I533129

(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 I533129 B**

(45)公告日： 中華民國 105 (2016) 年 05 月 11 日

(21)申請案號：101110082 (22)申請日： 中華民國 101 (2012) 年 03 月 23 日

(51)Int. Cl.： *G06F12/08 (2006.01)* *G06F15/76 (2006.01)*
G06F9/06 (2006.01) *G06F9/30 (2006.01)*

(30)優先權：2011/03/25 美國 61/467,944

(71)申請人：軟體機器公司 (美國) SOFT MACHINES, INC. (US)
美國

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

(74)代理人：李宗德

(56)參考文獻：

US 6216215B1 US 2009/0150647A1

審查人員：張發祥

申請專利範圍項數：33 項 圖式數：23 共 71 頁

(54)名稱

使用可分割引擎實體化的虛擬核心執行指令序列程式碼區塊

EXECUTING INSTRUCTION SEQUENCE CODE BLOCKS BY USING VIRTUAL CORES INSTANTIATED BY PARTITIONABLE ENGINES

(57)摘要

一種用於使用一處理器之複數個虛擬核心來執行指令的方法。該方法包括使用一通用前端排程器接收一輸入的指令序列，並將該輸入的指令序列分割成複數個指令的程式碼區塊。該方法另包括產生複數個繼承向量，其描述該等程式碼區塊的指令之間的相關性，且將該等程式碼區塊分配至該處理器的複數個虛擬程式碼，其中每一虛擬核心包含複數個可分割引擎之資源的個別子集合。該等程式碼區塊使用該等可分割引擎根據一虛擬核心模式及根據該等個別繼承向量來執行。

A method for executing instructions using a plurality of virtual cores tor a processor. The method includes receiving an incoming instruction sequence using a global front end scheduler, and partitioning the incoming instruction sequence into a plurality of code blocks of instructions. The method further includes generating a plurality of inheritance vectors describing interdependencies between instructions of the code blocks, and allocating the code blocks to a plurality of virtual cores of the processor, wherein each virtual core comprises a respective subset of resources of a plurality of partitionable engines. The code blocks are executed by using the partitionable engines in accordance with a virtual core mode and in accordance with the respective inheritance vectors.

指定代表圖：

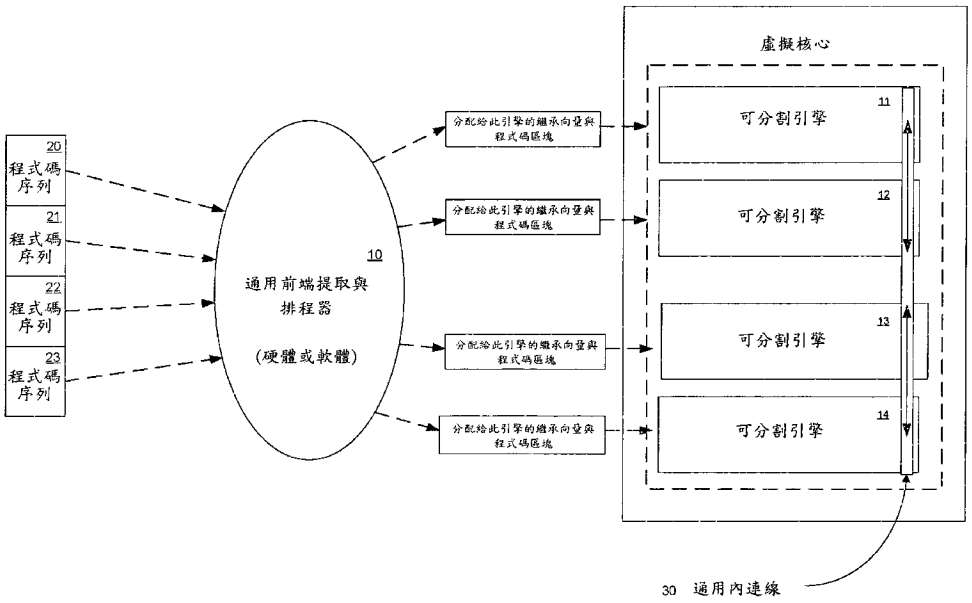

圖 1A

符號簡單說明：

10・・・通用前端提取與排程器

11-14・・・可分割引擎

20-23・・・程式碼序列

六、發明說明：

本申請案主張由 Mohammad A. Abdallah 於 2011 年 3 月 25 日立案之共同申請共同受讓的美國臨時專利申請案編號 61/467,944 之優先權，其名為「使用可分割引擎實體化的虛擬核心執行指令序列程式碼區塊」(EXECUTING INSTRUCTION SEQUENCE CODE BLOCKS BY USING VIRTUAL CORES INSTANTIATED BY PARTITIONABLE ENGINES)，在此完整引述併入。

【相關申請案之對照】

本申請案關於由 Mohammad A. Abdallah 於 2007 年 4 月 12 日立案之共同申請共同受讓的美國專利申請案編號 2009/0113170，其名為「處理在關連作業中平行指定之指令矩陣的裝置與方法」(APPARATUS AND METHOD FOR PROCESSING AN INSTRUCTION MATRIX SPECIFYING PARALLEL IN DEPENDENT OPERATIONS)，在此完整引述併入。

本申請案關於由 Mohammad A. Abdallah 於 2007 年 11 月 14 日立案之共同申請共同受讓的美國專利申請案編號 2010/0161948，其名為「在支援多種內容切換模式與虛擬化方案之多執行緒架構中處理複雜指令格式的裝置與方法」(APPARATUS AND METHOD FOR PROCESSING COMPLEX INSTRUCTION FORMATS IN A MULTITHREADED ARCHITECTURE SUPPORTING VARIOUS CONTEXT SWITCH MODES AND

VIRTUALIZATION SCHEMES)，在此完整引述併入。

【發明所屬之技術領域】

本發明概略關於數位電腦系統，尤指一種用於選擇包含一指令序列之指令的系統與方法。

【先前技術】

要處理相關或完全無關之多項工作即需要處理器。這種處理器之內部狀態通常由暫存器所構成，其可在每一次程式執行的特定瞬間保存不同的數值。在每一次程式執行的瞬間，該內部狀態圖像被稱為該處理器的架構狀態。

當程式碼執行被切換來運作另一項功能時(例如另一執行緒、程序或程式)，則該機器/處理器之狀態必須被儲存，使得該項新的功能可利用該等內部暫存器來建構其新的狀態。一旦該項新功能終止時，則其狀態即被丟棄，而該先前內容的狀態將被恢復，並重新開始執行。這種切換程序被稱為一內容切換，且通常包括 10 餘次或數百次循環，特別是利用到採用大量暫存器(例如 64、128、256)的現今架構及/或無順序執行時。

在感知執行緒(thread-aware)的硬體架構中，通常該硬體可對於有限數目的硬體支援執行緒來支援多種內容狀態。在此例中，該硬體對於每一支援的執行緒複製所有架構狀態元件。此即可在當執行一新執行緒時無需內容切換。但是，如此仍具有多項缺點，即對於在硬體中支援的

每一額外的執行緒要複製所有架構狀態元件(即暫存器)之區域、電力與複雜性。此外，如果軟體執行緒的數目超過明確支援之硬體執行緒的數目，則仍必須執行該內容切換。

此即為需要大量執行緒的一微細粒度基礎(fine granularity basis)之下所常見需要的平行化(parallelism)。具有複製內容狀態硬體儲存之硬體執行緒感知架構無助於非執行緒的軟體程式碼，且僅能對於被執行緒化的軟體減少其內容切換的次數。但是，那些執行緒通常針對粗粒度平行化所建構，並在初始化與同步化、離開微細粒度平行化(例如函數呼叫與迴路平行執行)時造成沉重的軟體負擔，而無法有效率的執行緒初始化/自動產生。這些描述的負擔在使用針對非明確地/簡易地平行化/執行緒的軟體程式碼之現今的編譯器或使用者平行化技術時，伴隨著這些程式碼之自動平行化的困難性。

【發明內容】

在一具體實施例中，本發明實施成一種針對一處理器使用複數虛擬程式碼執行指令之方法。該方法包括使用一通用前端排程器接收一輸入的指令序列，並將該輸入的指令序列分割成複數個指令的程式碼區塊。該方法另包括產生複數個繼承向量，其描述該等程式碼區塊的指令之間的相關性，且將該等程式碼區塊分配至該處理器的複數個虛擬程式碼，其中每一虛擬核心包含複數個可分割引擎之資

源的個別子集合。該等程式碼區塊使用該等可分割引擎根據一虛擬核心模式及根據該等個別繼承向量來執行。

本發明之其它具體實施例利用一共用排程器、一共用暫存器檔案及一共用記憶體子系統來針對處理器之多個可分割引擎實施片段化的位址空間。該等可分割引擎可用於實施複數個虛擬核心。片段化藉由允許額外的虛擬核心來共同合作執行指令序列而能夠調整微處理器的效能。該片段化階層在每一快取階層皆相同(例如 L1 快取、L2 快取及該共用暫存器檔案)。該片段化階層可使用位址位元將該位址空間區分成片段，其中使用該等位址位元使得該等片段在快取線邊界之上及頁面邊界之下。每一片段可設置成利用一多埠式記憶庫結構做儲存。

前述內容為一總結，因此包含必要的簡化、一般化與細節之省略；因此，本技術專業人士將可瞭解到該總結僅為例示性，而並非為任何型式的限制。僅由該等申請專利範圍所定義之本發明的其它態樣、創新特徵與好處將可在以下提出的非限制性詳細說明中做瞭解。

