I733746

(19)中華民國智慧財產局

(12)發明說明書公告本 **(11)證書號數：TW I733746 B**

(45)公告日： 中華民國 110 (2021) 年 07 月 21 日

(21)申請案號：106102675 (22)申請日： 中華民國 106 (2017) 年 01 月 24 日

(51)Int. Cl.： ***G06F9/30*** ***(2018.01)***

(30)優先權：2016/02/24 美國 15/052,765

(71)申請人：美商英特爾股份有限公司 (美國) INTEL CORPORATION (US)
美國

(72)發明人：華希 麥特 WALSH, MATT S. (US)；基德 泰勒 KIDD, TAYLOR W. (US)

(74)代理人：林志剛

(56)參考文獻：

TW	I427533	TW	I467477
US	2008/0140750A1	US	2013/0067196A1
US	2015/0339110A1		

審查人員：林琮烈

申請專利範圍項數：20 項 圖式數：7 共 39 頁

(54)名稱

在運行時最佳化指令的處理器、由處理器在運行時最佳化指令的方法及非暫態機器可讀媒體

(57)摘要

揭示相關於在運行時將指令情境最佳化的處理器之方法及設備。在一實施例中，處理器包括提取電路，用以從指令儲存器提取指令，該指令之格式包括運算碼、第一來源運算元識別符、及第二來源運算元識別符；其中該指令儲存器包括一序列之次最佳化指令，在其之前係序列開頭之指令且在其之後係序列終端之指令。本揭示處理器進一步包括解碼電路，用以解碼指令、用以偵測序列開頭之指令與序列終端之指令、用以緩衝在其間的該序列之次最佳化指令、用以存取查找表以識別一或多最佳化指令以替代該序列之次最佳化指令的一或多者、及用以選擇解碼指令或該序列之一或多最佳化指令以調度到執行電路。

Methods and apparatuses relating to processors that contextually optimize instructions at runtime are disclosed. In one embodiment, a processors includes a fetch circuit to fetch an in-struction from an instruction storage, a format of the instruction including an opcode, a first source operand identifier, and a second source operand identifier; wherein the instruction storage includes a sequence of sub-optimal instructions preceded by a start-of-sequence instruction and followed by an end-of-sequence instruction. The disclosed processor further includes a decode circuit to decode the instruction, to detect the start-of-sequence instruction and the end-of-sequence instruction, to buffer the sequence of sub-optimal instructions there between, to access a lookup table to identify one or more optimized instructions to substitute for one or more of the sequence of sub-optimal instructions, and to select either the decoded instruction or the sequence of one or more optimized instructions to dispatch to an execution circuit.

指定代表圖：

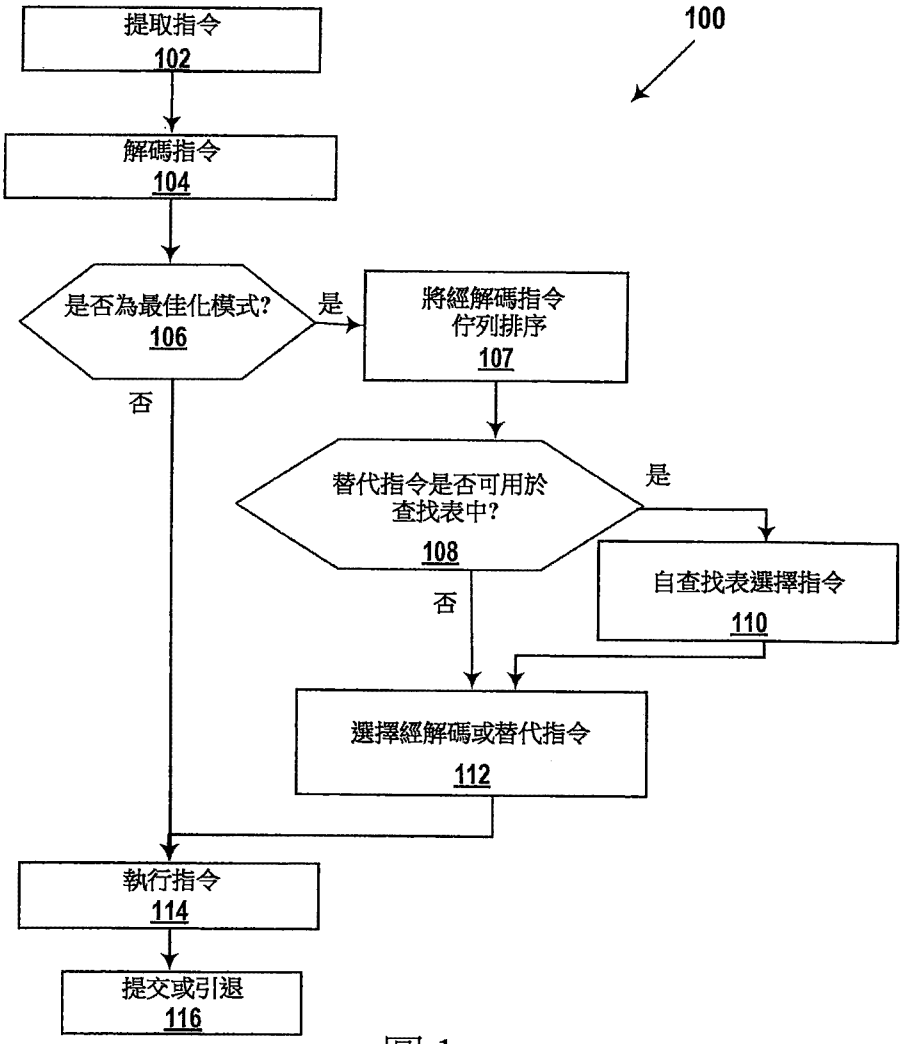

100
提取指令
102
解碼指令
104
是否為最佳化模式?
106
是
將經解碼指令
佇列排序
107
否
替代指令是否可用於
查找表中?
108
是
自查找表選擇指令
110
否
選擇經解碼或替代指令
112
執行指令
114
提交或引退
116

圖 1

專利案號：106102675

公告本　民國 109 年 12 月 1 日修正

I733746

發明摘要

※申請案號：106102675

※申請日：106 年 01 月 24 日　**※IPC 分類：**

【發明名稱】(中文/英文)

在運行時最佳化指令的處理器、由處理器在運行時最佳化指令的方法及非暫態機器可讀媒體

PROCESSOR TO OPTIMIZE INSTRUCTIONS AT RUN-TIME, METHOD OF OPTIMIZING INSTRUCTIONS BY A PROCESSOR AT RUN-TIME AND NON-TRANSITORY MACHINE-READABLE MEDIUM

【中文】

揭示相關於在運行時將指令情境最佳化的處理器之方法及設備。在一實施例中，處理器包括提取電路，用以從指令儲存器提取指令，該指令之格式包括運算碼、第一來源運算元識別符、及第二來源運算元識別符；其中該指令儲存器包括一序列之次最佳化指令，在其之前係序列開頭之指令且在其之後係序列終端之指令。本揭示處理器進一步包括解碼電路，用以解碼指令、用以偵測序列開頭之指令與序列終端之指令、用以緩衝在其間的該序列之次最佳化指令、用以存取查找表以識別一或多最佳化指令以替代該序列之次最佳化指令的一或多者、及用以選擇解碼指令或該序列之一或多最佳化指令以調度到執行電路。

【英文】

Methods and apparatuses relating to processors that contextually optimize instructions at runtime are disclosed. In one embodiment, a processors includes a fetch circuit to fetch an instruction from an instruction storage, a format of the instruction including an opcode, a first source operand identifier, and a second source operand identifier; wherein the instruction storage includes a sequence of sub-optimal instructions preceded by a start-of-sequence instruction and followed by an end-of-sequence instruction. The disclosed processor further includes a decode circuit to decode the instruction, to detect the start-of-sequence instruction and the end-of-sequence instruction, to buffer the sequence of sub-optimal instructions there between, to access a lookup table to identify one or more optimized instructions to substitute for one or more of the sequence of sub-optimal instructions, and to select either the decoded instruction or the sequence of one or more optimized instructions to dispatch to an execution circuit.

