公告本

年 月 日 修正 I266238

申請日期：93、4、12	IPC分類
申請案號：93110089	G06F9/30

(以上各欄由本局填註)

發明專利說明書

一、發明名稱	中 文	適用於超長指令字元之階層式指令編碼方法及其解碼器
	英 文	
二、發明人(共5人)	姓 名(中文)	1. 任建葳 2. 林泰吉 3. 張金祺
	姓 名(英文)	1. 2. 3.
	國 籍(中英文)	1. 中華民國 TW 2. 中華民國 TW 3. 中華民國 TW
	住居所(中 文)	1. 新竹市大學路1003號4樓 2. 高雄縣大寮鄉萬隆街14號 3. 台北市忠孝東路五段980號4樓
	住居所(英 文)	1. 2. 3.
三、申請人(共1人)	名稱或姓 名(中文)	1. 國立交通大學
	名稱或姓 名(英文)	1.
	國 籍(中英文)	1. 中華民國 TW
	住居所(營業所)(中 文)	1. 新竹市大學路1001號 (本地址與前向貴局申請者相同)
	住居所(營業所)(英 文)	1.
	代表人(中文)	1. 張俊彥
	代表人(英文)	1.

I266238

93 4 27

申請日期：	IPC分類
申請案號：	

(以上各欄由本局填註)

發明專利說明書

一、發明名稱	中 文	
	英 文	
二、發明人(共5人)	姓 名(中文)	4.趙至敏 5.劉志尉
	姓 名(英文)	4. 5.
	國 籍(中英文)	4.中華民國 TW 5.中華民國 TW
	住居所(中 文)	4.高雄市苓雅區苓雅一路31號6樓之3 5.新竹市南大路808巷7弄24號
	住居所(英 文)	4. 5.
三、申請人(共1人)	名稱或姓 名(中文)	
	名稱或姓 名(英文)	
	國 籍(中英文)	
	住居所(營業所)(中 文)	
	住居所(營業所)(英 文)	
	代表人(中文)	
	代表人(英文)	

一、本案已向

國家(地區)申請專利	申請日期	案號	主張專利法第二十四條第一項優先權

無

二、□主張專利法第二十五條之一第一項優先權:

申請案號:

無

日期:

三、主張本案係符合專利法第二十條第一項□第一款但書或□第二款但書規定之期間

日期:

四、□有關微生物已寄存於國外:

寄存國家:

寄存機構:

無

寄存日期:

寄存號碼:

□有關微生物已寄存於國內(本局所指定之寄存機構):

寄存機構:

寄存日期:

無

寄存號碼:

□熟習該項技術者易於獲得,不須寄存。

95 9. -8

四、中文發明摘要　（發明名稱：適用於超長指令字元之階層式指令編碼方法及其解碼器）

本發明提供一種適用於超長指令字元之階層式指令編碼方法及其對應之解碼器架構，編碼方法係將超長指令字元(VLIW)之各子指令依照需要轉換成長短不一之指令碼以縮短指令長度，接著將這些長度不固定的指令拆解成固定長度之頭及可變長度之尾欄位，最後加上控制資訊帽(cap)並依特有之排列方式置入一固定長度之指令包裹內，以方便程式記憶體之讀取及指令解碼。控制資訊帽中係包含一有效欄位來移除NOP指令，並同時解決一般可變長度VLIW需額外加入位置分配碼的問題；另外，本發明所特有之指令碼排列方式可採用對數移位器(logarithmic shifter)及漸進式移位器(incremental shifter)組成解碼器，具有簡易且高效率之優點。

五、英文發明摘要　（發明名稱：）

六、指定代表圖

（一）、本案代表圖為：第四(b)圖

（二）、本案代表圖之元件代表符號簡單說明：無

五、發明說明（1）

【發明所屬之技術領域】

本發明係有關一種超長指令集編碼方式及其相對應可行解碼器架構，特別是關於一種可用於高性能超長指令集處理器架構等需精簡程式碼之設計中的指令編解碼技術。

【先前技術】

隨著多媒體及通訊系統資料處理的發展，超長指令字元（Very long instruction word，VLIW）的設計架構已廣泛地運用於一般數位訊號處理器中，在通用處理器也可見其蹤影。然而VLIW指令集之程式碼因二大因素而比一般精簡指令集肥大：第一，基本指令使用固定長度編碼，造成所有的指令需屈就資訊量最多的指令而使編碼效率不彰。第二，各子指令係依其在固定長度之超長指令字元中的位置平行分派至專屬之功能模組執行，當程式碼平行度不足時，必需在超長指令字元中插入NOP指令，而使得程式碼的大小在封裝成超長指令字元後暴增。因此，如何減少程式碼的大小已成為VLIW設計的一大課題。目前常見移除NOP的方式分為下列兩大類：

（A）索引法（Index method）

如第一圖所示，此方法係在處理器配置一有限大小的程式記憶體儲存超長指令字元，外部仍使用一般字長之指令，可大幅降低處理器指令匯流排的頻寬。VLIW程式執行則以此內部程式暫存器的索引值代表其相對應的超長指令字元減少動態之程式碼。因NOP指令不需填入此程式暫存器中，故其初始化所用的靜態程式碼亦不會有傳統VLIW中

五、發明說明（2）

的NOP負擔。但如何決定合適的程式暫存器大小及如何動態配置空間予VLIW指令降低初始化的負擔尚無有效的方法。另外，程式暫存器亦是處理器動態的內部狀態，一旦發生中斷（interrupt）則需將整個程式暫存器備份，代價極高。

（B）移除NOP

如第二圖所示，此方法只編入需被執行運算的基本指令於超長指令中，將無作用的NOP移除以壓縮程式碼。其組合成的超長指令可依可平行執行指令的多寡而有所不同的長度，當平行度不足而必須指派NOP予某些指令槽（slot）時，其程式記憶體不需負擔NOP所佔之空間。此方法的缺失為每個基本指令需加入額外的資訊，用來標明是否為一超長指令的開頭或結尾分隔不同的超長指令；另外必須標明每個基本指令需在那個運算單元執行，以上兩者均會增加程式碼的長度，並且使指令解碼器複雜度增加。

而可變長度指令編碼最常見的是同時提供多套指令集，除了一般三十二位元指令集外，另外提供一組十六位元較短的指令集，某些非關鍵的程式區段可使用短的指令群組撰寫，以減小程式碼大小，有些多指令集之處理器亦支援同一程式區段混雜使用多種指令。其缺陷是只有少數幾種指令長度可選擇，編碼效率無法提升太多。另一類技術係每個指令均有不同長度的編碼，由指令所需之資訊量來決定其長度，其缺點為解碼過程相當複雜，尤其是管線

五、發明說明（3）

化處理器設計，需先對齊（align）指令開頭，才能進行接續的指令處理。

請再參閱第三圖所示，HAT格式是一種改良的可變長度編碼方式，其係將指令分為固定長度的頭（head）和可變長度的尾（tail），並將尾的長度記錄在頭中，再將每個指令的頭由右向左疊加且尾由左向右疊加；最後再將多個不同長度指令包覆在一固定長度的指令包裹（bundle）中。VLIW-HAT則是將上述的HAT格式應用於VLIW架構中，先將基本指令以HAT編碼，而超長指令字元中可平行執行的指令包裝為一可變長度的指令包裹。接著再以不同長度的指令包裹為單位，將其再次拆解為固定長度的頭和可變長度的尾，並再次利用HAT格式組裝成為一個超級包裹（super bundle）。此方式缺陷為其解碼架構窒礙難行。

有鑑於固定長度之指令及程式平行度不足時需添加NOP指令致使傳統VLIW處理器的程式碼密度（code density）極差，而習知使用可變長度編碼或索引法改良之處理器架構亦存在許多問題，本發明係針對上述之種種問題，提出一種適用於超長指令字元之階層式指令編碼及其解碼器，以有效克服習知之該等缺失。