【實施方式】

雖然本發明已經配合一具體實施例做說明，本發明並非要被限制於此處所提出的該等特定型式。相反地，本發明係要涵蓋其它這些選項、修正及同等者，其係被合理地包括在由該等附屬申請專利範圍所定義之本發明的範圍之內。

在以下的詳細說明中，已提出許多特定細節，例如特定方法順序、結構、元件與連接。但是應瞭解到這些與其它特定細節不需要被用來實施本發明之具體實施例。在其它狀況下，熟知的結構、元件或連接已被省略或並未特別詳細說明係為了避免不必要地混淆本說明。

在本說明書中的「一(one、an)具體實施例」係要代表配合該具體實施例所描述的一特定特徵、結構或特性被包括在本發明的至少一具體實施例中。在本說明書中多處有用語「在一具體實施例中」的出現並非一定都參照到相同的具體實施例，也非為與其它具體實施例相互排斥的獨立或其它的具體實施例。再者，所述之多種特徵可由一些具體實施例呈現而不出現於其它具體實施例。同樣地，所述之多種需求可為一些具體實施例的需求而非其它具體實施例所需要。

該等詳細說明的一些部份在以下係以程序、步驟、邏輯方塊、處理，以及其它對於一電腦記憶體內資料位元之作業的符號表示來呈現。這些說明及表示為在資料處理技術中那些專業人士所使用來最佳地傳遞他們工作的實質內容到本技藝中其他專業人士的手段。概言之，在此處的程序、電腦可執行步驟、邏輯方塊及程序等，其應視為可達到所想要結果之步驟或指令的一自我符合的順序。該等步驟為那些需要實體量的實體操縱。通常但非必要，這些數量可採取一電腦可讀取儲存媒體的電子或磁性信號之

型式，並能夠在一電腦系統中被儲存、轉換、組合、比較，及另可進行操縱。主要為了共通用法的原因，較為方便地是將這些信號稱為位元、數值、元件、符號、字元、項目、數目或類似者。

但是應要注意到所有這些及類似術語係要關聯於該等適當實體數量，並僅為應用到這些數量的便利標記。除非在以下討論中可瞭解者之外有特定地陳述，將可瞭解到在整個本發明討論中所利用的術語，例如「處理」或「存取」或「寫入」或「儲存」或「重製」或類似者，皆代表一電腦系統或類似的電子運算裝置之動作及程序，其可操縱及轉換表示成該電腦系統之暫存器及記憶體及其它電腦可讀取媒體中的實體(電子)數量的資料成為類似地表示成在該電腦系統記憶體、或暫存器、或其它像是資訊儲存、傳輸或顯示裝置內的實體數量之其它資料。

本發明之具體實施例利用一共用通用前端排程器、複數個分段的暫存器檔案及一記憶體子系統來針對一多核心處理器之多個核心實施片段化的位址空間。在一具體實施例中，片段化可藉由允許額外的虛擬核心(例如軟核心)來共同合作地執行包含一或多個執行緒的指令序列而能夠調整微處理器之效能。該片段化階層對於每一快取階層皆相同(例如 L1 快取、L2 快取及該共用暫存器檔案)。該片段化階層可使用位址位元將該位址空間區分成片段，其中使用該等位址位元使得該等片段由在快取線邊界之上及頁面邊界之下的位元做辨識。每一片段設置成利用一多

埠式記憶庫結構做儲存。本發明之具體實施例在以下由圖1A與圖1B做進一步說明。

圖1A所示為根據本發明一具體實施例之一種處理器的概述圖。如圖1A所示，該處理器包括一通用前端提取與排程器10及複數個可分割引擎11-14。

圖1A所示為該通用前端產生程式碼區塊與繼承向量來在它們個別的可分割引擎上支援程式碼序列之執行的方式概述。每一程式碼序列20-23根據該特定虛擬程式碼執行模式而屬於相同的邏輯核心/執行緒或屬於不同的邏輯核心/執行緒。該通用前端提取與排程器將處理程式碼序列20-23來產生程式碼區塊與繼承向量。這些程式碼區塊與繼承向量被分配給特定可分割引擎11-14，如所示。

該等可分割引擎根據一種選擇的模式實施虛擬核心。一可分割引擎包括一節段、一片段與一些執行單元。在該等可分割引擎之內的該等資源可用於實施具有多種模式的虛擬核心。如該虛擬核心模式所提供者，可實施一軟核心或許多軟核心來支援一邏輯核心/執行緒。在圖1A的具體實施例中，根據該選擇的模式，該等虛擬核心可支援一邏輯核心/執行緒或四邏輯核心/執行緒。在該等虛擬核心支援四邏輯核心/執行緒的一具體實施例中，每一虛擬核心的該等資源被分散橫跨每一可分割引擎。在該等虛擬核心支援一邏輯核心/執行緒的一具體實施例中，所有該等引擎的該等資源係專屬於該核心/執行緒。該等引擎被分

割，使得每一引擎提供包含每一虛擬核心的該等資源之一子集合。換言之，一虛擬核心將包含該等引擎 11-14 之每一者的該等資源之一子集合。該等引擎 11-14 之每一者的該等資源之間的通訊由一通用內連線結構 30 所提供，藉以實施此程序。另外，引擎 11-14 可用於實施一實體模式，其中引擎 11-14 之該等資源係專屬於支援一專屬的核心/執行緒之執行。依此方式，由該等引擎實施的該等軟核心包含具有分散橫跨該等引擎之每一者的資源之虛擬核心。該等虛擬核心執行模式另於下述在後續的圖示中做進一步說明。

必須注意到在一種習用的核心實施中，僅有一核心/引擎內的資源僅被分配至一邏輯執行緒/核心。相反地，在本發明之具體實施例中，任何引擎/核心之該等資源可被分割成與其它引擎/核心分割共同地實體化被分配至一邏輯執行緒/核心的一虛擬核心。此外，本發明之具體實施例可實施多種虛擬執行模式，其中那些相同引擎可被分割成支援許多專屬的核心/執行緒、許多動態分配的核心/執行緒、或是所有引擎之所有該等資源支援一單一核心/執行緒之執行的一具體實施例。這些具體實施例在下述之說明中做進一步說明。

圖 1B 所示為根據本發明一具體實施例中針對一多核心處理器之可分割引擎及它們的組件之概要圖，其中包括分段的排程器與暫存器檔案、通用內連線與一片段化記憶體子系統。如圖 1 所示，顯示有四個片段 101-104。該片

段化階層對於每一快取階層皆相同(例如 L1 快取、L2 快取及該負載儲存緩衝器)。資料可經由記憶體通用內連線 110a 於該等 L1 快取之每一者、該等 L2 快取之每一者、及該等負載儲存緩衝器之每一者之間交換。

該記憶體通用內連線包含一路由矩陣，其允許複數個核心(例如位址計算與執行單元 121-124)來存取可能儲存在該分段的快取階層(例如 L1 快取、負載儲存緩衝器與 L2 快取)中任何一點處的資料。圖 1 亦描述了片段 101-104 之每一者可由位址計算與執行單元 121-124 經由記憶體通用內連線 110a 存取之方式。

執行通用內連線 110b 類似地包含一路由矩陣，其可允許該等複數個核心(例如位址計算與執行單元 121-124)來存取可能儲存在該等分段的暫存器檔案之任何一處的資料。因此，該等核心可經由記憶體通用內連線 110a 或執行通用內連線 110b 存取儲存在該等片段之任何一者中的資料及儲存在該等節段之任何一者中的資料。此外，必須注意到在一具體實施例中，另一通用內連線存在於該等共用分割提取與排程器之每一者之間。此係由在每一共用分割提取與排程器之間連接的該等水平箭頭所示。

圖 1B 另顯示一通用前端提取與排程器 150，其為整個機器的視圖，且其管理該等暫存器檔案節段與該等片段化記憶體子系統之運用。位址產生包含片段定義之基礎。該通用前端提取與排程器藉由分配指令序列至每一節段

的分割排程器來運作。然後該共用分割排程器分派那些指令序列在位址計算與執行單元 121-124 上執行。

必須注意到在一具體實施例中，該等共用分割提取與排程器之功能性可被加入到通用前端排程器 150。在這種具體實施例中，該等節段未包括個別的共用分割提取與排程器，且它們之間將不需要一內連線。

此外，必須注意到圖 1A 所示的該等可分割引擎可用一階層方式巢化。在這種具體實施例中，一第一級可分割引擎將包括一局部前端提取與排程器，及與其連接的多個次級可分割引擎。

圖 2 所示為根據本發明一具體實施例之排程器流程圖。如圖 2 所示，顯示一桶緩衝器(bucket buffer)，其包括推測式執行緒桶指標、桶來源及目的地清單。該等排程器與執行桶包括一桶分派選擇器及該虛擬暫存器匹配與讀取，其包括一暫存器階層與一暫存器快取的可能性。該後端為已執行的桶被記錄及在沈除之前強制異常排序的地方。該暫存器階層/快取亦做為該等執行的桶結果之一中間儲存，直到它們為非推測性，並可更新該架構狀態。下述揭示一種該前端、該分派階段與已執行的桶被記錄處的該後端的可能實施。

圖 2 所示為由管理少量緊密耦合的執行緒之一桶緩衝器調整成管理多個桶緩衝器與執行緒之硬體電路的觀念

該運算子直接由該相對應的執行緒暫存器節段讀取。如果該暫存器檔案為一虛擬暫存器，包括使用標籤的一實體上分段的暫存器檔案，則一標籤匹配必須做為該虛擬暫存器讀取的一部份來完成。如果該標籤匹配，則該讀取由該分段的暫存器檔案發生。

