[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 9/45 (2006.01)
G06T 1/20 (2006.01)

[12] 发明专利申请公布说明书

[21] 申请号 200610148697.2

[43] 公开日 2007 年 4 月 11 日

[11] 公开号 CN 1945536A

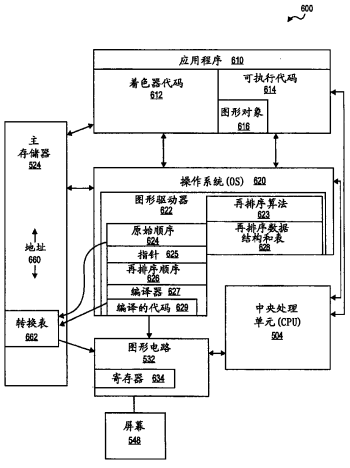
[22] 申请日 2006.9.29
[21] 申请号 200610148697.2
[30] 优先权
[32] 2005. 9.30 [33] US [31] 11/240984
[71] 申请人 英特尔公司
地址 美国加利福尼亚州
[72] 发明人 S·斯瓦米 O·海姆
[74] 专利代理机构 中国专利代理(香港)有限公司
代理人 王 岳 王 勇
权利要求书 5 页 说明书 18 页 附图 6 页

[54] 发明名称

利用指针来再排序指令的存储器布局

[57] 摘要

实施例包括在原始顺序的存储器中的地址处存储图形指令，并在存储器中存储与指向原始顺序的指令地址的每个指令相关的指针。 与第一图形指令相关的第一指针可以接着从指向第一图形指令的第一地址，被移动到指向第二图形指令的第二地址。同样的，通过在移动第一指针之前访问第一指针，与第二图形指令相关的第二指针可以接着从指向第二地址被移动到指向第一地址，以确定第二指针是指向第一地址的(例如，在被移动之前的第一指令指向的地址)。 然后，通过根据指针来转换它们到不同的地址，指令可以被再排序为优化的顺序，以用于编译。



1、一种方法，包括:

在存储器中存储多个地址上的多个指令;

在存储器中存储指向该多个地址的多个指针,每个指针与一指令相关;

从指向第一地址到指向第二指令的第二地址,移动与第一指令相关的第一指针;

从指向第二地址到指向第一地址,移动与第二指令相关的第二指针,

其中移动第二指针包括在移动第一指针之前访问第一指针,以确定第二指针是指向第一地址的。

2、权利要求1的方法，还包括:

在移动第二指针之前访问第二指针,以确定第一指针是指向第二地址的。

3、权利要求2的方法，还包括:

根据移动指令到再排序顺序的再排序算法,来选择访问第二指针以移动第一指针;

由于第一指针是根据第二指针移动的,所以选择访问第一指针以移动第二指针。

4、权利要求2的方法，还包括:

将原始顺序再排序为在多个地址的多个指令的再排序顺序,其中再排序包括:

将存储器中第一地址的第一指令移动到第二地址,其中移动第一指令包括访问第一指针;

将存储器中第二地址的第二指令移动到第一地址,其中移动第二指令包括访问第二指针;

实时（JIT）编译再排序的多个指令为由电路执行的编译代码。

5、权利要求2的方法，还包括:

从指向第三地址到指向第一地址,移动与第三指令相关的第三指针，其中移动第三指针包括访问第二指针，以确定第三指针是指向第一地址的;

从指向第一地址到指向第三地址，移动第二指针;

其中移动第二指针包括访问第三指针，以确定第二指针是指向第三地址的。

6、权利要求5的方法，还包括：

将原始顺序再排序为在多个地址的多个图形指令的再排序顺序，其中再排序包括：

将存储器中第一地址的第一指令移动到第二地址，其中移动第一指令包括访问第一指针；

将存储器中第二地址的第二指令移动到第三地址，其中移动第二指令包括访问第二指针；

将存储器中第三地址的第三指令移动到第一地址，其中移动第三指令包括访问第三指针；

实时（JIT）编译再排序的多个指令为由电路执行的编译代码。

7、权利要求2的方法，还包括：

从指向第三地址到指向第二地址，移动与第三指令相关的第三指针，其中移动第三指针包括访问第一指针，以确定第三指针是指向第二地址的；

从指向第二地址到指向第三地址，移动第一指针；

其中移动第一指针包括访问第三指针，以确定第一指针是指向第三地址的。

8、权利要求7的方法，还包括：

将原始顺序再排序为在多个地址的多个指令的再排序顺序，其中再排序包括：

将存储器中第一地址的第一指令移动到第三地址，其中移动第一指令包括访问第一指针；

将存储器中第二地址的第二指令移动到第一地址，其中移动第二指令包括访问第二指针；

将存储器中第三地址的第三指令移动到第二地址，其中移动第三指令包括访问第三指针；

实时（JIT）编译再排序的多个指令为由电路执行的编译代码。

9、权利要求1的方法，其中第一指针包括指向原始顺序的第一指令位置的第一原始位置指针，以及指向再排序顺序的第一指令位置的第一再排序位置指针；且其中第二指针包括指向原始顺序的第二

指令位置的第二原始位置指针，以及指向再排序顺序的第二指令位置的第二再排序位置指针。

10、权利要求9的方法，还包括：

将原始顺序再排序为在多个地址的多个指令的再排序顺序，其中再排序包括：

将第一指令从由第一原始位置指针指向的地址移动到由第一再排序位置指针指向的位置；以及

将第二指令从由第二原始位置指针指向的地址移动到由第二再排序位置指针指向的位置。

11、权利要求9的方法，其中第一指针还包括指向第一指令的第一指令指针，且其中第二指针还包括指向第二指令的第二指令指针。

12、权利要求9的方法，其中所述多个指令是多个图形指令，第一指令是第一图形指令，且第二指令是第二图形指令。

13、一种包含指令序列的计算机可访问介质，当执行指令序列时，使机器：

执行包括图形着色器指令的应用程序；

以指令的原始顺序在存储器中存储指令；

将原始顺序再排序为存储器中指令的再排序顺序，其中再排序包括：

移动多个指针，每个指针与一指令相关，其中移动包括从指向在存储器中的位置到指向在存储器中的不同位置，移动每个指针；

移动与指针相关的每个指令到不同的位置。

14、权利要求13的机器可访问介质，其中再排序包括选择第一指令和第二指令来转换在存储器中的位置；且其中移动包括：

访问与第二指令相关的第二指针，以识别在存储器中的第二位置移动到第一指针所指向的；

访问与第一指令相关的第一指针，以识别在存储器中的第一位置移动到第二指针所指向的；

从指向第一位置到指向第二指令，移动第一指针；

从指向第二位置到指向第一位置，移动第二指针；

15、权利要求13的机器可访问介质，其中指令的序列在执行时，

还使机器:

对于每个指令,在与指令相关的存储器中存储原始指针和再排序指针;

将每个原始指针指向与原始顺序的指令相关的地址;

根据算法和在指令中的信息,移动多个再排序指针;接着

根据再排序指针,将指令从原始顺序移动到再排序顺序。

16、权利要求13的机器可访问介质,其中指令的序列在执行时,还使机器:

实时编译指令的再排序顺序为编译的代码;

在图形处理单元(GPU)的寄存器中存储编译的代码;

在存储器中存储一张表,以将图形着色器指令转换为编译的代码。

17、一种系统,包括:

具有其中安装了操作系统的计算机单元;

具有多个地址的存储器,存储了:

原始顺序的图形应用程序的多个图形着色器指令,以及

多个指针,每个指针与一指令相关,且包括指向原始顺序的指令的第一地址的原始指针,以及指向移动到的指令的不同第二地址的再排序指针。

18、权利要求17的系统,还包括:

图形驱动器,根据再排序算法将指令移动到再排序顺序;以及

图形处理单元(GPU),具有用于存储编译代码的寄存器。

19、权利要求17的系统,还包括:

连接到存储器的处理器,用于执行在存储器中的操作系统;以及

实时编译器,用于根据再排序指针,在已经移动了指令后,将指令的再排序顺序编译为编译的代码。

20、一种系统,包括:

处理器;

第一存储器,用于存储由处理器执行的图形应用程序,该第一存储器连接到处理器;

图形硬件,用于根据存储在第二存储器中的图形指令的再排序顺序来显示图形,该图形硬件连接到处理器;

第二存储器，利用多个指针来存储要再排序的原始顺序的第一序列的指令，每个指针与一指令相关，且指向序列中的位置；以及

操作系统，使指针根据图形应用程序和再排序算法来指向不同的位置，以在编译指令之前，再排序指令为不同的第二序列。

21、权利要求20的系统，还包括：

图形编译器，在指令根据指针被再排序为第二序列之后，编译该指令。

22、权利要求21的系统，还包括：

图形电路和寄存器，用于根据编译的再排序第二序列的指令，来显示图形应用程序的图形。

利用指针来再排序指令的存储器布局

技术领域

本领域通常涉及在应用程序执行期间的实时编译图形着色器指令。

背景技术

计算机图形硬件的一个重要的测试是编程的执行速度。例如，如果图形硬件不能执行部分应用程序以实时显示更新的视频或图像，则会影响显示质量，例如不显得平滑以及要求观测器。特别地，在图像或视频数据的当前帧中出现的对象的运动可以具有适当的形状、颜色，或者可以具有到随后帧的“跳动”转换。