【發明內容】

本發明之主要目的，係在提出一種適用於超長指令字元之階層式指令編碼及其解碼器，以同時實現使指令編碼更精簡且提升單位指令字長所提供之有效運算之目的。

本發明之另一目的，係在有效達成壓縮放置於外部儲

五、發明說明 (4)

存空間的程式碼之功效，並使其解碼器具有架構簡易且可實行、高效能之功效者，以提供一高效能之處理器。

本發明之再一目的，係在大幅節省程式記憶體，特別在平行度不高的應用中，可得到更佳效能的改善。

為達到上述之目的，本發明之適用於超長指令字元之階層式指令編碼方法包括下列步驟：提供複數個超長指令字元(VLIW)指令；以及將該等VLIW指令中可平行執行的基本指令包裝成一指令封包（instruction packet），在每一該指令封包前端內置入一控制資訊帽(Cap)，該控制資訊帽包含一指令有效欄位，用以標明每一指令槽(instruction slot)是否含有有效基本指令。

上述編碼方法所對應之解碼器包括一資訊帽位移器，以便在進行每一解碼循環(cycle)時，將每一控制資訊帽往前位移來對齊下一資訊帽的位置；一資訊帽解碼器係在進行每一該解碼循環時，取所有資訊帽中的前N位元來解碼；數個頭位移器分別依據前述資訊帽中的有效欄位，將有效指令所對應之頭由該指令包裏中取出；指令頭解碼器負責將該等頭位移器所取出之頭進行解碼；以及一尾位移器負責將所有的尾一次取出，提供該有效指令解碼器解碼，該尾位移器對於資料之位移長度係由相對應資訊帽中之尾總長度欄位所決定。

底下藉由具體實施例配合所附的圖式詳加說明，當更容易瞭解本發明之目的、技術內容、特點及其所達成之功效。

五、發明說明 (5)

【實施方式】

本發明係依照不同指令的意涵（semantic）給予各自不同長度的指令編碼，且使用一有效標籤標明有效指令，以移除NOP指令且將用不到的指令槽予以失效(disable)，配合一控制資訊帽(Cap)之設計，使指令編碼更精簡，且提升高效能處理器單位指令字長所提供之有效運算。

適用於超長指令字元之階層式指令編碼方法將詳述於下，首先，將針對指令（instruction）、指令封包（instruction packet）及指令包裹（instruction bundle）之定義及編碼安排作說明。如第四(a)圖所示，其係為一ADD/SUB之指令編碼，在本方法中，指令係指VLIW處理器中每週期所需之單一指令封包中的每個基本命令，其編碼方式為可變長度編碼，依其所需資訊多寡付予恰好之長度，且由固定長度的頭（head）和可變長度的尾（tail）所構成。其中頭主要包含了操作碼（opcode）及其運算元（operand）的資訊，假如頭的長度不足以將所需的資訊含入，則將剩餘的資訊編入尾中。

請再參考第四(b)圖所示，指令封包係為VLIW處理器每週期所需要的所有指令資訊，其包含數量不定且長度不定的多個指令者，而其內含之指令個數係依照該週期指令之平行度決定，對單一週期最多可分派N個指令的微處理器而言，一指令封包為最多有N個指令的集合。在指令封包中，每一指令係根據其順序將其頭和尾填入封包中；在每一指令封包中均包含有一專屬的控制資訊帽(以下簡稱

五、發明說明（6）

Cap），Cap以指令封包類別（Cap Type）為首，主要包含有兩大類別，分別為VLIW指令及流程控制指令。VLIW指令之Cap另外包含了有效欄位（valid）、尾總長度和硬體特殊資訊；流程控制指令之Cap則儲存所需之控制資訊。VLIW指令中的有效欄位用來標明此指令封包中有多少個有效指令及其位置，以避免使用額外的空間來作為NOP指令編碼；而尾總長度則記錄在此指令封包中所有指令之尾長度的總和，用以計算下一個指令封包在指令包裹的位置；硬體特殊資訊則記錄此指令封包在分派時，硬體上所需要的額外資訊，視不同硬體設計而添加。

再來，為了方便記憶體儲取，再將複數個指令封包包成一固定長度的指令包裹（instruction bundle）。封包的包裝方式為依序將每一指令封包的資訊帽與頭-尾分開置入一指令包裹的前端及尾端，Cap係由前端往尾端疊加而頭-尾則是由尾端往前端疊加逆向排入，直至剩餘空間無法儲放下一指令封包為止，而每個指令封包中的封包類別亦用來標明一指令包裹的結束。

適用於超長指令字元之階層式指令編碼方法如第五圖所示，為其編碼流程圖，首先，如步驟S10開始進行編碼，提供複數個超長指令字元(VLIW)指令，接著如步驟S12，查看有效指令，且判斷該等指令是否為流程控制指令，若為一般指令而非流程控制指，則將每一VLIW指令拆解成固定長度的頭及可變長度的尾而形成一頭-尾配置；接著如步驟S14，對於該等VLIW指令，將可平行執行的每N

五、發明說明 (7)

個VLIW指令包裝成一指令封包，且在每一指令封包之前端內置入一Cap，該Cap係包含指令封包類別、有效欄位、尾總長度及硬體特殊資訊四欄位；其中，若指令經判斷後係為流程控制指令，由於流程控制指令無法平行執行，一次只會有一個指令被執行，即整個指令封包中只有一個指令，因此不採用頭-尾之方式來編碼，利用Cap中原本用來儲存頭-尾解碼資訊的欄位來編碼。亦即在Cap中緊接著封包類別欄位即是指令資訊，不含有效欄位等VLIW指令之資訊，其若Cap放不下所有資訊，則仿效VLIW指令將剩下資訊放至尾，尾可隨資訊量大小改變。換句話說，可變長度之流程控制指令亦被拆解為一固定長度之Cap及一可變長度之尾。

在完成該等指令封包之包裝後，接下來進入步驟S16之在目前包裹中分派空間之步驟，此步驟係將該等指令封包包成一固定長度的指令包裹，依序將每一指令封包的Cap與該頭-尾分開，以便分別置入該指令包裹的前、尾端內；當每一指令封包完成插入於該指令包裹之後，則進行步驟S18之詢問步驟，判斷此包裹之空間是否已用完，若已用完，則如步驟S20終止包裹且將其標示為"11"，並進行步驟S22，提供一新包裹之空間，以便再繼續進行步驟S16。

在步驟S18中，若此包裹之空間尚未用完，則進行步驟S24，繼續將下一指令封包進行拆解即分派而置入此包裹內，且回到步驟S12，並重複上述步驟，直至剩餘空間

五、發明說明 (8)

無法儲放下一該指令封包為止，當所有指令皆已包裹後，則如步驟S26，結束編碼。

其中，在步驟S16中，將數個指令封包包成一固定長度的指令包裹之包裹方法如第六圖所示，首先進行步驟S160，依序將每一指令封包的Cap與該頭-尾分開，以便分別置入該指令包裹的前、尾端內，此時係將一指令封包中之所有"Cap"由該左往右依序排入；之後如步驟S162，將每一指令封包中之所有"頭"由該右往左依序排入；旋即步驟S164，在該等頭之後插入剩餘之"尾"。

第七圖為本發明之階層式指令編碼相對應的指令解碼器架構方塊示意圖，此解碼器係適用於具有複數固定長度之Cap的指令包裹(bundle)，每一指令包裹之該等Cap的資料寬度為M位元(bits)，此解碼器包括負責解譯下一個Cap資訊的Cap解碼架構12以及一頭-尾解碼架構14；頭-尾解碼架構14係包含四層的頭位移器142以取出固定長的頭，且有一尾位移器144再將可變長度的尾一次取出，而一起送至尾解碼器148進行解碼。

其中，Cap解碼架構12包含一粗略分枝位移器（coarse branch shifter）120、一Cap位移器122及一Cap解碼器124，另在頭-尾解碼架構14中亦包含有一粗略分枝位移器140，二粗略分枝位移器120、140係負責在碰到分枝指令時，在該等Cap送至Cap位移器122處理之前，先由指令中帶的分頁號碼（page number）和索引（index）算出下一指令封包的位置，並將其Cap和頭分別