【代表圖】

【本案指定代表圖】：第(1)圖。

【本代表圖之符號簡單說明】：無

【本案若有化學式時，請揭示最能顯示發明特徵的化學式】：無

第 106102675 號　　民國 109 年 12 月 1 日修正

發明專利說明書

（本說明書格式、順序，請勿任意更動）

【發明名稱】(中文/英文)

在運行時最佳化指令的處理器、由處理器在運行時最佳化指令的方法及非暫態機器可讀媒體

PROCESSOR TO OPTIMIZE INSTRUCTIONS AT RUN-TIME, METHOD OF OPTIMIZING INSTRUCTIONS BY A PROCESSOR AT RUN-TIME AND NON-TRANSITORY MACHINE-READABLE MEDIUM

【技術領域】

本文所述實施例一般相關於處理器。更明確而言，所述實施例一般相關於經組態以在運行時將指令情境最佳化之處理器。

【先前技術】

並行處理通常比一次一資料點之純量執行更快。具有對多資料點執行相同操作的多處理元件之單指令多資料（SIMD）電腦達到效能增益，其係藉由利用多並行執行核心之並行與同時使用之優勢。

SIMD 處理器利用執行數學運算及移動資料的平行性之優勢。SIMD 處理器可同時地載入或儲存多資料項，相較於一次載入或儲存一資料之較緩慢的純量處理器，結果導致效能增益。當對具有並行資源之處理器執行電腦程式

時，運用 SIMD 指令提供比運用純量指令更佳之效能。

然而使用 SIMD 指令集架構（ISA）程式化可能係具有挑戰的。舉例而言，SIMD ISA 通常為處理器專用。使用 SIMD 指令之程式可能需要被重新編寫及客製化以適用於新的處理器世代。將純量指令適用於新的指令集架構所需要的工作，包括重新編寫該碼，記錄該碼；賦能編譯器發出該碼，訓練使用者使用該碼，及用以除錯並收集碼執行之追蹤，可能需要部分或全部被重複用於以各個新世代的指令集架構（例如 MMX、SSE、SSE2、SSE3、SSE4、AVX、AVX2、AVX 3.1、及 AVX 3.2）再使用。因此，所需要的係一種允許程式設計師可以利用 SIMD 指令集架構又同時避免在習知解決方案中固有之挑戰的方式。

此外，由於習知解決方案提前靜態地將碼最佳化，而非在執行期間動態地將碼最佳化，故其係受限的。編譯器嘗試將特定碼序列之執行最佳化，但其在靜態環境中操作，而未清楚機器或暫存器之狀態。即使傳統地手動編碼之 SIMD 碼亦無法根據機器與暫存器之運行期間狀態而將碼最佳化。因此，所需要的係一種在知悉暫存器之狀態與其內容的前提下在運行期間將指令最佳化的方法。

【圖式簡單說明】

圖 1 係根據一實施例說明用於處理器在運行時情境化地最佳化指令的程序之方塊流程圖。

圖 2 係根據一實施例說明由處理器在運行時情境化地

最佳化指令所使用的處理組件之方塊流程圖。

圖 3 根據一實施例說明指令及其各種欄位。

圖 4 根據實施例說明藉由將表 2 之 Vectorbeam 碼最佳化的處理器所執行之例示性暫存器分配。圖 5 根據一實施例說明用以並行處理圖 4 之經分配暫存器所使用之多計算資源之例示性分配。

圖 6 依據一實施例說明表列對於純量指令之向量指令替代的查找表之一部分。

圖 7 根據一實施例說明向量處理器暫存器檔案。

【發明內容與實施方式】

在以下說明中陳述數種特定細節。然而，應瞭解本揭示之實施例可被操作而不具有此些特定細節。在其他情況中，已知電路、結構及技術並未被詳細顯示以避免模糊對本說明之理解。

在本說明書中對「一實施例」、「各實施例」、「例示實施例」等之參照指示該所述實施例可包括特定特徵、結構、或特性，但每一個實施例不必然需要包括該特定特徵、結構、或特性。此外，此類詞語不必然係在參照同一個實施例。此外，當特定特徵、結構、或特性係結合實施例而被描述時，無論有無明確描述，對結合其他實施例之此類特徵、結構、或特性做出改變係被理解為在該領域中具有通常知識者之知識內的。

一（例如，硬體）處理器、或一組的處理器執行來自

指令集（例如，指令集架構（ISA））之指令。指令集為相關於程式化之電腦架構之部分，且一般包括本機（native）資料類型、指令、暫存器架構、定址模式、記憶體架構、中斷及異常處理、以及外部輸入及輸出（I/O）。應注意本文術語「指令」可指稱巨集指令，例如被提供到處理器用於執行之指令。處理器（例如，具有一或多核心以解碼及/或執行指令）可對資料操作，例如，執行算數、邏輯、資料移動或其他功用。

運行時之情境最佳化

根據本揭示，處理器在運行時將指令情境最佳化。更明確地，處理器在運行時動態地將一序列之次最佳化指令最佳化。本文使用之次最佳化指令係指使用少於全部處理器可用資源之指令，或其可經最佳化以利用處理器之指令集架構。在一實施例中，數個序列之次最佳化指令被儲存在指令儲存器中，且被開頭定界及終端定界之指令包圍。一般而言，為揭示之目的，被序列開頭及序列終端之指令包圍的該序列之次最佳化碼在本文內被稱為 Vectorbeam 碼。本文之申請專利範圍並未受限於 Vectorbeam 之名稱。在替代實施例中，可藉由不同名稱來引用該碼。Vectorbeam 指令並未受限於其特定運算碼或助憶符之選擇。

根據揭示實施例，處理器經組態以偵測 Vectorbeam 碼序列之次最佳化指令，以緩衝處理該序列，及以存取查

找表以識別一或多指令以替換並將該 Vectorbeam 碼序列之次最佳化指令之一或多者最佳化。

定界符：序列開頭與序列終端

在一實施例中，序列開頭之指令及序列終端之指令可被選擇以提供暗示給處理器，該暗示有關如何將碼序列最佳化。在一實施例中，處理器可臆測該序列之次最佳化指令目的係為迭代循環（iterative loop），這是因為該序列之次最佳化指令之前係序列開頭之定界指令（「foreach」），且之後係序列終端之定界指令（「next」）。相似地在替代實施例中，序列開頭之指令可暗示該序列之次最佳化指令係迭代式的，其係藉由使用定界指令（「foreach」、或「do until」、或「repeat until」、或「do while」）而迭代。序列終端之指令可相似地被命名以提示其功用（例如，「end」、「exit」、「stop」、「continue」、「return」）。序列開頭與序列終端之指令亦可具有不是人類可讀取之預判定值、或係數字的、或被自動或隨機地選擇而無人干涉。

序列之次最佳化指令

如本文使用的一序列之次最佳化指令意指利用少於全部的處理器中可用的並行資源之計算指令，該些指令係意在被執行於該處理器上。

舉例而言，該序列之次最佳化指令可為迭代式、純量

循環，其如所撰寫的使用比處理器中可用資源更少的並行資源。

作為進一步實例，該序列之次最佳化指令可被撰寫以使用 64 位元之暫存器，但處理器上那些將被執行之指令具有 128 位元之暫存器可用。

作為進一步實例，該序列之次最佳化指令相關於多媒體操作，諸如將像素之螢幕的明亮度調暗。在此類實例中，純量數學運算可被寬的向量指令替代。

元資料

根據本揭示，序列開頭之指令可包括將元資料提供到處理器之運算元，該處理器將實作該 Vectorbeam 碼序列。在一實施例中，元資料給予處理器有關如何將碼最佳化之一暗示。在一實施例中，該序列之次最佳化指令之前係序列開頭之指令「for each rax, 0, 64, 1」。元資料「rax, 0, 64, and 1」提供處理器一暗示，其有關該次最佳化碼應：使用暫存器 rax 作為循環索引；使 rax 變化 64 次；從 0 開始；針對每迭代具有 1 的跨距，而被循環。

說明性實施例之詳細說明

圖 1 係根據一實施例說明用於處理器在運行時情境化地最佳化指令的程序之方塊流程圖。更明確地，根據方塊流程圖 100，經組態以執行程序以在運行時將指令情境化地最佳化之處理器在 102 提取指令，在 104 解碼指令，並

在 106 測試其是否在最佳化模式中。在一實施例中，將被提取之指令被儲存在指令緩衝器中。在替代實施例中，指令可被儲存在指令暫存器、通用暫存器、程式堆疊、或包括靜態及動態隨機存取記憶體之記憶體中。執行流程圖 100 之程序的處理器在 104 將經提取之指令解碼，以產生一或多微操作、微碼進入點、微指令、其他指令、或其他控制訊號作為輸出。