尽管各种因素都会影响程序的执行速度,但是一个最重要的因素是由图形着色器的实时（JIT）编译器提供的执行最优化水平。图形“着色器”可以被描述为图形着色器代码、指令、程序和/或软件的次序、序列、顺序、区域和/或部分。在很多情况中，可以获得的最优化水平很大程度上是由在编译前的物理存储器中的着色器代码指令的存储器布局确定的。在许多存储器布局中，在编译之前，指令的原始顺序按次序布局，在单个树结构中，和/或在“n元树数据结构”中。然而，这些布局不必提供JIT编译器的最有效顺序来将着色器代码编译为机器可执行代码。因此，希望可以从原始顺序（例如，在程序或应用程序中指令的顺序或次序）到更有效编译的再排序顺序来再排序着色器代码指令，同时减少执行这种再排序所要求的存储器的数量。

附图说明

从以下详细的说明、一组权利要求和相应的附图，各种特征、方面和优点会变得更加明显，其中：

附图1是利用着色器来显示图形对象的过程的流程图。

附图2是利用指针来再排序和编译图形着色器代码指令的过程的

流程图。

附图3是利用指针来再排序图形着色器代码指令的第一个例子。

附图4示出了利用指针来再排序图形着色器代码指令的第二个例子。

附图5是利用指针来再排序和编译着色器代码指令的系统。

附图6是简化的存储器着色器代码编译器。

具体实施方式

这里的描述包括一种利用指针来从原始顺序到再排序顺序再排序图形着色器代码指令的方法和存储器布局，以便编译器可以快速和有效地编译再排序指令为中间语言或机器语言代码。例如，附图1是利用着色器来显示图形对象的过程的流程图。附图1示出了利用着色器代码来显示应用程序的图形对象的过程100。在框110，加载具有可执行代码和着色器代码的应用程序，例如通过操作系统将应用程序存储在计算机设备的存储器中。设想的计算设备包括个人计算机（PC）、桌面计算机、计算系统、便携式计算设备、手持计算设备、电话、蜂窝电话、游戏设备、相关计算设备的互联网、服务器、数字视频磁盘（DVD）播放器、置顶盒、以及视频存储器和编辑设备。在某些情况中，计算机设备可以具有连接到处理器的主存储器、由处理器执行的操作系统，以及存储编译的着色器代码的指令在寄存器中以在监视器或屏幕上显示应用程序的对象的图形电路。

设想的具有着色器代码的应用程序包括具有一个或多个图形着色器的那些着色器代码；或着色器代码的部分，或区域。每个着色器可以包括着色器代码、着色器代码指令和/或着色器指令，例如在高级语言、文本或其它在工业中已知的着色器语言或协议中。着色器代码提供如何在监视器或屏幕上显示图形对象、形状、图元的形状、颜色、阴影等的“规则”或信息。例如，应用程序可以是游戏或其它软件程序或应用程序，它们具有随时间要在计算机屏幕或监视器上显示的图形对象以显得平滑，且吸引用户去查看那些图形。图形“对象”可以是部分的、全部的、或者包括图形、形状、颜色、底纹、帧、阴影、图元等。应用程序可以具有能够在许多操作系统

上执行或运行的可执行代码（例如，以机器语言），例如发送对象、形状、颜色等到图形驱动器、操作系统和/或要在监视器上显示的图形电路。应用程序还可以包括着色器代码，例如除了应用程序执行或运行的计算设备的图形电路之外的图形电路所特有的高级语言。这种着色器代码，或者包括这种着色器代码的应用程序，可以被描述为第三方代码或应用程序。

在框120，执行应用程序的可执行代码。例如，其中加载了应用程序的计算设备的处理器可以执行那个应用程序的可执行代码部分。可执行代码部分可以包括用于显示对象、形状或图元的信息。

在框130，图形对象和显示图形对象的着色器代码被发送到图形驱动器，例如作为计算设备的操作系统的一部分的驱动器。框130可以包括发送图形对象和用于显示该对象的图形着色器（或图形着色器代码）的处理器或操作系统（例如，根据框120的可执行代码）。图形对象和着色器代码可以与识别用于绘制该对象的着色器代码的消息和信息一起发送。这种图形驱动器的一个例子是通过加利福尼亚的Cupertino的Inter公司的Extreme Graphics Drive®。

在判定框140中，确定转换表是否相对代码中的着色器而存在。例如，框140可能包括图形驱动器或操作系统，其确定转换表是否已经事先被存储在主存储器中，以便转换着色器代码到编译的代码。同样，框140可以包括存储由以下相对于框160和170而描述的过程而产生的表。

如果框140确定转换表不存在，则过程继续到框160。在框160，着色器代码利用指针被再排序并被编译为编译的代码。指针可以是位置指针，例如指向在存储器中地址的指针，和/或指向着色器或着色器指令次序、序列、顺序、区域，和/或部分的位置的指针。同样，指针可以是识别或包含存储器中位置的地址的变量，例如哪个位置是分配的对象或分配的数据的开始点，例如对象、值类型、位或字节长度、或阵列的元素。变量可以是符号或代表值的名字，例如数值、字符、字符串、虚拟存储地址或物理硬件存储器地址。在某些情况中，指针可以指向数据结构，例如能够以简单阵列或顺序存储例如在存储器中的数据结构，且该数据结构能够被递增、更新或变

化，而不改变在其阵列或序列中的指针的位置。更新指针可以包括在现有的值上执行算术操作、创建新值，或者以新的值代替当前的或旧的值。

框160可以包括如这里所描述的利用软件、操作系统代码、数据结构、转换代码、存储器、指针、再排序算法、图形驱动器、编译器以及实时编译器，例如以下附图2-6所描述的。例如，着色器是以特定顺序的着色器来指令的序列。指令的原始顺序可以根据一算法来再排序，例如根据“Linear Scan Registration Allocation”，计算机科学实验室的Massimiliano Poletto，国际商用机器公司（IBM）的密歇根州技术学院（MIT）Vivek Sakar的Thomas J. 华盛顿研究中心，ACM Transactions on Programming Language and Systems（TOPLAS），第21卷，1999年9月提交的，第895到913页，ISSN: 0164-0925（这里“MIT算法”）。

再排序指令之后，着色器代码指令的再排序顺序或次序可以被编译，例如在显示该对象之前，通过实时（JIT）编译为由计算设备的图形电路执行的编译代码。例如，着色器代码可以是在通过实时（JIT）编译器编译为由例如图形芯片、图形卡、图形处理单元（GPU）等执行的图形硬件机器语言之前，被再排序为优化的或更有效顺序的着色器指令。编译的代码可以使用一种图形电路特定的语言（例如，机器语言，图形电路能够执行），同时未编译的着色器代码对于图形电路不是特定的或自然的（例如，不同图形电路的语言，或者当前图形电路例如在不用编译的情况下不能执行的语言）。同样要考虑的是，指令可以是着色器指令和/或图形程序的其它指令。图形程序可以被描述为具有指令的程序，其多数被指定为图形电路。框160可以包括存储转换表在例如主存储器、随机存取（RAM）存储器和/或高速缓冲存储器的存储器中。

在框170，包括用于转换着色器代码为编译的代码的信息的转换表被存储，例如存储在主存储器中。框170可以包括存储转换表在例如主存储器、随机存取（RAM）存储器和/或高速缓冲存储器的存储器中。在框170后，过程继续到框180。

可替换地，如果判定框140确定了转换表的确存在，则过程可以

进行到框150。在框150，利用转换表将着色器代码转换成编译的代码。框150可以包括利用如以上相对于框160和170描述而生成的转换表，或者转换着色器、着色器代码、着色器指令、着色器代码指令、或部分的着色器应用程序，以为图形电路编译代码。框150之后，过程继续进行到框180。

在框180，编译的代码被存储在图形电路的寄存器中。例如，框180可以包括图形驱动器或者操作系统，用于使编译的再排序着色器指令存储在图形电路的寄存器中，使得电路可以根据着色器“规则”来显示对象。图形电路可以是图形处理单元（GPU）、图形卡、图形芯片、图形硬件，例如以下相对于附图5的图形电路532描述的。图形电路可以是部分的计算设备。

在框190，图形对象是利用图形电路的显示。例如，图形电路可以根据存储在图形电路的寄存器中的再排序的编译图形指令来在监视器上显示图形对象。在框190后，过程100的处理会返回到框120。例如，在框190后，可以执行应用程序的附加可执行代码。随后，附加可执行代码会要求利用相同或不同的着色器或着色器代码来显示附加图形对象。因此，在框140，为了显示附加图形对象，如果着色器代码是在应用程序的执行期间事先经历的代码，则转换表可以存在于主存储器中（例如，如在框170描述的），且过程会继续到框150。可替换地，对于显示的附加图形对象，如果着色器没有被事先编译，则处理会继续到框160，其中着色器代码的指令可以被适当地再排序和编译。