五、發明說明 (9)

位移到最前端和最尾端，再使該指令包裹由該Cap位移器122及該等頭位移器142進行處理。Cap位移器122係在進行每一解碼循環(cycle)時，將每一Cap往前位移，以對齊下一Cap的位置；Cap解碼器124則係在進行每一解碼循環時，取該M位元中的前N位元來解碼，所取出解碼之N位元係可大於每一Cap的長度，以利用多取的長度來判斷目前之指令封包是否已為指令包裹結尾。

在前述四層的頭位移器142中，每一頭位移器142係為對數位移器，其係依據前一時序解出Cap中的有效欄位，將有效指令所對應之頭由指令包裹中取出，以便傳送至一頭解碼器146將頭進行解碼；尾位移器144亦為對數位移器，其係負責將所有的尾一次取出，提供尾解碼器148進行解碼，尾位移器144對於資料之位移長度係由相對應Cap中之尾總長度欄位所決定。其中，頭解碼器146及尾解碼器148亦可合併為一有效指令解碼器。

在本編碼方法中，Cap中之封包類別欄係用以在進行解碼時，處理器可區別目前所抓取(Fetch)之指令封包為一般可重複解譯指令或流程控制指令，此欄位係為一預先處理機制，藉以在每一循環抓取至尾巴時，當發現下一指令封包為控制指令時，可立即停止位移，以便先行作相關處理。

至此，本發明之適用於超長指令字元之階層式指令編碼方法及其解碼器的精神已說明完畢，以下特以一具體範例來詳細驗證說明本發明之編碼方法及其相對應之解碼

五、發明說明（10）

器，以使熟習此項技術者將可參酌此範例之描述而獲得足夠的知識而據以實施。

（A）編碼方法

以下為一4-way VLIW的實現範例。指令編碼主要分VLIW指令和流程控制指令。

第八圖示範VLIW指令編碼方式，每個指令先依其資訊量決定編碼長度，編碼原則為先擺入操作碼（opcode）和運算模式（function code），依續為運算元（operand），常數則擺最後。當所有指令編碼完後，選定頭之長度，以便將指令拆為固定長度的頭和可變長度的尾；若頭放不下所有資訊，則依剩下資訊量決定可變長度尾的大小。以圖中的ADD/SUB 指令為例，因其需三個運元各為4 bits，再加上操作碼和運算模式已超出頭長度，因而將其中一運算元放於尾中。

在每個基本指令均以頭-尾編碼完後，將同一週期中可平行執行的數個指令和一Cap包裝成指令封包，如第九圖所示，每一Cap中之四個欄位分別代表指令封包的種類、有效欄位、尾總長度及硬體特殊資訊。封包類別欄位中，內含一般指令的指令封包其值為00。依序為有效欄位，其所佔位元數同等於同一週期中最多可同時平行運算指令數，此例為4-wayVLIW，即每週期最多有四個指令可同時平行運算，所以其有效欄位佔4bits，而其中1代表相對應位置之指令為有效指令，在解碼時需被處理；0代表NOP指令，其指令不佔用指令編碼空間。Cap中的第三個欄

五、發明說明（11）

位為尾總長度，而此例中以4bits為尾之基本長度單位，即所有指令尾之總和必為四的倍數，此欄位共佔四位元，因此最大尾總長度為15 * 4 = 60 bits。另外，最後尾端的偏移量(offset)欄位則為硬體額外控制訊號。

以第九圖右下方之指令為例，因其為一般指令，所以Cap前2bits為00；而4個指令中前2個指令為NOP，所以有效欄位為0011；2個ADD 指令分別帶4 bits的尾，尾總長度為8 bits，故Cap中尾總長度欄位其值為 2 = $(0010)_2$。另一類指令為流程控制指令，由於其無法平行執行，一次只會有一個指令被執行，即整個指令封包中只有一個指令，因此不採用頭-尾之方式來編碼，改為利用Cap中原本用來儲存頭-尾解碼資訊的欄位來編碼。其前2 bits依然代表封包類別，其中10和01分表代表不同程式流程控制指令，11代表此為指令包裹結尾，而剩下的位元則用來存放指令資訊，編碼原則為先擺入操作碼（opcode）和運算模式（function code），依續為運算元（operand），常數則擺最後，若Cap放不下所有資訊，則仿效VLIW指令則將剩下資訊放至尾，尾可隨資訊量大小改變，換句話說，可變長度之流程控制指令亦被拆解為一固定長度之Cap及一可變長度之尾。如第十圖所示。

而最後一類的指令封包類別為11，為一標示性質指令封包，代表已至指令包裹結尾，其中不含任何指令，亦不含一般指令之指令封包中的頭-尾解碼資訊的欄位。

（B）解碼器架構

五、發明說明 (12)

以一4-way的VLIW為例，假設Cap和所有基本指令其頭-尾長度都已選定，因為指令封包最多有4個指令可同時執行，所以有效欄位佔4個位元，其中1代表有效指令在解碼時需被處理；0代表NOP指令，不佔用指令編碼空間。每一指令的尾長度以4 bits為單位，Cap中尾總長度的欄位為4 bits，所以最多可表示 15 * 4 = 60 bits。假設每個指令包裹的長度為1024 bits且最多可含32個指令封包，則其解碼器可設計如第十一圖。此解碼器因每個Cap的長度是12 bits，而一個指令包裹最多含有32個指令封包，所以其Cap解碼器的資料寬度為12 * 32 = 384 (bits)。

在進行指令抓取(Fetching)時，在每一循環(cycle)下，Cap位移器會固定移除12 bits資料，以對齊下一個Cap的位置；Cap解碼器每次取384 bits的前14 bits來解碼，多取的2 bits其一功用為判斷下一指令封包是否已為指令包裹結尾，以決定下一週期是否抓取新的指令包裹。而另一目的為當發現下一指令封包為控制指令時，可立即停止位移，以便先行作相關處理。

而右半邊的四個頭位移器負責依前一個時序解出Cap中的有效欄位，將頭由指令包裹中取出。以圖中有效欄位之0011為例，代表目前所抓取之指令封包的4個指令中，前2個指令為NOP，不佔用指令封包編碼空間，後2個指令為有效指令，需被解碼處理，故H0位移器不動作且將頭-尾資訊往H1位移器送；由於第二個指令亦為NOP，故H1位

五、發明說明 (13)

移器送亦不動作且將頭-尾資訊繼續往H2位移器送；第三個指令為有效指令，故H2位移器將頭資訊位移12bits至頭解碼器；第四個指令亦為有效指令，故H3位移器將頭資訊位移12bits至頭解碼器。接下來，由對數位移之尾位移器將所有的尾一次取出以便提供尾解碼器解碼，其位移長度由相對應Cap中尾長度欄位決定。

在上述解碼架構中，更可包含一Cap緩衝器及一頭-尾緩衝器，以增加資料傳輸的速度及穩定性。

在上述解碼架構中，更可包含一Cap緩衝器及一頭-尾緩衝器，以增加資料傳輸的速度及穩定性。

本發明依照不同指令的意涵（semantic）給予各自不同長度的指令編碼，且利用一有效欄位將NOP指令從指令編碼中移除，可節省大約32%~50%程式記憶體，若在平行度不高的應用中，所得到的改善效能更佳。

因此，本發明針對移除NOP的問題，在編碼時加入有效欄位標明有效指令，係可解決指令分配器需找出指令與對應執行單元的問題。另，針對可變長度編碼，有別於習知HAT-format編碼方式，利用Cap提供控制資訊，可達成易於解碼的指令封裝模式，且簡易之解碼器架構得以實現。故本發明同時實現使指令編碼更精簡且提升單位指令字長所提供之有效運算之目的，以有效達成壓縮放置於外部儲存空間的程式碼之功效，並使其解碼器具有架構簡易且可實行、高效能之功效者，以提供一高效能之處理器。