所揭示者為可支援軟體執行緒、硬體產生的執行緒、VLIW 執行、SIMD & MIMD 執行以及無順序超純量執行之模擬的暫存器架構。雖然其實體上為分段，但可看作一統一的架構資源。此分段的暫存器為該虛擬暫存器檔案的一部份，其可包括一暫存器階層與一暫存器快取，以及儲存與檢查暫存器標籤的機制。如果我們使用一位置為主的方案來利用該相關性繼承向量，則可排除該標籤存取。該方案之運作使得當該執行的桶編號於分派階段期間被廣播時，後續指令的所有該等來源執行一 CAM(內容可定址匹配，Content addressable match)，其比較它們的來源桶與該剛被分派/執行的桶來設定該來源的該預備旗標。此處該桶被執行之實際位置亦可連同該暫存器編號來傳遞，使得可解決任何的混淆。

例如，考慮一種具有四個暫存器檔案節段的實施，其每一者包含 16 個暫存器。例如，在分派一桶#x 至節段 2 時，該桶編號 x 被廣播至該桶緩衝器，該節段#2 亦隨其被廣播，使得與桶 x 有相關性的所有來源將記錄其寫入所有其暫存器在節段 2 中。當來到分派那些指令時，它們知道它們需要由節段 2 而非任何其它節段讀取它們的暫存器，

即使相同的暫存器編號存在於該等其它節段中。此亦應用至該暫存器快取來避免使用標籤。我們可延伸此觀念至該通用前端，其中除了該執行緒資訊之外，該繼承向量可指定寫入到此暫存器之該指令桶被分配在那一個引擎中。

圖5所示為根據本發明一具體實施例中橫跨許多虛擬核心之指令分配的另一種實施。圖5顯示一運行時間最佳化器排程器550，其藉由分佈繼承向量編碼節段至該等虛擬核心來運作。在一具體實施例中，該最佳化器察看指令的一些程式碼區塊，並重新排程指令橫跨所有該等程式碼區塊，以產生程式碼節段與繼承向量。該最佳化器的目標將是使得程式碼節段在它們個別的虛擬核心上重疊執行之執行效率可最大化。

圖6為根據本發明一具體實施例中具有相對應複數的暫存器檔案與運算子結果緩衝器之複數個暫存器節段。如圖6所示，該執行通用內連線可連接每一暫存器節段至複數個位址計算與執行單元。

圖6的該等暫存器節段可用於實施以下三種執行模式之一：由該編譯器/程式化器被群組在一起來形成一 MIMD 超指令矩陣，或是每一矩陣在一執行緒的模式中被獨立地執行，其中個別的執行緒在該等四個硬體節段之每一者之上同時地執行。可能的最後執行模式為有能力使用一硬體相關性檢查由一單一執行緒動態地執行四個不同的指令

矩陣，以確保在該等四個不同硬體節段上同時執行的那些不同矩陣之間不會存在有相關性。

圖 6 中的該等暫存器檔案另可根據該執行模式來設置。在一種模式中，該等暫存器當案係以用於四個節段的一 MIMD 寬度的一 MIMD 分段的暫存器檔案來看待，或是它們做為四個各自的暫存器檔案，其每一者用於一獨立執行緒。該等暫存器檔案亦支援一動態執行模式，其中該等四個節段為一統一的暫存器檔案，其中被寫入到一特定節段中任何暫存器的資料可由該等其它節段中所有單元存取。那些模式之間的切換可為無縫隙式，只要不同的執行模式可於各自的執行緒基線指令矩陣與 MIMD 超指令矩陣執行緒之間交替。

在一多執行緒執行模式中，執行一執行緒的每一暫存器檔案與其執行單元整體皆無關於其它暫存器檔案與它們的執行緒。此係類似於每一執行緒具有其本身的暫存器狀態。但是，可指定那些執行緒之間的相關性。屬於一執行緒的每一矩陣將在該執行緒的暫存器檔案之該執行單元中執行。如果在該硬體上僅有執行一執行緒或非執行緒的單一程式，則使用以下的方法來允許屬於該單一執行緒/程式的平行矩陣能夠存取被寫入到該等其它節段中該等暫存器當中的該等結果。其完成的方法為藉由允許任何矩陣寫入結果到該等四個暫存器檔案節段之任何一者當中，以在該等其它暫存器檔案節段中產生那些暫存器的複本。實際上，此係藉由延伸每一節段的該等寫入埠到該等

元內連線 1201 的片段。那些片段元件可藉由片段化與分佈其集中的資源到數個引擎當中來被建構在一核心/處理器之內，或者它們可由在一多核心/多處理器組態中的不同核心/處理器之元件來建構。那些片段 1211 之一在圖中顯示為片段編號 1；該等片段可調整成一大的數目(概略為圖中所示的 N 個片段)。

此機制亦用於那些引擎/核心/處理器之間該記憶體架構的一種同調性方案。此方案開始於來自在一片段/核心/處理器中該等位址計算單元之一者的一位址要求。例如，假設該位址由片段 1(1211)所要求。其可使用屬於其本身片段的位址暫存器及/或使用位址內連線匯流排 1200 橫跨其它片段的暫存器來得到並計算其位址。在計算該位址之後，其產生用於存取快取與記憶體之 32 位元位址或 64 位元位址的該參照位址。此位址通常被分段成一標籤欄位與一集合與線欄位。此特定片段/引擎/核心將儲存該位址到其負載儲存緩衝器及/或 L1 及/或 L2 位址陣列 1202 當中，同時其將藉由使用一壓縮技術產生該標籤的一壓縮版本(其比該位址之原始標籤欄位具有較少數目的位元)。

有更多不同的片段/引擎/核心/處理器將使用該集合欄位或該集合欄位的一子集合做為一索引來辨識該位址被維護在那一個片段/核心/處理器中。此藉由該位址集合欄位位元之該等片段的索引化可確保在一特定片段/核心/引擎中該位址之擁有權的排除性，即使對應於該位址之記憶體資料可存在於另一個或多個其它片段/引擎/核心/處

理器中。即使該等位址 CAM/標籤陣列 1202/1206 被顯示在要耦合於資料陣列 1207 的每一片段中，它們可能僅耦合在實際上放置或佈置的鄰近處，或甚至事實上兩者屬於一特定引擎/核心/處理器，但在被保持在該等位址陣列中的位址與在一片段內該等資料陣列中的資料之間並無關係。

圖 8 為根據本發明一具體實施例如何使用一位址的位元由位址產生來列舉片段的示意圖。在本具體實施例中，片段係由在頁面邊界之上及快取線邊界之下的該等位址位元所定義，如圖 8 所示。本發明較佳地是維持在該等頁面邊界之上來避免於該等虛擬位址轉譯成實體位址其間造成 TLB 遺漏。該程序保持在該快取線邊界之下，藉以具有完整的快取線來正確地配合在該硬體快取階層當中。例如，在利用 64 位元組快取線的系統中，該片段邊界將避免使用最後六個位址位元。相較於利用 32 位元組快取線的一種系統，該片段邊界將避免使用最後五個位元。一旦定義之後，該片段階層在橫跨該處理器的所有快取階層當中皆相同。

圖 9 為本發明之具體實施例如何處理負載與儲存之示意圖。如圖 9 所示，每一片段係關聯於其負載儲存緩衝器與儲存沈除緩衝器。對於任何給定的片段，指定關聯於該片段或另一片段的一位址範圍之負載與儲存將被傳送至該片段的負載儲存緩衝器做處理。必須注意到它們將未依順序到達，因為該等核心並無順序地執行指令。在每一核

心之內，該核心不僅可存取到其本身的暫存器檔案，亦可存取到每一個其它核心之暫存器檔案。

本發明之具體實施例實施一分散式負載儲存排序系統。該系統被分佈橫跨多個片段。在一片段之內，局部資料相關性檢查由該片段執行。此係因為該片段僅載入與儲存在該特定片段的該儲存汰除緩衝器之內。此限制了必須察看其它片段來維持資料同調性的需求。依此方式，自一片段內的資料相關性被局部地強制。

關於資料一致性，該儲存分派閘極根據嚴格的程式內順序記憶體一致性規則來強制儲存汰除。儲存為無順序地抵達該負載儲存緩衝器。負載亦為無順序地抵達該負載儲存緩衝器。同時，該等無順序的負載與儲存被轉送至該等儲存汰除緩衝器做處理。必須注意到雖然儲存在一給定片段內依順序汰除，因為它們進入該儲存分派閘極，它們可無順序地來自該等多個片段。該儲存分派閘極強制實施一政策，其可確保即使儲存可無順序地存在於橫跨儲存汰除緩衝器，且即使該等緩衝器可相對於其它緩衝器的儲存為無順序地轉送儲存至該儲存分派閘極，該分派閘極可確保它們被嚴格地依順序轉送至片段記憶體。此係因為該儲存分派閘極具有儲存汰除的一整體概觀，並僅允許儲存依順序橫跨所有該等片段(例如通用地)離開至該記憶體之通用可見側。依此方式，該儲存分派閘極係做為一通用觀察者來確保該等儲存最終橫跨所有片段依序地返回到記憶體。

圖 10 為根據本發明一具體實施例中那些片段可被分成兩個或更多區域之方法。圖 10 所示為一單一片段可被分成多個區域之方法。區域區分可經由該位址產生程序來實施。區域區分改變了負載儲存檢查必須在一片段內完成的方式，因為在此例中相對於橫跨該整個片段，它們僅必須針對每個區域來完成。區域區分亦有好處在於其可使得單一埠的記憶體之行為可像是多埠記憶體，其中該單一埠係對不同的區域來存取。

