[19] 中华人民共和国国家知识产权局

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

[12] 发明专利说明书

专利号 ZL 200510071234.6

[45] 授权公告日 2009 年 5 月 20 日

[11] 授权公告号 CN 100489769C

[22] 申请日 2005.5.13
[21] 申请号 200510071234.6
[30] 优先权
[32] 2004. 5.21 [33] US [31] 10/851,929
[73] 专利权人 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 查理斯 F·雪洛
[56] 参考文献
US4517640A 1985.5.14
US6363473B1 2002.3.26
US5551054A 1996.8.27
US5659703A 1997.8.19
US2004/0078550A1 2004.4.22
审查员 徐 春

[74] 专利代理机构 北京市柳沈律师事务所
代理人 钱大勇

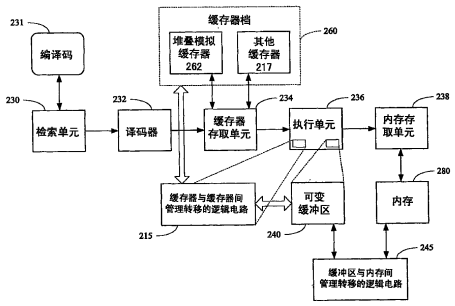
权利要求书 2 页 说明书 14 页 附图 5 页

[54] 发明名称

在缓存器架构处理器中管理堆栈转移的装置与方法

[57] 摘要

本发明揭示用于模拟一堆栈的一部分的装置与方法，本发明的某些具体实施例是管理模拟堆栈顶部的处理器缓存器与包含堆栈其它部分的内存间数据项的转移。 一些具体实施例是利用一可变缓冲区，此可变缓冲区被配置于处理器缓存器与内存间以作缓冲之用。 储存与可变缓冲区中实际的数据项量具有弹性，因此可变缓冲区与处理器缓存器间的转移可维持有效的堆栈数据项填入处理器缓存器(假设堆栈数据项存在)。 然而，可变缓冲区与内存间的转移可被配置在只有可变缓冲区超出可填入的容量时发生。



1.一处理器，其特征在于该处理器包含：

多数个缓存器，该多数个缓存器被配置来模拟一内存堆栈的一顶端部分；

一缓冲区；

一第一逻辑电路，该第一逻辑电路被配置来管理该多数个缓存器与该缓冲区间的转移；以及

一第二逻辑电路，该第二逻辑电路被配置来管理该缓冲区与一内存间的转移；

所述缓冲区的有效堆栈数据项超出一第一预设数量时，数据是从该缓冲区被转移至该内存，且在该缓冲区的有效堆栈数据项小于一第二预设数量时并且在该内存中含有有效堆栈数据项时，数据是从该内存被转移至该缓冲区，其中该第二预设数量是小于该第一预设数量。

2.如权利要求1所述的处理器，其特征在于，所述该多数个缓存器的任一未包含有效的堆栈数据项并且该缓冲区包含至少一有效堆栈数据项时，数据会从缓冲区被转移至该多数个缓存器。

3.一系统，其特征在于，该系统包含：

一处理器与一内存；

一模拟逻辑电路，该模拟逻辑电路是被配置以该处理器的多数个缓存器模拟一内存堆栈的一顶端部分；

一缓冲区；以及

一转移逻辑电路，该转移逻辑电路是用来管理该多数个缓存器与该内存的一部分间的数据转移，该内存的该部分是被配置来储存该内存堆栈的其它部分；

所述缓冲区中的堆栈数据项超出一第一预设数量时，该转移逻辑电路为响应一推入运算，将一堆栈数据项从该缓冲区转移至该内存，以及所述的缓冲区中的堆栈数据项低于一第二预设数量时，该转移逻辑电路为响应一推出运算，将一堆栈数据项从该内存转移至该缓冲区，其中该第二预设数量小于该第一预设数量。

4. 如权利要求 3 所述的系统，其特征在于，所述该转移逻辑电路储存一可变数量的堆栈数据项于该缓冲区中。

5.如权利要求 3 所述的系统，其特征在于，所述的转移逻辑电路的功能，至少包含下列两种功能中的某一种：

为响应该推入运算，将一堆栈数据项从一处理器缓存器转移至该缓冲区；以及

在该多数个缓存器中每一个都被有效堆栈数据项所使用时，为响应推入运算而将一堆栈数据项从该多数个缓存器其中之一转移至该缓冲区。

6.如权利要求 3 所述的系统，其特征在于，所述的转移逻辑电路是在一个新的堆栈数据项被移入该处理器缓存器前，将堆栈数据从该处理器缓存器转移至该缓冲区。

7.如权利要求 3 所述的系统，其特征在于，所述缓冲区中的堆栈数据项低于一第二预设数量时，该转移逻辑电路为响应一推出运算，将一堆栈数据项从该内存转移至该缓冲区。

8.一种用于一内存与一处理器间管理堆栈转移的方法，其特征在于，包含：

使用多数个处理器缓存器模拟一内存堆栈的一顶端部分；以及

实施一深度可变缓冲区来管理在该多数个处理器缓存器与该内存间的转移；

在回应一推入运算时：

在该深度可变缓冲区中的堆栈数据项超出一第一预设数量时，自该深度可变缓冲区转移一堆栈数据项至该内存；以及

从该多数个缓存器的一缓存器转移一堆栈数据项至该深度可变缓冲区；以及

在回应一推出运算时：

从该深度可变缓冲区转移一堆栈数据项至该多数个缓存器的一缓存器；以及

在该可变缓冲区中的堆栈数据项的数量低于一第二预设数量时，从该内存转移一堆栈数据项至该深度可变缓冲区；

其中该第二预设数量小于该第一预设数量。

在缓存器架构处理器中管理堆栈转移的装置与方法

技术领域

本发明是关于一处理器，特别是关于一用于在内存与处理器的缓存器间模拟部分内存堆栈的管理堆栈转移的装置与方法。

背景技术

处理器(如微处理器)是众所皆知且被广泛使用于许多产品与应用中，例如从桌上型计算机到可携式电子设备，如行动电话与个人数字助理(PDA; Personal digital assistant)。在目前现有的处理器有一些功能强大(如在高阶计算机工作站的处理器)，而其它处理器则是具有简单的设计，以用于较低阶、较不昂贵的应用与产品。

跨平台(platform-independent)的程序语言(如 Sun Microsystems, Inc. 出的 Java)提供了不同于传统、特定平台(platform-specific)程序语言的结构与运算特征，跨平台的程序语言典型地运用跨平台程序代码(机器可读的指令)，可适用于多种硬件平台，而不去考虑硬件平台的特定指令集。一个硬件平台典型地包含一个或多个处理器(如微处理器或微控制器)，此一个或多个处理器是用以执行具有特定格式的特定指令集，有时特定指令集也被称为原生指令集(native instruction set)。相对地，特定平台语言是运用编译器来产生给一特定硬件平台的原生(active)程序代码。在一些实例中，当相同的源码被以不同的特定平台的编译器编译成用于多种平台的适当程序代码时，产生的程序代码并非跨平台。

指令集中有一类是包含了一种在运作时使用堆栈架构方法(stack-based approach)来储存与操作数据项的指令集，在支持这样堆栈架构的指令集的数据项处理系统，堆栈可储存一连串的数据项(如运算值)，这些数据项是被以一特定顺序放入堆栈，再被以相反顺序从堆栈取出，因此最

