(19)中華民國智慧財產局
(12)發明說明書公開本 (11)公開編號：TW 201329862 A1
(43)公開日： 中華民國 102 (2013) 年 07 月 16 日

(21)申請案號：101101270 (22)申請日： 中華民國 101 (2012) 年 01 月 12 日
(51)Int. Cl.： *G06F9/32 (2006.01)* *G06F12/02 (2006.01)*
G06F21/55 (2013.01)
(71)申請人：國立交通大學 (中華民國) NATIONAL CHIAO TUNG UNIVERSITY (TW)
新竹市大學路 1001 號
(72)發明人：王繼偉 WANG, CHI WEI (TW)；謝續平 SHIEH, SHIUH PYNG (TW)；劉晏如 LIU, YEN JU (TW)
(74)代理人：詹銘文；葉璟宗
申請實體審查：有 申請專利範圍項數：28 項 圖式數：10 共 56 頁

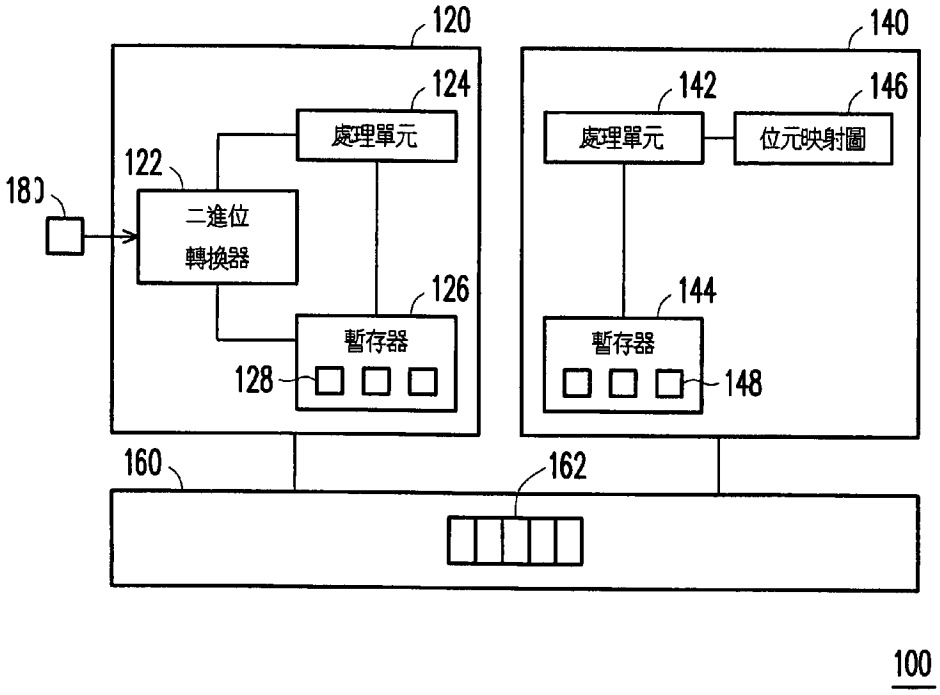
(54)名稱

低耦合的資訊流動的追蹤方法與電腦系統

DECOUPLED METHOD FOR TRACKING INFORMATION FLOW AND COMPUTER SYSTEM THEREOF

(57)摘要

一種追蹤資訊流動的電腦系統與方法。此電腦系統將資訊流動追蹤工作分解成低耦合的兩工作，並由兩程序來執行。第一程序用以模擬指令執行，並根據指令的執行順序將其分割為程式碼區塊。第一程序還將程式碼區塊的指令轉譯為資訊流動碼並傳送至第二程序。第一程序也將程式碼區塊的指令轉譯為動態模擬指令，並執行這些動態模擬指令以產生動態定址指令的定址結果。而第二程序根據這些定址結果執行資訊流動碼以模擬程式碼區塊的指令。並且，此方法還試圖減少第一程序在執行模擬工作時，與第二程序之間的傳輸量。藉此可以提升追蹤資訊流動的效率。



100：電腦系統
120：第一核心
122：二進位轉換器
124：處理單元
126：暫存器
128：動態模擬指令
140：第二核心
142：處理單元
144：暫存器
146：位元映射圖
148：資訊流動碼
160：共用記憶體
162：資料結構
180：指令

201329862

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：101101270 G06F 9/32 (2006.01)

※申請日：101. 1.12 ※IPC 分類：G06F 12/02 (2006.01)

G06F 21/55 (2013.01)

一、發明名稱：

低耦合的資訊流動的追蹤方法與電腦系統

DECOUPLED METHOD FOR TRACKING INFORMATION FLOW AND COMPUTER SYSTEM THEREOF

二、中文發明摘要：

一種追蹤資訊流動的電腦系統與方法。此電腦系統將資訊流動追蹤工作分解成低耦合的兩工作，並由兩程序來執行。第一程序用以模擬指令執行，並根據指令的執行順序將其分割為程式碼區塊。第一程序還將程式碼區塊的指令轉譯為資訊流動碼並傳送至第二程序。第一程序也將程式碼區塊的指令轉譯為動態模擬指令，並執行這些動態模擬指令以產生動態定址指令的定址結果。而第二程序根據這些定址結果執行資訊流動碼以模擬程式碼區塊的指令。並且，此方法還試圖減少第一程序在執行模擬工作時，與第二程序之間的傳輸量。藉此可以提升追蹤資訊流動的效率。

三、英文發明摘要：

A computer system and a method for tracking information flow are provided. The computer system divides a task of tracking information flows into two decoupled tasks which are executed by two procedures. The first procedure is configured to simulate executions of instructions, and divides the instructions into code blocks according to the execution order of the instructions. The first procedure translates the instructions of the code blocks into information flow codes and transmits them to the second procedure. The first procedure also translates the instructions of the code blocks into dynamic simulation instructions, and executes the dynamic simulation instructions to generate addressing results of dynamic addressing instructions. The second procedure executes the information flow code based on the addressing results to simulate the instructions of the code blocks. Moreover, the method also tries to reduce transportations between the two procedures when the first procedure executes the task of simulation. Therefore, the efficiency of tracking the information flows is improved.

四、指定代表圖：

（一）本案之指定代表圖：圖 1

（二）本代表圖之元件符號簡單說明：

100：電腦系統

120：第一核心

122：二進位轉換器

124：處理單元

126：暫存器

128：動態模擬指令

140：第二核心

142：處理單元

144：暫存器

146：位元映射圖

148：資訊流動碼

160：共用記憶體

162：資料結構

180：指令

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

六、發明說明：

【發明所屬之技術領域】

本發明是有關於一種資訊流動的追蹤技術，且特別是有關於一種將追蹤資訊流動的工作分割成兩個程序來執行的資訊流動的追蹤方法與電腦系統。

【先前技術】

隨著網路技術越來越發達，電腦之間的資料傳輸也越來越普及，使得多數的電腦都已連接至網際網路。但這也伴隨著網路攻擊的問題，也就是說一個電腦系統可能會接收到不可信任的資料，若執行這些資料可能會導致此電腦系統的機密資料被盜取，或者是被用以繼續進行其他網路攻擊。習知的解決辦法分為兩種，一種是在電腦系統中安裝偵測惡意程式的軟體，此種方法可以有效的偵測到電腦系統上各種情形(例如，指令的執行或是記憶體的使用狀況)。但由於此偵測軟體與電腦系統非常緊密的耦合，使得惡意程式可以較容易的規避偵測軟體。另一種方法是在電腦系統以外執行偵測軟體，並攔截此電腦系統上所有的網路連線動作，此方法可以避免惡意程式規避偵測軟體，但對電腦系統上的情況掌握較差。此兩種方法皆有其優缺點。然而，另一種方法亦被提出，其是在電腦系統上安裝虛擬機器，並在虛擬機器上執行一作業系統。而偵測軟體則是執行在虛擬機器監控系統(Virtual Machine Monitor, VMM)的層級。藉此，偵測軟體可以完整的掌握作業系統

上惡意程式的狀況，也不會被此惡意程式偵測到並進而規避。

然而，偵測軟體執行在虛擬機器監控系統的方法有效能上的問題。由於偵測軟體必須完全的模擬作業系統上每個指令的執行，藉此偵測到惡意程式所污染(taint)的暫存器、記憶體或硬碟。也就是說一個指令可能會被執行兩次，一次用以正常執行，一次用以模擬此指令並追蹤此指令的資訊流動(information flow)。因此，如何有效的追蹤資訊流動，為此領域研究人員所關心的議題。

【發明內容】

本發明提出一種低耦合的資訊流動的追蹤方法與電腦系統，其可以更有效率的追蹤資訊流動。

本發明的一實施例提出一種資訊流動的追蹤方法，用於一電腦系統，且此電腦系統包括一第一程序與一第二程序。此方法包括：由第一程序接收多個指令；由第一程序根據這些指令的執行順序，將指令分割為至少一程式碼區塊，程式碼區塊的這些指令包括多個動態定址指令與多個靜態定址指令；以及，由第一程序將程式碼區塊的這些指令轉譯為程式碼區塊的多個資訊流動碼，這些資訊流動碼對應至程式碼區塊的這些指令。此方法也包括：由第一程序將程式碼區塊的這些指令轉譯為程式碼區塊的多個動態模擬指令；由第一程序將這些資訊流動碼傳送至第二程序；由第一程序執行這些動態模擬指令以產生這些動態定

址指令的多個定址結果，並傳送這些定址結果至第二程序；以及，由第二程序根據這些定址結果執行這些資訊流動碼以追蹤程式碼區塊的這些指令的多個資訊流動。

在本發明之一實施例中，上述電腦系統包括一共用記憶體，且此共用記憶體包括一資料結構，此資料結構為先進先出的佇列。上述由第一程序傳送定址結果至第二程序的步驟包括：由第一程序將定址結果寫進資料結構；以及由第二程序從資料結構中讀取這些定址位置。

在本發明之一實施例中，上述程式碼區塊的動態模擬指令包括多個模擬執行指令，這些模擬執行指令是對應至程式碼區塊的指令。第一程序執行這些模擬執行指令時，等同於執行所對應的指令並產生所對應的動態定址指令的定址結果。

在本發明之一實施例中，上述程式碼區塊的動態模擬指令包括多個定址結果傳送指令。第一程序執行這些定址結果傳送指令時，會將定址結果傳送至第二程序。

在本發明之一實施例中，上述程式碼區塊的動態模擬指令包括多個計數指令。第一程序執行這些計數指令時會計算一指令個數。此指令個數為已被成功執行的程式碼區塊的模擬執行指令的個數。

在本發明之一實施例中，上述程式碼區塊的動態模擬指令包括一指令個數傳送指令。第一程序執行指令個數傳送指令時會將上述指令個數傳送至第二程序。

在本發明之一實施例中，上述程式碼區塊的動態模擬

指令包括一區塊序號傳送指令。第一程序執行區塊序號傳送指令時會將程式碼區塊的一區塊序號傳送至共用記憶體。

在本發明之一實施例中，上述由第二程序根據定址結果執行資訊流動碼的步驟包括：由第二程序根據區塊序號以決定追蹤程式碼區塊的這些資訊流動的順序；以及由第二程序根據指令個數來追蹤程式碼區塊的資訊流動。

在本發明之一實施例中，上述資訊流動碼符合一多位元組格式，此多位元組格式包括來源運算元格式欄位、目標運算元格式欄位、寬度欄位、效果欄位、來源運算元欄位、以及目標運算元欄位。來源運算元格式欄位用以記錄一來源運算元為記憶體或是暫存器。目標運算元格式欄位用以記錄一目標運算元為記憶體或是暫存器。寬度欄位用以記錄一運算子的一運算寬度。效果欄位用以記錄運算子的一運算種類。來源運算元欄位用以記錄來源運算元。目標運算元欄位用以記錄目標運算元。

在本發明之一實施例中，上述資訊流動碼符合一單位元格式，此單位元格式包括效果欄位、來源運算元格式欄位、來源運算元欄位、來源運算元偏移欄位、目標運算元格式欄位、目標運算元欄位、以及目標運算元偏移欄位。效果欄位用以記錄運算子的運算種類。來源運算元格式欄位用以記錄來源運算元為記憶體或是暫存器。來源運算元欄位用以記錄來源運算元。來源運算元偏移欄位用以記錄來源運算元的偏移量。目標運算元格式欄位用以記錄目標

