I637320

(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 I637320 B

(45)公告日： 中華民國 107 (2018) 年 10 月 01 日

(21)申請案號：106106344 (22)申請日： 中華民國 102 (2013) 年 06 月 14 日

(51)Int. Cl.： *G06F9/38 (2006.01)* *G06F9/46 (2006.01)*

(30)優先權：2012/06/15 美國 61/660,526

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

(72)發明人：艾伯戴爾拉 摩翰麥德 ABDALLAH, MOHAMMAD (US)

(74)代理人：林志剛

(56)參考文獻：

TW	201042544A	CN	1728087A
US	2002/0199063A1	US	2010/0281220A1

審查人員：李榮祥

申請專利範圍項數：20 項 圖式數：25 共 81 頁

(54)名稱

根據儲存資歷實施來自不同執行緒轉送的執行緒不可知之載入儲存緩衝區

A LOAD STORE BUFFER AGNOSTIC TO THREADS IMPLEMENTING FORWARDING FROM DIFFERENT THREADS BASED ON STORE SENIORITY

(57)摘要

本發明揭示一種在一處理器中使用共享記憶體資源用於記憶體一致性模型內失序載入之一執行緒未知統一儲存佇列與一統一載入佇列方法。該方法包含：實施一記憶體資源，其可由複數個非同步核心存取，其中該等複數個核心共享一統一儲存佇列及一統一載入佇列；及實施一存取遮罩，其作用在於追蹤一快取線的哪個字元透過一載入來存取，其中該快取線包含該記憶體資源，其中存取該快取線的一字元時，該載入設定該存取遮罩內的一遮罩位元，且其中該遮罩位元阻擋來自複數個核心的其他載入之存取。該方法更包含：在執行從該等複數個核心至該快取線的後續儲存時檢查該存取遮罩，其中可將來自不同執行緒的儲存轉送至不同執行緒的載入，同時依序維持記憶體一致性語意；及當該部分快取線的一後續儲存發現來自該存取遮罩內一載入的一先前遮罩時，導致一錯失預測，其中該後續儲存使用一追蹤暫存器以及一執行緒 ID 暫存器，發信號通知對應該載入的一載入佇列輸入。

In a processor, a thread agnostic unified store queue and a unified load queue method for out of order loads in a memory consistency model using shared memory resources. The method includes implementing a memory resource that can be accessed by a plurality of asynchronous cores, wherein the plurality of cores share a unified store queue and a unified load queue; and implementing an access mask that functions by tracking which words of a cache line are accessed via a load, wherein the cache line includes the memory resource, wherein the load sets a mask bit within the access mask when accessing a word of the cache line, and wherein the mask bit blocks accesses from other loads from a plurality of cores. The method further includes checking the access mask upon execution of subsequent stores from the plurality of cores to the cache line, wherein stores from different threads can forward to loads of different threads while still maintaining in order memory consistency semantics; and causing a miss prediction when a subsequent store

to the portion of the cache line sees a prior mark from a load in the access mask, wherein the subsequent store will signal a load queue entry corresponding to that load by using a tracker register and a thread ID register.

指定代表圖：

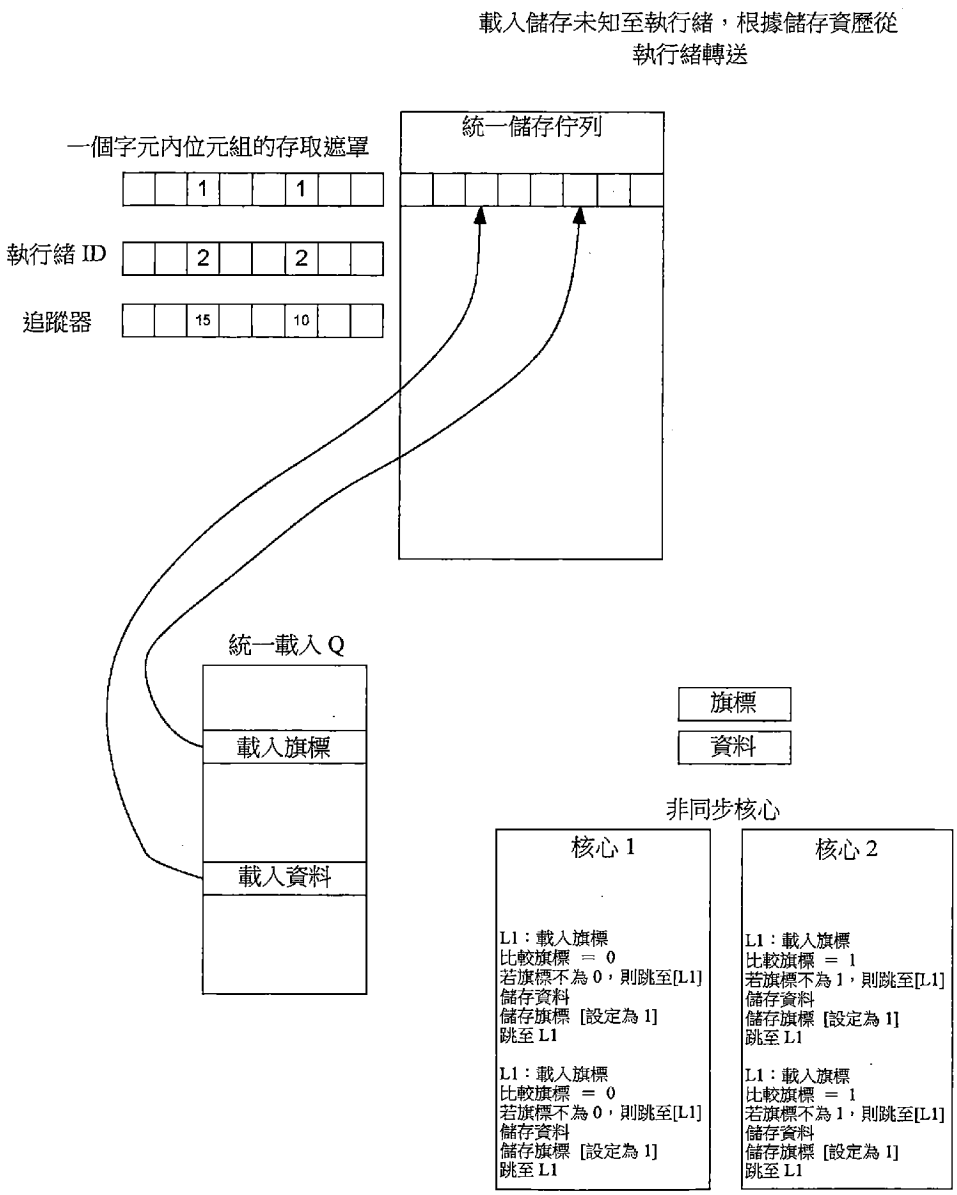

圖 20

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)

根據儲存資歷實施來自不同執行緒轉送的執行緒不可知之載入儲存緩衝區

A load store buffer agnostic to threads implementing forwarding from different threads based on store seniority

【技術領域】

[0001] 本發明一般係關於數位電腦系統，尤其係關於選擇包括一指令序列的指令之系統及方法。

【先前技術】

[0002] 處理器需要處理多個相依或完全獨立的任務。這種處理器的內部狀態通常由暫存器構成，這些暫存器可在程式執行的每一特定實例(instacne)上保有不同值。在程式執行的每一實例上，內部狀態影像(interal state image)稱為處理器的架構狀態。

[0003] 程式碼執行切換成運行另一功能時(例如另一執行緒、處理或程式)，則必須儲存機器/處理器的狀態，如此該新功能可運用內部暫存器來建立自己的新狀態。一旦已經終止該新功能，則可忽略其狀態，且將復原先前關聯的狀態然後恢復執行。這種切換處理稱為關聯切換(context switch)，且通常包含數十個或數百個循環，尤其

是具有運用大量暫存器(例如 64、128、256)及/或失序(out of order)執行的現代架構。

[0004] 在執行緒認知(thread-aware)硬體架構內，正常硬體可支援有限數量硬體支援執行緒的多個關聯狀態。在此狀況下，該硬體將所有架構狀態元件複製給每一支援的執行緒。這免除執行新執行緒時關聯切換的需求。不過，這仍舊具有多項缺點，換言之就是複製所有架構狀態元件(即是暫存器)每硬體內支援的每一額外執行緒之區域、功率與複雜度。此外，若軟體執行緒的數量超出明確支援硬體執行緒的數量，則仍舊必須執行關聯切換。

[0005] 這對於需要大量執行緒的細微粒度基礎上需要並行處理來說已經司空見慣。該等硬體執行緒認知架構含有複製關聯狀態硬體儲存，對於非執行緒軟體程式碼並無幫助，只會減少已經運行執行緒的軟體之關聯切換數量。不過，這些執行緒通常建構用於粗粒度並行處理，導致軟體在初始化以及同步化期間有非常重的負擔，使得例如函數呼叫以及迴圈並行執行這類細粒度並行處理不具備有效的執行緒初始化/自動產生。這種描述的負擔伴隨難以將最新編譯器或使用者並行處理技術，用於非明確/簡易並行處理/執行緒軟體程式碼，讓這些程式碼自動並行處理。

【發明內容】

[0006] 在一具體實施例中，本發明實施為一種使用

共享記憶體資源用於記憶體一致性模型內失序載入之一執行緒未知統一儲存佇列與一統一載入佇列方法。該方法包含：實施一記憶體資源，其可由複數個非同步核心存取，其中該等複數個核心共享一統一儲存佇列及一統一載入佇列；及實施一存取遮罩，其作用在於追蹤一快取線的哪個字元透過一載入來存取，其中該快取線包含該記憶體資源，其中存取該快取線的一字元時，該載入設定該存取遮罩內的一遮罩位元，且其中該遮罩位元阻擋來自複數個核心的其他載入之存取。該方法更包含：在執行從該等複數個核心至該快取線的後續儲存時檢查該存取遮罩，其中可將來自不同執行緒的儲存轉送至不同執行緒的載入，同時依序維持記憶體一致性語意；及當該部分快取線的一後續儲存發現來自該存取遮罩內一載入的一先前遮罩時，導致一錯失預測，其中該後續儲存使用一追蹤暫存器以及一執行緒 ID 暫存器，發信號通知對應該載入的一載入佇列輸入。

[0007] 上述為總結，因此必須簡單扼要且省略細節；因此精通技術人士將了解，該總結僅為例示，並不成為任何限制。從下列非限制的詳細說明當中，將了解如同申請專利範圍所單獨定義的本發明之其他態樣、發明特色以及優點。

