(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 111177648 A
(43)申请公布日 2020.05.19

(21)申请号 201910954609.5

(22)申请日 2019.10.09

(30)优先权数据
16/186,378 2018.11.09 US

(71)申请人 英特尔公司
地址 美国加利福尼亚州

(72)发明人 A·F·海内克 R·凡伦天
M·J·查尼 R·萨德
M·阿德尔曼 Z·斯波伯
A·格雷德斯廷 S·卢巴诺维奇

(74)专利代理机构 上海专利商标事务所有限公司 31100
代理人 陈依心 何焜

(51)Int.Cl.
G06F 17/16(2006.01)

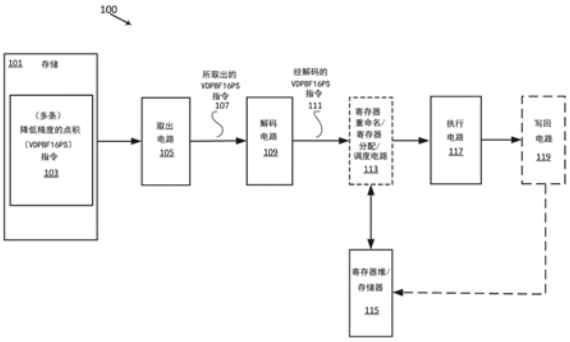
权利要求书2页 说明书24页 附图20页

(54)发明名称

用于执行16位浮点向量点积指令的系统和方法

(57)摘要

所公开实施例涉及用于执行16位浮点向量点积指令的系统和方法。在一个示例中，处理器包括：取出电路，用于取出指令，该指令具有用于指定操作码以及第一源向量、第二源向量和目的地向量的位置的字段，该操作码用于指示执行电路用于将所指定的第一源和第二源的N对16位浮点格式化元素相乘，并且将所得的乘积与所指定的目的地的对应的单精度元素的先前内容累加；解码电路，用于对所取出的指令解码；以及执行电路，用于如该操作码所指定地对经解码的指令作出响应。



CN 111177648 A

1.一种处理器,包括:

取出电路,用于取出指令,所述指令具有用于指定操作码以及第一源向量、第二源向量和目的地向量的位置的字段,所述操作码用于指示执行电路用于生成所述第一源向量和所述第二源向量的N对16位浮点元素的乘积,并且将所述乘积与所述目的地的对应的单精度元素的先前内容累加;

解码电路,用于对所取出的指令解码;以及

执行电路,用于根据所述操作码执行所述指令。

2.如权利要求1所述的处理器,其特征在于,所述16位浮点格式为bfloat16或binary16。

3.如权利要求1所述的处理器,其特征在于,所述16位浮点元素各自包括符号位、8位的指数以及尾数,所述尾数包括7个显式位和第八个隐式位。

4.如权利要求1-3中的任一项所述的处理器,其特征在于,N由所述指令指定并且具有4、8、16和32中的一个的值。

5.如权利要求1-3中的任一项所述的处理器,其特征在于,所述执行电路用于在没有饱和的情况下利用无限精度生成所述乘积,并且在上溢的情况下将所述累加的结果饱和至正无穷或负无穷并在任何下溢的情况下将所述累加的结果饱和至零。

6.如权利要求1-3中的任一项所述的处理器,其特征在于,所述第一源向量、所述第二源向量和所述目的地向量中的每一个位于寄存器中或位于存储器中。

7.如权利要求1-3中的任一项所述的处理器,其特征在于,所述执行电路用于并行地生成所述目的地的所有N个元素。

8.一种方法,包括:

使用取出电路取出指令,所述指令具有用于指定操作码以及第一源向量、第二源向量和目的地向量的位置的字段,所述操作码用于指示执行电路用于生成所述第一源和所述第二源的N对16位浮点格式化元素的乘积,并且将所述乘积与所述目的地的对应的单精度元素的先前内容累加;

使用解码电路对所取出的指令解码;以及

利用执行电路根据所述操作码对经解码的指令作出响应。

9.如权利要求8所述的方法,其特征在于,所述16位浮点格式为bfloat16或binary16。

10.如权利要求8所述的方法,其特征在于,所述16位浮点格式包括符号位、8位的指数以及尾数,所述尾数包括7个显式位和第八个隐式位。

11.如权利要求8-10中的任一项所述的方法,其特征在于,所述源向量和所述目的地向量中的每一个的位置在寄存器中或在存储器中。

12.如权利要求8-10中的任一项所述的方法,其特征在于,N由所述指令指定并且具有4、8、16中的一个的值。

13.如权利要求8-10中的任一项所述的方法,其特征在于,所述执行电路用于在没有饱和的情况下利用无限精度生成所述乘积,并且在上溢的情况下将所述累加的结果饱和至正无穷或负无穷并在任何下溢的情况下将所述累加的结果饱和至零。

14.如权利要求8-10中的任一项所述的方法,其特征在于,所述执行电路用于并行地生成所述目的地的所有N个元素。

15.一种包括存储器和处理器的系统,所述处理器包括:

取出电路,用于取出指令,所述指令具有用于指定操作码以及第一源向量、第二源向量和目的地向量的位置的字段,所述操作码用于指示执行电路用于生成所述第一源和所述第二源的N对16位浮点格式化元素的乘积,并且将所述乘积与所述目的地的对应的单精度元素的先前内容累加;

解码电路,用于对所取出的指令解码;以及

执行电路,用于根据所述操作码对经解码的指令作出响应。

16.如权利要求15所述的系统,其特征在于,所述16位浮点格式为bfloat16或binary16。

17.如权利要求15所述的系统,其特征在于,所述16位浮点格式包括5位的指数或8位的指数。

18.如权利要求15-17中的任一项所述的系统,其特征在于,所述源向量和所述目的地向量中的每一个的位置在所述处理器中的寄存器中或在所述存储器中。

19.如权利要求15-17中的任一项所述的系统,其特征在于,所述执行电路用于并行地生成所述目的地的所有N个元素。

20.一种机器可读介质,包括代码,所述代码当被执行时使机器执行如权利要求8-14中的任一项所述的方法。

用于执行16位浮点向量点积指令的系统和方法

技术领域

[0001] 本发明的领域一般涉及计算机处理器架构，并且更具体地涉及用于执行16位浮点向量点积指令的系统和方法。

背景技术

[0002] 指令集或指令集架构(ISA)是计算机架构中与编程有关的部分，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、中断和异常处置以及外部输入和输出(I/O)。指令集包括一种或多种指令格式。给定的指令格式定义各种字段(位的数目、位的位置)以指定将要被执行的操作以及将要对其执行那个操作的(多个)操作数，等等。给定的指令使用给定的指令格式来表达，并且指定操作和操作数。指令流是特定的指令序列，其中，该序列中的每条指令是指令按指令格式的出现。

[0003] 科学、金融、自动向量化的通用、RMS(识别、挖掘以及合成)/可视和多媒体应用程序(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同一操作的指令类型。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(这些数据元素中的每个数据元素表示单独的值)的处理器。例如，512位寄存器中的位可以被指定为要作为十六个单独的32位单精度浮点数据元素被操作的源操作数。作为另一示例，256位寄存器中的位可以被指定为要作为十六个单独的16位浮点紧缩数据元素、八个单独的32位紧缩数据元素(双字尺寸数据元素)、或三十二个单独的8位数据元素(字节(B)尺寸数据元素)被操作的源操作数。该数据类型被称为紧缩数据类型或向量数据类型，并且该数据类型的操作数被称为紧缩数据操作数或向量操作数。换句话说，紧缩数据项或向量指的是紧缩数据元素的序列；并且紧缩数据操作数或向量操作数是SIMD指令(也称为紧缩数据指令或向量指令)的源操作数或目的地操作数。

[0004] 作为示例，一种类型的SIMD指令指定了将以纵向方式对两个源向量操作数执行单个向量操作以生成具有相同尺寸的、具有相同数量的数据元素的以及按照相同数据元素的顺序的目的地向量操作数。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数据元素或结果数据元素。这些源向量操作数具有相同的尺寸，并包含相同宽度的数据元素，并且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的位的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对单独地执行由该SIMD指令所指定的操作，以生成匹配数量的结果数据元素，并且因此每一对源数据元素都具有对应的结果数据元素。由于操作是纵向的，并且由于结果向量操作数尺寸相同，具有相同数量的数据元素，并且结果数据元素以与源向量操作数相同的数据元素顺序来存储，因此，结果数据元素处于结果向量操作数的、与这些结果数据元素的对应源数据元素对在源向量操作数中的位的位置相同的位的位置处。除这种示例性类型的SIMD指令之外，还存在各种其他类型

的SIMD指令。

[0005] 包含16位浮点元素的向量的点积乘法在多个算法中是有用的，这些算法对16位源执行乘法并且将乘法结果与32位目的地向量元素累加。

附图说明

[0006] 图1是图示根据实施例的用于执行VDPBF16PS指令的处理组件的框图；

[0007] 图2是图示根据实施例的VDPBF16PS指令的执行的框图；

[0008] 图3A是图示根据实施例的VDPBF16PS指令的示例性执行的伪代码；

[0009] 图3B是图示根据实施例的VDPBF16PS指令的示例性执行的伪代码；

[0010] 图3C是图示根据实施例的用于与图3A和图3B的伪代码一起使用的助手函数的伪代码；

[0011] 图4是图示根据实施例的处理器对VDPBF16PS指令作出响应的流程图；

[0012] 图5是图示根据实施例的VDPBF16PS指令的格式的框图；

[0013] 图6A-图6B是图示根据本发明的一些实施例的通用向量友好指令格式及其指令模板的框图；

[0014] 图6A是图示根据本发明的一些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；

[0015] 图6B是图示根据本发明的一些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；

[0016] 图7A是图示根据本发明的一些实施例的示例性专用向量友好指令格式的框图；

[0017] 图7B是图示根据一个实施例的构成完整操作码字段的具有专用向量友好指令格式的字段的框图；

[0018] 图7C是图示根据一个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；

[0019] 图7D是图示根据一个实施例的构成扩充操作字段的具有专用向量友好指令格式的字段的框图；

[0020] 图8是根据一个实施例的寄存器架构的框图；

[0021] 图9A是图示根据一些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；

[0022] 图9B是图示根据一些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/执行架构核两者的框图；

[0023] 图10A-图10B图示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)；

[0024] 图10A是根据一些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第二级(L2)高速缓存的本地子集的框图；

[0025] 图10B是根据一些实施例的图10A中的处理器核的一部分的展开图；

[0026] 图11是根据一些实施例的可具有多于一个的核、可具有集成存储器控制器、并且可具有集成图形的处理器的框图；

[0027] 图12-图15是示例性计算机架构的框图；

[0028] 图12示出根据一些实施例的系统的框图；

[0029] 图13是根据一些实施例的更具体的第一示例性系统的框图；

[0030] 图14是根据一些实施例的更具体的第二示例性系统的框图；

[0031] 图15是根据一些实施例的芯片上系统(SoC)的框图；以及

[0032] 图16是根据一些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

[0033] 在下列描述中，阐述了众多特定细节。然而，应该理解，一些实施例可在没有这些特定细节的情况下实施。在其他实例中，未详细示出公知的电路、结构和技术，以免使对本描述的理解模糊。

[0034] 说明书中对“一个实施例”、“实施例”、“示例实施例”等的引用表明所描述的实施例可以包括特征、结构或特性，但是每个实施例可能不一定都包括该特征、结构或特性。此外，此类短语不一定是指同一个实施例。此外，当关于实施例描述特征、结构或特性时，认为影响关于如果被明确描述的其他实施例的此类特征、结构或特性是在本领域技术人员的知识范围之内的。

[0035] 如上文所提及，包含16位浮点元素的、将结果与32位目的地向量元素累加的向量的点积乘法在多个算法中是有用的。本文中所公开并且通过附图所图示的是向量紧缩数据指令(VDPBF16PS，助记符指示“VDP”＝VDPBF16PS Vector Dot Product(向量点积)，“BF16”＝格式化为BFloat16的源，将乘积累加到Packed Single-precision(紧缩单精度))，该向量紧缩数据指令实现两个源向量中的多对16位浮点元素的点积乘法。该指令取得针对每个元素具有成对的bfloat16值的两个源向量(即，成对的16位元素构成32位源元素)，并且生成具有单精度元素的目的地向量(即，也是32位，因此源寄存器和目的地寄存器是平衡的)。乘法输出是32位，并且被饱和并与目的地寄存器的先前内容累加。

[0036] 与为源元素和目的地元素两者都使用单精度的算法相比，所公开的VDPBF16PS指令预期实现可比拟的质量，但具有减少的存储器利用和存储器带宽要求，这将用于改善性能和功率效率，尤其在机器学习情境中。

[0037] 如上文所提及，所公开的VDPBF16PS指令通过以下操作来对向量执行点积操作：将两个源寄存器的16位浮点(例如，bfloat16)元素相乘并且将结果累加在单精度目的地向量中。在不具有所公开的VDPBF16PS指令的情况下，利用现有指令执行该功能将需要指令序列来转换数据元素，执行乘法，并且利用饱和将乘法结果与目的地数据累加。替代地，所公开实施例在一条指令中实现该功能。

相关浮点格式

[0038] 由所公开实施例使用的16位浮点格式包括bfloat16(由美国加利福尼亚州山景城的谷歌公司定义)和binary16(由电气和电子工程师协会公布为IEEE754-2008)，bfloat16在本文中有时被称为“bf16”或“BF16”，binary16在本文中有时被称为“半精度”或“fp16”。由所公开实施例使用的32位浮点格式包括binary32(也被公布为IEEE754-2008的一部分)，binary32在本文中有时被称为“单精度”或“fp32”。

[0039] 表1列出相关数据格式之间的一些相关特性和区别。如所示，所有三种格式都包括

一个符号位。binary32、binary16和bfloat16具有分别为8位、5位和8位的指数宽度以及分别为24位、11位和8位的有效数(在本文中有时被称为“尾数”或“分数”)位。为了腾出空间以适配有效数,binary32、binary16和bfloat16格式中的每一种为有效数使用一个隐式位,有效数的剩余位被显式地包括。bfloat16相对于fp16的一个优势在于,可以截断fp32数并且具有有效的bfloat16数。

表1

格式	位	符号	指数	有效数
Binary32	32	1	8位	24位
Binary16	16	1	5位	11位
Bfloat16	16	1	8位	8位

[0040] 实现所公开的VDPBF16PS指令的处理器将包括取出电路,该取出电路用于取出指令,该指令具有用于指定操作码、以及第一源向量、第二源向量和目的地向量的位置的字段。至少参考图5、图6A-6B和图7A-7D进一步图示和描述VDPBF16PS指令的格式。所指定的源向量和目的地向量可以位于向量寄存器中或位于存储器中。操作码用于使处理器将所指定的第一源和第二源的N对16位浮点格式化(例如,bfloat16)元素相乘,其中在一些实施例中,N是4、8和16中的任一个(但是本发明不对N设置上限,N可以是32、64或更大,并且与具有1024、2048或更多位的源向量相关联。在一些实施例中,N是大于4的偶数。),并且将所得的乘积中的每一个与所指定的目的地的对应的单精度(即,FP32或binary32)元素的先前内容累加。这种处理器将进一步包括解码电路和执行电路,该解码电路用于对所取出的指令解码,该执行电路用于如操作码所指定地对经解码的指令作出响应。下面至少参考图1-4、图9A-9B和图10A-10B进一步描述和图示执行电路。

[0041] 图1是图示根据一些实施例的用于执行VDPBF16PS指令的处理组件的框图。如所示,存储101存储要执行的(多条)VDPBF16PS指令103。在一些实施例中,计算系统100是用于同时处理诸如矩阵的紧缩数据向量的多个元素的SIMD处理器。

[0042] 在操作中,(多条)VDPBF16PS指令103由取出电路105从存储101取出。如下文进一步所描述,至少参考图2和图5,VDPBF16PS指令具有用于指定操作码以及第一源向量、第二源向量和目的地向量的位置的字段,该操作码用于指示执行电路用于将所指定的第一源和第二源的N对16位浮点格式化元素相乘,并且将所得的乘积与所指定的目的地的对应的单精度元素的先前内容累加。所取出的VDPBF16PS指令107由解码电路109解码。至少参考图5、图6A-6B和图7A-7D进一步图示和描述的VDPBF16PS指令格式具有用于指定第一源向量、第二源向量和目的地向量的位置的字段(此处未示出)。解码电路109将所取出的VDPBF16PS指令107解码为一个或多个操作。在一些实施例中,该解码包括:生成将由执行电路(诸如,执行电路117)执行的多个微操作。解码电路109还对指令后缀和前缀进行解码(如果使用)。具有对寄存器堆和存储器115的访问的执行电路117用于如操作码所指定地对经解码的指令111作出响应,并且在下面至少参考图3-4、图9A-9B和图10A-10B进一步进行描述和图示。

[0043] 在一些实施例中,寄存器重命名、寄存器分配和/或调度电路113提供用于以下一项或多项的功能:1)将逻辑操作数值重命名为物理操作数值(例如,一些实施例中的寄存器别名表);2)将状态位和标志分配给经解码的指令;和3)(例如,在一些实施例中,使用预留站)调度经解码的VDPBF16PS指令111供在指令池外部的执行电路117上执行。

