[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
G06T 1/20

[12] 发明专利申请公开说明书

[21] 申请号 200510070176.5

[43] 公开日 2005 年 10 月 19 日

[11] 公开号 CN 1684104A

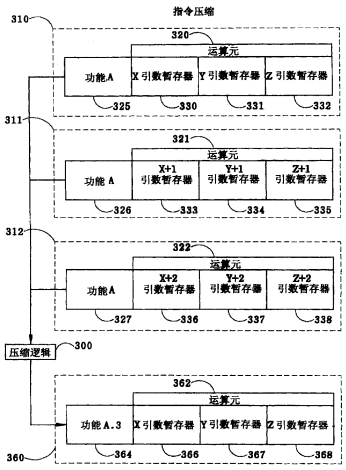
[22] 申请日 2005.5.9
[21] 申请号 200510070176.5
[30] 优先权
[32] 2004. 7.26 [33] US [31] 10/898,784
[71] 申请人 威盛电子股份有限公司
地址 台湾省台北县新店市
[72] 发明人 萨伊德·荷圣
[74] 专利代理机构 中原信达知识产权代理有限责任公司
代理人 文 琦 陈肖梅

权利要求书 3 页 说明书 17 页 附图 11 页

[54] 发明名称 计算器系统中压缩及解压缩指令的方法与装置

[57] 摘要

本发明是一种计算器系统中压缩及解压缩指令的方法及装置，该装置及方法可以提升计算机的性能，通过压缩多个具有相同功能且为连续地址运算元的指令，及通过复制已增加的运算元，解压缩已压缩指令。 本发明将实现及改进对于指令压缩及解压缩的系统及方法的效率。



ISSN1008-4274

1. 一种计算器系统中压缩及解压缩指令的装置，其特征是，包含：

一压缩逻辑，用以压缩复数个指令，其中每一该指令包含至少一运算元，其中每一该指令的运算元被配置于复数个连续暂存器，以及其中一已压缩指令包含压缩资料；

一解压缩逻辑，用以解压缩该已缩压指令，其中该压缩数据用以产生复数个解压缩指令；及

一指令队列逻辑，用以储存该复数个解压缩指令，该指令队列逻辑更分割一指令暂存器用以处理位于一第一指令队列及一第二指令队列中的该复数个解压缩指令。

2. 如权利要求 1 所述的计算器系统中压缩及解压缩指令的装置，其特征是，上述的运算元包含至少一引数数据域。

3. 如权利要求 2 所述的计算器系统中压缩及解压缩指令的装置，其特征是，上述的压缩逻辑更用以决定一压缩时指令数，其中该压缩时指令数储存于该已压缩指令的一复制值栏。

4. 如权利要求 3 所述的计算器系统中压缩及解压缩指令的装置，其特征是，上述每一该解压缩指令包含该已压缩指令之一复制，其中该复制已压缩指令包含一修改运算元，其系由修改该已压缩指令运算元所产生。

5. 如权利要求 4 所述的计算器系统中压缩及解压缩指令的装置，其特征是，上述的修改运算元对于每一对应复制特殊指令功能包含累进该已压缩指令运算元。

6. 如权利要求 5 所述的计算器系统中压缩及解压缩指令的装置，

其特征是，上述的解压缩逻辑更包含：

决定一最大复制值，其中该最大复制值等于在该第一指令队列的该已压缩指令以及在该第二指令队列的该已压缩指令两者中的较大复制值；

在该第一指令队列插入无运算指令，其中该无运算指令的数目系等于该最大复制值与该第一指令队列的复制值之差值；及

在该第二指令队列插入无运算指令，其中无运算指令的数目系等于该最大复制值与该第二指令队列的复制值之差值。

7. 如权利要求 6 所述的计算器系统中压缩及解压缩指令的装置，其特征是，上述每一该第一指令队列及每一该第二指令队列所储存的指令系用于不同的处理器之执行线。

8. 一种计算器系统中压缩及解压缩指令的方法，包含：

压缩复数个指令，产生一已压缩指令，其特征是，该复数个指令包含一指令功能应用于复数个连续寻址运算元；

传递该已压缩指令至一指令暂存器；及

解压缩该已压缩指令，其中该解压缩包含产生解压缩指令至复数个指令队列。

9. 如权利要求 8 所述的计算器系统中压缩及解压缩指令的方法，其特征是，上述的复数个指令队列包含一第一指令队列及一第二指令队列，其中该第一指令队列及该第二指令队列都各包含相同资料储存容量。

10. 如权利要求 9 所述的计算器系统中压缩及解压缩指令的方法，其特征是，上述的第一指令队列更包含一第一指令译码器，且该第二指令队列更包含一第二指令译码器。

11. 如权利要求 10 所述的计算器系统中压缩及解压缩指令的方

法，其特征是，上述的第一指令队列和该第二指令队列用以接收相同数量的解压缩指令数，

如果在该第二指令队列的解压缩指令数目大于储存在该第一指令队列的解压缩指令数目，则将含有无运算的指令储存于该第一指令队列中，

如果在该第一指令队列的解压缩指令数目大于储存在该第二指令队列的解压缩指令数目，则将含有无运算的指令储存于该第二指令队列中。

12. 如权利要求 11 所述的计算器系统中压缩及解压缩指令的方法，其特征是，上述的已压缩指令更包含：

一第一压缩数据域，用以储存该已压缩指令的数目；及

一第二压缩数据域，用以储存一运算元栏识别符，其中该运算元栏识别符包含修改相关的运算元之选择资料。

13. 如权利要求 12 所述的计算器系统中压缩及解压缩指令的方法，其特征是，上述的解压缩步骤更包含：

复制该已压缩指令，其中该已压缩指令的复制次数等于被压缩的指令数目；

在解压缩指令中，修改该已压缩指令运算元，其中该指令栏每复制一次，则该指令运算元就累进；及

储存一复制指令于该复数个指令队列其中之一。

计算器系统中压缩及解压缩指令的方法与装置

技术领域

本发明涉及计算器系统，特别是关于在计算器系统中压缩及解压缩指令的方法及装置。

背景技术

就一般所知，三维空间（3D）计算机绘图技术所关心的是如何产生或呈像(rendering)3D 对象（object）于二维空间（2D）影像的显示器或展示用的显示元件或监视器，如阴极射线管（CRT）或液晶显示器（LCD）。对象可能为一简单的几何元件，如：一个点、一线段（line segment）、三角形（triangle）或多边形（polygon）。而通过一系列连接的平面多边形的对象也可呈像较复杂的对象，举例而言，可通过一系列连接的平面三角形。所有的几何基元（geometry primitives）都可被一个顶点（vertex）或一组顶点所表示，举例而言，坐标(x,y,z)定义一个点，如一线段的端点或一多边形的一个角。

产生一资料集在计算机监视器或其它显示元件以显示表示三维基元的二维投影，基元的顶点为经由一系列的运算处理或在绘图呈像管线（graphics-rendering pipeline）处理阶段（stage）。一般所说的管线仅为一系列串接的处理单元或阶段，其中前阶段的输出提供后续阶段的输入。在绘图处理器的背景（context）下，这些阶段包含如：每一顶点运算、基元组合（primitive assembly）运算、像素（pixel）运算、纹理组合（texture assembly）运算、描绘处理器（rasterization）运算及片段（fragment）运算。

一典型的绘图显示系统中，影像数据库（如：命令列表（command list））可储存景物（scene）中对象的描述（description）。对象可为

一数量的多边形所描述，其中使用多边形覆盖对象表面使用相同的手法（manner）-如使用一数量的分割图块（tile）可以覆盖一整面墙（wall）或其它表面。而每一多边形可描述成一列顶点坐标（X,Y,Z 为一"模型"坐标）及一些实质面规格的特性（如：颜色、纹理及光亮（shininess）等），同样地使用一般的向量（vector）也可以描述对于每一顶点的表面。对于具有复杂曲面的三维对象，多边形必须为三角形或四角形（quadrilateral），其中四角形为方便分解成许多成对的三角形。

转换引擎（transformation engine）转换对象坐标，以在使用者的输入中对使用者观看所选择的角度作反应。另外，为包含或删除所需的背景，使用者可能明确说明观看的区域（field）、被产生的影像大小及视觉范围（viewing volume）后端。