圖 11 為根據本發明一具體實施例中該處理器之一種作業模式，其中該等可分割引擎之該等硬體資源係用於做為類似在執行應用程式中的邏輯核心。在此具體實施例中，該等虛擬核心之該等引擎的該等硬體資源被設置成實體核心。在圖 11 的模式中，其每一實體核心被設置成做為一邏輯核心。多執行緒應用程式與多執行緒功能性係根據該應用程式的軟體之執行緒化的可程式性。

圖 12 為根據本發明一具體實施例中該處理器之一種作業模式，其中軟核心被用於像是在執行應用程式時的邏輯核心來運作。在此具體實施例中，虛擬核心的該等可分割引擎將支援複數個軟核心。在圖 12 的模式中，每一軟核心被設置成做為一邏輯核心。多執行緒應用程式與多執行緒功能性係根據該應用程式的軟體之執行緒化的可程式性。

圖 13 為根據本發明一具體實施例中該處理器之一種作業模式，其中該等軟核心被用於像是在執行應用程式時的一單一邏輯核心來運作。在圖 13 的模式中，每一軟核心被設置成做為一單一邏輯核心。在這種實施中，一單一執行緒的應用程式將其指令序列分開，並分配在該等虛擬核心之間，其中它們被協同地執行來達成高單一執行緒效能。依此方式，單一執行緒的效能可隨著加入額外的軟核心來調整。

在選擇該處理器之操作模式時可使用一些策略。對於具有大量引擎(例如 8 引擎、12 引擎等)之一處理器，一些軟核心可設置成做為一單一邏輯核心，而該等其餘的核心可在該等其它模式中運作。此屬性可允許一種資源的智慧型分割來確保該硬體之最大利用率及/或最低浪費的電力消耗。例如，在一具體實施例中，核心(例如軟或邏輯核心)可根據正在執行的應用程式之種類來以每個執行緒為基礎做分配。

圖 14 為根據本發明一具體實施例中用於支援邏輯核心與虛擬核心功能之片段分段的示例性實施。如上所述，該片段分段化可允許該處理器設置成支援不同的虛擬核心執行模式，如上所述。

該通用內連線允許核心的執行緒來存取任何的埠 1401。必須注意到此處所使用的術語「執行緒」(thread)

代表來自不同邏輯核心的指令序列、來自相同邏輯核心的指令序列，或是兩者之某種混合。

該等執行緒利用埠 1401 之一來存取該負載儲存緩衝器之方式可根據該等仲裁器之政策而調整，如所示。因此，使用埠 1401 中任何一者的一執行緒經由埠 1402 可較大量或較少量地存取該負載儲存緩衝器。該分配的大小與該分配被管理的方式由該仲裁器控制。該仲裁器可動態地根據一特定執行緒的需求而分配存取該等埠。

該負載儲存緩衝器設置成具有散佈橫跨該等埠之複數個入口。至該負載儲存緩衝器之存取由該仲裁器控制。依此方式，該仲裁器可動態地分配在該負載儲存緩衝器中的入口至該等不同的執行緒。

圖 14 亦顯示了在負載儲存緩衝器與該 L1 快取之間的該等埠之上的仲裁器。因此，利用上述之該負載儲存緩衝器，使用該等埠 1403 中任何一者的一執行緒經由埠 1404 可較大量或較少量地存取該 L1 快取。該分配的大小與該分配被管理的方式由該仲裁器控制。該仲裁器可動態地根據一特定執行緒的需求而分配存取該等埠。

該 L1 快取設置成具有散佈橫跨該等埠之複數條路線。該 L1 快取之存取由該仲裁器控制。依此方式，該仲裁器可動態地分配在該 L1 快取中的入口至該等不同的執行緒。

在一具體實施例中，該等仲裁器設置成與用於追蹤功能性的複數個計數器 1460 及提供一限制功能之複數個臨界值限制暫存器 1450 運作。該限制功能指定一給定執行緒之最高資源分配百分比。該追蹤功能追蹤在任何給定時間時分配給一給定執行緒的該等實際資源。這些追蹤與限制功能影響了該負載儲存緩衝器、L1 快取、L2 快取或該等通用內連線之每一執行緒入口、路線或埠之數目的分配。例如，分配給每一執行緒之該負載儲存緩衝器中入口的總數可對於一可變臨界值做動態地檢查。此可變臨界值可根據一給定的執行緒之轉送進度來更新。例如，在一具體實施例中，減慢的執行緒(例如大數目或 L2 遺失等)以造成緩慢轉送進度來量化，因此它們個別的資源分配臨界值被降低，其包括該等入口臨界值、該等路線臨界值與該等埠臨界值。

圖 14 亦顯示出一共享的 L2 快取。在本具體實施例中，該共享的 L2 快取具有一固定埠配置，而在來自該 L1 快取的存取之間沒有任何的仲裁。在該處理器上執行的執行緒皆共享存取該 L2 快取及該 L2 快取的該等資源。

圖 15 為根據本發明一具體實施例中實施一多實體對多邏輯模式之一示例性四片段處理器的一片段記憶體。

一示例性邏輯核心與其與該處理器之該等資源的關係由圖 15 之陰影部所示。在圖 11 的操作模式中，該多實

體核心對多邏輯核心模式，其中該等實體核心用於像是在執行應用程式中的邏輯核心來運作，每一邏輯核心將設置成具有該負載儲存緩衝器與該 L1 快取之該等資源的一固定比例。該等埠可被特定地指定至每一執行緒或核心。在該負載儲存緩衝器中的入口可被特定地對每一執行緒或核心來保留。在該 L1 快取內的路線可被特定地對每一執行緒或核心來保留。多執行緒應用程式與多執行緒功能性係根據該應用程式的軟體之執行緒化的可程式性。此顯示成一個邏輯核心具有該等片段之每一者的該儲存緩衝器與該 L1 快取的一分配的埠與一分配的部份。依此方式，該邏輯核心包含每一片段的該等資源之一固定分配的片層。

在一具體實施例中，在該多實體核心對多邏輯核心模式中，該等四個片段可根據存取每一片段的埠之數目(例如埠 1401)來分割。例如，在每一片段具有六個埠的一具體實施例中，每一片段的該等資源，及每一分割的該等資源將引擎者，即可以這種方式區分來支援橫跨該等四個片段的六個實體核心與該等四個分割雙引擎。每一分割可被分配其本身的埠。同樣地，該負載儲存緩衝器與該 L1 快取的該等資源將以這種方式分配來支援六個實體核心。例如，在該負載儲存緩衝器具有 48 個入口的一具體實施例中，該等 48 個入口可被分配成使得每一實體核心有 12 個入口來支援實施有四個實體核心的一種模式，或是它們可分配成使得在實施有六個實體核心的狀況中每一實體核心有八個入口。

圖 16 為根據本發明另一具體實施例中實施一多實體對多邏輯模式之一示例性四片段處理器的一片段記憶體。

配合圖 15，一個示例性邏輯核心與其與該處理器之該等資源的關係如圖 16 的陰影部所示。在圖 11 的操作模式中，該多實體核心對多邏輯核心模式，一整個分割表引擎係專屬於支援一單一邏輯核心的執行。此係由圖 16 中的陰影部所顯示。該實體資源引擎係用於類似在執行應用程式中的邏輯核心來運作。

圖 17 為根據本發明一具體實施例中實施一多軟體核心對多邏輯核心模式之一示例性四片段處理器的一片段記憶體。

一示例性邏輯核心與其與該處理器的該等資源之關係係如圖 17 的陰影部所示。在圖 12 的操作模式中，該多軟核心對多邏輯模式，其中虛擬核心用於類似在執行應用程式中的邏輯核心來運作，該負載儲存緩衝器的該等資源之分配的大小與該分配被管理的方式由該仲裁器控制。該仲裁器可動態地根據一特定執行緒或核心的需求而分配存取該等埠。同樣地，該 L1 快取的該等資源之分配的大小與該分配被管理的方式由該仲裁器控制。該仲裁器可動態地根據一特定執行緒或核心的需求而分配存取該等埠。因此，在任何給定實例中，該邏輯執行緒/核心(例如陰影部)可使用不同的仲裁器與不同的埠。

圖 19 為根據本發明一具體實施例中實施一多實體對多邏輯模式之一示例性四片段處理器之位址計算與執行單元、運算子/結果緩衝器、執行緒的暫存器檔案與共用分割排程器。

一示例性邏輯核心與其與該處理器之該等資源的關係如圖 19 之陰影部所示。在圖 11 的操作模式中，該多實體核心對多邏輯核心模式，其中該等實體核心用於像是在執行應用程式中的邏輯核心來運作，每一邏輯核心將設置成具有該等位址計算單元、運算子/結果緩衝器、執行緒的暫存器檔案與共用分割排程器之該等資源的一固定比例。多執行緒應用程式與多執行緒功能性係根據該應用程式的軟體之執行緒化的可程式性。此係顯示成一個邏輯核心具有一分配的位址計算與執行單元、一分配的執行緒暫存器檔案與一分配的共用分割排程器。依此方式，該邏輯核心包含一固定分配的節段。但是在一具體實施例中，在此操作模式下，該等位址計算與執行單元仍可被共享(例如代表該等位址計算與執行單元之每一者將不會被遮影)。

