(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 115048144 A
(43)申请公布日 2022.09.13

(21)申请号 202210391590.X

(22)申请日 2007.09.24

(30)优先权数据
11/525,981 2006.09.22 US

(62)分案原申请数据
200710305776.4 2007.09.24

(71)申请人 英特尔公司
地址 美国加利福尼亚

(72)发明人 M·朱利耶 J·格雷 S·米克斯
M·塞科尼 S·陈努帕蒂

(74)专利代理机构 永新专利商标代理有限公司
72002
专利代理师 刘瑜

(51)Int.Cl.
G06F 9/38(2006.01)

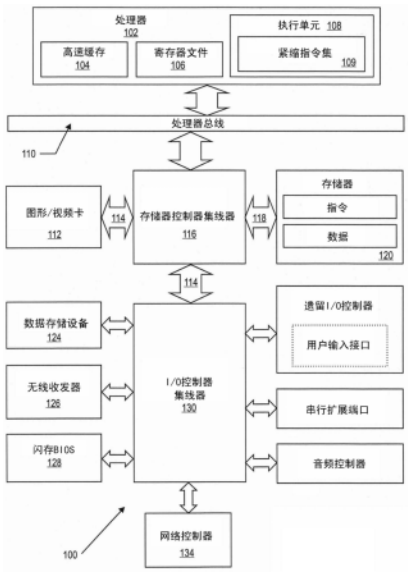
权利要求书6页 说明书16页 附图13页

(54)发明名称

用于处理文本串的指令和逻辑

(57)摘要

本发明提供一种用于处理文本串的指令和逻辑。用于执行串比较操作的方法、装置和程序模块。在一个实施例中，一种装置包括用来执行第一指令的执行资源。响应于所述第一指令，所述执行资源存储分别与第一和第二文本串相对应的第一和第二操作数的每个数据元素之间的比较结果。



CN 115048144 A

1.一种处理器,包括:

多级高速缓存,其包括一级(L1)高速缓存;

译码器,其用于对单指令多数据(SIMD)比较指令进行译码,所述SIMD比较指令具有用于识别第一源SIMD寄存器的第一字段,具有用于识别第二源SIMD寄存器的第二字段,并具有用于识别第三源寄存器的第三字段,所述第一源SIMD寄存器用于存储第一多个数据元素,所述第二源SIMD寄存器用于存储第二多个数据元素,所述第三源寄存器用于存储对应于所述第一多个数据元素中的不同数据元素的多个位;

执行单元,其与所述译码器耦合,所述执行单元用于执行对应于所述SIMD比较指令的操作,所述操作包括:

对于所述多个位中为1的每一个位:

将所述第一多个数据元素中的相应数据元素与所述第二多个数据元素中的至少一个数据元素进行比较;以及

在目的地中存储相应的比较结果,所述相应的比较结果指示所述第一多个数据元素中的相应数据元素是否与所述第二多个数据元素中的任意一个数据元素匹配。

2.根据权利要求1所述的处理器,其中,对于所述多个位中为1的每一个位,所述执行单元用于将所述第一多个数据元素中的相应数据元素与所述第二源SIMD寄存器中的所有数据元素进行比较。

3.根据权利要求1或2所述的处理器,其中,所述执行单元执行对应于所述SIMD比较指令的操作用于在多个标志中存储基于所述结果的指示。

4.根据权利要求1至3中的任意一项所述的处理器,其中,所述第一多个数据元素和所述第二多个数据元素是8位数据元素,并且其中,对于所述多个位中为1的每一个位,所述执行单元用于将所述第一多个数据元素中的相应数据元素与所述第二多个数据元素中的16个数据元素进行比较。

5.根据权利要求1至4中的任意一项所述的处理器,其中,所述第一多个数据元素和所述第二多个数据元素是16位数据元素,并且其中,对于所述多个位中为1的每一个位,所述执行单元用于将所述第一多个数据元素中的相应数据元素与所述第二多个数据元素中的8个数据元素进行比较。

6.根据权利要求1至5中的任意一项所述的处理器,其中,所述第一多个数据元素是8位数据元素或者16位数据元素。

7.根据权利要求1至6中的任意一项所述的处理器,其中,所述结果包括掩码值。

8.根据权利要求1至7中的任意一项所述的处理器,其中,所述处理器是精简指令集计算(RISC)处理器。

9.根据权利要求1至8中的任意一项所述的处理器,其中,所述第一多个数据元素是字符串,并且其中,所述指令是串处理指令。

10.根据权利要求1至9中的任意一项所述的处理器,其中,所述SIMD比较指令是串处理指令。

11.一种方法,包括:

接收单指令多数据(SIMD)比较指令;以及

执行操作以实施所述SIMD比较指令,包括:

从第一SIMD寄存器接收第一字符串的第一多个数据元素;

从第二SIMD寄存器接收第二字符串的第二多个数据元素;

从第三源寄存器接收对应于所述第一多个数据元素中的不同数据元素的多个位;以及

对于所述多个位中为1的每一个位:

将所述第一多个数据元素中的相应数据元素与所述第二多个数据元素中的至少一个数据元素进行比较;以及

在结果中存储相应的比较结果,所述相应的比较结果指示所述第一多个数据元素中的相应数据元素是否与所述第二多个数据元素中的任意一个数据元素匹配。

12.根据权利要求11所述的方法,其中,所述第一多个数据元素是字符串,其中,对于所述多个位中为1的每一个位,所述比较包括将所述第一多个数据元素中的相应数据元素与所述第二源SIMD寄存器中的所有数据元素进行比较。

13.根据权利要求11或12所述的方法,其中,所述第一多个数据元素和所述第二多个数据元素是16位数据元素,其中,对于所述多个位中为1的每一个位,所述比较包括将所述第一多个数据元素中的相应数据元素与所述第二多个数据元素中的8个数据元素进行比较,并且所述方法还包括使用所述比较结果来检测所述第一串中的字与所述第二串中的字匹配。

14.一种机器可读介质,其上存储有指令,所述指令在由机器执行时使得所述机器执行包括以下的操作:

接收单指令多数据(SIMD)比较指令;以及

执行操作以实施所述SIMD比较指令,包括:

从第一SIMD寄存器接收第一字符串的第一多个数据元素;

从第二SIMD寄存器接收第二字符串的第二多个数据元素;

从第三源寄存器接收对应于所述第一多个数据元素中的不同数据元素的多个位;以及

对于所述多个位中为1的每一个位:

将所述第一多个数据元素中的相应数据元素与所述第二多个数据元素中的至少一个数据元素进行比较;以及

在结果中存储相应的比较结果,所述相应的比较结果指示所述第一多个数据元素中的相应数据元素是否与所述第二多个数据元素中的任意一个数据元素匹配。

15.根据权利要求14所述的机器可读介质,其中,所述指令在由所述机器执行时使得所述机器执行包括以下的操作:对于所述多个位中为1的每一个位,将所述第一多个数据元素中的相应数据元素与所述第二源SIMD寄存器中的所有数据元素进行比较,其中,所述第一多个数据元素中的相应数据元素是文本串的一部分。

16.根据权利要求14或15所述的机器可读介质,其中,在由所述机器执行时使得所述机器执行将所述第一多个数据元素中的相应数据元素与所述第二多个数据元素中的至少一个数据元素进行比较的所述指令还包括:在由所述机器执行时使得所述机器将所述第一多个数据元素中的相应数据元素与所述第二多个数据元素中的8个16位数据元素进行比较的指令,并且还包括在由所述机器执行时使得所述机器执行包括使用所述比较结果来检测所述第一串中的字与所述第二串中的字匹配的操作的指令。

17.一种系统,包括:

存储器控制器;以及

处理器,其与所述存储器控制器耦合,所述处理器包括:

多级高速缓存,其包括一级(L1)高速缓存;

译码器,其用于对单指令多数据(SIMD)比较指令进行译码,所述SIMD比较指令具有用于识别第一源SIMD寄存器的第一字段,具有用于识别第二源SIMD寄存器的第二字段,并具有用于识别第三源寄存器的第三字段,所述第一源SIMD寄存器用于存储第一多个数据元素,所述第二源SIMD寄存器用于存储第二多个数据元素,所述第三源寄存器用于存储对应于所述第一多个数据元素中的不同数据元素的多个位;

执行单元,其与所述译码器耦合,所述执行单元用于执行对应于所述SIMD比较指令的操作,所述操作包括:

对于所述多个位中为1的每一个位:

将所述第一多个数据元素中的相应数据元素与所述第二多个数据元素中的至少一个数据元素进行比较;以及

在目的地中存储相应的比较结果,所述相应的比较结果指示所述第一多个数据元素中的相应数据元素是否与所述第二多个数据元素中的任意一个数据元素匹配。

18.根据权利要求17所述的系统,其中,对于所述多个位中为1的每一个位,所述执行单元用于将所述第一多个数据元素中的相应数据元素与所述第二源SIMD寄存器中的所有数据元素进行比较,并且所述系统还包括与所述处理器耦合的图形组件。

19.根据权利要求17或18所述的系统,其中,所述执行单元执行对应于所述SIMD比较指令的操作用于在多个标志中存储基于所述结果的指示,并且其中,所述SIMD比较指令是串处理指令,并且所述系统还包括与所述处理器耦合的输入/输出控制器。

20.根据权利要求17至19中的任意一项所述的系统,其中,所述第一多个数据元素和所述第二多个数据元素是8位数据元素,并且其中,对于所述多个位中为1的每一个位,所述执行单元用于将所述第一多个数据元素中的相应数据元素与所述第二多个数据元素中的16个数据元素进行比较,并且所述系统还包括与所述处理器耦合的音频控制器。

21.根据权利要求17至20中的任意一项所述的系统,其中,所述第一多个数据元素和所述第二多个数据元素是16位数据元素,并且其中,对于所述多个位中为1的每一个位,所述执行单元用于将所述第一多个数据元素中的相应数据元素与所述第二多个数据元素中的8个数据元素进行比较,其中,所述结果包括掩码值,并且所述系统还包括与所述处理器耦合的到网络控制器的接口。

22.一种系统,包括:

系统存储器;以及

处理器,其与所述系统存储器耦合,所述处理器包括:

多级高速缓存,其包括一级(L1)高速缓存;

译码器,其用于对单指令多数据(SIMD)比较指令进行译码,所述SIMD比较指令具有用于识别第一源SIMD寄存器的第一字段,具有用于识别第二源SIMD寄存器的第二字段,并具有用于识别第三源寄存器的第三字段,所述第一源SIMD寄存器用于存储第一多个数据元素,所述第二源SIMD寄存器用于存储第二多个数据元素,所述第二源寄存器用于存储对应于所述第一多个数据元素中的不同数据元素的多个位;

执行单元,其与所述译码器耦合,所述执行单元用于执行对应于所述SIMD比较指令的操作,所述操作包括:

对于所述多个位中为1的每一个位:

将所述第一多个数据元素中的相应数据元素与所述第二多个数据元素中的至少一个数据元素进行比较;以及

在目的地中存储相应的比较结果,所述相应的比较结果指示所述第一多个数据元素中的相应数据元素是否与所述第二多个数据元素中的任意一个数据元素匹配。

23.根据权利要求22所述的系统,其中,对于所述多个位中为1的每一个位,所述执行单元用于将所述第一多个数据元素中的相应数据元素与所述第二源SIMD寄存器中的所有数据元素进行比较,其中,所述系统存储器包括动态随机存取存储器(DRAM),并且所述系统还包括与所述处理器耦合的网络控制器。

24.根据权利要求22或23所述的系统,其中,所述执行单元执行对应于所述SIMD比较指令的操作用于在多个标志中存储基于所述结果的指示,并且其中,所述SIMD比较指令是串处理指令,所述系统还包括与所述处理器耦合的大容量存储设备。

25.根据权利要求22至24中的任意一项所述的系统,其中,所述第一多个数据元素和所述第二多个数据元素是8位数据元素,并且其中,对于所述多个位中为1的每一个位,所述执行单元用于将所述第一多个数据元素中的相应数据元素与所述第二多个数据元素中的16个数据元素进行比较,所述系统还包括与所述处理器耦合的盘驱动器。

26.根据权利要求22至25中的任意一项所述的系统,其中,所述第一多个数据元素和所述第二多个数据元素是16位数据元素,并且其中,对于所述多个位中为1的每一个位,所述执行单元用于将所述第一多个数据元素中的相应数据元素与所述第二多个数据元素中的8个数据元素进行比较,并且其中,所述结果包括掩码值,所述系统还包括与所述处理器耦合的I/O设备。

27.一种装置,包括:

多个寄存器,用于存储128位紧缩数据操作数;

译码电路,其用于对包括第一指令的指令进行译码,其中,所述第一指令用于识别所述多个寄存器中的第一128位源紧缩数据操作数、以及存储器或所述多个寄存器中的第二128位源紧缩数据操作数,其中,所述第一128位源紧缩数据操作数和所述第二128位源紧缩数据操作数中的有效数据元素由所述第一128位源紧缩数据操作数和所述第二128位源紧缩数据操作数中的空元素来隐含地指示,并且其中,所述第一指令具有8位立即字段,所述8位立即字段包括:

第一两位,其中,所述第一两位等于00B指示所述第一128位源紧缩数据操作数和所述第二128位源紧缩数据操作数的数据元素是无符号字节,等于01B指示所述数据元素是无符号字,等于10B指示所述数据元素是有符号字节,等于11B指示所述数据元素是有符号字;

第二两位,其中,所述第二两位等于00B指示要执行的比较是要确定所述第一128位源紧缩数据操作数和所述第二128位源紧缩数据操作数是否有任意有效数据元素相等,等于01B指示所述比较涉及范围,等于11B指示所述比较是要确定所述第一128位源紧缩数据操作数和所述第二128位源紧缩数据操作数是否具有相等的排序的多个数据元素;

第三两位,用于控制求负操作,其中,所述第三两位等于00B指示所述比较的结果的所

有位不会被求负,等于01B指示所述比较的结果的所有位会被求负,等于11B指示所述比较的结果中只有与所述第一128位源紧缩数据操作数和所述第二128位源紧缩数据操作数其中之一的有效数据元素相对应的位会被求负;以及

第四位,用于指示是所述求负操作的结果的最低有效置位的索引、还是所述求负操作的结果的最高有效置位的索引将被存储在寄存器中;

执行电路,其与所述译码电路耦合,用于执行对应于所述第一指令的操作。

28.一种装置,包括:

多个寄存器,用于存储128位紧缩数据操作数;

译码电路,其用于对包括第一指令的指令进行译码,其中,所述第一指令用于识别所述多个寄存器中的第一128位源紧缩数据操作数、以及存储器或所述多个寄存器中的第二128位源紧缩数据操作数,其中,所述第一128位源紧缩数据操作数和所述第二128位源紧缩数据操作数中的有效数据元素由所述第一128位源紧缩数据操作数和所述第二128位源紧缩数据操作数中的空元素来隐含地指示,并且其中,所述第一指令具有8位立即字段,所述8位立即字段包括:

第一两位,其中,所述第一两位等于00B指示所述第一128位源紧缩数据操作数和所述第二128位源紧缩数据操作数的数据元素是无符号字节,等于01B指示所述数据元素是无符号字,等于10B指示所述数据元素是有符号字节,等于11B指示所述数据元素是有符号字;