【圖式簡單說明】

[0008] 本發明藉由範例進行說明且不受其限制，在

附圖中的數據以及其中相同的參考編號指示相同的元件。

[0009] 圖 1 顯示根據本發明之一具體實施例的一載入佇列與一儲存佇列。

[0010] 圖 2 顯示根據本發明之一具體實施例載入與儲存指令分割的第一圖式。

[0011] 圖 3 顯示根據本發明之一具體實施例載入與儲存指令分割的第二圖式。

[0012] 圖 4 顯示一處理步驟的流程圖，其中根據本發明之一具體實施例，繪製用於從導因於負載儲存重新排序與最佳化的推測性轉送錯失預測/錯誤當中實施復原之規則。

[0013] 圖 5 顯示根據本發明之一具體實施例，例示其中使用一處理器的載入佇列與儲存佇列資源來實施處理 300 的該等規則之方式圖式。

[0014] 圖 6 顯示根據本發明之一具體實施例，例示其中使用一處理器的載入佇列與儲存佇列資源來實施處理 300 的該等規則之方式另一圖式。

[0015] 圖 7 顯示根據本發明之一具體實施例，例示其中使用一處理器的載入佇列與儲存佇列資源來實施處理 300 的該等規則之方式另一圖式。

[0016] 圖 8 顯示根據本發明之一具體實施例，其中在一載入之後派遣一儲存的該派遣功能概要之處理流程圖。

[0017] 圖 9 顯示根據本發明之一具體實施例，其中

在一儲存之後派遣一載入的該派遣功能概要之處理流程圖。

[0018] 圖 10 顯示根據本發明之一具體實施例的統一載入佇列之圖式。

[0019] 圖 11 顯示根據本發明之一具體實施例顯示該滑動載入派遣窗的統一載入佇列。

[0020] 圖 12 顯示根據本發明之一具體實施例的一分配載入佇列。

[0021] 圖 13 顯示根據本發明之一具體實施例具有一依序連續窗的一分配載入佇列。

[0022] 圖 14 顯示根據本發明之一具體實施例用於一多核心處理器的一分段記憶體子系統之圖式。

[0023] 圖 15 顯示本發明具體實施例如何操控載入與儲存的圖式。

[0024] 圖 16 顯示根據本發明之一具體實施例的一儲存篩選演算法之圖式。

[0025] 圖 17 顯示根據本發明之一具體實施例，具有構成從記憶體內依序讀取載入的記憶體一致性模型內失序載入之旗標語實施。

[0026] 圖 18 顯示根據本發明之一具體實施例，利用使用一鎖定式模型與一交易式模型，失序載入至構成從記憶體內依序讀取載入的記憶體一致性模型內。

[0027] 圖 19 顯示根據本發明之一具體實施例的多核心分段記憶體子系統之複數個核心。

[0028] 圖 20 顯示根據本發明之一具體實施例非同步核心存取一統一儲存佇列之圖式，其中執行緒根據儲存資歷賦予儲存。

[0029] 圖 21 顯示根據本發明之一具體實施例，說明其中儲存的資歷超越其他執行緒內對應儲存的功能性之圖式。

[0030] 圖 22 顯示根據本發明之一具體實施例的一免消歧義失序載入佇列除役實施。

[0031] 圖 23 顯示根據本發明之一具體實施例的一免消歧義失序載入佇列重新排序實施之重新排序實施。

[0032] 圖 24 顯示根據本發明之一具體實施例的一指令序列(例如軌跡)重新排序推測性執行實施。

[0033] 圖 25 顯示根據本發明之一具體實施例的示範微處理器管線之圖式。

【實施方式】

[0034] 雖然本發明已經搭配具體實施例來說明，但本發明並不用於限制到此處所公佈的特殊形式。相反地，吾人想要如申請專利範圍內所定義將改變、修改以及同等配置合理包含在本發明的範疇內。

[0035] 在下列詳細說明當中，許多特定細節，例如特定方法序列、結構、元件以及連接都已經公布。不過吾人要了解，這些與其他特定細節並不需要用於實現本發明的具體實施例。在其它環境中，已知的結構、元件或連接

都已經省略或未詳細說明，以避免模糊本說明。

[0036] 規格書內參考本發明的「一具體實施例」或「具體實施例」用於表示，與該具體實施例有關連所說明的特定功能、結構或特性包含在本發明的至少一具體實施例內。出現在整個規格書內許多地方的「在一具體實施例內」一詞，並不一定全都參照到同一具體實施例，也不是與其他具體實施例互斥的個別或替代具體實施例。再者，說明可由某些具體實施例展示且其他沒有的許多特色。同樣，說明可為某些具體實施例所需但是其他具體實施例不需的許多需求。

[0037] 某些詳細說明部分都以可在電腦記憶體上執行的資料位元上操作之程序、步驟、邏輯區塊、處理以及其他符號表示之方式來呈現。這些說明與代表為精通資料處理技術人士用來將其工作內容灌輸給其他精通此技術人士的最有效方式。此處的程序、電腦可執行步驟、邏輯區塊、處理等等一般係認為是導致所要結果的自洽步驟或指令序列。這些步驟為所需的物理量之物理操縱。通常，雖然非必要，不過這些量採用電腦可讀取儲存媒體且可以在電腦系統內儲存、傳輸、結合、比較以及操縱的電或磁性信號形式。為了時間上方便起見，原則上因為常用，所以這些信號代表位元、數值、元件、符號、字元、詞彙、數字等等。

[0038] 不過，吾人應該瞭解，所有這些與類似詞彙都與適當的物理量相關連，且僅為適用這些量的便利符

號。除非特別說明，否則從下列討論中可瞭解，整個說明書的討論運用像是「處理」、「存取」、「寫入」、「儲存」或「複製」等詞表示電腦系統或類似電子計算裝置的動作以及處理，其操縱以及轉換代表電腦系統暫存器、記憶體和其他電腦可讀取媒體內物理(電子)量的資料成為類似代表電腦系統記憶體、暫存器或其他這種資訊儲存、傳輸或顯示裝置內物理量的其他資料。

[0039] 本發明的具體實施例實施一種失序指令排程處理，其中一旦處理器資源可用來執行一輸入指令序列內的指令時，則允許失序地發出該指令。本發明的具體實施例可確保外部代理主機(agent)發現指令依序執行(例如記憶體一致性規則/模型)。對於該等外部代理主機來說，可發現依序執行指令，則可確保正確無誤的程式執行。本發明的具體實施例確保處理器的記憶體階層(例如 L1 快取、L2 快取、系統記憶體等等)發現該等指令的一致性依序執行。

[0040] 圖 1 顯示根據本發明之一具體實施例的一載入佇列與一儲存佇列。圖 1 也顯示一輸入指令序列。如上述，該處理器的記憶體階層(例如 L1 快取、L2 快取、系統記憶體等等)發現該等指令的一致性依序執行。該載入佇列與該儲存佇列，此後稱為載入/儲存佇列，可用來維持依序執行的語意(semantics)。

[0041] 此外，該等載入與儲存的失序執行傾向於推測性(speculative)執行。實施推測性執行時，機器需要認

知推測性錯誤。在圖 1 的具體實施例中，載入/儲存佇列提供一系統，用於從導因於負載儲存重新排序與最佳化的推測性轉送或錯失預測/錯誤當中實施復原。該載入/儲存佇列包含該硬體支援，允許由於轉送、分支與錯誤結果，從導因於載入儲存重新排序/最佳化的推測性錯誤當中復原。若要允許該機器從推測性錯誤當中復原，則在該載入佇列與該儲存佇列內維護該推測性執行的結果。該載入佇列與該儲存佇列保留該推測性執行的結果，直到修正錯誤，且該儲存結果可從記憶體當中除役(retire)。外部代理主機並無法發現該載入佇列與該儲存佇列的推測性執行內容。關於可見性，儲存必須依序從記憶體中除役。

[0042] 圖 2 顯示根據本發明之一具體實施例載入與儲存指令分割的第一圖式。本發明的一特徵在於，載入分成兩個巨集指令，第一個巨集指令進行位址計算且擷取至一暫時位置(載入儲存佇列)，第二個巨集指令為將該記憶體位址內容(資料)載入一暫存器或一 ALU 目的地。請注意，雖然用將載入與儲存指令分成兩個別巨集指令並重新排序的關聯來說明本發明的具體實施例，不過利用將載入與儲存指令分成兩個別巨集指令且在微碼關聯內重新排序，可實施相同的方法及系統。

[0043] 該功能性同樣適用於該等儲存。儲存也分成兩個巨集指令，第一指令為儲存位址與擷取，第二指令為將資料儲存在該位址上。分開儲存以及兩個指令都遵照底下用於載入所說明之相同規則。

[0044] 將載入分成兩個指令允許一執行時間最佳化器排程該位址計算，且在一已知指令序列內較早擷取指令。如此利用預先擷取資料進入與該快取階層分開的暫時緩衝區內，就可輕易從記憶體遺失當中復原。在此使用暫時緩衝區，以保證預先擷取資料在 LA/SA 與 LD/SD 之間一對一對應之可用性。若與該載入位址與該載入資料之間窗內的先前儲存有所偏差(例如若從先前儲存當中偵測到一轉送情況)，或若對於位址計算有任何問題，則重新發出對應的載入資料指令。此外，將載入分成兩個指令也可包含將資訊複製到兩個指令內。這種資訊可為位址資訊、來源資訊、其他額外識別碼等等。這種複製允許在缺乏 LA/SA 之下，單獨派遣兩個指令的 LD/SD。

[0045] 不用等待該載入資料返回，就可從實際機器除役窗當中將該載入位址與擷取指令除役，藉此允許該機器即使快取遺失該位址(例如本章節開頭時提到的該載入位置)的情況下進行轉送。例如：一快取遺失該位址時(例如位址 X)，該機器可停止(stall)數百個循環，等待從該記憶體階層擷取該資料。在不用等待該載入資料返回之下，從該實際機器除役窗當中將該載入位址與擷取指令除役，該機器仍舊可進行轉送。

[0046] 請注意，指令分離賦予本發明具體實施例一關鍵優勢，就是提早重新排序該等 LA/SA 指令，且讓該指令序列進一步脫離 LD/SD，以便提早派遣與執行該等載入與該等儲存。

[0047] 圖 3 顯示根據本發明之一具體實施例載入與儲存指令分割的第二圖式。圖 2 的具體實施例顯示如何使用複製功能，以便制定(enact)該等載入指令分離。在此具體實施例中，載入分成兩個巨集指令，第一進行位址計算且擷取至一暫時位置(載入儲存佇列)，第二個為將該記憶體位址內容(資料)載入一暫存器或一 ALU 目的地。請注意，雖然用將載入與儲存指令複製到兩個別巨集指令並重新排序的關聯來說明本發明的具體實施例，不過利用將載入與儲存指令複製到兩個別巨集指令且在微碼關聯內重新排序，可實施相同的方法及系統。