后置入的数据项会被最先取出。堆栈架构处理器可以提供包含用于数据项写入与数据项读出的复数个可寻址堆栈项(addressable stack entry)的堆栈，可寻址堆栈项是关联于一堆栈指针，此堆栈指针指示着堆栈内顶端的位置。堆栈指针明确指出在堆栈内的一参考点，该参考点表示最新被存入堆栈的数据项，并通过此被参照来对堆栈做其它的存取。

Sun Microsystem Inc 所制定的 Java 虚拟机器指令集便是堆栈架构指令集的其中一例，Java 程序语言追求的是一种不需要更动 Java 软件便能将以 Java 撰写的计算机软件执行于许多不同硬件处理平台上的环境。

另一种指令集包含在运作时使用缓存器架构方法来储存与操作数据项的指令集，英格兰(England) ARM limited of Cambridge 的 ARM 处理器便是这类缓存器架构系统的一例。ARM 指令是对储存在处理器的缓存器上的操作数(operand)执行数学运算、加载、储存或其它的运算(operation)，其中操作数是被指定于指令中的缓存器字段(register field)。

有些被设计来执行缓存器架构指令的数据项处理系统也支持堆栈架构指令的执行，在这样的系统下，堆栈架构指令被转换成一连串动作，由处理器核心使用在缓存器库(register bank)或缓存器集合(resister set)的缓存器以执行这一连串的动作。那些操作所需的堆栈中的数据项是从堆栈储存到缓存器库的缓存器中使其可以被处理器核心所使用。典型地，在缓存器库内的一组缓存器会被配置用来持有从部分堆栈而来的堆栈数据项。复数个不同的对照状态(mapping states)可以被提供用来持有从堆栈不同部分而来的相应的堆栈操作数于缓存器库中不同的缓存器。对照状态可以依运算而改变，这些运算会在用于堆栈的该缓存器合中加入或移除堆栈操作数，是以提供功能类似堆栈的堆栈指针的方式来达成，这样方法所追求的是降低在缓存器架构的处理器提供类似堆栈储存(stack-like storage)的处理负担。

在这样的系统中，堆栈中的堆栈项可以被视为具有固定的大小，并且在缓存器集中被配置用来持有堆栈操作数的每一个缓存器可以被安排来只对一相应的堆栈项储存数据项。然而处理器核心中专用于堆栈操作数储存的缓存器可能被限制于提供其它缓存器需求的功能，如堆栈架构指令译码成用来在缓存器架构的处理器执行运算的管理的功能，与像是

在堆栈架构处理系统中可以找到的变量指针或常数区(constant pool)指针的控制变量模拟。也就是这些情形可能发生在要将缓存器集合中被持有的迭操作数移回缓存器(在堆栈中)以提供空间给新的堆栈操作数置入缓存器集合中。

现有系统中有些使用缓存器来实施部分内存堆栈，其在效率获益上超出了内存堆栈的传统使用，例如在缓存器间的数据项移动会快于缓存器与内存间的数据项移动。然而这些现有的施做方式遭遇到许多缺点，其中一个缺点显然是缓存器会不足以被堆栈填入。在堆栈缓存器被填入后，更进一步从堆栈的推入与推出会导致对外部内存的过度读写(每个推入与推出都要一次)，造成内存流量增加与过度的电力消耗。尤其在可携式(如以电池运作)的装置，对电力消耗的尽可能的改进是很重要的需求。

发明内容

鉴于上述的发明背景中，为了符合产业上某些利益的需求，本发明提供一种用于模拟部分堆栈的装置与方法，可用以解决上述传统的堆栈未能达成的标的。

为达成某些优点与新颖特征，本发明说明模拟一堆栈的部分的方法与装置，本发明的某些具体实施例是管理模拟堆栈顶部的处理器缓存器与包含堆栈其它部分的内存间数据项的转移。一些具体实施例是利用一可变缓冲区，此可变缓冲区被配置于处理器缓存器与内存间做缓充之用。储存与可变缓冲区中实际的数据项量具有弹性，以可变缓冲区与处理器缓存器间的转移来维持有效地堆栈数据项填入处理器缓存器(假设堆栈数据项存在)。然而，可变缓冲区与内存间的转移可被配置在只有可变缓冲区超出可填入的容量时发生。

特别是如果可变缓冲区的堆栈数据项低于一低一预设量时，堆栈数据项将被从内存读出至可变缓冲区。并且如果缓冲区的堆栈数据项高于一第二预设量时，堆栈数据项被从可变缓冲区写至内存。

本发明的具体实施例并提供管理在处理缓存器与内存间堆栈数据项转移的方法。

附图说明

图 1A 是为在先前技术中管线处理器内一些阶段的功能区块示意图；

图 1B 是为在先前技术中以一些缓存器模拟堆栈内存的一部分的功能区块示意图；以及

图 2 是为在本发明的一具体实施例中相似于图 1A 的一管线处理器的功能区块示意图；以及

图 3 是类似图 1B，为本发明的一具体实施例的运算示意图；以及

图 4 是为本发明的一具体实施例的上层功能运算流程示意图。

主要部分的代表符号：

110 指令检索单元

120 译码单元

130 执行单元

132 产生中断信号的逻辑电路

134 中断信号

136 产生分支信号的逻辑电路

138 分支信号

140 内存存取单元

150 缓存器回写单元

160 缓存器文件

163 时间点

162 缓存器

164 时间点

165 时间点

166 推入运算

167 加运算

180 内存

182 堆栈位置

215 缓存器与缓冲区间管理转移的逻辑电路

217 其它缓存器

230 检索单元

231 编译码

232 译码单元

234 缓存器存取单元

236 执行单元

238 内存存取单元

240 可变缓冲区

245 缓冲区与内存间管理转移的逻辑电路

260 缓存器文件

262 堆栈模拟缓存器

263 时间点

264 时间点

265 时间点

266 时间点

267 时间点

291 推入运算

292 推入运算

293 推出运算

294 推出运算

280 内存

具体实施方式

为了能彻底地了解本发明，将在下列的描述中提出详尽的步骤及其组成。显然地，本发明的施行并未限定于相关的技艺者所熟习的特殊细节。另一方面，众所周知的组成或步骤并未描述于细节中，以避免造成本发明不必要的限制。本发明的较佳实施例会详细描述如下，然而除了这些详细描述之外，本发明还可以广泛地施行在其它的实施例中，且本发明的范围不受限定，其以之后的专利范围为准。

参考图 1A 所示，该图是用来描述一个用来执行指令的五阶段管线

处理器架构的功能区块图，其它像是具有更多或更少管线阶段与/或不同配置的管线架构亦可以本发明的概念与教示来实施。在图 1A 的架构中举例了一指令检索单元 110、一译码单元 120、一执行单元 130、一内存存取单元 140、一缓存器回写单元 150，除了这里所描述的之外，这些单元(或逻辑区块)是为熟悉该项技艺者所熟知，并不需要在此更进一步描述。