運算元為記憶體或是暫存器。目標運算元欄位用以記錄目標運算元。目標運算元偏移欄位用以記錄目標運算元的偏移量。

在本發明之一實施例中，上述電腦系統包括迴圈暫存器，迴圈暫存器的值對應至基礎頁面。並且，與基礎頁面相鄰的多個虛擬頁面中包括一邊端頁面。上述追蹤方法還包括：當迴圈暫存器的數值改變時、由第一程序根據迴圈暫存器的值計算出所對應的基礎頁面與邊端頁面；以及由第一程序將迴圈暫存器的值、基礎頁面的實體位址、以及邊端頁面的實體位址傳送至第二程序。

在本發明之一實施例中，上述追蹤方法還包括：由第二程序根據迴圈暫存器的值、基礎頁面的實體位址、以及邊端頁面的實體位址，計算出包括迴圈暫存器的動態定址指令的定址結果。

在本發明之一實施例中，上述電腦系統使用 IA-32 中央處理器的架構，迴圈暫存器為 ebp 暫存器。

在本發明之一實施例中，上述電腦系統包括一位元映射圖，此追蹤方法還包括：由第二程序在追蹤這些資訊流動時，將被污染的記憶體位置、暫存器或是硬碟位置記錄在位元映射圖。

以另外一個角度來說，本發明的一實施例提出一種追蹤資訊流動的電腦系統。此電腦系統包括第一核心與第二核心，第二核心是耦接至第一核心。。第一核心用以接收多個指令，並根據這些指令的執行順序，將這些指令分割

為至少一程式碼區塊。其中程式碼區塊的這些指令包括多個動態定址指令與多個靜態定址指令。其中，第一核心將程式碼區塊的這些指令轉譯為程式碼區塊的多個資訊流動碼，這些資訊流動碼對應至程式碼區塊的指令。第一核心更用以將程式碼區塊的指令轉譯為程式碼區塊的多個動態模擬指令，並將資訊流動碼傳送至電腦系統的第二核心。第一核心還用以執行動態模擬指令以產生動態定址指令的多個定址結果，並傳送這些定址結果至第二核心。其中，第二核心根據這些定址結果執行這些資訊流動碼以追蹤程式碼區塊的這些指令的多個資訊流動。

在本發明之一實施例中，上述電腦系統更包括一共用記憶體，其耦接至第一核心與第二核心。此共用記憶體包括一資料結構，且此資料結構為先進先出的佇列。

在本發明之一實施例中，上述第一核心更用以將定址結果寫進資料結構，並且第二核心從資料結構中讀取這些定址位置。

在本發明之一實施例中，上述程式碼區塊的動態模擬指令包括多個模擬執行指令，這些模擬執行指令是對應至程式碼區塊的指令。第一核心執行這些模擬執行指令時，等同於執行所對應的指令並產生所對應的動態定址指令的定址結果。

在本發明之一實施例中，上述程式碼區塊的動態模擬指令包括多個定址結果傳送指令。第一核心執行這些定址結果傳送指令時，會將定址結果傳送至第二核心。

在本發明之一實施例中，上述程式碼區塊的動態模擬指令包括多個計數指令。第一核心執行這些計數指令時會計算一指令個數。此指令個數為已被成功執行的程式碼區塊的模擬執行指令的個數。

在本發明之一實施例中，上述程式碼區塊的動態模擬指令包括一指令個數傳送指令。第一核心執行指令個數傳送指令時會將上述指令個數傳送至第二核心。

在本發明之一實施例中，上述程式碼區塊的動態模擬指令包括一區塊序號傳送指令。第一核心執行區塊序號傳送指令時會將程式碼區塊的一區塊序號傳送至共用記憶體。

在本發明之一實施例中，上述第二核心更用以根據區塊序號以決定追蹤程式碼區塊的這些資訊流動的順序，並且第二核心根據指令個數來追蹤程式碼區塊的這些資訊流動。

在本發明之一實施例中，上述資訊流動碼符合一多位元組格式，此多位元組格式包括來源運算元格式欄位、目標運算元格式欄位、寬度欄位、效果欄位、來源運算元欄位、以及目標運算元欄位。來源運算元格式欄位用以記錄一來源運算元為記憶體或是暫存器。目標運算元格式欄位用以記錄一目標運算元為記憶體或是暫存器。寬度欄位用以記錄一運算子的一運算寬度。效果欄位用以記錄運算子的一運算種類。來源運算元欄位用以記錄來源運算元。目標運算元欄位用以記錄目標運算元。

在本發明之一實施例中，上述資訊流動碼符合一單位元格式，此單位元格式包括效果欄位、來源運算元格式欄位、來源運算元欄位、來源運算元偏移欄位、目標運算元格式欄位、目標運算元欄位、以及目標運算元偏移欄位。效果欄位用以記錄運算子的運算種類。來源運算元格式欄位用以記錄來源運算元為記憶體或是暫存器。來源運算元欄位用以記錄來源運算元。來源運算元偏移欄位用以記錄來源運算元的偏移量。目標運算元格式欄位用以記錄目標運算元為記憶體或是暫存器。目標運算元欄位用以記錄目標運算元。目標運算元偏移欄位用以記錄目標運算元的偏移量。

在本發明之一實施例中，上述電腦系統包括一迴圈暫存器，迴圈暫存器的數值對應至一基礎頁面，與基礎頁面相鄰的多個虛擬頁面中包括一邊端頁面。其中當迴圈暫存器的數值改變時，第一核心更用以根據迴圈暫存器的值計算出所對應的基礎頁面與邊端頁面。並且第一核心會將迴圈暫存器的值、基礎頁面的實體位址、以及邊端頁面的實體位址傳送至第二核心。

在本發明之一實施例中，上述第二核心更用以根據迴圈暫存器的值、基礎頁面的實體位址、以及邊端頁面的實體位址，計算出包括迴圈暫存器的這些動態定址指令的這些定址結果。

在本發明之一實施例中，上述電腦系統使用 IA-32 中央處理器的架構，上述的迴圈暫存器為 ebp 暫存器。

在本發明之一實施例中，上述第二核心包括一位元映射圖。第二核心在追蹤這些資訊流動時，將被污染的記憶體位置、暫存器或是硬碟位置記錄在此位元映射圖中。

基於上述，本發明所提出的追蹤方法與電腦系統，可以將指令分割為程式碼區塊。第一核心與第二核心皆以程式碼區塊為單位來執行各自的運作，如此一來，可以減少第一核心與第二核心之間的傳輸量。並且，第一核心只傳輸必要的資訊，例如地址結果，迴圈暫存器的數值等給第二核心。藉此更進一步的減少第一核心與第二核心之間的傳輸量，增加追蹤指令的資訊流動的效率。

為讓本發明之上述特徵和優點能更明顯易懂，下文特舉實施例，並配合所附圖式作詳細說明如下。

【實施方式】

圖 1 為依照本發明一實施例所繪式的電腦系統方塊圖。

請參考圖 1，電腦系統 100 包括有第一核心 120、第二核心 140 與共用記憶體 160。電腦系統 100 是用以執行多個指令 180，並追蹤指令 180 的資訊流動(information flow)。特別是，電腦系統 100 包括有一第一程序與一第二程序(未匯示)。本發明是將高耦合的追蹤資訊流動的工作切割，並由此第一程序與第二程序來執行。例如，第一程序與第二程序是電腦系統 100 所包含的兩個執行緒(thread)。在本實施例中，第一程序是由第一核心 120 來執

行，而第二程序是由第二核心 140 來執行。也就是說，追蹤資訊流動的工作是交由兩個核心來執行，藉此提升效率。值得注意的是，在本實施例中將以第一核心 120 與第二核心 140 的執行來說明如何追蹤資訊流動，但也等同於說明第一程序與第二程序如何追蹤資訊流動。也就是說，第一程序即包含了第一核心 120 所執行的步驟；第二程序即包含了第二核心 140 所執行的步驟。另一方面，電腦系統 100 也可以包含更多數目的核心。為了負載的平衡，電腦系統 100 可以將第一程序移到第一核心 120 以外的核心上執行，第二程序也可以移到第二核心 140 以外的核心上執行。本發明並不限制第一程序與第二程序是由哪一個核心所執行。本實施例中的一個特點在於電腦系統 100 是使用第一核心 120 執行指令 180(以下說明第一核心的動作時，也表示第一程序所包含的動作)，並使用第二核心 140 執行追蹤的步驟(以下說明第二核心的動作時，也表示第二程序所包含的動作)。

圖 2 為依照本發明一實施例說明追蹤資訊流動的示意圖。

請參考圖 2，當電腦系統 100 從網路取得一資料並儲存在記憶體 200 的第一部分 202 時，第一部分 202 會被標記為污染(taint)，表示第一部份 202 中的資料可能並不安全。接著，若電腦系統執行了一些指令，使得第一部分 202 的資料被複製到第二部分 204，則第二部分 204 也會被標記為污染。然而，電腦系統 100 也可能計算第一部份 202

中的資料，並將計算結果儲存在第二部份 204，此時第二部份 204 也會被標記為汙染，本發明並不限制污染第二部份 204 的方式。另一方面，第一部分 202 的資料透過輸入/輸出(input/output, I/O)指令的執行也可能污染暫存器或是硬碟，本發明並不在此限。本發明中將上述第一部份 202 影響到第二部分 204 的過程稱為所執行指令的資訊流動。

請參考回圖 1。具體來說，第一核心 120 將指令 180 轉譯成特定格式的資訊流動碼並將資訊流動碼傳送給第二核心。這些資訊流動碼對應至指令 180，包括了資訊流動的資訊，而第二核心 140 則用以執行這些資訊流動碼來執行追蹤資訊流動的步驟。也就是說，本發明中第一核心 120 與第二核心 140 是平行處理，第一核心用以執行指令 180，而第二核心用以追蹤指令 180 所對應的資訊流動，藉此提昇追蹤資訊流動的效率。

第一核心 120 包括二進位轉換器 122、處理單元 124 與暫存器 126。二進位轉換器 122 用以接收指令 180，並依照指令 180 的執行順序將指令 180 切割成至少一個程式碼區塊。電腦系統 100 是以程式碼區塊為單位來執行指令 180 並模擬指令 180 的資訊流動。請參照圖 3，圖 3 為依照本發明一實施例說明程式碼區塊的示意圖。程式碼區塊 300 中包含了多個指令，然而本發明並不限制一個程式碼區塊中指令的個數，也不限制指令 180 被切割成的程式碼區塊的個數。而程式碼區塊 300 中包含多個指令，其中包括動態定址指令與靜態定址區塊。動態定址指令為必須要在執

行期才能決定定址的指令。例如，指令 181、指令 183 與指令 184 為動態定址指令，這些指令的定址都必須要在執行期才能被決定。另一方面，靜態定址指令為能夠在編譯期就決定定址的指令，例如，指令 182 為靜態定址指令。

二進位轉換器 122 還以程式碼區塊為單位將指令 180 轉譯為資訊流動碼，並將資訊流動碼傳送給第二核心 140。另一方面，二進位轉換器 122 還以程式碼區塊為單位將指令 180 轉換為動態模擬指令 128，並將動態模擬指令 128 存在暫存器 126 中。處理單元 124 用以執行動態模擬指令 128，並且在執行動態模擬指令時等同於執行指令 180，並且產生關於程式碼區塊的資訊以及動態定址的定址結果。例如，處理單元 124 執行動態模擬指令 128 時，會產生指令 181、183、184 的定址結果。而關於程式碼區塊的資訊可包括程式碼區塊 300 的序號，此序號是用以決定執行程式碼區塊的順序。處理單元 124 還將所產生的關於程式碼區塊的資訊以及定址結果傳送至第二核心 140。

第二核心 140 包括處理單元 142、暫存器 144 與位元映射圖 146。第二核心 140 將所接收到的資訊流動碼 148 儲存在暫存器 144 中。處理單元 142 根據所接收到的定址結果執行資訊流動碼 148 以追蹤指令 180 的資訊流動。另一方面，處理單元 142 還將追蹤指令 180 的資訊流動的結果儲存在位元映射圖 146。例如，處理單元 142 是紀錄被污染的記憶體位置、暫存器或是硬碟位置於位元映射圖 146 中。舉例來說，電腦系統 100 每一個記憶體位置、暫

存器或硬碟位置都用一個位元來表示，若此位元為"1"，則表示所對應的位置或暫存器被標記為汙染。然而，在本發明的其他實施例中也可以用其他方式將汙染的記憶體位置、硬碟位置或暫存器儲存在位元映射圖 146 中，本發明並不限制位元映射圖 146 的儲存方式。

