(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 104137059 B
(45)授权公告日 2018.10.09

(21)申请号 201180076436.3
(22)申请日 2011.12.23
(65)同一申请的已公布的文献号
申请公布号 CN 104137059 A
(43)申请公布日 2014.11.05
(85)PCT国际申请进入国家阶段日
2014.08.22
(86)PCT国际申请的申请数据
PCT/US2011/067276 2011.12.23
(87)PCT国际申请的公布数据
WO2013/095669 EN 2013.06.27
(73)专利权人 英特尔公司
地址 美国加利福尼亚州
(72)发明人 A·杰哈
(74)专利代理机构 上海专利商标事务所有限公司 31100
代理人 张东梅
(51)Int.Cl.
G06F 9/30(2006.01)
G06F 9/305(2006.01)
(56)对比文件
US 2002/0007449 A1,2002.01.17,
US 2009/0249026 A1,2009.10.01,
CN 102103483 A,2011.06.22,
US 2004/0236920 A1,2004.11.25,
US 2002/0007449 A1,2002.01.17,
审查员 陈敏

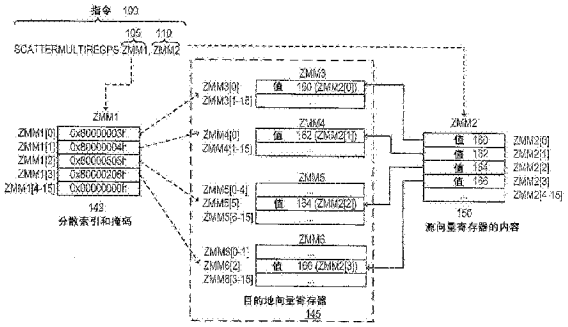
权利要求书2页 说明书30页 附图31页

(54)发明名称

多寄存器分散指令

(57)摘要

处理器获取多寄存器分散指令，其包括源操作数和目的地操作数。源操作数指定包括多个源数据元素的源向量寄存器。目的地操作数标识多个目的地数据元素，且每个目的地数据元素指定目的地向量寄存器和至该目的地向量寄存器的索引。解码并执行该指令，这导致对于这些标识的目的地数据元素中的每一个，在与该目的地数据元素的位置对应的源向量寄存器中的位置中的源数据元素之一被存储在由该目的地数据元素指定的索引处的目的地向量寄存器中。



CN 104137059 B

1.一种在计算机处理器中执行多寄存器分散指令的方法,包括:

获取多寄存器分散指令,所述多寄存器分散指令包括源操作数和目的地操作数,其中所述源操作数指定包括要分散到多个目的地向量寄存器的多个源数据元素的源向量寄存器,其中所述目的地操作数标识第一多个目的地数据元素,其中所述第一多个目的地数据元素中的每一个从所述多个目的地向量寄存器指定目的地向量寄存器和至该所指定目的地向量寄存器的索引;

解码所获取的多寄存器分散指令;以及

执行解码的多寄存器分散指令,导致对于所述第一多个目的地数据元素中的每一个,在与该目的地数据元素的位置对应的源向量寄存器中的位置中的源数据元素之一被存储在由该目的地数据元素指定的索引处的所述目的地向量寄存器中。

2.如权利要求1所述的方法,其特征在于,所述目的地操作数指定标识所述第一多个目的地数据元素的向量寄存器。

3.如权利要求2所述的方法,其特征在于,由所述目的地操作数指定的向量寄存器包括第二多个目的地数据元素,其包括第一多个目的地数据元素以及各自指示作为执行解码的多寄存器分散指令的结果而在与目的地数据元素对应的源向量寄存器中的位置中的源数据元素之一不被存储在该目的地向量寄存器中的至少一个目的地数据元素。

4.如权利要求2所述的方法,其特征在于,所述源向量寄存器和由目的地操作数指定的向量寄存器各自为512位。

5.如权利要求4所述的方法,其特征在于,所述第一多个目的地数据元素中的每一个是32位,其中8位指示目的地向量寄存器,且8位指示至该目的地向量寄存器的索引。

6.如权利要求1所述的方法,其特征在于,所述目的地操作数指定标识所述第一多个目的地数据元素的存储器位置。

7.如权利要求1所述的方法,其特征在于,所述源向量寄存器是512位。

8.一种处理器核,包括:

硬件解码单元,被配置为用于解码多寄存器分散指令,其中所述多寄存器分散指令包括源操作数和目的地操作数,其中所述源操作数指定包括要分散到多个目的地向量寄存器的多个源数据元素的源向量寄存器,其中所述目的地操作数标识第一多个目的地数据元素,其中所述第一多个目的地数据元素中的每一个从所述多个目的地向量寄存器指定目的地向量寄存器和至该所指定目的地向量寄存器的索引;以及

执行引擎单元,其耦合至所述硬件解码单元,并被配置为用于执行解码的多寄存器分散指令,这导致对于所述第一多个目的地数据元素中的每一个,在与该目的地数据元素的位置对应的源向量寄存器中的位置中的源数据元素之一被存储在由该目的地数据元素指定的索引处的所述目的地向量寄存器中。

9.如权利要求8所述的处理器核,其特征在于,所述目的地操作数指定标识所述第一多个目的地数据元素的向量寄存器。

10.如权利要求9所述的处理器核,其特征在于,由所述目的地操作数指定的向量寄存器包括第二多个目的地数据元素,其包括第一多个目的地数据元素以及各自指示作为执行解码的多寄存器分散指令的结果而在与所述目的地数据元素对应的源向量寄存器中的位置中的源数据元素之一不被存储在该目的地向量寄存器中的至少一个目的地数据元素。

11.如权利要求9所述的处理器核，其特征在于，所述源向量寄存器和由目的地操作数指定的向量寄存器各自为512位。

12.如权利要求11所述的处理器核，其特征在于，所述第一多个目的地数据元素中的每一个是32位，其中8位指示目的地向量寄存器，且8位指示至该目的地向量寄存器的索引。

13.如权利要求8所述的处理器核，其特征在于，所述目的地操作数指定标识所述第一多个目的地数据元素的存储器位置。

14.如权利要求8所述的处理器核，其特征在于，所述源向量寄存器是512位。

15.一种在计算机处理器中执行多寄存器分散指令的设备，包括：

指令获取装置，被配置为用于获取多寄存器分散指令，所述多寄存器分散指令包括源操作数和目的地操作数，其中所述源操作数指定包括要分散到多个目的地向量寄存器的多个源数据元素的源向量寄存器，其中所述目的地操作数标识第一多个目的地数据元素，其中所述第一多个目的地数据元素中的每一个从所述多个目的地向量寄存器指定目的地向量寄存器和至该所指定目的地向量寄存器的索引；以及

指令解码装置，其耦合至所述指令获取装置，并被配置为解码所获取的多寄存器分散指令；以及

指令执行装置，其耦合至所述指令解码装置，并被配置为用于执行解码的多寄存器分散指令，这导致对于所述第一多个目的地数据元素中的每一个，在与该目的地数据元素的位置对应的源向量寄存器中的位置中的源数据元素之一被存储在由该目的地数据元素指定的索引处的所述目的地向量寄存器中。

16.如权利要求15所述的设备，其特征在于，所述目的地操作数指定标识所述第一多个目的地数据元素的向量寄存器。

17.如权利要求16所述的设备，其特征在于，由所述目的地操作数指定的向量寄存器包括第二多个目的地数据元素，其包括第一多个目的地数据元素以及各自指示作为执行解码的多寄存器分散指令的结果而在与目的地数据元素对应的源向量寄存器中的位置中的源数据元素之一不被存储在目的地向量寄存器中的至少一个目的地数据元素。

18.如权利要求16所述的设备，其特征在于，所述源向量寄存器和由目的地操作数指定的向量寄存器各自为512位。

19.如权利要求18所述的设备，其特征在于，所述第一多个目的地数据元素中的每一个是32位，其中8位指示目的地向量寄存器，且8位指示至该目的地向量寄存器的索引。

20.如权利要求15所述的设备，其特征在于，所述目的地操作数指定标识所述第一多个目的地数据元素的存储器位置。

多寄存器分散指令

技术领域

[0001] 本发明的领域一般涉及计算机处理器架构，且尤其涉及多寄存器分散指令。

背景技术

[0002] 指令集、或指令集架构(ISA)是涉及编程的计算机架构的一部分，并且可包括原生数据类型、指令、寄存器架构、寻址模式、存储器架构、中断和异常处理、以及外部输入和输出(I/O)。应注意术语指令在本文中一般指的是宏指令——即提供给处理器以供执行的指令——与从处理器的解码器解码宏指令得到的微指令或微操作不同。指令集架构与微架构不同，微架构是实现ISA的处理器的内部设计。具有不同微架构的处理器可共享公共指令集。

[0003] 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(位数、位位置)以指定要执行的操作以及将对其进行该操作的操作数等。给定指令是使用给定指令格式来表达的，并指定操作和操作数。指令流是特定指令序列，其中该序列中的每一指令都是指令格式的指令出现。

[0004] 科学、金融、自动向量化的通用RMS(识别、挖掘以及合成)/可视和多媒体应用(例如，2D/3D图形、图像处理、视频压缩/解压缩、语音识别算法和音频操纵)常常需要对大量的数据项执行相同操作(被称为“数据并行性”)。单指令多数据(SIMD)是指使处理器对多个数据项执行相同操作的一种指令。SIMD技术尤其适用于可将寄存器中的多个位逻辑地划分成多个固定尺寸的数据元素的处理器，其中每个数据元素表示单独的值。例如，64位寄存器中的位可以被指定为作为四个单独的16位数据元素来操作的源操作数，每一个数据元素都表示单独的16位值。作为另一个示例，256位寄存器中的位可以被指定为作为四个单独的64位打包数据元素(四字(Q)尺寸的数据元素)、八个单独的32位打包数据元素(双字(D)尺寸的数据元素)、十六个单独的16位打包数据元素(字(W)尺寸的数据元素)、或三十二个单独的8位数据元素(字节(B)尺寸的数据元素)来操作的源操作数。这种类型的数据被称为打包数据类型或向量数据类型，并且这种数据类型的操作数被称为打包数据操作数或向量操作数。换句话说，打包数据项或向量指的是打包数据元素序列；并且打包数据操作数或向量操作数是SIMD指令(也称为打包数据指令或向量指令)的源操作数或目的地操作数。

[0005] 作为示例，一种类型的SIMD指令指定了将要以纵向方式对两个源向量操作数执行的单个向量操作，用于生成具有相同尺寸的、具有相同数量的数据元素并且按照相同数据元素次序的目的地向量操作数(也被称为结果向量操作数)。源向量操作数中的数据元素被称为源数据元素，而目的地向量操作数中的数据元素被称为目的地或结果数据元素。这些源向量操作数具有相同尺寸并且包含相同宽度的数据元素，因此它们包含相同数量的数据元素。两个源向量操作数中的相同位位置中的源数据元素形成数据元素对(也称为对应的数据元素；即，每个源操作数的数据元素位置0中的数据元素相对应，每个源操作数中的数据元素位置1中的数据元素相对应，以此类推)。对这些源数据元素对中的每一个分别执行该SIMD指令指定的操作，以产生匹配数量的结果数据元素，并且因此每一对源数据元素具

有相应的结果数据元素。由于该操作是纵向的，且由于结果向量操作数是相同尺寸、具有相同数量的数据元素并且结果数据元素按照与源向量操作数相同的数据元素顺序被存储，所以结果数据元素处于结果向量操作数中与它们在源向量操作数中的相应源数据元素对相同的位位置中。除此示例性类型的SIMD指令之外，还有各种其他类型的SIMD指令(例如，只有一个或具有两个以上的源向量操作数的；以横向方式操作的；生成不同尺寸的结果向量操作数的，具有不同尺寸的数据元素的，和/或具有不同的数据元素顺序的)。应该理解，术语目的地向量操作数(或目的地操作数)被定义为执行由指令所指定的操作的直接结果，包括将该目的地操作数存储在某一位置(寄存器或由该指令所指定的存储器地址)，以便它可以作为源操作数由另一指令访问(由另一指令指定相同位置)。

[0006] 某些指令集架构允许多个向量和标量操作并行完成并更新指令集架构寄存器集。存在这样的操作：其中一旦在向量寄存器中计算值，其元素就在不同向量寄存器之间分散。通常，利用在不同的寄存器组上的长且独立的置换和混洗链来执行分散操作，这是昂贵且复杂的。

[0007] 附图简述

[0008] 本发明是通过示例说明的，而不仅局限于各个附图的图示，在附图中，类似的参考标号表示类似的元件，其中：

[0009] 图1示出根据一个实施例的多寄存器分散指令的示例性执行；

[0010] 图2示出根据一个实施例的示例性分散索引和掩码值格式；

[0011] 图3示出根据一个实施例的多寄存器分散指令的另一个示例性执行；

[0012] 图4是示出根据一个实施例的示例性操作的流程图，示例性操作用于通过在处理器中执行多寄存器分散指令，将来自单个向量寄存器的多个值分散到多个向量寄存器；

[0013] 图5是示出根据一个实施例出现多寄存器分散指令时用于执行的示例性操作的流程图，其中目的地操作数指定向量寄存器；

[0014] 图6是示出根据一个实施例出现多寄存器分散指令时用于执行的示例性操作的流程图，其中目的地操作数指定存储器位置；

[0015] 图7示出根据一个实施例的多寄存器分散指令的示例性执行；

[0016] 图8示出根据一个实施例的示例性分散索引和掩码值格式；

[0017] 图9示出根据一个实施例的多寄存器分散指令的另一个示例性执行；

[0018] 图10是示出根据一个实施例的示例性操作的流程图，示例性操作用于通过在处理器中执行多寄存器分散指令，将来自单个向量寄存器的一个或多个值分散到多个向量寄存器；

[0019] 图11是示出根据一个实施例出现多寄存器分散指令时用于执行的示例性操作的流程图，其中目的地操作数指定向量寄存器；

[0020] 图12是示出根据一个实施例出现多寄存器分散指令时用于执行的示例性操作的流程图，其中目的地操作数指定存储器位置；

[0021] 图13a示出根据一个实施例的示例性AVX指令格式，包括VEX前缀、实操作码字段、Mod R/M字节、SIB字节、位移字段以及IMM8；

[0022] 图13B示出根据一个实施例来自图13A的哪些字段构成完整操作码字段和基础操作字段；

[0023] 图13C示出根据一个实施例来自图13A的哪些字段构成寄存器索引字段；

[0024] 图14A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；

[0025] 图14B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；

[0026] 图15A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；

[0027] 图15B是示出根据本发明的一个实施例的构成完整操作码字段的具有图15a的专用向量友好指令格式的字段的框图；

[0028] 图15C是示出根据本发明的实施例的构成寄存器索引字段的具有专用向量友好指令格式的字段的框图；

[0029] 图15D是示出根据本发明的一个实施例的构成扩充(augmentation)操作字段的具有专用向量友好指令格式的字段的框图；

[0030] 图16是根据本发明的一个实施例的寄存器架构的框图；

[0031] 图17A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无序发布/执行流水线两者的框图；

[0032] 图17B是示出根据本发明的各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无序发布/执行架构核的框图；

[0033] 图18A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的连接及其二级(L2)高速缓存的本地子集的框图；

[0034] 图18B是根据本发明的实施例的图18A中的处理器核的一部分的展开图；

[0035] 图19是根据本发明的实施例的可具有超过一个的核、可具有集成的存储器控制器、并且可具有集成图形器件的处理器的框图；

[0036] 图20是根据本发明一个实施例的系统的框图；

[0037] 图21是根据本发明的实施例的第一更具体的示例性系统的框图；

[0038] 图22是根据本发明的实施例的第二更具体的示例性系统的框图；

[0039] 图23是根据本发明的实施例的SoC的框图；以及

[0040] 图24是根据本发明的实施例的对比使用软件指令变换器将源指令集中的二进制指令变换成目标指令集中的二进制指令的框图。

[0041] 详细描述

[0042] 在以下描述中，陈述了多个具体细节。然而，应当理解的是，可不通过这些具体细节来实施本发明的实施例。在其它实例中，未详细示出公知的电路、结构以及技术，以免模糊对本描述的理解。

[0043] 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，这些短语不一定表示同一实施例。此外，当联系实施例描述特定的特征、结构或特性时，认为本领域普通技术人员能够知晓结合其它实施例来实现这种特征、结构或特性，无论是否明确描述。

[0044] 如前面详细描述地，通常利用复杂且长的置换和混洗依存(dependent)链来执行在向量寄存器中分散所计算的值的分散操作，这与目的地寄存器的数量成比例，且对于程

序器或编译器而言是耗时的且导致长指令序列。

[0045] 以下详细描述多寄存器分散指令(ScatterMultiReg)的实施例以及可用于执行该指令的系统、架构、指令格式的实施例。多寄存器分散指令包括源操作数和目的地操作数。源操作数指定包括将被存储在多个目的地向量寄存器中的多个源数据元素的源向量寄存器。目的地操作数标识多个目的地数据元素,每个目的地数据元素指定目的地向量寄存器和至该目的地向量寄存器的索引。多寄存器分散指令在执行时导致处理器将来自单个源向量寄存器的多个数据元素存储到多个目的地向量寄存器。

[0046] 在一些实施例中,目的地操作数指定标识该目的地数据元素的向量寄存器。在其它实施例中,目的地操作数指定标识该目的地数据元素的存储器位置。

[0047] 该指令的一个示例是“ScatterMultiReg[PS/PD]zmm1,zmm2”,其中zmm1和zmm2是向量寄存器(诸如128位、256位、512位寄存器)。向量寄存器zmm2包括多个源数据元素(例如,16个数据元素,假设每个数据元素是32位且zmm2是512位寄存器),根据向量寄存器zmm1的内容将它们中的至少一些存储在目的地向量寄存器中。向量寄存器zmm1包括多个数据元素,每个数据元素可指定另一个向量寄存器和至该向量寄存器的索引,当执行该指令时该向量寄存器将存储向量寄存器zmm2的与向量寄存器zmm1的数据元素相对应的位置处的数据元素。指令的“PS”部分指示标量浮点(4字节),而指令的“PD”部分指示双浮点(8字节)。用于整数向量形式的多寄存器分散指令的另一个示例还可用于诸如分散打包DWORD或QWORD整数元素的“ScatterMultiReg[D/Q]zmm1,zmm2”之类的实施例。

[0048] 该指令的另一个示例是“ScatterMultiReg[PS/PD]<memory>,zmm2”,其中<memory>是存储器中的位置且zmm2是向量寄存器(诸如128位、256位、512位寄存器)。向量寄存器zmm2包括多个源数据元素(例如,16个数据元素,假设每个数据元素是32位且zmm2是512位寄存器),根据存储器位置<memory>的内容将它们中的至少一些存储在目的地向量寄存器中。存储器位置<memory>标识多个数据元素,每个数据元素可指定向量寄存器和至该向量寄存器的索引,该向量寄存器将存储向量寄存器zmm2的数据元素。指令的“PS”部分指示标量浮点(4字节),而指令的“PD”部分指示双浮点(8字节)。用于整数向量形式的多寄存器分散指令的另一个示例还用于诸如分散打包DWORD或QWORD整数元素的“ScatterMultiReg[D/Q]<memory>,zmm2”之类的实施例。

[0049] 图1示出根据一个实施例的多寄存器分散指令的示例性执行。多寄存器分散指令100包括目的地操作数105和源操作数110。多寄存器分散指令100属于指令集架构,且在指令流内指令100的每次“出现”将包括目的地操作数105和源操作数110内的值。在该示例中,目的地操作数105和源操作数110二者均是向量寄存器(诸如128位、256位、512位寄存器)。向量寄存器可以是具有16个32位数据元素的zmm寄存器,然而,可使用其它数据元素和寄存器尺寸,诸如xmm或ymm寄存器和16位或64位数据元素。

[0050] 由源操作数(如所示的zmm2)指定的源向量寄存器的内容150包括多个源数据元素。如图1所示,在zmm2的索引0处的源数据元素包括值160,在zmm2的索引1处的源数据元素包含值162,在zmm2的索引2处的源数据元素包括值164,且在zmm2的索引3处的源数据元素包含值166。在zmm2的索引4-15处的源数据元素也包含值;然而它们未在图1中示出,因为它们未作为zmm1内容的结果而分散,以下将更详细地描述。

[0051] 向量寄存器zmm1包括多个数据元素,每个数据元素可指定分散索引和掩码值142。

每个分散索引和掩码值142可指定目的地向量寄存器和值该向量寄存器的索引，且可进一步指定是否将处于寄存器zmm2中与数据元素的位置相对应的位置处的源数据元素存储在指定的目的地向量寄存器中。对于标量浮点(PS)，每个分散索引和掩码值142是4字节(32位)。在一些实施例中，将较低的16位用于表示向量寄存器号和至该向量寄存器的索引，且最低有效位指示是否采取动作(是否将与分散索引和掩码值对应的位置中的源数据元素复制到指定索引处的指定向量寄存器中)。因此，由目的地操作数105指定的向量寄存器的数据元素指示目的地向量寄存器145和至这些寄存器的索引。

[0052] 图2示出示例性分散索引和掩码值格式210。分散索引和掩码格式210的较低8位指示向量寄存器号210。接下来的较高8位表示寄存器索引215。最高有效位是随时备用(actionable)位210，其指示是否采取行动(是否将源向量寄存器中的与分散索引和掩码值对应的位置中的源存储在寄存器号210标识的寄存器的寄存器索引215标识的目的地数据元素处)。

[0053] 例如，利用图2所示的分散索引和掩码值格式，zmm1寄存器的索引0处标识的数据元素具有分散索引和掩码值(按十六进制表示法)0x80000003h，其表示向量寄存器3(例如，zmm3)及其索引0，且是可随时备用的。Zmm1向量寄存器的索引1处标识的数据元素具有分散索引和掩码值(按十六进制表示法)0x80000004h，其表示向量寄存器4(例如，zmm4)及其索引0，且是可随时备用的。Zmm1向量寄存器的索引2处标识的数据元素具有分散索引和掩码值0x80000505h，其表示向量寄存器5(例如，zmm5)及其索引5，且是可随时备用的。zmm1向量寄存器的索引3处标识的数据元素具有分散索引和掩码值0x800000206h，其表示向量寄存器6(例如，zmm6)及其索引2，且是可随时备用的。Zmm1向量寄存器的索引4-15处标识的数据元素各自具有分散索引和掩码值0x00000000h，其表示向量寄存器0(例如，zmm0)及其索引0，且是可随时备用的(表示源向量寄存器zmm2的索引4-15处的元素将不会被分散)。

[0054] 对哪些位表示向量寄存器号且哪些位表示该向量寄存器的索引的选择在不同的实施例中可为不同选择。例如，较低的16位可表示向量寄存器号，而较高的16位可形成寄存器索引，这允许未来的ISA扩展。在这种情况下，指令还可包括指示是否随时备用的掩码寄存器。

[0055] 在一些实施例中，作为指令执行的结果，由与分散索引和掩码值的值相对应的位置中的源操作数指定的向量寄存器的源数据元素被存储在由分散索引和掩码值标识的索引处由分散索引和掩码值标识的向量寄存器中(假设分散索引和掩码值指示它是随时可用的)。例如，在这一实施例中，因为数据元素zmm1[0]的分散索引和掩码值(zmm1寄存器的索引0)是0x80000003h，作为指令执行的结果，值160(即在与分散索引和掩码值相同位置处的源操作数110指定的向量寄存器处的源数据元素值(zmm2[0]))存储在向量寄存器zmm3的索引0处的目的地数据元素中。

[0056] 如图1所示，作为指令执行的结果，如zmm1[0]的分散索引和掩码值所指示的，值160(即在zmm2[0]处的源数据元素的值)存储在zmm3[0]处的数据元素中。作为指令执行的结果，如zmm1[1]的分散索引和掩码值所指示的，值162(即在zmm2[1]处的源数据元素的值)存储在zmm4[0]处的数据元素中。作为指令执行的结果，如zmm1[2]的分散索引和掩码值所指示的，值164(即在zmm2[2]处的源数据元素的值)存储在zmm5[5]处的数据元素中。作为指令执行的结果，如zmm1[3]的分散索引和掩码值所指示的，值166(即在zmm2[3]处的源数据

元素的值)存储在zmm6[2]处的数据元素中。因为,在zmm1[4-15]处的每个数据元素的分散索引和掩码值指示不采取动作,所以作为指令执行的结果,在zmm2[4-15]处的源数据元素将不被存储在目的地数据元素中。

[0057] 因此,在所执行的指令100出现之后,来自zmm2[0]的值160存储在zmm3[0]中(作为执行指令100的结果,数据元素zmm3[1-15]不变),来自zmm2[1]的值162存储在zmm4[0]中(作为执行指令100的结果,数据元素zmm4[1-15]不变),来自zmm2[2]的值164存储在zmm5[5]中(作为执行指令100的结果,数据元素zmm5[0-4]和zmm5[6-15]不变),来自zmm2[3]的值166存储在zmm6[2]中(作为执行指令100的结果,数据元素zmm6[0-1]和zmm6[3-15]不变)。