根據本揭示，由一序列之次最佳化指令組成的 Vectorbeam 碼序列被儲存在記憶體中，在其之前係序列開頭之指令且在其之後係序列終端之指令。在一實施例中，序列開頭之指令可被編碼於人類可讀取之組合碼，並具有助憶符，其經定義以使得以建議該次最佳化指令應如何被最佳化。舉例而言，序列開頭之指令可能係「for each」、或「do until」、或「repeat until」、或「do while」，上述僅為舉例。序列終端之指令可相似地被命名以提示其功用（例如，「end」、「exit」、「stop」、「continue」、「return」、「abort」、「undo」、或「commit」）。序列開頭與序列終端之指令亦可具有不是人類可讀取之預判定值、或係數字的、或被自動或隨機地選擇而無人干涉。

在一實施例中，Vectorbeam 碼之次最佳化碼序列被撰寫以模仿純量助憶符，並呈現為具有純量及/或序列之執行流，使其易於瞭解及除錯。以易於瞭解之定界符將該次最佳化碼序列定界可增強碼之可讀性，且亦可建議該處理

器如何在運行時將碼最佳化。

若在 106 處理器判定其未在最佳化模式中運行，則其在 114 調度將被執行的經解碼指令，並在 116 使該指令提交或引退。

當在 104 所解碼之指令係序列開頭之指令時，如上述，在一實施例中進入最佳化模式在運行期間發生。在另一實施例中，處理器在通電時或在重啟後進入最佳化模式。在另一實施例中，在 102 提取特定序列之指令、在 104 將其解碼、並偵測到一或多次最佳化指令（諸如純量指令、或來自較舊指令集架構之指令）後，處理器隱含的進入最佳化模式。在替代實施例中，回應於預定義運行期間事件，處理器隱含的在運行期間進入最佳化模式。

當在 104 所解碼之指令係序列終端之指令時，如上述，在一實施例中退出最佳化模式在運行期間發生。在替代實施例中，在諸如針對一時間識別替代指令失敗的一條件之發生後，處理器隱含的退出最佳化模式。在替代實施例中，處理器無期限地保持在最佳化模式。

若在 106 處理器判定其已經在最佳化模式中，或假設該處理器偵測到序列開頭之指令，則其在 107 將該經解碼指令佇列排序到指令緩衝器中。在一實施例中的指令緩衝器係一系列之通用暫存器。在其他實施例中，指令緩衝器可運用指令暫存器、純量暫存器、移位暫存器、堆疊記憶體、靜態或動態 RAM 記憶體、或快取記憶體。

在操作中並在運行期間，在 108 處理器將指令緩衝器

中的指令與查找表做比較，以判定適當替代指令是否為可用，該等指令包括其將產出較佳效能之替代指令。在 108 評估適當替代指令是否為可用時，在一實施例中的處理器將由元資料引導，該元資料與序列開頭之定界符被包括。在 108 評估適當替代指令是否為可用時，在一實施例中的處理器將由暫存器檔案之狀態與內容引導，將參照圖 7 而有進一步說明。

以下參照表 1 及表 2 說明被元資料引導並實作 Vectorbeam 碼之本揭示處理器之實例。在一實施例中，Vectorbeam 碼之次最佳化指令包含純量指令，並且該替代指令包含向量指令或 SIMD 指令。在一實施例中，Vectorbeam 碼之次最佳化指令包含來自較舊世代處理器之指令集架構的指令，並且該替代指令係選自較近期處理器之指令集。在一實施例中，Vectorbeam 碼之該序列之次最佳化指令由指令之迭代式循環所組成，且該替代指令以較少之循環完成該操作，或使用向量指令以完成該操作。在另一實施例中，在指令緩衝器中的該序列之次最佳化指令由條件式循環組成，且該替代指令以較少之循環完成該操作，或使用向量指令以完成該操作。若在 108 該處理器判定產出較佳效能之替代指令係可用的，則其在 110 選擇一或多替代指令。

在 112，若在 108 無可用的替代指令，則處理器選擇經解碼指令用於執行，或選擇在 110 所選的該一或多替代指令。在一實施例中，處理器保持一序列之指令於指令緩

衝器中，並當在 108 評估潛在替代時考慮多指令。在另一實施例中，處理器限制儲存在指令緩衝器中的指令數，並且假若該指令數超越該限制，則該處理器自指令緩衝器移除一或多指令並將其調度為將在 114 被執行。

在 114，處理器執行經解碼指令或一或多替代指令。在執行後，在 116 處理器將該指令引退或提交，確保執行結果被寫入或已寫入到其目的地，並釋放或釋出資源供稍後使用。

圖 2 係根據一實施例說明由處理器在運行時情境化地最佳化 Vectorbeam 碼指令所使用的處理組件之方塊圖。更明確地，方塊圖 200 包括指令儲存器 202、提取電路 204、解碼電路 206、執行電路 220、及引退/提交電路 226。解碼電路 206 包括解碼邏輯 208、最佳化模式偵測器 210、指令緩衝器 212、查找表 214、替代評估邏輯 216、及指令選擇器 218。方塊圖 200 進一步包括執行電路 220、暫存器 222、記憶體 224、及引退/提交電路 226。

在操作中，處理器在運行期間運用提取電路 204 以自指令儲存器 202 提取指令。在一實施例中，指令儲存器 202 係暫存器檔案。在替代實施例中，指令儲存器 202 可為指令緩衝器、指令暫存器、通用暫存器、程式堆疊、或靜態或動態隨機存取記憶體。

處理器將所提取指令傳到解碼電路 206，其運用解碼邏輯 208 將所提取指令解碼。包括其運算碼及可選運算元

之指令將在以下參照圖 3 而被說明。處理器使用最佳化模式偵測器 210 來偵測其是否在最佳化模式中。假若處理器並未在最佳化模式中，則處理器調度經解碼指令到執行電路 220。

但假若最佳化模式偵測器 210 偵測到該處理器係在最佳化模式或將該處理器設定成在最佳化模式，則經解碼指令被佇列於指令緩衝器 212 中。如所示，指令緩衝器 212 具有四個條目，但條目數係可變：其可少於四個，多於四個，或者可被動態地調整，如同在該領域中具有通常知識者所理解般。在運行期間，處理器使用替代評估邏輯 216 以評估在指令緩衝器 212 中經解碼之指令。

更明確地，在運行期間替代評估邏輯 216 存取查找表 214 以判定是否有任何可用替代指令。在一實施例中的查找表將在指令緩衝器中的次最佳化 Vectorbeam 指令之助憶符與一列表之替代指令做比較。在一實施例中，替代指令為向量指令。在另一實施例中，替代指令為 SIMD 指令。在另一實施例中，替代指令為來自較次最佳化指令來源更新的指令集架構。在評估替代物時，在一實施例中的替代評估邏輯 216 係由以序列開頭之指令所提供的元資料所引導。在評估替代物時，在一實施例中的替代評估邏輯 216 評估處理器暫存器之運行期間狀態。舉例而言，假設多暫存器被使用於判定記憶體位址，且假若該些記憶體位址落在相同快取線中，則替代評估邏輯 216 以向量記憶體存取來替代與該些記憶體位址相關之純量記憶體存取。

在一實施例中，假設替代評估邏輯 216 判定替代指令係可用的，則替代指令被傳到調度選擇器 218 以被調度用於執行。在替代實施例中，將替代指令寫入到指令緩衝器 212 中，將其增加到或替代其中之指令。如所示，經解碼電路 206 使用調度選擇器 218 以自指令緩衝器 212 或替代評估邏輯 216 選擇要調度之指令。

在一實施例中的執行電路 220 係向量處理器。在替代實施例中，執行電路 220 可包括多核心及並行硬體。在一實施例中的執行電路 220 運用暫存器 222 及記憶體 224 以儲存中間結果，並用以另外支援執行。在執行後，引退/提交電路 226 確保執行結果被寫入或已寫入到其目的地，並釋放或釋出資源用於之後的用途。

圖 3 根據實施例說明指令及其各種欄位。更明確地，指令 300 包括運算碼 302 及分別可選的第一運算元識別符 304、第二運算元識別符 306、及第三運算元識別符 308。運算碼 302 識別將被執行之指令及/或操作，以及運算元之類型（例如，「序列開頭」或「序列終端」）。第一運算元識別符 304、第二運算元識別符 306、及第三運算元識別符 308 係可選的。在一實施例中，運算碼 302 係「序列開頭」之指令且第一運算元識別符 304、第二運算元識別符 306、及第三運算元識別符 308 代表元資料，其描述之後跟隨之次最佳化指令，以提供有關如何將次最佳化指令最佳化之暗示或建議給處理器。在替代實施例中，運算碼 302 係「序列開頭」之指令，其識別跟隨在後之迭代序

