(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 104169907 A
(43)申请公布日 2014.11.26

(21)申请号 201380014427.0
(22)申请日 2013.03.07
(30)优先权数据
13/421,448 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/IB2013/051811 2013.03.07
(87)PCT国际申请的公布数据
WO2013/136233 EN 2013.09.19
(71)申请人 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 T. 斯雷格
E.M. 施瓦茨 M.K. 格施温德
(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 15/76(2006.01)

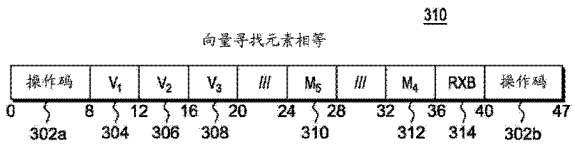
权利要求书2页 说明书35页 附图15页

(54)发明名称

向量寻找元素相等指令

(57)摘要

促进字符数据的处理。提供寻找元素相等指令，其比较多个向量的数据的相等性，且如果存在相等性，则提供相等性的指示。将与相等元素相关联的索引存储在目标向量寄存器中。另外，同一指令（该寻找元素相等指令）还搜索选定向量以寻找空值(null)元素，该等空值元素也被称作零元素。该指令的结果取决于是提供该空值搜索或仅该比较。



CN 104169907 A

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：

由处理器获得供执行的机器指令，该机器指令是根据计算机架构而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，该操作码识别向量寻找元素相等操作；

扩展字段，其用于指定一个或多个寄存器；

第一寄存器字段，其与该扩展字段的第一部分组合以指定第一寄存器，该第一寄存器包含第一操作元；

第二寄存器字段，其与该扩展字段的第二部分组合以指定第二寄存器，该第二寄存器包含第二操作元；

第三寄存器字段，其与该扩展字段的第三部分组合以指定第三寄存器，该第三寄存器包含第三操作元；

掩码字段，该掩码字段包含要在该机器指令的执行期间使用的一个或多个控制；以及

执行该机器指令，该执行包含：

确定该掩码字段是否包括零元素控制设定以指示对零元素的搜索；

基于该掩码字段包括该零元素控制设定以指示对零元素的该搜索，搜索该第二操作元以寻找零元素，该搜索提供空值索引，该空值索引包括在该搜索中寻找到的零元素的索引或未寻找到零元素的指示中的一个；

比较该第二操作元的一个或多个元素与该第三操作元的一个或多个元素的相等性，该比较提供比较索引，该比较索引包括基于该比较寻找到相等元素的相等元素的索引或基于该比较未寻找到相等元素的无相等性的指示中的一个；

提供结果，该结果是基于是否执行对零元素的该搜索，其中该结果包括以下各者中的一个：

基于未执行对零元素的该搜索，该结果包括该比较索引；或

基于执行对零元素的该搜索，该结果包括该比较索引或该空值索引中的一个。

2. 如权利要求 1 的方法，其中该结果是针对元素，该元素为零元素或相等元素，且该方法进一步包含：

调整该结果，该调整包含对该结果执行至少一个操作以提供经调整的结果，该经调整的结果包含该元素的第一字节的字节索引；及

将该经调整的结果存储在该第一操作元中。

3. 如权利要求 2 的方法，其中该机器指令进一步包含另一掩码字段，该另一掩码字段包括元素大小控制，该元素大小控制指定该第一操作元、该第二操作元或该第三操作元中的至少一个中的元素的大小，且其中该大小用于该调整中。

4. 如权利要求 1 的方法，其中该结果包括指示该第二操作元的大小的值，且该方法进一步包含将该结果存储在该第一操作元中。

5. 如权利要求 1 的方法，其中该掩码字段包含条件码设定控制，且其中该方法包含：

确定该条件码设定控制是否经设定；及

基于该条件码设定控制经设定，设定条件码以用于执行该机器指令。

6. 如权利要求 5 的方法，其中该设定该条件码包含以下各者中的一个：

将该条件码设定为指示在比任何相等比较低的索引元素中的零元素的检测的值；

将该条件码设定为指示匹配元素的值

;及

将该条件码设定为指示无匹配的值，且基于该零元素控制经设定，未寻找到零元素。

7. 如权利要求 1 的方法，其中该执行包含 :确定用于该比较的方向，其中该方向为自左至右或自右至左中的一个。

8. 如权利要求 7 的方法，其中该确定在运行时间执行，并且该确定包含通过该机器指令存取方向控制以确定该方向。

9. 如权利要求 1 的方法，其中该第二操作元及该第三操作元包含 N 个字节，且其中该比较包含并行地比较该第二操作元的该 N 个字节与该第三操作元的该 N 个字节，且其中元素的大小包含一个字节、两个字节或四个字节中的一个。

10. 如权利要求 1 的方法，其中该零元素的该索引包含字节索引，该字节索引为该零元素的第一字节的索引。

11. 如权利要求 1 的方法，其中该不相等元素的该索引包含字节索引，该字节索引为该不相等元素的字节的索引。

12. 如权利要求 1 的方法，其中包含在该空值索引或该比较索引中的至少一个中的未寻找到零元素的该指示或无相等性的该指示中的至少一个包括表示该第二操作元的大小的数字。

13. 一种包括适于执行根据任一前述方法权利要求的方法的所有步骤的部件的系统。

14. 一种计算机程序，包含当所述计算机程序在计算机系统上执行时用于执行根据任一前述方法权利要求的方法的所有步骤的指令。

向量寻找元素相等指令

技术领域

[0001] 本发明的一个方面大体是关于文本处理，且更具体地，是关于促进与字符数据相关联的处理。

背景技术

[0002] 文本处理常常需要比较字符数据，包括（但不限于）比较字符数据串。通常，用于比较字符数据的指令一次比较数据的单个字节。

[0003] 另外，文本处理常常需要其他类型的字符数据处理，包括寻找终止点（例如，串的结尾）、确定字符数据的长度、寻找特定字符等。执行这些类型的处理的当前指令趋于效率低下。

发明内容

[0004] 通过提供一种用于执行机器指令的计算机程序产品来克服先前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一种方法的指令。举例而言，该方法包括：由处理器获得供执行的机器指令，该机器指令是根据计算机架构而定义以用于计算机执行，该机器指令包括（例如）：至少一个操作码字段，其提供操作码，该操作码识别向量寻找元素相等操作；扩展字段，其用于指定一个或多个寄存器；第一寄存器字段，其与该扩展字段的第一部分组合以指定第一寄存器，该第一寄存器包括第一操作元；第二寄存器字段，其与该扩展字段的第二部分组合以指定第二寄存器，该第二寄存器包括第二操作元；第三寄存器字段，其与该扩展字段的第三部分组合以指定第三寄存器，该第三寄存器包括第三操作元；掩码字段，该掩码字段包括要在该机器指令的执行期间使用的一个或多个控制；及执行该机器指令，该执行包括：确定该掩码字段是否包括零元素控制设定以指示对零元素的搜索；基于该掩码字段包括该零元素控制设定以指示对零元素的搜索，搜索该第二操作元以寻找零元素，该搜索提供空值索引，该空值索引包括在该搜索中寻找到的零元素的索引或未寻找到零元素的指示中的一个；比较该第二操作元的一个或多个元素与该第三操作元的一个或多个元素的相等性，该比较提供比较索引，该比较索引包括基于该比较寻找到相等元素的相等元素的索引或基于该比较未寻找到相等元素的无相等性的指示中的一个；提供结果，该结果是基于是否执行对零元素的该搜索，其中该结果包括以下各者中的一个：基于未执行对零元素的该搜索，该结果包括该比较索引；或基于执行对零元素的该搜索，该结果包括该比较索引或该空值索引中的一个。

[0005] 本文中还描述及主张关于本发明的一个或多个方面的方法及系统。另外，本文中还描述且可主张关于本发明的一个或多个方面的服务。

[0006] 通过本发明的技术实现额外特征及优势。本发明的其他实施例及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] 特别指出本发明的一个或多个方面且在本说明书完结时在权利要求中作为示例清楚地主张本发明的一个或多个方面。自以下结合附图进行的详细描述，本发明的前述及其他目标、特征及优势是显而易见的，其中：

[0008] 图 1 描绘并入且使用本发明的一个或多个方面的计算环境的一个示例；

[0009] 图 2A 描绘并入且使用本发明的一个或多个方面的计算环境的另一示例；

[0010] 图 2B 描绘根据本发明的一个方面的图 2A 的存储器的其他细节；

[0011] 图 3 描绘根据本发明的一个方面的向量寻找元素相等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寻找元素相等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执行图 4 的逻辑的各种处理块的一个实施例；

[0014] 图 6 描绘根据本发明的一个方面的寄存器文件的一个示例；

[0015] 图 7 描绘根据本发明的一个方面的向量寻找任何相等指令的格式的一个实施例；

[0016] 图 8 描绘根据本发明的一个方面的与向量寻找任何相等指令相关联的逻辑的一个实施例；

[0017] 图 9 描绘并入本发明的一个或多个方面的计算机程序产品的一个实施例；

[0018] 图 10 描绘并入且使用本发明的一个或多个方面的主机计算机系统的一个实施例；

[0019] 图 11 描绘并入且使用本发明的一个或多个方面的计算机系统的又一示例；

[0020] 图 12 描绘并入且使用本发明的一个或多个方面的包含计算机网络的计算机系统的另一示例；

[0021] 图 13 描绘并入且使用本发明的一个或多个方面的计算机系统的各种元件的一个实施例；

[0022] 图 14A 描绘并入且使用本发明的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；

[0023] 图 14B 描绘并入且使用本发明的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；

[0024] 图 14C 描绘并入且使用本发明的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

[0025] 图 15 描绘并入且使用本发明的一个或多个方面的模拟主机计算机系统的一个实施例。

具体实施方式

[0026] 根据本发明的一个方面，提供用于促进字符数据的处理的能力，字符数据包括（但不限于）：任一语言下的字母字符；数字；标点符号；和 / 或其他符号。字符数据可为或可不为数据串。标准与字符数据相关联，标准的示例包括（但不限于）：美国信息交换标准码 (ASCII)；Unicode，包括（但不限于）Unicode 变换格式 (UTF)8；UTF16；等。

[0027] 在一个示例中，提供寻找元素相等指令，其比较多个向量的数据的相等性且提供相等性的指示（如果存在相等性）。在一个示例中，与相等元素相关联的索引存储在目标向

量寄存器中。

[0028] 如本文中所描述，向量寄存器（也被称作向量）的元素的长度为一个、两个或四个字节，作为示例；且向量操作元为（例如）具有多个元素的单指令多数据 (SIMD) 操作元。在其他实施例中，元素可具有其他大小；且向量操作元不需要为 SIMD，和 / 或可包括元素。

[0029] 在又一个实施例中，同一指令（寻找元素不相等指令）还搜索选定向量以寻找空值元素，空值元素也被称作零元素（例如，全部元素为零）。空值或零元素指示字符数据的终止；例如，特定数据串的结尾。指令的结果取决于是提供空值搜索或仅比较。

[0030] 在又一个实施例中，提供向量寻找任何相等指令，其搜索向量以寻找特定字符和 / 或零元素，并且返回匹配的字符或元素的掩码或字节索引。

[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）通过（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）及一个或多个输入 / 输出 (I/O) 设备和 / 或接口 106。

[0032] 在一个示例中，处理器 102 是基于由国际商用机器公司供应的 z/Architecture，且为服务器的部分，诸如也由国际商用机器公司供应且实施 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® 公开案（IBM® 公开案第 SA22-7832-08 号，第九版，2010 年 8 月）中，该公开案在此以其全文引用的方式并入本文中。在一个示例中，该处理器执行操作系统，诸如，也由国际商用机器公司供应的 z/OS。IBM®、Z/ARCHITECTURE® 及 Z/OS® 为国际商用机器公司 (Armonk, New York, USA) 的注册商标。本文中使用的其他名称可为国际商用机器公司或其他公司的注册商标、商标或产品名称。

[0033] 在又一个实施例中，处理器 102 是基于由国际商用机器公司供应的 Power Architecture。Power Architecture 的一个实施例描述于”Power ISATM 第 2.06 版修订 B”（国际商用机器公司，2010 年 7 月 23 日）中，该文件在此以其全文引用的方式并入本文中。POWER ARCHITECTURE® 为国际商用机器公司的注册商标。

[0034] 在又一个实施例中，处理器 102 是基于由 Intel Corporation 供应的 Intel 架构。Intel 架构的一个实施例描述于“Intel® 64and IA-32Architectures Developer’s Manual：第 2B 卷，Instructions Set Reference, A-L”（序号 253666-041US，2011 年 12 月）及“Intel® 64and IA-32Architectures Developer’s Manual：第 2B 卷，Instructions Set Reference, M-Z”（序号 253667-041US，2011 年 12 月）中，该等文件中的每一个在此以其全文引用的方式并入本文中。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方面的计算环境的另一个实施例。在此示例中，计算环境 200 包括（例如）通过（例如）一个或多个总线 208 和 / 或其他连接耦接至彼此的原生中央处理单元 202、存储器 204 及一个或多个输入 / 输出设备和 / 或接口 206。作为示例，计算环境 200 可包括：由国际商用机器公司 (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 / 或基于由国际商用机器公司、Hewlett Packard、Intel、Oracle 或其他者供应的架构的其他机器。

[0036] 原生中央处理单元 202 包括在该环境内的处理期间使用的一个或多个原生寄存器 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一特定时间点的环境状态的信息。

[0037] 此外,原生中央处理单元 202 执行存储在存储器 204 中的指令及代码。在一特定示例中,中央处理单元执行存储在存储器 204 中的模拟器代码 212。此代码使得以架构配置的处理环境能够模拟另一架构。举例而言,模拟器代码 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 服务器或其他者) 模拟 z/Architecture 及执行基于 z/Architecture 开发的软件及指令。

[0038] 参看图 2B 描述关于模拟器代码 212 的其他细节。客体指令 250 包含经开发以欲于不同于原生 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,客体指令 250 可能已经设计以在 z/Architecture 处理器 102 上执行,但实情为,正在原生 CPU 202(其可为 (例如)Intel Itanium II 处理器) 上模拟客体指令 250。在一个示例中,模拟器代码 212 包括指令提取单元 252 以自存储器 204 获得一个或多个客体指令 250,及视情况提供用于所获得的指令的本地缓冲。模拟器代码 212 还包括指令转译例程 254 以确定已获得的客体指令的类型且将客体指令转译成一个或多个对应原生指令 256。此转译包括 (例如) 识别待由客体指令执行的函数及选择原生指令以执行该函数。

[0039] 另外,模拟器 212 包括模拟控制例程 260 以使得执行原生指令。模拟控制例程 260 可使得原生 CPU 202 执行模拟一个或多个先前所获得的客体指令的原生指令的例程且,在此执行完结时,将控制返回至指令提取例程以模拟下一个客体指令或一群客体指令的获得。原生指令 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 ;或执行某一类型的算术或逻辑运算 (如由转译例程确定)。

[0040] 每一例程 (例如) 以软件来实施,该软件存储在存储器中且由原生中央处理单元 202 来执行。在其他示例中,例程或运算中的一个或多个以固件、硬件、软件或其某一组合来实施。可使用原生 CPU 的寄存器 210 或通过使用存储器 204 中的位置模拟经模拟的处理器的寄存器。在实施例中,客体指令 250、原生指令 256 及模拟器代码 212 可常驻于相同存储器中或可在不同存储器设备间分配。

[0041] 如本文中所使用,固件包括 (例如) 处理器的微码、毫码和 / 或巨集码。固件包括 (例如) 在较高阶机器码的实施中使用的硬件层级指令和 / 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] 在一个示例中,所获得的、经转译及经执行的客体指令 250 为本文中所描述的指令之一。自存储器提取具有架构 (例如,z/Architecture) 的指令,将该指令转译及表示为具有另一架构 (例如,PowerPC、pSeries、xSeries、Intel 等) 的一系列原生指令 256。接着执行这些原生指令。

[0043] 在一个实施例中,本文中所描述的指令为根据本发明的一个方面提供的向量指令,其为向量设施的部分。向量设施提供 (例如) 在从一至十六个元素的范围内的固定大小的向量。每一向量包括由设施中所定义的向量指令进行运算的数据。在一个实施例中,

如果向量由多个元素组成，则将每一元素与其他元素并行地处理。直至所有元素的处理完成后方出现指令完成。

[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 等的各种架构的部分。尽管本文中所描述的实施例是针对 z/Architecture，但向量指令及本发明的一个或多个方面可基于许多其他架构。z/Architecture 仅为一个示例。

[0045] 在将向量设施实施为 z/Architecture 的部分的一个实施例中，为了使用向量寄存器及指令，将指定控制寄存器（例如，控制寄存器 0）中的向量启用控制及寄存器控制设定为（例如）一。如果安装了向量设施且在未设定启用控制的情况下执行向量指令，则识别到数据例外状况。如果未安装向量设施且执行向量指令，则识别到操作例外状况。

[0046] 向量数据出现于存储器中，例如，以与其他数据格式相同的自左至右序列。编号为 0 至 7 的数据格式的位构成存储器中的最左侧（最低编号）字节位置中的字节，位 8 至 15 形成下一个顺序位置中的字节，等等。在又一示例中，向量数据可以另一系列（诸如，自右至左）出现于存储器中。

[0047] 通过向量设施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 的该字段包括用于向量寄存器指定的操作元中的每一个的最高有效位。用于未由指令指定的寄存器指定的位将颠倒，且经设定为零。

[0048] 在一个示例中，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定的最高有效位。

[0050] 1- 用于指令的第二向量寄存器指定（如果有的话）的最高有效位。

[0051] 2- 用于指令的第三向量寄存器指定（如果有的话）的最高有效位。

[0052] 3- 用于指令的第四向量寄存器指定（如果有的话）的最高有效位。

[0053] 由（例如）汇编器取决于寄存器编号将每一位设定为零或一。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，每一 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] 在又一个实施例中，RXB 字段包括额外位，且将一个以上位用作用于每一向量或位置的扩展。

[0056] 根据本发明的一个方面提供的包括 RXB 字段的指令为向量寻找元素不相等指令，其一个示例描绘于图 3 中。在一个示例中，向量寻找元素不相等指令 300 包括：操作码字段 302a（例如，位 0 至 7）、302b（例如，位 40 至 47），其指示向量寻找元素不相等操作；第一向量寄存器字段 304（例如，位 8 至 11），其用于指定第一向量寄存器（V_1）；第二向量寄存器字段 306（例如，位 12 至 15），其用于指定第二向量寄存器（V_2）；第三向量寄存器字段 308（例如，位 16 至 19），其用于指定第三向量寄存器（V_3）；第一掩码字段（M_5）310（例如，位 24 至 27）；第二掩码字段（M_4）312（例如，位 32 至 35）；及一 RXB 字段 314（例如，位 36 至 39）。在一个示例中，字段 304 至 314 中的每一个分开且独立于操作码字段。另外，在一个实施例中，该等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] 在一个示例中，由操作码字段 302a 指定的操作码的选定位（例如，前两个位）指