根据某些实施例，框170和框150是可选的，例如其中在每种情况下，图形代码指令被再排序和编译，以显示图形对象。

在某些情况下，例如为了利用算法（例如，上述MIT算法）来执行再排序，通过存储两组指令在存储器中并再排列第二组指令的顺序，同时保持第一组指令的原始顺序作为参考，图形指令可以被再排序。然而，通过利用指针来追踪每个指令的再排序或移动（例如，以下对于附图2的框225的描述），可以获得更多存储器的有效的实施例。接着，当移动或再排序完成时，可以根据指针将指令移动为再排序顺序。再排序顺序可以接着被编译，例如通过JIT编译器（例

如，以上对于附图1的框160的描述）来编译。

例如，附图2是利用指针来再排序和编译图形着色器代码指令的过程的流程图。附图2是过程200的流程图，其可以是也可以不是如以上对于附图1的框160所描述的过程，例如用于再排序和编译图形着色器代码指令的过程。在框210，图形指令以原始顺序被存储在存储器中的地址上。例如，框210可以包括存储着色器代码的图形指令，如以上对于附图1的框110和130所描述的。图形指令可以是向量指令、着色器代码指令或由计算设备所使用的用来绘制图形的一部分着色器的其它指令或信息，如以上对于附图1的框110到190所描述的。

同样，在存储器中的地址可以是物理地址、虚拟地址、在存储器中的位置或以一种顺序的位置，例如应用程序中的指令序列的原始顺序，或者应用程序中包括着色器以及具有对象的图形驱动器的着色器代码的部分，如对于附图1的框130所描述的。还要考虑的是，框210可以包括从应用程序以原始顺序来读取图形指令（例如，对于附图1的框110-130所描述的）。

在某些情况中，存储器的地址可以通过逐行地读取着色器程序的行，并对于程序中的每一行检查指令中使用的寄存器，以及如下述方法填充间隔表来被填充。如果寄存器是目的寄存器，则将新寄存器元件加上当前行号以作为开始行。如果寄存器是源寄存器，则在间隔表对象中寻找寄存器的实例，并用当前行号更新其末行条目。当程序的所有行都已经处理时，所有的寄存器的生命期可从间隔表中得到，并且硬件寄存器能够以线性方式被分配。

例如，应用程序的着色器或着色器代码部分的每一行可以包括以单独的指令结构存储的指令操作码、目的寄存器和源寄存器。特别地，每个着色器指令可以包括操作码（“OPCODE”）、目的寄存器（“DST”）、第一源寄存器（“SRC0”）、第二源寄存器（“SRC1”）和第三源寄存器（“SRC2”）。同样，每个指令可以被包含在指令结构中，例如在包括以原始顺序存储在存储器的地址上的图形指令和指向在存储器中的指令地址和顺序的指令位置的指针的结构中，其中每个指针与指令之一相关。指针可以包括指向原始顺序的指令

地址的原始位置指针，以及指向在根据一算法移动后指令的地址或位置的再排序位置指针，例如该算法按照算法标准根据指令信息（例如操作码、目的寄存器和源寄存器信息）来转换两个指令的地址。在指令被存储、移动或再排序在存储器中之前，再排序位置指针可以被初始化为“NULL”、非活动的、或逻辑低（例如，逻辑“0”）。可替换地，在存储指令期间或之后，再排序位置指针可以被设置为指向原始位置指针地址。还可以预料的是，原始位置指针或指针的附加或第三指针可以包括执行自身指令的指针。

在框220，指向存储在存储器中的图形指令的地址的指针还被存储在存储器中。例如，框228可以包括将如上面对于框210的描述的指针存储在存储图形指令的同一存储器中，其中每个指针与指令之一相关。因此，每个指针可以包括对应于、或相关于、或配对于、或用于单一的图形指令的一个或多个指令。与指令相关的每个指针可以被描述为包括以下中的一个或多个：指向指令的指针；指向原始顺序的存储器中的地址或指针的位置的原始位置指针；以及指向地址或位置的再排序位置指针，指令按再排序顺序被移动到所述地址或位置从而被编译为编译的代码，例如以上对于附图1的框160所描述的。再排序位置指针可以指向从初识位置（例如NULL、逻辑“0”或与原始顺序中的位置相同的地址）移动再排序位置指针到再排序顺序的最后地址的过程期间而更新的地址或位置，例如根据算法（例如根据上述MIT算法）或其它选择标准（例如选择要按原始顺序转换的各种地址，直到获得期望的再排序次序），在指针位置的移动、更新或再排序期间。

接着，在框225，选择第一指令和第二指令来转换地址（例如，根据上述的MIT算法）。例如，框225可以包括选择指令，以使它们在存储器中的地址被转换，或使指令的次序或顺序中的位置被转换，例如在再排序或移动指令或指向指令的指针期间、在从原始顺序移动指令到再排序顺序的过程期间、在编译指令和编译代码或用于显示图形之前（例如，见附图1）。

在框230，访问与第二图形指令相关的第二指针的第二地址，例如被存储在存储器中，或者被保存以便稍后在过程200中使用。框230

可以包括访问第二地址，以确定或识别指向第一指令的第一指针被移动到第二地址，以便影响该转换。可以理解的是，第二地址应该是在移动第二指针到不同地址之前被访问，例如引起指向第一地址的第二指针更新的移动（例如，见以下附图3的行322）。

在框240，访问与第一图形指令相关的第一指针的第一地址，例如被存储在存储器中，或者被保存以便稍后在过程200中使用。框240可以包括访问第一地址，以确定或识别指向第二指令的第二指针被移动到第一地址，以便影响该转换。可以理解的是，第一地址应该是在移动第一指针到不同地址之前而被访问，例如引起指向第二地址的第一指针更新的移动（例如，见以下附图3的行320）。因此，可以根据（或包括访问）在移动指针之前的第一和第二指针，来描述用于转换指针或指令的再排序顺序地址或位置的第一和第二指针的后续移动，从而影响框255的转换。

在框250，第一指针从第一地址移动到所访问的第二地址。框250可以包括从指向例如NULL、逻辑“0”的第一地址或原始顺序的第一指令的地址到指向第二图形指令的地址或位置，来移动或更新由与第一图形指令相关的指针（例如，再排序顺序指针）指向的位置的地址，以按次序或顺序，例如存储在存储器中的顺序转换第一和第二图形指令的位置或地址。

在框260，第二指针从第二地址移动到所访问的第一地址。框250可以包括从指向例如NULL、逻辑“0”的第二地址或原始顺序的第二指令的地址到指向第一图形指令的地址或位置，来移动或更新由与第二图形指令相关的指针（例如，再排序顺序指针）指向的位置的地址，以按次序或顺序，例如存储在存储器中的顺序转换第二和第一图形指令的位置或地址。

在框270，确定指令的转换是否完成了。框270可以包括确定着色器的指针的位置的转换、或者应用程序包括着色器代码的部分（例如，以上附图1的框110到130和框210到220所描述的），或者其相关的指针是否通过所期望的再排序顺序被移动、更新或再排序，例如根据一种算法（例如，上述MIT算法，以上附图1的框160和框225）。

如果在框270指令的转换没有完成，则过程200会返回到框225，

以便可以转换两种以上的指令（其可以包括，也可以不包括最新转换的第一和/或第二指令），如以上对于框225到260所描述的。例如，在框270后，当返回到框225时，与第三图形指令相关的第三指针可以从指向第三地址到指向第一图形指令的第二地址而被移动（例如，在框225的第二选择期间包括选择第三指令和第一指令，来转换地址）。移动第三指针可以包括访问第一指针以确定第三指针是指向第二地址的（例如，已经转换了第一和第二指针，且现在指向这些其它的地址）。同样，从指向第二地址到指向第三地址移动第一指针可以包括访问第三指针以确定第一指针是指向第三地址的。

可替换地，如果在框270指令的转换完成，过程继续到框280。在框280，原始指令可以被再排序以根据指针来读取指令的顺序。框280可以包括通过从由原始位置指针指向的地址或位置移动每个具有指针的指令到由再排序位置指针指向的地址或位置，来再排序指令的原始顺序为存储器中的指令的再排序顺序或指令的再排序次序。可以理解的是，某些指令以顺序或次序移动，而其它指令可以从远离它们的位置按原始顺序移动1、2或任意数量的地址或位置。此外，在某些情况中，尽管与指令相关的指针（例如再排序位置指针）可以被多次移动或更新，但是仅需要指令被再排序或移动一次，例如移动到由再排序顺序指针定义的最终地址或位置。可以理解的是，这可以提供对于处理资源和时间更有效的再排序。

在框290，图形指令的再排序顺序可以例如通过JIT编译器被编译，以编译代码。编译的代码可以被编译为机器语言，例如图形硬件机器语言，由计算设备的图形电路、图形处理单元（GPU）、图形卡、图形芯片、图形硬件（例如，以下对于附图5的图形电路532所描述的）来执行。

如以上对附图1描述的，在某些实施例中，代替利用指针，可以预想的是，两组图形指令可以原始顺序被存储在存储器的地址。在那些情况中，在框225选择之后，在存储器中的第二组指令可以如所选择的那样被移动，而不利用指针。然而，根据指令的大小，可以理解的是，指针的利用可以节约存储器的存储空间。例如，在每个指令包括四个四字节部分（例如，操作码、DST、SRC1和SRC0）的情