在现有技术中，指令检索单元 110 进行指令内存检索，此单元被配置来决定程序计数器(位于缓存器文件 160 中)的内容或值，用于循序(in-order)的指令执行，同样地也用于例外向量(exception vectors)、分支(branches)与返回(returns)。指令检索单元 110 也被配置来决定返回地址(return address)给所有例外(exceptions)与分支连结(branch-link)指令，并写入或储存此返回地址至缓存器文件 160 内的指定缓存器。指令检索的寻址可以是直接指向内存的实体地址，或使用实体或虚拟寻址的指令快取(instruction cache)。虽然缓存器文件 160 的内部架构并未显示，此缓存器文件 160 包含各样被处理器使用的缓存器。在现有技术中，这样的缓存器可以包含一般用途(general-purpose)缓存器或特殊用途(special-purpose)缓存器(像是状态缓存器、程序计数器或其它)。再者，缓存器文件 160 内的缓存器可以是编派成排(bankded)或非编派成排(unbanked)的。在现有技术中，一个非编派成排的缓存器会参照一个单一实体缓存器，此单一实体缓存器对运算的所有处理器模式都是有效的。典型地，非编派成排的缓存器皆为一般用途，在架构中不具有特殊用途。当然程序设计师必须要确保，在运算(或当处理分支例行程序(routine)或其它子程序(subroutine))的模式改变时，缓存器的内容被储存(如推入一个堆栈)，和当从一被改变的操作模式中返回时被复原。

在此考量下，缓存器文件 160 可能包含复数个缓存器 162(在本例中标示为 R0 至 R7)，复数个缓存器 162 伴随着复数个其它缓存器(未示于图标)，这些缓存器负责传统处理器缓存器的功能与运算。缓存器 162 是被配置与被控制来模拟复数个(在本例中为 8 个)在内存堆栈最上层 8 个数据项(缓存器 R0-R7)，这些缓存器 162 的流程与运算将于图 1B 被提供。

译码单元 120 被运算来对由指令检索单元 110 送来的指令译码，并产生必要的控制信号给执行单元 130，以进行个别指令的执行。译码单元

120 的特定架构为处理器相关，其一般运算与组成为熟悉相关技术者可推知。同样地，执行单元 130 的结构与运算也是处理器相关，为熟悉相关技术者可推知。一般而言，一个执行单元 130 包含进行指令运算的电路，该电路是以译码单元 120 所产生的控制信号来决定指令的执行。

如图 1A 所示，例中实施例的执行单元 130 可能包含逻辑电路 132 来产生一个或多个中断信号 134，同样地，逻辑电路 136 产生一个或多个分支信号 138。由名称可知，中断信号 134 代表一中断条件(如 IRQ、FIRQ 或其它)。同样地，分支信号 138 代表一个分支条件(也可以是代表从一分支返回)，这些信号代表接连的乱序(out-of-order)指令。

内存存取单元 140 是外部数据项内存为响应被执行单元 130 所执行的指令所做的读写界面。当然，并非所有指令需要内存存取，但是对需要者而言，内存存取单元 140 为进行外部内存存取的必要条件。这样的内存存取可以是直接或通过由实体或虚拟寻址来透过一数据高速缓存来完成。

最后，缓存器回写单元 150 负责储存或写入(从指令执行所产生的)内容至缓存器文件 160 中(指定)的缓存器。例如，一指令的执行为将两一般用途缓存器的内容相加，并将相加的内容存入第三个一般用途缓存器。在这样的指令执行后，缓存器回写单元 150 使得加总所得到的值被写入第三个一般用途缓存器。

参考图 1B，该图举例堆栈模拟缓存器 162 的运算，其中堆栈模拟缓存器 162 是关连于位于内存 180 中的传统堆栈。在图 1B 的例子中，假设有四个用来处理堆栈数据项的缓存器(R0 至 R3)，在此考量下，这些缓存器模拟内存堆栈中最上面四个堆栈位置。在图标中另外举例了一内存 180，其中一部分 182 可配置来处理复数个堆栈数据项，内存 180 中的该部分 182 是用以实施复数个堆栈位置，该部分 182 可以被动态配置来因应增加的堆栈数据项而增长。

图 1B 的图示中举例了在三个连续时间点(以数值 163、164 与 165 来标示)的堆栈模拟缓存器 162 与内存 180。在第一个时间点 163，堆栈模拟缓存器 162 包含数值 F、E、D 与 C。要注意的是，在这些个别缓存器中的值是以字母代表，与例子中的实际内容或数值无关。内存 180 中的

堆栈部分 182 包含两值(B 与 A)，分别储存在堆栈位置 S0 与 S1 中。习惯上，四个堆栈模拟缓存器 162 中缓存器 R0 被指定为堆栈位置的顶端。同样地，堆栈位置 S0 被指定为内存 180 中缓存器位置的顶端，并且在需要额外堆栈位置时，额外堆栈位置被加入并标示为 S1、S2...等等。因此，在 163 所标示的时间点，“A”为存在堆栈中最旧的值(或第一个值)。当每一个后续值(B、C...等等)被推入堆栈时，A 值被连续地往堆栈深处推下。例如，当 A 内容最先被推入堆栈时，其被推入堆栈模拟缓存器 R0，在同时，没有有效的堆栈数据项会被包含于缓存器 R1、R2 或 R3，也没有任何有效的堆栈数据项存在于内存 180 中。

以数字 164 标示的时间点举例了一个 PUSH G 运算在堆栈模拟缓存器与内存堆栈的相关部分 182 间的影响。因为每一个堆栈模拟缓存器 162 都个别被有效堆栈数据项所占据，最旧项目(在本例中为 C)被从堆栈模拟缓存器 162 移至内存 180 的堆栈部分 182，在这样考量下，数值 C 被移入内存堆栈的顶端位置。先前分别占据堆栈模拟缓存器 R2、R1、R0 的堆栈数据项 D、E 与 F 会被分别移至堆栈模拟缓存器 R3、R2、R1，而新的堆栈数据项(G)接下来被移入堆栈模拟缓存器 R0，来当作堆栈的顶端位置。

参考数值 165 标示相应于一加法运算 167 的堆栈模拟缓存器与堆栈部分 182 的内容。在现有技术中，一加法运算是以将堆栈位置最上方两个位置相加，并且将结果存于堆栈顶端。因此，相应于加法运算 167，堆栈模拟缓存器 R0 会具有 G+F 的内容，如此便开放了堆栈模拟缓存器 R1(例如使其能够持有新数据项)。接下来，在缓存器 R1 的下的堆栈内容便会向上移动，因此缓存器 R2 与 R3 的内容会被分别移动至 R1 与 R2。同样地，内存堆栈 180 的堆栈部分 182 的顶端会被移动至堆栈模拟缓存器 R3，每一个在内存 180 的堆栈部分 182 中后面的堆栈数据项也被往上移动。

显然地，堆栈模拟缓存器与内存的堆栈部分 182 有效地组合成一动态堆栈，然而在堆栈模拟缓存器中数据项的移动与在堆栈模拟缓存器 162 与内存 180 间数据项的常态移动(当堆栈模拟缓存器装满时)会导致处理器频宽的超出与不希望的内存 180 读写所产生的电力消耗。

参照图 2，其为相似于图 1A 的本发明的一具体实施例的功能区块

示意图。图 2 的具体实施例举例了相关于传统管线阶段或单元，像是一检索单元 230、一译码器 232、缓存器存取单元 234、一执行阶段 236 与内存存取阶段 238。显然地，本发明应可施行于其它传统管线架构。图 2 并举例了检索单元 230 所检索的编译码 231，编译码 231 由此开始在各管线阶段的处理。典型地，编译码 231 被写入并被编译来执行于一特定的硬件架构之上。在本具体实施例的一较佳例子中，这样的硬件架构包含一堆栈架构处理器的架构。本发明在此所描述的特征是运算于这样的码，像是被编译成与本发明的此特定架构非相关(independent)的编译码 231。

