寄存器单元。这些寄存器单元可以提供架构向量寄存器、向量掩码寄存器、和通用寄存器。物理寄存器堆单元3158与引退单元3154重叠以示出可以用来实现寄存器重命名和乱序执行的各种方式(例如,使用重新排序缓冲器和引退寄存器堆;使用将来的文件、历史缓冲器和引退寄存器堆;使用寄存器映射和寄存器池等等)。

引退单元3154和物理寄存器堆单元3158耦合到执行群集3160。执行群集3160包括一组一个或多个执行单元3162和一组一个或多个存储器访问单元3164。执行单元3162可以对各种类型的数据(例如,标量浮点、紧缩整数、紧缩浮点、向量整数、向量浮点)执行各种操作(例如,移位、加法、减法、乘法)。尽管一些实施例可以包括专用于特定功能或功能组的数个执行单元,但是其他实施例可以仅包括一个执行单元或全部都执行所有功能的多个执行单元。调度器单元3156、物理寄存器组单元3158和执行群集3160被示为可能有多个,因为某些实施例为某些类型的数据/操作创建分开的流水线(例如,各自具有其自身的调度器单元、物理寄存器组单元和/或执行群集的标量整数流水线、标量浮点/紧缩整数/紧缩浮点/向量整数/向量浮点流水线和/或存储器访问流水线——以及在分开的存储器访问流水线的情况下,实现其中仅该流水线的执行群集具有存储器访问单元3164的某些实施例)。还应当理解,在使用分开的流水线的情况下,这些流水线中的一个或多个可以是乱序发布/执行的,而其余的是有序的。

[0351] 存储器访问单元3164的集合耦合到存储器单元3170,该存储器单元包括耦合到数据高速缓存单元3174的数据TLB单元3172,其中数据高速缓存单元耦合到第2级(L2)高速缓存单元3176。在一个示例性实施例中,存储器访问单元3164可包括加载单元、存储地址单元和存储数据单元,其中的每一个均耦合至存储器单元3170中的数据TLB单元3172。指令高速缓存单元3134还耦合到存储器单元3170中的第2级(L2)高速缓存单元3176。L2高速缓存单元3176被耦合到一个或多个其他级的高速缓存,并最终被耦合到主存储器。

[0352] 作为示例,示例性寄存器重命名的、乱序发布/执行核架构可以如下实现流水线3100:1)指令取出3138执行取出和长度解码级3102和3104;2)解码单元3140执行解码级3106;3)重命名/分配器单元3152执行分配级3108和重命名级3110;4)调度器单元3156执行调度级3112;5)物理寄存器堆单元3158和存储器单元3170执行寄存器读取/存储器读取级3114;执行群集3160执行执行级3116;6)存储器单元3170和物理寄存器堆单元3158执行写回/存储器写入级3118;7)各单元可牵涉到异常处理级3122;以及8)引退单元3154和物理寄存器堆单元3158执行提交级3124。

[0353] 核3190可支持一个或多个指令集(例如,x86指令集(具有与较新版本一起添加的一些扩展);加利福尼亚州桑尼维尔市的MIPS技术公司的MIPS指令集;加利福尼州桑尼维尔市的ARM控股的ARM指令集(具有诸如NEON等可选附加扩展)),其中包括本文中描述的各指令。在一个实施例中,核3190包括用于支持紧缩数据指令集扩展(例如,AVX1、AVX2)的逻辑,进而允许由许多多媒体应用使用的操作通过使用紧缩数据来执行。

[0354] 应当理解,核可以支持多线程操作(执行两个或更多个并行的操作或线程的集合),并且可以按各种方式来完成该多线程操作,各种方式包括时分多线程操作、同步多线程操作(其中,单个物理核为物理核正在同步进行多线程操作的多个线程中的每一个线程提供逻辑核)或其组合(例如,时分取出和解码以及此后诸如利用Intel®超线程技术的同步多线程操作)。

[0355] 尽管在乱序执行的上下文中描述了寄存器重命名,但是,应当理解,寄存器重命名可以用于有序架构中。尽管所示出的处理器的实施例还包括分开的指令和数据高速缓存单元3134/3174以及共享L2高速缓存单元3176,但替代实施例可以具有用于指令和数据两者的单个内部高速缓存,诸如例如第1级(L1)内部高速缓存或多个级别的内部高速缓存。在一

些实施例中，系统可以包括内部高速缓存以及在核和/或处理器外部的外部高速缓存的组合。或者，全部高速缓存都可以在核和/或处理器的外部。

[0356] 具体的示例性有序核架构

[0357] 图32A-B示出更具体的示例性有序核架构的框图，该核将是芯片中的多个逻辑块中的一个(包括相同类型和/或不同类型的其他核)。根据应用，这些逻辑块通过高带宽的互连网络(例如，环形网络)与一些固定的功能逻辑、存储器I/O接口和其它必要的I/O逻辑通信。

[0358] 图32A是根据本发明的各实施例的单个处理器核以及它与管芯上互连网络3202的连接及其第2级(L2)高速缓存的本地子集3204的框图。在一个实施例中，指令解码器3200支持具有紧缩数据指令集扩展的x86指令集。L1高速缓存3206允许对进入标量和向量单元中的高速缓存存储器的低等待时间访问。尽管在一个实施例中(为了简化设计)，标量单元3208和向量单元3210使用分开的寄存器集合(分别为标量寄存器3212和向量寄存器3214)，并且在这些寄存器之间转移的数据被写入到存储器并随后从第一级(L1)高速缓存3206读回，但是本发明的替代实施例可以使用不同的方法(例如，使用单个寄存器集合或包括允许数据在这两个寄存器堆之间传输而无需被写入和读回的通信路径)。

[0359] L2高速缓存的本地子集3204是全局L2高速缓存的一部分，该全局L2高速缓存被划分成多个分开的本地子集，即每个处理器核一个本地子集。每个处理器核具有到其自己的L2高速缓存的本地子集3204的直接访问路径。被处理器核读出的数据被存储在其L2高速缓存子集3204中，并且可以与其它处理器核访问其自己的本地L2高速缓存子集并行地被快速访问。被处理器核写入的数据被存储在其自己的L2高速缓存子集3204中，并在必要的情况下从其它子集转储清除。环形网络确保共享数据的一致性。环形网络是双向的，以允许诸如处理器核、L2高速缓存和其它逻辑块之类的代理在芯片内彼此通信。每个环形数据路径为每个方向1012位宽。

[0360] 图32B是根据本发明的实施例的图32A中的处理器核的一部分的展开图。图32B包括L1高速缓存3204的L1数据高速缓存3206A部分，以及关于向量单元3210和向量寄存器3214的更多细节。具体地说，向量单元3210是16宽向量处理单元(VPU)(见16宽ALU 3228)，该单元执行整型、单精度浮点以及双精度浮点指令中的一个或多个。该VPU通过混合单元3220支持对寄存器输入的混合、通过数值转换单元3222A-B支持数值转换、并通过复制单元3224支持对存储器输入的复制。写掩码寄存器3226允许断言所得的向量写入。

[0361] 具有集成存储器控制器和图形器件的处理器

[0362] 图33是根据本发明的实施例的可具有多于一个的核、可具有集成存储器控制器、以及可具有集成图形器件的处理器3300的框图。图33的实线框示出了处理器3300，处理器3300具有单个核3302A、系统代理3310、一组一个或多个总线控制器单元3316，而可选附加的虚线框示出了替代处理器3300，其具有多个核3302A-N、系统代理单元3310中的一组一个或多个集成存储器控制器单元3314以及专用逻辑3308。

[0363] 因此，处理器3300的不同实现可包括：1)CPU，其中专用逻辑3308是集成图形和/或科学(吞吐量)逻辑(其可包括一个或多个核)，并且核3302A-N是一个或多个通用核(例如，通用的有序核、通用的乱序核、这两者的组合)；2)协处理器，其中核3302A-N是旨在主要用于图形和/或科学(吞吐量)的多个专用核；以及3)协处理器，其中核3302A-N是多个通用有

序核。因此，处理器3300可以是通用处理器、协处理器或专用处理器，诸如例如网络或通信处理器、紧缩引擎、图形处理器、GPGPU(通用图形处理单元)、高吞吐量的集成众核(MIC)协处理器(包括30个或更多核)、或嵌入式处理器等。该处理器可以被实现在一个或多个芯片上。处理器3300可以是一个或多个衬底的一部分，和/或可以使用诸如例如BiCMOS、CMOS或NMOS等的多个工艺技术中的任何一个技术将该处理器实现在一个或多个衬底上。

[0364] 存储器层次结构包括核内的一个或多个层级的高速缓存、一组或一个或多个共享高速缓存单元3306以及耦合到集成存储器控制器单元3314的集合的外部存储器(未示出)。共享高速缓存单元3306的集合可以包括一个或多个中级高速缓存，诸如，第2级(L2)、第3级(L3)、第4级(L4)，或其他层级的高速缓存，末级高速缓存(LLC)，和/或上述各项的组合。尽管在一个实施例中，基于环的互连单元3312将集成图形逻辑3308、共享高速缓存单元3306的集合以及系统代理单元3310/集成存储器控制器单元3314互连，但替代实施例可使用任何数量的公知技术来将这些单元互连。在一个实施例中，可以维护一个或多个高速缓存单元3306和核3302-A-N之间的一致性。

[0365] 在一些实施例中，核3302A-N中的一个或多个能够实现多线程。系统代理3310包括协调并操作核3302A-N的那些组件。系统代理单元3310可包括例如功率控制单元(PCU)和显示单元。PCU可以是或可包括用于调节核3302A-N和集成图形逻辑3308的功率状态所需的逻辑和组件。显示单元用于驱动一个或多个从外部连接的显示器。

[0366] 核3302A-N在架构指令集方面可以是同构的或异构的；即，这些核3302A-N中的两个或更多个核可能能够执行相同的指令集，而其它核可能能够执行该指令集的仅仅子集或不同的指令集。

[0367] 示例性计算机架构

[0368] 图34-37是示例性计算机架构的框图。本领域已知的对膝上型设备、台式机、手持PC、个人数字助理、工程工作站、服务器、网络设备、网络集线器、交换机、嵌入式处理器、数字信号处理器(DSP)、图形设备、视频游戏设备、机顶盒、微控制器、蜂窝电话、便携式媒体播放器、手持设备以及各种其它电子设备的其它系统设计和配置也是合适的。一般地，能够包含本文中所公开的处理器和/或其它执行逻辑的多个系统或电子设备一般都是合适的。

[0369] 现在参见图34，所示是根据本发明的一个实施例的系统3400的框图。系统3400可以包括一个或多个处理器3410、3415，这些处理器耦合到控制器中枢3420。在一个实施例中，控制器中枢3420包括图形存储器控制器中枢(GMCH)3490和输入/输出中枢(IOH)3450(其可以在分开的芯片上)；GMCH 3490包括存储器和图形控制器，存储器3440和协处理器3445耦合到该存储器和图形控制器；IOH 3450将输入/输出(I/O)设备3460耦合到GMCH 3490。或者，存储器和图形控制器中的一个或两者被集成在处理器内(如本文中所描述的)，存储器3440和协处理器3445直接耦合到处理器3410以及控制器中枢3420，该控制器中枢与IOH 3450处于单个芯片中。

[0370] 在图34中以虚线表示附加的处理器3415的可选的性质。每一个处理器3410、3415可包括本文中描述的处理核中的一个或多个，并且可以是处理器3300的某一版本。

[0371] 存储器3440可以是例如动态随机存取存储器(DRAM)、相变存储器(PCM)或这两者的组合。对于至少一个实施例，控制器中枢3420经由诸如前端总线(FSB)之类的多分支总线、诸如快速通道互连(QPI)之类的点对点接口、或者类似的连接3495与处理器3410、3415

进行通信。

[0372] 在一个实施例中，协处理器3445是专用处理器，诸如例如高吞吐量MIC处理器、网络或通信处理器、紧缩引擎、图形处理器、GPGPU、或嵌入式处理器等等。在一个实施例中，控制器中枢3420可以包括集成图形加速器。

[0373] 在物理资源3410、3415之间会存在包括架构、微架构、热、功耗特性等的一系列品质度量方面的各种差异。

[0374] 在一个实施例中，处理器3410执行控制一般类型的数据处理操作的指令。协处理器指令可嵌入在这些指令中。处理器3410将这些协处理器指令识别为应当由附连的协处理器3445执行的类型。因此，处理器3410在协处理器总线或者其它互连上将这些协处理器指令(或者表示协处理器指令的控制信号)发布到协处理器3445。协处理器3445接受并执行所接收的协处理器指令。

[0375] 现在参见图35，所示是根据本发明的实施例的第一更具体的示例性系统3500的框图。如图35所示，多处理器系统3500是点对点互连系统，并且包括经由点对点互连3550耦合的第一处理器3570和第二处理器3580。处理器3570和3580中的每一个都可以是处理器3300的某一版本。在本发明的一个实施例中，处理器3570和3580分别是处理器3410和3415，而协处理器3538是协处理器3445。在另一实施例中，处理器3570和3580分别是处理器3410和协处理器3445。

[0376] 处理器3570和3580被示为分别包括集成存储器控制器(IMC)单元3572和3582。处理器3570还包括点对点(P-P)接口3576和3578作为其总线控制器单元的部分；类似地，第二处理器3580包括P-P接口3586和3588。处理器3570、3580可以经由使用点对点(P-P)接口电路3578、3588的P-P接口3550来交换信息。如图35所示，IMC 3572和3582将处理器耦合至相应的存储器，即，存储器3532和存储器3534，它们可以是本地连接到相应的处理器的主存储器的部分。

[0377] 处理器3570、3580可各自经由使用点对点接口电路3576、3594、3586、3598的各个P-P接口3552、3554与芯片组3590交换信息。芯片组3590可任选地经由高性能接口3539与协处理器3538交换信息。在一个实施例中，协处理器3538是专用处理器，诸如例如高吞吐量MIC处理器、网络或通信处理器、压缩引擎、图形处理器、GPGPU、或嵌入式处理器等等。

[0378] 共享高速缓存(未示出)可被包括在任一处理器中，或在两个处理器的外部但经由P-P互连与这些处理器连接，从而如果处理器被置于低功率模式，则任一个或这两个处理器的本地高速缓存信息可被存储在该共享的高速缓存中。

[0379] 芯片组3590可经由接口3596耦合至第一总线3516。在一个实施例中，第一总线3516可以是外围组件互连(PCI)总线，或诸如PCI Express总线或其它第三代I/O互连总线之类的总线，但本发明的范围并不受此限制。

[0380] 如图35所示，各种I/O设备3514可以连同总线桥3518被耦合至第一总线3516，总线桥3518将第一总线3516耦合至第二总线3520。在一个实施例中，诸如协处理器、高吞吐量MIC处理器、GPGPU的处理器、加速器(例如，图形加速器或数字信号处理(DSP)单元)、现场可编程门阵列或任何其他处理器之类的一个或多个附加处理器3515耦合到第一总线3516。在一个实施例中，第二总线3520可以是低引脚数(LPC)总线。在一个实施例中，各种设备可以耦合到第二总线3520，各种设备包括例如，键盘和/或鼠标3522、通信设备3527以及存储单

元3528,存储单元3528诸如,可包括指令/代码和数据3530的磁盘驱动器或其他大容量存储设备。此外,音频I/O 3524可以被耦合至第二总线3520。注意,其他架构是可能的。例如,系统可实现多分支总线或者其他此类架构,而不是图35中的点对点架构。

[0381] 现在参考图36,所示为根据本发明的实施例的第二更具体的示例性系统3600的框图。图35和36中的类似元件使用类似附图标记,且在图36中省略了图35的某些方面以避免混淆图36的其它方面。

[0382] 图36示出处理器3570、3580可分别包括集成存储器和I/O控制逻辑("CL")3572和3582。因此,CL 3572、3582包括集成存储器控制器单元并包括I/O控制逻辑。图36示出不仅存储器3532、3534耦合到CL 3572、3582,而且I/O设备3614也耦合到控制逻辑3572、3582。传统I/O设备3615被耦合至芯片组3590。

[0383] 现在参照图37,所示出的是根据本发明的实施例的SoC 3700的框图。图33中相似的部件具有同样的附图标记。另外,虚线框是更先进的SoC的可选特征。在图37中,互连单元3702被耦合至:应用处理器3710,该应用处理器包括一个或多个核202A-N的集合以及共享高速缓存单元3306;系统代理单元3310;总线控制器单元3316;集成存储器控制器单元3314;一组或一个或多个协处理器3720,其可包括集成图形逻辑、图像处理器、音频处理器和视频处理器;静态随机存取存储器(SRAM)单元3730;直接存储器存取(DMA)单元3732;以及用于耦合至一个或多个外部显示器的显示单元3740。在一个实施例中,协处理器3720包括专用处理器,诸如例如网络或通信处理器、紧缩引擎、GPGPU、高吞吐量MIC处理器、或嵌入式处理器等等。

[0384] 本文公开的机制的各实施例可以被实现在硬件、软件、固件或这些实现方法的组合中。本发明的实施例可实现为在可编程系统上执行的计算机程序或程序代码,该可编程系统包括至少一个处理器、存储系统(包括易失性和非易失性存储器和/或存储元件)、至少一个输入设备以及至少一个输出设备。

[0385] 可将程序代码(诸如图35中示出的代码3530)应用于输入指令,以执行本文描述的各功能并生成输出信息。可以按已知方式将输出信息应用于一个或多个输出设备。为了本申请的目的,处理系统包括具有诸如例如数字信号处理器(DSP)、微控制器、专用集成电路(ASIC)或微处理器之类的处理器的任何系统。

[0386] 程序代码可以用高级程序化语言或面向对象的编程语言来实现,以便与处理系统通信。在需要时,也可用汇编语言或机器语言来实现程序代码。事实上,本文中描述的机制不限于任何特定编程语言的范围。在任一情形下,该语言可以是编译语言或解释语言。

[0387] 至少一个实施例的一个或多个方面可以由存储在机器可读介质上的表示性指令来实现,指令表示处理器中的各种逻辑,指令在被机器读取时使得该机器制作用于执行本文所述的技术的逻辑。可将被称为"IP核"的此类表示存储在有形的机器可读介质上,并将其提供给各种客户或生产设施以加载到实际制造该逻辑或处理器的制造机器中。