定指令的长度及格式。在此特定示例中，选定位指示长度为三个半字组，且格式为通过扩展的操作码字段进行的向量寄存器及寄存器运算。向量 (V) 字段中的每一个以及由 RXB 指定的其对应寄存器扩展位指定向量寄存器。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）指定含有操作元的寄存器。举例而言，如果四位字段为 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元。举例而言，与向量寄存器 V_1 相关联的下标编号 1 表示第一操作元，等等。寄存器操作元的长度为一个寄存器，其为（例如）128 个位。

[0059] 具有（例如）四个位 0 至 3 的 M_4 字段在（例如）位 1 至 3 中指定元素大小控制。元素大小控制指定向量寄存器操作元中的元素的大小。作为示例，元素大小控制指定字节、半字组（例如，2 个字节）或字组（例如，4 个字节）的任一。举例而言，0 指示字节 ；1 指示半字组 ；且 2 指示字组，还称为，全字组。如果指定颠倒值，则识别到规范例外状况。

[0060] M_5 字段为（例如）四位字段 - 位 0 至 3，包括（例如）：

[0061] 零搜索字段 (ZS，位 2)，其如果为一，则还将第二操作元的每一元素与零（或空值）比较。（在又一示例中，将第三操作元或另一操作元中的每一元素与零比较）；及

[0062] 条件码设定字段 (CC，位 3)，其如果为零，则不设定条件码且条件码保持不变。如果为一，则如下文所指定来设定条件码（作为一个示例）：

[0063] 0- 如果设定零搜索位，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元中的零元素 ；

[0064] 1- 比较在一些元素中检测到第二和第三操作元之间的匹配。如果设定零搜索位，在具有比零比较元素低或等于零比较元素的索引的元素中出现该匹配。

[0065] 2---

[0066] 3- 无元素比较起来相等。

[0067] 在向量寻找元素相等指令的一个实施例的执行中，在一个实施例中自左至右继续进行，将第二操作元（包括于由 V_2 加上其 RXB 位指定的寄存器中）的不带正负号的二进位整数元素与第三操作元（包括于由 V_3 加上其 RXB 位指定的向量寄存器中）的对应的不带正负号的二进位整数元素比较。如果两个元素相等，则将最左侧相等元素的第一字节的字节索引置放于第一操作元（包括在由 V_1 加上其 RXB 位指定的寄存器中）的选定字节（例如，字节 7）中。将零存储至第一操作元的剩余字节。

[0068] 举例而言，如果元素大小为一个字节，则传回最左侧相等元素的字节索引（例如，如果存在 16 个元素 0 至 15，且元素 6 相等，则传回字节索引 6）。类似地，如果元素大小为半字组，且存在 8 个元素 0 至 7，且元素三的字节 6 或 7 两者相等，则传回字节索引 6。同样地，如果元素大小为全字组且存在四个元素 0 至 3，且元素的字节 4 至 7 中的所有相等，则传回字节索引 4。

[0069] 如果没有发现字节相等，或者设定零搜索，则将等于向量大小（例如，以字节的数目计，例如，16）的索引存储于第一操作元的指定字节（例如，字节 7）中。将零存储在剩余字节中。

[0070] 如果零搜索位设定于 M_5 字段中，则还比较第二操作元（或在另一实施例中，诸如第三操作元的另一操作元）中的每一元素与零（或者空值、串结尾）的相等性。如果在发

现第二和第三操作元的任一其他元素相等之前在第二操作元中寻找到零元素，则将发现为零的元素的第一字节的字节索引存储在第一操作元的指定字节（例如，字节 7）中，并且将零存储在所有其他字节位置中。如果条件码设定标记为一，则条件码设定为零。

[0071] 在一个实施例中，并行地执行元素的比较。举例而言，如果正进行比较的向量寄存器的长度为 16 个字节，则并行地比较 16 个字节。另外，在一个实施例中，在运行时间提供向量的方向 - 自左至右或自右至左。举例而言，作为示例，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部分，但在运行时间将其提供至指令。

[0072] 在又一个实施例中，指令不包括 RXB 字段。实情为，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），或作为指令的另一字段的部分提供扩展。

[0073] 参看图 4 描述关于处理向量寻找元素不相等指令的一个实施例的其他细节。在一个示例中，计算环境的处理器正执行此逻辑。

[0074] 最初，作出关于是否将执行对空值（还称为，零元素、串结尾、终止符等）的搜索的确定（询问 400）。如果将执行对空值的搜索，则进行与空值字符（也就是说，针对零元素）的比较（步骤 402），且将结果输出至 nullidx（403）。举例而言，如果元素大小为字节且在字节 5 中寻找到零元素，则将寻找到零元素所在的字节的索引（例如，5）置放于 nullidx 中。类似地，如果元素大小为半字组，且存在 8 个元素 0 至 7，且元素三（也就是说，字节 6 至 7）为零，则将 6（针对字节索引 6）置放于 nullidx 中。同样地，如果元素大小为全字组且存在四个元素 0 至 3，且元素一（也就是说，字节 4 至 7）为零，则将 4（针对字节索引 4）置放于 nullidx 中。如果未寻找到空值元素，则，在一个示例中，将向量的大小（例如，以字节计；例如，16）置放于 nullidx 中。

[0075] 另外，或如果将不执行空值搜索，则并行地执行基于比较操作而比较 A 与 B 的多个比较（例如，16）（步骤 404）。在一个示例中，A 为第二操作元的内容，且 B 为第三操作元的内容，且比较操作是不相等。

[0076] 将比较的结果存储在变量 406（取决于搜索是自左或自右而被称作左索引（cmpidxl），或右索引（cmpidxr））中。举例而言，如果比较为相等比较，则搜索为自左至右，且比较导致一个或多个相等，将与最低相等元素的第一字节相关联的索引置放于 cmpidxl 中。作为一个示例，如果元素大小为字节且向量中存在 16 个元素（0 至 15），且在元素 6 中发现等性，则将 6 存储在 cmpidxl 中。类似地，如果元素大小为半字组，且向量中存在 8 个元素（0 至 7），且在元素 3 中（例如，在字节 6 或 7 处）发现等性，则传回元素的第一字节的索引（字节 6）。同样地，如果元素大小为全字组且存在四个元素（0 至 3），且在元素 1 中（例如，在字节 4 至 7 处）发现等性，则传回元素的第一字节的索引（字节 4）。如果不存在相等比较，则，在一个实施例中，取决于比较的方向，将 cmpidxl 或 cmpidxr 设定为等于向量的大小（例如，以字节计；例如，16）。

[0077] 此后，作出关于搜索是自左或自右的确定（询问 408）。如果搜索是自左，则将变量 cmpidx 设定为等于 cmpidxl（步骤 410）；否则，将 cmpidx 设定为等于 cmpidxr（步骤 412）。

[0078] 在设定 cmpidx 之后，作出关于是否执行对空值字符的搜索的确定（询问 414）。如果不存在对空值字符的搜索，则将变量 idx 设定为（例如）比较索引 cmpidx（步骤 416）。如果搜索到空值，则将 idx 设定为比较索引或空值索引（nullidx）中的最小者（步骤 418）。

此情形结束处理。

[0079] 用于图 4 的处理的块逻辑的一个示例描绘于图 5 中。在此示例中，存在两个输入：向量 B 500 及向量 A 502。将两个输入输入至并行地执行比较（例如，相等）的比较逻辑 504。另外，还将一输入（向量 A）输入至执行空值处理的零检测逻辑 506。

[0080] 将比较逻辑的输出 idxL 或 idxR 508 以及零检测逻辑的输出 nullidx 510 输入至结果确定逻辑 512。结果确定逻辑还将以下控制作为输入：右 / 左 514，其指示搜索的方向；零检测 516，其指示是否将执行空值处理；及元素大小 518，其提供每一元素的大小（例如，字节、半字组、字组）；且产生所得索引 520-resultidx，将其存储在输出向量 522 中（例如，在字节 7 中）。

[0081] 另外，结果确定逻辑包括条件码处理 523，其视情况输出条件码 524。

[0082] 用于比较逻辑 504 的示例伪码如下：

[0083] idxL = 16 ;idxR = 16

[0084] For i = 0to vector_length

[0085] If A[i] = to B[i]THEN

[0086] idxL = i

[0087] Done

[0088] For i = vector_length downto 0

[0089] If A[i] ! = to B[i]THEN

[0090] idxR = i

[0091] done

[0092] 如所展示，取决于方向，将变量 idxL 或 idxR 初始化至向量的大小（例如，以字节的数目计；例如，16）。接着，将向量 A 的每一元素与向量 B 的对应元素比较。在一个示例中，比较为字节比较，因此针对 16 个字节 (i) 中的每一个进行比较。在此示例中，比较操作是相等，且如果发现相等性，则将不相等字节的索引存储在 idxL（如果自左搜索）或 idxR（如果自右搜索）中。

[0093] 用于零检测逻辑 506 的示例伪码如下：

[0094] nullidx = 16

[0095] FOR j = 0to vector_length

[0096] IF A[j] = 0THEN

[0097] nullidx = j x element_size

[0098] Done

[0099] 如所展示，测试向量的每一元素 (j) 以查看其是否等于零。如果元素等于零，则将 nullidx 设定为等于该元素的索引乘以元素大小所得值。举例而言，如果元素大小为半字组 (2 个字节)，且在元素 3 中检测到空值字符，则将 3 乘以 2，且将 nullidx 设定为 6，其表示字节 6。类似地，如果元素大小为全字组 (4 个字节)，且在元素 3 中检测到空值字符，则将 3 乘以 4，且将 nullidx 设定为 12。

[0100] 同样地，用于结果确定逻辑 512 的示例伪码如下：

[0101] IF Left/Right = Left THEN

[0102] cmpidx = idxL

[0103] ELSE

[0104] cmpidx = idxR

[0105] IF zero_detect = ON THEN

[0106] resultidx = min(cmpidx,nullidx)

[0107] IF set_CC = ON&&nullidx< = cmpidx THEN

[0108] CC = 0

[0109] ELSE

[0110] resultidx = cmpidx

[0111] IF element_size = byte THEN element_size_mask = '11111'b

[0112] IF element_size = 2byte THEN element_size_mask = '11110'b

[0113] IF element_size = 4byte THEN element_size_mask = '11100'b

[0114] resultidx = resultidx&element_size_mask

[0115] IF SetCC = ON THEN

[0116] IF nullidx<cmpidx THEN

[0117] CC = 0

[0118] ELSE IF cmpidx<16THEN

[0119] CC = 1

[0120] ELSE

[0121] CC = 3

[0122] 如所展示,如果左 / 右控制指示左,则将 cmpidx 设定为等于 idxL ;否则,将 cmpidx 设定为等于 idxR。另外,如果零检测指示符在作用中,则将 resultidx 设定为等于 cmpidx 或 nullidx 中的最小者 ;且如果条件码设定控制在作用中且 cmpidx 大于 nullidx,则将条件码设定为零。否则,如果零检测不在作用中,则将 resultidx 设定为等于 cmpidx。

[0123] 另外,如果元素大小等于字节,则将元素大小掩码设定为“11111”;如果元素大小等于 2 个字节,则将掩码设定为“11110”,且如果元素大小等于 4 个字节,则将掩码设定为“11100”。

[0124] 此后,将 resultidx 设定为等于 resultidx 与元素大小掩码进行和 (AND) 运算所得值。举例而言,如果元素大小为半字组且字节 7 为 resultidx,则 resultidx = 00111AND 11110,从而提供 00110 ;因此,将 resultidx 设定为等于 6(也就是说,二进位的 00110),其为元素的第一字节。

[0125] 另外,视情况设定条件码。如果将指令的设定条件码控制设定为在作用中,则提供条件码 ;否则,不设定条件码。作为示例,如果将控制设定为在作用中,则如果 nullidx<cmpidx,则将条件码设定为 0。否则,如果 cmpidx<16,则将条件码设定为 1 ;否则,将条件码设定为 3。

[0126] 上文所描述的示例为用于促进字符数据处理的向量指令的一个示例。如本文中所描述,对于 128 位向量,比较逻辑仅执行 16 个字节比较,而非 (例如)256 个比较。此情形提供对于较大向量的按比例调整。另外,可将左 / 右控制作为运行时间值来提供且不在指令内予以编码。又,作为结果传回的值为字节位置,而非元素索引。另外,支持 4 个字节比较,以及 1 字节及 2 个字节比较。

[0127] 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等向量寄存器的象限。举例而言，如图 6 中所展示，如果存在包括 32 个向量寄存器 602 的寄存器文件 600 且每一寄存器的长度为 128 个位，则长度为 64 个位的 16 个浮点寄存器 604 可覆迭该等向量寄存器。因此，作为一个示例，当修改浮点寄存器 2 时，接着还修改向量寄存器 2。用于其他类型的寄存器的其他映射也是可能的。

[0128] 在本发明的进一步方面中，根据本发明的一个方面提供有向量设施并且使用的另一指令向量寻找任何相等指令，其中输入向量中的所有字符（或者另一实施例中的子集）与另一输入向量中的每个字符（或者选定字符）比较。输出记录为对于匹配的字符的掩码或索引。这是有用的，例如，当解析诸如数据串的字符数据时。当解析时，执行的一个操作是在进到下一个解析步骤之前寻找若干特定字符。该指令使得能够一次搜索若干字符。

[0129] 图 7 中描绘向量寻找任何相等指令的示例，其中在一个实施例中，向量寻找任何相等指令 700 包括例如指示向量寻找任何相等操作的操作码字段 702a（例如，位 0-7）、702b（例如，位 40-47）；用于指定第一矢量寄存器 (V_1) 的第一矢量寄存器字段 704（例如，位 8-11）；用于指定第二矢量寄存器 (V_2) 的第二矢量寄存器字段 706（例如，位 12-15）；用于指定第三矢量寄存器 (V_3) 的第三矢量寄存器字段 708（例如，位 16-19）；第一掩码字段 (M_5) 710（例如，位 24-27）；第二掩码字段 (M_4) 712（例如，位 32-35）；和 RXB 字段 714（例如，位 36-39）。在一个示例中，字段 704-714 的每个是与操作码字段分离并且独立的。此外，在一个实施例中，它们是相互分离和独立的；然而，在其他实施例中，超过一个字段可以组合。下面描述关于这些字段的使用的进一步信息。

[0130] 与以上类似，在该示例中，由操作码字段 702a 执行的操作码的选定位（例如，头两位）指定指令的长度和格式。在该示例中，选定的位指示长度是三个半字组，且格式为通过扩展的操作码字段进行的向量寄存器及寄存器运算。向量 (V) 字段中的每一个以及由 RXB 指定的其对应扩展位指定向量寄存器。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）指定含有操作元的寄存器。举例而言，如果四位字段为 0110 且扩展位为 0，则五位字段 00110 指示寄存器编号 6。

[0131] 与指令的字段相关联的下标编号表示该字段适用的操作元。举例而言，与向量寄存器 V_1 相关联的下标编号 1 表示第一操作元，等等。寄存器操作元的长度为一个寄存器，其为（例如）128 个位。

[0132] 具有（例如）四个位 0 至 3 的 M_4 字段在（例如）位 1 至 3 中指定元素大小控制。元素大小控制指定向量寄存器操作元中的元素的大小。在一个示例中，元素大小控制可指定字节、半字组（例如，2 个字节）或字组（例如，4 个字节）。举例而言，0 指示字节；1 指示半字组；且 2 指示字组，还称为，全字组。如果指定颠倒值，则识别到规范例外状况。

[0133] M_5 字段为（例如）四位字段 - 位 0 至 3，包括（例如）：

[0134] 结果类型字段 (RT，位 1)，其如果为零，则每个得到元素是关于该元素比较的所有范围的掩码。如果为一，则字节索引存储在第一操作元的指定字节（例如，字节 7），并且零存在在所有其他元素中；

[0135] 零搜索字段 (ZS，位 2)，其如果为一，则还将第二操作元（或另一操作元）的每一元素与零比较；以及

[0136] 条件码设定字段 (CC，位 3)，其如果为零，则不设定条件码且条件码保持不变。如

果为一，则如下文所指定来设定条件码（作为一个示例）：

[0137] 0- 如果设定零搜索位，则在比第二操作元中的零较低的索引元素中不存在匹配。

[0138] 1- 第二操作元中的一些元素匹配第三操作元中的至少一个元素。

[0139] 2- 第二操作元中的所有元素匹配第三操作元中的至少一个元素。

[0140] 3- 第二操作元中没有元素匹配第三操作元中的任何元素。

[0141] 在向量寻找任何相等指令的一个实施例的执行中，在一个实施例中自左至右继续进行，将第二操作元（包括于由 V_2 加上 RXB 指定的向量寄存器中）的不带正负号的二进位整数元素与第三操作元（包括于由 V_3 加上 RXB 指定的向量寄存器中）的每个不带正负号的二进位整数元素比较，并且可选地，为零，如果在 M_5 字段中设定零搜索标记。

[0142] 如果 M_5 字段中的结果类型标记为零，那么对于匹配第三操作元中任何元素的第二操作元中的每个元素，或者可选的零，第一操作元（包括于由 V_1 加上 RXB 指定的向量寄存器中）中对应元素的位位置设定位一；否则，它们设定位零。

[0143] 如果 M_5 字段中的结果类型标记为一，那么匹配第三操作元中元素的第二操作元中的最左侧元素的字节索引（例如，元素的第一字节的字节索引）或者零存储在第一操作元的指定字节（例如，字节 7）。

[0144] 如果 M_5 字段中的结果类型标记为一，并且没有发现字节相等，或者如果设定零搜索标记为零，那么等于向量的大小（例如，以字节计；例如，16）的索引存储在第一操作元的指定字节（例如，字节 7）。

[0145] 在一个实施例中，在运行时间提供向量的方向 - 自左至右或自右至左。举例而言，作为示例，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部分，但在运行时间将其提供至指令。

[0146] 在又一个实施例中，指令不包括 RXB 字段。实情为，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），或作为指令的另一字段的部分提供扩展。

[0147] 参看图 8 描述关于处理向量寻找任何相等指令的一个实施例的其他细节。在一个示例中，计算环境的处理器正执行此逻辑。

[0148] 最初，初始化称为 zeroidx 802、resultidx 804 和 resultmask 806 的变量，步骤 800。例如，zeroidx 设定为等于第二操作元的大小（例如，16）；resultidx 设定为等于第二操作元的大小（例如，16）；并且 resultmask 设定为全部等于零。Resultmask 在一个实施例中包括对应于第二操作元的 128 位的 128 位。

