(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 104185839 B
(45)授权公告日 2017.06.06

(21)申请号 201380014276.9

(22)申请日 2013.03.07

(65)同一申请的已公布的文献号
申请公布号 CN 104185839 A

(43)申请公布日 2014.12.03

(30)优先权数据
13/421,599 2012.03.15 US

(85)PCT国际申请进入国家阶段日
2014.09.15

(86)PCT国际申请的申请数据
PCT/EP2013/054608 2013.03.07

(87)PCT国际申请的公布数据
WO2013/135556 EN 2013.09.19

(73)专利权人 国际商业机器公司
地址 美国纽约阿芒克

(72)发明人 J.D.布拉德伯里 M.K.格施温德 T.斯雷格 E.M.施瓦茨 C.雅各比

(74)专利代理机构 北京市柳沈律师事务所 11105
代理人 邸万奎

(51)Int.Cl.
G06F 9/30(2006.01)

(56)对比文件
US 2002/0026569 A1,2002.02.28,
US 2001/0018731 A1,2001.08.30,
CN 102298515 A,2011.12.28,
WO 2008/124473 A1,2008.10.16,
US 7877582 B2,2011.01.25,
US 6334176 B1,2001.12.25,
CN 1487409 A,2004.04.07,
US 5881260 A,1999.03.09,
WO 2009/087160 A1,2009.07.16,
CN 102077195 A,2011.05.25,

审查员 张力

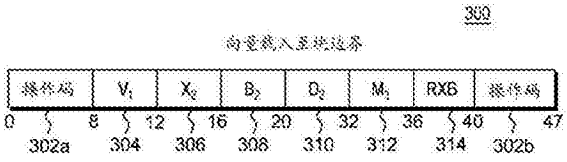
权利要求书1页 说明书29页 附图12页

(54)发明名称

将数据载入寄存器的方法和系统

(57)摘要

提供“载入至块边界”指令,其将数据的可变数目个字节载入至寄存器中,同时确保指定存储器边界不跨越。基于边界的指定类型和执行该指令的处理器的一个或者多个特性(诸如由处理器使用的高速缓冲存储器线大小或者页大小)动态确定该边界。



CN 104185839 B

1.一种用于在中央处理单元中执行机器指令的方法，该方法包含以下步骤：

由处理器获得用于执行的机器指令，该机器指令根据计算机架构定义以用于计算机执行，该机器指令包含：

至少一个操作码字段，其提供操作码，该操作码识别“载入至块边界”操作；

寄存器字段，其用以指明寄存器，该寄存器包含第一操作数；

用于在主存储器中定位第二操作数的至少一个字段；以及

块边界类型指示符，其用以指示第二操作数的块边界的指定类型；以及

执行该机器指令，该执行包括：

用在基于块边界的指定类型和处理器的一个或者多个特性动态确定的主存储器的块内的该第二操作数的对应字节仅载入该第一操作数的字节，其中该仅载入包括：在确保仅将第二操作数的块内的数据载入第一操作数的同时，将可变量的数据从该第二操作数的块载入至该第一操作数，其中，从该第二操作数的块载入在该第二操作数的块中的开始地址处开始，该开始地址由机器指令提供，并且其中，载入在该第二操作数的块的确定块边界处或者该第二操作数的块的确定块边界之前结束，其中，可变量的载入数据基于开始地址和确定块边界，基于块边界的指定类型和处理器的一个或者多个特性动态确定该确定块边界，

其中该第二操作数的地址为从其要将数据载入于该第一操作数中的存储器中的开始地址，并且其中，该执行还包含：确定载入要在其处停止的结束地址，其中该载入停止于该结束地址处；以及

其中该确定该结束地址包含如下计算该结束地址：

结束地址＝(开始地址+(边界大小-(开始地址AND NOT边界掩码))，开始地址+寄存器大小)中的最小值，其中该边界大小为该块边界，边界掩码等于0-边界大小，且寄存器大小为该寄存器的指定长度。

2.如权利要求1所述的方法，其中该至少一个字段包含位移字段、基本字段和索引字段，用于定位通用寄存器的该基本字段和该索引字段具有要添加至该位移字段的内容以形成该第二操作数的地址的内容。

3.如权利要求1所述的方法，其中，该机器指令还包含掩码字段，该掩码字段包括块边界类型指示符。

4.如权利要求1所述的方法，其中，该一个或者多个特性包含处理器的高速缓冲存储器线大小或者处理器的页大小中的一个。

5.如权利要求1所述的方法，其中，该执行包含使用第二操作数的地址确定块的块边界，其中，在数据结构查找中使用该地址，以确定块的块边界。

6.如权利要求1所述的方法，其中该载入包含以下中的一个：自左至右载入该第一操作数，或自右至左载入该第一操作数。

7.如权利要求6所述的方法，其中在执行时提供该载入的方向。

8.如权利要求1所述的方法，其中该机器指令还包含要于指明一个或多个寄存器时使用的扩展字段，且其中该寄存器字段与该扩展字段的至少一部分组合以指明该寄存器。

9.一种包含适配用于实施根据任一前述方法权利要求的方法的所有步骤的构件的系统。

将数据载入寄存器的方法和系统

技术领域

[0001] 本发明的方面总体涉及数据处理，尤其涉及将数据载入至寄存器中。

背景技术

[0002] 数据处理包括各种类型的处理，包括将数据载入至寄存器中。数据至寄存器中的载入包括(但不限于)字符数据(诸如，字符数据串)；整数数据；或任何其他类型的数据的载入。然后能够使用和/或操纵所载入的数据。

[0003] 执行各种类型的处理(包括将数据载入至寄存器中)的当前指令倾向于效率低下。

发明内容

[0004] 经由提供用于执行机器指令的计算机程序产品来克服现有技术的缺点并提供优势。该计算机程序产品包括计算机可读存储介质，其可由处理电路读取且存储用于由该处理电路执行以用于执行方法的指令。该方法包括(例如)：由处理器获得用于执行的机器指令，该机器指令根据计算机架构定义以用于计算机执行，该机器指令包含：至少一个操作码字段，其提供操作码，该操作码识别“载入至块边界”操作；寄存器字段，其用以指明寄存器，该寄存器包含第一操作数；用于找出第二操作数在主存储器中的位置的至少一个字段；以及执行该机器指令，该执行包含：仅用在基于块边界的指定类型和处理器的一个或多个特性动态确定主存储器的块内的该第二操作数的对应字节载入该第一操作数的字节。

[0005] 本文中也描述和主张关于本发明的一个或多个方面的方法和系统。另外，本文中也描述且可主张关于本发明的一个或多个方面的服务。

[0006] 经由本发明的技术实现额外特征和优势。本发明的其他实施例和方面在本文中得以详细描述且被视为所主张的本发明的一部分。

附图说明

[0007] 在说明书的结论处的权利要求中作为实例具体指出并且明显主张本发明的一个或者多个方面。依据结合附图的以下详细描述，本发明的前面和其他目的、特征和优点显而易见，在附图中：

[0008] 图1描绘并入且使用本发明的一个或者多个方面的计算环境的实例；

[0009] 图2A描绘并入且使用本发明的一个或者多个方面的计算环境的另一实例；

[0010] 图2B描绘根据本发明的一个方面的图2A的存储器的进一步细节；

[0011] 图3描绘根据本发明的一个方面的“向量载入至块边界”指令的格式的一个实施例；

[0012] 图4描绘根据本发明的一个方面的与“向量载入至块边界”指令相关联的逻辑的一个实施例；

[0013] 图5描绘根据本发明的一个方面的要载入至向量寄存器中的数据的一个实例；

[0014] 图6描绘根据本发明的一个方面的寄存器文件的一个实例；

[0015] 图7描绘并入本发明的一个或多个方面的计算机程序产品的实施例；

[0016] 图8描绘并入且使用本发明一个或多个方面的主机计算机系统的一个实施例；

[0017] 图9描绘并入且使用本发明一个或多个方面的计算机系统的另一实例；

[0018] 图10描绘并入且使用本发明一个或多个方面的包含计算机网络的计算机系统的另一实例；

[0019] 图11描绘并入且使用本发明一个或多个方面的计算机系统的各种元件的一个实施例；

[0020] 图12A描绘并入且使用本发明一个或多个方面的图11的计算机系统的执行单元的一个实施例；

[0021] 图12B描绘并入且使用本发明一个或多个方面的图11的计算机系统的分支单元的一个实施例；

[0022] 图12C描绘并入且使用本发明一个或多个方面的图11的计算机系统的载入/存储单元的一个实施例；以及

[0023] 图13描绘并入且使用本发明一个或多个方面的仿真主机计算机系统的一个实施例。

具体实施方式

[0024] 根据本发明的一个方面，提供用于促进数据在寄存器中的载入的能力。作为实例，数据包括字符数据、整数数据和/或其他类型的数据。另外，寄存器为向量寄存器或另一类型的寄存器。

[0025] 字符数据包括(但不限于)任何语言中的字母字符；数字；标点符号；和/或其他符号。字符数据可为或可不为数据的串。标准与字符数据相关联，标准的实例包括(但不限于)：ASCII(美国信息交换标准码)；Unicode，包括(但不限于)UTF(Unicode变换格式)8；UTF16；等。

[0026] 向量寄存器(也被称作向量)包括一个或多个元素，且作为实例，每个元素的长度为一个、两个或四个字节。另外，向量操作数(例如)为具有多个元素的SIMD(单指令多数据)操作数。在其他实施例中，元素可以是其他大小的；且向量操作数不需要为SIMD，和/或可包括一个元素。

[0027] 在一个实例中，提供“向量载入至块边界”指令，其将来自存储器的数据的可变数目个字节载入至向量寄存器中，同时确保不跨越从其数据正被载入的存储器的指定边界。可通过指令(例如，指令文本中的可变值、编码于操作码中的固定指令文本值、在指令中指定的基于寄存器的边界等)明确地指定边界；或可由机器动态地确定边界。例如，指令指定数据将被载入至页面或高速缓冲存储器边界，且机器确定高速缓冲存储器线或页面大小(例如，在(例如)翻译后备缓冲器中查找以确定页面大小)，且载入至该点。

[0028] 作为另一实例，此指令也用以使数据存取与选定边界对准。

[0029] 在一个实施例中，该指令仅用第二操作数的在由块边界的类型(例如，高速缓冲存储器线或者页)和执行该指令的处理器的一个或者多个特性(诸如高速缓冲存储器线大小或者页大小)动态确定的主存储器(也称为主储存器)的块内的对应字节载入向量寄存器(第一操作数)的字节。如本文中所使用，主存储器的块为具有指定大小的任何存储器块。指

定大小也被称作块的边界，边界为块的末端。

[0030] 在另一实施例中，载入其他类型的寄存器。即，正被载入的寄存器不是向量寄存器，而是另一类型的寄存器。在此上下文中，该指令被称作“载入至块边界”指令，其用以将数据载入至寄存器中。

[0031] 参看图1描述并入且使用本发明的一个或多个方面的计算环境的一个实施例。计算环境100包括(例如)经由(例如)一个或多个总线108和/或其他连接 耦接至彼此的处理器102(例如，中央处理单元)、存储器104(例如，主存储器)和或多个输入/输出(I/O)器件和/或接口106。

[0032] 在一个实例中，处理器102基于由International Business Machines Corporation供应的z/Architecture，且为服务器的一部分，诸如也由International Business Machines Corporation供应且实施z/Architecture的System z服务器。z/Architecture的一个实施例描述于题为“z/Architecture Principles of Operation”的IBM®公开号(SA22-7832-08号，第九版，2010年8月)中。在一个实例中，该处理器执行操作系统，诸如，也由International Business Machines Corporation供应的z/OS。IBM®、Z/ARCHITECTURE®和Z/OS®为International Business Machines Corporation(Armonk，New York，USA)的注册商标。本文中使用的其他名称可为International Business Machines Corporation或其他公司的注册商标、商标或产品名称。

[0033] 在另一实施例中，处理器102基于由International Business Machines Corporation供应的Power架构。Power架构的一个实施例描述于“Power ISA™第2.06版修订B”(International Business Machines Corporation，2010年7月23日)中。POWER ARCHITECTURE®为International Business Machines Corporation的注册商标。

[0034] 在另一实施例中，处理器102基于由Intel Corporation供应的Intel架构。Intel架构的一个实施例描述于“Intel®64 and IA-32 Architectures Developer’s Manual:第2B卷，Instructions Set Reference，A-L”(序号253666-041US，2011年12月)和“Intel®64and IA-32 Architectures Developer’s Manual:第2B卷，Instructions Set Reference，M-Z”(序号253667-041US，2011年12月)中。Intel®为Intel Corporation(Santa Clara，California)的注册商标。

[0035] 参看图2A描述并入且使用本发明的一个或多个实施例的计算环境的另一实施例。在此实例中，计算环境200包括(例如)经由(例如)一个或多个总线208和/或其他连接耦接至彼此的本地中央处理单元202、存储器204和一个或多个输入/输出器件和/或接口206。作为实例，计算环境200可包括：由International Business Machines Corporation(Armonk，New York)供应的PowerPC处理器、pSeries服务器或xSeries服务器；由Hewlett Packard Co.(Palo Alto，California)供应的具有Intel Itanium II处理器的HP Superdome；和/或基于由International Business Machines Corporation、Hewlett Packard、Intel、Oracle 或其他供应的架构的其他机器。

[0036] 本地中央处理单元202包括在该环境内的处理期间使用的一个或多个本地寄存器210，诸如，一个或多个通用寄存器和/或一个或多个专用寄存器。这些寄存器包括表示在任

何特定时间点的环境状态的信息。

[0037] 此外，本地中央处理单元202执行存储于存储器204中的指令和程序代码。在一个特定实例中，中央处理单元执行存储于存储器204中的仿真器代码212。此代码使得以一个架构中配置的处理环境能够仿真另一架构。举例而言，仿真器代码212允许基于不同于z/Architecture的架构的机器(诸如，PowerPC处理器、pSeries服务器、xSeries服务器、HP Superdome服务器或其他)仿真z/Architecture和执行基于z/Architecture开发的软件和指令。

[0038] 参看图2B描述关于仿真器代码212的进一步细节。客户指令250包含经开发以欲于不同于本地CPU 202的架构的架构中执行的软件指令(例如，机器指令)。举例而言，客户指令250可能已经设计以在z/Architecture处理器102上执行，但相反，正在本地CPU 202(其可为(例如)Intel Itanium II处理器)上仿真客户指令250。在一个实例中，仿真器代码212包括指令取得单元252以自存储器204获得一个或多个客户指令250，并且可选地提供用于所获得的指令的本地缓冲。仿真器代码212也包括指令翻译例程254以确定已获得的客户指令的类型且将客户指令翻译成一个或多个对应本地指令256。此翻译包括(例如)识别要由客户指令执行的功能并选择(多个)本地指令以执行该功能。

[0039] 另外，仿真器212包括仿真控制例程260以使得执行本地指令。仿真控制例程260可使得本地CPU 202执行仿真一个或多个先前所获得的客户指令的本地指令的例程，并且，在这种执行完成时，将控制返回至指令取得例程以仿真下一个客户指令或群客户指令的获得。本地指令256的执行可包括将数据自存储器204载入至寄存器中；将数据自寄存器存储回至存储器；或执行某类型的算术或逻辑运算(如由翻译例程确定那样)。

[0040] 每个例程(例如)以软件来实施，该软件存储于存储器中且由本地中央处理单元202来执行。在其他实例中，例程或运算中的一个或多个以固件、硬件、软件或其某组合来实施。可使用本地CPU的寄存器210或通过使用存储器204中的位置仿真仿真处理器的寄存器。在实施例中，客户指令250、本地指令256和仿真器代码212可常驻于相同存储器中或可在不同存储器器件间分配。

[0041] 如本文中所使用，固件包括(例如)处理器的微码、毫码和/或宏码。固件包括(例如)在高级机器码的实施中使用的硬件级指令和/或数据结构。在一个实施例中，固件包括(例如)通常作为微码递送的专属码，该微码包括受信任软件或基础硬件所特有的微码且控制操作系统对系统硬件的存取。

[0042] 在一个实例中，所获得的、经翻译并经执行的客户指令250为本文中所描述的指令。自存储器取得是一个架构(例如，z/Architecture)的指令，将该指令翻译且表示为另一架构(例如，PowerPC、pSeries、xSeries、Intel等)的本地指令256的序列。然后执行这些本地指令。

[0043] 在一个实施例中，本文中所描述的指令为根据本发明的实施例提供的向量指令，其为向量工具的一部分。向量工具提供(例如)在自一个至十六个元素的范围内的固定大小的向量。每向量包括由工具中所定义的向量指令进行运算的数据。在一个实施例中，若向量由多个元素组成，则将每个元素与其他元素并行地处理。指令完成不出现，直至所有元素的处理完成。

[0044] 如本文中所描述，可将向量指令实施为包括(但不限于)z/Architecture、Power、

Intel等的各种架构的一部分。尽管本文中所描述的实施例针对z/Architecture，但向量指令和本发明的一个或多个实施例可基于许多其他架构。z/Architecture仅为实例。

[0045] 在将向量工具实施为z/Architecture的一部分的一个实施例中，为了使用向量寄存器和指令，将指定控制寄存器(例如，控制寄存器0)中的向量启用控制和寄存器控制设定为(例如)一。若安装了向量工具且在未设定启用控制的情况下执行向量指令，则看作为数据例外。若未安装向量工具且执行向量指令，则看作为操作例外。

[0046] 向量数据例如以与其他数据格式相同的自左至右顺序显现于储存器中。编号为0至7的数据格式的位构成储存器中的最左侧(最低编号)字节位置中的字节，位8至15形成下个顺序位置中的字节，等等。在另一实例中，向量数据可以另一顺序(诸如，自右至左)显现于储存器中。

[0047] 用向量工具提供的向量指令中的许多向量指令具有指定位的字段。被称作寄存器扩展位或RXB的此字段包括用于向量寄存器指明的操作数中的每一个的最高有效位。用于未由指令指定的寄存器指明的位将保留，且被设定为零。