一旦这观看区域（viewing area）被选择，剪辑逻辑（clipping logic）删除超出观看区域的多边形（如：三角形）及"剪掉"具有部分在观看区域内部及外部的多边形。这些被剪辑的多边形相当于观看区域内部多边形部分的边缘，而这些边缘也相当于观看区域的边缘。接着，多边形的顶点，相当于浏览屏幕坐标（X、Y 坐标）以及每一顶点的相关深度（即 Z 坐标），被传送至下一阶段。接着在一典型系统中，一光影模块（lighting model）在考量一光源下将被加以应用。这些多边形包含其颜色值被传送至一描绘处理器（rasterizer）。

对于每一多边形而言，描绘处理器决定哪些像素位置被多边形所覆盖，且试图将相关颜色值及深度值(即 Z 值)写入至讯框缓冲器(frame buffer）。描绘处理器（rasterizer）用以比较在处理中的多边形深度值（Z）与已被写入讯框缓冲器的像素的深度值。如果新多边形像素的深度值较小，则表示已经在原多边形之前，于是写入讯框缓冲器，然后它的值将取代在讯框缓冲器的值，因为新多边形将模糊（obscure）先前处理的多边形及写入讯框缓冲器。此过程重复着直到所有的多边

形被描绘出（rasterized）。此时，一影像控制器（video controller）展示一讯框缓冲器的内容依据描绘的顺序一次以一扫描线展示出。

有关于此技术的背景，请参阅图 1，其显示在计算机绘图系统中具有绘图管线的元件的功能流程图。需注意的是系统中绘图管线是可变化的，而实现的方式也是可变化的。如一般所知，主机 10（或在主机计算机上执行的绘图应用程序介面）可产生一命令列表（command list）12，其包含了一连续的绘图命令及资料以呈像一"环境"在一绘图显示区。具绘图管线的元件可运算命令列表 12 的资料及命令以呈像一屏幕至绘图显示区。

就此，剖析器（parser）14 可撷取来自命令列表 12 的资料且"剖析（parse）"资料以解释命令及经由（或进入）绘图管线传递定义的绘图基元的资料。就此，绘图基元不但可经由地址资料（如：坐标 x,y,z 及 w），也可通过光度及纹理信息来定义。所有这些信息对于每一基元可从命令列表 12，再经剖析器 14 撷取，并传递至顶点着色器 16。一般来说，在接收来自命令列表 12 的绘图资料，顶点着色器 16 可执行不同的转换（transformation）。就此，资料可能转换来自世界坐标（World coordinate）到模型观看坐标（Model View coordinate），到投影坐标（Projection coordinate）而最终至屏幕坐标（screen coordinate）。顶点着色器 16 的实质功用在此已经说明清楚将不在赘述，接下来绘图资料可传递到描绘处理器 18，也由于描绘处理器 18 在前面也已经叙述在此就不在赘述。

再接下来，Z-测试（Z-test）20 执行在每一运算中的基元的像素。就像一般所知，由比较现在 Z-值（如对于现在基元的所给像素的 Z 值）与比较对应于像素位置的储存的 Z-值，来执行这个 Z-测试。储存的 Z-值对先前呈像的基元提供深度值以对所给像素的位置。如果现在 Z-值显示一深度比储存的 Z-值接近于观看者的眼睛，然后现在 Z-值将取代储存的 Z-值及现在绘图信息（如：颜色），也将取代在对应的讯框缓

冲器像素位置（由像素着色器 22 决定）的颜色信息。假设现在 z 值相比于所储存的 z 值，较为不接近于目前的观看点（viewpoint），则讯框缓冲器及 z 缓冲器的内容不需要被取代，先前已呈像的像素将被视作在现在像素的前面。

再一次对于像素中呈像及决定的基元接近于观看点而不接近于先前储存的像素，信息相关于基元是传递在像素着色器 22。像素着色器 22 然后决定颜色信息对于每一像素具有决定的基元接近现在观看点。一旦颜色信息经像素着色器 22 计算，信息储存在讯框缓冲器 24 以方便接着而来的显示。

上面所讨论的计算机绘图处理功能中特别强调资料及指令，因此本发明将实现及改进对于指令压缩及解压缩的系统及方法的效率。

发明内容

本发明的目的是提供一种计算器系统中压缩及解压缩指令的方法及装置，本发明将实现及改进对于指令压缩及解压缩的系统及方法的效率。

为达上述目的，本发明的具体实施例提供一计算器系统具有压缩逻辑用以压缩复数个指令，其中该复数个指令包含复数个运算元，其中每一该指令包含一复数个运算元，其中该复数个运算元位于复数个连续暂存器，其中每一该指令更包含一特定指令功能应用于该复数个指令；解压缩逻辑用以解压缩已压缩指令，其中该已压缩指令展开为复数个解压缩指令；以及指令队列逻辑用以储存该复数个指令，及用以储存已压缩指令，更用以分割一指令暂存器，其中该指令暂存器包含复数个指令队列，其中该复数个指令队列包含一第一指令队列，且其中该复数个指令队列更包含一第二指令队列。

本发明的具体实施例也可视为提供压缩及解压缩计算器指令的方法，由此可知，在众多实施例中一实施例的方法大致可以归纳成下列步骤：

压缩复数个指令，其中该复数个指令包含一指令应用于复数个元件值，其中该复数个指令压缩成一已压缩指令；及解压缩该已压缩指令，其中该解压缩包含决定一数量该复数个元件值，其中该解压缩更包含决定哪一个已压缩指令的元件的修改，以及其中该解压缩更包含储存解压缩指令于一指令暂存器。

本发明其它的系统、方法、特征及优点将或清楚的显现技术于接下来的附图以及详细的说明。而本发明不但保护接下来详细说明中附加的系统、方法、特征及优点，也保护所述的权利要求范围所包含的内容。

附图说明

本发明可参考所示附图及说明以帮助了解，图中的元件并未按照比例绘制，反而用以强调以及清楚地显示本发明原理。再者，在图中参考数字标示对应的部分也使用在许多地方。

图 1 显示已知的绘图管线的方块图；

图 2 显示压缩/解压缩系统选择元件的方块示意图；

图 3 显示指令压缩方法的一实施例的方块示意图；

图 4 显示在实施例中指令处理的方块示意图；

图 5 显示指令暂存器及指令队列格式的实施例示意图；

图 6 显示已压缩指令的实施例示意图；

图 7 显示一实施例中提供运算元栏识别符值的定义表；

图 8 显示栏识别符逻辑的实施例方块示意图；

图 9 显示指令解压缩的实施例方块示意图；

图 10 显示指令解压缩逻辑的实施例方块示意图；及

图 11 显示在解压缩中指令的复制及修改示意图。

图中符号说明

10 主机（绘图应用程序介面）

12 命令列表

14 剖析器

16 顶点着色器

18 描绘处理器

20 Z-测试

22 像素着色器

24 讯框缓冲器

200 数据处理单元

202 编译器

204 压缩逻辑

206 指令快取

208 解压缩逻辑

210 队列逻辑

212 指令队列

300 压缩逻辑

310 第一指令

311 第二指令

312 第三指令

320-322 运算元

325-327 功能 A

330 X 引数暂存器

331 Y 引数暂存器

332 Z 引数暂存器

333 X+1 引数暂存器

334 Y+1 引数暂存器

335 Z+1 引数暂存器

336 X+2 引数暂存器

337 Y+2 引数暂存器

338 Z+2 引数暂存器

360 已压缩指令

362 运算元

364 功能 A.3

366 X 引数暂存器

367 Y 引数暂存器

368 Z 引数暂存器

410 增加程序计数器

420 指向指令快取内的指令

430 提取 128 位元指令

440 储存 128 位元指令于指令暂存器内的两个 64 位元指令

450 解压缩 64 位元指令至指令队列

510 128 位元指令暂存器（127-0）

512 64 位元指令（127-64）

514 64 位元指令（63-0）

520 指令队列

521 N 分割（N=2）

522 指令队列（左）

524 指令队列（右）

526 L 位阶

528 L 位阶

610 复制数

620 栏识别符

630 运算元引数

801 位元 54

802 位元 53

810 互斥或

812 无复制

820 复制

830 修改 X 引数及 Z 引数

840 位元 53=1

850 修改 Y 引数

870 无修改 Y 引数

910 复制及修改左指令队列及右指令队列

920 从压缩指令读复制值

930 读压缩指令运算码

940 从压缩指令读栏识别符

950 复制指令与引数栏增加

960 决定最大的复制数

970 插入"无运算"功能以平衡左队列及右队列

1001 128 位元指令暂存器（127-0）

1002 64 位元指令（127-64）

1003 64 位元指令（63-0）

1010 复制及修改（左）

1012 读复制值 P=（62..61）

1014 插入无运算指令 M-P 次

1020 复制及修改（右）

1022 读复制值 Q（62..61）

1024 插入无运算指令 M-P 次

1030 决定最大复制数 M=MAX（P,Q）

1040 指令队列（左）

1050 指令队列（右）

具体实施方式

归纳本发明各种变化的例子，可参阅本发明的附图及详尽的描述，且在对照发明描述及附图时并不受限于这些具体实施例及在此揭露的例子，相对地在此更包含由申请专利范围所述的包含本发明精神及范围的可替代性例子、可修改性例子以及相等的例子。