第二两位,其中,所述第二两位指示要对所述第一128位源紧缩数据操作数和所述第二128位源紧缩数据操作数的数据元素执行的比较的类型;

第三两位,用于控制求负操作,其中,所述第三两位等于00B指示所述比较的结果的所有位不会被求负,等于01B指示所述比较的结果的所有位会被求负,等于11B指示所述比较的结果中只有与所述第一128位源紧缩数据操作数和所述第二128位源紧缩数据操作数其中之一的有效数据元素相对应的位会被求负;以及

第四位,用于指示是所述求负操作的结果的最低有效置位的索引、还是所述求负操作的结果的最高有效置位的索引将被存储在寄存器中;

执行电路,其与所述译码电路耦合,用于执行对应于所述第一指令的操作。

29.一种处理器,包括:

多级高速缓存,其包括一级(L1)高速缓存;

多个单指令多数据(SIMD)寄存器,其包括用于存储包括多个数据元素的第一源操作数的第一源SIMD寄存器、以及用于存储包括多个数据元素的第二源操作数的第二源SIMD寄存器;

第一寄存器,其用于存储多个位,每个位对应于所述第一源操作数的不同数据元素;

译码器,其用于对包括SIMD比较指令的指令进行译码,所述SIMD比较指令具有用于识别所述第一源SIMD寄存器的第一字段,并具有用于识别所述第二源SIMD寄存器的第二字段;以及

执行单元,其与所述译码器耦合并与所述多个SIMD寄存器耦合,响应于对所述SIMD比较指令的译码,所述执行单元用于:

将所述第二源操作数中的第一数据元素与所述第一源操作数中所述第一寄存器的相应位为1的第一多个数据元素进行比较;以及

在目的地寄存器中存储结果，所述结果的一部分指示所述第一数据元素与所述第一源操作数中所述第一寄存器的相应位为1的第一多个数据元素的比较结果，其中，所述结果不存储针对所述第一源操作数中所述第一寄存器的相应位为0的数据元素的比较结果。

30.根据权利要求29所述的处理器，其中，所述SIMD比较指令具有多个字段用来指定多个不同类型的比较。

31.根据权利要求29所述的处理器，其中，响应于对所述SIMD比较指令的译码，所述执行单元用于使用标志来作出基于所述结果的指示。

32.根据权利要求29所述的处理器，其中，所述第一源操作数的数据元素是整数，所述整数允许是有符号的或无符号的。

33.根据权利要求29所述的处理器，其中，所述第一源操作数的数据元素能够是8位、16位、以及32位中的任意一种。

34.根据权利要求29所述的处理器，其中，响应于对所述SIMD比较指令的译码，所述执行单元用于将所述第二源操作数中的每一个数据元素与所述第一源操作数中的至多8个数据元素进行比较。

35.一种系统，包括：

第一存储器，其用于存储SIMD比较指令；

处理器，其用于执行所述SIMD比较指令以确定第一紧缩操作数中的哪些数据元素和第二紧缩操作数中的哪些数据元素是有效的，将由所述SIMD比较指令指示的所述第一紧缩操作数的每个有效数据元素与所述第二紧缩操作数的每个有效数据元素进行比较，并仅针对在所述第一紧缩操作数和所述第二紧缩操作数的有效数据元素之间进行的比较来存储所述SIMD比较指令的结果。

36.一种处理器，包括：

指令获取逻辑，其用于获取一个或多个指令；

指令译码逻辑，其用于对所述一个或多个指令进行译码；

寄存器文件，其包括一组128位紧缩数据寄存器，用于存储紧缩单精度浮点(SPFP)数据元素和紧缩整数数据元素；以及

一个或多个执行单元，其用于执行比较指令以将存储在第一紧缩数据寄存器中的第一多个紧缩无符号整数数据元素与存储在第二紧缩数据寄存器中的第二多个紧缩无符号整数数据元素进行比较，并且在第三紧缩数据寄存器中存储多个数据元素，其中，所述多个数据元素用于表示在所述第一多个紧缩无符号整数数据元素和所述第二多个紧缩无符号整数数据元素之间的比较的结果，并且其中，所述比较指令包括掩码信号，所述掩码信号用于指示是生成零扩展掩码还是扩展掩码作为所述比较的结果。

37.根据权利要求36所述的处理器，其中，所述指令用于使得仅将所述第一多个紧缩无符号整数数据元素和所述第二多个紧缩无符号整数数据元素的明确有效的数据元素进行比较。

38.根据权利要求36所述的处理器，其中，所述指令用于使得仅将所述第一多个紧缩无符号整数数据元素和所述第二多个紧缩无符号整数数据元素的隐含有效的数据元素进行比较。

用于处理文本串的指令和逻辑

[0001] 本申请是申请日为2007年9月24日、申请号为201610875178.X的同名专利申请的分案申请。

技术领域

[0002] 本公开内容属于执行逻辑和数学操作的处理装置以及相关软件和软件序列的领域。

背景技术

[0003] 在我们的社会中，计算机系统已经日益变得普遍。计算机的处理能力提高了许多职业范围内的工作者的效率和生产率。随着购买和拥有计算机的费用持续下降，越来越多的消费者能够利用更新和更快的机器。此外，许多人因为自由而热衷于使用笔记本电脑。移动计算机允许用户在他们离开办公室或旅行时容易地运送他们的数据和工作。这种情况对于销售人员、团体执行人员甚至学生都是十分常见的。

[0004] 随着处理器技术的发展，还产生了要在具有这些处理器的机器上运行的更新的软件代码。用户通常希望和需要从他们的计算机获得更高的性能，而不考虑所使用的软件类型。从所述处理器内实际执行的多种指令和操作将会产生一个这样的问题。基于操作的复杂度和/或所需的电路类型，特定类型的操作需要更多的时间来完成。这提供了一个机会来优化在处理器内执行特定复杂操作的方式。

[0005] 通信应用已经推动了微处理器发展超过十年。实际上，计算和通信之间的界线已经变得越来越模糊，这部分地是由于文本通信应用的使用。文本应用在消费者层面内很普遍，并且在从蜂窝电话到个人计算机的多种设备中要求越来越块的文本信息处理。文本通信设备继续以诸如Microsoft® Instant Messenger™的应用、诸如Microsoft® Outlook™的电子邮件应用以及蜂窝电话文本应用的形式在计算和通信设备中得以应用。结果是，未来的个人计算和通信体验将具有更为丰富的文本能力。

[0006] 因此，对于目前的计算和通信设备而言，在计算或通信设备之间传输的文本信息的处理或解析变得越来越重要。特别是，由通信或计算设备进行的文本信息串的解释包括一些在文本数据上执行的最为重要的操作。这样的操作可以是计算密集型的，但是提供能够通过使用各种数据存储设备的有效实现方式而利用的高度的数据并行性，所述数据存储设备诸如例如单指令多数据(SIMD)寄存器。多种当前的体系结构还要求多个操作、指令或子指令(常称作“微操作”或“uop”)来对多个操作数执行各种逻辑和数学操作，从而减少吞吐量并增加执行所述逻辑和数学操作所需的时钟周期数。

[0007] 例如，可能需要由多个指令构成的指令序列来执行解释文本串的特定词语所需的一个或多个操作，包括将由处理装置、系统或计算机程序内的各种数据类型所表示的两个或多个文本词语进行比较。然而，这样的现有技术可能需要许多处理周期，并且可能导致处理器或系统为了生成结果而消耗不必要的能量。此外，一些现有技术可能对可在其上进行操作的操作数数据类型受到限制。

附图说明

[0008] 本发明通过实例进行说明,而不局限于附图中:

[0009] 图1A是根据本发明一个实施例的计算机系统的框图,所述计算机系统具有处理器,所述处理器包括用来执行串比较操作的指令的执行单元;

[0010] 图1B是根据本发明可选实施例的另一个示例性计算机系统的框图;

[0011] 图1C是根据本发明另一个可选实施例的又一个示例性计算机系统的框图;

[0012] 图2是根据本发明一个实施例的处理器的微体系结构的框图,所述处理器包括用来执行一个或多个串比较操作的逻辑电路;

[0013] 图3A表示根据本发明一个实施例的在多媒体寄存器中的各种紧缩数据类型(packed data type)表示;

[0014] 图3B表示根据可选实施例的紧缩数据类型;

[0015] 图3C说明根据本发明一个实施例的在多媒体寄存器中的各种有符号和无符号紧缩数据类型表示;

[0016] 图3D表示操作编码(操作码)格式的一个实施例;

[0017] 图3E表示可选的操作编码(操作码)格式;

[0018] 图3F表示又一个可选的操作编码格式;

[0019] 图4是根据本发明一个实施例的用来对一个或多个单精度紧缩数据操作数执行至少一个串比较操作的逻辑的框图;

[0020] 图5是根据一个实施例的可以被用来执行至少一个串比较操作的阵列的框图;以及

[0021] 图6示出了可以在本发明一个实施例中执行的操作。

具体实施方式

[0022] 以下说明描述了用来执行处理装置、计算机系统或软件程序内的文本或串元素之间的比较操作的技术的实施例。在以下的说明中,为了提供对本发明的更为全面的理解,阐述了诸如处理器类型、微体系结构状况、事件、使能机制等的许多特定细节。然而,本领域技术人员将会意识到,没有这些特定细节也可以实施本发明。此外,为了避免不必要地使本发明难以理解,一些已知的结构、电路等没有详细示出。

[0023] 虽然以下实施例是参考处理器进行描述的,但是其它实施例也可以应用于其它类型的集成电路和逻辑设备。本发明的相同技术和教导能够容易地应用到其它类型的能够受益于更高的流水线吞吐量和改进的性能的电路或半导体设备。本发明的教导能够应用于执行数据操作的任意处理器或机器。然而,本发明并不局限于执行256位、128位、64位、32位或16位数据操作的处理器或机器,并且能够应用于其中需要操作紧缩数据的任意处理器和机器。

[0024] 在以下的说明中,出于解释的目的,阐述了许多特定细节以便提供对本发明的全面理解。然而,本领域普通技术人员将会意识到,这些特定细节对于实施本发明并不是必需的。在其它实例中,为了避免不必要地使本发明难以理解,没有特别详细地阐述已知的电结构和电路。此外,以下的说明提供了示例,而附图则出于说明的目的而示出了各个示例。然而,这些示例不应当被认为是限制性的,因为它们仅仅用于提供本发明的示例而不是提供

本发明的所有可能的实施方式的排他性列表。

[0025] 虽然以下示例描述了执行单元和逻辑电路环境中的指令处理和分配，但是本发明的其它实施例能够利用软件实现。在一个实施例中，本发明的方法被体现为机器可执行指令。所述指令能够用来使得利用所述指令进行编程的通用或专用处理器执行本发明的步骤。本发明可以被提供为可以包括其上存储有指令的机器或计算机可读介质的计算机程序产品或软件，所述指令可以用来对计算机(或其它电子设备)进行编程而使其执行根据本发明的处理。或者，本发明的步骤可以由包含用于执行这些步骤的硬连线逻辑的特定硬件组件来执行，或者由编程的计算机组件和定制硬件组件的任意组合来执行。这样的软件可以存储在系统中的存储器内。类似地，所述代码可以经由网络或利用其它计算机可读介质来分发。

[0026] 因此，机器可读介质可以包括用于存储或传输机器(例如，计算机)可读形式的信息的任何机制，但并不局限于软盘、光盘、紧凑盘、只读存储器(CD-ROM)以及磁光盘、只读存储器(ROM)、随机访问存储器(RAM)、可擦除可编程只读存储器(EPROM)、电可擦除可编程只读存储器(EEPROM)、磁或光卡、闪存、因特网传输、电、光、声或其它形式的传播信号(例如，载波、红外信号、数字信号等)等等。因此，所述计算机可读介质包括适于存储或传输机器(例如，计算机)可读形式的电子指令或信息的任意类型的媒介/机器可读介质。此外，本发明还可以作为计算机程序产品下载。这样，所述程序可以从远程计算机(例如，服务器)传输到请求计算机(例如，客户端)。所述程序的传输可以利用嵌入在载波或其它传播介质中的电、光、声或其它形式的数据信号经由通信链路(例如，调制解调器、网络连接等)进行。

[0027] 设计可以经历各种阶段，从创建到仿真到制造。表示设计的数据可以以多种方式对设计进行表示。首先，如在仿真中有用的，可以使用硬件描述语言或另外的功能描述语言来表示硬件。此外，可以在设计过程的一些阶段生成具有逻辑和/或晶体管门电路的电路级模型。而且，大多数设计在某个阶段达到了表示硬件模型中各种设备的物理布局的数据级别。在使用传统的半导体制造技术的情况下，表示硬件模型的数据可以是用于指定各种特征在掩模的不同掩模层上是否存在的数据，所述掩模用于生成集成电路。在设计的任意表示中，所述数据可以存储在任意形式的机器可读介质中。被调制或以别的方式生成来传输这样的信息的光或电波、存储器或诸如盘片的磁或光存储装置可以作为所述机器可读介质。任意这些介质可以“携带”或“指示”所述设计或软件信息。当指示或携带代码或设计的电载波被传输，到执行电信号的复制、缓冲或重传的程度时，进行新的复制。这样，通信提供商或网络提供商可以对实现本发明的技术的物体(载波)进行复制。

[0028] 在现代的处理器中，多个不同的执行单元被用来处理和执行多种代码和指令。并不是所有的指令都被平等地创建，一些完成较快而其它则占用很多时钟周期。指令的吞吐量越快，处理器的整体性能就越好。因此，使许多指令尽可能快地执行是有利的。然而，还是存在某些指令具有较大的复杂度并且要求较多的执行时间和处理器资源。例如，存在浮点指令、加载/存储操作、数据移动等。

[0029] 随着越来越多的计算机系统在因特网、文本和多媒体应用中使用，随时间而引入了其它的处理器支持。例如，单指令多数据(SIMD)整数/浮点指令和流SIMD扩展(SSE)是减少执行特定程序任务所需的指令总数的指令，而这又减少了功耗。这些指令能够通过并行操作多个数据元素而加速软件性能。结果是，能够在包括视频、语音和图像/照片处理的广

泛的应用中实现性能提高。SIMD指令在微处理器和类似类型的逻辑电路中的实现方式通常涉及许多问题。此外，SIMD操作的复杂度经常导致需要附加电路，以便正确处理和操作数据。

[0030] 目前，还无法获得对至少两个紧缩操作数(packed operand)的每个数据元素进行比较的SIMD指令。没有例如由一个实施例所执行的SIMD紧缩比较指令的存在，在例如文本解释、压缩/解压缩、处理和操作的应用中可能需要大量指令和数据寄存器来完成同样的结果。在此所公开的实施例可互换地涉及文本或串比较。然而，实施例可以应用于任意信息(文本、数字或其它数据)串。

[0031] 因此，根据本发明实施例的至少一个串比较指令能够减少代码开销和资源需求。本发明的实施例提供了一种将文本解析操作实现为使用SIMD相关硬件的算法的方法。目前，对SIMD寄存器中的数据执行文本解析操作是有些困难和乏味的。一些算法所要求的用来排列算术操作的数据的指令比用来执行那些操作的指令的实际数目更多。通过实施根据本发明实施例的文本比较操作的实施例，实现文本处理所需的指令数目能够大大减少。