在一實施例中，第一核心 120 是將動態定址指令的定址結果傳送至共用記憶體 160，而第二核心從共用記憶體 160 中讀取這些定址結果。例如，共用記憶體 160 中包含一資料結構 162，資料結構 162 為先進先出(First in First out, FIFO)的佇列(queue)。而第一核心 120 是將定址結果傳送至資料結構 162，且第二核心 140 是從資料結構 162 中讀取定址結果。第一核心 120 將定址結果寫入至資料結構 162 的順序是符合指令 180 的執行順序；並且，由於資料結構 162 是 FIFO 的佇列，因此第二核心 120 從資料結構 162 讀取的順序也符合指令 180 的執行順序。然而，在其他實施例中第一核心 120 與第二核心 140 皆耦接至一匯流排(未繪示)，第一核心 120 藉由匯流排而將定址結果傳送至第二核心 140，本發明並不限制第一核心 120 傳送定址桔果給第二核心 140 的方法。

為詳細定義動態模擬指令 128 與資訊流動碼 148 的格式與內容，本發明還將指令 180 分類為四種類別，並根據指令 180 的類別來產生資訊流動碼 148。在本實施例中，電腦系統 100 是使用 IA-32 中央處理器(central computing unit, CPU)的架構，指令 180 為 IA-32 中央處理器的指令

集，然而在其他實施例中電腦系統 100 也可以使用其他中央處理器的架構，本發明不在此限。

圖 4 為依照本發明一實施例說明指令類別的示意圖。

請參照圖 4，指令 180 被分類為位元方式覆寫 410、位元方式附加 420、遞增混合 430 與全部混合 440。每種類別皆代表一種資訊流動的方式。以下以 A[i]表示為目標運算元的一位元組，B[i]表示為來源運算元的一位元組，i 為運算元中的一位元組的位置，i 的數值是從 0 到 n。

圖 5A 至圖 5D 為依照本發明一實施例說明指令的資訊流動的示意圖。

被分類為位元方式覆寫 410 的指令包括單純移動暫存記憶體的指令。也就是說，來源運算元的一位元組只會影響相同位置的目標運算元中的位元組。例如，B[0]只會影響 A[0]，B[n]只會影響 A[n]。若指令的類別是位元方式覆寫 410，則記號為 $\Leftarrow_n$。請參考圖 5A。指令"mov eax, ebx"是將暫存器 ebx 的值複製到暫存器 eax。例如，暫存器 ebx 為 32 位元的暫存器，包括了位元組 502a、502b、502c、502d。而暫存器 eax 也是 32 位元的暫存器，包括了位元組 504a、504b、504c、504d。在執行完指令"mov eax, ebx"以後，暫存器 ebx 的每個位元組便會複製到 eax 所對應的位元組。而如果暫存器 ebx 包括了被污染的位元組，則暫存器 eax 也會包括被污染的位元組。例如，位元組 502a 是被汙染的暫存器，則在執行完指令"mov eac, ebx"以後，位元組 504a 也會變成被汙染的暫存器。圖 5A 中是以箭頭代表

資訊流動的方向，例如，位元組 502a 指向位元祖 504a，表示位元組 502a 會影響位元組 504a。

而位元方式附加 420 的類別包括了目標運算元與來源運算元會重複的指令。也就是說，A 同時為目標運算元與來源運算元；A[0]同時會被 A[0]與 B[0]所影響。若一指令被分類為位元方式附加，則記號為 $\overset{+}{\Leftarrow}_n$。請參考圖 5B，指令"xor eax, ebx"是將暫存器 eax 與暫存器 ebx 的內容做 xor 的運算，並將運算結果存入暫存器 eax 中。因此，若暫存器 ebx 或暫存器 eax 包括了被污染的位元組，則暫存器 eax 也會包括被污染的位元組。圖 5B 中是以箭頭代表資訊流動的方向。例如，位元組 506a 是被汙染的位元祖，則在執行完指令"xor eac, ebx"以後，位元組 506b 也會變成被汙染的位元祖。

遞增混合 430 的類別包括了低位元組會影響高位元組的指令。也就是說，高位元組的 A[n]不只會被 B[n]影響，也會被低位元組的 A[n-1]影響。若一指令被分類為遞增混合 430，則記號為 $\overset{A}{\Leftarrow}_n$。請參考圖 5C，指令"add eax, ebx"是將暫存器 eax 與暫存器 ebx 的值相加，並將相加的結果儲存在暫存器 eax 中。若暫存器 eax 與暫存器 ebx 包括了被污染的位元組，則暫存器 eax 也會包括被污染的位元組，並且低位元組會汙染高位元組。圖 5C 中是以箭頭代表資訊流動的方向。例如位元組 508a 是被汙染的位元組，則在執行指令"add eax, ebx"以後，位元組 508b、506C、508d 都會變成被汙染的位元組。

全部混合 440 則包括了不屬於位元方式復寫 410、位元方式附加 420 以及遞增混合 430 的其他指令。若一指令被分類為全部混合 440，則記號為 $\overset{*}{\Leftarrow}_n$。請參考圖 5D，指令"mul ebx"是將暫存器 eax 與暫存器 ebx 的值相乘，並將相乘的結果放在暫存器 eax 與暫存器 edx 當中。值得注意的是，由於暫存器 eax 與暫存器 ebx 皆是 32 位元的暫存器，因此相乘的結果為 64 位元，必須使用兩個暫存器才可以儲存相乘的結果。圖 5D 中是以箭頭代表資訊流動的方向。例如，暫存器 ebx 的位元組 510a 是被汙染的位元組，則在執行指令"mul ebx"以後，位元組 510b、510c、510d、510e、510f 都會變成被汙染的位元組。值得注意的是，為了圖 5D 的可閱讀性，圖 5D 中一些箭頭已被省略。

值得注意的是，上述的分類皆是以暫存器為例子。然而，關於記憶體存取的指令也可以被分類為上述的分類。例如，指令"mov eax, [ebp+0x8]"是將暫存器 ebp 的值加上 0x8 作為定址結果，並根據此定址結果從記憶體中讀取資料至暫存器 eax 中。指令"mov eax, [ebp+0x8]"可被分類為位元方式覆寫 410。若對應此定址結果的記憶體中包括被污染的位元組，則暫存器 eax 中也會包括被污染的位元組。另一方面，由於暫存器 ebp 的值必須在執行期才能決定，因此定址結果必須在執行期才能確定，指令"mov eax, [ebp+0x8]"便是動態定址指令。在其他實施例中，指令 180 也可能包括存取電腦系統 100 上的任何輸入/輸出(input/output, I/O)裝置(例如，硬碟)。因此，電腦系統上的

I/O 裝置也有可能被污染。

在將指令 180 分類為位元方式覆寫 410、位元方式附加 420、遞增混合 430 或全部混合 440 以後，二位轉換器 122 便可以依照分類將指令 180 的分類產生特定格式的資訊流動碼。資訊流動碼中會包含指令 180 的資訊流動資訊(例如，哪個記憶體位置會汙染哪些暫存器，或相反)，第二核心 140 藉由執行資訊流動碼來追蹤指令 180 的資訊流動。

資訊流動碼可能符合一種多位元組格式，多位元組格式是用以當一指令同時處理多個位元組時。例如，指令"mov eax, ebx"是將暫存器 ebx 的四個位元組複製到暫存器 eax 的四個位元組，此指令便可符合多位元組格式。請參照圖 6A，圖 6A 為依照本發明一實施例說明多位元組格式的示意圖。當資訊流動碼 602 符合多位元組格式時，資訊流動碼 602 為固定長度，例如，32 位元。而多位元組格式包括了表頭欄位 610、目標運算元格式欄位 620、來元運算元格式欄位 630、寬度欄位 640、效果欄位 650、目標運算元欄位 660、以及來源運算元欄位 670。

其中表頭欄位 610 用以記錄資訊流動碼 602 的格式，例如，當表頭欄位 610 裡為"00"時，表示資訊流動碼 602 為多位元組格式。

目標運算元格式欄位 620 用以記錄目標運算元為格式或是記憶體，例如，指令"mov eax, ebx"的目標運算元是暫存器 eax，則目標運算元格式欄位 620 則被寫入"0"用以表

示目標運算元是暫存器。若目標運算元是記憶體，則在目標運算元格式欄位 620 寫入"1"。來源運算元格式欄位 630 用以記錄來源運算元為暫存器或記憶體，其紀錄方式與目標運算元格式欄位 620 類似，在此便不再贅述。然而，目標運算元格式欄位 620 與來源運算元格式欄位 630 也可使用其他數值來表示對應的運算元為記憶體或暫存器，本發明並不在此限。

寬度欄位 640 用以記錄運算子的運算寬度，例如，指令"mov eax, ebx"的運算子為"mov"，其會移動 32 位元的資料。因此，寬度欄位 640 則被寫入"11"，用以表示運算子的運算寬度為 32 位元。然而，當運算子的運算寬度為其他數值時，寬度欄位 640 可被寫入其他的值以代表其他運算寬度，本發明不在此限。效果欄位 650 用以記錄運算子的運算種類。例如，運算種類為位元方式覆寫 410、位元方式附加 420、遞增混合 430 或全部混合 440(如圖 4 所示)。效果欄位 650 可被寫入不同的值以代表不同的運算種類。

目標運算元欄位 660 用以記錄目標運算元。例如，指令"mov eax, ebx"的目標運算元是暫存器 eax，則在目標運算元欄位 660 可以寫入"0x0001"用以表示暫存器 eax。然而，暫存器 eax 可以用其他的數值表示，本發明不在此限。來源運算元欄位 670 用以記錄來源運算元，其紀錄方式類似於目標運算元欄位 660，在此便不再贅述。也就是說，在一程式碼區塊中，來源運算元與目標運算元皆可以用固定的數值來表示(例如，"0x0001"表示暫存器 eax)，因此，

第二核心 140 在執行資訊流動碼 602 時，便可以根據來源運算元欄位 670 與目標運算元欄位 660 取得來源運算元與目標運算元。

如此一來，第二核心 140 在執行資訊流動碼 602 時，可從寬度欄位 640 與效果欄位 650 取得資訊流動的資訊，而不需要計算出資訊流動碼 602 所對應的指令的結果。舉例來說，資訊流動碼 602 對應至指令"mov eax, ebx"，而第二核心 140 在執行資訊流動碼 602 時，不需要計算出指令"mov eax, ebx"的結果，也可以根據資訊流動碼 602 得知暫存器 eax 可能會被暫存器 ebx 所污染。並且，第二核心 140 中包括了位元映射圖 146，用以追蹤被污染的暫存器、記憶體位置或是硬碟位置。因此第二核心 140 便可以藉由資訊流動碼 602 來追蹤指令"mov eax, ebx"的資訊流動。值得注意的是，指令"mov eax, ebx"為靜態定址指令，也就是其定址可以在編譯期就決定，因此第二核心 140 在追蹤指令"mov eax, ebx"時，第一核心 120 與第二核心 140 之間並不需要傳送任何資訊。如此一來，第一核心 120 與第二核心 140 之間所需要傳送的資訊便減少了，使電腦系統 100 可以有效的追蹤指令 180 的資訊流動。

然而，指令 180 可能包括一些特殊的指令。例如，將暫存器 eax 的最低位元組複製到暫存器 ebx 的最高位元組。類似這樣的特殊指令便無法用多位元組格式的資訊流動碼來表示。因此本發明中，資訊流動碼還可能符合單位元組格式，其可以處理任何資訊流動的情況。

圖6B與圖6C為依照本發明一實施例說明單位元組格式的示意圖。

請參考圖 6B。當資訊流動碼 604 符合單位元組格式時，資訊流動碼 604 為固定長度，例如，32 位元。而資訊流動碼 604 包括了表頭欄位 612、目標運算元格式欄位 620、來元運算元格式欄位 630、效果欄位 650、目標運算元欄位 660、來源運算元欄位 670、目標運算元偏移欄位 680、以及來源運算元偏移欄位 690。其中目標運算元格式欄位 620、來元運算元格式欄位 630、效果欄位 650、目標運算元欄位 660 以及來源運算元欄位 670 已詳細描述如上，在此便不再贅述。