[0044] 在一些实施例中，写回电路119用于写回所执行的指令的结果。写回电路119和寄存器重命名/调度电路113可在不同时刻发生或可根本不发生，在该程度上而言，写回电路119和寄存器重命名/调度电路113是可选的，如由其虚线边界所指示。

[0045] 图2是图示根据实施例的VDPBF16PS指令的执行的框图。如所示，计算装置200(例如，处理器)用于接收、取出并解码(取出和解码电路在此处未示出，但是至少参考图1和图9A-9B图示和描述)VDPBF16PS指令201，该指令包括用于指定操作码202(VDPBF16PS)以及目的地204向量、第一源206向量和第二源208向量的位置的字段。至少参考图5、图6A-6B和图7A-7D进一步图示和描述VDPBF16PS指令201的格式。还示出所指定的第一源向量212A和第二源向量212B、包括乘法器215A、215B和累加器216的执行电路214、以及所指定的目的地向量218。

[0046] 在操作中，计算装置200(例如，处理器)用于使用取出和解码电路(未示出)来取出并解码指令201，该指令具有用于指定操作码202以及第一源206向量、第二源208向量和目的地204向量的位置的字段，该操作码用于指示计算装置(例如，处理器)用于将所指定的第一源和第二源的N对16位浮点(例如，bfloat16或binary16)元素相乘，并且将所得的乘积与所指定的目的地的对应的单精度(例如，binary32)元素的先前内容累加。此处，N被示出为等于8，因此所指定的第一源和第二源是256位向量，这些向量可以位于向量寄存器中或位于存储器中，这些向量包含8对16位浮点格式化数。如至少参考图5、图6A-6B和图7A-7D进一步图示和描述，在其他实施例中，指令201可以指定不同的向量长度，诸如128位、512位或1024位。执行电路214在此处用于如操作码202所指定地对经解码的指令作出响应。在一些实施例中，执行电路214根据需要对乘法器215A和215B以及累加器216的结果执行饱和。在一些实施例中，执行电路在没有饱和的情况下利用无限精度执行乘法，并且在上溢的情况下将累加的结果饱和至正无穷或负无穷并在任何下溢的情况下将累加的结果饱和至零。在一些实施例中，执行电路214用于并行地生成所有N个目的地元素。

[0047] 图3A是图示根据实施例的VDPBF16PS指令的示例性执行的伪代码。如伪代码300中所示，VDPBF16PS函数具有用于指定第一源向量src1和第二源向量src2以及目的地srcdest的字段，第一源向量src1和第二源向量src2可以是128位、256位和512位中的任一个，目的地srcdest还用作用于累加的源。伪代码300还示出对写掩码的使用以控制其中经掩码的元素被归零还是合并以对目的地元素中的每一个进行掩码。(如至少参考图5、图6A-6B和图7A-7D进一步图示和描述，在一些实施例中，VDPBF16PS指令包括用于指定掩码和用于控制归零还是合并的字段。)至少参考图2、图3B、图4和图9A-9B进一步图示和描述VDPBF16PS指令的执行。

[0048] 图3B是图示根据实施例的VDPBF16PS指令的示例性执行的伪代码。如所示，伪代码310类似于伪代码300(图3A)，但是在将偶源元素的乘积与目的地累加之前将奇源元素的乘积与目的地累加。

[0049] 图3C是图示根据实施例的用于与图3A的伪代码一起使用的助手函数的伪代码。此处，伪代码320定义助手函数make_fp32(x)，该助手函数从bfloat16格式转换为binary32格式。代码图示bfloat16格式相对于binary16或半精度格式的至少一个优势，即，转换可以简单地将bfloat16数的16位紧缩到双字的高16位中，而将低16位归零。可以通过仅截断binary32数的低16位来执行逆转换。

[0050] 图4是图示根据实施例的处理器对VDPBF16PS指令作出响应的过程流程图。如流程400中所示，在401处，处理器用于使用取出电路来取出指令，该指令具有用于指定操作码以及第一源向量、第二源向量和目的地向量的位置的字段，该操作码用于指示执行电路用于将所指定的第一源和第二源的N对半精度元素相乘，并且将所得的乘积与所指定的目的地的对应的单精度元素的先前内容累加。在403处，处理器用于使用解码电路对所取出的指令进行解码。在一些实施例中，在405处，处理器用于如操作码所指定地调度经解码的指令的执行。在407处，处理器用于使用执行电路对经解码的指令作出响应。在一些实施例中，执行电路用于根据需要使乘法和累加的结果饱和。在一些实施例中，执行电路在没有饱和的情况下利用无限精度执行乘法，并且在上溢的情况下将累加的结果饱和至正无穷或负无穷并在任何下溢的情况下将累加的结果饱和至零。在一些实施例中，在409处，处理器用于提交所执行的指令的结果。

[0051] 操作405和409可在不同时刻发生或可根本不发生，在该程度上而言，操作405和409是可选的，如由其虚线边界所指示。

[0052] 图5是图示根据实施例的VDPBF16PS指令的格式的框图。如所示，VDPBF16PS指令500包括用于指定操作码502、以及目的地504向量、第一源506向量和第二源508向量的位置的字段。源向量和目的地向量可以各自位于寄存器中或位于存储器中。操作码502的示例为VDPBF16PS*。

[0053] 操作码502被示出为包括星号，该星号表示各种可选字段可以作为前缀或后缀被添加到操作码。即，VDPBF16PS指令500进一步包括用于影响指令行为的可选参数，包括掩码{k}510、归零{Z}512、元素格式514和向量尺寸(N)516。指令修饰符510、512、514和516中的一个或多个可以通过对操作码502使用前缀或后缀来指定。

[0054] 在一些实施例中，可选的指令修饰符510、512、514和516中的一个或多个被编码在任选地被包括在指令500中的立即数字段(未示出)中。例如，可选的指令修饰符510、512、514和516中的每一个可以由32位立即数的不同的8位部分指示。在一些实施例中，可选的指令修饰符510、512、514和516中的一个或多个经由配置寄存器来指定，该配置寄存器诸如被包括在指令集架构中的模型专用寄存器(MSR)。

[0055] 至少参考图6A-6B和图7A-7D进一步图示和描述VDPBF16PS指令500的格式。

指令集

[0056] 指令集可包括一种或多种指令格式。给定的指令格式可定义各种字段(例如，位的数量、位的位置)以指定将要执行的操作(例如，操作码)以及将对其执行该操作的(多个)操作数和/或(多个)其他数据字段(例如，掩码)，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该指令格式的字段(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，因为较少的字段被包括)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。由此，ISA的每一条指令使用给定的指令格式(并且如果经定义，则按照该指令格式的指令模板中的给定的一个指令模板)来表达，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，该特定的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操作数字段中具有选择特定操作数的特定的内容。已经推出和/或发布了被称为

高级向量扩展(AVX)(AVX1和AVX2)和利用向量扩展(VEX)编码方案的SIMD扩展集(参见例如2014年9月的英特尔®64和IA-32架构软件开发者手册;并且参见2014年10月的英特尔®高级向量扩展编程参考)。

示例性指令格式

[0057] 本文中所描述的(多条)指令的实施例能以不同的格式体现。另外,在下文中详述示例性系统、架构和流水线。(多条)指令的实施例可在此类系统、架构和流水线上执行,但是不限于详述的那些系统、架构和流水线。

通用向量友好指令格式

[0058] 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,但是替代实施例仅使用通过向量友好指令格式的向量操作。

[0059] 图6A-图6B是图示根据本发明的一些实施例的通用向量友好指令格式及其指令模板的框图。图6A是图示根据本发明的一些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6B是图示根据本发明的一些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600定义A类和B类指令模板,这两者都包括无存储器访问605的指令模板和存储器访问620的指令模板。在向量友好指令格式的上下文中的术语"通用"是指不束缚于任何特定指令集的指令格式。

[0060] 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: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个双字尺寸的元素组成,或者替代地由8个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);但是替代实施例可支持更大、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)。

[0061] 图6A中的A类指令模板包括:1)在无存储器访问605的指令模板内,示出无存储器访问的完全舍入控制型操作610的指令模板、以及无存储器访问的数据变换型操作615的指令模板;以及2)在存储器访问620的指令模板内,示出存储器访问的时效性625的指令模板和存储器访问的非时效性630的指令模板。图6B中的B类指令模板包括:1)在无存储器访问605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612的指令模板以及无存储器访问的写掩码控制的vsize型操作617的指令模板;以及2)在存储器访问620的指令模板内,示出存储器访问的写掩码控制627的指令模板。

[0062] 通用向量友好指令格式600包括以下列出的按照在图6A-6B中图示的顺序的如下字段。

[0063] 格式字段640——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。

[0064] 基础操作字段642——其内容区分不同的基础操作。

[0065] 寄存器索引字段644——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)寄存器堆中选择N个寄存器。尽管在一个实施例中N可多达三个源寄存器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,可支持多达两个源,其中这些源中的一个源还用作目的地;可支持多达三个源,其中这些源中的一个源还用作目的地;可支持多达两个源和一个目的地)。

[0066] 修饰符(modifier)字段646——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605的指令模板与存储器访问620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。

[0067] 扩充操作字段650——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。在一些实施例中,该字段被分成类字段668、α字段652和β字段654。扩充操作字段650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。

[0068] 比例字段660——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)的索引字段的内容的按比例缩放。

[0069] 位移字段662A——其内容用作存储器地址生成的一部分(例如,用于使用($2^{比例}$*索引+基址+位移)的地址生成)。

[0070] 位移因数字段662B(注意,位移字段662A直接在位移因数字段662B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;它指定将按比例缩放存储器访问的尺寸(N)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尺寸(N)以生成将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674(稍后在本文中描述)和数据操纵字段654C确定。位移字段662A和位移因数字段662B不用于无存储器访问605的指令模板和/或不同的实施例可实现这两者中的仅一个或不实现这两者中的任一个,在这个意义上,位移字段662A和位移因数字段662B是任选的。

[0071] 数据元素宽度字段664——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;在其他实施例中只用于指令中的一些指令)。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。

[0072] 写掩码字段670——其内容逐数据元素位置地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-写掩码和归零-写掩码两者。当合并时,向量掩码允许在执行(由基础操作和扩充操作指定的)任何操作期间保护目的地中的任何元素集免于更新;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

(由基础操作和扩充操作指定的)任何操作期间使目的地中的任何元素集归零;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正被执行的操作的向量长度的能力(即,从第一个到最后一个正被修改的元素的跨度),然而,被修改的元素不一定要是连续的。由此,写掩码字段670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670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670的内容间接地标识要执行的掩码)的本发明的实施例,但是替代实施例替代地或附加地允许掩码写字段670的内容直接指定要执行的掩码。

[0073] 立即数字段672——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。

[0074] 类字段668——其内容在不同类的指令之间进行区分。参考图6A-图6B,该字段的内容在A类和B类指令之间进行选择。在图6A-图6B中,圆角方形用于指示特定的值存在于字段中(例如,在图6A-图6B中分别用于类字段668的A类668A和B类668B)。

A类指令模板

[0075] 在A类非存储器访问605的指令模板的情况下,α字段652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610和无存储器访问的数据变换型操作615的指令模板分别指定舍入652A.1和数据变换652A.2)的RS字段652A,而β字段654区分要执行所指定类型的操作中的哪一种。在无存储器访问605的指令模板中,比例字段660、位移字段662A和位移比例字段662B不存在。

无存储器访问的指令模板——完全舍入控制型操作

[0076] 在无存储器访问的完全舍入控制型操作610的指令模板中,β字段654被解释为其(多个)内容提供静态舍入的舍入控制字段654A。尽管在本发明的所述实施例中舍入控制字段654A包括抑制所有浮点异常(SAE)字段656和舍入操作控制字段658,但是替代实施例可支持这两个概念,可将这两个概念编码为同一字段,或仅具有这些概念/字段中的一个或另一个(例如,可仅具有舍入操作控制字段658)。

[0077] SAE字段656——其内容区分是否禁用异常事件报告;当SAE字段656的内容指示启用抑制时,给定的指令不报告任何种类的浮点异常标志,并且不唤起任何浮点异常处置程序。

[0078] 舍入操作控制字段658——其内容区分要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658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一些实施例中,舍入操作控制字段650的内容覆盖(override)该寄存器值。

无存储器访问的指令模板－数据变换型操作

[0079] 在无存储器访问的数据变换型操作615的指令模板中,β字段654被解释为数据变换字段654B,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、混合、广播)。

[0080] 在A类存储器访问620的指令模板的情况下,α字段652被解释为驱逐提示字段652B,其内容区分要使用驱逐提示中的哪一个(在图6A中,对于存储器访问时效性625的指令模板和存储器访问非时效性630的指令模板分别指定时效性的652B.1和非时效性的

652B.2)，而β字段654被解释为数据操纵字段654C，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620的指令模板包括比例字段660，并任选地包括位移字段662A或位移比例字段662B。

[0081] 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以及向存储器的向量存储。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从/向存储器传输数据，其中实际被传输的元素由被选为写掩码的向量掩码的内容规定。

存储器访问的指令模板——时效性的

[0082] 时效性的数据是可能足够快地被重新使用以从高速缓存操作受益的数据。然而，这是提示，并且不同的处理器能以不同的方式实现它，包括完全忽略该提示。

存储器访问的指令模板——非时效性的

[0083] 非时效性的数据是不太可能足够快地被重新使用以从第一级高速缓存中的高速缓存操作受益且应当被给予驱逐优先级的数据。然而，这是提示，并且不同的处理器能以不同的方式实现它，包括完全忽略该提示。

B类指令模板

[0084] 在B类指令模板的情况下，α字段652被解释为写掩码控制(Z)字段652C，其内容区分由写掩码字段670控制的写掩码应当是合并还是归零。

[0085] 在B类非存储器访问605的指令模板的情况下，β字段654的一部分被解释为RL字段657A，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612的指令模板和无存储器访问的写掩码控制VSIZE型操作617的指令模板分别指定舍入657A.1和向量长度(VSIZE)657A.2)，而β字段654的其余部分区分要执行所指定类型的操作中的哪一种。在无存储器访问605的指令模板中，比例字段660、位移字段662A和位移比例字段662B不存在。

[0086] 在无存储器访问的写掩码控制部分舍入控制型操作610的指令模板中，β字段654的其余部分被解释为舍入操作字段659A，并且禁用异常事件报告(给定的指令不报告任何种类的浮点异常标志，并且不唤起任何浮点异常处置程序)。

[0087] 舍入操作控制字段659A——正如舍入操作控制字段658，其内容区分要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659A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一些实施例中，舍入操作控制字段650的内容覆盖该寄存器值。

[0088] 在无存储器访问的写掩码控制VSIZE型操作617的指令模板中，β字段654的其余部分被解释为向量长度字段659B，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)。

[0089] 在B类存储器访问620的指令模板的情况下，β字段654的一部分被解释为广播字段657B，其内容区分是否要执行广播型数据操纵操作，而β字段654的其余部分被解释为向量长度字段659B。存储器访问620的指令模板包括比例字段660，并任选地包括位移字段662A或位移比例字段662B。

[0090] 针对通用向量友好指令格式600，示出完整操作码字段674包括格式字段640、基础操作字段642和数据元素宽度字段664。尽管示出了其中完整操作码字段674包括所有这些

字段的一个实施例，但是在不支持所有这些字段的实施例中，完整操作码字段674包括少于所有的这些字段。完整操作码字段674提供操作代码(操作码)。

[0091] 扩充操作字段650、数据元素宽度字段664和写掩码字段670允许逐指令地以通用向量友好指令格式指定这些特征。

[0092] 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。

[0093] 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，不同处理器或处理器内的不同核可支持仅A类、仅B类、或者可支持这两类。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，并且旨在用于通用计算和图形和/或科学(吞吐量)计算两者的核可支持A类和B类两者(当然，具有来自这两类的模板和指令的一些混合、但是并非来自这两类的所有模板和指令的核在本发明的范围内)。同样，单个处理器可包括多个核，这多个核全部都支持相同的类，或者其中不同的核支持不同的类。举例而言，在具有单独的图形核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。将使以高级语言编写的程序成为(例如，及时编译或静态编译)各种不同的可执行形式，这些可执行形式包括：1)仅具有由用于执行的目标处理器支持的(多个)类的指令的形式；或者2)具有替代例程并具有控制流代码的形式，该替代例程使用所有类的指令的不同组合来编写，该控制流代码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来执行。

示例性专用向量友好指令格式

[0094] 图7A是图示根据本发明的一些实施例的示例性专用向量友好指令格式的框图。图7A示出专用向量友好指令格式700，其指定各字段的位置、尺寸、解释和次序、以及那些字段中的一些字段的值，在这个意义上，该专用向量友好指令格式700是专用的。专用向量友好指令格式700可用于扩展x86指令集，并且由此字段中的一些字段与如在现有的x86指令集及其扩展(例如，AVX)中所使用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、位移字段和立即数字段一致。图示来自图6A-图6B的字段，来自图7A的字段映射到来自图6A-图6的字段。