[0058] 图3示出多寄存器分散指令的另一示例性执行。多寄存器分散指令300包括目的地操作数305和源操作数310。多寄存器分散指令300属于指令集架构,且在指令流内指令300的每次"出现"将包括目的地操作数305和源操作数310内的值。在该示例中,源操作数310是向量寄存器(诸如,128位、256位、512位寄存器)且目的地操作数305是标识多个目的地数据元素的存储器位置,每个目的地数据元素可指定分散索引和掩码值142,该分散索引和掩码值可指定目的地向量寄存器和至该向量寄存器的索引,且还可指定与数据元素的位置对应的寄存器zmm2的位置中的源数据元素是否被存储在指定的目的地向量寄存器中。在一个实施例中,每个分散索引和掩码值342的格式与参考图1和2描述的相同。

[0059] 由源操作数(如所示的zmm2)指定的源向量寄存器的内容350包括多个源数据元素。如图3所示,在zmm2的索引0处的源数据元素包括值360,在zmm2的索引1处的源数据元素包含值362,在zmm2的索引2处的源数据元素包括值364,且在zmm2的索引3处的源数据元素包含值366。在zmm2的索引4-15处的源数据元素包含值;然而它们未在图3中示出,因为它们未作为由目的地操作数指定的存储器位置的内容的结果而分散,以下将更详细地描述。

[0060] 利用图2所示的分散索引和掩码值格式,由操作数305指定的存储器位置的索引0处标识的数据元素具有分散索引和掩码值(按十六进制表示法)0x80000003h,其表示向量寄存器3(例如,zmm3)及其索引0,且是可随时备用的。由操作数305指定的存储器位置的索引1处标识的数据元素具有分散索引和掩码值(按十六进制表示法)0x80000004h,它表示向量寄存器4(例如,zmm4)及其索引0,且是随时备用的。由操作数305指定的存储器位置的索引2处标识的数据元素具有分散索引和掩码值0x80000505h,它表示向量寄存器5(例如,zmm5)及其索引5,且是随时备用的。由操作数305指定的存储器位置的索引3处标识的数据元素具有分散索引和掩码值0x800000206h,它表示向量寄存器6(例如,zmm6)及其索引2,且是随时备用的。由操作数305指定的存储器位置的索引4-15处标识的数据元素各自具有分散索引和掩码值0x0h,它指示将不会分散源向量寄存器zmm2的索引4-15中的元素。

[0061] 如图3所示,作为指令执行的结果,如由操作数305指定的存储器位置的索引0的分散索引和掩码值所指示的,值360(即在zmm2[0]处的源数据元素的值)存储在zmm3[0]处的数据元素中。作为指令执行的结果,如由操作数305指定的存储器位置的索引1的分散索引和掩码值所指示的,值362(即在zmm2[1]处的源数据元素的值)存储在zmm4[0]处的数据元素中。作为指令执行的结果,如由操作数305指定的存储器位置的索引2的分散索引和掩码值所指示的,值364(即在zmm2[2]处的源数据元素的值)存储在zmm5[5]处的数据元素中。作为指令执行的结果,如由操作数305指定的存储器位置的索引3的分散索引和掩码值所指示

的，值366(即在zmm2[3]处的源数据元素的值)存储在zmm6[2]处的数据元素中。因为由操作数305指定的存储器位置的每个数据元素索引4-15的分散索引和掩码值指示不采取动作，所以作为指令执行的结果，在zmm2[4-15]处的源数据元素将不被存储在目的地数据元素中。

[0062] 因此，利用单个指令，多寄存器分散指令将来自单个向量寄存器的不同源数据元素分散到多个向量寄存器的数据元素。在一些实施例中，优化访问，使得当在特定通道中时，在一次扫描中，从该通道的所有寄存器复制值。因为利用单个指令，多寄存器分散指令将来自单个向量寄存器的元素分散到多个向量寄存器，所以它去除了先前必须的昂贵的混洗和置换，从而提高性能。

[0063] 多寄存器分散指令可通过编译器自动生成，或者可由软件开发者手动编码。单个多寄存器分散指令不仅节省指令数，还降低编程复杂度。单个多寄存器分散指令还减小执行端口压力并减少内部缓冲器(诸如，RS(保留站)、ROB(重排序缓冲器)、获取和解码缓冲器)的使用，从而给予提高的性能和降低的功耗。

[0064] 图4是示出根据一个实施例的示例性操作的流程图，示例性操作用于通过在处理器中执行多寄存器分散指令，将来自单个向量寄存器的多个值分散到多个向量寄存器。在操作410，通过处理器获取多寄存器分散指令(例如，通过处理器的获取单元)。多寄存器分散指令包括源操作数和目的地操作数。源操作数指定包括将被分散到多个目的地向量寄存器(例如，xmm、ymm或zmm寄存器)的多个源数据元素的源向量寄存器。目的地操作数标识多个目的地数据元素，且每个目的地数据元素指定目的地向量寄存器和至该目的地向量寄存器的索引。

[0065] 例如，在一个实施例中，目的地操作数指定向量寄存器(例如，xmm、ymm或zmm寄存器)或存储器位置，该向量寄存器或存储器位置标识指定目的地向量寄存器和至该目的地向量寄存器的索引的多个数据元素。每个数据元素还指定来自源向量寄存器的数据元素(由源操作数指定)是否应被复制到指定的目的地向量寄存器。

[0066] 流程从操作410移动到操作415，其中处理器解码多寄存器分散指令。例如，在一些实施例中，处理器包括硬件解码单元，指令被提供给该解码单元(例如，通过处理器的取出单元)。对于解码单元，可使用各种不同的公知解码单元。例如，该解码单元可以将多寄存器分散指令解码为单一宽微指令。作为另一个示例，该解码单元可以将多寄存器分散指令解码为多个宽微指令。作为特别适于无序处理器流水线的另一示例，该解码单元可以将多寄存器分散指令解码为一个或多个微操作(micro-op)，其中每个微操作可被发出并无序执行。而且，解码单元可以被实现为具有一个或多个解码器，并且每个解码器可被实现为可编程逻辑阵列(PLA)，如本领域所公知的。作为示例，给定解码单元可以：1)具有导引逻辑以便将不同的宏指令定向到不同的解码器；2)第一解码器，可解码指令集的子集(但是比第二、第三和第四解码器解码得更多)，并且每次生成两个微操作；3)第二、第三和第四解码器，可各自仅解码完整指令集的子集，并且每次仅生成一个微操作；4)微序列发生器ROM，可以仅解码完整指令集的子集并且每次生成四个微操作；以及5)由解码器和微序列发生器ROM提供馈送的复用逻辑，确定谁的输出被提供至微操作队列。解码单元的其他实施例可具有解码更多或更少指令和指令子集的更多或更少的解码器。例如，一个实施例可具有第二、第三和第四解码器，该第二、第三和第四解码器可每次各生成两个微操作；并且可包括每次生成

8个微操作的微序列发生器ROM。

[0067] 流程然后移动到操作420，其中处理器执行经解码的多寄存器分散指令，导致对于通过目的地操作数标识的每个目的地数据元素，在与目的地数据元素的位置对应的源向量寄存器中的位置中的由源操作数指定的源数据元素之一被存储在由目的地数据元素指定的索引处的目的地向量寄存器中。

[0068] 图5是示出根据一个实施例出现多寄存器分散指令时用于执行的示例性操作的流程图，其中目的地操作数指定向量寄存器。在一个实施例中，参考图5描述的操作结合操作420进行。

[0069] 在操作510，处理器读取由目的地操作数指定的向量寄存器的第一数据元素的值(分散索引和掩码值)。在一个实施例中，该值采用分散索引格式210的形式。流程然后移动到操作515，其中处理器确定分散索引和掩码值是否指示应从源操作数指定的源向量寄存器的对应位置复制数据元素。例如，参考分散索引和掩码值格式210，处理器确定在分散索引和掩码值中随时备用位220是否被置位。如果分散索引和掩码值指示应从源向量寄存器复制数据元素，则流程移动到操作530，否则流程移动到操作520。

[0070] 在操作520，处理器读取由目的地操作数指定的向量寄存器的后续数据元素的值(下一分散索引和掩码值)。流程然后移动到操作525，其中处理器确定分散索引和掩码值是否指示应从源操作数指定的源向量寄存器的对应位置复制数据元素。如果是，则流程移动到操作530，否则流程移动到操作540。

[0071] 在操作530，处理器确定由分散索引和掩码值指示的目的地向量寄存器和寄存器的索引。例如，遵循分散索引和掩码值格式210的格式，值的较低8位标识向量寄存器(通过寄存器号)，接下来的较高8位标识至该寄存器的索引。流程然后移动到操作535，其中处理器在所确定的索引处的所确定的目的地向量寄存器中存储在与目的地操作数指定的数据元素的位置(由目的地操作数指定的向量寄存器中的分散索引和掩码值的位置)相对应的位置中的由源操作数指定的源向量寄存器的源数据元素。流程从操作535移动到操作540。

[0072] 在操作540，处理器确定在目的地向量寄存器中是否有另一个数据元素。如果是，则流程移动回到操作520。如果不是，则操作完成。

[0073] 图6是示出根据一个实施例出现多寄存器分散指令时用于执行的示例性操作的流程图，其中目的地操作数指定存储器位置。在一个实施例中，参考图6描述的操作结合操作420进行。

[0074] 在操作610，处理器检索目的地操作数中标识的存储器位置所指定的数据元素。在一个实施例中，所检索的数据元素各自采用分散索引和掩码格式210的形式。流程然后移动到操作615，其中处理器确定第一检索的数据元素的分散索引和掩码值是否指示应从源操作数指定的源向量寄存器的对应位置复制数据元素。如果分散索引和掩码值指示应从源向量寄存器复制数据元素，则流程移动到操作630，否则流程移动到操作620。

[0075] 在操作620，处理器读取由目的地操作数指定的从存储器位置检索的后续数据元素的值(下一分散索引和掩码值)。流程然后移动到操作625，其中处理器确定分散索引和掩码值是否指示应从源操作数指定的源向量寄存器的对应位置复制数据元素。如果是，则流程移动到操作630，否则流程移动到操作640。

[0076] 在操作630，处理器确定由分散索引和掩码值指示的寄存器的索引和目的地向量

寄存器。流程然后移动到操作635，其中处理器在所确定的索引处的所确定的目的地向量寄存器中存储在与目的地操作数指定的数据元素的位置(由目的地操作数指定的存储器位置中的分散索引和掩码值的位置)相对应的位置中的由源操作数指定的源向量寄存器的源数据元素。流程从操作635移动到操作640。

[0077] 在操作640，处理器确定在目的地操作数指定的存储器位置中是否有另一个数据元素。如果是，则流程移动回到操作620。如果不是，则操作完成。

[0078] 替换实施例

[0079] 尽管以上描述的实施例描述了多寄存器分散指令，其中对应于所指定的数据元素中的分散索引和掩码值的位置的源向量寄存器中的位置确定要存储在目的地中的源数据元素，但在替换实施例中，掩码值允许选择源向量寄存器的索引。

[0080] 例如，该指令的一个示例是“ScatterMultiRegVar[PS/PD]zmm1,zmm2”，其中zmm1和zmm2是向量寄存器(诸如128位、256位、512位寄存器)。向量寄存器zmm2包括多个源数据元素(例如，16个数据元素，假设每个数据元素是32位且zmm2是512位寄存器)，根据向量寄存器zmm1的内容将它们中的至少一个存储在目的地向量寄存器中。向量寄存器zmm1包括多个数据元素，其中的每一个可指定另一个向量寄存器(目的地向量寄存器)和至该向量寄存器的索引(目的地索引)，且还指定至源向量寄存器的索引(源索引)。作为执行指令的结果，对应于所指定的源索引的源数据元素将被存储在目的地索引处的目的地向量寄存器中。指令的“PS”部分指示标量浮点(4字节)，而指令的“PD”部分指示双浮点(8字节)。用于整数向量形式的多寄存器分散指令的另一个示例还用于诸如分散打包DWORD或QWORD整数元素的“ScatterMultiReg[D/Q]zmm1,zmm2”之类的实施例。

[0081] 该指令的另一个示例是“ScatterMultiRegVar[PS/PD]<memory>,zmm2”，其中<memory>是存储器中的位置且zmm2是向量寄存器(诸如128位、256位、512位寄存器)。向量寄存器zmm2包括多个源数据元素(例如，16个数据元素，假设每个数据元素是32位且zmm2是512位寄存器)，根据存储器位置<memory>的内容将它们中的至少一个存储在目的地向量寄存器中。存储器位置<memory>标识多个数据元素，其中的每一个可指定向量寄存器(目的地向量寄存器)和至该向量寄存器的索引(目的地索引)，且还指定至该存储器的索引(源索引)。作为执行指令的结果，对应于所指定的源索引的源数据元素将被存储在目的地索引处的目的地向量寄存器中。指令的“PS”部分指示标量浮点(4字节)，而指令的“PD”部分指示双浮点(8字节)。用于整数向量形式的多寄存器分散指令的另一个示例还用于诸如分散打包DWORD或QWORD整数元素的“ScatterMultiReg[D/Q]<memory>,zmm2”之类的实施例。

[0082] 图7示出根据一个实施例的多寄存器分散指令的示例性执行。多寄存器分散指令700包括目的地操作数705和源操作数710。多寄存器分散指令700属于指令集架构，且在指令流内指令700的每次“出现”将包括目的地操作数705和源操作数710内的值。在该示例中，目的地操作数705和源操作数710二者均是向量寄存器(诸如128位、256位、512位寄存器)。向量寄存器可以是具有16个32位数据元素的zmm寄存器，然而，可使用其它数据元素和寄存器尺寸，诸如xmm或ymm寄存器和16位或64位数据元素。

[0083] 由源操作数(如所示的zmm2)指定的源向量寄存器的内容750包括多个源数据元素。如图7所示，在zmm2的索引0处的源数据元素包括值760，在zmm2的索引1处的源数据元素包含值762，且在zmm2的索引2处的源数据元素包括值764。在zmm2的索引3-15处的源数据元

素也可包含值;然而它们未在图7中示出,因为它们未作为zmm1内容的结果而分散,以下将更详细地描述。

[0084] 向量寄存器zmm1包括多个数据元素,每个数据元素可指定分散索引和掩码值742。每个分散的索引和掩码值742可指示以下内容:目的地向量寄存器、至该目的地向量寄存器的索引(目的地寄存器索引)和至源向量寄存器的索引(源寄存器索引)。每个分散的索引和掩码值742还可指示指令的执行是否导致与源寄存器索引相对应的数据元素被存储在目的地向量寄存器中。对于标量浮点(PS),每个分散索引和掩码值742是4字节(32位)。在一些实施例中,较低的16位用于表示目的地向量寄存器的数量和至该向量寄存器的索引,较高的8位用于表示至源向量寄存器的索引,且最高有效位指示是否采取动作(是否将与源向量寄存器的索引相对应的源数据元素复制到指定索引处的指定的目的地向量寄存器中)。因此,由目的地操作数705指定的向量寄存器的数据元素指示目的地向量寄存器745、至这些寄存器的索引,还指示将要复制的源数据元素。

[0085] 图8示出示例性分散索引和掩码值格式810。分散索引和掩码格式810的较低8位指示目的地寄存器号810。接下来的较高8位表示目的地寄存器索引815。接下来的较高8位表示源寄存器索引825。最高有效位是随时备用的位810,其指示是否采取行动(是否将源寄存器索引825所标识源寄存器索引处的数据元素存储在寄存器号810所标识寄存器的寄存器索引所标识的目的地数据元素处)。因此,与不提供源寄存器索引的示例性分散索引和掩码值格式210不同,分散索引和掩码值格式810允许从将要分散的源向量寄存器750选择源数据元素。例如,这允许单个数据元素被分散到多个不同的向量寄存器。

[0086] 例如,利用图8所示的分散索引和掩码值格式,zmm1寄存器的索引0处标识的数据元素具有分散索引和掩码值(按十六进制表示法)0x80000003h,其表示目的地向量寄存器3(例如,zmm3)及其索引0,以及(由源操作数710指定的向量寄存器zmm2的)源寄存器索引0,且是可随时备用的。zmm1向量寄存器的索引1处标识的数据元素具有分散索引和掩码值(按十六进制表示法)0x80010004h,其表示目的地向量寄存器4(例如,zmm4)及其索引0,(向量寄存器zmm2的)源寄存器索引1,且是可随时备用的。zmm1向量寄存器的索引2处标识的数据元素具有分散索引和掩码值0x80010505h,其表示向量寄存器5(例如,zmm5)及其索引5,(向量寄存器zmm2的)源寄存器索引1,且是可随时备用的。zmm1向量寄存器的索引3处标识的数据元素具有分散索引和掩码值0x800200206h,其表示向量寄存器6(例如,zmm6)及其索引2,(向量寄存器zmm2的)源寄存器索引2,且是可随时备用的。zmm1向量寄存器的索引4-15处标识的数据元素各自具有分散索引和掩码值0x00000000h,其表示向量寄存器0(例如,zmm0)及其索引0、源寄存器索引0,且是随时备用的(表示作为zmm1寄存器的索引4-15的分散索引和掩码值的结果没有被分散的元素)。

[0087] 选择哪些位表示目的地向量寄存器号、哪些位表示该目的地向量寄存器的索引、哪些位表示源向量寄存器的索引等在不同的实施例中是不同的。

[0088] 如图7所示,作为指令执行的结果,如zmm1[0]的分散索引和掩码值所指示的,值760(即在zmm2[0]处的源数据元素的值)存储在zmm3[0]处的数据元素中。作为指令执行的结果,如zmm1[1]的分散索引和掩码值所指示的,值762(即在zmm2[1]处的源数据元素的值)存储在zmm4[0]处的数据元素中。作为指令执行的结果,如zmm1[2]的分散索引和掩码值所指示的,值762(即在zmm2[1]处的源数据元素的值)也被存储在zmm5[5]处的数据元素中。因

此，值762已经被分散到多个目的地数据元素。作为指令执行的结果，如zmm1[3]的分散索引和掩码值所指示的，值764(即在zmm2[2]处的源数据元素的值)存储在zmm6[2]处的数据元素中。

[0089] 因此，在所执行的指令700出现之后，来自zmm2[0]的值760存储在zmm3[0]中(作为执行指令700的结果，数据元素zmm3[1-15]不变)，来自zmm2[1]的值162存储在zmm4[0]和zmm5[5]中(作为执行指令700的结果，数据元素zmm4[1-15]和zmm5[0-4;6-15]不变)，且来自zmm2[2]的值164存储在zmm6[2]中(作为执行指令700的结果，数据元素zmm6[0-1]和zmm6[3-15]不变)。

[0090] 图9示出根据一个实施例的多寄存器分散指令的另一个示例性执行。多寄存器分散指令900包括目的地操作数905和源操作数910。多寄存器分散指令900属于指令集架构，且在指令流内指令900的每次“出现”将包括目的地操作数905和源操作数910内的值。在该示例中，源操作数910是向量寄存器(诸如128位、256位、512位寄存器)，且目的地操作数905是存储器中的位置，其标识多个目的地数据元素，每个目的地数据元素可指定分散索引和掩码值942。在一个实施例中，分散索引和掩码值942的格式与参考图7和8描述的相同。

[0091] 由源操作数(如所示的zmm2)指定的源向量寄存器的内容950包括多个源数据元素。如图9所示，在zmm2的索引0处的源数据元素包括值960，在zmm2的索引1处的源数据元素包含值962，且在zmm2的索引2处的源数据元素包括值964。在zmm2的索引4-15处的源数据元素也可包含值；然而它们未在图9中示出，因为它们未作为由目的地操作数905指定的存储位置的内容的结果而分散，以下将更详细地描述。

[0092] 由操作数905指定的存储器位置包括多个数据元素，每个数据元素可指定分散索引和掩码值942。每个分散的索引和掩码值942可指示以下内容：目的地向量寄存器、至该目的地向量寄存器的索引(目的地寄存器索引)和至源向量寄存器的索引(源寄存器索引)。因此，由目的地操作数905指定的存储器位置的数据元素指示目的地向量寄存器945、至这些寄存器的索引，还指示将要复制的源数据元素。

[0093] 例如，利用图9所示的分散索引和掩码值格式，由操作数905指定的存储器位置的索引0处标识的数据元素具有分散索引和掩码值(按十六进制表示法)0x80000003h，其表示目的地向量寄存器3(例如，zmm3)及其索引0，以及(由源操作数710指定的向量寄存器zmm2的)源寄存器索引0，且是可随时备用的。由操作数905指定的存储器位置的索引1处标识的数据元素具有分散索引和掩码值(按十六进制表示法)0x80010004h，其表示目的地向量寄存器4(例如，zmm4)及其索引0，(向量寄存器zmm2的)源寄存器索引1，且是可随时备用的。由操作数905指定的存储器位置的索引2处标识的数据元素具有分散索引和掩码值0x80010505h，其表示向量寄存器5(例如，zmm5)及其索引5，(向量寄存器zmm2的)源寄存器索引1，且是可随时备用的。由操作数905指定的存储器位置的索引3处标识的数据元素具有分散索引和掩码值0x800200206h，其表示向量寄存器6(例如，zmm6)及其索引2，(向量寄存器zmm2的)源寄存器索引2，且是可随时备用的。由操作数905指定的存储器位置的索引4-15处标识的数据元素各自具有分散索引和掩码值0x0h，其指示对于这些数据元素将不分散元素。

[0094] 如图9所示，作为指令执行的结果，如由操作数905指定的存储器位置的索引0的分散索引和掩码值所指示的，值960(即在zmm2[0]处的源数据元素的值)存储在zmm3[0]处的

数据元素中。作为指令执行的结果，如由操作数905指定的存储器位置的索引1的分散索引和掩码值所指示的，值962(即在zmm2[1]处的源数据元素的值)存储在zmm4[0]处的数据元素中。作为指令执行的结果，如由操作数905指定的存储器位置的索引2的分散索引和掩码值所指示的，值962(即在zmm2[1]处的源数据元素的值)也存储在zmm5[5]处的数据元素中。因此，值962已经被分散到多个目的地数据元素。作为指令执行的结果，如由操作数905指定的存储器位置的索引3的分散索引和掩码值所指示的，值964(即在zmm2[2]处的源数据元素的值)存储在zmm6[2]处的数据元素中。

[0095] 因此，参考图7-9描述的多寄存器分组指令导致利用单个指令将来自单个向量寄存器的多个或多个源数据元素存储在多个向量寄存器的数据元素中。在一些实施例中，优化访问，使得当在特定通道中时，在一次扫描中，从该通道的所有寄存器复制值。因为利用单个指令，多寄存器分散指令将来自单个向量寄存器的元素分散到多个向量寄存器，所以它去除了先前必须的昂贵的混洗和置换，从而提高性能。

[0096] 图10是示出根据一个实施例的示例性操作的流程图，示例性操作用于通过在处理器中执行多寄存器分散指令，将来自单个向量寄存器的一个或多个值分散到多个向量寄存器。在操作1010，通过处理器获取多寄存器分散指令(例如，通过处理器的获取单元)。多寄存器分散指令包括源操作数和目的地操作数。源操作数指定包括多个源数据元素的源向量寄存器；多个源数据元素中的一个或多个将被分散到多个目的地向量寄存器(例如，xmm、ymm或zmm寄存器)。目的地操作数标识多个目的地数据元素，每个目的地数据元素指定目的地向量寄存器、至该目的地向量寄存器的索引、以及至由源操作数指定的源向量寄存器的索引。

[0097] 例如，在一个实施例中，目的地操作数指定向量寄存器(例如，xmm、ymm或zmm寄存器)或存储器位置，该向量寄存器或存储器位置标识指定目的地向量寄存器、至该目的地向量寄存器的索引以及至由源操作数标识的源向量寄存器的索引的多个数据元素。每个数据元素还可指定来自源向量寄存器的数据元素是否应被复制到指定的目的地向量寄存器。

[0098] 流程从操作1010移动到操作1015，其中处理器解码多寄存器分散指令。例如，在一些实施例中，处理器包括硬件解码单元，指令被提供给该解码单元(例如，通过处理器的取出单元)。如上所述，对于解码单元，可使用各种不同的公知解码单元。流程然后移动到操作1020，其中处理器执行经解码的多寄存器分散指令，导致对于目的地数据元素中的每一个，与源向量寄存器的所指定的索引对应的源数据元素被存储在由目的地数据元素指定的索引处的目的地向量寄存器中。

[0099] 图11是示出根据一个实施例出现多寄存器分散指令时用于执行的示例性操作的流程图，其中目的地操作数指定向量寄存器。在一个实施例中，参考图11描述的操作结合操作1020进行。

[0100] 在操作1110，处理器读取由目的地操作数指定的向量寄存器的第一数据元素的值(分散索引和掩码值)。在一个实施例中，该值采用分散索引和掩码值格式810的形式。流程然后移动到操作1115，其中处理器确定分散索引和掩码值是否指示应从源向量寄存器复制数据元素。例如，参考分散索引和掩码值格式810，处理器确定在分散索引和掩码值中随时备用位820是否被置位。如果分散索引和掩码值指示应从源向量寄存器复制数据元素，则流程移动到操作1130，否则流程移动到操作1120。

[0101] 在操作1120，处理器读取由目的地操作数指定的向量寄存器的后续数据元素的值(下一分散索引和掩码值)。流程然后移动到操作1125，其中处理器确定分散索引和掩码值是否指示应从源操作数指定的源向量寄存器复制数据元素。如果是，则流程移动到操作1130，否则流程移动到操作1140。

[0102] 在操作1130，处理器确定由分散索引和掩码值指示的目的地向量寄存器、至该目的地向量寄存器的索引和源向量寄存器的索引。例如，遵循分散索引和掩码值格式810的格式，值的较低8位标识目的地向量寄存器(通过寄存器号)，接下来的较高8位标识至该目的地向量寄存器的索引，并且接下来的较高8位标识至该源向量寄存器的索引。

