200809613

發明專利說明書

（本說明書格式、順序及粗體字，請勿任意更動，※記號部分請勿填寫）

※ 申請案號：P6010639P4

※ 申請日期：P6.2.16　※IPC 分類：G06F9/32 (2006.01)

一、發明名稱：（中文/英文）

由輸入來源接收指令直接執行的方法/

EXECUTION OF INSTRUCTIONS DIRECTLY FROM INPUT SOURCE

二、申請人：（共1人）

姓名或名稱：（中文/英文）

科技資產股份有限公司/TECHNOLOGY PROPERTIES LIMITED

代表人：（中文/英文）

F. 艾力克 桑德斯/F. ERIC SAUNDERS

住居所或營業所地址：（中文/英文）

美國加州 95014 庫普堤諾市 20400 史帝芬溪大道第 500 室

國　籍：（中文/英文）

美國/US

三、發明人：（共1人）

姓　名：（中文/英文）

查理斯 H. 莫爾/CHARLES H. MOORE

國　籍：（中文/英文）

美國/US

四、聲明事項：

☐ 主張專利法第二十二條第二項☐第一款或☐第二款規定之事實，其事實發生日期為：　年　月　日。

☑ 申請前已向下列國家（地區）申請專利：

【格式請依：受理國家（地區）、申請日、申請案號　順序註記】

☑ 有主張專利法第二十七條第一項國際優先權：

1.美國、2006/2/16、11/355,495

2.美國、2006/2/16、11/355,513

3.美國、2006/3/31、60/788,265

4.美國、2006/5/3、60/797,345

5.美國、2006/5/26、11/441,784

6.美國、2006/5/26、11/441,812

7.美國、2006/5/26、11/441,818

☐ 無主張專利法第二十七條第一項國際優先權：

☐ 主張專利法第二十九條第一項國內優先權：

【格式請依：申請日、申請案號　順序註記】

☐ 主張專利法第三十條生物材料：

☐ 須寄存生物材料者：

國內生物材料 【格式請依：寄存機構、日期、號碼　順序註記】

國外生物材料 【格式請依：寄存國家、機構、日期、號碼　順序註記】

☐ 不須寄存生物材料者：

所屬技術領域中具有通常知識者易於獲得時，不須寄存。

九、發明說明：

【發明所屬之技術領域】

本發明係關於運算器及運算器處理器之領域，特別是關於用以當運算器從外部來源接收指令時，無須先儲存該指令，而能夠執行該指令之方法和裝置，以及一相關的方法用以使用該方法和裝置以協助運算器之間的通訊以及一運算器使用其他運算器之可用資源的能力。本發明之直接執行方法及裝置目前主要係用於將複數運算器整合於單一微晶片上，其中運作效率的重要不僅是因為想要增加運作速度，且也是因為要省電及減少熱能（其為更具效率之一結果）。

【先前技術】

在運算領域中，處理速度是一項重要的品質，且一直推動要製造出更快的運算器和處理器。但是，業界一般均知增加微晶片速度的界線很快就被達到，至少以目前已知的技術是如此。因此，漸漸關注於使用多處理器來增加整體運算器速度，其係藉由以這些處理器來分擔運算器任務來達成。

多處理器的使用造成了對於處理器之間通訊的需要。在處理器之間的確有大量通訊在進行，使得相當比例的時間是耗費在其間的指令和資料的傳輸。當這些通訊為數可觀時，每個額外的需執行完成的指令，都造成處理延遲增加，其累積起來可能很可觀。傳統上，用以從一運算器通訊指令或資料到另一運算器的方法，係先將該資料或指令

儲存在接收端運算器，並繼而將之呼叫出以執行(當其為指令)或處理(當其為資料)。

減少運算器之間傳輸、接收及使用資料(以資料或指令的形式)所需的步驟數是有用的。但是，就發明人所知，沒有一個習知系統可以有效簡化上述程序。

而且，在習知技術中，已知必須時時「取得運算器的注意」。其係為，有時即使一運算器正忙於某一工作，另一時間敏感工作需要可能產生，其必須暫時使該運算器從該第一工作中分心。其例子包括，但不限於，當一使用者輸入裝置用於提供輸入至該運算器。在此例中，該運算器可能必須暫時要告知收到該輸入及/或依據該輸入動作。繼之，該運算器可以繼續它在該輸入之前的動作，或依據該輸入改變他正在執行的動作。雖然在此係以一外部輸入為例，當在運算器內部元件之間對 ALU 的投入有一潛在衝突產生時，也有同樣的狀況發生。

當從輸入輸出埠接收資料及改變狀態時，習知可用兩種方法因應。其一為查詢(poll)該埠，其係在固定期間讀取該埠的狀態，以決定是否有任何資料被接收或者發生任何狀態改變。但是，該埠之查詢耗費相當多時間及資源，而這些時間和資源可以用以執行其他工作。使用「中斷(interrupt)」常被視為是較佳的替代方案。當使用中斷時，處理器執行被指定的工作，依據一位元組被接收或狀態改變，一輸入輸出埠/裝置需要處理時，其傳送一中斷要求(IRQ)至該處理器。當該處理器接收該中斷要求時，就停止其目前的指令，將一些工作置於佇列中，並執行該適當

的中斷服務任務(ISR)，其可以將該位元組從該埠移除，並將其置於緩衝區中。當該 ISR 完成時，該處理器回到原處。使用這種方法，該處理器無須浪費時間去注意是否輸入輸入裝置需要處理，反之，該裝置僅需於其需要處理時服務該中斷。但是，由於伴隨著中斷的使用會產生許多經常成本，因此中斷之使用本身在許多狀況下是不當的。例如，每當有中斷發生時，運算器必須要暫存某些和其正在執行的工作相關的資料，並載入和該中斷相關的資料，並於該中斷處理完後，再重新載入該先前工作必須的該資料。顯然地，需要減少或消除這些時間和資源的經常成本。然而，尚無習知技術能夠減少對中斷之需要。

【發明內容】

因此，本發明之目的在於提供一裝置和方法，以在兩個以上運算器互相通訊資料及/或指令時，增加處理速度。

本發明之另一目的在於提供一裝置及方法，以廉價地提供大量運算能力。

本發明之另一目的在於提供一裝置及方法，以在最短時間內完成大量運算的工作。

本發明之另一目的在於提供一運算器裝置，其提供大量的運算能力。

本發明之另一目的在於提高運算器及運算器控制裝置之間的通訊效率。

本發明之另一目的在於提高運算器及運算器之間的通訊效率。

本發明之另一目的在於提高運算器相互通訊及與其他裝置(例如使用者輸入裝置等)通訊之方式的效率。

簡言之，本發明之一實施例係為一具有其本身之記憶體的運算器，如此其可以執行獨立運算功能。依據一實施例，複數運算器配置於一陣列中。為了合作完成任務，該運算器必須互相傳遞資料及/或指令。由於所有同時運作的運算器通常可以提供的運算能力遠大於大部分任務所需，又因為不論使用何種演算法或方法來分配任務給數個運算器，幾乎都會導致分配不均，可預見的是，在特定時間點，至少某些(也許是大部分)的運算器並未積極投入任務的完成。因此，必須找到一種方法應用於這些未充分使用的運算器，使其能夠「借出」其運算資源及/或記憶空間，以協助鄰近的忙碌的運算器。為了使此種關係更具效率及且有用，需要使鄰近的運算器之間的通訊和互動能夠盡可能地快速有效率。因此，本發明提供用於一運算器之一裝置及方法，以執行由另一運算器所提供之指令及/或處理資料，而無須在執行/處理之前接收並儲存該資料及/或指令。需注意的是，本發明亦適用於作為中介的指令，其使得一運算器將從另一運算器傳來的指令或資料再傳給別個運算器。

依據本發明實施例，為了避免不必要的能量消耗及產熱，當一運算器欲與一或多個鄰近運算器通訊時，其可以進入幾乎不耗能的睡眠模式，直到鄰近之一或多個運算器執行完成該通訊。但是，這並非本發明必須執行的。而且，為了達到所欲之能量節省及減少產熱，可使起始的運算器

在等待通訊傳訊完成時，先暫停(或至少使其大幅減少)其能量消耗。當可理解這可以藉由數種方法中任一種達成。例如，若運算器由一內建或外部時脈來定時，則在該期間，該時脈可以變慢或暫停。亦可以理解到上述實施例可以應用於本發明所述之外的原因，雖然在此所述之實施例是目前發明人所知最佳且最有效率的實施方式。

依據在此所述之本發明一實施例，不論指令和資料的來源為該運算器之內部記憶體，或者，該指令及資料是從另一來源(例如另一個運算器、一外部通訊埠等)，指令和資料基本上是被同等對待的。這一點很重要，因為「額外」操作(例如儲存該資料或指令，並在之後將之從內部記憶體中呼叫出來)變得不需要了，因此可以減少需要的指令數，並增加相關運算器的操作速度。

上述實施例之另一方面為，很少的指令群可以同時傳送到另一個運算器，使得可以快速且容易地完成需要重複迴圈的相對簡單的操作。這使得運算器之間的通訊過程大幅加快。

上述實施例之另一方面為，因為有相當多的運算器可以執行各種任務，且因為在等待輸入時，一或多個運算器可以進入幾乎不耗能的睡眠狀態，此種運算器可以被指定為用以等候輸入，以減少或消除去中斷其他可能正在執行其他任務之運算器的必要。

為讓本發明之上述和其他目的、特徵、和優點能更明顯易懂，下文特舉出較佳實施例，並配合所附圖式，作詳細說明如下：

【實施方式】

為了讓本發明之目的、特徵、及優點能更明顯易懂，下文特舉較佳實施例，並配合所附圖示，做詳細之說明。本發明說明書提供不同的實施例來說明本發明不同實施方式的技術特徵。其中，實施例中的各元件之配置係為說明之用，並非用以限制本發明。且實施例中圖式標號之部分重複，係為了簡化說明，並非意指不同實施例之間的關聯性。

由個別運算器構成之陣列已知為可以執行本發明之一模式。該陣列係顯示於第 1 圖之示意圖中，並以一般參照號碼 10 標示之。該運算器陣列 10 具有複數個(在此實施例中為 20 個)運算器 12(有時在一陣列的例子中亦指稱為「核心」或「節點」)。在所示之例中，所有的運算器 12 都配置於一單一晶粒 14 上。依據本發明，每一個運算器 12 係為一通用獨立運作運算器，後文將詳述之。運算器 12 藉由複數(其數量之細節容後再述)連結資料匯流排 16 相連結。在此實施例中，資料匯流排 16 係為雙向，非同步，高速，平行之資料匯流排，雖然在本發明範圍中，亦可以採用其他連結裝置來達到此目的。在本實施例之陣列 10 中，不僅是運算器 12 之間的資料通訊是非同步的，個別的運算器 12 也可以運作於一內部非同步模式。發明人發現這可以提供重要的益處。例如，由於時脈訊號不必傳送到整個運算器陣列 10 中，所以省下了可觀的電力。而且，無須傳送時脈訊號消除了許多會限制陣列 10 的大小或導致其他已

知困難的定時問題。而且，因為沒有時鐘運作於其中，每一個運算器在不執行指令時，可以幾乎不耗能，所以個別運算器非同步運作之此一事實，節省大量電力，

熟悉該領域技藝者可以知道，在晶粒 14 上有其他的元件，其係為了簡單起見而在第 1 圖中被略去不表。這些其他的元件包括：電力匯流排、外部連接墊、及其他微處理器晶片的一般元件。