[0048] 此時說明根據本發明之一具體實施例的指令分離功能性之更詳細說明。在一具體實施例中，該指令集並沒有直接對應指令至 LA、SA、LD 或 SD。在這種具體實施例中，使用指令前置碼(prefix)、LAF、SAF、LASAF 和一協同後置碼(suffix)指令的組合來實現這些概念。且粗略映射到該 LA 上的指令集具有 LAD，且至 SA 的具有 SAD，且可實施一組合的 LADSAD。這些概念也可實施為微碼內的微指令。

a) 在此定義成 LAF 前置碼+後置碼指令可描述為「LD」。

b) 在此定義成 SAF 前置碼+後置碼指令可描述為「SD」。

c) 在此定義成 LAD 指令可描述為「LA」。

d) 在此定義成 SAD 指令可描述為「SA」。

e) 進一步具有一 LASAF 指令以及一 LADSAD 指令，這些指令包含一組合的 LAF/SAF 前置碼+後置碼指令，可用來實施旗標語(semaphore)(鎖定的原子(locked-atomic))操作。其也可定義一組合式 LAD-SAD 指令，再次預先擷取該等記憶體運算元，而具有硬體內的結果複雜度。

[0049] LAD 代表「LA 解除融合(LA-defused)」

[0050] LAD 指令啟動一資料預先擷取進入該執行管線。這與正常預先擷取不同，其直接載入至該執行管線，而可承擔比第一階快取還要低的執行延遲。在一具體實施例中，藉由使用一固定儲存體用於該 LA-LD 配對，其可使用 LA-LD 對對之間的該 ID 連結(例如該 QID 編號)來標記，以便實施此功能。該 LAD 指令計算一有效記憶體位址(例如從潛在的複雜規格)，指定運算元大小(位元組、半字元、字元、雙字元或更大)、起始該記憶體參照、通過該 TLB 與快取階層。記錄例外(分頁通行遺失、特權、保護)以便在 LAF+後置碼執行上回報，或一替代實施例中可取消該 Q 輸入或使無效、強制該 LAF+後置碼指令重新執行且採用該等例外。

[0051] 該 LAD 指令具備一般格式以及運算元：

LAD(os)　　QID,M[EA]

- EA 為該有效位址規格，可為基礎暫存器、索引暫存器、位移因數及/或索引偏移，例如 M[B,RI,sf,offset]，的組合。

- os—為要讀取的位元組數指示
- QID—為用於該記憶體參照操作的該載入記憶體QID。其也可用於連結該 LAD 的操作以及後續 LAF 前置碼指令。該 QID 位於 1 至 N 的範圍內，N 為一實施特定值。預期值為 31、63、127。QID=0 保留用於 LAF 指令沒有先前 LAD 的特殊情況。LAF 會立即採用 QID=0，因此 LAD 無法使用。

[0052] LAF 代表「LA 融合(LA-fused)」。

[0053] LAF 為一指令前置碼，表示必須直接與後置碼指令連結(或融合)。該後置碼指令可單獨存在。該後置碼指令可為具有至少一來源暫存器的任何指令。然後必須連結當成前置碼的 LAF。LAF 前置碼改變了該後置碼指令的性質(nature)。利用該前置碼，將一或多個其暫存器運算元重新定義為一記憶體佇列識別碼(QID，memory queue identifier)。進一步，源自於該暫存器的相關聯資料，現在源自於該記憶體佇列。

[0054] LAF 前置碼+後置碼指令可具有或沒有先前的 LAD 指令。若 QID = 0，則該 LAF 沒有先前的 LAD，若 QID ≠ 0，則該 LAF 具有先前的 LAD。想要將該載入指令分成 LA 和 LD 時，則 LAF 將具有 QID!=≠ 0 且 LAD 將實現成在 LAF 之前具有相同 QID (例如這基本上建立了分離的 LA/LD 配對)。

[0055] 在 LAF/後置碼執行且具有 QID == 0 時，該記憶體佇列的該 0 輸入用來進行「LA」操作、記憶體讀

取、儲存資料至該記憶體佇列，然後完成將該資料載入該後置碼指令來源，且套用的該操作可能與其他來源結合，且將該結果寫入該後置碼指令目的地暫存器。在另一方面，若 QID ≠ 0，則諮詢(查找)該記憶體佇列找尋一匹配的 QID，若存在，則從該記憶體佇列讀取該資料，且套用該後置碼指令的操作，且將結果寫入該目的地暫存器。若該匹配 QID 有效但是不完整，則閒置該資料直到資料可用。若該 QID 無效，則該 LAF 具有充分資訊(位址與資料運算元大小)來重新啟動該操作。因為許多因素，所以可能並不存在匹配的 QID，某些因素為：

a) 先前的 LAD 未執行、不良編碼或其他原因。

b) 例外或中斷打斷了 LAF 與先前 LAD 之間的執行流。

c) 一介入儲存操作以該 LAD 的位址為假名(aliased)，且使其資料整體性無效。

[0056] 在任何這些情況中，該 LAF 前置碼+後置碼具有足夠資訊來重複 LAD (LA)操作。此能力讓我們的 LAD 指令得到一提示，該 LAD 並未成功執行，或就此而言，甚至超越 NOP 來實施，讓正確程式碼使用。

[0057] 含後置碼指令的 LAF 指令一般格式或運算元為：

LAF M[ea]

SUFFIX(os) Rt,QID,....

[0058] 該 LAF 指令從該後置碼指令的編碼當中借到

運算元大小、QID。若該後置碼為 SIMD，則也從該後置碼借到該操作的 SIMD 寬度。該 QID 總是在該後置碼指令的該來源暫存器規格欄位之一者內編碼。在 SMI 的特定實施當中，這總是位元 23:18，但是並不需要是如此。

[0059] SAD 代表「SA 解除融合(SA-defused)」

[0060] SAD 為 LAD 的並行指令，只用於儲存。預先擷取資料，將資料帶入快取進行修改。進一步，其建立一記憶體儲存佇列輸入。SAD 主要具有 2 種主要用途：

a) 當成一預先擷取，讀取用於資料修改

b) 維持正確記憶體順序，且在一儲存(寫入)之前提昇一載入(讀取)之後揭露與處理潛在寫入後讀取危險

[0061] SAD 為提示指令。該 SAD 指令計算一有效記憶體位址(從潛在的複雜規格)，指定運算元大小(位元組、半字元、字元、雙字元、...)、啟動該記憶體參照、通過該 TLB、快取/記憶體階層。例外(分頁通行遺失、特權、保護)都記錄在 SAF+後置碼執行上，以便重新執行且採用該等例外。

[0062] 該 SAD 指令具備一般格式以及運算元：

SAD(os) M[ea],QID

- Ea 為該有效位址規格，可為基礎暫存器、索引暫存器、位移因數及/或索引偏移，例如 M[B,RI,sf,offset]，的組合。
- os—為要讀取至該 Ea 的位元組數指示
- QID—為用於該記憶體參照操作的該儲存記憶體

QID。其也可用於連結該 SAD 的操作以及後續 SAF 前置碼指令。該 QID 位於 1 至 N 的範圍內，N 為一實施特定值。預期值為 31、63、127。QID=0 保留用於 SAF 指令沒有先前 SAD 的特殊情況。總是由 SAF 立即使用此 QID。

[0063] SAF 代表「SA 融合(SA-fused)」

[0064] SAF 為 LAF 前置碼的並行前置碼，只用於儲存。身為前置碼，則必須直接與後置碼指令連結(或融合)。該後置碼指令可單獨存在。該後置碼指令可為具有至少一目標暫存器的任何指令。然後必須連結當成前置碼的 SAF。SAF 改變了該後置碼指令的性質：一或多個該等目的地暫存器運算元為正常暫存器選擇索引至一記憶體儲存佇列識別碼(QID)，且該操作從目標在一暫存器變成目標在一記憶體(更精準的記憶體佇列輸入)。如此，將暫存器操作改變為儲存記憶體操作。

[0065] 該 SAF 指令可具有或沒有一先前 SAD。若 QID = 0，則該 SAF 沒有先前的 SAD，若 QID ≠ 0，則該 SAF 具有先前的 SAD。在 SAF/後置碼執行且若 QID = 0 時，該記憶體佇列的該 0 輸入用來進行「SA」操作、記憶體寫入、儲存資料至該記憶體佇列，然後完成儲存該後置碼指令來源供應的該資料。在另一方面，若 QID ≠ 0，則諮詢(查找)該記憶體佇列找尋一匹配的 QID，若存在，則在套用該後置碼指令的操作時，將該資料寫入該記憶體佇列輸入。若該匹配 QID 有效但是不完整，則閒置該資料到資料可用。若該 QID 無效，則該 LAF 具有充分資訊

(位址與資料運算元大小)來重新啟動該操作，且完成該記憶體寫入操作。因為許多因素，所以可能並不存在匹配的QID，某些因素為：

a) 先前的 SAD 未執行、不良編碼或其他原因。

b) 例外或中斷打斷了 SAF 與先前 SAD 之間的執行流。

[0066] 在任何這些情況中，該 SAF 前置碼+後置碼具有足夠資訊來重複 SAD (SA)操作。此能力讓我們的 SAD 指令得到一提示，該 SAD 並未成功執行，或就此而言，甚至超越 NOP 來實施，讓正確程式碼使用。

[0067] LASAF 為一指令前置碼

[0068] LASAF 為一前置碼，將具有相同暫存器的指令修改至為一來源與一目的地。LASAF 將這種指令改變成一基本記憶體參考讀取/寫入一次操作。使用來自該載入記憶體佇列的其中之一者，以及來自該儲存記憶體佇列的其中另一者。在此並無先前的 LAD 或 SAD 指令。

```
LASAF    M[ea3]
ADD      QID1,QID2,R1
```

[0069] LASAF 在該載入與儲存記憶體佇列內建立 QID 輸入，然後使用 QID2 讀取記憶體[ea3]、加入 R1 且將結果除存在儲存記憶體 QID1 內，達成 M[ea3]的基本讀取修改寫入。

[0070] 該 SMI 實施(若要實施這個)應該需要 QID1=QID2=0，但是本發明不欲侷限於此。

[0071] 可擁有 LASAD 指令，但是必須窺探進入該記憶體佇列的所有通道來達成。使得窺探上命中的輸入無效。然後在該 LASAF 上重新執行載入/儲存。

[0072] 範例用途：

[0073] A.在提昇遠在使用該資料之前的一載入之後，節省暫存器資源。

假設原始碼為：

```
LDR R1,M[ea1]
ADD32  Rt,R1,R2
```

為了隱藏記憶體存取延遲，要盡早在執行流內提昇該 LDR 在使用 R1 資料(該 ADD)之上。

```
LDR R1,M[ea1]
...許多指令
ADD32  Rt,R1,R2
```