[0388] 此类机器可读存储介质可以包括但不限于通过机器或设备制造或形成的制品的非暂态的有形安排,其包括存储介质,诸如,硬盘;任何其他类型的盘,包括软盘、光盘、紧致盘只读存储器(CD-ROM)、紧致盘可重写(CD-RW)以及磁光盘;半导体器件,诸如,只读存储器(ROM)、诸如动态随机存取存储器(DRAM)和静态随机存取存储器(SRAM)之类的随机存取存储器(RAM)、可擦除可编程只读存储器(EPROM)、闪存、电可擦除可编程只读存储器

(EEPROM)；相变存储器(PCM)；磁卡或光卡；或适于存储电子指令的任何其他类型的介质。

[0389] 因此，本发明的各实施例还包括非暂态有形机器可读介质，该介质包含指令或包含设计数据，诸如硬件描述语言(HDL)，它定义本文中描述的结构、电路、装置、处理器和/或系统特征。这些实施例也被称为程序产品。

[0390] 仿真(包括二进制变换、代码变形等)

[0391] 在一些情况下，指令转换器可用来将指令从源指令集转换至目标指令集。例如，指令转换器可以变换(例如使用静态二进制变换、包括动态编译的动态二进制变换)、变形、仿真或以其它方式将指令转换成将由核来处理的一个或多个其它指令。指令转换器可以用软件、硬件、固件、或其组合实现。指令转换器可以在处理器上、在处理器外、或者部分在处理器上且部分在处理器外。

[0392] 图38是根据本发明的实施例的对照使用软件指令转换器将源指令集中的二进制指令转换成目标指令集中的二进制指令的框图。在所示的实施例中，指令转换器是软件指令转换器，但作为替代，该指令转换器可以用软件、固件、硬件或其各种组合来实现。图38示出可以使用x86编译器3804来编译高级语言3802形式的程序以生成可由具有至少一个x86指令集核的处理器3816原生地执行的x86二进制代码3806。具有至少一个x86指令集核的处理器3816表示能通过兼容地执行或以其他方式处理以下内容来执行与具有至少一个x86指令集核的英特尔处理器基本相同的功能的任何处理器：(1)英特尔x86指令集核的指令集的本质部分，或(2)目标为在具有至少一个x86指令集核的英特尔处理器上运行以实现与具有至少一个x86指令集核的英特尔处理器基本相同的结果的应用或其他软件的目标代码版本。x86编译器3804表示用于生成x86二进制代码3806(例如，目标代码)的编译器，该x86二进制代码3806可利用或不利用附加的链路处理而在具有至少一个x86指令集核的处理器3816上执行。类似地，图38示出可以使用替代的指令集编译器3808来编译高级语言3802的程序以生成可由不具有至少一个x86指令集核的处理器3814(例如，具有执行加利福尼亚州桑尼维尔市的MIPS技术公司的MIPS指令集和/或执行加利福尼州桑尼维尔市的ARM控股公司的ARM指令集的核的处理器)原生地执行的替代的指令集二进制代码3810。指令转换器3812用于将x86二进制代码3806转换成可以由不具有x86指令集核的处理器3814原生地执行的代码。该转换后的代码不大可能与替代的指令集二进制代码3810相同，因为能够这样做的指令转换器难以制造；然而，转换后的代码将完成通用操作，并且将由来自替代指令集的指令构成。因此，指令转换器3812表示软件、固件、硬件或它们的组合，这些软件、固件、硬件或它们的组合通过仿真、模拟或任何其他过程允许不具有x86指令集处理器或核的处理器或其他电子设备执行x86二进制代码3806。

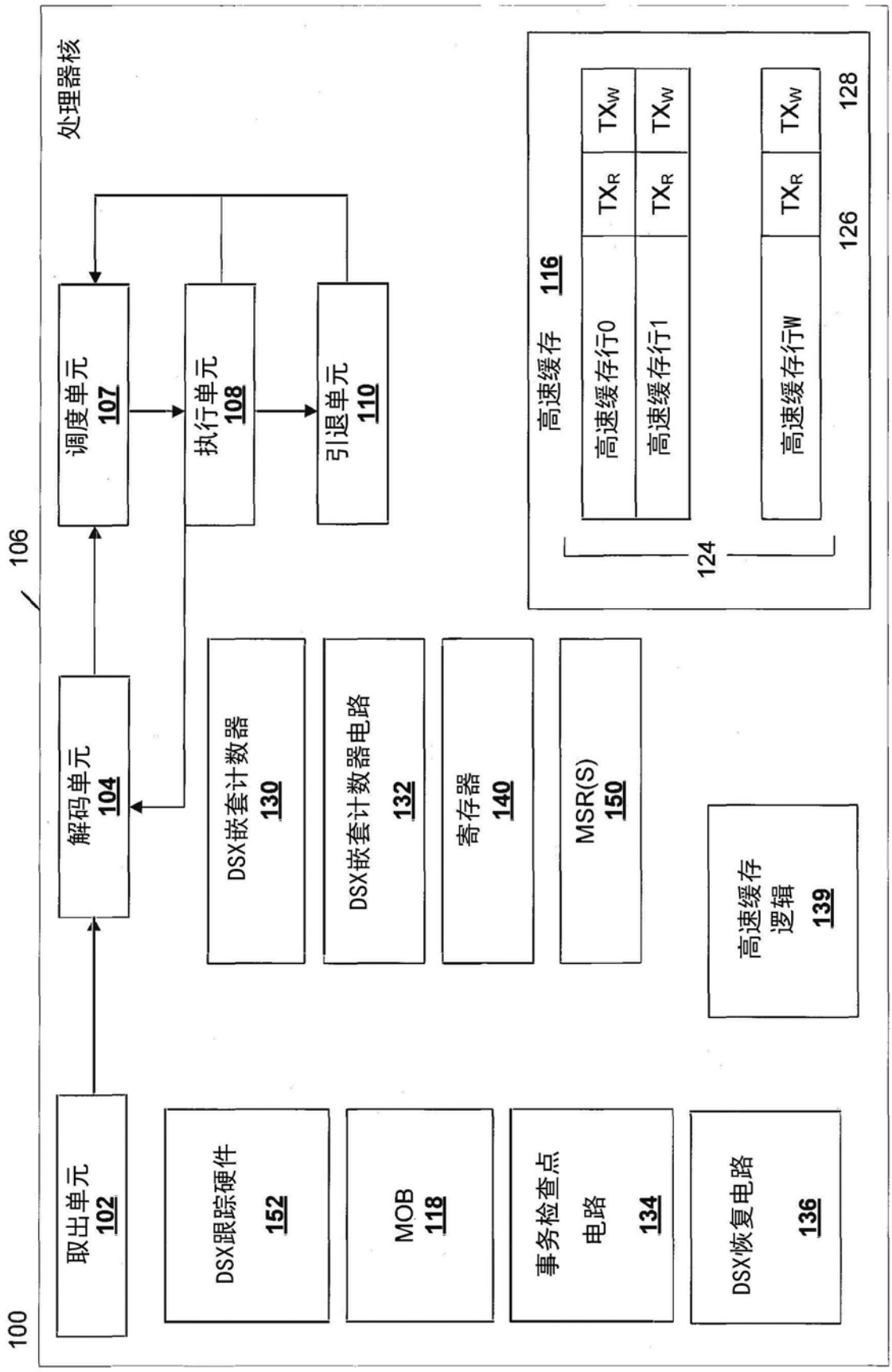

100
处理器核
106
取出单元 102
解码单元 104
调度单元 107
执行单元 108
引退单元 110
高速缓存 116
高速缓存行0
高速缓存行1
高速缓存行W
TX_R
TX_W
124
126
128
DSX嵌套计数器 130
DSX嵌套计数器电路 132
寄存器 140
MSR(S) 150
高速缓存逻辑 139
DSX跟踪硬件 152
MOB 118
事务检查点电路 134
DSX恢复电路 136

图1

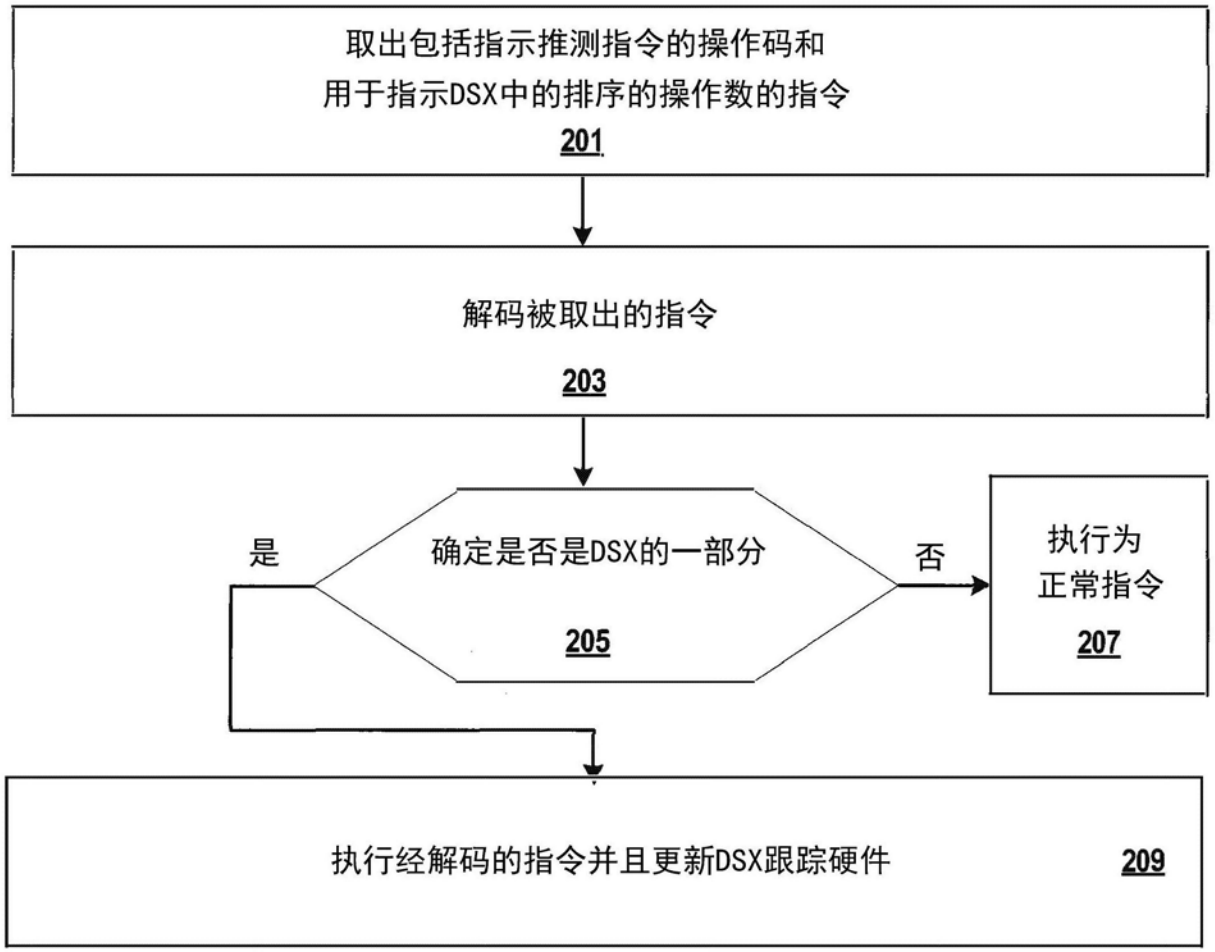

取出包括指示推测指令的操作码和
用于指示DSX中的排序的操作数的指令
201
解码被取出的指令
203
确定是否是DSX的一部分
205
是
否
执行为
正常指令
207
执行经解码的指令并且更新DSX跟踪硬件
209

图2

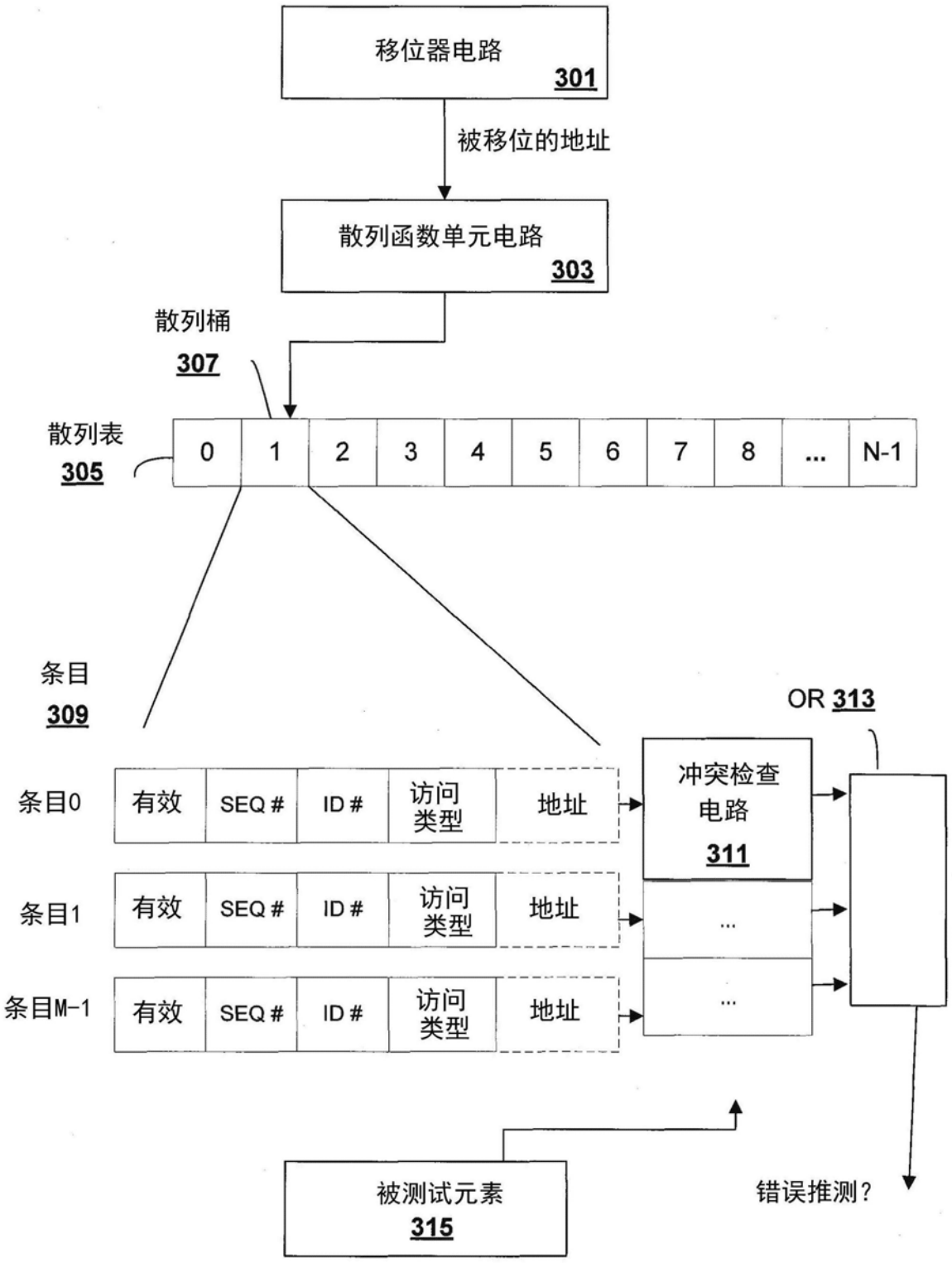

移位器电路
301
被移位的地址
散列函数单元电路
303
散列桶
307
散列表
305
0
1
2
3
4
5
6
7
8
...
N-1
条目
309
OR 313
条目0
有效
SEQ #
ID #
访问类型
地址
冲突检查电路
311
条目1
有效
SEQ #
ID #
访问类型
地址
...
条目M-1
有效
SEQ #
ID #
访问类型
地址
...
被测试元素
315
错误推测?