運算器 12e 為運算器 12 之一例，其不位於陣列 10 的周邊。亦即，運算器 12e 具有 4 個直角相鄰的運算器 12a、12b、12c、及 12d。這一群運算器 12a~12e 係作為一例，以說明陣列 10 中運算器 12 之間通訊通訊的細節。如第 1 圖所示，類似如運算器 12e 的內部運算器可以藉由匯流排 16 直接和 4 個其他的運算器 12 通訊。在後續討論中，討論的原理可適用於所有的運算器 12，除了陣列 10 邊緣的運算器 12，其可能僅能和個或 2 個(例如角落的運算器 12)運算器 12 直接通訊。

第 2 圖為第 1 圖一部份之細部圖，其僅顯示一些運算器 12，並特別包含運算器 12a~12e。第 2 圖亦顯示資料匯流排 16，其中每一者均具有一讀取線(read line)18、一寫入線(write line)20 及複數(在此例中為 18)條資料線(data line)22。資料線 22 通常可以平行地同時傳送一個 18 位元指令文字之所有位元。需注意的是，在本發明實施例中，某些運算器 12 是鄰近運算器的鏡像(mirror image)。但是，不論運算器 12 是全部同向或為鄰近運算器之鏡像，均非本發明之特徵。因此，為了更佳地描述本發

明，這個可能的複雜狀況不在討論之列。

依據本發明方法，運算器 12(例如運算器 12e)可以高設 1 條、2 條、3 條、或 4 條讀取線 18，以準備從個別的 1 個、2 個、3 個、4 個鄰近的運算器 12 接收資料。同樣地，某個運算器 12 也可以高設 1 條、2 條、3 條、或 4 條寫入線 20。雖然本案發明人並不認為將運算器 12 之 1 條以上的寫入線 20 高設有什麼實際上的價值，不過，這麼做也不超出本發明範圍，其係因為可以理解到在未來有可能會出現此種運用。

當鄰接的運算器 12a、12b、12c、或 12d 高設其本身和運算器 12e 之間的寫入線 20，若運算器 12e 已經高設對應之讀取線 18，則一文字在對應之資料線 22 上，從運算器 12a、12b、12c、或 12d 傳送到運算器 12e。繼之，該傳送端運算器 12 釋出寫入線 20，而接收端運算器 12(在此為運算器 12e)則拉低寫入線 20 及讀取線 18。後者的動作告知傳送端運算器 12 資料已經被接收。注意到上述描述並非指定事件發生之順序。在實際運作中，接收端運算器可以在稍早於傳送端運算器 12 釋出(停止拉高)其寫入線 20 之時，嘗試將寫入線 20 設低。在此，傳送端運算器 12 一釋出其寫入線 20，該寫入線 20 就被接收端運算器 12e 拉低。

在本實施例中，只有程式錯誤可能導致在資料匯流排 16 兩端之兩個運算器 12 嘗試要高設其間的讀取線 18。而且，在資料匯流排 16 兩端之兩個運算器 12 同時嘗試要高設其間的讀取線 18 也會導致錯誤。同樣地，如上所述，目前並不認為有必要使單一運算器 12 將其 4 條寫入線 20 之

一條以上高設。但是，目前認為有可能要將不同組合的讀取線 18 高設，使得運算器 12 中之一者處於等待狀態，以等待資料從第一個選取的運算器 12 高設其對應之寫入線 20。

在上述例子中，運算器 12e 被敘述為在鄰接的運算器(從運算器 12a、12b、12c、及 12d 中選取至少一者)以高設其寫入線 20 時，高設其讀取線 18 中至少一者。但是，此程序絕對可以相反順序發生。例如，若運算器 12e 嘗試要寫入運算器 12a，則運算器 12e 可以高設在運算器 12e 和運算器 12a 之間的寫入線 20。若在那時，在運算器 12e 和運算器 12a 之間的讀取線 18 尚未被運算器 12a 高設，則運算器 12e 等待，直到運算器 12a 高設該讀取線 18。繼之，如上所述，當對應的寫入線 20 和讀取線 18 都被高設，則由資料線 22 傳送等待被傳送的資料。之後，當傳送端運算器 12e 釋出寫入線 20 時，接收端運算器 12(在此為運算器 12a)將兩運算器(在此為運算器 12a 及 12e)之間的寫入線 20 和讀取線 18 都設低。

當一類似如運算器 12e 之運算器 12 已高設其寫入線 20 之一以等待寫入時，其只是等待，且基本上不耗能，直到如上述一般，對應之鄰接運算器 12 要求該資料，除非欲傳送資料之目的的運算器 12 已高設其讀取線 18，在該狀況下資料立刻被傳送。同樣地，當一運算器 12 已高設其讀取線 18 之一以等待讀取時，其只是等待，且基本上不耗能，直到連接到一選定運算器 12 之寫入線 20 變高以在該兩個運算器 12 之間傳送一指令串。

如上述，可以有數種可能裝置及/或方法來使得運算器12如所述般運作。但是在本實施例中，由於運算器12內部非同步地運作(再加上在彼此之間以如所述之非同步方式傳送資料)，所以其運作簡單。其係為，指令一般係依序完成。當寫入或讀取指令發生時，直到該指令被完成(或者直到他被放棄，例如被重設等)之前都不會有進一步的動作。以習知技術來說，沒有規律的時脈脈衝。而是，僅有當該被執行的指令不是讀取或寫入(已知一讀取或寫入型的指令常會需要由另一個體完成)，或者當該讀取或寫入操作事實上已完成時，一脈衝係被產生以完成次一指令。

第3圖顯示如第1圖及第2圖之運算器12之一者的一般配置的方塊圖。如第3圖所示，基本上，每一個運算器12是一個完整的運算器，其包含自己的RAM 24及ROM 26。如前所述，由於在本實施例中，其係組合在一單晶片中，運算器12有時可以稱之為個別的「節點」。

運算器12的其他基本元件為：回歸堆疊(return stack)28(包含一R寄存器29，容後再敘)、一指令區域30、一計算邏輯單元(ALU或處理器)32、一資料堆疊34及一用以解碼指令之解碼邏輯段落36。熟悉該技術者通常熟悉基於運算器(如本實施例之運算器12)之堆疊的運作。該運算器12為雙堆疊運算器，其具有該資料堆疊34及該分離的回歸堆疊28。

在本實施例中，運算器12具有4個通訊埠38，用以和相鄰的運算器12通訊。該通訊埠38係為三態驅動器，具有一關閉狀態、一接收狀態(用以將訊號驅入運算器

12)、及一傳送狀態(用以將訊號驅出運算器 12)。當然，若某一運算器 12 不在該陣列(第 1 圖)內部，如運算器 12e 之例，則一個或以上的通訊埠 38 不會被用在該運算器，至少用於上述用途。然而，這些不緊靠在晶粒 14 邊緣的通訊埠 38 可以具有額外的電路，其可以設計到該運算器 12 中或在運算器 12 外而依附之，以使得此種通訊埠 38 作為一外部輸入輸出埠 39(第 1 圖)。此種外部輸入輸出埠 39 之例包括，但不限於，USB(Universal Serial Bus)埠、RS232 串列匯流排埠、平行通訊埠、類比到數位及/或數位到類比轉換埠、及其他可能變化。不論何種額外的或改良的電路被用於該用途，依據本實施例之操作該外部輸入輸出埠 39 的方法係關於指令及/或從之接收的資料之處理係和內部通訊埠 38 所述相關。在第 1 圖中，顯示一邊緣運算器 12f 連接於介面電路 80(以方塊顯示)，用以透過一外部輸入輸出埠 39 與一外部裝置 82 通訊。

在本實施例中，指令區域 30 包含數個寄存器(register)40，其在本例中包含：A 寄存器 40a、B 寄存器 b、及 P 寄存器 c。在本實施例中，A 寄存器 40a 為一完整 18 位元寄存器，B 寄存器 b 及 P 寄存器 c 則為 9 位元寄存器。

雖然本發明並不以此實施例為限，該運算器 12 係用以執行原生型第 4 代語言指令。熟悉該第 4 代電腦語言者都知道，複雜的第 4 代指令(第 4 代文字，Fourth words)係從設計到電腦中的原生型處理器指令而來。第 4 代文字的集合則為一字典(dictionary)。在其他語言中，其可能稱

之為字庫(library)。如後將描述之細節，運算器 12 一次從 RAM 24、ROM 26 或直接從匯流排 16(第 2 圖)中之一，讀取 18 位元。但是，由於在第 4 代中，大部分指令(稱之為無運算域指令(operand-less instruction))直接從堆疊 28 及 34 中取得其運算域，其長度通常只有 5 位元，使得多至 4 個指令可以被包含在單一個 18 位元指令文字中，其條件為最該群之中的最後一個指令係從一限制的指令集合中選出，其僅需要 3 位元。(在本實施例中，一位於最後面的指令之最後兩個重要位元(significant bit)係假設為「01」。)在第 3 圖之方塊圖中亦顯示一位置排序器(slot sequencer)42。

在本實施例中，資料堆疊 34 為一用於被 ALU32 操作之參數的後進先出(last-in-first-out)堆疊，而回歸堆疊 28 則為由 CALL 和 RETURN 指令所使用之巢狀回歸位址(nested return address)的後進先出堆疊。該回歸堆疊 28 亦為 PUSH、POP、NEXT 指令所用，其細節容後再敘。資料堆疊 34 及回歸堆疊 28 並非如在許多習知的運算器中一般，為在可由一堆疊指向器(stack pointer)存取之記憶體中的陣列。在資料堆疊 34 中最上面兩個寄存器為 T 寄存器 44 及 S 寄存器 46。資料堆疊 34 的其他部分包含一環狀寄存器陣列 34a，其具有 8 個額外硬體寄存器，標示為 S_2～S_9。這 8 個在環狀寄存器陣列 34a 中的寄存器之其中一個，可以在任何時間被選取為在 S 寄存器 46 之下的寄存器。在選取該堆疊寄存器為在 S 之下的該移位寄存器中的值不能被軟體讀取或寫入。同樣地，在回歸堆疊 28 之最上面位置

為專用 R 寄存器 29，而回歸堆疊 28 的其他部分具有一環狀寄存器陣列 28a，其具有 12 個額外硬體寄存器(圖未顯示)其在本例中標示為 R_1～R_{11}。

在本實施例中，並無針對堆疊溢出(overflow)或不足(underflow)狀態的硬體偵測。一般而言，習知的處理器具有堆疊指向器及記憶體管理之類的，使得當一堆疊指向器超出該堆疊分配的記憶體範圍時，錯誤狀態被標示。其係因為，若該堆疊位於記憶體中，一個溢出或不足會覆寫或使用為不被視為堆疊之一部分的一堆疊項目。但是，由於在本發明中在堆疊 28 及 34 的底部具有環狀陣列 28a 及 34a，堆疊 28 和 34 不會溢出或不足於該堆疊區域。相反地，環狀陣列 28a 及 34a 只會繞著該寄存器的環狀陣列。因為堆疊 28 和 34 具有限定的深度，將任何東西推到堆疊 28 或 34 的頂端就表示在底部的某個東西被覆寫。將多於 10 個項目推到資料堆疊 34，或將多於 13 個項目推到回歸堆疊 28，一定會使得堆疊 28 或 34 底部的項目被覆寫。追蹤堆疊 28 和 34 中的項目數量，並且不將多於堆疊 28 和 34 分別能容納的項目推入，是軟體的責任。硬體並不會偵測堆疊底部的項目溢出或將其標示為錯誤。但是，需注意的是，軟體可以藉由各種方法，利用在堆疊 28 和 34 底部的環狀陣列 28a 及 34a。試舉一例，軟體可以假設堆疊 28 或 34 在任何時刻都是空的。無須將舊項目從該堆疊中清除，其係因為，其會被往下推到底部，而當堆疊滿時，其就會被丟失。因此，一程式無須初始化來假設該堆疊是空的。