[0048] 在一个实例中，RXB字段包括四个位(例如，位0至3)，且将位定义如下：

[0049] 0-用于指令的第一向量寄存器指明的最高有效位。

[0050] 1-用于指令的第二向量寄存器指明(若有的话)的最高有效位。

[0051] 2-用于指令的第三向量寄存器指明(若有的话)的最高有效位。

[0052] 3-用于指令的第四向量寄存器指明(若有的话)的最高有效位。

[0053] 由(例如)组译器取决于寄存器编号将每位设定为零或一。举例而言，对于寄存器0至15，将位设定为0；对于寄存器16至31，将位设定为1等等。

[0054] 在一个实施例中，每个RXB位为用于包括一个或多个向量寄存器的指令中的特定位置的扩展位。举例而言，在一个或多个向量指令中，RXB的位0为位置8至11的扩展位，其被指派至(例如)V_1；RXB的位1为位置12至15的扩展位，其被指派至(例如)V_2；等等。

[0055] 在另一实施例中，RXB字段包括额外位，且将一个以上位用作用于每个向量或位置的扩展。

[0056] 根据本发明的一个方面提供的包括RXB字段的指令为“向量载入至块边界”指令，其实例描绘于图3中。在一个实例中，“向量载入至块边界”指令300包括：操作码字段302a(例如，位0至7)、302b(例如，位40至47)，其指示“向量载入至块边界”操作；向量寄存器字段304(例如，位8至11)，其用以指明向量寄存器(V_1)；索引字段(X_2)306(例如，位12至15)；基本字段(B_2)308(例如，位16至19)；位移(displacement)字段(D_2)310(例如，位20至31)；掩码字段(M_3)312(例如，位32至35)；和RXB字段314(例如，位36至39)。在一个实例中，字段304至314中的每一个分开且独立于(多个)操作码字段。另外，在一个实施例中，这些字段分开且独立于彼此；然而，在其他实施例中，可组合一个以上字段。下文描述关于这些字段的使用的其他信息。

[0057] 在一个实例中，由操作码字段302a指明的操作码的选定位(例如，前两个位)指定指令的长度和格式。在此特定实例中，长度为三个半字，且格式为用扩展的操作码字段的向量寄存器和索引存储运算。向量(V)字段以及由RXB指定的其对应扩展位指明向量寄存器。具体地，对于向量寄存器，使用(例如)寄存器字段的四位字段(其中添加寄存器扩展位(RXB)作为最高有效位)指定含有操作数的寄存器。举例而言，若四位字段为0110且扩展位

为0，则五位字段00110指示寄存器编号6。

[0058] 与指令的字段相关联的下标编号表示该字段适用的操作数。举例而言，与V_1相关联的下标编号1表示第一操作数，等等。寄存器操作数的长度为一 个寄存器，其为(例如)128个位。

[0059] 在一个实例中，在“向量寄存器和索引储存器运算”指令中，将由X_2和B_2字段指明的通用寄存器的内容添加至D_2字段的内容以形成第二操作数地址。在一个实例中，将针对“向量载入至块边界”指令的位移D_2作为12-位无符号整数对待。

[0060] 在一个或者多个实施例中使用M_3字段，以确定要载入到的存储器中的边界(下文也称为边界大小)。例如，在由指令指定边界的一个实施例中，M_3字段指定用以用信号向CPU通知关于载入至的块边界的代码。若指定保留值，则看作为指定例外。实例码和对应值如下：

[0061]

码	边界
0	64-字节
1	128-字节
2	256-字节
3	512-字节
4	lK-字节
5	2K-字节
6	4K-字节

[0062] 然而，在由执行指令的处理器动态确定边界的另一实施例中，M_3字段作为实例包括边界类型(诸如高速缓冲存储器或者页边界)的指示。然后处理器基于该类型和一个或者多个处理器特性(诸如由处理器使用的高速缓冲存储器线或者页大小)确定边界大小。处理器可以使用大小的固定值或者可以动态确定大小。例如，如果M_3字段指示该类型是页边界，则处理器可以例如在翻译后备缓冲器中执行开始地址的表查找，以获得页大小。

[0063] 在另一实例中，不提供M_3字段，并且由指令的另一字段或者依据指令之外的控制，指示该类型。

[0064] 在“向量载入至块边界(VLBB)”指令的一个实施例的执行中，在一个实施例中，自左至右进行，以零索引字节元素开始，以来自第二操作数的字节载入第一操作数(在通过V_1字段加上扩展位指明的寄存器中指定)。第二操作数为通过第二操作数地址(也被称作开始地址)指明的存储器位置。载入自该存储器位置开始，且继续至由指令(或处理器)计算出的结束地址，如下文所描述。若遇到边界条件，则其为依赖于对待第一操作数的其余部分的方式 的模型。未看作为关于未载入的一个字节的存取例外。在一个实例中，未载入的字节不可预测。

[0065] 在上述实例指令中，开始地址通过索引寄存器值(X_2)+基寄存器值(B_2)+位移(D_2)来确定；然而，在其他实施例中，通过以下来提供开始地址：寄存器值；指令地址+指令文本指定偏移(offset)；寄存器值+位移；或寄存器值+索引寄存器值；(仅作为一些实例)。另外，在一个实施例中，指令不包括RXB字段。相反，不使用扩展或以另一方式提供扩展(诸如，自

指令外部的控制)，或提供扩展作为指令的另一字段的一部分。

[0066] 参看图4描述处理“向量载入至块边界”指令的一个实施例的进一步细节。在一个实例中，计算环境的处理器正执行此逻辑。

[0067] 在一个实施例中，最初，计算开始地址，其指示从其载入将开始的存储器中的位置(步骤400)。作为实例，可通过以下数据来提供开始地址402：寄存器值；指令地址加上指令文本指定偏移；寄存器值加上位移；寄存器值加上索引寄存器值；或寄存器值加上索引寄存器值加上位移。在本文中提供的指令中，由X_2字段、B_2字段和D_2字段来提供开始地址。即，将通过X_2和B_2指明的寄存器的内容添加至通过D_2指示的位移以提供开始地址。计算开始地址的上文所指示的方式仅为实例；其他实例也可能。

[0068] 此后，做出关于该边界是否要动态确定的确定(询问404)。如果否，则使用M_3字段中指定的值作为边界大小(BdySize)。否则，处理器动态确定边界大小(步骤406)。例如，M_3字段指定边界的类型，并且基于该类型和处理器的一个或者多个特性(例如，处理器的高速缓冲存储器线大小、处理器的页大小等)，处理器确定该边界。作为实例，基于该类型，处理器使用边界的固定大小(例如，处理器的预定义固定高速缓冲存储器线或者页大小)，或者基于该类型，处理器确定边界。例如，如果该类型是页边界，则处理器在TLB中查找开始地址，并且从其确定页边界。也存在其他实例。

[0069] 确定边界大小之后，要么动态要么通过指定的指令，建立用以确定对指定边界的接近性的边界掩码(BdyMask)(步骤410)。为了建立掩码，在一个实例中，采用边界大小(BdySize)(408)的2的补数否定数，从而建立边界掩码412(例如，BdyMask＝0-BdySize)。

[0070] 接下来，计算结束地址，其指示自何处停止载入(步骤420)。此计算的输入(例如)为边界大小408、开始地址402、向量大小414(例如，以字节计； 例如，16)和边界掩码412。在一个实例中，如下计算结束地址422：

[0071]

$$\text{EndAddress} = \min(\text{StartAddress} + (\text{BdySize} - (\text{StartAddress} \,\&\, \neg \text{BdyMask})), \text{StartAddress} + \text{vec\_size})。$$

[0072] 此后，以索引字节0开始，自始于开始地址且终止于结束地址的存储器载入第一操作数(即，指明的向量寄存器)(步骤430)。此情形使得能够将可变数目个字节自存储器载入至向量中，而不跨越指明的存储器边界。举例而言，若存储器边界在64个字节处，且开始地址在58个字节处，则在向量寄存器中载入字节58至64。

[0073] 根据本发明的实施例的要载入至向量寄存器中的数据的一个实例描绘于图5中。如所指示，无数据系经过由虚垂直线指明的边界载入。经过边界的位置不可存取且没有例外发生。在一个特定实施例中，自左至右载入向量。然而，在另一实施例中，可自右至左载入向量。在一个实施例中，在执行时(runtime)提供向量的方向-自左至右或自右至左。举例而言，作为实例，指令存取寄存器、状态控制或指示处理的方向为自左至右或自右至左的其他实体。在一个实施例中，不将此方向控制编码为指令的一部分，而是在执行时将其提供至指令。

[0074] 如本文描述，用来自主储存器的块内的数据的字节载入向量寄存器。块的边界被视为块的末尾。可以使用开始地址(StartAddress)和边界掩码计算块的开头。例如，由StartAddress AND BdyMask计算块的开头。

[0075] 上文描述是载入指令的一个实例。当载入数据(诸如,串数据)时,常常不知道该串是否会于页面边界之前结束。载入直到该边界而不跨越的能力通常需要首先检查串的结尾。一些实施也可具有对于跨越边界的处罚,且软件可能想要避免这些情形。因此,载入直到若干边界的能力有用。提供将可变数目个字节载入至向量寄存器中,同时确保不载入跨越指定边界的数据的指令。

[0076] 在一个实施例中,存在32个向量寄存器,且其他类型的寄存器可映射至该向量寄存器的象限。举例而言,如图6中所展示,若存在包括32个向量寄存器602的寄存器文件600且每寄存器的长度为128个位,则长度为64个位的16个浮点寄存器604可重叠这些向量寄存器。因此,作为实例,当修改浮点寄存器2时,然后也修改向量寄存器2。用于其他类型的寄存器的其他映射也可能。

[0077] 本文中,除非另有明确注释或由上下文注释,否则可互换地使用存储器、主存储器、储存器与主储存器。

[0078] 作为下文进一步描述的此实施方式的一部分提供关于向量工具的额外细节(包括其他指令的实例)。

[0079] 如本领域技术人员将了解,本发明的一个或多个实施例可体现为系统、方法或计算机程序产品。因此,本发明的一个或多个实施例可采用完全硬件实施例、完全软件实施例(包括固件、常驻软件、微码等)或组合软件与硬件实施例的实施例的形式,这些实施例在本文中大体上皆可被称作“电路”、“模块”或“系统”。此外,本发明的一个或多个实施例可采用体现于一个或多个计算机可读介质(其具有体现于其上的计算机可读程序代码)中的计算机程序产品的形式。

[0080] 可利用一个或多个计算机可读介质的任何组合。计算机可读介质可为计算机可读存储介质。举例而言,计算机可读存储介质可为(但不限于)电子、磁性、光学、电磁、红外线或半导体系统、装置或器件或前述中的任何合适组合。计算机可读存储介质的更特定实例(非详尽清单)包括以下:具有一个或多个电线的电连接、便携型计算机磁盘、硬盘、随机存取存储器(RAM)、只读存储器(ROM)、可擦除可编程只读存储器(EPROM或快闪存储器)、光纤、便携型光盘-只读存储器(CD-ROM)、光学存储器件、磁性存储器件或前述的任何合适组合。在此文件的上下文中,计算机可读存储介质可为可含有或存储供指令执行系统、装置或器件使用或结合指令执行系统、装置或器件而使用的程序的任何有形介质。

[0081] 现参看图7,在一个实例中,计算机程序产品700包括(例如)一个或多个非暂时性计算机可读存储介质702以在其上存储计算机可读程序代码构件或逻辑704,以提供和促进本发明的一个或多个实施例。

[0082] 可使用适当介质(包括(但不限于)无线、有线、光纤缆线、RF等或前述的任何合适组合)传输体现于计算机可读介质上的程序代码。

[0083] 可以一个或多个程序设计语言的任何组合来编写用于进行本发明的一个或多个方面的操作的计算机程序代码,该一个或多个程序设计语言包括诸如Java、Smalltalk、C++等的面向对象的程序设计语言,和诸如“C”程序设计语言、组译器或类似程序设计语言的常规程序性程序设计语言。程序代码可完全在用户的计算机上执行、部分地在用户的计算机上执行、作为独立软 件包而执行、部分地在用户的计算机上且部分地在远程计算机上执行,或完全在远程计算机或服务器上执行。在后者情形下,远程计算机可经由任一类型的网

络连接至用户的计算机，该任一类型的网络包括局域网(LAN)或广域网(WAN)，或可进行至外部计算机的连接(例如，使用因特网服务提供者经由因特网)。

[0084] 本文中参考根据本发明的实施例的方法、装置(系统)和计算机程序产品的流程图说明和/或方块图描述本发明的一个或多个方面。应理解，可由计算机程序指令来实施流程图说明和/或方块图的每个块和这些流程图说明和/或方块图中的块的组合。可将这些计算机程序指令提供至通用计算机、专用计算机或其他可编程数据处理装置的处理器以产生机器，以使得经由该计算机或其他可编程数据处理装置的处理器执行的指令产生用于实施流程图和/或方块图块中所指定的功能/动作的构件。

[0085] 这些计算机程序指令也可存储于计算机可读介质中，该计算机可读介质可指导计算机、其他可编程数据处理装置或其他器件以特定方式起作用，使得存储于该计算机可读介质中的指令产生制造物件，其包括实施流程图和/或方块图方块中所指定的功能/动作的指令。

[0086] 也可将这些计算机程序指令载入至计算机、其他可编程数据处理装置或其他器件上以使得在该计算机、其他可编程装置或其他器件上执行一系列操作步骤以产生计算机实施的处理程序，使得在该计算机或其他可编程装置上执行的指令提供用于实施流程图和/或方块图块中所指定的功能/动作的处理程序。

[0087] 这些图中的流程图和方块图说明根据本发明的一个或多个方面的各种实施例的系统、方法和计算机程序产品的可能实施的架构、功能性和操作。就此而言，流程图或方块图中的每个块可表示模块、区段或代码的一部分，其包含用于实施(多个)指定逻辑功能的一个或多个可执行指令。也应注意，在一些替代实施中，块中所注释的功能可能不以诸图中所注释的次序发生。举例而言，取决于所涉及的功能性，实际上可实质上同时执行连续展示的两个块，或有时可以相反次序执行这些块。也将注意，方块图和/或流程图说明的每个块和方块图和/或流程图说明中的块的组合可由执行指定功能或动作的基于专用硬件的系统或专用硬件与计算机指令的组合来实施。

[0088] 除上述内容之外，本发明的一个或多个实施例也可由供应客户环境的管 理的服务提供者来提供、供应、部署、管理、服务等。举例而言，服务提供者可为一个或多个客户建立、维护、支持等执行本发明的一个或多个实施例的计算机代码和/或计算机基础结构。作为回报，作为实例，服务提供者可在订用和/或收费协议下自客户收取付款。此外或替代地，服务提供者可自广告内容销售至一个或多个第三方而收取付款。

[0089] 在本发明的一个方面中，可部署用于执行本发明的一个或多个方面的应用程序。作为实例，应用程序的部署包含提供可操作以执行本发明的一个或多个方面的计算机基础结构。

[0090] 作为本发明的另一方面，可部署计算基础结构，其包含将计算机可读代码整合至计算系统内，其中与该计算系统组合的代码能够执行本发明的一个或多个方面。

[0091] 作为本发明的另一方面，可提供用于整合计算基础结构的处理，其包含将计算机可读代码整合至计算机系统中。计算机系统包含计算机可读介质，其中计算机介质包含本发明的一个或多个方面。与该计算机系统组合的代码能够执行本发明的一个或多个方面。

[0092] 尽管上文描述了各种实施例，但这些仅为实例。举例而言，其他架构的计算环境可并入且使用本发明的一个或多个方面。另外，可使用其他大小的寄存器，且在不脱离本发明

的精神的情况下，可作出对指令的改变。

[0093] 另外，其他类型的计算环境可受益于本发明的一个或多个方面。作为实例，适合于存储和/或执行程序代码的数据处理系统为可使用，其包括直接或经由系统总线间接耦接至存储器元件的至少两个处理器。这些存储器元件包括(例如)在程序代码的实际执行期间使用的本地存储器、大容量储存器和高速缓冲存储器，高速缓冲存储器提供至少一些程序代码的临时存储以便减少在执行期间必须自大容量储存器取得代码的次数。

[0094] 输入/输出端或I/O器件(包括(但不限于)键盘、显示器、指点器件、DASD、磁带、CD、DVD、随身碟(thumb drive)和其他存储器介质等)可直接或经由介入的I/O控制器而耦接至系统。网络适配器也可耦接至系统以使得数据处理系统能够经由介入的私有或公用网络而变得耦接至其他数据处理系统或远程打印机或储存器件。调制解调器、电缆调制解调器和以太网络卡仅为可用类型的网络适配器中的少数几种。

[0095] 参看图8，描绘实施本发明的一个或多个方面的主机计算机系统5000的 代表性组件。代表性主机计算机5000包含与计算机存储器(即，中央储存器)5002通信的一个或多个CPU 5001，以及用于与其他计算机或SAN等通信的至存储介质器件5011和网络5010的I/O接口。CPU 5001符合具有构建的指令集和构建的功能性的架构。CPU 5001可具有动态地址翻译(DAT)5003，以用于将程序地址(虚拟地址)变换成存储器的真实地址。DAT通常包括翻译后备缓冲器(TLB)5007以用于高速缓冲翻译，使得对计算机存储器5002的块的稍后存取不需要延迟地址翻译。通常，高速缓冲存储器5009用于计算机存储器5002与处理器5001之间。高速缓冲存储器5009可为层级式的，其具有可用于一个以上CPU的大型高速缓冲存储器和在大型高速缓冲存储器与每个CPU之间的较小较快(较低层级)高速缓冲存储器。在一些实施中，将较低层级高速缓冲存储器分割以提供用于指令取得和数据存取的单独的低层级高速缓冲存储器。在一个实施例中，由指令取得单元5004经由高速缓冲存储器5009自存储器5002取得指令。在指令解码单元5006中解码指令，且将指令(在一些实施例中，其他指令)分派给一个或多个指令执行单元5008。通常使用若干个执行单元5008，例如，算术执行单元、浮点执行单元和分支指令执行单元。由执行单元执行指令，从而按需要自指令指定的寄存器或存储器存取操作数。若将自存储器5002存取(载入或存储)操作数，则载入/存储单元5005通常在正被执行的指令的控制下处置存取。可在硬件电路中或在内部微码(固件)中或由两者的组合来执行指令。