图3

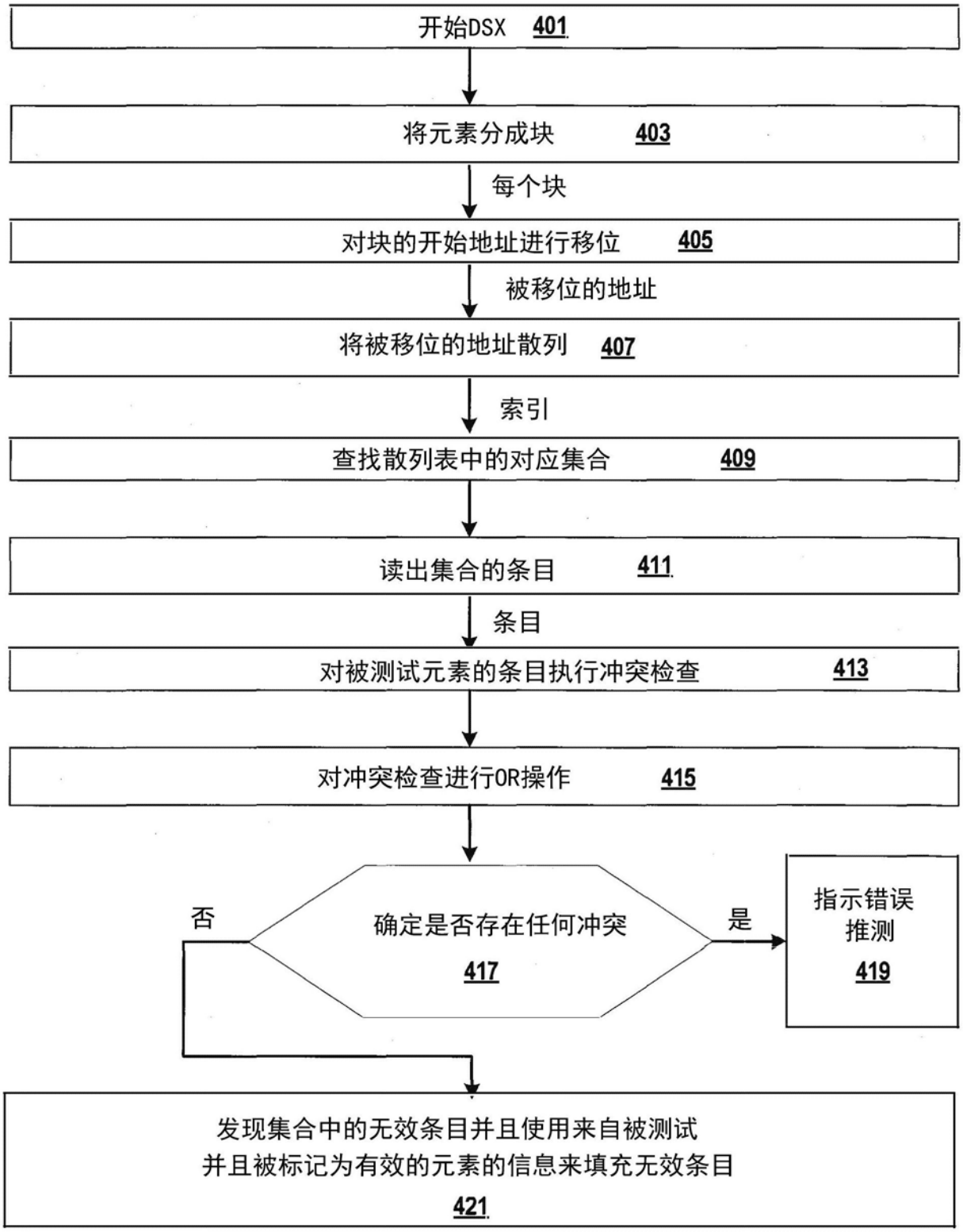

开始DSX 401
将元素分成块 403
每个块
对块的开始地址进行移位 405
被移位的地址
将被移位的地址散列 407
索引
查找散列表中的对应集合 409
读出集合的条目 411
条目
对被测试元素的条目执行冲突检查 413
对冲突检查进行OR操作 415
否
确定是否存在任何冲突
417
是
指示错误
推测
419
发现集合中的无效条目并且使用来自被测试
并且被标记为有效的元素的信息来填充无效条目
421

图4

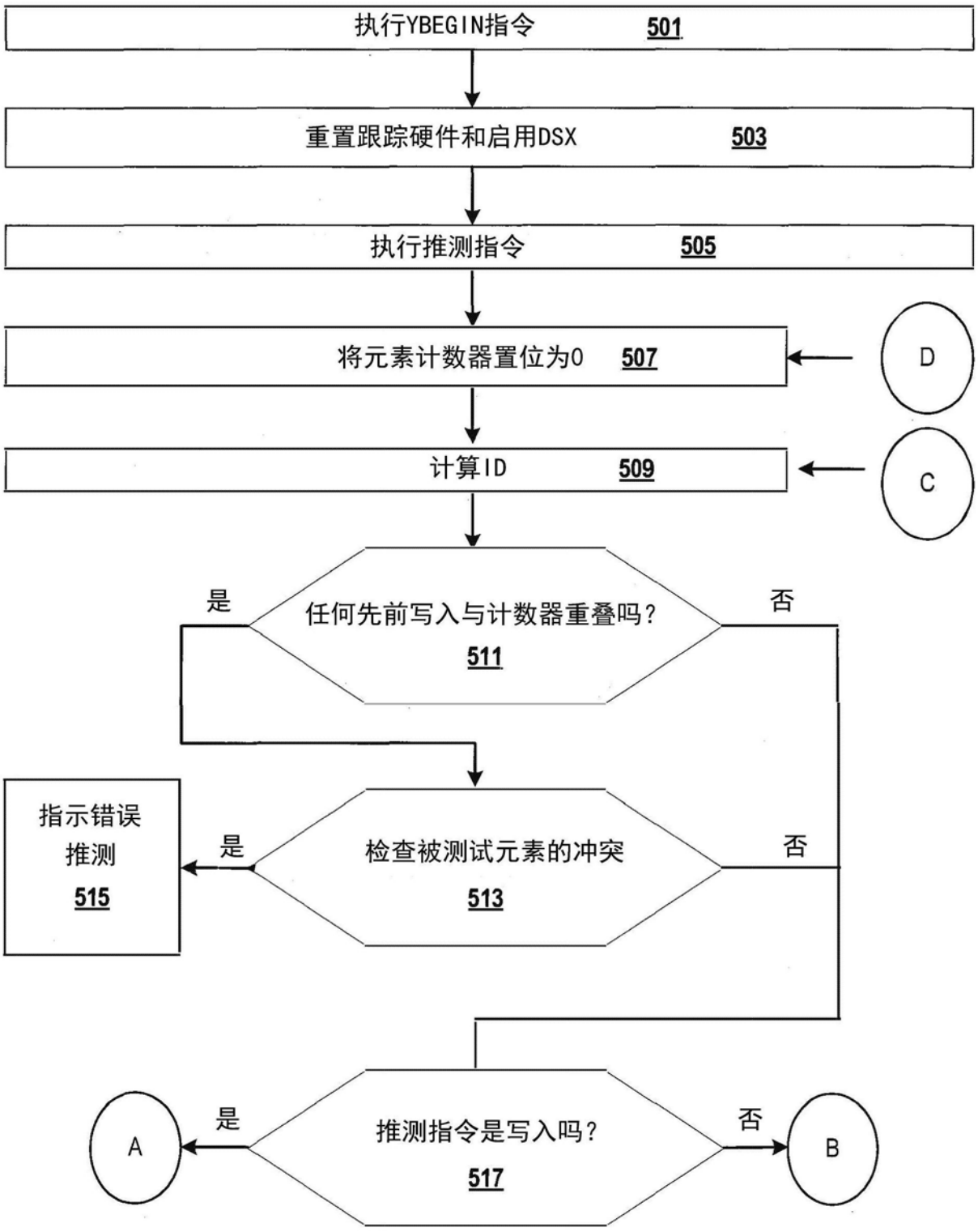

执行YBEGIN指令 501
重置跟踪硬件和启用DSX 503
执行推测指令 505
将元素计数器置位为0 507
D
计算ID 509
C
是
任何先前写入与计数器重叠吗？ 511
否
指示错误推测 515
是
检查被测试元素的冲突 513
否
A
是
推测指令是写入吗？ 517
否
B

图5A

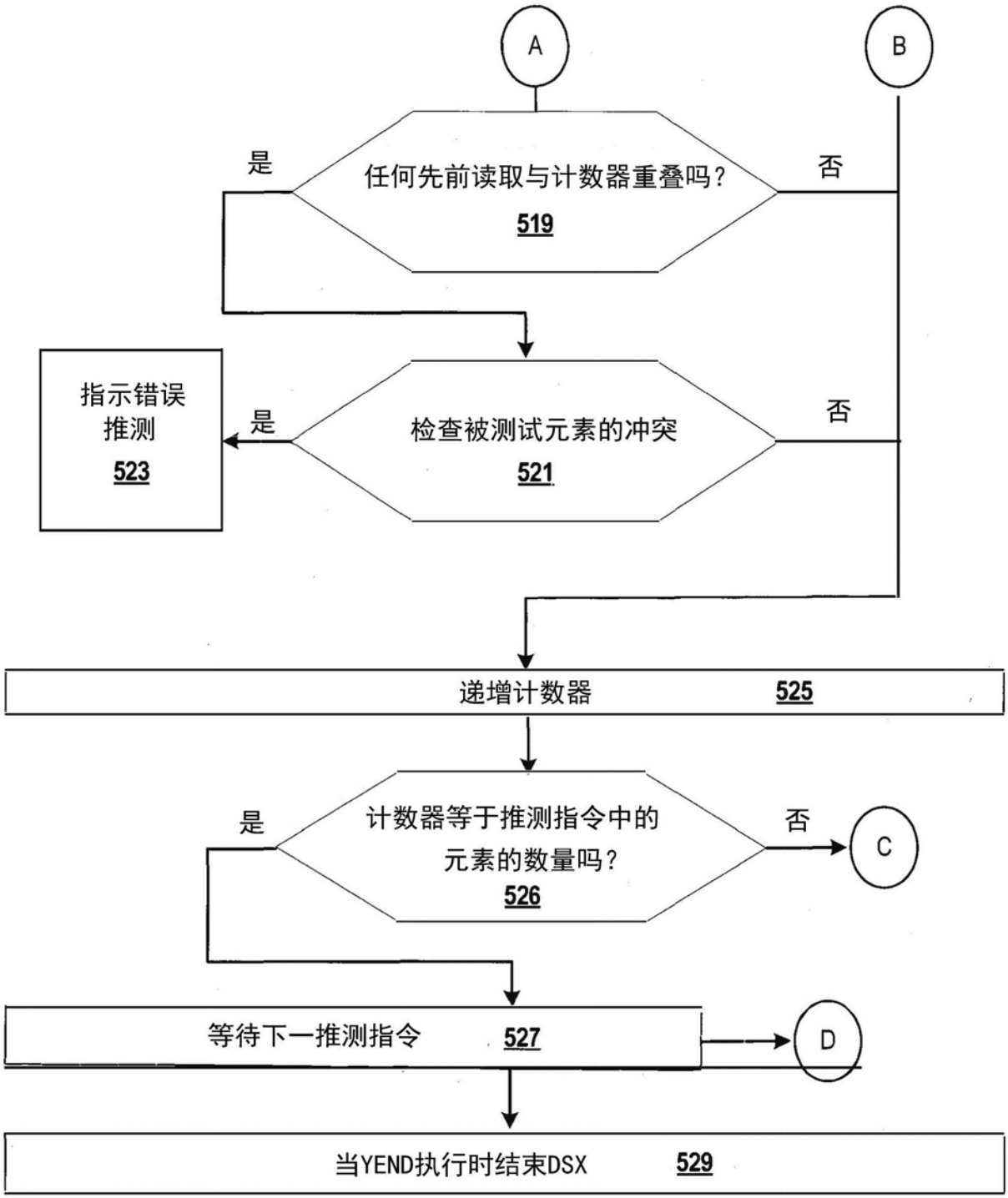

A
B
是
任何先前读取与计数器重叠吗？
519
否
指示错误
推测
523
是
检查被测试元素的冲突
521
否
递增计数器 525
是
计数器等于推测指令中的
元素的数量吗？
526
否
C
等待下一推测指令 527
D
当YEND执行时结束DSX 529

图5B

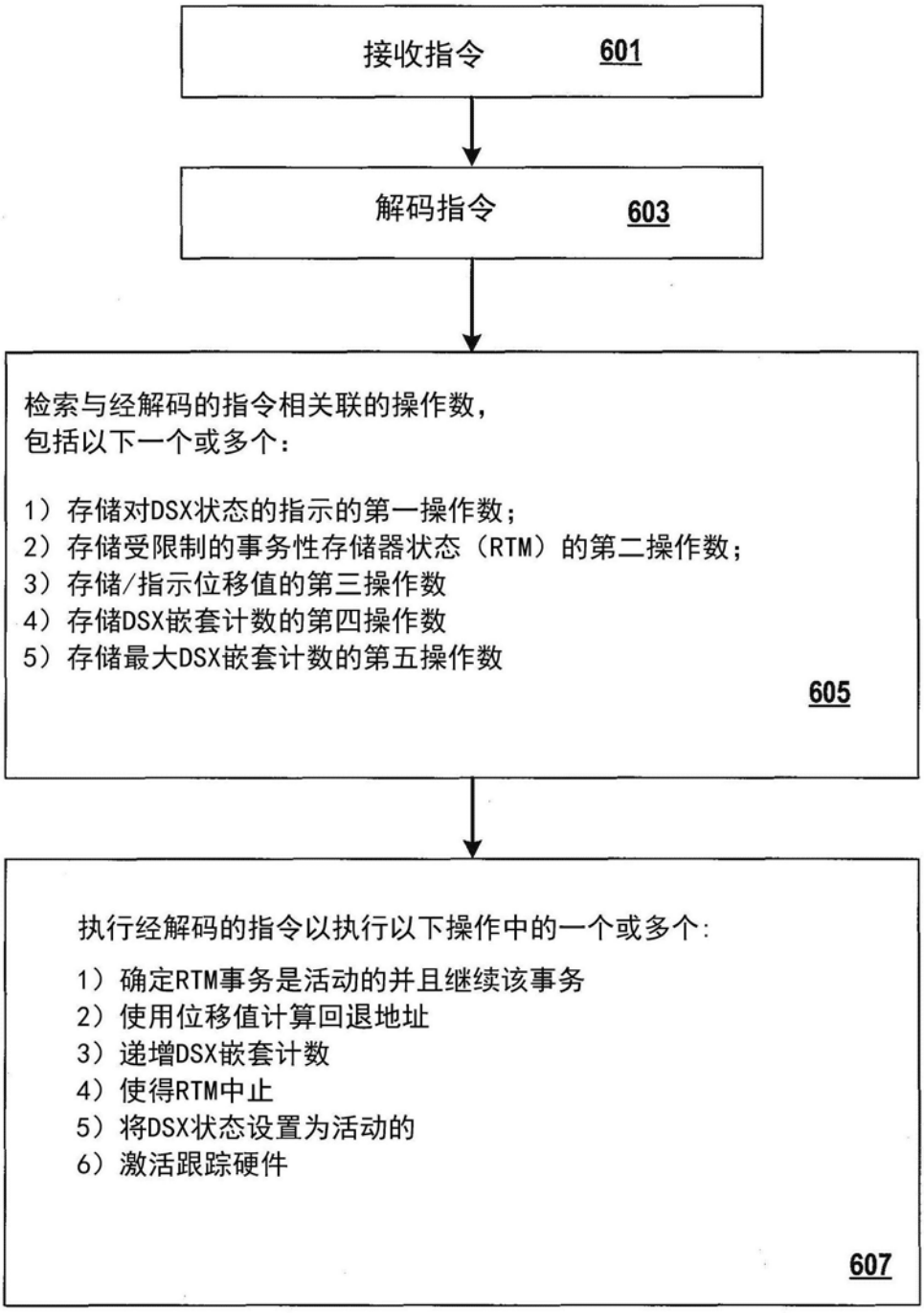

接收指令 601
解码指令 603
检索与经解码的指令相关联的操作数，
包括以下一个或多个：
1）存储对DSX状态的指示的第一操作数；
2）存储受限制的事务性存储器状态（RTM）的第二操作数；
3）存储/指示位移值的第三操作数
4）存储DSX嵌套计数的第四操作数
5）存储最大DSX嵌套计数的第五操作数
605
执行经解码的指令以执行以下操作中的一个或多个：
1）确定RTM事务是活动的并且继续该事务
2）使用位移值计算回退地址
3）递增DSX嵌套计数
4）使得RTM中止
5）将DSX状态设置为活动的
6）激活跟踪硬件
607

图6

操作码	显式操作数
701 YBEGIN	位移
703 YBEGIN	DSX状态寄存器，位移
705 YBEGIN	DSX嵌套计数寄存器，位移
707 YBEGIN	DSX状态寄存器，DSX嵌套计数寄存器，位移
709 YBEGIN	DSX状态寄存器，DSX嵌套计数寄存器， RTM状态寄存器，位移

图7

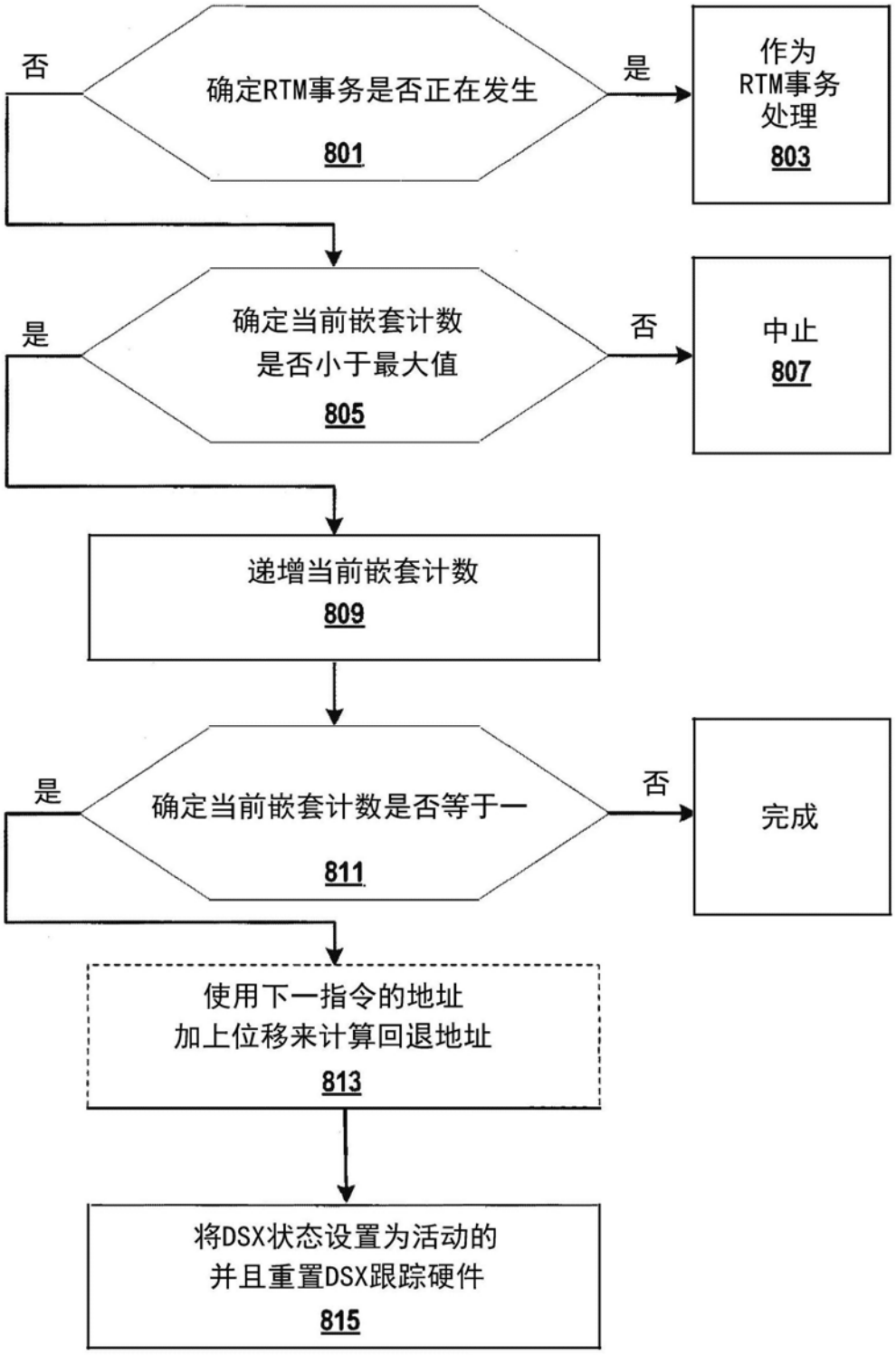

否
确定RTM事务是否正在发生
801
是
作为
RTM事务
处理
803
是
确定当前嵌套计数
是否小于最大值
805
否
中止
807
递增当前嵌套计数
809
是
确定当前嵌套计数是否等于一
811
否
完成
使用下一指令的地址
加上位移来计算回退地址
813
将DSX状态设置为活动的
并且重置DSX跟踪硬件
815