如图 1A 所描述，一缓存器文件 160 典型地包含一缓存器架构处理器的一部分，这样的缓存器文件 260 亦被举例于图 2 的具体实施例中。缓存器文件 260 中的缓存器包含堆栈模拟缓存器 262 与其它缓存器 217，这些缓存器是惯见于缓存器文件 260 中。在一具体实施例中，更进一步包含例中四个堆栈模拟缓存器 262，然而与本发明的领域与精神一致的额外或较少缓存器可以被实行这些缓存器的堆栈模拟功能。

本具体实施例的一中心特征包含一可变缓冲区 240 的运用，此可变缓冲区 240 是被设置于堆栈模拟缓存器 262 与内存间 280。如图 1B 的现有技术中相关的描述，堆栈模拟缓存器 162 与内存 180 的堆栈部分 182 共同合作来定义了一内存堆栈。以类似的方式，堆栈模拟缓存器 262 与内存 280 的堆栈部分也组合来形成图 2 的系统的一工作堆栈。再者，可变缓冲区 240 提供复数个堆栈位置给额外的堆栈数据项，堆栈数据项在堆栈模拟缓存器 262 与内存 280 的堆栈部分间经由可变缓冲区 240 的管理提供了先前技术系统的系统效能与较低电力消耗的改进。为这目的，本发明的一具体实施例更包含逻辑电路 215，逻辑电路 215 是用以管理堆栈数据项在堆栈模拟缓存器 262 与可变缓冲区 240 间的转移。同样地，逻辑电路 245 可被提供用来管理可变缓冲区 240 与内存 280 的堆栈部分间堆栈数据项的转移的管理。在本例的具体实施例中，例中可变缓冲区 240 与逻辑电路 215 是为执行单元 236 的一部分，与例中逻辑电路 245 分开。在另一具体实施例(未被特别举例)中，可变缓冲区 240、逻辑电路 215 与逻辑电路 245 是全施做为执行单元 236 的一部分。在另一具体实施例中(未被特别举例)，这些组件可以被实施在处理器管线的其它区域，再者，这

些组件也可以被实施于非管线架构。

在一具体实施例中，可变缓冲区 240 被设定大小能储存八个堆栈数据项，“可变”一词是用以描述缓冲区 240，在缓冲区内所包含的堆栈数据项的数目为变动，其变动是相依于数据项是否推入至堆栈，或从堆栈推出。在这考量下，可变缓冲区 240 被设置来识别或利用在 JAVA 堆栈中组件的暂时位置。在此考量下，以堆栈架构编写或编译的码(如 JAVA)在被写入时会对在堆栈中相邻的数据项做经常性地参照。例如一个加指令仅对堆栈的最上面二个值相加并将结果储存在堆栈的最上面。相关于图 1B，堆栈模拟缓存器一旦满时，接下来的运算会造成堆栈模拟缓存器与内存间过度的数据项移动，使得不希望的过度内存存取量产生。这样的过度内存存取造成电力被内存过度地消耗，这是特别不希望发生在可携式电子装置上的，如行动电话、个人数字助理(PDA; personal digital assistants)或其它以电池运作的装置。

因此，本发明一具体实施例是使用一可变缓冲区 240，其大小能够持有八个堆栈数据项，在可变缓冲区 240 装满前(或已经超出某一门坎限值)，堆栈数据项并不会从可变缓冲区推至内存 280 的堆栈部分。逻辑电路 245 用以运算来管理缓冲区 240，除非四个(或其它预设量)或更少的堆栈数据项存在于目前的可变缓冲区 240，否则堆栈数据项不会从内存 280 的堆栈部分推出并转移到可变缓冲区 240。这样的可变缓冲区相当地小，在实作上不会占到大量的空间，同时跟先前技术比较起来效能有相当地改善(尤其在降低电力消耗方面)。

显然地，“用于管理转移的逻辑电路”(215 与 245)会较偏好管理数据项的地址，使得标准编译码 231 可以假设为是以一般的堆栈管理(在堆栈缓存器之外)。然而，这样的寻址可以被逻辑电路 215 与 245 修改来转译成缓存器识别(register identifiers)与/或被修正的地址(在缓冲区内储存的堆栈数据项的偏移量(offset)架构下)。

参照图 3，本发明的一具体实施例是举例于其中。图 3 是类似图 1B，在其中显示了在堆栈模拟缓存器 262、可变缓冲区 240 与内存 280 的堆栈部分 282 中的堆栈内容，这些堆栈内容是分别位于数个连续的时间点(分别以数值 263、264、265、266 与 267 来标示)。假设在第一个时间点 263，

内存 280 的堆栈部分 282 包含数据项 A 与 B，缓冲区储存位置 B5 至 B0 包含堆栈数据项 C、D、E、F、G 与 H，堆栈模拟缓存器 262 包含堆栈数据项 I、J、K 与 L。在这样的配置下，堆栈数据项 L 位于堆栈顶端，并且堆栈数据项 A 位于堆栈底端(如堆栈中最旧的组件)。如所示，在时间点 263 时，可变缓冲区 240 底部两个位置 B6 与 B7 尚可使用(或尚未被有效堆栈数据项所使用)。如果两个连续的推入运算 291(PUSH M 与 PUSH N)被执行时，整个堆栈将会如数值 264 所标示的时间点所示。

在这考量下，堆栈数据项 M 与 N 会被推入堆栈顶端，且位于堆栈模拟缓存器 262 中，而堆栈的其余内容会被往下推。由于有两个开放或未用位置为于可变缓冲区，则位在可变缓冲区 240 中最旧的堆栈数据项(C 与 D)会被移下到可变缓冲区 240 底部的缓存器 B7 与 B6，其它的内容也会被适当地向下移动。先前存在堆栈模拟缓存器 262 的缓存器 R3 与 R2 中的数据项 I 与 J 会被转变至可变缓冲区 240 的顶端两个位置 B1 与 B0。然而值得注意的是，两项目被推入堆栈中，内存 280 却没有任何的写入，因此节省了原本要转移至内存 280 的电力消耗。

在图 3 揭示的具体实施例的运算是为了说明堆栈数据项在堆栈模拟缓存器与可变深度缓冲区间被以移动方式移动时的操作，然而，显然地数据项(在实际上)可以是以指标指向数据项来有效地移动，而不是真的从缓存器至缓存器或从位置至位置来搬动数据项。这样指标的管理可以用很多种方式来实作，其中一种例子被描述在另一个美国申请案中，其序号为 10/827,662，并且申请日期为 2004 年 4 月 19 日，该内容在此被参照。

然而当可变缓冲区 240 满了时，接下来的推入 292(PUSH)会导致堆栈数据项被有效地如涟漪般影响堆栈模拟缓存器 262 与可变缓冲区 240，使用一堆栈数据项 C 被写入内存 280。参照图示中 265 所标示，显示了在 PUSH O 运算后的状况，在这考量下，O 的数据项内容被推入堆栈顶端(进入堆栈模拟缓存器中的缓存器 R0)。然而在转移数据项 O 进入缓存器之前，其它的数据项必需被移动以空出空间给该数据项。如此导致数据项值 C 被从可变缓冲区 240 的底部转移至内存 280 的堆栈部分 282 的顶端。如图标，堆栈部分 282 可如任何传统堆栈随意地延展或增长至内存 280 中。堆栈数据项 K 被从堆栈模拟缓存器的 R3 位置转移到可变缓冲区 B0