[0096] 如所注释，计算机系统包括在本地(或主)储存器中的信息，以及寻址、保护和参考和改变记录。寻址的一些方面包括地址的格式、地址空间的概念、地址的各种类型，以及将地址的一种类型翻译至地址的另一类型的方式。主储存器中的一些储存器包括永久指派的存储位置。主储存器为系统提供数据的可直接寻址的快速存取存储。在可处理数据和程序两者之前将数据和程序两者载入至主储存器中(自输入器件)。

[0097] 主储存器可包括一个或多个较小的快速存取缓冲储存器(有时称为高速缓冲存储器)。高速缓冲存储器通常与CPU或I/O处理器相关联。通过程序大体上不可观测到不同存储介质的实体构造和用途的效应(除了对性能的效应外)。

[0098] 可维持针对指令和针对数据操作数的单独的高速缓冲存储器。将高速缓冲存储器内的信息以相邻字节维持于被称为高速缓冲块或高速缓冲存储器 线(或简称为线)的整体边界上。模型可提供“EXTRACT CACHE ATTRIBUTE(提取高速缓冲存储器属性)”指令，其返回

以字节计的高速缓冲存储器线的大小。模型也可提供“PREFETCH DATA(预先取得数据)”和“PREFETCH DATA RELATIVE LONG(预取得数据相对长度)”指令，其实现储存器至数据或指令高速缓冲存储器中的预取得或数据自高速缓冲存储器的释放。

[0099] 将储存器视为长水平位串。对于多数操作，对储存器的存取以自左至右顺序进行。将该位的串再分成八个位的单元。八位单元被称为字节，其为所有信息格式的基本建立块。储存器中的每个字节位置通过唯一非负整数来识别，该唯一非负整数为该字节位置的地址或简称为字节地址。邻近字节位置具有连续地址，其以左侧的0开始且以自左至右顺序进行。地址为无符号二进制整数，且为24个、31个或64个位。

[0100] 在储存器与CPU或通道子系统之间一次一字节或字节群组地传输信息。除非另有指定，否则，在(例如)z/Architecture中，储存器中的字节群组由该群组的最左侧字节来寻址。通过要执行的操作隐含或明确指定该群组中的字节的数目。当在CPU操作中使用时，字节群组被称为字段。在每个字节群组内，在(例如)z/Architecture中，以自左至右顺序对位编号。在z/Architecture中，最左侧位有时被称作“高阶”位，且最右侧位有时被称作“低阶”位。然而，位编号不是存储地址。可仅寻址字节。为了对储存器中的字节的单独位操作，存取整个字节。将字节中的位自左至右编号为0至7(在(例如)z/Architecture中)。对于24位地址，可将地址中的位编号为8至31或40至63，或对于31位地址，可将地址中的位编号为1至31或33至63；对于64位地址，可将地址中的位编号为0至63。在多个字节的任何其他固定长度的格式内，自0开始对构成该格式的位连续地编号。为了错误检测的目的，且优选地为了校正的目的，可将一个或多个检查位与每字节或与字节群组一起传输。由机器自动地产生这种检查位，且不可直接由程序来控制这种检查位。以字节的数目来表达存储容量。当通过指令的操作码隐含存储操作数字段的长度时，认为该字段具有固定长度，其可为个、两个、四个、八个或十六个字节。对于一些指令，可隐含更大字段。当不隐含而是明确叙述存储操作数字段的长度时，认为该字段具有可变长度。可变长度操作数的长度可以一个字节的增量变化(或对于一些指令，以两个字节的倍数或其他倍数的增量变化)。当 将信息放置于储存器中时，替换包括于指明的字段中的仅那些字节位置的内容，即使至储存器的物理路径的宽度可能大于正存储的字段的长度也如此。

[0101] 某些信息单元将在储存器中的整体边界上。当其存储地址为该单元的长度(以字节计)的倍数时，将边界称为信息单元的整体。对整体边界上的2个、4个、8个和16个字节的字段给予特殊名称。半字为在两字节边界上的两个连续字节的群组，且为指令的基本建立块。字为四字节边界上的四个连续字节的群组。双字为八字节边界上的八个连续字节的群组。四倍字为16字节边界上的16个连续字节的群组。当储存器地址指明半字、字、双字和四倍字时，地址的二进制表示分别含有一个、两个、三个或四个最右侧零位。指令将在两-字节整体边界上。多数指令的存储操作数不具有边界对准要求。

[0102] 在实施针对指令和数据操作数的单独的高速缓冲存储器的器件上，若程序存储至高速缓冲存储器线中(随后自该高速缓冲存储器线取得指令)，则可经历显著延迟，而不管储存器是否更改随后取得的指令。

[0103] 在一个实施例中，可通过软件(有时称为经授权内部码、固件、微码、毫码、微微码(pico-code)等，前述中的任一个将与本发明的一个或多个方面一致)来实践本发明。参看图8，体现本发明的一个或多个方面的软件程序代码可由主机系统5000的处理器5001自长

期存储介质器件5011(诸如,CD-ROM驱动、磁带机或硬驱动)存取。软件程序代码可体现于多种已知介质中的任一个上,以用于供数据处理系统(诸如,磁片、硬驱动或CD-ROM)使用。该代码可分布于这种介质上,或可经由网络5010自计算机系统的计算机存储器5002或储存器至其他计算机系统地分发至用户,以供这种其他系统的用户使用。

[0104] 软件程序代码包括操作系统,其控制各种计算机组件和一个或多个应用程序的功能和交互。通常将程序代码自存储介质器件5011分页至相对较高速计算机储存器5002,在该相对较高速计算机储存器5002处,程序代码可用于由处理器5001处理。用于在存储器中、在物理介质上体现软件程序代码和/或经由网络分发软件程序代码的技术和方法已熟知,且在本文中将不进一步加以讨论。当在有形介质(包括(但不限于)电子存储器模块(RAM)、快闪存储器、光盘(CD)、DVD、磁带等)上建立和存储程序代码时,程序代码常常被称作“计算机程序产品”。计算机程序产品介质通常可由优选在计算机系统中的处理电路来读取,以用于由处理电路执行。

[0105] 图9说明可实践本发明的一个或多个方面的代表性工作站或服务器硬件系统。图9的系统5020包含代表性基础计算机系统5021(诸如,个人计算机、工作站或服务器),包括选用的外围器件。基础计算机系统5021包括一个或多个处理器5026和总线,该总线用以根据已知技术连接处理器5026与系统5021的其他组件并实现(多个)处理器5026与系统5021的其他组件之间的通信。总线将处理器5026连接至存储器5025和长期储存器5027,长期储存器5027可包括(例如)硬驱动(包括(例如)磁性介质、CD、DVD和快闪存储器中的任一个)或磁带驱动。系统5021也可能包括用户接口适配器,用户接口适配器经由总线将微处理器5026连接至或多个接口器件(诸如,键盘5024、鼠标5023、打印机/扫描仪5030和/或其他接口器件),这些接口器件可为诸如触敏式屏幕、数字化键入板(entry pad)等的任何用户接口器件。总线也经由显示器适配器将诸如LCD屏幕或监视器的显示器件5022连接至微处理器5026。

[0106] 系统5021可借助于能够与网络5029通信(5028)的网络适配器与其他计算机或计算机网络通信。实例网络适配器为通信频道、令牌环、以太网络或调制解调器。替代地,系统5021可使用诸如CDPD(蜂窝式数字分组数据)卡的无线接口通信。系统5021可与局域网(LAN)或广域网(WAN)中的这种其他计算机相关联,或系统5021可为具有另一计算机的客户端/服务器布置中的客户端等。所有这些配置以及适当通信硬件和软件为本领域已知。

[0107] 图10说明可实践本发明的一个或多个方面的数据处理网络5040。数据处理网络5040可包括多个单独网络(诸如,无线网络和有线网络),这些网络中的每一个可包括多个单独工作站5041、5042、5043、5044。另外,如本领域技术人员将了解,可包括一个或多个LAN,其中LAN可包含耦接至主机处理器的多个智能型工作站。

[0108] 仍参看图10,网络也可包括大型计算机或服务器,诸如,网关计算机(客户端服务器5046)或应用程序服务器(远程服务器5048,其可存取数据储存库且也可自工作站5045直接存取)。网关计算机5046充当至每个单独网络的入口点。当将一个联网协议连接至另一联网络协议时,需要网关。网关5046可优选借助于通信链路耦接至另一网络(例如,因特网5047)。也可使用通信链路将网关5046直接耦接至一个或多个工作站5041、5042、5043、5044。可利用可购自International Business Machines Corporation的IBM eServer™ System z服务器来实施网关计算机。

[0109] 同时参看图9和图10，可体现本发明的一个或多个实施例的软件程序代码可由系统5020的处理器5026自长期存储介质5027(诸如，CD-ROM驱动或硬驱动)存取。软件程序代码可体现于多种已知介质中的任一个上，以用于供数据处理系统(诸如，磁盘、硬驱动或CD-ROM)使用。程序代码可分发于这种介质上，或可经由网络自一个计算机系统的存储器或储存器至其他计算机系统地分发至用户5050、5051，以供这种其他系统的用户使用。

[0110] 可替代地，程序代码可体现于存储器5025中，且由处理器5026使用处理器总线来存取。此程序代码包括操作系统，其控制各种计算机组件和一个或多个应用程序5032的功能和交互。通常将程序代码自存储介质5027分页至高速存储器5025，在高速存储器5025处，程序代码可用于由处理器5026处理。用于在存储器中、在物理介质上体现软件程序代码和/或经由网络分发软件程序代码的技术和方法已熟知且在本文中将不进一步加以讨论。当在有形介质(包括(但不限于)电子存储器模块(RAM)、快闪存储器、光盘(CD)、DVD、磁带等)上建立和存储程序代码时，程序代码常常被称作“计算机程序产品”。计算机程序产品介质通常可由优选在计算机系统中的处理电路来读取，以用于由处理电路执行。

[0111] 最容易用于处理器的高速缓冲存储器(通常比处理器的其他高速缓冲存储器快且小)为最低(L1或层级一)高速缓冲存储器，且主储存器(主存储器)为最高层级高速缓冲存储器(若存在3个层级，则为L3)。常常将最低层级高速缓冲存储器划分成保持要执行的机器指令的指令高速缓冲存储器(I-高速缓冲存储器)和保持数据操作数的数据高速缓冲存储器(D-高速缓冲存储器)。

[0112] 参看图11，针对处理器5026描绘例示性处理器实施例。通常，使用高速缓冲存储器5053的一个或多个层级缓冲存储器块以便改进处理器性能。高速缓冲存储器5053为保持有可能使用的存储器数据的高速缓冲存储器线的高速缓冲器。典型的高速缓冲存储器线为64个、128个或256个字节的存储器数据。除用于对数据进行高速缓冲外，单独的高速缓冲存储器也常常用于对指令进行高速缓冲。常常通过本领域中熟知的各种“窥探(snoop)”算法来提供高速缓冲一致性(存储器和高速缓冲存储器中的线的复制的同步)。处理器系统的主存储器储存器5025常常被称作高速缓冲存储器。在具有4个层级的高速缓冲存储器5053的处理器系统中，主储存器5025有时被称作层级5(L5)高速缓冲存储器，这是因为其通常较快且仅保持可用于计算机系统的非易失 性储存器(DASD、磁带等)的部分。主储存器5025对由操作系统分页入和分页出主储存器5025的数据页“进行高速缓冲”。

[0113] 程序计数器(指令计数器)5061保持追踪要执行的当前指令的地址。z/Architecture处理器中的程序计数器为64个位，且可经截断至31或24个位以支持先前寻址限制。程序计数器通常体现于计算机的PSW(程序状态字)中，使得程序计数器在上下文切换期间持续。因此，具有程序计数器值的进行中程序可由(例如)操作系统来中断(自程序环境至操作系统环境的上下文切换)。在程序不起作用时，程序的PSW维持程序计数器值，且在操作系统正执行时，使用操作系统的程序计数器(在PSW中)。通常，以等于当前指令的字节的数目的量来使程序计数器递增。RISC(精简指令集计算)指令通常为固定长度，而CISC(复杂指令集计算)指令通常为可变长度。IBM z/Architecture的指令为具有2个、4个或6个字节长度的CISC指令。举例而言，通过上下文切换操作或分支指令的分支选取操作来修改程序计数器5061。在上下文切换操作中，将当前程序计数器值连同关于正执行的程序的其他状态信息(诸如，条件码)一起保存于程序状态字中，且载入新程序计数器值从而指向要执行

的新程序模块的指令。执行分支选取操作以便通过将分支指令的结果载入至程序计数器5061中而准许程序作出决策或在程序内循环。

[0114] 通常，指令读取单元5055用以代表处理器5026取得指令。取得单元取得“下一个顺序指令”、分支选取指令的目标指令或在上下文切换之后的程序的第一指令。现代指令取得单元常常使用预取得技术以基于可能使用经预先取得的指令的可能性而推测性地预取得指令。举例而言，取得单元可取得包括下一个顺序指令的指令的16个字节和再下一个顺序指令的额外字节。

[0115] 然后由处理器5026执行所取得的指令。在一个实施例中，将(多个)所取得的指令传递至取得单元的分派单元5056。分派单元解码(多个)指令且将关于(多个)经解码的指令的信息转递至适当单元5057、5058、5060。执行单元5057通常将自指令取得单元5055接收关于经解码的算术指令的信息，且将根据指令的操作码对操作数执行算术运算。优选自存储器5025、构建的寄存器5059或自正执行的指令的立即字段，将操作数提供至执行单元5057。当存储执行的结果时，将执行的结果存储于存储器5025、寄存器5059中或其他机器硬件(诸如，控制寄存器、PSW寄存器等)中。

[0116] 处理器5026通常具有用于执行指令的功能的一个或多个单元5057、5058、5060。参看图12A，执行单元5057可借助于接口逻辑5071与构建的通用寄存器5059、解码/分派单元5056、载入存储单元5060和其他处理器单元5065通信。执行单元5057可使用若干个寄存器电路5067、5068、5069以保持算术逻辑单元(ALU)5066将进行运算的信息。ALU执行算术运算(诸如，加法、减法、乘法和除法)以及逻辑函数(诸如，“和”(and)、“或”(or)和“异或”(XOR)、旋转和移位)。优选地，ALU支持依赖于设计的专业化运算。其他电路可提供其他构建的工具5072，包括(例如)条件码和恢复支持逻辑。通常，将ALU运算的结果保持于输出寄存器电路5070中，输出寄存器电路5070可将结果转递至多种其他处理功能。存在处理器单元的许多布置，本发明描述仅意欲提供对一个实施例的代表性理解。

[0117] “加法”指令(例如)将在具有算术和逻辑功能性的执行单元5057中执行，而浮点指令(例如)将在具有专业化的浮点能力的浮点执行中执行。优选地，执行单元通过对操作数执行操作码定义的功能而对由指令识别的操作数进行运算。举例而言，“加法”指令可由执行单元5057对在由指令的寄存器字段识别的两个寄存器5059中发现的操作数执行。

[0118] 执行单元5057对两个操作数执行算术加法，且将结果存储于第三操作数中，其中第三操作数可为第三寄存器或两个源寄存器中的一个。执行单元优选利用算术逻辑单元(ALU)5066，算术逻辑单元(ALU)5066能够执行多种逻辑函数(诸如，移位、旋转、“和”(And)、“或”(Or)和“异或”(XOR))和多种代数函数(包括加法、减法、乘法、除法中的任一个)。一些ALU 5066经设计以用于标量运算，且一些ALU 5066经设计以用于浮点运算。取决于架构，数据可为大端法(Big Endian)(其中最低有效字节处于最高字节地址)或小端法(Little Endian)(其中最低有效字节处于最低字节地址)。IBM z/Architecture为大端法。取决于架构，无符号的字段可为符号和量值(1的补数或2的补数)。2的补数是有利的，因为ALU并不需要设计减法能力，这由于在ALU中，2的补数中的负值或正值仅需要加法。通常以速记法来描述数字，其中12位字段定义4,096字节块的地址，且通常描述为(例如)4Kbyte(千字节)块。

[0119] 参看图12B，用于执行分支指令的分支指令信息通常发送至分支单元5058，分支单元5058常常使用分支预测算法(诸如，分支历史表5082)以在其他条件运算完成之前预测分

支的结果。将取得当前分支指令的目标，且在条件运算完成之前推测性地执行当前分支指令的目标。当完成条件运算时，基 于条件运算的条件和所推测的结果，完成或放弃推测性执行的分支指令。典型分支指令可测试条件码，且在条件码满足分支指令的分支要求的情况下分支至目标地址，可基于包括(例如)在寄存器字段或指令的立即字段中发现的数字的若干个数字而计算目标地址。分支单元5058可使用具有多个输入寄存器电路5075、5076、5077和输出寄存器电路5080的ALU 5074。举例而言，分支单元5058可与通用寄存器5059、解码分派单元5056或其他电路5073通信。

[0120] 指令群组的执行可对于包括(例如)以下的多种原因被中断：由操作系统启动的上下文切换、引起上下文切换的程序例外或错误、引起上下文切换的I/O中断信号或多个程序的多线程活动(在多线程化环境中)。优选地，上下文切换动作保存关于当前正执行的程序的状态信息，且然后载入关于正被调用的另一程序的状态信息。举例而言，可将状态信息保存于硬件寄存器中或存储器中。状态信息优选包含指向要执行的下一个指令的程序计数器值、条件码、存储器翻译信息和构建的寄存器内容。上下文切换活动可单独或组合地通过硬件电路、应用程序、操作系统程序或固件程序代码(微码、微微码或经授权内部码(LIC))来训练。