[0149] 此后，从在此称为 opA 的操作元载入字符（例如，元素），opA 例如是指令的第二操作元，步骤 808。此后，对于是否设定零搜索字段指示零搜索和 zeroidx 等于 16 进行确定，询问 810。如果是，则执行零搜索，步骤 812，并且结果输出到 zeroidx 802 和 resultmask 806。例如，在 zeroidx 中指示零元素的最左字节的字节索引，并且 resultmask 中对应于该元素的位设定一。例如，对于 zeroidx，如果元素大小是字节，并且在字节 5 中发现零元素，那么在 zeroidx 中放置其中发现零元素的字节的索引（例如，5）。类似地，如果元素大小是半字组，并且存在 8 个元素，0-7，并且元素 3（即，字节 6-7）是零，那么 6（对于字节索引 6）放置在 zeroidx 中。类似地，如果元素大小是全字组，并且存在 4 个元素，0-3，并且元素 1（即，字节 4-7）是零，那么 4（对于字节索引 4）放置在 zeroidx 中。如果没有发现空值元素，那么，在一个示例中，向量的大小（例如，以字节计；例如，16）放置在 zeroidx 中。

[0150] 此后，或者如果不执行零搜索，那么载入的字符与操作元中的每个字符比较，操作元在此称为 opB，其例如是指令中的第三操作元，步骤 818。如果存在匹配，那么在 resultidx 804 和 resultmask 806 中指示匹配。例如，resultidx 设定为等于匹配的元素的第一字节，并且对应于该元素的 resultmask 中的位设定为等于一。作为一个示例，对于 resultidx，如果元素大小是字节，并且向量中存在 16 个元素 (0-15)，并且在元素 6 中发现相等，那么 6 存储在 resultidx 中。类似地，如果元素大小是半字组，并且向量中存在 8 个元素 (0-7)，并且在元素 3（例如，在字节 6 或 7）中发现相等，那么返回元素（字节 6）的第一字节的索引。类似地，如果元素大小是全字组，并且向量中存在 4 个元素 (0-3)，并且在元素 1（例如，在字节 4-7）中发现相等，那么返回元素（字节 4）的第一字节的索引。如果不存在相等比较，那么在一个实施例中，resultidx 设定为等于向量的大小（例如，以字节计；例如，16）。

[0151] 此后，对于是否已经到达 opA 的末端进行确定，询问 830。如果没有，那么变量 i 递增例如一，步骤 832，并且处理继续在步骤 808 进行。否则，在确定结果类型等于零的情况下处理继续，询问 834。如果结果类型等于零，那么在此也称为 opC 的操作元（例如指令的第一操作元）设定为等于 resultmask，步骤 836。否则，opC 的指定字节（例如，字节 7）设定为 resulttidx 和 zeroidx 的最小值（并且其他字节设定为零），步骤 838。

[0152] 在将结果放置在 opC 中之后，对于条件码设定字段是否指示条件码设定为零进行确定，询问 840。如果条件码设定字段设定为一，那么设定条件码，步骤 842。例如，如果 ZS 字段设定并且在比第二操作元中的零更低索引元素中不存在匹配，则条件码设定为零；如果第二操作元的一些元素匹配第三操作元中的至少一个元素，则设定为 1；如果第二操作元的所有元素匹配第三操作元中的至少一个元素，则设定为 2；并且如果没有第二操作元的元素匹配第三操作元中的任何元素，则设定为 3。否则，如果条件码设定字段等于零，那么不设定条件码。

[0153] 如在此描述的，在一个实施例中，提供矢量发现任何相等指令，其能够在提供和不提供零搜索之间切换，并且有条件地能够返回零元素或相等元素的字节索引。该字节索引是报告的元素的第一字节。如果搜索零元素，那么可以提供该零元素的位置。因此，提供具有一个特定操作码的一个指令，其中该指令能够执行与零搜索的比较或与非零搜索的比较。

[0154] 上面描述的是用于促进字符数据处理的向量指令的示例。如在此描述的，对于具有 128 位向量的向量寻找元素相等指令，比较逻辑仅执行 16 字节比较，而不是例如 256 比较。这对于更大向量提供缩放。此外，对于向量寻找元素相等指令或向量寻找任何相等指令，可以提供左 / 右控制作为运行时间值，并不在指令内编码。此外，作为结果返回的值是字节位置，而不是元素索引。此外，支持 4 字节比较连同 1 字节和 2 字节比较。

[0155] 根据本发明的一个方面，视情况，基于通过指令提供的控制提供条件码。通过允许不设定条件码，促进指令的调度。

[0156] 在又一个实施例中，零搜索并非条件，而实情为，当执行指令时，执行零搜索。基于或回应于执行该指令，执行零搜索，在一个示例中，且传回零元素的位置（例如，字节索引）和 / 或第一匹配元素的位置（例如，字节索引）。在一个实施例中，针对向量寻找元素相等指令所执行的比较的数目对应于向量的字节的数目，而不管实施例。举例而言，如果正搜索

或进行比较的向量为 16 个字节，则（例如）并行地执行至多 16 个比较。在又一个实施例中，一旦寻找到匹配或零元素，比较便停止。

[0157] 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、存储器与主存储器。

[0158] 作为下文进一步描述的实施方式的部分提供关于向量设施的额外细节（包括其他指令的示例）。

[0159] 如本领域技术人员将了解，本发明的一个或多个方面可体现为系统、方法或计算机程序产品。因此，本发明的一个或多个方面可呈完全硬件实施例、完全软件实施例（包括固件、常驻软件、微码等）或组合软件与硬件方面的实施例的形式，该等实施例在本文中大体上皆可被称作“电路”、“模组”或“系统”。此外，本发明的一个或多个方面可呈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代码）中的计算机程序产品之形式。

[0160] 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、装置或设备或前述各者的任何合适组合。计算机可读存储介质之更特定示例（非详尽清单）包括以下各者：具有一个或多个电线之电连接、携带型计算机磁盘、硬碟、随机存取存储器 (RAM)、唯读存储器 (ROM)、可抹除可程序化唯读存储器 (EPROM 或快闪存储器)、光纤、携带型致密盘 - 唯读存储器 (CD-ROM)、光学存储设备、磁性存储设备或前述各者的任何合适组合。在此文件的情况下，计算机可读存储介质可为可含有或存储供指令执行系统、装置或设备使用或结合指令执行系统、装置或设备而使用的程序的任何有形介质。

[0161] 现参看图 9，在一个示例中，计算机程序产品 900 包括（例如）一个或多个非暂时性计算机可读存储介质 902 以在其上存储计算机可读代码构件或逻辑 904，以提供及促进本发明的一个或多个方面。

[0162] 可使用适当介质（包括（但不限于）无线、有线、光纤缆线、RF 等或前述各者的任何合适组合）传输体现于计算机可读介质上的代码。

[0163] 可以一个或多个程序设计语言的任何组合来撰写用于进行本发明的一个或多个方面的操作的计算机代码，该一个或多个程序设计语言包括诸如 Java、Smalltalk、C++ 或其类似者的面向对象程序设计语言，及诸如“C”程序设计语言、汇编器或类似程序设计语言的传统程序性程序设计语言。代码可完全在使用者的计算机上执行、部分地在使用者的计算机上执行、作为一独立套装软件而执行、部分地在使用者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，远程计算机可通过任一类型的网络连接至使用者的计算机，该任一类型的网络包括局域网络 (LAN) 或广域网络 (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通过因特网）。

[0164] 本文中参考根据本发明的实施例的方法、装置（系统）及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 / 或方块图的每一块及该等流程图说明和 / 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程序化数据处理装置的处理器以产生机器，以使得通过该计算机或其他可程序化数据处理装置的处理器执行的指令产生用于实施该或该等流程图和 / 或方块图块中所指定的功能 / 动作的构件。

[0165] 这些计算机程序指令也可存储在计算机可读介质中，其可指导计算机、其他可程序化数据处理装置或其他设备以特定方式起作用，使得存储在该计算机可读介质中的指令产生制造物件，其包括实施该或该等流程图和／或方块图方块中所指定的功能／动作的指令。

[0166] 也可将该等计算机程序指令载入至计算机、其他可程序化数据处理装置或其他设备上以使得在该计算机、其他可程序化装置或其他设备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程序化装置上执行的指令提供用于实施该或该等流程图和／或方块图块中所指定的功能／动作的处理程序。

[0167] 该等图中的流程图及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、方法及计算机程序产品的可能实施的架构、功能性及操作。就此而言，流程图或方块图中的每一块可表示模组、区段或代码的部分，其包含用于实施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该等块。也应注意，方块图和／或流程图说明的每一块及方块图和／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。

[0168] 除上述内容之外，本发明的一个或多个方面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产生、维持、支持等执行本发明的一个或多个方面的计算机代码和／或计算机基础结构。作为回报，作为示例，服务提供者可在订用和／或收费协议下自客户收取付款。或者或另外，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。

[0169] 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。作为一个示例，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。

[0170] 作为本发明的又一个方面，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。

[0171] 作为本发明的又一个方面，可提供用于整合计算基础结构的处理程序，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。

[0172] 尽管上文描述了各种实施例，但这些实施例仅为示例。举例而言，其他架构的计算环境可并入及使用本发明的一个或多个方面。另外，可使用其他大小的向量，且在不脱离本发明的精神的情况下，可作出对指令的改变。此外，可使用不同于向量寄存器的寄存器，和／或数据可为不同于字符数据的数据，诸如，整数数据或其他类型的数据。

[0173] 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。作为一个示例，适合于存储和／或执行代码的数据处理系统为可使用的，其包括直接或通过系统总线间接耦接至存储器元件的至少两个处理器。该等存储器元件包括（例如）在代码的实际执行期间使用的本地存储器、大容量存储器及高速缓冲存储器，高速缓冲存储器提供至少一些代码的临时存储以便减少在执行期间必须自大容量存储器提取代码的次数。

[0174] 输入 / 输出或 I/O 设备（包括（但不限于）键盘、显示器、指标设备、DASD、磁带、CD、DVD、随身碟及其他存储器介质等）可直接或通过介入的 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通过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存储设备。调制解调器、电缆调制解调器及乙太网卡仅为可用类型的网络适配器中的少数几种。

[0175] 参看图 10，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（也就是说，中央存储器）5002 通信的一个或多个 CPU 5001，以及用于与其他计算机或 SAN 及其类似者通信的至存储介质设备 5011 及网络 5010 的 I/O 接口。CPU 5001 符合具有经建构的指令集及经建构的功能性的架构。CPU 5001 可具有动态地址转译 (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 通常包括转译后备缓冲器 (TLB) 5007 以用于快取转译，使得对计算机存储器 5002 的块的稍后存取不需要延迟地址转译。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 可为阶层式的，其具有可用于一个以上 CPU 的大高速缓冲存储器及在大高速缓冲存储器与每一 CPU 之间的较小的较快（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裂以提供用于指令提取及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，由指令提取单元 5004 通过高速缓冲存储器 5009 自存储器 5002 提取指令。在指令解码单元 5006 中解码指令，且将指令分派（在一些实施例中，通过其他指令）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、浮点执行单元及分支指令执行单元。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元。如果将自存储器 5002 存取（载入或存储）操作元，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。

[0176] 如所注释，计算机系统包括在本地（或主）存储器中的信息，以及寻址、保护及参考及改变记录。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，及将一个类型的地址转译至另一类型的地址的方式。主存储器中的一些存储器包括永久指派的存储位置。主存储器为系统提供数据的可直接寻址的快速存取存储。在可处理数据及程序两者之前将数据及程序两者载入至主存储器中（自输入设备）。

[0177] 主存储器可包括一个或多个较小的快速存取缓冲存储器（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 处理器实体相关联。通过程序大体上不可观测到相异存储介质的实体构造及用途的影响（除了对性能的影响外）。

[0178] 可维持针对指令及针对数据操作元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连字节维持于被称为快取块或快取行（或简称为行）的整体边界上。模型可提供“提取高速缓冲存储器属性 (EXTRACT CACHE ATTRIBUTE)”指令，其传回以字节计的快取行的大小。模型也可提供“预先提取数据 (PREFETCH DATA)”及“预先提取数据相对长度 (PREFETCH DATA RELATIVE LONG)”指令，其实现存储器至数据或指令高速缓冲存储器中的预先提取或数据自高速缓冲存储器的释放。

[0179] 将存储器视为长的水平位串。对于多数操作，对存储器的存取以自左至右序列进行。将该位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置块。存储器中的每一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 开始且以自左至右序列进行。地址为不带正负号的二进位整数，且为 24 个、31 个或 64 个位。

[0180] 在存储器与 CPU 或通道子系统之间一次一字节或一字节群组地传输信息。除非另有指定，否则，在（例如）z/Architecture 中，存储器中的字节群组由该群组的最左侧字节来寻址。通过待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。在每一字节群组内，在（例如）z/Architecture 中，以自左至右序列对位编号。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，位编号并非存储地址。可仅寻址字节。为了对存储器中的字节的个别位操作，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。在多个字节的任一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，且较佳地为了校正的目的，可将一个或多个检查位与每一字节或与字节群组一起传输。由机器自动地产生这些检查位，且不可直接由程序来控制这些检查位。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元字段的长度时，认为该字段具有固定长度，其可为一个、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。当并非隐含而是明确叙述存储操作元字段的长度时，认为该字段具有可变长度。可变长度操作元的长度可以一个字节的增量变化（或通过一些指令，以两个字节的倍数或其他倍数的增量变化）。当将信息置放于存储器中时，替换包括于指定的字段中的仅彼等字节位置的内容，即使至存储器的实体路径的宽度可能大于正存储的字段的长度也如此。

[0181] 某些信息单元将在存储器中的整体边界上。对于信息单元，当其存储地址为该单元的长度（以字节计）的倍数时，将边界称为整体的。对整体边界上的 2 个、4 个、8 个及 16 个字节的字段给予特殊名称。半字组为在两 - 字节边界上的两个连续字节的群组，且为指令的基本建置块。字组为四 - 字节边界上的四个连续字节的群组。双字组为八 - 字节边界上的八个连续字节的群组。四倍字组为 16- 字节边界上的 16 个连续字节的群组。当存储器地址指定半字组、字组、双字组及四倍字组时，地址的二进位表示分别含有一个、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元不具有边界对准要求。

[0182] 在实施针对指令及数据操作元的单独的高速缓冲存储器的设备上，如果程序存储至快取行中（随后自该快取行提取指令），则可经历显著延迟，而不管存储器是否更改随后提取的指令。

[0183] 在一个实施例中，可通过软件（有时指代经授权内部码、固件、微码、毫码、微微码 (pico-code) 及其类似者，前述各者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 10，体现本发明的一个或多个方面的软件代码可由主机系统 5000 的处理器 5001 自长期存储介质设备 5011（诸如，CD-ROM 驱动器、带驱动器或硬驱动器）存取。软件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、硬驱动器或 CD-ROM）使用。代码可散布于这些介质上，或可散布至使用者，通过网络 5010 自计算机系统的计算机存储器 5002 或存储器散布至其他计算机系统，以供这些其他系统的使用者使用。

[0184] 软件代码包括操作系统，其控制各种计算机组件及一个或多个应用程序的功能及交互。通常将代码自存储介质设备 5011 分页至相对较高速计算机存储器 5002，在该相对较高速计算机存储器 5002 处，代码可用于由处理器 5001 处理。用于在存储器中、在实体介质上体现软件代码和 / 或通过网络散布软件代码的技术及方法是已知的，且在本文中将不进一步加以论述。当在有形介质（包括（但不限于）电子存储器模组 (RAM)、快闪存储器、致密盘 (CD)、DVD、磁带及其类似者）上建立及存储代码时，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较佳计算机系统中的处理电路来读取，以用于由处理电路执行。

[0185] 图 11 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 11 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），包括可选的外围设备。基础计算机系统 5021 包括一个或多个处理器 5026，及总线，该总线用于根据已知技术连接处理器 5026 与系统 5021 的其他组件及实现处理器 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 及长期存储器 5027，长期存储器 5027 可包括（例如）硬驱动器（包括（例如）磁性介质、CD、DVD 及快闪存储器中的任一个）或带驱动器。系统 5021 可能还包括使用者接口适配器，使用者接口适配器通过总线将微处理器 5026 连接至一个或多个接口设备（诸如，键盘 5024、鼠标 5023、打印机 / 扫描仪 5030 和 / 或其他接口设备），该等接口设备可为诸如触敏式屏幕、数字化键入板 (entry pad) 等的任何使用者接口设备。总线还通过显示器适配器将诸如 LCD 屏幕或监视器的显示设备 5022 连接至微处理器 5026。

[0186] 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。示例网络适配器为通信信道、令牌环、乙太网络或调制解调器。或者，系统 5021 可使用诸如蜂窝式数字分组数据 (CDPD) 卡的无线接口通信。系统 5021 可与局域网络 (LAN) 或广域网络 (WAN) 中的这些其他计算机相关联，或系统 5021 可为具有另一计算机的用户端 / 服务器配置中的用户端等。所有这些配置以及适当通信硬件及软件为此项技术中已知的。

[0187] 图 12 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 可包括多个个别网络（诸如，无线网络及有线网络），这些网络中的每一个可包括多个个别工作站 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工作站。

[0188] 仍参看图 12，网络还可包括大型计算机或服务器，诸如，网关计算机（用户端服务器 5046) 或应用程序服务器（远程服务器 5048，其可存取数据存储库且还可自工作站 5045 直接存取）。网关计算机 5046 充当至每一个别网络的入口点。当将网络连接协议连接至另一网络连接协议时，需要网关。网关 5046 可较佳借助于通信链路耦接至另一网络（例如，因特网 5047)。还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。可利用可购自国际商用机器公司的 IBM eServer™ System z 服务器来实施网关计算机。

[0189] 同时参看图 11 及图 12，可体现本发明的一个或多个方面的软件代码可由系统 5020 的处理器 5026 自长期存储介质 5027（诸如，CD-ROM 驱动器或硬驱动器）存取。软件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、硬驱动器

或 CD-ROM) 使用。代码可散布于这些介质上，或可散布至使用者 5050、5051，通过网络自计算机系统的存储器或存储器散布至其他计算机系统，以供这些其他系统的使用者使用。

[0190] 或者，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。此代码包括操作系统，其控制各种计算机组件及一个或多个应用程序 5032 的功能及交互。通常将代码自存储介质 5027 分页至高速存储器 5025，在高速存储器 5025 处，代码可用于由处理器 5026 处理。用于在存储器中、在实体介质上体现软件代码和 / 或通过网络散布软件代码的技术及方法是已知的且在本文中将不进一步加以论述。当在有形介质（包括（但不限于）电子存储器模组 (RAM)、快闪存储器、致密盘 (CD)、DVD、磁带及其类似者）上建立及存储代码时，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较佳计算机系统中的处理电路来读取，以用于由处理电路执行。