其中表頭欄位 612 用以記錄資訊流動碼 604 的格式，例如，當表頭欄位 612 裡為"01"時，表示資訊流動碼 604 為單位元組格式。

目標運算元偏移欄位 680 用以記錄目標運算元的偏移量；來源運算元偏移欄位 690 用以記錄來源運算元的偏移量。舉例來說，資訊流動碼 604 對應至指令"mov ch, al"，而指令"mov ch, al"可用以將暫存器 eax 的位元組 692 複製到暫存器 ecx 的位元組 682(如圖 6C 所示)。也就是說，位元組 692 的偏移量與位元組 682 的偏移量並不相同，在此例子中，偏移量是相對於最小有效位元(least significant bit, LSB)來計算。因此，目標運算元偏移欄位 680 會被寫入"0x0001"，用以表示位元組 682 的偏移量為 1 個位元組；而來源運算元偏移欄位 690 會被寫入"0x0000"，用以表示

692 的偏移量為 0 個位元組。然而，在其他實施例中，偏移量也可以是相對於最大有效位元(most significant bit, MSB)來計算，且目標運算元偏移欄位 680 與來源運算元偏移欄位 690 可用其他數值來表示其偏移量，本發明應不在此限。

藉由執行單位元組格式與多位元格式的資訊流動碼，第二核心 140 便可以追蹤所對應的靜態定址指令的資訊流動。然而，若資訊流動碼所對應的為動態定址指令，例如，指令"mov eax, [ebp+0x8]"，則第二核心還需要此指令的定址結果(即，ebp+0x8 的計算結果)來追蹤此指令的資訊流動。

請參考回圖 1，第一核心 120 是藉由執行動態模擬指令 128 來產生第二核心 140 所需要的定址結果以及與程式碼區塊相關的資訊。

圖 7 為依照本發明一實施例說明指令、動態模擬指令與資訊流動碼的示意圖。

請參考圖 7。二進位轉換器 122 以程式碼區塊 300 為單位將指令 181~184 轉換為動態模擬指令 128 與資訊流動碼 148。動態模擬指令 128 由第一核心 120 來執行，而資訊流動碼 148 由第二核心 140 來執行。動態模擬指令包括了指令個數傳送指令、區塊序號傳送指令、模擬執行指令、計數指令以及定址結果傳送指令。

模擬執行指令 703、706、708、711 分別對應至指令 181、812、183、184。當第一核心 120 執行模擬執行指令

時，等同於執行對應的指令。並且，當模擬執行指令所對應的為動態定址指令時，第一核心 120 會產生相對應的定址結果。例如，當第一核心 120 執行模擬執行指令 708 時，等同於執行指令 183，並會產生指令 183 的定址結果(即，ebp+0x8 的計算結果)。相同地，第一核心 120 執行模擬執行指令 711 時，等同於執行指令 184，並會產生指令 184 的定址結果(即，ebp+0xc 的計算結果)。

定址結果傳送指令是在一動態定址指令所對應的模擬執行指令之後，用以傳送定址結果至第二核心 140。舉例來說，指令 181、183、184 為動態定址指令，而所對應的動態模擬指令分別為模擬執行指令 703、708、711。定址結果傳送指令 705、710、713 便分別在模擬執行指令 703、708、711 之後，用以傳送指令 181、183、184 的定址結果至第二核心 140。

計數指令 704、707、709、712 則分別在模擬執行指令 703、706、708、711 之後，用以計算第一核心 120 成功執行模擬執行指令的指令個數。舉例來說，計數指令 704 在模擬執行指令 703 之後，用以將指令個數加 1，在此例子中指令個數是以變數 counter 來表示。相同地，計數指令 712 在模擬執行指令 711 之後，用以將指令個數增加 1。換句話說，指令個數(例如，變數 counter)代表第一核心 120 已成功執行的模擬執行指令的個數。

指令個數傳送指令 701 在動態模擬指令 128 的最前端，用以將指令個數(例如，變數 counter)傳送至第二核心

140。此指令個數代表的是，第一核心在上一個動態模擬指令(未繪示)累加的成功執行的模擬執行指令的個數。由於任何程式碼皆有可以會產生例外(exception)，例如，頁面錯誤(page fault)或是被其他 I/O 裝置中斷。因此，藉由傳送指令個數至第二核心 140，可使第二核心 140 得知目前第一核心 120 的執行狀況(例如，第一核心已成功執行了多少個模擬執行指令)。

區塊序號傳送指令 702 是用以傳送動態模擬指令 128 所對應的程式碼區塊的區塊序號。也就是說，二進位轉換器 122 在將指令 180 切割成至少一個程式碼區塊時，會設定每個程式碼區塊擁有一個區塊序號(例如，變數 block_seq)，每個程式碼區塊所擁有的區塊序號都是唯一的。當第一核心 120 執行區塊序號傳送指令 702 時，便會將所對應的區塊序號(例如，變數 block_seq)傳送至第二核心 140。如此一來，第二核心 140 便可以根據此區塊序號來取得正確的程式碼區塊的順序，並從暫存器 144 中取得正確的資訊流動碼 148。

在本實施例中，第一核心 120 是將定址結果、指令個數以及區塊序號傳送至共用記憶體 160 的資料結構 162。也就是說，第一核心 120 是藉由 enqueue 指令將資料寫入至資料結構 162 中。而第二核心 140 可藉由 dequeue 指令從資料結構 162 中讀取所需的資料。然而，在其他實施例中，第一核心 120 可以透過匯流排(未繪示)來將資料傳送至第二核心 140，本發明應不在此限。

另一方面，資訊流動碼 721、722、723、724 分別對應至指令 181、182、183、184。當第二核心執行資訊流動碼時，便可以追蹤指令的資訊流動。舉例來說，當第二核心 140 執行資訊流動碼 723 時，便可以追蹤指令 183 的資訊流動。值得注意的是，由於指令 183 是動態定址指令，因此第二核心還需要指令 183 的定址結果。然而，根據上述，第一核心 120 在執行動態模擬指令 710 時，已將指令 183 的定址結果傳送至第二核心 140，因此第二核心 140 便可以正確的追蹤指令 183 的資訊流動。而資訊流動碼 721~724 的內容以及格式以詳細描述如上，在此並不再贅述。

值得一提的是，本發明是以程式碼區塊為單位來模擬指令的資訊流動。由於一次傳送尺寸較大的檔案，比傳送多次傳送尺寸較小的檔案來的有效率。因此，第一核心 120 是在將一個程式碼區塊中的指令轉譯為動態模擬指令與資訊流動碼之後，才將一個區塊程式碼中的資訊流動碼傳送給第二核心 140。如此一來，第一核心 120 與第二核心 140 之間的傳輸次數便可以減少，

除此之外，第一核心 120 還會將一迴圈暫存器的值傳送給第二核心 140。在一實施例中，電腦系統 100 使用的是 IA-32 中央處理器的架構，此迴圈暫存器為 ebp 暫存器。在 IA-32 的架構中，編譯器(未繪示)經常會使用暫存器 ebp 來作為迴圈的定址。舉例來說，指令 183(即，"mov eax, [ebp+0x8]")為常見的指令。在這樣的架構中，ebp 暫存器

的值並不會經常改變，編譯器會使用暫存器 ebp 加上一個補償值(例如，0x8)來作為定址結果。並且，補償值的範圍並不會超過一定的範圍。

圖 8A 至圖 8C 為依照本發明一實施例所繪示的迴圈暫存器的補償值的示意圖。

依照統計結果，迴圈暫存器的補償值多數會在-1024到+512 個位元組之間。這是由於一般的程式碼都會有區域參照(locality of reference)的特性，因此程式碼所使用的記憶體也會有區域性的特性。請參考圖 8A。若迴圈暫存器(例如，ebp 暫存器)的數值所指向的記憶體頁面是虛擬頁面 802，而且補償值的範圍並不會超過虛擬頁面 802 的範圍，則可以經由 ebp 暫存器的數值加上補償值來算出。值得注意的是，ebp 暫存器的數值所指的是虛擬地址(virtual address)，而在執行期時所需要的定址結果是實體地址(physical address)。但是由於迴圈暫存器與補償值的和並沒有超出虛擬頁面 802 的範圍，因此只需要虛擬頁面 802 的實體位址便可以計算出定址結果。

請參考圖 8B，若迴圈暫存器的數值所指向的是虛擬頁面 802 的邊緣，則將迴圈暫存器的值加上補償值以後可能會超出虛擬頁面 802 的範圍，例如，包括虛擬頁面 806。除此之外，雖然將迴圈暫存器的值加上補償值所計算出的虛擬位址是連續的，然而卻可能會對應至不同的實體位址，也就是說，虛擬頁面 802 與虛擬頁面 806 可能會對應到不同的實體位址。因此，要計算出定址結果除了需要虛

擬頁面 802 的實體位址，也需要虛擬頁面 806 的實體位址。請參考圖 8C，若迴圈暫存器的數值所指向的是虛擬頁面 802 的另一個邊緣，則將迴圈暫存器的值加上補償值以後可能會超出虛擬頁面 802 的範圍，例如，包括虛擬頁面 804。則要計算出定址結果除了需要虛擬頁面 802 的實體位址，也需要虛擬頁面 806 的實體位址。

值得注意的是，雖然圖 8B 與圖 8C 中的定址結果皆包括了不同的虛擬頁面，但由於補償值的範圍有限，因此所對應的虛擬頁面並不會超過兩個。因此，本發明是將迴圈暫存器所指向的虛擬頁面設定為基礎頁面，並且依照迴圈暫存器的值在基礎頁面的位址來決定一個鄰近的虛擬頁面為邊端頁面。舉例來說，當迴圈暫存器所指向的位址如圖 8B 所示時，設定虛擬頁面 802 為基礎頁面，虛擬頁面 806 為邊端頁面。另一方面，當迴圈暫存器所指向的位址如圖 8C 所示時，設定虛擬頁面 802 為基礎頁面，虛擬頁面 804 為邊端頁面。也就是說，邊端頁面必定是與基礎頁面相鄰的多個虛擬頁面中的其中一個。第一核心 120 會將迴圈暫存器、基礎頁面的實體位址以及邊端頁面的實體位置傳送給第二核心 140。第二核心 140 可以判斷迴圈暫存器的值加上補償值以後是否超出基礎頁面的範圍，並根據基礎頁面與邊端頁面的實體位址來計算出定址結果。

圖 9 為依照本發明實施例說明根據迴圈暫存器計算定址結果的程式碼。

請參考圖 9，本實施例中以 ebp 暫存器做為迴圈暫存

器當做例子。offset 為補償值，paddr_base 為基礎頁面的實體位址，paddr_siding 為邊端頁面的實體位址。在程式碼 901 中，判斷 ebp 暫存器加上補償值以後是否有超出 ebp 暫存器所指的虛擬頁面的範圍。若已超出範圍，在程式碼 902 中，根據 ebp 暫存器與補償值的和計算出在一虛擬頁面的位置，並以邊端頁面的實體位址做為基準，計算出定址結果。若判斷 ebp 暫存器加上補償值以後並沒有超出 ebp 暫存器所指的虛擬頁面的範圍，則以基礎頁面的實體位址作為基準來計算出定址結果。更具體來說，若 ebp 暫存器的值為 0x2fff，則第一核心 120 可判斷基礎頁面為虛擬頁面 802，邊端頁面為虛擬頁面 806(如圖 8B 所示)。此時若補償值為-3，則 ebp 暫存器的值加上補償值以後為地址 0x2ffc，並沒有超出基礎頁面的範圍(即，0x2000~0x2fff)，此時以虛擬頁面 802 的實體位址做為基準計算出定址結果。例如，虛擬位址與實體位址之間轉換的遮罩為 0xf000，則地址 0x2ffc 在基礎頁面實體位址的位移是 0x0ffc。將 0x0ffc 加上基礎頁面的實體位址，則可以得到正確的定址結果。另一方面，若補償值為 3，則 ebp 暫存器的值加上補償值以後為地址 0x3002，已超出基礎頁面的範圍，此時則以邊端頁面為基礎。例如，地址 0x3002 經過遮罩運算後為 0x0002，將邊端頁面(即，虛擬頁面 806)的實體位址加上 0x0002 便可得到定址結果。

如此一來，當第二核心 140 執行的資訊流動碼對應至包括迴圈暫存器的動態定址指令時，第二核心 140 可以根

據迴圈暫存器的值、基礎頁面的實體位址以及邊端頁面的實體位址計算出定址結果。因此，第一核心 120 與第二核心 140 之間便不需要傳送包括迴圈暫存器的動態定址指令的定址結果，藉此提升追蹤資訊流動的效率。第一核心 120 只需要在迴圈暫存器的值改變時，將迴圈暫存器的值、所對應的基礎頁面的實體位置、以及所對應的邊端頁面的實體位址傳送至第二核心 140。並且，由於迴圈暫存器的值並不會經常改變，因此上述傳輸並不會經常發生。