[0103] 流程然后移动到操作1135，其中处理器将所确定的源数据元素存储在所确定的索引处的所确定的目的地向量寄存器中。流程从操作1135移动到操作1140。

[0104] 在操作1140，处理器确定在目的地向量寄存器中是否有另一个数据元素。如果是，则流程移动回到操作1120。如果不是，则操作完成。

[0105] 图12是示出根据一个实施例出现多寄存器分散指令时用于执行的示例性操作的流程图，其中目的地操作数指定存储器位置。在一个实施例中，参考图12描述的操作结合操作1020进行。在操作1210，处理器检索目的地操作数中标识的存储器位置所指定的数据元素。在一个实施例中，所检索的数据元素各自采用分散索引和掩码格式810的形式。流程然后移动到操作1215，其中处理器确定第一检索的数据元素的分散索引和掩码值是否指示应从源操作数指定的源向量寄存器复制数据元素。如果分散索引和掩码值指示应从源向量寄存器复制数据元素，则流程移动到操作1230，否则流程移动到操作1220。

[0106] 在操作1220，处理器读取由目的地操作数指定的从存储器位置检索的后续数据元素的值(下一分散索引和掩码值)。流程然后移动到操作1225，其中处理器确定分散索引和掩码值是否指示应从源操作数指定的源向量寄存器复制数据元素。如果是，则流程移动到操作1230，否则流程移动到操作1240。

[0107] 在操作1230，处理器确定由分散索引和掩码值指示的目的地向量寄存器、至该目的地向量寄存器的索引和源向量寄存器的索引。流程然后移动到操作1235，其中处理器将所确定的源数据元素存储在所确定的索引处的所确定的目的地向量寄存器中。流程从操作1235移动到操作1240。

[0108] 在操作1240，处理器确定在目的地操作数指定的存储器位置中是否有另一个数据元素。如果是，则流程移动回到操作1220。如果不是，则操作完成。

[0109] 示例性指令格式

[0110] 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。

[0111] VEX指令格式

[0112] VEX编码允许指令具有两个以上操作数，并且允许SIMD向量寄存器比128位长。VEX前缀的使用提供了三个操作数(或者更多)句法。例如，先前的两操作数指令执行盖写源操作数的操作(诸如A＝A+B)。VEX前缀的使用使操作数执行非破坏性操作，诸如A＝B+C。

[0113] 图13A示出示例性AVX指令格式，包括VEX前缀1302、实操作码字段1330、MoD R/M字节1340、SIB字节1350、位移字段1362以及IMM81372。图13B示出来自图13A的哪些字段构成

完整操作码字段1374和基础操作字段1342。图13C示出来自图13A的哪些字段构成寄存器索引字段1344。

[0114] VEX前缀(字节0-2)1302以三字节形式进行编码。第一字节是格式字段1340(VEX字节0,位[7:0]),该格式字段包含明确的C4字节值(用于区分C4指令格式的唯一值)。第二-第三字节(VEX字节1-2)包括提供专用能力的多个位字段。具体地,REX字段1305(VEX字节1,位[7-5])由VEX.R位字段(VEX字节1,位[7]-R)、VEX.X位字段(VEX字节1,位[6]-X)以及VEX.B位字段(VEX字节1,位[5]-B)组成。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VEX.R、VEX.X以及VEX.B来形成Rrrr、Xxxx以及Bbbb。操作码映射字段1315(VEX字节1,位[4:0]-mmmmm)包括对隐含的前导操作码字节进行编码的内容。W字段1364(VEX字节2,位[7]-W)由记号VEX.W表示,并且提供取决于该指令而不同的功能。VEX.vvvv 1320(VEX字节2,位[6:3]-vvvv)的作用可包括如下:1)VEX.vvvv编码第一源寄存器操作数且对具有两个或两个以上源操作数的指令有效,第一源寄存器操作数以反转(1补码)形式被指定;2)VEX.vvvv编码目的地寄存器操作数,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)VEX.vvvv不编码任何操作数,保留该字段,并且应当包含1111b。如果VEX.L 1368尺寸字段(VEX字节2,位[2]-L)=0,则它指示128位向量;如果VEX.L=1,则它指示256位向量。前缀编码字段1325(VEX字节2,位[1:0]-pp)提供了用于基础操作字段的附加位。

[0115] 实操作码字段1330(字节3)还被称为操作码字节。操作码的一部分在该字段中被指定。

[0116] MOD R/M字段1340(字节4)包括MOD字段1342(位[7-6])、Reg字段1344(位[5-3])、以及R/M字段1346(位[2-0])。Reg字段1344的作用可包括如下:对目的地寄存器操作数或源寄存器操作数(Rrrr中的rrr)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1346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。

[0117] 缩放、索引、基址(SIB)—缩放字段1350(字节5)的内容包括用于存储器地址生成的SS1352(位[7-6])。先前已经针对寄存器索引Xxxx和Bbbb参考了SIB.xxx 1354(位[5-3])和SIB.bbb 1356(位[2-0])的内容。

[0118] 位移字段1362和立即数字段(IMM8)1372包含地址数据。

[0119] 通用向量友好指令格式

[0120] 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操作两者的实施例,但是替换实施例仅使用通过向量友好指令格式的向量操作。

[0121] 图14A-14B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。图14A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14B是示出了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。具体地,针对通用向量友好指令格式1400定义A类和B类指令模板,两者包括无存储器访问1405的指令模板和存储器访问1420的指令模板。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。

[0122] 尽管将描述其中向量友好指令格式支持64字节向量操作数长度(或尺寸)与32位

(4字节)或64位(8字节)数据元素宽度(或尺寸)(并且由此,64字节向量由16双字尺寸的元素或者替换地8四字尺寸的元素组成)、64字节向量操作数长度(或尺寸)与16位(2字节)或8位(1字节)数据元素宽度(或尺寸)、32字节向量操作数长度(或尺寸)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(或尺寸)、以及16字节向量操作数长度(或尺寸)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(或尺寸)的本发明的实施例,但是替换实施例可支持更大、更小、和/或不同的向量操作数尺寸(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)。

[0123] 图14A中的A类指令模板包括:1)在无存储器访问1405的指令模板内,示出无存储器访问的完全舍入(round)控制型操作1410的指令模板、以及无存储器访问的数据变换型操作1415的指令模板;以及2)在存储器访问1420的指令模板内,示出存储器访问的时间性1425的指令模板和存储器访问的非时间性1430的指令模板。图14B中的B类指令模板包括:1)在无存储器访问1405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1412的指令模板以及无存储器访问的写掩码控制的vsize型操作1417的指令模板;以及2)在存储器访问1420的指令模板内,示出存储器访问的写掩码控制1427的指令模板。

[0124] 通用向量友好指令格式1400包括以下列出以在图14A-14B中示出的顺序的如下字段。

[0125] 格式字段1440—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,该字段在无需仅有通用向量友好指令格式的指令集的意义上是任选的。

[0126] 基础操作字段1442—其内容区分不同的基础操作。

[0127] 寄存器索引字段1444-其内容直接或者通过地址生成来指定源或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)个寄存器组选择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,但是替换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)。

[0128] 修饰符(modifier)字段1446—其内容将以指定存储器访问的通用向量指令格式出现的指令与不指定存储器访问的通用向量指令格式出现的指令区分开;即在无存储器访问1405的指令模板与存储器访问1420的指令模板之间。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,但是替换实施例可支持更多、更少或不同的方式来执行存储器地址计算。

[0129] 扩充操作字段1450—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。该字段是上下文专用的。在本发明的一个实施例中,该字段被分成类字段1468、α字段1452、以及β字段1454。扩充操作字段1450允许在单一指令而非2、3或4个指令中执行多组共同的操作。

[0130] 缩放字段1460—其内容允许用于存储器地址生成(例如,用于使用$2^{缩放}$*索引+基址

的地址生成)的索引字段的内容的按比例缩放。

[0131] 位移字段1462A—其内容被用作存储器地址生成的一部分(例如,用于使用$2^{缩放}$*索引+基址+位移的地址生成)。

[0132] 位移因数字段1462B(注意,位移字段1462A直接在位移因数字段1462B上的并置指示了使用一个或另一个)——其内容被用作地址生成的一部分,它指定通过存储器访问尺寸(N)按比例缩放的位移因数,其中N是存储器访问中的字节的数量(例如,用于使用$2^{缩放}$*索引+基址+经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尺寸(N)以生成在计算有效地址中使用的最终位移。N的值由处理器硬件在运行时基于完整操作码字段1474(稍候在本文中描述)和数据操纵字段1454C确定。位移字段1462A和位移因数字段1462B在它们不用于无存储器访问1405的指令模板和/或不同的实施例可实现两者中的仅一个或均未实现的意义上是任选的。

[0133] 数据元素宽度字段1464—其内容区分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,在其他实施例中只用于一些指令)。该字段在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不需要的意义上是任选的。

[0134] 写掩码字段1470—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,而B类指令模板支持合并写掩码和填零写掩码两者。当合并的向量掩码允许在执行任何操作(由基础操作和扩充操作指定)期间保护目的地中的任何元素集免于更新时,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,当填零向量掩码允许在执行任何操作(由基础操作和扩充操作指定)期间使目的地中的任何元素集填零时,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。由此,写掩码字段1470允许部分向量操作,这包括加载、存储、算术、逻辑等。尽管描述了其中写掩码字段1470的内容选择了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1470的内容间接地标识了要执行的掩码操作)的本发明的实施例,但是替换实施例相反或另外允许掩码写字段1470的内容直接地指定要执行的掩码操作。

[0135] 立即数字段1472—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的意义上是任选的。

[0136] 类字段1468—其内容在不同类的指令之间进行区分。参考图14A-B,该字段的内容在A类和B类指令之间进行选择。在图14A-B中,圆角方形用于指示专用值存在于字段中(例如,在图14A-B中分别用于类字段1468的A类1468A和B类1468B)。

[0137] A类指令模板

[0138] 在A类非存储器访问1405的指令模板的情况下,α字段1452被解释为其内容区分要执行不同扩充操作类型中的哪一种(例如,针对无存储器访问的舍入型操作1410和无存储器访问的数据变换型操作1415的指令模板分别指定舍入1452A.1和数据变换1452A.2)的RS字段1452A,而β字段1454区分要执行指定类型的操作中的哪一种。在无存储器访问1405指令模板中,缩放字段1460、位移字段1462A以及位移缩放字段1462B不存在。

[0139] 无存储器访问的指令模板—完全舍入控制型操作

[0140] 在无存储器访问的完全舍入控制型操作1410的指令模板中，β字段1454被解释为其内容提供静态舍入的舍入控制字段1454A。尽管在本发明的所述实施例中舍入控制字段1454A包括抑制所有浮点异常(SAE)字段1456和舍入操作控制字段1458，但是替换实施例可支持、可将这些概念两者都编码成相同的字段或者只有这些概念/字段中的一个或另一个(例如，可只有舍入操作控制字段1458)。

[0141] SAE字段1456－其内容区分是否停用异常事件报告；当SAE字段1456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。

[0142] 舍入操作控制字段1458－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1458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1450的内容覆盖该寄存器值。

[0143] 无存储器访问的指令模板－数据变换型操作

[0144] 在无存储器访问的数据变换型操作1415的指令模板中，β字段1454被解释为数据变换字段1454B，其内容区分要执行多个数据变换中的哪一个(例如，无数据变换、拌和、广播)。

[0145] 在A类存储器访问1420的指令模板的情况下，α字段1452被解释为驱逐提示字段1452B，其内容区分要使用驱逐提示中的哪一个(在图14A中，为存储器访问时间1425指令模板和存储器访问非时间1430的指令模板分别指定时间1452B.1和非时间1452B.2)、而β字段1454被解释为数据操纵字段1454C，其内容区分要执行大量数据操纵操作(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1420的指令模板包括缩放字段1460、以及任选的位移字段1462A或位移缩放字段1462B。

[0146] 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。

[0147] 存储器访问的指令模板－时间性的

[0148] 时间性的数据是可能很快地重新使用足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。

[0149] 存储器访问的指令模板－非时间性的

[0150] 非时间性的数据是不可能很快地重新使用足以从第一级高速缓存中的高速缓存受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。

[0151] B类指令模板

[0152] 在B类指令模板的情况下，α字段1452被解释为写掩码控制(Z)字段1452C，其内容区分由写掩码字段1470控制的写掩码应当是合并还是填零。

[0153] 在B类非存储器访问1405的指令模板的情况下，β字段1454的一部分被解释为RL字段1457A，其内容区分要执行不同扩充操作类型中的哪一种(例如，针对无存储器访问的写掩码控制部分舍入控制类型操作1412的指令模板和无存储器访问的写掩码控制VSIZE型操作1417的指令模板分别指定舍入1457A.1和向量长度(VSIZE)1457A.2)，而β字段1454的其

余部分区分要执行指定类型的操作中的哪一种。在无存储器访问1405指令模板中，缩放字段1460、位移字段1462A以及位移缩放字段1462B不存在。

[0154] 在无存储器访问的写掩码控制的部分舍入控制型操作1410的指令模板中，β字段1454的其余部分被解释为舍入操作字段1459A，并且停用异常事件报告(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)。

[0155] 舍入操作控制字段1459A—只作为舍入操作控制字段1458，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1459A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1450的内容覆盖该寄存器值。

[0156] 在无存储器访问的写掩码控制VSIZE型操作1417的指令模板中，β字段1454的其余部分被解释为向量长度字段1459B，其内容区分要执行多个数据向量长度中的哪一个(例如，128字节、256字节、或512字节)。

[0157] 在B类存储器访问1420的指令模板的情况下，β字段1454的一部分被解释为广播字段1457B，其内容区分是否要执行广播型数据操纵操作，而β字段1454的其余部分被解释为向量长度字段1459B。存储器访问1420的指令模板包括缩放字段1460、以及任选的位移字段1462A或位移缩放字段1462B。

[0158] 针对通用向量友好指令格式1400，示出完整操作码字段1474包括格式字段1440、基础操作字段1442以及数据元素宽度字段1464。尽管示出了其中完整操作码字段1474包括所有这些字段的一个实施例，但是完整操作码字段1474包括在不支持所有这些字段的实施例中的少于所有的这些字段。完整操作码字段1474提供操作码(opcode)。

[0159] 扩充操作字段1450、数据元素宽度字段1464以及写掩码字段1470允许在每一指令的基础上以通用向量友好指令格式指定这些特征。

[0160] 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。

[0161] 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，期望用于通用计算的高性能通用无序核可仅支持B类，期望主要用于图形和/或科学(吞吐量)计算的核可仅支持A类，并且期望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，在具有分离的图形和通用核的处理器中，图形核中的期望主要用于图形和/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期望用于通用计算的仅支持B类的无序执行和寄存器重命名的高性能通用核。没有单独的图形核的另一处理器可包括支持A类和B类两者的一个或多个通用有序或无序核。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。以高级语言撰写的程序可被输入(例如，及时编译或者统计编译)到各种不同的可执行形式，包括：1)具有用于执行的目标处理器支持的类的指令的形式；或者2)具有使用所有类的指令的不同组合而编写的替换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。

[0162] 示例性专用向量友好指令格式

[0163] 图15A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15A示出在其指定位置、尺寸、解释和字段的次序、以及那些字段中的一些字段的值的意义上是专用的专用向量友好指令格式1500。专用向量友好指令格式1500可用于扩展x86指令集，并且由此一些字段类似于在现有x86指令集及其扩展(例如，AVX)中使用的那些字段或与之相同。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14的字段，来自图15的字段映射到来自图14的字段。

[0164] 应当理解，虽然出于说明的目的在通用向量友好指令格式1400的上下文中，本发明的实施例参考专用向量友好指令格式1500进行了描述，但是本发明不限于专用向量友好指令格式1500，声明的地方除外。例如，通用向量友好指令格式1400构想各种字段的各种可能的尺寸，而专用向量友好指令格式1500被示为具有特定尺寸的字段。作为具体示例，尽管在专用向量友好指令格式1500中数据元素宽度字段1464被示为一位字段，但是本发明不限于此(即，通用向量友好指令格式1400构想数据元素宽度字段1464的其他尺寸)。

[0165] 通用向量友好指令格式1400包括以下列出的按照图15A中示出的顺序的如下字段。

[0166] EVEX前缀(字节0-3)1502—以四字节形式进行编码。

[0167] 格式字段1440(EVEX字节0，位[7:0])—第一字节(EVEX字节0)是格式字段1440，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)。

[0168] 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。

[0169] REX字段1505(EVEX字节1，位[7-5])—由EVEX.R位字段(EVEX字节1，位[7]-R)、EVEX.X位字段(EVEX字节1，位[6]-X)以及(1457BEX字节1，位[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(多个)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。

[0170] REX'字段1410—这是REX'字段1410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；本发明的替换实施例不以反转的格式存储该指示的位以及其他指示的位。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。

[0171] 操作码映射字段1515(EVEX字节1，位[3:0]-mmmm)-其内容对隐含的领先操作码字节(0F、0F 38、或0F 3)进行编码。

[0172] 数据元素宽度字段1464(EVEX字节2，位[7]-W)—由记号EVEX.W表示。EVEX.W用于定义数据类型(32位数据元素或64位数据元素)的粒度(尺寸)。

[0173] EVEX.vvvv 1520(EVEX字节2，位[6:3]-vvvv)—EVEX.vvvv的作用可包括如下：1)EVEX.vvvv对以反转((多个)1补码)的形式指定的第一源寄存器操作数进行编码且对具有

两个或两个以上源操作数的指令有效;2)EVEX.vvvv针对特定向量位移对以(多个)1补码的形式指定的目的地寄存器操作数进行编码;或者3)EVEX.vvvv不对任何操作数进行编码,保留该字段,并且应当包含1111b。由此,EVEX.vvvv字段1520对以反转((多个)1补码)的形式存储的第一源寄存器指定符的4个低阶位进行编码。取决于该指令,额外不同的EVEX位字段用于将指定符尺寸扩展到32个寄存器。

[0174] EVEX.U 1468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。

[0175] 前缀编码字段1525(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,这也具有压缩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前缀格式的SIMD前缀(66H、F2H、F3H)的传统SSE指令,这些传统SIMD前缀被编码成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PLA可执行传统和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作为操作码扩展,但是为了一致性,特定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。替换实施例可重新设计PLA以支持2位SIMD前缀编码,并且由此不需要扩展。

[0176] α字段1452(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;还被示为具有α)—如先前所述的,该字段是上下文特定的。

[0177] β字段1454(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,还被示为具有βββ)—如先前所述的,该字段是上下文特定的。

[0178] REX'字段1410—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寄存器进行编码的EVEX.R'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。

[0179] 写掩码字段1470(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(这可以各种方式实现,包括使用硬连线到所有的写掩码或者旁路掩码硬件的硬件来实现)的特别行为。

[0180] 实操作码字段1530(字节4)还被称为操作码字节。操作码的一部分在该字段中被指定。

[0181] MOD R/M字段1540(字节5)包括MOD字段1542、Reg字段1544、以及R/M字段1546。如先前所述的,MOD字段1542的内容将存储器访问和非存储器访问操作区分开。Reg字段1544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1546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。

[0182] 缩放、索引、基址(SIB)字节(字节6)—如先前所述的,缩放字段1450的内容用于存储器地址生成。SIB.xxx 1554和SIB.bbb 1556—先前已经针对寄存器索引Xxxx和Bbbb提及了这些字段的内容。

[0183] 位移字段1462A(字节7-10)—当MOD字段1542包含10时,字节7-10是位移字段1462A,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。

[0184] 位移因数字段1462B(字节7)—当MOD字段1542包含01时,字节7是位移因数字段1462B。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,因此它可只在-128和127字节偏移量之间寻址;在64字节高速缓存线的方面,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1462B是disp8的重新解释;当使用位移因数字段1462B时,通过位移因数字段的内容乘以存储器操作数访问的尺寸(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(用于位移但具有大得多的范围的单一字节)。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1462B替代传统x86指令集8位位移。由此,位移因数字段1462B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,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尺寸按比例缩放位移量以获得字节式地址偏移量)。

[0185] 立即数字段1472如先前所述地操作。

[0186] 完整操作码字段

[0187] 图15B是示出根据本发明的实施例的构成完整操作码字段1474的具有专用向量友好指令格式1500的字段的框图。具体地,完整操作码字段1474包括格式字段1440、基础操作字段1442、以及数据元素宽度(W)字段1464。基础操作字段1442包括前缀编码字段1525、操作码映射字段1515以及实操作码字段1530。

[0188] 寄存器索引字段

[0189] 图15C是示出根据本发明的实施例的构成寄存器索引字段1444的具有专用向量友好指令格式1500的字段的框图。具体地,寄存器索引字段1444包括REX字段1505、REX'字段1510、MODR/M.reg字段1544、MODR/M.r/m字段1546、VVVV字段1520、xxx字段1554以及bbb字段1556。

[0190] 扩充操作字段

[0191] 图15D是示出根据本发明的一个实施例的构成扩充(augmentation)操作字段1450的具有专用向量友好指令格式1500的字段的框图。当类(U)字段1468包含0时,它表明EVEX.U0(A类1468A);当它包含1时,它表明EVEX.U1(B类1468B)。当U=0且MOD字段1542包含11(表明无存储器访问操作)时,α字段1452(EVEX字节3,位[7]-EH)被解释为rs字段1452A。当rs字段1452A包含1(舍入1452A.1)时,β字段1454(EVEX字节3,位[6:4]-SSS)被解释为舍入控制字段1454A。舍入控制字段1454A包括一位SAE字段1456和两位舍入操作字段1458。当rs字段1452A包含0(数据变换1452A.2)时,β字段1454(EVEX字节3,位[6:4]-SSS)被解释为三位数据变换字段1454B。当U=0且MOD字段1542包含00、01或10(表达存储器访问操作)时,α字段1452(EVEX字节3,位[7]-EH)被解释为驱逐提示(EH)字段1452B且β字段1454(EVEX字节3,位[6:4]-SSS)被解释为三位数据操纵字段1454C。

[0192] 当U=1时,α字段1452(EVEX字节3,位[7]-EH)被解释为写掩码控制(Z)字段1452C。

当U＝1且MOD字段1542包含11(表明无存储器访问操作)时,β字段1454的一部分(EVEX字节3,位[4]-S_0)被解释为RL字段1457A;当它包含1(舍入1457A.1)时,β字段1454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1459A,而当RL字段1457A包含0(VSIZE 1457.A2)时,β字段1454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1459B(EVEX字节3,位[6-5]-L_{1-0})。当U＝1且MOD字段1542包含00、01或10(表明存储器访问操作)时,β字段1454(EVEX字节3,位[6:4]-SSS)被解释为向量长度字段1459B(EVEX字节3,位[6-5]-L_{1-0})和广播字段1457B(EVEX字节3,位[4]-B)。

[0193] 示例性编码成具体的向量友好指令格式

[0194] 示例性寄存器架构

[0195] 图16是根据本发明的一个实施例的寄存器架构1600的框图。在所示出的实施例中,有32个512位宽的向量寄存器1610;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1500对这些覆盖的寄存器组操作,如在以下表格中所示的。

[0196]

可调节向量长度	类	操作	寄存器
不包括向量长度字段1459B的指令模板	A(图14A;U=0)	1410,1415,1425,1430	zmm寄存器(向量长度是64字节)
	B(附图14B;U=1)	1412	zmm寄存器(向量长度是64字节)
包括向量长度字段1459B的指令模板	B(附图14B;U=1)	1417,1427	zmm、ymm、或xmm寄存器(向量长度是64字节、32字节、或16字节),取决于向量长度字段1459B

[0197] 换句话说,向量长度字段1459B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1459B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1500的B类指令模板对打包或标量单/双精度浮点数据以及打包或标量整数数据操作。标量操作是在zmm/ymm/xmm寄存器中的最低阶数据元素位置上执行的操作;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填零。

[0198] 写掩码寄存器1615－在所示的实施例中,存在8个写掩码寄存器(k0至k7),每一写掩码寄存器的尺寸是64位。在替换实施例中,写掩码寄存器1615的尺寸是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。

[0199] 通用寄存器1625——在所示出的实施例中，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寻址存储器操作数。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。

[0200] 标量浮点堆栈寄存器组(x87堆栈)1645，在其上面使用了别名MMX打包整数平坦寄存器组1650——在所示出的实施例中，x87堆栈是用于使用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堆栈；而使用MMX寄存器来对64位打包整数数据执行操作，以及为在MMX和XMM寄存器之间执行的某些操作保存操作数。

[0201] 本发明的替换实施例可以使用较宽的或较窄的寄存器。另外，本发明的替换实施例可以使用更多、更少或不同的寄存器组和寄存器。

[0202] 示例性核架构、处理器和计算机架构

[0203] 处理器核可以用出于不同目的的不同方式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无序核；3)主要旨在用于图形和/或科学(吞吐量)计算的专用核。不同处理器的实现可包括：包括预期用于通用计算的一个或多个通用有序核和/或预期用于通用计算的一个或多个通用无序核的CPU；以及2)包括主要预期用于图形和/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。