[0191] 最易于可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 或第一层级）高速缓冲存储器，且主存储器（主存储器）为最高层级高速缓冲存储器（如果存在 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待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）及保持数据操作元的数据高速缓冲存储器 (D- 高速缓冲存储器）。

[0192] 参看图 13，针对处理器 5026 描绘示例性处理器实施例。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良处理器性能。高速缓冲存储器 5053 为保持有可能使用的存储器数据的快取行的高速缓冲器。典型的快取行为 64 个、128 个或 256 个字节的存储器数据。除用于对数据进行快取外，单独的高速缓冲存储器还常常用于对指令进行快取。常常通过此项技术中已知的各种“窥探”演算法来提供快取一致性（存储器及高速缓冲存储器中的行的复制的同步）。处理器系统的主存储器存储器 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，主存储器 5025 有时被称作第 5 层级 (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存储器 (DASD、磁带等）的一部分。主存储器 5025 对由操作系统页入及页出主存储器 5025 的数据页“进行快取”。

[0193] 程序计数器（指令计数器）5061 追踪待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程序状态字组 (PSW) 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（从程序环境至操作系统环境的上下文切换）。在程序并非活动的时，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。精简指令集计算 (RISC) 指令的长度通常为固定的，而复杂指令集计算 (CISC) 指令的长度通常为可变的。IBM z/Architecture 的指令为长度为 2 个、4 个或 6 个字节的 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组中，且载入新程序计数器值从而指向待执行的新程序模组的指令。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。

[0194] 通常，指令提取单元 5055 用于代表处理器 5026 提取指令。提取单元提取“接下来

的顺序指令”、分支选取指令的目标指令或程序的第一指令（在上下文切换之后）。现代指令提取单元常常使用预先提取技术以基于可能使用经预先提取的指令的可能性而推测性地预先提取指令。举例而言，提取单元可提取包括下一个顺序指令的指令的 16 个字节及其他顺序指令的额外字节。

[0195] 接着由处理器 5026 执行所提取的指令。在一个实施例中，将所提取的指令传递至提取单元的分派单元 5056。分派单元解码指令且将关于经解码的指令的信息转递至适当单元 5057、5058、5060。执行单元 5057 通常将自指令提取单元 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元执行算术运算。较佳自存储器 5025、经建构的寄存器 5059 或自正执行的指令的立即字段，将操作元提供至执行单元 5057。当存储执行的结果时，将执行的结果存储在存储器 5025、寄存器 5059 中或其他机器硬件（诸如，控制寄存器、PSW 寄存器及其类似者）中。

[0196] 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 14A，执行单元 5057 可借助于介接逻辑 5071 与经建构的通用寄存器 5059、解码 / 分派单元 5056、载入存储单元 5060 及其他处理器单元 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、乘法及除法）以及逻辑函数（诸如，“及”(and)、“或”(or) 及“互斥或”(XOR)、旋转及移位）。较佳地，ALU 支持相依于设计的专业化运算。其他电路可提供其他经建构的设施 5072，包括（例如）条件码及恢复支持逻辑。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配置，本发明描述仅意欲提供对一个实施例的代表性理解。

[0197] “加法”指令（例如）将在具有算术及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。较佳地，执行单元通过对通过指令识别的操作元执行操作码定义的功能而对操作元进行运算。举例而言，“加法”指令可由执行单元 5057 对在通过指令的寄存器字段识别的两个寄存器 5059 中发现的操作元执行。

[0198] 执行单元 5057 对两个操作元执行算术加法，且将结果存储在第三操作元中，其中第三操作元可为第三寄存器或两个源寄存器中的一个。执行单元较佳利用算术逻辑单元 (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、“及”(And)、“或”(Or) 及“互斥或”(XOR)）以及多种代数函数（包括加法、减法、乘法、除法中的任一个）。一些 ALU 5066 经设计以用于纯量运算且一些 ALU 5066 经设计以用于浮点运算。取决于架构，数据可为大端序 (Big Endian)（其中最低有效字节处于最高字节地址）或小端序 (Little Endian)（其中最低有效字节处于最低字节地址）。IBM z/Architecture 为大端序。取决于架构，带正负号的字段可为正负号及量值 (1 的补数或 2 的补数）。2 的补数是有利的，此在于 :ALU 并不需要设计减法能力，这是由于在 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。

[0199] 参看图 14B，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 常常使用分支预测演算法（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。将提取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及所推测的结果，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（例如）在寄存器字段或指令的立即字段中发现之若干个数字（包括一）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 及输出寄存器电路 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。

[0200] 指令群组的执行可因包括（例如）以下各者之多种原因而被中断：由操作系统起始的上下文切换、引起上下文切换的程序例外状况或错误、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）。较佳地，上下文切换动作保存关于当前正执行的程序的状态信息，且接着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。状态信息较佳包含指向待执行的下一个指令的程序计数器值、条件码、存储器转译信息及经建构的寄存器内容。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代码（微码、微微码或经授权内部码 (LIC)）来训练。

[0201] 处理器根据指令定义的方法来存取操作元。指令可使用指令的一部分的值来提供立即操作元，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元的隐含的寄存器。指令可将存储器位置用于操作元。操作元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 长位移设施 (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元在存储器中的位置的基底寄存器、索引寄存器及立即字段（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（主存储器）中的位置。

[0202] 参看图 14C，处理器使用载入 / 存储单元 5060 来存取存储器。载入 / 存储单元 5060 可通过获得目标操作元在存储器 5053 中的地址且在寄存器 5059 或另一存储器 5053 的位置中载入操作元来执行载入操作，或可通过获得目标操作元在存储器 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在存储器 5053 中的目标操作元位置中来执行存储操作。载入 / 存储单元 5060 可为推测性的，且可以相对于指令序列而言无序的序列存取存储器，然而，载入 / 存储单元 5060 对于程序维持按次序执行指令的出现。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 及控制逻辑 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出现为已按次序执行的功能性，如此项技术中所熟知的。

[0203] 较佳地，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”及“有效地址”。这些虚拟地址为虚拟的在于：其通过多种动态地址转译 (DAT) 技术中的一个而重新导向至实体存储器位置，这些 DAT 技术包括（但不限于）仅对虚拟地址加偏移值 (offset value) 作为报头、通过一个或多个转译表转译虚拟地址，转译表较佳单独或组合地包含至少段表及页表，较佳地，段表具有指向页表的项目。在 z/Architecture 中，提供转译阶层，包括区第一表、区第二表、区第三表、段表及可选的页表。常常通过利用转译后备缓冲器 (TLB)（其包含将虚拟地址映射至相关联的实体存储器位置的项目）来改良地

址转译的性能。当 DAT 使用转译表转译虚拟地址时，建立这些项目。虚拟地址的随后使用可接着利用快速 TLB 的项目，而非缓慢顺序转译表存取。可通过包括最近最少使用 (LRU) 的多种替换演算法来管理 TLB 内容。

[0204] 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 及存储器的共用资源互锁以达成一致性的责任。通常，在维持快取一致性中将利用“窥探”技术。在窥探环境中，可将每一快取行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、改变之状态、无效状态及其类似者。

[0205] I/O 单元 5054（图 13）为处理器提供用于附接至外围设备（例如，包括磁带、致密盘、打印机、显示器及网络）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，通道适配器及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设备之间的通信。

[0206] 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。作为一个示例，环境可包括模拟器（例如，软件或其他模拟机制），在该模拟器中模拟特定架构（包括（例如）指令执行、经建构的功能（诸如，地址转译），及经建构的寄存器）或其子集（例如，在具有处理器及存储器的原生计算机系统上）。在此环境中，模拟器的一个或多个模拟函数可实施本发明的一个或多个方面，即使执行该模拟器的计算机可具有不同于正模拟的能力的架构也如此。作为一个示例，在模拟模式下，解码特定指令或正模拟的操作，且建置适当模拟函数以实施个别指令或操作。

[0207] 在模拟环境中，主机计算机包括（例如）：存储器，其存储指令及数据；指令提取单元，其自存储器提取指令且视情况提供所提取的指令的本地缓冲；指令解码单元，其接收所提取的指令且确定已提取的指令的类型；及指令执行单元，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；或执行某一类型的算术或逻辑运算（如由解码单元确定）。在一个示例中，以软件来实施每一单元。举例而言，将正由这些单元执行的操作实施为模拟器软件内的一个或多个子例程。

[0208] 更明确而言，在大型计算机中，经建构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。存储在存储介质中的这些指令可原生地在 z/Architecture IBM® 服务器中或者在执行其他架构的机器中执行。可在现有及未来 IBM® 大型计算机服务器中及在 IBM® 的其他机器（例如，Power Systems 服务器及 System x® 服务器）上模拟这些指令。可于在使用由 IBM®、Intel®、AMD™ 及其他者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，还可使用 Linux，以及使用由 Hercules、UMX 或 FSI(Fundamental Software, Inc) 进行的模拟的机器，其中执行大体上处于模拟模式下。在模拟模式下，由原生处理器执行模拟软件以模拟经模拟的处理器的架构。

[0209] 原生处理器通常执行包含固件或原生操作系统的模拟软件以执行经模拟的处理器的模拟。模拟软件负责提取及执行经模拟的处理器架构的指令。模拟软件维持经模拟的程序计数器以追踪指令边界。模拟软件可一次提取一个或多个经模拟的机器指令，且将该一个或多个经模拟的机器指令转换至对应的原生机器指令群组，以用于由原生处理器执行。可对这些经转换的指令进行快取，使得可实现较快速转换。尽管如此，模拟软件仍将维

持经模拟的处理器架构的架构规则以便确保操作系统及针对经模拟的处理器撰写的应用程序正确地操作。此外，模拟软件将提供通过经模拟的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）段表及页表的动态地址转译功能、中断机制、上下文切换机制、当日时间 (TOD) 时钟及至 I/O 子系统的经建构的接口），使得操作系统或经设计以在经模拟的处理器上执行的应用程序可在具有模拟软件的原生处理器上执行。

[0210] 解码正进行模拟的特定指令，且调用子例程以执行个别指令的功能。模拟经模拟的处理器的功能的模拟软件功能是（例如）按以下各者来实施：“C”子例程或驱动程序，或在理解较佳实施例的描述之后将在本领域技术人员的技术内的提供用于特定硬件的驱动程序的某一其他方法。包括（但不限于）以下各者的各种软件及硬件模拟专利说明达成针对不同机器建构的指令格式用于可用于本领域技术人员的目标机器的模拟的多种已知方式：Beausoleil 等人的题为“Multiprocessor for Hardware Emulation”的美国专利证书第 5,551,013 号；及 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；及 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；及 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；及 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；及 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）；及许多其他专利证书。

[0211] 在图 15 中，提供经模拟的主机计算机系统 5092 的一个示例，其模拟主机架构的主机计算机系统 5000’。在经模拟的主机计算机系统 5092 中，主机处理器 (CPU) 5091 为经模拟的主机处理器（或虚拟主机处理器），且包含模拟处理器 5093，其具有不同于主机计算机 5000’ 的处理器 5091 的原生指令集架构的原生指令集架构。经模拟的主机计算机系统 5092 具有模拟处理器 5093 可存取的存储器 5094。在示例实施例中，将存储器 5094 分割成主机计算机存储器 5096 部分及模拟例程 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经模拟的主机计算机 5092 的程序。模拟处理器 5093 执行不同于经模拟的处理器 5091 的原生指令的架构的经建构的指令集的原生指令，这些原生指令是自模拟例程存储器 5097 获得，且可通过使用在序列及存取 / 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，序列及存取 / 解码例程可解码所存取的主机指令以确定用于模拟所存取的主机指令的功能的原生指令执行例程。举例而言，针对主机计算机系统 5000’ 的架构定义的其他设施可通过经建构的设施例程来模拟，包括诸如通用寄存器、控制寄存器、动态地址转译及 I/O 子系统支持及处理器高速缓冲存储器的设施。模拟例程还可利用可用于模拟处理器 5093 中的功能（诸如，通用寄存器及虚拟地址的动态转译）以改良模拟例程的性能。还可提供特殊硬件及卸载引擎以辅助处理器 5093 模拟主

机计算机 5000’ 的功能。

[0212] 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”及“该”意欲也包括复数形式。应进一步理解，当术语“包含”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 / 或组件之存在，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 / 或其群组之存在或添加。

[0213] 以下权利要求中的所有构件或步骤加功能元件的对应结构、材料、动作及等效物（如果有的话）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。已出于说明及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揭示之形式下的本发明。在不脱离本发明的范畴及精神的情况下，许多修改及变化对于一般本领域技术人员将为显而易见的。选择并描述了实施例以便最佳地解释本发明的原理及实务应用，且使其他一般本领域技术人员能够针对具有如适合于所预期之特定用途的各种修改的各种实施例来理解本发明。

[0214] 向量串指令

[0215] 向量串设施

[0216] .

[0217] .

[0218] .

[0219] 指令

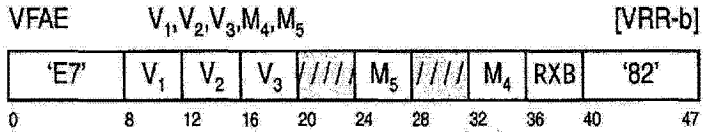
[0220] 除非另有指定，否则所有操作元为向量寄存器操作元。汇编器语法中的“V”指定向量操作元。

[0221]

名称	助忆符号	特性							操作码	
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

[0222] 向量寻找任何相等

[0223]



[0224] 自左至右进行，比较第二操作元的每个不带正负号的二进位整数元素与第三操作元的每个不带正负号的二进位整数元素的相等性，且如果在 M_5 字段中设定了零搜索标记，

则视情况与零比较。

[0225] 如果 M_5 字段中的结果类型 (RT) 标记为零，则对于匹配第三操作元中的任一元素或视情况匹配零的第二操作元中的每一元素，将第一操作元中对应的元素的位位置设定为一，否则，将其设定为零。

[0226] 如果 M_5 字段中的结果类型 (RT) 标记为一，则将匹配第三操作元中的元素或零的第二操作元中的最左侧元素的字节索引存储在第一操作元的字节七中。

[0227] 每一指令具有扩展的助忆符号区段，其描述推荐的扩展的助忆符号及其对应的机器汇编器语法。

[0228] 程序设计注释：对于视情况设定条件码的所有指令，如果设定条件码，则性能可能劣化。

[0229] 如果 M_5 字段中的结果类型 (RT) 标记为一且未发现字节相等，或为零（如果设定了零搜索标记），则将与向量中的字节的数目相等的索引存储在第一操作元的字节七中。

[0230] 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元中的元素的大小。如果指定颠倒值，则识别到规范例外状况。

[0231] 0- 字节

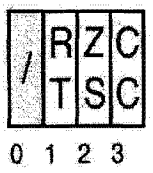
[0232] 1- 半字组

[0233] 2- 字组

[0234] 3 至 15- 保留

[0235] M_5 字段具有以下格式：

[0236]



[0237] 如下定义 M_5 字段的位：

[0238] • 结果类型 (RT)：如果为零，则每一所得元素为关于该元素的所有范围比较的掩码。如果为一，则将字节索引存储至第一操作元的字节七中，且将零存储在所有其他元素中。

[0239] • 零搜索 (ZS)：如果为一，则还将第二操作元的每一元素与零比较。

[0240] • 条件码设定 (CC)：如果为零，则不设定条件码且条件码保持不变。如果为一，则如以下段中所指定来设定条件码。

[0241] 特殊条件

[0242] 如果出现以下各者中的任一个，则识别规范例外状况且不采取其他行动：

[0243] 1. M_4 字段含有自 3 至 15 的值。

[0244] 2. M_5 字段的位 0 并非零。

[0245] 所得条件码：

[0246] 如果 CC 标记为零，则码保持不变。

[0247] 如果 CC 标记为一，则如下来设定码：

[0248] 0 如果设定了 ZS- 位，则在第二操作元中比零低的索引元素中不存在匹配。

[0249] 1 第二操作元的一些元素匹配第三操作元中的至少一个元素。

[0250] 2 第二操作元的所有元素匹配第三操作元中的至少一个元素。

[0251] 3 第二操作元中无元素匹配第三操作元中的任何元素。

[0252] 程序例外状况：

[0253] 1 具有 DXC FE 的数据，向量寄存器

[0254] • 在未安装向量扩展设施的情况下的操作

[0255] • 规范（保留的 ES 值）

[0256] • 异动约束

[0257] 扩展的助忆符号：

[0258]

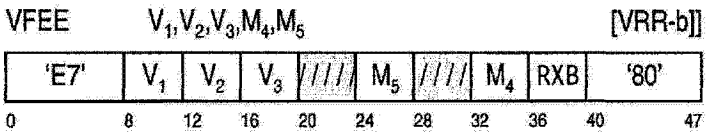
VFAEB V_1,V_2,V_3,M_5	VFAE $V_1,V_2,V_3,0,M_5$
VFAEH V_1,V_2,V_3,M_5	VFAE $V_1,V_2,V_3,1,M_5$
VFAEF V_1,V_2,V_3,M_5	VFAE $V_1,V_2,V_3,2,M_5$
VFAEBS V_1,V_2,V_3,M_5	VFAE $V_1,V_2,V_3,0,(M_5 \mid X'1')$
VFAEHS V_1,V_2,V_3,M_5	VFAE $V_1,V_2,V_3,1,(M_5 \mid X'1')$
VFAEFS V_1,V_2,V_3,M_5	VFAE $V_1,V_2,V_3,2,(M_5 \mid X'1')$

[0259]

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 $V_1,V_2,V_3,2,(M_5 \mid X'3')$

[0260] 向量寻找元素相等

[0261]



[0262] 自左至右进行，将第二操作元的不带正负号的二进位整数元素与第三操作元的对应的不带正负号的二进位整数元素比较。如果两个元素相等，则将最左侧相等元素的第一字节的字节索引置放于第一操作元的字节七中。将零存储在第一操作元的剩余字节中。如果未发现字节相等，或如果未发现字节为零（如果设定了零比较），则将与向量中的字节的数目相等的索引存储在第一操作元的字节七中。将零存储在剩余字节中。

[0263] 如果在 M_5 字段中设定了零搜索 (ZS) 位，则还比较第二操作元中的每一元素与零的相等性。如果在发现第二操作元及第三操作元的任何其他元素相等之前在第二操作元中寻找到零元素，则将发现为零的元素的第一字节的字节索引存储在第一操作元的字节七中，且将零存储在所有其他字节位置中。如果条件码设定 (CC) 标记为一，则将条件码设定为零。

[0264] 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元中的元素的大小。如果指定颠倒值，则识别到规范例外状况。