除了上述之寄存器之外，指令區域 30 亦包含一 18 位

元的指令寄存器 30a 以儲存目前使用的指令文字 48，以及一外加的 5 位元的操作碼寄存器 30b 以儲存在該特定指令中目前被執行之指令。

第 4 圖係顯示指令文字 48 的示意圖。(需注意的是，指令文字 48 可以實際上包含指令、資料、或其組合。)該指令文字 48 由 18 位元 50 構成。此為一二進位的運算器，每一位元 50 為 0 或 1。如前所述，18 位元寬的指令文字 48 可以包含至多 4 個指令 52，位於稱之為位置零 54a、位置一 54b、位置二 54c、及位置三 54d 之 4 個位置 54。依據本發明實施例，該 18 位元指令文字 48 總是以一整體被讀取。因此，由於在指令文字 48 總是有多至 4 個指令的可能，一個不操作指令被包含於運算器 12 的指令組中，以提供類似如當不需要或不想要使用所有的可用位置 54 時之用。需注意的是，依據本發明實施例，在間隔位置(特別是位置一 54b 及位置二 54c)之位元 50 的極性(活性高相較於活性低)是相反的。但是，此並非本發明之必須特性，因此，為了更清楚說明本發明，在下列討論中不涉及此一可能的複雜狀況。

第 5 圖顯示如第 3 圖所示之位置排序器(slot sequencer)42 的示意圖。如第 5 圖所示，位置排序器 42 包含配置為環狀之複數(在此例中為 14)個轉換器(inverter)56 及一 NAND 閘 58，使得當一訊號通過 14 個轉換器 56 及 NAND 閘 58 時被轉換奇數次。當送到 OR 閘 60 的兩個輸入中任一者升高時，一訊號從位置排序器 42 開始。第一 OR 閘輸入 62 係從被執行之指令 52 的位元 i4 66(第 4

圖)而來。若位元 i4 高，則特定指令 52 為一 ALU 指令，而 i4 位元 66 為 1。當該 i4 位元為 1，則該第一 OR 閘輸入 62 高，且位置排序器 42 被觸發以開始一脈衝，其將會導致次一指令 52 的執行。

當位置排序器 42 被觸發，不論其是由第一 OR 閘輸入 62 升高或是由第二 OR 閘輸入 64 升高(詳細如後述)所引起，則一訊號會傳過該位置排序器 42 兩次，每次都會在位置排序器輸出 68 產生一輸出。當該訊號第一次通過位置排序器輸出 68，其值為低，當該訊號第二次通過位置排序器輸出 68，其值為高。位置排序器輸出 68 之較寬的輸出係提供給脈衝產生器 70(方塊圖中未顯示)，其產生一窄時脈脈衝(narrow timing pulse)作為其輸出。熟悉該領域者應知，要準確開始運算器 12 的操作需要有窄時脈脈衝。

當被執行之該特定指令 52 為一讀取或寫入指令，其任何其他指令，並不希望被執行之指令 52 觸發次一指令 52 的立刻執行，則 i4 為原 66 為 0(低)及該第一 OR 閘輸入 62 因此亦為低。熟悉該領域者應知，一裝置(如運算器 12)之事件的時間是很重要的，且此並無例外。在檢視位置排序器 42 時，熟悉該領域者應知，OR 閘 60 的輸出必須維持高，直到該訊號通過 NAND 閘 58 以開始該環之第 2 處理(lap)。之後，在該第 2 處理中，OR 閘 60 的輸出會降低，以防止該電路有不必要的繼續振盪。

由上述可知，當 i4 位元 66 為 0，則位置排序器 42 不會被觸發-假設第二 OR 閘輸入 66 不高(細節後述)。

如上所述，每一指令 52 的 i4 位元 66 係依據該指令是

否為讀取型或寫入型指令而定，相對於該指令為不需要輸入或輸出者。指令 52 中剩下的位元 50 提供用於該指令之該特定操作碼的剩下部分。當其為讀取型或寫入型指令時，該位元中一或多個會被用於指定資料從運算器 12 中之何處讀取，或寫入何處。在本發明本實施例中，欲寫入之資料總是來自 T 寄存器 44(在資料堆疊 34 之頂部)，但是資料也可以選擇性地讀到 T 寄存器 44 或指令區域 30，並從那裡執行之。其係由於，在本實施例中，資料或指令可以藉由所述的方法傳送，而指令因此可以直接從資料匯流排 16 執行。

位元 50 中一或多個可以被用於指定通訊埠 38 中哪一者(若有的話)被設定為要讀取或寫入。此一後續操作係藉由一或多個位元來指定一寄存器 40(如 A 寄存器 40a、B 寄存器 b 等)以選擇性執行。在此例中，寄存器 40 可以事先載入資料，其具有一位元以對應於每一個通訊埠 38(並且，任何運算器 12 可能會與之通訊的其他可能的單元，例如一記憶體(RAM 24、ROM 26)、一個外部通訊埠 39 等)。例如在寄存器 40 中 4 個位元的每一者可以對應於上通訊埠 38a、右通訊埠 38b、左通訊埠 38c、或下通訊埠 38d 中每一者。在此，當這些位元位置上出現一個 1 時，就可以設定要透過對應的通訊埠 38 來執行通訊。如上所述，在本實施例中，可以預期到在單一指令中一個讀取操作碼可以設定多於一個通訊埠 38 來進行通訊，但是，雖然是可能的，並不會預期在單一指令中一寫入操作碼設定多於一個通訊埠 38 來進行通訊。

後述的實施例假設一通訊，其中運算器 12e 欲寫入運算器 12c，不過此例可適用於任何相鄰運算器 12 之間的通訊。當一寫入指令在寫入運算器 12e 中執行時，該被選取的寫入線 20(在此例中，在運算器 12e 和 12c 之間的寫入線 20)被高設，若該對應的讀取線 18 已經為高，則立刻透過該被選取的通訊埠 38 從該被選取的位置傳送資料。或者，若對應的讀取線 18 並未被高設，則運算器 12e 只是暫停運作，直到對應的讀取線 18 變高。當有一讀取型或寫入型指令時，該運算器 12a 停止(或者，更正確地說，不使進一步的操作開始)的機制在前面已經討論過。簡單地說，指令 52 的操作碼在位元位置 i4 66 係為 0，且 OR 閘 60 的第一 OR 閘輸入 62 為低，且位置排序器 42 未被觸發產生一啟動脈衝。

至於當讀取型或寫入型指令完成時，運算器 12e 的操作如何恢復，其機制如後：當運算器 12e 和 12c 之間的讀取線 18 和對應的寫入線 20 為高，則讀取線 18 和寫入線 20 都會由使其保持為高的運算器 12 分別釋出。(在此例中，傳送端運算器 12e 保持讀取線 18 為高，而接收端運算器 12c 保持寫入線 20 為高)繼之，接收端運算器 12c 會將讀取線 18 及寫入線 20 拉低。在實際運用中，接收端運算器 12c 會試圖在傳送端運算器 12e 釋出讀取線 18 之前，將讀取線 18 和寫入線 20 拉低。但是，因為讀取線 18 和寫入線 20 被拉高，並只有微弱地維持(拴鎖)為低，所以直到讀取線 18 或寫入線 20 由將其保持為高的運算器 12 釋出，任何要拉低讀取線 18 或寫入線 20 的嘗試實際上都不會成功。

當資料匯流排 16 中的讀取線 18 和寫入線 20 都被拉低，即為一通知狀態。在通知狀態下，運算器 12e 和 12c 都會將其內部的通知線(knowledge line)72 高設。如第 5 圖所示，通知線 72 提供第二 OR 閘輸入 64。因為到 OR 閘 60 之輸入 62 或 64 的輸入會使得 OR 閘 60 的輸出變高，此將會以如前所述之方式開始位置排序器 42 的運作，而使得在指令文字 48 之次一位置 54 的指令 52 被執行。直到次一指令 52 被解碼，通知線 72 維持為高，以防止假的位址到達位址匯流排。

在任何一個當被執行的指令 52 是在指令文字 48 之位置三的位置的狀況下，運算器 12 會擷取下一個等候中的 18 位元指令文字 48，當然，除非位元 i4 66 為 0，或者，除非是在位置三的指令就是次一指令，其細節將於後述。

在實際運用中，本發明機制包含用以事先擷取指令的方法及裝置，其使得該擷取可以在指令文字 48 中所有指令 52 執行完畢之前進行。但是，此並非本發明必要特徵。

上述例子已被清楚敘述，其中運算器 12e 寫入運算器 12c。由上述可知，不論運算器 12e 先試圖要寫到運算器 12c，或是運算器 12c 先試圖要從運算器 12e 讀取，其操作基本上是相同的。該操作直到運算器 12e 和 12c 兩者準備好才能完成，且不論運算器 12e 或 12c 先準備好，也只是先進入睡眠狀態，直到另一運算器 12e 或 12c 完成該傳送。上述程序之另一種解釋方式為，事實上，當其執行該寫入及讀取指令時，傳送端運算器 12e 和接收端運算器 12c 都分別進入睡眠，但是在當讀取線 18 和寫入線 20 都高時，

最後一個進入該交換運作的，幾乎立刻又被喚醒，使得第一個開始該交換運作的運算器 12 可以維持在睡眠狀態，直到第二個運算器 12 準備好完成該程序。

發明人相信要達成在裝置之間有效率的非同步通訊的關鍵，在於某種通知訊號或狀態。在習知技術中，在裝置之間大部分的通訊係已被定時(clocked)，且對於傳送端裝置，沒有直接的方法使其知道接收端裝置已經正確接收該資料。類似如檢查碼(checksum)操作的方法可以被用於確認資料被正確接收，但是傳送端裝置並無直接的訊息可知該操作已經完成。如同在此所述，本發明方法提供必須的通知狀態，其允許(或至少實現)裝置之間的非同步通訊。更且，該通知狀態也使得一個或多個裝置可以進入睡眠狀態，直到該通知狀態產生。當然，通知狀態可以藉由在運算器 12 之間(透過連結資料匯流排 16 或透過獨立的訊號線)傳送的一獨立訊號，在運算器 12 之間通訊，如此，一通知訊號可以包含在本發明範圍中。但是，依據本發明實施例可以知道，此係為更加經濟的作法，該用於通知的方法不需要任何額外的訊號、時脈週期、時脈脈衝、或任何此類的超過已描述之資源來實現該通訊。

由於指令文字 48 可以包含 4 個指令 52，並由於依據本發明，一個完整的指令文字 48 可以一次在運算器 12 之間傳送，此提供一個絕好的機會，以在一操作中傳送一非常小的城市。例如，大部分的小型「For/Next」迴圈可以在單一個指令文字 48 中實現。第 6 圖顯示微迴圈 100 之示意圖。微迴圈 100 不像其他習知技術迴圈，其具有一個 FOR