位置，而可变缓冲区中其它的内容依此下移。

假设有三个连续的推出(POP)运算 293 被执行，包含堆栈顶端的堆栈模拟缓存器 262 顶端三个项目会被移出，在堆栈模拟缓存器 262 与可变缓冲区 240 中的数据项则被往上移动至这些堆栈位置。这会使得数据项 K、J 与 I 被从可变缓冲区 240 转移至堆栈模拟缓存器 R1、R2 与 R3，如此可变缓冲区 240 底部三个位置(B5、B6 与 B7)便没有被用到。

如上所述，在本具体实施例中，只要超过四个项目存在可变缓冲区 240 中，则内存 280 的堆栈部分 282 的堆栈内容便不送至可变缓冲区 240。因此，堆栈数据项 C、B 与 A 仍分别在堆栈位置 S1、S2 与 S3。然而，如果另一个推出(POP)运算 294 被执行时，数据项 L 会被从堆栈模拟缓存器的 R0 位置移出，并且缓存器 R1、R2 与 R3 中的内容 K、J 与 L 会被往上移动。可变缓冲区 240 顶端的内容 H 则会被转移至堆栈模拟缓存器 262 的 R3 位置，使得可变缓冲区 240 中只有四个堆栈数据项，这四个项目为 G、F、E 与 D。因为只有四个项目在可变缓冲区中，根据本发明的一具体实施例，在内存 280 的堆栈部分 282 顶端的堆栈数据项 C 被检索并移至可变缓冲区 240 的 B4 位置。

图 3 所示的运算显然地仅为可实施用来与本发明的领域与精神一致的数种具体实施例中的一种，例如可变缓冲区 240 与内存 280 间的数据项传输不一定一次一个项目(如字节或字符)，可变缓冲区 240 与内存 280 间的数据项传输可以是一群，在此考量下，两个、四个、甚至更多的数据项可以一次传输。更进一步，可便缓冲区 240 可与图 3 不同尺寸如可便缓冲区 240 的尺寸可存放十六个堆栈数据项逻辑电路 245(图 2)可被配置来使得 12 个或更多的堆栈数据项储存在可变缓冲区中，在收到一个推入运算时，一次将四个堆栈数据项传输至内存 280 的堆栈部分 282，并且相对于推出运算，一次将四个堆栈数据项从内存 280 的堆栈部分 282 读出至缓冲区 240 中四个可用的位置。

另外需要注意的是，在缓冲区 240 与内存 280 间的转移较偏好在关键路径(critical path)的外，也就是当数据项被处理器从堆栈推出或推入时，能够让在初始处理(initial transaction)上要尽量可能的快(例如当数据项由处理器管线至堆栈或由堆栈至处理器管线被处理出去)，使得管线运

算能够持续。参照图 2，在可变缓冲区 240 与内存 280 的数据项处理可能发生在其它时候，使得在两个装置间的处理，并未中断管线的数据项流通。

参考图 4，其举例了本发明的一具体实施例于上层运算的流程示意图。相应于一堆栈运算 302，一方法可决定涉及数据项推入堆栈或数据项推出堆栈的运算 304。如果运算涉及将数据项推出堆栈，本具体实施例可立刻从堆栈顶端的堆栈缓存器 R0 推出或拉出一堆栈数据项，堆栈缓存器 R0 为内存顶端位置(步骤 306)。紧接着在步骤 306 之后，其余堆栈模拟缓存器的内容会朝上被移动一个位置，之后的运算被进行来决定目前是否有堆栈数据项在缓冲区 240 中(步骤 308)。如果没有，便没有更进一步的运算需要被执行。然而，如果在缓冲区 240 中有堆栈数据项，则缓冲区 240 顶端项目被移入可用的堆栈模拟缓存器 R3(步骤 310)，这样会让堆栈模拟缓存器尽可能地随时都在装满的状态，使得堆栈模拟缓存器的运用达到最大。接下来的运算是对维持在缓冲区 240 内堆栈数据项的数量进行最佳化，在这考量下，于本发明的一具体实施例中，若缓冲区 240 中少于四个项目时，则在内存的堆栈部分中有效的堆栈数据项可以被移至缓冲区 240。因此，该方法判断是否缓冲区内的堆栈数据项是否少于四个(步骤 312)，如果不是，则不需要进行下一步。然而如果在缓冲区内的项目少于五个，则该方法判断是否在内存 280 的堆栈部分 282 中目前是否有堆栈数据项(步骤 314)。如果有，则一个(或多个)堆栈数据项会被从内存 280 被移入缓冲区 240 中(步骤 315)。

在步骤 304 的评估中所判断堆栈运算为推入运算时，一堆栈数据项会被推入堆栈中，然而在推入一个项目进入堆栈模拟缓存器前，必需先将空间挪出来(如果堆栈模拟器装满了时)。因此在一具体实施例中，该方法对堆栈模拟缓存器是否全满做出判断(步骤 320)，如果没有装满，该方法会立刻将新堆栈数据项推入在堆栈模拟缓存器中的堆栈顶端(步骤 322)。当然，其余堆栈模拟缓存器的内容在数据项被推入堆栈位置顶端时不会被覆盖。然而如果步骤 320 判断出堆栈模拟缓存器是完全被有效堆栈数据项所占据，则在堆栈模拟缓存器底部位置的堆栈数据项会被移至可变缓冲器 240，以空出空间来让新数据项被推入堆栈模拟缓存器。然而，

在数据项从堆栈模拟缓存器移至缓冲区 240 前，该方法会先判断缓冲区 240 中是否有可用空间。在此考量下，该方法可评估来判断缓冲区是否是满的(如目前是否有 8 个项目存在缓冲区 240 中)(步骤 325)。如果没有，则在堆栈模拟缓存器底部位置中的项目会被移至可变缓冲区 240 内的一个可用位置(未显示于图标)。

在此考量下，如图 3 所示，任何存在于可变缓冲区 240 的项目会被往下移动一个位置，以空出在缓冲区 240 的最顶端的可用空间，来接收从堆栈模拟缓存器 R3 传送的数据项。如果步骤 325 判断出可变缓冲区事实上是满的，缓冲区最底端项目会被移到内存 280 的堆栈部分 282(步骤 327)。在将数据项移入内存后，缓冲区中剩下的数据项会被移动，以空出空间让来自堆栈模拟缓存器的数据项移入缓冲区 240。接下来，在缓存器 R3 的数据项会被移入可变缓冲区的缓冲区位置 B0(步骤 328)。再接下来，由于堆栈模拟缓存器给新数据项的可用空间已空出，应该被推入堆栈的新数据项便会被推入堆栈模拟缓存器(步骤 322)。

虽然上述具体实施例所描述的是一些在新堆栈数据项分别在被推入或推出堆栈而向下或向上有效移动堆栈数据项的具体实施例，凡与本发明的领域与精神一致的其它方式亦可依此施行。例如，除了在堆栈模拟缓存器的不同缓存器移动堆栈数据项，或在可变缓冲区 240 的不同位置移动堆栈数据项外，也可以用指针来指向一连串的位置以代表最高(或最低)的一些堆栈位置。这样的实作会避免在缓存器间或其它数据项位置间做不必要的数据项移动，在时间上较有效率。在此考量下，在图 3 所示的具体实施例与其中的描述是为了帮助得自本发明的具体实施例的相关优点的呈现，其中本发明的具体实施例是通过由可变缓冲区 240 的运作来达成。

显然地，依照上面实施例中的描述，本发明可能有许多的修正与差异。因此需要在其附加的权利要求项的范围内加以理解，除了上述详细的描述外，本发明还可以广泛地在其它的实施例中施行。上述仅为本发明的较佳实施例而已，并非用以限定本发明的申请专利范围；凡其它未脱离本发明所揭示的精神下所完成的等效改变或修饰，均应包含在下述申请专利范围内。