图8

```
YBEGIN DISPLACEMENT
IF (RTM_ACTIVE)
  GOTO XBEGIN
IF(DSX_NESTING_COUNT < MAX_DSX_NESTING_COUNT)
 DSX_NEST_COUNT++
IF(DSX_NEST_COUNT ==1)
 FALLBACK_IP = CALC_AND_FAULT_CHECK_IP (RIP, DISPLACEMENT)
 DSX_ACTIVE = 1
ENTER_DSX_STATE(REGISTER, MEMORY TRACKING)
FI
ELSE
YABORT
FI
```

图9

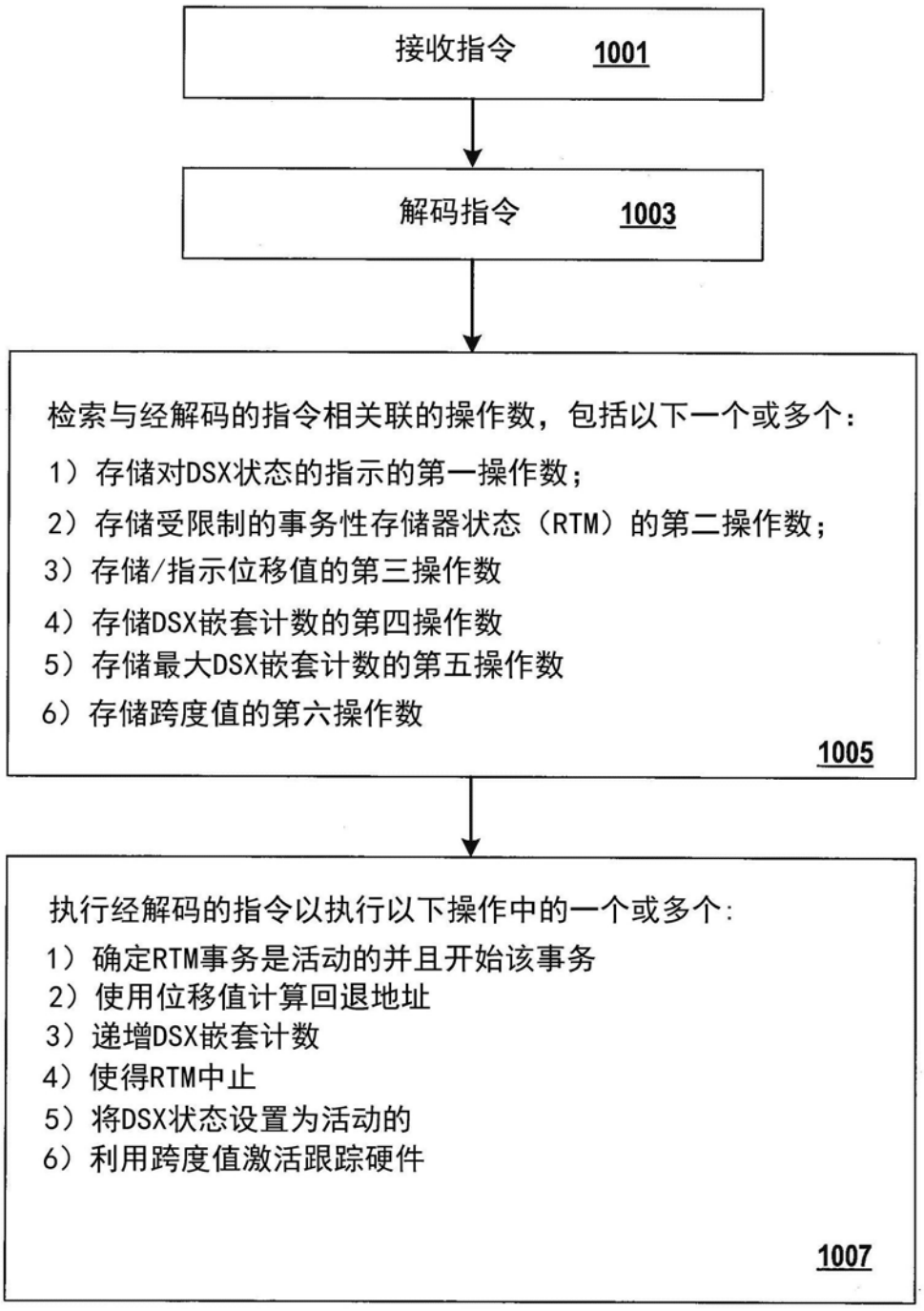

接收指令 1001
解码指令 1003
检索与经解码的指令相关联的操作数，包括以下一个或多个：
1）存储对DSX状态的指示的第一操作数；
2）存储受限制的事务性存储器状态（RTM）的第二操作数；
3）存储/指示位移值的第三操作数
4）存储DSX嵌套计数的第四操作数
5）存储最大DSX嵌套计数的第五操作数
6）存储跨度值的第六操作数
1005
执行经解码的指令以执行以下操作中的一个或多个：
1）确定RTM事务是活动的并且开始该事务
2）使用位移值计算回退地址
3）递增DSX嵌套计数
4）使得RTM中止
5）将DSX状态设置为活动的
6）利用跨度值激活跟踪硬件
1007

图10

操作码	显式操作数
1101 YBEGIN WITH STRIDE	位移，跨度
1103 YBEGIN WITH STRIDE	DSX状态寄存器，位移，跨度
1105 YBEGIN WITH STRIDE	DSX嵌套计数寄存器，位移，跨度
1107 YBEGIN WITH STRIDE	DSX状态寄存器，DSX嵌套计数寄存器，位移，跨度
1109 YBEGIN WITH STRIDE	DSX状态寄存器，DSX嵌套计数寄存器， RTM状态寄存器，位移，跨度

图11

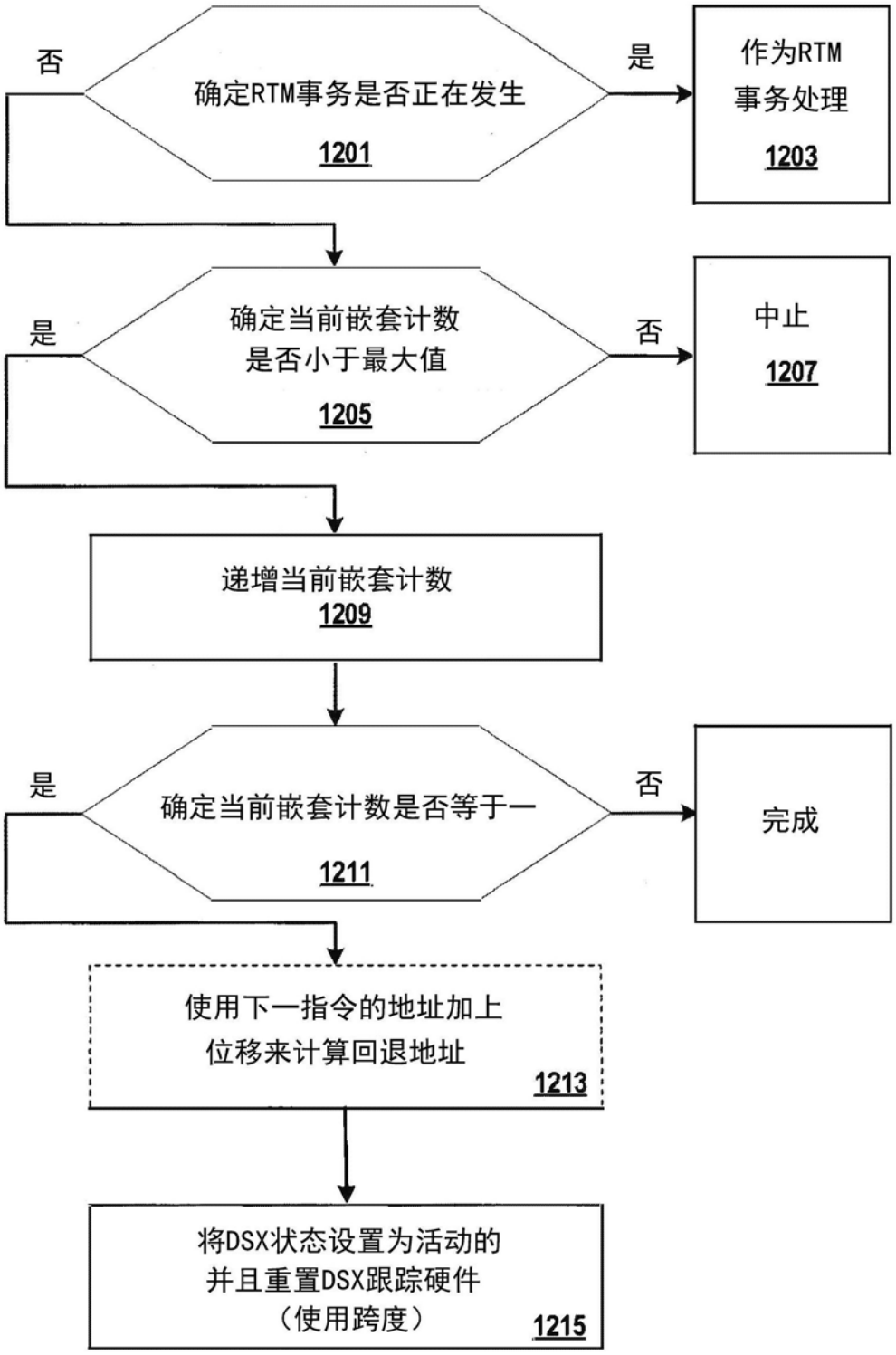

否
确定RTM事务是否正在发生
1201
是
作为RTM
事务处理
1203
是
确定当前嵌套计数
是否小于最大值
1205
否
中止
1207
递增当前嵌套计数
1209
是
确定当前嵌套计数是否等于一
1211
否
完成
使用下一指令的地址加上
位移来计算回退地址
1213
将DSX状态设置为活动的
并且重置DSX跟踪硬件
（使用跨度）
1215

图12

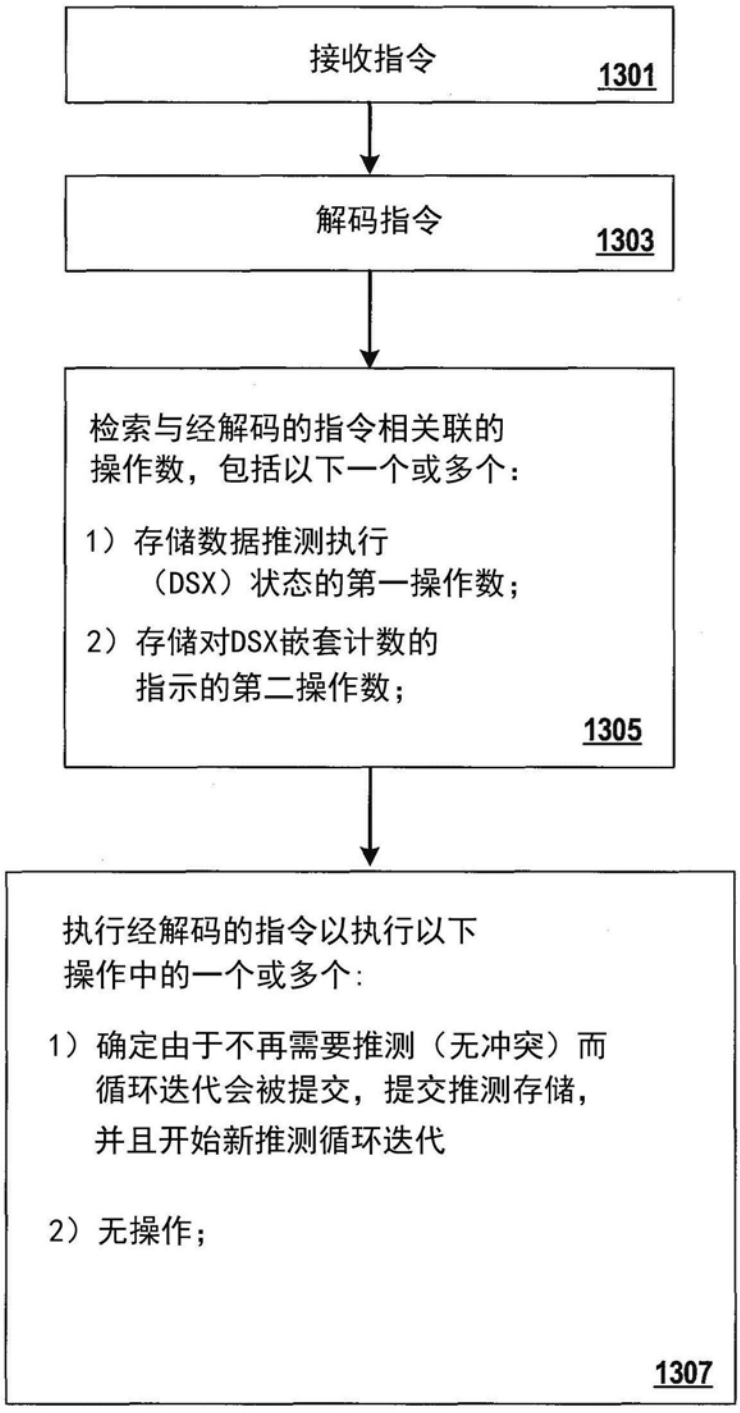

接收指令
1301
解码指令
1303
检索与经解码的指令相关联的
操作数，包括以下一个或多个：
1）存储数据推测执行
（DSX）状态的第一操作数；
2）存储对DSX嵌套计数的
指示的第二操作数；
1305
执行经解码的指令以执行以下
操作中的一个或多个：
1）确定由于不再需要推测（无冲突）而
循环迭代会被提交，提交推测存储，
并且开始新推测循环迭代
2）无操作；
1307

图13

操作码		显式操作数
1401	YCONTINUE	
1403	YCONTINUE	DSX状态寄存器
1405	YCONTINUE	DSX嵌套计数寄存器
1407	YCONTINUE	DSX状态寄存器，DSX嵌套计数寄存器

图14

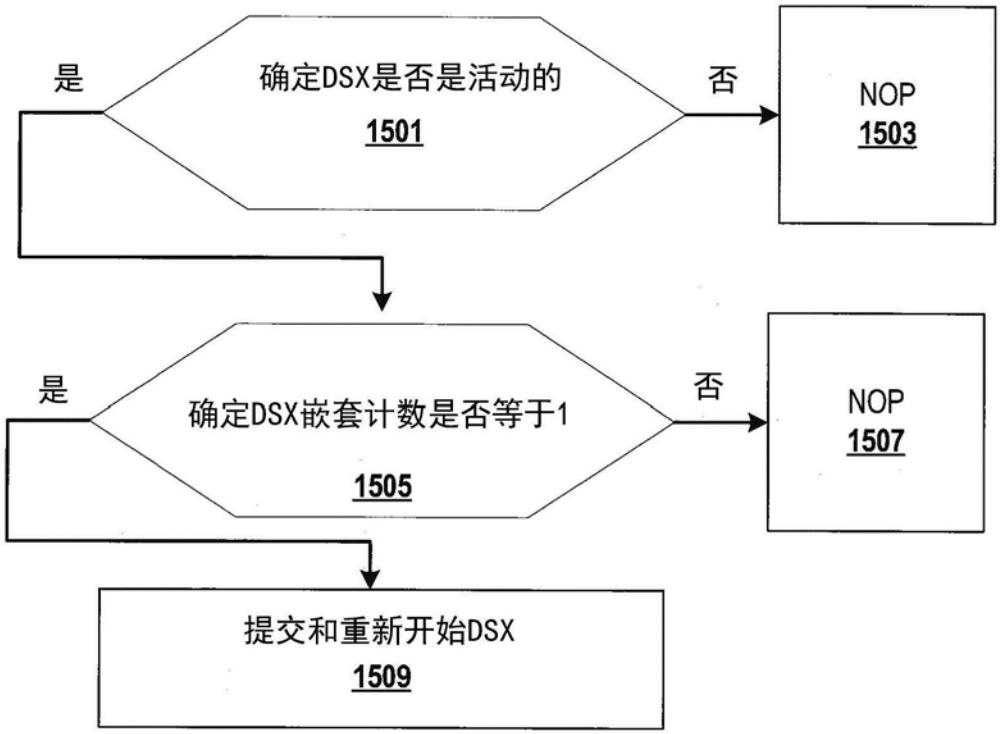


图15

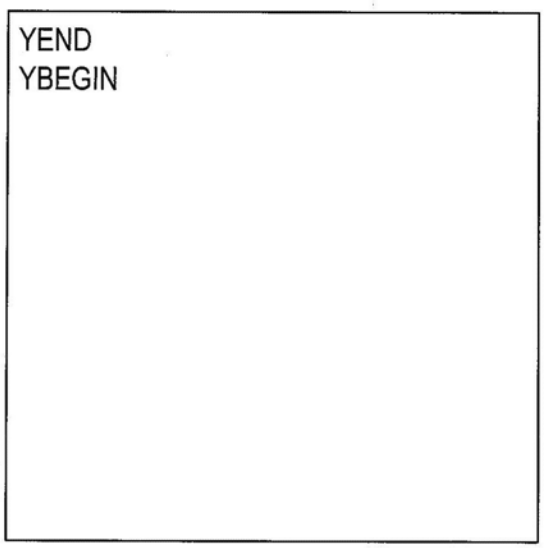


图16

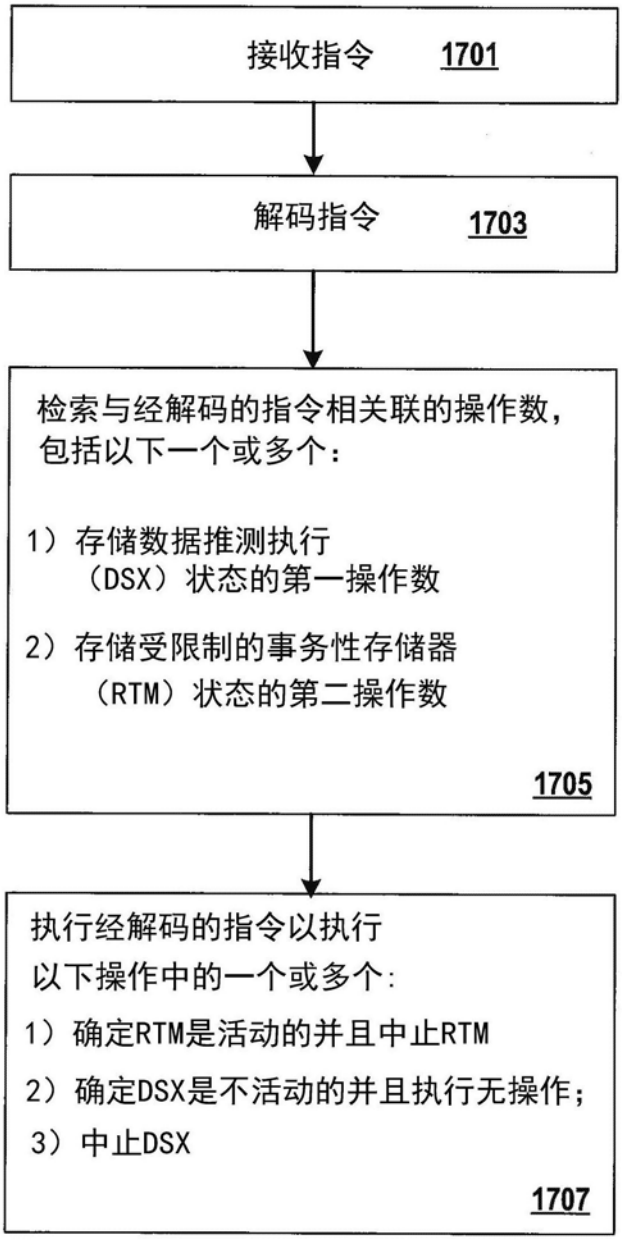

接收指令 1701
解码指令 1703
检索与经解码的指令相关联的操作数，
包括以下一个或多个：
1）存储数据推测执行
（DSX）状态的第一操作数
2）存储受限制的事务性存储器
（RTM）状态的第二操作数
1705
执行经解码的指令以执行
以下操作中的一个或多个：
1）确定RTM是活动的并且中止RTM
2）确定DSX是不活动的并且执行无操作；
3）中止DSX
1707