[0265] 0- 字节

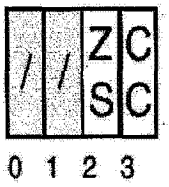
[0266] 1- 半字组

[0267] 2- 字组

[0268] 3 至 15- 保留

[0269] M_5 字段具有以下格式：

[0270]



[0271] 如下定义 M_5 字段的位：

[0272] • 保留：保留位 0 至 1 且位 0 至 1 必须为零。否则，识别到规范例外状况。

[0273] • 零搜索 (ZS)：如果为一，则还将第二操作元的每一元素与零比较。

[0274] • 条件码设定 (CC)：如果为零，则条件码保持不变。如果为一，则如在以下段中所指定来设定条件码。

[0275] 特殊条件

[0276] 如果出现以下各者中的任一个，则识别规范例外状况且不采取其他行动：

[0277] 1. M_4 字段含有自 3 至 15 的值。

[0278] 2. M_5 字段的位 0 至 1 并非零。

[0279] 所得条件码：

[0280] 如果将 M_5 字段的位 3 设定为一，则如下设定码：

[0281] 0 如果设定了零比较位，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元中的零元素。

[0282] 1 比较在一些元素中检测到第二操作元与第三操作元之间的匹配。如果设定了零比较位，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。

[0283] 2--

[0284] 3 无元素比较起来相等。

[0285] 如果 M_5 字段的位 3 为零，则码保持不变。

[0286] 程序例外状况：

[0287] • 具有 DXC FE 的数据，向量寄存器

[0288] • 在未安装向量扩展设施的情况下的操作

[0289] • 规范（保留的 ES 值）

[0290] • 异动约束

[0291] 扩展的助忆符号：

[0292]

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 $V_1,V_2,V_3,2,(M_5 \mid X'1')$
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 $V_1,V_2,V_3,2,(M_5 \mid X'3')$

[0293] 程序设计注释：

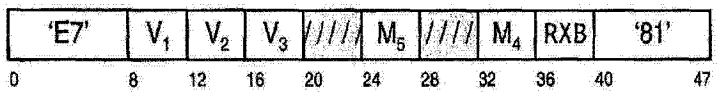
[0294] 1. 对于任一元素大小，始终将字节索引存储至第一操作元中。举例而言，如果将元素大小设定为半字组且比较出第 2 个索引的半字组相等，则将存储字节索引 4。

[0295] 2. 第三操作元不应含有具有零值的元素。如果第三操作元确实含有零且在任何其他相等比较之前与第二操作元中的零元素匹配，则设定条件码一，而不管零比较位设定。

[0296] 向量寻找元素不相等

[0297]

VFENE V_1,V_2,V_3,M_4,M_5 [VRR-b]



[0298] 自左至右进行，将第二操作元的不带正负号的二进位整数元素与第三操作元的对应的不带正负号的二进位整数元素比较。如果两个元素不相等，则将最左侧不相等元素的字节索引置放于第一操作元的字节七中，且将零存储至所有其他字节。如果将 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元较大。如果所有元素相等，则将等于向量大小的字节索引置放于第一操作元的字节七中，且将零置放于所有其他字节位置中。如果 CC 位为一，则设定条件码三。

[0299] 如果在 M_5 字段中设定了零搜索 (ZS) 位，则还比较第二操作元中的每一元素与零的相等性。如果在发现第二操作元的任一其他元素不相等之前在第二操作元中寻找到零元素，则将发现为零的元素的第一字节的字节索引存储在第一操作元的字节七中。将零存储在所有其他字节中，且设定条件码 0。

[0300] 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元中的元素的大小。如果指定颠倒值，则识别到规范例外状况。

[0301] 0- 字节

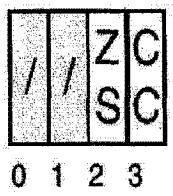
[0302] 1- 半字组

[0303] 2- 字组

[0304] 3 至 15- 保留

[0305] M_5 字段具有以下格式：

[0306]



[0307] 如下定义 M_5 字段的位：

[0308] • 零搜索 (ZS)：如果为一，则还将第二操作元的每一元素与零比较。

[0309] • 条件码设定 (CC)：如果为零，则不设定条件码且条件码保持不变。如果为一，则如以下段中所指定来设定条件码。

[0310] 特殊条件

[0311] 如果出现以下各者中的任一个，则识别规范例外状况且不采取其他行动：

[0312] 1. M_4 字段含有自 3 至 15 的值。

[0313] 2. M_5 字段的位 0 至 1 并非零。

[0314] 所得条件码：

[0315] 如果将 M_5 字段的位 3 设定为一，则如下设定码：

[0316] 0 如果设定了零比较位，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元中的零元素

[0317] 1 检测到元素失配，且 VR2 中的元素小于 VR3 中的元素

[0318] 2 检测到元素失配，且 VR2 中的元素大于 VR3 中的元素

[0319] 3 所有元素比较起来相等，且如果设定了零比较位，则在第二操作元中未寻找到零元素。

[0320] 如果 M_5 字段的位 3 为零，则码保持不变。

[0321] 程序例外状况：

[0322] • 具有 DXC FE 的数据，向量寄存器

[0323] • 在未安装向量扩展设施的情况下的操作

[0324] • 规范（保留的 ES 值）

[0325] • 异动约束

[0326] 扩展的助忆符号：

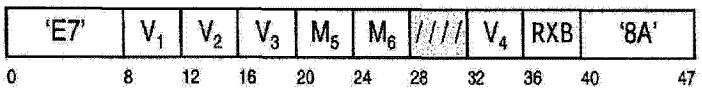
[0327]

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 $V_1,V_2,V_3,2,(M_5 \mid X'3')$

[0328] 向量串范围比较

[0329]

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



[0330] 自左至右进行，将第二操作元中的不带正负号的二进位整数元素与由第三操作元及第四操作元中的偶数—奇数对元素定义的值范围比较。结合来自第四操作元的控制值定义待执行的比较的范围。如果元素匹配由第三操作元及第四操作元指定的范围中的任一个，则将其视为匹配。

[0331] 如果 M_6 字段中的结果类型 (RT) 标记为零，则如果第一操作元中对应于第二操作元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。

[0332] 如果将 M_6 字段中的结果类型 (RT) 标记设定为一，则第二操作元中匹配由第三操作元及第四操作元指定的范围中的任一个或零比较（如果将 ZS 标记设定为一）的第一元素的字节索引置放于第一操作元的字节七中，且将零存储在剩余字节中。如果无元素匹配，则将等于向量中的字节的数目的索引置放于第一操作元的字节七中，且将零存储在剩余字节中。

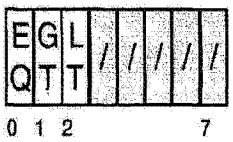
[0333] M_6 字段中的零搜索 (ZS) 标记，如果设定为一，则将第二操作元元素与零的比较添加至由第三操作元及第四操作元提供的范围。如果为在比任何其他真比较低的索引元素中的零比较，则将条件码设定为零。

[0334] 这些操作元含有具有由 M_5 字段中的元素大小控制指定的大小的元素。

[0335] 第四操作元元素具有以下格式：

[0336] 如果 ES 等于 0：

[0337]

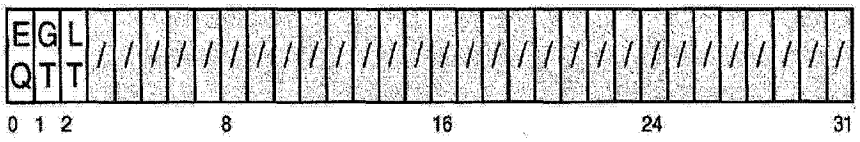


[0338] 如果 ES 等于 1：

[0339]

[0340] 如果 ES 等于 2：

[0341]



[0342] 如下定义第四操作元元素中的位：

[0343] • 相等 (EQ)：当为一时，进行相等性比较。

[0344] • 大于 (GT)：当为一时，执行大于比较。

[0345] • 小于 (LT)：当为一时，执行小于比较。

[0346] • 保留所有其他位且所有其他位应为零以确保未来相容性。

[0347] 可以任何组合使用这些控制位。如果这些位中无一个经设定，则比较将始终产生假结果。如果所有这些位经设定，则比较将始终产生真结果。

[0348] 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元中的元素的大小。如果指定颠倒值，则识别到规范例外状况。

[0349] 0- 字节

[0350] 1- 半字组

[0351] 2- 字组

[0352] 3 至 15- 保留

[0353] M_6 字段具有以下格式：

[0354]

[0355] 如下定义 M_6 字段的位：

[0356] • 反转结果 (IN)：如果为零，则继续进行与控制向量中的成对值的比较。如果为一，则反转在该等范围中的该等成对的比较的结果。

[0357] • 结果类型 (RT)：如果为零，则每一所得元素为关于该元素的所有范围比较的掩码。如果为一，则将索引存储至第一操作元的字节七中。将零存储在剩余字节中。

[0358] • 零搜索 (ZS)：如果为一，则还将第二操作元的每一元素与零比较。

[0359] • 条件码设定 (CC) :如果为零,则不设定条件码且条件码保持不变。如果为一,则如以下段中所指定来设定条件码。

[0360] 特殊条件

[0361] 如果出现以下各者中的任一个,则识别规范例外状况且不采取其他行动 :

[0362] 1. M_4 字段含有自 3 至 15 的值。

[0363] 所得条件码 :

[0364] 0 如果 ZS = 1 且在比任何比较低的索引元素中发现零

[0365] 1 发现比较

[0366] 2--

[0367] 3 未发现比较

[0368] 程序例外状况 :

[0369] • 具有 DXC FE 的数据,向量寄存器

[0370] • 在未安装向量扩展设施的情况下的操作

[0371] • 规范 (保留的 ES 值)

[0372] • 异动约束

[0373] 扩展的助忆符号 :

[0374]

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 $V_1,V_2,V_3,V_4,2,(M_6 \mid X'3')$

[0375]

	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				

[0376] 图 23-1.

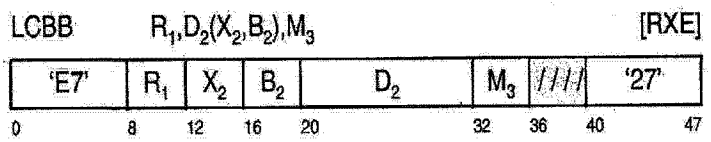
[0377] ES = 1, ZS = 0

[0378] VR1(a)RT = 0 的结果

[0379] VR1(b)RT = 1 的结果

[0380] 将计数载入至块边界

[0381]



[0382] 将含有可能自第二操作元位置载入而不与指定块边界交叉的字节的数目的 32- 位不带正负号的二进位整数（覆盖于十六处）置放于第一操作元中。

[0383] 将位移作为 12- 位不带正负号的整数对待。

[0384] 第二操作元地址不用于寻址数据。

[0385] M_3 字段指定用于用信号向 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。如果指定颠倒值，则识别到规范例外状况。

[0386] 码边界

[0387] 0 64- 字节

[0388] 1 128- 字节

[0389] 2 256- 字节

[0390] 3 512- 字节

[0391] 4 1K- 字节

[0392] 5 2K- 字节

[0393] 6 4K- 字节

[0394] 7 至 15 保留

[0395] 所得条件码：

[0396] 0 操作元一为十六

[0397] 1 --

[0398] 2 --

[0399] 3 操作元一小于十六

[0400] 所得条件码：

[0401] 程序例外状况：

[0402] • 在未安装向量扩展设施的情况下的操作

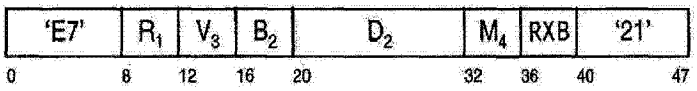
[0403] • 规范

[0404] 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。

[0405] 自 VR 元素的向量载入 GR

[0406]

VLGV $R_1,V_3,D_2(B_2),M_4$ [VRS-c]



[0407] 将第三操作元的具有由 M_4 字段中的 ES 值指定的大小且通过第二操作元地址编索引的元素置放于第一操作元位置中。第三操作元为向量寄存器。第一操作元为通用寄存器。如果通过第二操作元地址指定的索引大于第三操作元中具有指定元素大小的最高编号的元素，则第一操作元中的数据为不可预测的。

[0408] 如果向量寄存器元素小于双字组，则该元素在 64- 位通用寄存器中恰好对准，且零填充剩余位。

[0409] 第二操作元地址不用于寻址数据；实情为，该地址的最右侧 12 个位用于指定元素在第二操作元内的索引。

[0410] 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元中的元素的大小。如果指定颠倒值，则识别到规范例外状况。

[0411] 0- 字节

[0412] 1- 半字组

[0413] 2- 字组

[0414] 3- 双字组

[0415] 4 至 15- 保留不变。

[0416] 所得条件码：码不变。

[0417] 程序例外状况：

[0418] • 具有 DXC FE 的数据，向量寄存器

[0419] • 在未安装向量扩展设施的情况下的操作

[0420] • 规范（保留的 ES 值）

[0421] • 异动约束

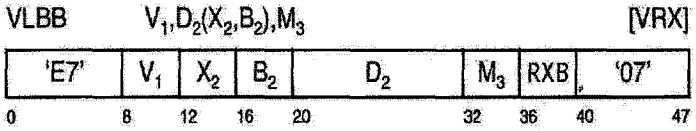
[0422] 扩展的助忆符号：

[0423]

VLGVB	$R_1,V_3,D_2(B2)$	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

[0424] 向量载入至块边界

[0425]



[0426] 以零索引字节元素开始，以来自第二操作元的字节载入第一操作元。如果遇到边界条件，则第一操作元的其余部分为不可预测的。未识别到关于未载入的字节的存取例外状况。

[0427] 将针对 VLBB 的位移作为 12- 位不带正负号的整数对待。

[0428] M_3 字段指定用于用信号向 CPU 通知关于块边界大小以载入至块边界的码。如果指定颠倒值，则识别到规范例外状况。

[0429] 码边界

[0430] 0 64- 字节

[0431] 1 128- 字节

[0432] 2 256- 字节

[0433] 3 512- 字节

[0434] 4 1K- 字节

[0435] 5 2K- 字节

[0436] 6 4K- 字节

[0437] 7 至 15 保留

[0438] 所得条件码 :码保持不变。

[0439] 程序例外状况 :

[0440] • 存取 (提取，操作元 2)

[0441] • 具有 DXC FE 的数据，向量寄存器

[0442] • 在未安装向量扩展设施的情况下的操作

[0443] • 规范 (保留的块边界码)

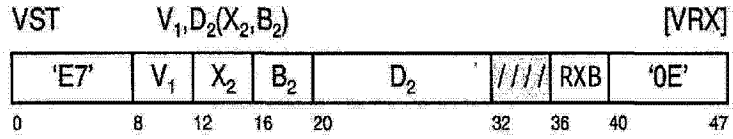
[0444] • 异动约束

[0445] 程序设计注释 :

[0446] 1. 在某些情况下，可经过块边界载入数据。然而，如果不存在关于彼数据的存取例外状况，则将仅发生此情形。

[0447] 向量存储

[0448]



[0449] 将第一操作元中的 128- 位值存储至通过第二操作元指定的存储位置。将针对 VST 的位移作为 12- 位不带正负号的整数对待。

[0450] 所得条件码 :码保持不变。

[0451] 程序例外状况 :

[0452] • 存取（存储，操作元 2）

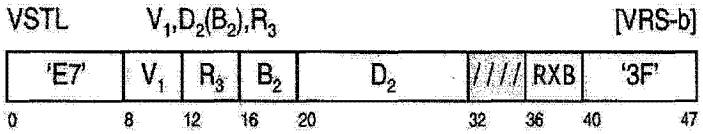
[0453] • 具有 DXC FE 的数据，向量寄存器

[0454] • 在未安装向量扩展设施的情况下的操作

[0455] • 异动约束

[0456] 具有长度的向量存储

[0457]



[0458] 自左至右进行，将来自第一操作元的字节存储在第二操作元位置处。指定第三操作元的通用寄存器含有 32- 位不带正负号的整数，该整数含有表示存储的最高索引字节的值。如果第三操作元含有大于或等于向量的最高字节索引的值，则存储第一操作元的所有字节。

[0459] 仅识别到关于存储的字节的存取例外状况。

[0460] 将针对具有长度的向量存储 (VECTOR STORE WITH LENGTH) 的位移作为 12- 位不带正负号的整数对待。

[0461] 所得条件码：条件码保持不变。

[0462] 程序例外状况：

[0463] • 存取（存储，操作元 2）

[0464] • 具有 DXC FE 的数据，向量寄存器

[0465] • 在未安装向量扩展设施的情况下的操作

[0466] • 异动约束

[0467] RXB 描述

[0468] 所有向量指令具有在指令的位 36 至 40 中的字段，其标注为 RXB。此字段含有用于所有向量寄存器指定的操作元的最高有效位。保留用于未通过指令指定的寄存器指定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 - 位寄存器指定的左侧以建立五 - 位向量寄存器指定。

[0469] 如下定义该等位：

[0470] 0. 在指令的位 8 至 11 中用于向量寄存器指定的最高有效位。

[0471] 1. 在指令的位 12 至 15 中用于向量寄存器指定的最高有效位。

[0472] 2. 在指令的位 16 至 19 中用于向量寄存器指定的最高有效位。

[0473] 3. 在指令的位 32 至 35 中用于向量寄存器指定的最高有效位。

[0474] 向量启用控制

[0475] 如果将控制寄存器零中的向量启用控制（位 46）及 AFP 寄存器控制（位 45）皆设定为一，则可仅使用向量寄存器及指令。如果安装了向量设施且在未设定启用位的情况下执行向量指令，则识别到具有 DXC FE 十六进位的数据例外状况。如果未安装向量设施，则识别到操作例外状况。