指令 102 及 NEXT 指令 104。因為一個指令文字 48(第 4 圖)包含至多 4 個指令 52，一個指令文字 48 可以在單一個指令文字 48 中包含 3 個操作指令 106。操作指令 106 基本上可以是程式設計者想要包含在微迴圈 100 中的任何可得的指令。可能從一運算器 12 傳送到另一者的微迴圈 100 的典型例子，係為一組用於從第二運算器 12 之 RAM 24 讀取或寫入的指令，使得第一運算器 12 可以借用可得的 RAM 24 容量。

FOR 指令 102 將一值推入代表所欲重複次數的回歸堆疊 28 中。亦即，位於資料堆疊 34 頂部 T 寄存器 44 中之值被推入回歸堆疊 28 之 R 寄存器 29。FOR 指令 102，雖然常位於指令文字 48 的位置三 54d，但事實上也可以位於認為位置 54。當 FOR 指令 102 不位於位置三 54d 時，則在指令文字 48 中其他的指令 52 可以在進入微迴圈 100 之前執行，其通常為次一個載入的指令文字 48。

依據本發明實施例，第 6 圖中所示之 NEXT 指令 104 為一特定種類的 NEXT 指令 104。因為其係位於位置三 54d(第 4 圖)。依據本發明實施例，假設在正常 NEXT 指令(圖未顯示)後面的指令文字 48 中的所有資料為一位址(亦即 for/next 迴圈開始的位址)。不論 NEXT 指令 104 係位於 4 個位置 54 中的哪一者，其操作碼是一樣的(有一明顯的例外為，開頭的 2 個數字為假設為其係位於位置三 54d，而不是明確寫明，如前所述)。但是，當 NEXT 指令 104 是位於位置三 54d 時，由於可以沒有位址資料接在 NEXT 指令 104 後面，所以也可以假設在位置三 54d 的 NEXT 指令 104

為一 MICRO-NEXT 指令 104a。該 MICRO-NEXT 指令 104a 使用第一指令 52 的位址，其位於其所屬之同一指令文字 48 的位置零 54a，作為回復的位址。MICRO-NEXT 指令 104a 也從 R 寄存器 29 中取得數值(其先由 FOR 指令 102 推至該處)，將其減少 1，並將之傳回 R 寄存器 29。當 R 寄存器 29 中的數值到達一預定值(例如 0)，則 MICRO-NEXT 指令將載入指令文字 48，並繼續如上述之程序。然而，當 MICRO-NEXT 指令 104a 從 R 寄存器 29 讀取一大於該預定值之數值時，他就會於其本身的指令文字 48 之位置零 54a 重開始運作，並執行其中位於位置零到位置三之該三個指令 52。亦即，依據本發明實施例，MICRO-NEXT 指令 104a 總是執行三個操作指令 106。因為，在某些狀況下，不想要使用所有的三個可得的指令 52，則需要一「非操作」指令去填滿一個或兩個位置 54。

需注意的是，微迴圈 100 可以在單一運算器 12 中被完全使用。事實上，整組可用的機械語言指令可以用作操作只另 106，而微迴圈 100 之應用及使用僅係由程式設計者之想像所限。然而，當在單一指令文字 48 中執行一完整的微迴圈 100 的能力，和使得一運算器 12 將指令文字 48 傳送到一鄰近運算器 12 以直接從資料匯流排 16 執行指令 52 的能力結合時，就提供了一有力的工具，使得運算器 12 使用其鄰近裝置之資源。

該微迴圈 100，都包含於該單一指令文字 48 中，可以如在此所述，在運算器 12 之間傳送，且其可以直接從接收端運算器 12 的通訊埠 38 被執行，如同前述之其他任何包

含於指令文字 48 中的指令組。雖然此種微迴圈 100 有多種用途，典型的用途為，當一運算器 12 將一些資料存入一鄰近運算器 12 的記憶體中時。例如，其可以先傳送一指令至該鄰近運算器，告知他要將一輸入的資料文字儲存於一特定記憶位址中，繼之遞增該位址，繼之重複預定次數的迴圈(欲傳送之資料文字的數目)。為了讀回該資料，該第一運算器可以使用較小的微迴圈，僅指示該第二運算器(在此為用於儲存者)將該儲存的資料寫回該第一運算器。

藉由使用微迴圈 100 結構和在此所述之直接執行的技術，當資料儲存的需求超出每個個別的運算器 12 中內建的小容量時，一運算器 12 可以使用一不然就在休息的鄰近運算器 12 來儲存多出的資料。雖然本實施例係以資料儲存為例，同樣的技術也可以同樣應用於讓運算器 12 使其鄰近裝置分享其運算資源，其係藉由產生一微迴圈 100，使得其他的運算器 12 執行一些操作，儲存其結果，並重複執行預定次數。由此可知，該創新的微迴圈 100 結構在未來可以有幾乎無限多種使用方式。

如前所述，依據本發明實施例，資料或指令可以以在此所述之方式來通訊，而且，指令因此可以幾乎是直接從資料匯流排 16 執行。亦即，無須將指令儲存於 RAM 24，在將之於執行前呼叫出來。相反地，依據本發明之本技術特徵，由通訊埠 38 接收的一指令文字 48 被處理的方式，並未不同於將其從 RAM 24 或 ROM 26 中呼叫出來。雖然此一相同處已於前文述及，關於運算器 12 之所述的操作，後文更詳細描述如何擷取及使用指令文字 48，以協助瞭解本

發明。

可得的機械語言指令之一係為擷取(FETCH)指令。FETCH指令使用在A寄存器40a的位址以決定要從何處擷取一18位元文字。當然，該程式必須已經提供用以將正確的位址置於A寄存器40a。如前所述，A寄存器40a係為一18位元寄存器，使得以有足夠範圍的位址資料，而能夠區分一擷取動作能夠發生的任何一個可能的資源。亦即，有一範圍的位址被指定給ROM，一不同範圍的位址指定給RAM，對於每一個通訊埠38及外部I/O通訊埠39指定一特定的位址。一FETCH指令總是將其擷取之該18位元置於T寄存器44。

相反地，如前所述，可執行指令(相對於資料)係暫時儲存於指令寄存器30a。並無特別的命令用來擷取一18位元指令文字48到該指令寄存器30a中。而是，當指令寄存器30a中沒有留下可執行指令時，運算器會自動擷取次一個指令文字48。該次一個指令文字的位置係由程式計數器(P寄存器c)決定。P寄存器c常為自動遞增，就像是當要從RAM 24或ROM 26中擷取一列指令文字48時一樣。但時，此一通則有數個例外。例如，一JUMP或CALL指令會使得在該JUMP或CALL指令之後，將由目前載入的指令文字48的剩下部分所指定的位址載入該P寄存器c，而不是遞增。當對應於一或多個通訊埠38的位址載入P寄存器c，則次一指令文字48從通訊埠38載入到指令寄存器30a。當一指令文字48才從通訊埠38被擷取到指令寄存器30a中，P寄存器c也不遞增。而是，其會繼續保持同樣的埠位址，

直到一特定的 JUMP 或 CALL 指令被執行以改變 P 寄存器 c。亦即，一旦運算器 12 被告知要從通訊埠 38 搜尋其次一指令，則其持續從同樣的通訊埠 38 搜尋指令，直到其被告知要從別處搜尋次一個指令文字 48，例如回到記憶體(RAM 24 或 ROM 26)。

如上所述，運算器 12 知道，當在目前的指令文字 48 中沒有留下可執行指令，則被擷取的次一 18 位元要放置在指令寄存器 30a。預設是，在 JUMP 或 CALL 指令之後(或在特定的其他在此未指定的指令之後)時，則在目前的指令文字 48 中沒有留下可執行指令，因為，依據定義，在依 JUMP 或 CALL 指令之後的該留下的 18 位元指令文字，係專用於 JUMP 或 CALL 指令所指定之該位址。另一種敘述方式為，上述程序在許多方面都是獨特的，包含但不限於，一 JUMP 或 CALL 指令可以選擇性地為一通訊埠 38，而非僅是一記憶體位址等的事實。

應該記得的是，如前所述，運算器 12 可以從一個通訊埠 38 或一群通訊埠 38 中任何一者搜尋其次一指令。因此，提供的位址係對應於通訊埠 38 的各種組合。例如，當一運算器被告知要從一群通訊埠 38 中擷取一指令時，則其會接受從該選取的通訊埠 38 中任何一者而來之第一個可得的指令文字 48。如果沒有鄰近的運算器 12 已經準備要寫入該通訊埠 38 中任一者，則所討論之該運算器 12 會進入睡眠，如前所述，直到一個鄰近裝置寫入該選取的通訊埠 38。

第 7 圖顯示依據上述之直接執行方法 120 之一例之流程圖。如前所述，當在指令寄存器 30a 中沒有留下可執行

指令時，一個正常操作流程會開始。此時，運算器 12 會依據「擷取文字(fetch word)操作 122」，擷取另一指令文字(需注意該用詞擷取(fetch)，在此係為一般含意，其中並無使用真正的 FETCH 指令)。該操作係依據該 P 寄存器 c 中的位址而完成，如第 7 圖中之一「位址？」決定操作 124 所示。若在 P 寄存器 c 中的位址為一 RAM 24 或 ROM 26 位址，則該次一指令文字 48 會在一「從記憶體擷取操作」中，從該指定的記憶體位置擷取。另一方面，如果在 P 寄存器 c 中的位址為一通訊埠 38 或數個通訊埠 38(非記憶體位址)，則該次一指令文字 48 會在「從埠擷取操作 128」中，從該指定的埠位置擷取。不論是哪一者，被擷取的該指令文字 48 都是在「擷取指令文字」操作 130 中，被放置在指令寄存器 30a 中。在一「執行指令文字」操作 132 中，在指令文字 48 之位置 54 之該指令依序被完成，如前所述。

在一「跳躍(jumping)？」決定操作 134 中，決定是否在該指令文字 48 中有一操作為 JUMP 指令，或其他會將操作導出前述該繼續的正常流程的指令。若是，則在指令文字 48 之 JUMP(或其他此類)指令之後的該位址，係在一「載入 P 寄存器」操作 136 中，被提供給 P 寄存器 c，而該序列再度於「擷取文字(fetch word)」操作 122 開始，如第 7 圖所示。若否，則該次一動作係依據該最後一個指令是從通訊埠 38 或一記憶體位址擷取而定，如「埠位址？」決定操作 138 所示。若該最後一個被擷取的指令係來自通訊埠 38，則不會對 P 寄存器 30a 做改變，且該序列再度開始於「擷取文字(fetch word)操作 122」另一方面，若被擷

取的最後一個指令是來自一記憶體位址(RAM 24 或 ROM 26)，則在「擷取文字(fetch word)操作 122」完成之前，在 P 寄存器 30a 中的位址被遞增，如第 7 圖中「遞增 P 寄存器操作 140」所示。

上述敘述並非用以界定實際的操作步驟。而是，其係為依據本發明實施例執行之各種決定和所導致之操作的示意圖。事實上，該流程圖不應被解讀為描述及顯示的每一操作必須要一分離的不同之序列步驟。事實上，在第 7 圖之流程圖中敘述之操作中有多個在實際上會幾乎是被同時完成。