以上所述係藉由實施例說明本發明之特點，其目的在

五、發明說明（14）

使熟習該技術者能瞭解本發明之內容並據以實施，而非限定本發明之專利範圍，故，凡其他未脫離本發明所揭示之精神所完成之等效修飾或修改，仍應包含在以下所述之申請專利範圍中。

圖式簡單說明

圖式說明：

第一圖為習知索引法之編碼架構示意圖。

第二圖為習知移除NOP以壓縮程式碼之編碼架構示意圖。

第三圖為習知HAT format之編碼架構示意圖。

第四(a)圖為本發明對於指令之編碼安排的配置示意圖。

第四(b)圖為本發明之指令封包及指令包裹之編碼安排的關係圖。

第五圖為本發明之編碼流程圖。

第六圖為本發明將指令封包包成指令包裹之方法流程圖。

第七圖為本發明之指令解碼器的架構方塊示意圖。

第八圖為本發明之VLIW指令編碼實施例。

第九為本發明之VLIW指令封包與指令包裹關係的實施例。

第十為本發明之流程控制指令的編碼實施例。

第十一圖為本發明之解碼器實施例。

圖號說明：

12 Cap解碼架構	120 粗略分枝位移器
122 Cap位移器	124 Cap解碼器
14 頭-尾解碼架構	140 粗略分枝位移器
142 頭位移器	144 尾位移器
146 頭解碼器	148 尾解碼器

六、申請專利範圍

1. 一種適用於超長指令字元之階層式指令編碼方法，包括下列步驟：

提供複數個超長指令字元(VLIW)指令；以及

將該等VLIW指令中可平行執行的基本指令包裝成一指令封包（instruction packet），在每一該指令封包前端內置入一控制資訊帽(Cap)，該控制資訊帽包含一指令有效欄位，以標明每一指令槽(instruction slot)是否含有有效基本指令。

2. 如申請專利範圍第1項所述之階層式指令編碼方法，其中，在進行指令封包之包裝步驟前，更包括一步驟，係將該等VLIW指令中之每一該基本指令拆解成固定長度的頭及可變長度的尾而形成一頭-尾配置。

3. 如申請專利範圍第2項所述之階層式指令編碼方法，其中，在完成該等指令封包之包裝後，更包括一步驟，係為將該等指令封包包成一固定長度的指令包裹之步驟，其係包含：

依序將每一該指令封包的控制資訊帽與該頭-尾分開而分別置入該指令包裹的前、尾端內，使該等控制資訊帽由該前端往該尾端依序排入，且該等頭由該尾端往該前端依序排入，最後該等尾則接續排入該等頭之後；以及

將每一該指令封包重覆同樣動作填入該指令包裹，直至剩餘空間無法儲放下一該指令封包為止。

4. 如申請專利範圍第3項所述之階層式指令編碼方法，其中，該控制資訊帽更包括一指令封包類別欄位，以標明該

六、申請專利範圍

指令封包為一般指令、流程控制指令及指令包裹的結束其中之一類別者。

5. 如申請專利範圍第3項所述之階層式指令編碼方法，其中，每一該指令包裹的結束係在該封包類別中以特定碼標示。

6. 如申請專利範圍第1項所述之階層式指令編碼方法，其中，該控制資訊帽更包含一尾總長度欄位，以記錄該指令封包中之所有指令的尾長度的總和。

7. 如申請專利範圍第1項所述之階層式指令編碼方法，其中，在該控制資訊帽內更包括一硬體特殊資訊欄，以記錄該指令封包在分派時硬體上所需要的額外資訊。

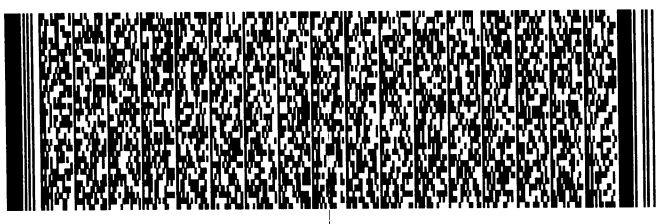
8. 如申請專利範圍第1項所述之階層式指令編碼方法，其係使用該有效欄位移除NOP指令。

9. 如申請專利範圍第1項所述之階層式指令編碼方法，其中，該有效欄位中之有效指令及無效指令分別係以1及0表示。

10. 如申請專利範圍第1項所述之階層式指令編碼方法，其係用於可平行執行的指令之編碼。

11. 如申請專利範圍第1項所述之階層式指令編碼方法，其中，在進行該指令封包之包裝步驟時，係將可平行執行的每N個該VLIW指令包裝成一係用於可平行執行的指令之編碼。

12. 一種解碼器，其係適用於具有複數固定長度之控制資訊帽的指令包裹(bundle)，每一該指令包裹之該等控制資



六、申請專利範圍

訊帽的資料寬度為M位元(bits)，每一該控制資訊帽係代表一指令封包之控制資訊，其中包含一有效欄位以標明指令為有效/無效及其位置，該解碼器包括：

一資訊帽位移器，其係在進行每一解碼循環(cycle)時，將每一該控制資訊帽往前位移，以對齊下一該控制資訊帽的位置；

一資訊帽解碼器，其係在進行每一該解碼循環時，取該M位元中的前N位元來解碼；

數個頭位移器，每一該頭位移器係依據前一時序解出該控制資訊帽中的有效欄位，將有效指令所對應之頭由該指令包裹中取出；

一有效指令解碼器，其係將該等頭位移器所取出之頭進行解碼；以及

一尾位移器，其係負責將所有的尾一次取出，提供該有效指令解碼器解碼，該位移器對於資料之位移長度係由相對應資訊帽中之尾總長度欄位所決定。

13. 如申請專利範圍第12項所述之解碼器，其中，更包括至少一粗略分枝位移器（coarse branch shifter），以在碰到分枝指令時，在該等控制資訊帽送至該資訊帽位移器處理之前，先將當，由指令中帶的分頁號碼（page number）和索引（index）算出下一指令封包的位置，並將其資訊帽和頭分別位移到最前端和最尾端，再使該指令包裹由該資訊帽位移器及該等頭位移器進行處理。