列之次最佳化指令，及第一運算元識別符 304、第二運算元識別符 306、及第三運算元識別符 308 建議循環迭代數、用以改變各迭代之暫存器、及各迭代期間該變數要增加的跨距。在替代實施例中，運算碼 302 對應於「序列開頭」之指令，且第一運算元識別符 304、第二運算元識別符 306、及第三運算元識別符 308 之一或多者並未被使用。在替代實施例中，運算碼 302 對應於「序列終端」之指令，且第一運算元識別符 304、第二運算元識別符 306、及第三運算元識別符 308 並未被使用。

表 1 說明根據一實施例經最佳化的例示性序列之次最佳化指令。更明確地，表 1 之並排的碼比較說明三種版本之指令：純量碼、AVX 碼、及 Vectorbeam 碼，用以實作以下循環，標記為碼 1：

碼1.　　*for (int i = 0; i < 64; i++) { outp[i] = inp1[i] + inp2[i]; }*

純量碼	
..LOOP	標記循環
movss (%rsp,%rax,4), %xmm0	將64位元從(memory)移動到xmm0
movss 256(%rsp,%rax,4), %xmm1	將64位元從(memory+256)移動到xmm1
addss %xmm1, %xmm0	xmm0 += xmm1
movss %xmm0, 512(%rsp,%rax,4)	將64位元從xmm0移動到(memory + 512)
incq %rax	增加rax
cmpq $64, %rax	比較rax與64
jl ..LOOP	循環64次
AVX 碼	
..LOOP	標記循環
vmovups (%rsp,%rax,4), %ymm0	將256位元從(memory)向量移動到ymm0
vmovups 256(%rsp,%rax,4), %ymm1	將256位元從(memory+256)向量移動到ymm1
vaddps %ymm1, %ymm0	向量相加 ymm0 += ymm1
vmovups %ymm0,512(%rsp,%rax,4)	將256位元從ymm0向量移動到(memory + 512)
addq 8, %rax	將rax加上8
cmpq $64, %rax	比較rax與64
jl ..LOOP	循環8次
Vectorbeam 碼	
foreach rax, 0, 64, 1	序列開頭：以每次增加1之方式對rax從0循環至63
movss (%rsp,%rax,4), %xmm0	將64位元從(memory)移動到xmm0
movss 256(%rsp,%rax,4), %xmm1	將64位元從(memory+512)移動到xmm1
addss %xmm1, %xmm0	xmm0 += xmm1
movss %xmm0, 512(%rsp,%rax,4)	將64位元從xmm0移動到(memory + 512)
next	

表1.

如表 1 所示，純量碼包括將被執行 64 次之循環，其每次對暫存器 rax 增加 1。由於純量碼在其 64 循環迭代期間可能無法執行處理器之所有並行資源，故其為次最佳化的。

AVX 碼在許多態樣方面與純量碼不同。AVX 暫存器（ymm0 與 ymm1）具有不同名稱與不同大小。AVX 運算碼具有不同之名稱。特別有關於與 AVX 相關處理器之產生，係以每次 8 個而非每次 1 個來迭代通過資料。一般而言，發明純量指令（諸如表 1 中的純量指令）而言，記錄之、賦能編譯器以將之發出；訓練使用者使用它們及除錯並收集其執行之追蹤，上述所做的工作在每次指令集架構（例如，MMX、SSE、SSE2、SSE3、SSE4、AVX、AVX2、AVX 3.1、及 AVX 3.2）之新產生可能需要被部分或全部重複，以為了重複使用。

然而根據本揭示，實質上具有實質相同格式之同一純量指令被使用於 Vectorbeam 碼中。更明確地，Vectorbeam 碼本體被序列開頭之指令（在此為「foreach rax, 0, 64, 1」）與序列終端之指令包圍（在此為「next」）。如所編碼的，在序列開頭之指令中的「foreach」運算碼對實作流程 100（圖 1）之處理器指示該之後跟隨之次最佳化碼目的就是迭代碼。如所編碼的，運算元「rax」、「0」、「64」、及「 1」對實作流程 100（圖 1）之處理器提供元資料建議，以使用暫存器 rax 作為循環索引；使 rax 變化從 0 到 63；及針對每迭代具有

1 的跨距，來循環該次最佳化碼。換言之，序列開頭之指令建議實作流程 100（圖 1）之處理器以執行次最佳化碼 64 次，每次以 1 之間隔改變 rax。原始純量碼之作者可製備 Vectorbeam 碼序列具有實質相同運算碼與實質相同格式，並交給處理器以將該碼最佳化。處理器將 Vectorbeam 碼最佳化以利用其特定硬體能力及指令集架構之優勢。以此方法，用以實作碼 1 之碼序列可被移植到較新世代之處理器及指令集架構，而無須費力將該碼再撰寫、再記錄、再賦能、再測試、再訓練、再除錯、與再發表。當新處理器變可用時，開發商亦可能花較少時間來學習新運算碼與新暫存器檔案及與新指令集架構相關之其他細節。

圖 4 根據實施例說明藉由將表 1 之 Vectorbeam 碼最佳化的處理器所執行之例示性暫存器更名。更明確地，在此實施例中的 Vectorbeam 碼將被 128 位元處理器（例如，SSE）運行，該處理器具有預設策略以將 Vectorbeam 內文擴展 4。因此，第一次通過處理器必須執行用於具有值{0, 1, 2, 3}之 rax 的碼。當處理器到達 Vectorbeam 指令時，movss (%rsp,%rax,4), %xmm0，其必須執行四次載入：

*xmm0[0] = (rsp + 0*4);*

*xmm0[1] = (rsp + 1*4);*

*xmm0[2] = (rsp + 2*4);*

*xmm0[3] = (rsp + 3*4);*

雖然表 1 之 Vectorbeam 碼意指架構暫存器 xmm0 到

xmm1，執行圖 4 之 Vectorbeam 碼 402 的處理器將架構暫存器更名成實體暫存器。（如同對該領域具有通常知識者而言為已知的，執行該碼之處理器可能具有大量的實體暫存器以作用為各種程式執行緒中參照的架構暫存器，該等執行緒可同時地運行或失序地運行。）如所示，處理器存取邏輯到實體暫存器表 404 以判定如何將架構暫存器對映到實體暫存器。根據表 406，處理器指定 xmm0 及 xmm1 到實體暫存器檔案位置 4-7 及 12-15。

圖 5 根據一實施例說明用以並行處理圖 4 之經分配暫存器所使用之多計算資源之例示性分配。本文之 128 位元 SSE 處理器將對其可用硬體資源之知識應用以最佳化該碼及暫存器更名。如所示，根據本揭示之處理器將架構 xmm 暫存器更名成實體暫存器檔案 502。在此實施例中的所選暫存器分配允許一次四個 ALU 並行地執行加法，如圖 5 之 504、506、508、及 510 所指示。

表 2 說明根據一實施例經最佳化的例示性序列之 Vectorbeam 次最佳化指令。更明確地，表 2 之並排的碼比較說明四種版本之指令：純量碼、SSE 碼、AVX3 碼、及 Vectorbeam 碼，用以實作碼 2 之有條件式循環，參見以下：

碼2. *for (int i = 0; i < PTS; i++) { if (cond[i]) {outp[i] = inp1[i] + inp2[i];}}*

純量碼	SSE 碼

..LOOP
cmpl $0, 768(%rsp,%rax,4)
je ..FALSE
movss (%rsp,%rax,4), %xmm0
addss 256(%rsp,%rax,4), %xmm0
movss %xmm0, 512(%rsp,%rax,4)
..FALSE
incq %rax
cmpq $64, %rax
jl ..LOOP
	..LOOP
	movups (%rsp,%rax,4), %xmm3
	movdqu 768(%rsp,%rax,4), %xmm0
	pcmpe4 %xmm1, %xmm0
	addps 256(%rsp,%rax,4), %xmm3
	movups 512(%rsp,%rax,4), %xmm4
	pxor %xmm2, %xmm0
	blendvps %xmm0, %xmm3, %xmm4
	movups %xmm4, 512(%rsp,%rax,4)
	addq $4, %rax
	cmpq $64, %rax
	jb ..LOOP

AVX3 碼	Vectorbeam 碼

..LOOP
vmovups (%rsp,%rax), %zmm1
addb $16, %dl
vpcmpd $4, 512(%rsp,%rax), %zmm0, %k1
vaddps 256(%rsp,%rax), %zmm1, %zmm2
vmovups %zmm2, 1624(%rsp,%rax){%k1}
addq $64, %rax
cmpb $64, %dl
jb ..LOOP
	foreach rax, 0, 64, 1
	cmpl $0, 768(%rsp,%rax,4)
	je ..FALSE
	movss (%rsp,%rax,4), %xmm0
	addss 256(%rsp,%rax,4), %xmm0
	movss %xmm0, 512(%rsp,%rax,4)
	..FALSE
	Next

表2.