圖 20 為根據本發明一具體實施例中實施一多實體對多邏輯模式之一示例性四片段處理器之位址計算與執行單元、運算子/結果緩衝器、執行緒的暫存器檔案與共用分割排程器之另一種實施。

一示例性邏輯核心與其與該處理器之該等資源的關係如圖 20 之陰影部所示。但是在圖 20 的具體實施例中，一實體核心的該等資源被分散橫跨該等片段之每一者與該等可分割引擎之每一者。此係顯示為一個邏輯核心具有該等位址計算與執行單元之一分配的部份、該等執行緒的暫存器檔案之一分配的部份，及橫跨該等節段之每一者的共用分割排程器之一分配的部份。此外，圖 20 顯示出一個邏輯核心如何將被分配該等位址計算執行單元之每一者的該等資源之部份。依此方式，該邏輯核心包含該等節段之每一者的一固定分配的部份。

圖 21 為根據本發明一具體實施例中實施一多軟核心對多邏輯核心模式之一示例性四片段處理器之位址計算與執行單元、暫存器檔案與共用分割排程器。

一示例性邏輯核心與其與該處理器之該等資源的關係如圖 21 之陰影部所示。在圖 12 的操作模式中，該多軟核心對多邏輯核心模式，其中該等軟核心用於像是在執行應用程式中的邏輯核心來運作，每一邏輯核心將設置成具有對於該等位址計算單元之任何一者、該等運算子/結果緩衝器之一動態分配部份、執行緒的暫存器檔案與共用分割排程器之一共享的存取。多執行緒應用程式與多執行緒功能性係根據該應用程式的軟體之執行緒化的可程式性。

圖 22 為根據本發明一具體實施例中實施一多軟核心對一邏輯核心模式之一示例性四片段處理器之位址計算與執行單元、暫存器檔案與共用分割排程器。

一示例性邏輯核心與其與該處理器之該等資源的關係如圖 22 之陰影部所示。在圖 13 的操作模式中，該多軟核心對一邏輯核心模式，其中該等軟核心用於像是在執行應用程式中一單一邏輯核心來運作，每一邏輯核心將設置成具有對於所有該等位址計算單元、及所有該等運算子/結果緩衝器、執行緒的暫存器檔案與共用分割排程器之一共享的存取。在這種實施中，一單一執行緒的應用程式將其指令序列分開，並分配在該等虛擬核心之間，其中它們被協同地執行來達成高單一執行緒效能。依此方式，單一執行緒的效能可隨著加入額外的軟核心來調整。

圖 23 為根據本發明一具體實施例之一示例性微處理器管線 2300 的示意圖。微處理器管線 2300 包括一提取模組 2301，其實施該程序的功能來辨識與擷取包含一執行的該等指令，如上所述。在圖 23 的具體實施例中，該提取模組接著為一解碼模組 2302、一分配模組 2303、一分派模組 2304、一執行模組 2305 與一汰除模組 2306。必須注意到微處理器管線 2300 僅為可實施上述之本發明之具體實施例的功能之管線的一種示例。本技術專業人士將可瞭解到可實施其它的微處理器管線來包括上述之該解碼模組的功能。

為了解釋的目的，前述的說明已經參照特定具體實施例來說明。但是，以上之例示性討論並非窮盡式或限制本發明於所揭示之明確型式。在以上的教示之下可瞭解其有可能許多修改及變化。該等具體實施例係被選擇及描述來最佳地解釋本發明及其實際應用的原理，藉此使得本技術中其它專業人士可在多種具體實施例及多種修正中最佳地利用本發明，使其可適用於所考慮的特定用途。

【圖式簡單說明】

本發明藉由範例來例示，但並非限制，在附屬圖面的圖形中類似的參考編號代表類似的元件。

圖 1A 為該通用前端產生程式碼區塊與繼承向量來在它們個別的可分割引擎上支援程式碼序列之執行的方式概述。

圖 1B 為根據本發明一具體實施例中針對一多核心處理器之可分割引擎及它們的組件之概要圖，其中包括分段的排程器與暫存器檔案、通用內連線與一片段化記憶體子系統。

圖 2 為根據本發明一具體實施例之排程器流程圖。

圖 3 為根據本發明一具體實施例之示例性硬體電路圖，其中顯示有儲存運算子與結果的一分段的暫存器檔案並具有一內連線。

圖 4 為根據本發明一具體實施例之一通用前端提取及排程器之示意圖。

圖 5 為根據本發明一具體實施例中橫跨許多虛擬核心之指令分配的另一種實施。

圖6為根據本發明一具體實施例中具有相對應複數的暫存器檔案與運算子及結果緩衝器之複數個暫存器節段。

圖7為根據本發明一具體實施例之一多核心處理器之一片段化記憶體子系統之細部示意圖。

圖8為根據本發明一具體實施例如何使用一位址的位元由位址產生來列舉片段的示意圖。

圖9為本發明之具體實施例如何處理負載與儲存之示意圖。

圖 10 為根據本發明一具體實施例中那些片段可被分成兩個或更多區域之方法。

圖 11 為根據本發明一具體實施例中該處理器之一種作業模式，其中虛擬核心被設置成對應於在執行應用程式時邏輯核心之實體核心。

圖 12 為根據本發明一具體實施例中該處理器之一種作業模式，其中虛擬核心被設置成對應於在執行應用程式時邏輯核心之軟核心。

圖 13 為根據本發明一具體實施例中該處理器之一種作業模式，其中該等虛擬核心被設置成對應於在執行應用程式時一單一邏輯核心之軟核心。

圖 14 為根據本發明一具體實施例中用於支援邏輯核心與虛擬核心功能之片段分段的示例性實施。

圖 15 為根據本發明一具體實施例中實施一多實體對多邏輯模式之一示例性四片段處理器的一片段記憶體。

圖 16 為根據本發明另一具體實施例中實施一多實體對多邏輯模式之一示例性四片段處理器的一片段記憶體。

圖 17 為根據本發明一具體實施例中實施一多軟核心對多邏輯核心模式之一示例性四片段處理器的一片段記憶體。

圖 18 為根據本發明一具體實施例中實施一多軟核心對一邏輯核心模式之一示例性四片段處理器的一片段記憶體。

圖 19 為根據本發明一具體實施例中實施一實體對邏輯模式之一示例性四片段處理器之位址計算與執行單元、運算子/結果緩衝器、執行緒的暫存器檔案與共用分割排程器。

圖 20 為根據本發明一具體實施例中實施一多實體對多邏輯模式之一示例性四片段處理器之位址計算與執行單元、運算子/結果緩衝器、執行緒的暫存器檔案與共用分割排程器之另一種實施。

圖 21 為根據本發明一具體實施例中實施一多軟核心對多邏輯模式之一示例性四片段處理器之位址計算與執行單元、暫存器檔案與共用分割排程器。

圖 22 為根據本發明一具體實施例中實施一多軟核心對一邏輯核心模式之一示例性四片段處理器之位址計算與執行單元、暫存器檔案與共用分割排程器。

圖 23 為根據本發明一具體實施例之一示例性微處理器管線的示意圖。

【主要元件符號說明】

10	通用前端提取與排程器
11-14	可分割引擎

20-23	程式碼序列
30	通用內連線結構
101-104	片段
110a	記憶體通用內連線
110b	執行通用內連線
121-124	位址計算與執行單元
150	通用前端提取與排程器
550	運行時間最佳化器排程器
850	桶
852	執行緒分配指標陣列
852	執行緒桶分配指標
853	目的地繼承向量
854	桶執行旗標
855	桶預備位元
856	桶繼承向量 B_iv
857	汰除執行緒指標
901	桶繼承向量
902	執行緒標頭
903	轉送
904	引擎/核心或處理器
905	局部暫存器檔案
906	執行緒 2
910	向量
1200	位址暫存器內連線

1200	位址內連線匯流排
1201	位址計算單元內連線
1202	位址陣列
1206	標籤陣列
1207	資料陣列
1211	片段
1401、1402、1403、1404	埠
1450	臨界值限制暫存器
1460	計數器
2300	微處理器管線
2301	提取模組
2302	解碼模組
2303	分配模組
2304	分派模組
2305	執行模組
2306	汰除模組

I533129

公告本

發明專利說明書

（本說明書格式、順序，請勿任意更動，※記號部分請勿填寫）

※ 申請案號：101110082

G06F 12/08 (2006.01)
15/76 (2006.01)
9/06 (2006.01)
9/30 (2006.01)

申請日：101 年 3 月 23 日　　※IPC 分類：

一、發明名稱：（中文/英文）

使用可分割引擎實體化的虛擬核心執行指令序列程式碼區塊

EXECUTING INSTRUCTION SEQUENCE CODE BLOCKS BY USING VIRTUAL CORES INSTANTIATED BY PARTITIONABLE ENGINES

二、中文發明摘要：

一種用於使用一處理器之複數個虛擬核心來執行指令的方法。該方法包括使用一通用前端排程器接收一輸入的指令序列，並將該輸入的指令序列分割成複數個指令的程式碼區塊。該方法另包括產生複數個繼承向量，其描述該等程式碼區塊的指令之間的相關性，且將該等程式碼區塊分配至該處理器的複數個虛擬程式碼，其中每一虛擬核心包含複數個可分割引擎之資源的個別子集合。該等程式碼區塊使用該等可分割引擎根據一虛擬核心模式及根據該等個別繼承向量來執行。

三、英文發明摘要：

A method for executing instructions using a plurality of virtual cores for a processor. The method includes receiving an incoming instruction sequence using a global front end scheduler, and partitioning the incoming instruction sequence into a plurality of code

blocks of instructions. The method further includes generating a plurality of inheritance vectors describing interdependencies between instructions of the code blocks, and allocating the code blocks to a plurality of virtual cores of the processor, wherein each virtual core comprises a respective subset of resources of a plurality of partitionable engines. The code blocks are executed by using the partitionable engines in accordance with a virtual core mode and in accordance with the respective inheritance vectors.