[0074] 如此有一缺點，就是將該 R1 暫存器維持在「忙碌」狀態來等待資料，這樣就無法用於其他用途。該記憶體佇列擴充資源庫來保留資料。如此將 LDR 轉換成一 LAD 以及一後續 LAD：

```
LAD QID,M[ea1]
...許多指令
LAF M[ea1]
ADD32 Rt,QID,R2
```

因為載入佇列輸入 QID 已經用過，所以 R1 可用於其他用途。

若用 Rt-QID 的差載入 Rt，或若 QID 不存在，則從減去 R2 的 M[ea1]當中重新載入資料，且將結果放入 Rt。

[0075] 請注意，在上述實施中，該記憶體位址計算在這 2 次載入之間並不需要匹配。若兩個位址計算不同，且該 QID 仍舊有效，則最有可能是程式編輯錯誤。該 OS (在範例 32 內)也不用在 LAD 與 LAF 之間匹配。該 LAD 會讀取比所需要更多的資料，在此情況下，將使用所讀取資料的最低有效位元組。或者該 LAF+後置碼會比該 LAD 讀取需要更多資料，在此情況下，將使用該 LAD 所讀取的最低有效位元組，後面接著 0 直到該後置碼操作足夠。進一步，該等位址計算運算元不用在該 LAD 與 LAF 之間匹配，不過為了有良好的編碼，應該取得相同的結果有效位址。

[0076] B.利用將正常 ALU 暫存器-暫存器操作逐漸變成 ALU 暫存器記憶體操作，以節省執行循環。

[0077] 在此使用該特殊 QID=0 (%0)，將該正常暫存器-暫存器 ADD 指令改變為一記憶體參照。因為 LAF 為前置碼且必須直接連結至該 ADD，所以之間並無指令。該 QID=0 總是立即可用。

```
LAF M[ea1]
ADD32   Rt,%q0,R2
```

LAF 將上面的指令有效變成

```
ADD32       Rt,M[ea1],R2
```

在此也可使用 SAF 將 ALU 暫存器-暫存器操作變成將

該 ALU 操作結果儲存到記憶體的操作。

SAF M[ea1]

ADD %q0,R2,R3

將加入的 R2 和 R3 之結果儲存到記憶體內位址 ea1 上。

[0078] C.提昇載入位於儲存之上時，保留順序語意。

[0079] 另一問題是我們要提昇該載入(LDR)在一儲存(STR)之上，可具有或沒有該載入位址的別名。別名：ea1 的某些或全部資料位址都與 ea2 相同。

STR M[ea2]

LDR R1,M[ea1]

ADD32 Rt,R1,R2

變成

LDR R1,M[ea1]

0-to-many instructions

STR M[ea2],R3

0-to-many instructions

ADD32 Rt,R1,R2

[0080] 要安全地執行此動作(保證正確編碼)，需要有工具。因此使用 LAD 和 SAD 指令及其個別 LAF 和 SAF 前置碼+後置碼指令，且可保留執行順序並保證正確編碼。

LOOP:

SAD M[ea2],R3

0-to-many instructions

a)

LAD R1,M[ea1]

0-to-many instructions

b)

SAF M[ea1],R3

0-to-many instructions

c)

saf-suffix

LAF M[ea1]

d)

BRN LOOP

[0081] 在上面的程式碼中，必須提昇 LAD 和 SAD 且維持相同順序。這會發生什麼事？在 a、b、c、d 每一點上，分別指示一替代項。

a1) 中斷、讓該 SAD 無效，後續 SAF 將必須重新執行

a2) LAD 別名 SAD、使得 LAD 無效或要插入該記憶體佇列內

b1) 中斷、使得 SAD 和 LAD 無效

b2) SAF 別名 LAD 且使得 LAD 無效

b3) SAF 同時使用仍舊有效的 SAD 或重新執行。

c1) 中斷、使得 LAD 無效，

c2) 若仍舊有效的 LAF 使用 LAD 的資料，否則重新執行。

c) 迴圈、執行硬體功能、標記 IP 與執行順序 ID 的組合以及正確管理 QID、LAD/SAD/LAF/SAF。

[0082] 在上面有關 LA/SA 和 LD/SD 的說明當中，使用 LA 和 SA 相對程式順序位置，針對轉送目的強制該順序。在另一具體實施例中，使用該 LD/SD 相對程式順序位置，針對轉送目的強制該順序(如底下所述)。

[0083] 圖 4 顯示一處理 400 的步驟流程圖，其中根據本發明之一具體實施例，繪製用於從導因於負載儲存重新排序與最佳化的推測性轉送錯失預測/錯誤當中實施復原之規則。

[0084] 如步驟 401 所例示，本發明具體實施例的目標為在一儲存與一載入之間位址匹配時，找出轉送至該載入的儲存。在步驟 402，將最早的儲存(例如在機器順序中)轉送至該載入。

[0085] 在步驟 403，以機器順序分配 LD/SD 時，會更新 LA/SA 的實際時間歷程(ages)。該 LA/SA 實際時間歷程會指派給與該 LD/SD 時間歷程相同之值。該 LD/SD 維持實際時間歷程，且強制(enforce)該原始程式順序語意。

[0086] 步驟 404-407 顯示維持程式順序語意同時支援推測性執行的規則。步驟 404-407 顯示彼此水平排列，以指示同時實施這些規則功能的機制。如步驟 404 內所示，若一儲存具有實際時間歷程，但是該載入尚未獲得實際時間歷程，則該儲存比該載入還要早。如步驟 405 內所示，若一載入具有實際時間歷程，但是該儲存尚未獲得實

際時間歷程，則該載入比該儲存還要早。如步驟 66 內所示，若該載入或該儲存都未獲得實際時間歷程，則將使用一虛擬識別碼(VID，virtual identifier)來找出哪個比較早(例如在某些具體實施例中，與該載入/儲存指令相關聯的該 QID 代表該 VID)。如步驟 407 內所示，若載入與儲存都已經獲得實際時間歷程，則使用該實際時間歷程找出哪個較早。

[0087] 請注意，圖 4 具體實施例所描述用來決定載入與儲存之間相對時間歷程的演算法也可用來決定複數個儲存之間的相對時間歷程。這在底下圖 4 以及後續圖式內說明的更新該儲存時間歷程戳記非常有用。

[0088] 圖 5 顯示根據本發明之一具體實施例，例示其中使用一處理器的載入佇列與儲存佇列資源來實施處理 400 的該等規則之方式圖式。圖 5 具體實施例顯示一範例，其中將指令迴圈展開成兩個一致的指令序列 401-402。請注意，該 SA 和 LA 可自由重新排序，不過，該 SD 和 LD 則必須維持其相對程式順序。較早的儲存可轉送至較晚的載入。較早意味著較小的 VID (例如該虛擬 ID 表內所維護)或較小時間歷程。若一 SA 具有一 VID 但是無時間歷程，則該 SA 比具有時間歷程的一載入年輕。LA/SA 的實際時間歷程會在分配 LD/SD 時更新，並指派與該 LD/SD 相同的時間歷程。若一儲存或一載入具有實際時間歷程，則與該實際時間歷程比較，否則使用 VID 時間歷程。

[0089] 請注意，該 VID 表的作用為利用儲存該 LA/SA 對應機器 ID 以及對應每一 VID 唯一識別碼的機器資源，追蹤該 LA/SA 與 LD/SD 指令之間的關聯性。另請注意，「VID」一詞與圖 2A 和圖 2B 的討論中所描述之「QID」意義相同。

[0090] 此時描述圖 4 具體實施例的操作範例。一開始，考慮其中該分配指標器 410 一開始為 3 的案例。V3 LA 已經在該載入 Q 輸入第 4 號內派遣且分配。V1 SA 和 V2 SA 都已經派遣，其與 V3 LA 比較，且因為 V2 SA 小於且比 V1 SA 更靠近 V3 LA，則其潛在轉送至 V3 LA，如此其針對 V3 LA 載入 Q 輸入來更新該儲存初始時間歷程。

[0091] 此時該分配指標器移動至 6。此時 V2 SA (5 號)的儲存實際時間歷程更新該 V3 LA 載入 Q 輸入(因為 V2 SA 為已經標記成轉送至此載入的紀錄儲存)。此時派遣 V4 SA 並與該載入初始時間歷程比較，且因為 V4 大於 V3 LA，所以不會轉送。此時該分配指標器移動至 11。在 V3 LD 的分配時間上，用 V3 LD (7 號)的實際時間歷程更新該載入 Q 輸入 4 號。此時派遣 V1 SA 11 號。此時因為 V3 LA 1 號具有實際時間歷程，但是 V1 SA 11 號沒有，則該載入比該儲存還要早，如此不可能轉送。

[0092] 該預測表用於偵測該預設假設不正確的情況。該預設假設就是並無儲存轉送至一載入。一旦偵測到用於一載入儲存配對的轉送，則記錄該載入儲存配對的該

程式計數器，如此該載入將總是等待派遣該儲存位址，且計算位址以找出載入味至是否匹配儲存位置，因此需要從此處轉送。

[0093] 請注意在一具體實施例內所描述的特徵，其中允許缺少該 LA/SA 時派遣該 LD/SD，幫助在分支之前或在已知指令序列下的分支範疇內重新排列 LA/SA。若隨分支結果省略該 LA 和 SA，或因為導致記憶體例外的結果而省略，該 LD 和 SD 仍舊可正確運作，因為其包含派遣兩次的必要資訊：首先當成一 LA/SA，再來當成一 LD/SD。在這種情況下，該 LD/SD 的第一次派遣執行位址計算(例如載入位址)。因此，相同 LD/SD 可再次派遣，以滿足該載入或儲存的消耗部分(例如載入資料)。此機制可稱為該等載入與儲存指令的「雙派遣」。

[0094] 請注意在一具體實施例中，該對應的解除融合 LA/SA 不存在時(例如在具有融合 LD/SD 的情況下)發生該 LD/SD 的雙派遣，或若由於分支結果省略該 LA/SA，或由於導致一記憶體例外的結果而忽略等等。

[0095] 上述雙派遣功能確保 LD/SD 正確執行，獨立於遺失、忽略或省略的 LA/SA。上述特徵所提供的優點在於：利用較早排程該 LA/SA，甚至在分支存在、潛在錯誤、例外等等之時，由該載入/儲存指定的資料預先擷取可在程式順序中較早開始(例如減少延遲)。