况中，每个指令可以要求在存储器中的16字节。对于两组图形指令被存储在存储器中的地址（例如“静态编译过程”）以再排序指令的情况，指令从原始顺序移动到再排序顺序所必需的存储空间可以是指令的16字节加上1字节，以识别每组指令存储器的指令，对于在存储器中的两组指令，乘以2。因此，在每个指令的静态编译过程中，每个指令需要34字节，以使用静态编译过程的n元树型数据结构。

可替换地，在一个实施例中，利用在相同的16字节指令上的指针，其中原始位置指针使用4字节，且再排序位置指针也使用4字节，每个指令仅仅需要24字节来存储指令和指针（例如，用于指令的16字节，加上用于指针的8字节）。因此，参考附图1和2所描述的过程对于每个指令可以节约10字节的存储器存储空间，同时再排序指令为优化的（例如，为根据算法或过程的优化顺序，例如上述的MIT线性扫描算法），或用于编译，且在显示图形对象之前的JIT编译期间（例如，对于附图1的框190的描述）。可以理解的是，这种存储器存储空间的节约会乘以在着色器中的着色器指令的数量，且对于节约存储器存储空间、增加性能会变得更重要，且在再排序前和/或编译着色器代码期间，可以有效使用处理器、电源和时间。

利用指针来再排序着色器指令的另一个效用是对于JIT编译器能够在编译前确定需要为着色器、着色器代码或在应用程序中的部分着色器代码分配的寄存器的最少量。这是很重要的，因为减少了使用的寄存器的数量，通常增加了性能、速度，并减少了需要的处理资源。特别地，编译的代码可以存储在图形电路的寄存器中。因此，除了提供更多的存储器有效实施例，利用指针的实施例还提供了更有效的设计和预测存储编译的代码所必需的图形电路的寄存器的数量。例如，期望最小化或事前得知对于存储编译的代码在图形电路中所需的寄存器的数量，例如以减少图形电路的硬件需要，或允许用更多可预测的寄存器数量来设计的图形电路。此外，在某些情况中，可以限制在图形电路中存在的或可用的寄存器的数量。在这些情况下，在编译前、编译期间和编译后由着色器的编译代码使用的寄存器的已知数量可以允许操作系统、图形驱动器或图形电路来保存更少的图形电路的寄存器，由此提供对可用电路的寄存器的有限

总量的更有效和优化的使用。可以理解的是，可以注册不存在或使用计算系统的存储器、主存储器或高速缓冲存储器的图形电路的寄存器。特别地，寄存器可以仅仅作为部分的图形电路存在（例如，见以下对于附图5的图形电路532所描述的图形电路的说明）。

附图3是利用指针来再排序图形着色器代码指令的第一个例子。附图3示出了包括表302和转换305的例子300。表302作为地址1、地址2和地址3的存储器的一个例子。以原始顺序304示出了指令A、B和C，其中指令A是在地址1或位于地址1，指令B是在地址2或位于地址2，指令C是在地址3或位于地址3。表302还示出了再排序顺序306，例如在转换在存储器地址或位于存储器地址的指令的地址、位置或顺序后获得的顺序。特别地，再排序顺序306具有在地址1的指令B、在地址2的指令C和在地址3的指令A。

转换305示出了选择转换308、指令A指针370、指令B指针380和指令C指针390的列。例如，在转换305的行311，指针370、380和390指向原始地址，其指令A、B和C是以原始顺序304。特别地，指针370包括指向地址1的原始位置指针371，以及也指向地址1的再排序位置指针372。可替换地，如以上对于附图所指出的，指针372可以初始为NULL、非活动的、“0”或其它值。同样，指针380包括原始位置指针381和再排序位置指针382，都指向了地址2。接着，指针390包括原始位置指针391和再排序位置指针392，都指向了地址3。指针382和392可替换地被设置为NULL、非活动的、“0”或其它值，如对于指针372所描述的。

可以理解的是，例子300可以对应于以上对于附图1和2的说明。例如，指针370可以是第一指针，指针380可以是第二指针，且指针390可以是第三指针，如以上对于附图2的框220到270所描述的。类似地，指针371、381和391可以是原始位置指针，且指针372、382和392可以是再排序位置指针，如对于附图2的框210到280所描述的。此外，再排序顺序306可以是要编译的指令的再排序顺序，如以上对于附图1的框160和附图2的框290所描述的。

在行312，选择了以指令的顺序或次序来用指令B转换指令A（例如，在这点上，它是从原始顺序304到下一顺序的初始或第一转换，

其可以包括，也可以不包括转换，例如下一顺序是要编译的再排序顺序）。为了转换指令，在行314，访问了由指令B指针382指向的地址。在行314，这个地址是2。

在行316，访问了由指令A指针372指向的地址。在这点上，这个地址是1。在行320，指令A指针372被移动到所访问的指令B指针382（其是2）。因此，在行320，指针372被示出了从1变化到2。

接着，在行322，指令B指针382被移动到所访问的指令A指针372，其是1。通过将指针382从2变化到1在行322上示出了这种移动。行331示出了在第一转换（在指令A和B之间的转换）之后指向的地址。

在行332，选择了在指令A和指令C之间的转换。在行334，访问了由指令C指针392指向的地址（具有当前值3）。在行336，访问了由指令A指针372指向的地址（具有当前值2）。在行340，指令A指针372被移动到访问的指令C指针392（其是3）。通过将指针372从2移动到3在行340上示出了这种移动。

在行342，指令C指针392被移动到所访问的指令A指针372（具有值2）。通过将指针392从3移动到2在行342示出了这种移动。在行351，示出了指针的再排序地址。

行360包括根据再排序指针（例如在行351的再排序地址）来移动指令。对于指令A，示出了从地址1到地址3的移动。对于指令B，示出了从地址2到地址1的移动。对于指令C，示出了从地址3到地址2的移动。因此，指令A被移动或再排序到地址3，B到地址1，C到地址2，如再排序顺序306所示。

还需要指出的是，由于在行336，在移动在行320的指令A指针372之后，以及在移动在行340的指令A指针372之前，访问由指令A指针372指向的地址，因此在行342正确地发生了指令C指针392到指令A的地址2的移动。无需在此时访问这个指针，直观地，指令C可以已经被移动到地址1，以原始顺序的指向指令A的指针372的地址；或者地址3，在行340后指向地址A的指针372的地址。

在某些实施例中，每个指令仅需要使用一个指针。在这些情况中，指令的指针或与指令相关的指针可以被描述为或定义为再排序位置指针。因此，在单个指针实施例中，对于指令A，指针是指针372，

对于指令B是指针382，且对于指令C，指针是指针392。可替换地，实施例可以被描述为指针370包括指针372和至少一个其它的指针；指针380包括指针382和至少一个其它的指针；以及指针390包括指针392和至少一个其它的指针。在这个可替换的实施例中，其它指针可以是原始位置指针（例如指针371、381和391），指向指令的指针（例如，以上附图2所描述的实际的16字节），和/或期望的其它指针。

与对于附图2的框280描述相似的，在行360的指令的移动可以根据指针370、380和390或根据指针372、382或392来描述。

附图4示出了利用指针来再排序图形着色器代码指令的第二个例子。附图4示出了包括表402和转换405的例子400。可以理解的是在行440，当转换指令B指针482时，在行434通过访问由指令B指针482指向的地址，指令C的指针492被正确移动到地址1。否则直观地，由于指令B在原始顺序的地址2，因此通过转换在地址2和3处的指令，这可能被错误地转换（例如，用指令A的位置转换指令C的位置），则可能产生错误。

附图5是利用指针来再排序和编译着色器代码指令的系统。附图5示出了系统500，例如计算机系统、计算设备、计算机单元、具有处理器和存储器的机器或系统、或其它计算设备，如以上对附图1的框110-130所描述的。参考附图5，系统500包括处理器或中央处理单元（CPU）504。CPU 504可以包括用于执行计算的算术逻辑单元（ALU）、用于临时存储数据和指令的寄存器集合、以及用于控制系统500操作的控制单元。在一个实施例中，CPU 504包括x86、奔腾或其它处理器（例如，在现有技术中已知的PC）的任意一种。CPU 504不限于微处理器，但还可以采用其它形式，例如微控制器、数字信号处理器、精简指令集计算机（RISC）、特定用途集成电路等。尽管示出了一个CPU 504，系统500可以可替换地例如通过包括协处理器来包括多处理单元。

借助CPU总线508，CPU 504被连接到总线控制器512。总线控制器512包括集成其上的存储控制器516，通过存储控制器516可以是在总线控制器512的外部。存储控制器516提供了一种接口，用于通过存储总线520由CPU 504或其它设备访问系统存储器524。在一个实施例

中，系统存储器524包括随机访问存储器（RAM），例如同步动态随机访问存储器（SDRAM），且可以包括高速缓冲存储器。系统存储器524可选地包括任意附加的或可替换的高速存储设备或存储电路。系统存储器524可以被描述为系统500的“主存储器”，例如存储要由处理器（例如CPU 504）执行的应用程序，或执行操作系统（例如，从大容量存储设备552安装和加载的）的存储器。