[0095] 应当理解，虽然出于说明的目的在通用向量友好指令格式600的上下文中参考专用向量友好指令格式700描述了本发明的实施例，但是本发明不限于专用向量友好指令格式700，除非另有声明。例如，通用向量友好指令格式600构想了各种字段的各种可能的尺寸，而专用向量友好指令格式700示出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700中数据元素宽度字段664被图示为一位字段，但是本发明不限于此(即，通用向量友好指令格式600构想数据元素宽度字段664的其他尺寸)。

[0096] 通用向量友好指令格式600包括以下列出的按照图7A中图示的顺序的如下字段。

[0097] EVEX前缀(字节0-3)702——以四字节形式进行编码。

[0098] 格式字段640(EVEX字节0，位[7:0])——第一字节(EVEX字节0)是格式字段640，并

且它包含0x62(在一些实施例中,为用于区分向量友好指令格式的唯一值)。

[0099] 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。

[0100] REX字段705(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(657BEX字节1,位[5]-B)组成。EVEX.R、EVEX.X和EVEX.B位字段提供与对应的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和bbb)进行编码,由此可通过对EVEX.R、EVEX.X和EVEX.B相加来形成Rrrr、Xxxx和Bbbb。

[0101] REX'字段710A——这是REX'字段710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')。在一些实施例中,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BOUND指令进行区分,该BOUND指令的实操作码字节是62,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以下其他指示的位。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。

[0102] 操作码映射字段715(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。

[0103] 数据元素宽度字段664(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)。

[0104] EVEX.vvvv 720(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(1补码)形式指定的第一源寄存器操作数进行编码,并且对具有两个或更多个源操作数的指令有效;2)EVEX.vvvv对针对特定向量位移以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,该字段被预留,并且应当包含1111b。由此,EVEX.vvvv字段720对以反转(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。

[0105] EVEX.U 668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。

[0106] 前缀编码字段725(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前缀仅需要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;并且在运行时在被提供给解码器的PLA之前被扩展成传统SIMD前缀(因此,在无需修改的情况下,PLA既可执行传统格式的这些传统指令又可执行EVEX格式的这些传统指令)。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,特定实施例以类似的方式扩展,但允许由这些传统SIMD前缀指定的不同含义。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。

[0107] α字段652(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图示)——如先前所述,该字段是针对上下文的。

[0108] β字段654(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图示)——如前所述,此字段是针对上下文的。

[0109] REX'字段710B——这是REX'字段710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。

[0110] 写掩码字段670(EVEX字节3,位[2:0]-kkk)——其内容指定写掩码寄存器中的寄存器的索引,如先前所述。在一些实施例中,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(这能以各种方式实现,包括使用硬连线到所有对象的写掩码或绕过掩码硬件的硬件来实现)。

[0111] 实操作码字段730(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。

[0112] MOD R/M字段740(字节5)包括MOD字段742、Reg字段744和R/M字段746。如先前所述的,MOD字段742的内容将存储器访问操作和非存储器访问操作区分开。Reg字段744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,并且不用于对任何指令操作数进行编码。R/M字段746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。

[0113] 比例、索引、基址(SIB)字节(字节6)——如先前所述的,比例字段650的内容用于存储器地址生成。SIB.xxx 754和SIB.bbb 756——先前已经针对寄存器索引Xxxx和Bbbb提及了这些字段的内容。

[0114] 位移字段662A(字节7-10)——当MOD字段742包含10时,字节7-10是位移字段662A,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。

[0115] 位移因数字段662B(字节7)——当MOD字段742包含01时,字节7是位移因数字段662B。该字段的位置与以字节粒度工作的传统x86指令集8位位移(disp8)的位置相同。由于disp8是符号扩展的,因此它仅能在-128和127字节偏移之间寻址;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662B是disp8的重新解释;当使用位移因数字段662B时,通过将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(单个字节用于位移,但具有大得多的范围)。此类经压缩的位移基于有效位移是存储器访问的粒度的倍数的假设,并且由此地址偏移的冗余低阶位不需要被编码。换句话说,位移因数字段662B替代传统x86指令集8位位移。由此,位移因数字段662B以与x86指令集8位位移相同的方式被编码(因此,在ModRM/SIB编码规则中没有变化),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方面没有变化,而仅在有硬件对位移值的解释方面有变化(这需要将位移按比例缩放存储器操作数的尺寸以获得字节式地址偏移)。立即数字段672如先前所述地操作。

完整操作码字段

[0116] 图7B是图示根据一些实施例的构成完整操作码字段674的具有专用向量友好指令

格式700的字段的框图。具体地，完整操作码字段674包括格式字段640、基础操作字段642和数据元素宽度(W)字段664。基础操作字段642包括前缀编码字段725、操作码映射字段715和实操作码字段730。

寄存器索引字段

[0117] 图7C是图示根据一些实施例的构成寄存器索引字段644的具有专用向量友好指令格式700的字段的框图。具体地，寄存器索引字段644包括REX字段705、REX'字段710、MODR/M.reg字段744、MODR/M.r/m字段746、VVVV字段720、xxx字段754和bbb字段756。

扩充操作字段

[0118] 图7D是图示根据一些实施例的构成扩充操作字段650的具有专用向量友好指令格式700的字段的框图。当类(U)字段668包含0时，它表明EVEX.U0(A类668A)；当它包含1时，它表明EVEX.U1(B类668B)。当U＝0且MOD字段742包含11(表明无存储器访问操作)时，α字段652(EVEX字节3，位[7]-EH)被解释为rs字段652A。当rs字段652A包含1(舍入652A.1)时，β字段654(EVEX字节3，位[6:4]-SSS)被解释为舍入控制字段654A。舍入控制字段654A包括一位SAE字段656和两位舍入操作字段658。当rs字段652A包含0(数据变换652A.2)时，β字段654(EVEX字节3，位[6:4]-SSS)被解释为三位数据变换字段654B。当U＝0且MOD字段742包含00、01或10(表明存储器访问操作)时，α字段652(EVEX字节3，位[7]-EH)被解释为驱逐提示(EH)字段652B，并且β字段654(EVEX字节3，位[6:4]-SSS)被解释为三位数据操纵字段654C。

[0119] 当U＝1时，α字段652(EVEX字节3，位[7]-EH)被解释为写掩码控制(Z)字段652C。当U＝1且MOD字段742包含11(表明无存储器访问操作)时，β字段654的一部分(EVEX字节3，位[4]-S_0)被解释为RL字段657A；当它包含1(舍入657A.1)时，β字段654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659A，而当RL字段657A包含0(VSIZE657.A2)时，β字段654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659B(EVEX字节3，位[6-5]-L_{1-0})。当U＝1且MOD字段742包含00、01或10(表明存储器访问操作)时，β字段654(EVEX字节3，位[6:4]-SSS)被解释为向量长度字段659B(EVEX字节3，位[6-5]-L_{1-0})和广播字段657B(EVEX字节3，位[4]-B)。

示例性寄存器架构

[0120] 图8是根据一些实施例的寄存器架构800的框图。在所图示的实施例中，有32个512位宽的向量寄存器810；这些寄存器被引用为zmm0到zmm31。较低的16个zmm寄存器的较低阶256个位覆盖(overlay)在寄存器ymm0-16上。较低的16个zmm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700对这些被覆盖的寄存器堆操作，如在以下表格中所图示。

可调节向量长度	类	操作	寄存器
不包括向量长度字段659B的指令模板	A(图6A；U=0)	610、615、625、630	zmm寄存器(向量长度是64字节)

	B（图6B;U=1)	612	zmm寄存器(向量长度是64字节)
包括向量长度字段659B的指令模板	B（图6B;U=1)	617、627	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659B

[0121] 换句话说,向量长度字段659B在最大长度与一个或多个其他较短长度之间进行选择,其中每一个此类较短长度是前一长度的一半,并且不具有向量长度字段659B的指令模板在最大向量长度上操作。此外,在一个实施例中,专用向量友好指令格式700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;取决于实施例,较高阶数据元素位置要么保持与在指令之前相同,要么归零。

[0122] 写掩码寄存器815——在所图示的实施例中,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。在替代实施例中,写掩码寄存器815的尺寸是16位。如先前所述,在一些实施例中,向量掩码寄存器k0无法用作写掩码;当将正常指示k0的编码用作写掩码时,它选择硬连线的写掩码0xFFFF,从而有效地禁止写掩码用于那条指令。

[0123] 通用寄存器825——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寻址模式一起使用以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。

[0124] 标量浮点栈寄存器堆(x87栈)845,在其上面重叠了MMX紧缩整数平坦寄存器堆850——在所图示的实施例中,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与XMM寄存器之间执行的一些操作保存操作数。

[0125] 替代实施例可以使用更宽的或更窄的寄存器。另外,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。

示例性核架构、处理器和计算机架构

[0126] 处理器核能以不同方式、出于不同的目的、在不同的处理器中实现。例如,此类核的实现可以包括: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)CPU,其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;以及2)协处理器,其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。此类不同的处理器导致不同的计算机系统架构,这些计算机系统架构可包括:1)在与CPU分开的芯片上的协处理器;2)在与CPU相同的封装中但在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,此类协处理器有时被称为专用逻辑或被称为专用核,该专用逻辑诸如,集成图形和/或科学(吞吐量)逻辑);以及4)芯片上系统,其可以将所描述的CPU(有时被称为(多个)应用核或(多个)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。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。

示例性核架构

有序和乱序核框图

[0127] 图9A是图示根据本发明的一些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9B是示出根据本发明的一些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。图9A-图9B中的实线框图示有序流水线和有序核，而虚线框的任选增加图示寄存器重命名的、乱序发布/执行流水线和核。考虑到有序方面是乱序方面的子集，将描述乱序方面。

[0128] 在图9A中，处理器流水线900包括取出级902、长度解码级904、解码级906、分配级908、重命名级910、调度(也被称为分派或发布)级912、寄存器读取/存储器读取级914、执行级916、写回/存储器写入级918、异常处置级922和提交级924。

[0129] 图9B示出处理器核990，该处理器核990包括前端单元930，该前端单元930耦合到执行引擎单元950，并且前端单元930和执行引擎单元950两者都耦合到存储器单元970。核990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、或混合或替代的核类型。作为又一选项，核990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、通用计算图形处理单元(GPGPU)核、图形核，等等。

[0130] 前端单元930包括分支预测单元932，该分支预测单元932耦合到指令高速缓存单元934，该指令高速缓存单元934耦合到指令转换后备缓冲器(TLB)936，该指令转换后备缓冲器936耦合到指令取出单元938，该指令取出单元938耦合到解码单元940。解码单元940(或解码器)可对指令解码，并且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940可使用各种不同的机制来实现。合适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990包括存储用于某些宏指令的微代码的微代码ROM或其他介质(例如，在解码单元940中，或以其他方式在前端单元930内)。解码单元940耦合到执行引擎单元950中的重命名/分配器单元952。

[0131] 执行引擎单元950包括重命名/分配器单元952，该重命名/分配器单元952耦合到引退单元954和一个或多个调度器单元的集合956。(多个)调度器单元956表示任何数量的不同调度器，包括预留站、中央指令窗等。(多个)调度器单元956耦合到(多个)物理寄存器堆单元958。(多个)物理寄存器堆单元958中的每一个物理寄存器堆单元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，作为要执行的下一条指令的地址的指令指针)等等。在一个实施例中，(多个)物理寄存器堆单元958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。(多个)物理寄存器堆单元958由引退单元954重叠，以图示可实现寄存器重命名和乱序执行的各种方式(例如，使用(多个)重排序缓冲器和(多个)引退寄存器堆；使用(多个)未来文件、(多个)历史缓冲器、(多个)引退寄存器堆；使用寄存器映射和寄存器池，等等)。引退单元954和(多个)物理寄存器堆单元958耦合到(多个)执行集群960。(多个)执行集群960包括一个或多个执行单元的集合962以及一个或多个存储器访问

单元的集合964。执行单元962可执行各种操作(例如,移位、加法、减法、乘法)并可对各种数据类型(例如,标量浮点、紧缩整数、紧缩浮点、向量整数、向量浮点)执行。尽管一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,但是其他实施例可包括仅一个执行单元或全都执行所有功能的多个执行单元。(多个)调度器单元956、(多个)物理寄存器堆单元958和(多个)执行集群960示出为可能有多个,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,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/或各自具有其自身的调度器单元、(多个)物理寄存器堆单元和/或执行集群的存储器访问流水线——并且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集群具有(多个)存储器访问单元964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,并且其余流水线可以是有序的。

[0132] 存储器访问单元的集合964耦合到存储器单元970,该存储器单元970包括数据TLB单元972,该数据TLB单元972耦合到数据高速缓存单元974,该数据高速缓存单元974耦合到第二级(L2)高速缓存单元976。在一个示例性实施例中,存储器访问单元964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到存储器单元970中的数据TLB单元972。指令高速缓存单元934还耦合到存储器单元970中的第二级(L2)高速缓存单元976。L2高速缓存单元976耦合到一个或多个其他级别的高速缓存,并最终耦合到主存储器。

[0133] 作为示例,示例性寄存器重命名的乱序发布/执行核架构可如下所述地实现流水线900:1)指令取出938执行取出级902和长度解码级904;2)解码单元940执行解码级906;3)重命名/分配器单元952执行分配级908和重命名级910;4)(多个)调度器单元956执行调度级912;5)(多个)物理寄存器堆单元958和存储器单元970执行寄存器读取/存储器读取级914;执行集群960执行执行级916;6)存储器单元970和(多个)物理寄存器堆单元958执行写回/存储器写入级918;7)各单元可牵涉到异常处置级922;以及8)引退单元954和(多个)物理寄存器堆单元958执行提交级924。

[0134] 核990可支持一个或多个指令集(例如,x86指令集(具有已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;加利福尼亚州桑尼维尔市的ARM控股公司的ARM指令集(具有诸如NEON的任选的附加扩展)),其中包括本文中描述的(多条)指令。在一个实施例中,核990包括用于支持紧缩数据指令集扩展(例如,AVX1、AVX2)的逻辑,由此允许使用紧缩数据来执行由许多多媒体应用使用的操作。

[0135] 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,各种方式包括时分多线程化、同时多线程化(其中单个物理核为物理核正在同时多线程化的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的诸如英特尔®超线程化技术中的同时多线程化)。

[0136] 尽管在乱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。尽管所图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934/974以及共享的L2高速缓存单元976,但是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,第一级(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,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。

具体的示例性有序核架构

[0137] 图10A-图10B图示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(包括相同类型和/或不同类型的其他核)中的一个逻辑块。取决于应用，逻辑块通过高带宽互连网络(例如，环形网络)与一些固定的功能逻辑、存储器I/O接口和其他必要的I/O逻辑进行通信。

[0138] 图10A是根据本发明的一些实施例的单个处理器核以及它至管芯上互连网络1002的连接及其第二级(L2)高速缓存的本地子集1004的框图。在一个实施例中，指令解码器1000支持具有紧缩数据指令集扩展的x86指令集。L1高速缓存1006允许对进入标量和向量单元中的、对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1008和向量单元1010使用分开的寄存器集合(分别为标量寄存器1012和向量寄存器1014)，并且在这些寄存器之间传输的数据被写入到存储器，并随后从第一级(L1)高速缓存1006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)。

[0139] L2高速缓存的本地子集1004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，每个处理器核一个本地子集。每个处理器核具有到其自身的L2高速缓存的本地子集1004的直接访问路径。由处理器核读取的数据被存储在其L2高速缓存子集1004中，并且可以与其他处理器核访问其自身的本地L2高速缓存子集并行地被快速访问。由处理器核写入的数据被存储在其自身的L2高速缓存子集1004中，并在必要的情况下从其他子集转储清除。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高速缓存和其他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。

[0140] 图10B是根据本发明的一些实施例的图10A中的处理器核的一部分的展开图。图10B包括L1高速缓存1004的L1数据高速缓存1006A部分，以及关于向量单元1010和向量寄存器1014的更多细节。具体地，向量单元1010是16宽向量处理单元(VPU)(见16宽ALU 1028)，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通过混合单元1020支持对寄存器输入的混合，通过数值转换单元1022A-B支持数值转换，并且通过复制单元1024支持对存储器输入的复制。写掩码寄存器1026允许掩蔽所得的向量写入。

[0141] 图11是根据本发明的一些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1100的框图。图11中的实线框图示具有单个核1102A、系统代理1110、一个或多个总线控制器单元的集合1116的处理器1100，而虚线框的任选增加图示具有多个核1102A-N、系统代理单元1110中的一个或多个集成存储器控制器单元的集合1114以及专用逻辑1108的替代处理器1100。

[0142] 因此，处理器1100的不同实现可包括：1)CPU，其中专用逻辑1108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1102A-N是一个或多个通用核(例如，通用有序核、通用乱序核、这两者的组合)；2)协处理器，其中核1102A-N是旨在主要用于图形和/或科学(吞吐量)的大量专用核；以及3)协处理器，其中核1102A-N是大量通用有序核。因此，处理器1100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)、嵌入式处理器，等等。该处理器可以被实现在一个或多个芯片上。