[0096] 圖 6 顯示根據本發明之一具體實施例，例示其中使用一處理器的載入佇列與儲存佇列資源來實施處理

400 的該等規則之方式另一圖式。在圖 6 具體實施例中，考慮其中該分配指標器一開始為 3 的案例。V3 LA 已經在該載入 Q 輸入第 4 號內派遣且分配。此時該分配指標器移動至 6。此時 V1 和 V2 (4 號、5 號)的儲存實際時間歷程更新具有機器 ID 2 和 3 的該對應 SA 之時間歷程。此時派遣 V4 SA 並與該載入初始時間歷程比較，且因為 V4 SA 大於 V3 LA，所以不會轉送。此時該分配指標器移動至 11。在 V3 LD 的分配時間上，用 V3 LD (7 號)的實際時間歷程更新該載入 Q 輸入 4 號。此時派遣 ID 10 的 V1 LA。

[0097] 此時已經派遣機器 ID 2 的 V1 SA 以及機器 ID 3 的 V2 SA，這兩者與 ID 10 的 V1 LA 比較，且因為 ID 10 的 V1 LA 並無機器時間歷程(尚未分配其對應 LD)，而機器 ID 2 的 V1 SA 以及機器 ID 3 的 V2 SA 都具有實際時間歷程，因此吾人了解 V1 和 V2 儲存都比 V1 還要早/老。然後將這兩個儲存最新的一(V2)轉送至 ID 10 的 V1。此時派遣 SA (V2) 11 號。因為 V1 LA 和 V2 SA 沒有實際時間歷程，所以使用其 VID 的時間歷程來比較，且未偵測到轉送。此時該分配指標器移動至 16。此時派遣 ID 16 的 V4 SA 且與 ID 10 的 V1 LA 比較，因為該 V1 LA 具有實際時間歷程但是 V4 SA 沒有，所以該 V4 SA 在 V1 LA 之後。因此，不可能從此儲存轉送至此較早儲存。

[0098] 圖 7 顯示根據本發明之一具體實施例，例示其中使用一處理器的載入佇列與儲存佇列資源來實施處理 400 的該等規則之方式另一圖式。在圖 7 具體實施例中，

考慮其中該分配指標器一開始為 3 的案例。V1 LA 和 V2 SA 已經在該載入 Q 輸入第 4 號和第 5 號內派遣且分配。此時該分配指標器移動至 6 且已經派遣 V4 SA。V1 SA 和 V2 SA 兩者都取得 4 和 5 的實際時間歷程。

[0099] 此時該分配指標器移動至 11。V3 LA 取得 7 的實際時間歷程。此時已經派遣 V1 SA 10 號和 V2 SA 11 號。V3 LA 已經派遣且將其位址與該儲存 Q 輸入比較，找出跨越 V1 SA、V2 SA 和 V4 SA 以及 V2 SA 11 號的匹配。因為 V3 LA 擁有 7 的實際時間歷程，則將其實際時間歷程與屬於 V2 SA 的最接近儲存時間歷程比較，就是時間歷程 5，如此仍舊從此儲存轉送載入，且將此標記在該載入 Q 內。

[0100] 圖 8 顯示根據本發明之一具體實施例，其中在一載入之後派遣一儲存的該派遣功能概要之處理 800 流程圖。

[0101] 處理 800 是從步驟 801 開始，其中一儲存指令分成 SA 和 SD。如稍早所描述，該 SA 指令維持與該 SD 指令的語意，允許在該分離 SA 與該剛分配 SD 之間該 VID 表內無匹配的事件中雙派遣。在步驟 802，SA 重新排序成較早機器可見程式順序，且使用一 VID 表來追蹤該 SA，以獲得該原始 SD 程式順序。在步驟 803，該 SA 的派遣上，針對該儲存佇列內所有載入檢查位址是否匹配該 SA。

[0102] 在步驟 804，當位址匹配，藉由使用該等載入

的該 VID 編號或使用該等載入與該等儲存的實際時間歷程，該匹配載入的該程式順序與該 SA 的程式順序比較。此程序已經在稍早圖 3 的討論中說明。若一儲存具有實際時間歷程但是載入沒有，則該儲存比該載入要早。若一載入具有實際時間歷程但是儲存沒有，則該載入比該儲存要早。若一載入或一儲存都沒有實際時間歷程，則使用一虛擬識別碼(VID)來找出哪一較早。若一載入與一儲存都有實際時間歷程，則使用該實際時間歷程找出哪個較早。如上述，該 VID 編號允許追蹤原始程式順序以及該重新排序的 SA 和 LA。該 VID 表內的輸入允許該對應 SD 和 LD 取得與已經指派給該 SA 和 LA 的機器資源(在這些資源已經分配時)之關聯。

[0103] 在步驟 805，對於在該程式順序內屬於稍後的載入而言，該儲存將檢查來查看該等載入是否已經由其他儲存轉送。在步驟 806，若是，則該儲存檢查先前已經轉送至此載入的該儲存之戳記，以查看該儲存在程式順序內是否早於本身。在步驟 807，若是，則該儲存檢查先前已經轉送至此載入的該儲存之戳記，以查看該儲存在程式順序內是否早於本身。在步驟 808，若否，則該儲存不會轉送至此載入。

[0104] 圖 9 顯示根據本發明之一具體實施例，其中在一儲存之後派遣一載入的該派遣功能概要之處理 900 流程圖。

[0105] 在步驟 901，一載入指令以上述方式分成 LA

和 LD。在步驟 902，LA 重新排序成較早機器可見程式順序，且使用該 VID 表來追蹤，如上面所述。反之步驟 903，該 LA 針對該儲存順序內所有儲存，檢查位址是否匹配該載入。

[0106] 在步驟 904，當位址匹配，藉由使用該載入和該儲存的該 VID 編號或使用該載入與該儲存的實際時間歷程，該匹配載入的該程式順序與該儲存的程式順序比較。此程序已經在稍早圖 3 的討論中說明。若一儲存具有實際時間歷程但是載入沒有，則該儲存比該載入要早。若一載入具有實際時間歷程但是儲存沒有，則該載入比該儲存要早。若一載入或一儲存都沒有實際時間歷程，則使用一虛擬識別碼(VID)來找出哪一較早。若一載入與一儲存都有實際時間歷程，則使用該實際時間歷程找出哪個較早。如上述，該 VID 編號允許追蹤原始程式順序以及該重新排序的 SA 和 LA。接著在步驟 905，該載入消耗來自程式順序中最接近其自己程式順序的該儲存之資料。

[0107] 圖 10 顯示根據本發明之一具體實施例的統一載入佇列之圖式。虛擬載入/儲存佇列的目標為允許該處理器使用其載入/儲存佇列的實際大小，在該機器內分配比可容納還要多的載入/儲存。相對地，這允許該處理器除了超越其載入/儲存佇列的處理器實際大小限制的載入/儲存以外，分配其他指令。即使某些該等載入/儲存在該載入/儲存佇列內仍舊沒有空間，這些其他指令仍舊可派遣與執行。

[0108] 隨著載入從該載入佇列當中除役，該載入派遣窗移動至該序列中後續指令，且將包含比從該載入佇列移除的載入數量還要多的已分配載入派遣考量。在此圖式中，該載入派遣窗將從左往右移動。

[0109] 在一具體實施例中，該載入派遣窗總是包含等於該載入佇列內輸入數量的載入數量。任何時間上都沒有載入可以派遣到該載入派遣窗之外。該排程器窗內載入以外的其他指令(例如減、加等等)都可派遣。該載入派遣窗內的所有載入只要備妥之後都可派遣。

[0110] 圖 11 顯示根據本發明之一具體實施例顯示該滑動(sliding)載入派遣窗的統一載入佇列。圖 11 顯示相較於圖 10 在時間內的一後續實例。隨著載入從該載入佇列當中除役，該載入派遣窗移動至該序列中後續指令，且將包含比從該載入佇列移除的載入數量還要多的已分配載入派遣考量。該載入派遣窗總是包含等於該載入佇列內輸入數量的載入數量。任何時間上都沒有載入可以派遣到該載入派遣窗之外。該排程器窗內載入以外的其他指令(例如減、加等等)都可派遣。該載入派遣窗內的所有載入只要備妥之後都可派遣。如此用此方法所獲得的一好處為：若已經超出該載入或該儲存佇列容量，對於該排程器的分配並不會停止，而是繼續將指令分配至包含載入與儲存的排程器，而不管是否超出該載入或儲存佇列容量，該載入與儲存動態窗將確保不會派遣該載入或儲存佇列容量之外的載入或儲存。

[0111] 圖 12 顯示根據本發明之一具體實施例的一分配載入佇列。圖 12 具體實施例的目標為實施一分配載入佇列以及一分配儲存佇列，其維持單一程式/執行緒依序語意，但是仍舊允許載入與儲存失序派遣通過多個核心/記憶體片段。

[0112] 圖 12 圖解顯示避免死結的一載入佇列擴充解決方案。該載入/儲存佇列的擴充已經建立，且從導致該死結的該載入/儲存點(從此點算起)，以程式順序將死結載入/儲存分配至擴充佇列，直到該載入/儲存佇列可自由輸入為止。在圖 12 的案例中，LD 3 根據 SD 載入，而後者根據 LD 2 (具有映射至載入_Q B 的位址)載入，而因為該載入_Q B 已滿，所以無法派遣。在此死結案例中，在偵測到死結時，允許以一接著一的順序派遣 LD 1 和 LD 2 至保留部分 B 然後除役。一種分散載入/儲存佇列的保存政策為替每一載入/儲存保留每一載入/儲存分配佇列內一輸入。在此圖中，每一已分配載入需要在載入_Q A 內保留一輸入，且在載入_Q B 內保留另一輸入。

[0113] 請注意在已分配的載入/儲存佇列內，有一關於已分配載入/儲存的問題，就是在分配時並不知道其位址。因為此問題，只有在一已知載入或儲存將佔用該已分配佇列的失序派遣期間才會知道位址。

[0114] 本發明的具體實施例可運用三種不同的解決方案，讓該已分配載入/儲存佇列避免失序派遣時的死結：

1. 導致一錯失預測並將產生死結(沒有空間派遣至該載入/儲存緩衝區)的最早載入/儲存清除(flush)，且依序開始派遣載入/儲存一段時間，或利用保留分配，其中每一載入/儲存都分配所有已分配佇列內的空間。一旦已知該載入/儲存的位址(在派遣時)，如此已知將接收該載入/儲存的該特定載入佇列，然後可姜琦他佇列內的保留空間解除分配。

2. 該載入/儲存佇列的擴充已經建立，且從導致該死結的該載入/儲存點，以程式順序將死結載入/儲存分配至擴充佇列(圖 9)。

3. 動態派遣窗調整大小，其中在該連續窗之外未派遣載入的總和應該小於或等於該特定載入佇列內自由未保留空間的數量(例如圖 11)。

[0115] 圖 13 顯示根據本發明之一具體實施例具有一依序連續窗的一分配載入佇列。決定動態派遣窗調整大小，如此在該連續窗之外未派遣載入的總和應該小於或等於該特定載入佇列內自由未保留空間的數量。每一載入佇列都將如本說明書所示，使用其個別派遣窗來追蹤其輸入。任何時間上用於每一載入佇列的動態窗大小 = 該佇列加上虛擬輸入的實際大小(在此情況下為 6 + 4 =10)，如此在此情況下，該窗大小應該只覆蓋 10 個載入。請注意，其他佇列的載入並未算在內(例如 LD 4)。