需注意的是附图中提供了本发明具体实施例的特征及外观（aspect），文字的描述更提供了符合本发明精神及范围的一些可变化的实施例及可实现的方式。

依据以上的归纳，本发明的应用为直接在计算机系统中，以下是压缩及解压缩指令的设备、系统及方法的具体实施例。

参阅一简略绘制的图 2，其为表示指令压缩/解压缩方法或设备中的一具体实施例的选择系统元件的流程图。如图所示，压缩/解压缩系统其特征为数据处理单元 200 利用应用于压缩逻辑（compression logic）204 的编译器（compiler）202 以多工指令。压缩逻辑 204 压缩多个指令使其成为一个指令，然后放入指令快取（instruction cache）206。压缩的执行也可经由压缩逻辑 204 及在数据处理单元 200 外部的编译器来完成。如图 4 所示（步骤 410 及 420 中），已压缩指令保持在指令快取 206 直到程序计数器（program counter）指到此指令才执行。

当程序计数器指到在指令快取 206 的已压缩指令（步骤 420），已压缩指令置于指令队列（instruction queue）212 及使用解压缩逻辑（decompression logic）208 解压缩。指令队列 212 是经队列逻辑（queuing logic）210 的无运算指令（no-operation instructions）管理及平衡（balance）。熟知此技艺者必然知道，解压缩逻辑 208 及队列逻辑 210 可以在指令队列 212 外部或是整合于指令队列 212 内。

参阅图 3 其显示一指令压缩方法的具体实施例。未压缩指令 310-312 有相同的指令功能 A（function A）元件 325-327 及运算元（operand）320-322 位于连续的暂存器（register）中。在一系列具有连续运算元的重复指令中典型的如：像素着色器（pixel shader）、Z 测试（Z-test）及顶点着色器（vertex shader）的绘图呈像管线阶段（graphics rendering pipeline stage）。

本实施例的特征为指令具有运算元 320-322，其中每一运算元有三个引数（argument 自变量）330-338。在三个未压缩指令中的每一个指令所对应的引数（自变量）为在连续的地址暂存器。举例而言，第一指令 310 的"引数"在 X 引数暂存器 330，为对应于第二指令 311 的"引数"在连续的地址 X+1 引数暂存器 333，以及对应于第三指令 312 的"引数"在 X+2 引数暂存器 336。依此类推，在第一指令 310 的"Y"及"Z"引数 331-332 也有相同对应连续的"+1"及"+2"在第二及第三指令。

由压缩逻辑 300 压缩多个具有相同指令功能的指令及连续的地址运算元至一单一压缩指令 360。已压缩指令 360 有一运算元 362 其包含引数 366-368 且其等于第一指令（未压缩）310 的引数 330-332。已压缩指令 360 为修改功能 A.3 364 以更包含资料及识别符（identifier）信息，其中资料是关于压缩中压缩指令撷取的数量，识别符信息是关于已压缩指令的数据域。

举例而言，以下为 Microsoft™DX 汇编语言指令

MUL R3.xyz, R4.yzw, R5.xyz

其可以经编译器展开成为以下三个指令

MUL R20, R12, R16

MUL R21, R13, R17

MUL R22, R14, R18

这些指令则可压缩成

FMUL.3 R20, R12, R16

需注意的是，在此压缩情况下是需要连续的暂存器位置，因此，如果码为

MUL R10.xzy, R10.xyz, R.yzw

时，将不会被压缩，因为它的暂存器位置不为连续。压缩一个不具有连续暂存器的指令时，编译器将再指派一个暂存器因此减少压缩

情况的优点。需注意此例子中的运算元可为向量（vector）、向量阵列（array of vector）或它们之间的组合。

现在参阅另一简略制作的图 4，其为一方块图用以显示指令处理的具体实施例。增加程序计数器 410 指向指令快取（instruction cache）的指令 420，其中指令快取包含特定执行线（process thread）相关的指令。就如图 2 所述，对于特定执行线，指令从指令快取提取（fetch）（步骤 430）且放置于指令暂存器（步骤 440）。一指令压缩发生在每一个循环（cycle）且指令被压缩至指令队列后接下来便可执行了。

如图 5 所示，一具体实施例中指令暂存器 510 及指令队列 520 配置成 128 位元，且 128 位元被分成两个 64 位元指令 512、514。一替代性的实施例也可配置成一不同大小的指令暂存器，其中暂存器在每一资料流（data string 数据流）可包含多于两个指令。因此分割的数目"N" 521 可为除了二以外之数。举例而言，在 128 位元系统利用少资料密集指令功能及引数（自变量）时，指令暂存器可包含四个 332 位元指令，如："N" 521 等于四。无论指令暂存器 510 的大小及分割 521 的数目为何时，每一个别的指令队列仅处理系统中的一个不同的执行线（process thread）。

每一分开的 64 位元指令队列 522，524 都有用于指令解压缩的功能。每一队列同时地加载对应的 64 位元指令。一实施例中，指令队列 520 有七个位阶（level）深，如："L" 526，528 为七。如果少于四个位阶也是可行的，如执行线结合了那些会暂停（stall）直到在每一队列中最少四个位阶都可以时的指令。四个位阶的需求是必要的，因为除了接收已压缩指令，指令队列 522，524 也储存多个指令，其为解压缩已压缩的指令。因此，指令队列 522，524 必须有容量储存已压缩指令功能外更要储存解压缩指令。

在一些已知的技术中，有一些指令队列配置不是七的，如上面所述的例子，如："L" 526，528 不为七。相同地，解压缩指令的最大数及因避免暂停的最小可能指令队列容量也可能都不为四。此数目的值是必须少于全部的位阶数"L" 526，528 及关联于在一特殊实施例中可压缩的指令的最大值。

参阅图 6，其显示此实施例的已压缩指令的资料格式。已压缩指令 600 利用栏 612 中位元 62 及位元 61 的值-去显示在解压缩所需求的复制数 610。复制数 610 对应于未压缩指令的数目，其中未压缩指令为经编译器压缩。此外这个值如以下所讨论是用以决定介于左及右指令队列的最大复制数。

位元 54 及位元 53 为定义数据域 622，其用以储存栏识别符 620。栏识别符 620 辨认指令是否适合压缩，如果已经压缩，它们的运算元引数 630 在解压缩时需修改。在此实施例中，引数 632、634 及 636 可以被定义为三个八位元栏，其为根据有多少引数为特定指令功能所需要。这将使不同的功能可以运算在每一指令的一或多个运算元引数。指令功能是要定义的，如在图 6 没讨论的其中一个栏。另外，在图中，X、Y 及 Z 标示着对应的引数 632、634 及 636 没有相关的坐标系统标示，且其目的仅用以区别在相同指令中每一引数与其它引数的不同。

使用上述已压缩指令的例子，复制数的值，如位元 62 及 61 可为"11"，以对应于由三个指令压缩成的一个已压缩指令的二进制表示式。因为运算为暂存器-暂存器方式，栏识别符 620 在位元 54 及 53 可为"01"。因此，运算元引数 630 中位元范围 46-39、19-12 及 7-0 将分别为暂存器 R20、R12 及 R16 的地址。指令功能码"MUL"将储存于位元范围 38-20 的栏。

一实施例中，栏识别符 620 可定义如图 7 的列表。此例子，为一具体实施例的压缩方法，而此结果仅当指令为暂存器-暂存器运算或暂存器-立即运算的压缩。限制指令的压缩于此两个运算型态将有较大的效益，因为压缩的方法依赖于运算元位于连续的地址暂存器。