如所示，表 1 與表 2 中所示之純量碼、相似純量碼為次最佳化的，這是因為其需要 64 個循環。若純量碼或要在當代向量處理器上運行，則其將使用較少之可用並行硬體資源。

如同 AVX 碼（表 1），將純量碼遷移成 SSE 碼之編程者將需學習新運算碼、新暫存器集、及不提供跳越旁路之 SSE 碼。一般而言，對發明純量指令而言，記錄、賦能編譯器以將之發出、訓練使用者使用它們、及除錯並收集該碼之追蹤，上述所做的工作在每次指令集架構（例如，MMX、SSE、SSE2、SSE3、SSE4、AVX、AVX2、AVX 3.1、及 AVX 3.2）之產生可能需要被部分或全然地重複。

然而，SSE 碼具有些許的缺點。舉其一例而言，由於 SSE 碼不具有跳越旁路，故其一視同仁地對所有巷道執行數學計算，並接著使用 BLEND 指令以適當地將未觸及非條件式值與該等撰寫於條件式方塊中者合併（merge）。效能受損（BLEND 係較一般儲存緩慢大約 1.5 倍）－更多的暫存器被需要（5 vs. 2）。如同 AVX 碼（表 1）一般，SSE 碼產生需要較精細之編譯器。

AVX3 碼（相似於 AVX 碼（表 1））在許多態樣方面與純量碼不同。AVX3 暫存器（ymm0...n）具有不同名稱與不同大小。AVX3 運算碼具有不同的名稱。AVX3 使用「k」個暫存器，其經設定為比較之結果並接著結合儲存/載入而被使用。與 SSE 碼相同，AVX3 碼不具有分支；在所有條件下，處理器執行全部的運算碼。一般而言，接

著，對純量指令而言，發明之：記錄之、賦能編譯器以將之發出、訓練使用者使用之、及除錯並收集該碼之追蹤，上述所作的工作在針對每次指令集架構（例如，MMX、SSE、SSE2、SSE3、SSE4、AVX、AVX2、AVX 3.1、及AVX 3.2）之產生時，可能需要被部分或全部重複。

然而根據本揭示，與（表 1）實質相同之純量指令可以實質相同格式被使用，於 Vectorbeam 碼中。更明確地，Vectorbeam 碼可以序列開頭之指令（在此為「foreach rax, 0, 64, 1」）與序列終端之指令（在此為「next」）開始。如所編碼的，「foreach」運算碼對實作流程 100（圖 1）之處理器指示該之後跟隨之次最佳化碼係迭代碼。如所編碼的，運算元「rax」、「0」、「64」、及「 1」對實作流程 100（圖 1）之處理器提供元資料建議，以使用暫存器 rax 作為循環索引；使 rax 變化從 0 到 63；及針對每迭代具有 1 的跨距，來循環該次最佳化 Vectorbeam 碼。換言之，實作流程 100（圖 1）之處理器將被指示以執行次最佳化碼 64 次，每次以 1 之間隔改變 rax。以此方法，用以實作碼 1 之碼序列可被移植到新世代之指令集架構，而無須費力將該碼再撰寫、再測試、與再發表。開發商亦可能花較少時間來學習新運算碼與新暫存器檔案及與新指令集架構相關之其他細節。

此外，根據本揭示，CPU 建構者（其具有與特定 CPU 相關之特定資源、能力、及指令集架構之知識）可選擇如何實作所述在該序列之次最佳化 Vectorbeam 指令中的功

能。此外，運算碼 302（圖 3）及元資料 304、306、及 308（圖 3）可提供有關選擇哪個指令要替代該序列之次最佳化 Vectorbeam 指令的暗示到處理器。舉例而言，序列開頭之指令可允許處理器知道一些記憶體負載或儲存運用相同的偏移。又或者，序列開頭之指令可提供有關要通往相同快取線之一些記憶體負載或儲存的暗示。又或者，序列開頭之指令可提供一些數學運算使用與可能在諸如藉由增強明亮度來統一調整像素之多媒體或圖形碼常式之例子中相同運算元的暗示。

圖 6 依據一實施例說明表列對於純量指令之向量指令替代的查找表之一部分。更明確地，表 600 表列純量指令及其對應向量等效物用於執行數學運算：加法、減法、乘法、除法、平方根、最大值、最小值、倒數、及平方根之倒數。在一實施例中，參照並排之表 600，儲存在指令緩衝器 212（圖 2）中的該序列之次最佳化 Vectorbeam 指令包括純量加法、ADDSS、及替代評估邏輯 216（圖 2），其以向量加法、ADDPS 來替代次最佳化 Vectorbeam 指令之一或多者。

圖 7 根據一實施例說明向量處理器暫存器檔案。更明確地，暫存器檔案 700 包括向量暫存器 702 及通用暫存器 704。如所示，向量暫存器 702 包括 32 zmm 暫存器 706（各為 512 位元寬）、16 ymm 暫存器 708（各為 256 位元寬），及 16 xmm 暫存器 710（各為 128 位元寬）。作為在指令緩衝器 212（圖 2）中用於次最佳化 Vectorbeam

指令之評估替代之部分，在一實施例中的替代評估邏輯 216（圖 2）指定 8 個 32 位元純量操作之結果到 256 位元之 ymm 暫存器。

在替代實施例中，Vectorbeam 序列之次最佳化指令被來自較新指令集架構之指令替換。在一實施例中，運用 128 位元 XMM 暫存器 710 之一序列之次最佳化 SSE 指令被一序列之 AVX 替換，運行使用 256 位元 YMM 暫存器 708 之指令，並達到更佳之效能而無須次最佳化碼被其撰寫者重新撰寫。在另一實施例中，運用 YMM 暫存器 708 之一序列之次最佳化 AVX 指令被一序列之 AVX-512 指令替換，其使用 ZMM 暫存器 706，並達到更佳之效能而無須次最佳化碼被其撰寫者重新撰寫。

在替代實施例中，運算碼 302（圖 3）可識別與儲存在指令緩衝器 212（圖 2）中的次最佳化指令相關之特定指令集架構。替代評估邏輯 216（圖 2）可接著以與處理器之指令集架構相關之最佳化指令來替代次最佳化指令。

在一實施例中，次最佳化指令與替代指令皆為向量指令，但是替代指令係來自較新的世代並使用較寬之暫存器。

【符號說明】

100：方塊流程圖

200：方塊圖

202：指令儲存器

204：提取電路

206：解碼電路

208：解碼邏輯

210：最佳化模式偵測器

212：指令緩衝器

214：查找表

216：替代評估邏輯

218：指令選擇器

220：執行電路

222：暫存器

224：記憶體

226：引退/提交電路

300：指令

302：運算碼

304：第一運算元識別符

306：第二運算元識別符

308：第三運算元識別符

402：Vectorbeam 碼

404：表

406：表

502：實體暫存器檔案

504：ALU(算術邏輯單元)#1

506：ALU(算術邏輯單元)#2

508：ALU(算術邏輯單元)#3

510：ALU(算術邏輯單元)#4

600：表

700：暫存器檔案

702：向量暫存器

704：通用暫存器

706：zmm 暫存器

708：ymm 暫存器

710：xmm 暫存器

第 106102675 號　　民國 109 年 12 月 1 日修正

申請專利範圍

1. 一種在運行時最佳化指令的處理器，包含：

提取及解碼電路，用以提取及解碼該等指令；

指令緩衝器，包含至少四個條目用以緩衝經解碼的指令；

替代評估電路，用以在運行期間操作，用以：

判定操作模式是否為最佳化模式；

在緩衝指令之中識別一序列之次最佳化指令，其包含三或更多指令且由序列開頭之指令所定界；

存取處理器情境資料，包括暫存器檔案的內容；

存取替代指令的查找表；及

根據該操作模式、該序列之次最佳化指令、該處理器情境資料及該查找表的內容，為該等緩衝指令的每一者在該緩衝指令與一或多個替代指令之間選擇；及

執行電路，用以執行該等選擇指令的每一者。

2. 如申請專利範圍第 1 項之處理器，其中，該序列之次最佳化指令之所以次最佳化，是因為其使用比該查找表中的指令較舊世代的指令集架構（ISA）。

3. 如申請專利範圍第 1 項之處理器，其中，該序列之次最佳化指令之所以次最佳化，是因為其包括為使用比該處理器上可用的向量暫存器更小的向量暫存器而被撰寫的向量指令。

4. 如申請專利範圍第 1 項之處理器，其中，該序列

之次最佳化指令包括純量多媒體操作，且該等替代指令包括向量指令。

5. 如申請專利範圍第 1 項之處理器，其中，該序列之次最佳化指令之所以次最佳化，是因為該處理器情境資料指示該等次最佳化指令正在使用多於一個記憶體存取來存取相同快取線。