[0032] 本发明的实施例涉及用于实施一个或多个串比较操作的指令。文本比较操作通常涉及将来自两个数据串的数据元素进行比较以确定哪些数据元素匹配。可以对一般文本比较算法进行其它变形，这将在这里进行讨论。在一般意义中，应用到表示两个数据串的两个紧缩操作数中的各个数据元素的文本比较操作的一个实施例可以一般性地表示为：

[0033] DEST 1←SRC 1cmp SRC2；

[0034] 对于紧缩SIMD数据操作数而言，该一般性操作能够应用到每个操作数的每个数据元素位置。

[0035] 在以上操作中，“DEST”和“SRC”是用来表示对应数据或操作的目的和源的一般性术语。在一些实施例中，它们可以由寄存器、存储器或具有与所描述的不同的名称或功能的其它存储区域来实现。例如，在一个实施例中，DEST1可以是临时存储寄存器或其它存储区域，而SRC1和SRC2可以是第一和第二目的存储寄存器或其它存储区域，等等。在其它实施例中，两个或更多个SRC和DEST存储区域可以对应于相同存储区域(例如，SIMD寄存器)内的不同数据存储元素。

[0036] 此外，在一个实施例中，串比较操作可以生成一个指示符，以指示源寄存器中的一个的每个元素是否等同于另一个源寄存器的每个元素，并且将所述指示符存储到例如DEST1的寄存器内。在一个实施例中，所述指示符为索引值，而在其它实施例中，所述指示符可以是掩码值。在其它实施例中，所述指示符可以表示其它数据结构或指针。

[0037] 图1A是根据本发明一个实施例的示例性计算机系统的框图，所述计算机系统具有处理器，所述处理器包括用来执行串比较操作的指令的执行单元。根据本发明，例如根据在此所述的实施例，系统100包括诸如处理器102的组件以使用执行单元，所述执行单元包括用来执行处理数据的算法的逻辑。系统100代表基于可从加利福尼亚圣克拉拉的Intel公司获得的PENTIUM® III、PENTIUM® 4、Xeon™、Itanium®、XScale™和/或StrongARM™微处理器的处理系统，当然也可以使用其它系统(包括具有其它微处理器的PC、工程工作站、机顶盒等)。在一个实施例中，样例系统100可以执行可从华盛顿州雷德蒙德的Microsoft公司获得的WINDOWS™操作系统版本，当然也可以使用其它操作系统(例如，UINX和Linux)、嵌入式软件和/或图形用户界面。因此，本发明的实施例并不局限于任何硬件电

路和软件的特定组合。

[0038] 实施例并不局限于计算机系统。本发明的可选实施例能够用在其它设备中，例如手持设备和嵌入式应用。手持设备的一些示例包括蜂窝电话、因特网协议设备、数码相机、个人数字助理(PDA)和手持PC。嵌入式应用可以包括微控制器、数字信号处理器(DSP)、片上系统、网络计算机(NetPC)、机顶盒、网络集线器、广域网(WAN)交换机或对操作数执行串比较操作的任何其它系统。此外，已经实现了一些体系结构使得指令能够同时对若干个数据进行操作，以提高多媒体应用的效率。随着数据类型和数据量的增加，必须增强计算机及其处理器以更为有效的方法操作数据。

[0039] 图1A是根据本发明一个实施例的计算机系统100的框图，计算机系统100形成有处理器102，处理器102包括一个或多个执行单元108，用来执行将来自一个或多个操作数的数据元素进行比较的算法。可以在单处理器桌面或服务器系统的情况下对一个实施例进行描述，但是可选实施例可以包括在多处理器系统中。系统100是集线器体系结构的一个示例。计算机系统100包括用来处理数据信号的处理器102。处理器102可以例如是复杂指令集计算机(CISC)微处理器、精简指令集计算(RISC)微处理器、超长指令字(VLIW)微处理器、实现指令集组合的处理器或比如数字信号处理器的任何其它处理器设备。处理器102耦合至能够在处理器102和系统100中的其它部件之间传送数据信号的处理器总线110。系统100的元件执行它们为本领域技术人员所熟知的常规功能。

[0040] 在一个实施例中，处理器102包括一级(L1)内部高速缓存存储器104。取决于体系结构，处理器102能够具有单个内部高速缓存或多级内部高速缓存。作为选择，在另一个实施例中，高速缓存存储器能够位于处理器102的外部。根据特定实现方式和需要，其它的实施例还可以包括内部和外部高速缓存的组合。寄存器文件106能够将不同类型的数据存储在包括整数寄存器、浮点寄存器、状态寄存器和指令指针寄存器的各种寄存器中。

[0041] 包括用来执行整数和浮点操作的逻辑的执行单元108也位于处理器102中。处理器102还包括存储特定宏指令的微代码的微代码(ucode)ROM。对于该实施例而言，执行单元108包括用来处理紧缩指令集109的逻辑。在一个实施例中，紧缩指令集109包括用于比较多个操作数的元素的紧缩串比较指令。通过在通用处理器102的指令集中包括紧缩指令集109，连同用来执行这些指令的相关电路，许多多媒体应用所使用的操作可以使用通用处理器102中的紧缩数据来执行。因此，通过使用用于对紧缩数据执行操作的处理器数据总线的全带宽，能够加速并更有效地执行许多多媒体应用。这能够消除为了同时对一个数据元素执行一个或多个操作而需要跨处理器数据总线传输较小数据单元的需要。

[0042] 执行单元108的可选实施例还能够用在微控制器、嵌入式处理器、图形设备、DSP和其它类型的逻辑电路中。系统100包括存储器120。存储器120能够是动态随机访问存储器(DRAM)设备、静态随机访问存储器(SRAM)设备、闪存设备或其它存储器设备。存储器120能够存储指令和/或数据，所述指令和/或数据由能够由处理器102执行的数据信号来表示。

[0043] 系统逻辑芯片116耦合至处理器总线110和存储器120。所图示的实施例中的系统逻辑芯片116是存储器控制器集线器(MCH)。处理器102能够经由处理器总线110与MCH 116进行通信。MCH 116为指令和数据存储以及为图形命令、数据和构造的存储提供了到存储器120的高带宽存储器路径118。MCH 116用来引导处理器102、存储器120和系统100中的其它部件之间的数据信号，并且用来桥接处理器总线110、存储器120和系统I/O 122之间的数据

信号。在一些实施例中，系统逻辑芯片116能够提供用于耦合至图形控制器112的图形端口。MCH 116通过存储器接口118耦合至存储器120。图形卡112通过加速图形端口(AGP)互连114耦合至MCH 116。

[0044] 系统100使用专有集线器接口总线122将MCH 116耦合至I/O控制器集线器(ICH)130。ICH 130提供经由本地I/O总线到一些I/O设备的直接连接。所述本地I/O总线是用于将外围设备连接至存储器120、芯片组和处理器102的高速I/O总线。一些示例是音频控制器、固件集线器(闪存BIOS)128、无线收发器126、数据存储设备124、包含用户输入和键盘接口的遗留I/O控制器、诸如通用串行总线(USB)的串行扩展端口以及网络控制器134。数据存储设备124可以包括硬盘驱动器、软盘驱动器、CD-ROM设备、闪存设备或其它大容量存储设备。

[0045] 对于系统的另一个实施例而言，能够在片上系统使用用来执行具有串比较指令的算法的执行单元。片上系统的一个实施例包括处理器和存储器。一个这样的系统的存储器是闪存。所述闪存能够与处理器和其它系统组件位于相同的管芯上。此外，例如存储器控制器或图形控制器之类的其它逻辑模块也能够位于片上系统上。

[0046] 图1B表示实现本发明一个实施例原理的数据处理系统140。本领域技术人员很容易理解，在此所描述的实施例能够用于可替换的处理系统，而不脱离本发明的范围。

[0047] 计算机系统140包括能够执行包括串比较操作的SIMD操作的处理核心159。对于一个实施例而言，处理核心159表示任意类型体系结构的处理单元，包括CISC、RISC或VLIW类型体系结构，但并不局限于此。处理核心159还适于以一种或多种工艺技术来制造，并且通过足够详细地表示在机器可读介质上，可能适于方便所述制造。

[0048] 处理核心159包括执行单元142、一组寄存器文件145和译码器144。处理核心159还包括对于理解本发明不必需的附加电路(未示出)。执行单元142用于执行由处理核心159所接收的指令。除了识别典型的处理器指令之外，执行单元142还能够识别用于对紧缩数据格式执行操作的紧缩指令集143中的指令。紧缩指令集143包括用于支持串比较操作的指令，并且还包括其它紧缩指令。执行单元142通过内部总线耦合至寄存器文件145。寄存器文件145表示存储核心159上用于存储包括数据的信息的存储区域。如之前所提及的，应当理解，用于存储紧缩数据的存储区域并不是关键的。执行单元142耦合至译码器144。译码器144用于将处理核心159所接收的指令译码为控制信号和/或微代码进入点。响应于这些控制信号和/或微代码进入点，执行单元142执行适当的操作。

[0049] 处理核心159与总线141相耦合以与各种其它系统设备进行通信，例如，所述其它系统设备可以包括同步动态随机访问存储器(SDRAM)控制器146、静态随机访问存储器(SRAM)控制器147、突发闪存(burst flash)接口148、个人计算机存储卡国际协会(PCMCIA)/紧密闪存(CF)卡控制器149、液晶显示(LCD)控制器150、直接存储器访问(DMA)控制器151和可选的总线主接口152，但并不局限于此。在一个实施例中，数据处理系统140还可以包括用于经由I/O总线153与各种I/O设备进行通信的I/O桥154。例如，这样的I/O设备可以包括通用异步接收器/发射器(UART)155、通用串行总线(USB)156、蓝牙无线UART 157和I/O扩展接口158，但并不局限于此。

[0050] 数据处理系统140的一个实施例提供移动、网络和/或无线通信，并且处理核心159能够执行包括串比较操作的SIMD操作。处理核心159可以利用包括诸如Walsh-Hadamard变换、快速傅立叶变换(FFT)、离散余弦变换(DCT)及其各自的反变换之类的离散变换的各种

音频、视频、成像和通信算法;诸如颜色空间变换、视频编码运动估计或视频解码运动补偿之类的压缩/解压缩技术;以及诸如脉冲编码调制(PCM)之类的调制/解调(MODEM)功能,来进行编程。

[0051] 图1C表示能够执行SIMD串比较操作的数据处理系统的又一个可选实施例。根据一个可选实施例,数据处理系统160可以包括主处理器166、SIMD协处理器161、高速缓存存储器167和输入/输出系统168。输入/输出系统168可选地耦合至无线接口169。SIMD协处理器161能够执行包括串比较操作的SIMD操作。处理核心170可适于以一种或多种工艺技术来制造,并且通过足够详细地表示在机器可读介质上,可以适于方便包括处理核心170的数据处理系统160的全部或部分制造。

[0052] 对于一个实施例而言,SIMD协处理器161包括执行单元162和一组寄存器文件164。主处理器165的一个实施例包括译码器165,用来识别包括由执行单元162执行的SIMD串比较指令的指令集163的指令。对于可选实施例而言,SIMD协处理器161还包括译码器165B的至少一部分,用来译码指令集163的指令。处理核心170还包括对于理解本发明实施例来说不需要的附加电路(未示出)。

[0053] 在操作时,主处理器166执行一个数据处理指令流,其控制包括与高速缓存存储器167和输入/输出系统168的交互的一般类型的数据处理操作。SIMD协处理器指令嵌入在所述数据处理指令流中。主处理器166的译码器165将这些SIMD协处理器指令识别为应当由所附的SIMD协处理器161执行的类型。因此,主处理器166将这些SIMD协处理器指令(或表示SIMD协处理器指令的控制信号)发射到协处理器总线166上,任意所附的SIMD协处理器从协处理器总线166上接收这些SIMD协处理器指令。在这种情形下,SIMD协处理器161将接受并执行任何接收的针对其的SIMD协处理器指令。

[0054] 可以经由无线接口169接收数据以供SIMD协处理器指令处理。对于一个示例,可以以数字信号的形式接收语音通信,所述数字信号可以由SIMD协处理器指令处理以再次生成表示语音通信的数字音频采样。对于另一个示例,可以以数字比特流的形式接收压缩的音频和/或视频,所述数字比特流可以由SIMD协处理器指令处理以再次生成数字音频采样和/或运动视频帧。对于处理核心170的一个示例而言,主处理器166和SIMD协处理器161集成为包括执行单元162、一组寄存器文件164和译码器165的单个处理核心170,以识别包括SIMD串比较指令的指令集163的指令。

[0055] 图2是根据本发明一个实施例的处理器200的微体系结构的框图,其中处理器200包括用来执行串比较指令的逻辑电路。对于串比较指令的一个实施例而言,所述指令能够将第一操作数的每个数据元素与第二操作数的每个数据元素进行比较,并且存储指示每个比较是否匹配的指示符。在一些实施例中,串比较指令能够被实现为对大小为字节、字、双字、四字等以及数据类型为诸如整数和浮点数据类型的数据元素进行操作。在一个实施例中,顺序前端201是处理器200的一部分,其读取待执行的宏指令并且准备它们以备稍后在处理器流水线中使用。前端201可以包括若干个单元。在一个实施例中,指令预取器226从存储器读取宏指令并且将它们提供至指令译码器228,而指令译码器228又将它们译码为机器能够执行的称为微指令或微操作(也称为micro op或uops)的原语。在一个实施例中,追踪缓存(trace cache)230取得译码的微指令并且将它们汇编到微指令队列234中的程序排序的序列或踪迹中以供执行。当追踪缓存230遇到复杂的宏指令时,微代码ROM 232提供完成

该操作所需的微指令。

[0056] 许多宏指令被转换为单个微操作，而其它的则需要若干个微指令来完成整个操作。在一个实施例中，如果需要多于四个的微操作来完成宏指令，则译码器228访问微代码ROM 232来执行所述宏指令。对于一个实施例而言，紧缩串比较指令能够被译码为在指令译码器228处进行处理的少数微操作。在另一个实施例中，若完成操作需要多个微操作，则用于紧缩串比较算法的指令能够存储在微代码ROM 232内。追踪缓存230参照进入点可编程逻辑阵列(PLA)以确定用于读取微代码ROM 232中的串比较算法的微代码序列的正确微指令指针。在微代码ROM 232完成对当前宏指令的微操作进行排序之后，机器的前端201继续从追踪缓存230读取微指令。

[0057] 一些SIMD和其它多媒体类型的指令被认为是复杂的指令。大多数浮点相关指令也是复杂指令。这样，当指令译码器228遇到复杂的宏指令时，在适当的位置访问微代码ROM 232以获取该宏指令的微代码序列。执行该宏指令所需的各个微操作被传输至乱序执行引擎203，以在适当的整数和浮点执行单元进行执行。