14. 如申請專利範圍第12項所述之解碼器，其中，該資訊

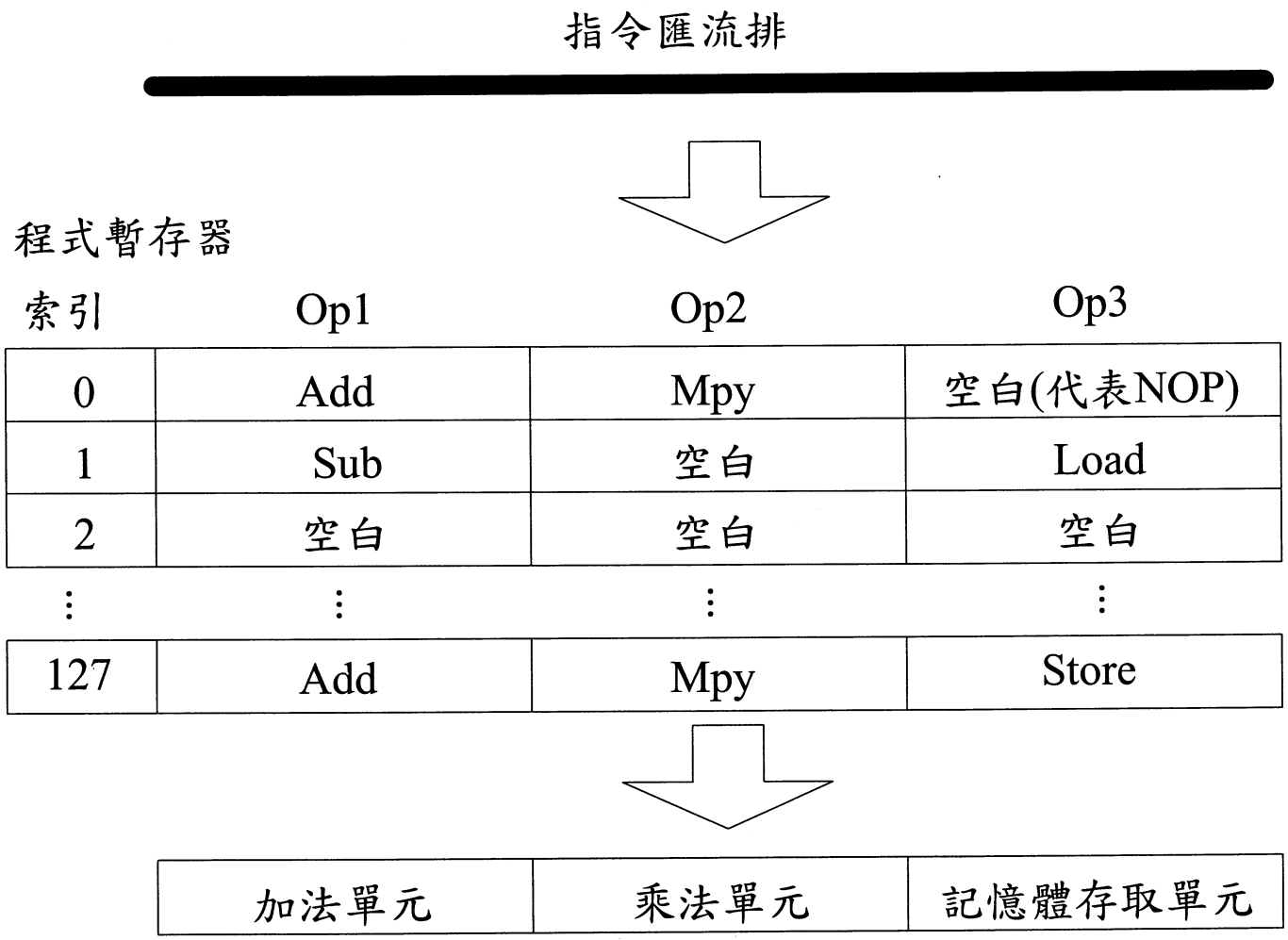
六、申請專利範圍

帽解碼器在進行每一該解碼循環時，所取出解碼之N位元係大於每一該資訊帽的長度，以利用多取的長度來判斷目前之指令封包是否已為指令包裹結尾。

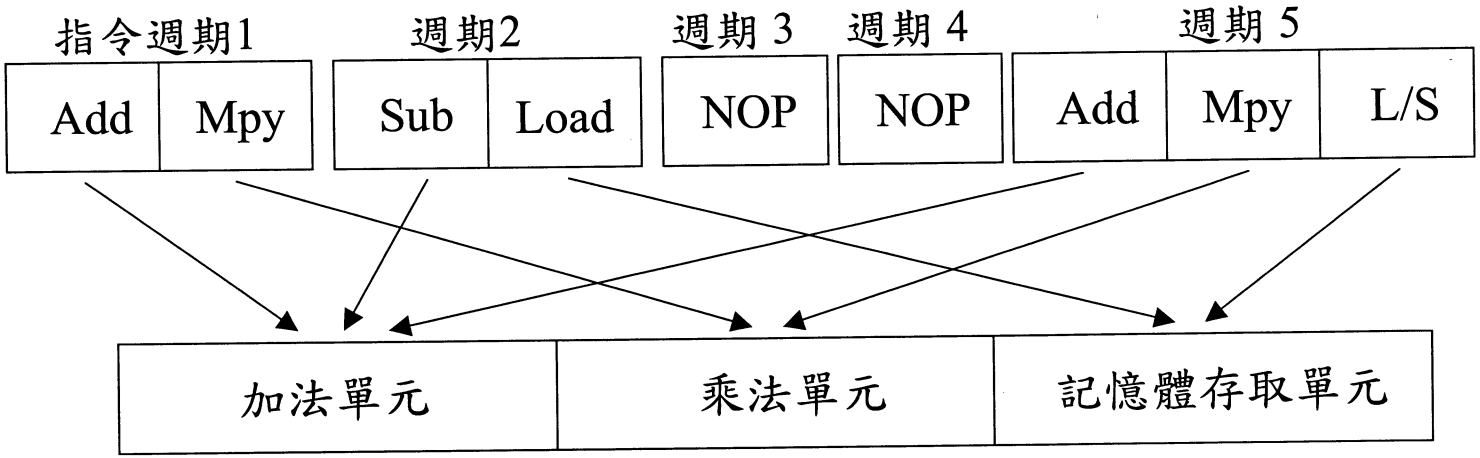
15. 如申請專利範圍第12項所述之解碼器，其中，該等頭位移器係為對數位移器。

16. 如申請專利範圍第12項所述之解碼器，其中，該尾位移器為對數位移器。

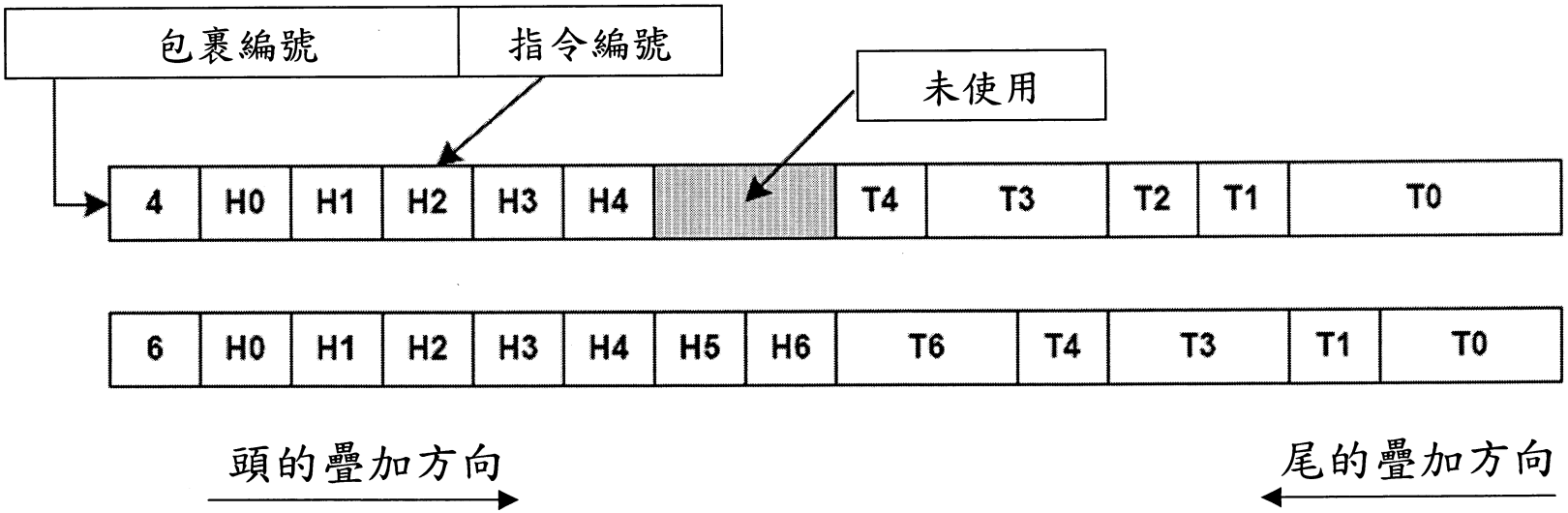
17. 如申請專利範圍第12項所述之解碼器，其中，該有效指令解碼器包括一頭解碼器及一尾解碼器。


指令匯流排
程式暫存器
索引
Op1
Op2
Op3
0
Add
Mpy
空白(代表NOP)
1
Sub
空白
Load
2
空白
空白
空白
127
Add
Mpy
Store
加法單元
乘法單元
記憶體存取單元