图17

操作码	显式操作数
1801 YABORT	
1803 YABORT	DSX状态寄存器
1805 YABORT	DSX状态寄存器，RTM状态寄存器

图18

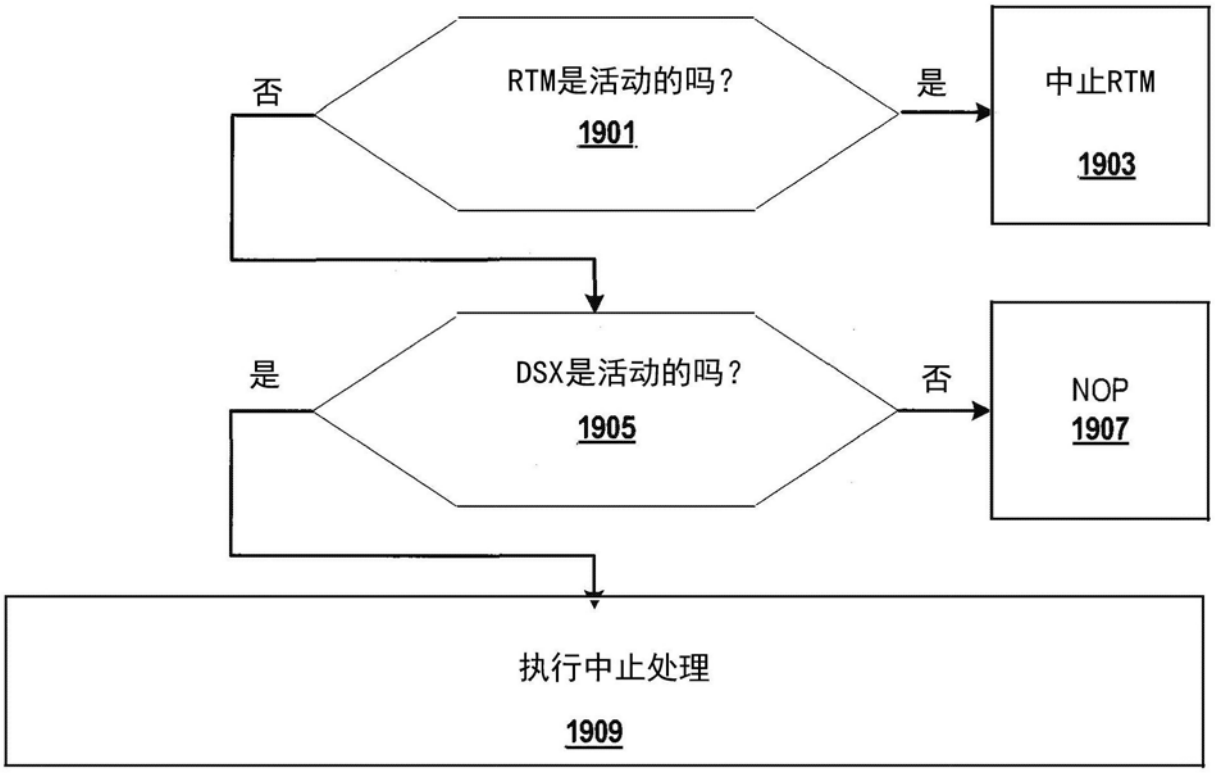


图19

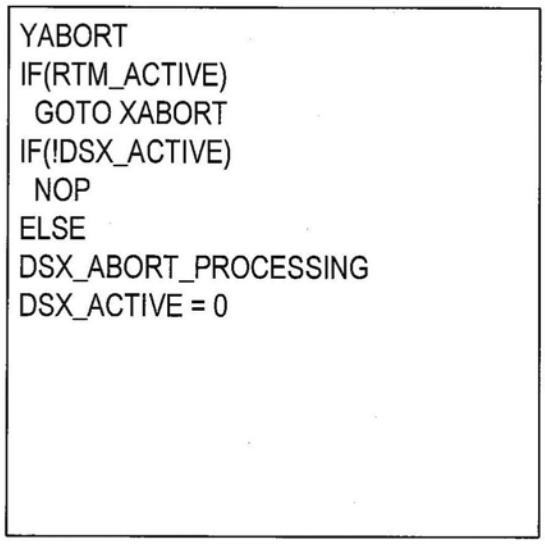

YABORT
IF(RTM_ACTIVE)
GOTO XABORT
IF(!DSX_ACTIVE)
NOP
ELSE
DSX_ABORT_PROCESSING
DSX_ACTIVE = 0

图20

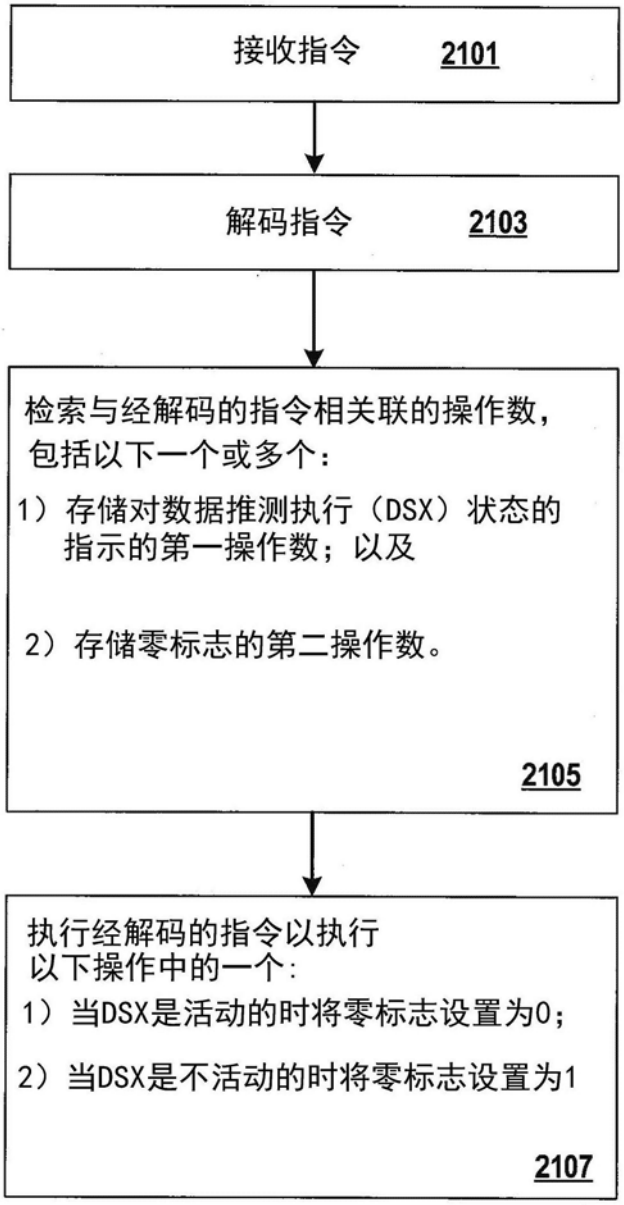

接收指令 2101
解码指令 2103
检索与经解码的指令相关联的操作数，
包括以下一个或多个：
1）存储对数据推测执行（DSX）状态的
指示的第一操作数；以及
2）存储零标志的第二操作数。
2105
执行经解码的指令以执行
以下操作中的一个：
1）当DSX是活动的时将零标志设置为0；
2）当DSX是不活动的时将零标志设置为1
2107

图21

操作码	显式操作数
2201 YTEST	
2203 YTEST	DSX状态寄存器
2205 YTEST	标志寄存器
2207 YTEST	DSX状态寄存器，标志寄存器

图22

```
YTEST
If (DSX_ACTIVE)
  then ZF=0
ELSE
  ZF=1
```

图23


接收指令
2401
解码指令
2403
检索与经解码的指令相关联的操作数，
包括以下一个或多个：
1）存储数据推测执行（DSX）嵌套
计数的第一操作数；
2）存储对DSX状态的指示的第二操作数；
3）存储受限制的事务性存储器（RTM）
状态的第三操作数
2405
执行经解码的指令以执行以下
操作中的一个或多个：
1）使得与DSX相关联的推测写入终结；
2）无操作；
3）中止；
4）结束RTM事务；
5）递减DSX嵌套计数
2407

图24

操作码	显式操作数
2501 YEND	
2503 YEND	DSX状态寄存器
2505 YEND	DSX嵌套计数寄存器
2507 YEND	DSX状态寄存器，DSX嵌套计数寄存器
2509 YEND	DSX状态寄存器，DSX嵌套计数寄存器，RTM状态寄存器

图25

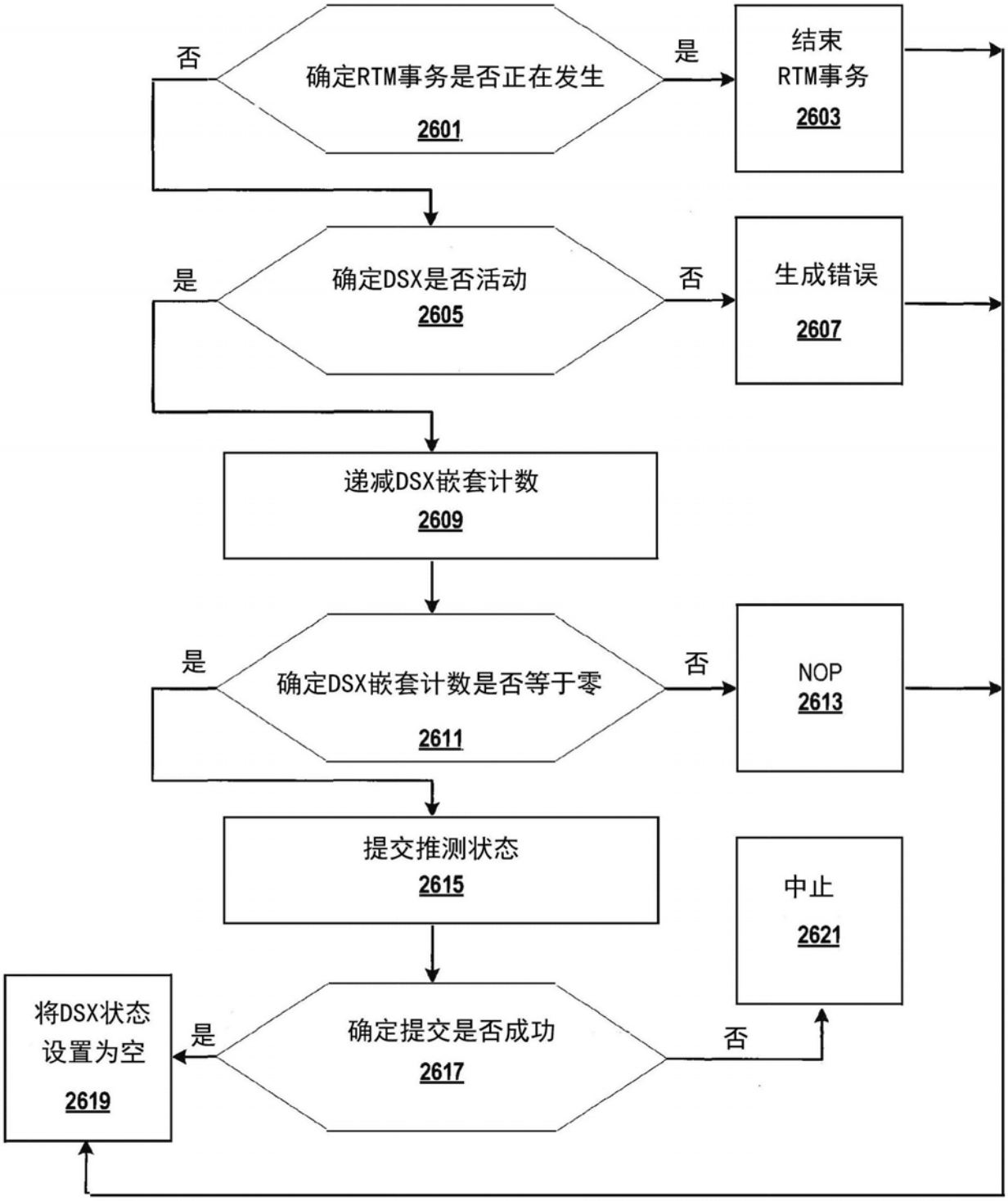

否
确定RTM事务是否正在发生
2601
是
结束
RTM事务
2603
是
确定DSX是否活动
2605
否
生成错误
2607
递减DSX嵌套计数
2609
是
确定DSX嵌套计数是否等于零
2611
否
NOP
2613
提交推测状态
2615
中止
2621
将DSX状态
设置为空
2619
是
确定提交是否成功
2617
否

图26

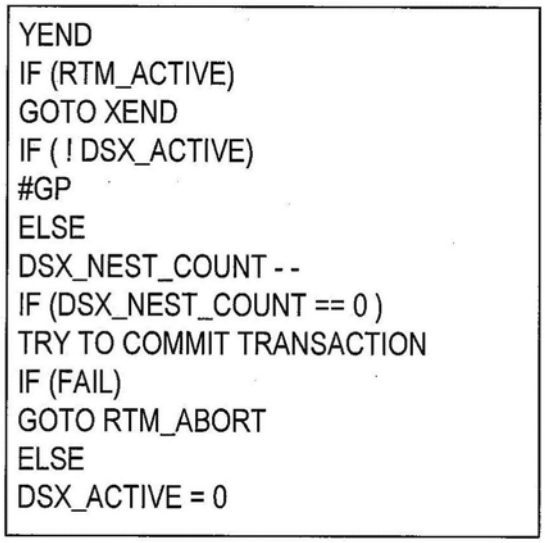

YEND
IF (RTM_ACTIVE)
GOTO XEND
IF (! DSX_ACTIVE)
#GP
ELSE
DSX_NEST_COUNT - -
IF (DSX_NEST_COUNT == 0)
TRY TO COMMIT TRANSACTION
IF (FAIL)
GOTO RTM_ABORT
ELSE
DSX_ACTIVE = 0

图27

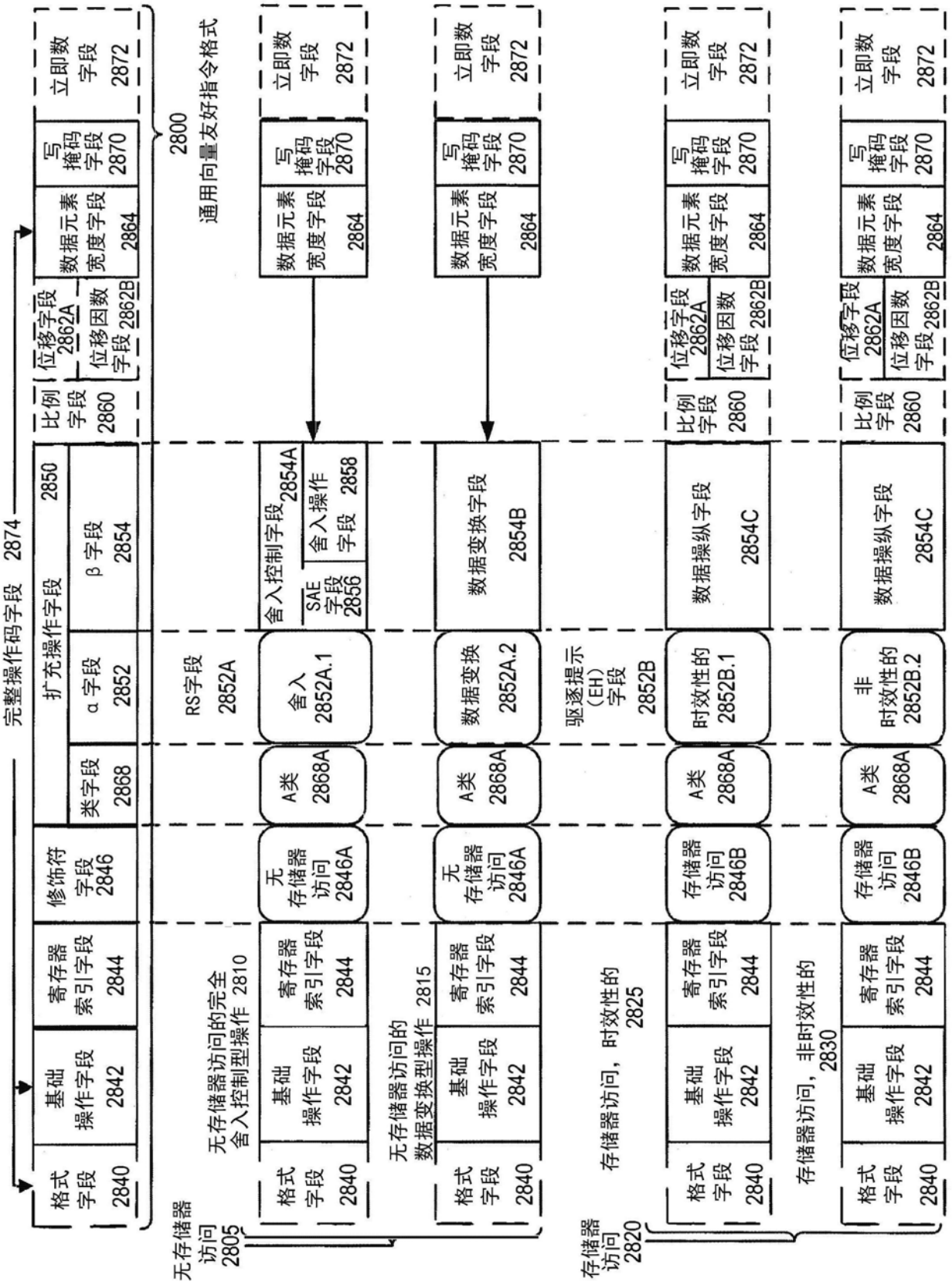

完整操作码字段 2874
格式字段 2840
基础操作字段 2842
寄存器索引字段 2844
修饰符字段 2846
扩充操作字段 2850
类字段 2868
α字段 2852
β字段 2854
比例字段 2860
位移字段 2862A
位移因数字段 2862B
数据元素宽度字段 2864
写掩码字段 2870
立即数字段 2872
2800
通用向量友好指令格式
无存储器访问 2805
无存储器访问的完全舍入控制型操作 2810
RS字段 2852A
A类 2868A
无存储器访问 2846A
舍入 2852A.1
舍入控制字段 2854A
SAE字段 2856
舍入操作字段 2858
无存储器访问的数据变换型操作 2815
数据变换 2852A.2
数据变换字段 2854B
存储器访问 2820
存储器访问，时效性的 2825
驱逐提示（EH）字段 2852B
存储器访问 2846B
时效性的 2852B.1
数据操纵字段 2854C
存储器访问，非时效性的 2830
非时效性的 2852B.2