处理器1100可以是一个或多个基板的一部分，和/或可使用多种工艺技术(诸如例如，BiCMOS、CMOS、或NMOS)中的任何技术被实现在一个或多个基板上。

[0143] 存储器层次结构包括核内的一个或多个级别的高速缓存、一个或多个共享高速缓存单元的集合1106、以及耦合到集成存储器控制器单元的集合1114的外部存储器(未示出)。共享高速缓存单元的集合1106可包括一个或多个中间级别的高速缓存，诸如，第二级(L2)、第三级(L3)、第四级(L4)或其他级别的高速缓存、末级高速缓存(LLC)和/或以上各项的组合。虽然在一个实施例中，基于环的互连单元1112将集成图形逻辑1108(集成图形逻辑1108是专用逻辑的示例并且在本文中也称为专用逻辑)、共享高速缓存单元的集合1106以及系统代理单元1110/(多个)集成存储器控制器单元1114互连，但是替代实施例可使用任何数量的公知技术来互连此类单元。在一个实施例中，在一个或多个高速缓存单元1106与核1102A-N之间维持一致性。

[0144] 在一些实施例中，一个或多个核1102A-N能够实现多线程化。系统代理1110包括协调和操作核1102A-N的那些部件。系统代理单元1110可包括例如功率控制单元(PCU)和显示单元。PCU可以是对核1102A-N以及集成图形逻辑1108的功率状态进行调节所需的逻辑和部件，或可包括这些逻辑和部件。显示单元用于驱动一个或多个外部连接的显示器。

[0145] 核1102A-N在架构指令集方面可以是同构的或异构的；即，核1102A-N中的两个或更多个核可能能够执行相同的指令集，而其他核可能能够执行该指令集的仅仅子集或不同的指令集。

示例性计算机架构

[0146] 图12-15是示例性计算机架构的框图。本领域中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他电子设备的其他系统设计和配置也是合适的。一般地，能够包含如本文中所公开的处理器和/或其他执行逻辑的各种各样的系统或电子设备一般都是合适的。

[0147] 现在参考图12，所示出的是根据本发明一个实施例的系统1200的框图。系统1200可以包括一个或多个处理器1210、1215，这些处理器耦合到控制器中枢1220。在一个实施例中，控制器中枢1220包括图形存储器控制器中枢(GMCH)1290和输入/输出中枢(IOH)1250(其可以在分开的芯片上)；GMCH 1290包括存储器和图形控制器，存储器1240和协处理器1245耦合到该存储器和图形控制器；IOH 1250将输入/输出(I/O)设备1260耦合到GMCH 1290。或者，存储器和图形控制器中的一个或这两者被集成在(如本文中所描述的)处理器内，存储器1240和协处理器1245直接耦合到处理器1210，并且控制器中枢1220与IOH 1250处于单个芯片中。

[0148] 附加的处理器1215的任选性在图12中通过虚线来表示。每一处理器1210、1215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1800的某一版本。

[0149] 存储器1240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1220经由诸如前端总线(FSB)之类的多分支总线、诸如快速路径互连(QPI)之类的点对点接口、或者类似的连接1295来与(多个)处理器1210、1215进行通信。

[0150] 在一个实施例中，协处理器1245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、嵌入式处理器，等等。在一个实施例中，控制器中枢1220可以包括集成图形加速器。

[0151] 在物理资源1210、1215之间可以存在包括架构、微架构、热、功耗特性等一系列品质度量方面的各种差异。

[0152] 在一个实施例中，处理器1210执行控制一般类型的数据处理操作的指令。嵌入在这些指令内的可以是协处理器指令。处理器1210将这些协处理器指令识别为具有应当由附连的协处理器1245执行的类型。因此，处理器1210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1245。(多个)协处理器1245接受并执行所接收的协处理器指令。

[0153] 现在参见图13，所示出的是根据本发明的实施例的第一更具体的示例性系统1300的框图。如图13中所示，多处理器系统1300是点对点互连系统，并且包括经由点对点互连1350耦合的第一处理器1370和第二处理器1380。处理器1370和1380中的每一个都可以是处理器1100的某一版本。在一些实施例中，处理器1370和1380分别是处理器1210和1215，而协处理器1338是协处理器1245。在另一实施例中，处理器1370和1380分别是处理器1210和协处理器1245。

[0154] 处理器1370和1380示出为分别包括集成存储器控制器(IMC)单元1372和1382。处理器1370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1376和1378；类似地，第二处理器1380包括P-P接口1386和1388。处理器1370、1380可以经由使用点对点(P-P)接口电路1378、1388的P-P接口1350来交换信息。如图13中所示，IMC 1372和1382将处理器耦合到相应的存储器，即存储器1332和存储器1334，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。

[0155] 处理器1370、1380可各自经由使用点对点接口电路1376、1394、1386、1398的各个P-P接口1352、1354来与芯片组1390交换信息。芯片组1390可以任选地经由高性能接口1392来与协处理器1338交换信息。在一个实施例中，协处理器1338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、嵌入式处理器，等等。

[0156] 共享高速缓存(未示出)可被包括在任一处理器中，或在这两个处理器的外部但经由P-P互连与这些处理器连接，使得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共享高速缓存中。

[0157] 芯片组1390可以经由接口1396耦合到第一总线1316。在一个实施例中，第一总线1316可以是外围部件互连(PCI)总线或诸如PCI快速总线或另一第三代I/O互连总线之类的总线，但是本发明的范围不限于此。

[0158] 如图13中所示，各种I/O设备1314可连同总线桥1318一起耦合到第一总线1316，该总线桥1318将第一总线1316耦合到第二总线1320。在一个实施例中，诸如协处理器、高吞吐量MIC处理器、GPGPU、加速器(诸如例如，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的一个或多个附加处理器1315耦合到第一总线1316。在一个实施例中，第二总线1320可以是低引脚数(LPC)总线。在一个实施例中，各种设备可耦合到第二总线1320，这些设备包括例如键盘和/或鼠标1322、通信设备1327以及存储单元1328，该存储单元1328诸如可包括指令/代码和数据1330的盘驱动器或者其他大容量存储设备。此

外，音频I/O 1324可以被耦合到第二总线1320。注意，其他架构是可能的。例如，代替图13的点对点架构，系统可以实现多分支总线或其他此类架构。

[0159] 现在参考图14，示出的是根据本发明的实施例的第二更具体的示例性系统1400的框图。图13和14中的类似元件使用类似的附图标记，并且从图14中省略了图13的某些方面以避免混淆图14的其他方面。

[0160] 图14图示处理器1370、1380可分别包括集成存储器和I/O控制逻辑(“CL”)1472和1482。因此，CL 1472、1482包括集成存储器控制器单元，并包括I/O控制逻辑。图14图示不仅存储器1332、1334耦合到CL 1472、1482，而且I/O设备1414也耦合到控制逻辑1472、1482。传统I/O设备1415被耦合到芯片组1390。

[0161] 现在参考图15，示出的是根据本发明的实施例的SoC 1500的框图。图11中的类似要素使用类似的附图标记。另外，虚线框是更先进的SoC上的任选的特征。在图15中，(多个)互连单元1502被耦合到：应用处理器1510，其包括一个或多个核的集合1102A-N以及(多个)共享高速缓存单元1106，一个或多个核的集合1102A-N包括高速缓存单元1104A-N；系统代理单元1110；(多个)总线控制器单元1116；(多个)集成存储器控制器单元1114；一个或多个协处理器的集合1520，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1530；直接存储器访问(DMA)单元1532；以及用于耦合到一个或多个外部显示器的显示单元1540。在一个实施例中，(多个)协处理器1520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。

[0162] 本文公开的机制的各实施例可以被实现在硬件、软件、固件或此类实现方式的组合中。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。

[0163] 可将程序代码(诸如，图13中图示的代码1330)应用于输入指令，以执行本文中描述的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，处理系统包括具有处理器的任何系统，该处理器诸如例如，数字信号处理器(DSP)、微控制器、专用集成电路(ASIC)或微处理器。

[0164] 程序代码可以用高级的面向过程的编程语言或面向对象的编程语言来实现，以便与处理系统通信。如果需要，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的编程语言的范围。在任何情况下，该语言可以是编译语言或解释语言。

[0165] 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，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造用于执行本文中所述的技术的逻辑。被称为“IP核”的此类表示可以被存储在有形的机器可读介质上，并可被供应给各个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。

[0166] 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、有形布置，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)、可重写紧致盘(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)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。

[0167] 因此，本发明的实施例还包括非暂态的有形机器可读介质，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。

仿真(包括二进制变换、代码变形等)

[0168] 在一些情况下，指令转换器可用于将指令从源指令集转换至目标指令集。例如，指令转换器可以将指令变换(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、仿真或以其他方式转换成要由核处理的一条或多条其他指令。指令转换器可以用软件、硬件、固件、或其组合来实现。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。

[0169] 图16是根据本发明的一些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。在所图示的实施例中，指令转换器是软件指令转换器，但替代地，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16示出可使用x86编译器1604来编译高级语言1602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1616原生执行的x86二进制代码1606。具有至少一个x86指令集核的处理器1616表示通过兼容地执行或以其他方式处理以下各项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实质部分，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的应用或其他软件的目标代码版本。x86编译器1604表示可操作用于生成x86二进制代码1606(例如，目标代码)的编译器，该二进制代码可通过或不通过附加的链接处理在具有至少一个x86指令集核的处理器1616上执行。类似地，图16示出可以使用替代的指令集编译器1608来编译高级语言1602形式的程序，以生成可以由不具有至少一个x86指令集核的处理器1614(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/或执行加利福尼亚州桑尼维尔市的ARM控股公司的ARM指令集的核的处理器)原生执行的替代的指令集二进制代码1610。指令转换器1612用于将x86二进制代码1606转换成可以由不具有x86指令集核的处理器1614原生执行的代码。该转换后的代码不大可能与替代的指令集二进制代码1610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，并且由来自替代指令集的指令构成。因此，指令转换器1612通过仿真、模拟或任何其他过程来表示允许不具有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1606的软件、固件、硬件或其组合。

进一步的示例

[0170] 示例1提供一种示例性处理器，该示例性处理器包括：取出电路，用于取出指令，该指令具有用于指定操作码以及第一源向量、第二源向量和目的地向量的位置的字段，该操作码用于指示执行电路用于将所指定的第一源和第二源的N对16位浮点格式化元素相乘，并且将所得的乘积与所指定的目的地的对应的单精度元素的先前内容累加；解码电路，用于对所取出的指令解码；以及执行电路，用于如该操作码所指定地对经解码的指令作出响应。

[0171] 示例2包括示例1的示例性处理器的实质内容，其中所指定的源向量和目的地向量中的每一个的位置在寄存器中或在存储器中。

[0172] 示例3包括示例1的示例性处理器的实质内容，其中16位浮点格式包括符号位、8位的指数、以及尾数，该尾数包括7个显式位和第八个隐式位。

[0173] 示例4包括示例1的示例性处理器的实质内容，其中N由该指令指定并且具有在一些实施例中等于4、8和16中的任一个的值(但是本发明不对N设置上限，N可以是32、64或更大，并且与具有1024、2048或更多位的源向量相关联)。

[0174] 示例5包括示例1的示例性处理器的实质内容，其中执行电路用于在没有饱和的情况下利用无限精度执行乘法，并且在上溢的情况下将累加的结果饱和至正无穷或负无穷并在任何下溢的情况下将累加的结果饱和至零。

[0175] 示例6包括示例1的示例性处理器的实质内容，其中16位浮点格式为bfloat16或binary16。

[0176] 示例7包括示例1的示例性处理器的实质内容，其中执行电路用于并行地生成所指定的目的地的所有N个元素。

[0177] 示例8提供一种示例性方法，该示例性方法包括：使用取出电路取出指令，该指令具有用于指定操作码以及第一源向量、第二源向量和目的地向量的位置的字段，该操作码用于指示执行电路用于将所指定的第一源和第二源的N对16位浮点格式化元素相乘，并且将所得的乘积与所指定的目的地的对应的单精度元素的先前内容累加；使用解码电路对所取出的指令解码；以及利用执行电路如该操作码所指定地对经解码的指令作出响应。

[0178] 示例9包括示例8的示例性方法的实质内容，其中所指定的源向量和目的地向量中的每一个的位置在寄存器中或在存储器中。

[0179] 示例10包括示例8的示例性方法的实质内容，其中16位浮点格式包括符号位、8位的指数、以及尾数，该尾数包括7个显式位和第八个隐式位。

[0180] 示例11包括示例8的示例性方法的实质内容，其中N由该指令指定并且具有在一些实施例中等于4、8和16中的任一个的值(但是本发明不对N设置上限，N可以是32、64或更大，并且与具有1024、2048或更多位的源向量相关联)。

[0181] 示例12包括示例8的示例性方法的实质内容，其中执行电路用于在没有饱和的情况下利用无限精度执行乘法，并且在上溢的情况下将累加的结果饱和至正无穷或负无穷并在任何下溢的情况下将累加的结果饱和至零。

[0182] 示例13包括示例8的示例性方法的实质内容，其中16位浮点格式为bfloat16或binary16。

[0183] 示例14包括示例8的示例性方法的实质内容，其中执行电路用于并行地生成所指定的目的地的所有N个元素。

[0184] 示例15提供一种示例性系统，该示例性系统包括存储器和处理器，该处理器包括：取出电路，用于取出指令，该指令具有用于指定操作码以及第一源向量、第二源向量和目的地向量的位置的字段，该操作码用于指示执行电路用于将所指定的第一源和第二源的N对16位浮点格式化元素相乘，并且将所得的乘积与所指定的目的地的对应的单精度元素的先前内容累加；解码电路，用于对所取出的指令解码；以及执行电路，用于如该操作码所指定地对经解码的指令作出响应。

[0185] 示例16包括示例15的示例性系统的实质内容，其中所指定的源向量和目的地向量中的每一个的位置在该处理器中的寄存器中或在存储器中。

[0186] 示例17包括示例15的示例性系统的实质内容，其中16位浮点格式包括5位的指数或8位的指数。

[0187] 示例18提供一种用于由处理器执行的示例性方法，该方法包括：使用取出电路取出指令，该指令具有用于指定操作码以及第一源向量、第二源向量和目的地向量的位置的字段，该操作码用于指示处理器用于将所指定的第一源和第二源的N对16位浮点格式化元素相乘，并且将所得的乘积与所指定的目的地的对应的单精度元素的先前内容累加；使用解码电路对所取出的指令解码；以及利用执行电路如该操作码所指定地对经解码的指令作出响应。

[0188] 示例19包括示例18的示例性非暂态机器可读介质的实质内容，其中该指令进一步包括用于指定N的字段，其中N在一些实施例中等于4、8和16中的任一个(但是本发明不对N设置上限，N可以是32、64或更大，并且与具有1024、2048或更多位的源向量相关联。)，并且其中执行电路用于并行地生成所指定的目的地的所有N个元素。

[0189] 示例20包括示例18的示例性非暂态机器可读介质的实质内容，其中执行电路用于在没有饱和的情况下利用无限精度执行乘法，并且在上溢的情况下将累加的结果饱和至正无穷或负无穷并在任何下溢的情况下将累加的结果饱和至零。