八、圖式：

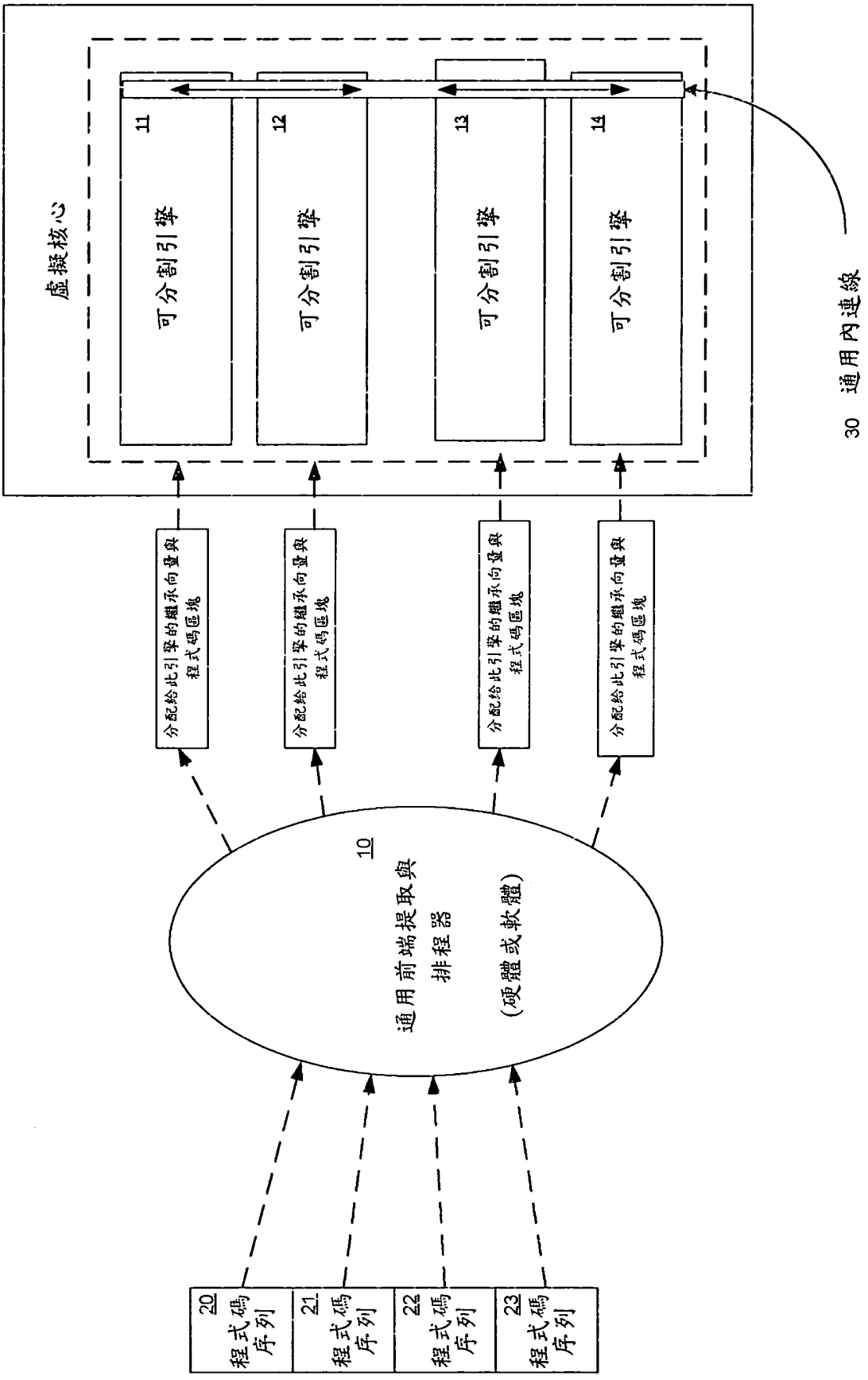
虛擬核心
11
可分割引擎
12
可分割引擎
13
可分割引擎
14
可分割引擎
30 通用內連線
分配給此引擎的繼承向量與程式碼區塊
分配給此引擎的繼承向量與程式碼區塊
分配給此引擎的繼承向量與程式碼區塊
分配給此引擎的繼承向量與程式碼區塊
10
通用前端提取與排程器
(硬體或軟體)
20 程式碼序列
21 程式碼序列
22 程式碼序列
23 程式碼序列

圖 1A

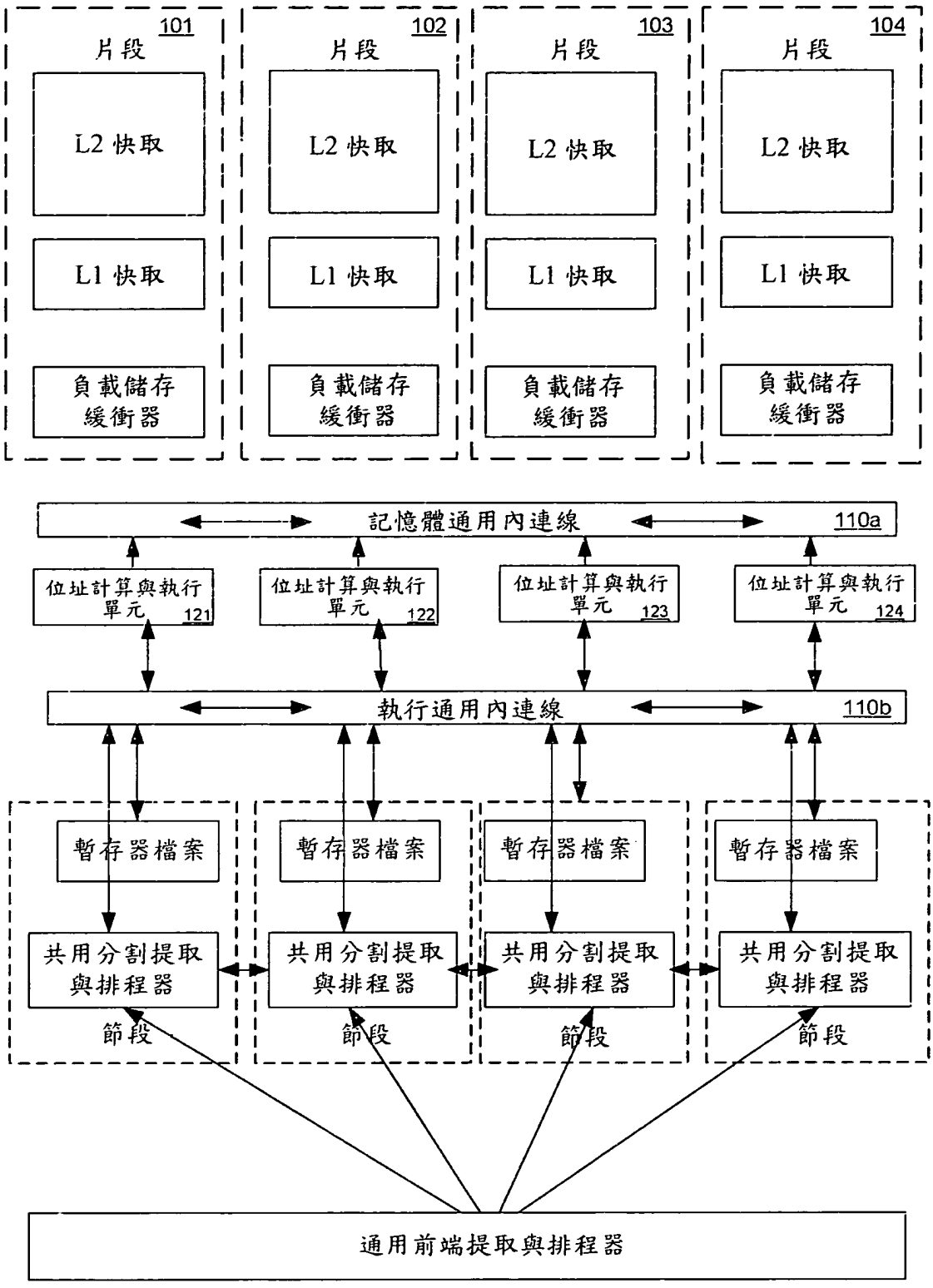
101
片段
L2 快取
L1 快取
負載儲存緩衝器
102
片段
L2 快取
L1 快取
負載儲存緩衝器
103
片段
L2 快取
L1 快取
負載儲存緩衝器
104
片段
L2 快取
L1 快取
負載儲存緩衝器
記憶體通用內連線
110a
位址計算與執行單元 121
位址計算與執行單元 122
位址計算與執行單元 123
位址計算與執行單元 124
執行通用內連線
110b
暫存器檔案
共用分割提取與排程器
節段
通用前端提取與排程器