第 8 圖顯示依據本發明實施例警示一運算器之方法的流程圖。如前所述，上述實施例中的運算器 12 當其等候輸入時會進入睡眠。此一輸入可以是從一鄰近運算器 12 而來，如同第 1～5 圖相關的實施例敘述所言。或者，亦如前所述，緊鄰晶粒 14 邊緣且具有通訊埠 38 之該運算器 12 可以具有額外的電路，其可以設計於該運算器 12 中或在運算器 12 外部且與其相接，以使得此一通訊埠 38 動作如外部通訊埠 39。不論哪一者，該創新的組合可以提供額外的益處，使得睡眠中的運算器 12 可以做好準備並準備好被喚醒以於接收輸入時投入某些預設的活動。因此，本發明亦提供對於使用中斷來處理輸入之一種替代方案，不論此種輸入是來自一外部輸入裝置，或來自陣列 10 中的另一個運算器 12。

不使運算器 12 停止其正在執行的工作以處理一中斷，在此所述之本發明使得一運算器 12 處於一「睡眠但警

覺」的狀態，如上所述。因此，可以指定一或多台運算器12去接收並反應特定輸入。雖然有多種方法可以使用此一技術特徵，參見第8圖以說明一例用於解釋此「運算器警示方法」之一種，其係以數字150標示。如第8圖所示，在一「進入警覺狀態」操作152中，使得一運算器12進入睡眠，而使其等候從一鄰近運算器12傳來的輸入，或多於一個(至多4個)鄰近的運算器12，或者，若為邊緣運算器12則為一外部輸入，或為外部輸入及/或鄰近運算器12之輸入的組合。如前所述，一運算器12可以進入睡眠來等待一讀取或寫入操作的完成。如前所述，當該運算器12被用來等待某些可能的輸入時，自然可以假設該等候的運算器已將其讀取線18高設，等候從一鄰近或外部來源傳來的一寫入。事實上，已知此為通常的狀況。然而，等候的運算器12已將其寫入線20高設，並因此其可以當一鄰近或外部來源從其讀取時被喚醒，係在本發明範圍中。

在一「喚醒」操作154中，因為鄰近的運算器12或外部裝置39已完成其等候中的交換，而使得睡眠中的運算器12重新開始操作。若等候中的交換係為接收欲執行之指令文字48，則運算器12會繼續執行其中的指令。若等候中的該交換為接收資料，則該運算器12繼續執行次一排序中的指令，其可以是在目前指令文字48之次一位置54的指令，或是將載入的次一指令文字48，且該次一指令係位於次一指令文字48的位置零。在任一狀況中，當依據所述方式使用時，則次一指令將開始一或多指令之一序列指令，以處理剛接收的該輸入。處理此輸入之選擇可以包含在內

部反應以執行某些預定的功能，和一或多個在陣列 10 中的其他運算器 12 通訊，或甚至忽略該輸入(如習知的中斷可以在特定狀況下被忽略)。該選擇係顯示於第 8 圖之「處理輸入」操作 156。需注意的是，在某些狀況下，輸入的內容可能不重要。例如，在某些狀況下，關注焦點其可能僅是外部裝置嘗試要通訊之此一事實。

若運算器 12 被指定運作為一「警覺」運算器的工作，則依據第 8 圖所示之方式，其通常會回到「睡眠但警覺」的狀態，如第 8 圖所示。然而，亦可以指定運算器 12 一些其他的任務，例如當其不再需要監視其正在監視的特定輸入時，或當將該任務轉移給該陣列中其他運算器 12 時。

熟悉該領域技藝之人士可知，上述之操作模式較之於傳統的中斷之使用，為更有效的替代方案。當運算器 12 將其一或多個讀取線 18(或寫入線 20)高設時，其可以稱之為處於「警覺」狀態。在警覺狀態中，該運算器 12 準備好可以立刻執行任藉由對應於高社之讀取線 18 的資料匯流排 16 何傳送給他的指令，或者，處理透過資料匯流排 16 傳送的資料。當可使用運算器 12 之陣列時，可以在任何時間使用一或多個為上述的警覺狀態，使得任何預定的輸入可以觸發其進入執行。此係較使用傳統的的中斷技術來達成使運算器「注意」的技術，因為一中斷會使得運算器必須對應於該中斷要求儲存特定資料，下載特定資料等。但是，依據本發明，一運算器可以被置於警覺狀態，並專用於等候所欲之輸入，使得不浪費一指令期間於開始執行開輸入所觸發之該指令。而且，注意到，在本實施例中，處於警

覺狀態之運算器事實上係處於「睡眠但警覺」的狀態，亦即，其幾乎不使用電力而處於「睡眠」狀態，但其處於「警覺」狀態而可以由一輸入立刻被觸發進入執行。然而，即使一運算器並沒有睡眠，該警覺狀態可以在該運算器中實現，此亦在本發明範圍中。上述警覺狀態可以使用於任何可以使用傳統的中斷技術(硬體中斷或軟體中斷)之情況。

第 9 圖顯示另一例之運算器警示方法 150a。此僅為一例，其中監視運算器 12f(第 1 圖)和另一指令其他任務之運算器 12g(第 1 圖)的互動係為所欲或必須。如第 9 圖所示，有兩個獨立的流程圖，分別針對運算器 12f 及運算器 12g。此係指出本發明合作的共同運算方法的本質，其中運算器 12 中每一者具有其本身指定任務，其一般為獨立執行，除非當互動如在所所述之被完成時。

針對運算器 12f，「進入警覺狀態」操作 152、「喚醒」操作 154 及「處理輸入」操作均依據前述運算器警示方法 150 之第一例而完成。但是，因為此例預期有可能要進行運算器 12f 和運算器 12g 之間的通訊，所以在「處理輸入」操作 156 之後，運算器 12f 進入「傳送資料？」決定操作 158，其中依據其程式，決定是否剛才接收的輸入需要另一運算器 12g 的注意。若否，則運算器 12f 回到警覺狀態，若其他前述的狀態。若是，則運算器 12f 如前述般，在「傳送給其他」操作 160 中開始和運算器 12g 通訊。需注意的是，依據程式設計者的選擇，運算器 12f 可以傳送指令，例如其內部對應於外部裝置 82 所產生的指令，又例如其可以從該外部裝置 82 接收。或者，運算器 12f 可以傳送資料

給運算器 12g，而此一資料可以為運算器 12f 內部產生，或從外部裝置傳來。另一種可能是，在某些情況下，當運算器 12f 從外部裝置 82 接收一輸入時，可以嘗試要從運算器 12g 讀取。該程式設計者可得到所有這些機會。

同時，運算器 12g 通常執行程式碼以完成其被指定之主要任務，不管那是什麼，如「執行主要功能」操作 162 所示。然而，若該程式設計者以決定需要偶而執行運算器 12f 和運算器 12g 之間的互動，則程式設計者使得運算器 12g 偶而停下來確認是否他的鄰近裝置中有一或多個嘗試要通訊，如「尋找輸入」操作 166 所示。如「輸入資料？」決定操作 158 所示，若有一通訊則等待(例如，若運算器 12f 已經開始一寫入到運算器 12g)。若已有一通訊開始(是)，則運算器 12g 會在「從其他裝置接收」操作 170 中完成該通訊，如前所述。若否，則運算器 12g 會回到其主要功能的執行 162，如第 9 圖所示。在「從其他裝置接收」操作 170 之後，該運算器 12g 在「處理輸入」操作 172 中處理該輸入。如上述，程式設計者可以使得運算器 12g 等待指令輸入，其中，運算器 12g 可以如前述般執行該指令。或者，運算器 12g 可以等待資料以處理之。

在第 9 圖之例中，在「處理輸入」操作 172 之後，運算器 12g 回到其主要功能之完成(亦即，其回到「執行主要功能」操作 162)。但是，也有可能出現更複雜的例子。例如，該程式可以為，從運算器 12f 輸入的特定輸入使其中止其原先被指定的主要功能，並開始一新的功能，或者，其可以暫時停止並等待其他輸入。熟悉該領域技藝者可以

知道，在此執行之多種可能僅受限於程式設計者的想像。

需注意的是，依據在此所述之本發明的實施例，一既定的運算器 12 當其正執行一任務時，不需被中斷，其係因為其他運算器 12 被指定了可能需要一中斷的監視及處理輸入的任務。但是，亦需注意，忙於處理其他任務的運算器 12 也不能被打擾，除非且直到其程式使得他去詢問通訊埠 38 是否有輸入。因此，有時需要使得運算器 12 暫停以尋找其他輸入。應知，在此所述者係為運算方法之一例，其可以描述為「合作多工」，其中先前由單一處理器完成之工作以新且有趣的方式，在多個處理器之間被分割。

可以對本發明做多種變化，而無損於其價值或範圍。例如，在此以運算器 12 為例說明本發明，許多或所有的發明特徵係可以修改適用於其他運算器設計，其他種類的運算器陣列等。

同樣地，雖然在此主要以單晶片 14 之陣列 10 中的運算器 12 之間的通訊來描述本發明，同樣的原理和方法可以使用或修正以使用於完成其他裝置間的通訊，例如，在運算器 12 和其專用記憶體之間，或在陣列 10 之運算器 12 及一外部裝置之間。

在此已描述本發明運算器陣列 10、運算器 12、微迴圈 100、直接執行方法 120 及相關裝置、及運算器警示方法 150 之特定例子，可推知會有多種未見之對此之應用。事實上，本發明的優點之一，就是本發明方法和裝置可以適用於多種狀況。

雖然本發明已以較佳實施例揭露如上，然其並非用以

限定本發明，任何熟習此技藝者，在不脫離本發明之精神和範圍內，當可作些許之更動與潤飾，因此本發明之保護範圍當視後附之申請專利範圍所界定者為準。

產業應用性

本發明運算器陣列、運算器 12、微迴圈 100、直接執行方法 120 及相關裝置、及運算器警示方法 150 係適用於多種運算器應用。預期其在需要特定運算電力，且電力消耗和產熱至關重要的應用中特別有用。

如前所述，本發明的應用為，在速度和可變性方面，提升在一陣列之運算器間資料和資源的分享。而且，依據所描述的方法和裝置，同時也增進在一運算器陣列和其他裝置之間的通訊。

由於運算器陣列 10、運算器 12、微迴圈 100、直接執行方法 120 及相關裝置、及運算器警示方法 150 可以被產生並整合於現存的工作，輸入/輸出裝置等，並因為其提供在此所述之優點，在業界廣被接受是可期的。因此，本發明的有用性和產業利用性在範圍和長時間沿用方面，是可期的。

【圖式簡單說明】

第 1 圖顯示依據本發明實施例的運算器陣列之示意圖。

第 2 圖顯示第 1 圖所示之運算器的部分之細部及第 1 圖中連結資料匯流排的詳細圖。

第 3 圖顯示如第 1 圖及第 2 圖之運算器的一般配置的方塊圖。

第 4 圖係顯示依據本發明實施例之一指令文字的示意圖。

第 5 圖顯示如第 3 圖所示之位置排序器(slot sequencer)42 的示意圖。

第 6 圖顯示依據本發明實施例之微迴圈之流程圖。

第 7 圖顯示依據本發明實施例之從一埠執行指令的方法之流程圖。

第 8 圖顯示依據本發明實施例警示一運算器之方法的流程圖。

第 9 圖顯示喚醒一不活動處理器並將輸入資料從該被喚醒的處理器傳送到一欲處理該輸入資料之執行處理器的方法流程圖。

【主要元件符號說明】

10~運算器陣列；
12~運算器；
14~晶粒；
16~資料匯流排；
18~讀取線；
20~寫入線；
22~資料線；
24~RAM；
26~ROM；
28~回歸堆疊；
28a~環狀寄存器陣列；
29R~寄存器；
30~指令區域；
30a~指令寄存器；
30b~操作碼寄存器；
32~ALU；
34~資料堆疊；
34a~環狀寄存器陣列；