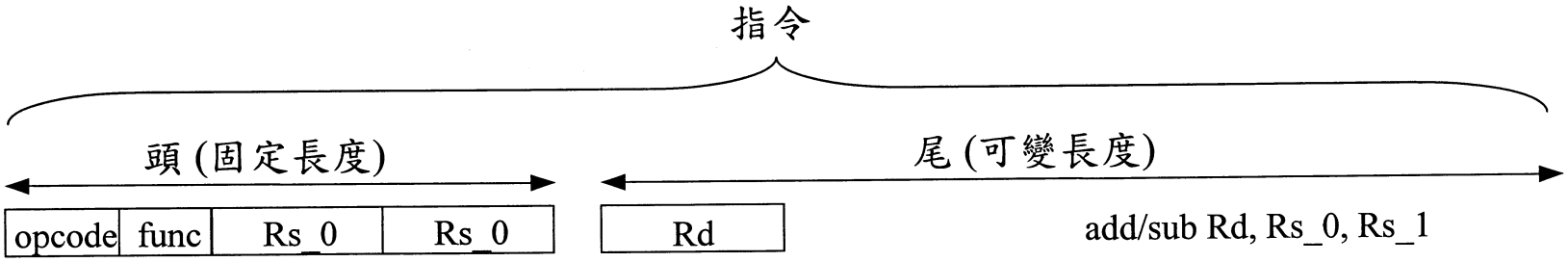
第一圖


指令週期1
週期2
週期 3
週期 4
週期 5
Add
Mpy
Sub
Load
NOP
NOP
Add
Mpy
L/S
加法單元
乘法單元
記憶體存取單元

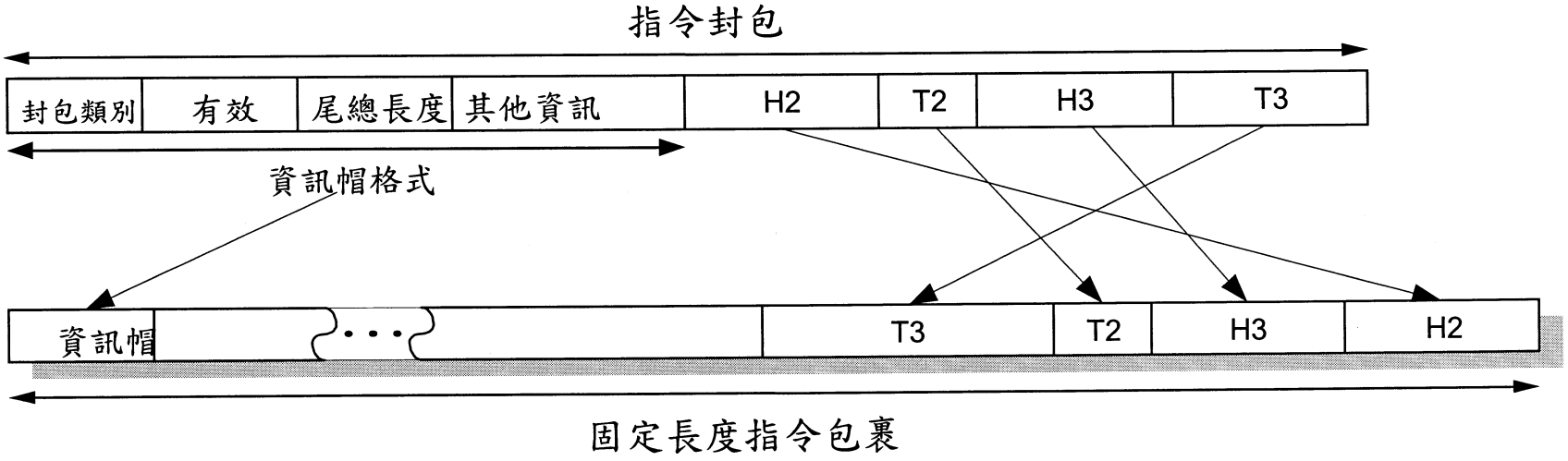
第二圖


包裹編號
指令編號
未使用
4
H0
H1
H2
H3
H4
T4
T3
T2
T1
T0
6
H0
H1
H2
H3
H4
H5
H6
T6
T4
T3
T1
T0
頭的疊加方向
尾的疊加方向