[0204] 示例性核架构

[0205] 有序和无序核框图

[0206] 图17A是示出根据本发明的实施例的示例性有序流水线和示例性寄存器重命名、无序发布/执行流水线二者的框图。图17B是示出根据本发明的实施例的有序架构核的示例性实施例以及包括在处理器中的示例性寄存器重命名的无序发布/执行架构核两者的方框图。图17A-B中的实线框示出有序流水线和有序核，而任选增加的虚线框示出寄存器重命名的无序发布/执行流水线和核。考虑到有序方面是无序方面的子集，将描述无序方面。

[0207] 在图17A中，处理器流水线1700包括获取(fetch)级1702、长度解码级1704、解码级1706、分配级1708、重命名级1710、调度(也称为分派或发布)级1712、寄存器读取/存储器读取级1714、执行级1716、写回/存储器写入级1718、异常处理级1722和提交级1724。

[0208] 图17B示出处理器核1790，该核1790包括耦合到执行引擎单元1750的前端单元1730，并且两者耦合到存储器单元1770。核1790可以是精简指令集合计算(RISC)核、复杂指令集合计算(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1790可以是专用核，诸如例如网络或通信核、压缩引擎、协处理器核、通用计算图形处理器单元(GPGPU)核、图形核等等。

[0209] 前端单元1730包括耦合到指令高速缓存单元1734的分支预测单元1732，该指令高

速缓存单元1734被耦合到指令转换后备缓冲器(TLB)1736，该指令转换后备缓冲器1736被耦合到指令获取单元1738，指令获取单元1738被耦合到解码单元1740。解码单元1740(或解码器)可解码指令，并生成从原始指令解码出的、或以其他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、其他指令、或其他控制信号作为输出。解码单元1740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。在一个实施例中，核1790包括存储(例如，在解码单元1740中或否则在前端单元1730内的)特定宏指令的微代码的微代码ROM或其他介质。解码单元1740耦合至执行引擎单元1750中的重命名/分配器单元1752。

[0210] 执行引擎单元1750包括重命名/分配器单元1752，该重命名/分配器单元1752耦合至引退单元1754和一个或多个调度器单元(多个)1756的集合。调度器单元1756表示任何数目的不同调度器，包括预留站(reservations stations)、中央指令窗等。调度器单元1756被耦合到物理寄存器组单元1758。每个物理寄存器组单元1758表示一个或多个物理寄存器组，其中不同的物理寄存器组存储一种或多种不同的数据类型，诸如标量整数、标量浮点、打包整数、打包浮点、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，物理寄存器组单元1758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。物理寄存器组单元1758与引退单元1754重叠以示出可以用来实现寄存器重命名和无序执行的各种方式(例如，使用记录器缓冲器和引退寄存器组；使用将来的文件、历史缓冲器和引退寄存器组；使用寄存器映射和寄存器池等等)。引退单元1754和物理寄存器组单元1758被耦合到执行群集1760。执行群集1760包括一个或多个执行单元1762的集合和一个或多个存储器访问单元1764的集合。执行单元1762可以执行各种操作(例如，移位、加法、减法、乘法)，以及对各种类型的数据(例如，标量浮点、打包整数、打包浮点、向量整数、向量浮点)执行。尽管某些实施例可以包括专用于特定功能或功能集合的多个执行单元，但其他实施例可包括全部执行所有函数的仅一个执行单元或多个执行单元。调度器单元1756、物理寄存器组单元1758和执行群集1760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(例如，标量整数流水线、标量浮点/打包整数/打包浮点/向量整数/向量浮点流水线，和/或各自具有其自己的调度器单元、物理寄存器组单元和/或执行群集的存储器访问流水线——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1764的某些实施例)创建分开的流水线。还应当理解，在分开的流水线被使用的情况下，这些流水线中的一个或多个可以为无序发布/执行，并且其余流水线可以为有序发布/执行。

[0211] 存储器访问单元1764的集合被耦合到存储器单元1770，该存储器单元1770包括耦合到数据高速缓存单元1774的数据TLB单元1772，其中该数据高速缓存单元1774耦合到二级(L2)高速缓存单元1776。在一个示例性实施例中，存储器访问单元1764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1770中的数据TLB单元1772。指令高速缓存单元1734还耦合到存储器单元1770中的二级(L2)高速缓存单元1776。L2高速缓存单元1776被耦合到一个或多个其他级的高速缓存，并最终耦合到主存储器。

[0212] 作为示例，示例性寄存器重命名的、无序发布/执行核架构可以如下实现流水线1700：1)指令获取1738执行取指和长度解码级1702和1704；2)解码单元1740执行解码级

1706;3)重命名/分配器单元1752执行分配级1708和重命名级1710;4)调度器单元1756执行调度级1712;5)物理寄存器组单元1758和存储器单元1770执行寄存器读取/存储器读取级1714;执行群集1760执行执行级1716;6)存储器单元1770和物理寄存器组单元1758执行写回/存储器写入级1718;7)各单元可牵涉到异常处理级1722;以及8)引退单元1754和物理寄存器组单元1758执行提交级1724。

[0213] 核1790可支持一个或多个指令集合(例如,x86指令集合(具有与较新版本一起添加的某些扩展);加利福尼亚州桑尼维尔市的MIPS技术公司的MIPS指令集合;加利福尼州桑尼维尔市的ARM控股的ARM指令集合(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1790包括用于支持打包数据指令集扩展(例如,AVX1、AVX2和/或先前描述的一些形式的一般向量友好指令格式(U=0和/或U=1))的逻辑,从而允许很多多媒体应用使用的操作能够使用打包数据来执行。

[0214] 应当理解,核可支持多线程化(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化,此各种方式包括时分多线程化、同步多线程化(其中单个物理核为物理核正同步多线程化的各线程中的每一个线程提供逻辑核)、或其组合(例如,时分取指和解码以及此后诸如用Intel®超线程化技术来同步多线程化)。

[0215] 尽管在无序执行的上下文中描述了寄存器重命名,但应当理解,可以在有序架构中使用寄存器重命名。尽管所例示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1734/1774以及共享L2高速缓存单元1776,但替换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,诸如例如一级(L1)内部高速缓存或多个级别的内部缓存。在某些实施例中,该系统可包括内部高速缓存和在核和/或处理器外部的外部高速缓存的组合。或者,所有高速缓存都可以在核和/或处理器的外部。

[0216] 具体的示例性有序核架构

[0217] 图18A-B示出了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)。这些逻辑块通过高带宽的互连网络(例如,环形网络)与某些固定的功能逻辑、存储器I/O接口和其它必要的I/O逻辑通信,这依赖于应用。

[0218] 图18A是根据本发明的实施例的连接到片上互联网络1802且具有第二级(L2)高速缓存1804的本地子集的单一处理器内核的方框图。在一个实施例中,指令解码器1800支持具有打包数据指令集合扩展的x86指令集。L1高速缓存1806允许对高速缓存存储器的低等待时间访问进入标量和向量单元。尽管在一个实施例中(为了简化设计),标量单元1808和向量单元1810使用分开的寄存器集合(分别为标量寄存器1812和向量寄存器1814),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一级(L1)高速缓存1806读回,但是本发明的替换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组之间传输而无需被写入和读回的通信路径)。

[0219] L2高速缓存的本地子集1804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1804的本地子集的直接访问路径。被处理器核读出的数据被存储在其L2高速缓存子集1804中,并且可以被快速访问,该访问与其他处理器核访问它们自己的本地L2高速缓存子集并行。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1804中,并在必

要的情况下从其它子集清除。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。

[0220] 图18B是根据本发明的实施例的图18A中的处理器核的一部分的展开图。图18B包括L1高速缓存1804的L1数据高速缓存1806A部分、以及关于向量单元1810和向量寄存器1814的更多细节。具体地说，向量单元1810是16宽向量处理单元(VPU)(见16宽ALU 1828)，该单元执行整数、单精度浮点以及双精度浮点指令中的一个或多个。该VPU支持通过拌和单元1820混合寄存器输入、通过数值转换单元1822A-B进行数值转换，以及通过复制单元1824进行对存储器输入的复制。写掩码寄存器1826允许断言(predicating)所得的向量写入。

[0221] 具有集成存储器控制器和图形器件的处理器

[0222] 图19是根据本发明的实施例的处理器1900的框图，该处理器可具有一个以上的核，可具有集成的存储器控制器，且可具有集成的图形。图19的实线框示出了处理器1900，处理器1900具有单个核心1902A、系统代理1910、一组一个或多个总线控制器单元1916而可选附加的虚线框示出了替换式的处理器1900，具有多个核心1902A-N、系统代理单元1910中的一组一个或多个集成存储器控制器单元1914以及专用逻辑1908。

[0223] 因此，处理器1900的不同实现可包括：1)CPU，其中专用逻辑1908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1902A-N是一个或多个通用核(例如，通用的有序核、通用的无序核、这两者的组合)；2)协处理器，其中核1902A-N是主要旨在用于图形和/或科学(吞吐量)的大量专用核；以及3)协处理器，其中核1902A-N是大量通用有序核。因此，处理器1900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、压缩引擎、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1900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等的多个加工技术中的任何一个技术将其实现在一个或多个衬底上。

[0224] 存储器层次结构包括在各核内的一个或多个级别的高速缓存、一组或一个或多个共享高速缓存单元1906、以及耦合至集成存储器控制器单元1914的集合的外部存储器(未示出)。该共享高速缓存单元1906的集合可以包括一个或多个中间级高速缓存，诸如二级(L2)、三级(L3)、四级(L4)或其他级别的高速缓存、末级高速缓存(LLC)、和/或其组合。尽管在一个实施例中，基于环的互连单元1912将集成图形逻辑1908、共享高速缓存单元1906的集合以及系统代理单元1910/集成存储器控制器单元1914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，在一个或多个高速缓存单元1906与核1902-A-N之间维持相干性。

[0225] 在某些实施例中，核1902A-N中的一个或多个核能够多线程化。系统代理1910包括协调和操作核1902A-N的那些组件。系统代理单元1910可包括例如功率控制单元(PCU)和显示单元。PCU可以是或包括调整核1902A-N和集成图形逻辑1908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外部连接的显示器。

[0226] 核1902A-N在架构指令集合方面可以是同构的或异构的；即，这些核1902A-N中的两个或更多个核可以能够执行相同的指令集合，而其他核可以能够执行该指令集合的仅仅子集或不同的指令集合。

[0227] 示例性计算机架构

[0228] 图20-23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、网络中枢、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他电子设备的其他系统设计和配置也是合适的。一般来说,能够含有本文中所公开的处理器和/或其它执行逻辑的大量系统和电子设备一般都是合适的。

[0229] 现在参考图20,所示出的是根据本发明一实施例的系统2000的框图。系统2000可以包括一个或多个处理器2010、2015,这些处理器耦合到控制器中枢2020。在一个实施例中,控制器中枢2020包括图形存储器控制器中枢(GMCH)2090和输入/输出中枢(IOH)2050(其可以在分开的芯片上);GMCH 2090包括存储器和图形控制器,存储器2040和协处理器2045耦合到该图形控制器;IOH 2050将输入/输出(I/O)设备2060耦合到GMCH 2090。或者,存储器和图形控制中的一个或两个集成在处理器内(如本文所述),存储器2040和协处理器2045直接耦合到处理器2010和在单个芯片中具有IOH2050的控制器中枢2020。

[0230] 附加处理器2015的可选性质用虚线表示在图20中。每一处理器2010、2015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1900的某一版本。

[0231] 存储器2040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2020经由诸如前侧总线(FSB)之类的多分支总线(multi-drop bus)、诸如快速通道互连(QPI)之类的点对点接口、或者类似的连接2095与处理器2010、2015进行通信。

[0232] 在一个实施例中,协处理器2045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。在一个实施例中,控制器中枢2020可以包括集成图形加速器。

[0233] 按照包括架构、微架构、热、功耗特征等等优点的度量谱,物理资源2010、2015之间存在各种差别。

[0234] 在一个实施例中,处理器2010执行控制一般类型的数据处理操作的指令。嵌入在这些指令中的可以是协处理器指令。处理器2010将这些协处理器指令识别为应当由附连的协处理器2045执行的类型。因此,处理器2010在协处理器总线或者其他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2045。协处理器2045接受并执行所接收的协处理器指令。

[0235] 现在参照图21,所示出的是根据本发明一个实施例的更具体的第一示例性系统2100的框图。如图21所示,多处理器系统2100是点对点互连系统,且包括经由点对点互连2150耦合的第一处理器2170和第二处理器2180。处理器2170和2180中的每一个都可以是处理器1900的某一版本。在本发明的一个实施例中,处理器2170和2180分别是处理器2010和2015,而协处理器2138是协处理器2045。在另一实施例中,处理器2170和2180分别是处理器2010和协处理器2045。

[0236] 处理器2170和2180被示为分别包括集成存储器控制器(IMC)单元2172和2182。处理器2170还包括作为其总线控制器单元的一部分的点对点(P-P)接口2176和2178;类似地,第二处理器2180包括点对点接口2186和2188。处理器2170、2180可以使用点对点(P-P)接口电路2178、2188经由P-P接口2150来交换信息。如图21所示,IMC 2172和2182将处理器耦合

到相应的存储器，即存储器2132和存储器2134，这些存储器可以是本地附连到相应处理器的主存储器的部分。

[0237] 处理器2170、2180可各自经由使用点对点接口电路2176、2194、2186、2198的各个P-P接口2152、2154与芯片组2190交换信息。芯片组2190可以可选地经由高性能接口2139与协处理器2138交换信息。在一个实施例中，协处理器2138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。

[0238] 共享高速缓存(未示出)可以被包括在两个处理的任一个之内或被包括两个处理器外部但仍经由P-P互连与这些处理器连接，从而如果将某处理器置于低功率模式时，可将任一处理器或两个处理器的本地高速缓存信息存储在该共享高速缓存中。

[0239] 芯片组2190可经由接口2196耦合至第一总线2116。在一个实施例中，第一总线2116可以是外围部件互连(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。

[0240] 如图21所示，各种I/O设备2114可以连同总线桥2118耦合到第一总线2116，总线桥2118将第一总线2116耦合至第二总线2120。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(诸如例如图形加速器或数字信号处理器(DSP)单元)、现场可编程门阵列或任何其他处理器的一个或多个附加处理器2115被耦合到第一总线2116。在一个实施例中，第二总线2120可以是低引脚计数(LPC)总线。各种设备可以被耦合至第二总线2120，在一个实施例中这些设备包括例如键盘/鼠标2122、通信设备2127以及诸如可包括指令/代码和数据2130的盘驱动器或其它海量存储设备的存储单元2128。此外，音频I/O 2124可以被耦合至第二总线2120。注意，其它架构是可能的。例如，代替图21的点对点架构，系统可实现多分支总线或者其他此类架构。

[0241] 现在参照图22，所示出的是根据本发明实施例的更具体的第二示例性系统1200的框图。图21和22中的类似元件使用类似附图标记，且在图22中省略了图21的某些方面以避免混淆图22的其它方面。

[0242] 图22示出处理器2170、2180可分别包括集成存储器和I/O控制逻辑(“CL”)2172和2182。因此，CL 2172、2182包括集成存储器控制器单元并包括I/O控制逻辑。图22示出：不仅存储器2132、2134耦合至CL 2172、2182，I/O设备2214也耦合至控制逻辑2172、2182。传统I/O设备2215被耦合至芯片组2190。

[0243] 现在参照图23，所示出的是根据本发明一个实施例的SoC 2300的框图。图19中的类似元件具有相似的附图标记。另外，虚线框是更先进的SoC的可选特征。在图23中，互连单元2302被耦合至：应用处理器2310，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1906；系统代理单元1910；总线控制器单元1916；集成存储器控制器单元1914；一组或一个或多个协处理器2320，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2330；直接存储器存取(DMA)单元2332；以及用于耦合至一个或多个外部显示器的显示单元2340。在一个实施例中，协处理器2320包括专用处理器，诸如例如网络或通信处理器、压缩引擎、GPGPU、高吞吐量MIC处理器、或嵌入式处理器等等。

[0244] 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。

[0245] 诸如图21所示的代码2130之类的程序代码可应用于输入指令,以执行本文中所描述的功能并生成输出信息。输出信息可以按已知方式被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。

[0246] 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。程序代码也可以在需要的情况下用汇编语言或机器语言来实现。事实上,本文中描述的机制不仅限于任何特定编程语言的范围。在任一情形下,语言可以是编译语言或解译语言。

[0247] 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代表性指令来实现,该指令表示处理器中的各种逻辑,该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。被称为"IP核"的这些表示可以被存储在有形的机器可读介质上,并被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。

[0248] 这样的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瞬态、有形配置,其包括存储介质,诸如硬盘;任何其它类型的盘,包括软盘、光盘、紧致盘只读存储器(CD-ROM)、紧致盘可重写(CD-RW)的以及磁光盘;半导体器件,例如只读存储器(ROM)、诸如动态随机存取存储器(DRAM)和静态随机存取存储器(SRAM)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它类型的介质。

[0249] 因此,本发明的各实施例还包括非瞬态、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/或系统特性。这些实施例也被称为程序产品。

[0250] 仿真(包括二进制变换、代码变形等)

[0251] 在某些情况下,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(morph)、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部分在处理器外。

[0252] 图24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24以高级语言2402示出了程序,该程序可使用x86编译器2404来编译以生成x86二进制代码2406,该二进制代码可天然地由具有至少一个x86指令集核心的处理器2416来执行。具有至少一个x86指令集核的处理器2416表示任何处理器,这些处理器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:1)英特尔x86指令集核的指令集的本质部分(substantial portion),或2)目标旨在在具有至少一个x86指令集核的英特尔处理器上运行的应用或其它程序的对象代码版本,以便取得与具有至少一个x86指令集核的英特尔处理器基本相同的结果。x86编译器2404表示

用于生成x86二进制代码2406(例如,对象代码)的编译器,该二进制代码2406可通过或不通过附加的可链接处理在具有至少一个x86指令集核的处理器2416上执行。类似地,图24以高级语言2402示出了程序,该程序可使用替换指令集编译器2408来编译以生成替换指令集二级制代码2410,替换指令集二级制代码2410可由不具有至少一个x86指令集核的处理器2414(诸如,具有可执行加利福尼亚州桑尼威尔的MIPS技术公司的MIPS指令集的处理器和/或执行加利福尼亚州桑尼威尔的ARM控股公司的ARM指令集的处理器)来天然地执行。指令转换器2412被用来将x86二进制代码2406转换成可以由不具有x86指令集核的处理器2414原生执行的代码。该经转换的代码不大可能与替换性指令集二进制代码2410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一般操作并由来自替换性指令集的指令构成。因此,指令转换器2412表示:通过仿真、模拟或任何其它过程来允许不具有x86指令集处理器或核的处理器或其它电子设备得以执行x86二进制代码2406的软件、固件、硬件或其组合。

[0253] 尽管附图中的流程图示出本发明的某些实施例的特定操作顺序,应该理解该顺序是示例性的(例如,可选实施例可按不同顺序执行操作、组合某些操作、使某些操作重叠等)。

[0254] 在以上描述中,为解释起见,阐明了众多具体细节以提供对本发明的实施例的透彻理解。然而,将对本领域技术人员明显的是,没有这些具体细节中的一些也可实践一个或多个其他实施例。提供所描述的具体实施例不是为了限制本发明而是为了说明本发明的实施例。本发明的范围不是由所提供的具体示例确定,而是仅由所附权利要求确定。

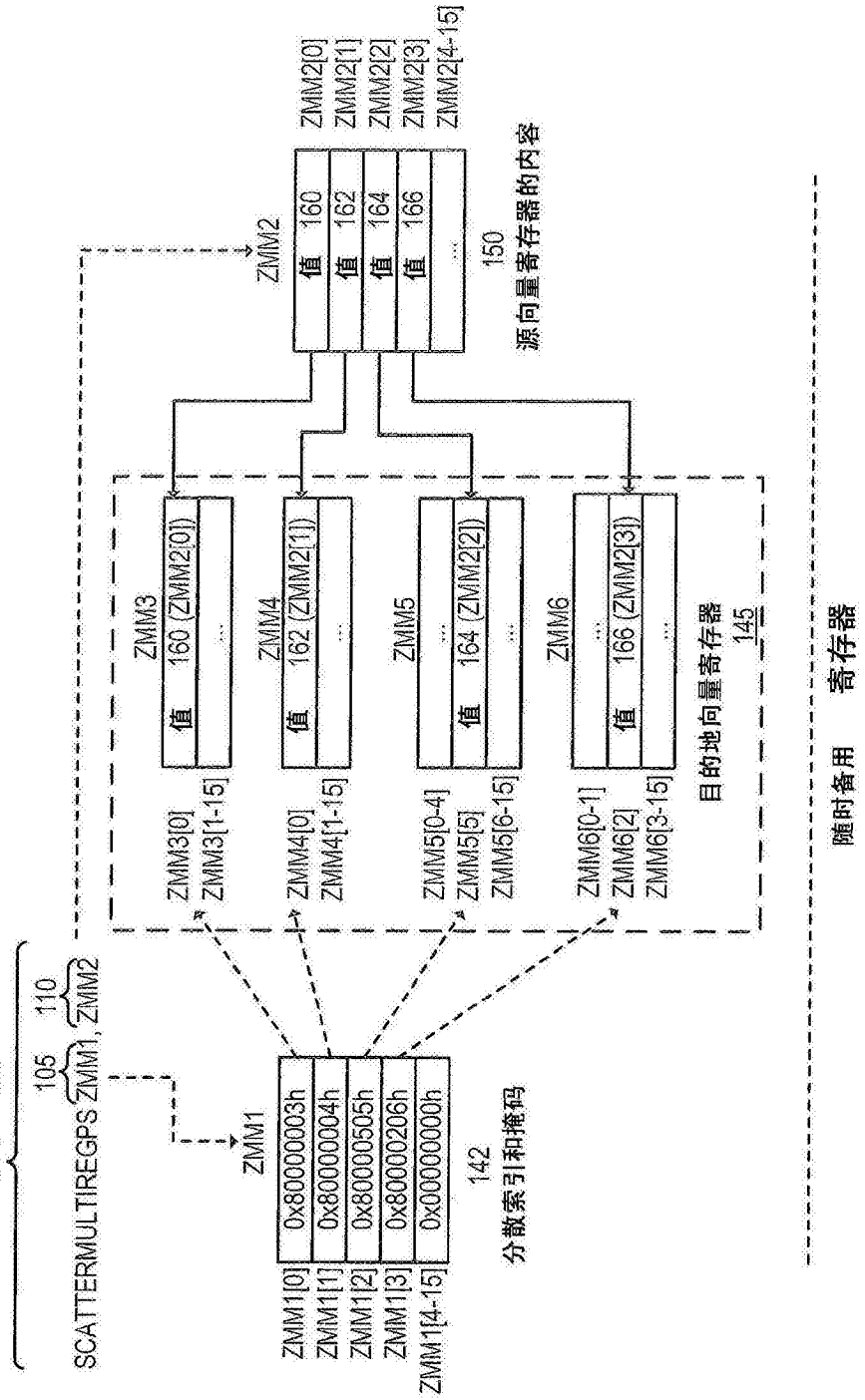

指令 100
SCATTERMULTIREGPS ZMM1, ZMM2
105
110
ZMM1
ZMM2
ZMM1[0] 0x80000003h
ZMM1[1] 0x80000004h
ZMM1[2] 0x80000505h
ZMM1[3] 0x80000206h
ZMM1[4-15] 0x00000000h
142
分散索引和掩码
ZMM2[0] 值 160
ZMM2[1] 值 162
ZMM2[2] 值 164
ZMM2[3] 值 166
ZMM2[4-15] ...
150
源向量寄存器的内容
ZMM3 160 (ZMM2[0]) ...
ZMM4 162 (ZMM2[1]) ...
ZMM5 ... 164 (ZMM2[2]) ...
ZMM6 ... 166 (ZMM2[3]) ...
ZMM3[0] ZMM3[1-15]
ZMM4[0] ZMM4[1-15]
ZMM5[0-4] ZMM5[5] ZMM5[6-15]
ZMM6[0-1] ZMM6[2] ZMM6[3-15]
目的地向量寄存器
145
随时备用 寄存器