6. 如申請專利範圍第 1 項之處理器，其中，該緩衝指令包含純量指令，且一序列之一或多個最佳化指令包含向量指令。

7. 如申請專利範圍第 1 項之處理器，其中，該緩衝指令包含純量指令，且一序列之一或多個最佳化指令包含 SIMD（單指令多資料）指令。

8. 如申請專利範圍第 1 項之處理器，還包含最佳化模式偵測器電路，用以在運行時監控該指令緩衝器、在偵測到該序列開頭之指令時將該最佳化模式設定為 TRUE，在偵測到序列終端之指令時將該最佳化模式設定為 FALSE。

9. 如申請專利範圍第 8 項之處理器，該序列開頭之指令包括用以提供是否以及如何執行替代之提示的元資料。

10. 一種由處理器在運行時最佳化指令的方法，包含：

由替代評估電路判定處理器正在最佳化模式下運行；

使用提取電路及解碼電路提取及解碼該等指令；

在包含至少四個條目之指令緩衝器中緩衝經解碼的指令；

由替代評估電路識別包含三或更多由序列開頭之指令所定界的指令的一序列之次最佳化指令；

由該替代評估電路存取處理器情境資料，包括暫存器檔案的內容；

由該替代評估電路存取替代指令的查找表；

根據該序列之次最佳化指令、該處理器情境資料及該查找表的內容，為每個緩衝指令在該緩衝指令與一或多個替代指令之間選擇；及

由執行電路執行該等選擇指令。

11. 如申請專利範圍第 10 項之方法，其中，該序列之次最佳化指令之所以次最佳化，是因為其使用比該查找表中的指令較舊世代的指令集架構（ISA）。

12. 如申請專利範圍第 10 項之方法，其中，該序列之次最佳化指令之所以次最佳化，是因為其包括用以使用比該處理器上可用的向量暫存器更小的向量暫存器而被撰寫的向量指令。

13. 如申請專利範圍第 10 項之方法，其中，該序列之次最佳化指令包括純量多媒體操作，且該等替代指令包括向量指令。

14. 如申請專利範圍第 10 項之方法，其中，該序列之次最佳化指令之所以次最佳化，是因為根據該處理器情境資料，該等次最佳化指令正在使用多於一個記憶體存取

來存取相同快取線。

15. 如申請專利範圍第 10 項之方法，其中，該緩衝指令包含純量指令，且一序列之一或多個最佳化指令包含 SIMD（單指令多資料）指令。

16. 如申請專利範圍第 10 項之方法，其中，該序列開頭之指令包括用以提供是否以及如何執行替代之提示的元資料。

17. 一種非暫態機器可讀媒體，其包含指令，該等指令當由處理器執行時，使處理器在運行時藉由以下步驟最佳化指令：

判定該處理器以最佳化模式運行；

使用提取電路及解碼電路提取及解碼該等指令；

在包含至少四個條目之指令緩衝器中緩衝經解碼的指令；

識別包含三或更多由序列開頭之指令所定界的該等緩衝指令的一序列之次最佳化指令；

存取處理器情境資料，包括暫存器檔案的內容；

存取替代指令的查找表；

根據該序列之次最佳化指令、該處理器情境資料及該查找表的內容，為每個緩衝指令在該緩衝指令與一或多個替代指令之間選擇；及

執行電路，用以執行該等選擇指令。

18. 如申請專利範圍第 17 項之非暫態機器可讀媒體，其中，該序列之次最佳化指令包括純量指令，且該等

替代指令包括向量指令。

19. 如申請專利範圍第 17 項之非暫態機器可讀媒體，其中，，該一 或多個緩衝指令包含純量指令，且一序列之一或多個次最佳化指令包含 SIMD（單指令多資料）指令。

20. 如申請專利範圍第 17 項之非暫態機器可讀媒體，其中，該序列開頭之指令包括用以提供是否以及如何執行替代之提示的元資料。

圖式

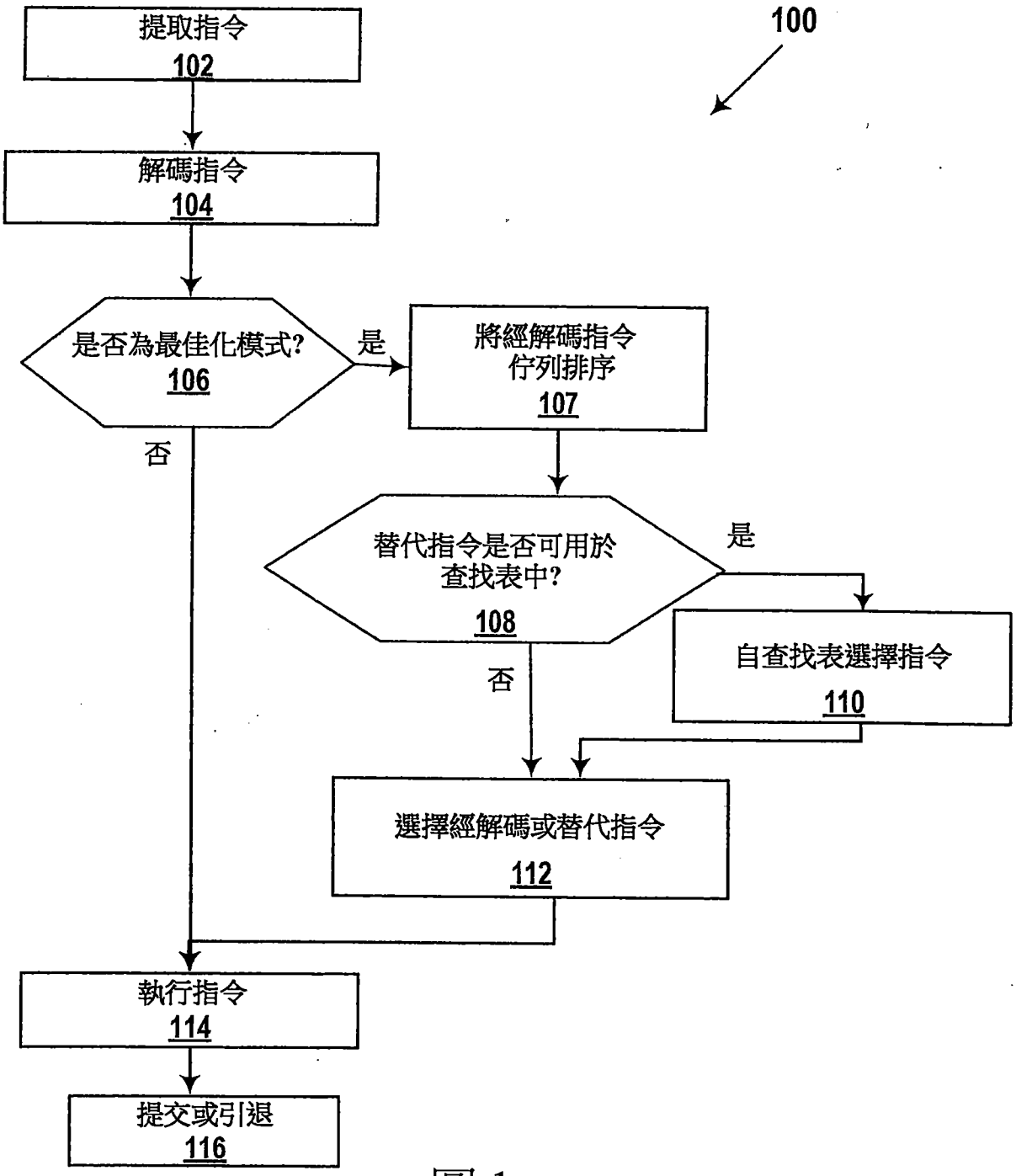

100
提取指令
102
解碼指令
104
是否為最佳化模式?
106
是
將經解碼指令
佇列排序
107
否
替代指令是否可用於
查找表中?
108
是
自查找表選擇指令
110
否
選擇經解碼或替代指令
112
執行指令
114
提交或引退
116