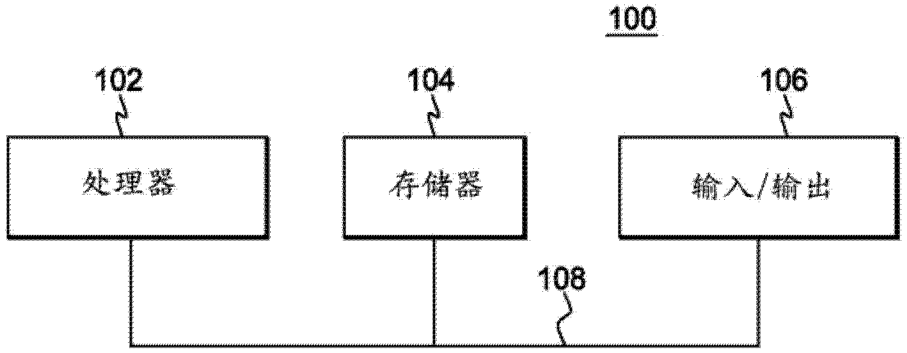

100
102
104
106
处理器
存储器
输入/输出
108

图 1

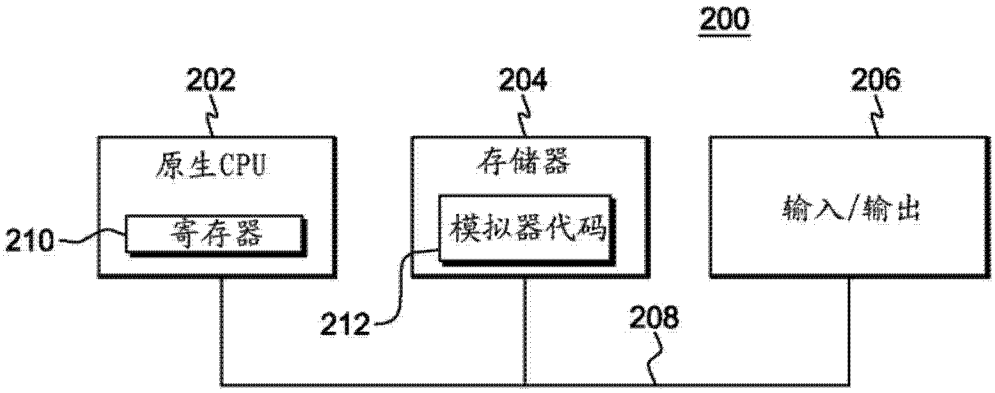

200
202
204
206
原生CPU
寄存器
存储器
模拟器代码
输入/输出
210
212
208

图 2A

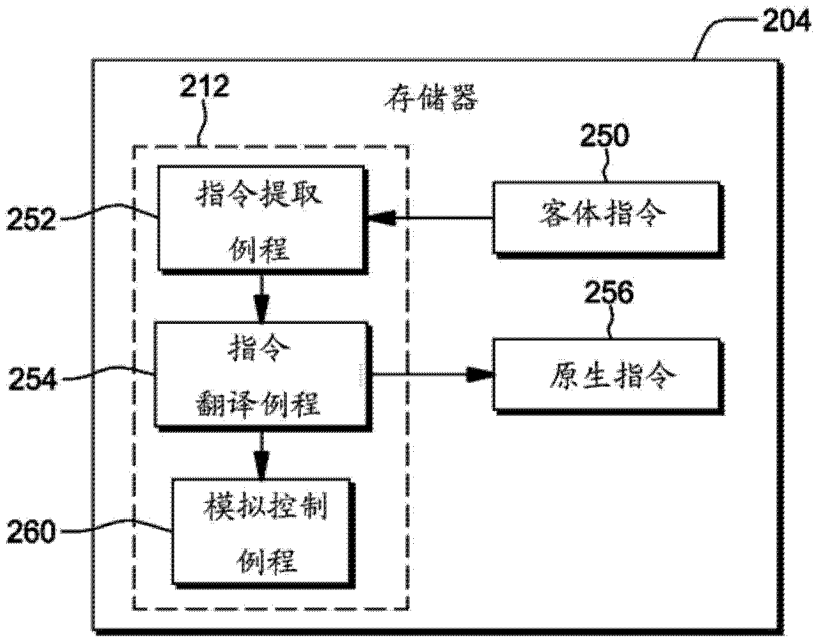

204
212
存储器
250
指令提取例程
客体指令
252
256
指令翻译例程
原生指令
254
模拟控制例程
260

图 2B

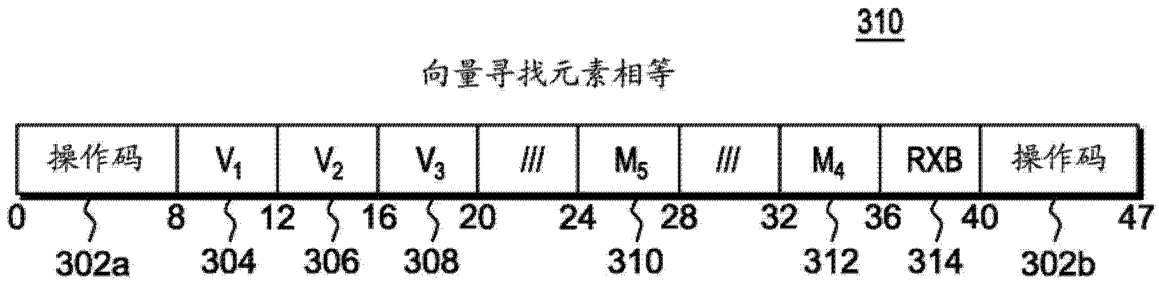

310
向量寻找元素相等
操作码
V1
V2
V3
///
M5
///
M4
RXB
操作码
0
8
12
16
20
24
28
32
36
40
47
302a
304
306
308
310
312
314
302b

图 3

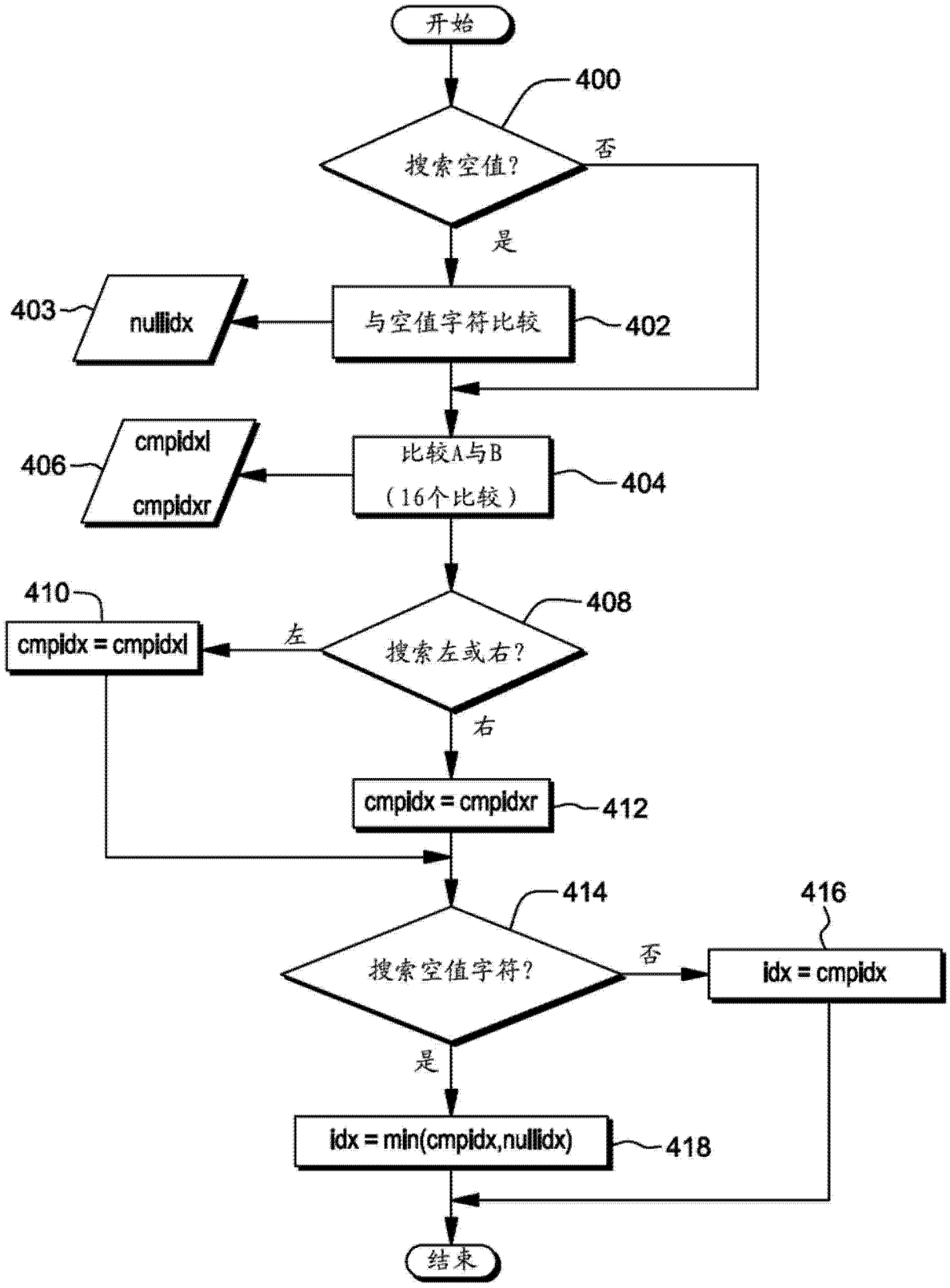

开始
400
搜索空值?
否
是
403
nullidx
与空值字符比较
402
406
cmpidxl
cmpidxr
比较A与B
(16个比较)
404
410
cmpidx = cmpidxl
左
408
搜索左或右?
右
cmpidx = cmpidxr
412
414
416
搜索空值字符?
否
idx = cmpidx
是
idx = min(cmpidx,nullidx)
418
结束

图 4

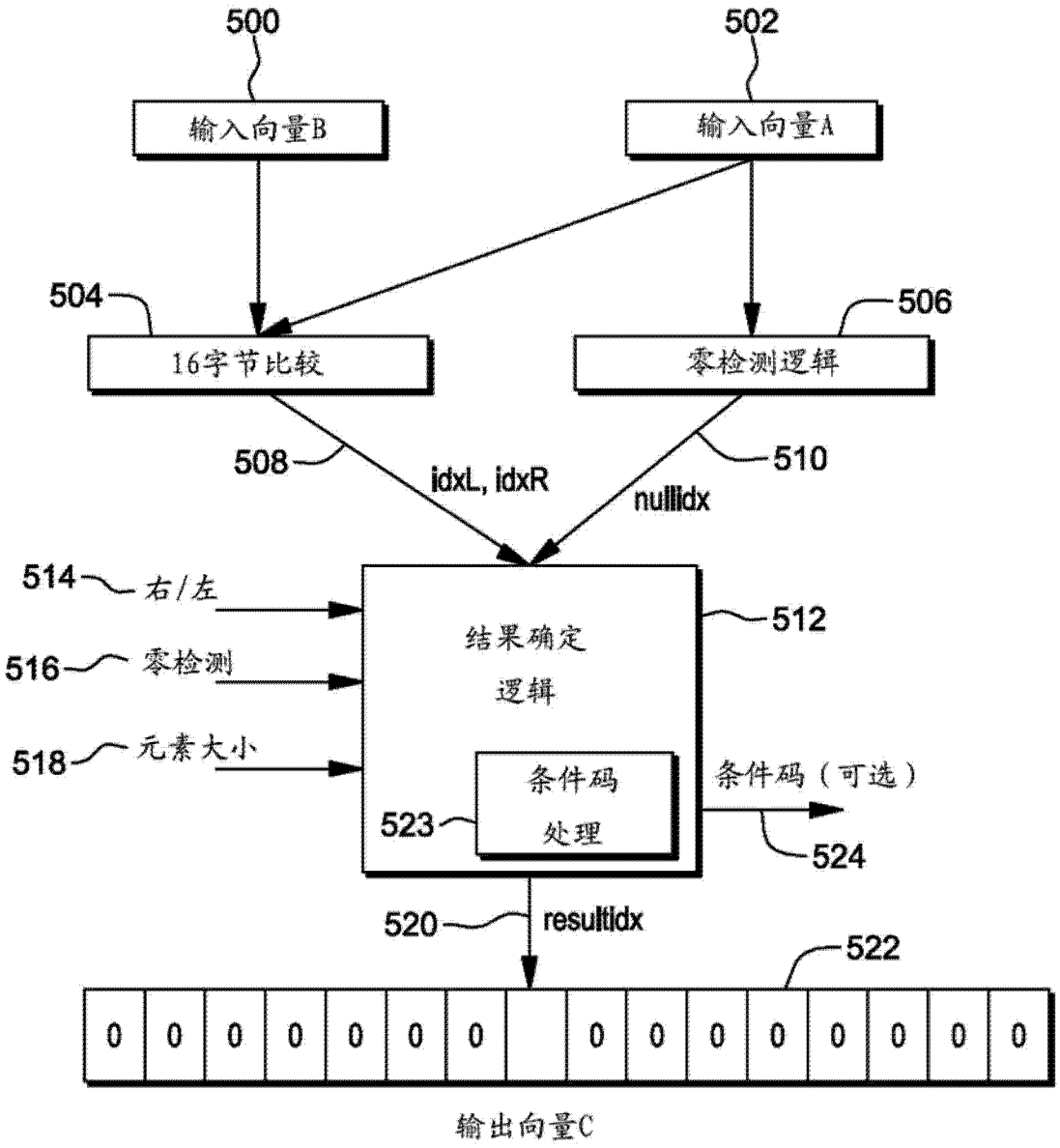

500
502
输入向量B
输入向量A
504
506
16字节比较
零检测逻辑
508
idxL, idxR
510
nullidx
514
右/左
512
结果确定
逻辑
516
零检测
518
元素大小
条件码
处理
523
条件码（可选）
524
520
resultidx
522
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
输出向量C

图 5

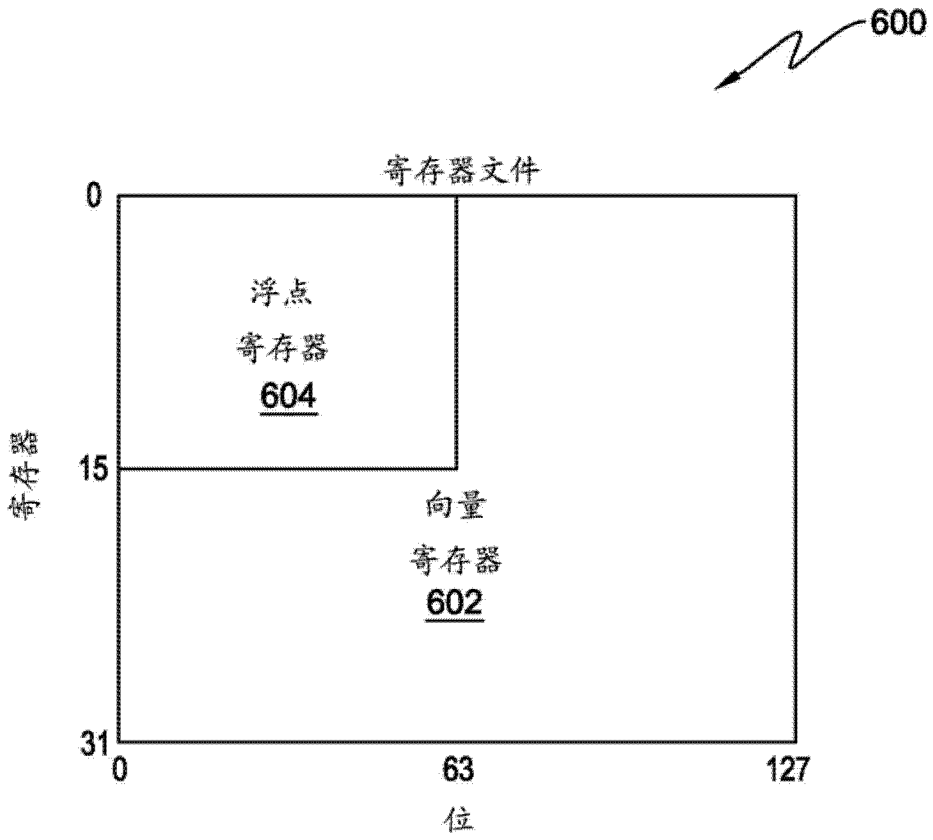

600
寄存器文件
0
浮点
寄存器
604
15
向量
寄存器
602
31
0
63
127
位

图 6

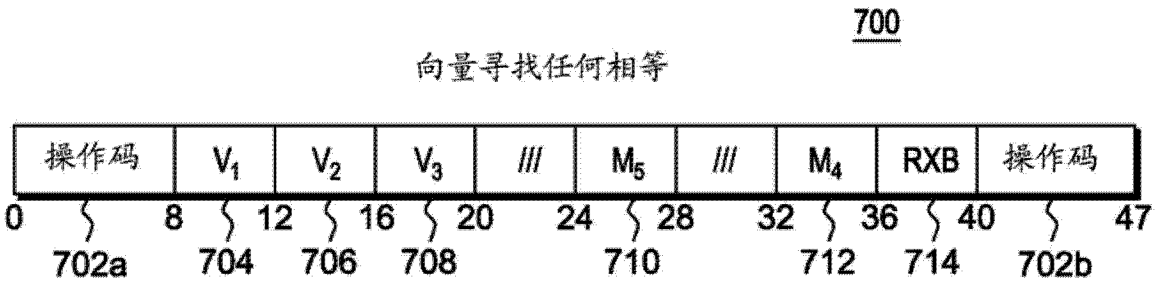

700
向量寻找任何相等
操作码
V1
V2
V3
///
M5
///
M4
RXB
操作码
0
8
12
16
20
24
28
32
36
40
47
702a
704
706
708
710
712
714
702b