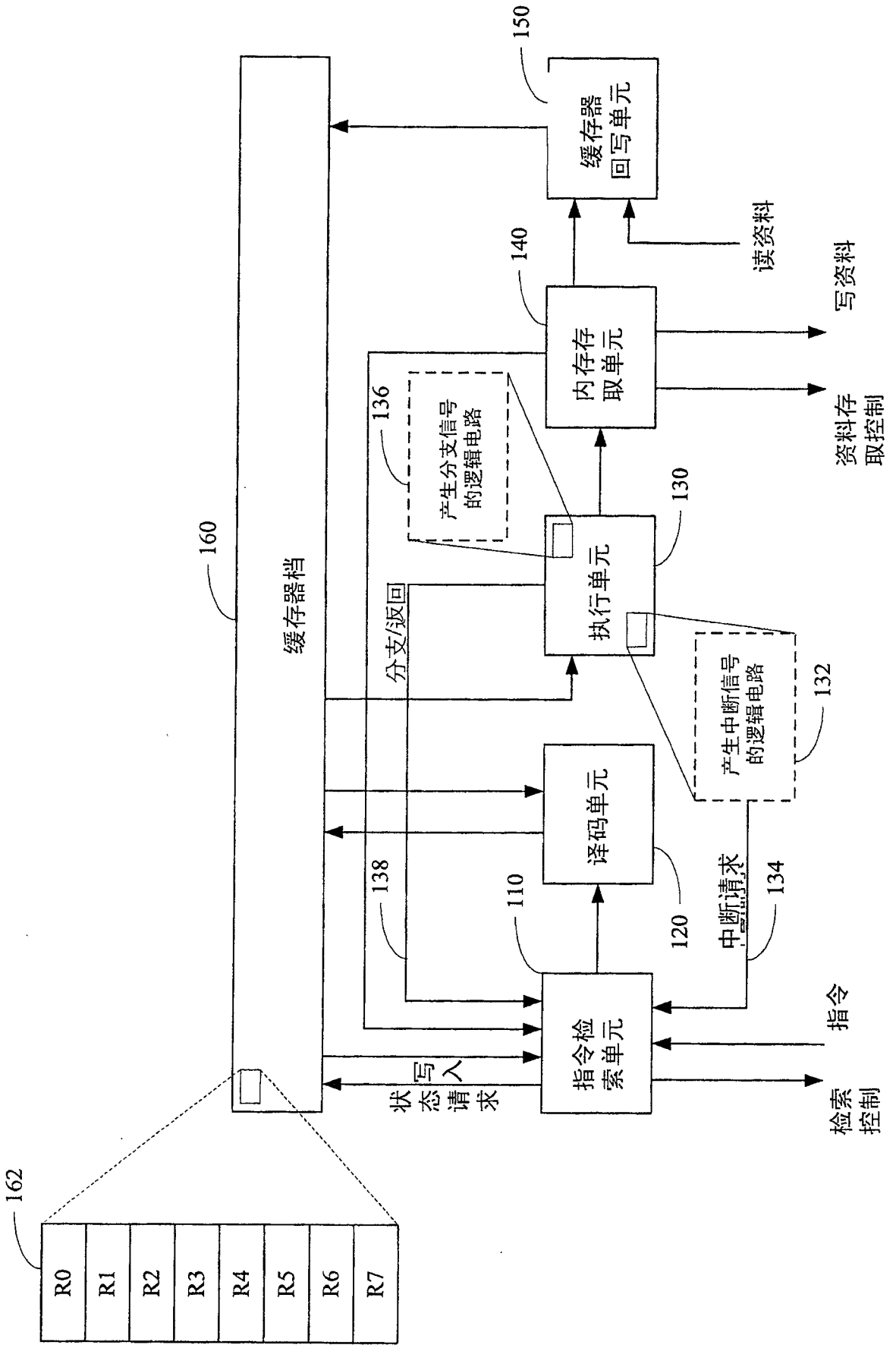

缓存器回写单元
150
读资料
写资料
内存存取单元
140
产生分支信号的逻辑电路
136
资料存取控制
执行单元
130
缓存器档
160
分支/返回
产生中断信号的逻辑电路
132
译码单元
120
中断请求
134
138
110
写入状态请求
指令检索单元
指令
检索控制
162
R0
R1
R2
R3
R4
R5
R6
R7