[0116] 保留的訂位比例(booking ratio)為 3。訂位比例為競爭每一保留空間的順序載入數量。在此範例中，只

有頭兩個依序未派遣載入(從左至右掃描該依序連續窗)可派遣至該保留部分(假設已經指派保留該佇列的 2 個輸入)。因此，虛擬輸入的數量 =(訂位比例 - 1) * 保留輸入的數量 =(3-1)*2=4。

[0117] 關於該依序連續窗調整大小，任何時間上尚未派遣給該載入佇列內一輸入(擷取空間)的載入數量(從最舊算到最新)加上已派遣至該保留空間的載入數量必須小於或等於(該訂位比例 * 保留輸入的輸入)。在此情況下，載入的數量必須小於或等於 3。該訂位比例為設計可設置效能公制(performace metric)，其決定可接受的保留空間比例(佔用對上訂位)的比例為何。這用在最早未派遣載入無法找出一佇列空間來派遣到該保留輸入之外的情況。在這種情況下，從最早(最舊)載入開始的這些載入將競爭該保留空間，該訂位比例決定有多少載入將等待佔用每一保留輸入，該等保留輸入總是先指派給最舊的未派遣載入，且一旦該載入除役，則次舊的載入佔用該輸入(該訂位比例決定從最早派遣的開始，一個接著一個佔用保留輸入的這些載入數量)。

[0118] 請注意在一具體實施例中，在該佇列的未保留部分內已經沒有空間時，來自每一佇列的該依序連續窗之載入可派遣至該佇列的保留空間(依序從最舊的載入開始)。另請注意在一具體實施例中，在該佇列的依序連續窗之外且在該佇列的動態派遣窗之內的載入無法派遣至該佇列的保留部分。

[0119] 另請注意，一旦該佇列的未保留部分內有空間，則該佇列的完整動態派遣窗內之任何載入都會失序地派遣至任何該已分配佇列的未保留部分之任何輸入。該佇列的該依序連續窗與該動態派遣窗之大小可隨每一循環調整，以反應上面在每一載入派遣或除役之後所提供的等式內所陳述之大小限制。

[0120] 圖 14 顯示根據本發明之一具體實施例用於一多核心處理器的一分段記憶體子系統之圖式。圖 13 顯示一般執行續之間及/或載入與儲存之間同步方法之各種方法與實施。該方法描述用於記憶體參照跨越載入/儲存架構及/或跨越記憶體參照及/或執行緒的記憶體存取之同步與免消歧義(disambiguation)較佳方法。在圖 15 中，顯示暫存器檔案(位址及/或資料暫存器)的多個區段，搭配執行單元、位址記算單元以及第 1 階快取片段及/或載入儲存緩衝區和第 2 階快取，以及位址暫存器互連 1200 和位址計算單元互連 1201。在一核心/處理器內利用將其集中化資源分成許多引擎，可建構這些片段的元件，或可從多核心/處理器組態內不同核心/處理器的元件，來建構這些元件。圖式內用片段編號 1 顯示這些片段 1211 之一者；該等片段可縮放為較大數量(一般如圖式內所示縮放為 N 個片段)。

[0121] 此機制也用來當成一致性方法，運用於這些引擎/核心/處理器之間的記憶體架構。此方法從一片段/核心/處理器內位址計算單元之一者要求一位址開始，例

如：假設由片段 1 (例如 1211)要求該位址。其可使用屬於其自己的片段之位置暫存器，或從使用位址互連匯流排 1200 跨越其他片段的暫存器，來獲得並計算其位址。計算該位址之後，建立用來存取快取與記憶體的 32 位元位址或 64 位元位址之參考位址。此位址通常分段成一標記欄位以及一集合與行欄位。此特定片段/引擎/核心會將該位址儲存到其載入儲存緩衝區及/或 L1 及/或 L2 位址陣列 1202，同時將使用一壓縮技術建立該標籤的壓縮版本(包含比該位址的原始標籤欄位還少的位元數)。

[0122] 再者，不同的片段/引擎/核心/處理器將使用該集合欄位或該集合欄位的子集，當成索引來識別該位址將維持在哪個片段/核心/處理器內。此利用位址集合欄位位元的片段索引確保特定片段/核心/引擎內該位址所有權的排他性，即使對應該位址的該記憶體資料可在另一或多個其他片段/引擎/核心/處理器內存活。即使該位址 CAM/標籤陣列 1202/1206 顯示在每一片段內要與資料陣列 1207 連結，其可只連結在位置與佈局的實體附近，或甚至依照事實，兩者都屬於特定引擎/核心/處理器，但是該位址陣列內所維持位址與一片段(fragment)內該資料陣列內的資料之間並無關聯。

[0123] 圖 15 顯示本發明具體實施例如何操控載入與儲存的圖式。如圖 15 內所描述，每一片段都關聯於其載入儲存緩衝區以及儲存除役緩衝區。針對任何給定的片段，指定一位址範圍關聯於該片段或另一片段的載入與儲

存已經傳送至該片段的載入儲存緩衝區進行處理。請注意，其可失序地到達，如同該核心執行指令失序一樣。在每一核心之內，該核心不僅存取本身的暫存檔，也存取每一其他核心的暫存檔。

[0124] 本發明的具體實施例實施一種分散式載入儲存排序系統，該系統分散於多個片段。在一片段中，由該片段執行局部資料相依性檢查。這是因為該片段只有該特定片段的儲存除役緩衝區內之載入與儲存。這樣限制了必須查找其他片段以維持資料一致性之需求。在此方式中，一片段內的資料相依性在本地(locally)實施。

[0125] 對於資料相依性，該儲存派遣閘依照嚴苛的程式內順序記憶體一致性規則來實施儲存除役。儲存在該載入儲存緩衝區上失序到達，載入也在該載入儲存緩衝區上失序到達。同時，該失序載入與儲存都轉送至該儲存除役緩衝區進行處理。注意，雖然在一給定片段內儲存依序除役，當其前往該儲存派遣閘，可失序來自多個片段。該儲存派遣閘實施一政策，確保即使儲存失序通過儲存除役緩衝區，且即使該等緩衝區可將儲存以關於其他緩衝區的儲存方式失序地轉送至該儲存派遣閘，該派遣閘確保嚴格依序轉送至片段記憶體。這是因為該儲存派遣閘具有儲存除役的整體視野(global view)，且只允許儲存依序離開至該記憶體的全體可見側(visible side)，通過所有片段，例如全體(globally)。在此方式中，該儲存派遣閘當成一全體觀察者，以確保儲存最終通過所有片段，依序回到記憶

體。

[0126] 圖 16 顯示根據本發明之一具體實施例的一儲存篩選演算法之圖式。圖 16 具體實施例的目標為過濾該等儲存，以避免所有儲存必須檢查該載入佇列內所有輸入。

[0127] 儲存針對位址匹配來窺探該等快取，以維持一致性。若執行緒/核心 X 載入從一快取線讀取，則標記其載入資料的該快取線部分。在另一執行緒/核心 Y 儲存窺探該等快取時，若任何這種儲存覆蓋該快取線部分，導致執行緒/核心 X 載入錯失預測。

[0128] 過濾這些窺探的一項作法為追蹤該載入佇列輸入的參考。在此情況下，儲存必須要窺探該載入佇列。若該儲存匹配該存取遮罩，則從該參考追蹤器獲得的載入佇列輸入將導致載入輸入進行錯失預測。

[0129] 在另一作法中(其中並無參考追蹤器)，若該儲存匹配該存取遮罩(access mask)，則儲存位址將窺探該載入佇列輸入，且將導致該匹配的載入輸入進行錯失預測。

[0130] 用這兩種作法，一旦從一快取線讀取一載入，則設定該個別存取遮罩位元。當該載入除役時，則重設該位元。

[0131] 圖 17 顯示根據本發明之一具體實施例，具有構成從記憶體內依序讀取載入的記憶體一致性模型內失序載入之旗標語實施。如本說明書內所使用，『旗標語』一詞就是提供存取控制用於多個執行緒/核心至共用資源的

資料建構。

[0132] 在圖 17 的具體實施例中，該存取遮罩用於由多個執行緒/核心控制存取至記憶體資源。該存取遮罩利用追蹤快取線的哪個字元具有等待中載入(pending load)來運作。存取該快取線的該字元時，一失序載入設定該遮罩位元，且在該載入除役時清除該遮罩位元。若來自另一執行緒/核心的儲存寫入至該字元，而該遮罩位元已經設定，則用信號通知對應該載入的該載入佇列輸入(例如透過追蹤器)，用其相關指令錯失預測/清除或除役。該存取遮罩也追蹤執行緒/核心。

[0133] 在此方式中，該存取遮罩確保正確實施該記憶體一致性規則。記憶體一致性規則規定儲存依序更新記憶體，且載入依序從記憶體讀取，讓此旗標語運作通過該兩個核心/執行緒。如此，將正確執行核心 1 和核心 2 執行的程式碼，其中這兩者都存取該記憶體位置「旗標(flag)」和「資料」。

[0134] 圖 18 顯示根據本發明之一具體實施例，利用使用一鎖定式模型與一交易式模型，失序載入至構成從記憶體內依序讀取載入的記憶體一致性模型內。

[0135] 如上述，記憶體一致性規則規定儲存依序更新記憶體，且載入依序參照記憶體，以便該兩個核心/執行緒正確通訊。在圖 18 的右下角內顯示兩個核心，分別是核心 1 和核心 2。此時使用旗標與資料這兩個記憶體資源，進行通訊且在核心 1 與核心 2 之間共享資料。例如：

核心 1 要將資料傳遞至核心 2 時，依照核心 1 內的程式碼所指示，將儲存該資料然後傳送該旗標。依照核心 2 內程式碼所指示，核心 2 將載入該旗標，且檢查該旗標是否等於 1。若該旗標不等於 1，則核心 2 將跳回且持續檢查該旗標，直到該旗標等於 1。在該時間點上，將載入該資料。

[0136] 在運用失序架構而載入與儲存以失序執行的情況下，可使用一鎖定型記憶體一致性模式，以確保兩個輸入(例如核心 1 和核心 2)維持依序記憶體一致性語意。這透過使用一存取遮罩、一執行緒 ID 暫存器以及該追蹤暫存器來顯示。利用設定程式碼關鍵段落內任何載入的對應存取遮罩位元，來設定該鎖定。若發生從另一執行緒/核心至該快取線字元的任何存取，則該鎖定將避免該存取。在一具體實施例中，這可利用將該存取當成一遺失來實施。該鎖定已經清除時，就允許存取該字元。