圖 1B

四、指定代表圖：

(一)本案指定代表圖為：圖 1A。

(二)本代表圖之元件符號簡單說明：

10　通用前端提取與排程器

11-14 可分割引擎

20-23 程式碼序列

五、本案若有化學式時，請揭示最能顯示發明特徵的化學式：無。

之方式。可被擴充來處理可能具有較少緊密互動的較大量之執行緒的那些電路被描述為一通用前端(例如圖 1 所示的通用前端排程器 150)。

該程序開始於提取一新執行緒矩陣/桶/區塊，然後該新執行緒桶被指定至該桶緩衝器中一空的桶槽。在執行緒分配指標陣列852中該等執行緒分配指標之每一者構成一桶間隔，使得該執行緒被允許實體上來將其指令的區塊/桶置於其中。那些執行緒之每一者以循環的方式保持分配桶至其連續空間之相對應間隔內的該桶緩衝器陣列中。在每一執行緒空間內該等桶/區塊被指定一新編號852，其在每一次一新的桶/區塊被指定時即遞增。對於桶850中每一有效來源。每一桶的該等有效來源具有一有效讀取位元「Rv」，代表此來源為此桶內該等指令所需要。利用相同方式，要由此桶中的指令寫回的每一目的地暫存器在該桶中具有一有效位元「Wv」，且其在一目的地繼承向量853中具有一欄位。當一新的桶要被提取到該桶緩衝器中時，其由執行緒桶分配指標852所指到的該先前分配的桶繼承該目的地繼承向量。該繼承向量自該先前分配的桶複製，然後其覆寫對應於將由那些桶指令更新的該等暫存器的那些有效目的地欄位。該等有效目的地將標示該目前桶編號，而該等無效目的地係自該桶內的該相對應繼承向量複製。然後該執行緒桶指標藉由遞增其指標(其纏繞在其間隔內)對該新的提取桶來更新。

在該桶分派與執行階段中，每當一桶被執行而沒有任何異常處理時，則桶執行旗標(包含該桶編號)854 被設定，並廣播到整個該桶緩衝器，並在每一桶內被閂鎖/監視，其具有以該桶編號為來源的一來源。亦可能連同該桶編號傳送其它相關的資訊，例如關於虛擬暫存器位置的資訊。當該等來源桶的所有該等執行旗標被設置在一桶內時，則桶預備位元 855 被設定，且該桶預備好被分派與執行。當該桶執行而沒有任何異常且其預備好以該程式的序列順序來更新該架構狀態時，則其汰除該桶，且汰除執行緒指標 857 被遞增到該陣列中的下一個桶。該汰除的桶位置可被指定給一新的桶。

那些緊密相關的執行緒皆可在該矩陣/桶/區塊緩衝器內共存；每一執行緒將佔據屬於該執行緒的連續桶之一間隔。該執行緒的分配指標以一循環方式移動到桶的此間隔內，以該所述的循環方式提取新的指令桶，並將其分配在該執行緒間隔內。利用這種間隔區段化，該整個桶緩衝器利用桶的不同或相等間隔長度來動態地區分。

此處所介紹的該繼承向量的概念係針對該指令桶以及該執行緒。每一指令矩陣/區塊/桶寫入到該等架構性暫存器當中特定的暫存器當中。在分配階段中每一新的桶更新此繼承向量，其寫入其本身的該執行緒與桶編號到此向量中，而使得其並未寫入其中的該等暫存器之該等欄位保持未更新。此桶繼承向量 B_iv 856 以程式順序由每一桶轉送至下一個桶。在圖 2 中，如果在該矩陣中該等指令寫入

到那些暫存器中時，每一矩陣將其本身的編號寫入到該等架構目的地暫存器中，否則其繼承來自在該執行緒中該先前的桶之該B_iv的該數值。

圖3所示為根據本發明一具體實施例之示例性硬體電路圖，其中顯示有儲存運算子與結果的一分段的暫存器檔案並具有一內連線。圖3所示為經由該執行通用內連線耦合至複數個執行單元的一運算子結果緩衝器。

圖4所示為根據本發明一具體實施例之一通用前端排程器之示意圖。該通用前端排程器設置成處理可能具有較少緊密互動之較多數目的執行緒(例如圖 1 所示之排程器150中的通用前端)。此圖所示為來自一邏輯核心的一序列的指令如何被分配橫跨許多虛擬核心。此程序將針對存在於該機器中每一邏輯核心來重複。必須注意到圖4的「引擎」包含一虛擬核心的該等組件，其中該暫存器檔案被明確地描述成顯示在該暫存器檔案階級處虛擬核心間的通訊之態樣。

例如，如圖4所示，該通用前端排程器可處理一執行緒標頭902，但並不需要該執行緒內該等實際指令來強制進行橫跨那些遠離的執行緒之相關性檢查。該執行緒的標頭與其桶的該等子標頭僅包含關於那些執行緒與桶寫入(至那些指令之目的地暫存器)的該等架構暫存器之資訊。在那些標頭中不需要包括實際指令或那些指令的來源。實際上其足以列出那些目的地暫存器或一位元向量，其中每

一個別位元針對為一指令之一目的地的每一暫存器做設定。該標頭並不需要被實際上放置成該等指令的一標頭；其可為任何格式的封包，或該等執行緒內該等指令之該等目的地暫存器之小型化表示，其可能或可能不儲存有該等指令資訊之其餘部份。

此通用前端僅以程式順序提取該等執行緒/區塊之該等標頭，並產生動態執行緒及/或桶繼承向量 901(Tiv 及/或 Biv)。每次分配一新執行緒時，那些繼承向量藉由保持該目前執行緒桶將不會寫入或更新之該等舊的欄位來轉送，如 903 所示。那些繼承向量被分佈到大量的引擎/核心或處理器 904，其每一者可包括一局部前端與一提取單元(其將提取與儲存由每一桶之相關性向量所產生的該等實際指令)及具有局部暫存器檔案 905 的一局部矩陣/區塊/桶緩衝器。然後該等局部前端提取該等實際指令，並使用來自由該通用前端取得的該等繼承向量之資訊來填充被帶入到那些引擎來執行的該等指令之該等指令來源的相關性資訊。圖 4 所示為一通用前端實施，及其僅使用關於該等指令之簡要資訊(例如其僅為那些指令寫入其中的該等暫存器)將該等繼承向量散佈到不同的引擎 904 的方式。其它要放置在該標頭中而有助益的資訊為關於在該等執行緒內或橫跨其間的該控制路徑中的變化之資訊。一通用分支預測器可用於預測橫跨那些執行緒之控制流程，所以這些標頭可包括該等分支的目的地與偏移量。除了該分支預測器來決定控制流程之外，該硬體/編譯器可決定橫跨一分支的兩條控制路徑來分派獨立的執行緒。這此例中，稍

後將使用該繼承向量合併那兩條路徑之執行。圖4亦顯示出當一新執行緒的一標頭由該通用前端提取時的轉送程序。例如執行緒2(906)將更新被轉送給它的相對應繼承向量901，造成暫存器1、2、3、4、6、0與7利用T2標記被更新的向量910。請注意在910中，暫存器5並非由T2桶寫入，因此其標記係由一先前繼承向量所繼承。

一項有趣的觀察為該等暫存器檔案允許該等核心/引擎之間橫跨通訊。由於橫跨引擎而需要的該等暫存器之一早期要求(來降低該存取潛時)只要在該執行緒的該等指令桶被提取與分配在該局部桶緩衝器中時即被放置。此時該來源相關性資訊出現(populate)，使得可能在該等實際指令被分派來執行之很久之前就發出橫跨引擎執行緒參照。在任何情況下，該指令將不會被分派，直到該交互參照的來源被轉送且抵達時。此交互參照的來源可被儲存在該局部多執行緒的暫存器檔案或暫存器快取中。雖然此交互參照的來源可被儲存在類似於該負載儲存緩衝器的一緩衝器中(其可重新使用該負載儲存緩衝器實體儲存器與相關性檢查機制，但做為一暫存器負載而非記憶體負載)。可使用許多拓樸來連接橫跨該等引擎/核心的該等暫存器檔案，其可為一環狀拓樸或橫桿拓樸或網格路由內連線。

以下的討論可例示暫存器檔案分段化可如何用於一引擎內且亦可橫跨引擎來使用。當該桶被分派時，其來源被傳送(同時或依序)至該暫存器檔案與該暫存器快取。如果該暫存器檔案被實體上統一，並直接支援執行緒化，則

其餘節段當中來完成。但是，此無法調整，因為我們無法利用具有四倍於單獨一節段所需要之該等寫入埠的每一記憶胞之一有效率的暫存器檔案。我們提出一種機制使得該暫存器檔案可建構成其將不會受到這種單一執行緒暫存器-廣播延伸的影響。

必須注意到關於在本發明之具體實施例中所使用的暫存器節段之額外態樣可見於 Mohammad A. Abdallah 於 2007 年 11 月 14 日所立案的美國專利申請案編號 2010/0161948，其名為「在支援多種內容切換模式與虛擬化方案之多執行緒架構中處理複雜指令格式的裝置與方法」(APPARATUS AND METHOD FOR PROCESSING COMPLEX INSTRUCTION FORMATS IN A MULTITHREADED ARCHITECTURE SUPPORTING VARIOUS CONTEXT SWITCH MODES AND VIRTUALIZATION SCHEMES)。

圖 7 為根據本發明一具體實施例之一多核心處理器之一片段化記憶體子系統之細部示意圖。圖 7 概略顯示出執行緒之間及/或負載與儲存器之間該同步化方案的一種完整方案與實施。該方案描述用於橫跨負載/儲存架構及/或橫跨記憶體參照及/或執行緒的記憶體存取之記憶體參照的同步化與歧義消除的一種較佳的方法。在圖 7 中，我們顯示了暫存器檔案之多個節段(位址及/或資料暫存器)、執行單元、位址計算單元、及第 1 級快取及/或負載儲存緩衝器與第 2 級快取及位址暫存器內連線 1200 與位址計算單