图 1A

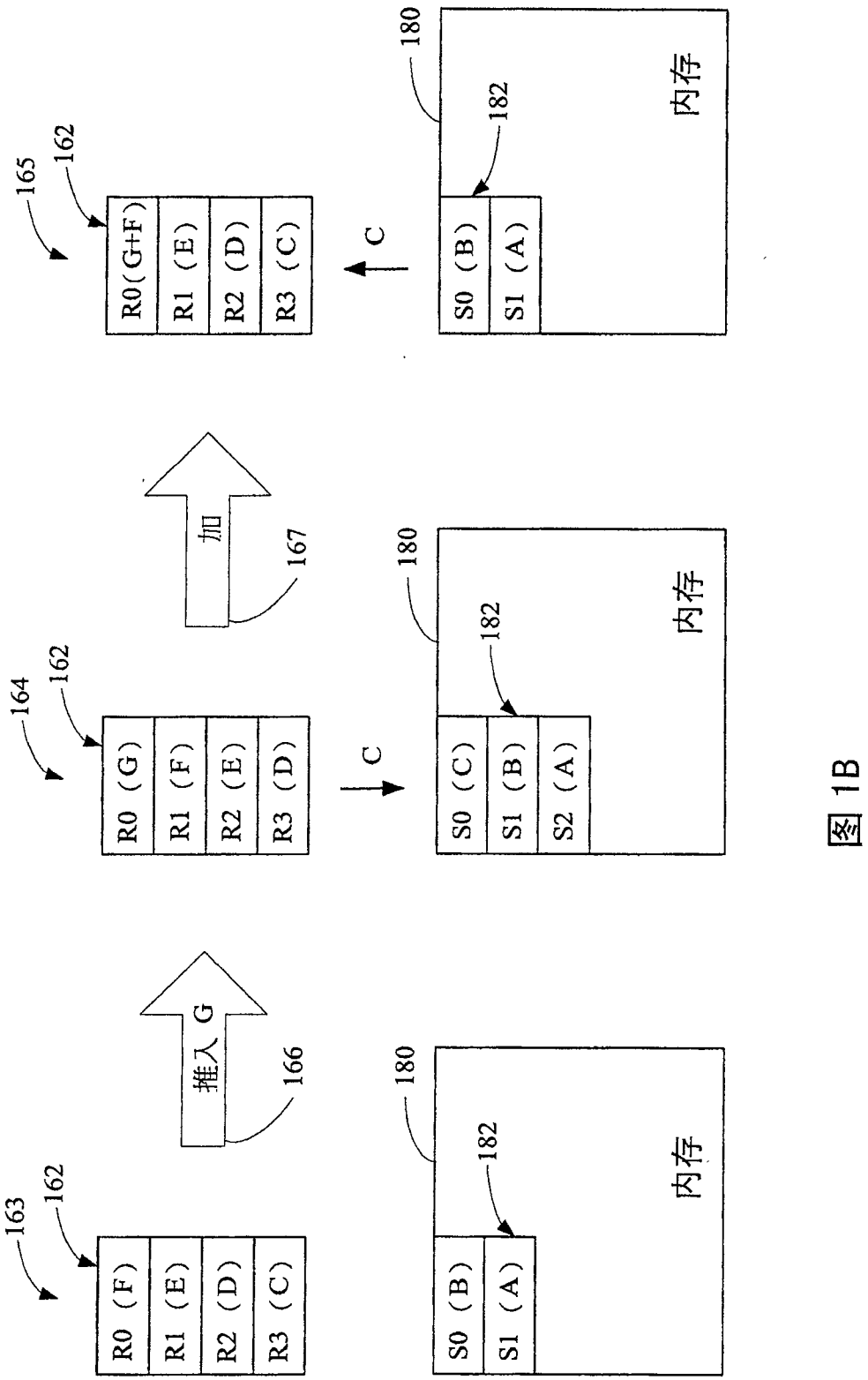

163
162
R0 (F)
R1 (E)
R2 (D)
R3 (C)
推入 G
166
180
182
S0 (B)
S1 (A)
内存
164
162
R0 (G)
R1 (F)
R2 (E)
R3 (D)
C
加
167
180
182
S0 (C)
S1 (B)
S2 (A)
内存
165
162
R0(G+F)
R1 (E)
R2 (D)
R3 (C)
C
180
182
S0 (B)
S1 (A)
内存

图 1B

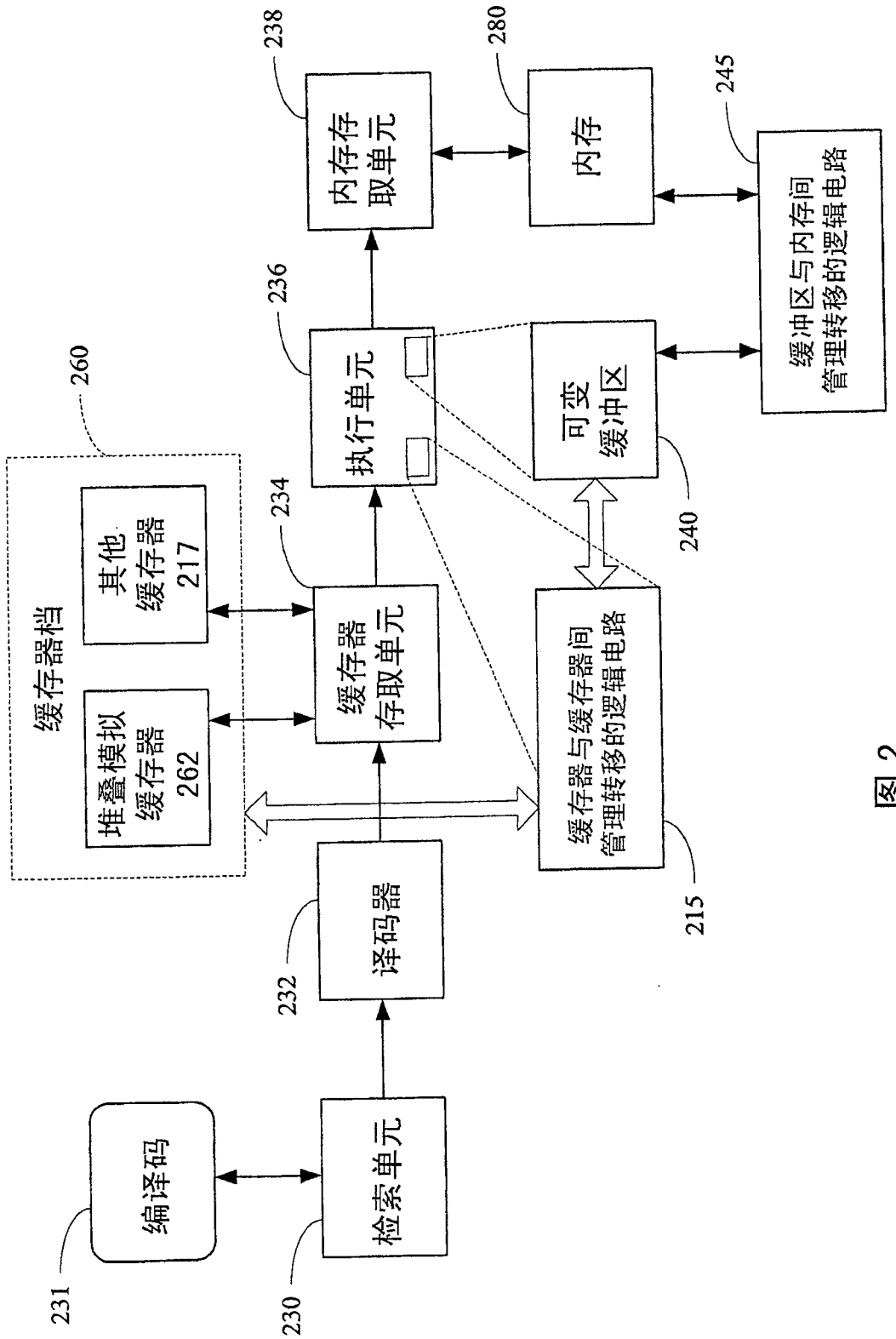

编译码
检索单元
译码器
缓存器存取单元
执行单元
内存存取单元
内存
缓冲区与内存间管理转移的逻辑电路
可变缓冲区
缓存器与缓存器间管理转移的逻辑电路
缓存器档
堆叠模拟缓存器262
其他缓存器217
231
230
232
234
236
238
280
245
240
215
260