图28A

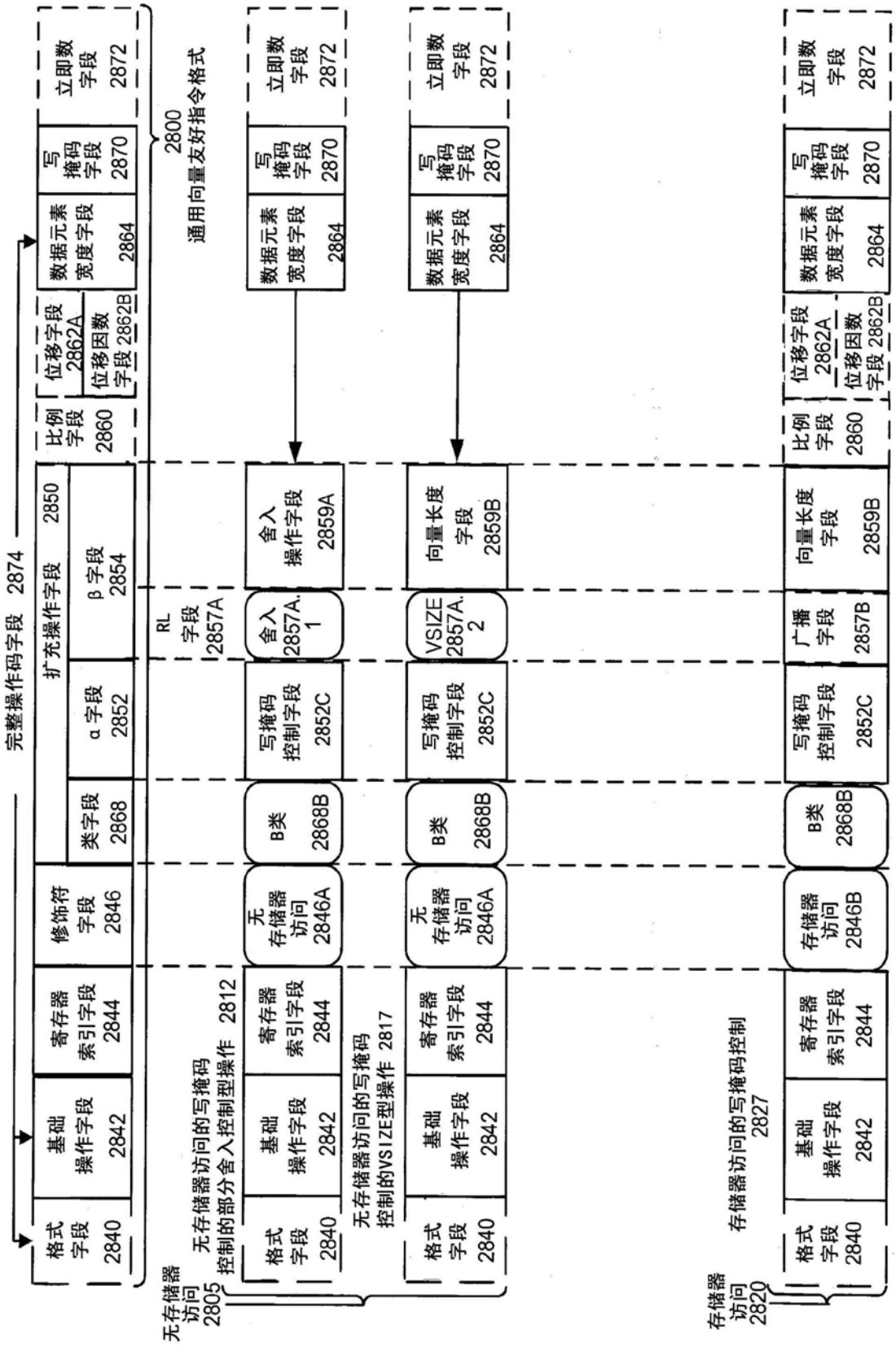

2800
通用向量友好指令格式
完整操作码字段 2874
格式字段 2840
基础操作字段 2842
寄存器索引字段 2844
修饰符字段 2846
类字段 2868
扩充操作字段 2850
α 字段 2852
β 字段 2854
比例字段 2860
位移字段 2862A
位移因数字段 2862B
数据元素宽度字段 2864
写掩码字段 2870
立即数字段 2872
RL 字段 2857A
无存储器访问 2805
无存储器访问的写掩码控制的部分舍入控制型操作 2812
无存储器访问 2846A
B类 2868B
写掩码控制字段 2852C
舍入操作字段 2859A
舍入 2857A.1
无存储器访问的写掩码控制的VSIZE型操作 2817
VSIZE 2857A.2
向量长度字段 2859B
存储器访问 2820
存储器访问的写掩码控制 2827
存储器访问 2846B
广播字段 2857B

图28B

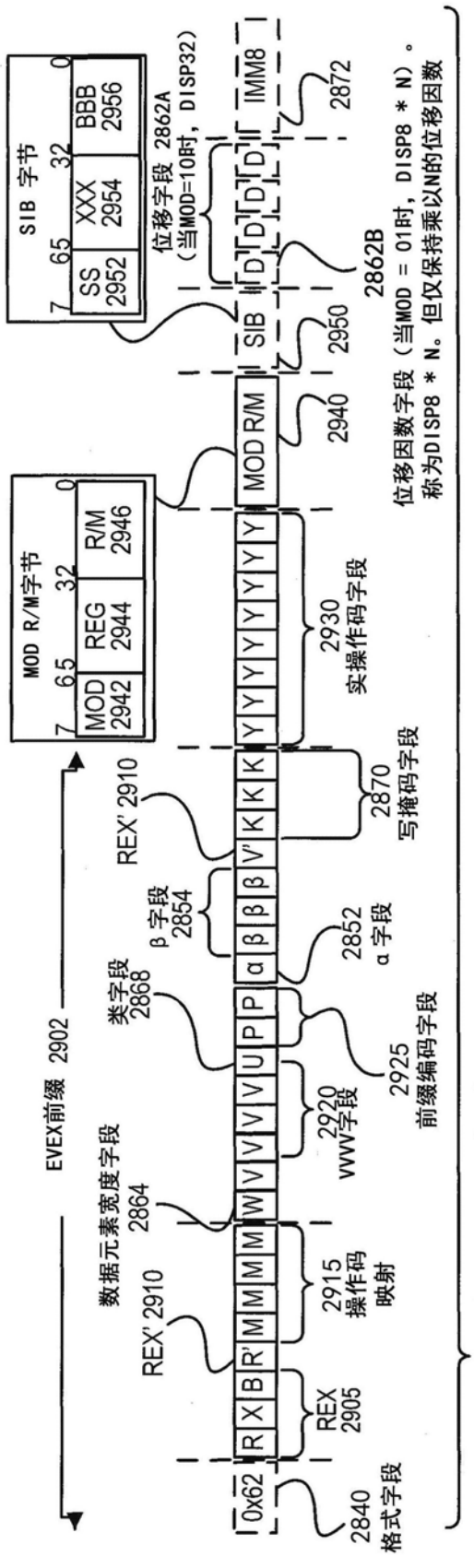

SIB 字节
SS 2952
XXX 2954
BBB 2956
位移字段 2862A
（当MOD=10时，DISP32）
MOD R/M字节
MOD 2942
REG 2944
R/M 2946
SIB
MOD R/M
2872
2940
2950
2862B
位移因数字段（当MOD = 01时，DISP8 * N）。
称为DISP8 * N。但仅保持乘以N的位移因数
EVEX前缀 2902
REX' 2910
类字段 2868
β 字段 2854
α 字段 2852
写掩码字段 2870
实操作码字段 2930
数据元素宽度字段 2864
REX' 2910
vvv字段 2920
前缀编码字段 2925
操作码映射 2915
REX 2905
格式字段 2840
0x62
2900 专用向量友好指令格式

图29A

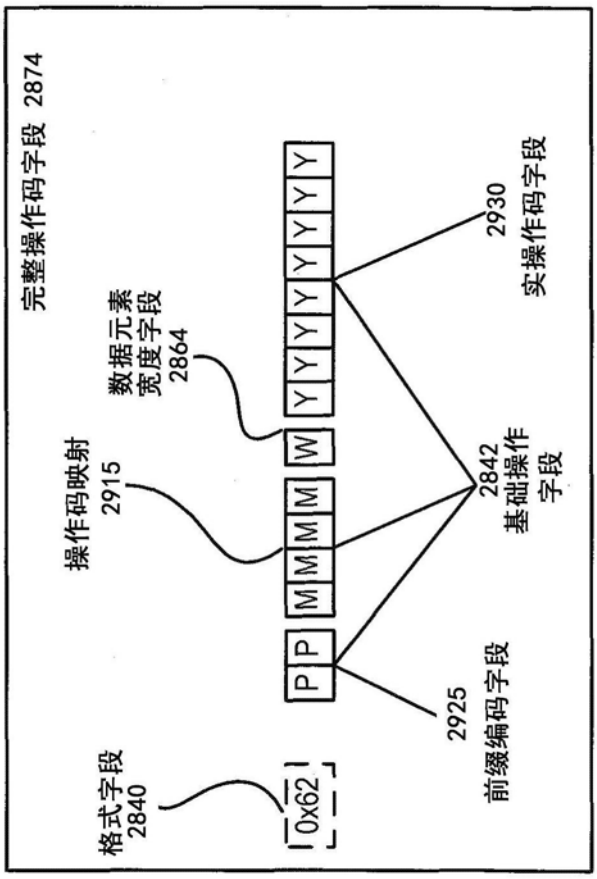

完整操作码字段 2874
格式字段
2840
0x62
操作码映射
2915
数据元素
宽度字段
2864
P P
M M M
W
Y Y Y Y
Y Y Y Y
2925
前缀编码字段
2842
基础操作
字段
2930
实操作码字段

图29B

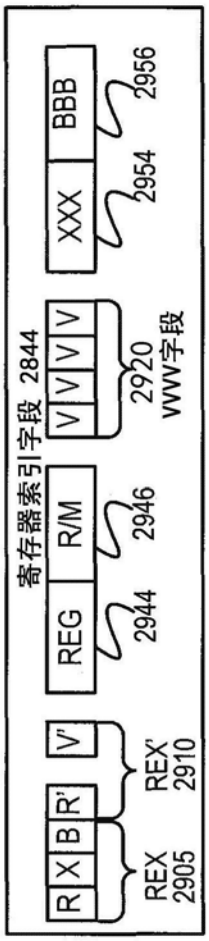

寄存器索引字段 2844
R
X
B
R'
REX
2905
V'
REX'
2910
REG
R/M
2944
2946
V
V
V
V
2920
VVV字段
XXX
BBB
2954
2956

图29C

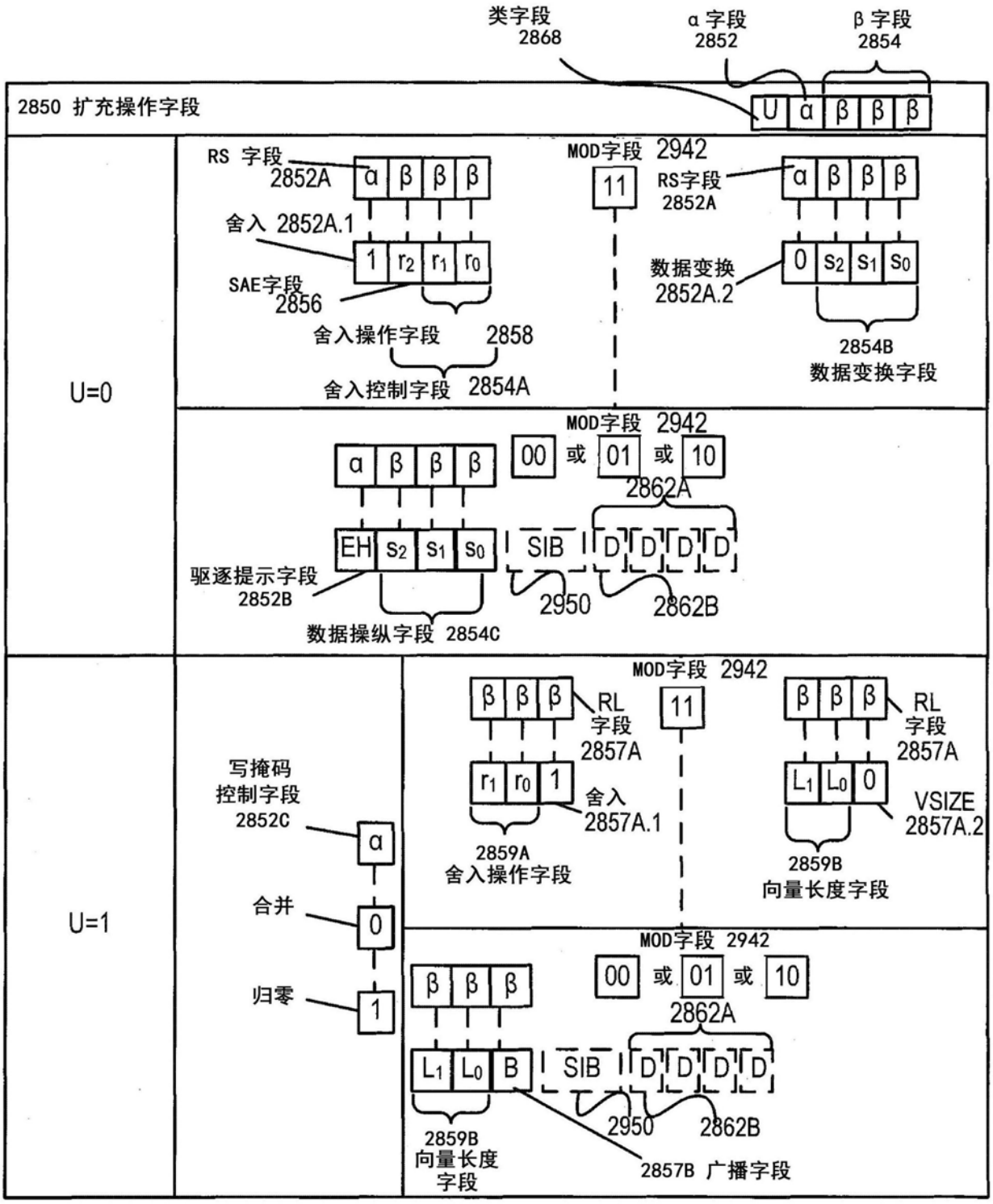

类字段 2868
α 字段 2852
β 字段 2854
2850 扩充操作字段
U α β β β
U=0
RS 字段 2852A
α β β β
舍入 2852A.1
1 r2 r1 r0
SAE字段 2856
舍入操作字段 2858
舍入控制字段 2854A
MOD字段 2942
11
RS字段 2852A
α β β β
数据变换 2852A.2
0 S2 S1 S0
2854B 数据变换字段
MOD字段 2942
α β β β
00 或 01 或 10
2862A
EH S2 S1 S0
SIB D D D D
驱逐提示字段 2852B
2950
2862B
数据操纵字段 2854C
U=1
写掩码控制字段 2852C
α
合并
0
归零
1
MOD字段 2942
β β β
RL 字段 2857A
11
r1 r0 1
舍入 2857A.1
2859A 舍入操作字段
β β β
RL 字段 2857A
L1 L0 0
VSIZE 2857A.2
2859B 向量长度字段
MOD字段 2942
β β β
00 或 01 或 10
2862A
L1 L0 B
SIB D D D D
2950
2862B
2859B 向量长度字段
2857B 广播字段