现在参阅图 8，其为一流程图用以显示栏识别符资料如何决定在解压缩中哪一引数要被修正的具体实施例。如以上所讨论的，栏识别符包含位元 54 801 及位元 53 802。因为当两个位都为 0 或都为 1 时没有执行压缩，在位元 54 801 及位元 53 802 执行互斥或（XOR）810 功能用以决定结合了解压缩的复制是否需要。如果互斥或结果为零，然后就无复制 812 也就无需压缩。如果互斥或结果为“1”，然后就是有指令的复制 820。在此实施例中，对每一复制，引数 X 及 Z 在引数栏 632 及 636（图 6）作修改 830。更进一步说，如果位元 53 为"1" 830 然后在 Y 引数栏的值也作修改 850。相同的，如果位元 53 为"0"则在 Y 引数栏的值就不需修改 870。

使用上述的例子，运算栏识别符为"01"时，因此互斥或 810 功能将为一，在此表示指令的复制发生。因为无论栏识别符是否为"01"或"10"，引数储存在"X"及"Y"位置将被改变，互斥或为"1"要求此例子中的 R20 及 R16 引数必须改变。接着而来的解压缩指令改变 R20 引数以产生对应于在压缩之前的原始指令引数 R21 及 R22。对照的，R16 引数被修改以产生在接下来的解压缩指令的 R17 及 R18。另外，由于栏识别符的值为"01"，位元 53 的值为“1”，然后执行关系于引数 R12 的相应运算。因此位元 53 的位元测试区别了如图 7 所示的暂存器-暂存器运算及暂存器-立即运算。

图 9 显示指令解压缩的实施例的方块图。为达到对于左及右指令队列 910 指令的复制及修改，复制值为从压缩指令 920 读出。如先前讨论的，复制值的决定得视有多少个指令压缩成已压缩指令来决定，而复制值则决定了在解压缩时有多少个解压缩指令会产生。如上述的

码的例子，复制值可为三或"11"。同时，已压缩指令运算码及栏识别符被读出 930，940。已压缩指令运算码的读出是因为在复制的指令中它将不需修改而再产生。举例而言，在上述的例子中指令运算码在每一复制的指令将对应于"MUL"功能。

栏识别符被读取以决定在复制的指令中，哪一个运算元栏要被修改，其中复制的指令在解压缩时产生。如上述所讨论，栏识别符值"01"需要在复制的指令中修改所有三个运算栏。拥有复制数或拥有来自运算码及栏识别符资料的指令功能，指令的复制是与对应于引数栏每一次复制的增加 950。

决定在左及右指令队列的最大的复制数 960 中，此最大值通过在任何队列插入"无运算"功能以平衡左及右指令队列 970，其中复制数是少于最大值。因此，包含无运算指令，所有指令队列在每一循环（cycle）储存相同的指令数。此平衡功能是需要保持适当的指令序列，因为在一实施例中，左及右指令队列是在一单一数据块（single data block）加载及存取。

在实施例中，假设上述的已压缩指令有一复制值为三且在左队列，而另一个已压缩指令有一复制值为二且在右队列。将此两不同的复制值互相比较以决定最大值，而此例子为三。无运算指令数插入左指令队列及右指令队列为最大值（此例子为三）以及各自队列的复制值的差。因此，在此实施例中，左队列因为它具有最大的指令数，将不会有任何无运算指令插入。相对的，右指令队列将有一无运算指令队列插入，以使全部的指令数从二提升到三以符合最大指令数。

现在参阅图 10，其显示一指令解压缩过程实施例的方块图。128 位元指令暂存器 1001 由位元 127-0 定义，且逻辑地分割为两个 64 位元指令分别由 127-64 1002 及 63-0 1003 来定义。经分割 128 位元资料流（data string），指令暂存器同时地支持多执行线。在左及右指令队

列 1040，1050 的背景下，64 位元指令由位元 63-0 个别的定义。在一已知技术中可以知道指令暂存器可被分割成以适应大于二个的具有相同数据流的指令。

复制值储存于每一左及右指令的位元 62-61，而对于每一个别的指令队列 1040，1050 被读出 1012，1022。在每一队列的指令与相对应的修改 1010，1020 复制以及写入左及右指令队列 1040，1050。最大复制值 1012，1022 的决定是经由比较 1030 对每一左及右指令队列 1040，1050 的复制值 1012，1022。无运算指令数插入 1014，1024，其插入无运算指令数为左及右指令队列的最大复制数与对于每一相对指令队列复制数的差值。因此，在事件中，一指令有一较大的复制值大于其它的指令时，有较少的值的指令队列将插入一或多个无运算指令以平衡在每一队列的循环中加载的位阶数。换句话说，如果两个指令有相同的复制值，然后最大值为相同时则不需插入无运算指令。

经由插入无运算指令后，左及右队列在解压缩后得以平衡。平衡的队列因为解压缩在每一次循环执行一次而改善了效率。如：在队列中不平衡的 128 位元资料流可能引起暂停而影响所有结合了整个资料流的执行线。此种暂停的发生，起因于在解压缩时一指令队列的容量少于所需的容量。暂停的结果将影响所有经过此指令暂存器线（thread）的处理。

一复制及修改逻辑的具体实施例显示于图 11。已压缩指令 1100 包含一复制值其为在位元 62-61 1102，包含一栏识别符其为在位元 54-53 1104，包含多达三个不同引数栏 1106-1108 以及包含一运算码栏其包含指令功能 1110。复制数 1102 决定有多少个复制指令 1120，1140，1160 的产生。如上所述，复制值相对应于指令压缩至已压缩指令的数目。

栏识别符位 1104 选择性的致能（enable）8 位增量器（incrementer）1180，1182，1184 以修改由栏识别符值决定的引数。如一例子中，增量器 1180 致能且复制值 1102 为三，已压缩指令 1100 的位元 46-39 1106 的引数增加以产生对每一对应的复制指令 1120，1140，1160 的修正引数 1126，1146，1166。套用这个于上述例子，R20 引数将会增加以产生修正的引数 R21 及 R22。相同的，R12 也将增加以产生修正的引数 R13 及 R14，而 R16 增加以产生修正的引数 R17 及 R18。

尽管上述实施例使用三个或更少的复制数为例子，在已知技艺中可得知复制值及结合栏可以被修正为压缩一指令的最大数，其可能大于三或小于三。相同的，已知技艺中可知道上述讨论的指令格式仅作为示范性例子且这些方法的实行并不限制于此格式，因此不同数目的资料格式，容量以及栏的形式都可包含。

本发明的方法可被实现于硬件（hardware）、软件（software）、韧体（firmware）或它们之间的组合。在一较佳实施例中，压缩及解压缩逻辑是由软件或韧体实现，其做法为储存在内存且由一适当的指令执行系统来执行。如果使用硬件替代，其逻辑可由下列熟知的技术以任何的组合来实现。如具有逻辑门的数字逻辑电路（discrete logic circuit）以实现资料讯号的逻辑功能，具有适当组合逻辑门的特殊应用集成电路（ASIC），可程序门阵列（PGA）及现场可程序门阵列，（FPGA），等

任何处理描述或流程图的方块需了解其所表示的模块，区段或包含一或多个在处理中实现特殊逻辑功能或步骤可执行的指令码部分。可替代性的实现也可包含本发明较佳实施例的范围中，所显示及所讨论的功能执行顺序的替换，其可为本质的同时的或相反的顺序，此乃依据其所使用的功能且可从本发明这些技术中得到。

这里要强调的是上述本发明的实施例中，特别是在任何的"较佳"实施例中，仅为本发明实施的可行例子，也仅是帮助了解本发明原理。不同的变化或修改上述本发明的实施例而本质上没有脱离本发明精神或理论。这些变化或修改可包含于此揭露的范围以及本发明以及被接下来的申请专利范围所保护。