36~解碼段落；　38~內部通訊埠；

38a~上方通訊埠；　38b~右方通訊埠；

38c~左方通訊埠；　38d~下方通訊埠；

39~外部輸入輸出埠；　40~寄存器；

40a~A 寄存器；　40b~B 寄存器；

40c~P 寄存器；　42~位置排序器；

44~T 寄存器；　46~S 寄存器；

48~指令文字；　50~位元；

54~位置；　54a~位置零；

54b~位置一；　54c~位置二；

54d~位置三；　56~轉換器；

58~NAND 閘；　60~OR 閘；

62~第一 OR 閘輸入；　64~第二 OR 閘輸入；

66~i4 位元；　68~位置排序器輸出；

70~脈衝產生器；　72~通知線；

80~介面電路；　82~外部裝置；

100~微迴圈；　102~FOR 指令；

104~NEXT 指令；　106~操作指令；

108~MICRO-NEXT 指令；　120~直接執行方法；

124~「位址？」決定操作；　128~「從埠擷取」操作；

150~運算器警示方法；　150a~運算器警示方法；

154~「喚醒」操作；　156~「處理輸入」操作；

160~「傳送給其他」操作；　166~「尋找輸入」操作；

168~「輸入」決定操作；　172~「處理輸入」操作；

162~「執行主要功能」操作；

126~「從記憶體擷取」操作；

130~「擷取指令文字」操作；

132~「執行指令文字」操作；

136~「載入P寄存器」操作；

138~「埠位址？」決定操作；

140~「遞增P寄存器」操作；

152~「進入警覺狀態」操作；

158~「輸入資料？」決定操作；

170~「從其他裝置接收」操作；

122~擷取文字(fetch word)操作；

134~「跳躍(jumping)？」決定操作。

五、中文發明摘要：

一運算器陣列(10)包括複數運算器(12)。該運算器(12)非同步地互相通訊，且該運算器(12)本身內部以一大致非同步的方式運作。當一運算器(12)欲與另一運算器通訊時，其轉入睡眠直到其他運算器(12)準備好要完成該交換，以節省電力並減少熱產生。該睡眠運算器(12)可以等候資料或指令(12)。若為指令，則睡眠運算器(12)可以等待以儲存該指令，或立刻執行該指令。在後者，當該指令被接收時，該指令被載入一指令寄存器(30a)，並從那裡被執行，而不先將該指令載入記憶體中。該指令可以包含一微迴圈(micro-loop)(100)，其可以重複執行一系列操作。在一應用中，該睡眠運算器(12)被一輸入喚醒，使得其可以開始執行一動作，不然該活動需要運作中運算器之一中斷。

六、英文發明摘要：

A computer array (10) has a plurality of computers (12). The computers (12) communicate with each other asynchronously, and the computers (12) themselves operate in a generally asynchronous manner internally. When one computer (12) attempts to communicate with another it goes to sleep until the other computer (12) is ready to complete the transaction, thereby saving power and reducing heat production. The sleeping computer (12) can be awaiting data or instructions (12). In the case of instructions, the sleeping computer (12) can be waiting to store the instructions or to immediately execute the instructions. In the later case, the instructions are placed in an instruction register (30a) when they are received and executed therefrom, without first placing the instructions first into memory. The instructions can include a micro-loop (100) which is capable of performing a series of operations repeatedly. In one application, the sleeping computer (12) is awakened by an input such that it commences an action that would otherwise have required an interrupt of an otherwise active computer.