另一方面，本發明還提出一種資訊流動的追蹤方法。

圖 10 為本發明實施例之追蹤方法的流程圖。

在步驟 S1002 中，接收多個指令。並且根據這些指令的執行順序，將指令分割為至少一個程式碼區塊(步驟 S1004)。接著在步驟 S1006 中，將程式碼區塊的指令轉譯為資訊流動碼，並將程式碼區塊的指令轉譯為動態模擬指令(步驟 S1008)。

除此之外，在步驟 S1010 中，將資訊流動碼傳送至電腦系統的第二核心。接著在步驟 S1012 中，設定電腦系統的第一核心執行動態模擬指令以產生動態定址指令的定址結果，並傳送這些定址結果至第二核心。而在步驟 S1014 中，設定第二核心根據這些定址結果執行這些資訊流動碼來追蹤程式碼區塊的指令的資訊流動。

然而，此追蹤方法各步驟的實施方式已詳細說明如上，在此便不再贅述。

綜上所述，本發明實施例所提出的追蹤資訊流動的電

腦系統與追蹤方法，可以讓電腦系統的第一核心與第二核心平行運作。由於指令被切割為程式碼區塊，使得兩核心之間的傳輸次數減少。並且由於設計了資訊流動碼以及動態模擬指令，使得兩核心之間只需要傳送動態的定址結果。除此之外，迴圈暫存器的數值是當改變時才需要傳送至第二核心，藉此進一步的減少兩核心之間的傳輸量。基於這些理由，電腦系統可以更有效率的追蹤指令的資訊流動。並且，除了工作之平行化之優點外，還可以消除原本高耦合性質的動態資訊流動追蹤工作，所造成的快取失誤以及轉譯出指令碼的品質低落兩大缺點，進而達成超過 2 倍以上之效能提升。

雖然本發明已以實施例揭露如上，然其並非用以限定本發明，任何所屬技術領域中具有通常知識者，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，故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。

【圖式簡單說明】

圖 1 為依照本發明一實施例所繪式的電腦系統方塊圖。

圖 2 為依照本發明一實施例說明程式碼區塊的示意圖。

圖 3 為依照本發明一實施例說明程式碼區塊的示意圖。

圖 4 為依照本發明一實施例說明指令類別的示意圖。

圖 5A 至圖 5D 為依照本發明一實施例說明指令的資訊流動的示意圖。

圖 6A 為依照本發明一實施例說明多位元組格式的示意圖。

圖 6B 與圖 6C 為依照本發明一實施例說明單位元組格式的示意圖。

圖 7 為依照本發明一實施例說明指令、動態模擬指令與資訊流動碼的示意圖。

圖 8A 至圖 8C 為依照本發明一實施例所繪示的迴圈暫存器的補償值的示意圖。

圖 9 為依照本發明實施例說明根據迴圈暫存器計算定址結果的程式碼。

圖 10 為本發明實施例之追蹤方法的流程圖。

【主要元件符號說明】

100：電腦系統

120：第一核心

122：二進位轉換器

124：處理單元

126：暫存器

128：動態模擬指令

140：第二核心

142：處理單元

144：暫存器

146：位元映射圖

148：資訊流動碼

160：共用記憶體

162：資料結構

180：指令

200：記憶體

202：第一部分

204：第二部分

300：程式碼區塊

181、183、184：動態定址指令

182：靜態定址指令

410：位元方式覆寫

420：位元方式附加

430：遞增混合

440：全部混合

502a、502b、502c、502d、504a、504b、504c、504d、506a、506b、508a、508b、508c、508d、510a、510b、510c、510d、510e、510f、682、692：位元組

610、612：表頭欄位

620：目標運算元格式欄位

630：來源運算元格式欄位

640：寬度欄位

650：效果欄位

660：目標運算元欄位

670：來源運算元欄位

680：目標運算元偏移欄位

690：來源運算元偏移欄位

701：指令個數傳送指令

702：區塊序號傳送指令

703、706、708、711：模擬執行指令

704、707、709、712：計數指令

705、710、713：定址結果傳送指令

721、722、723、724：資訊流動碼

802、804、806：虛擬頁面

901、902、903：程式碼

S1002、S1004、S1006、S1008、S1010、S1012、S1014：追蹤方法的步驟。

七、申請專利範圍：

1. 一種低耦合的資訊流動的追蹤方法，用於一電腦系統，該電腦系統包括一第一程序與一第二程序，該方法包括：

由該第一程序接收多個指令；

由該第一程序根據該些指令的執行順序，將該些指令分割為至少一程式碼區塊，該至少一程式碼區塊的該些指令包括多個動態定址指令與多個靜態定址指令；

由該第一程序將該至少一程式碼區塊的該些指令轉譯為該至少一程式碼區塊的多個資訊流動碼，該些資訊流動碼對應至該至少一程式碼區塊的該些指令；

由該第一程序將該至少一程式碼區塊的該些指令轉譯為該至少一程式碼區塊的多個動態模擬指令；

由該第一程序將該些資訊流動碼傳送至該電腦系統的該第二程序；

由該第一程序執行該些動態模擬指令以產生該些動態定址指令的多個定址結果，並傳送該些定址結果至該第二程序；以及

設定該第二程序根據該些定址結果執行該些資訊流動碼以追蹤該至少一程式碼區塊的該些指令的多個資訊流動。

2. 如申請專利範圍第 1 項所述之追蹤方法，其中該電腦系統包括一共用記憶體，該共用記憶體包括一資料結構，該資料結構為先進先出的佇列，其中該第一程序將該

些資訊流動碼傳送至該電腦系統的該第二程序的步驟包括：

由該第一程序將該些定址結果寫進該資料結構；以及

由該第二程序從該資料結構中讀取該些定址位置。

3. 如申請專利範圍第 1 項所述之追蹤方法，其中該至少一程式碼區塊的該些動態模擬指令包括多個模擬執行指令，該些模擬執行指令是對應至該至少一程式碼區塊的該些指令，該第一程序執行該些模擬執行指令時，等同於執行所對應的該些指令並產生所對應的該些動態定址指令的該些定址結果。

4. 如申請專利範圍第 3 項所述之追蹤方法，其中該至少一程式碼區塊的該些動態模擬指令包括多個定址結果傳送指令，該第一程序執行該些定址結果傳送指令時，會將該些定址結果傳送至該第二程序。

5. 如申請專利範圍第 3 項所述之追蹤方法，其中該至少一程式碼區塊的該些動態模擬指令包括多個計數指令，該第一程序執行該些計數指令時會計算一指令個數，該指令個數為已成功執行的該至少一程式碼區塊的該些模擬執行指令的個數。

6. 如申請專利範圍第 5 項所述之追蹤方法，其中該至少一程式碼區塊的該些動態模擬指令包括一指令個數傳送指令，該第一程序執行該指令個數傳送指令時會將該指令個數傳送至該第二程序。

7. 如申請專利範圍第 6 項所述之追蹤方法，其中該至

少一程式碼區塊的該些動態模擬指令包括一區塊序號傳送指令，該第一程序執行該區塊序號傳送指令時會將該至少一程式碼區塊的一區塊序號傳送至該共用記憶體。

8. 如申請專利範圍第 7 項所述之追蹤方法，其中由該第二程序根據該些定址結果執行該些資訊流動碼的步驟包括：

由該第二程序根據該區塊序號以決定追蹤該至少一程式碼區塊的該些資訊流動的順序；以及

由該第二程序根據該指令個數來追蹤該至少一程式碼區塊的該些資訊流動。

9. 如申請專利範圍第 1 項所述之追蹤方法，其中該些資訊流動碼符合一多位元組格式，該多位元組格式包括：

一來源運算元格式欄位，用以記錄一來源運算元為記憶體或是暫存器；

一目標運算元格式欄位，用以記錄一目標運算元為記憶體或是暫存器；

一寬度欄位，用以記錄一運算子的一運算寬度；

一效果欄位，用以記錄該運算子的一運算種類；

一來源運算元欄位，用以記錄該來源運算元；以及

一目標運算元欄位，用以記錄該目標運算元。

10. 如申請專利範圍第 1 項所述之追蹤方法，該些資訊流動碼符合一單位元格式，該單位元格式包括：

一效果欄位，用以記錄一運算子的一運算種類；

一來源運算元格式欄位，用以記錄一來源運算元為記

憶體或是暫存器；

一來源運算元欄位，用以記錄該來源運算元；

一來源運算元偏移欄位，用以記錄該來源運算元的一偏移量；

一目標運算元格式欄位，用以記錄一目標運算元為記憶體或是暫存器；

一目標運算元欄位，用以記錄該目標運算元；以及

一目標運算元偏移欄位，用以記錄該目標運算元的一偏移量。

11. 如申請專利範圍第 1 項所述之追蹤方法，該電腦系統包括一迴圈暫存器，該迴圈暫存器的值對應至一基礎頁面，與該基礎頁面相鄰的多個虛擬頁面中包括一邊端頁面，其中該追蹤方法還包括：

當該迴圈暫存器的數值改變時、由該第一程序根據該迴圈暫存器的值計算出所對應的該基礎頁面與該邊端頁面；以及

由該第一程序將該迴圈暫存器的值、該基礎頁面的實體位址、以及該邊端頁面的實體位址傳送至該第二程序。

12. 如申請專利範圍第 11 項所述之追蹤方法，包括：

由該第二程序根據該迴圈暫存器的值、該基礎頁面的實體位址、以及該邊端頁面的實體位址，計算出包括該迴圈暫存器的該些動態定址指令的該些定址結果。

13. 如申請專利範圍第 12 項所述之追蹤方法，其中該電腦系統使用 IA-32 中央處理器的架構，該迴圈暫存器為

ebp 暫存器。

14. 如申請專利範圍第 1 項所述之追蹤方法，其中該電腦系統包括一位元映射圖，該追蹤方法還包括：

由該第二程序在追蹤該些資訊流動時，將被污染的記憶體位置、暫存器或是硬碟位置記錄在該位元映射圖。

15. 一種追蹤資訊流動的電腦系統，包括：

一第一核心，用以接收多個指令，並根據該些指令的執行順序，將該些指令分割為至少一程式碼區塊，其中該至少一程式碼區塊的該些指令包括多個動態定址指令與多個靜態定址指令；以及

一第二核心，耦接至該第一核心；

其中，該第一核心將該至少一程式碼區塊的該些指令轉譯為該至少一程式碼區塊的多個資訊流動碼，該些資訊流動碼對應至該至少一程式碼區塊的該些指令；

其中，該第一核心將該至少一程式碼區塊的該些指令轉譯為該至少一程式碼區塊的多個動態模擬指令；

其中，該第一核心將該些資訊流動碼傳送至該電腦系統的一第二核心；

其中，該第一核心執行該些動態模擬指令以產生該些動態定址指令的多個定址結果，並傳送該些定址結果至該第二核心；

其中，該第二核心根據該些定址結果執行該些資訊流動碼以追蹤該至少一程式碼區塊的該些指令的多個資訊流動。

16. 如申請專利範圍第 15 項所述之電腦系統，更包括：

一共用記憶體，耦接至該第一核心與該第二核心，該共用記憶體包括一資料結構，該資料結構為先進先出的佇列，

其中該第一核心將該些定址結果寫進該資料結構，並且該第二核心從該資料結構中讀取該些定址位置。

17. 如申請專利範圍第 15 項所述之電腦系統，其中該至少一程式碼區塊的該些動態模擬指令包括多個模擬執行指令，該些模擬執行指令是對應至該至少一程式碼區塊的該些指令，該第一核心執行該些模擬執行指令時，等同於執行所對應的該些指令並產生所對應的該些動態定址指令的該些定址結果。

18. 如申請專利範圍第 17 項所述之電腦系統，其中該至少一程式碼區塊的該些動態模擬指令包括多個定址結果傳送指令，該第一核心執行該些定址結果傳送指令時，會將該些定址結果傳送至該第二核心。

19. 如申請專利範圍第 18 項所述之電腦系統，其中該至少一程式碼區塊的該些動態模擬指令包括多個計數指令，該第一核心執行該些計數指令時會計算一指令個數，該指令個數為已被成功執行的該至少一程式碼區塊的該些模擬執行指令的個數。

20. 如申請專利範圍第 19 項所述之電腦系統，其中該至少一程式碼區塊的該些動態模擬指令包括一指令個數傳

送指令，該第一核心執行該指令個數傳送指令時會將該指令個數傳送至該第二核心。

21. 如申請專利範圍第 20 項所述之電腦系統，其中該至少一程式碼區塊的該些動態模擬指令包括一區塊序號傳送指令，該第一核心執行該區塊序號傳送指令時會將該至少一程式碼區塊的一區塊序號傳送至該共用記憶體。