总线控制器512被连接到可以是外围部件互连（PCI）总线、工业标准结构（ISA）总线等的系统总线528。连接到系统总线528的是图形电路532、大容量存储设备552、通信接口设备556、一个或多个输入/输出（I/O）设备568_1-568_N和扩展总线控制器572。图形电路532被示出连接到屏幕548，例如用于显示图形、图形对象、图像、帧、视频等的计算机屏幕或监视器。

根据某些实施例，图形电路532可以是图形处理单元（GPU）、图形处理器、图形卡、图形芯片、图形电路、图形硬件、图形控制器或图形引擎。图形电路532可以包括在一个或多个载体衬底上的一个或多个硬件芯片、印刷电路板（PCB）、母板或其它电路支持硬件。作为电路，图形电路532可以包括各种开关、控制器、处理器、驱动器、晶体管、电阻器、电感器、电容、互连和其它适合的设备和结构以执行如这里对于附图1-4和6所描述的功能。

图形电路532被连接到视频存储器536（例如，8兆字节）和视频BIOS 540，所有这些可以被集成到单个卡或设备上，如数字544指定的。视频存储器536和视频BIOS 540包括用于控制图形电路532的代码或视频服务。在另一个实施例中，图形电路532通过加速图形接口（AGP）总线被连接到CPU 504。图形电路532提供了视频信号到显示屏幕548。例如，电路532可以发送图形图像、对象或帧，以显示在由个人或系统500的操作者观看的显示屏幕548上。

大容量存储设备552包括（但并不限于）硬盘、软盘、光盘只读存储器（CD-ROM）、数字视频磁盘只读存储器（DVD-ROM）、磁带、高密度软盘、高容量可移动介质、低容量可移动介质、固态存储设备和其结合。通信接口设备556包括网卡、调制解调器接口或类似的通信设备，用于通过通信链接560来访问网络563。另外，通信接口

设备556可以包括通信端口，例如串行端口（例如，IEEE RS-232）、并行端口（例如，IEEE-1284）、通用串行总线（USB）端口和红外（IR）端口。

（I/O）设备568_1-568_N包括键盘、鼠标、音频/声卡、打印机等。例如，音频声卡可以被连接到一个或多个扬声器。同样，键盘、鼠标等可以被用于输入数据。

扩展总线控制器572被连接到非易失性存储器575，其包括系统固件576。系统固件576包括系统BIOS，尤其用于控制在系统500中的硬件设备。系统固件576还包括ROM580和闪存（或EEPROM）584。扩展总线控制器572也被连接到具有RAM、ROM和/或闪存（未示出）的扩展存储器588。系统500可以另外包括连接到总线控制器512的存储模块590。在一个实施例中，存储模块590包括ROM592和闪存（或EEPROM）594。

此外，系统500可以包括、存储、加载和执行或运行计算机程序、操作系统、图形驱动器、应用程序、软件或指令序列来执行这里对于附图1-4和6所描述的功能。

如本领域技术人员熟知的，系统500可以包括图形驱动器（例如一部分的操作系统（OS））和至少一个应用程序，在一个实施例中，至少一个应用程序从大容量存储设备552被加载到系统存储器524，并在开机自测（POST）后启动。OS可以包括任意类型的OS，包括但不限于或不约束为磁盘操作系统（DOS）、Windows、Unix、Linux、OS/2、OS/9、Xenx等。每个操作系统和/或图形驱动器可以是一个或多个控制系统500的操作和资源分配的程序集合。应用程序可以包括一个或多个具有要显示给用户的图形、图形对象、图像、图形帧和/或视频的软件程序集合。

附图6是简化的存储器着色器代码编译器。附图6示出了可以是也可以不是一部分系统500的编译器600。同样，编译器600可以包括以上对于系统500没有描述的组件或特征。示出的编译器600包括链接到应用程序610的操作系统620、主存储器524、图形电路532和中央处理单元504。示出的应用程序610包括着色器代码612和具有对象616的可执行代码614。应用程序610可以是以上对于附图1的框110到

130、附图2的框210、和/或对于附图5所描述的应用程序。主存储器524包括地址660，例如用于存储应用程序在这里所述的代码、着色器代码、指针、可执行代码、操作系统代码、图形驱动器代码、原始顺序、再排序顺序、编译代码、再排序算法、再排序数据结构和表，以及转换表的地址或位置。图形电路532包括寄存器634，例如用于加载按着色器代码的再排序顺序或次序编译的编译代码的寄存器。

示出的操作系统620包括图形驱动器622。图形驱动器622包括原始顺序624、指针625、再排序顺序626、编译器627、再排序算法623和再排序数据结构和表628。编译器627可以包括编译代码629。

操作系统620可以包括例如华盛顿的Redmond的微软公司的WINDOWS XP®或另一种WINDOWS®操作系统。操作系统620还可以包括加利福尼亚的Cupertino的苹果公司的MACINTOSH®操作系统；或其它的以上对于系统500的操作系统（例如在附图5加载在系统存储器524中）所描述的操作系统。

编译器600可以包括适当的电子设备或计算机硬件和软件，用于执行以上对于附图1-5所描述的功能。例如，附图6可以利用对于附图1的框160、附图2的框和附图3和4的例子300和400所描述的指针，来执行再排序和编译着色器代码为编译的代码。特别地，操作系统620可以包括或访问包含指令序列（例如包括图形驱动器622和其组件）的机器可访问介质（例如附图5的存储器562），所述指令在由处理器（例如CPU504）执行时使得编译器600加载应用程序610（例如，对于在附图1的框110加载应用程序所描述的），执行可执行代码614（例如附图1的框120所描述的），发送图形对象616和着色器代码612（或其部分）到图形驱动器622（例如附图1的框130所描述的），确定转换表是否存在于用于着色器代码的转换表662中（例如附图1的框140所描述的），如果表存在，利用表662转换该着色器代码（例如附图1的框150所描述的），或者利用图形驱动器622（例如附图1的框160、附图2的框210-290和附图3和4所描述的）来再排序和编译着色器代码612为编译代码，利用原始顺序624和编译代码629来存储用于转换编译的着色器代码为在表662中的编译的代码的转

换表（例如附图1的框170所描述的），存储编译的代码629在电路532的寄存器634中（例如附图1的框180所描述的），并利用电路532和寄存器634，在屏幕548上显示对象616（例如附图1的框190和对于附图5所描述的）。

因此，编译器600可以执行以上对于附图1-5所描述的功能。特别地，编译器600可以是安装于其中的操作系统620的计算机单元，具有地址660的存储器524（例如主存储器、RAM存储器和/或CACHE存储器）。在地址660，存储器524可以具有物理地址；虚拟地址；存储原始顺序624、指针625、再排序顺序626、编译器627和/或编译代码629的位置。同样地，存储器524的地址600可以存储应用程序610、再排序算法623、图形驱动器622和再排序数据结构和表628的其它部分。因此，可以理解的是，编译器600具有应用程序610、操作系统620、存储器524、电路532、驱动器622、顺序624、指针625、顺序626、编译器627、算法623和表628，以利用这里所述的指针（例如以上对于附图2-4所描述的），从原始顺序到再排序顺序，来再排序高级语言或文本着色器的图形着色器指令，使得再排序顺序可以被编译为存储在寄存器634和表662中的编译的语言，以允许在屏幕548上显示对象616。

根据实施例，以上对于附图1-6所描述的概念可以应用于根据着色器代码在存储器中的地址和在次序中的位置来选择、转换和/或移动着色器代码指令的情况中。例如，这种情况可以包括代替选择要转换的指令，而是选择在存储器中的地址和在序列中的位置来转换（例如附图2的框225；附图3的线312和332；和/或附图4的线412和432）的情况。此外，在这些实施例中，在某些情况中，可能将指向地址或位置的指针转换为指向彼此，而不访问第一或第二地址（例如以上对于附图1的框230和240；附图3的线314、316、334和336；和/或附图4的线414、416、434和436）。例如，要转换的在第一地址或位置的指针和在第二地址或位置的指针的数据或信息可以被保存，直到第一地址或位置被转换为第二地址或位置，反之亦然。参考附图2，例如，框225可以包括选择地址，以转换它们的指令，但是框230和240是不必要的。代替的，处理可以简单地包括框250和260

来将指针从第一地址移动到第二地址，反之亦然，因为其移动的指令是不相干的。因此，最终结果是存储在第二地址的指令的指针现在指向了其它指令的地址，反之亦然。

尽管这里描述的概念是对于再排序图形着色器指令而描述的，但是该概念还可以应用到再排序其它类型的图形指令、除了图形指令之外的指令、或存储在存储器中的地址、位置或序列中的其它数据。例如，再排序指令可以被指定为图形电路或除了图形电路之外的电路。例如，再排序指令可以被指定为处理器、协处理器、算法处理器、存储器中的存储空间、通过网络的传输、或者用于除了着色或显示图形之外的使用，例如在屏幕上。同样，以上概念可以扩展到再排序其它类型的高级代码、高级指令、数据组、数据阵列、数据类型、或存储在位置、地址或序列中的信息，并被再排序为再排序顺序。可以预想的是，相似的概念可以被应用到指令为空指令或指令包括空指令或者不存在指向该位置的情况。代替地，在这些位置，可以存在其它类型的数据。另外，以上概念可以被应用于编译的再排序指令，除了由JIT编译之外。在某些情况中，该概念可以被应用到再排序指令，使得该指令可以再排序而执行，而不需编译。