十、申請專利範圍：

1.一種運算器，用以執行一系列指令，其特徵在於包含：

一指令寄存器(instruction register)，用以暫存一待執行指令群；以及

一程式指標(program counter)，用以儲存一位址，一指令群從該位址被擷取到該指令寄存器，

其中該程式指標之該位址可以為一記憶體位址或一寄存器位址。

2.如申請專利範圍第 1 項所述之運算器，其中在該程式指標中之該位址可以選擇性地指向複數寄存器。

3.如申請專利範圍第 1 項所述之運算器，其中該指令群包含一以上之指令。

4.如申請專利範圍第 1 項所述之運算器，其中當該指令寄存器中沒有待執行指令時，該運算器依據該程式指標中儲存的位址，擷取一指令群。

5.如申請專利範圍第 1 項所述之運算器，其中當該運算器從一記憶體中擷取一指令群時，該程式指標增額(incremented)；以及

當該運算器從一寄存器中擷取一指令群時，該程式指標未增額。

6.如申請專利範圍第 1 項所述之運算器，其中當該指令群中之一指令為跳躍指令時，該程式指標載入該跳躍指令指示之一位址。

7.如申請專利範圍第 6 項所述之運算器，其中該跳躍

指令指示之該位址跟隨在該指令群之跳躍指令之後。

8.一種用於在運算器中執行指令的方法，包括：

(a)從一位址擷取一指令群，其中該位址可以為一記憶體位置之位址或一埠之位址；

(b)將該指令群載入一指令寄存器以待執行；以及

(c)從該指令寄存器執行該指令群中至少一指令。

9.如申請專利範圍第 8 項所述之方法，其中該位址係從一程式指標中擷取出。

10.如申請專利範圍第 8 項所述之方法，其中當該指令群中所有指令均已被執行時，擷取另一指令群。

11.如申請專利範圍第 10 項所述之方法，其中當該位址為一記憶體位置之位址時，則在擷取該另一指令群之前，將該位址增額(incremented)；以及

當該位址為一埠之位址時，則在擷取該另一指令群之前，不將該位址增額。

12.如申請專利範圍第 8 項所述之方法，更包括：

其中該位址可以指定一個埠以上。

13.如申請專利範圍第 8 項所述之方法，更包括：

於步驟(c)中，執行該指令群中所有指令，或者，執行該指令群中之指令直到接到轉換指令；

其中該轉換指令係為一指令，其係用以從另一位置擷取一指令群。

14.如申請專利範圍第 13 項所述之方法，其中該轉換指令為一跳躍指令。

15.一種用於在運算器中處理資料區塊的方法，包括：

(a)從一位址擷取一資料區塊，其中該位址可以為一記憶體位置之位址或一埠之位址；

(b)若該資料區塊包含一指令群，則將該資料區塊載入一指令寄存器；以及

(c)若該資料區塊所包含之資料非為指令，則將該資料區塊提供至一資料堆疊(data stack)之頂。

16.如申請專利範圍第 15 項所述之方法，其中該指令群可以選擇性地同時包含資料。