[0137] 選擇性地，可使用一交易型方法以維持依序記憶體一致性語意。運用該交易型方法，利用設定一交易之內任何載入的對應存取遮罩位元，來設定基本單元(atomicity)。若發生任何從另一執行緒/核心或並行交易存取至該快取線字元，而該遮罩位元已經設定，則用信號通知對應該載入的該載入佇列輸入(例如透過追蹤器)，用其相關指令錯失預測/清除或除役。該存取遮罩也追蹤執行緒/核心。該交易結束時，將清除該遮罩位元。該執行緒 ID 暫存器用來追蹤哪個執行緒存取統一儲存佇列輸入的

哪個字元。

[0138] 圖 19 顯示根據本發明之一具體實施例的多核心分段(segmented)記憶體子系統之複數個核心。此具體實施例顯示來自多核心分段記憶體子系統的載入，如何避免存取已經標示為進行中交易一部分的字元(例如類似於鎖定的情況)。

[0139] 請注意，若此多核心分段子系統為大型叢集的一部分，則此處具有含共享記憶體子系統的外部處理器/核心/叢集。在此情況下，屬於其他外部處理器/核心/叢集的載入應進行，且若該記憶體位置為交易存取的一部分時，則無法避免意外從任何記憶體位置載入。不過，所有載入將標記該存取遮罩，將其為交易一部分的訊息告知未來的儲存。

[0140] 來自其他處理器的窺探儲存會將其位址與該遮罩比較，若一儲存知道其嘗試要儲存的該位址標記在來自另一執行緒載入(屬於交易一部分的載入)之存取遮罩內，則該儲存將導致該載入錯失預測。否則，會在該載入除役時清除該標記(例如藉此完成交易)。

[0141] 圖 20 顯示根據本發明之一具體實施例非同步核心存取一統一儲存佇列之圖式，其中儲存可根據儲存資歷將資料轉送至執行緒內的載入。

[0142] 如上述，記憶體一致性規則規定儲存依序更新記憶體，且依序從記憶體讀取載入，如此該核心/執行緒正確通訊。在圖 20 的右下角內顯示兩個核心，分別是

核心 1 和核心 2。該兩核心非同步且執行每一核心內指示的程式碼，以存取該旗標與該資料記憶體資源。

[0143] 在圖 20 的具體實施例中，該等複數個執行緒並不知道(agnostic)要存取的該統一儲存佇列。在此實施當中，來自不同執行緒的儲存可轉送至不同執行緒的載入，而藉由依循一組演算法規則，仍舊維持依序記憶體一致性語意。執行緒可根據儲存資歷彼此轉送。

[0144] 在相同執行緒內之前所有載入與儲存都已經執行時，則一儲存變成資深儲存。接收來自另一執行緒的轉送之執行緒無法獨立地除役載入/儲存。若其他執行緒接收的轉送具有錯失預測，則執行緒必須有條件錯失預測。一特定載入可從相同執行緒轉送儲存轉送，或若相同執行緒內無除存轉送給它時，從不同執行緒資深儲存轉送。

[0145] 運用圖 20 的方法，利用將任何存取的對應存取遮罩位元設定為該統一儲存佇列輸入內字元之內的位元組，來設定基本單元。若發生任何從另一執行緒/核心或並行交易存取至該儲存佇列輸入字元，而該遮罩位元已經設定，則用信號通知對應該載入的該載入佇列輸入(例如透過追蹤器)，用其相關指令錯失預測/清除或除役。該存取遮罩也追蹤執行緒/核心。該交易結束時，將清除該遮罩位元。

[0146] 圖 21 顯示根據本發明之一具體實施例，說明其中儲存具有資歷(seniority)的功能性之圖式。如圖 21 內

所描述，將從相同執行緒轉送儲存轉送一特定載入。若並無來自該執行緒的轉送，則可從不同的執行緒資深儲存轉送。此原理作用於多核心/執行緒存取共享記憶體的情況下。在這種情況下，不過，只有在從該執行緒內無轉送至一特定載入的情況下，儲存可根據儲存資歷，從執行緒轉送到從執行緒儲存。在相同執行緒內之前所有載入與儲存都已經執行時，則一儲存變成資深儲存。

[0147] 此外請注意，一執行緒無法單獨除役載入/儲存。另一執行緒接收一轉送儲存錯失預測或清除時，該執行緒必須載入錯失預測。

[0148] 圖 21 圖解兩非同步核心/執行緒(例如核心/執行緒 1 和核心/執行緒 2)之間的範例執行串流。線 2101-2105 顯示其中儲存根據旗資歷轉送至不同載入之情況。為了幫助例示資歷如何在儲存之運用，每一指令旁邊都加上編號，顯示不同的執行階段，進度從 0 至 14。尤其是請注意，根據上述規則，其中線 2103 所指示的該儲存轉送至相同執行緒內一載入。如此，如上述，在從本身執行緒轉送的載入無法從任何相鄰執行緒轉送。這會在轉送線上顯示黑色打叉。

[0149] 圖 22 顯示根據本發明之一具體實施例，非推測性的一免消歧義失序載入佇列除役實施(例如產生低功率、低晶粒面積以及較少時機關鍵)。

[0150] 儲存除役/重新排序緩衝區(SRB，store retirement/reorder buffer)可在兩種實施當中操作：除役實

施以及重新排序實施。

[0151] 在除役實施當中，儲存在儲存除役時以原始程式順序，從該儲存佇列載入該 SRB，如此在原始程式順序當中較早的該儲存位於該 SRB 的頂端上。然後找尋後續載入的位址匹配(例如使用位址 CAM)，且從該 SRB/儲存快取內的匹配輸入轉送。在具有二或多個位址匹配的情況下，具有優先權(priorty)的編碼器可利用掃描第一來找出正確轉送輸入。這樣省下一趟至記憶體的往返，且允許機器進行轉送。若一載入已經派遣且轉送給它的該儲存已經除役至該 SRB/儲存快取，則該載入從該 SRB/儲存快取轉送，且將該配對關係記錄在該預測表。若要偵測一載入在轉送給它的該儲存已經除役至該 SRB/儲存快取之前已經派遣之情況，則該載入必須建立一位址遮罩，其中標示其本身的位址。這可用不同方式實施(例如圖 17 的具體實施例)。

[0152] 如上述，圖 17 描述利用追蹤快取線的哪個字元具有等待中載入來運作的一存取遮罩。存取該快取線的該字元時，一失序載入設定該遮罩，且在該載入除役時清除該遮罩位元。若來自相同執行緒/核心的儲存偵測到其除役並寫入至該字元，而該遮罩位元已經設定，則用信號通知對應該載入的該載入佇列輸入(例如透過追蹤器)，用其相關指令錯失預測/清除或除役。該存取遮罩也追蹤執行緒/核心。

[0153] 圖 22 為一非消歧義載入儲存佇列，事實上其

並不包含對應硬體來消除失序載入與儲存的歧義。載入與儲存依照允許的機器資源進行失序派遣。傳統上，該載入佇列與該儲存佇列內都使用位址匹配與對應消歧義硬體，以確保正確的儲存佇列輸入已經轉送至該要求的載入佇列輸入，如上面所述(例如圖 5 和圖 6)。外部核心/執行緒無法發現該載入佇列與該儲存佇列的內容。

[0154] 在圖 22 中，已派遣的載入與儲存位址並不會相對於該儲存佇列或該載入佇列內的輸入消除歧義。此時該等載入/儲存佇列為精簡的緩衝區實施，含減少的晶粒面積、功率消耗以及時機需求。該 SRB 將執行該消歧義功能。隨著在該 SRB 內偵測到位址匹配，這些匹配用於將該儲存內的輸入傳遞至載入轉送預測表，以隨指令序列往前執行來實施該轉送。

[0155] 隨著載入已經派遣，檢查該預測表來查看是否與對應的儲存配對。若該載入已經配對且特定儲存已經派遣，則該載入如該預測表內所記錄，將從該儲存佇列輸入編號轉送。若該儲存尚未派遣，則該載入將其載入佇列輸入編號暫存在該預測表內，且將在該載入佇列內標記本身，以等待轉送該儲存資料。當稍後派遣該儲存時，其檢查該預測表，以獲得該載入佇列輸入編號並轉送至該載入。

[0156] 一旦偵測到用於一載入儲存配對的轉送，則記錄該 PC 和該載入儲存配對的該位址，如此確認該位址匹配。若該位址匹配，則直到該儲存資料已經派遣且該載

入將標記來轉送之後，才會派遣該載入。該預測臨界用於設定載入儲存配對之間該轉送關係內的機密等級。

[0157] 圖 23 顯示根據本發明之一具體實施例的一免消歧義失序載入佇列重新排序實施之重新排序實施。圖 23 也產生非推測性的低功綠、低晶粒面積以及較少時機關鍵。

[0158] 儲存除役/重新排序緩衝區(SRB，store retirement/reorder buffer)可在兩種實施當中操作：除役實施以及重新排序實施。

[0159] 在圖 23 的重新排序實施當中，從該儲存佇列以失序方式(例如依照允許的資源)將儲存位址載入該 SRB。隨著分配每一儲存，其接收一序號。然後該 SRB 根據序號將儲存重新排序，如此其以原始程式順序存在於該 SRB 內。在程式順序內較早的儲存位於該 SRB 的頂端上。然後，後續載入找尋位址匹配以及分配時間歷程(在載入與儲存分配時所賦予的該程式順序序號)。隨著載入已經派遣，其找尋該 SRB，若找到尚未派遣(尚未有位址計算)的較早儲存(相較於其本身序號)，則可實施兩種解決方案之一。

1. 該載入並未派遣，會等待所有較早儲存都已經派遣，才會輪到派遣此載入。

2. 該載入派遣且將其位址標記在該快取的該存取遮罩內(如圖 17 內所示)。後續儲存檢查該存取遮罩，且遵照如圖 17 內所述相同的方法。

[0160] 請注意，如上述用優先權編碼器來找出正確的轉送輸入。

[0161] 圖 24 顯示根據本發明之一具體實施例的一指令序列(例如軌跡)重新排序推測性執行實施。在推測性模式中，儲存在儲存除役時以原始程式順序，從該儲存佇列移動至該 SRB，如此在原始程式順序當中較早的該儲存位於該 SRB 的頂端上。然後找尋後續載入的位址匹配(例如使用位址 CAM)，且從該 SRB/儲存快取內的匹配輸入轉送。在具有二或多個位址匹配的情況下，具有優先權的編碼器可利用掃描第一來找出正確轉送輸入。這允許該機器進行轉送。若一載入已經派遣(第一次檢查該 SRB)且轉送給它的該儲存已經除役至該 SRB/儲存快取，則該載入從該 SRB/儲存快取轉送，且將該配對關係記錄在該預測表。若要偵測一載入在轉送給它的該儲存已經除役至該 SRB/儲存快取之前已經派遣之情況，則該除役的載入將再一次檢查該儲存佇列。若該載入發現一轉送儲存匹配，則用信號通知對應該載入的該載入佇列輸入，用其相關指令進行錯失預測/清除或除役。該轉送預測器將從此錯失轉送當中學習。