在前述说明中，描述了特定的实施例。然而，可以对其进行各种修改和变化，而不脱离如权利要求提出的实施例的更宽的精神或范围。因此，本说明书和附图是示意性的，而不是意味着限制性。

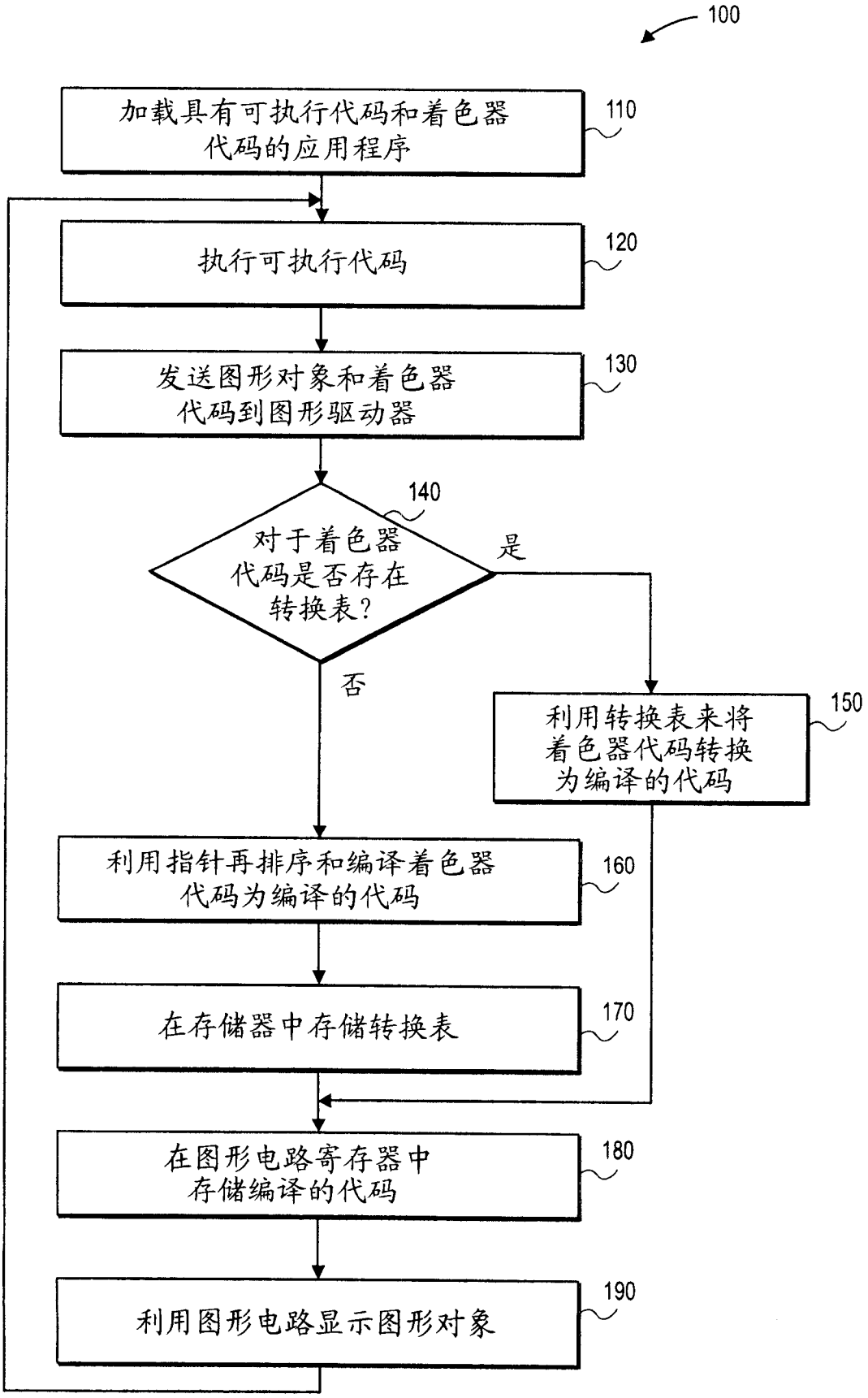

100
加载具有可执行代码和着色器代码的应用程序
110
执行可执行代码
120
发送图形对象和着色器代码到图形驱动器
130
对于着色器代码是否存在转换表?
140
是
否
利用转换表来将着色器代码转换为编译的代码
150
利用指针再排序和编译着色器代码为编译的代码
160
在存储器中存储转换表
170
在图形电路寄存器中存储编译的代码
180
利用图形电路显示图形对象
190

图 1

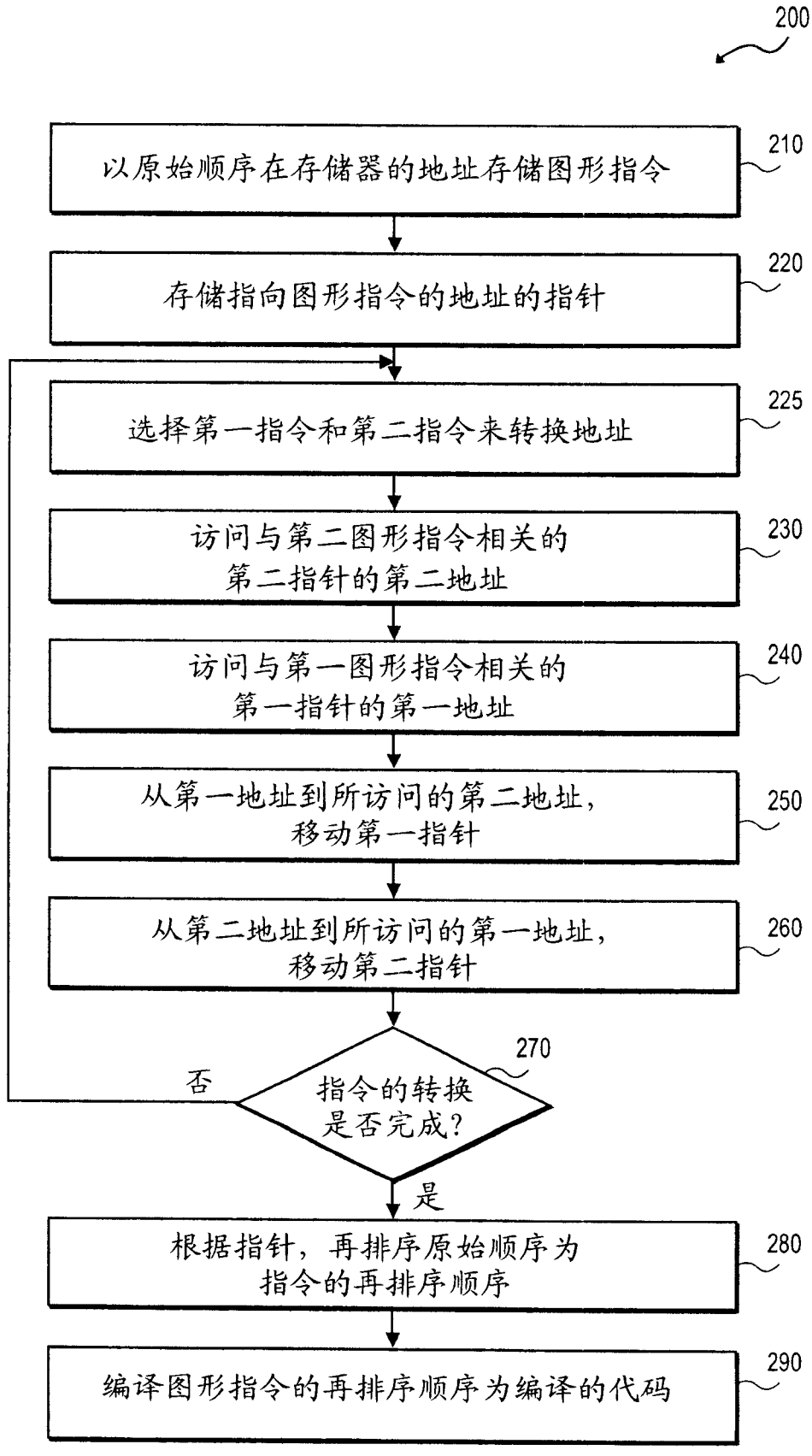

200
以原始顺序在存储器的地址存储图形指令
210
存储指向图形指令的地址的指针
220
选择第一指令和第二指令来转换地址
225
访问与第二图形指令相关的
第二指针的第二地址
230
访问与第一图形指令相关的
第一指针的第一地址
240
从第一地址到所访问的第二地址，
移动第一指针
250
从第二地址到所访问的第一地址，
移动第二指针
260
指令的转换
是否完成?
270
否
是
根据指针，再排序原始顺序为
指令的再排序顺序
280
编译图形指令的再排序顺序为编译的代码
290