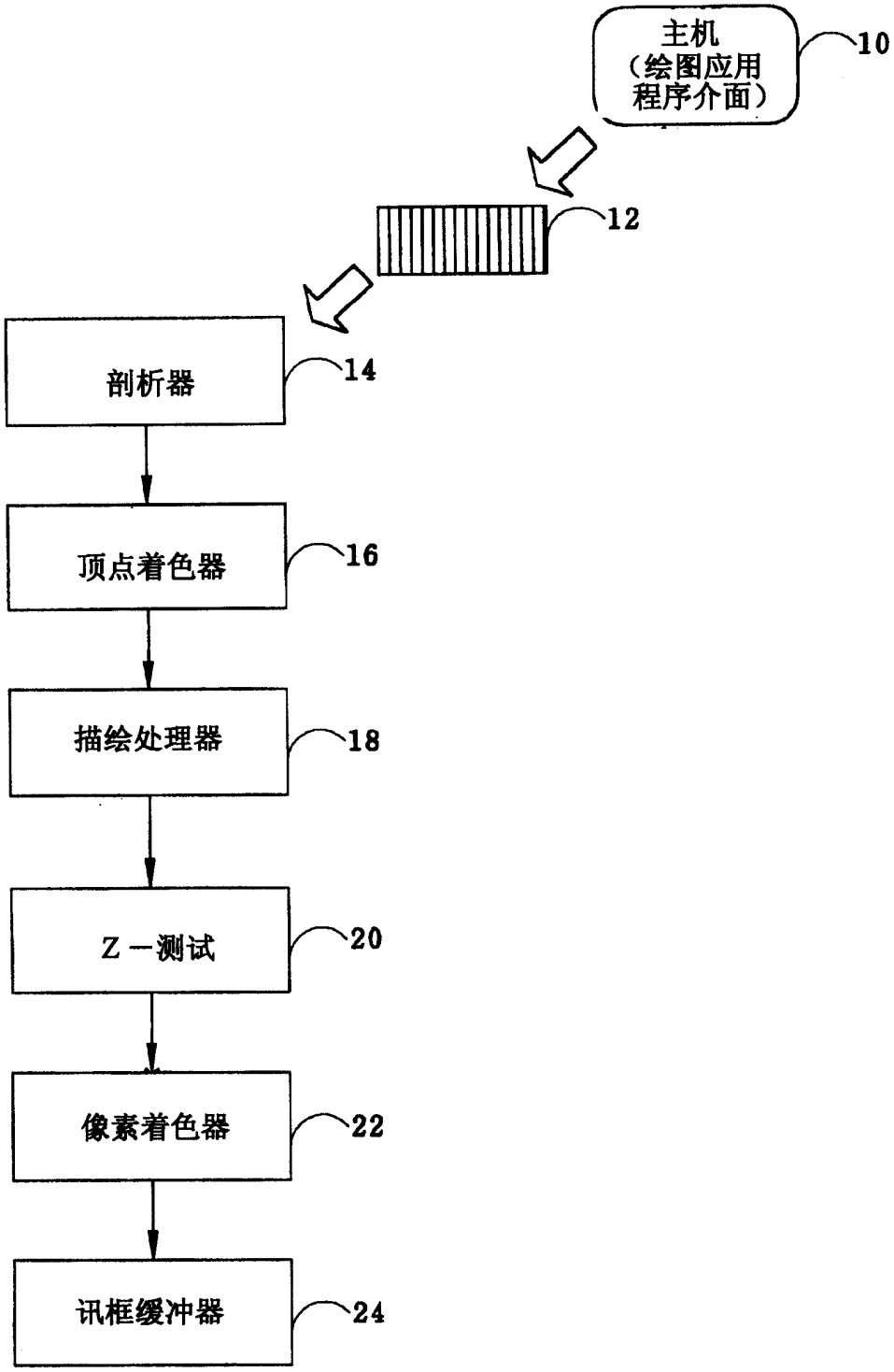

主机
（绘图应用
程序介面）
10
12
剖析器
14
顶点着色器
16
描绘处理器
18
Z－测试
20
像素着色器
22
讯框缓冲器
24

图1

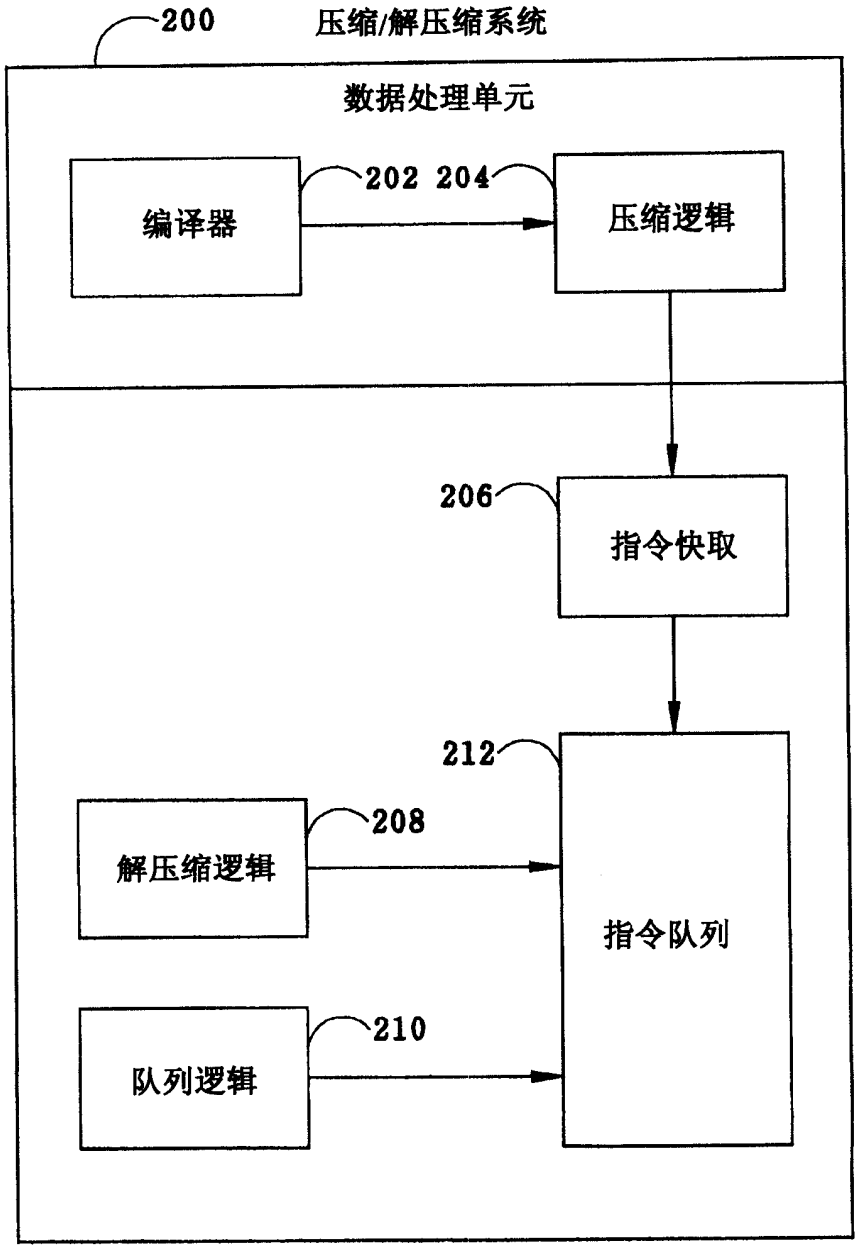

200
压缩/解压缩系统
数据处理单元
编译器
202 204
压缩逻辑
206
指令快取
212
解压缩逻辑
208
指令队列
队列逻辑
210

图2

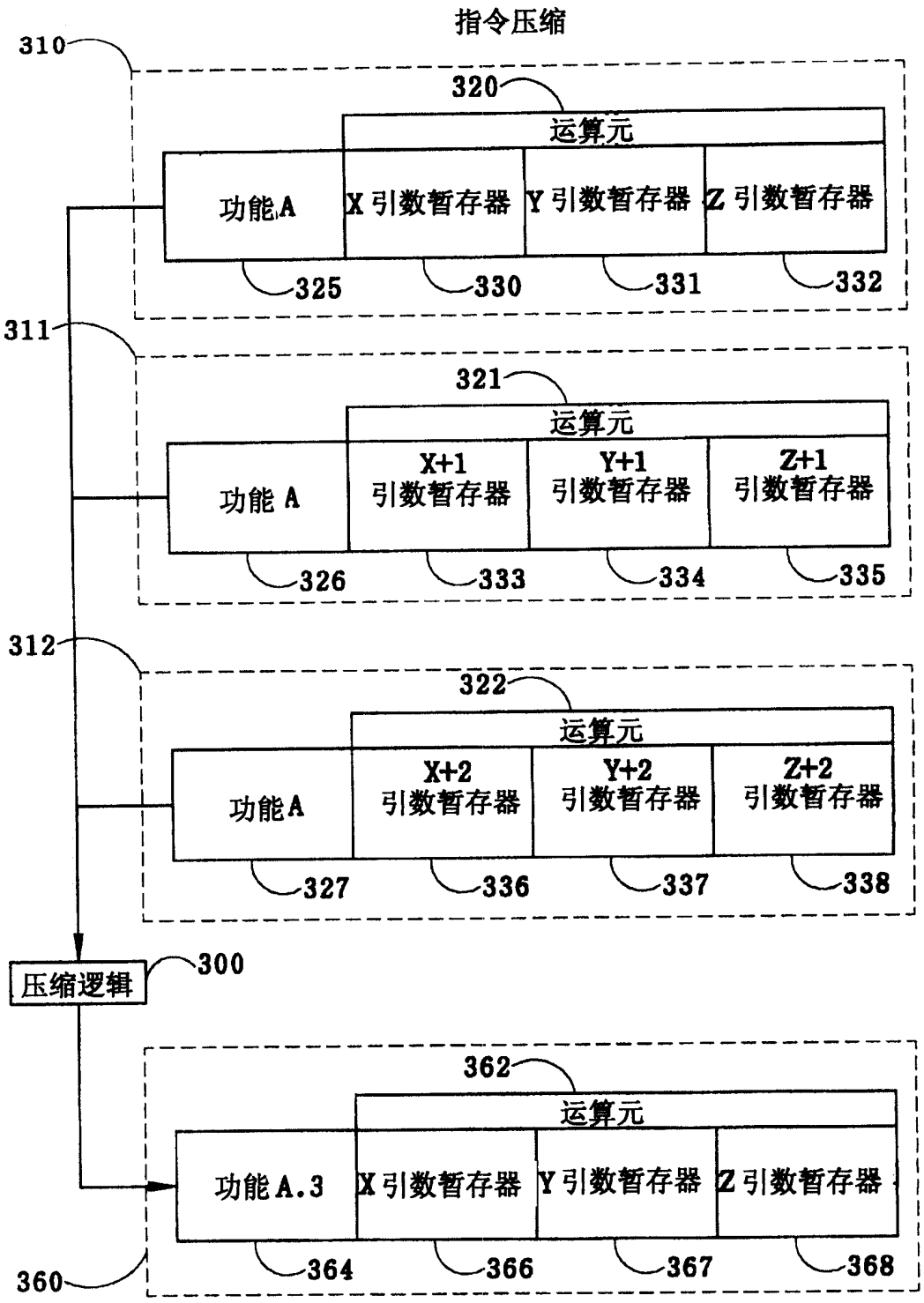

指令压缩
310
320
运算元
功能A
X 引数暫存器
Y 引数暫存器
Z 引数暫存器
325
330
331
332
311
321
运算元
功能 A
X+1
引数暫存器
Y+1
引数暫存器
Z+1
引数暫存器
326
333
334
335
312
322
运算元
功能A