图1

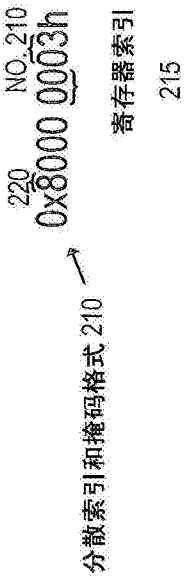


图2

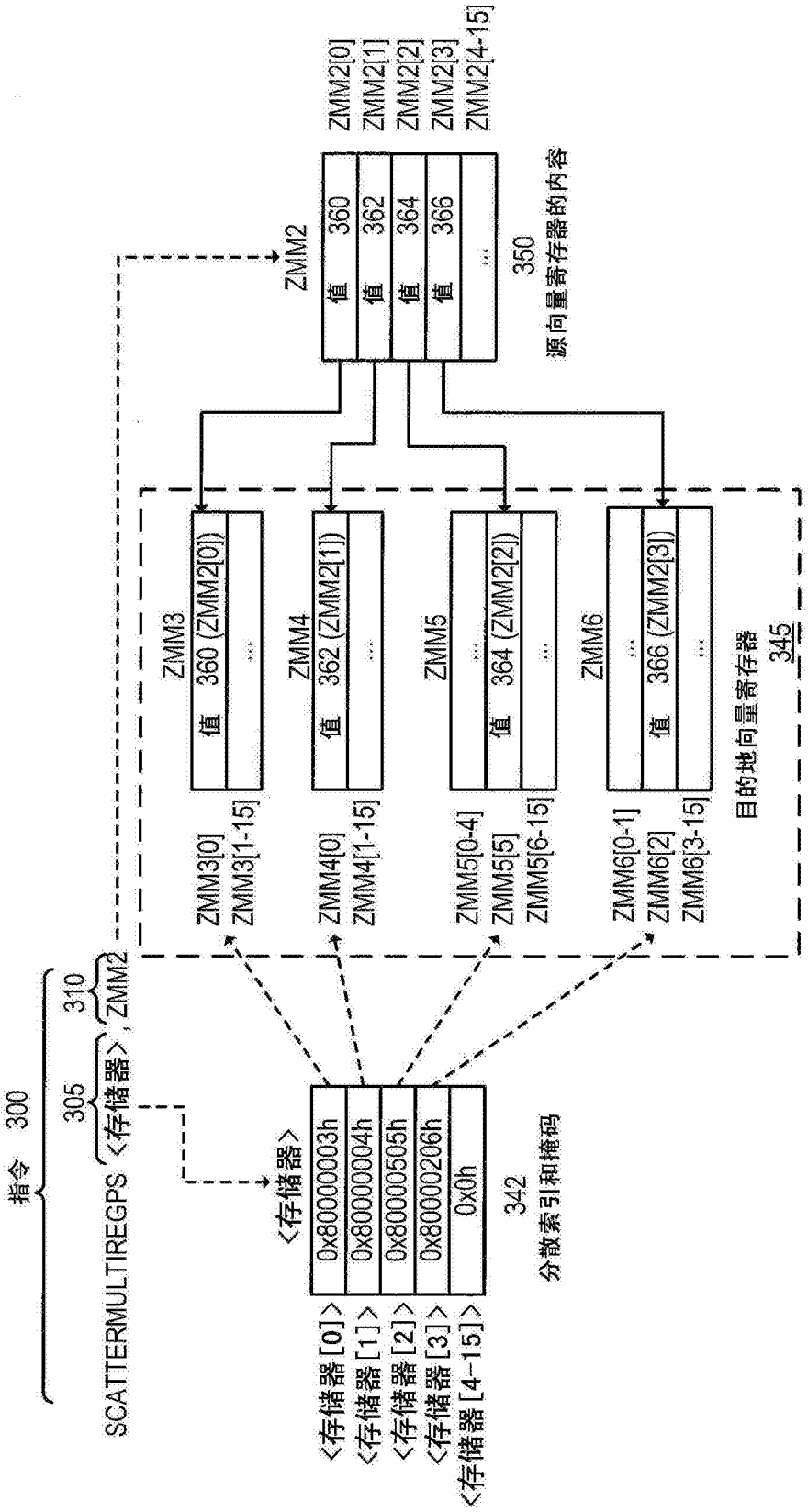

指令 300
SCATTERMULTIREGPS <存储器>, ZMM2
305
310
<存储器>
0x80000003h
0x80000004h
0x80000505h
0x80000206h
0x0h
342
分散索引和掩码
<存储器[0]>
<存储器[1]>
<存储器[2]>
<存储器[3]>
<存储器[4-15]>
ZMM2
值 360
值 362
值 364
值 366
350
源向量寄存器的内容
ZMM2[0]
ZMM2[1]
ZMM2[2]
ZMM2[3]
ZMM2[4-15]
ZMM3
值 360 (ZMM2[0])
ZMM4
值 362 (ZMM2[1])
ZMM5
值 364 (ZMM2[2])
ZMM6
值 366 (ZMM2[3])
目的地向量寄存器
345
ZMM3[0]
ZMM3[1-15]
ZMM4[0]
ZMM4[1-15]
ZMM5[0-4]
ZMM5[5]
ZMM5[6-15]
ZMM6[0-1]
ZMM6[2]
ZMM6[3-15]

图3

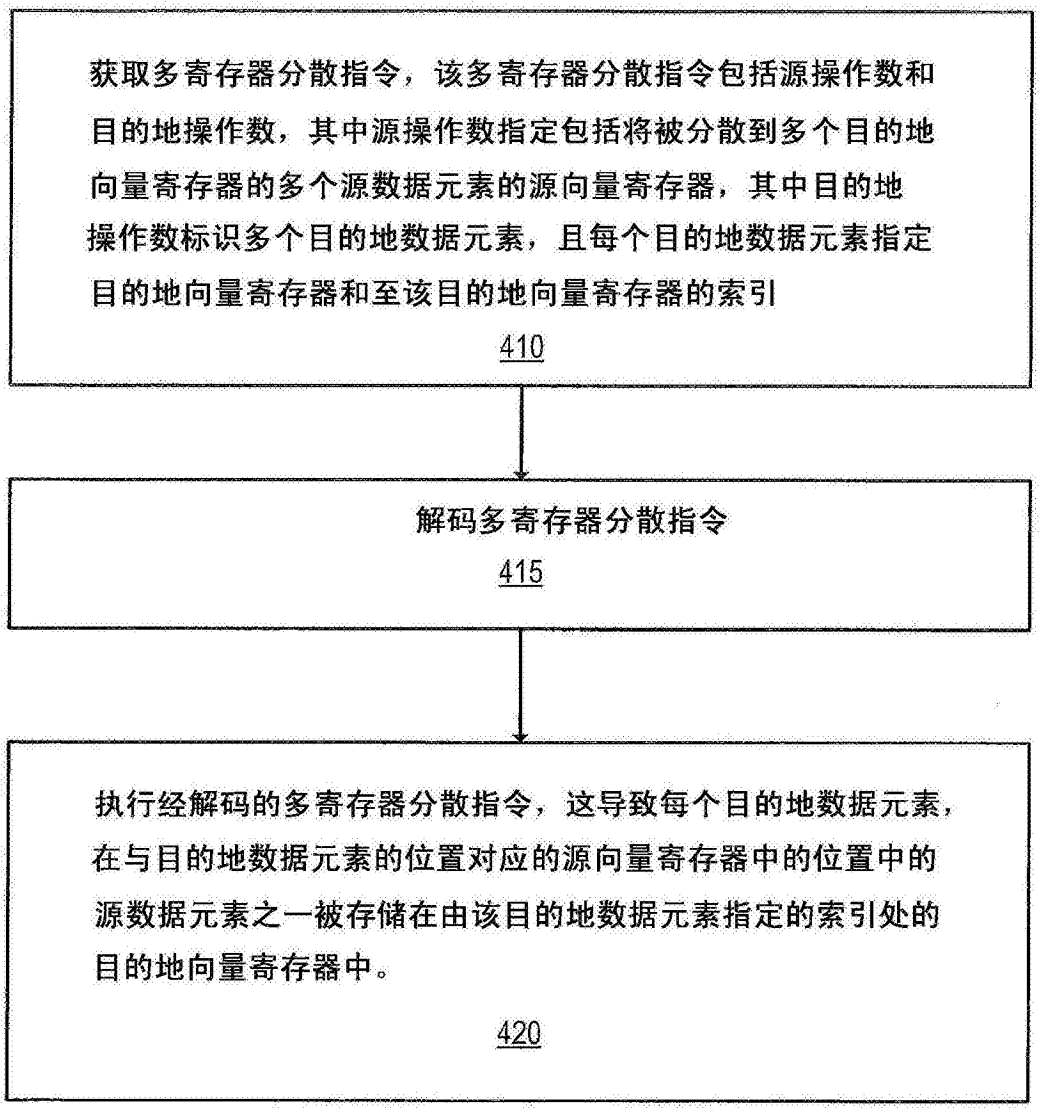

获取多寄存器分散指令，该多寄存器分散指令包括源操作数和目的地操作数，其中源操作数指定包括将被分散到多个目的地向量寄存器的多个源数据元素的源向量寄存器，其中目的地操作数标识多个目的地数据元素，且每个目的地数据元素指定目的地向量寄存器和至该目的地向量寄存器的索引
410
解码多寄存器分散指令
415
执行经解码的多寄存器分散指令，这导致每个目的地数据元素，在与目的地数据元素的位置对应的源向量寄存器中的位置中的源数据元素之一被存储在由该目的地数据元素指定的索引处的目的地向量寄存器中。
420

图4

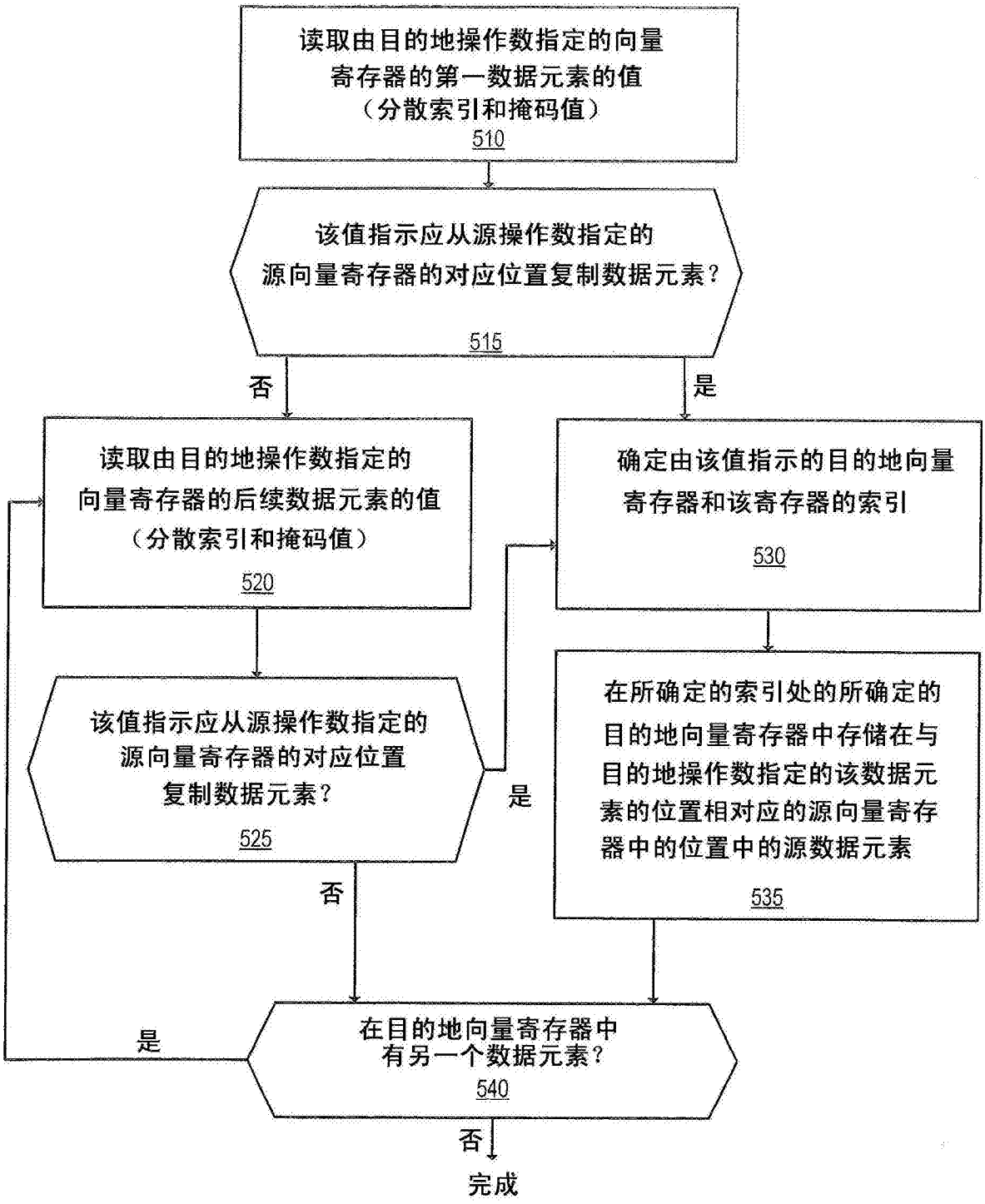

读取由目的地操作数指定的向量寄存器的第一数据元素的值（分散索引和掩码值）
510
该值指示应从源操作数指定的源向量寄存器的对应位置复制数据元素？
515
否
是
读取由目的地操作数指定的向量寄存器的后续数据元素的值（分散索引和掩码值）
520
确定由该值指示的目的地向量寄存器和该寄存器的索引
530
该值指示应从源操作数指定的源向量寄存器的对应位置复制数据元素？
525
是
否
在所确定的索引处的所确定的目的地向量寄存器中存储在与目的地操作数指定的该数据元素的位置相对应的源向量寄存器中的位置中的源数据元素
535
在目的地向量寄存器中有另一个数据元素？
540
是
否
完成

图5

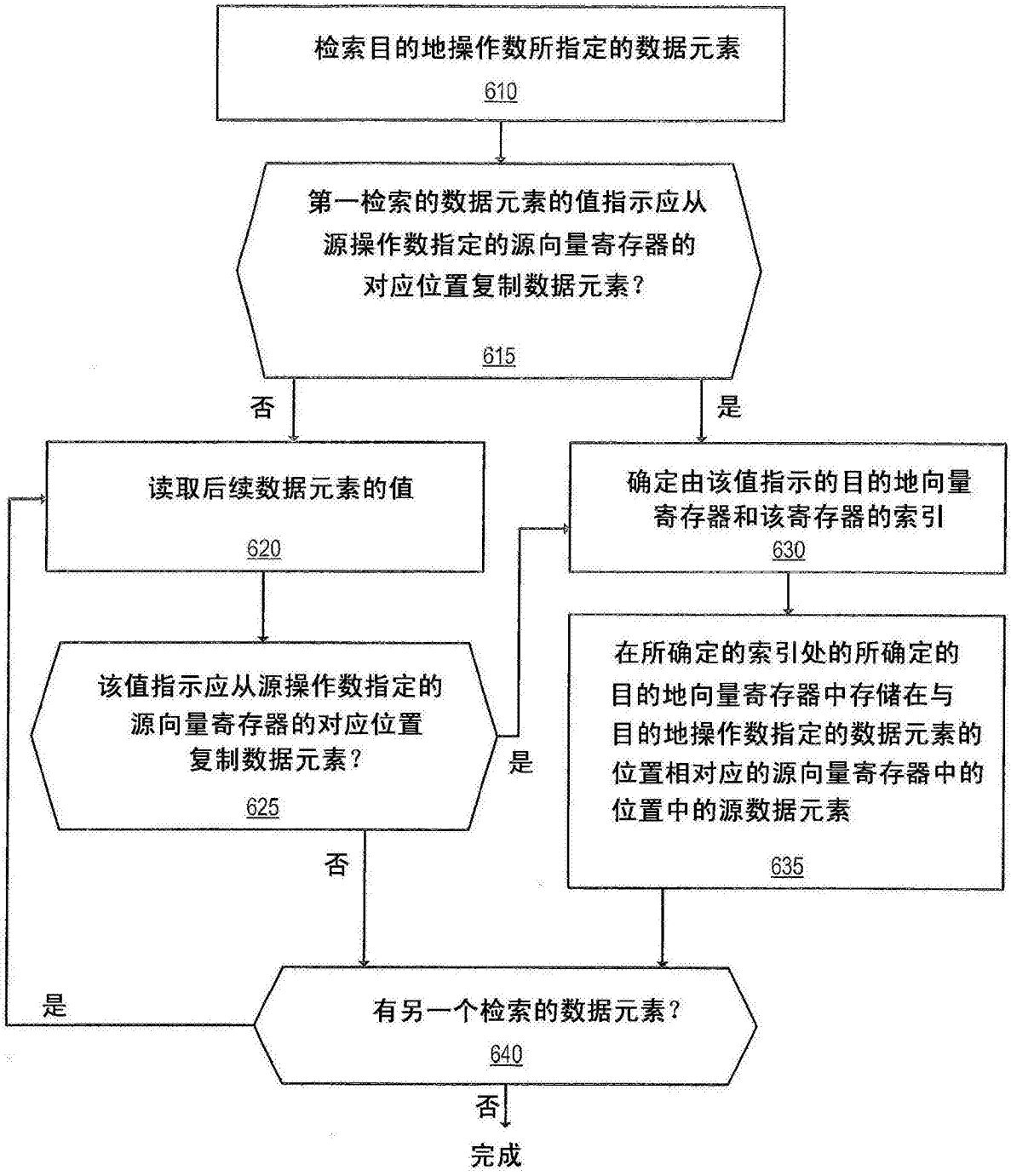

检索目的地操作数所指定的数据元素
610
第一检索的数据元素的值指示应从源操作数指定的源向量寄存器的对应位置复制数据元素？
615
否
是
读取后续数据元素的值
620
确定由该值指示的目的地向量寄存器和该寄存器的索引
630
该值指示应从源操作数指定的源向量寄存器的对应位置复制数据元素？
625
是
在所确定的索引处的所确定的目的地向量寄存器中存储在与目的地操作数指定的数据元素的位置相对应的源向量寄存器中的位置中的源数据元素
635
否
是
有另一个检索的数据元素？
640
否
完成

图6

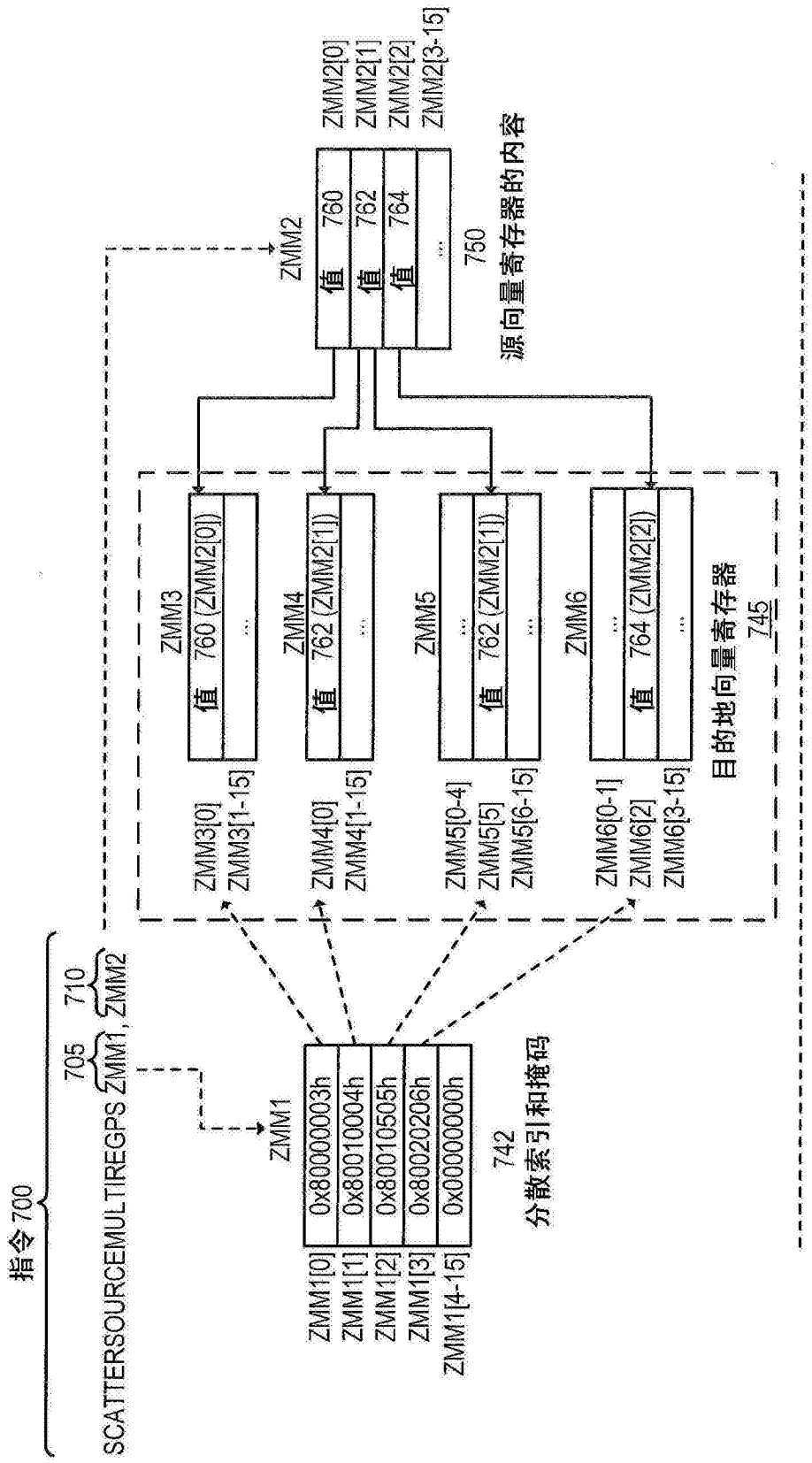

指令700
SCATTERSOURCEMULTIREGPS
705
ZMM1, ZMM2
710
ZMM1
ZMM1[0] 0x80000003h
ZMM1[1] 0x80010004h
ZMM1[2] 0x80010505h
ZMM1[3] 0x80020206h
ZMM1[4-15] 0x00000000h
742
分散索引和掩码
ZMM2
值 760
值 762
值 764
...
750
源向量寄存器的内容
ZMM2[0]
ZMM2[1]
ZMM2[2]
ZMM2[3-15]
ZMM3
值 760 (ZMM2[0])
ZMM4
值 762 (ZMM2[1])
ZMM5
值 762 (ZMM2[1])
ZMM6
值 764 (ZMM2[2])
ZMM3[0]
ZMM3[1-15]
ZMM4[0]
ZMM4[1-15]
ZMM5[0-4]
ZMM5[5]
ZMM5[6-15]
ZMM6[0-1]
ZMM6[2]
ZMM6[3-15]
目的地向量寄存器
745

图7

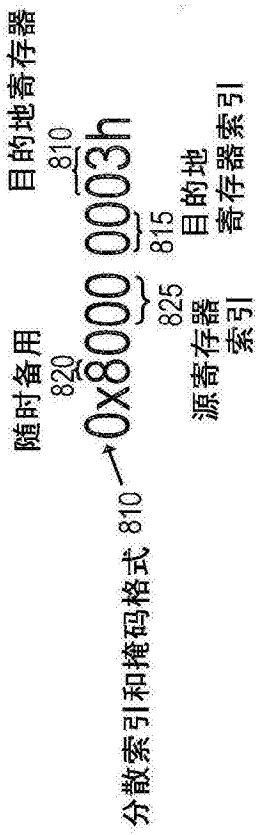

目的地寄存器
810
0x8000 0003h
随时备用
820
分散索引和掩码格式 810
825
源寄存器
索引
815
目的地
寄存器索引

图8

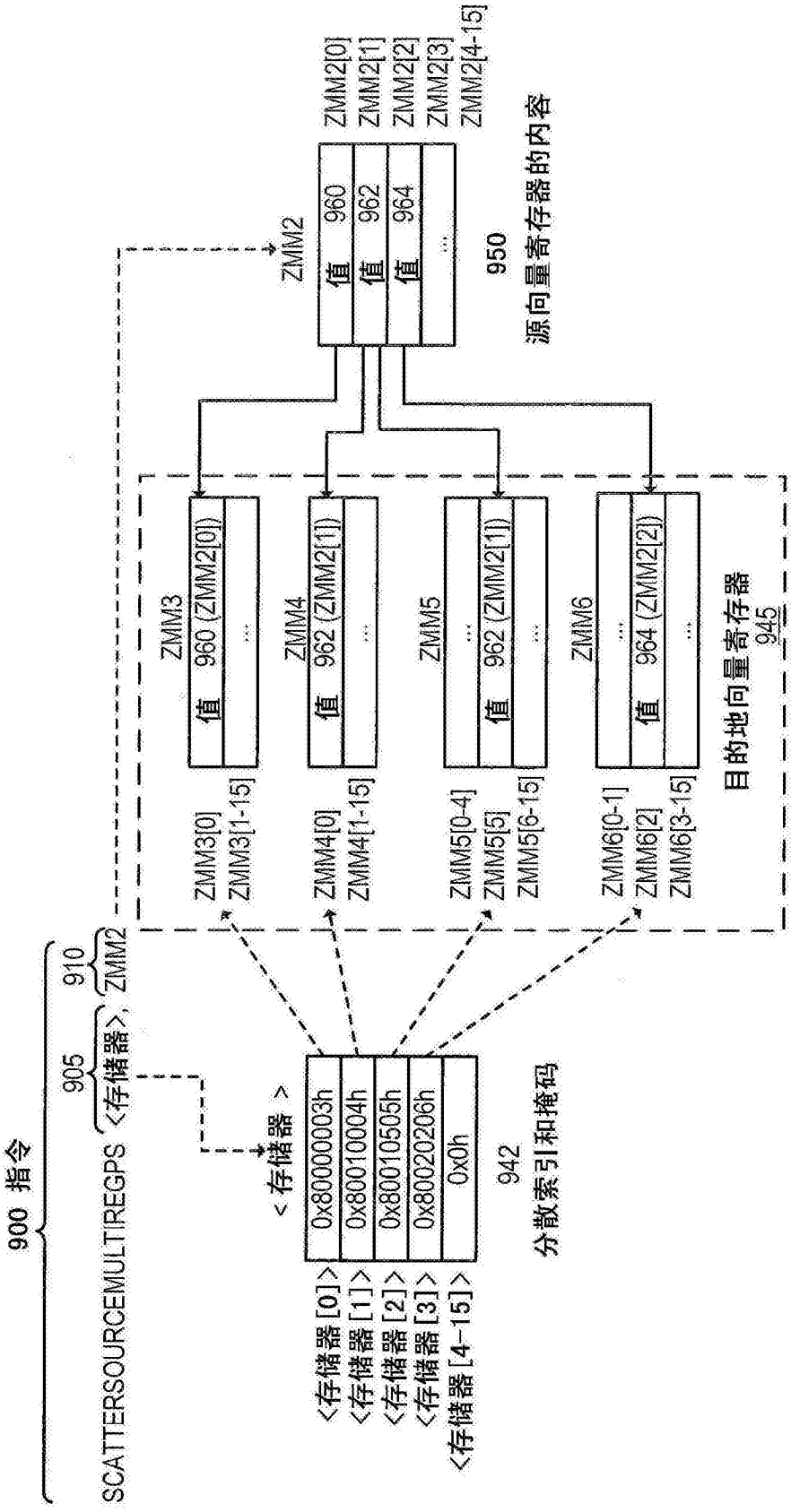

900 指令
SCATTERSOURCEMULTIREGPS
905
<存储器>,
910
ZMM2
< 存储器 >
0x80000003h
0x80010004h
0x80010505h
0x80020206h
0x0h
<存储器 [0]>
<存储器 [1]>
<存储器 [2]>
<存储器 [3]>
<存储器 [4-15]>
942
分散索引和掩码
ZMM2
950
源向量寄存器的内容
ZMM2[0]
ZMM2[1]
ZMM2[2]
ZMM2[3]
ZMM2[4-15]
值 960
值 962
值 964
...
ZMM3
值 960 (ZMM2[0])
ZMM4
值 962 (ZMM2[1])
ZMM5
值 962 (ZMM2[1])
ZMM6
值 964 (ZMM2[2])
ZMM3[0]
ZMM3[1-15]
ZMM4[0]
ZMM4[1-15]
ZMM5[0-4]
ZMM5[5]
ZMM5[6-15]
ZMM6[0-1]
ZMM6[2]
ZMM6[3-15]
目的地向量寄存器
945