圖 1

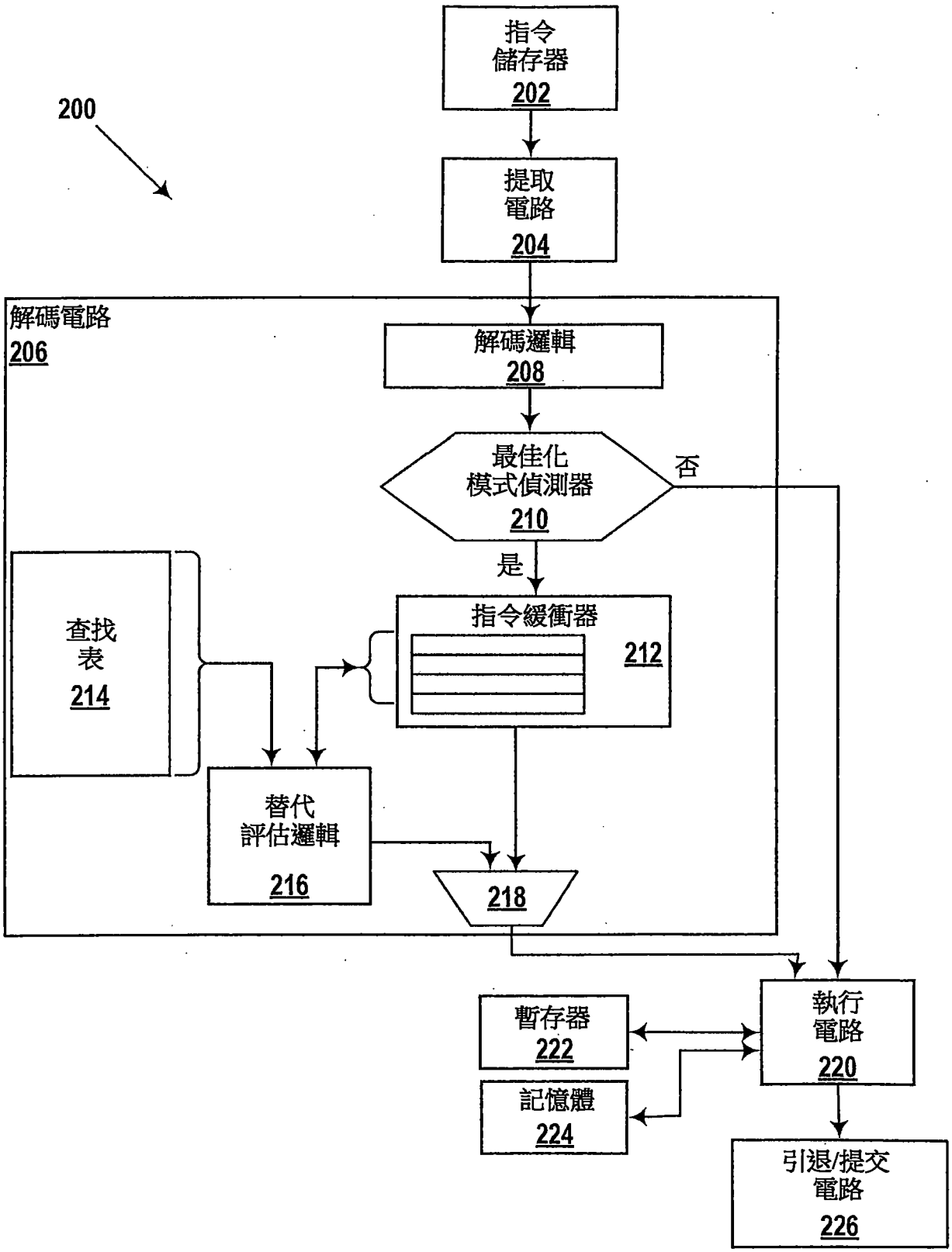

200
指令
儲存器
202
提取
電路
204
解碼電路
206
解碼邏輯
208
最佳化
模式偵測器
210
否
是
查找
表
214
指令緩衝器
212
替代
評估邏輯
216
218
暫存器
222
記憶體
224
執行
電路
220
引退/提交
電路
226

圖 2

運算碼 302	第一運算元 識別符 [可選] 304	第二運算元 識別符 [可選] 306	第三運算元 識別符 [可選] 308

圖 3

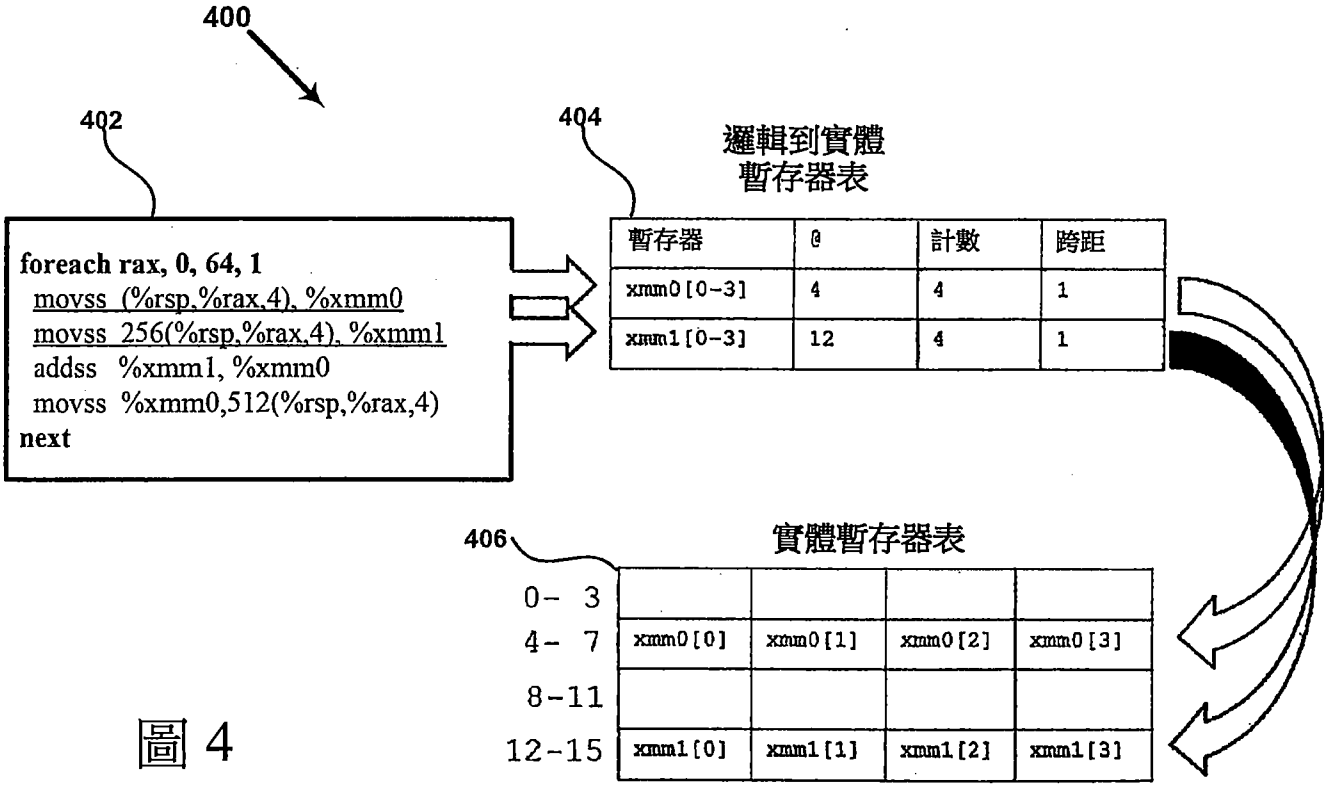

400
402
404
邏輯到實體
暫存器表
foreach rax, 0, 64, 1
movss (%rsp,%rax,4), %xmm0
movss 256(%rsp,%rax,4), %xmm1
addss %xmm1, %xmm0
movss %xmm0,512(%rsp,%rax,4)
next
暫存器
@
計數
跨距
xmm0[0-3]
4
4
1
xmm1[0-3]
12
4
1
406
實體暫存器表
0- 3
4- 7
8-11
12-15
xmm0[0]
xmm0[1]
xmm0[2]
xmm0[3]
xmm1[0]
xmm1[1]
xmm1[2]
xmm1[3]