22. 如申請專利範圍第 21 項所述之電腦系統，其中該第二核心根據該區塊序號以決定追蹤該至少一程式碼區塊的該些資訊流動的順序，並且該第二核心根據該指令個數來追蹤該至少一程式碼區塊的該些資訊流動。

23. 如申請專利範圍第 15 項所述之電腦系統，其中該些資訊流動碼符合一多位元組格式，該多位元組格式包括：

一來源運算元格式欄位，用以記錄一來源運算元為記憶體或是暫存器；

一目標運算元格式欄位，用以記錄一目標運算元為記憶體或是暫存器；

一寬度欄位，用以記錄一運算子的一運算寬度；

一效果欄位，用以記錄該運算子的一運算種類；

一來源運算元欄位，用以記錄該來源運算元；以及

一目標運算元欄位，用以記錄該目標運算元。

24. 如申請專利範圍第 15 項所述之電腦系統，該些資訊流動碼符合一單位元格式，該單位元格式包括：

一效果欄位，用以記錄一運算子的一運算種類；

一來源運算元格式欄位，用以記錄一來源運算元為記

憶體或是暫存器；

一來源運算元欄位，用以記錄該來源運算元；

一來源運算元偏移欄位，用以記錄該來源運算元的一偏移量；

一目標運算元格式欄位，用以記錄一目標運算元為記憶體或是暫存器；

一目標運算元欄位，用以記錄該目標運算元；以及

一目標運算元偏移欄位，用以記錄該目標運算元的一偏移量。

25. 如申請專利範圍第 15 項所述之電腦系統，該電腦系統包括一迴圈暫存器，該迴圈暫存器的數值對應至一基礎頁面，與該基礎頁面相鄰的多個虛擬頁面中包括一邊端頁面；

其中當該迴圈暫存器的數值改變時，該第一核心根據該迴圈暫存器的值計算出所對應的該基礎頁面與該邊端頁面，並且該第一核心將該迴圈暫存器的值、該基礎頁面的實體位址、以及該邊端頁面的實體位址傳送至該第二核心。

26. 如申請專利範圍第 25 項所述之電腦系統，其中該第二核心根據該迴圈暫存器的值、該基礎頁面的實體位址、以及該邊端頁面的實體位址，計算出包括該迴圈暫存器的該些動態定址指令的該些定址結果。

27. 如申請專利範圍第 26 項所述之電腦系統，其中該電腦系統使用 IA-32 中央處理器的架構，該迴圈暫存器為 ebp 暫存器。

28. 如申請專利範圍第 15 項所述之電腦系統，其中該第二核心包括一位元映射圖，該第二核心在追蹤該些資訊流動時，將被污染的記憶體位置、暫存器或是硬碟位置記錄在該位元映射圖。

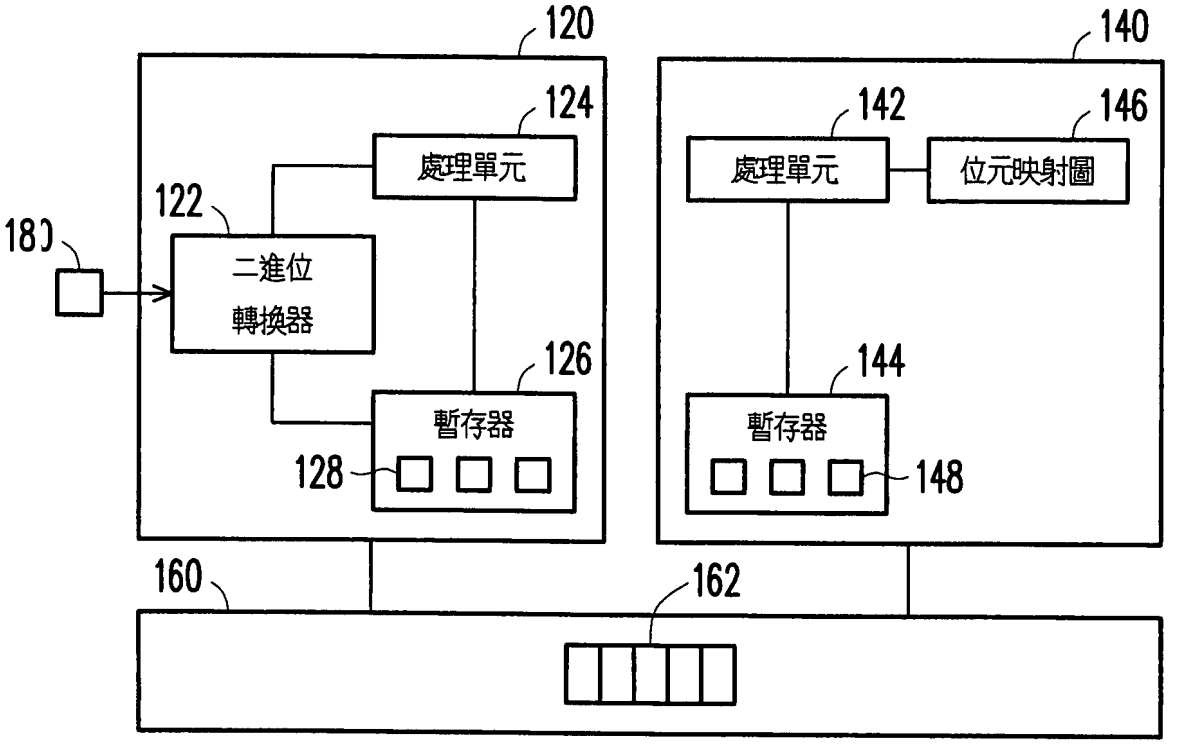

120
140
124
142
146
處理單元
處理單元
位元映射圖
122
180
二進位
轉換器
126
144
暫存器
暫存器
128
148
160
162
100

圖 1

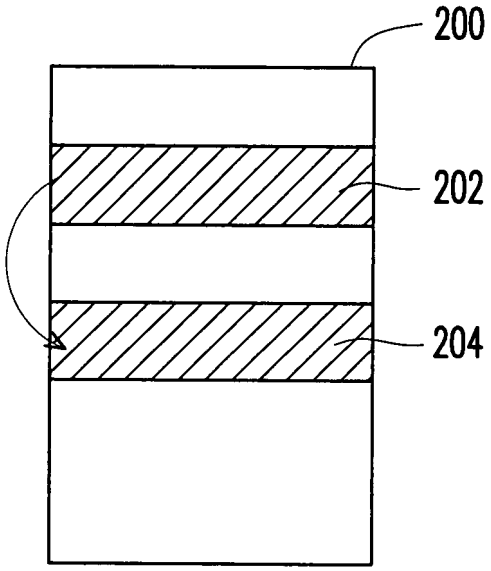

200
202
204

圖 2

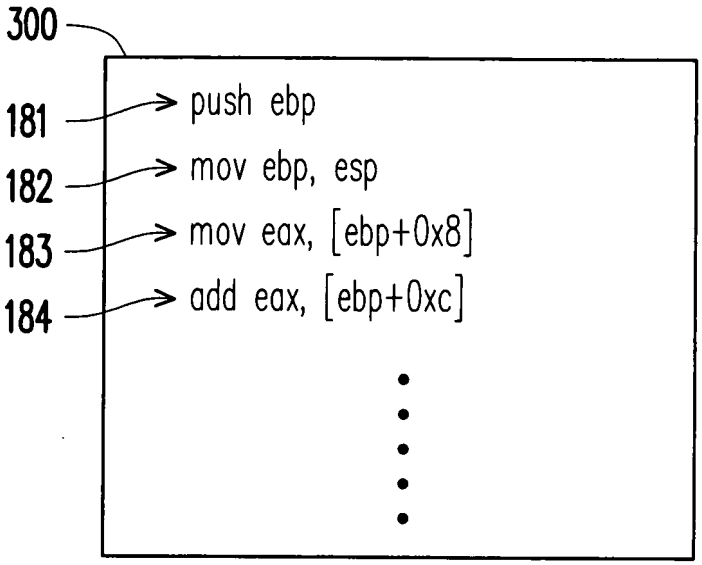

300
181
push ebp
182
mov ebp, esp
183
mov eax, [ebp+0x8]
184
add eax, [ebp+0xc]

圖 3

300

	類別	資訊流動	記號
410	位元方式覆寫	$A[0] \leftarrow B[0]$ $A[1] \leftarrow B[1]$ ⋮ $A[n] \leftarrow B[n]$	$A \Leftarrow nB$
420	位元方式附加	$A[0] \overset{+}{\leftarrow} B[0]$ $A[1] \overset{+}{\leftarrow} B[1]$ ⋮ $A[n] \overset{+}{\leftarrow} B[n]$	$A \overset{+}{\Leftarrow} nB$
430	遞增混合	$A[0] \overset{+}{\leftarrow} B[0]$ $A[1] \overset{+}{\leftarrow} A[0],\ A[1] \overset{+}{\leftarrow} B[1]$ $A[2] \overset{+}{\leftarrow} A[1],\ A[2] \overset{+}{\leftarrow} B[2]$ ⋮ $A[n] \overset{+}{\leftarrow} A[n-1],\ A[n] \overset{+}{\leftarrow} B[n]$	$A \overset{A}{\Leftarrow} nB$
440	遞增混合	$T \leftarrow A[0],\ T \overset{+}{\leftarrow} A[1], \cdots, T \overset{+}{\leftarrow} A[n]$ $T \overset{+}{\leftarrow} B[0],\ T \overset{+}{\leftarrow} B[1], \cdots, T \overset{+}{\leftarrow} B[n]$ $A[0] \leftarrow T,\ A[1] \leftarrow T, \cdots, A[n] \leftarrow T$ $B[0] \leftarrow T,\ B[1] \leftarrow T, \cdots, B[n] \leftarrow T$	$A \overset{+}{\Leftarrow} nB$

圖 4

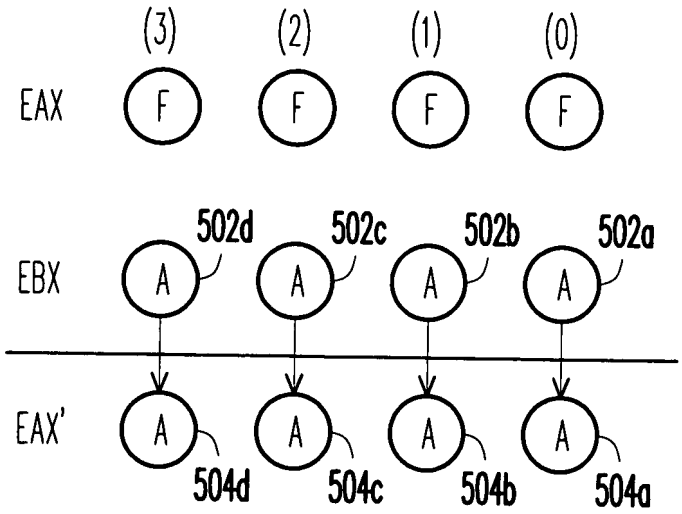

(3)
(2)
(1)
(0)
EAX
F
F
F
F
EBX
502d
502c
502b
502a
A
A
A
A
EAX'
A
A
A
A
504d
504c
504b
504a
mov eax, ebx

圖 5A

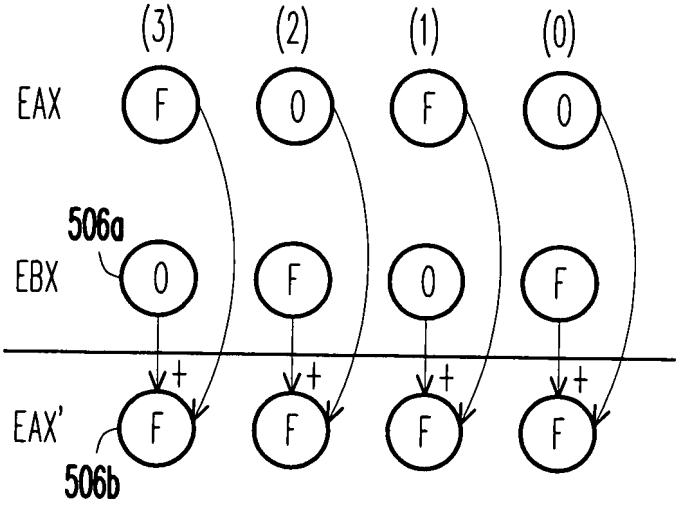

(3)
(2)
(1)
(0)
EAX
F
0
F
0
506a
EBX
0
F
0
F
+
+
+
+
EAX'
F
F
F
F
506b
xor eax, ebx