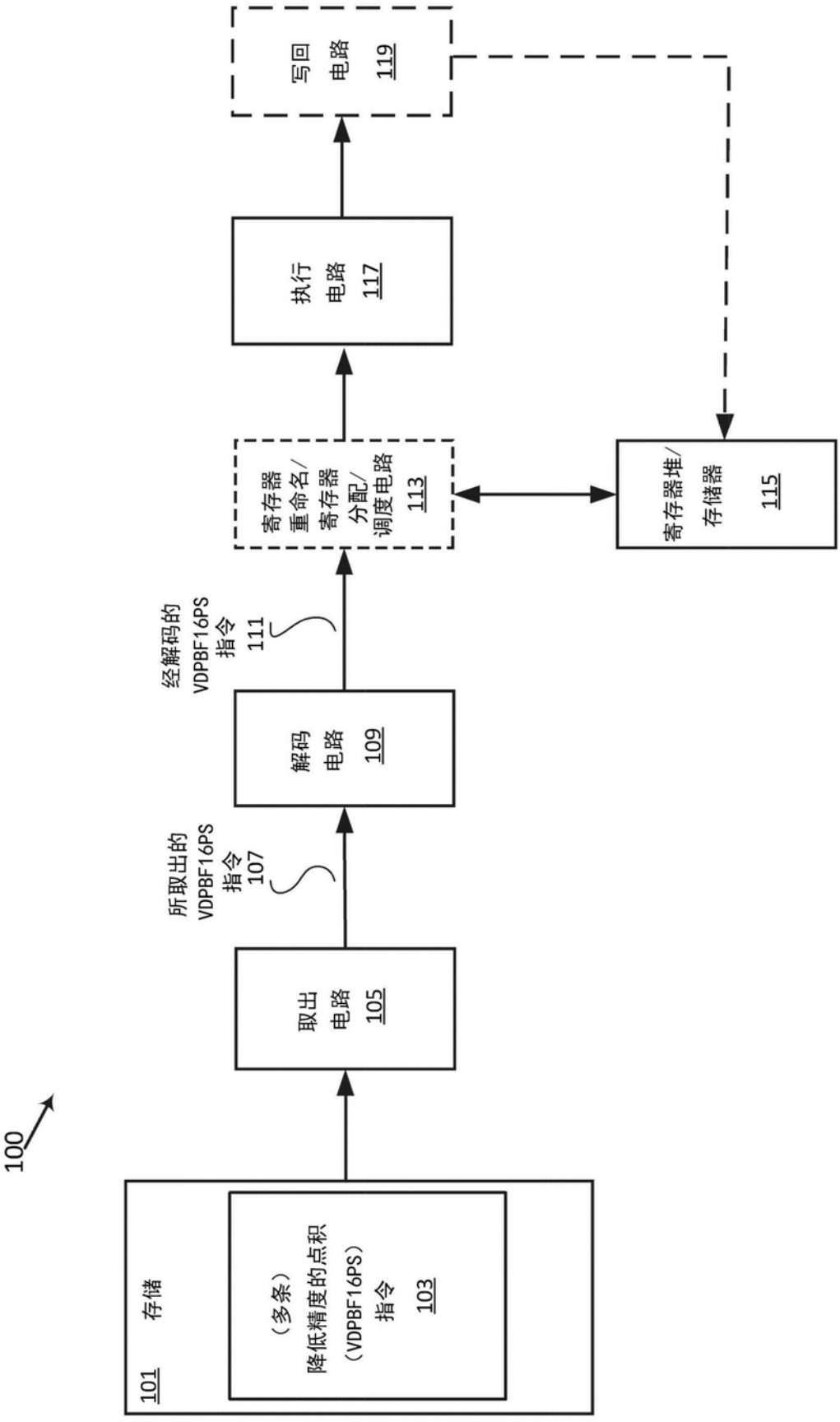

100
101 存储
（多条）
降低精度的点积
（VDPBF16PS）
指令
103
取出
电路
105
所取出的
VDPBF16PS
指令
107
解码
电路
109
经解码的
VDPBF16PS
指令
111
寄存器
重命名/
寄存器
分配/
调度电路
113
寄存器堆/
存储器
115
执行
电路
117
写回
电路
119

图1

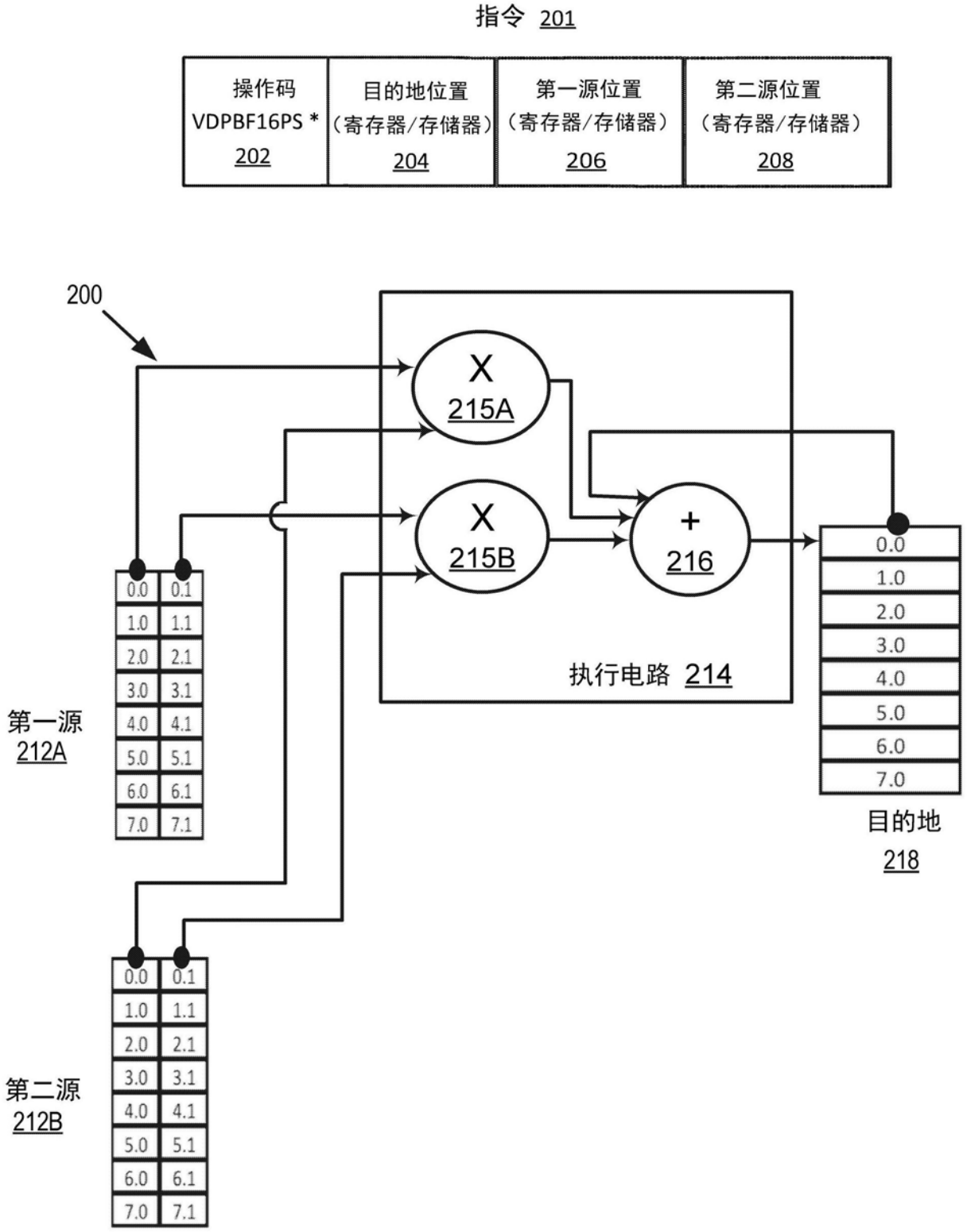

指令 201
操作码 VDPBF16PS * 202
目的地位置（寄存器/存储器） 204
第一源位置（寄存器/存储器） 206
第二源位置（寄存器/存储器） 208
200
X 215A
X 215B
+ 216
执行电路 214
第一源 212A
第二源 212B
目的地 218
0.0 0.1
1.0 1.1
2.0 2.1
3.0 3.1
4.0 4.1
5.0 5.1
6.0 6.1
7.0 7.1

图2

伪代码

300

```
VDPBF16PS srcdest, src1, src2
        VL = (128,256,512)
        kl = vl/32
        origdest := srcdest
        for i := 0 to kl-1:
                if k1[i] or *no writemask*:
                        if src2 is memory and evex.b == 1:
                                t := src2.dword[0]
                        else:
                                t := src2.dword[i]

                        //具有daz输入、ftz输出和RNE舍入的FP32 FMA。
                        //MXCSR既不被查阅也不被更新。

                        srcdest.fp32[i] += make_fp32(src1.bfloat16[2*i+0]) *
                                           make_fp32(t.bfloat[0])
                        srcdest.fp32[i] += make_fp32(src1.bfloat16[2*i+1]) *
                                           make_fp32(t.bfloat[1])
                else if *zeroing*:
                        srcdest.dword[i] := 0

                else: //合并掩码，目的地元素未被改变
                        srcdest.dword[i] := origdest.dword[i]

        srcdest[max_vl-1:vl] := 0
```

图3A

伪代码

310

```
VDPBF16PS srcdest, src1, src2
        VL = (128,256,512)
        kl = vl/32
        origdest := srcdest
        for i := 0 to kl-1:
                if k1[i] or *no writemask*:
                        if src2 is memory and evex.b == 1:
                                t := src2.dword[0]
                        else:
                                t := src2.dword[i]

                        //具有daz输入、ftz输出和RNE舍入的FP32 FMA。
                        //MXCSR既不被查阅也不被更新。

                        srcdest.fp32[i] += make_fp32(src1.bfloat16[2*i+1]) *
                                           make_fp32(t.bfloat[1])
                        srcdest.fp32[i] += make_fp32(src1.bfloat16[2*i+0]) *
                                           make_fp32(t.bfloat[0])
                else if *zeroing*:
                        srcdest.dword[i] := 0

                else: //合并掩码，目的地元素未被改变
                        srcdest.dword[i] := origdest.dword[i]

        srcdest[max_vl-1:vl] := 0
```

图3B

伪代码

320

助手函数

```
define make_fp32(x):
//x参数是bfloat16。将其紧缩为双字的高16位。
//位模式是合法的fp32值。返回该位模式
dword: = 0
dword[31:16] := x
return dword
```

图3C

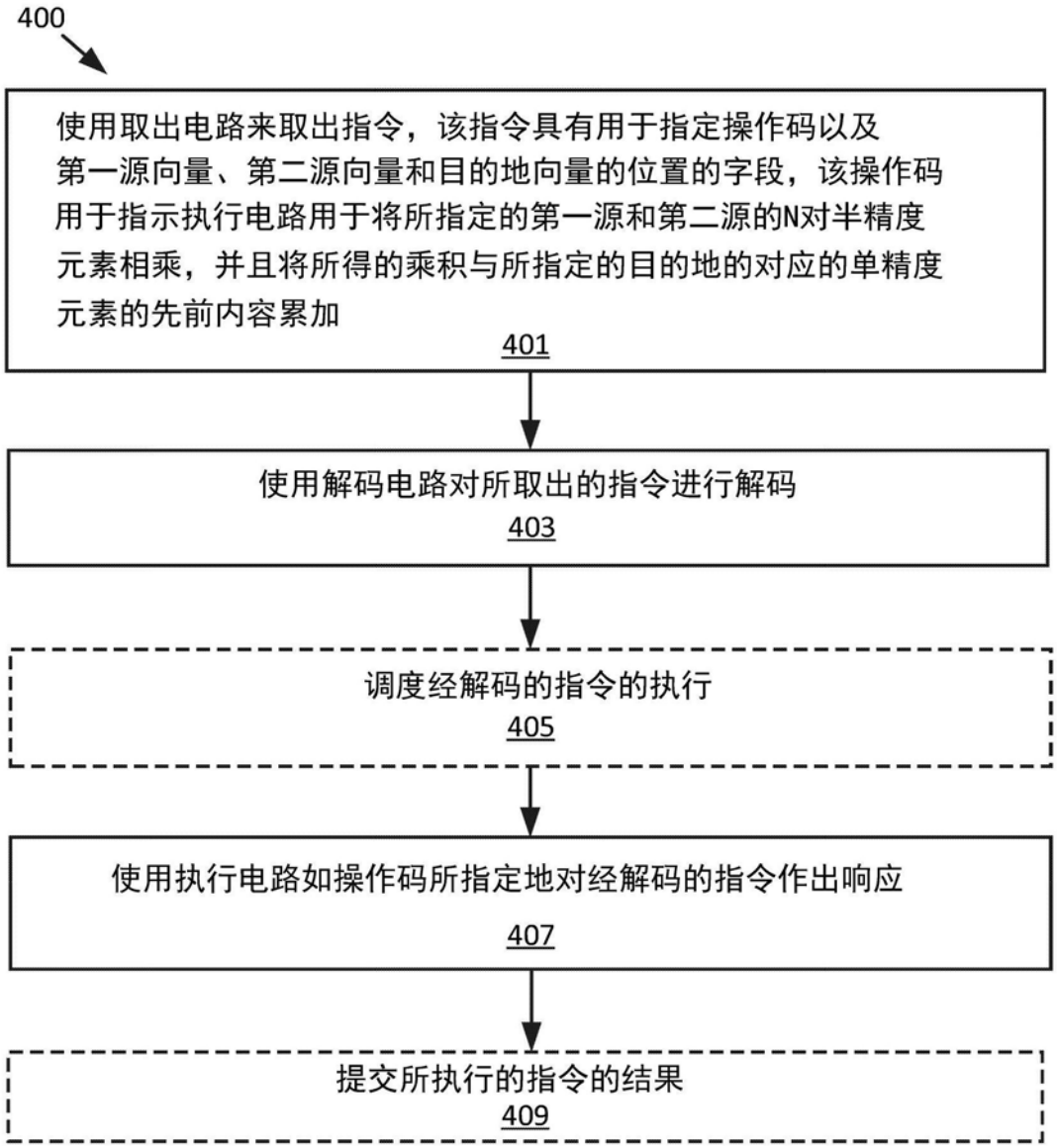


图4

VDPBF16PS指令 500

操作码 VDPBF16PS * 502	目的地位置（寄存器/存储器） 504	第一源位置（寄存器/存储器） 506	第二源位置（寄存器/存储器） 508	掩码 {k} 510	归零 {Z} 512	元素格式 514	向量尺寸 (N) 516

图5

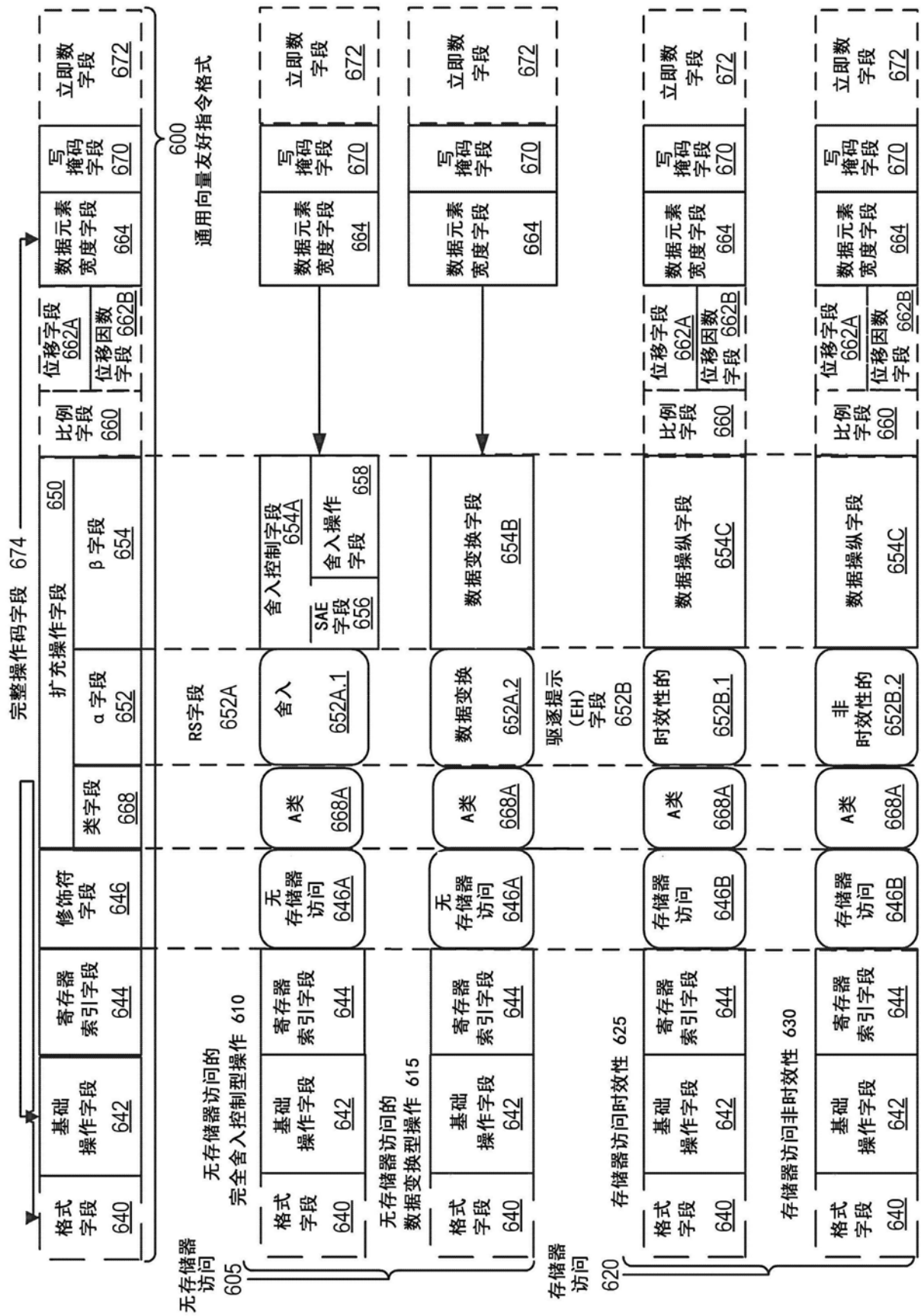

600
通用向量友好指令格式
完整操作码字段 674
格式字段 640
基础操作字段 642
寄存器索引字段 644
修饰符字段 646
扩充操作字段 650
类字段 668
α 字段 652
β 字段 654
比例字段 660
位移字段 662A
位移因数字段 662B
数据元素宽度字段 664
写掩码字段 670
立即数字段 672
RS字段 652A
舍入控制字段 654A
SAE字段 656
舍入操作字段 658
舍入 652A.1
数据变换字段 654B
数据变换 652A.2
驱逐提示(EH)字段 652B
数据操纵字段 654C
时效性的 652B.1
非时效性的 652B.2
A类 668A
无存储器访问 646A
存储器访问 646B
无存储器访问 605
无存储器访问的完全舍入控制型操作 610
无存储器访问的数据变换型操作 615
存储器访问 620
存储器访问时效性 625
存储器访问非时效性 630