图 7

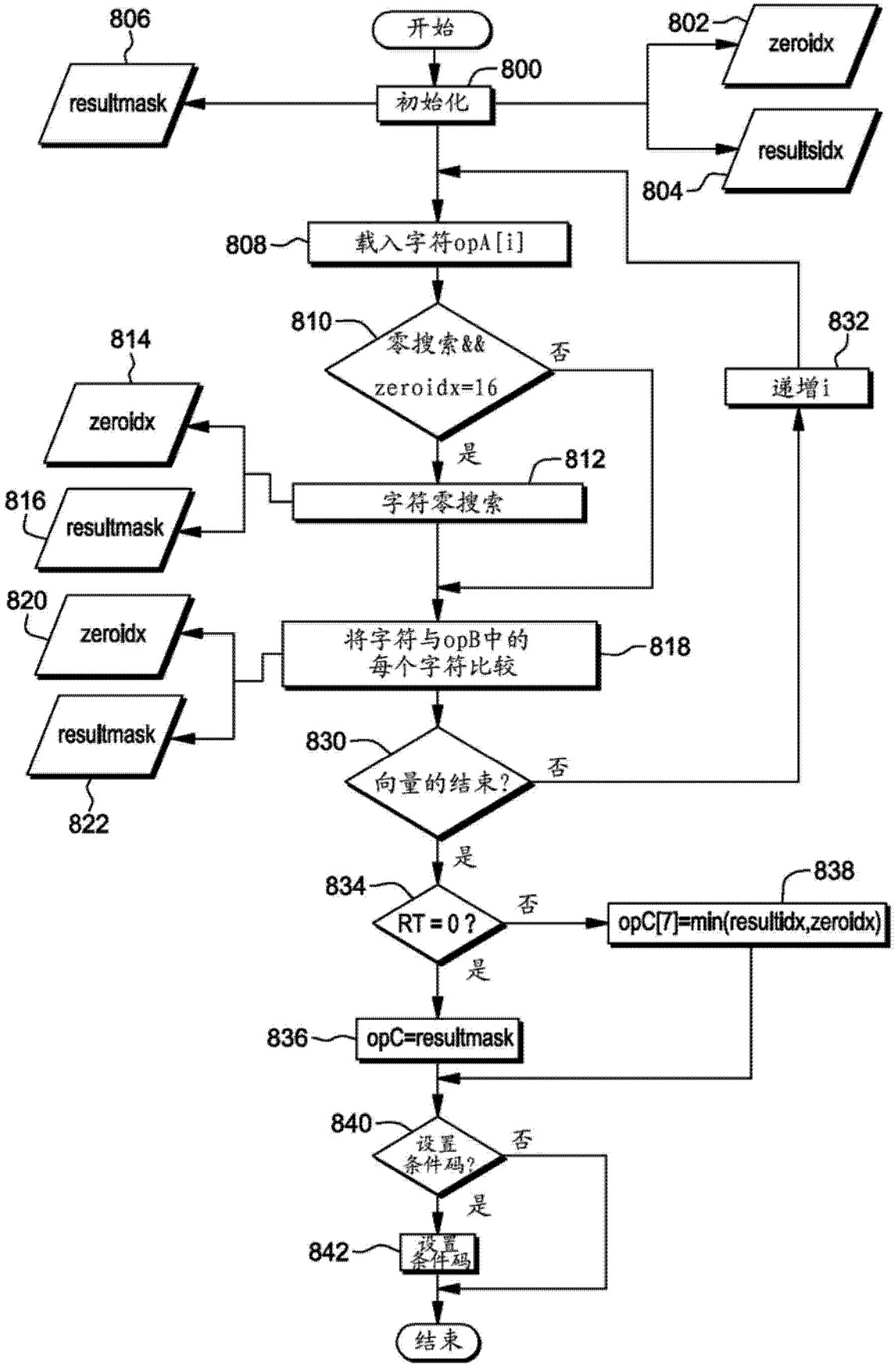

开始
初始化
800
resultmask
806
zeroidx
802
resultsidx
804
载入字符opA[i]
808
零搜索&&
zeroidx=16
810
否
是
递增i
832
zeroidx
814
resultmask
816
字符零搜索
812
将字符与opB中的
每个字符比较
818
zeroidx
820
resultmask
822
向量的结束?
830
否
是
RT = 0 ?
834
否
是
opC[7]=min(resultidx,zeroidx)
838
opC=resultmask
836
设置
条件码?
840
否
是
设置
条件码
842
结束

图 8

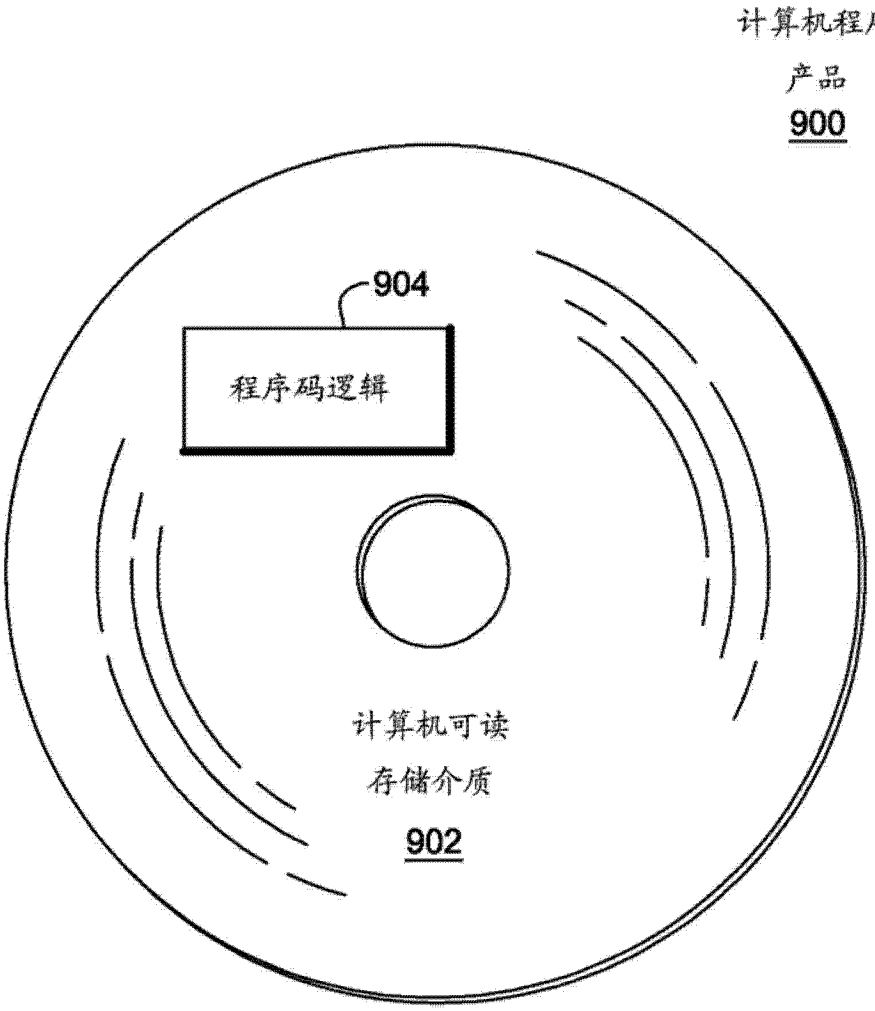


图 9

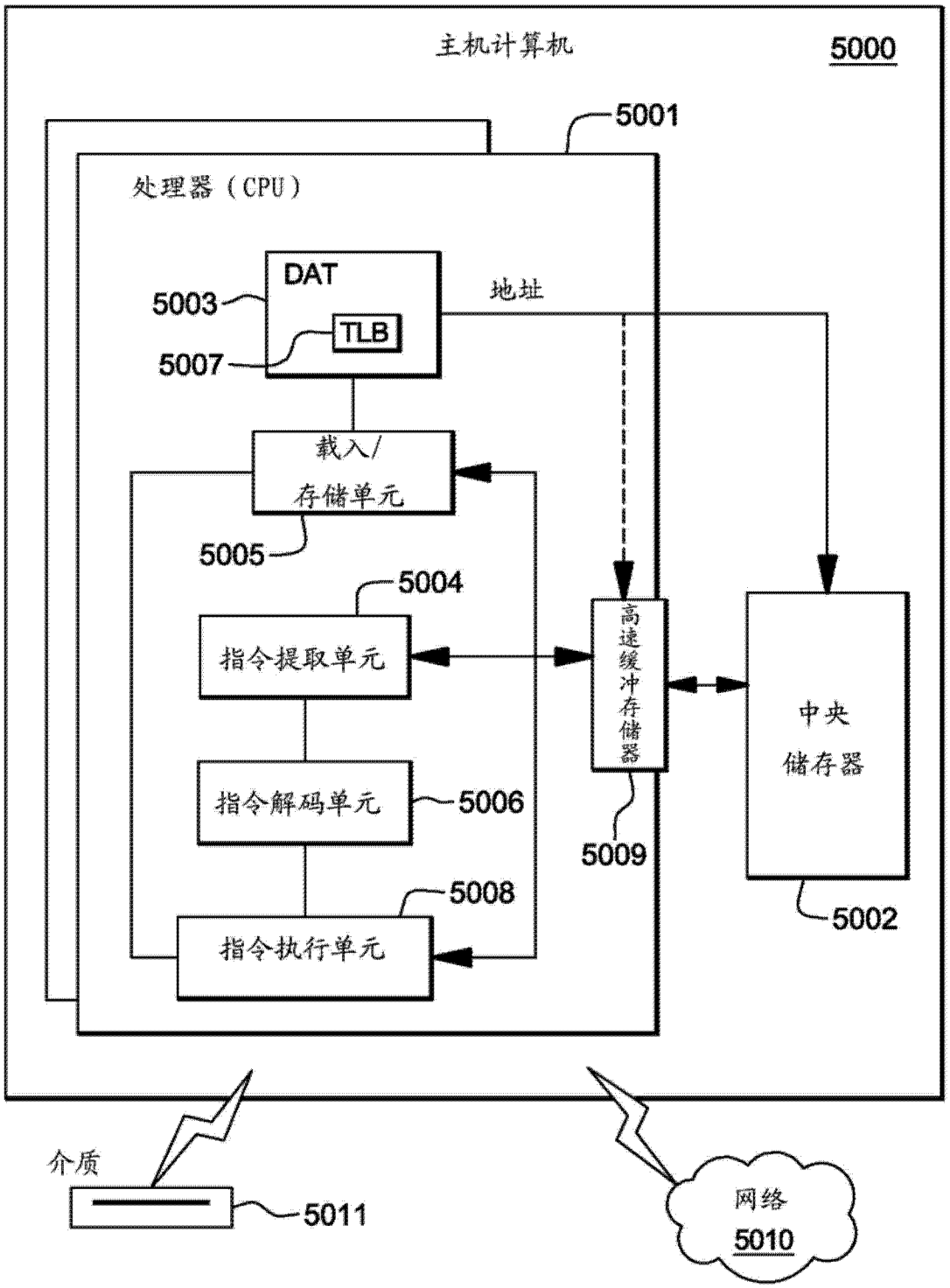

主机计算机
5000
5001
处理器（CPU）
DAT
TLB
5003
5007
地址
载入/
存储单元
5005
5004
指令提取单元
指令解码单元
5006
5008
指令执行单元
高速缓冲存储器
5009
中央
储存器
5002
介质
5011
网络
5010

图 10

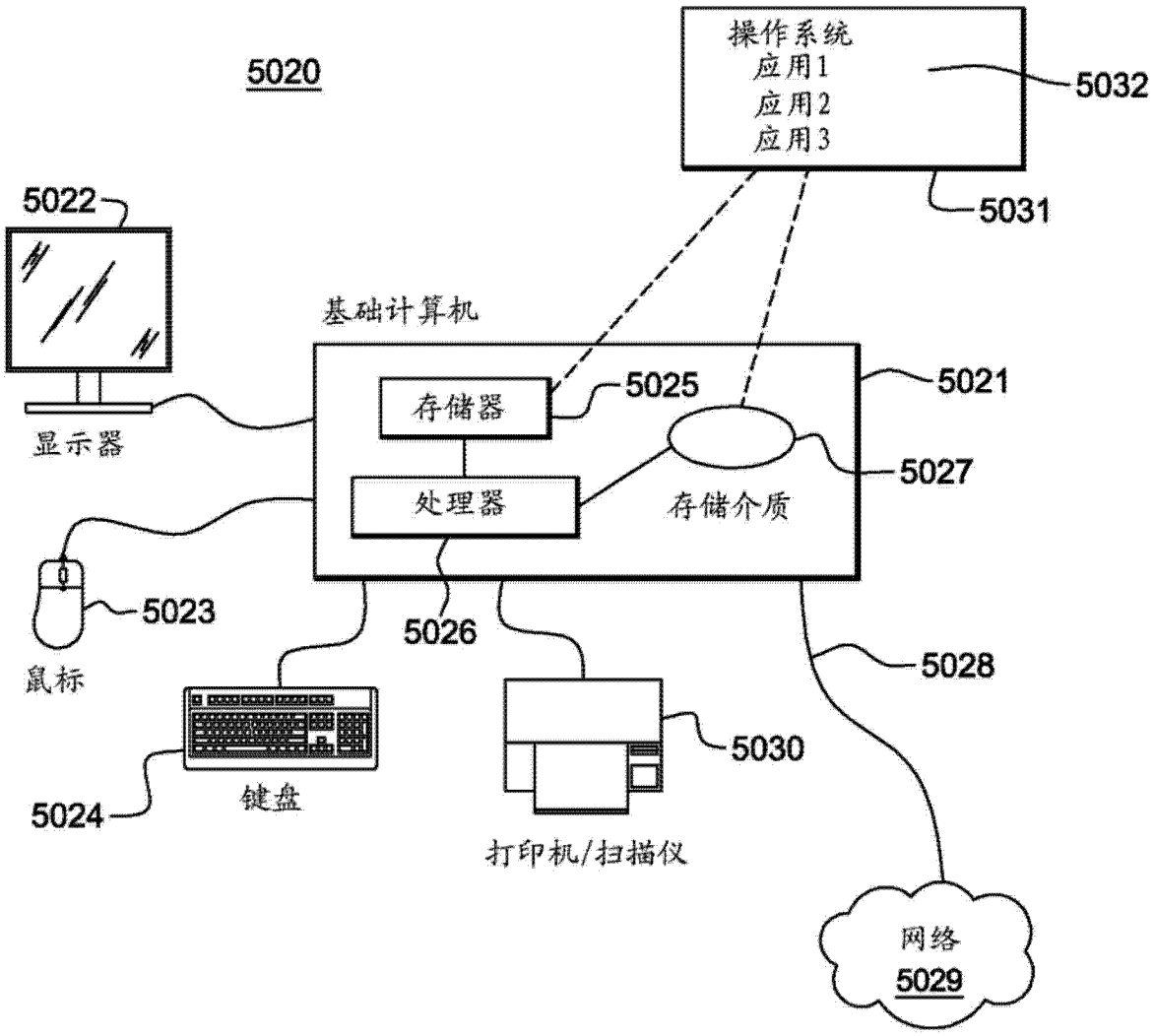

5020
操作系统
应用1
应用2
应用3
5032
5031
5022
基础计算机
存储器
5025
5021
处理器
存储介质
5027
显示器
5023
鼠标
5026
5028
5030
5024
键盘
打印机/扫描仪
网络
5029

图 11

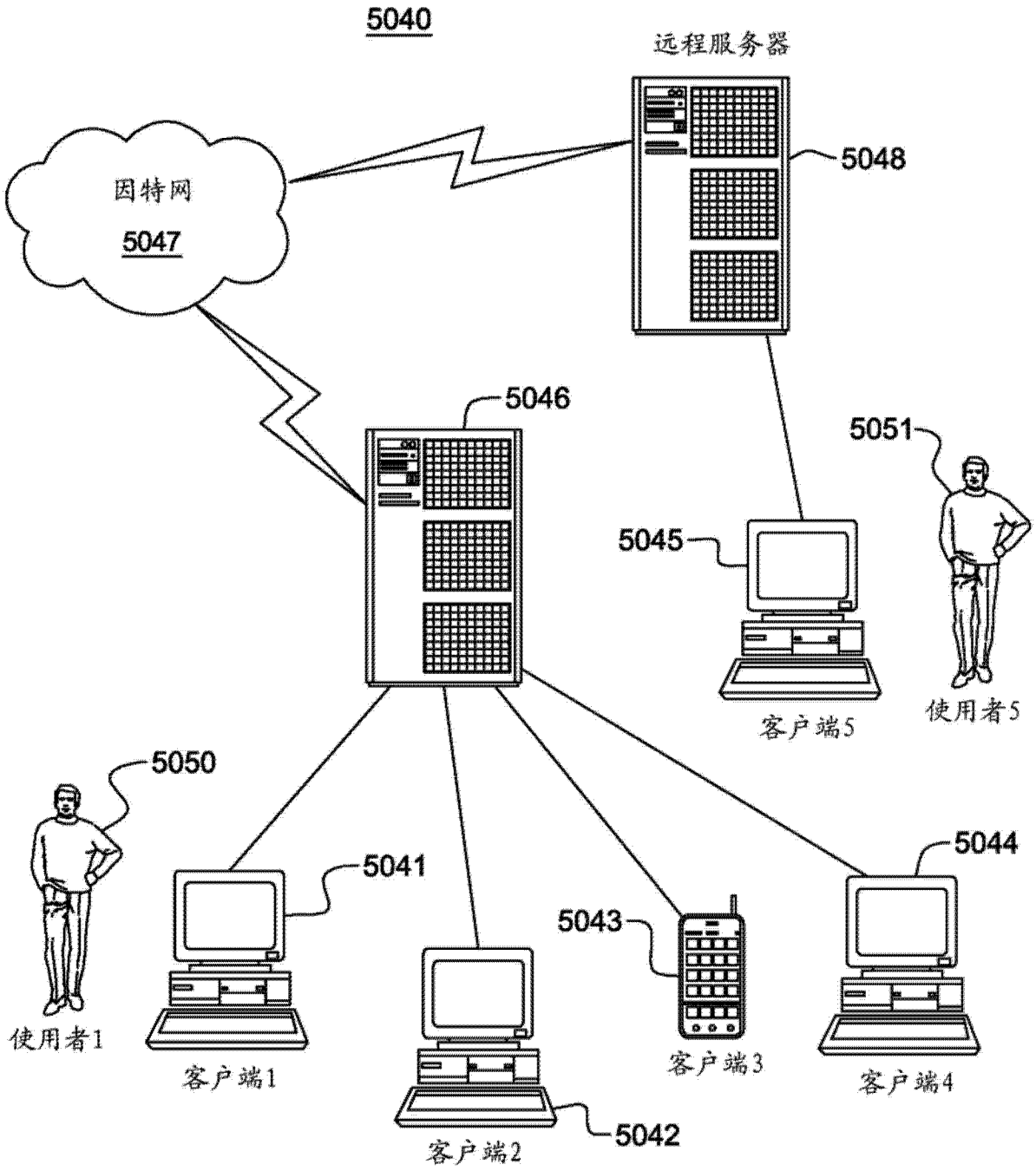

5040
远程服务器
5048
因特网
5047
5046
5051
5045
客户端5
使用者5
5050
5041
5044
5043
使用者1
客户端1
客户端3
客户端4
客户端2
5042