图29D

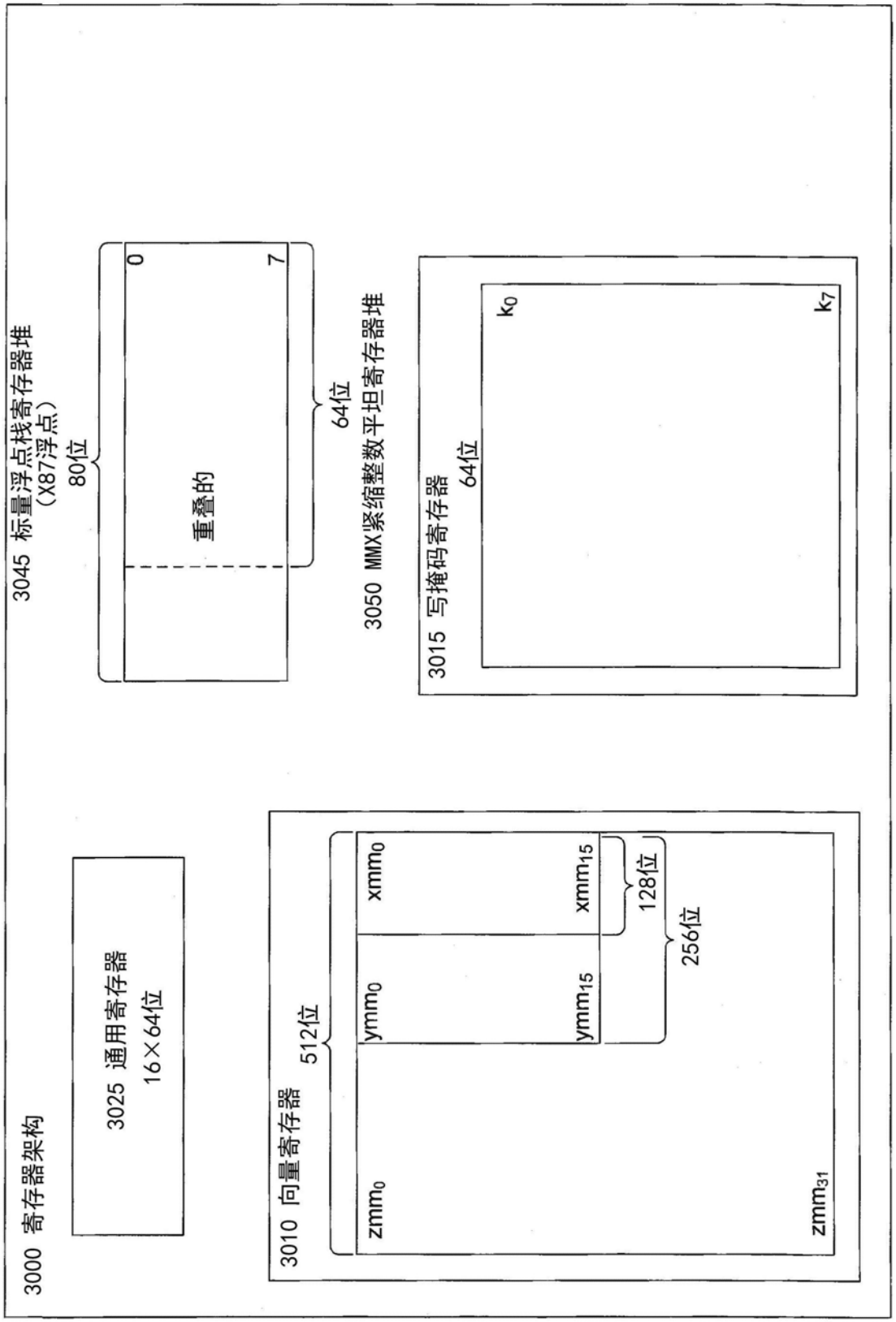

3000 寄存器架构
3025 通用寄存器
16×64位
3045 标量浮点栈寄存器堆
(X87浮点)
80位
0
7
重叠的
64位
3050 MMX紧缩整数平坦寄存器堆
3015 写掩码寄存器
64位
k0
k7
3010 向量寄存器
512位
zmm0
zmm31
ymm0
ymm15
xmm0
xmm15
128位
256位

图30

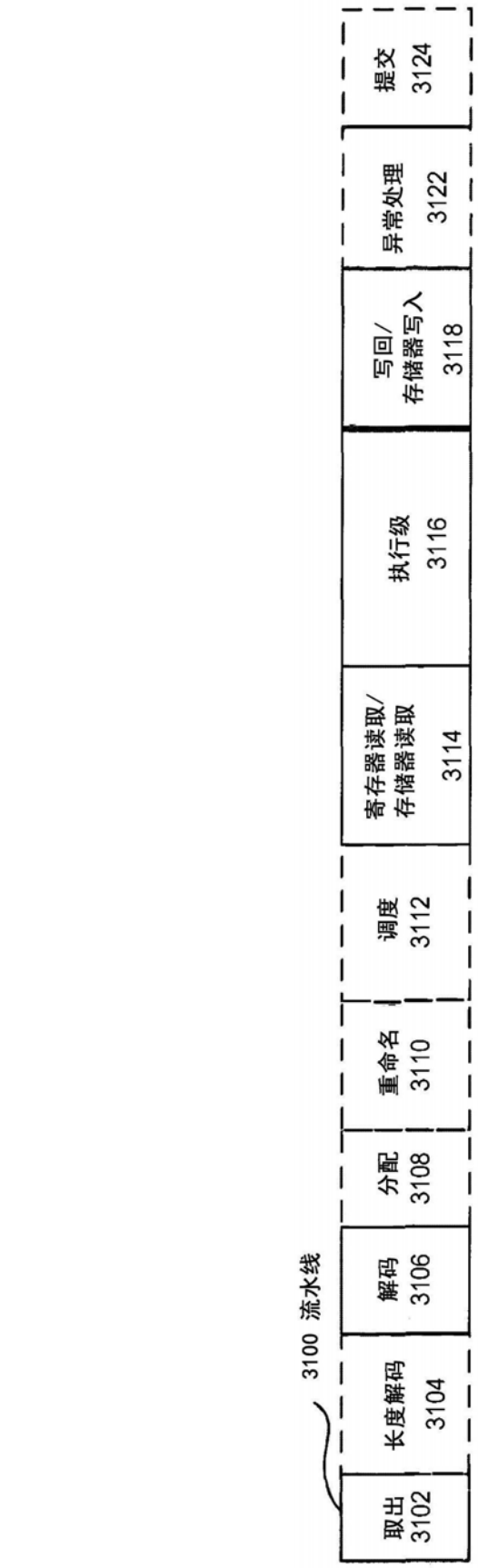

3100 流水线
取出 3102
长度解码 3104
解码 3106
分配 3108
重命名 3110
调度 3112
寄存器读取/存储器读取 3114
执行级 3116
写回/存储器写入 3118
异常处理 3122
提交 3124

图31A

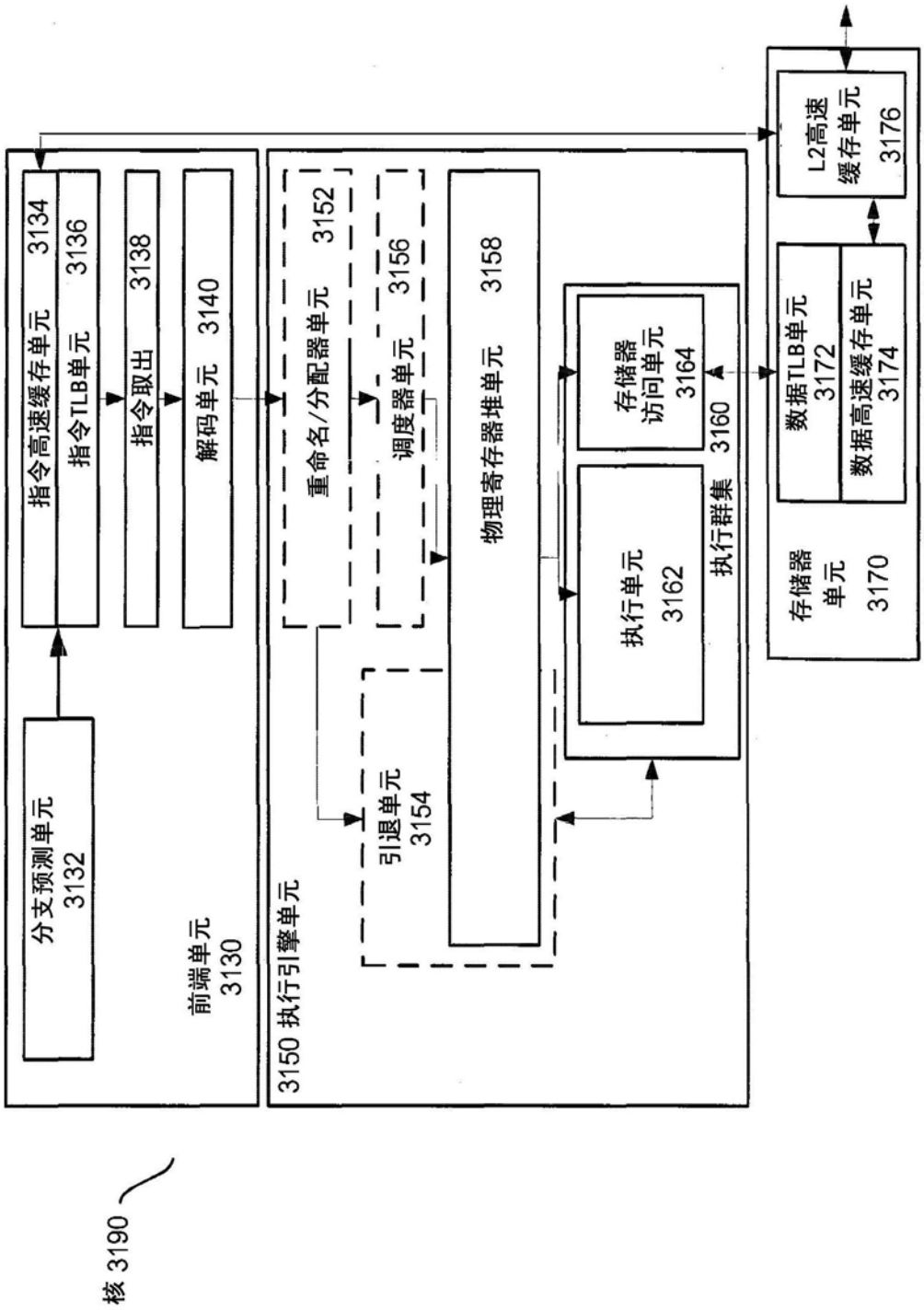

核 3190
前端单元 3130
分支预测单元 3132
指令高速缓存单元 3134
指令TLB单元 3136
指令取出 3138
解码单元 3140
3150 执行引擎单元
重命名/分配器单元 3152
引退单元 3154
调度器单元 3156
物理寄存器堆单元 3158
执行单元 3162
存储器访问单元 3164
执行群集 3160
存储器单元 3170
数据TLB单元 3172
数据高速缓存单元 3174
L2高速缓存单元 3176

图31B

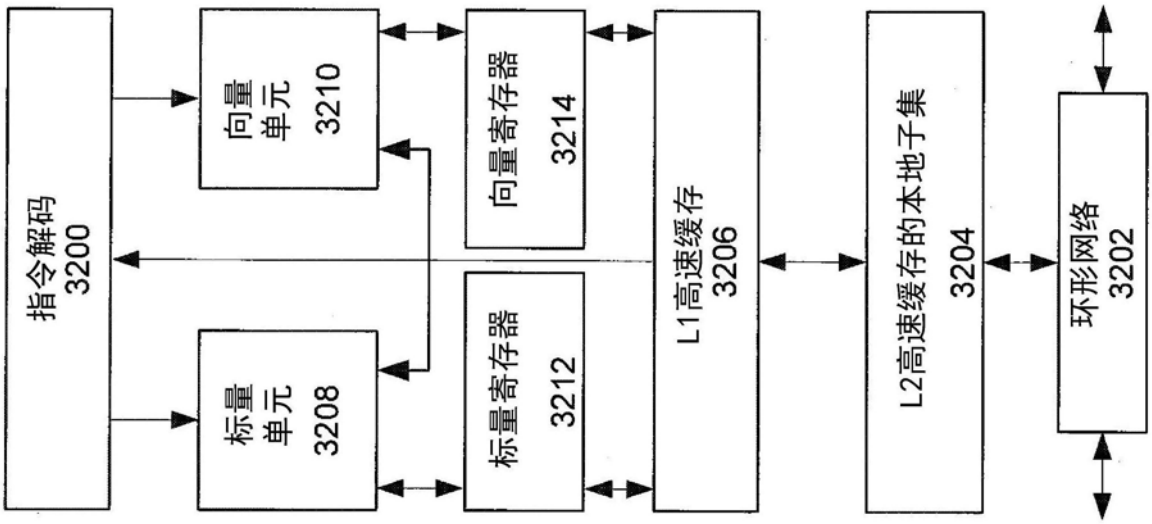

指令解码
3200
向量
单元
3210
标量
单元
3208
向量寄存器
3214
标量寄存器
3212
L1高速缓存
3206
L2高速缓存的本地子集
3204
环形网络
3202

图32A

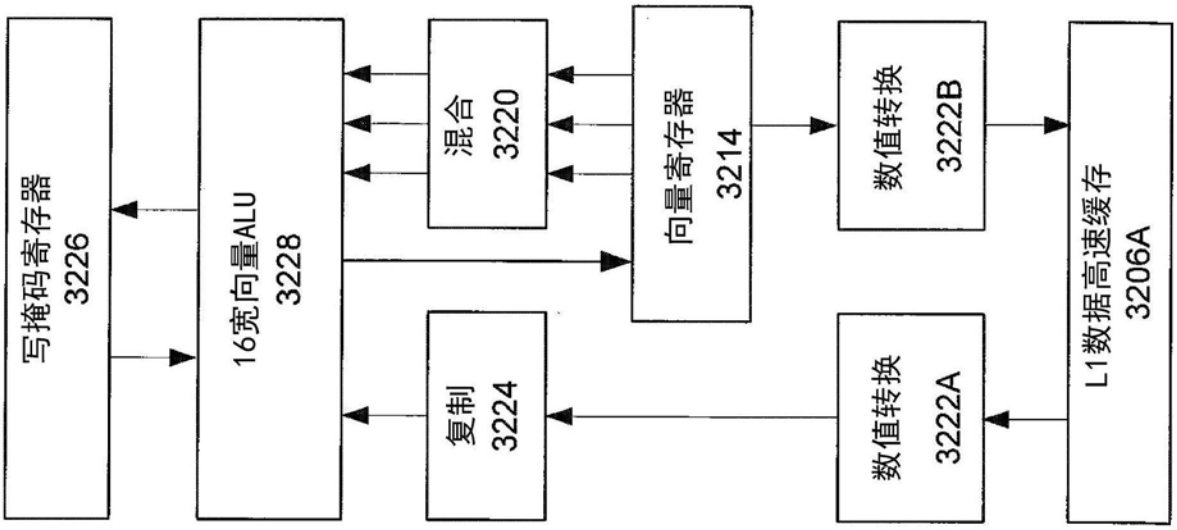

写掩码寄存器
3226
16宽向量ALU
3228
混合
3220
向量寄存器
3214
数值转换
3222B
L1数据高速缓存
3206A
复制
3224
数值转换
3222A

图32B

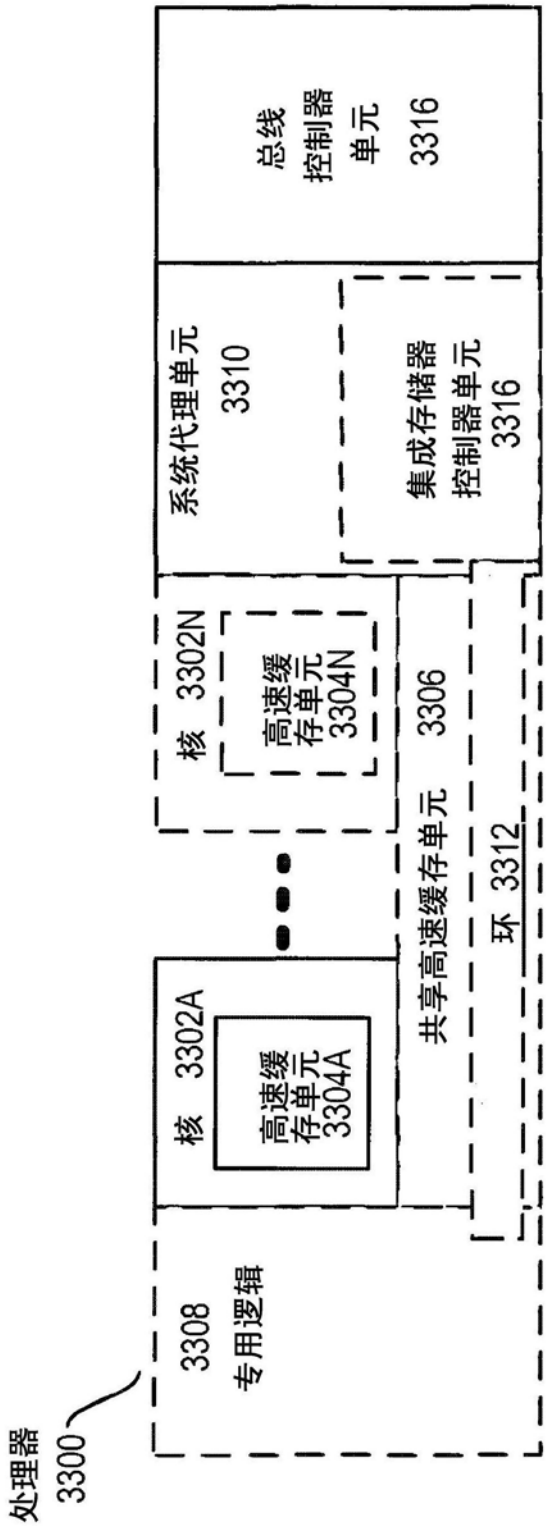

处理器
3300
3308
专用逻辑
核 3302A
高速缓存单元 3304A
核 3302N
高速缓存单元 3304N
共享高速缓存单元 3306
环 3312
系统代理单元 3310
集成存储器控制器单元 3316
总线控制器单元 3316