图6A

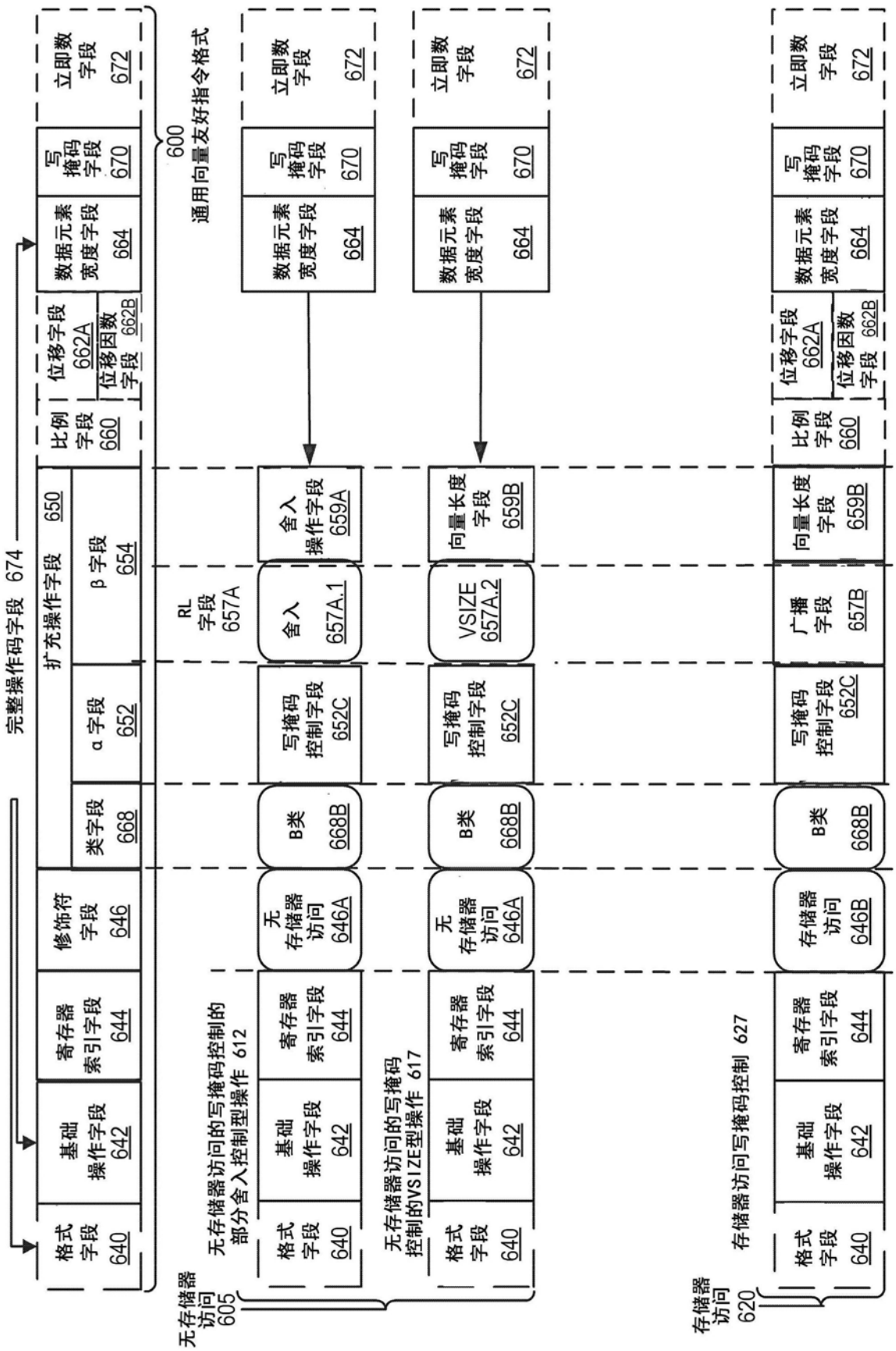

600
通用向量友好指令格式
完整操作码字段 674
格式字段 640
基础操作字段 642
寄存器索引字段 644
修饰符字段 646
扩充操作字段 650
类字段 668
α 字段 652
β 字段 654
比例字段 660
位移字段 662A
位移因数字段 662B
数据元素宽度字段 664
写掩码字段 670
立即数字段 672
无存储器访问 605
无存储器访问的写掩码控制的部分舍入控制型操作 612
无存储器 646A
B类 668B
写掩码控制字段 652C
RL字段 657A
舍入 657A.1
舍入操作字段 659A
无存储器访问的写掩码控制的VSIZE型操作 617
VSIZE 657A.2
向量长度字段 659B
存储器访问 620
存储器访问写掩码控制 627
存储器 646B
广播字段 657B

图6B

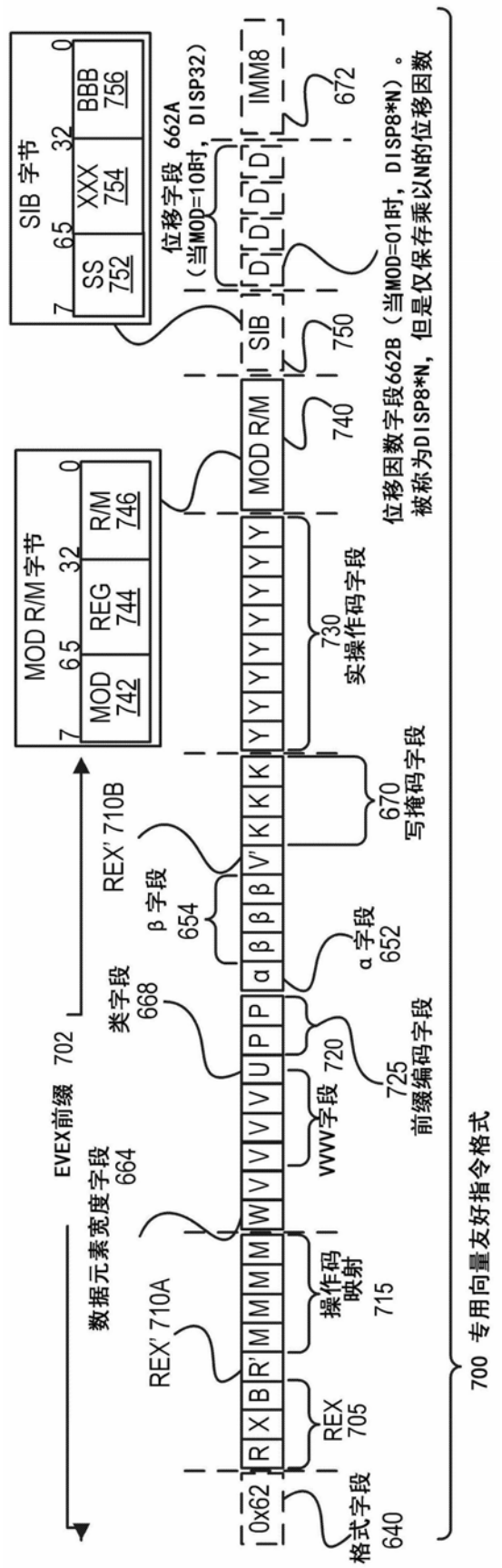

SIB 字节
SS 752
XXX 754
BBB 756
位移字段 662A
（当MOD=10时，DISP32）
MOD R/M 字节
MOD 742
REG 744
R/M 746
SIB
MOD R/M
740
750
672
IMM8
位移因数字段662B（当MOD=01时，DISP8*N）。
被称为DISP8*N，但是仅保存乘以N的位移因数
EVEX前缀 702
类字段 668
REX' 710B
β 字段 654
α 字段 652
实操作码字段 730
写掩码字段 670
数据元素宽度字段 664
REX' 710A
操作码映射 715
VVV字段 720
前缀编码字段 725
0x62
R X B R' M M M W V V V V U P P α β β V' K K K Y Y Y Y Y Y Y
格式字段 640
REX 705
700 专用向量友好指令格式

图7A

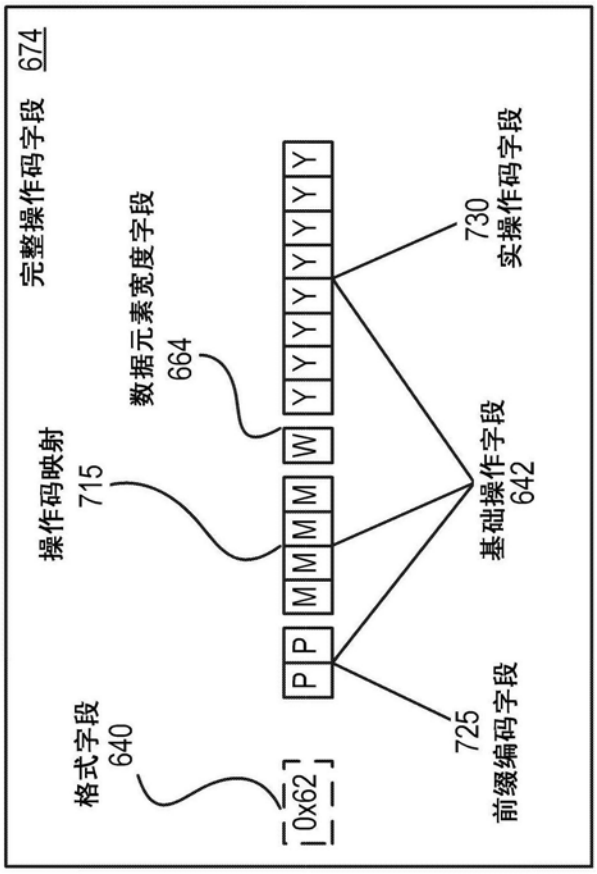

完整操作码字段 674
格式字段 640
操作码映射 715
数据元素宽度字段 664
0x62
P P M M M W Y Y Y Y Y Y Y Y
基础操作字段 642
730 实操作码字段
725 前缀编码字段

图7B

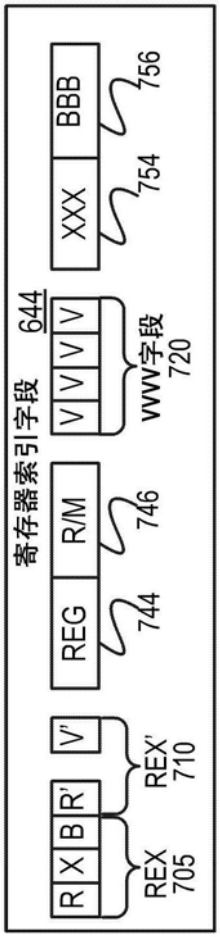

寄存器索引字段 644
R
X
B
R'
V'
REG
R/M
V
V
V
V
XXX
BBB
REX 705
REX' 710
744
746
vvv字段 720
754
756

图7C

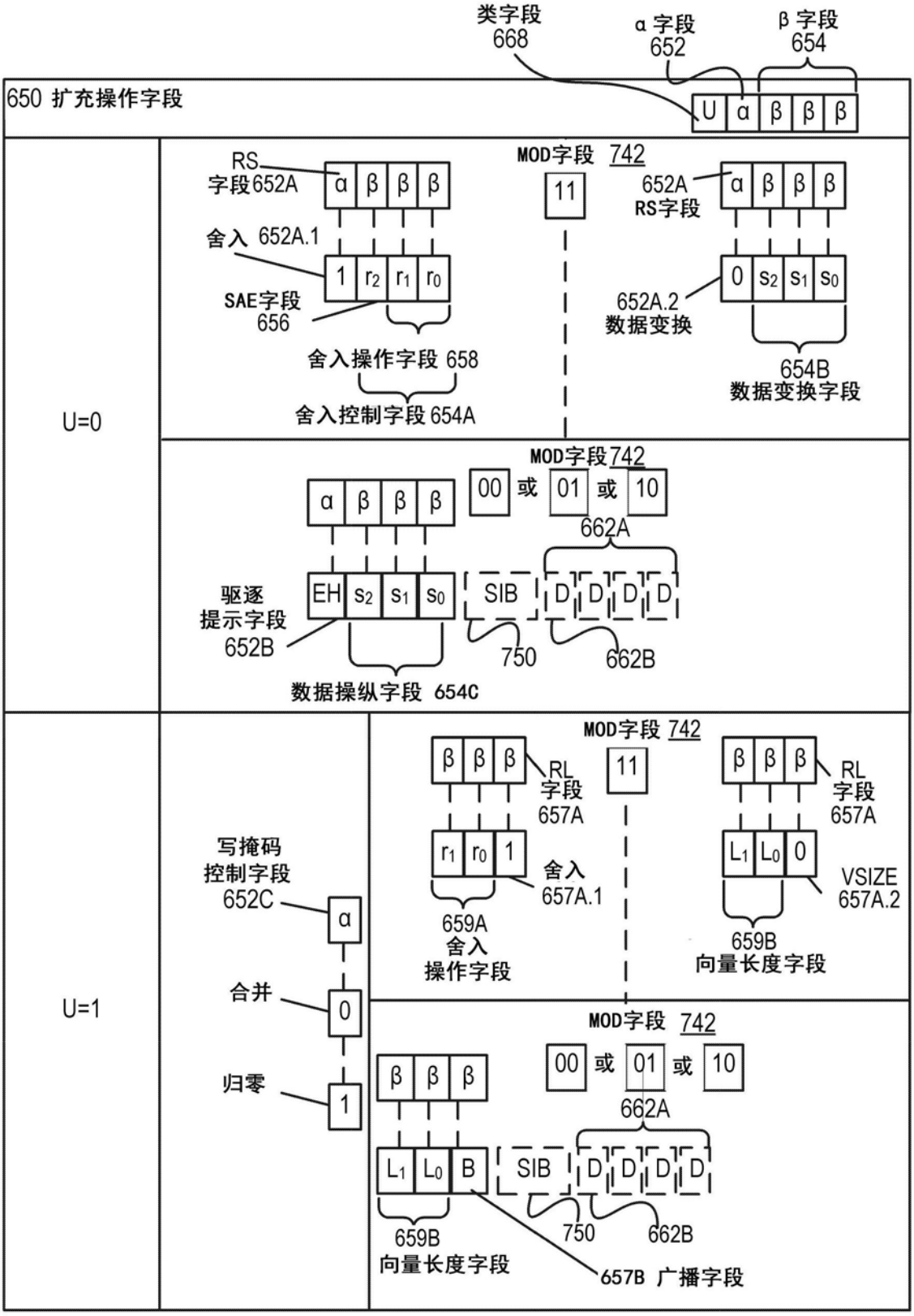

类字段
668
α 字段
652
β 字段
654
650 扩充操作字段
U α β β β
U=0
RS
字段652A
α β β β
舍入 652A.1
1 r2 r1 r0
SAE字段
656
舍入操作字段 658
舍入控制字段 654A
MOD字段 742
11
652A
RS字段
α β β β
652A.2
数据变换
0 s2 s1 s0
654B
数据变换字段
MOD字段742
00 或 01 或 10
662A
α β β β
EH s2 s1 s0
SIB D D D D
驱逐
提示字段
652B
750
662B
数据操纵字段 654C
U=1
写掩码
控制字段
652C
α
合并
0
归零
1
MOD字段 742
β β β
RL
字段
657A
11
r1 r0 1
舍入
657A.1
659A
舍入
操作字段
β β β
RL
字段
657A
L1 L0 0
VSIZE
657A.2
659B
向量长度字段
MOD字段 742
00 或 01 或 10
662A
β β β
L1 L0 B
SIB D D D D
659B
向量长度字段
750
662B
657B 广播字段