[0121] 处理器根据指令定义的方法来存取操作数。指令可使用指令的一部分的值来提供立即操作数，可提供明确指向通用寄存器或专用寄存器(例如，浮点寄存器)的一个或多个寄存器字段。指令可利用通过操作码字段识别为操作数的隐含的寄存器。指令可将存储器位置用于操作数。操作数的存储器位置可由寄存器、立即字段或寄存器与立即字段的组合来提供，如通过z/Architecture长位移工具(long displacement facility)举例说明，其中指令定义(例如)相加在一起以提供操作数在存储器中的地址的基寄存器、索引寄存器和立即字段(位移字段)。除非另有指示，否则本文中的位置通常隐含主存储器(主储存器)中的位置。

[0122] 参看图12C，处理器使用载入/存储单元5060来存取储存器。载入/存储单元5060可通过获得目标操作数在存储器5053中的地址且在寄存器5059或另一存储器5053的位置中载入操作数来执行载入操作，或可通过获得目标操作数在存储器5053中的地址且将自寄存器5059或另一存储器5053的位置获得的数据存储于存储器5053中的目标操作数位置中来执行存储操作。载入/存储单元5060可为推测性的，且可以相对于指令顺序而言无序的顺序存取存储 器，然而，载入/存储单元5060对于程序维持按次序执行指令的显现。载入/存储单元5060可与通用寄存器5059、解码/分派单元5056、高速缓冲存储器/存储器接口5053或其他元件5083通信，且包含各种寄存器电路、ALU 5085和控制逻辑5090以计算存储地址且提供管线定序以保持操作按次序。一些操作可能为无序的，但载入/存储单元提供使得无序操作对于程序显现为已按次序执行的功能性，如本领域中所熟知。

[0123] 优选地，应用程序“看见”的地址常常被称作虚拟地址。虚拟地址有时被称作“逻辑地址”和“有效地址”。这些虚拟地址为虚拟是因为：它们通过多种动态地址翻译(DAT)技术中的一个而重新导向至物理存储器位置，这些DAT技术包括(但不限于)仅对虚拟地址加偏移值作为前缀、经由一个或多个翻译表翻译虚拟地址，翻译表优选单独或组合地包含至少段表和页表，优选地，段表具有指向页表的入口。在z/Architecture中，提供翻译层级，包括区第一表、区第二表、区第三表、段表和选用的页表。常常通过利用翻译后备缓冲器(TLB)

(其包含将虚拟地址映射至相关联的物理存储器位置的入口)来改进地址翻译的性能。当DAT使用翻译表翻译虚拟地址时,建立这些入口。然后,虚拟地址的随后使用可利用快速TLB的入口,而不是缓慢依序翻译表存取。可通过包括LRU(最近最少使用)的多种替换演算法来管理TLB内容。

[0124] 在处理器为多处理器系统的处理器的状况下,每处理器具有保持诸如I/O、高速缓冲存储器、TLB和存储器的共用资源互锁以达成一致性的责任。通常,在维持高速缓冲一致性中将利用"窥探"技术。在窥探环境中,可将每个高速缓冲存储器线标记为处于以下状态中的任一个以便促进共用:共用状态、互斥状态、改变的状态、无效状态等。

[0125] I/O单元5054(图11)为处理器提供用于附接至外围器件(例如,包括磁带、光盘、打印机、显示器和网络)的构件。I/O单元常常由软件驱动程序呈现至计算机程序。在大型计算机(诸如,来自IBM®的System z®)中,通道适配器和开放系统适配器为大型计算机的I/O单元,这些I/O单元提供操作系统与外围器件之间的通信。

[0126] 另外,其他类型的计算环境可受益于本发明的一个或多个方面。作为实例,环境可包括仿真器(例如,软件或其他仿真机制),在该仿真器中仿真特定架构(包括(例如)指令执行、构建的功能(诸如,地址翻译)和构建的寄存器)或其子集(例如,在具有处理器和存储器的本地计算机系统上)。在这种环境 中,仿真器的一个或多个仿真功能可实施本发明的一个或多个方面,即使执行该仿真器的计算机可具有不同于正仿真能力的架构也如此。作为一个实例,在仿真模式下,解码特定指令或正仿真操作,且构建适当仿真功能以实施单独指令或操作。

[0127] 在仿真环境中,主机计算机包括(例如):存储器,其存储指令和数据;指令取得单元,其自存储器取得指令且可选地提供所取得的指令的本地缓冲;指令解码单元,其接收所取得的指令且确定已取得的指令的类型;和指令执行单元,其执行这些指令。执行可包括:将数据自存储器载入至寄存器中;将数据自寄存器存储回至存储器;或执行某类型的算术或逻辑运算(如由解码单元确定)。在一个实例中,以软件来实施每个单元。举例而言,将正由这些单元执行的操作实施为仿真器软件内的一个或多个子例程。

[0128] 更具体地,在大型计算机中,构建的机器指令常常借助于编译应用程序而由程序员(现今通常为"C"程序员)使用。存储于存储介质中的这些指令可本地地在z/Architecture IBM®服务器中或者在执行其他架构的机器中执行。可在现有和未来IBM®大型计算机服务器中和在IBM®的其他机器(例如,Power Systems服务器和System x®服务器)上仿真这些指令。可于在使用由 IBM®、Intel®、AMD™和其他制造的硬件的广泛多种机器上执行Linux的机器中执行这些指令。除了在z/Architecture下在该硬件上执行外,也可使用Linux,以及使用由Hercules、UMX或FSI(Fundamental Software,Inc)进行的仿真机器,其中执行大体上处于仿真模式下。在仿真模式下,由本地处理器执行仿真软件以仿真仿真处理器的架构。

[0129] 本地处理器通常执行包含固件或本地操作系统的仿真软件以执行仿真处理器的仿真。仿真软件负责取得和执行仿真处理器架构的指令。仿真软件维持仿真程序计数器以追踪指令边界。仿真软件一次可取得一个或多个仿真机器指令,且将该一个或多个仿真机器指令转换至对应的本地机器指令群组,以用于由本地处理器执行。可对这些经转换的指

令进行高速缓冲，使得可实现较快速转换。尽管如此，仿真软件仍将维持仿真处理器架构的架构规则以便确保操作系统和针对仿真处理器编写的应用程序正确地操作。此外，仿真软件将提供通过仿真处理器架构识别的资源(包括(但不限于)控制寄存器、通用寄存器、浮点寄存器、包括(例如)段表和页表的动态地址翻译功能、中断机制、上下文切换机制、当日时间(TOD)时钟和至I/O子系统的构建的接 口)，使得操作系统或经设计以在仿真处理器上执行的应用程序可在具有仿真软件的本地处理器上执行。

[0130] 解码正进行仿真特定指令，且调用子例程以执行个别指令的功能。仿真仿真处理器的功能的仿真软件功能(例如)按以下来实施："C"子例程或驱动程序，或在理解优选实施例的描述之后将在本领域的技术人员的技术内的提供用于特定硬件的驱动程序的某其他方法。包括(但不限于)以下的各种软件和硬件仿真专利说明达成针对不同机器构建的指令格式用于可用于本领域技术人员的目标机器的仿真的多种已知方式：Beausoleil等人的题为"Multiprocessor for Hardware Emulation"的美国专利证书第5,551,013号；和Scalzi等人的题为"Preprocessing of Stored Target Routines for Emulating Incompatible Instructions on a Target Processor"的美国专利证书第6,009,261号；和Davidian等人的题为"Decoding Guest Instruction to Directly Access Emulation Routines that Emulate the Guest Instructions"的美国专利证书第5,574,873号；和Gorishek等人的题为"Symmetrical Multiprocessing Bus and Chipset Used for Coprocessor Support Allowing Non-Native Code to Run in a System"的美国专利证书第6,308,255号；和Lethin等人的题为"Dynamic Optimizing Object Code Translator for Architecture Emulation and Dynamic Optimizing Object Code Translation Method"的美国专利证书第6,463,582号；和Eric Traut的题为"Method for Emulating Guest Instructions on a Host Computer Through Dynamic Recompilation of Host Instructions"的美国专利证书第5,790,825号(前述专利证书中的每一个在此以其全文引用的方式并入本文中)；和许多其他。

[0131] 在图13中，提供仿真主机计算机系统5092的实例，其仿真主机架构的主机计算机系统5000'。在仿真主机计算机系统5092中，主机处理器(CPU)5091为仿真主机处理器(或虚拟主机处理器)，且包含仿真处理器5093，其具有不同于主机计算机5000'的处理器5091的本地指令集架构的本地指令集架构。仿真主机计算机系统5092具有仿真处理器5093可存取的存储器5094。在实例实施例中，将存储器5094分割成主机计算机存储器5096部分和仿真例程5097部分。主机计算机存储器5096可用于根据主机计算机架构的仿真主机计算机5092的程序。仿真处理器5093执行不同于仿真处理器5091的本地指令的架构的构建的指令集的本地指令，这些本地指令自仿真例程存储器5097获得，且 可通过使用在序列和存取/解码例程中获得的一个或多个指令自主机计算机存储器5096中的程序存取主机指令以用于执行，序列和存取/解码例程可解码(多个)所存取的主机指令以确定用于仿真所存取的主机指令的功能的本地指令执行例程。举例而言，针对主机计算机系统5000'的架构定义的其他工具可通过构建的工具例程来仿真，包括诸如通用寄存器、控制寄存器、动态地址翻译和I/O子系统支持和处理器高速缓冲存储器的工具。仿真例程也可利用可用于仿真处理器5093中的功能(诸如，通用寄存器和虚拟地址的动态翻译)以改进仿真例程的性能。也可提供特殊硬件和卸载引擎以辅助处理器5093仿真主机计算机5000'的功能。

[0132] 本文中所使用的术语仅用于描述特定实施例的目的，且并不意欲为本发明的限制。如本文中所使用，除非上下文另有清晰指示，否则单数形式“一”、“一个”和“该”意欲也包括复数形式。应进一步理解，当术语“包含”和/或“包括”用于此说明书中时，其指定所叙述特征、整数、步骤、操作、元件和/或组件的存在，但并不排除一个或多个其他特征、整数、步骤、操作、元件、组件和/或其群组的存在或添加。

[0133] 以下权利要求书中的所有构件或步骤加功能元件的对应结构、材料、动作和等效物(若有的话)意欲包括用于结合如特别主张的其他所主张元件执行功能的任何结构、材料或动作。已出于说明和描述的目的呈现本发明的一个或多个方面的描述，但该描述并不意欲为详尽的或限于所公开的形式下的本发明。在不脱离本发明的范畴和精神的情况下，许多修改和变化对于本领域技术人员将为显而易见。选择并描述了实施例以便最佳地解释本发明的原理和实践应用，且使其他本领域技术人员能够针对具有如适合于所预期的特定用途的各种修改的各种实施例来理解本发明。

[0134] 向量串工具

[0135] ⋮

[0136] 指令

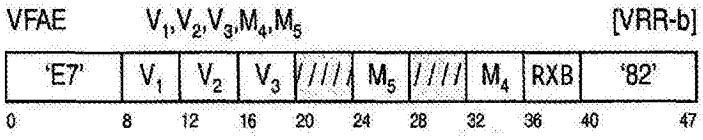
[0137] 除非另有指定，否则所有操作数为向量寄存器操作数。汇编器语法中的“V”指明向量操作数。

[0138]

名称	记忆码	特性					操作码	页
向量寻找任何相等	VFAE	VRR-b	C*	VF	a^9	SP	Dv	E782	23-1
向量寻找元素相等	VFEE	VRR-b	C*	VF	a^9	SP	Dv	E780	23-2
向量寻找元素不相等	VFENE	VRR-b	C*	VF	a^9	SP	Dv	E781	23-3
向量串范围比较	VSTRC	VRR-d	C*	VF	a^9	SP	Dv	E78A	23-4

[0139] 向量寻找任何相等

[0140]



[0141] 自左至右进行，比较第二操作数的每个无符号二进制整数元素与第三操作数的每个无符号二进制整数元素的相等性，且若在M_5字段中设定了零搜索标志，则视情况与零比较。

[0142] 若M_5字段中的结果类型(RT)标志为零，则对于匹配第三操作数中的任一元素或视情况匹配零的第二操作数中的每个元素，将第一操作数中对应的元素的位位置设定为一，否则，将其设定为零。

[0143] 若M_5字段中的结果类型(RT)标志为一，则将匹配第三操作数中的元素或零的第二操作数中的最左侧元素的字节索引存储于第一操作数的字节七中。

[0144] 每个指令具有扩展的记忆码区段，其描述推荐的扩展的记忆码及其对应的机器组译器语法。

[0145] 程序设计注释：对于视情况设定条件码的所有指令，若设定条件码，则性能可能降

级。

[0146] 若M_5字段中的结果类型(RT)标志为一且未发现字节相等,或为零(若设定了零搜索标志),则将与向量中的字节的数目相等的索引存储于第一操作数的字节七中。

[0147] M_4字段指定元素大小控制(ES)。ES控制指定向量寄存器操作数中的元素的大小。若指定保留值,则看作为指定例外。

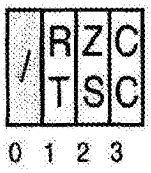
[0148] 0-字节

[0149] 1-半字

[0150] 2-字

[0151] 3至15-保留

[0152] M_5字段具有以下格式:

[0153] 

[0154] 如下定义M_5字段的位:

[0155] • 结果类型(RT):若为零,则每个所得元素为关于该元素的所有范围比较的掩码。若为一,则将字节索引存储至第一操作数的字节七中,且将零存储于所有其他元素中。

[0156] • 零搜索(ZS):若为一,则也将第二操作数的每个元素与零比较。

[0157] • 条件码设定(CC):若为零,则不设定条件码且条件码保持不变。若为一,则如以下段中所指定来设定条件码。

[0158] 特殊条件

[0159] 若出现以下中的任一个,则看作指定例外且不采取其他行动:

[0160] 1.M_4字段含有自3至15的值。

[0161] 2.M_5字段的位0不是零。

[0162] 所得条件码:

[0163] 若CC标志为零,则码保持不变。

[0164] 若CC标志为一,则如下来设定码:

[0165] 0若设定了ZS-位,则在第二操作数中比零低的索引元素中不存在匹配。

[0166] 1第二操作数的一些元素匹配第三操作数中的至少一个元素。

[0167] 2第二操作数的所有元素匹配第三操作数中的至少一个元素。

[0168] 3第二操作数中没有元素匹配第三操作数中的任何元素。

[0169] 程序例外:

[0170] 1具有DXC FE的数据,向量寄存器

[0171] • 在未安装向量扩展工具的情况下的操作

[0172] • 指定(保留的ES值)

[0173] • 事务约束(Transaction Constraint)

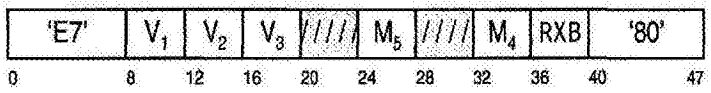
[0174] 扩展的记忆码:

[0175]

VFAEB V_1,V_2,V_3,M_5	VFAE $V_1,V_2,V_3,0,M_5$
VFAEH V_1,V_2,V_3,M_5	VFAE $V_1,V_2,V_3,1,M_5$
VFAEF V_1,V_2,V_3,M_5	VFAE $V_1,V_2,V_3,2,M_5$
VFAEBS V_1,V_2,V_3,M_5	VFAE $V_1,V_2,V_3,0,(M_5 \mid X'1')$
VFAEHS V_1,V_2,V_3,M_5	VFAE $V_1,V_2,V_3,1,(M_5 \mid X'1')$
VFAEFS V_1,V_2,V_3,M_5	VFAE $V_1,V_2,V_3,2,(M_5 \mid X'1')$
VFAEZB V_1,V_2,V_3,M_5	VFAE $V_1,V_2,V_3,0,(M_5 \mid X'2')$
VFAEZH V_1,V_2,V_3,M_5	VFAE $V_1,V_2,V_3,1,(M_5 \mid X'2')$
VFAEZF V_1,V_2,V_3,M_5	VFAE $V_1,V_2,V_3,2,(M_5 \mid X'2')$
VFAEZBS V_1,V_2,V_3,M_5	VFAE $V_1,V_2,V_3,0,(M_5 \mid X'3')$
VFAEZHS V_1,V_2,V_3,M_5	VFAE $V_1,V2,V3,1,(M_5 \mid X'3')$
VFAEZFS V_1,V_2,V_3,M_5	VFAE $V_1,V_2,V_3,2,(M_5 \mid X'3')$

[0176] 向量寻找元素相等

[0177] VFEE V_1,V_2,V_3,M_4,M_5 [VRR-b]]



[0178] 自左至右进行，将第二操作数的无符号二进制整数元素与第三操作数的对应的无符号二进制整数元素比较。若两个元素相等，则将最左侧相等元素的第一字节的字节索引放置于第一操作数的字节七中。将零存储于第一操作数的剩余字节中。若没有发现字节相等，或若没有发现字节为零(若设定了零比较)，则将与向量中的字节的数目相等的索引存储于第一操作数的字节七中。将零存储于剩余字节中。

[0179] 若在M_5字段中设定了零搜索(ZS)位，则也比较第二操作数中的每个元素与零的相等性。若在发现第二操作数和第三操作数的任何其他元素相等之前在第二操作数中寻找到零元素，则将发现为零的元素的第一字节的字节索引存储于第一操作数的字节七中，且将零存储于所有其他字节位置中。若条件码设定(CC)标志为一，则将条件码设定为零。

[0180] M_4字段指定元素大小控制(ES)。ES控制指定向量寄存器操作数中的元素的大小。若指定保留值，则看作为指定例外。

[0181] 0-字节

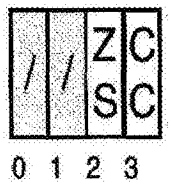
[0182] 1-半字

[0183] 2-字

[0184] 3至15-保留

[0185] M_5字段具有以下格式：

[0186]