X+2
引数暫存器
Y+2
引数暫存器
Z+2
引数暫存器
327
336
337
338
压缩逻辑
300
362
运算元
功能 A.3
X 引数暫存器
Y 引数暫存器
Z 引数暫存器
364
366
367
368
360

图3

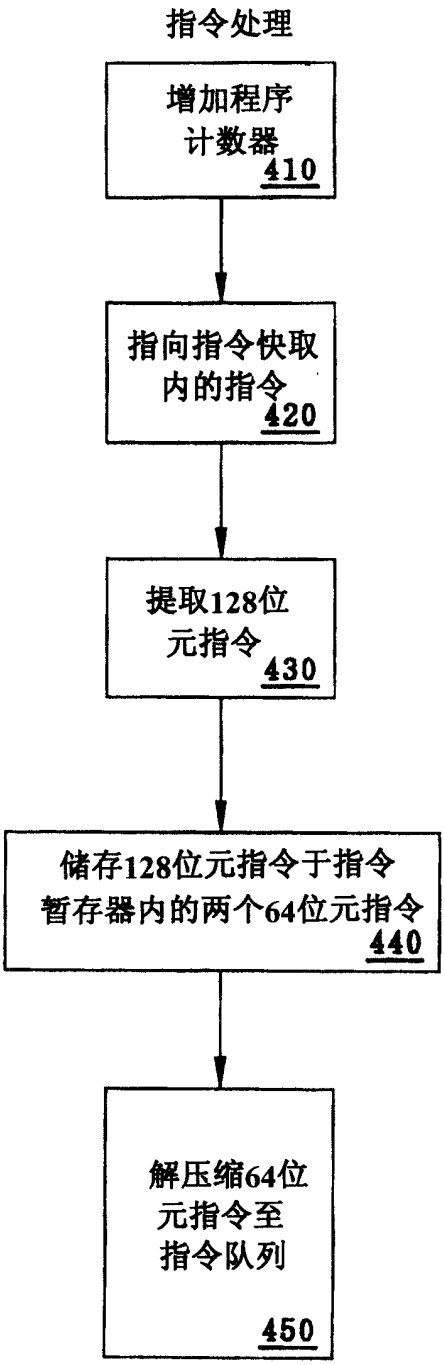

指令处理
增加程序计数器
410
指向指令快取内的指令
420
提取128位元指令
430
储存128位元指令于指令暂存器内的两个64位元指令
440
解压缩64位元指令至指令队列
450

图4

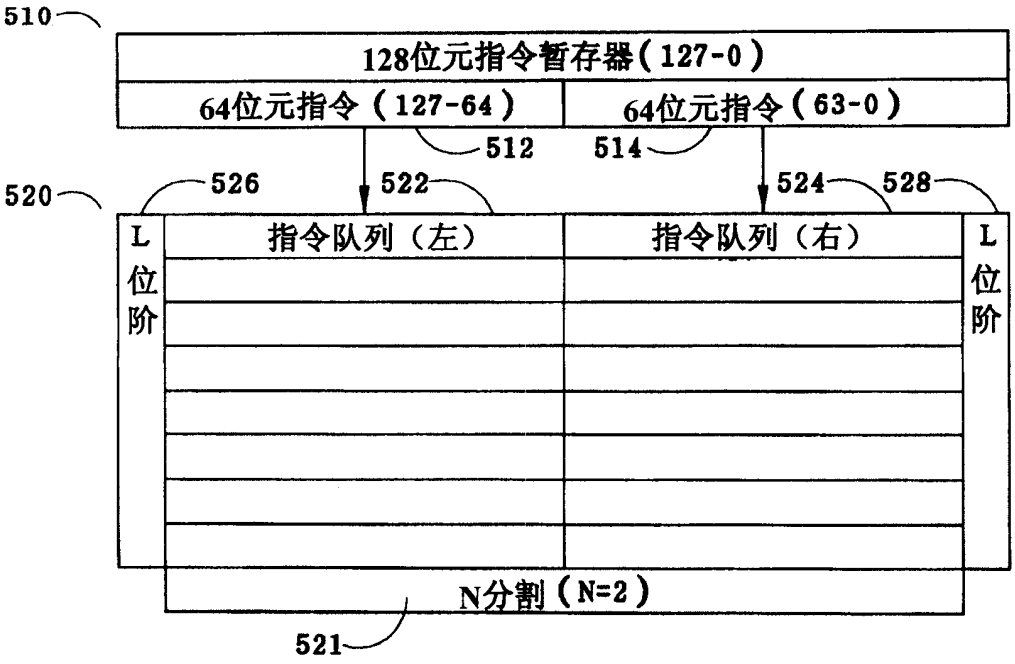

510
128位元指令暂存器(127-0)
64位元指令(127-64)
64位元指令(63-0)
512
514
520
526
522
524
528
L位阶
指令队列(左)
指令队列(右)
L位阶
N分割(N=2)
521