图9

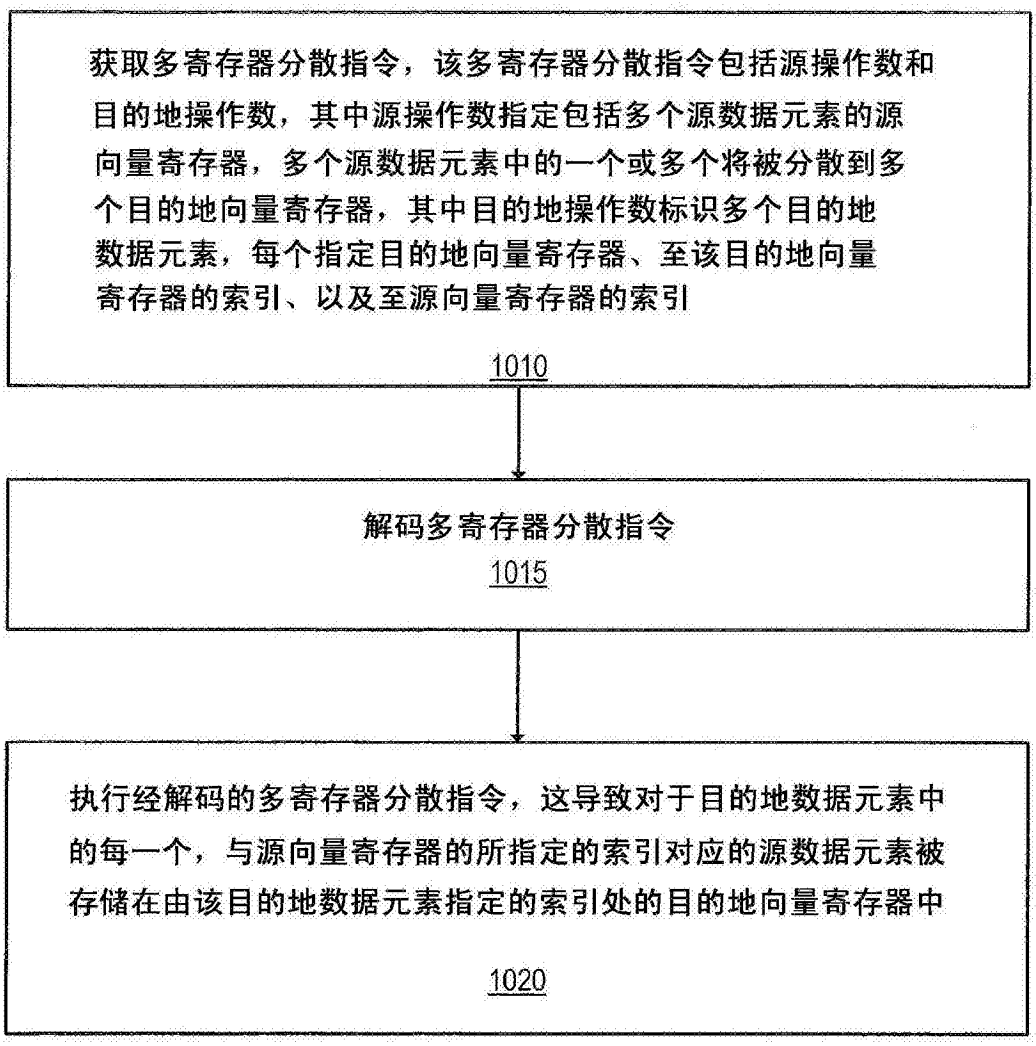

获取多寄存器分散指令，该多寄存器分散指令包括源操作数和目的地操作数，其中源操作数指定包括多个源数据元素的源向量寄存器，多个源数据元素中的一个或多个将被分散到多个目的地向量寄存器，其中目的地操作数标识多个目的地数据元素，每个指定目的地向量寄存器、至该目的地向量寄存器的索引、以及至源向量寄存器的索引
1010
解码多寄存器分散指令
1015
执行经解码的多寄存器分散指令，这导致对于目的地数据元素中的每一个，与源向量寄存器的所指定的索引对应的源数据元素被存储在由该目的地数据元素指定的索引处的目的地向量寄存器中
1020

图10

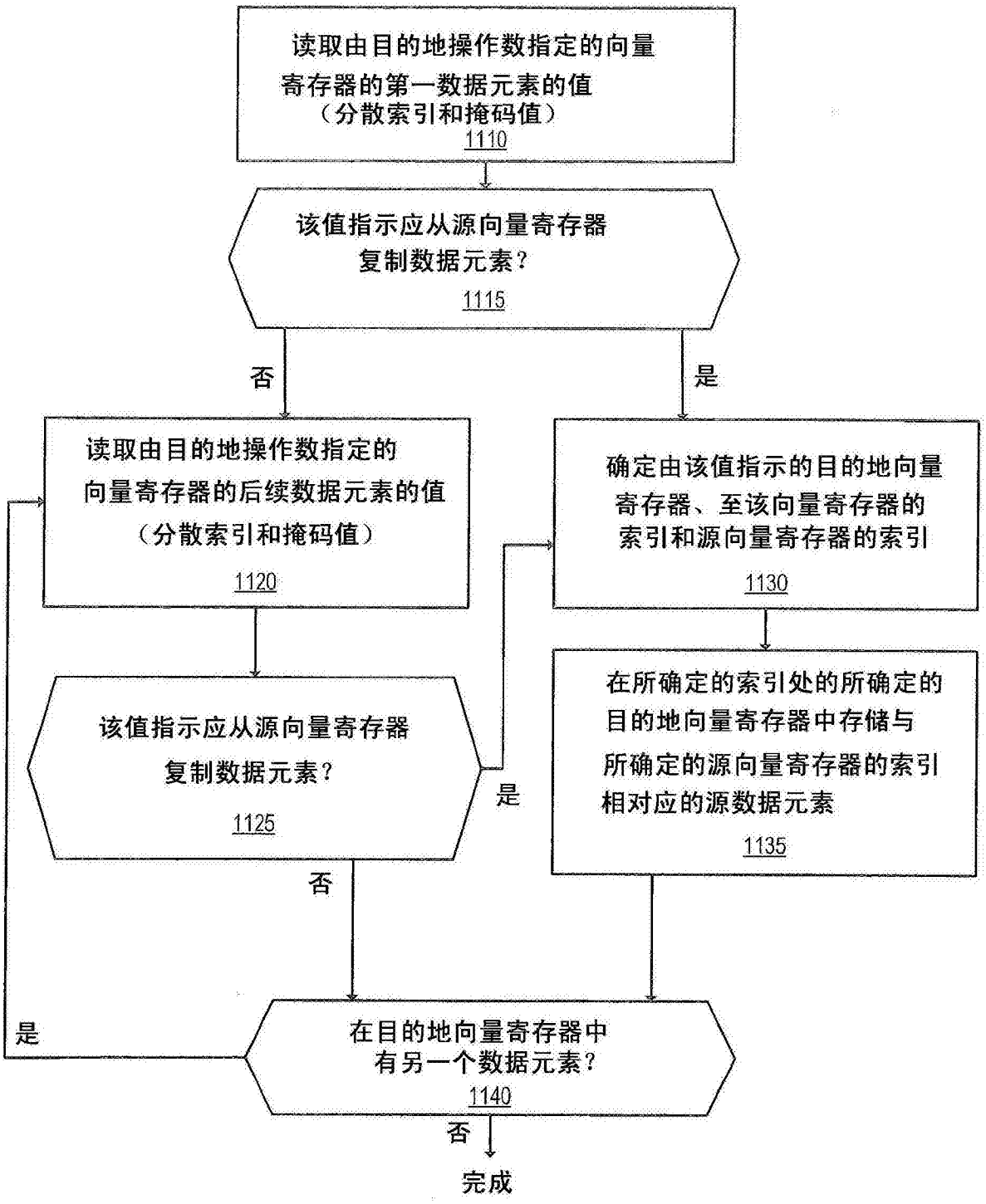

读取由目的地操作数指定的向量寄存器的第一数据元素的值（分散索引和掩码值）
1110
该值指示应从源向量寄存器复制数据元素？
1115
否
是
读取由目的地操作数指定的向量寄存器的后续数据元素的值（分散索引和掩码值）
1120
确定由该值指示的目的地向量寄存器、至该向量寄存器的索引和源向量寄存器的索引
1130
该值指示应从源向量寄存器复制数据元素？
1125
是
在所确定的索引处的所确定的目的地向量寄存器中存储与所确定的源向量寄存器的索引相对应的源数据元素
1135
否
是
在目的地向量寄存器中有另一个数据元素？
1140
否
完成

图11

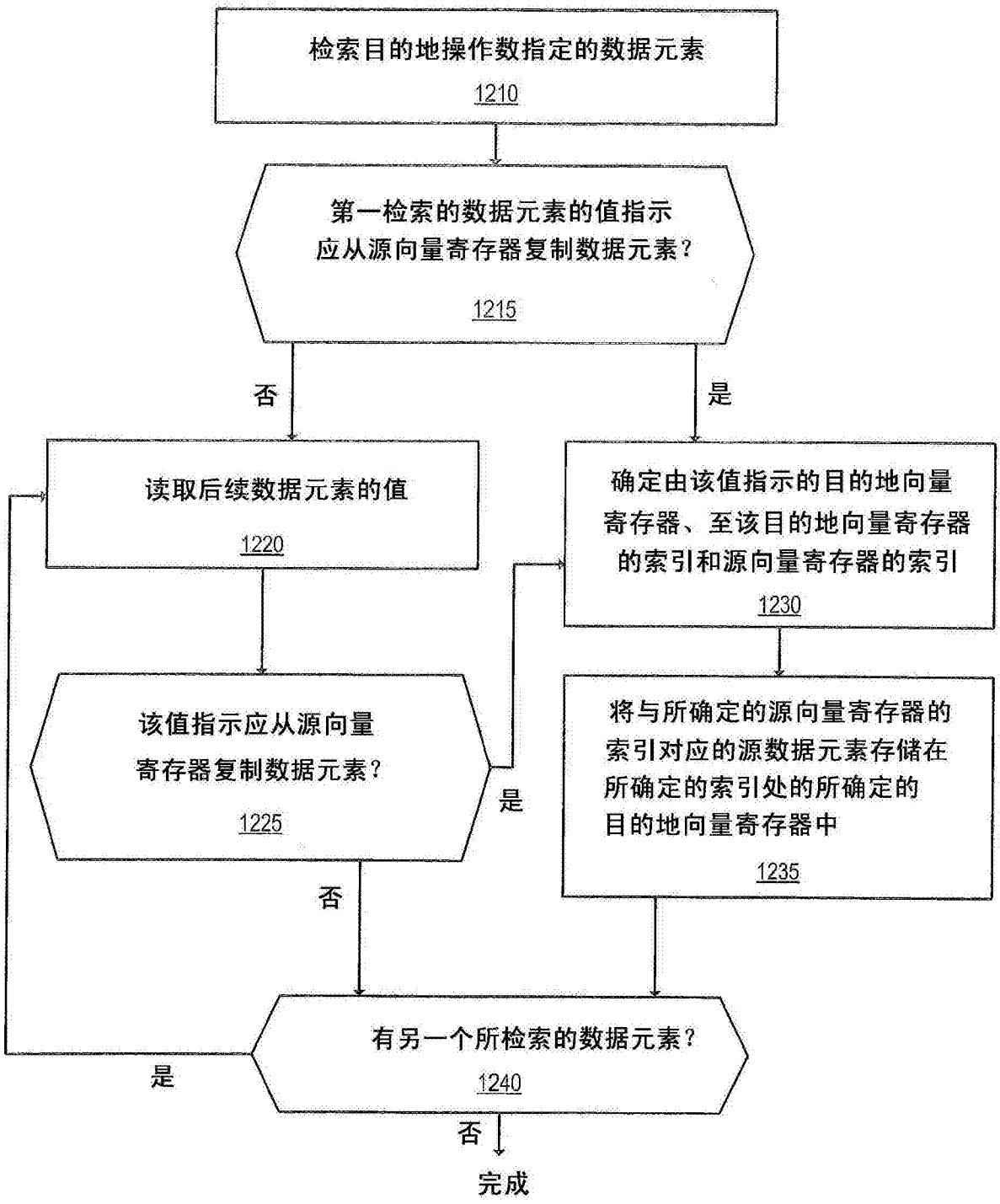

检索目的地操作数指定的数据元素
1210
第一检索的数据元素的值指示
应从源向量寄存器复制数据元素？
1215
否
是
读取后续数据元素的值
1220
确定由该值指示的目的地向量
寄存器、至该目的地向量寄存器
的索引和源向量寄存器的索引
1230
该值指示应从源向量
寄存器复制数据元素？
1225
是
否
将与所确定的源向量寄存器的
索引对应的源数据元素存储在
所确定的索引处的所确定的
目的地向量寄存器中
1235
有另一个所检索的数据元素？
1240
是
否
完成

图12

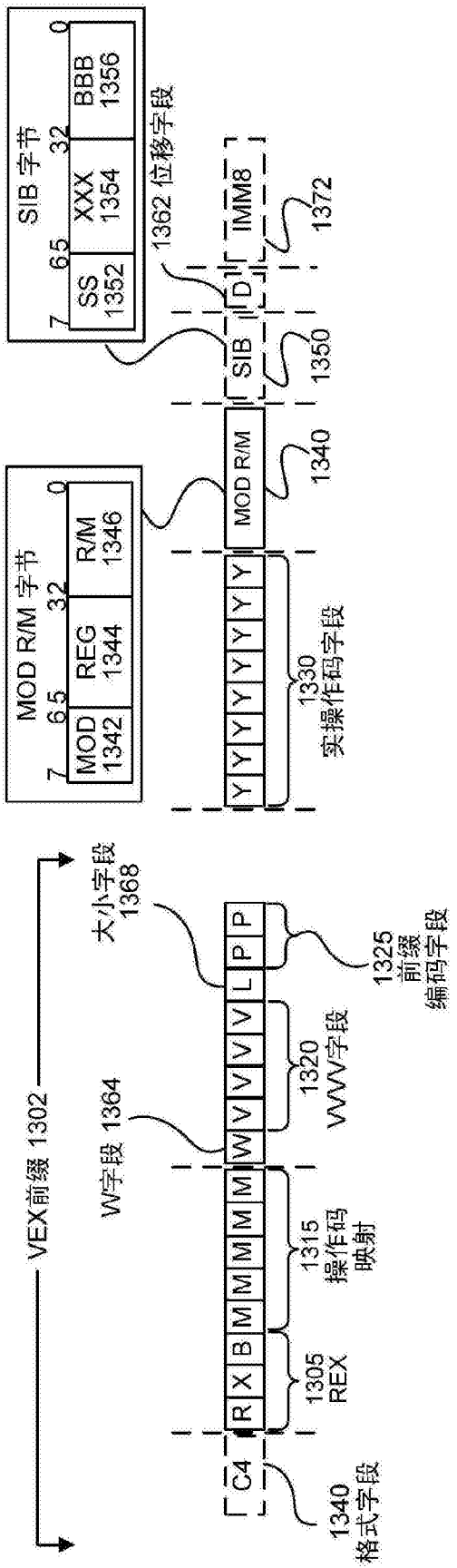

SIB 字节
7 6 5 3 2 0
SS 1352
XXX 1354
BBB 1356
1362 位移字段
MOD R/M 字节
7 6 5 3 2 0
MOD 1342
REG 1344
R/M 1346
SIB
D
IMM8
1372
1350
MOD R/M
1340
Y Y Y Y Y Y Y Y
1330 实操作码字段
VEX前缀 1302
大小字段 1368
W字段 1364
1320 VVVV字段
1325 前缀编码字段
C4
R X B M M M M M
W V V V V V L P P
1305 REX
1315 操作码映射
1340 格式字段

图13A

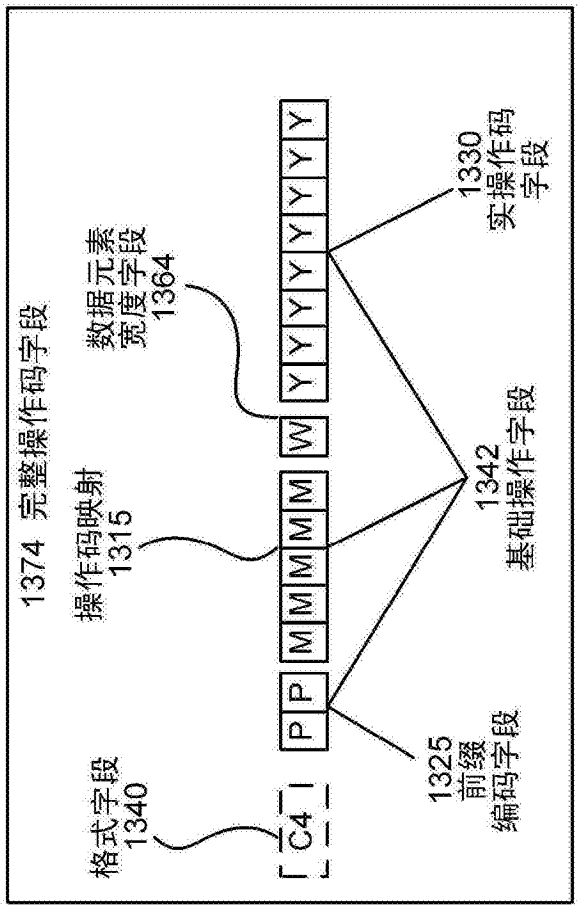

1374 完整操作码字段
格式字段
1340
操作码映射
1315
数据元素
宽度字段
1364
C4
P
P
M
M
M
M
M
W
Y
Y
Y
Y
Y
Y
Y
Y
1325
前缀
编码字段
1342
基础操作字段
1330
实操作码
字段

图13B

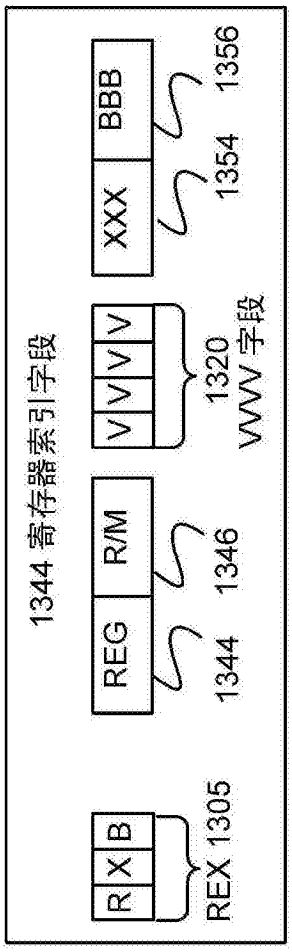

1344 寄存器索引字段
R
X
B
REX 1305
REG
R/M
1344
1346
V
V
V
V
1320
VVVV 字段
XXX
BBB
1354
1356

图13C

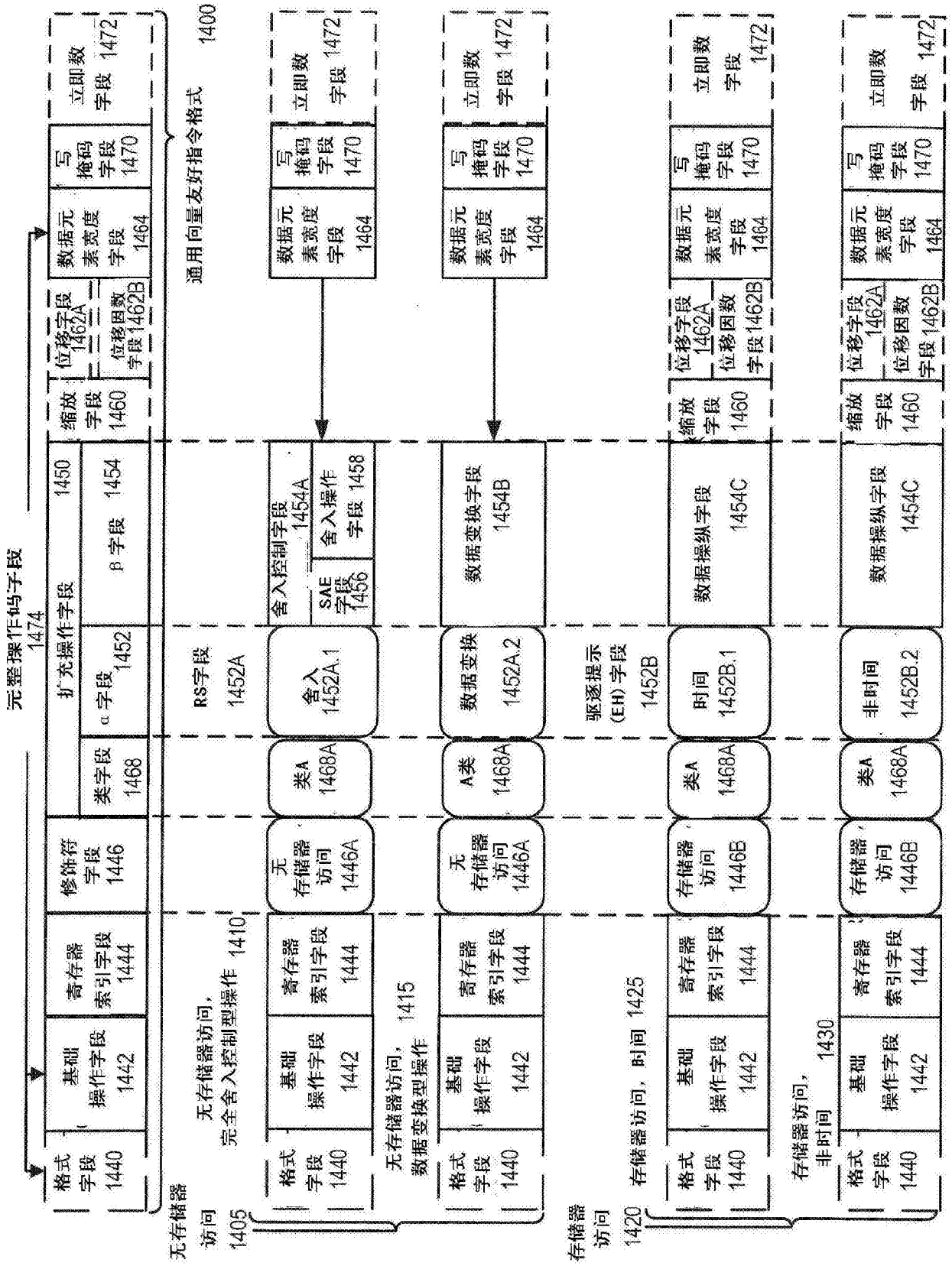

完整操作码字段 1474
通用向量友好指令格式 1400
无存储器访问 1405
格式字段 1440
基础操作字段 1442
寄存器索引字段 1444
修饰符字段 1446
扩充操作字段 1450
类字段 1468
α字段 1452
β字段 1454
缩放字段 1460
位移字段 1462A
位移因数字段 1462B
数据元素宽度字段 1464
写掩码字段 1470
立即数字段 1472
RS字段 1452A
舍入控制字段 1454A
SAE字段 1456
舍入操作字段 1458
数据变换字段 1454B
驱逐提示(EH)字段 1452B
数据操纵字段 1454C
无存储器访问，完全舍入控制型操作 1410
无存储器访问，数据变换型操作 1415
存储器访问 1420
存储器访问，时间 1425
存储器访问，非时间 1430
无存储器访问 1446A
存储器访问 1446B
类A 1468A
舍入 1452A.1
数据变换 1452A.2
时间 1452B.1
非时间 1452B.2