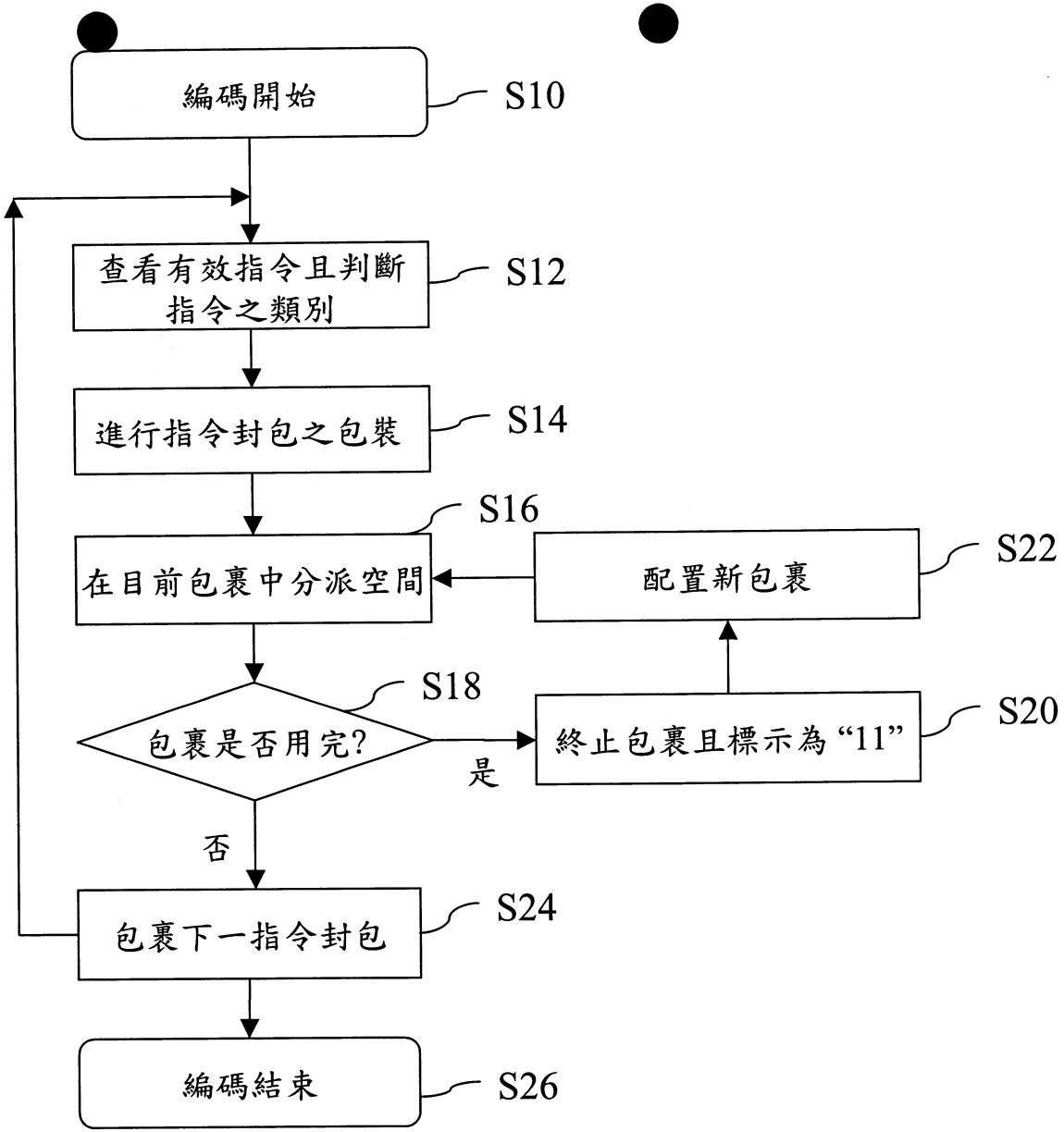
第三圖


指令
頭 (固定長度)
尾 (可變長度)
opcode
func
Rs_0
Rs_0
Rd
add/sub Rd, Rs_0, Rs_1

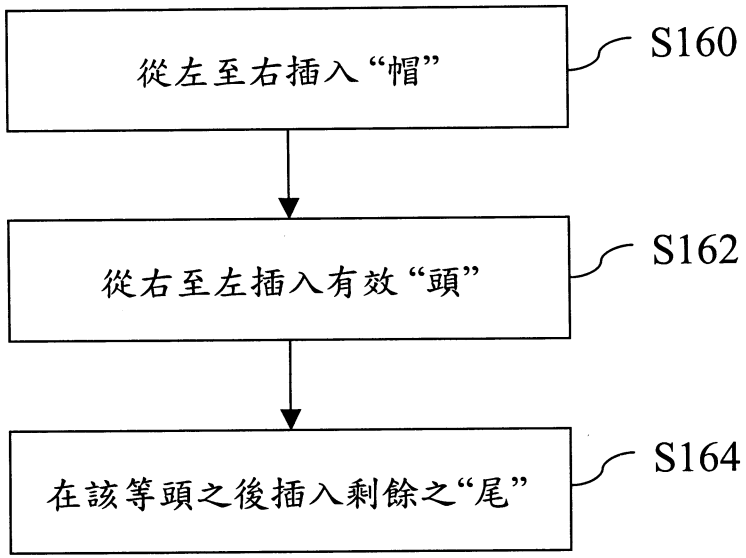
第四(a)圖


指令封包
封包類別
有效
尾總長度
其他資訊
H2
T2
H3
T3
資訊帽格式
資訊帽
T3
T2
H3
H2
固定長度指令包裹

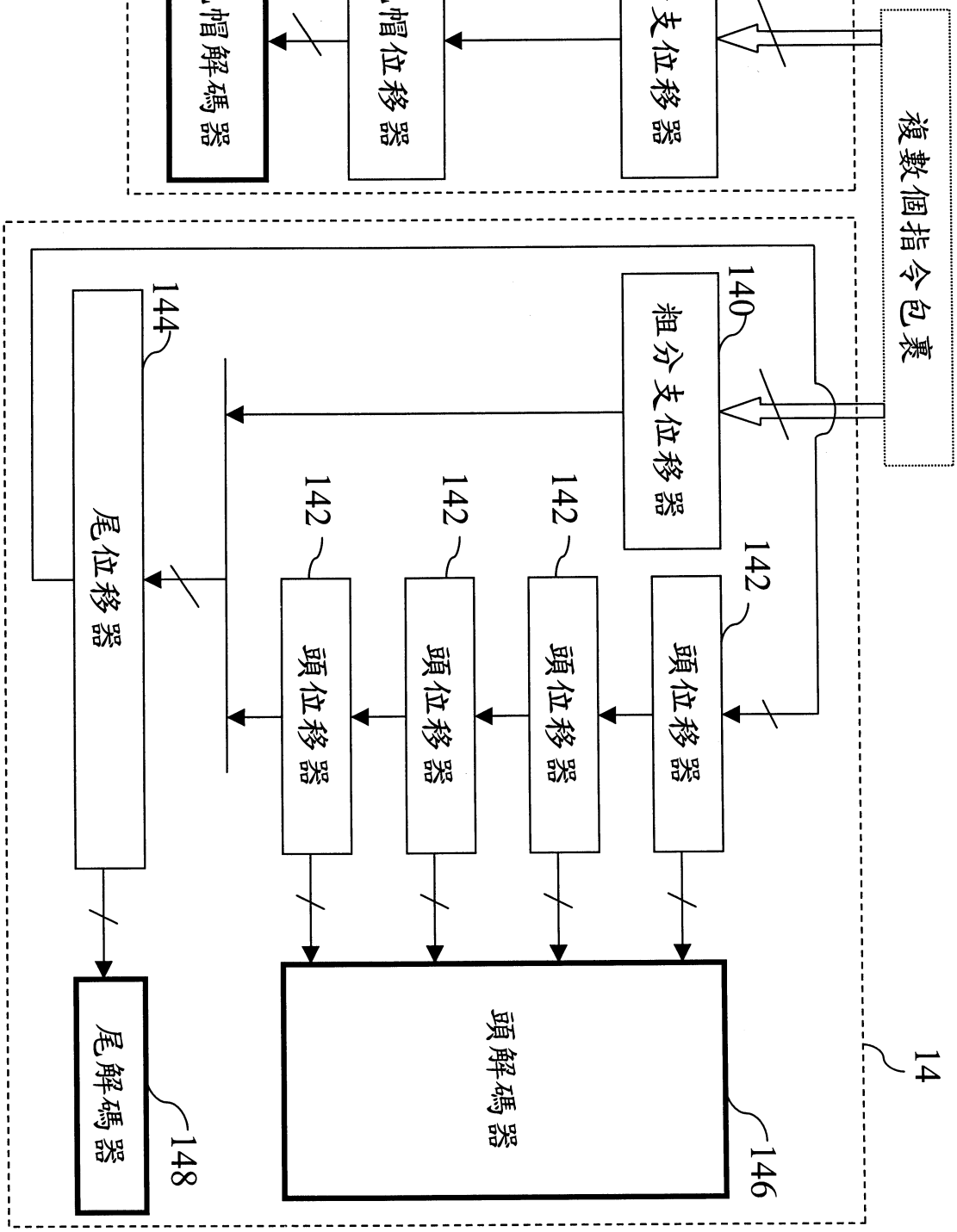
第四(b)圖


編碼開始
S10
查看有效指令且判斷指令之類別
S12
進行指令封包之包裝
S14
S16
在目前包裹中分派空間
配置新包裹
S22
S18
包裹是否用完?
終止包裹且標示為“11”
S20
是
否
包裹下一指令封包
S24
編碼結束
S26