17.如申請專利範圍第 15 項所述之方法，其中依據前一指令群之最後一項指令是否被執行，來決定該資料區塊係包含指令群或資料。

18.如申請專利範圍第 17 項所述之方法，其中當前一指令群之最後一項指令已被執行，則決定該資料區塊係包含指令群。

19.如申請專利範圍第 15 項所述之方法，其中該資料堆疊為一主要資料堆疊(primary data stack)。

20.如申請專利範圍第 15 項所述之方法，更包括：

若該資料區塊包含一指令群，則從該指令寄存器執行該指令群中之指令。

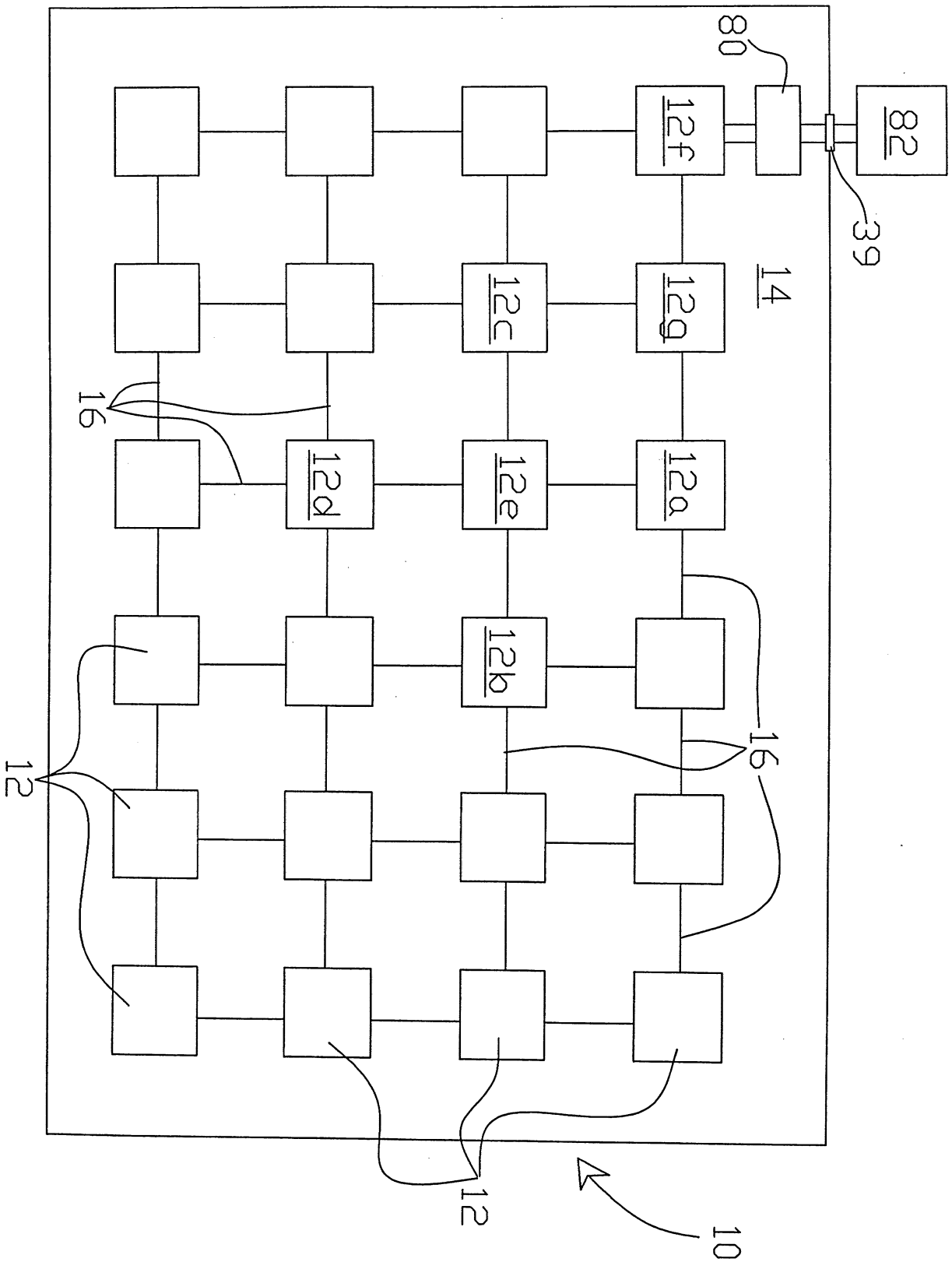
21.一種用以執行一系列指令之運算器，該運算器包括：

一指令寄存器，用以暫存一指令群；

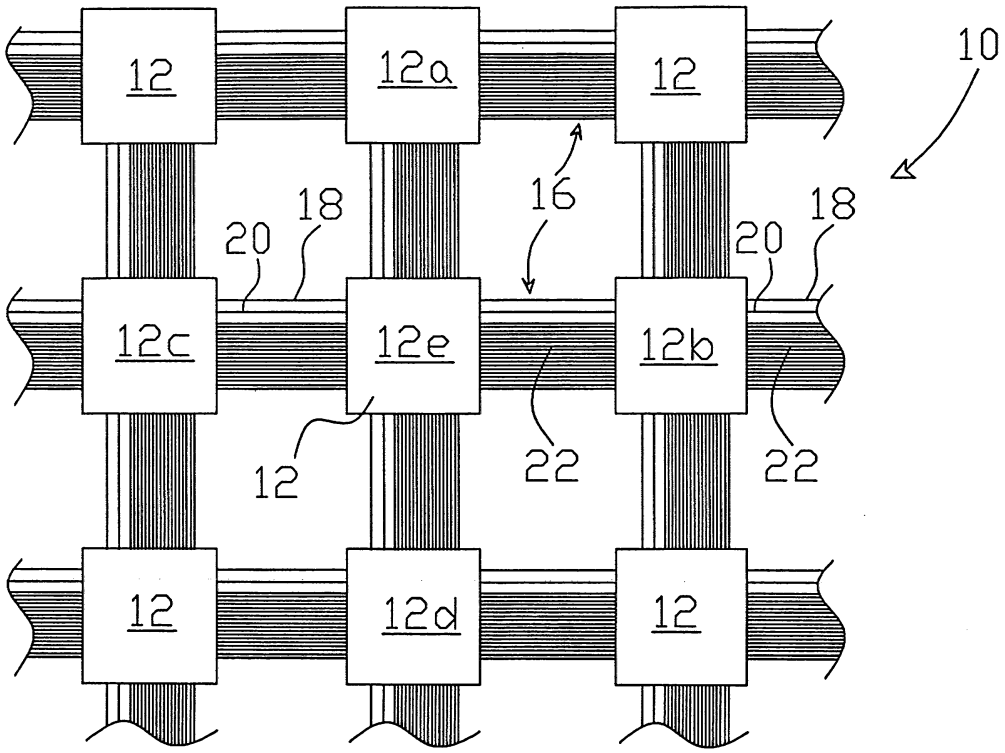
一邏輯單元，用以執行儲存於該指令寄存器中的指令；以及

一裝置，用以從一埠擷取該指令群到該指令寄存器。

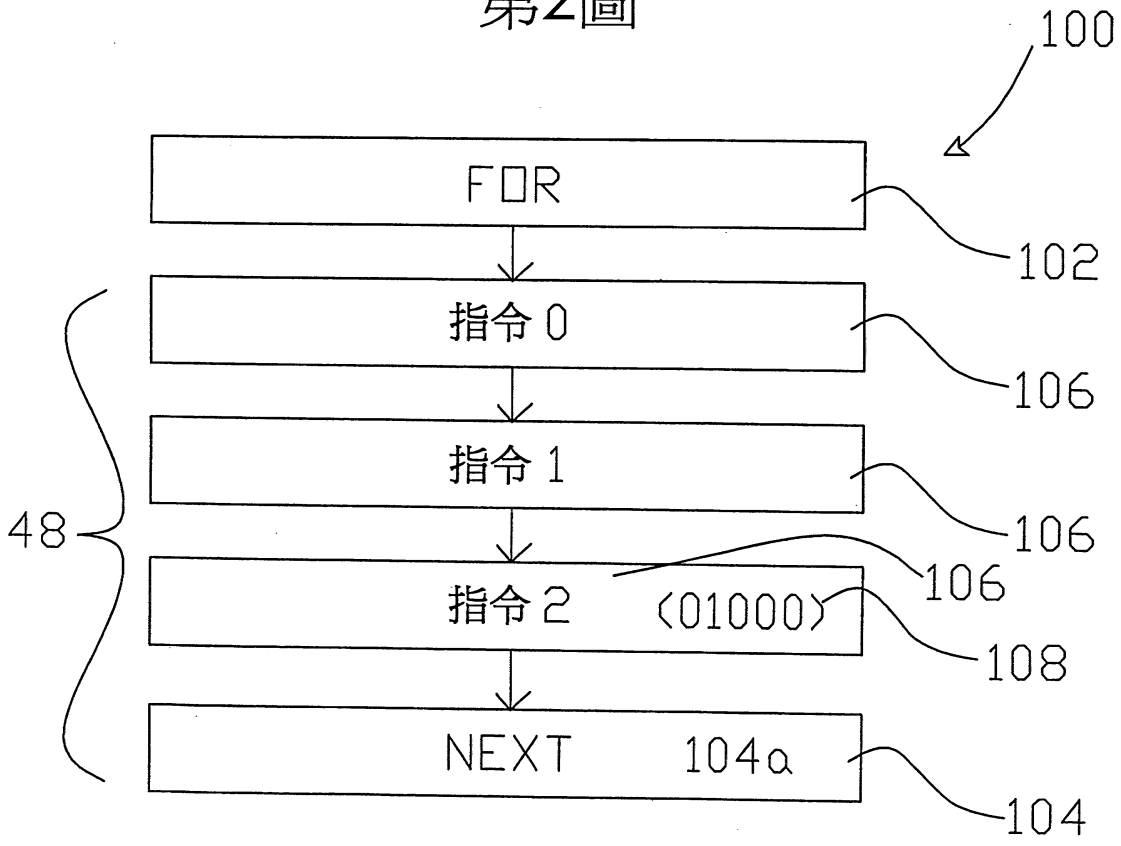
200809613



第1圖

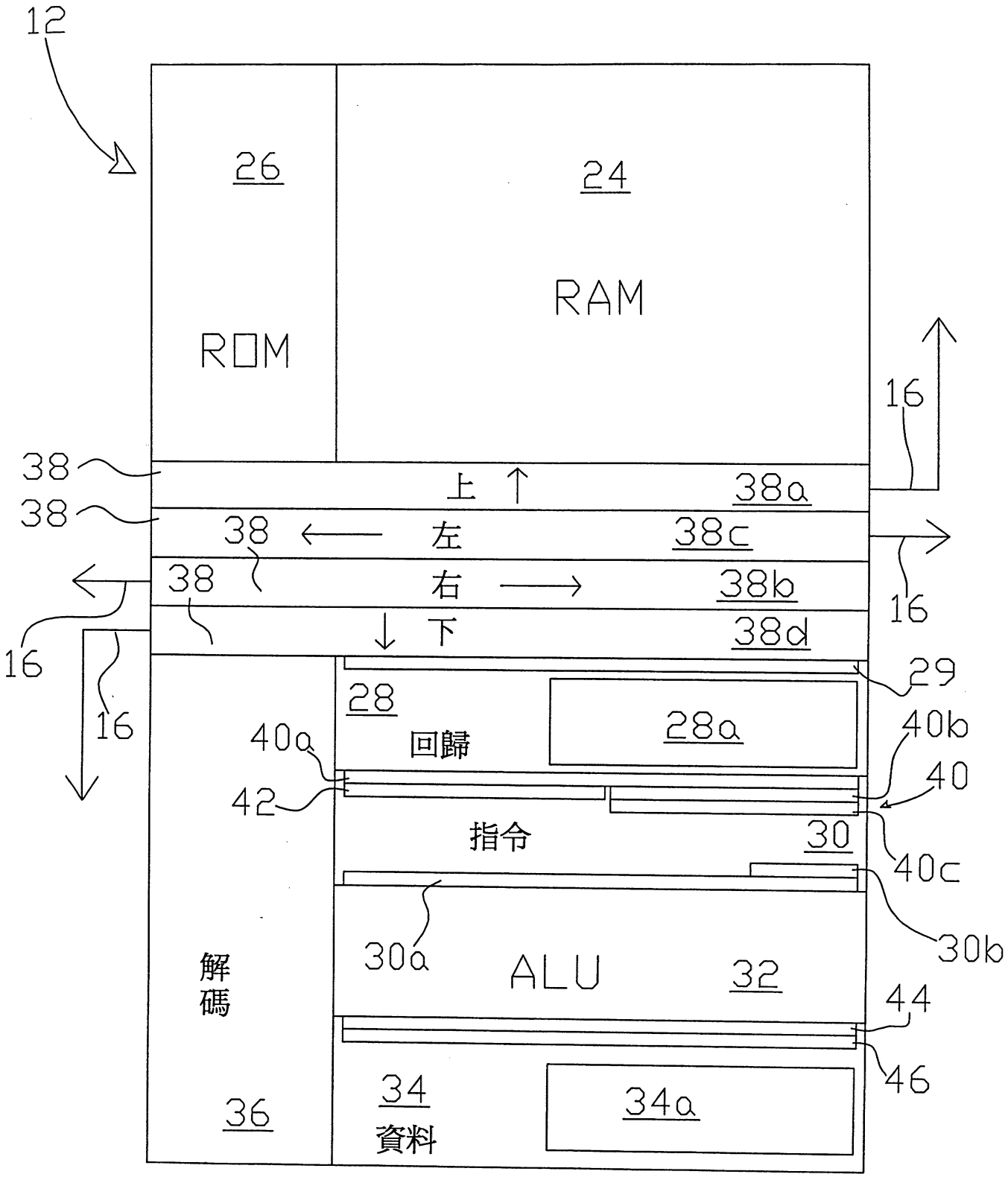

12
12a
12
10
16
18
20
18
20
12c
12e
12b
12
22
22
12
12d
12

第2圖


100
FOR
102
指令 0
106
指令 1
106
48
指令 2
106
(01000)
108
NEXT
104a
104

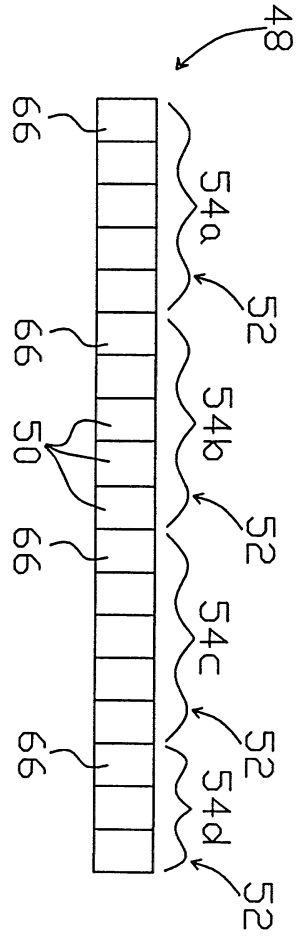
第6圖

200809613

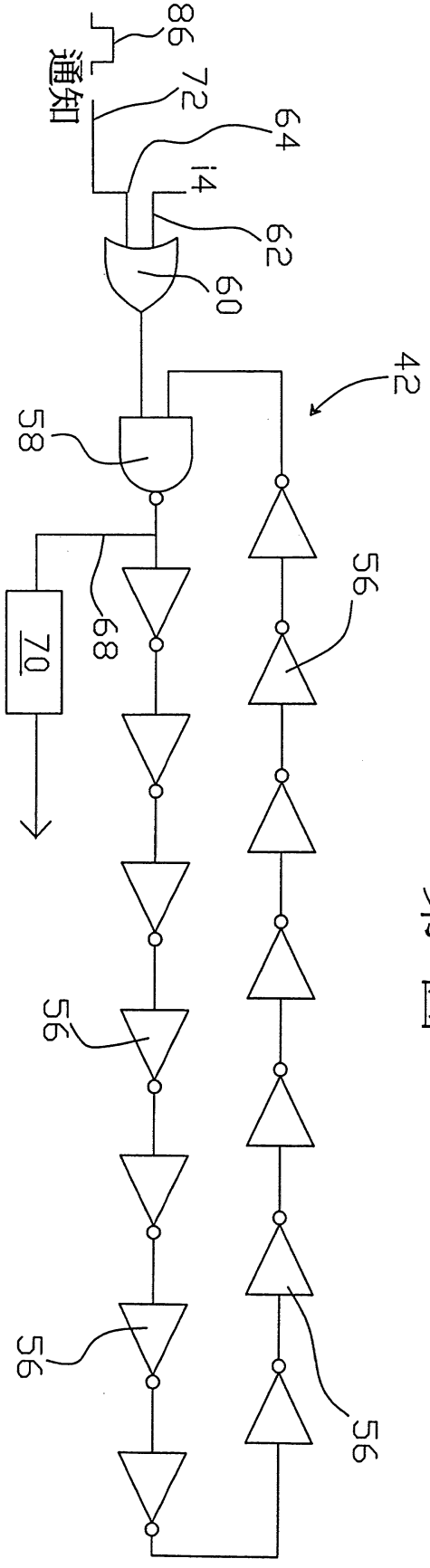


第3圖

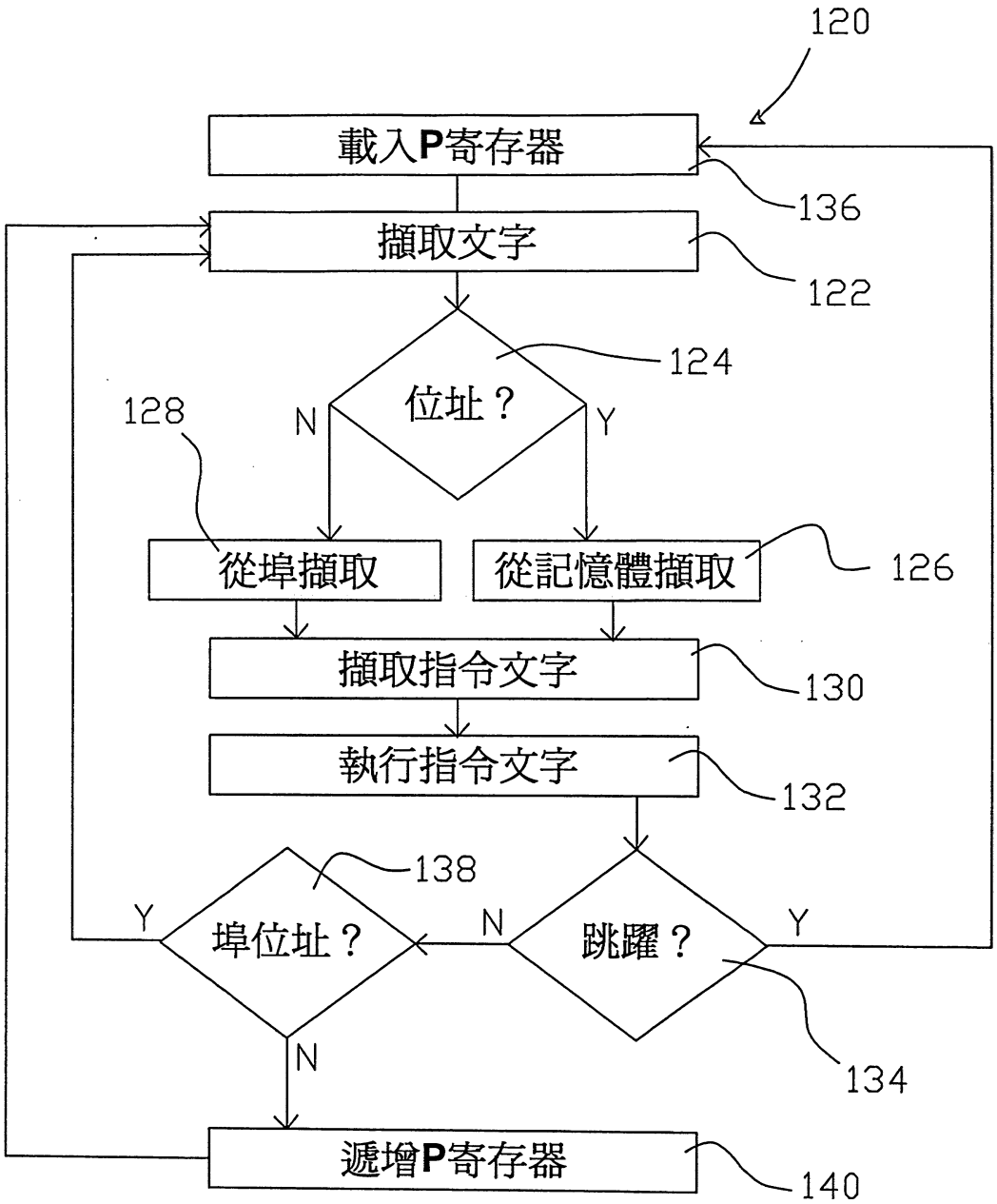
200809613



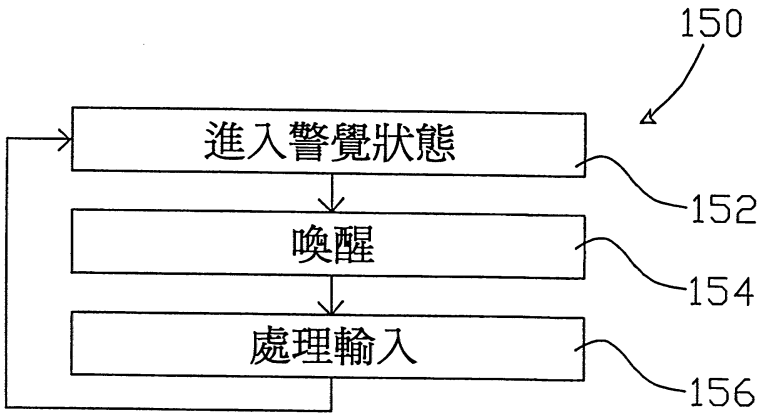
第4圖



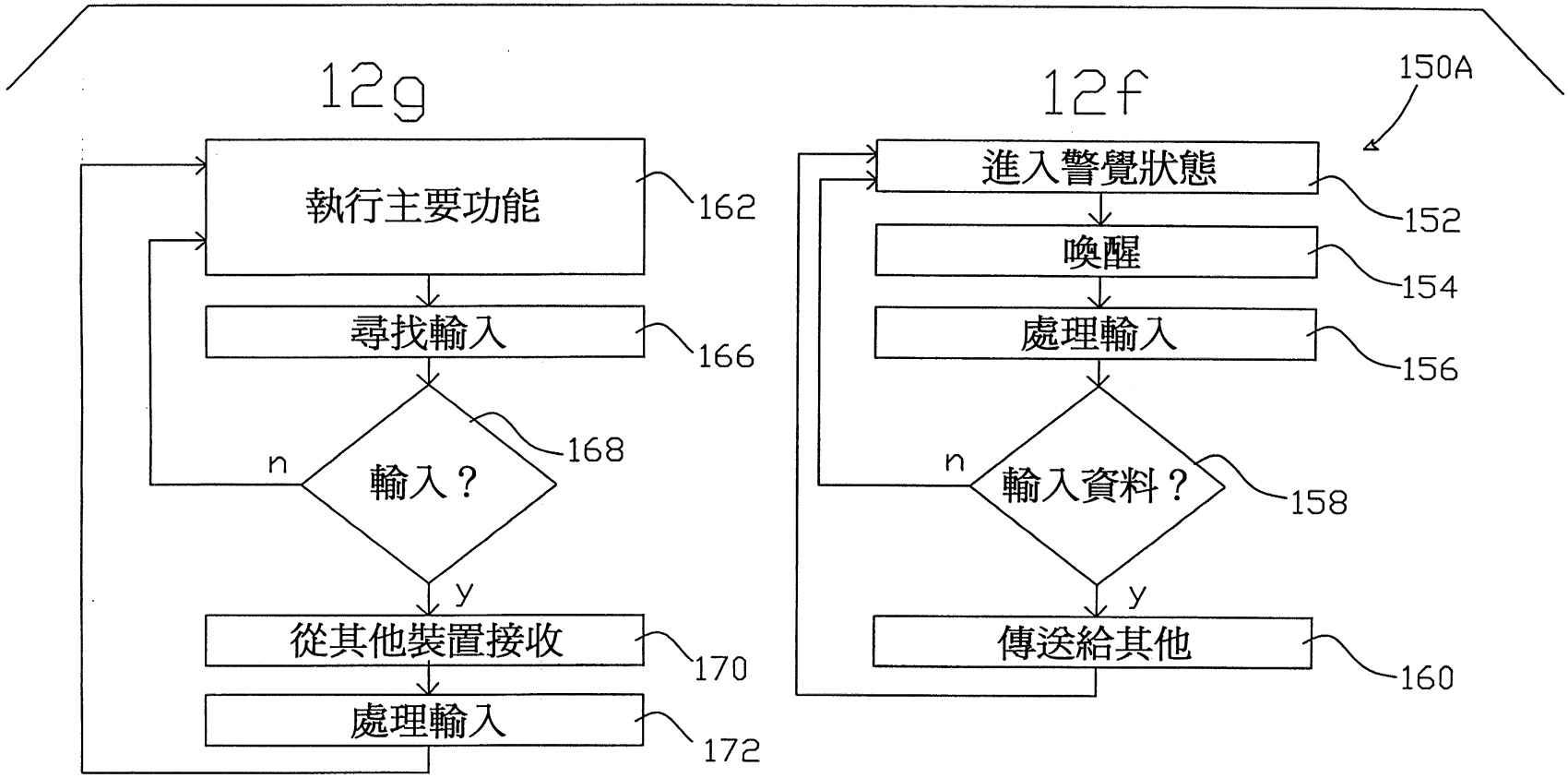
第5圖


120
載入P寄存器
136
擷取文字
122
124
位址？
N
Y
128
從埠擷取
從記憶體擷取
126
擷取指令文字
130
執行指令文字
132
138
埠位址？
Y
N
跳躍？
Y
134
N
遞增P寄存器
140

第7圖


150
進入警覺狀態
152
喚醒
154
處理輸入
156

第8圖



第9圖

七、指定代表圖：

（一）本案指定代表圖為：第（1）圖。

（二）本代表圖之元件符號簡單說明：

10 運算器陣列；　　14~晶粒；

16 資料匯流排；　　39 外部輸入輸出埠；

80 介面電路；　　82 外部裝置；

12、12a、12b、12c、12d、12e、12f、12g 運算器。

八、本案若有化學式時，請揭示最能顯示發明特徵的化學式：

無。