[0058] 乱序执行引擎203是准备微指令以供执行的地方。乱序执行逻辑具有多个缓冲器，用于在微指令沿流水线下行并得以调度执行的过程中，使微指令流平滑并对其重新排序以优化性能。分配器逻辑分配每个微指令执行所需要的机器缓冲器和资源。寄存器重命名逻辑将逻辑寄存器重命名至寄存器文件的项。在指令调度器：存储器调度器、快速调度器202、慢速/一般浮点调度器204和简单浮点调度器206之前，所述分配器还为两个微指令队列之一中的每个微指令分配项，所述两个微指令队列一个用于存储器操作，一个用于非存储器操作。微指令调度器202、204、206基于它们所依赖的输入寄存器操作数源的准备就绪以及微指令完成它们的操作所需的执行资源的可用性，来确定微指令何时准备好执行。该实施例的快速调度器202能够在主时钟周期的每半个周期进行调度，而其它调度器在每个主处理器时钟周期仅调度一次。这些调度器对用来调度供执行的微指令的分派端口进行仲裁。

[0059] 寄存器文件208、210位于调度器202、204、206和执行模块211中的执行单元212、214、216、218、220、222、224之间。存在分别用于整数和浮点操作的分离寄存器文件208、210。在其它实施例中，所述整数和浮点寄存器可以位于相同的寄存器文件中。该实施例的每个寄存器文件208、210还包括旁路网络，所述旁路网络能够绕过还没有被写入寄存器文件的刚完成的结果或将其转发至新的从属微指令。整数寄存器文件208和浮点寄存器文件210还能够互相传输数据。对于一个实施例而言，整数寄存器文件208被分为两个分离的寄存器文件，一个寄存器文件用于数据的低序32位，而第二寄存器文件用于数据的高序32位。因为浮点指令典型地具有64至128位宽度的操作数，所以一个实施例的浮点寄存器文件210具有128位宽的项。

[0060] 执行模块211包含实际执行指令的执行单元212、214、216、218、220、222、224。该部分包括存储微指令需要执行的整数和浮点数据操作数值的寄存器文件208、210。该实施例的处理器200由多个执行单元构成：地址生成单元(AGU)212、AGU 214、快速ALU 216、快速ALU 218、慢速ALU 220、浮点ALU 222、浮点移动单元224。对于该实施例而言，浮点执行模块222、224执行浮点、MMX、SIMD和SSE操作。该实施例的浮点ALU 222包括64位乘64位的浮点除法器，用来执行除法、平方根和余数微操作。对于本发明的实施例而言，涉及浮点数值的任何动作均利用浮点硬件进行。例如，整数格式和浮点格式之间的变换涉及浮点寄存器文件。

相似地，浮点除法操作在浮点除法器中进行。另一方面，非浮点数和整数类型利用整数硬件资源进行处理。简单的、非常频繁的ALU操作转至高速ALU执行单元216、218。该实施例的快速ALU 216、218能够以半个时钟周期的有效延迟执行快速操作。对于一个实施例而言，大多数复杂的整数操作转至慢速ALU 220，原因在于慢速ALU 220包括用于长延迟操作类型的整数执行硬件，例如乘法器、移位、标志逻辑和分支处理。存储器加载/存储操作由AGU 212、214执行。对于该实施例而言，在对64位数据操作数执行整数操作的情况下对ALU 216、218、220进行描述。在可选实施例中，ALU 216、218、220能够被实现为支持包括16、32、128、256等的多种数据位。类似地，浮点单元222、224能够被实现为支持具有各种位宽度的操作数范围。对于一个实施例而言，浮点单元222、224能够结合SIMD和多媒体指令对128位宽的紧缩数据操作数进行操作。

[0061] 在该实施例中，微指令调度器202、204、206在父加载执行完成之前分派附属操作。由于微指令在处理器200中预测地调度和执行，所以处理器200还包括用来处理存储器缺失的逻辑。如果数据加载在数据高速缓存中出现缺失，则可能在留下了具有临时错误数据的调度器的流水线中存在运行中的从属操作。重放机制追踪并重新执行使用错误数据的指令。仅需要重放从属操作，而允许独立操作完成。处理器的一个实施例的所述调度器和重放机制还被设计为捕捉用于串比较操作的指令序列。

[0062] 在此使用的术语“寄存器”是指被用作识别操作数的部分宏指令的板上处理器存储单元。换句话说，在此提到的寄存器是从处理器外部(从编程者的角度)可见的那些寄存器。然而，实施例的寄存器不应当在含义上局限于用于特定电路类型。而是，实施例的寄存器仅需要能够存储和提供数据，并且执行在此所描述的功能。在此所描述的寄存器可以由使用任意数量的不同技术的处理器内的电路来实现，例如专用物理寄存器、使用寄存器重命名的动态分配物理寄存器、专用和动态分配物理寄存器的组合等。在一个实施例中，整数寄存器存储32位整数数据。一个实施例的寄存器文件还包含8个用于紧缩数据的多媒体SIMD寄存器。对于以下的讨论而言，寄存器被理解为被设计成保存紧缩数据的数据寄存器，例如利用加利福尼亚圣克拉拉的Intel公司的MMX技术致能的微处理器中的64位宽度的MMX™寄存器(在一些实例中也被称作‘mm’寄存器)。这些可以是整数或浮点形式的MMX寄存器能够利用伴随SIMD和SSE指令的紧缩数据元素来进行操作。类似地，与SSE2、SSE3、SSE4或其后的(通常称作“SSEx”)技术有关的128位宽度的XMM寄存器也能够被用来保存这样的紧缩数据操作数。在该实施例中，在存储紧缩数据和整数数据时，所述寄存器不需要在两种数据类型之间进行区分。

[0063] 在以下附图的示例中，描述了多个数据操作数。图3A表示根据本发明一个实施例的多媒体寄存器中的各种紧缩数据类型表示。图3A表示了128位宽度的操作数的紧缩字节310、紧缩字320和紧缩双字(dword)330的数据类型。该示例的紧缩字节格式310为128位长度，并且包含16个紧缩字节数据元素。字节在此定义为8位数据。每个字节数据元素的信息如下存储：字节0存储在位7至位0，字节1存储在位15至位8，字节2存储在位23至位16，以及最后的字节15存储在位127至位120。因此，寄存器中的所有可用位都被使用。该存储安排增加了处理器的存储效率。而且，通过访问16个数据元素，现在能够并行地对16个数据元素执行一个操作。

[0064] 通常，数据元素是与相同长度的其它数据元素一同存储在单个寄存器或存储器单

元中的单条数据。在与SSEx技术相关的紧缩数据序列中，存储在XMM寄存器中的数据元素的数目是128位除以单个数据元素的位长度。类似地，在与MMX和SSE技术相关的紧缩数据序列中，存储在MMX寄存器中的数据元素数目是64位除以单个数据元素的位长度。虽然图3A中所示的数据类型为128位长度，但本发明的实施例还能够操作64位宽或其它大小的操作数。该示例中的紧缩字格式320为128位长度，并且包含8个紧缩字数据元素。每个紧缩字包含16位信息。图3A的紧缩双字格式330为128位长度，并且包含4个紧缩双字数据元素。每个紧缩双字数据元素包含32位信息。紧缩四字为128位长度，并且包含2个紧缩四字数据元素。

[0065] 图3B表示可选的寄存器内数据存储格式。每个紧缩数据能够包括不止一个独立数据元素。表示了3种紧缩数据格式：紧缩半倍341、紧缩单倍342和紧缩双倍343。紧缩半倍341、紧缩单倍342和紧缩双倍343的一个实施例包含定点数据元素。对于可选实施例而言，紧缩半倍(packed half)341、紧缩单倍(packed single)342和紧缩双倍(packed double)343中的一个或多个可以包含浮点数据元素。紧缩半倍341的一个可选实施例为包含8个16位数据元素的128位长度。紧缩单倍342的一个实施例为128位长度并且包含4个32位数据元素。紧缩双倍343的一个实施例为128位长度并且包含2个64位数据元素。可以理解，这样的紧缩数据格式可以进一步扩展到其它寄存器长度，例如扩展到96位、160位、192位、224位、256位或更多。

[0066] 图3C表示根据本发明一个实施例的多媒体寄存器中的各种有符号和无符号紧缩数据类型表示。无符号紧缩字节表示344示出了无符号紧缩字节在SIMD寄存器中的存储。每个字节数据元素的信息存储为：字节0存储在位7至位0，字节1存储在位15至位8，字节2存储在位23至位16，以及最后的字节15存储在位127至位120。因此，寄存器中的所有可用位都被使用。这样的存储安排能够增加处理器的存储效率。而且，通过访问16个数据元素，现在能够以并行的方式对16个数据元素执行一个操作。有符号紧缩字节表示345表示有符号紧缩字节的存储。注意，每个字节数据元素的第8位是符号指示符。无符号紧缩字表示346示出了字7至字0如何存储在SIMD寄存器中。有符号紧缩字表示347与无符号紧缩字寄存器内表示346相似。注意，每个字数据元素的第16位是符号指示符。无符号紧缩双字表示348示出了如何存储双字数据元素。有符号紧缩双字表示349与无符号紧缩双字寄存器内表示348相似。注意，必需的符号位是每个双字数据元素的第32位。在一个实施例中，一个或多个操作数可以是恒定的，从而在它们所关联的一个或多个指令的实例之间不发生改变。

[0067] 图3D是具有32位或更多位的操作编码(操作码)格式360以及与在可在万维网(www)intel.com/design/litcentr上从加利福尼亚圣克拉拉的Intel公司获得的“IA-32英特尔体系结构软件开发者手册，第二卷：指令集参考”(IA-32Intel Architecture Software Developer’s Manual Volume 2:Instruction Set Reference)中所描述的操作码格式类型相对应的寄存器/存储器操作数寻址模式的一个实施例的描述。在一个实施例中，可以通过一个或多个字段361和362对串比较操作进行编码。每个指令可以标识多达两个操作数的位置，包括多达两个源操作数标识符364和365。对于串比较指令的一个实施例而言，目标操作数标识符366与源操作数标识符364相同，而在其它实施例中它们不同。对于可选实施例而言，目标操作数标识符366与源操作数标识符365相同，而在其它实施例中它们不同。在串比较指令的一个实施例中，由源操作数标识符364和365标识的源操作数中的一个被串比较操作的结果所覆盖，而在其它实施例中，标识符364对应于源寄存器元素，并

且标识符365对应于目标寄存器元素。对于串比较指令的一个实施例而言，操作数标识符364和365可以被用来标识32位或64位源和目标操作数。

[0068] 图3E是具有40位或更多位的另一个可选操作编码(操作码)格式370的描述。操作码格式370与操作码格式360相对应，并且包括任选的前缀字节378。串比较操作的类型可以由字段378、371和372中的一个或多个进行编码。每个指令中多达两个操作数的位置可以由源操作数标识符374和375以及由前缀字节378来标识。对于串比较指令的一个实施例而言，前缀字节378可以被用来标识32位、64位或128位的源和目标操作数。对于串比较指令的一个实施例而言，目标操作数标识符376与源操作数标识符374相同，而在其它实施例中它们不同。对于可选实施例而言，目标操作数标识符376与源操作数标识符375相同，而在其它实施例中它们不同。在一个实施例中，所述串比较操作将由操作数标识符374和375所标识的操作数中的一个的每个元素与由操作数标识符374和375所标识的另一个操作数的每个元素进行比较，并且被所述串比较操作的结果所覆盖，而在其它实施例中，由标识符374和375所标识的操作数的串比较被写入另一个寄存器的另一个数据元素中。操作码格式360和370允许寄存器到寄存器、存储器到寄存器、寄存器经存储器、寄存器经寄存器、寄存器经立即、寄存器到存储器寻址，所述寻址部分地由MOD字段363和373以及任选的基址加比例变址(scale-index-base)和置换字节所指定。

[0069] 接下来转向图3F，在一些可选实施例中，64位单指令多数据(SIMD)算术操作可以通过协处理器数据处理(CDP)指令来执行。操作编码(操作码)格式380描述了一个这样的CDP指令，其具有CDP操作码字段382和389。对于串比较操作的可选实施例而言，CDP指令的类型可以由一个或多个字段383、384、387和388进行编码。每个指令可以标识多达三个操作数的位置，包括多达两个源操作数标识符385和390以及一个目标操作数标识符386。协处理器的一个实施例能够对8、16、32和64位数值进行操作。对于一个实施例而言，对整数数据元素执行串比较操作。在一些实施例中，可以使用条件字段381来有条件地执行串比较指令。对于一些串比较指令而言，源数据大小可以由字段383进行编码。在串比较指令的一些实施例中，能够在SIMD字段上进行零(Z)、负数(N)、进位(C)和溢出(V)检测。对于一些指令而言，饱和的类型可以由字段384进行编码。

[0070] 在一个实施例中，字段，或称“标志”可以用来指示串比较操作的结果何时非零。在一些实施例中，可以使用其它字段，诸如用来指示源元素何时无效的标志，以及用来指示串比较操作的结果的最低或最高有效位的标志。

[0071] 图4是根据本发明的用来对紧缩数据操作数执行串比较操作的逻辑的一个实施例的框图。本发明的实施例能够被实现为与比如如上所述的各种操作数类型进行操作。对于一种实施方式而言，根据本发明的串比较操作被实现为一组用来对特定数据类型进行操作的指令。例如，提供紧缩串比较指令来执行32位数据类型的比较，所述数据类型包括整数和浮点。类似地，提供紧缩串比较指令来执行64位数据类型的比较，所述数据类型包括整数和浮点。接下来的讨论和以下示例用来表示用于比较数据元素的比较指令的操作，而不关心元素表示什么。为了简要起见，一些示例将示出一个或多个串比较指令的操作，其中数据元素表示文本字。

[0072] 在一个实施例中，串比较指令将第一数据操作数DATA A 410的每个元素与第二数据操作数DATA B 420的每个元素进行比较，并且将每个比较的结果存储在RESULTANT 440

寄存器中。对于以下的讨论而言，DATA A、DATA B和RESULTANT通常被称作寄存器，但是并不局限于此，并且还包括寄存器、寄存器文件和存储器单元。在一个实施例中，文本串比较指令(例如，“PCMPxSTRy”)被译码为一个微操作。在可选实施例中，每个指令可以被译码为不同数量用来对数据操作数执行文本串比较操作的微操作。对于该示例而言，操作数410、420是存储在具有字宽度数据元素的源寄存器/存储器中的128位宽度的信息。在一个实施例中，操作数410、420保存在128位长度的SIMD寄存器中，例如128位SSEx XMM寄存器。对于一个实施例而言，RESULTANT 440也是XMM数据寄存器。在其它实施例中，RESULTANT 440可以是不同类型的寄存器，例如扩展寄存器(例如，“EAX”)或存储器单元。取决于特定的实施方式，所述操作数和寄存器可以是例如32、64和256位的其它长度，并且具有字节、双字或四字大小的数据元素。虽然该示例的数据元素是字大小，但是相同的概念可以扩展到字节和双字大小的元素。在一个实施例中，其中数据操作数为64位宽度，则使用MMX寄存器来取代XMM寄存器。

[0073] 在一个实施例中，第一操作数410由一组8个数据元素构成：A7、A6、A5、A4、A3、A2、A1和A0。第一和第二操作数的元素之间的每个比较可以对应于结果440中的数据元素位置。在一个实施例中，第二操作数420由另一组8个数据段构成：B7、B6、B5、B4、B3、B2、B1和B0。这里的数据段长度相等，并且每个包含单字(16位)数据。然而，数据元素和数据元素位置能够具有不同于字的其它粒度。要是每个数据元素为字节(8位)、双字(32位)或四字(64位)，则128位操作数将分别具有16个字节宽度、4个双字宽度或2个四字宽度的数据元素。本发明的实施例并不局限于特定长度的数据操作数或数据段，并且对于每种实施方式能够采用适当的大小。