图5

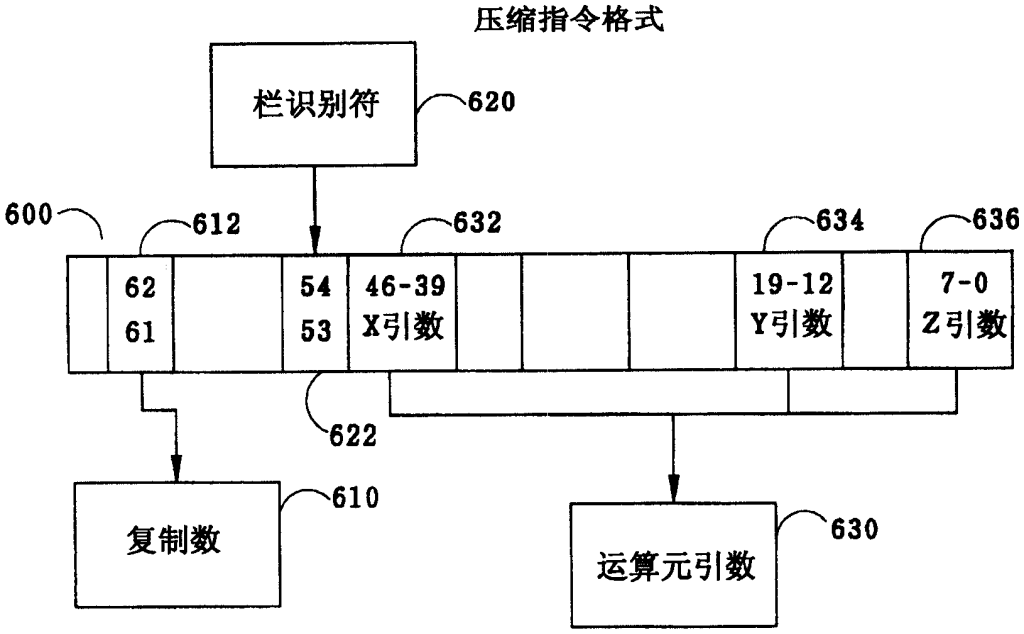

压缩指令格式
栏识别符
620
600
612
632
634
636
62
61
54
53
46-39
X引数
19-12
Y引数
7-0
Z引数
622
复制数
610
运算元引数
630

图6

位元 54	位元 53		
0	0	3个源运算	无压缩
0	1	暂存器－暂存器运算	压缩
1	0	暂存器－立即运算	压缩
1	1	分支运算	无压缩

图7

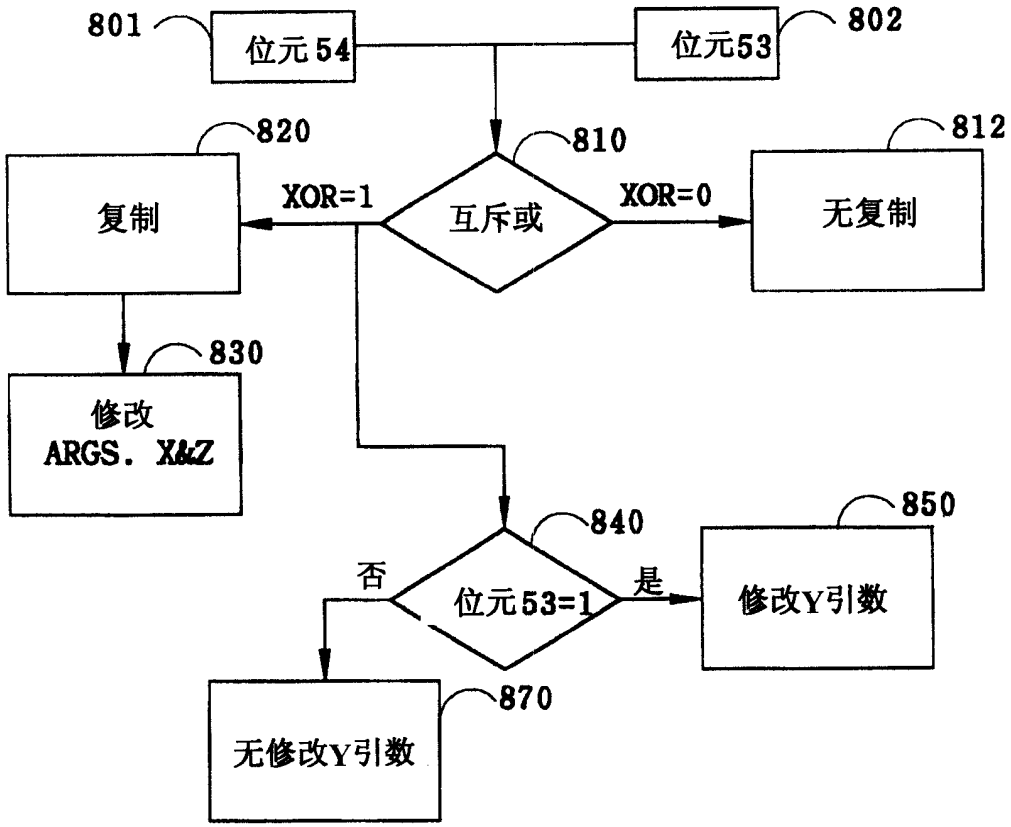

801
位元54
位元53
802
820
810
812
复制
XOR=1
互斥或
XOR=0
无复制
830
修改
ARGS. X&Z
840
850
否
位元53=1
是
修改Y引数
870
无修改Y引数

图8

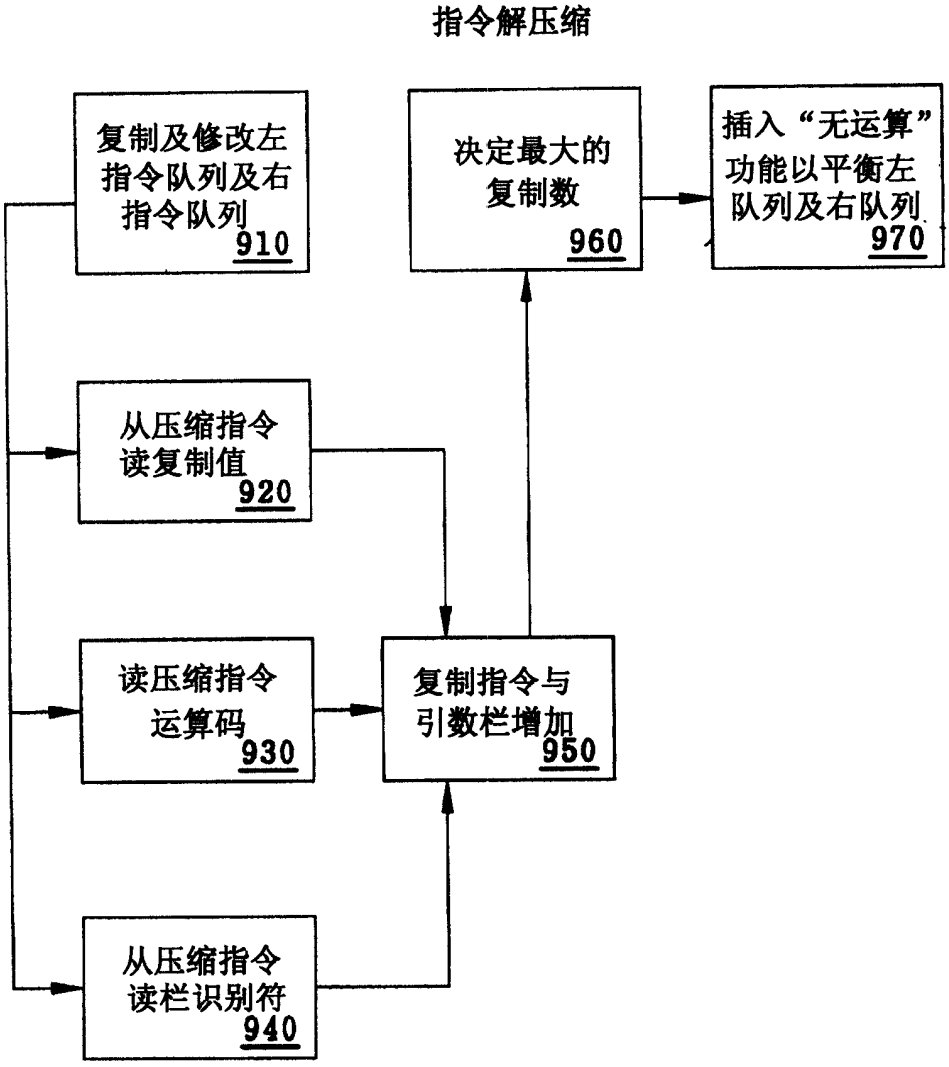

指令解压缩
复制及修改左指令队列及右指令队列
910
决定最大的复制数
960
插入“无运算”功能以平衡左队列及右队列
970
从压缩指令读复制值
920
读压缩指令运算码
930
复制指令与引数栏增加
950
从压缩指令读栏识别符
940