圖 4

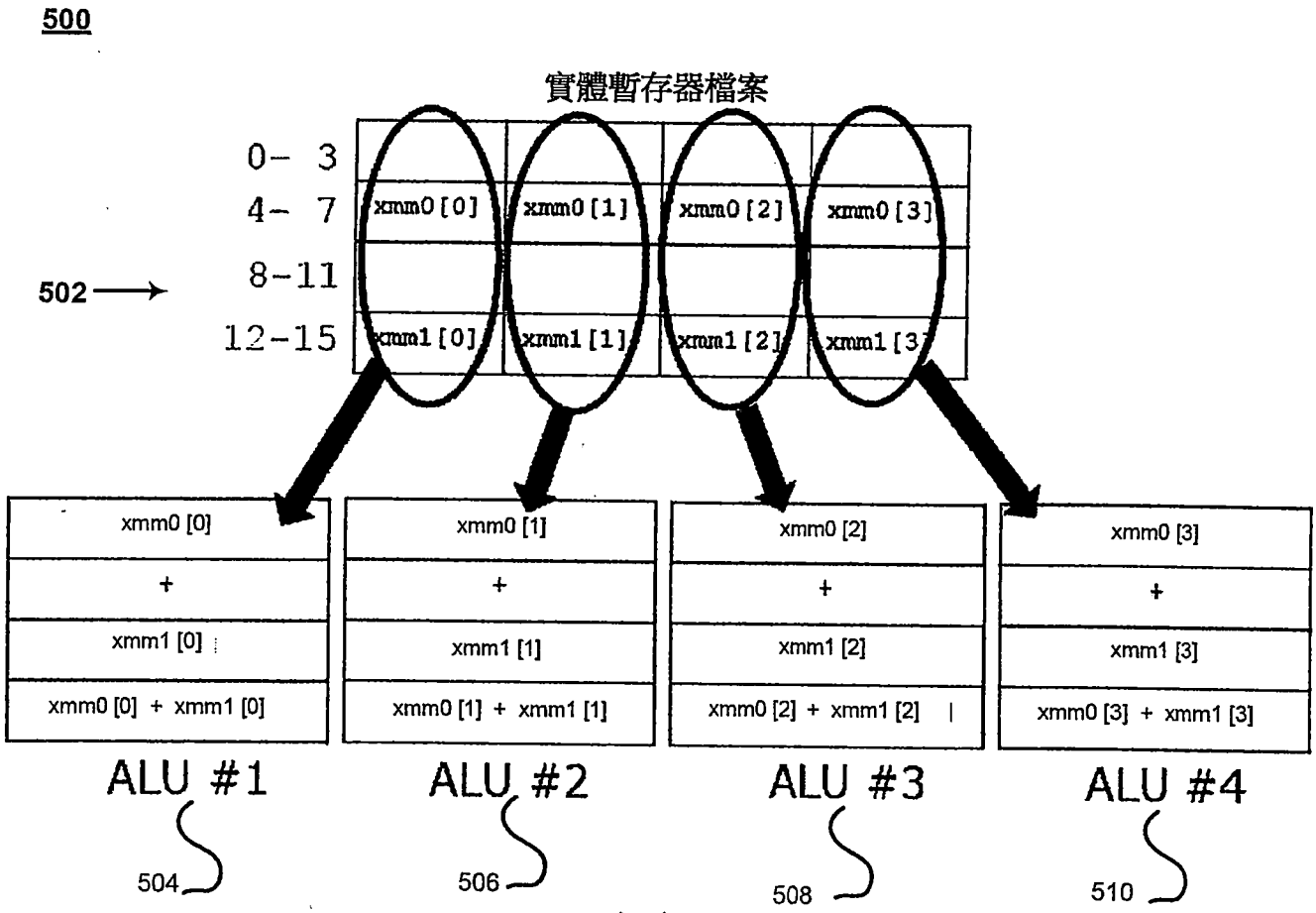

500
實體暫存器檔案
0- 3
4- 7
8-11
12-15
502
xmm0 [0]
xmm0 [1]
xmm0 [2]
xmm0 [3]
xmm1 [0]
xmm1 [1]
xmm1 [2]
xmm1 [3]
xmm0 [0]
+
xmm1 [0]
xmm0 [0] + xmm1 [0]
xmm0 [1]
+
xmm1 [1]
xmm0 [1] + xmm1 [1]
xmm0 [2]
+
xmm1 [2]
xmm0 [2] + xmm1 [2]
xmm0 [3]
+
xmm1 [3]
xmm0 [3] + xmm1 [3]
ALU #1
ALU #2
ALU #3
ALU #4
504
506
508
510

圖 5

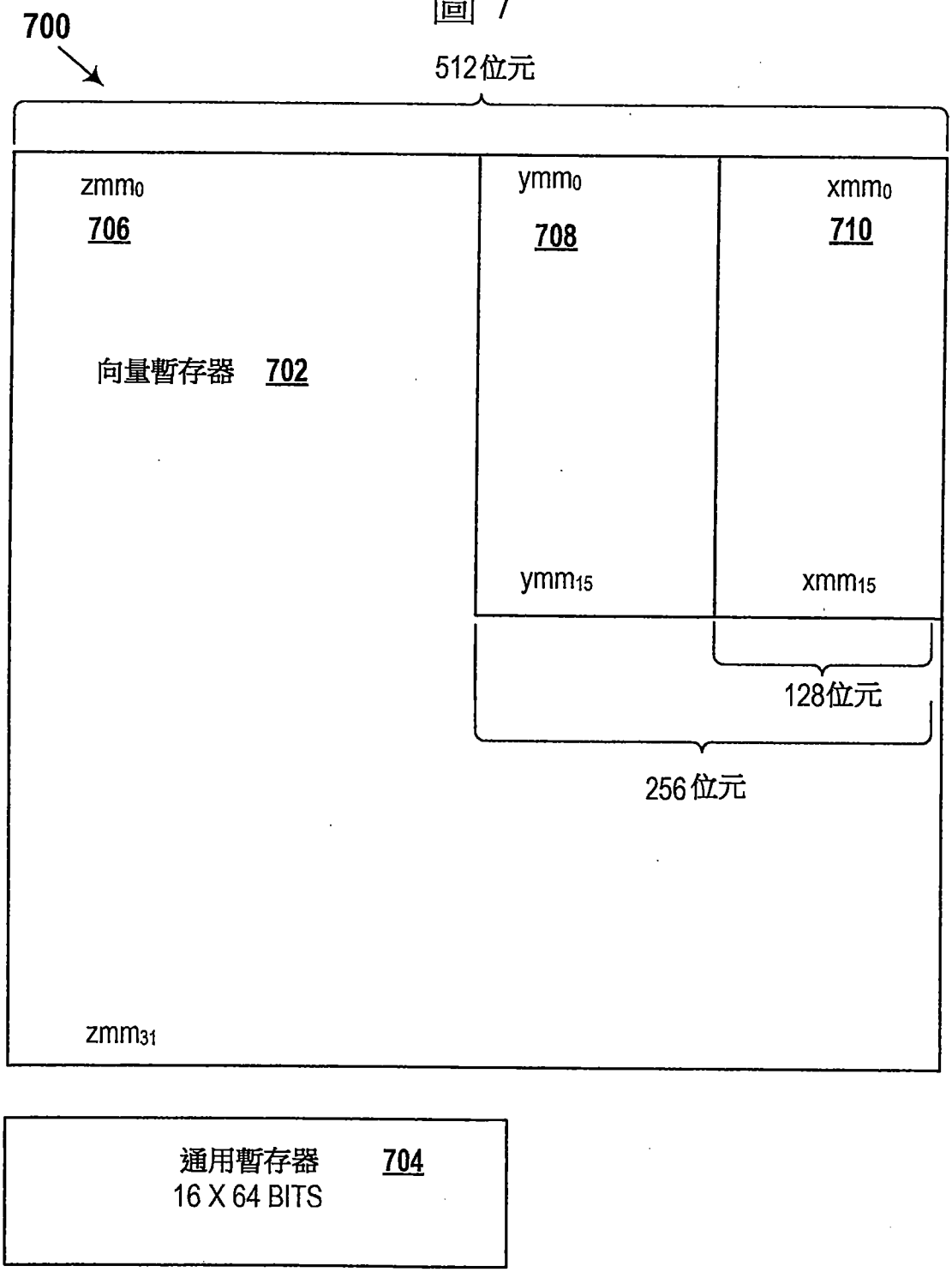
600

純量比照向量數學指令

純量¤	向量¤	說明¤	¤
ADDSS¤	ADDPS¤	加法運算元¤	¤
SUBSS¤	SUBPS¤	減法運算元¤	¤
MULSS¤	MULPS¤	乘法運算元¤	¤
DIVSS¤	DIVPS¤	除法運算元¤	¤
SQRTSS¤	SQRTPS¤	平方根運算元¤	¤
MAXSS¤	MAXPS¤	最大值算元¤	¤
MINSS¤	MINPS¤	最小值運算元¤	¤
RCPSS¤	RCPPS¤	倒數運算元¤	¤
RSQRTSS¤	RSQRTPS¤	平方根之倒數運算元¤	¤

圖 6

圖 7


700
512位元
zmm0
706
ymm0
708
xmm0
710
向量暫存器 702
ymm15
xmm15
128位元
256位元
zmm31
通用暫存器 704
16 X 64 BITS