[0074] 操作数410、420能够位于寄存器或存储器单元或寄存器文件或以上的组合中。数据操作数410、420连同文本串比较指令一起被发送至处理器中的执行单元的串比较逻辑430。在一个实施例中，在所述指令到达执行单元之前，所述指令可能已在处理器流水线中事先被译码。因此，所述串比较指令可以是微操作(uop)或一些其他译码格式的形式。对于一个实施例而言，在串比较逻辑430接收到两个数据操作数410、420。在一个实施例中，文本串比较逻辑生成两个数据操作数的元素是否相等的指示。在一个实施例中，仅比较每个操作数的有效元素，这可以由每个操作数中的每个元素的另一个寄存器或存储器单元进行指示。在一个实施例中，将操作数410的每个元素与操作数420的每个元素进行比较，这样生成的比较结果的数目等于操作数410的元素数乘以操作数420的元素数。例如，在每个操作数410和420均为32位数值的情况下，结果寄存器440将存储由串比较逻辑430所执行的文本比较操作的多达32×32个结果指示符。在一个实施例中，来自第一和第二操作数的数据元素是单精度的(例如，32位)，而在其它实施例中，来自第一和第二操作数的数据元素是双精度的(例如，64位)。在另一些其它实施例中，第一和第二操作数可以包括任意大小的整数元素，包括8、16和32位。

[0075] 对于一个实施例而言，所有数据位置的数据元素都被并行处理。在另一个实施例中，数据元素位置的特定部分能够同时在一起处理。在一个实施例中，结果440由操作数410和420中存储的每个数据元素之间进行的比较的多个结果构成。特别是，在一个实施例中，所述结果可以存储的比较结果数目等于操作数410或420之一中的数据元素数目的平方。

[0076] 在一个实施例中，所述结果可以仅存储在操作数410和420的有效数据元素之间进

行比较的比较结果。在一个实施例中，每个操作数的数据元素可以明确或隐含地被指示为有效。例如，在一个实施例中，每个操作数数据元素对应于一个有效性指示符，诸如存储在另一个存储区域(如有效寄存器)内的有效位。在一个实施例中，两个操作数的每个元素的有效位可以存储在同一个有效寄存器中，而在其它实施例中，一个操作数的有效位可以存储在第一有效寄存器中而另一操作数的有效位可以存储在第二有效寄存器中。在对操作数数据元素进行比较或者结合之前，确定两个数据元素是否都有效(例如通过检查对应的有效位)，从而仅在有效数据元素之间进行比较。

[0077] 在一个实施例中，可以通过使用一个或两个操作数内存储的空或“零”字段来隐含地指示每个操作数中的有效数据元素。例如，在一个实施例中，可以在元素中存储空字节(或其它大小)来指示比所述空字节更高位的所有数据元素均无效，而比所述空字节更低位的所有数据元素均有效从而应当将比所述空字节更低位的所有数据元素与另一操作数的对应有效数据元素进行比较。此外，在一个实施例中，一个操作数的有效数据元素可以被明确地指示(如之前所述)，而另一操作数的有效数据元素可以使用空字段来隐含地指示。在一个实施例中，由与一个或多个源操作数内的有效数据元素或子元素的数目相对应的计数来指示有效数据元素。

[0078] 不管指示每个操作数的有效数据元素的方法，在至少一个实施例中，仅对每个操作数的被指示为有效的数据元素进行比较。在各个实施例中，可以用多种方法执行仅对有效数据元素进行的比较。出于提供全面和可理解的说明的目的，以下对在两个文本串操作数之间仅比较有效数据元素的方法给出最佳概念化表达。然而，以下的描述仅仅是如何对仅比较文本串操作数的有效数据元素进行最佳的概念化表示或实施的一个示例。在其它实施例中，其它的概念化表示或方法可以用来说明如何对有效数据元素进行比较。

[0079] 在一个实施例中，不管操作数中的有效数据元素的数目是被明确指示(例如，经由有效寄存器中的有效位，或通过从最低有效位开始的有效字节/字数目的计数)或隐含表示(例如，经由操作数自身内的空字符)，仅对每个操作数的有效数据元素互相进行比较。在一个实施例中，有效性指示符和将要比较的数据元素的聚合可在图5中概念化表示。

[0080] 参考图5，在一个实施例中，阵列501和505包含分别指示第一操作数和第二操作数的每个数据元素是否有效的项。例如，在以上说明中，阵列501可以在第一操作数包含对应的有效数据元素的每个阵列元素中包含“1”。类似地，阵列505可以在第二操作数包含对应的有效数据元素的每个阵列元素中包含“1”。在一个实施例中，对于两个单独的操作数中的每一个中存在的每个有效元素，阵列501和505可以从阵列元素0开始包含若干个1。例如，在一个实施例中，如果第一操作数包含4个有效元素，则阵列501可以仅在开始的四个阵列元素中包含1，而阵列501的所有其它阵列元素可以是0。

[0081] 在一个实施例中，阵列501和505的大小均为16个元素，用来表示两个128位操作数的16个数据元素，所述数据元素每个大小为8位(1字节)。在其它实施例中，其中操作数的数据元素的大小为16位(1个字)，阵列501和505可以仅包含8个元素。在其它实施例中，根据其对应的操作数的大小，阵列501和505可以更大或更小。

[0082] 在一个实施例中，第一操作数的每个数据元素与第二操作数的每个数据元素进行比较，其结果可以由i×j阵列510来表示。例如，表示文本串的第一操作数的第一数据元素例如可以与表示另一个文本串的另一个操作数的每个数据元素进行比较，并且对应于第一

操作数的第一数据元素和第二操作数的每个数据元素之间的匹配，在阵列510的第一行内的每个阵列元素中存储“1”。可以对第一操作数中的每个数据元素重复该步骤，直至阵列510完成。

[0083] 在一个实施例中，可以生成具有i×j个项的第二阵列515来存储是否仅仅有效操作数数据元素相等的指示。例如，在一个实施例中，阵列510的顶行511的每个项可以与对应的有效阵列元素506以及有效阵列元素502进行逻辑“与”，并且将结果置于阵列515的对应元素516中。所述的“与”操作可以在阵列510的每个元素与有效阵列501和505中的对应元素之间完成，并且将结果置于阵列520的对应元素中。

[0084] 在一个实施例中，结果阵列520可以指示一个操作数中是否存在与另一操作数中的一个或多个数据元素有关的数据元素。例如，结果阵列520可以存储用来指示是否有任何数据元素处于由另一操作数中的数据元素所定义的一组范围之内的位，这是通过将来自阵列515的元素对进行“与”操作并且将所述“与”操作的所有结果进行“或”操作而进行的。

[0085] 图5还示出了结果阵列520，其用来存储与至少两个紧缩操作数的数据元素之间的比较相关的各种指示符。例如，结果阵列520可以存储用来指示在两个操作数之间是否存在任何相等数据元素的位，这是通过将阵列515的对应元素进行“或”操作而进行的。例如，如果阵列515的任意阵列元素包含“1”，即指示在操作数的有效数据元素之间存在匹配，那么这会反映在结果阵列520中，其元素还可以被进行“或”操作来确定多个操作数的是否有任意有效数据元素相等。

[0086] 在一个实施例中，通过检测结果阵列520内的相邻的“1”值而在结果阵列520中检测两个操作数的数据元素之间的有效匹配的连续串。在一个实施例中，这可以通过同时对两个连续的结果阵列元素进行“与”操作并且将一个“与”操作的结果与下一个结果项进行“与”操作直至检测到“0”为止来完成。在其它实施例中，可以使用其它逻辑来检测两个紧缩操作数内的数据元素的有效匹配的范围。

[0087] 在一个实施例中，结果阵列520可以通过例如在相应的结果阵列项中返回“1”来指示两个操作数的每个数据元素是否匹配。为了确定所有的项是否相等，可以对结果阵列的项执行异或(XOR)操作。在其它实施例中，可以使用其它逻辑来确定两个操作数的每个有效数据元素是否相等。

[0088] 在一个实施例中，可以通过将测试串与其它串的相等大小的部分进行比较并且在结果阵列内指示所述测试串与其它串的所述部分之间的匹配来检测数据元素串在另一个数据元素串内的某处是否出现。例如，在一个实施例中，与第一操作数中的三个数据元素相对应的3个字符的测试串与第二串的第一组三个数据元素进行比较。如果检测到匹配，则可以通过在对应于匹配的3个结果项中的一个或多个组中存储一个或多个“1”来在结果阵列中反映所述匹配。所述测试串可以接着与另一操作数的接下来三个数据元素进行比较，或者可以将两个之前的操作数数据元素和新的第三数据元素与所述测试串进行比较，从而测试串随着比较而沿其它操作数进行‘滑动’。

[0089] 在一个实施例中，根据应用程序，可以将结果阵列的项进行翻转或求负。在其它实施例中，仅有一些结果项可以被求负，例如仅有被指示为与两个操作数的数据元素之间的有效匹配相对应的那些项。在其它实施例中，可以对结果阵列520的结果项执行其它的操作。例如，在一些实施例中，结果阵列520可以表示为掩码值，而在其它实施例中，所述结果

阵列可以利用可被存储在例如寄存器的存储单元中的索引值来进行表示。在一个实施例中,索引可以由所述结果阵列的一组最高有效位表示,而在其它实施例中,所述索引可以由所述阵列的一组最低有效位表示。在一个实施例中,所述索引可以由所设置的相对于最低或最高有效位的偏移值表示。在一个实施例中,所述掩码可以是0扩展的,而在其它实施例中,其可以是字节/字掩码或其它一些粒度。

[0090] 在各个实施例中,将两个或更多SIMD操作数的每个元素进行比较时的上述每种变化可以作为单独的各个指令来执行。在其它实施例中,可以通过改变单个指令的属性来执行上述变化,例如与指令相关的立即字段。图6示出了由一个或多个指令执行来将两个或更多SIMD操作数的每个数据元素进行比较的各种操作。在一个实施例中,由图6中的操作所比较的每个操作数每个均表示一个文本串。在其它实施例中,所述操作数可以表示其它一些信息或数据。

[0091] 参考图6,在操作610,可以将第一SIMD操作数601和第二SIMD操作数605的每个元素互相比较。在一个实施例中,一个操作数可以存储在例如XMM寄存器的寄存器中,而另一个操作数可以存储在另一个XMM寄存器或存储器中。在一个实施例中,可以由与执行图6所示的操作的指令相对应的立即字段来控制比较的类型。例如,在一个实施例中,立即字段的两位(例如,IMM8[1:0])可以用来指示将要进行比较的数据元素是否为有符号字节、有符号字、无符号字节或无符号字。在一个实施例中,所述比较的结果可以生成i×j阵列(例如,BoolRes[i,j])或i×j阵列的一些部分。

[0092] 并行地,在操作613,找到由操作数601和605表示的每个串的末端,并且可以确定操作数601和605的每个元素的有效性。在一个实施例中,通过在寄存器或存储器单元内设置对应的一个或多个位而明确指示操作数601和605的每个元素的有效性。在一个实施例中,所述一个或多个位可以对应于从操作数601和605的最低有效位位置开始的连续有效的数据元素(例如,字节)的数目。例如,根据操作数的大小,可以使用诸如EAX或RAX寄存器之类的寄存器来存储指示第一操作数的每个数据元素的有效性的位。类似地,根据操作数的大小,可以使用诸如EDX或RDX之类的寄存器来存储指示第二操作数的每个数据元素的有效性的位。在另一个实施例中,可以通过该公开内容中已讨论的方式隐含地指示操作数601和605的每个元素的有效性。

[0093] 在一个实施例中,在操作615,可以通过聚合函数(aggregation function)合并比较和有效性信息,以生成比较两个操作数的元素的一些结果。在一个实施例中,所述聚合函数由与用来执行两个操作数的元素比较的指令相关联的立即字段确定。例如,在一个实施例中,所述立即字段可以指示所述比较是否用来指示两个操作数是否有任意数据元素相等、两个操作数中是否有任意范围(连续或非连续)的数据元素相等、两个操作数的每个数据元素是否相等或者多个操作数是否共享至少一些数据元素的相等排序。

[0094] 在一个实施例中,在操作620,可以对所述聚合函数的结果(例如,存储在IntRes1中)进行求负。在一个实施例中,立即字段的位(例如,IMM8[6:5])可以控制要对聚合函数结果执行的求负函数的类型。例如,立即字段可以指示聚合结果根本不会被求负、聚合函数的所有结果都要被求负或者只有与操作数的有效元素相对应的聚合结果才被求负。在一个实施例中,所述求负操作的结果可以存储在阵列(例如,IntRes2阵列)中。

[0095] 在一个实施例中,在操作625和630,所述求负操作所生成的结果阵列可以分别被

变换为索引或掩码值。如果所述求负操作的结果被转换为索引，则立即字段的位(例如，IMM8[6])可以控制比较结果的(多个)最高有效位或(多个)最低有效位是否被编码为索引，其结果可以存储到寄存器(例如，ECX或RCX)中。在一个实施例中，如果要利用掩码值来表示所述求负操作的结果，则立即字段的位(例如，IMM8[6])可以用来控制所述掩码是0扩展的还是被扩展到字节(或字)掩码。

[0096] 因此，已经公开了用于执行串比较操作的技术。虽然已经描述并在附图中示出了特定的示例性实施例，但是应当理解，这些实施例仅仅是说明性的，并不在宽泛的发明上进行限制，并且由于本领域普通技术人员在学习了该公开内容后可以进行各种其它的修改，所以本发明并不局限于所示出和描述的特定结构和配置。在发展迅速并且不能够轻易预见到进一步的发展的诸如本发明的技术领域，通过技术进步能够方便地对所公开的实施例在配置和细节上容易地进行修改，而不会背离本公开内容的原理或所附权利要求的范围。