圖 5B

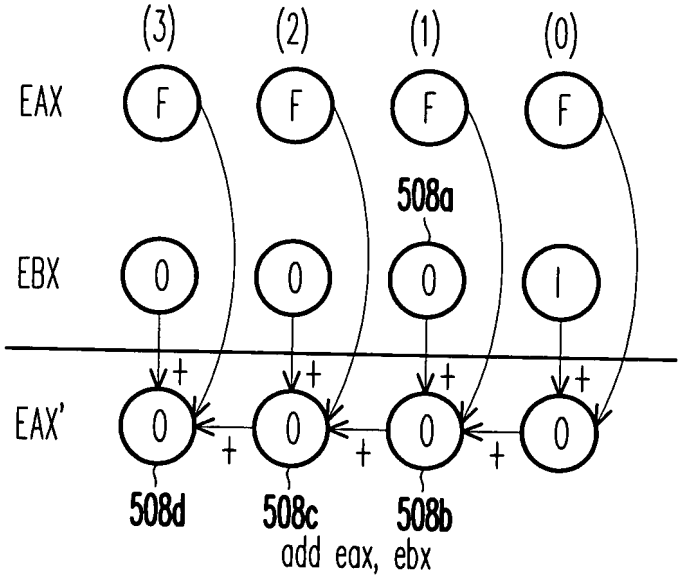

(3)
(2)
(1)
(0)
EAX
F
F
F
F
508a
EBX
0
0
0
1
+
+
+
+
EAX'
0
0
0
0
+
+
+
508d
508c
508b
add eax, ebx

圖 5C

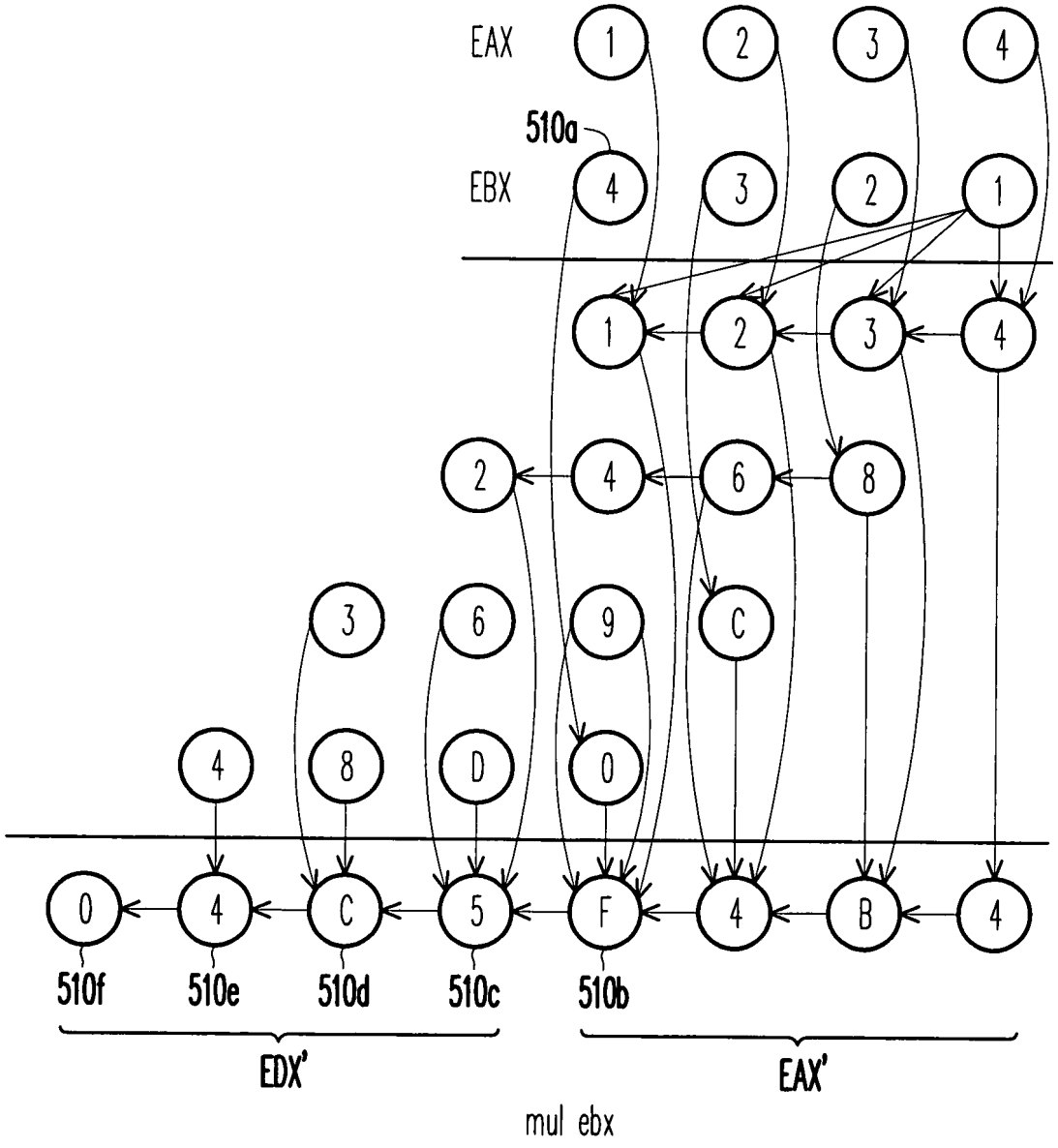

EAX
1
2
3
4
510a
EBX
4
3
2
1
1
2
3
4
2
4
6
8
3
6
9
C
4
8
D
0
0
4
C
5
F
4
B
4
510f
510e
510d
510c
510b
EDX'
EAX'
mul ebx

圖 5D

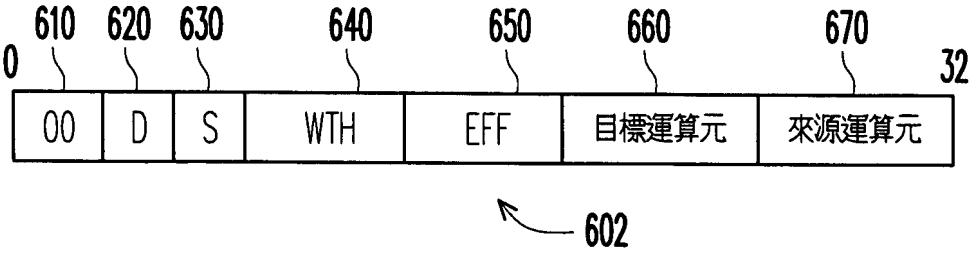

610
620
630
640
650
660
670
0
32
00
D
S
WTH
EFF
目標運算元
來源運算元
602

圖 6A

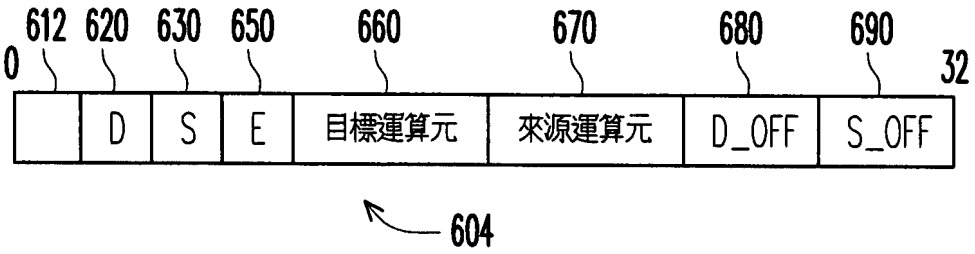

612
620
630
650
660
670
680
690
0
32
D
S
E
目標運算元
來源運算元
D_OFF
S_OFF
604

圖 6B

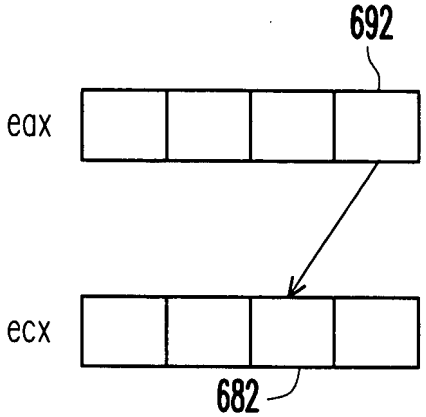

692
eax
ecx
682

圖 6C

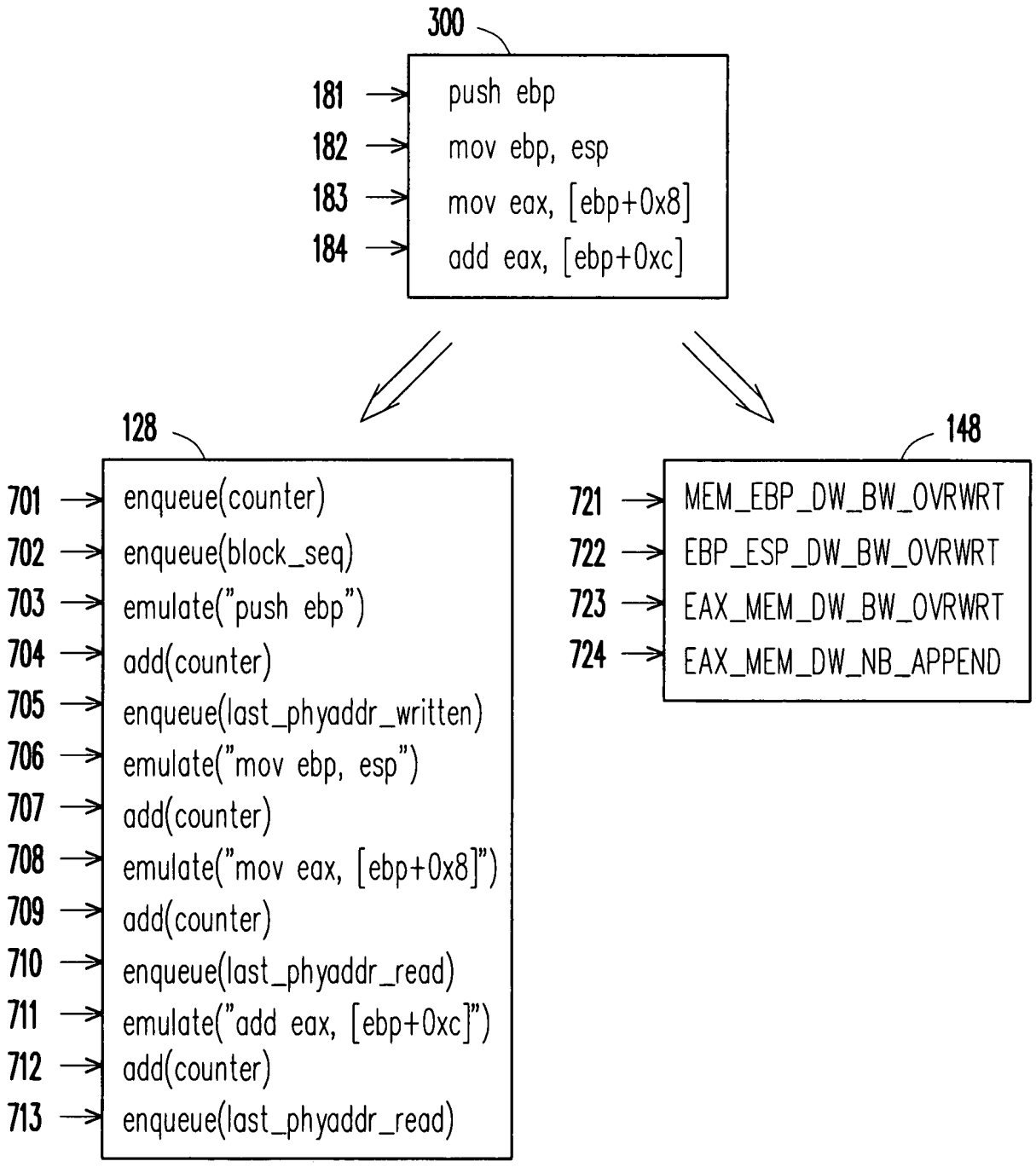

300
181
push ebp
182
mov ebp, esp
183
mov eax, [ebp+0x8]
184
add eax, [ebp+0xc]
128
701
enqueue(counter)
702
enqueue(block_seq)
703
emulate("push ebp")
704
add(counter)
705
enqueue(last_phyaddr_written)
706
emulate("mov ebp, esp")
707
add(counter)
708
emulate("mov eax, [ebp+0x8]")
709
add(counter)
710
enqueue(last_phyaddr_read)
711
emulate("add eax, [ebp+0xc]")
712
add(counter)
713
enqueue(last_phyaddr_read)
148
721
MEM_EBP_DW_BW_OVRWRT
722
EBP_ESP_DW_BW_OVRWRT
723
EAX_MEM_DW_BW_OVRWRT
724
EAX_MEM_DW_NB_APPEND