依此方式，存取該負載儲存緩衝器的該等資源與存取該 L1 快取的該等資源可以是更為政策導向，並可更為基於進行轉送進度之各自的執行緒或核心之該等需求。此顯示成一個邏輯核心具有該等片段之每一者的該儲存緩衝器與該 L1 快取的一動態分配的埠與一動態分配的部份。依此方式，該邏輯核心包含每一片段的該等資源之一非固定動態分配的片層。

圖 18 為根據本發明一具體實施例中實施一多軟核心對一邏輯核心模式之一示例性四片段處理器的一片段記憶體。

在圖 13 的操作模式中，該多軟核心對一邏輯核心模式，其中該等軟核心用於類似在執行應用程式中一單一邏輯核心來運作，該等軟核心之每一者設置成協同於該等其它軟核心運作成一單一邏輯核心。一單一執行緒或核心具有該等負載儲存緩衝器之所有該等資源與該等 L1 快取的所有該等資源。在這種實施中，一單一執行緒的應用程式將其指令序列分開，並分配在該等軟核心之間，其中它們被協同地執行來達成高單一執行緒效能。依此方式，單一執行緒的效能可隨著加入額外的軟核心來調整。此係顯示在圖 18 中，其中一個示例性邏輯核心與其與該處理器之該等資源的關係藉由遮影該處理器之所有該等資源來顯示。

七、申請專利範圍：

1. 一種用於使用一處理器之複數個虛擬核心來執行指令的方法，該方法包含：

使用一通用前端排程器接收一輸入的指令序列；

分割該輸入的指令序列成為複數個指令的程式碼區塊；

產生複數個繼承向量來描述該等程式碼區塊之指令之間的交互相關性；

分配該等程式碼區塊至該處理器的複數個虛擬核心，其中每一虛擬核心包含複數個可分割引擎的資源之一個別的子集合，其中每一可分割引擎包含一節段、一記憶體片段、及複數個執行單元，其中每一可分割引擎的資源被分割成與其他可分割引擎的分割資源實體化(instantiate)一虛擬核心，其中每一可分割引擎的資源間的通訊係由一通用內連線結構（global interconnecion structure）所支援；及

使用該等可分割引擎根據一虛擬核心模式及根據該等個別繼承向量來執行該等程式碼區塊。

2. 如申請專利範圍第 1 項之方法，其中每一可分割引擎節段另包含一共用分割提取與排程器。

3. 如申請專利範圍第 1 項之方法，其中每一節段另包含一暫存器檔案。

4. 如申請專利範圍第 1 項之方法，其中每一可分割引擎另包含一 L1 快取片段與 L2 快取片段與一負載儲存緩衝器。

5. 如申請專利範圍第1項之方法，其中該等複數個虛擬核心實施一執行模式，其中每一可分割引擎的一實體資源的子集合被分配成支援一邏輯核心的一單一邏輯執行緒之執行。

6. 如申請專利範圍第5項之方法，該等複數個虛擬核心實施複數個邏輯核心。

7. 如申請專利範圍第1項之方法，其中該等複數個虛擬核心實施一執行模式，其中每一可分割引擎的實體資源根據一可調整臨界值被動態地分配來支援一單一邏輯核心的一單一邏輯執行緒之執行。

8. 如申請專利範圍第7項之方法，該等複數個虛擬核心實施複數個邏輯核心。

9. 如申請專利範圍第1項之方法，其中該等複數個虛擬核心實施一執行模式，其中每一可分割引擎之實體資源的該子集合根據一可調整臨界值被分配來支援一單一邏輯執行緒之執行。

10. 一種用於使用一處理器之複數個虛擬核心來執行指令的系統，該系統包含：

一通用前端排程器，用於接收一輸入的指令序列，其中該通用前端排程器分割該輸入的指令序列成為複數個指令的程式碼區塊，並產生複數個繼承向量來描述該等程式碼區塊的指令之間之交互相關性；及

耦合來接收由該通用前端排程器分配的程式碼區塊之該

處理器的複數個虛擬核心，其中每一虛擬核心包含複數個可分割引擎的一個別的資源子集合，其中每一可分割引擎包含一節段、一記憶體片段、及複數個執行單元，其中每一可分割引擎的資源被分割成與其他可分割引擎的分割資源實體化一虛擬核心，其中每一可分割引擎的資源間的通訊係由一通用內連線結構所支援，其中該等程式碼區塊使用該等可分割引擎根據一虛擬核心模式與根據該等個別的繼承向量來執行。

11. 如申請專利範圍第10項之系統，其中每一可分割引擎節段另包含一共用分割提取與排程器。

12. 如申請專利範圍第10項之系統，其中每一節段另包含一暫存器檔案。

13. 如申請專利範圍第10項之系統，其中每一可分割引擎另包含一L1快取片段與L2快取片段與一負載儲存緩衝器。

14. 如申請專利範圍第10項之系統，其中該等複數個虛擬核心實施一執行模式，其中每一可分割引擎的一實體資源的子集合被分配成支援一邏輯核心的一單一邏輯執行緒之執行。

15. 如申請專利範圍第14項之系統，該等複數個虛擬核心實施複數個邏輯核心。

16. 如申請專利範圍第10項之系統，其中該等複數個虛擬核心實施一執行模式，其中每一可分割引擎的實體資源根據一可

調整臨界值被動態地分配來支援一單一邏輯核心的一單一邏輯執行緒之執行。

17. 如申請專利範圍第16項之系統，該等複數個虛擬核心實施複數個邏輯核心。

18. 如申請專利範圍第10項之系統，其中該等複數個虛擬核心實施一執行模式，其中每一可分割引擎之實體資源的該子集合根據一可調整臨界值被分配來支援一單一邏輯執行緒之執行。

19. 一種用於使用一處理器之複數個虛擬核心來執行指令的系統，該系統包含：

一運行時間最佳化器排程器，用於接收一輸入的指令序列，其中該運行時間最佳化器排程器分割該輸入的指令序列成為複數個指令的程式碼區塊，並產生複數個繼承向量來描述該等程式碼區塊的指令之間之交互相關性；及

耦合來接收由該運行時間最佳化器排程器分配的程式碼區塊之該處理器的複數個虛擬核心，其中每一虛擬核心包含複數個可分割引擎的一個別的資源子集合，其中每一可分割引擎包含一節段、一記憶體片段、及複數個執行單元，其中每一可分割引擎的資源被分割成與其他可分割引擎的分割資源實體化一虛擬核心，其中每一可分割引擎的資源間的通訊係由一通用內連線結構所支援，其中該等程式碼區塊使用該等可分割引擎根據一虛擬核心模式與根據該等個別的繼承向量來執行。

20. 如申請專利範圍第19項之系統，其中每一可分割引擎節段另包含一共用分割提取與排程器。

21. 如申請專利範圍第19項之系統，其中每一節段另包含一暫存器檔案。

22. 如申請專利範圍第19項之系統，其中每一可分割引擎另包含一L1快取片段與L2快取片段與一負載儲存緩衝器。

23. 如申請專利範圍第19項之系統，其中該等複數個虛擬核心實施一執行模式，其中每一可分割引擎的一實體資源的子集合被分配成支援一邏輯核心的一單一邏輯執行緒之執行。

24. 如申請專利範圍第23項之系統，該等複數個虛擬核心實施複數個邏輯核心。

25. 如申請專利範圍第19項之系統，其中該等複數個虛擬核心實施一執行模式，其中每一可分割引擎的實體資源根據一可調整臨界值被動態地分配來支援一單一邏輯核心的一單一邏輯執行緒之執行。

26. 如申請專利範圍第25項之系統，該等複數個虛擬核心實施複數個邏輯核心。

27. 如申請專利範圍第19項之系統，其中該等複數個虛擬核心實施一執行模式，其中每一可分割引擎之實體資源的該子集

合根據一可調整臨界值被分配來支援一單一邏輯執行緒之執行。

28. 一種用於使用一處理器之複數個虛擬核心來執行指令的方法，該方法包含：

使用一通用前端排程器接收一輸入的指令序列；

分割該輸入的指令序列成為複數個指令的程式碼區塊；

分配該等程式碼區塊至該處理器的複數個虛擬核心，其中每一虛擬核心包含複數個可分割引擎的資源之一個別的子集合，其中每一可分割引擎包含一節段、一記憶體片段、及複數個執行單元，其中每一可分割引擎的資源被分割成與其他可分割引擎的分割資源實體化一虛擬核心，其中每一可分割引擎的資源間的通訊係由一通用內連線結構所支援；及

使用該等可分割引擎根據一虛擬核心模式執行該等程式碼區塊。

29. 如申請專利範圍第28項之方法，其中該等複數個虛擬核心實施一執行模式，其中每一可分割引擎的一實體資源的子集合被分配成支援一邏輯核心的一單一邏輯執行緒之執行。

30. 如申請專利範圍第29項之方法，該等複數個虛擬核心實施複數個邏輯核心。

31. 如申請專利範圍第28項之方法，其中該等複數個虛擬核心實施一執行模式，其中每一可分割引擎的實體資源根據一可調整臨界值被動態地分配來支援一單一邏輯核心的一單一邏

輯執行緒之執行。

32. 如申請專利範圍第 31 項之方法，該等複數個虛擬核心實施複數個邏輯核心。

33. 如申請專利範圍第 28 項之方法，其中該等複數個虛擬核心實施一執行模式，其中每一可分割引擎之實體資源的該子集合根據一可調整臨界值被分配來支援一單一邏輯執行緒之執行。