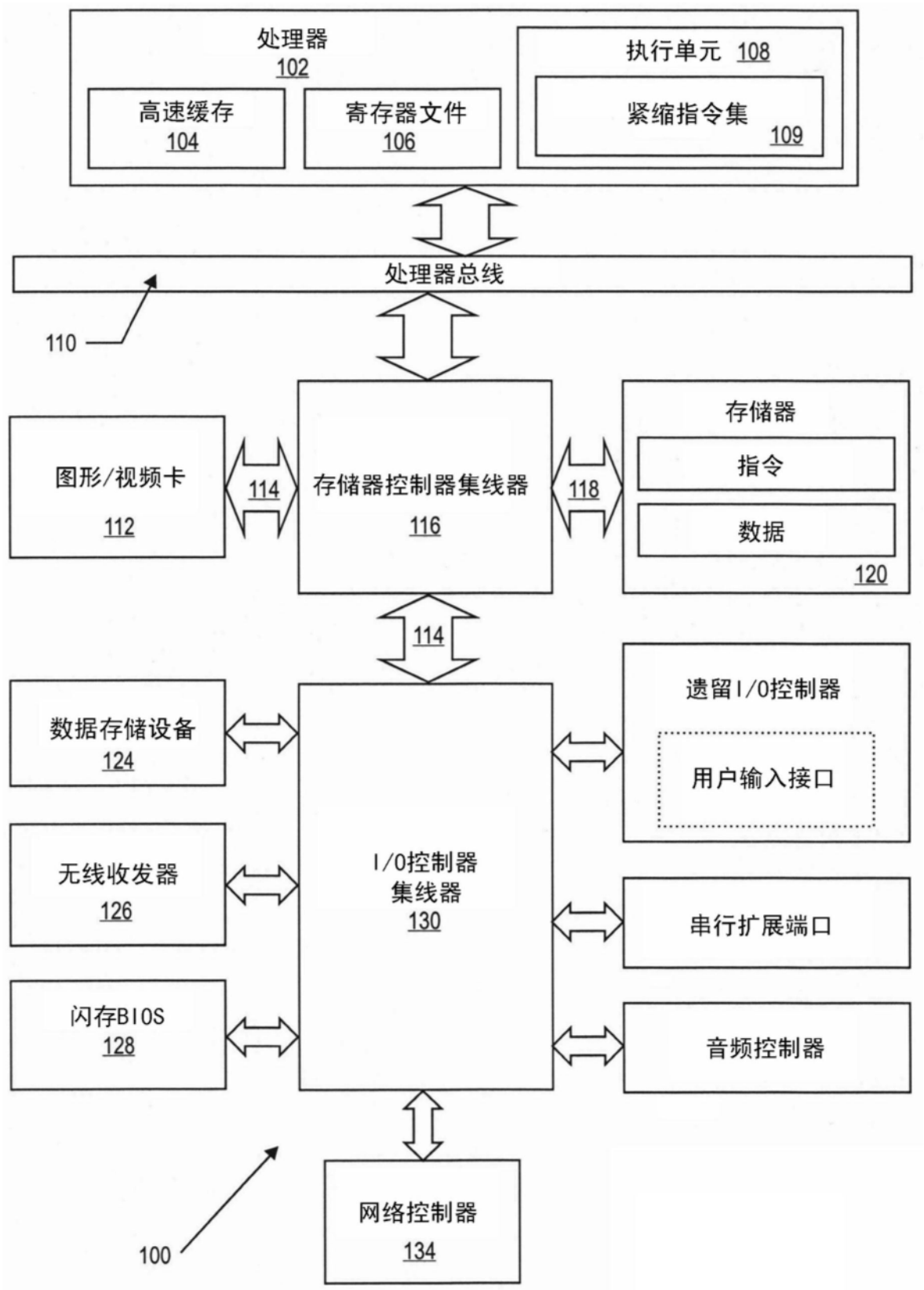

处理器
102
高速缓存
104
寄存器文件
106
执行单元 108
紧缩指令集
109
处理器总线
110
图形/视频卡
112
114
存储器控制器集线器
116
118
存储器
指令
数据
120
114
数据存储设备
124
无线收发器
126
闪存BIOS
128
I/O控制器
集线器
130
遗留I/O控制器
用户输入接口
串行扩展端口
音频控制器
网络控制器
134
100

图1A

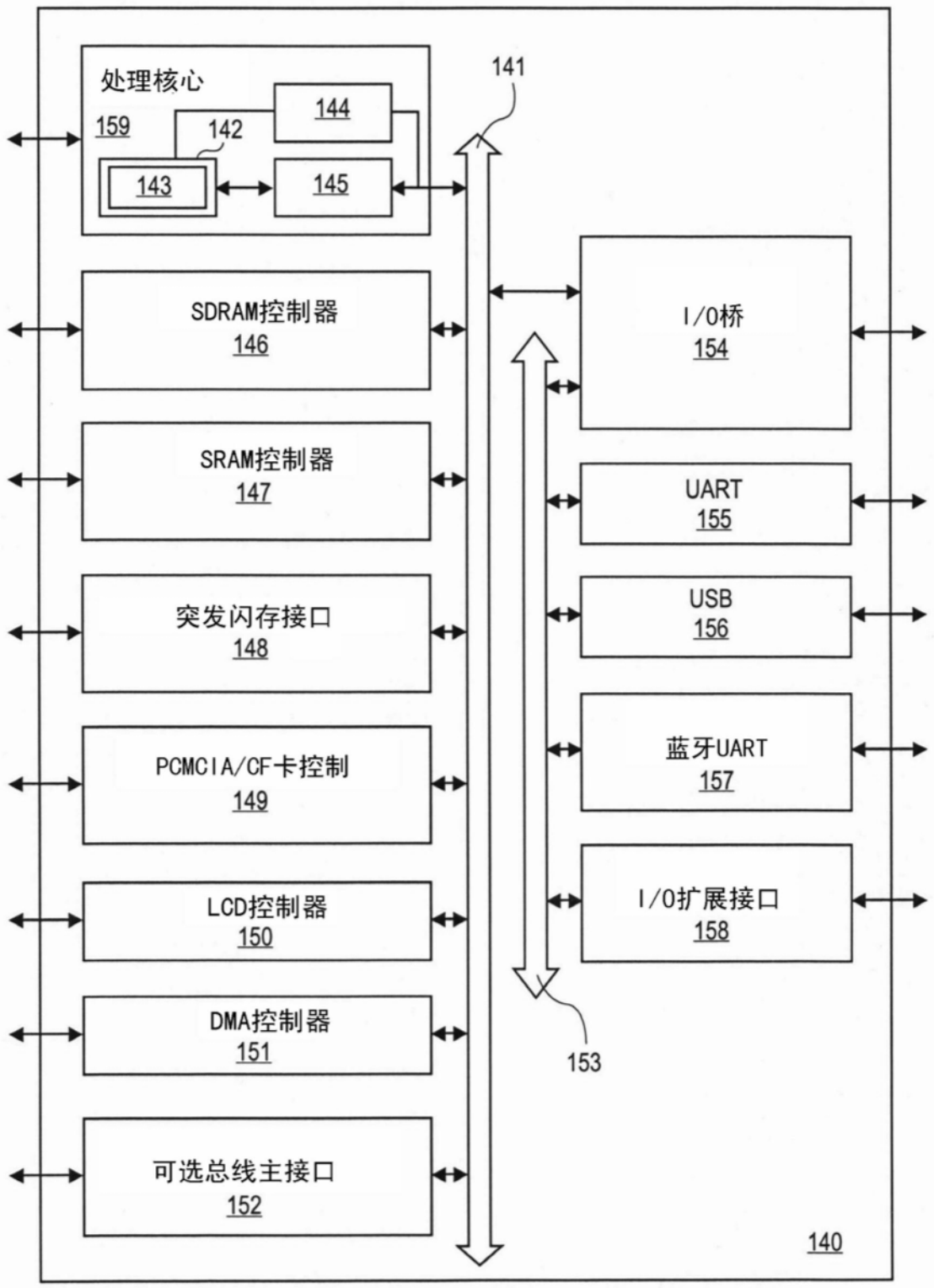

处理核心
159
142
144
143
145
141
SDRAM控制器
146
SRAM控制器
147
突发闪存接口
148
PCMCIA/CF卡控制
149
LCD控制器
150
DMA控制器
151
可选总线主接口
152
I/O桥
154
UART
155
USB
156
蓝牙UART
157
I/O扩展接口
158
153
140

图1B

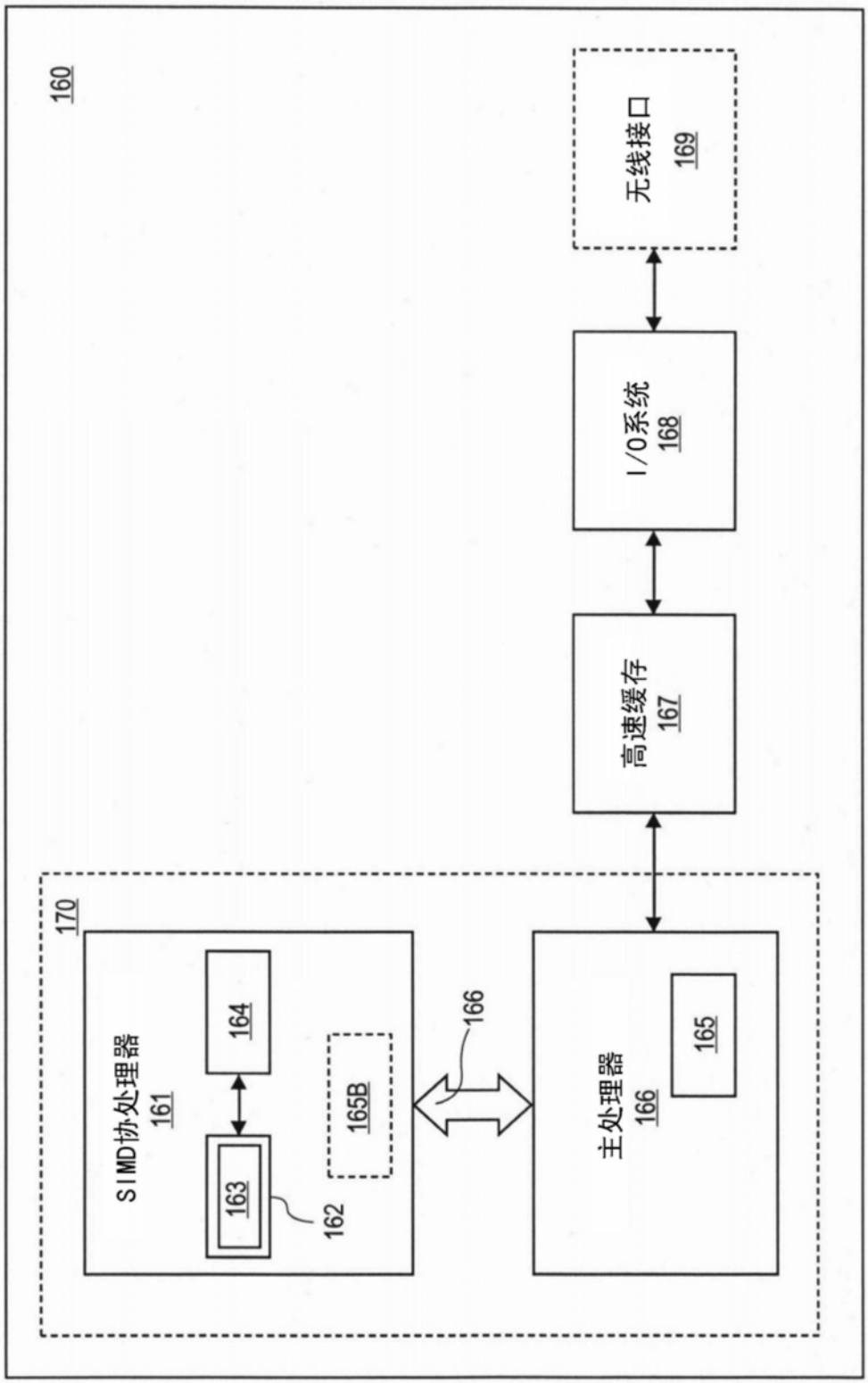

160
无线接口
169
I/O系统
168
高速缓存
167
170
SIMD协处理器
161
164
163
162
165B
166
主处理器
166
165

图1C

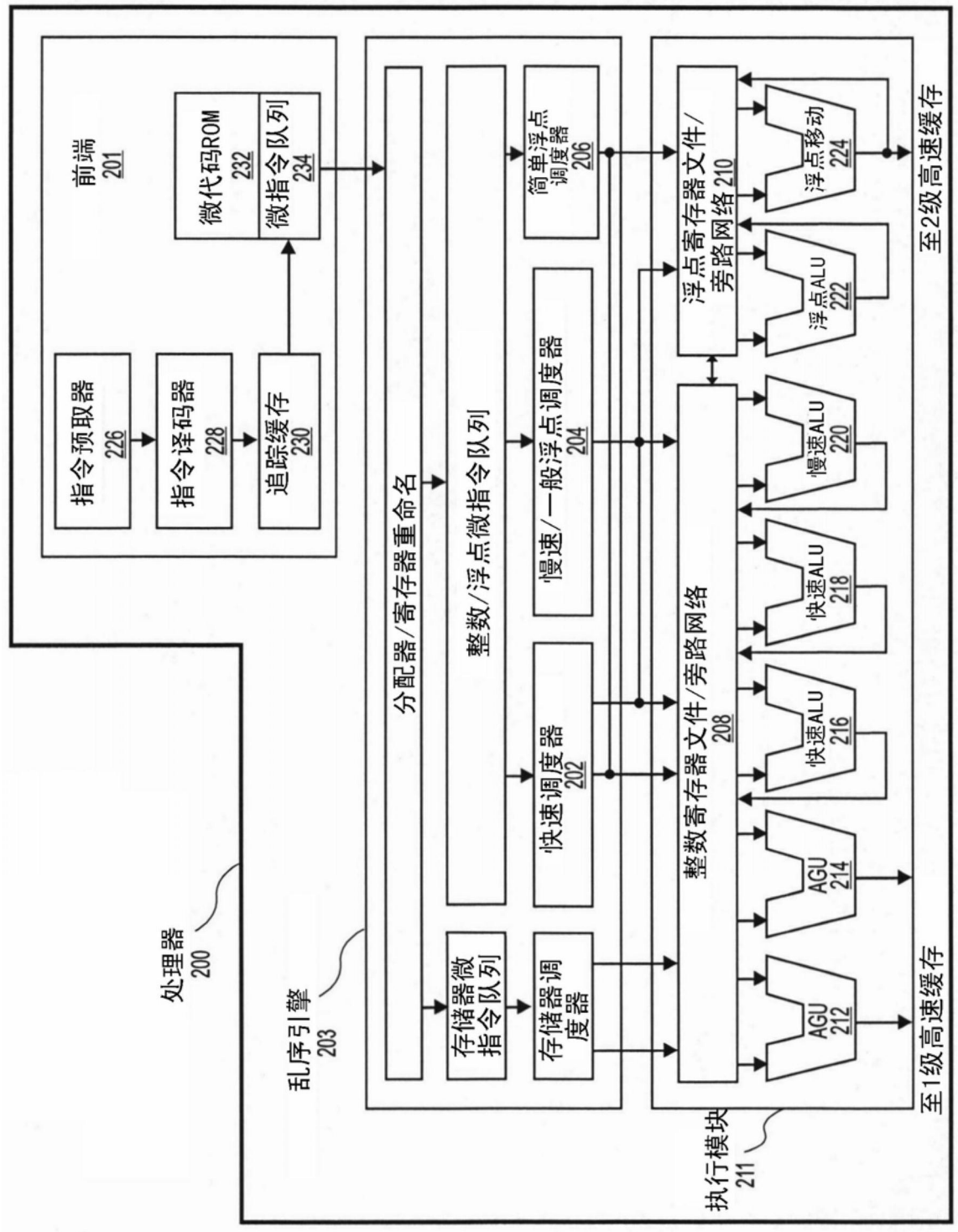

处理器
200
前端
201
指令预取器
226
指令译码器
228
追踪缓存
230
微代码ROM
232
微指令队列
234
乱序引擎
203
分配器/寄存器重命名
存储器微指令队列
存储器调度器
整数/浮点微指令队列
快速调度器
202
慢速/一般浮点调度器
204
简单浮点调度器
206
整数寄存器文件/旁路网络
208
浮点寄存器文件/旁路网络 210
AGU
212
AGU
214
快速ALU
216
快速ALU
218
慢速ALU
220
浮点ALU
222
浮点移动
224
执行模块
211
至1级高速缓存
至2级高速缓存