图7D

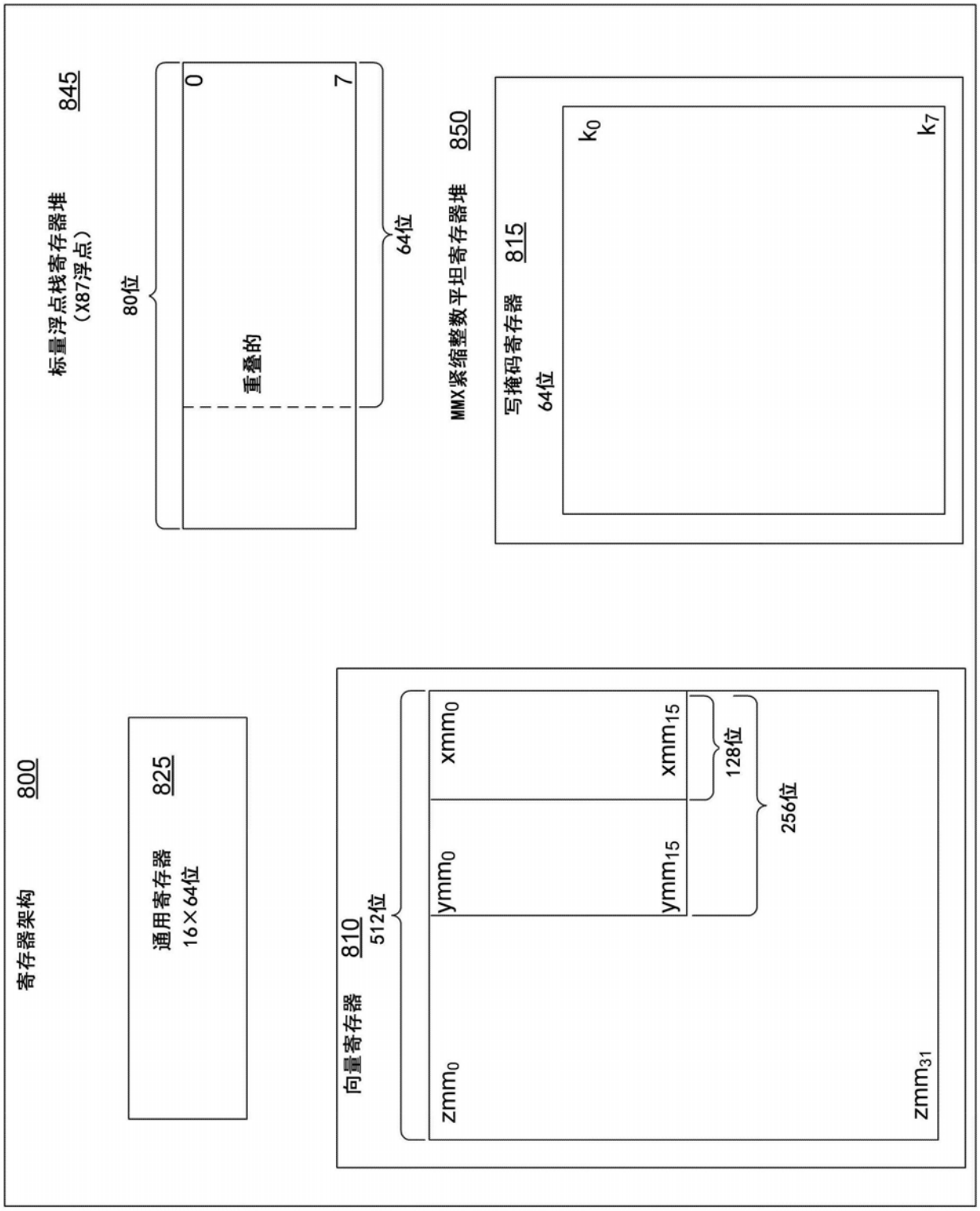

寄存器架构 800
通用寄存器 825
16×64位
向量寄存器 810
512位
zmm0
ymm0
xmm0
ymm15
xmm15
128位
256位
zmm31
标量浮点栈寄存器堆 845
(X87浮点)
80位
0
7
重叠的
64位
MMX紧缩整数平坦寄存器堆 850
写掩码寄存器 815
64位
k0
k7

图8

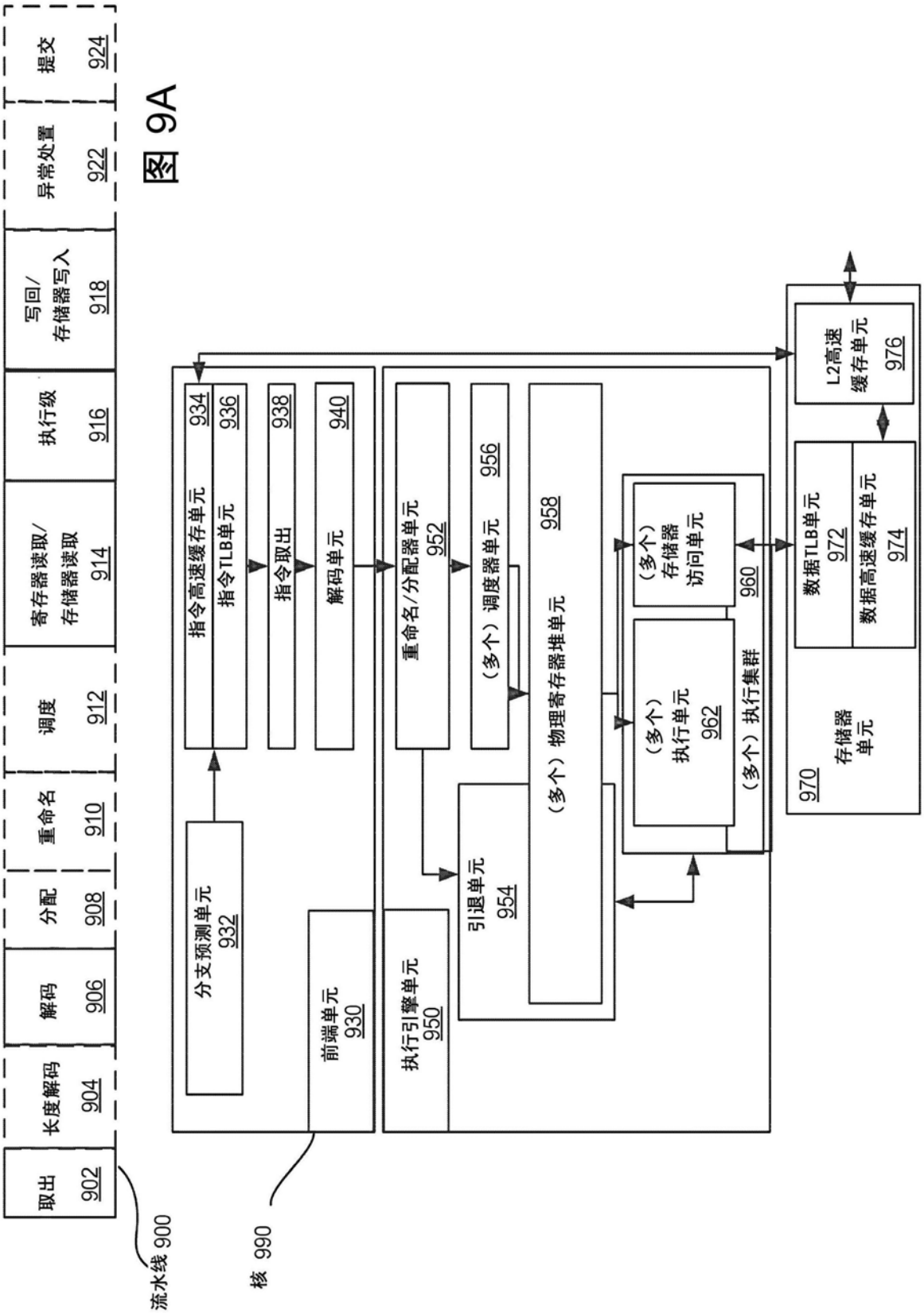

取出 902
长度解码 904
解码 906
分配 908
重命名 910
调度 912
寄存器读取/存储器读取 914
执行级 916
写回/存储器写入 918
异常处置 922
提交 924
流水线 900
核 990
指令高速缓存单元 934
指令TLB单元 936
指令取出 938
解码单元 940
分支预测单元 932
前端单元 930
重命名/分配器单元 952
(多个) 调度器单元 956
(多个) 物理寄存器堆单元 958
执行引擎单元 950
引退单元 954
(多个) 执行单元 962
(多个) 存储器访问单元 960
(多个) 执行集群
970 存储器单元
数据TLB单元 972
数据高速缓存单元 974
L2高速缓存单元 976

图 9A

图 9B

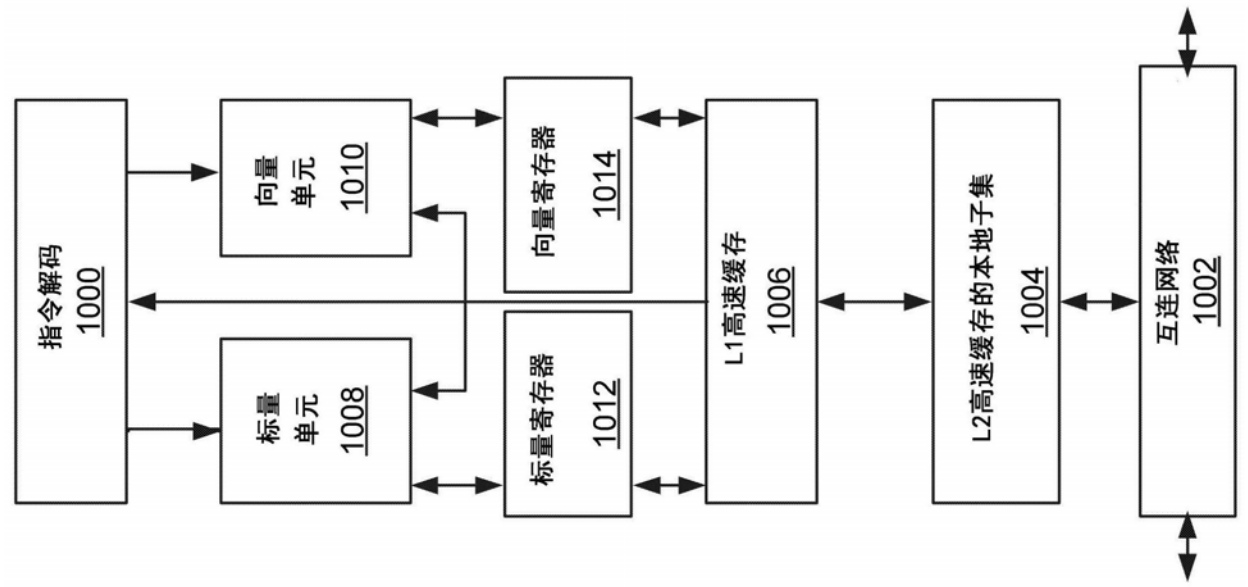

指令解码
1000
向量
单元
1010
标量
单元
1008
向量寄存器
1014
标量寄存器
1012
L1高速缓存
1006
L2高速缓存的本地子集
1004
互连网络
1002

图10A

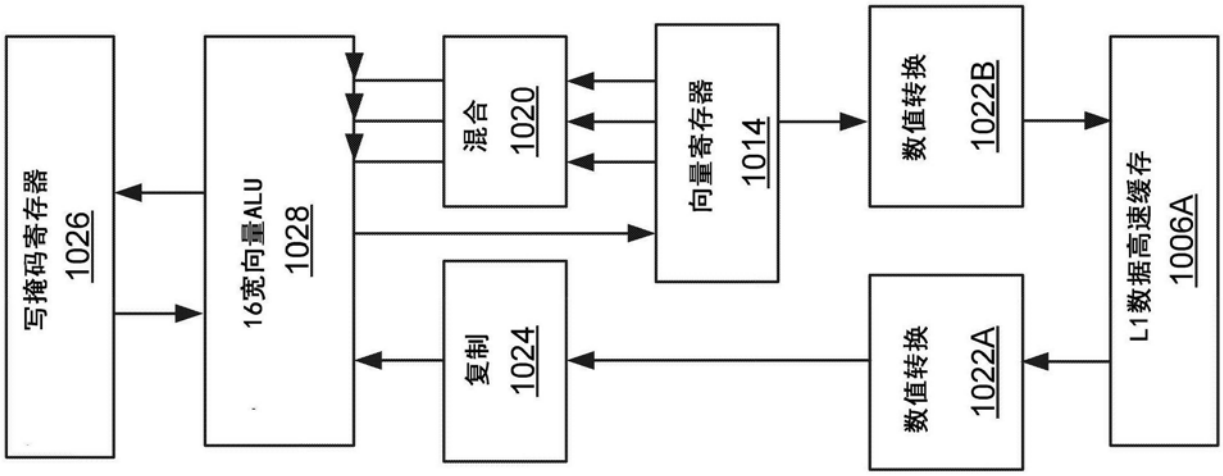

写掩码寄存器
1026
16宽向量ALU
1028
混合
1020
向量寄存器
1014
数值转换
1022B
L1数据高速缓存
1006A
复制
1024
数值转换
1022A

图10B

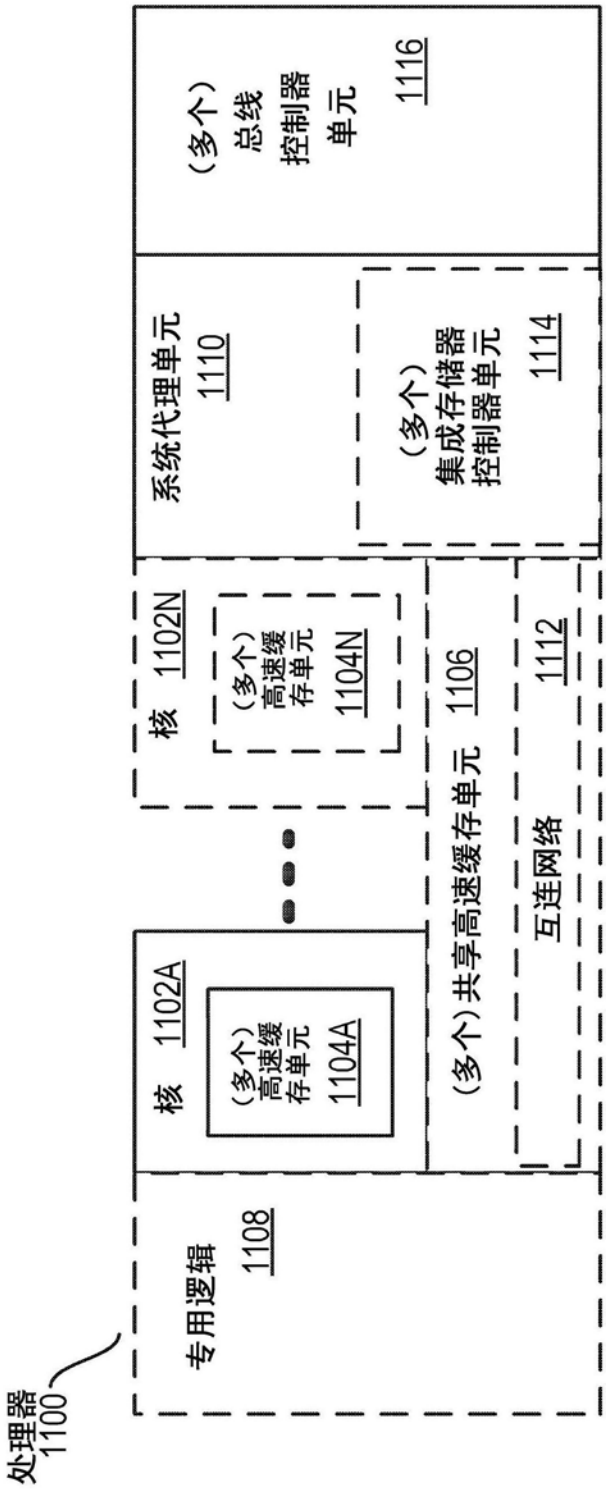

处理器
1100
专用逻辑
1108
核 1102A
(多个)
高速缓
存单元
1104A
核 1102N
(多个)
高速缓
存单元
1104N
(多个)共享高速缓存单元 1106
互连网络 1112
系统代理单元
1110
(多个)
集成存储器
控制器单元
1114
(多个)
总线
控制器
单元
1116

图11

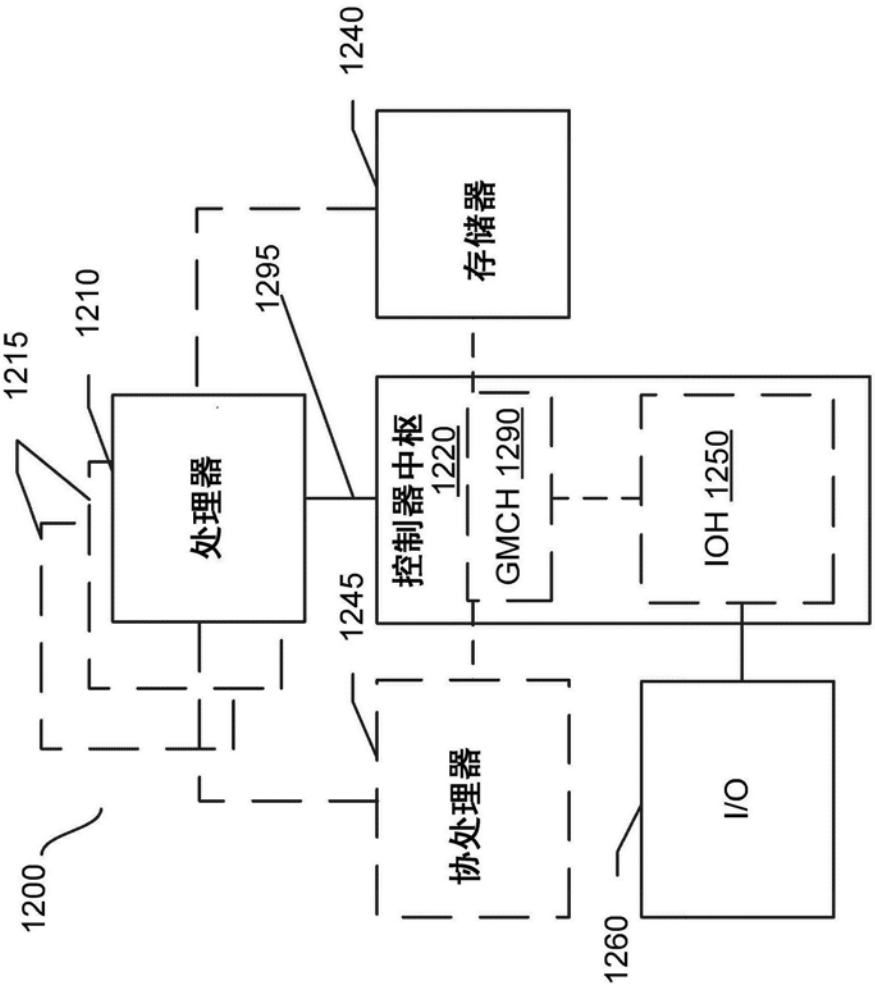

1240
存储器
1295
1210
1215
处理器
控制器中枢
1220
GMCH 1290
IOH 1250
1245
协处理器
I/O
1200
1260