[0187] 如下定义M_5字段的位：

[0188] • 保留：保留位0至1且位0至1必须为零。否则，看作为指定例外。

[0189] • 零搜索(ZS):若为一,则也将第二操作数的每个元素与零比较。

[0190] • 条件码设定(CC):若为零,则条件码保持不变。若为一,则如在以下段中所指定来设定条件码。

[0191] 特殊条件

[0192] 若出现以下中的任一个,则看作指定例外且不采取其他行动:

[0193] 1.M_4字段含有自3至15的值。

[0194] 2.M_5字段的位0至1不是零。

[0195] 所得条件码:

[0196] 若将M_5字段的位3设定为一,则如下设定码:

[0197] 0若设定了零比较位,则比较在具有比任何相等比较小的索引的元素中检测到第二操作数中的零元素。

[0198] 1比较在一些元素中检测到第二操作数与第三操作数之间的匹配。若设定了零比较位,则此匹配出现于具有小于或等于零比较元素的索引的元素中。

[0199] 2--

[0200] 3没有元素比较起来相等。

[0201] 若M_5字段的位3为零,则码保持不变。

[0202] 程序例外:

[0203] • 具有DXC FE的数据,向量寄存器

[0204] • 在未安装向量扩展工具的情况下的操作

[0205] • 指定(保留的ES值)

[0206] • 事务约束

[0207] 扩展的记忆码:

	VFEEB V_1,V_2,V_3,M_5	VFEE $V_1,V_2,V_3,0,M_5$
	VFEEH V_1,V_2,V_3,M_5	VFEE $V_1,V_2,V_3,1,M_5$
	VFEEF V_1,V_2,V_3,M_5	VFEE $V_1,V_2,V_3,0,(M_5 \mid X'1')$
	VFEEHS V_1,V_2,V_3,M_5	VFEE $V_1,V_2,V_3,1,(M_5 \mid X'1')$
	VFEEFS V_1,V_2,V_3,M_5	VFEE $V_1,V_2,V_3,2,(M_5 \mid X'1')$
[0208]	VFEEZB V_1,V_2,V_3,M_5	VFEE $V_1,V_2,V_3,0,(M_5 \mid X'2')$
	VFEEZH V_1,V_2,V_3,M_5	VFEE $V_1,V_2,V_3,1,(M_5 \mid X'2')$
	VFEEZF V_1,V_2,V_3,M_5	VFEE $V_1,V_2,V_3,2,(M_5 \mid X'2')$
	VFEEZBS V_1,V_2,V_3,M_5	VFEE $V_1,V_2,V_3,0,(M_5 \mid X'3')$
	VFEEZHS V_1,V_2,V_3,M_5	VFEE $V_1,V_2,V_3,1,(M_5 \mid X'3')$
	VFEEZFS V_1,V_2,V_3,M_5	VFEE $V_1,V_2,V_3,2,(M_5 \mid X'3')$

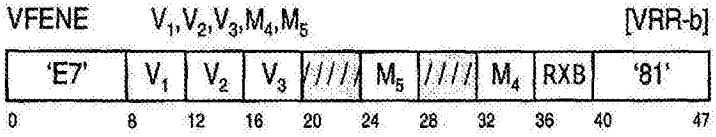
[0209] 程序设计注释:

[0210] 1.对于任一元素大小,始终将字节索引存储至第一操作数中。举例而言,若将元素大小设定为半字且比较出第2个索引的半字相等,则将存储字节索引4。

[0211] 2.第三操作数不应含有具有零值的元素。若第三操作数确实含有零且在任何其他相等比较之前与第二操作数中的零元素匹配,则设定条件码,而不管零比较位设定。

[0212] 向量寻找元素不相等

[0213]



[0214] 自左至右进行，将第二操作数的无符号二进制整数元素与第三操作数的对应的无符号二进制整数元素比较。若两个元素不相等，则将最左侧不相等元素的字节索引放置于第一操作数的字节七中，且将零存储至所有其他字节。若将M_5字段中的条件码设定(CC)位设定为一，则设定条件码以指示哪一操作数较大。若所有元素相等，则将等于向量大小的字节索引放置于第一操作数的字节七中，且将零放置于所有其他字节位置中。若CC位为一，则设定条件码三。

[0215] 若在M_5字段中设定了零搜索(ZS)位，则也比较第二操作数中的每个元素 与零的相等性。若在发现第二操作数的任一其他元素不相等之前在第二操作数中寻找到零元素，则将发现为零的元素的第一字节的字节索引存储于第一操作数的字节七中。将零存储于所有其他字节中，且设定条件码0。

[0216] M_4字段指定元素大小控制(ES)。ES控制指定向量寄存器操作数中的元素的大小。若指定保留值，则看作为指定例外。

[0217] 0-字节

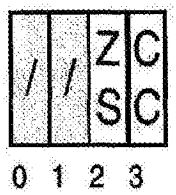
[0218] 1-半字

[0219] 2-字

[0220] 3至15-保留

[0221] M_5字段具有以下格式：

[0222]



[0223] 如下定义M_5字段的位：

[0224] • 零搜索(ZS)：若为一，则也将第二操作数的每个元素与零比较。

[0225] • 条件码设定(CC)：若为零，则不设定条件码且条件码保持不变。若为一，则如以下段中所指定来设定条件码。

[0226] 特殊条件

[0227] 若出现以下中的任一个，则看作指定例外且不采取其他行动：

[0228] 1.M_4字段含有自3至15的值。

[0229] 2.M_5字段的位0至1不是零。

[0230] 所得条件码：

[0231] 若将M_5字段的位3设定为一，则如下设定码：

[0232] 0若设定了零比较位，则比较在比任何不相等比较低的索引元素中检测到两个操作数中的零元素

[0233] 1检测到元素失配，且VR2中的元素小于VR3中的元素

[0234] 2检测到元素失配，且VR2中的元素大于VR3中的元素

[0235] 3所有元素比较起来相等，且若设定了零比较位，则在第二操作数中没有寻找到零元素。

[0236] 若M_5字段的位3为零，则码保持不变。

[0237] 程序例外：

[0238] • 具有DXC FE的数据，向量寄存器

[0239] • 在没有安装向量扩展工具的情况下的操作

[0240] • 指定(保留的ES值)

[0241] • 事务约束

[0242] 扩展的记忆码：

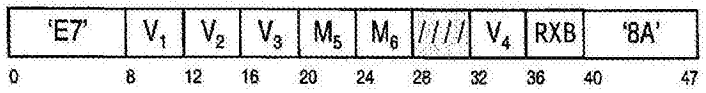
[0243]

VFENEB V_1,V_2,V_3,M_5	VFENE $V_1,V_2,V_3,0,M_5$
VFENEH V_1,V_2,V_3,M_5	VFENE $V_1,V_2,V_3,1,M_5$
VFENEF V_1,V_2,V_3,M_5	VFENE $V_1,V_2,V_3,2,M_5$
VFENEBS V_1,V_2,V_3,M_5	VFENE $V_1,V_2,V_3,0,(M_5 \mid X'1')$
VFENEHS V_1,V_2,V_3,M_5	VFENE $V_1,V_2,V_3,1,(M_5 \mid X'1')$
VFENEFS V_1,V_2,V_3,M_5	VFENE $V_1,V_2,V_3,2,(M_5 \mid X'1')$
VFENEZB V_1,V_2,V_3,M_5	VFENE $V_1,V_2,V_3,0,(M_5 \mid X'2')$
VFENEZH V_1,V_2,V_3,M_5	VFENE $V_1,V_2,V_3,1,(M_5 \mid X'2')$
VFENEZF V_1,V_2,V_3,M_5	VFENE V1,V2,V3,2,$(M_5 \mid X'2')$
VFENEZBS V_1,V_2,V_3,M_5	VFENE $V_1,V_2,V_3,0,(M_5 \mid X'3')$
VFENEZHS V_1,V_2,V_3,M_5	VFENE $V_1,V_2,V_3,1,(M_5 \mid X'3')$
VFENEZFS V_1,V_2,V_3,M_5	VFENE $V_1,V_2,V_3,2,(M_5 \mid X'3')$

[0244] 向量串范围比较

[0245]

VSTRC V_1,V_2,V_3,V_4,M_5,M_6 [VRR-d]



[0246] 自左至右进行，将第二操作数中的无符号二进制整数元素与由第三操作数和第四操作数中的偶数-奇数对元素定义的值范围比较。结合来自第四操作数的控制值定义要执行的比较的范围。若元素匹配由第三操作数和第四操作数指定的范围中的任一个，则将其视为匹配。

[0247] 若M_6字段中的结果类型(RT)标志为零，则若第一操作数中对应于第二操作数中正进行比较的元素的元素匹配这些范围中的任一个，则将该元素的位位置设定为一，否则，将其设定为零。

[0248] 若将M_6字段中的结果类型(RT)标志设定为一，则第二操作数中匹配由第 三操作数和第四操作数指定的范围中的任一个或零比较(若将ZS标志设定为一)的第一元素的字节索引放置于第一操作数的字节七中，且将零存储于剩余字节中。若没有元素匹配，则将等于向量中的字节的数目的索引放置于第一操作数的字节七中，且将零存储于剩余字节中。

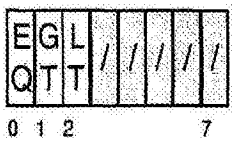
[0249] M_6字段中的零搜索(ZS)标志，若设定为一，则将第二操作数元素与零的比较添加

至由第三操作数和第四操作数提供的范围。若为在比任何其他真比较低的索引元素中的零比较，则将条件码设定为零。

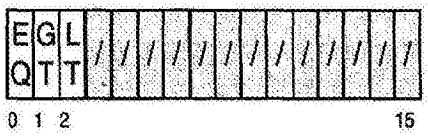
[0250] 这些操作数含有由M_5字段中的元素大小控制指定的大小的元素。

[0251] 第四操作数元素具有以下格式：

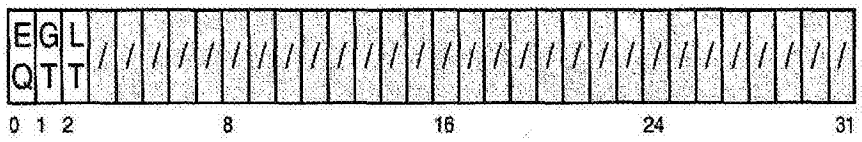
[0252] 若ES等于0：

[0253] 

[0254] 若ES等于1：

[0255] 

[0256] 若ES等于2：

[0257] 

[0258] 如下定义第四操作数元素中的位：

[0259] • 相等(EQ)：当为一时，进行相等性比较。

[0260] • 大于(GT)：当为一时，执行大于比较。

[0261] • 小于(LT)：当为一时，执行小于比较。

[0262] • 保留所有其他位且所有其他位应为零以确保未来相容性。

[0263] 可以任何组合使用这些控制位。若这些位都没有经设定，则比较将始终产生假结果。若所有这些位经设定，则比较将始终产生真结果。

[0264] M_5字段指定元素大小控制(ES)。ES控制指定向量寄存器操作数中的元素的大小。若指定保留值，则看作为指定例外。

[0265] 0-字节

[0266] 1-半字

[0267] 2-字

[0268] 3至15-保留

[0269] M_6字段具有以下格式：

[0270] 

[0271] 如下定义M_6字段的位：

[0272] • 反转结果(IN)：若为零，则继续进行与控制向量中的成对值的比较。若为一，则反转在这些范围中的这些成对的比较的结果。

[0273] • 结果类型(RT)：若为零，则每个所得元素为关于该元素的所有范围比较的掩码。

若为一，则将索引存储至第一操作数的字节七中。将零存储于剩余字节中。

[0274] • 零搜索(ZS)：若为一，则也将第二操作数的每个元素与零比较。

[0275] • 条件码设定(CC)：若为零，则不设定条件码且条件码保持不变。若为一，则如以下段中所指定来设定条件码。

[0276] 特殊条件

[0277] 若出现以下中的任一个，则看作指定例外且不采取其他行动：

[0278] 1.M_4字段含有自3至15的值。

[0279] 所得条件码：

[0280] 0若ZS＝1且在比任何比较低的索引元素中发现零

[0281] 1发现比较

[0282] 2--

[0283] 3没有发现比较

[0284] 程序例外：

[0285] • 具有DXC FE的数据，向量寄存器

[0286] • 在没有安装向量扩展工具的情况下的操作

[0287] • 指定(保留的ES值)

[0288] • 事务约束

[0289] 扩展的记忆码：

[0290]

VSTRCB V_1,V_2,V_3,V_4,M_6	VSTRC $V_1,V_2,V_3,V_4,0,M_6$
VSTRCH V_1,V_2,V_3,V_4,M_6	VSTRC $V_1,V_2,V_3,V_4,1,M_6$
VSTRCF V_1,V_2,V_3,V_4,M_6	VSTRC $V_1,V_2,V_3,V_4,2,M_6$
VSTRCBS V_1,V_2,V_3,V_4,M_6	VSTRC $V_1,V_2,V_3,V_4,0,(M_6 \mid X'1')$
VSTRCHS V_1,V_2,V_3,V_4,M_6	VSTRC $V_1,V_2,V_3,V_4,1,(M_6 \mid X'1')$
VSTRCFS V_1,V_2,V_3,V_4,M_6	VSTRC $V_1,V_2,V_3,V_4,2,(M_6 \mid X'1')$
VSTRCZB V_1,V_2,V_3,V_4,M_6	VSTRC $V_1,V_2,V_3,V_4,0,(M_6 \mid X'2')$
VSTRCZH V_1,V_2,V_3,V_4,M_6	VSTRC $V_1,V_2,V_3,V_4,1,(M_6 \mid X'2')$
VSTRCZF V_1,V_2,V_3,V_4,M_6	VSTRC $V_1,V_2,V_3,V_4,2,(M_6 \mid X'2')$
VSTRCZBS V_1,V_2,V_3,V_4,M_6	VSTRC $V_1,V_2,V_3,V_4,0,(M_6 \mid X'3')$
VSTRCZHS V_1,V_2,V_3,V_4,M_6	VSTRC $V_1,V_2,V_3,V_4,1,(M_6 \mid X'3')$
VSTRCZFS V_1,V_2,V_3,V_4,M_6	VSTRC $V_1,V_2,V_3,V_4,2,(M_6 \mid X'3')$

[0291]

	VR2 →	A	b	C	d	e	F	1	2
GE	A	T	T	T	T	T	T	F	F
LE	Z	T	F	T	F	F	T	F	F
GE	a	F	T	F	T	T	F	F	F
LE	c	T	T	T	F	F	T	T	T
LE	4	F	F	F	F	F	F	T	T
GE	0	T	T	T	T	T	T	T	T
EQ	d	F	F	F	T	F	F	F	F
EQ	d	F	F	F	T	F	F	F	F
VR4↑	VR3 ↑								
IN=0	VR1 (a)→	FFFF	FFFF	FFFF	FFFF	0000	FFFF	FFFF	FFFF
IN=1	VR1 (a)→	0000	0000	0000	0000	FFFF	0000	0000	0000
IN=0	VR1(b)→	0000	0000	0000	0000				
IN=1	VR1(b)→	0000	0000	0000	0008				
					index				

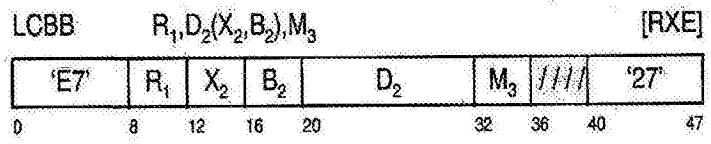
[0292] ES＝1,ZS＝0

[0293] VR1 (a) RT＝0的结果

[0294] VR1 (b) RT＝1的结果

[0295] 将计数载入至块边界

[0296]



[0297] 将含有可能自第二操作数位置载入而不与指定块边界跨越的字节的数目的32-位无符号二进制整数(覆盖于十六处)放置于第一操作数中。

[0298] 将位移作为12-位无符号整数对待。

[0299] 第二操作数地址不用以寻址数据。

[0300] M_3字段指定用以用信号向CPU通知关于块边界大小以计算载入的可能字节的数目的码。若指定保留值,则看作为指定例外。

[0301]

码	边界
0	64-字节
1	128-字节
2	256-字节
3	512-字节
4	1K-字节
5	2K-字节
6	4K-字节
7至15	保留

[0302] 所得条件码:

[0303] 0 操作数一为十六

[0304] 1 --

[0305] 2 --

[0306] 3 操作数一小于十六

[0307] 所得条件码：

[0308] 程序例外：

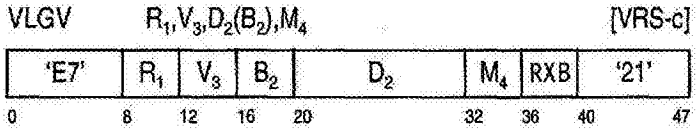
[0309] • 在没有安装向量扩展工具的情况下的操作

[0310] • 指定

[0311] 程序设计注释：期望结合向量载入至块边界(VECTOR LOAD TO BLOCK BOUNDARY)使用将计数载入至块边界(LOAD COUNT TO BLOCK BOUNDARY)以确定载入的字节的数目。

[0312] 自VR元素的向量载入GR

[0313]



[0314] 将具有由M4字段中的ES值指定的大小且通过第二操作数地址编索引的第三操作数的元素放置于第一操作数位置中。第三操作数为向量寄存器。第 一操作数为通用寄存器。若通过第二操作数地址指定的索引大于具有指定元素大小的第三操作数中最高编号的元素，则第一操作数中的数据为不可预测。

[0315] 若向量寄存器元素小于双字，则该元素在64-位通用寄存器中恰好对准，且零填充剩余位。

[0316] 第二操作数地址不用以寻址数据；相反，该地址的最右侧12个位用以指定元素在第二操作数内的索引。

[0317] M4字段指定元素大小控制(ES)。ES控制指定向量寄存器操作数中的元素的大小。若指定保留值，则看作为指定例外。

[0318] 0-字节

[0319] 1-半字

[0320] 2-字

[0321] 3-双字

[0322] 4至15-保留不变。

[0323] 所得条件码：码不变。

[0324] 程序例外：

[0325] • 具有DXC FE的数据，向量寄存器

[0326] • 在没有安装向量扩展工具的情况下的操作

[0327] • 指定(保留的ES值)