图2

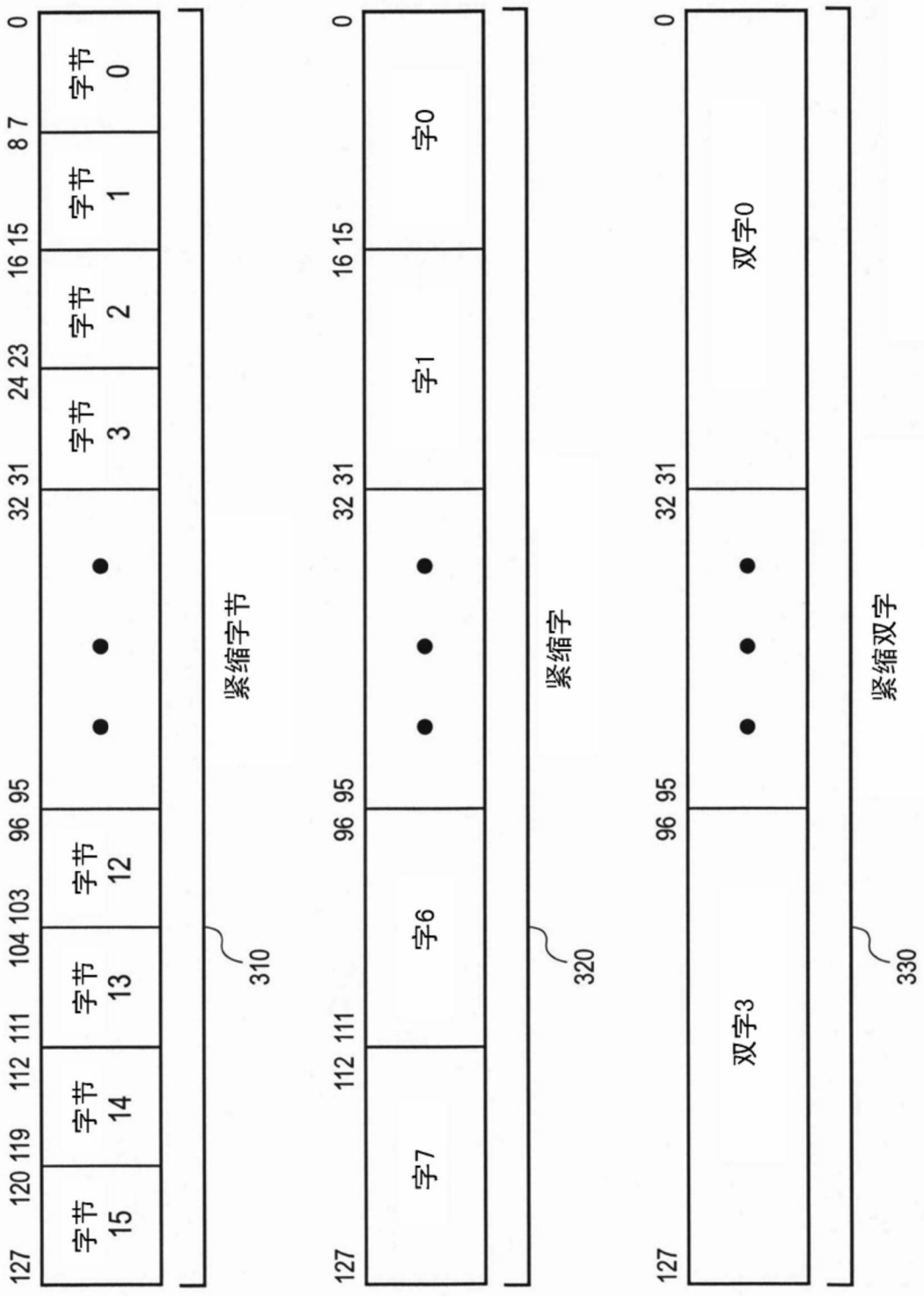

0
7 8
15 16
23 24
31 32
95 96
103 104
111 112
119 120
127
字节 0
字节 1
字节 2
字节 3
字节 12
字节 13
字节 14
字节 15
紧缩字节
310
0
15 16
31 32
95 96
111 112
127
字0
字1
字6
字7
紧缩字
320
0
31 32
95 96
127
双字0
双字3
紧缩双字
330

图3A

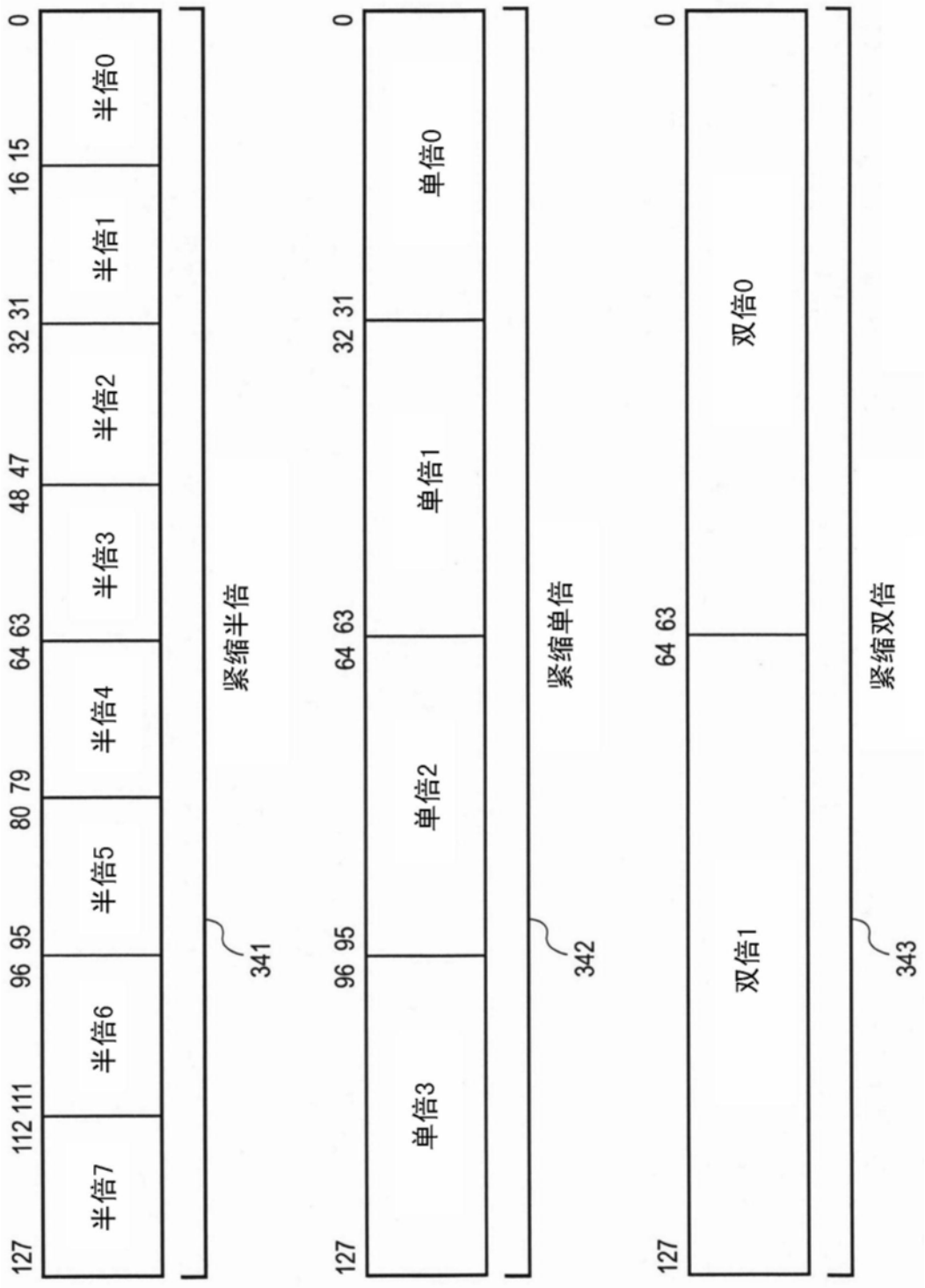
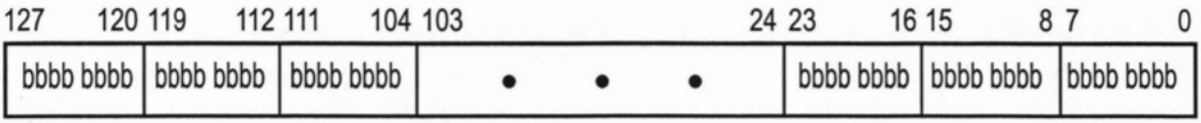
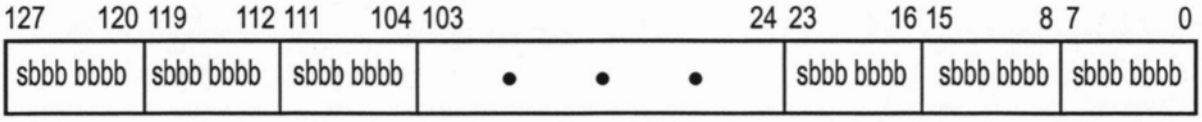

0
16 15
32 31
48 47
64 63
80 79
96 95
112 111
127
半倍0
半倍1
半倍2
半倍3
半倍4
半倍5
半倍6
半倍7
紧缩半倍
341
0
32 31
64 63
96 95
127
单倍0
单倍1
单倍2
单倍3
紧缩单倍
342
0
64 63
127
双倍0
双倍1
紧缩双倍
343