图12

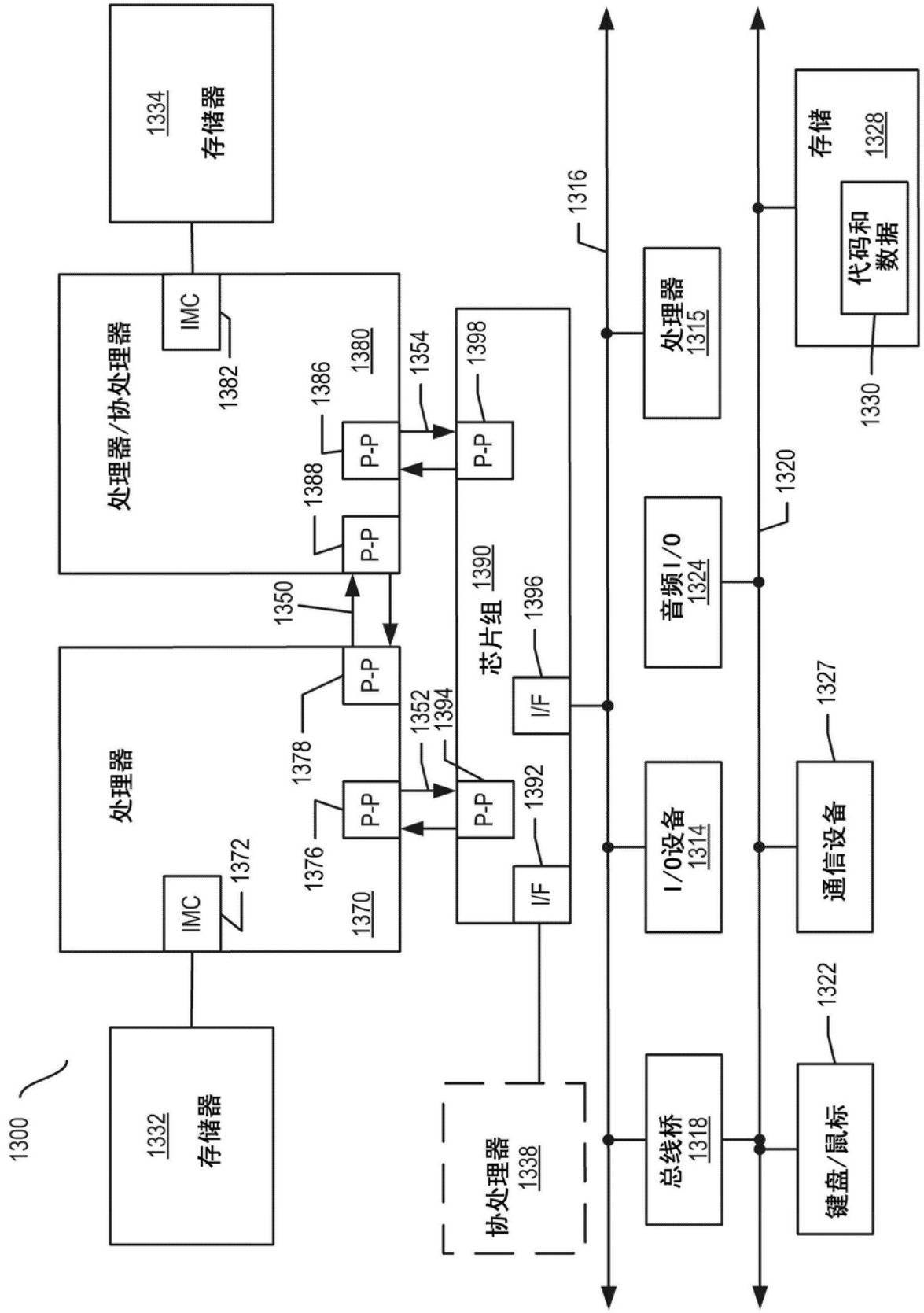

1300
1332
存储器
1334
存储器
处理器/协处理器
IMC
1382
1380
1386
1388
P-P
P-P
1354
1398
P-P
1350
处理器
IMC
1372
1370
1376
1378
P-P
P-P
1352
1394
芯片组
1390
1396
I/F
1392
I/F
协处理器
1338
1316
处理器
1315
音频I/O
1324
1320
I/O设备
1314
通信设备
1327
总线桥
1318
键盘/鼠标
1322
存储
1328
代码和
数据
1330

图13

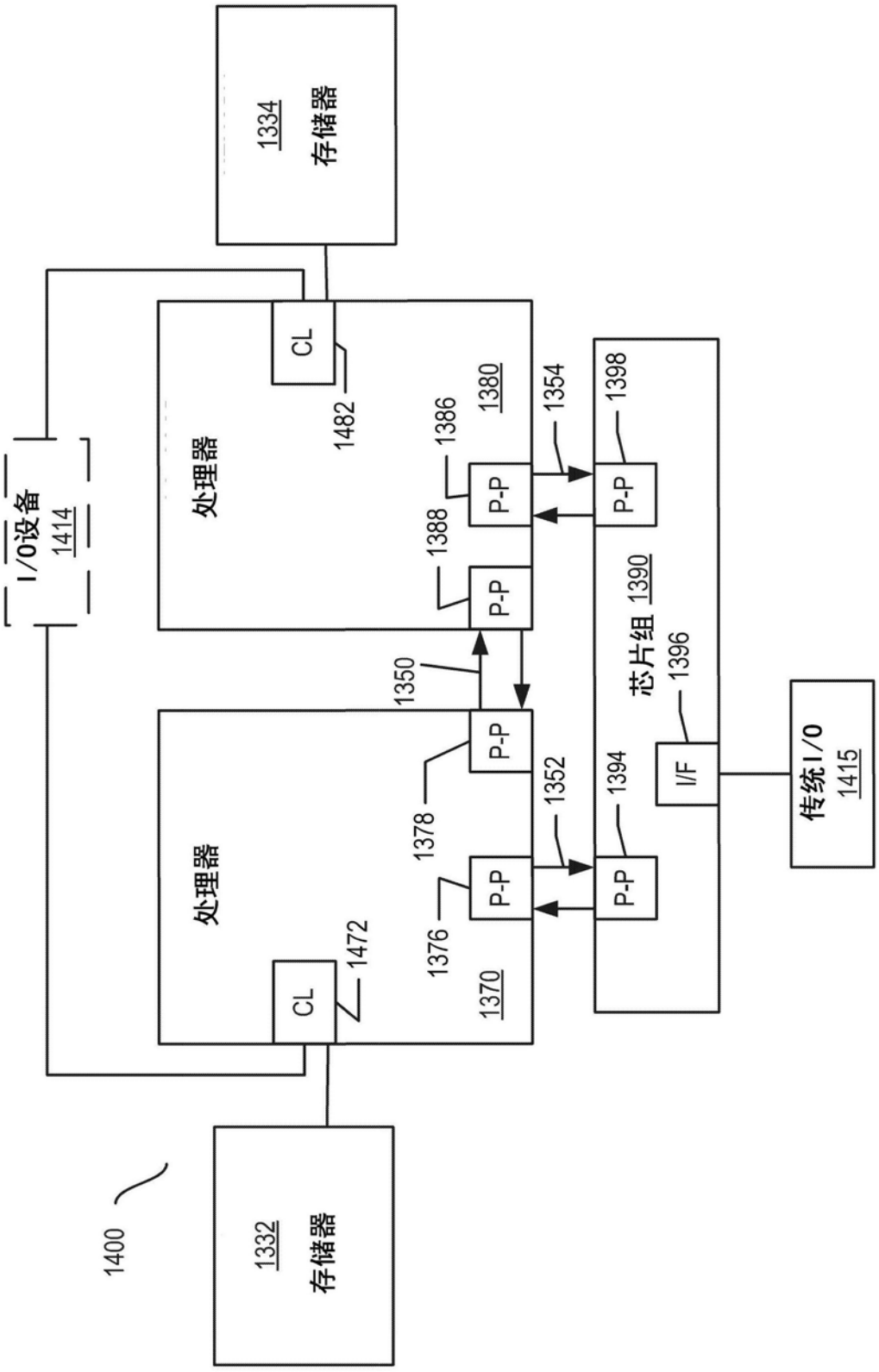

1334
存储器
CL
1482
处理器
1380
1386
1388
P-P
P-P
1354
1398
P-P
I/O设备
1414
芯片组 1390
1396
1350
P-P
1352
1394
1378
P-P
P-P
I/F
传统I/O
1415
处理器
1376
1370
CL
1472
1400
1332
存储器

图14

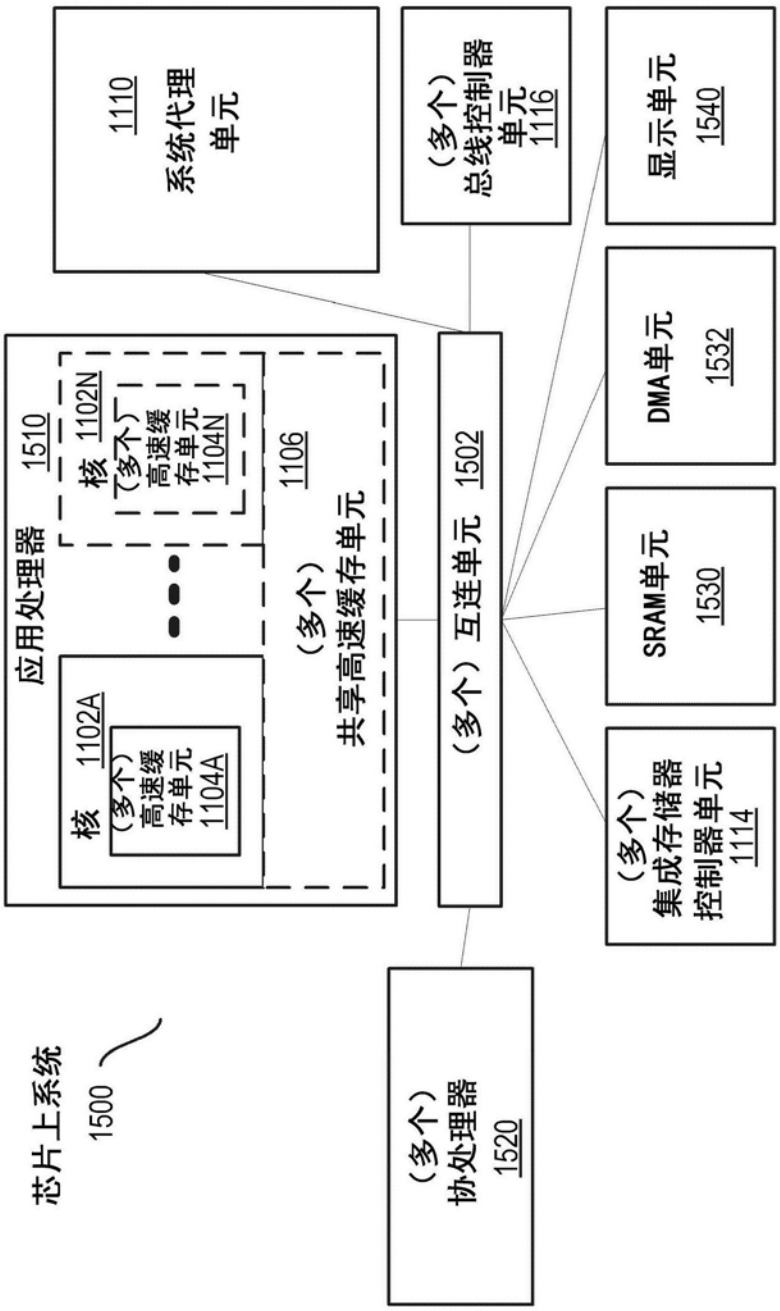

芯片上系统
1500
应用处理器 1510
核 1102A
(多个)高速缓存单元 1104A
核 1102N
(多个)高速缓存单元 1104N
1106
(多个)共享高速缓存单元
(多个)协处理器 1520
(多个)互连单元 1502
1110 系统代理单元
(多个)总线控制器单元 1116
(多个)集成存储器控制器单元 1114
SRAM单元 1530
DMA单元 1532
显示单元 1540

图15

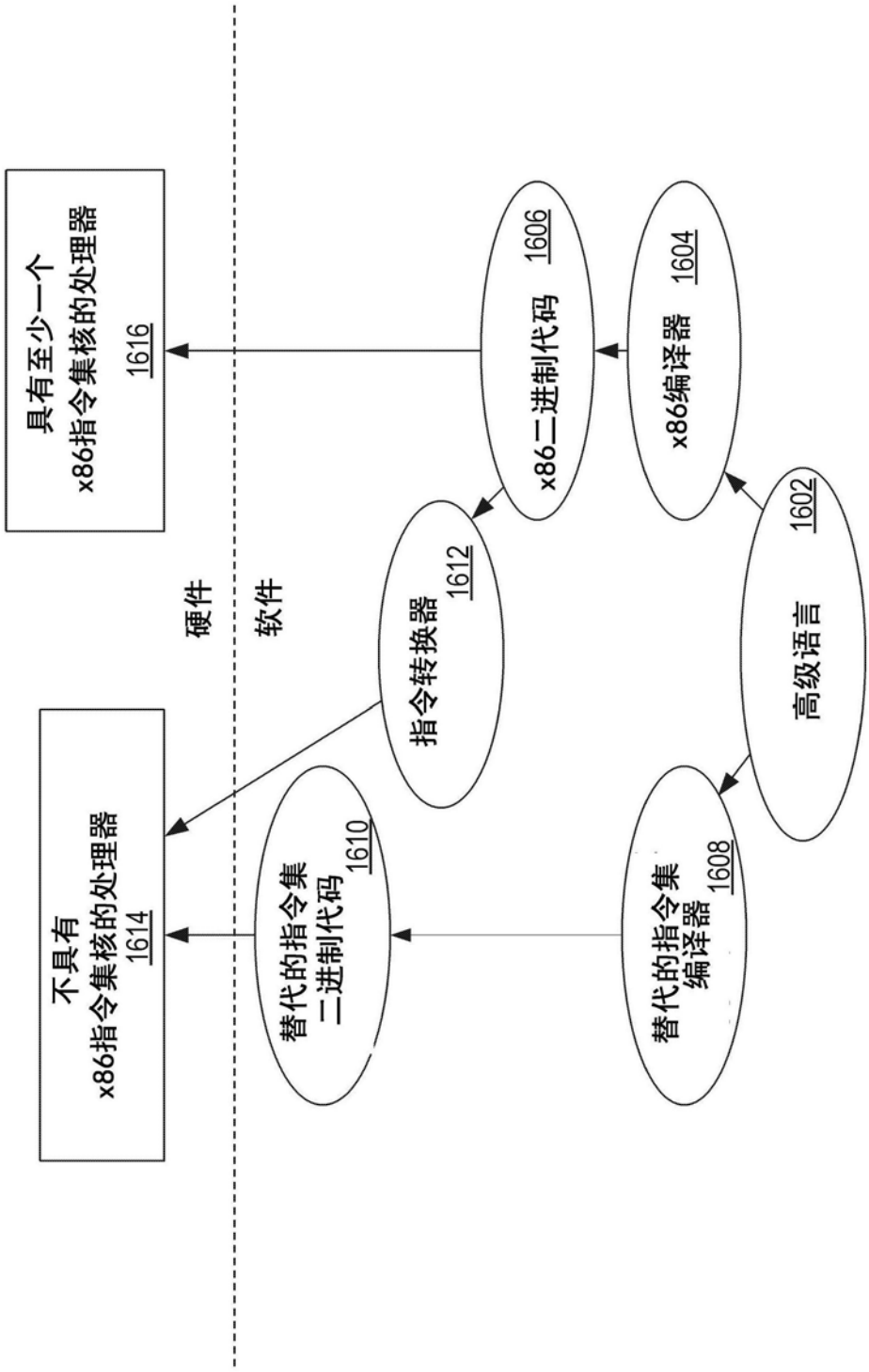

具有至少一个
x86指令集核的处理器
1616
x86二进制代码 1606
x86编译器 1604
指令转换器 1612
高级语言 1602
硬件
软件
不具有
x86指令集核的处理器
1614
替代的指令集
二进制代码 1610
替代的指令集
编译器 1608

图16