图 2

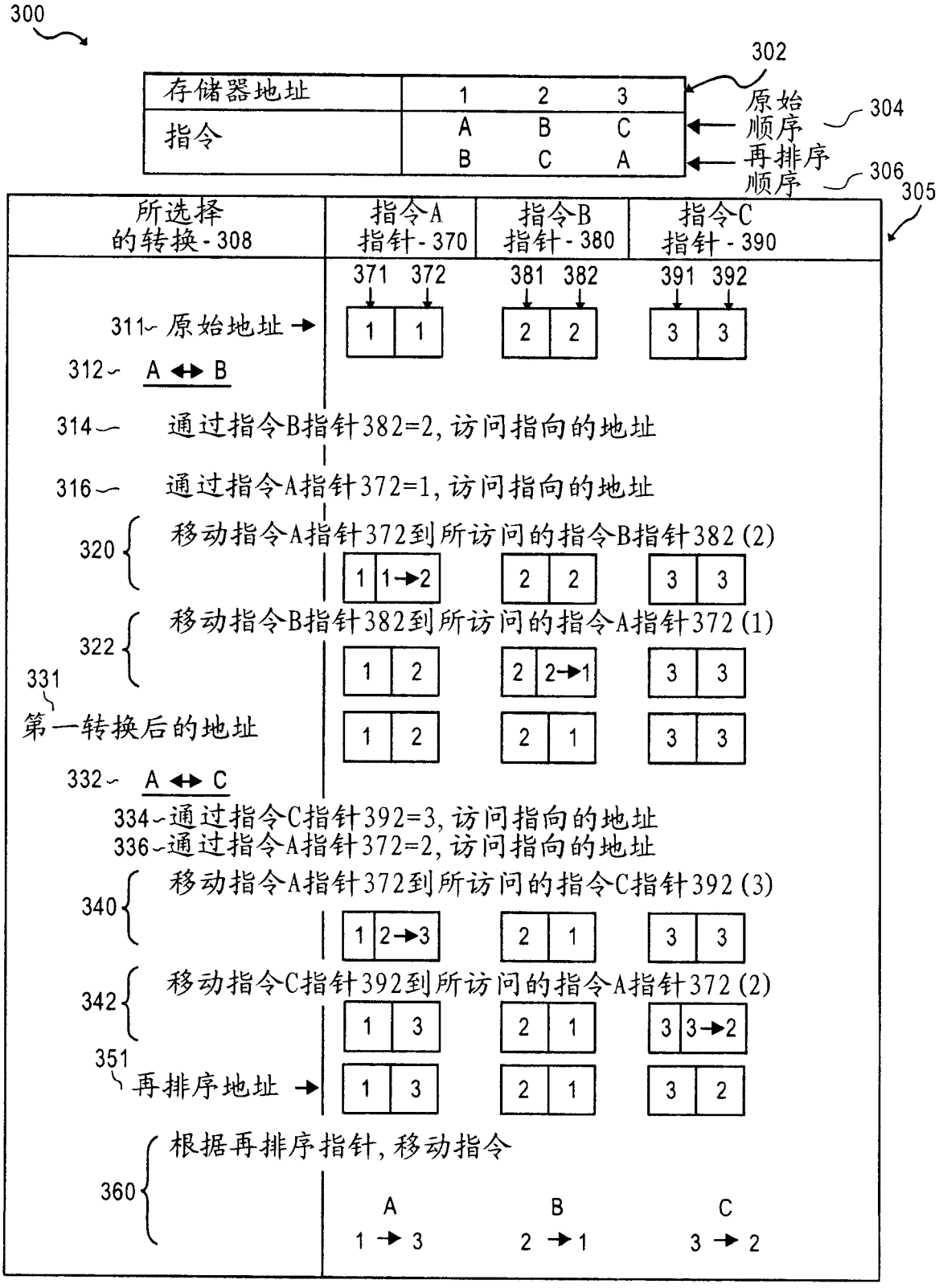

300
302
存储器地址
1 2 3
指令
A B C
B C A
原始顺序
再排序顺序
304
306
305
所选择的转换-308
指令A指针-370
指令B指针-380
指令C指针-390
371 372
381 382
391 392
311 原始地址
312 A ↔ B
314 通过指令B指针382=2,访问指向的地址
316 通过指令A指针372=1,访问指向的地址
320 移动指令A指针372到所访问的指令B指针382 (2)
322 移动指令B指针382到所访问的指令A指针372 (1)
331
第一转换后的地址
332 A ↔ C
334 通过指令C指针392=3,访问指向的地址
336 通过指令A指针372=2,访问指向的地址
340 移动指令A指针372到所访问的指令C指针392 (3)
342 移动指令C指针392到所访问的指令A指针372 (2)
351
再排序地址
360 根据再排序指针,移动指令
A 1 → 3
B 2 → 1
C 3 → 2

图 3

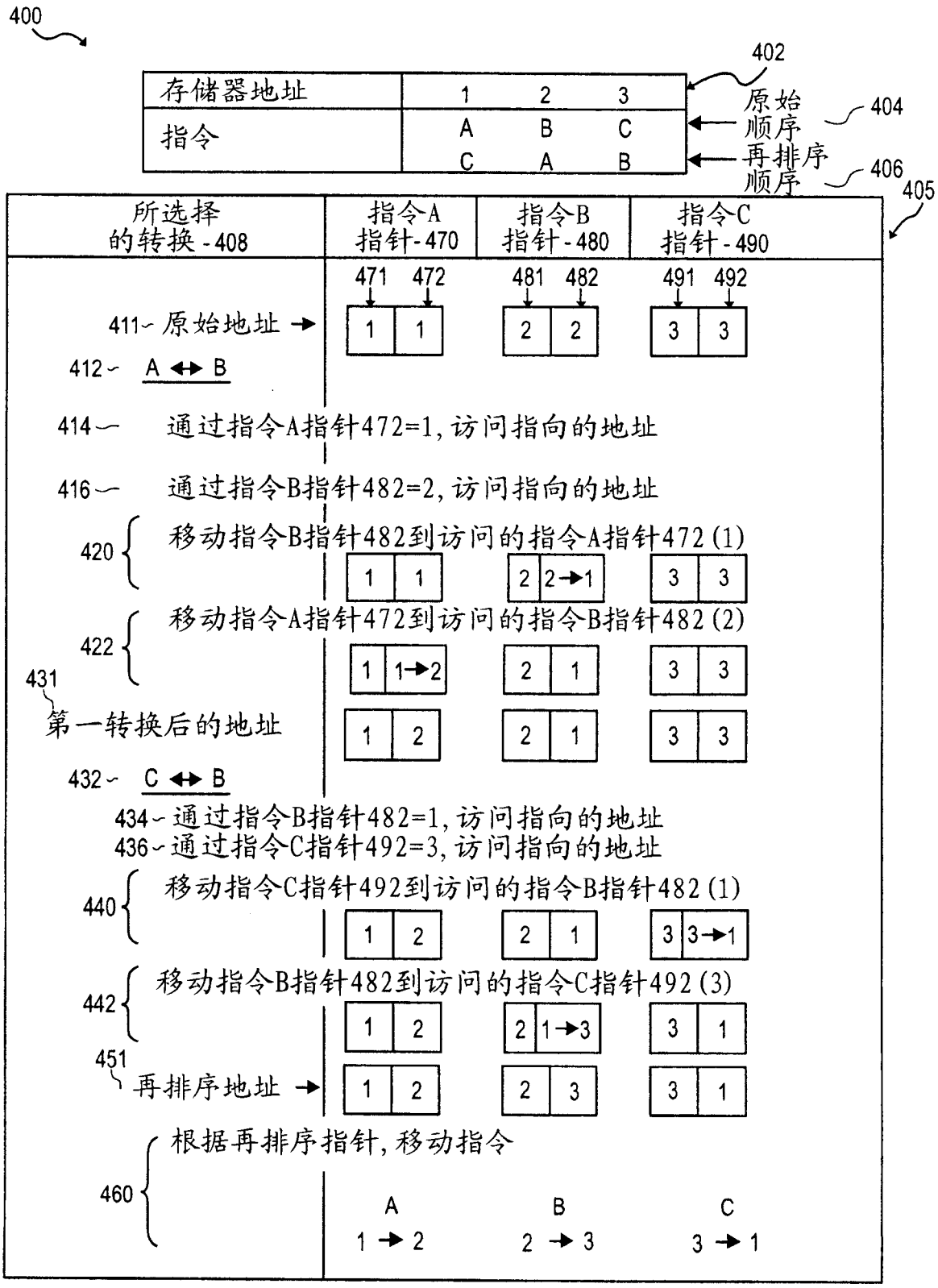

400
402
存储器地址
1 2 3
指令
A B C
C A B
原始顺序
再排序顺序
404
406
405
所选择的转换-408
指令A指针-470
指令B指针-480
指令C指针-490
471 472
481 482
491 492
411 原始地址
412 A ↔ B
414 通过指令A指针472=1,访问指向的地址
416 通过指令B指针482=2,访问指向的地址
420 移动指令B指针482到访问的指令A指针472(1)
422 移动指令A指针472到访问的指令B指针482(2)
431 第一转换后的地址
432 C ↔ B
434 通过指令B指针482=1,访问指向的地址
436 通过指令C指针492=3,访问指向的地址
440 移动指令C指针492到访问的指令B指针482(1)
442 移动指令B指针482到访问的指令C指针492(3)
451 再排序地址
460 根据再排序指针,移动指令
A 1→2
B 2→3
C 3→1