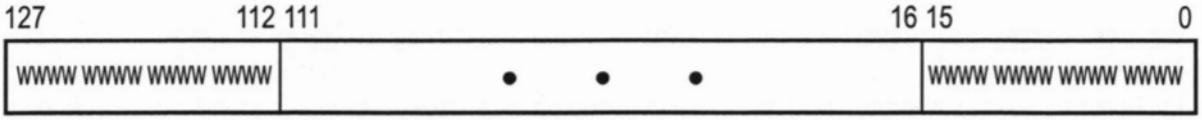
图3B



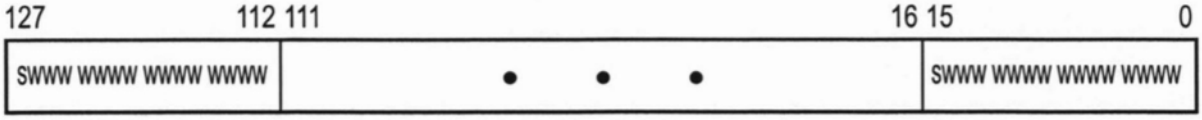
无符号紧缩字节表示344



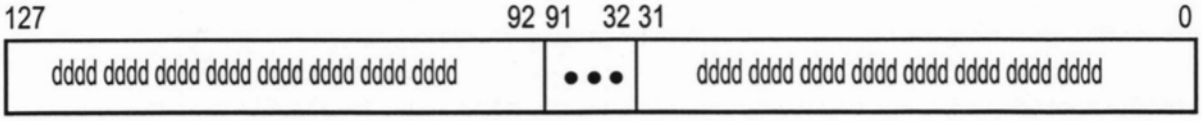
有符号紧缩字节表示345



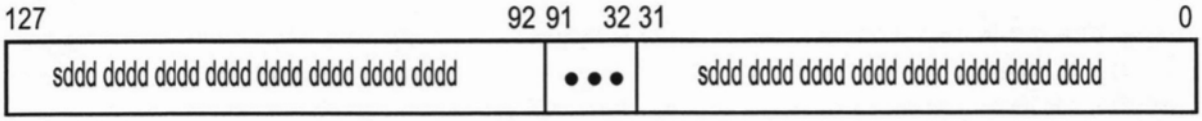
无符号紧缩字表示346



有符号紧缩字表示347



无符号紧缩双字表示348



有符号紧缩双字表示349

图3C

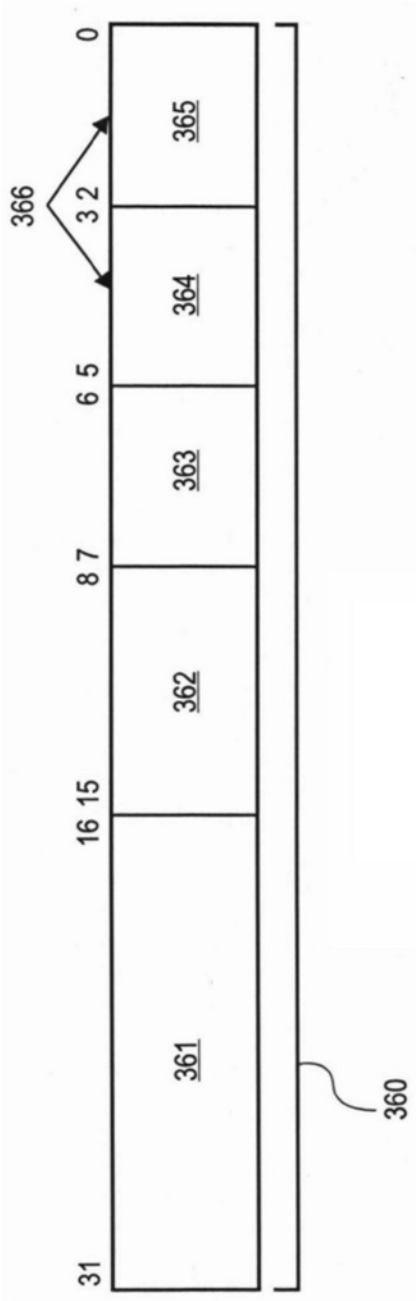


图3D

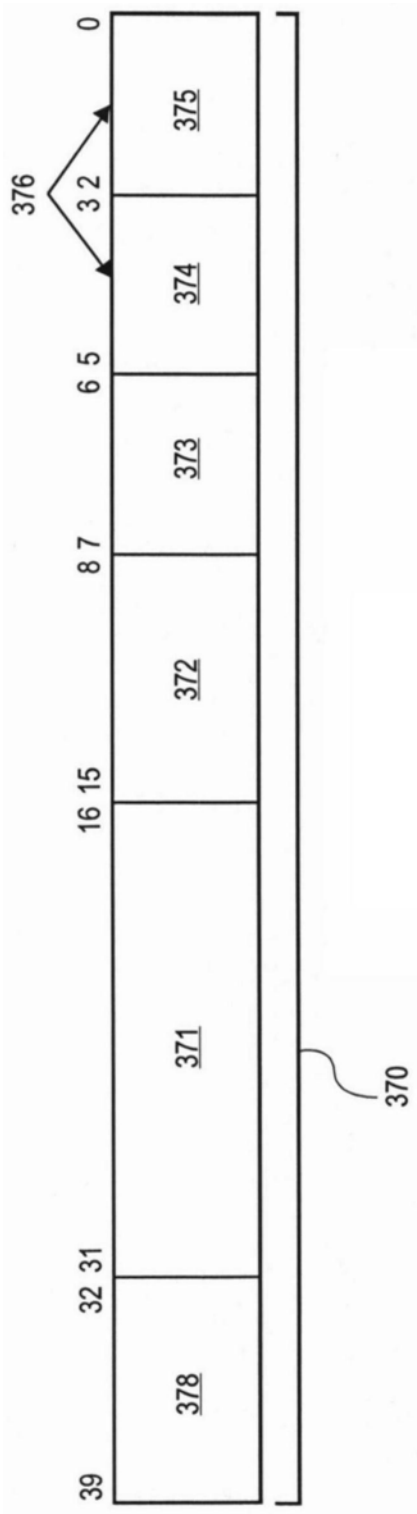


图3E

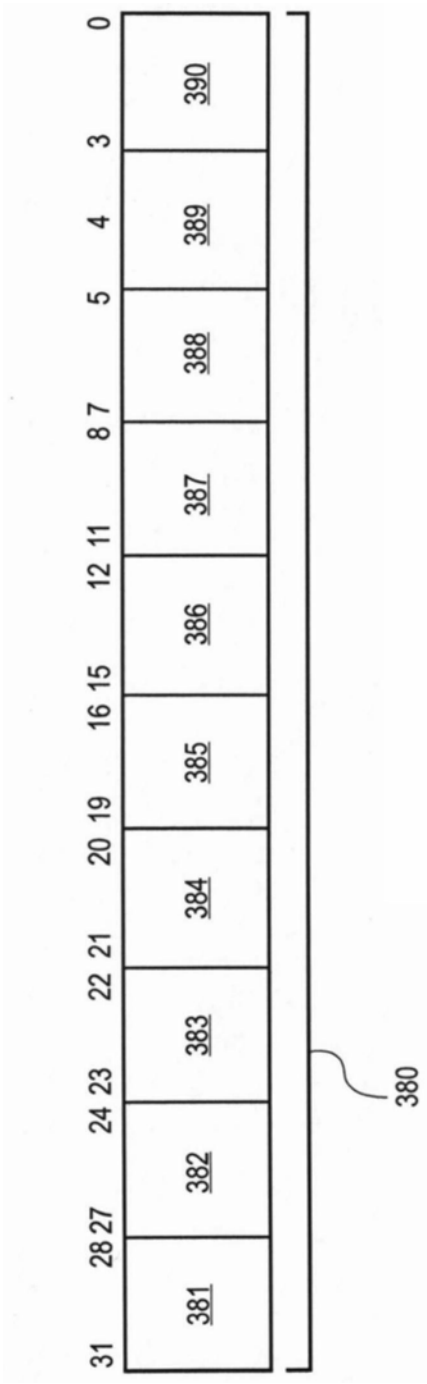

0
3
4
5
8 7
12 11
16 15
20 19
22 21
24 23
28 27
31
390
389
388
387
386
385
384
383
382
381
380

图3F

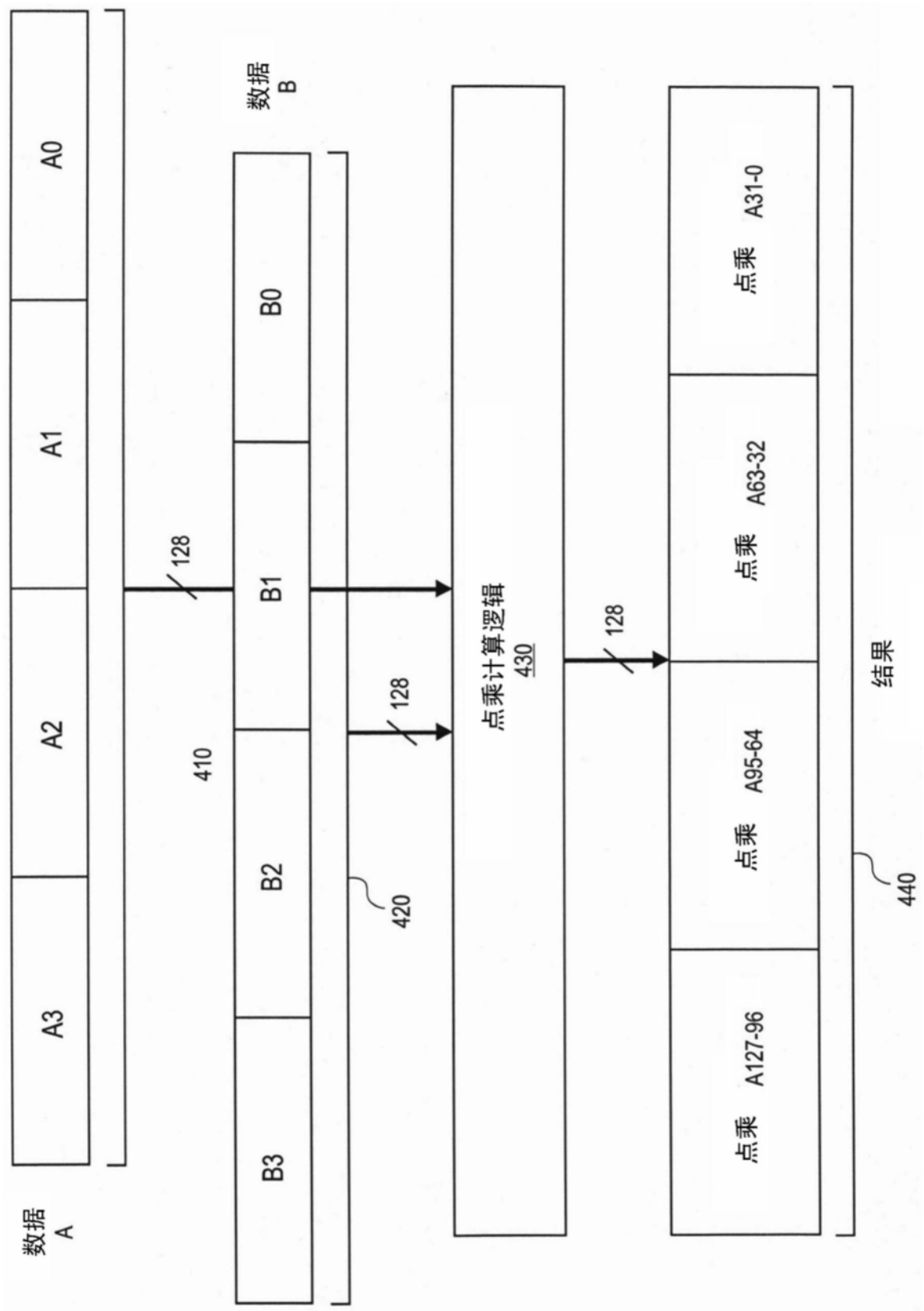

数据 A
A0
A1
A2
A3
数据 B
B0
B1
B2
B3
128
128
128
410
420
点乘计算逻辑
430
点乘 A31-0
点乘 A63-32
点乘 A95-64
点乘 A127-96
结果
440

图4

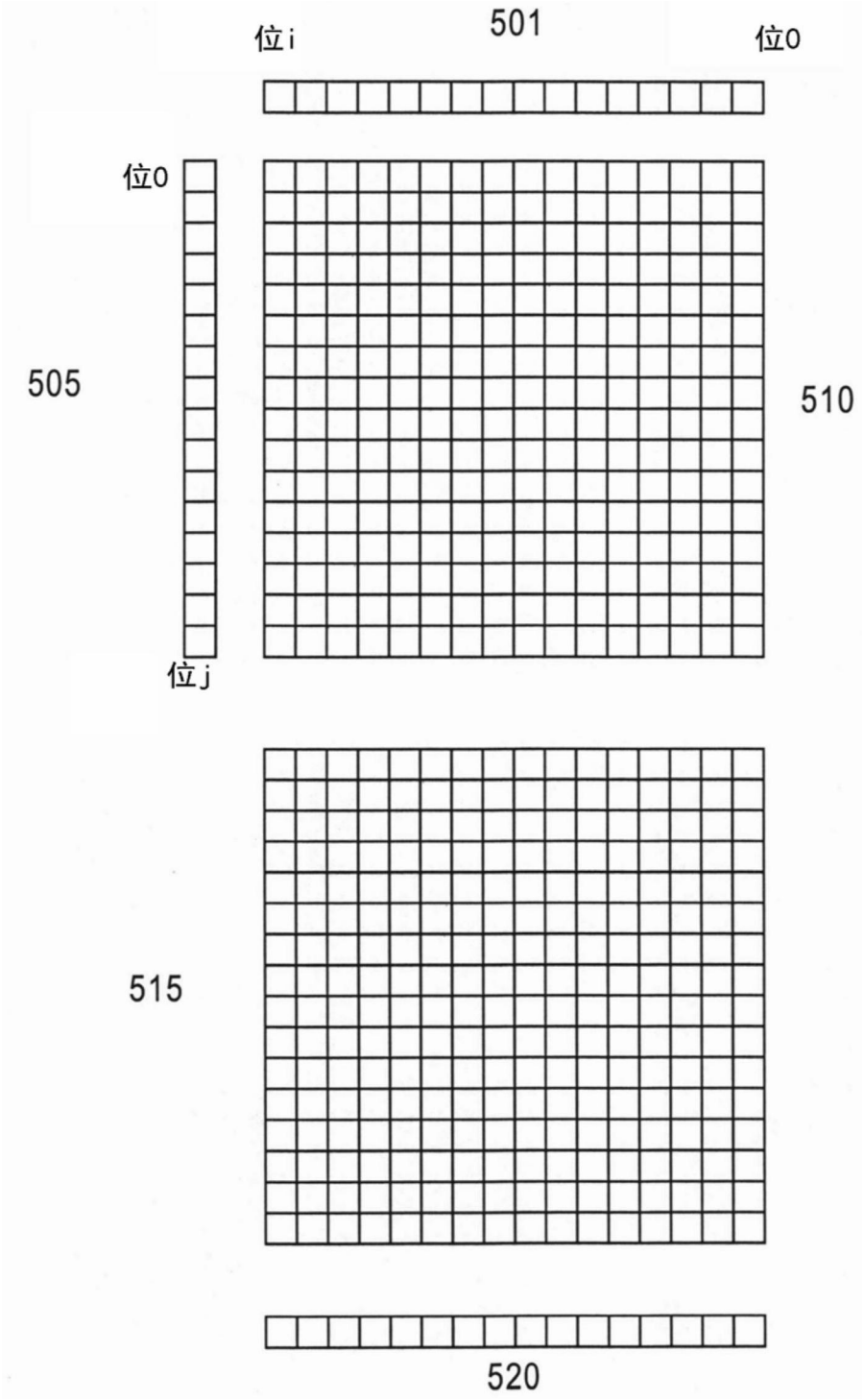


图5

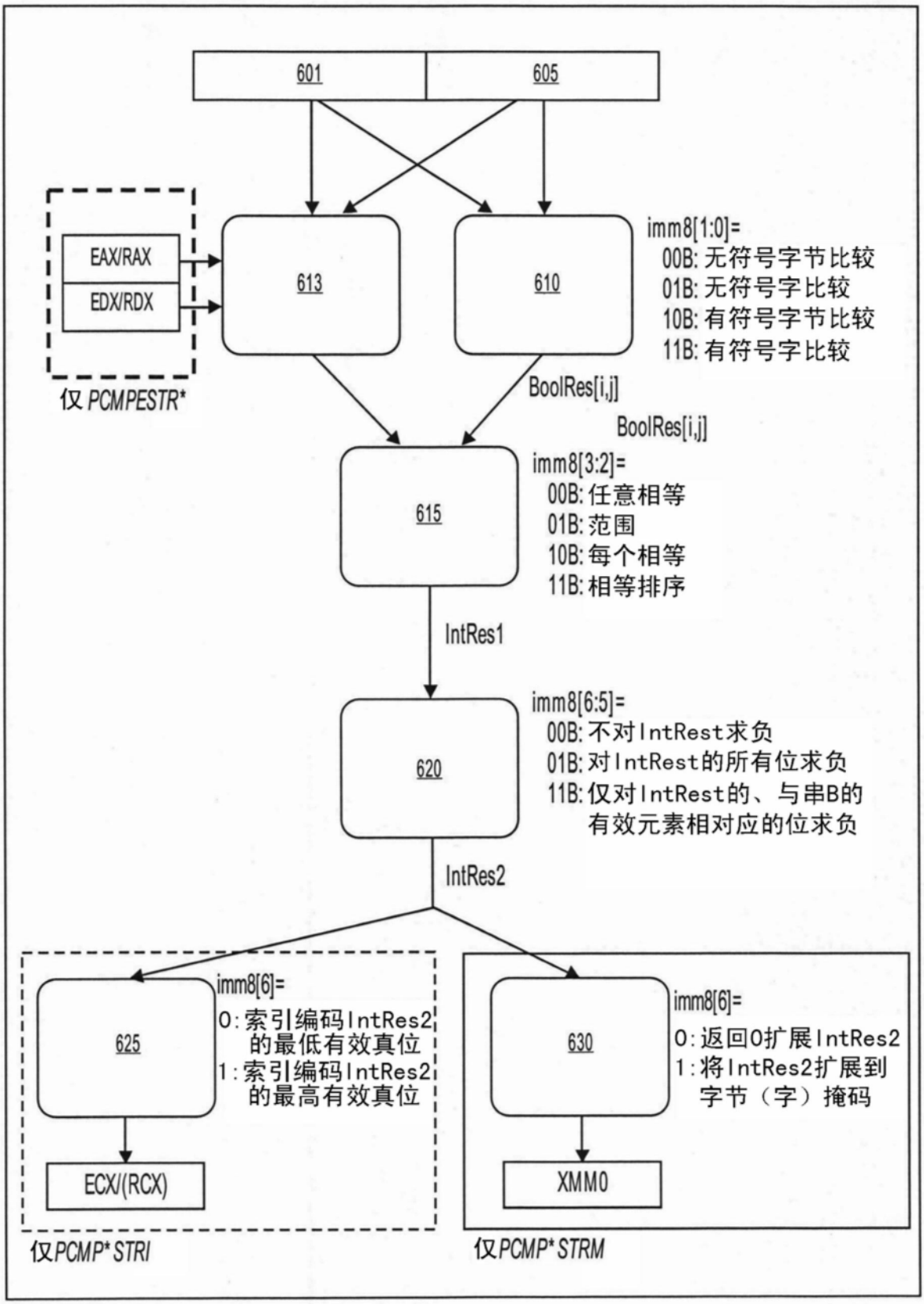

601
605
EAX/RAX
EDX/RDX
613
610
仅 PCMPESTR*
imm8[1:0]=
00B: 无符号字节比较
01B: 无符号字比较
10B: 有符号字节比较
11B: 有符号字比较
BoolRes[i,j]
BoolRes[i,j]
615
imm8[3:2]=
00B: 任意相等
01B: 范围
10B: 每个相等
11B: 相等排序
IntRes1
620
imm8[6:5]=
00B: 不对IntRest求负
01B: 对IntRest的所有位求负
11B: 仅对IntRest的、与串B的
有效元素相对应的位求负
IntRes2
625
imm8[6]=
0: 索引编码IntRes2
的最低有效真位
1: 索引编码IntRes2
的最高有效真位
ECX/(RCX)
仅PCMP*STRI
630
imm8[6]=
0: 返回0扩展IntRes2
1: 将IntRes2扩展到
字节（字）掩码
XMM0
仅PCMP*STRM

图6