图14A

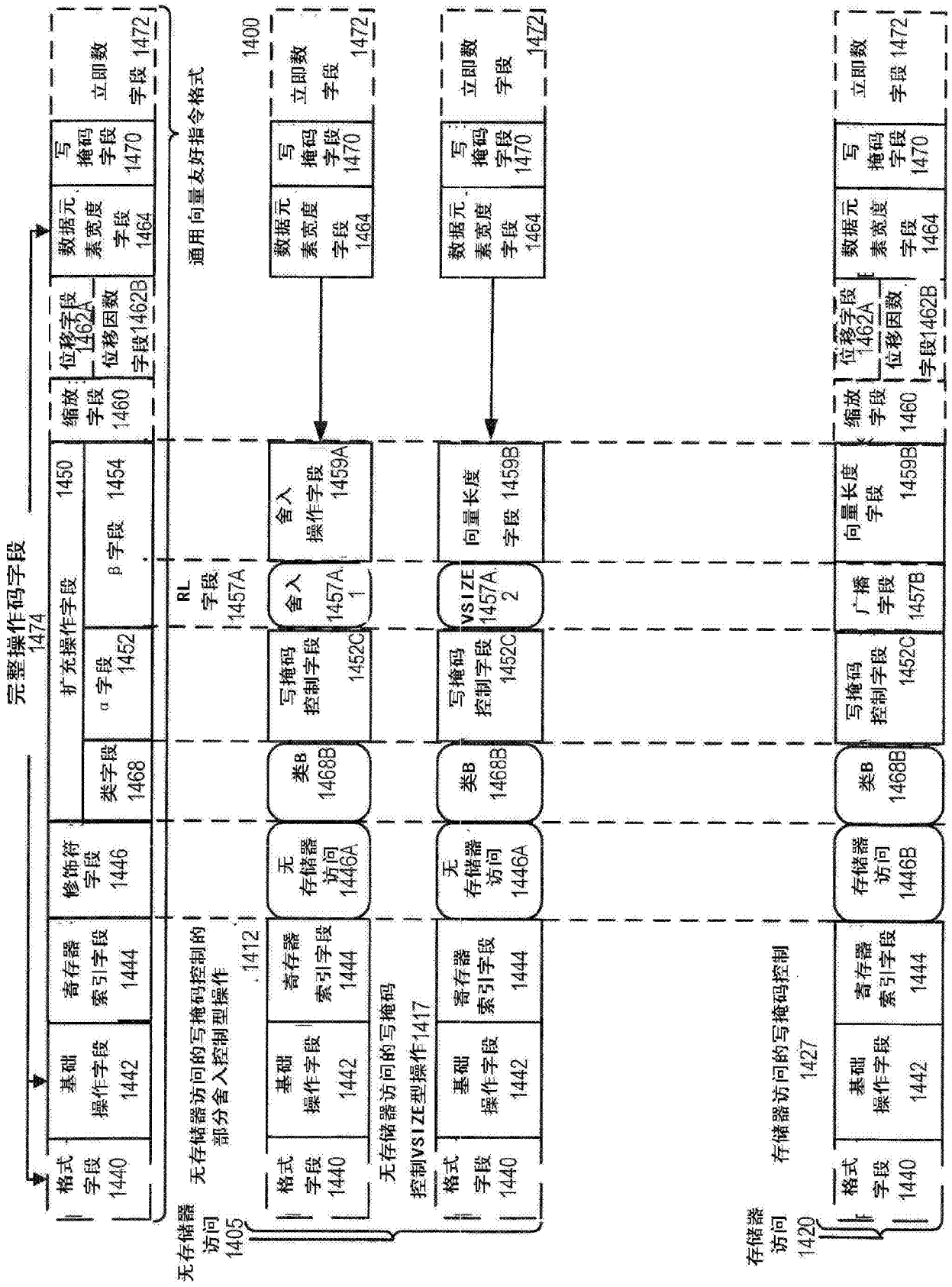

完整操作码字段
1474
通用向量友好指令格式
1400
无存储器访问
1405
无存储器访问的写掩码控制的部分舍入控制型操作
1412
无存储器访问的写掩码控制VSIZE型操作1417
存储器访问
1420
存储器访问的写掩码控制
1427
格式字段 1440
基础操作字段 1442
寄存器索引字段 1444
修饰符字段 1446
无存储器访问 1446A
存储器访问 1446B
类字段 1468
类B 1468B
扩充操作字段 1450
α字段 1452
β字段 1454
写掩码控制字段 1452C
RL字段 1457A
舍入 1457A.1
舍入操作字段 1459A
VSIZE 1457A.2
向量长度字段 1459B
广播字段 1457B
缩放字段 1460
位移字段 1462A
位移因数字段 1462B
数据元素宽度字段 1464
写掩码字段 1470
立即数字段 1472

图14B

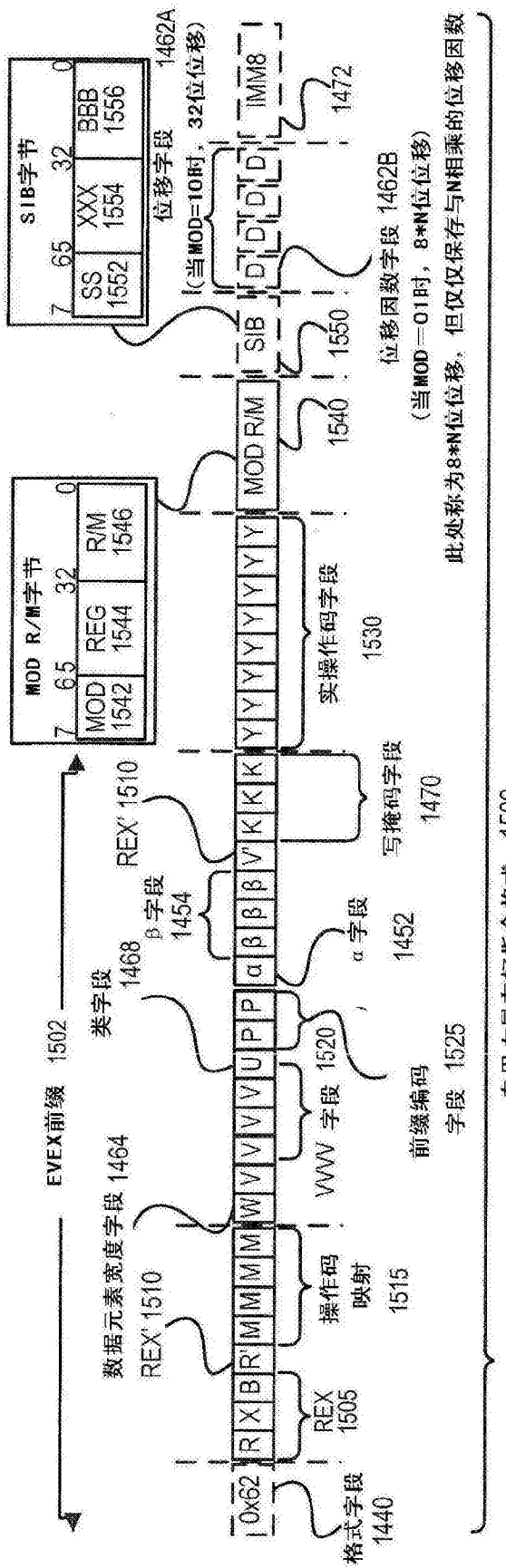

EVEX前缀 1502
格式字段 1440
0x62
REX 1505
操作码映射 1515
前缀编码字段 1525
数据元素宽度字段 1464
REX' 1510
类字段 1468
β字段 1454
REX' 1510
α字段 1452
写掩码字段 1470
实操作码字段 1530
MOD R/M字节
MOD 1542
REG 1544
R/M 1546
SIB字节
SS 1552
XXX 1554
BBB 1556
位移字段 1462A
(当MOD=10时，32位位移)
位移因数字段 1462B
(当MOD=01时，8*N位位移)
此处称为8*N位位移，但仅保存与N相乘的位移因数
专用向量友好指令格式 1500
SIB 1550
MOD R/M 1540
IMM8 1472

图15A

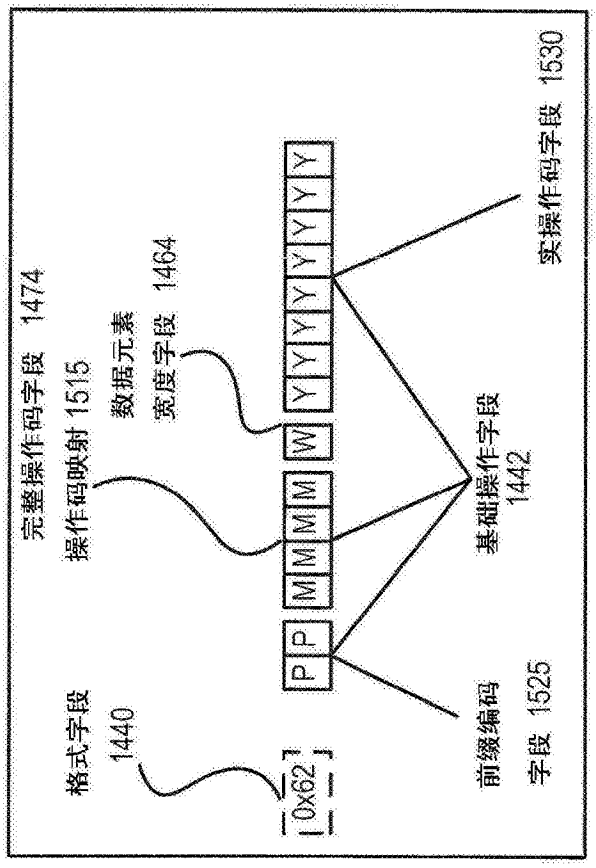

完整操作码字段 1474
操作码映射 1515
数据元素
宽度字段 1464
格式字段
1440
0x62
P P M M M W Y Y Y Y Y Y Y Y
基础操作字段
1442
前缀编码
字段 1525
实操作码字段 1530

图15B

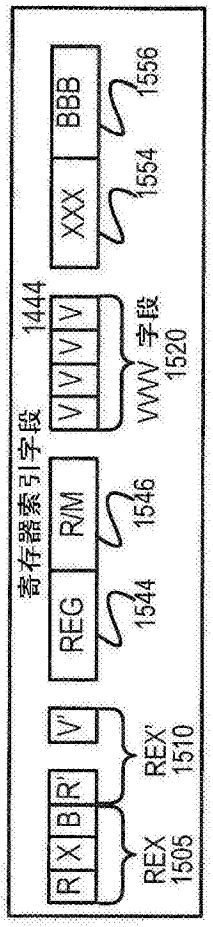

寄存器索引字段
1444
REG
R/M
1544
1546
V V V V
VVVV 字段
1520
XXX
BBB
1554
1556
R X B R' V'
REX
1505
REX'
1510

图15C

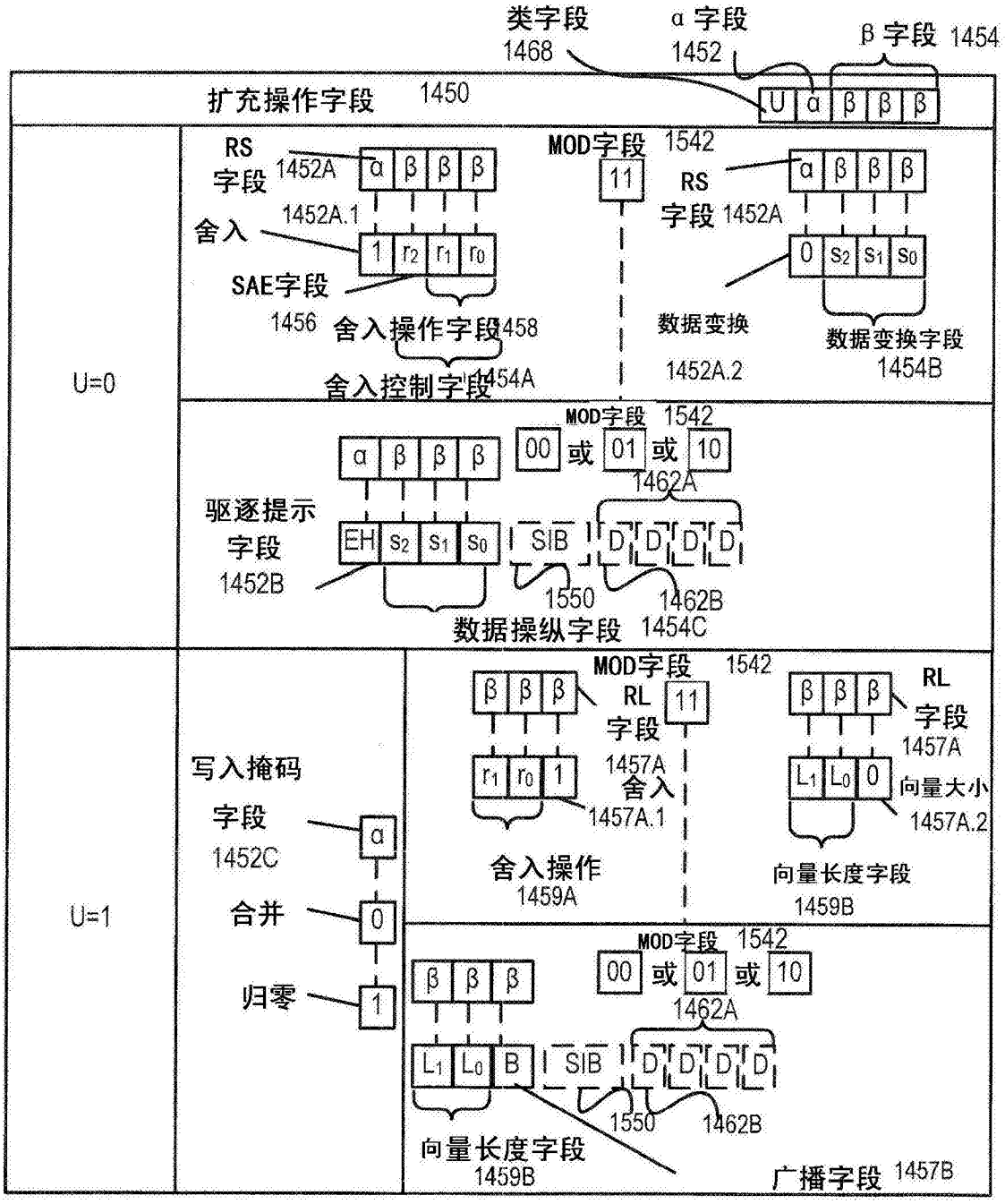

类字段
1468
α 字段
1452
β 字段 1454
扩充操作字段 1450
U α β β β
U=0
RS
字段 1452A
α β β β
1452A.1
舍入
1 r2 r1 r0
SAE字段
1456
舍入操作字段 1458
舍入控制字段 1454A
MOD字段 1542
11
RS
字段 1452A
α β β β
0 s2 s1 s0
数据变换
1452A.2
数据变换字段
1454B
MOD字段 1542
00 或 01 或 10
1462A
α β β β
驱逐提示
字段
1452B
EH s2 s1 s0
SIB D D D D
1550
1462B
数据操纵字段 1454C
U=1
写入掩码
字段
1452C
α
合并
0
归零
1
MOD字段 1542
β β β
RL
字段
11
1457A
r1 r0 1
舍入
1457A.1
舍入操作
1459A
β β β
RL
字段
1457A
L1 L0 0
向量大小
1457A.2
向量长度字段
1459B
MOD字段 1542
00 或 01 或 10
1462A
β β β
L1 L0 B
SIB D D D D
1550
1462B
向量长度字段
1459B
广播字段 1457B

图15D

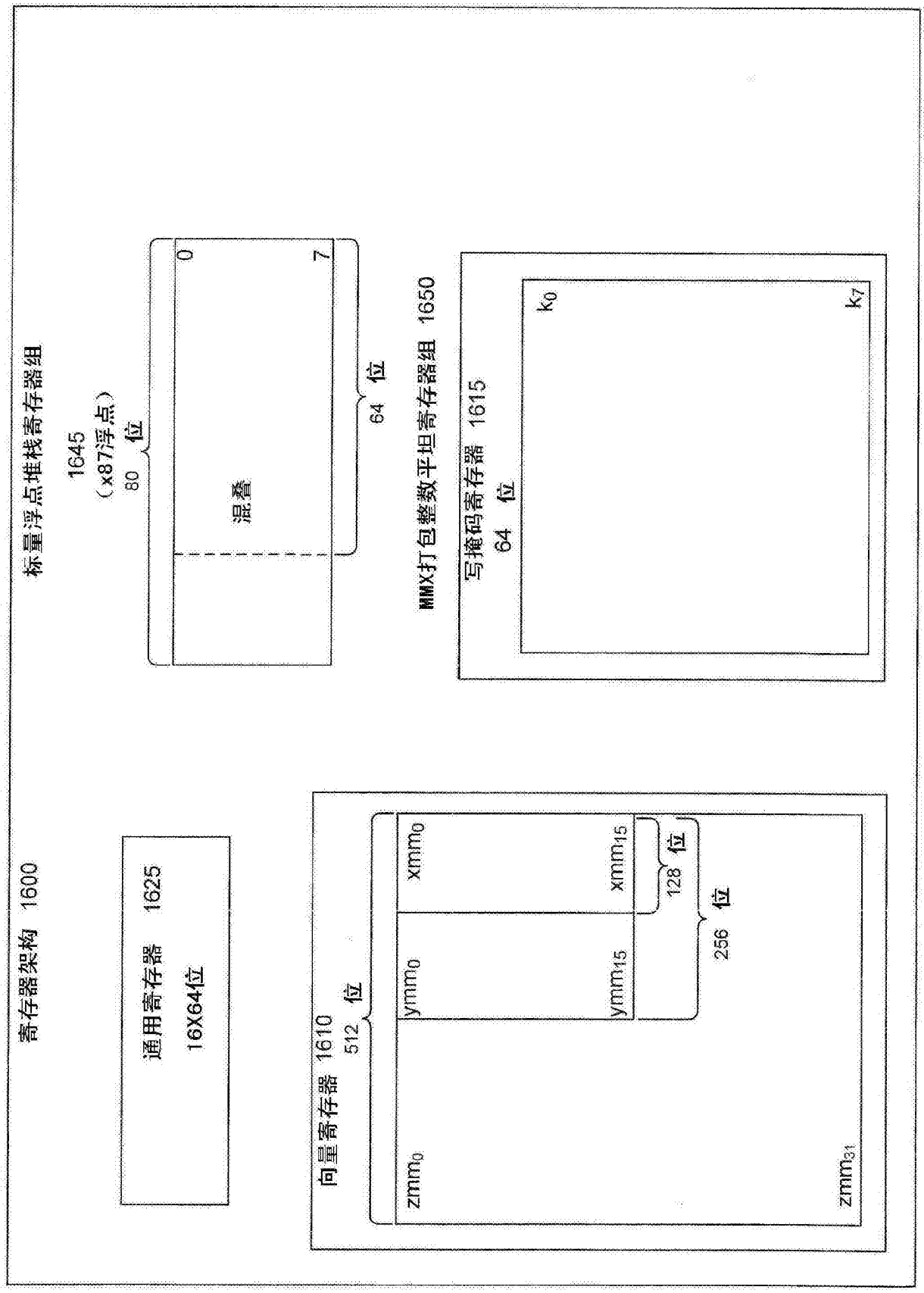

寄存器架构 1600
标量浮点堆栈寄存器组
1645
（x87浮点）
80 位
0
7
混叠
64 位
通用寄存器 1625
16X64位
向量寄存器 1610
512 位
zmm0
ymm0
xmm0
ymm15
xmm15
128 位
256 位
zmm31
MMX打包整数平坦寄存器 1650
写掩码寄存器 1615
64 位
k0
k7

图16

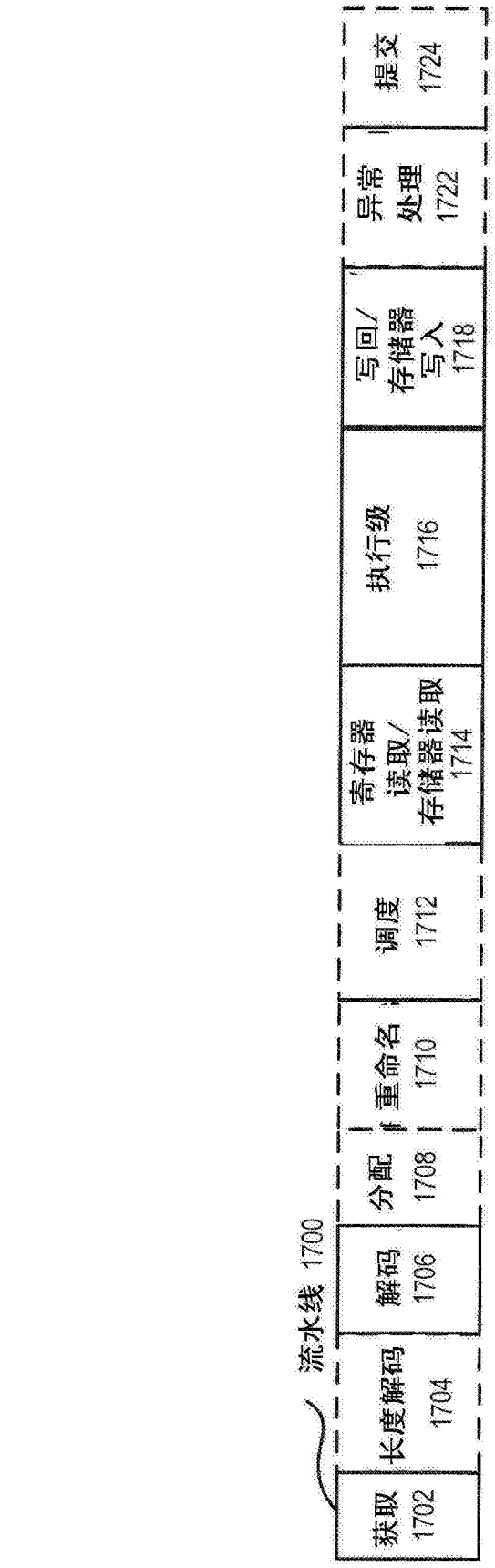

流水线 1700
获取 1702
长度解码 1704
解码 1706
分配 1708
重命名 1710
调度 1712
寄存器读取/存储器读取 1714
执行级 1716
写回/存储器写入 1718
异常处理 1722
提交 1724

图17A

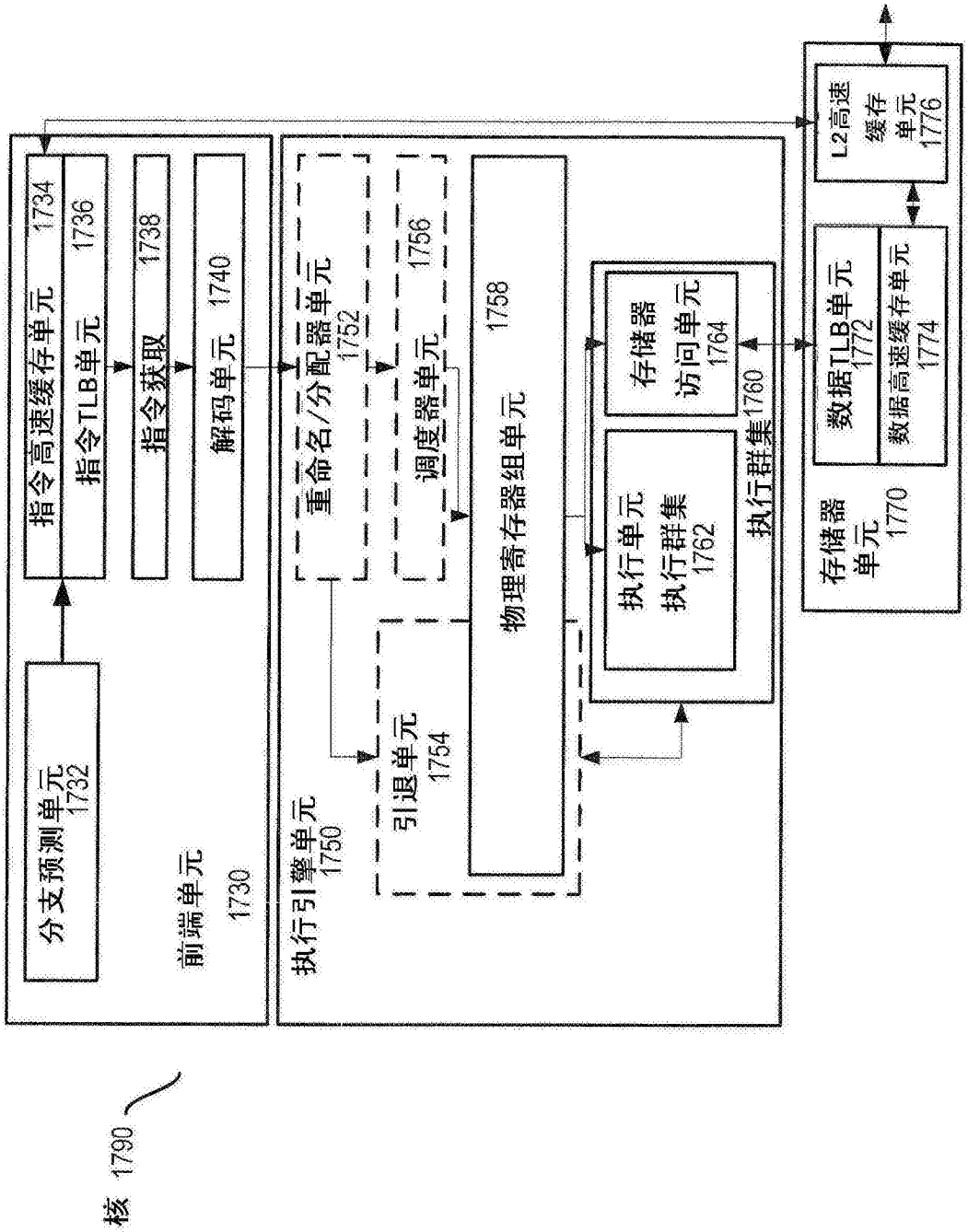

核 1790
前端单元 1730
分支预测单元 1732
指令高速缓存单元 1734
指令TLB单元 1736
指令获取 1738
解码单元 1740
执行引擎单元 1750
重命名/分配器单元 1752
引退单元 1754
调度器单元 1756
物理寄存器组单元 1758
执行单元 执行群集 1762
存储器访问单元 1764
执行群集 1760
存储器单元 1770
数据TLB单元 1772
数据高速缓存单元 1774
L2高速缓存单元 1776