[0328] • 事务约束

[0329] 扩展的记忆码：

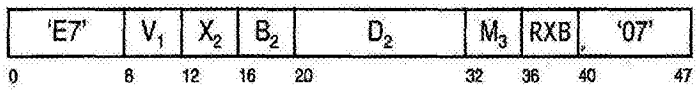
[0330]

VLGVB R1,V3,D2(B2)	VLGV R1,V3,D2(B2),0
VLGVH R1,V3,D2(B2)	VLGV R1,V3,D2(B2),1
VLGVF R1,V3,D2(B2)	VLGV R1,V3,D2(B2),2
VLGVG R1,V3,D2(B2)	VLGV R1,V3,D2(B2),3

[0331] 向量载入至块边界

VLBB $V_1,D_2(X_2,B_2),M_3$ [VRX]

[0332]



[0333] 以零索引字节元素开始,以来自第二操作数的字节载入第一操作数。若遇到边界条件,则第一操作数的其余部分为不可预测。没有看作为关于未载 入的字节的存取例外。

[0334] 将针对VLBB的位移作为12-位无符号整数对待。

[0335] M_3字段指定用以用信号向CPU通知关于用以载入至的块边界大小的码。若指定保留值,则看作为指定例外。

[0336]

码	边界
0	64-字节
1	128-字节
2	256-字节
3	512-字节
4	1K-字节
5	2K-字节
6	4K-字节
7至15	保留

[0337] 所得条件码:码保持不变。

[0338] 程序例外:

[0339] • 存取(取得,操作数2)

[0340] • 具有DXC FE的数据,向量寄存器

[0341] • 在没有安装向量扩展工具的情况下的操作

[0342] • 指定(保留的块边界码)

[0343] • 事务约束

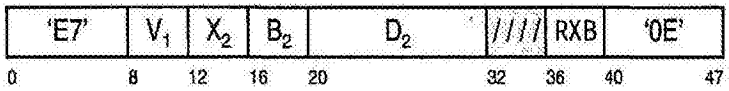
[0344] 程序设计注释:

[0345] 1.在某些情况下,可经过块边界载入数据。然而,若不存在关于该数据的存取例外,则将仅发生此情形。

[0346] 向量存储

VST $V_1,D_2(X_2,B_2)$ [VRX]

[0347]



[0348] 将第一操作数中的128-位值存储至通过第二操作数指定的存储位置。将针对VST的位移作为12-位无符号整数对待。

[0349] 所得条件码:码保持不变。

[0350] 程序例外:

[0351] • 存取(存储,操作数2)

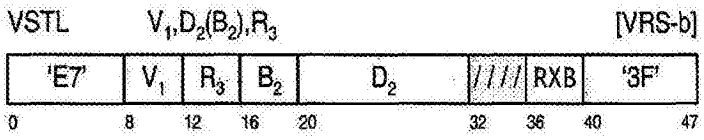
[0352] • 具有DXC FE的数据，向量寄存器

[0353] • 在没有安装向量扩展工具的情况下的操作

[0354] • 事务约束

[0355] 具有长度的向量存储

[0356]



[0357] 自左至右进行，将来自第一操作数的字节存储于第二操作数位置处。指定第三操作数的通用寄存器含有32-位无符号整数，该整数含有表示存储的最高索引字节的值。若第三操作数含有大于或等于向量的最高字节索引的值，则存储第一操作数的所有字节。

[0358] 仅看作为关于存储的字节的存取例外。

[0359] 将针对具有长度的向量存储(VECTOR STORE WITH LENGTH)的位移作为12-位无符号整数对待。

[0360] 所得条件码：条件码保持不变。

[0361] 程序例外：

[0362] • 存取(存储，操作数2)

[0363] • 具有DXC FE的数据，向量寄存器

[0364] • 在没有安装向量扩展工具的情况下的操作

[0365] • 事务约束

[0366] RXB描述

[0367] 所有向量指令具有在标注为RXB的指令的位36至40中的字段。此字段含有用于所有向量寄存器指明的操作数的最高有效位。保留用于没有通过指令指定的寄存器指明的位且应将其设定为零；否则，程序在未来无法相容地操作。将最高有效位串接至四-位寄存器指明的左侧以建立五-位向量寄存器指明。

[0368] 如下定义这些位：

[0369] 0.在指令的位8至11中用于向量寄存器指明的最高有效位。

[0370] 1.在指令的位12至15中用于向量寄存器指明的最高有效位。

[0371] 2.在指令的位16至19中用于向量寄存器指明的最高有效位。

[0372] 3.在指令的位32至35中用于向量寄存器指明的最高有效位。

[0373] 向量启用控制

[0374] 若将控制寄存器零中的向量启用控制(位46)和AFP寄存器控制(位45)二者设定为一，则可仅使用向量寄存器和指令。若安装了向量工具且在没有设定启用位的情况下执行向量指令，则看作为具有DXC FE十六进制的数据例外。若没有安装向量工具，则看作为操作例外。

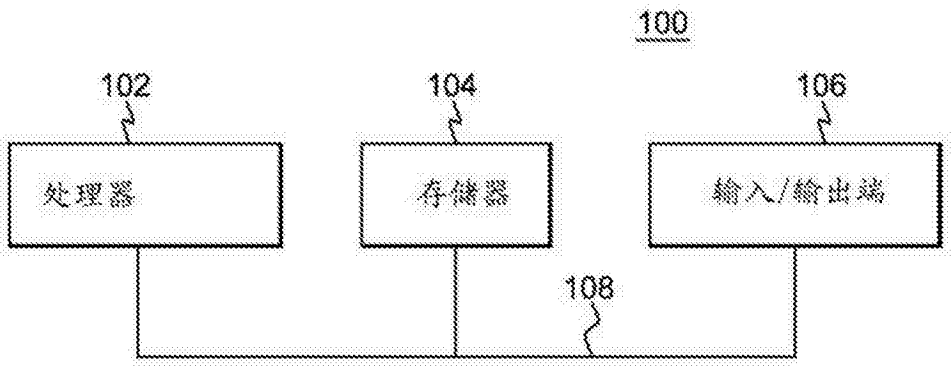

100
102
104
106
处理器
存储器
输入/输出端
108

图1

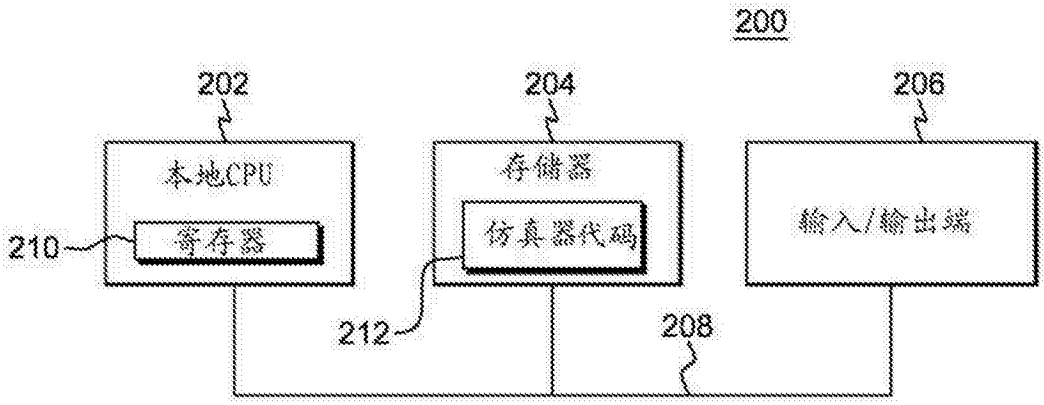

200
202
204
206
本地CPU
存储器
输入/输出端
210
寄存器
仿真器代码
212
208

图2A

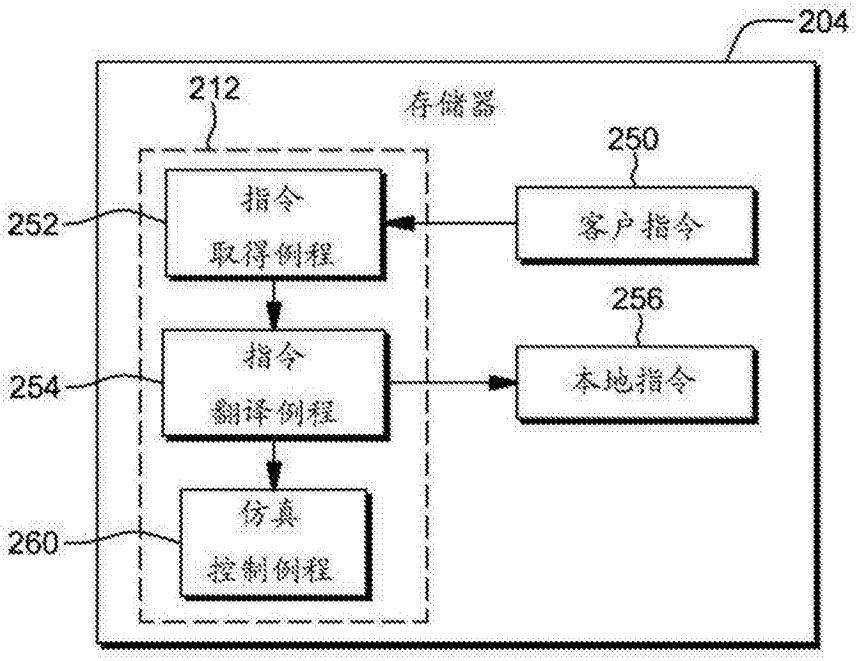

204
212
存储器
250
指令
取得例程
252
客户指令
256
指令
翻译例程
254
本地指令
仿真
控制例程
260

图2B

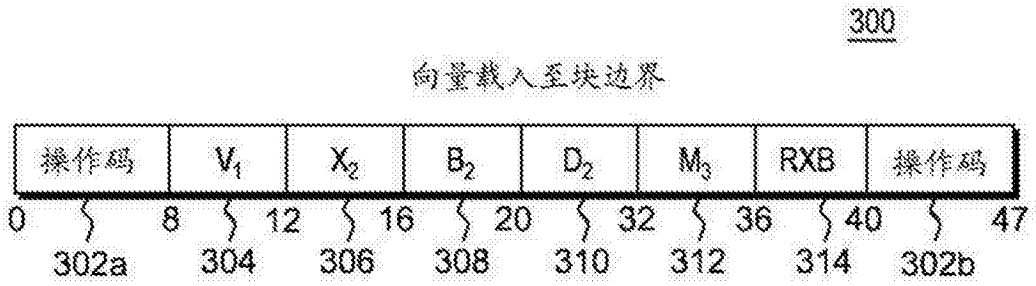

300
向量载入至块边界
操作码
V_1
X_2
B_2
D_2
M_3
RXB
操作码
0
8
12
16
20
32
36
40
47
302a
304
306
308
310
312
314
302b

图3

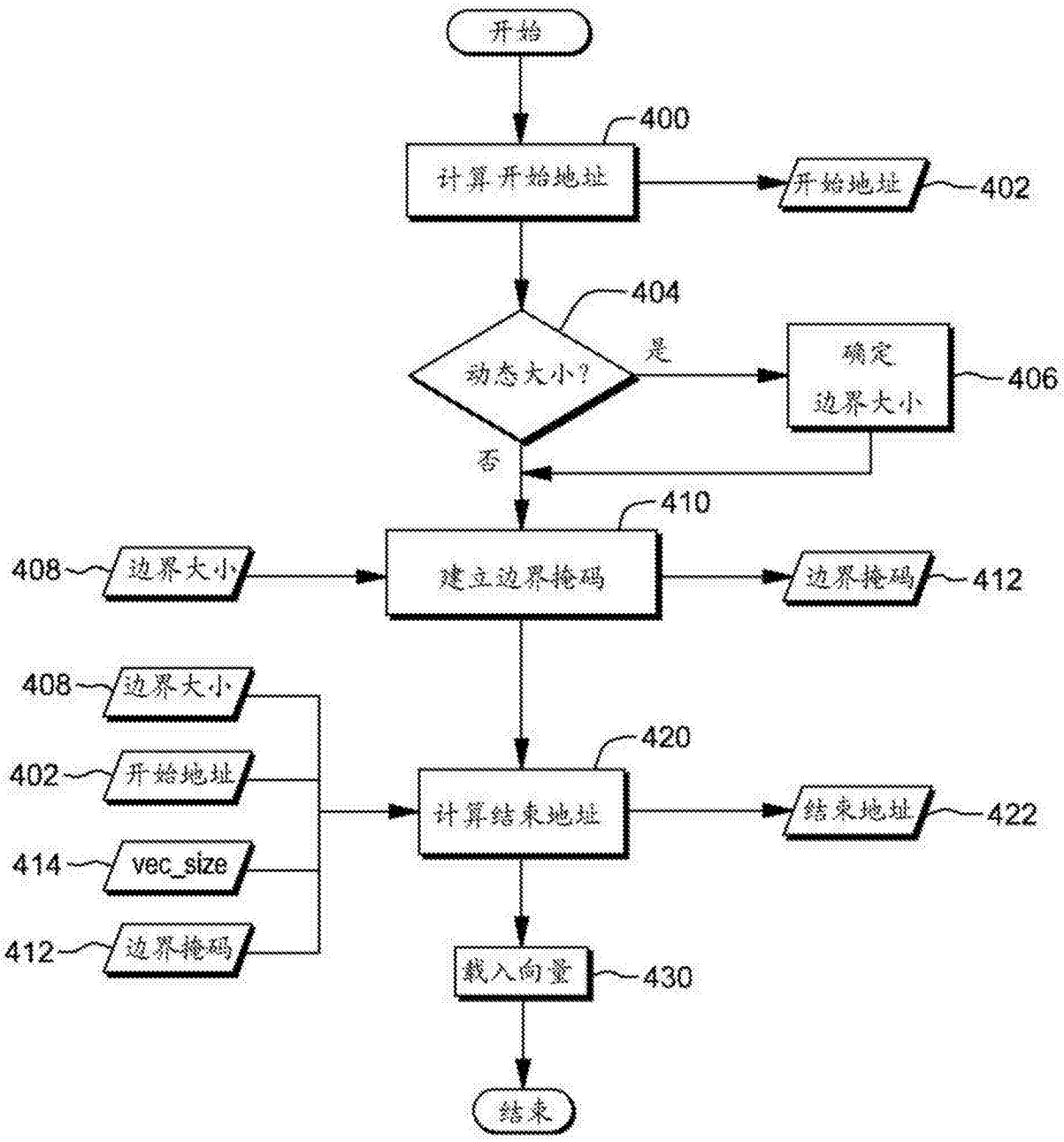

开始
400
计算开始地址
开始地址
402
404
动态大小?
是
确定
边界大小
406
否
410
408
边界大小
建立边界掩码
边界掩码
412
408
边界大小
420
402
开始地址
计算结束地址
结束地址
422
414
vec_size
412
边界掩码
载入向量
430
结束

图4

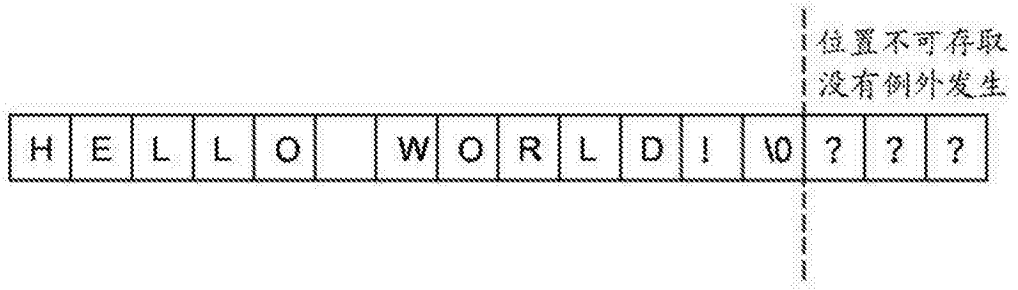

位置不可存取
没有例外发生
H
E
L
L
O
W
O
R
L
D
!
\0
?
?
?