圖 7

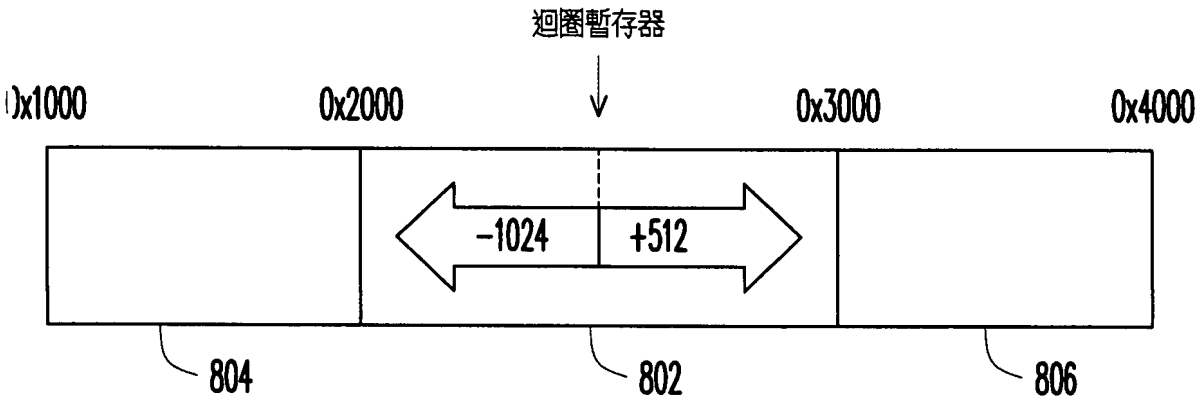

迴圈暫存器
)x1000
0x2000
0x3000
0x4000
-1024
+512
804
802
806

圖 8A

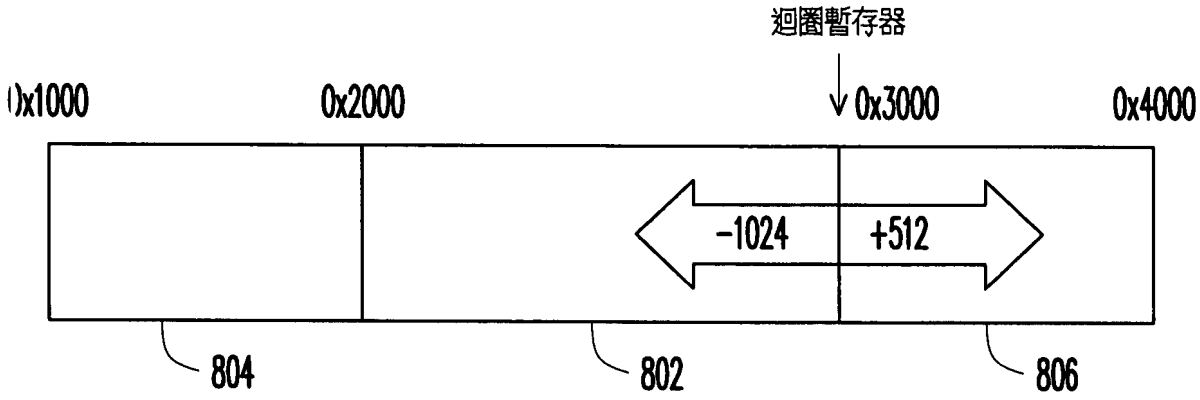

迴圈暫存器
)x1000
0x2000
0x3000
0x4000
-1024
+512
804
802
806

圖 8B

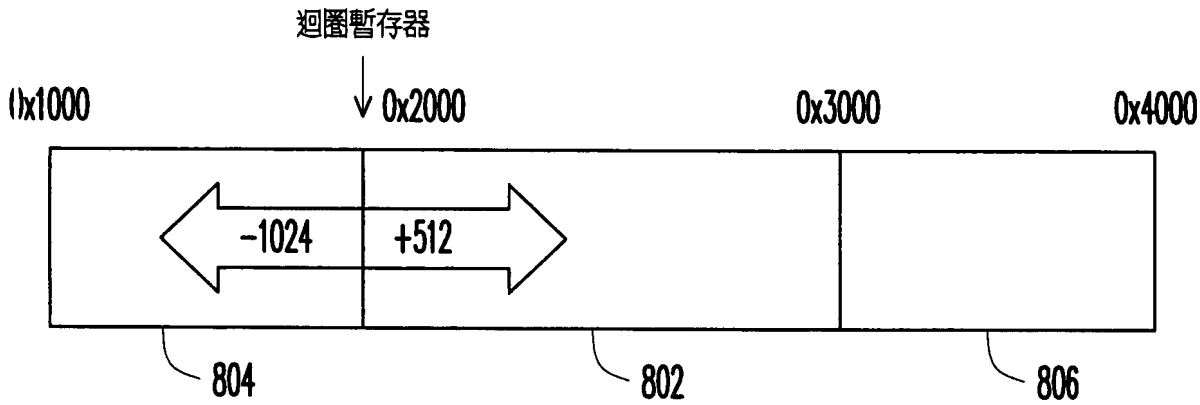

迴圈暫存器
)x1000
0x2000
0x3000
0x4000
-1024
+512
804
802
806

圖 8C

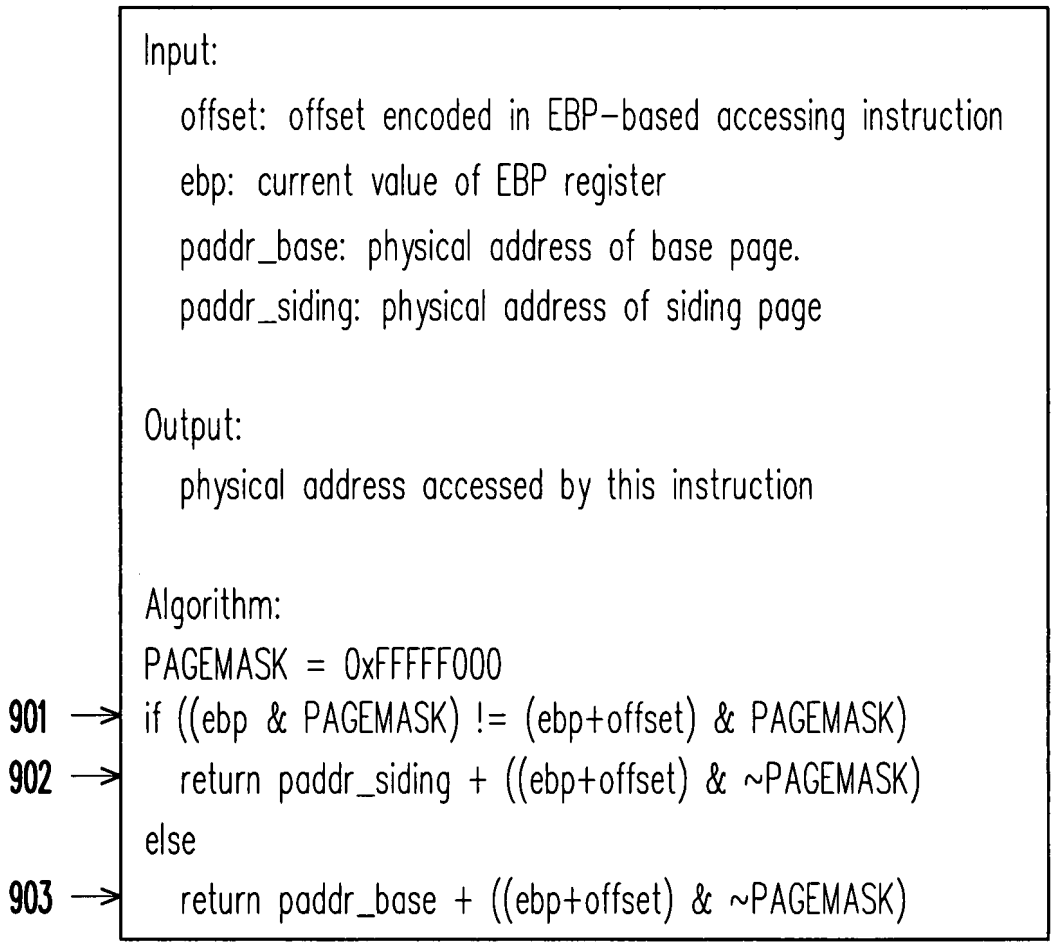

Input:
offset: offset encoded in EBP-based accessing instruction
ebp: current value of EBP register
paddr_base: physical address of base page.
paddr_siding: physical address of siding page
Output:
physical address accessed by this instruction
Algorithm:
PAGEMASK = 0xFFFFF000
901
if ((ebp & PAGEMASK) != (ebp+offset) & PAGEMASK)
902
return paddr_siding + ((ebp+offset) & ~PAGEMASK)
else
903
return paddr_base + ((ebp+offset) & ~PAGEMASK)

圖 9

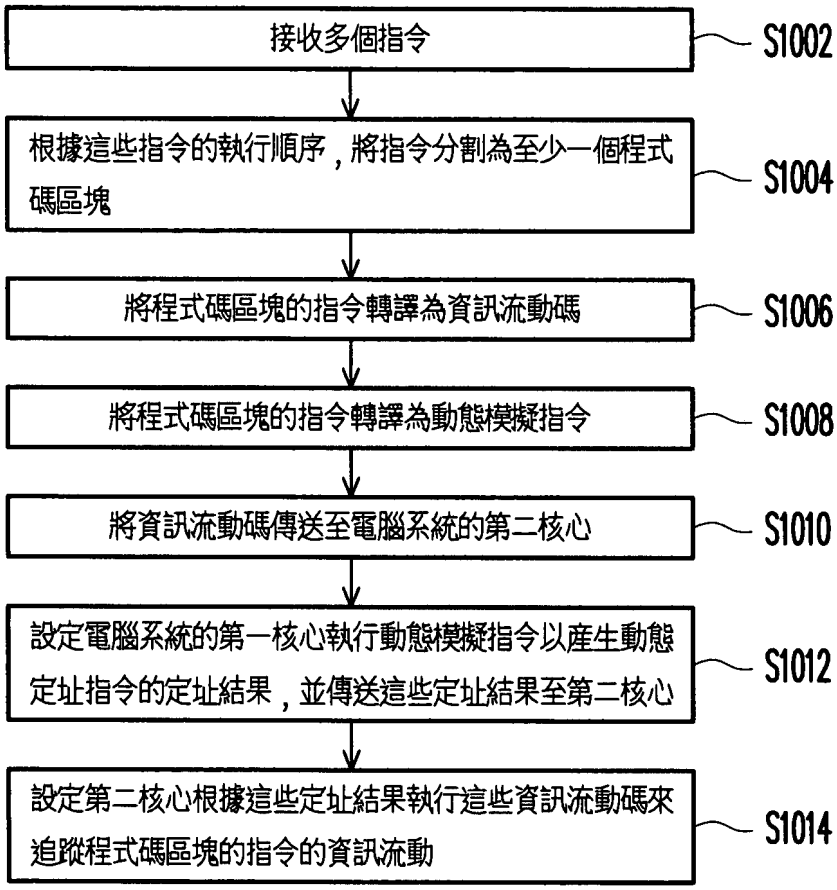

接收多個指令
S1002
根據這些指令的執行順序，將指令分割為至少一個程式碼區塊
S1004
將程式碼區塊的指令轉譯為資訊流動碼
S1006
將程式碼區塊的指令轉譯為動態模擬指令
S1008
將資訊流動碼傳送至電腦系統的第二核心
S1010
設定電腦系統的第一核心執行動態模擬指令以產生動態定址指令的定址結果，並傳送這些定址結果至第二核心
S1012
設定第二核心根據這些定址結果執行這些資訊流動碼來追蹤程式碼區塊的指令的資訊流動
S1014

圖 10