图9

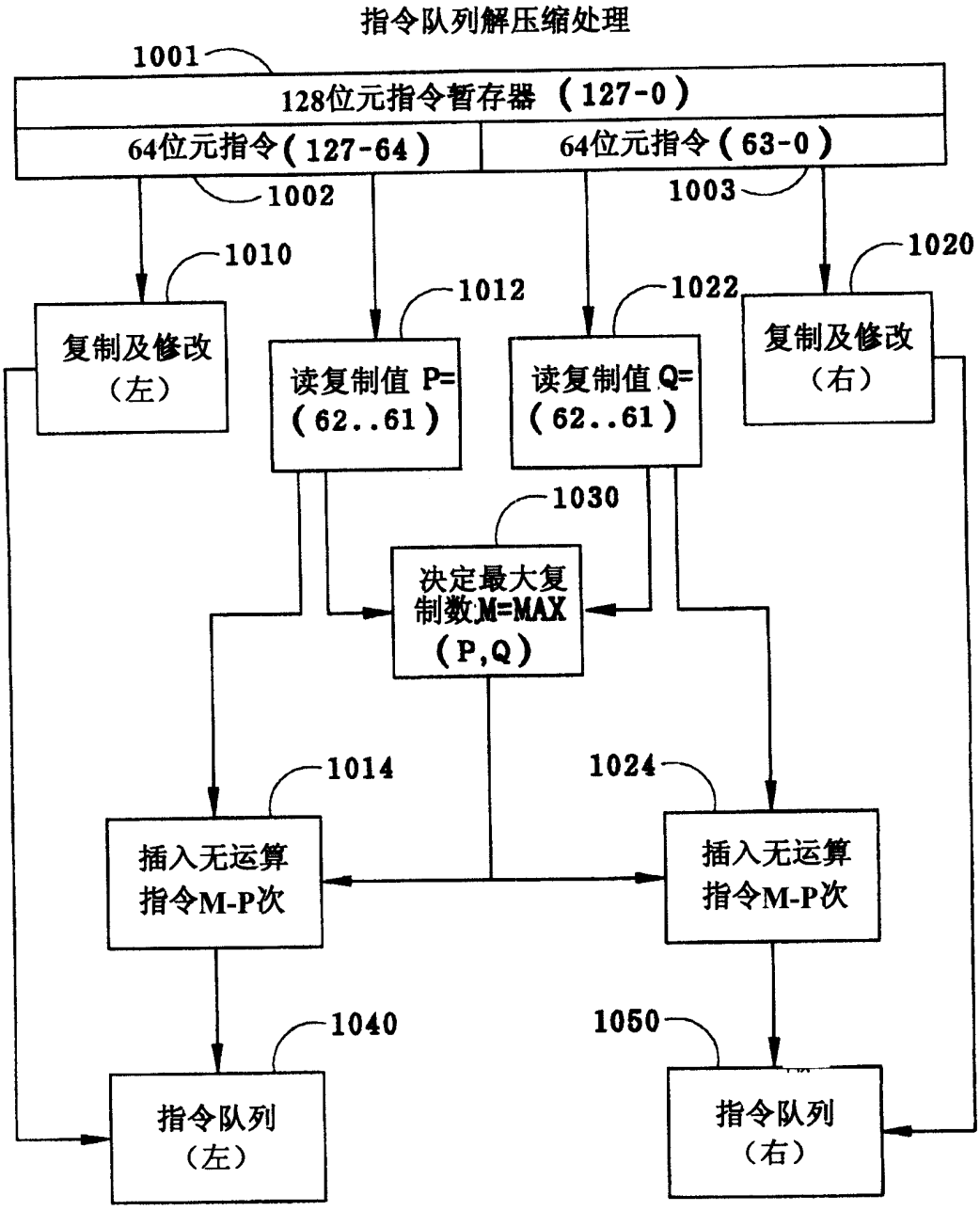

指令队列解压缩处理
1001
128位元指令暂存器（127-0）
64位元指令（127-64）
64位元指令（63-0）
1002
1003
1010
1012
1020
1022
复制及修改（左）
读复制值 P=（62..61）
读复制值 Q=（62..61）
复制及修改（右）
1030
决定最大复制数M=MAX（P,Q）
1014
1024
插入无运算指令M-P次
插入无运算指令M-P次
1040
1050
指令队列（左）
指令队列（右）

图10

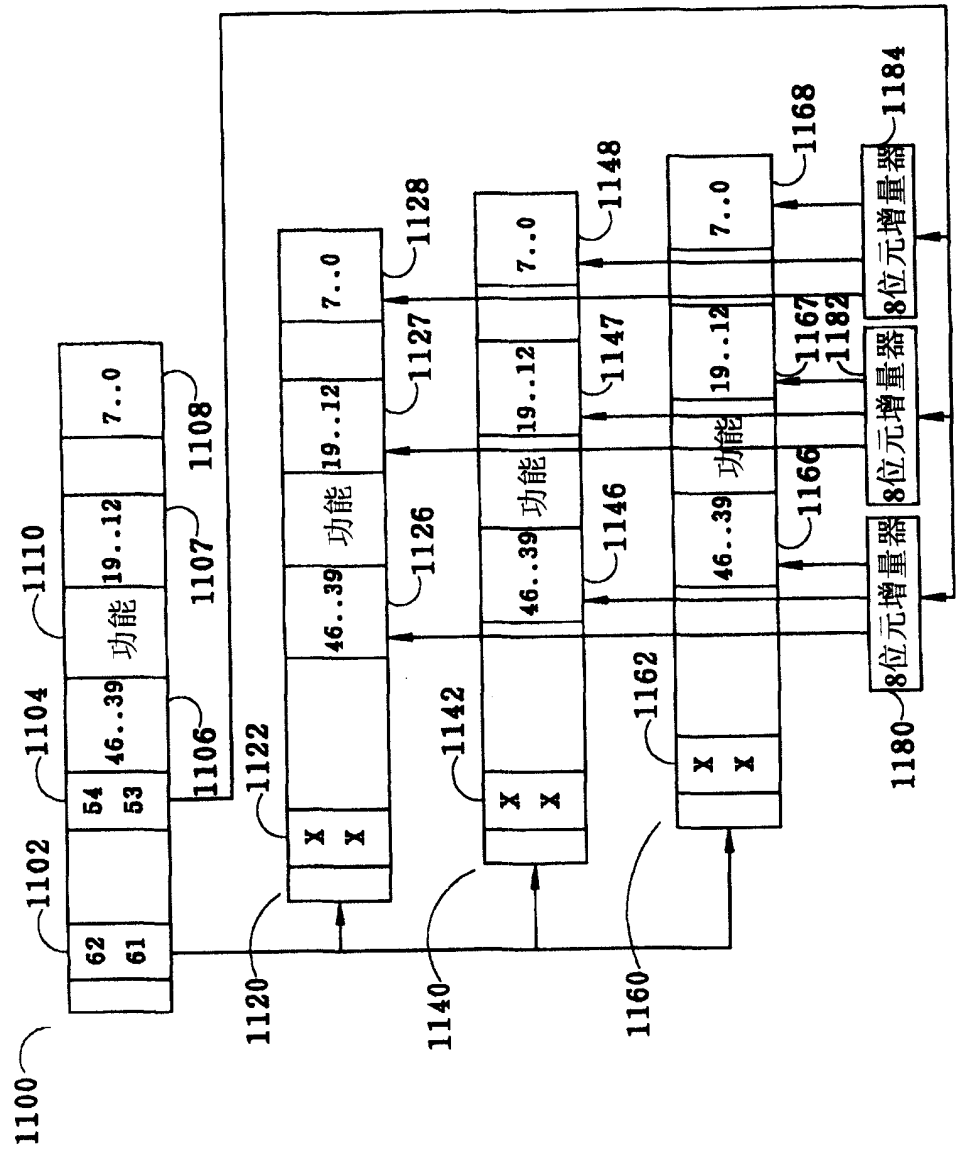

指令复制及修改
1100
1102
62
61
54
53
1104
46..39
功能
19..12
7..0
1110
1106
1107
1108
1120
1122
X
X
46..39
功能
19..12
7..0
1126
1127
1128
1140
1142
X
X
46..39
功能
19..12
7..0
1146
1147
1148
1160
1162
X
X
46..39
功能
19..12
7..0
1166
1167
1168
1180
8位元增量器
1182
8位元增量器
1184
8位元增量器

图11