图33

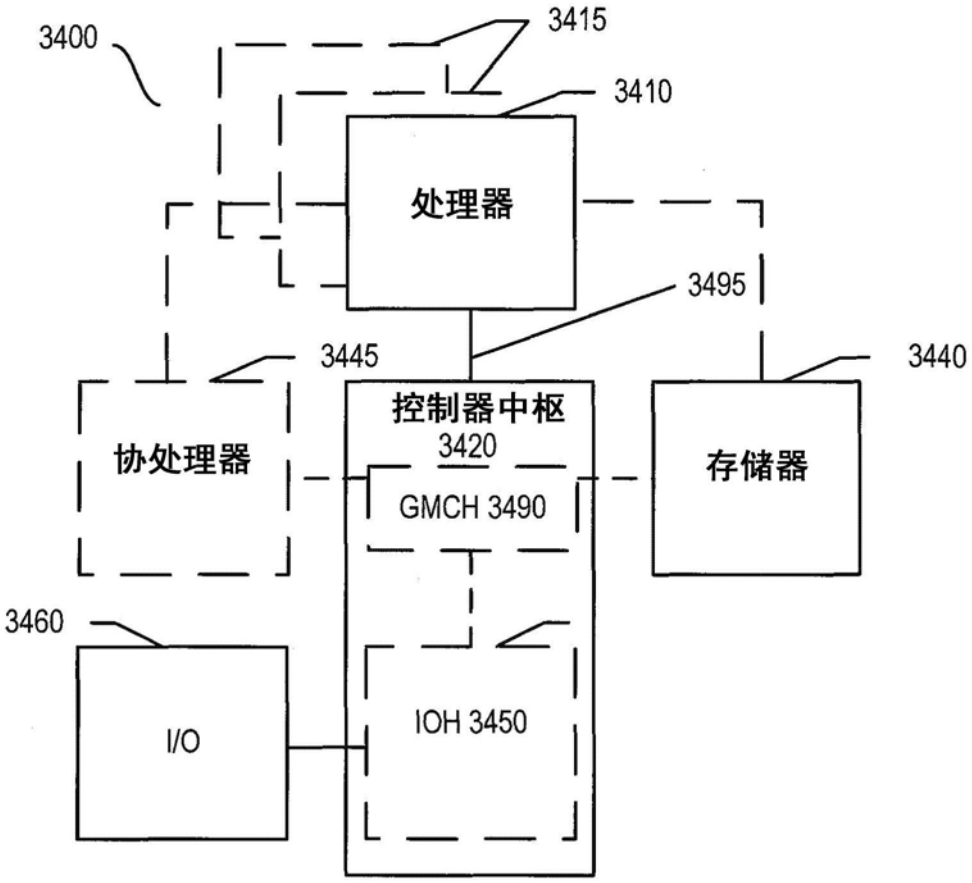

3400
3415
3410
处理器
3495
3445
3440
控制器中枢
3420
协处理器
GMCH 3490
存储器
3460
I/O
IOH 3450

图34

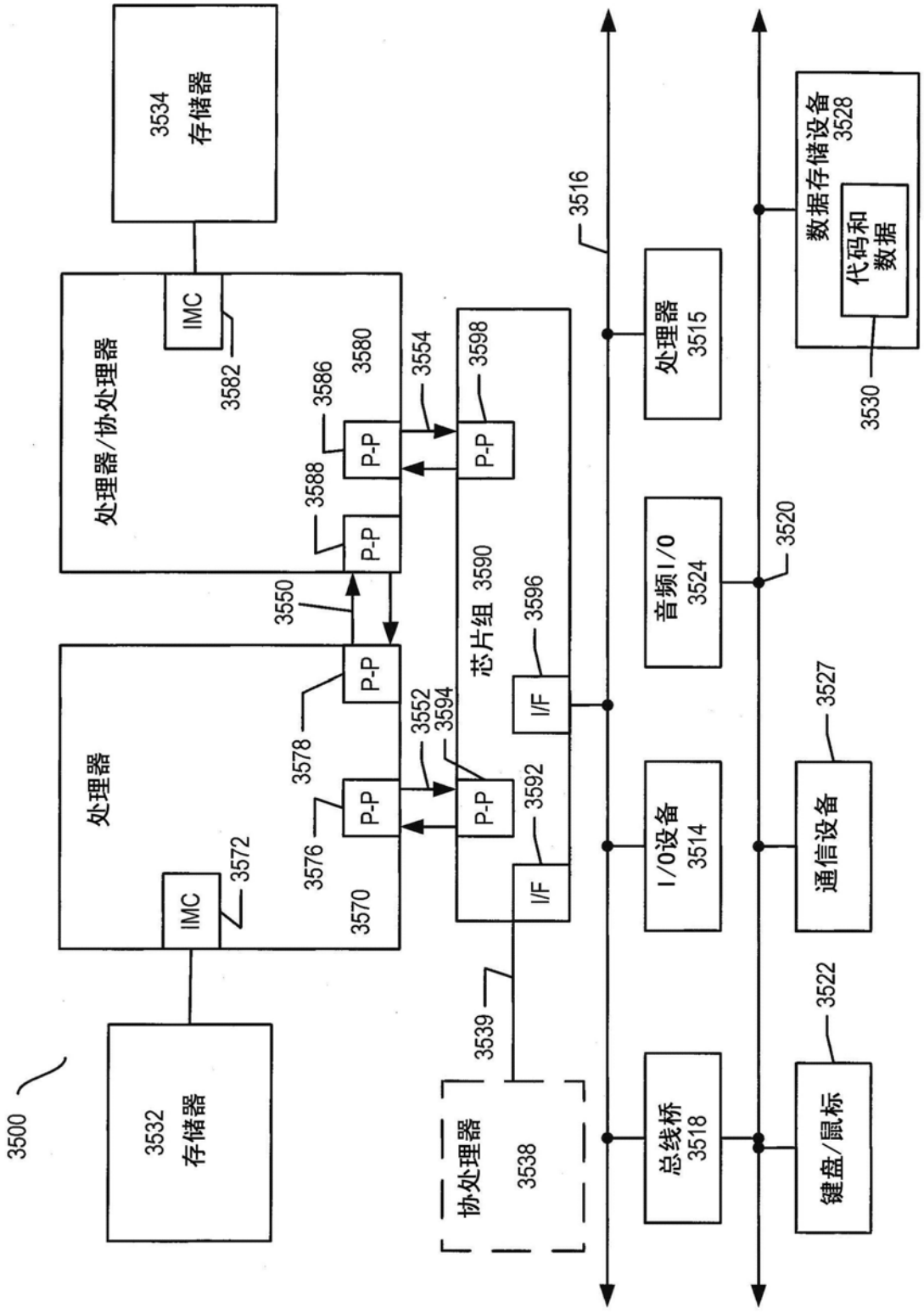

3500
3532 存储器
3534 存储器
处理器
处理器/协处理器
IMC
3572
3582
3570
3576
3578
3588
3586
3580
3554
3598
3550
3552
3594
3592
芯片组 3590
3596
I/F
3539
协处理器 3538
3516
处理器 3515
音频I/O 3524
I/O设备 3514
总线桥 3518
3520
3527
通信设备
3522
键盘/鼠标
数据存储设备 3528
代码和数据
3530

图35

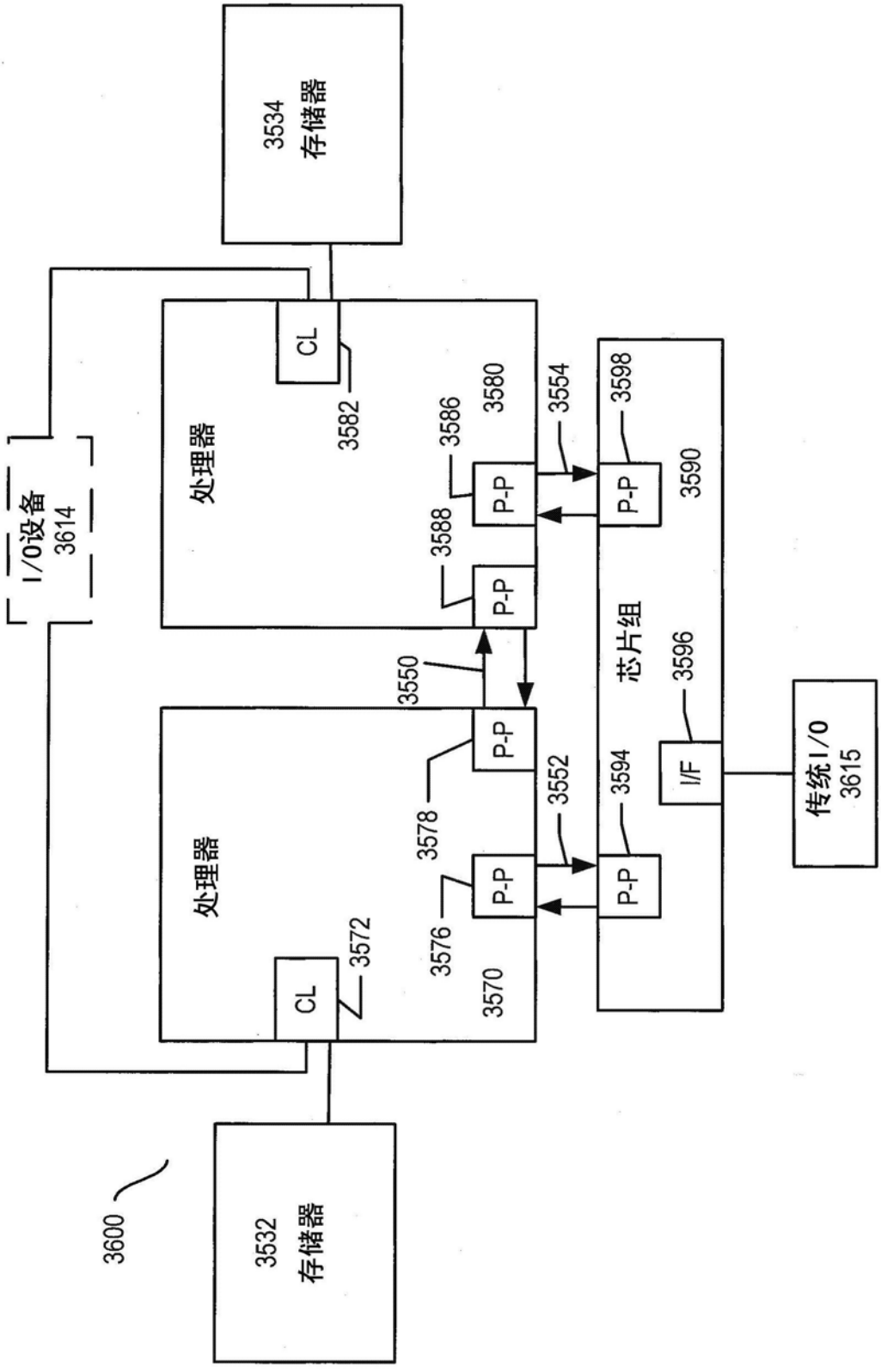

3534
存储器
CL
3582
处理器
3586
3580
3554
3598
P-P
P-P
3590
3588
P-P
I/O设备
3614
3550
芯片组
3596
P-P
3578
3552
3594
I/F
传统I/O
3615
处理器
P-P
P-P
3576
3572
CL
3570
3600
3532
存储器

图36

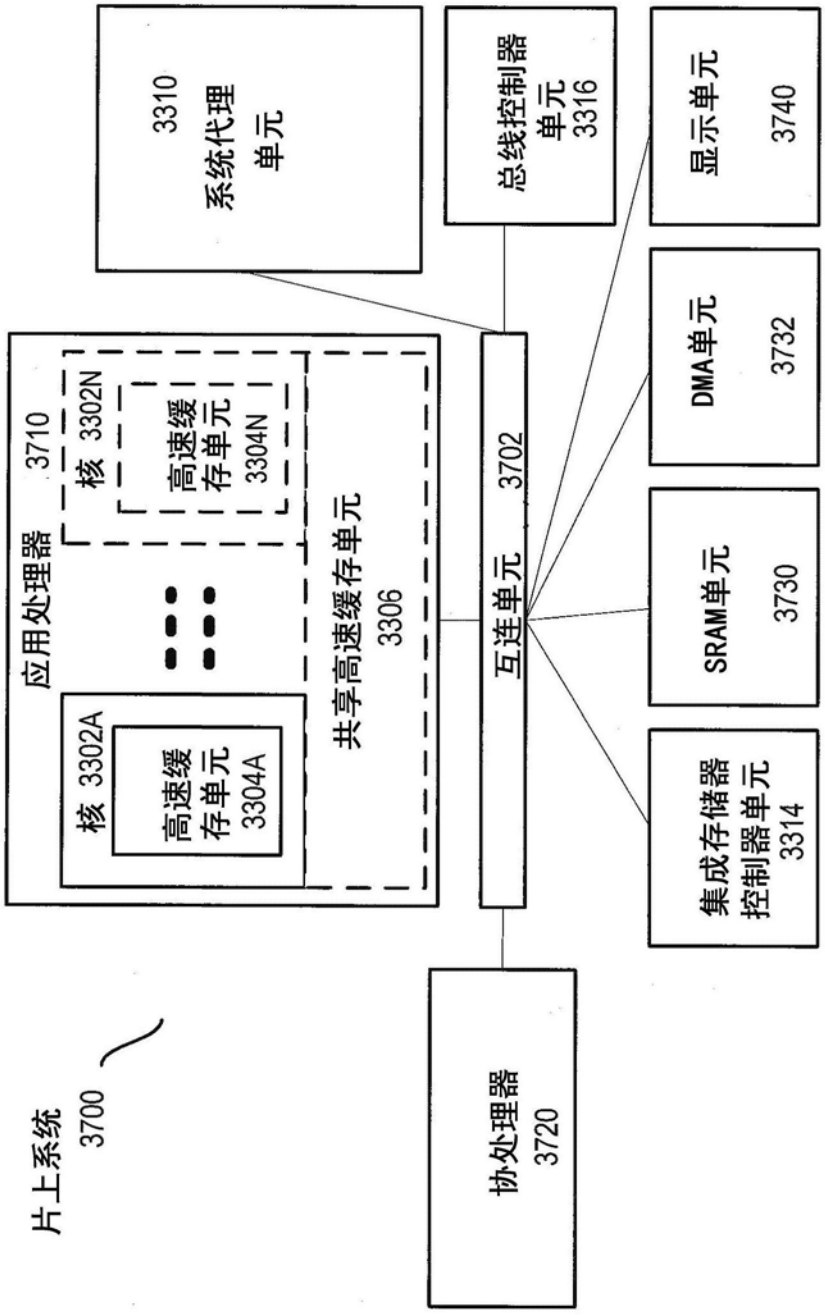

片上系统
3700
应用处理器
3710
核 3302A
高速缓存单元 3304A
核 3302N
高速缓存单元 3304N
共享高速缓存单元 3306
互连单元 3702
3310 系统代理单元
总线控制器单元 3316
显示单元 3740
DMA单元 3732
SRAM单元 3730
集成存储器控制器单元 3314
协处理器 3720

图37

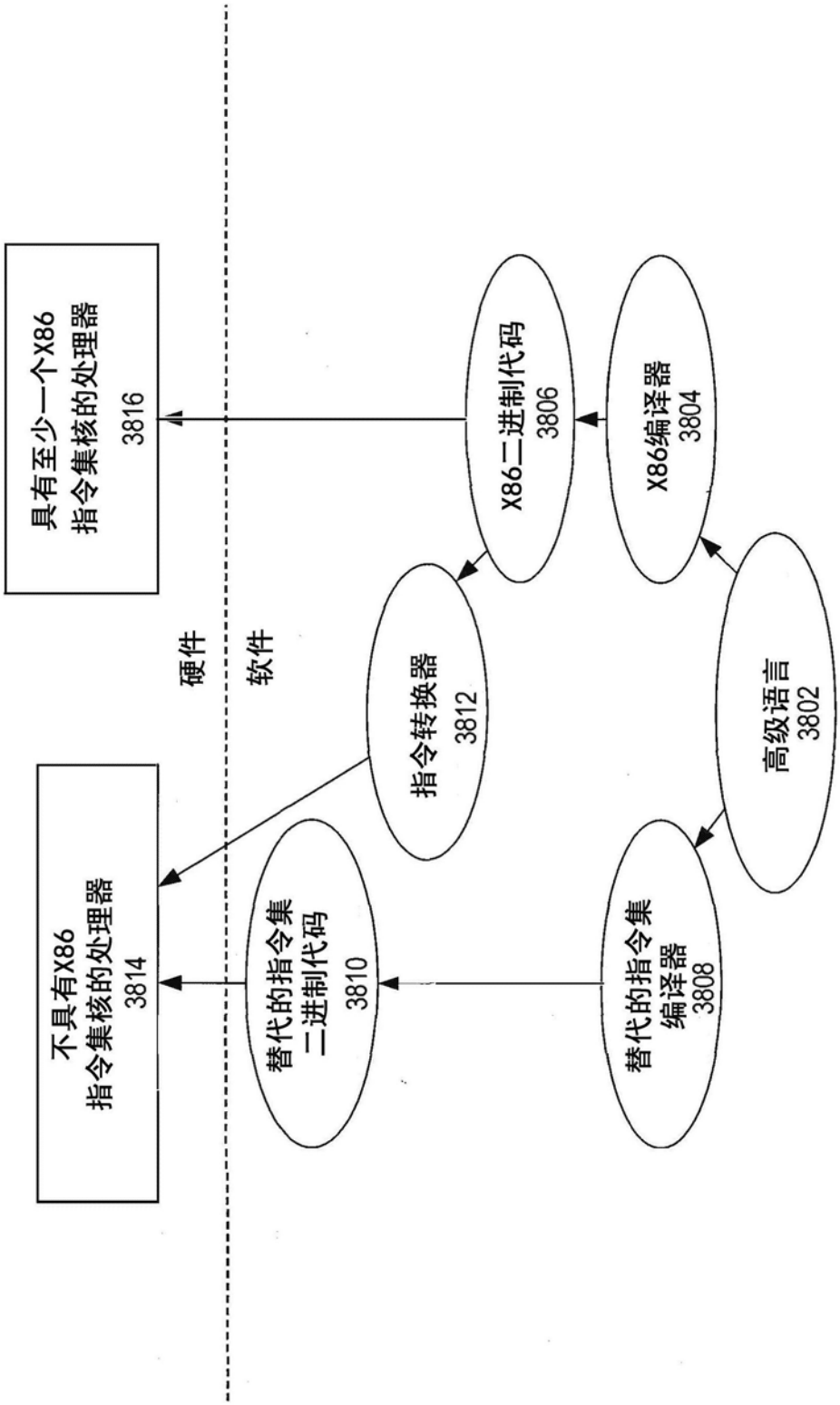

具有至少一个X86
指令集核的处理器
3816
硬件
软件
X86二进制代码
3806
X86编译器
3804
指令转换器
3812
高级语言
3802
不具有X86
指令集核的处理器
3814
替代的指令集
二进制代码
3810
替代的指令集
编译器
3808

图38