图 12

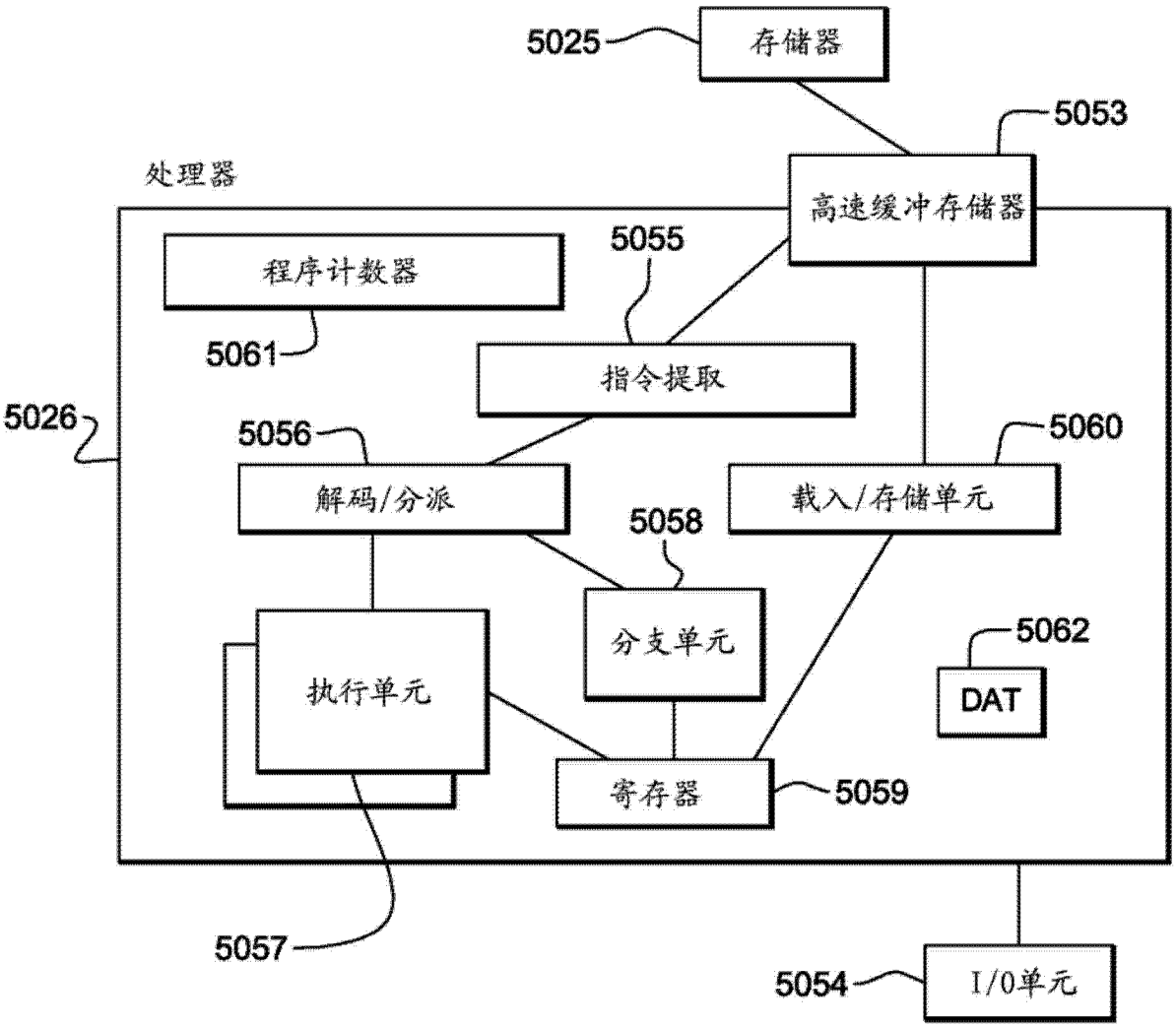

5025
存储器
5053
处理器
高速缓冲存储器
5055
程序计数器
5061
指令提取
5026
5056
5060
解码/分派
载入/存储单元
5058
分支单元
5062
DAT
执行单元
寄存器
5059
5057
5054
I/O单元

图 13

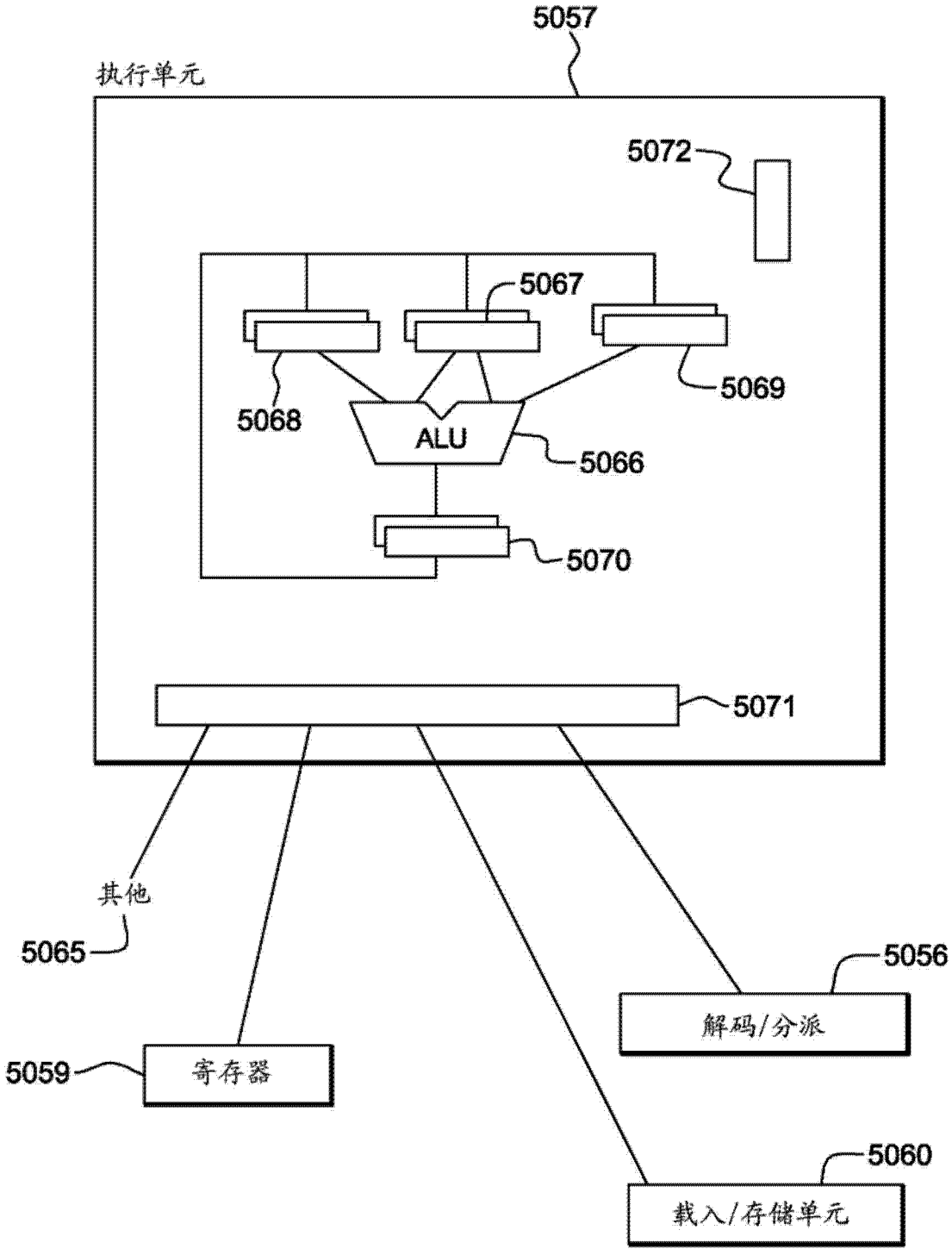

5057
执行单元
5072
5067
5068
ALU
5066
5069
5070
5071
其他
5065
5056
解码/分派
5059
寄存器
5060
载入/存储单元

图 14A

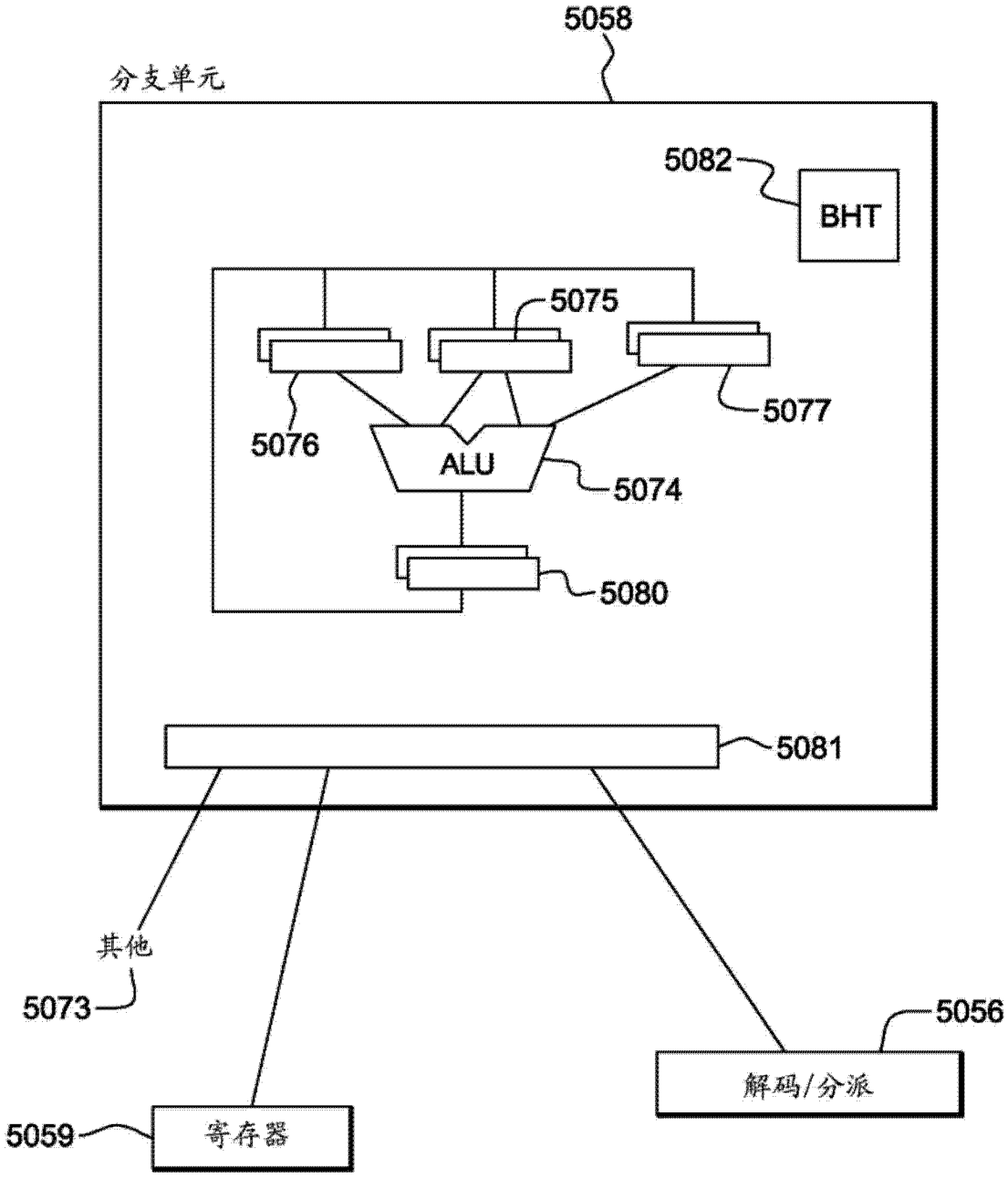

5058
分支单元
5082
BHT
5075
5076
5077
ALU
5074
5080
5081
其他
5073
5056
解码/分派
5059
寄存器

图 14B

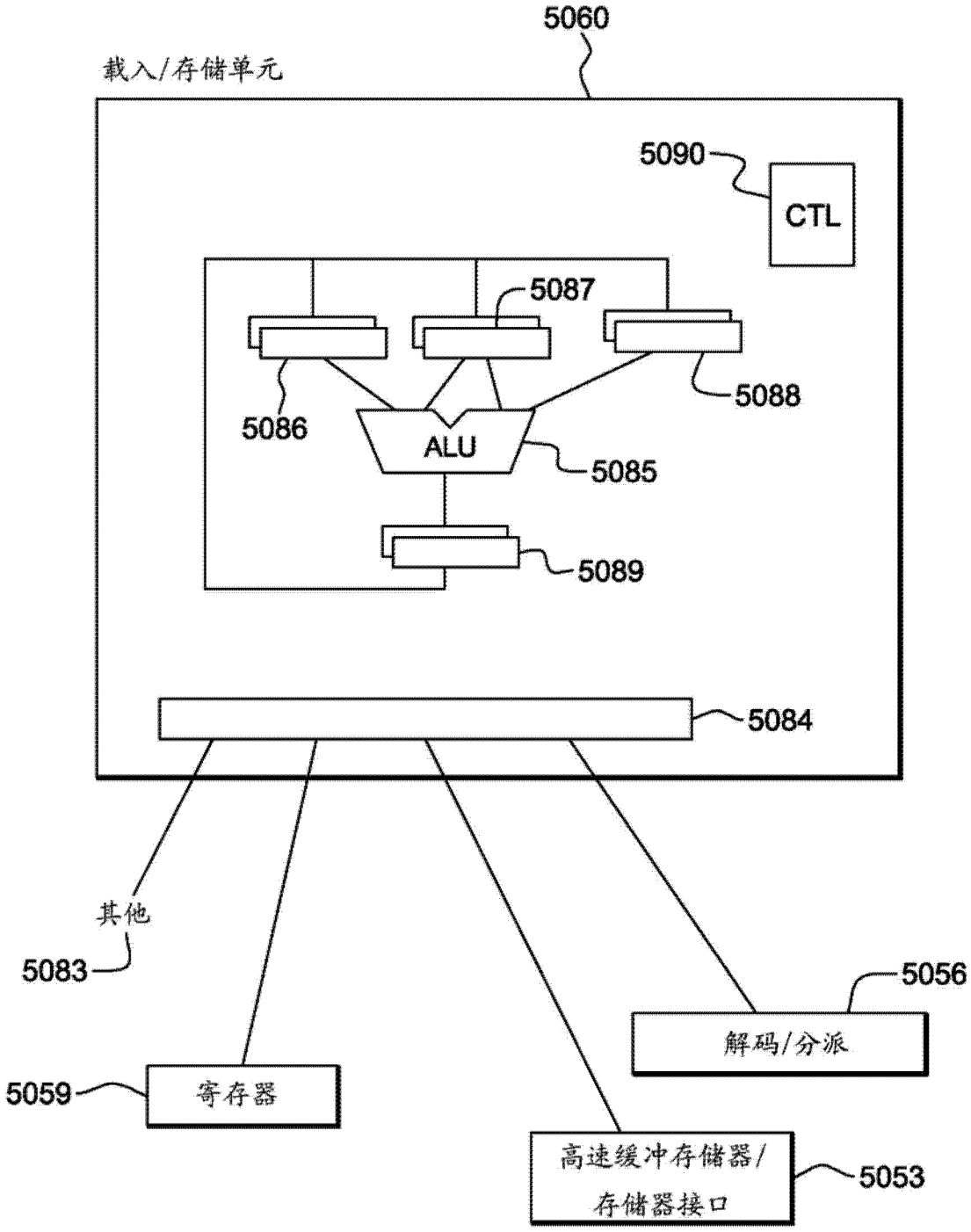

5060
载入/存储单元
5090
CTL
5087
5086
ALU
5088
5085
5089
5084
其他
5083
5056
解码/分派
5059
寄存器
高速缓冲存储器/
存储器接口
5053

图 14C

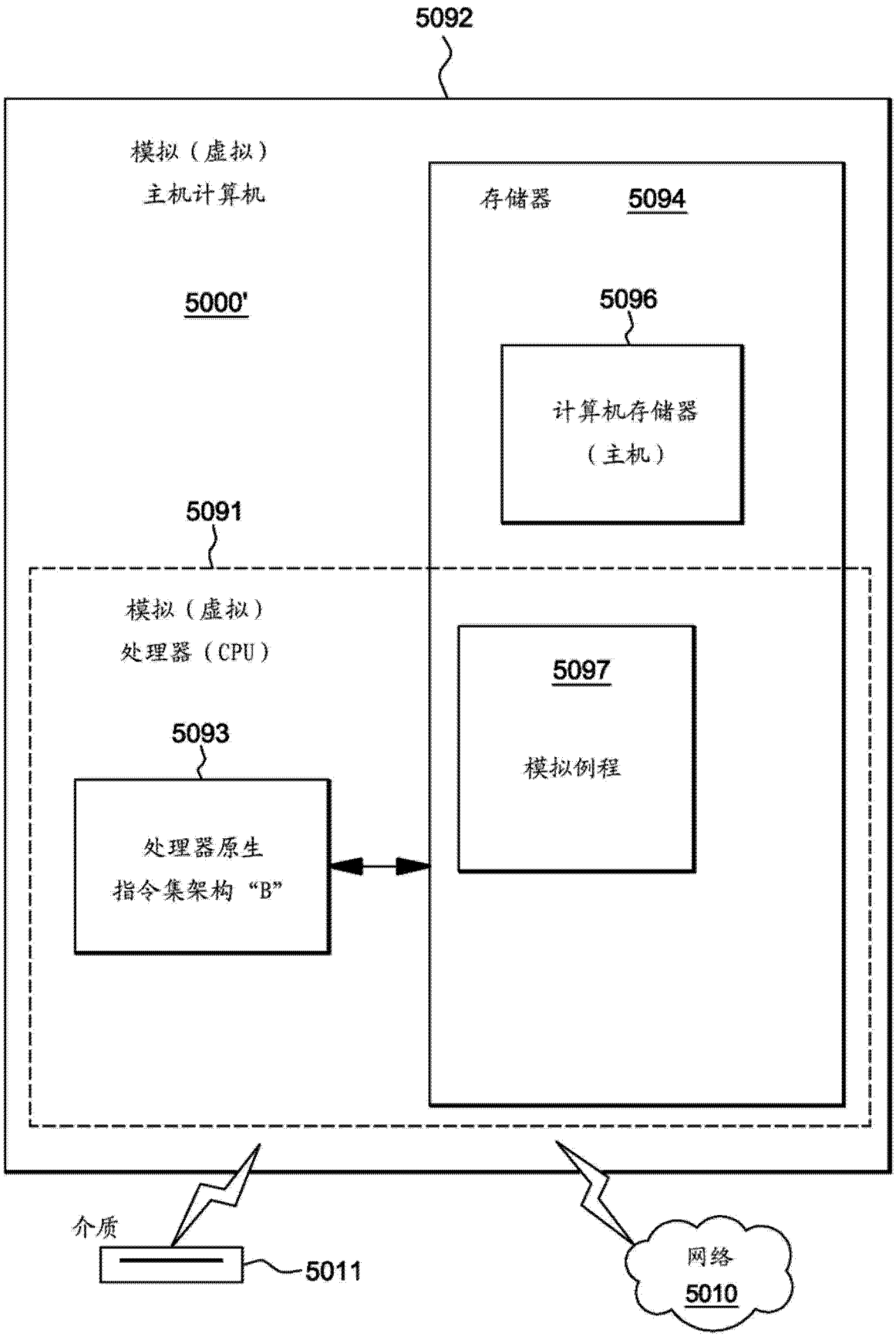

5092
模拟（虚拟）
主机计算机
5000'
存储器
5094
5096
计算机存储器
（主机）
5091
模拟（虚拟）
处理器（CPU）
5097
模拟例程
5093
处理器原生
指令集架构“B”
介质
5011
网络
5010

图 15