图5

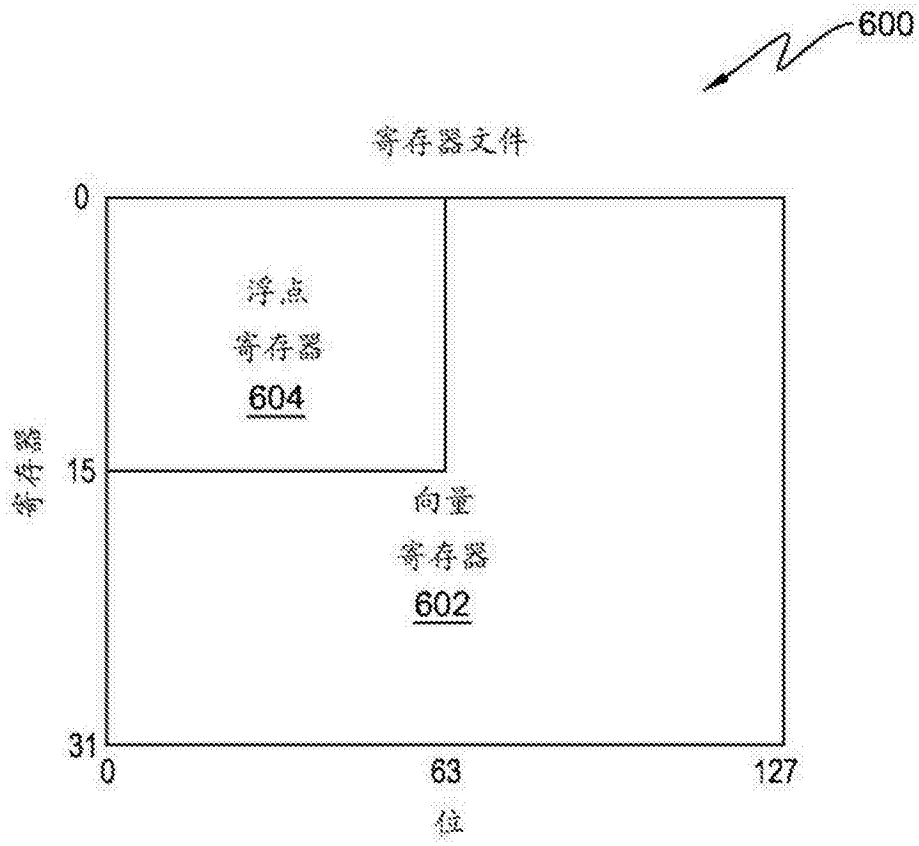

600
寄存器文件
0
浮点
寄存器
604
15
向量
寄存器
602
31
0
63
127
位
寄存器

图6

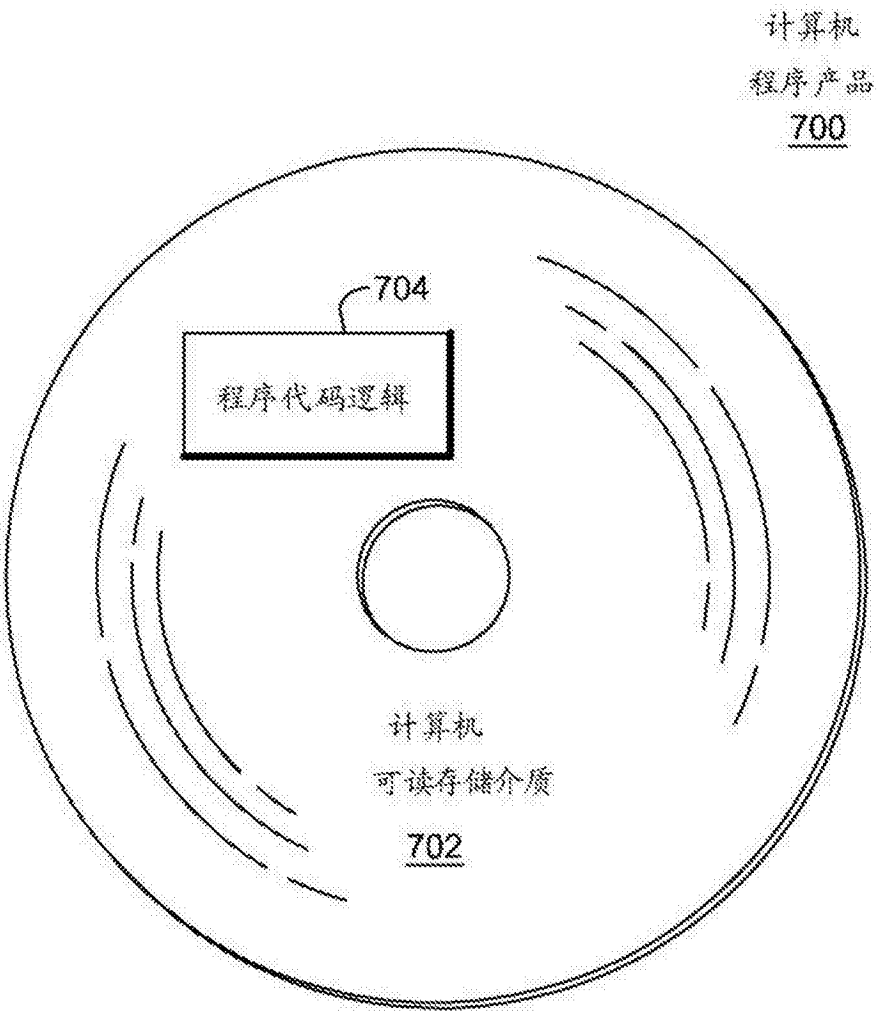


图7

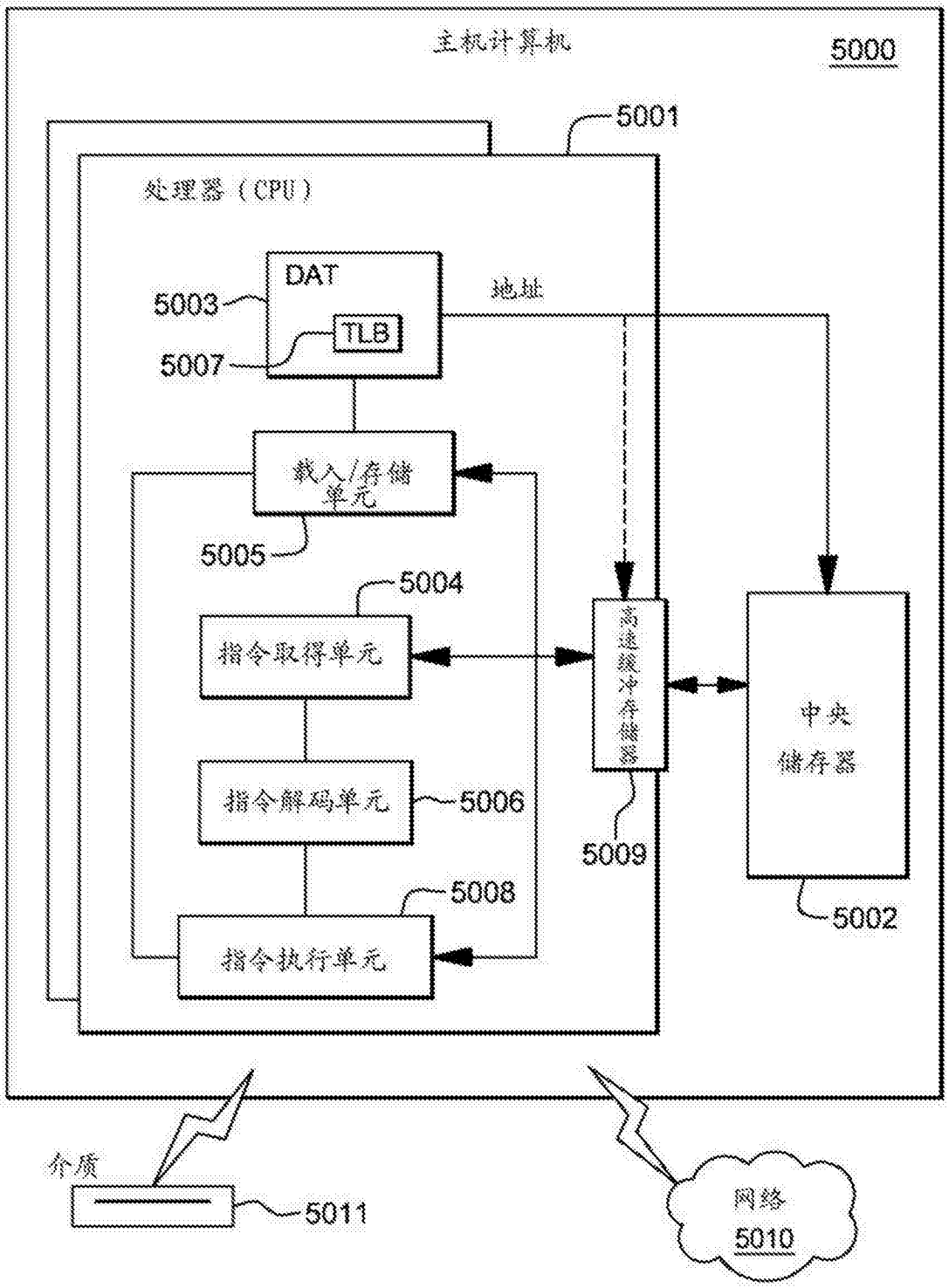

主机计算机
5000
5001
处理器（CPU）
DAT
5003
TLB
5007
地址
载入/存储单元
5005
5004
指令取得单元
指令解码单元
5006
5008
指令执行单元
高速缓冲存储器
5009
中央储存器
5002
介质
5011
网络
5010

图8

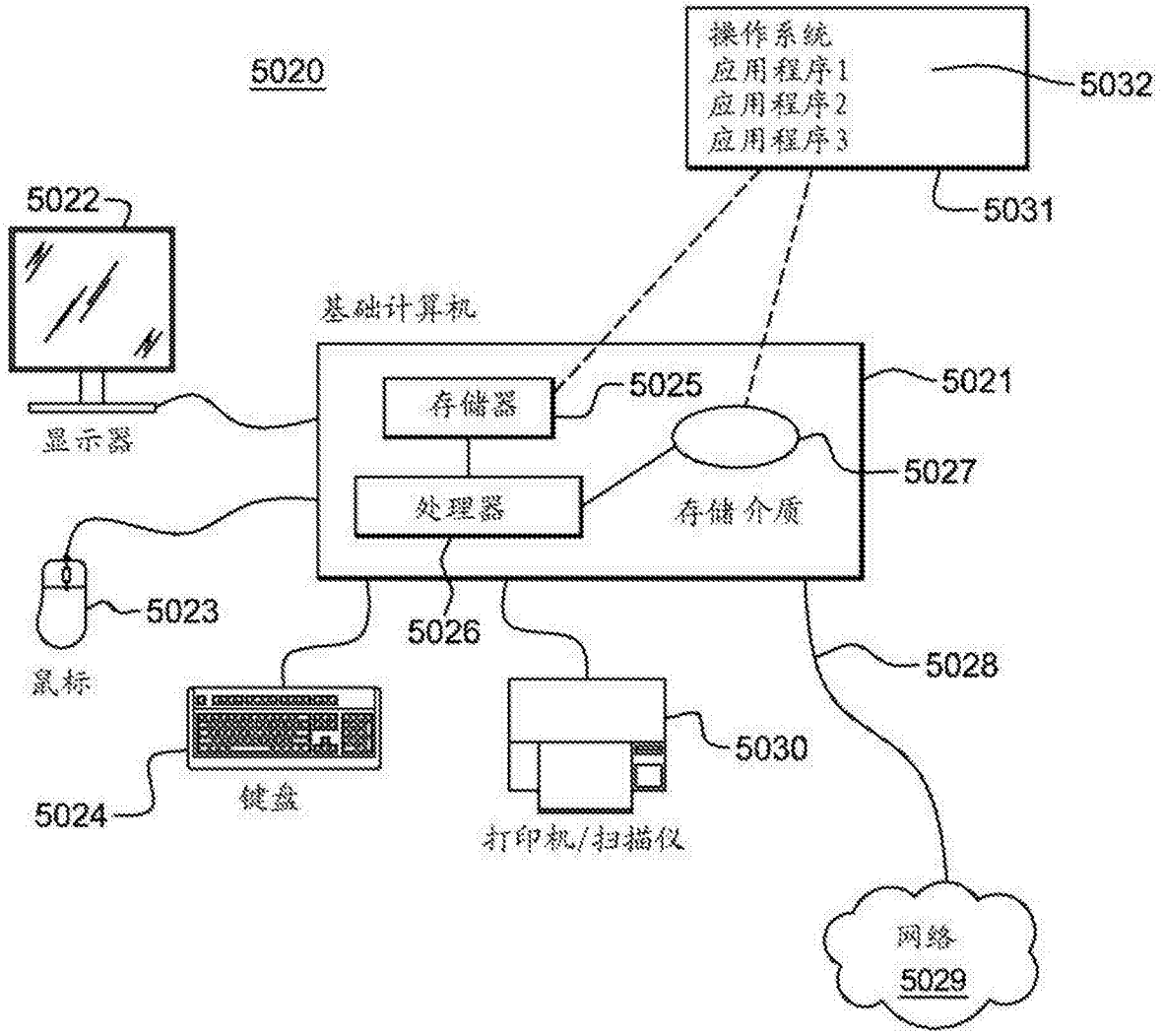

5020
操作系统
应用程序1
应用程序2
应用程序3
5032
5031
5022
基础计算机
存储器
5025
5021
显示器
处理器
存储介质
5027
5023
5026
5028
鼠标
5030
5024
键盘
打印机/扫描仪
网络
5029

图9

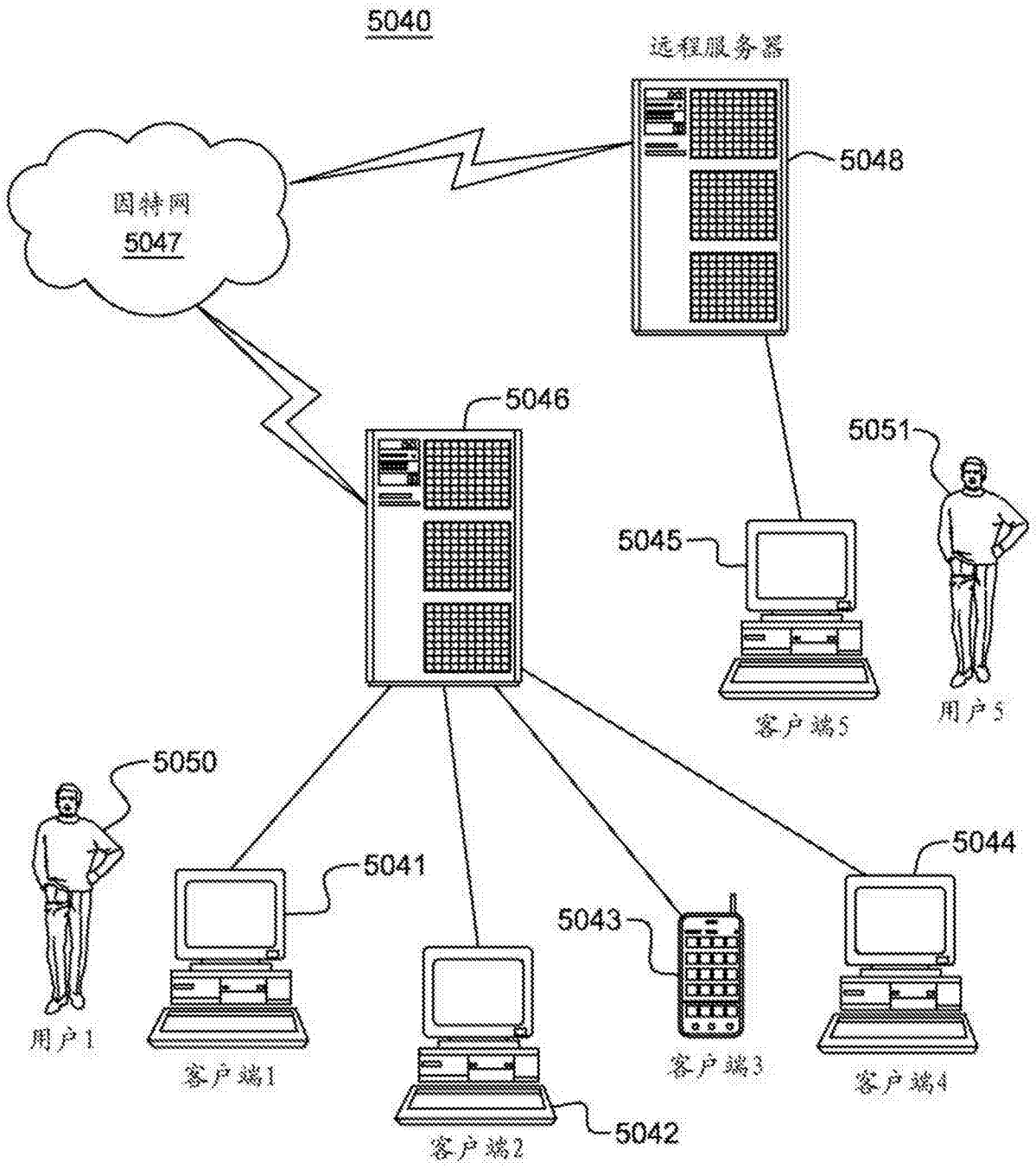

5040
远程服务器
5048
因特网
5047
5046
5051
5045
客户端5
用户5
5050
5041
5044
5043
用户1
客户端1
客户端3
客户端4
客户端2
5042