图 2

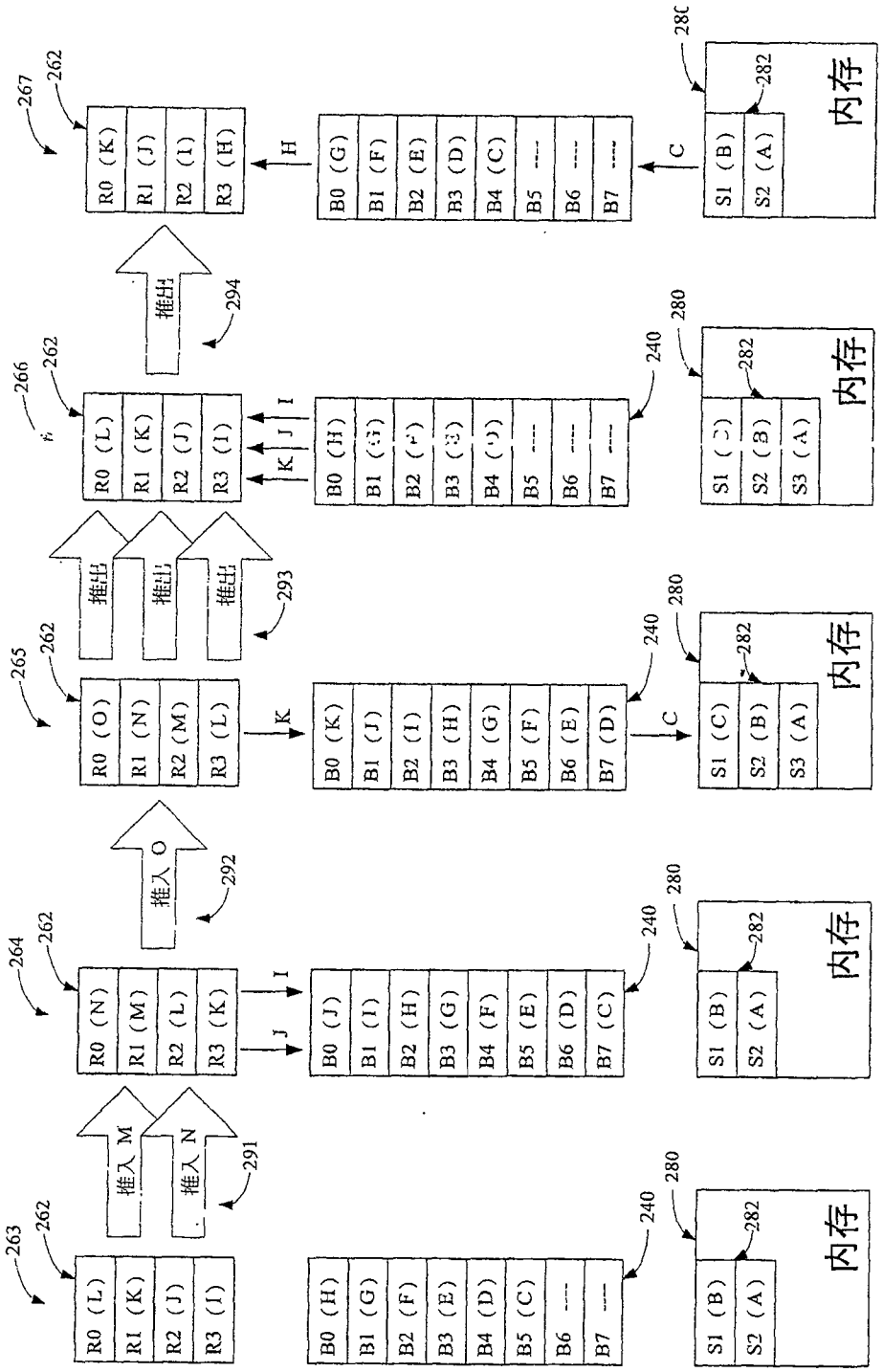

263
262
R0 (L)
R1 (K)
R2 (J)
R3 (I)
推入 M
推入 N
291
264
262
R0 (N)
R1 (M)
R2 (L)
R3 (K)
I
J
B0 (J)
B1 (I)
B2 (H)
B3 (G)
B4 (F)
B5 (E)
B6 (D)
B7 (C)
240
280
S1 (B)
S2 (A)
282
内存
推入 O
292
265
262
R0 (O)
R1 (N)
R2 (M)
R3 (L)
K
B0 (K)
B1 (J)
B2 (I)
B3 (H)
B4 (G)
B5 (F)
B6 (E)
B7 (D)
C
S1 (C)
S2 (B)
S3 (A)
推出
293
266
262
R0 (L)
R1 (K)
R2 (J)
R3 (I)
K
J
I
B0 (H)
B7 ---
S1 (C)
S2 (B)
S3 (A)
推出
294
267
262
R0 (K)
R1 (J)
R2 (I)
R3 (H)
H
B0 (G)
B7 ---
C
S1 (B)
S2 (A)

图 3

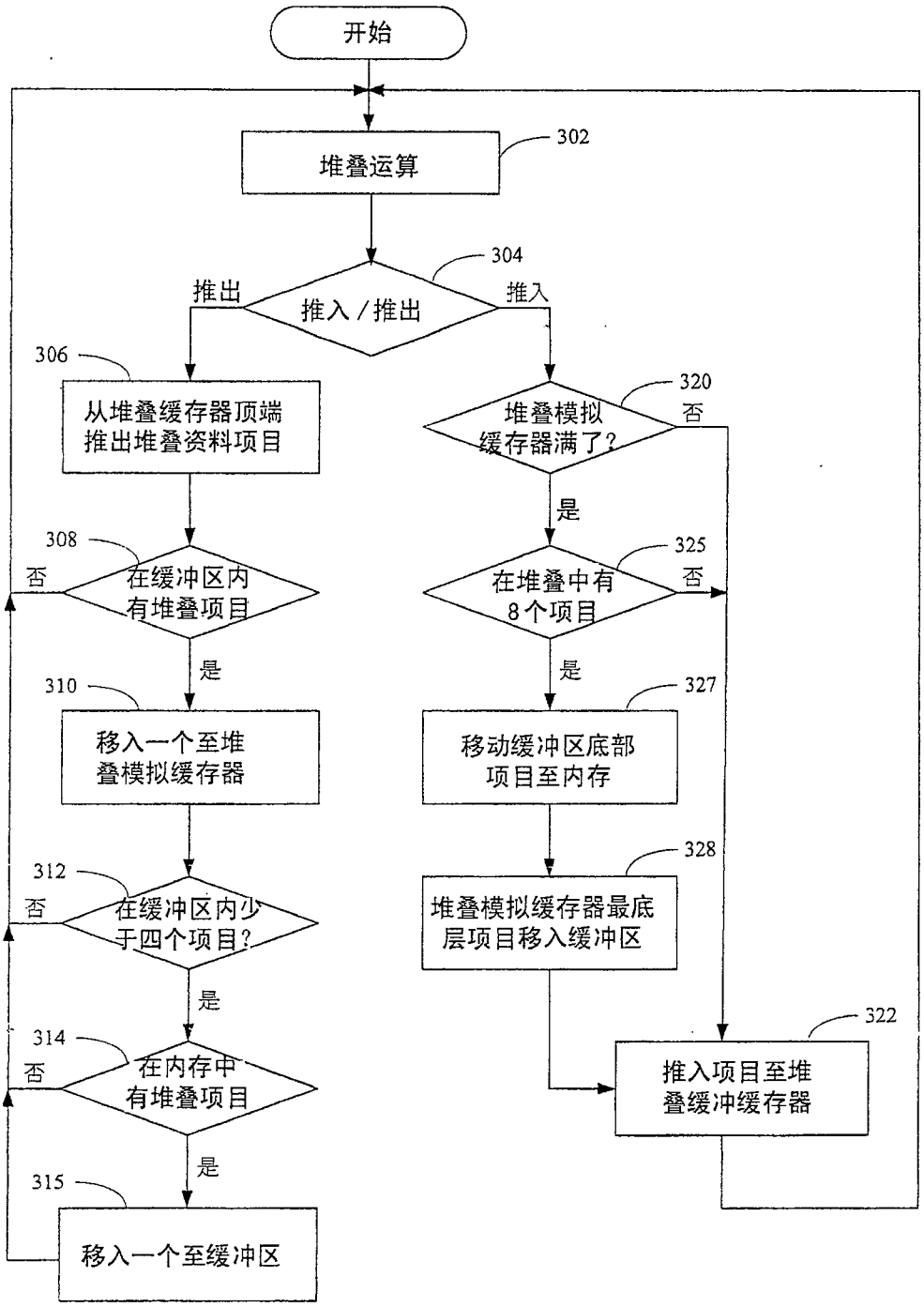

开始
堆叠运算
302
304
推入/推出
推出
推入
306
从堆叠缓存器顶端
推出堆叠资料项目
308
在缓冲区内
有堆叠项目
否
是
310
移入一个至堆
叠模拟缓存器
312
在缓冲区内少
于四个项目?
否
是
314
在内存中
有堆叠项目
否
是
315
移入一个至缓冲区
320
堆叠模拟
缓存器满了?
否
是
325
在堆叠中有
8个项目
否
是
327
移动缓冲区底部
项目至内存
328
堆叠模拟缓存器最底
层项目移入缓冲区
322
推入项目至堆
叠缓冲缓存器

图 4