图 4

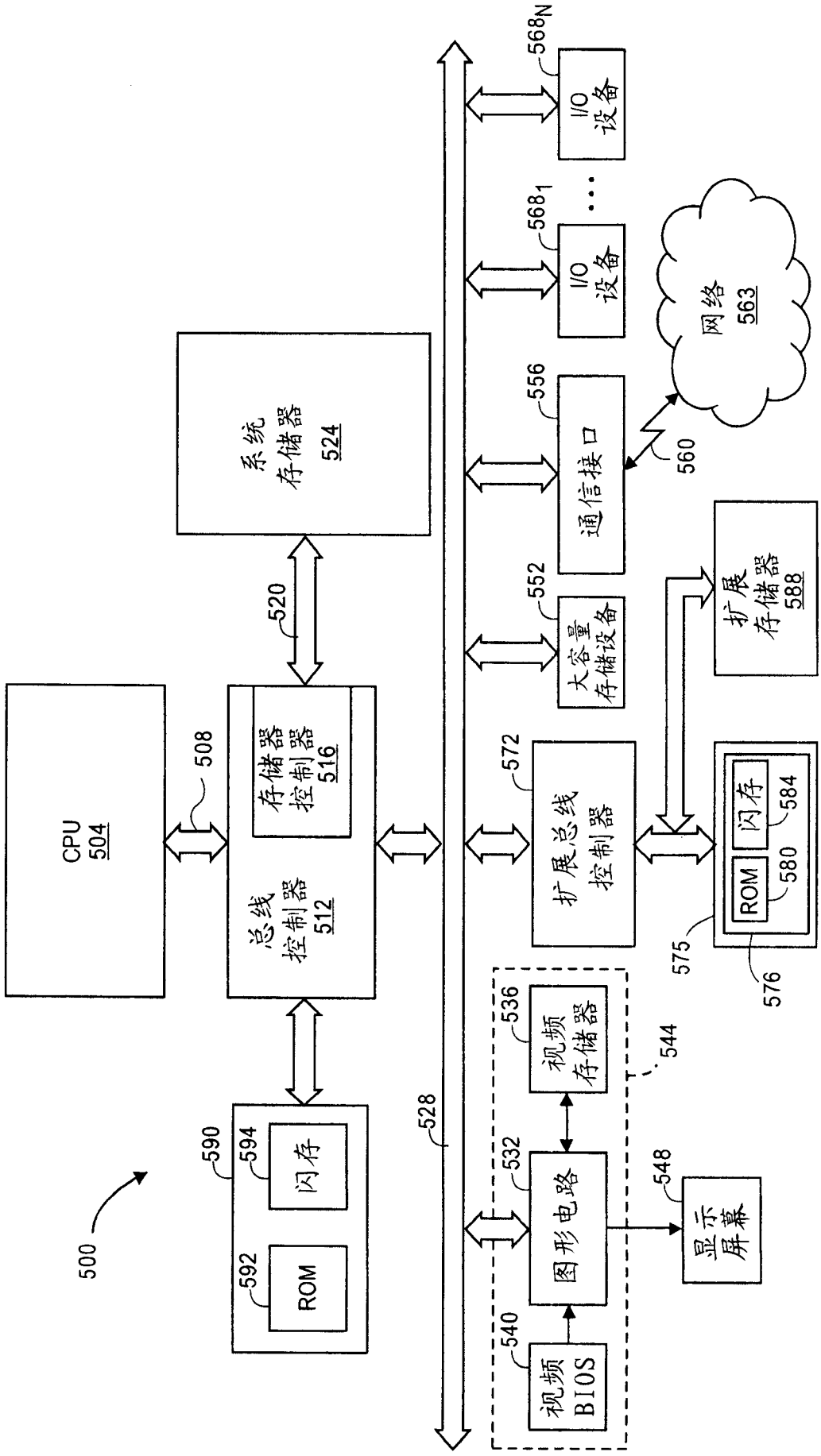

500
CPU
504
508
总线
控制器
512
存储器
控制器
516
520
系统
存储器
524
528
590
594
闪存
592
ROM
572
扩展总线
控制器
536
视频
存储器
532
图形电路
540
视频
BIOS
544
548
显示
屏幕
552
大容量
存储设备
556
通信接口
560
网络
563
568N
I/O
设备
568₁
扩展
存储器
588
575
576
ROM
580
闪存
584

图 5

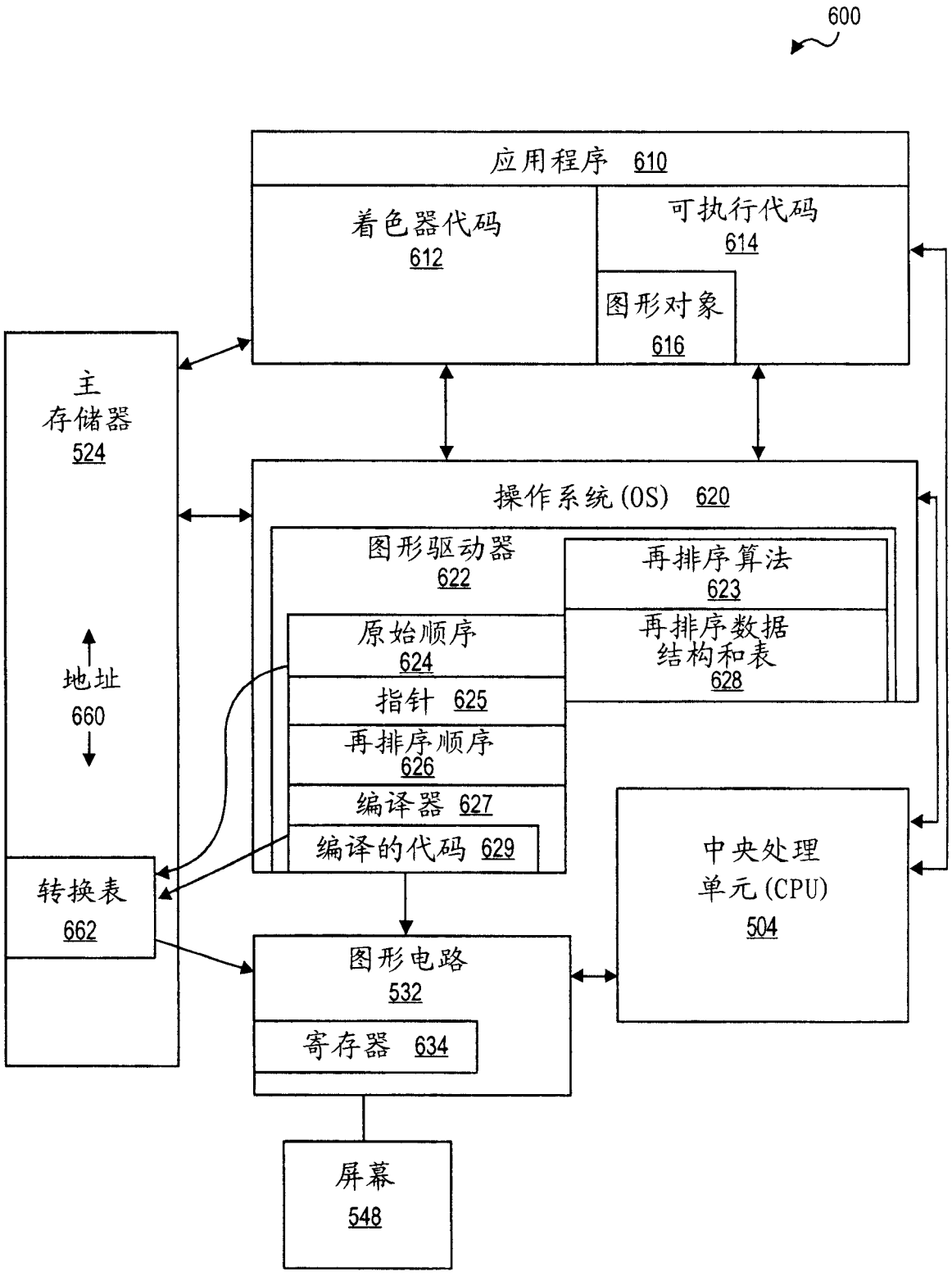

600
应用程序 610
着色器代码 612
可执行代码 614
图形对象 616
主存储器 524
操作系统(OS) 620
图形驱动器 622
再排序算法 623
原始顺序 624
再排序数据结构和表 628
指针 625
再排序顺序 626
编译器 627
编译的代码 629
地址 660
转换表 662
中央处理单元(CPU) 504
图形电路 532
寄存器 634
屏幕 548

图 6