[0162] 請注意，該載入將可針對先前儲存檢查該 SRB 的一匹配位址，因為直到包含所提及載入的該軌跡內所有指令都已經進入該軌跡提交(commit)狀態之前(例如全都變成非推測性，且全部軌跡都準備好提交)，SRB 內的所有儲存都不會提交至外部快取/儲存快取架構可見狀態

(讓記憶體可發現該 SRB 儲存裝置)。

[0163] 該儲存除役/重新排序緩衝區在功能上可進行推測性執行。推測性執行的結果可存在該儲存除役/重新排序緩衝區內，直到知道推測性結果為止。該推測性結果在架構上並不可見，一旦提交推測性狀態，儲存可提交至該儲存快取。在提交該狀態之前，任何例外或需要除役的載入與儲存都將發出例外或錯失預測，避免該狀態提交。儲存與對應載入之間的轉送錯失預測可以修復(例如利用讓錯失時清除機器的措施預測=轉送載入點等等)。

[0164] 由 Mohammad Abdallah 於 2012 年 1 月 27 日提出，名稱為「HARDWARE ACCELERATION COMPONENTS FOR TRANSLATING GUEST INSTRUCTIONS TO NATIVE INSTRUCTIONS」的第 13/360,024 號美國專利申請案內可找到 SRB 功能的額外說明。

[0165] 圖 25 顯示根據本發明之一具體實施例的示範微處理器管線 2500 之圖式。微處理器管線 2500 包括一擷取模組 2501，用於實施識別並擷取包括執行的指令之處理功能，如上述。在圖 25 的具體實施例中，該擷取模組接著一解碼模組 2502、一分配模組 2503、一派遣模組 2504、一執行模組 2505 以及一除役模組 2506。請注意，微處理器管線 2500 只是實施上述本發明具體實施例功能性的一範例，熟此技藝者應該了解，可實施包括上述解碼模組功能性的其他微處理器管線。

[0166] 為了解釋，上面參照特定具體實施例的說明並非用於排他或用於限制本發明。許多修改與變化都可和上述一致。具體實施例經過選擇與說明來最佳闡述本發明原理，且以許多具體實施例讓其他精通此技術的人士對本系統有最佳瞭解，這些具體實施例都適合特定使用期待。

【符號說明】

[0167]

400：處理

410：分配指標器

800：處理

900：處理

1200：互連

1201：互連

1211：片段

1202：位址陣列

1206：標籤陣列

1207：資料陣列

2101-2105：線

2500：微處理器管線

2501：擷取模組

2502：解碼模組

2503：分配模組

2504：派遣模組

2505：執行模組

2506：除役模組

I637320

發明摘要

※申請案號：106106344

※申請日：102 年 06 月 14 日　　※IPC 分類：*G06F 9/38* (2006.01)
G06F 9/46 (2006.01)

【發明名稱】(中文/英文)

根據儲存資歷實施來自不同執行緒轉送的執行緒不可知之載入儲存緩衝區

A load store buffer agnostic to threads implementing forwarding from different threads based on store seniority

【中文】

本發明揭示一種在一處理器中使用共享記憶體資源用於記憶體一致性模型內失序載入之一執行緒未知統一儲存佇列與一統一載入佇列方法。該方法包含：實施一記憶體資源，其可由複數個非同步核心存取，其中該等複數個核心共享一統一儲存佇列及一統一載入佇列；及實施一存取遮罩，其作用在於追蹤一快取線的哪個字元透過一載入來存取，其中該快取線包含該記憶體資源，其中存取該快取線的一字元時，該載入設定該存取遮罩內的一遮罩位元，且其中該遮罩位元阻擋來自複數個核心的其他載入之存取。該方法更包含：在執行從該等複數個核心至該快取線的後續儲存時檢查該存取遮罩，其中可將來自不同執行緒的儲存轉送至不同執行緒的載入，同時依序維持記憶體一致性語意；及當該部分快取線的一後續儲存發現來自該存取遮罩內一載入的一先前遮罩時，導致一錯失預測，其中該後續儲存使用一追蹤暫存器以及一執行緒 ID 暫存器，發信號通知對應該載入的一載入佇列輸入。

【英文】

In a processor, a thread agnostic unified store queue and a unified load queue method for out of order loads in a memory consistency model using shared memory resources. The method includes implementing a memory resource that can be accessed by a plurality of asynchronous cores, wherein the plurality of cores share a unified store queue and a unified load queue; and implementing an access mask that functions by tracking which words of a cache line are accessed via a load, wherein the cache line includes the memory resource, wherein the load sets a mask bit within the access mask when accessing a word of the cache line, and wherein the mask bit blocks accesses from other loads from a plurality of cores. The method further includes checking the access mask upon execution of subsequent stores from the plurality of cores to the cache line, wherein stores from different threads can forward to loads of different threads while still maintaining in order memory consistency semantics; and causing a miss prediction when a subsequent store to the portion of the cache line sees a prior mark from a load in the access mask, wherein the subsequent store will signal a load queue entry corresponding to that load by using a tracker register and a thread ID register.

申請專利範圍

1. 一種在處理器中使用共享記憶體資源對記憶體一致性模型內的失序載入作執行緒不可知統一儲存佇列與統一載入佇列的方法，包含：

實施一記憶體資源，其可由複數非同步核心存取，其中該等複數核心共享該統一儲存佇列及統一載入佇列；及

以存取遮罩追蹤，一快取線中的哪些字元被一載入所存取，其中該快取線包含該記憶體資源，其中當該載入存取該快取線的一字元時，在該存取遮罩內的一遮罩位元被設定，及其中該遮罩位元阻擋來自複數非同步核心的其他載入對該快取線的該字元的存取。

2. 如申請專利範圍第 1 項之方法，更包含：

在執行該等複數非同步核心之一對該快取線的該字元的後續儲存的同時，檢查該存取遮罩，其中可將來自不同執行緒的儲存轉送至不同執行緒的載入，同時仍維持依序記憶體一致性語意；及

當檢測到對應於為該失序載入所存取的該字元的該存取遮罩內的該遮罩位元被設定時，對該快取線的該字元的後續儲存造成一錯失預測。

3. 如申請專利範圍第 2 項之方法，其中該統一儲存佇列與統一載入佇列對可能存取它的該等複數執行緒之任一者為不可知的。

4. 如申請專利範圍第 2 項之方法，其中該統一儲存佇列與統一載入佇列對可能存取它的該等複數核心之任一

者為不可知的。

5. 如申請專利範圍第 2 項之方法，其中執行緒可根據儲存資歷從彼此轉送。

6. 如申請專利範圍第 5 項之方法，其中對於一給定執行緒，若並無來自該給定執行緒的轉送，則一載入可以接收來自不同執行緒的資深儲存的轉送。

7. 如申請專利範圍第 5 項之方法，其中在該相同執行緒內一給定儲存之前的所有載入與儲存都已經執行時，則該給定儲存為資深儲存。

8. 如申請專利範圍第 1 項之方法，其中該統一儲存佇列與統一載入佇列包含單一記憶體資料結構。

9. 如申請專利範圍第 1 項之方法，其中該記憶體資源可以為複數執行緒所存取。

10. 如申請專利範圍第 1 項之方法，其中該存取遮罩包含一組位元及在該組位元中的各個位元對應於在該快取線中的分開字元，以及，該快取線包含多重字元。

11. 如申請專利範圍第 10 項之方法，其中由快取線的一字元讀取的載入設定對應於該快取線的該字元的個別存取遮罩位元。

12. 如申請專利範圍第 11 項之方法，其中當該載入除役時，該個別存取遮罩位元被清除。

13. 如申請專利範圍第 1 項之方法，其中實施一載入佇列參考暫存器來追蹤載入佇列輸入參考，使得當一儲存將資料儲存到該快取線的對應該載入佇列輸入參考暫存器

內的匹配的字元時，導致該對應載入佇列輸入進行錯失預測。

14. 如申請專利範圍第 2 項之方法，其中該後續儲存將使用一追蹤暫存器，發信號通知對應該載入的一載入佇列輸入，且導致載入搭配載入相依指令進行錯失預測。

15. 一種微處理器，包含：

複數非同步核心與一載入儲存緩衝器，其中該載入儲存緩衝器實施使用共享記憶體資源在記憶體一致性模型內的失序載入之方法，透過以下步驟：

實施記憶體資源，其可由該等複數非同步核心存取，其中該等複數非同步核心共享統一儲存佇列與統一載入佇列；

以存取遮罩來追蹤，快取線的哪些字元被載入所存取，其中該快取線包含該記憶體資源，其中當該載入存取該快取線的一字元時，該存取遮罩內的一遮罩位元被設定，及其中該遮罩位元阻擋來自複數非同步核心的其他載入對該快取線的該字元的存取。

16. 如申請專利範圍第 15 項之微處理器，其中該存取遮罩包含一組位元及在該組位元中的各個位元對應於在該快取線中的分開字元，以及，該快取線包含多重字元。

17. 如申請專利範圍第 16 項之微處理器，其中由該快取線的一字元讀取的載入設定對應於該快取線的該字元的個別存取遮罩位元。

18. 如申請專利範圍第 17 項之微處理器，其中當該

載入除役時，該個別存取遮罩位元被清除。

19. 一種儲存指令的非暫態電腦可讀取儲存媒體，當所述指令為計算裝置的一組一或更多處理器所執行時，使得該計算裝置用以：

實施記憶體資源，其可由複數非同步核心存取，其中該等複數非同步核心共享統一儲存佇列與統一載入佇列；及

以存取遮罩來追蹤，快取線的哪些字元被載入所存取，其中該快取線包含該記憶體資源，其中當該載入存取該快取線的一字元時，該存取遮罩內的一遮罩位元被設定，及其中該遮罩位元阻擋來自複數非同步核心的其他載入對該快取線的該字元的存取。

20. 如申請專利範圍第 19 項之非暫態電腦可讀取儲存媒體，其中該存取遮罩包含一組位元及在該組位元中的各個位元對應於在該快取線中的分開字元，以及，該快取線包含多重字元。

【代表圖】

【本案指定代表圖】：第(20)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