图17B

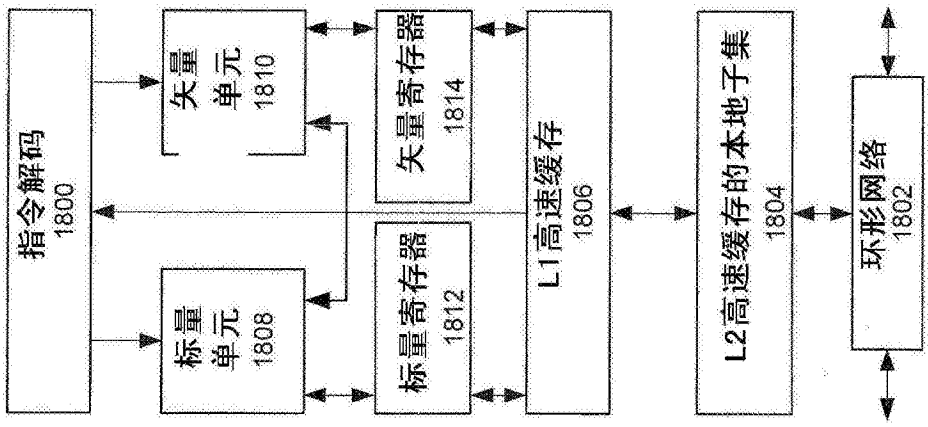

指令解码
1800
矢量
单元
1810
矢量寄存器
1814
标量
单元
1808
标量寄存器
1812
L1高速缓存
1806
L2高速缓存的本地子集
1804
环形网络
1802

图18A

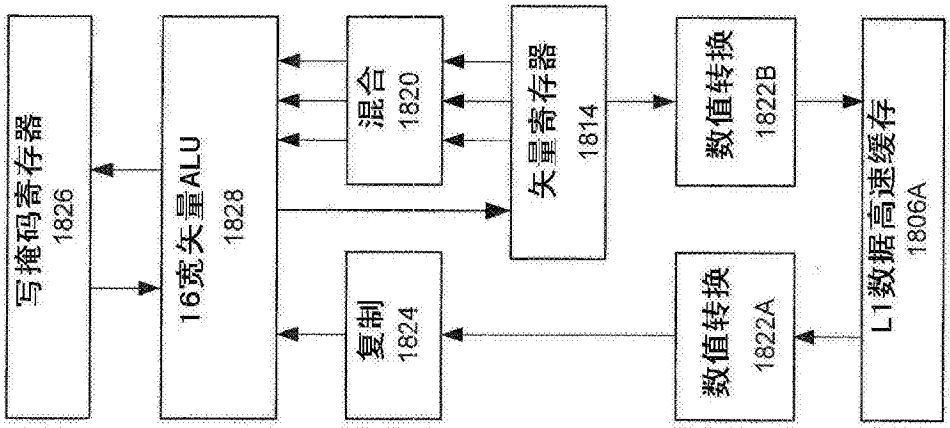

写掩码寄存器
1826
16宽矢量ALU
1828
混合
1820
矢量寄存器
1814
数值转换
1822B
L1数据高速缓存
1806A
复制
1824
数值转换
1822A

图18B

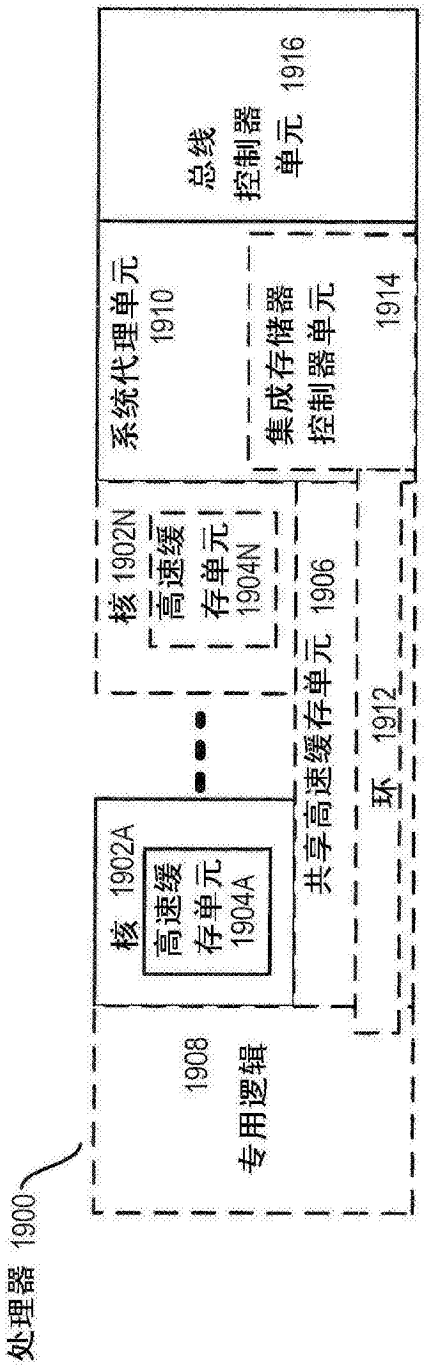

处理器 1900
1908
专用逻辑
核 1902A
高速缓存单元 1904A
核 1902N
高速缓存单元 1904N
共享高速缓存单元 1906
环 1912
系统代理单元 1910
集成存储器控制器单元 1914
总线控制器单元 1916

图19

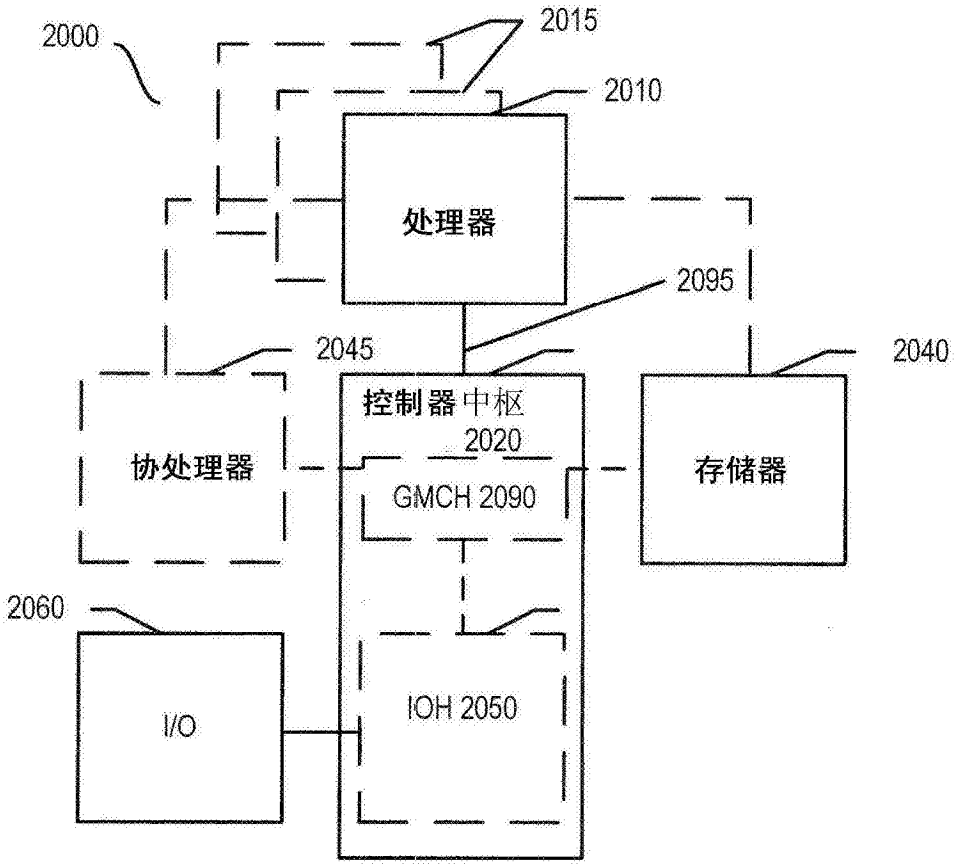

2000
2015
2010
处理器
2095
2045
2040
控制器中枢
2020
协处理器
GMCH 2090
存储器
2060
I/O
IOH 2050

图20

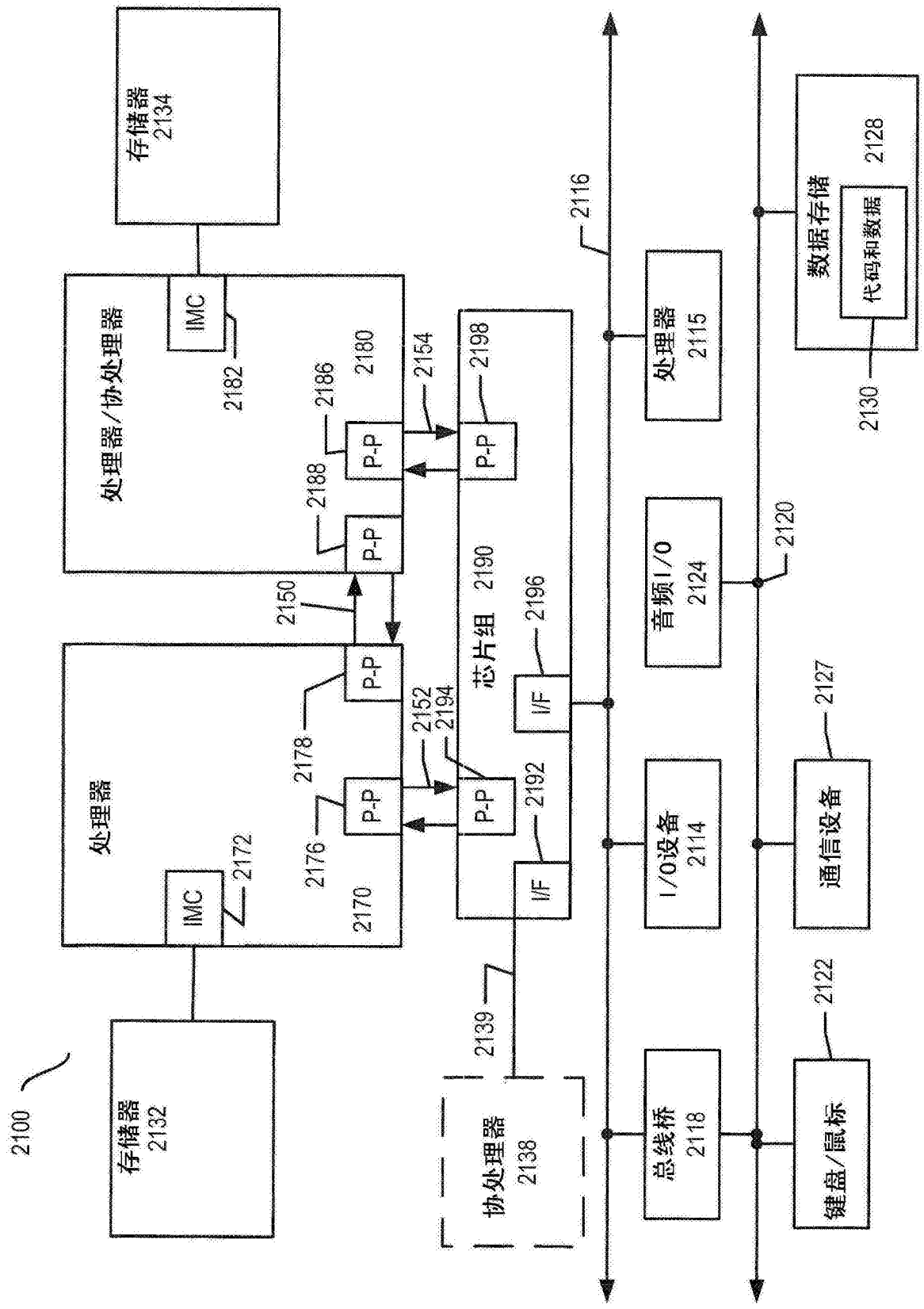

2100
存储器 2132
存储器 2134
处理器 2170
IMC 2172
处理器/协处理器 2180
IMC 2182
P-P 2176
P-P 2178
P-P 2186
P-P 2188
2150
2152
2154
芯片组 2190
P-P 2192
P-P 2194
P-P 2198
I/F 2196
I/F 2139
协处理器 2138
2116
处理器 2115
音频I/O 2124
I/O设备 2114
总线桥 2118
2120
2127
通信设备
2122
键盘/鼠标
数据存储 2128
代码和数据
2130

图21

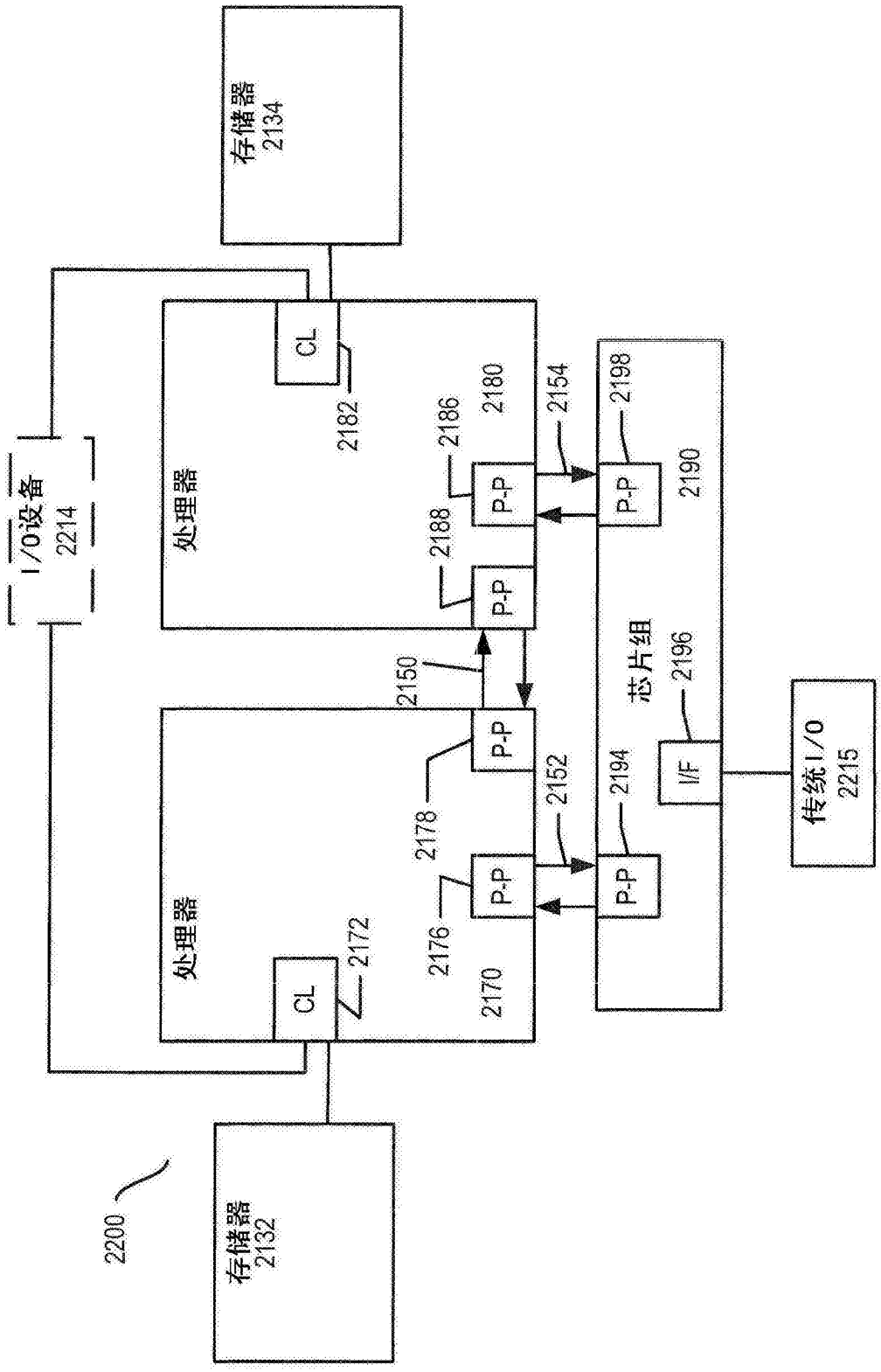

存储器
2134
CL
2182
处理器
2180
2186
P-P
2188
P-P
2154
2198
P-P
2190
芯片组
2196
I/F
传统I/O
2215
2150
P-P
2178
处理器
2152
2194
P-P
2176
P-P
2170
CL
2172
I/O设备
2214
2200
存储器
2132

图22

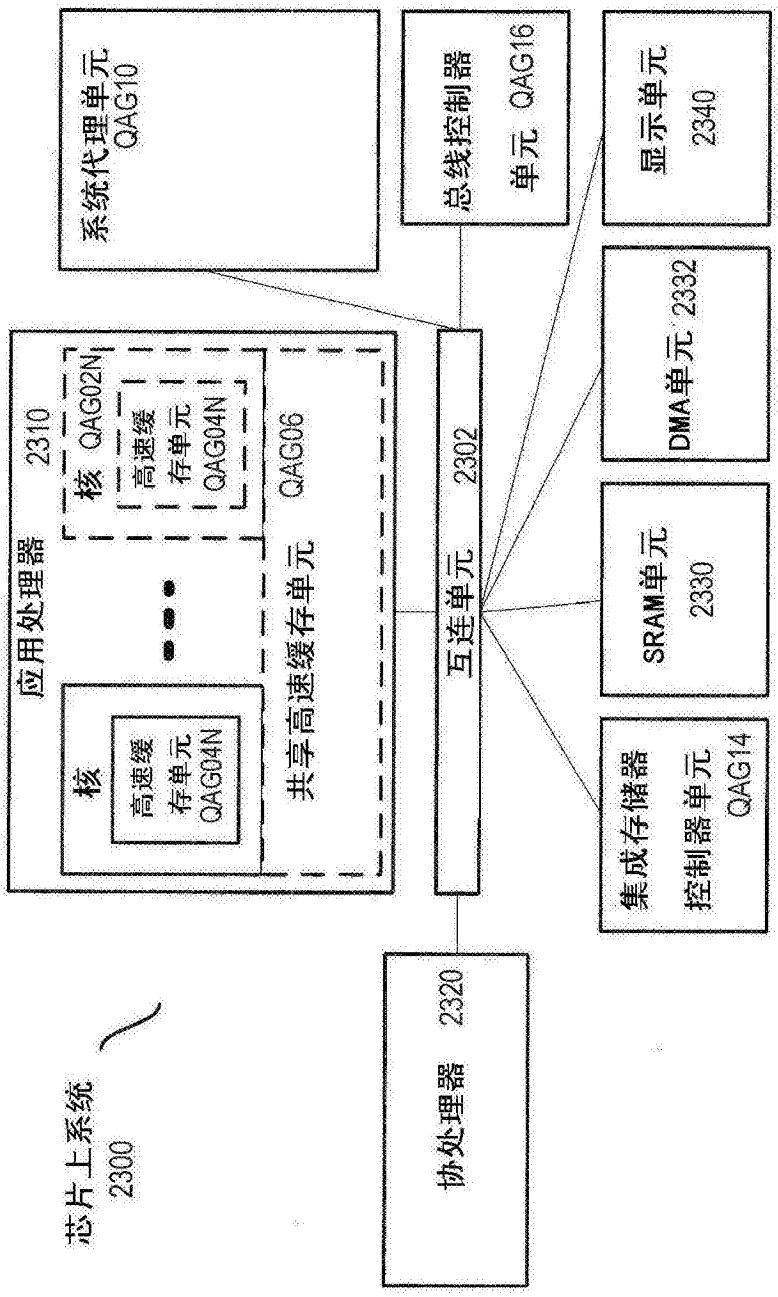

芯片上系统
2300
应用处理器 2310
核 QAG02N
高速缓存单元 QAG04N
核
高速缓存单元 QAG04N
共享高速缓存单元 QAG06
系统代理单元 QAG10
总线控制器单元 QAG16
互连单元 2302
协处理器 2320
显示单元 2340
DMA单元 2332
SRAM单元 2330
集成存储器控制器单元 QAG14

图23

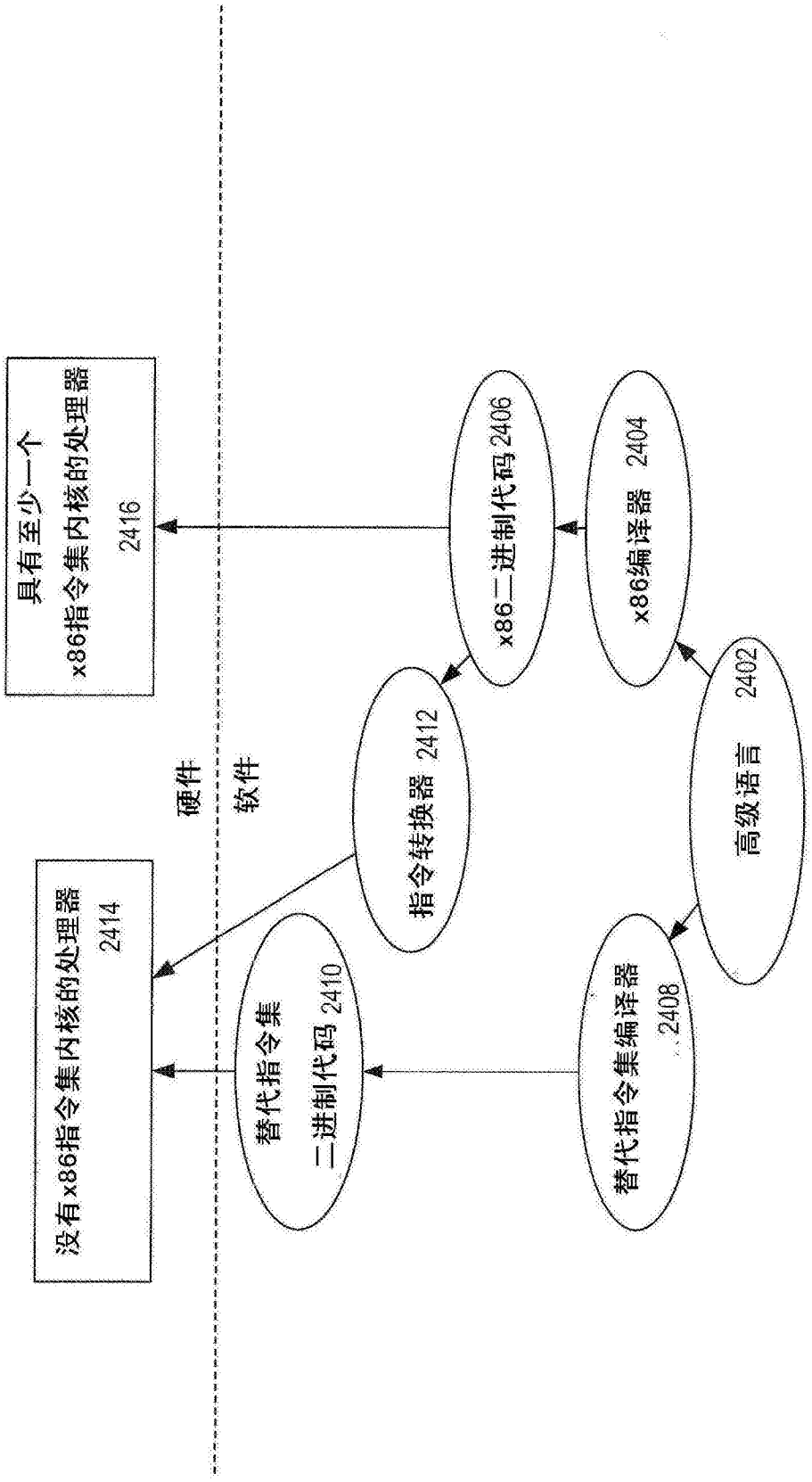

具有至少一个
x86指令集内核的处理器
2416
没有x86指令集内核的处理器
2414
硬件
软件
x86二进制代码 2406
x86编译器 2404
指令转换器 2412
高级语言 2402
替代指令集
二进制代码 2410
替代指令集编译器
2408

图24