图10

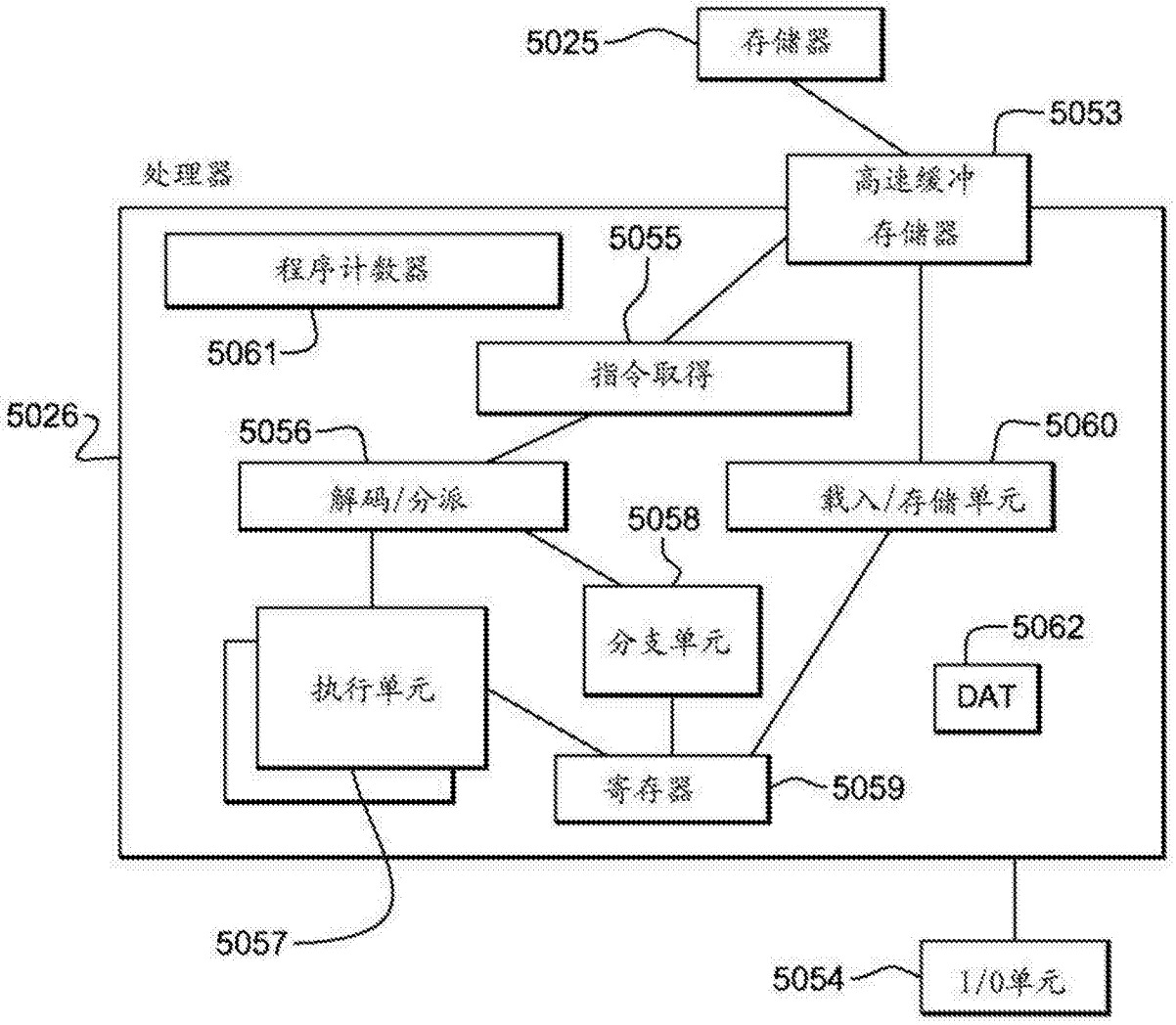

5025
存储器
5053
处理器
高速缓冲
存储器
5055
程序计数器
5061
指令取得
5026
5056
5060
解码/分派
载入/存储单元
5058
分支单元
5062
DAT
执行单元
寄存器
5059
5057
5054
I/O单元

图11

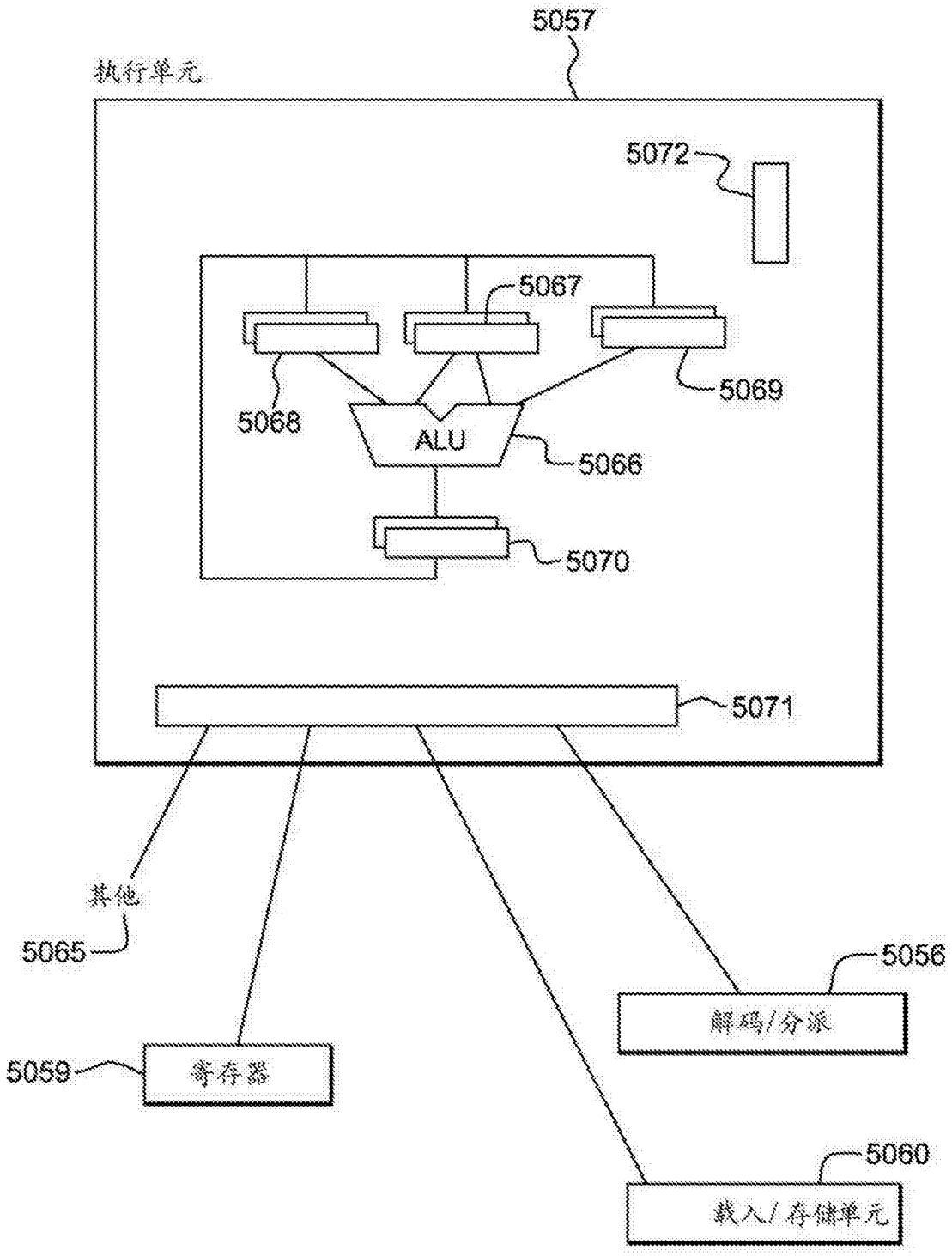

5057
执行单元
5072
5067
5068
ALU
5066
5069
5070
5071
其他
5065
5056
解码/分派
5059
寄存器
5060
载入/存储单元

图12A

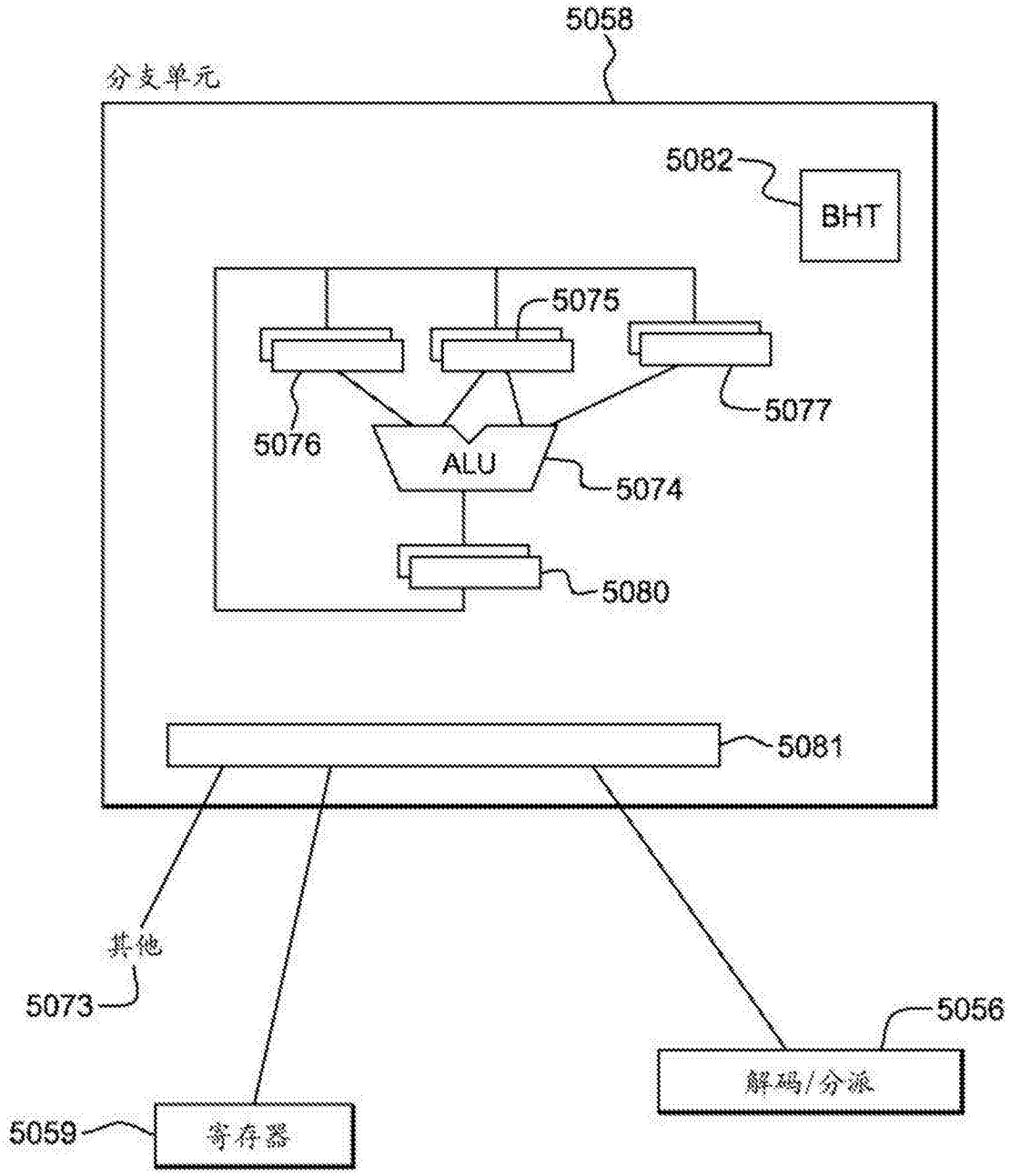

5058
分支单元
5082
BHT
5075
5076
5077
ALU
5074
5080
5081
其他
5073
5056
解码/分派
5059
寄存器

图12B

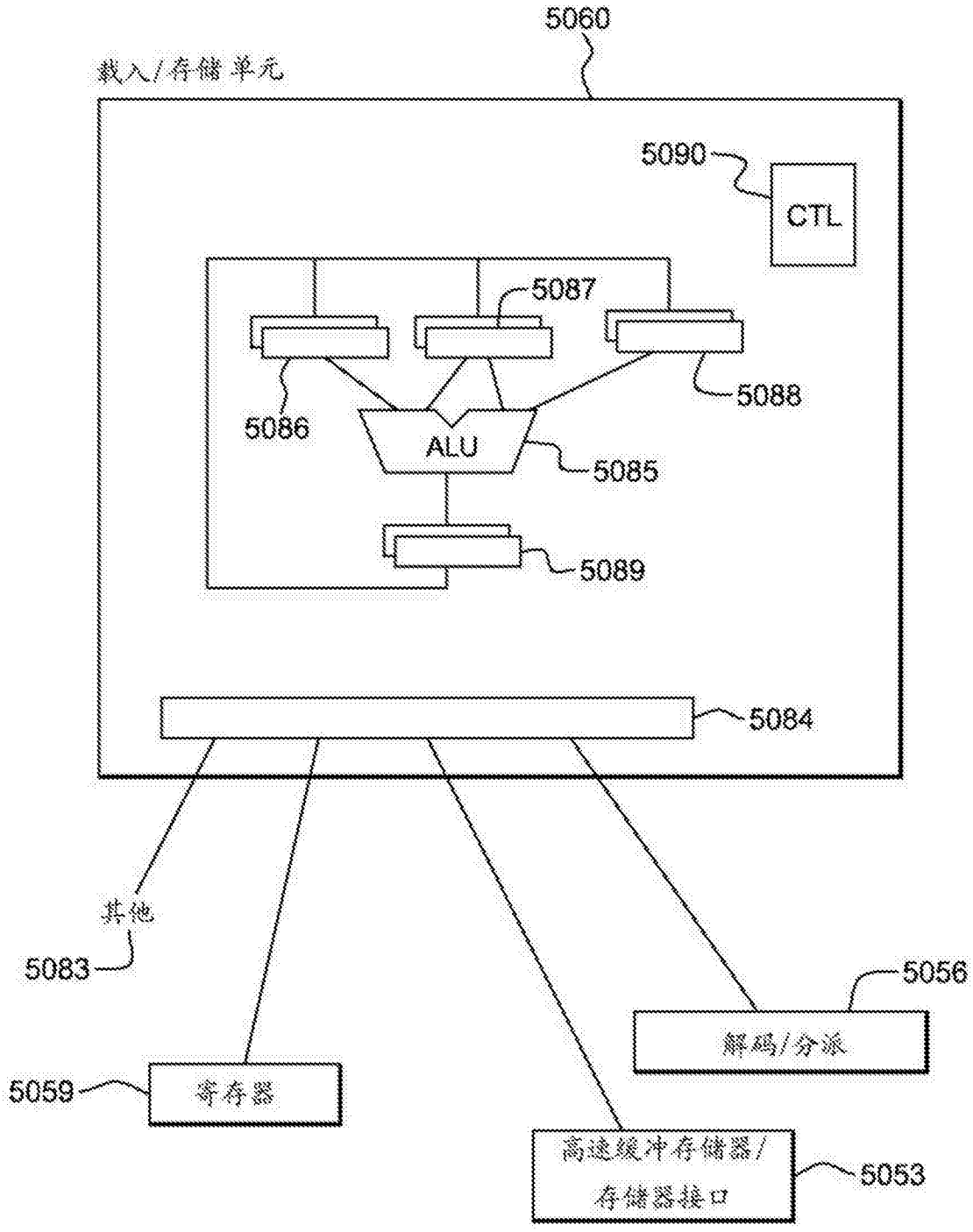

5060
载入/存储单元
5090
CTL
5087
5086
ALU
5088
5085
5089
5084
其他
5083
5056
解码/分派
5059
寄存器
高速缓冲存储器/
存储器接口
5053

图12C

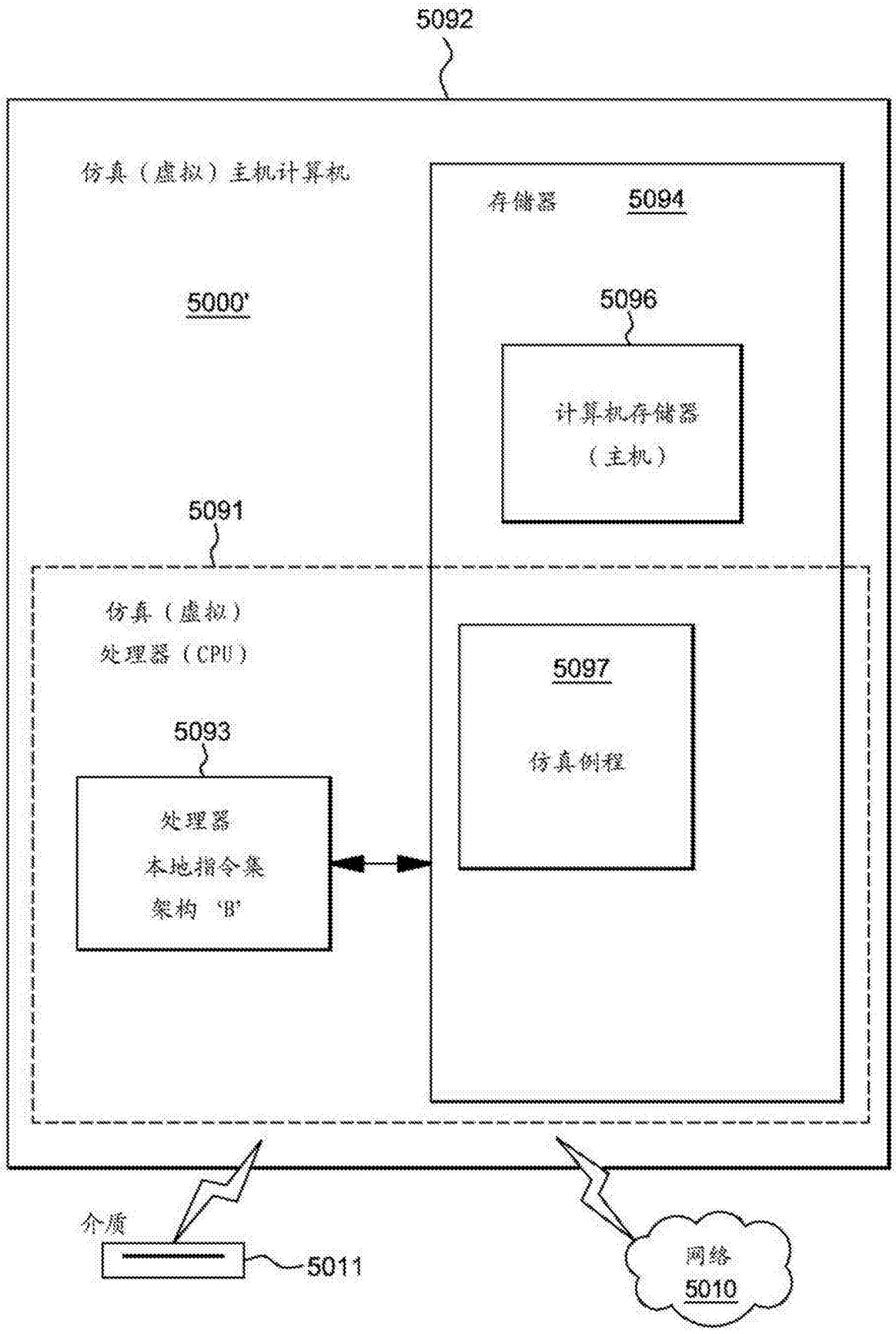

5092
仿真（虚拟）主机计算机
存储器
5094
5000'
5096
计算机存储器
（主机）
5091
仿真（虚拟）
处理器（CPU）
5097
仿真例程
5093
处理器
本地指令集
架构 'B'
介质
5011
网络
5010

图13