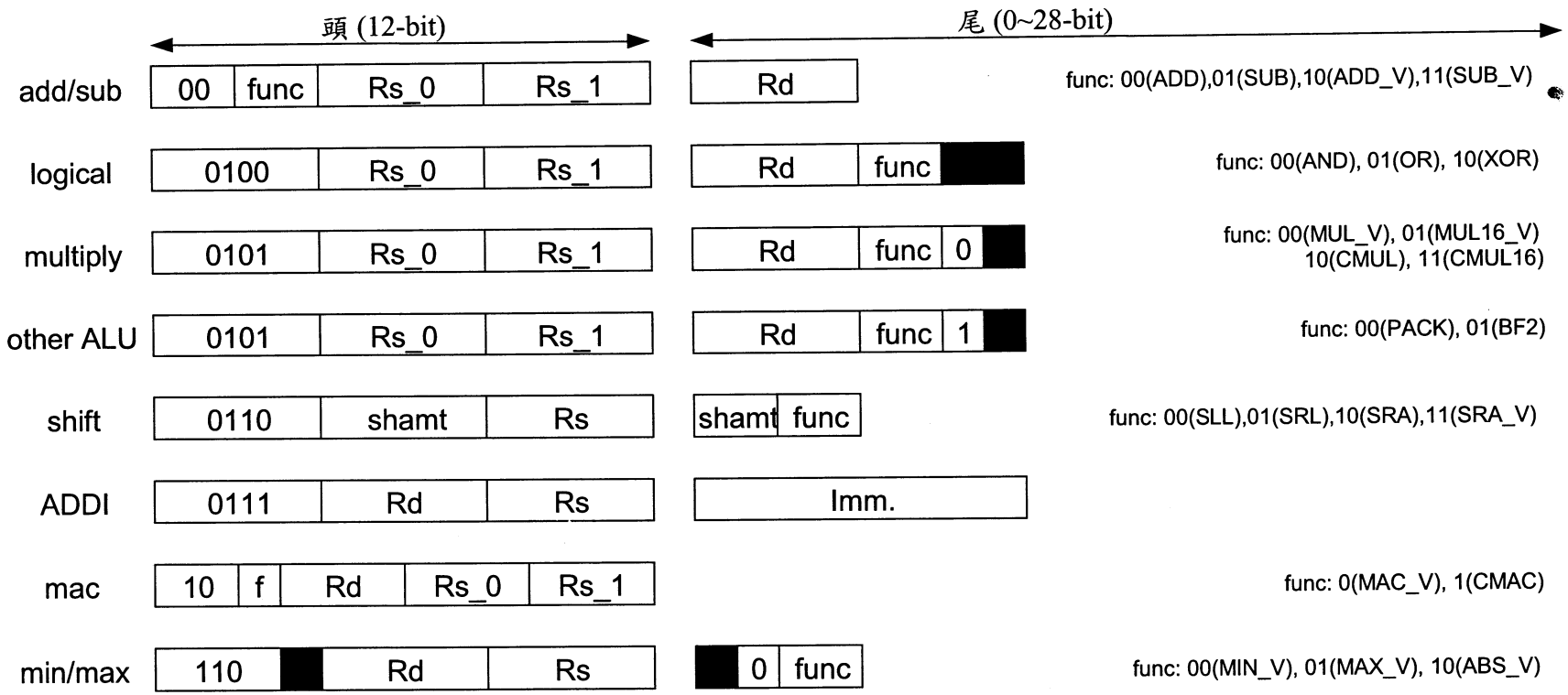
第五圖



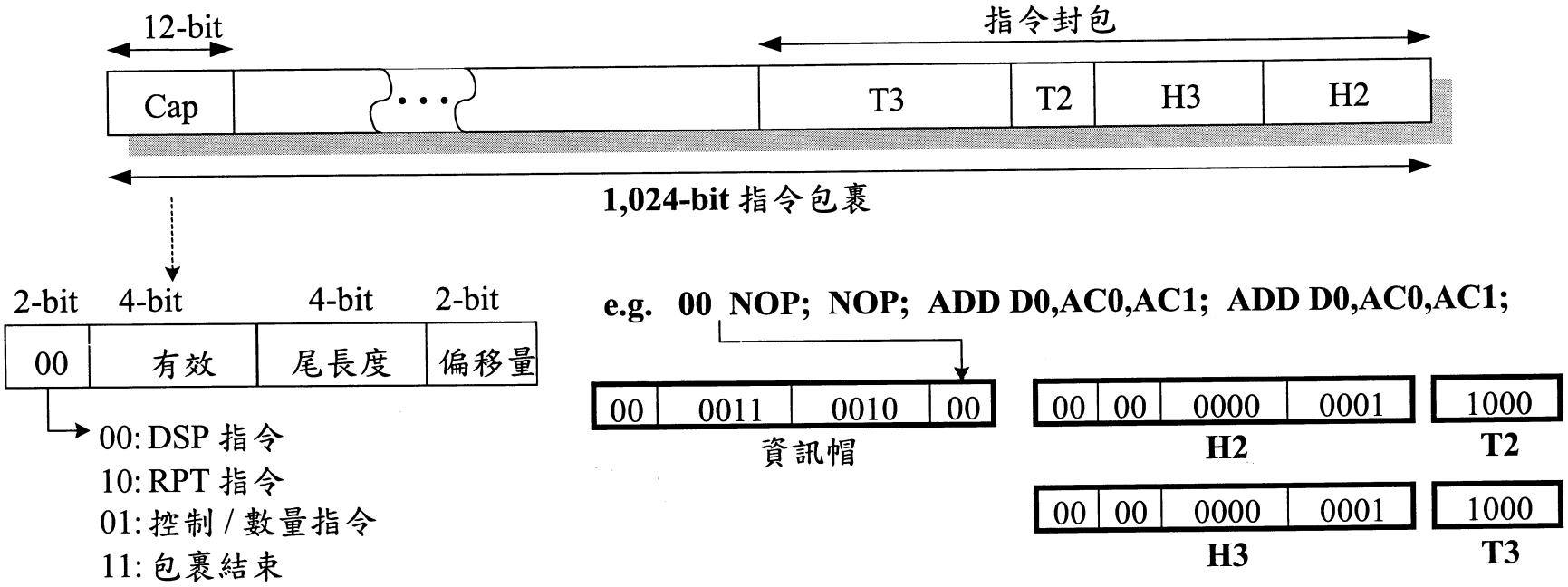
第六圖


複數個指令包裹
12
120
粗分支位移器
122
資訊帽位移器
124
資訊帽解碼器
14
140
粗分支位移器
142
頭位移器
142
頭位移器
142
頭位移器
144
尾位移器
頭解碼器
146
尾解碼器
148

第七圖

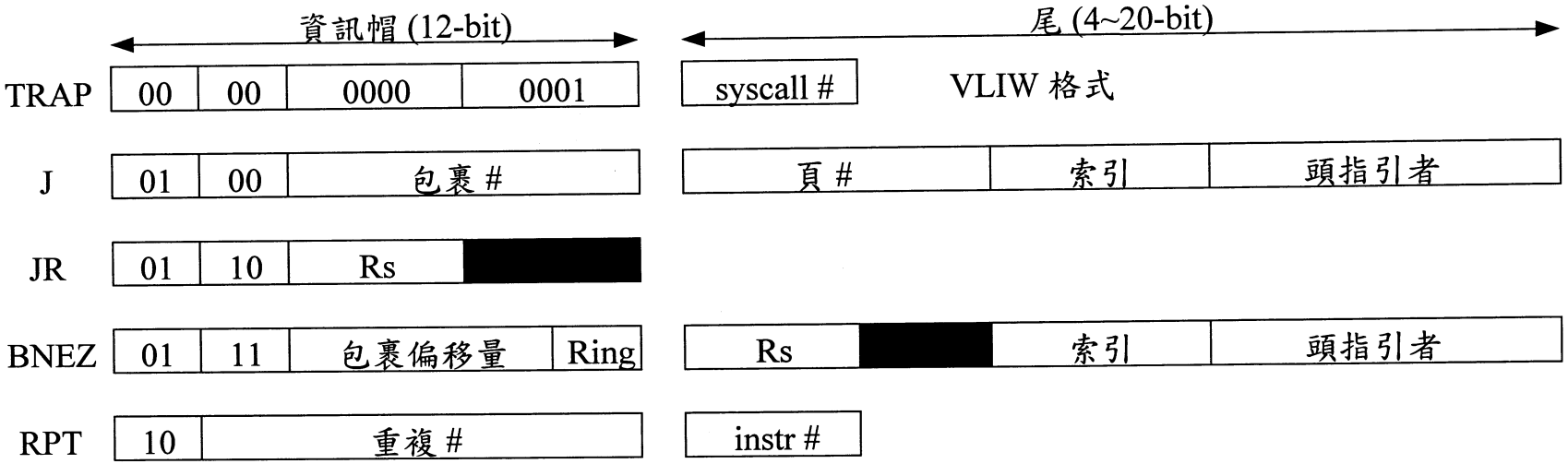

頭 (12-bit)
尾 (0~28-bit)
add/sub
00
func
Rs_0
Rs_1
Rd
func: 00(ADD),01(SUB),10(ADD_V),11(SUB_V)
logical
0100
Rs_0
Rs_1
Rd
func
func: 00(AND), 01(OR), 10(XOR)
multiply
0101
Rs_0
Rs_1
Rd
func
0
func: 00(MUL_V), 01(MUL16_V)
10(CMUL), 11(CMUL16)
other ALU
0101
Rs_0
Rs_1
Rd
func
1
func: 00(PACK), 01(BF2)
shift
0110
shamt
Rs
shamt
func
func: 00(SLL),01(SRL),10(SRA),11(SRA_V)
ADDI
0111
Rd
Rs
Imm.
mac
10
f
Rd
Rs_0
Rs_1
func: 0(MAC_V), 1(CMAC)
min/max
110
Rd
Rs
0
func
func: 00(MIN_V), 01(MAX_V), 10(ABS_V)

第八圖

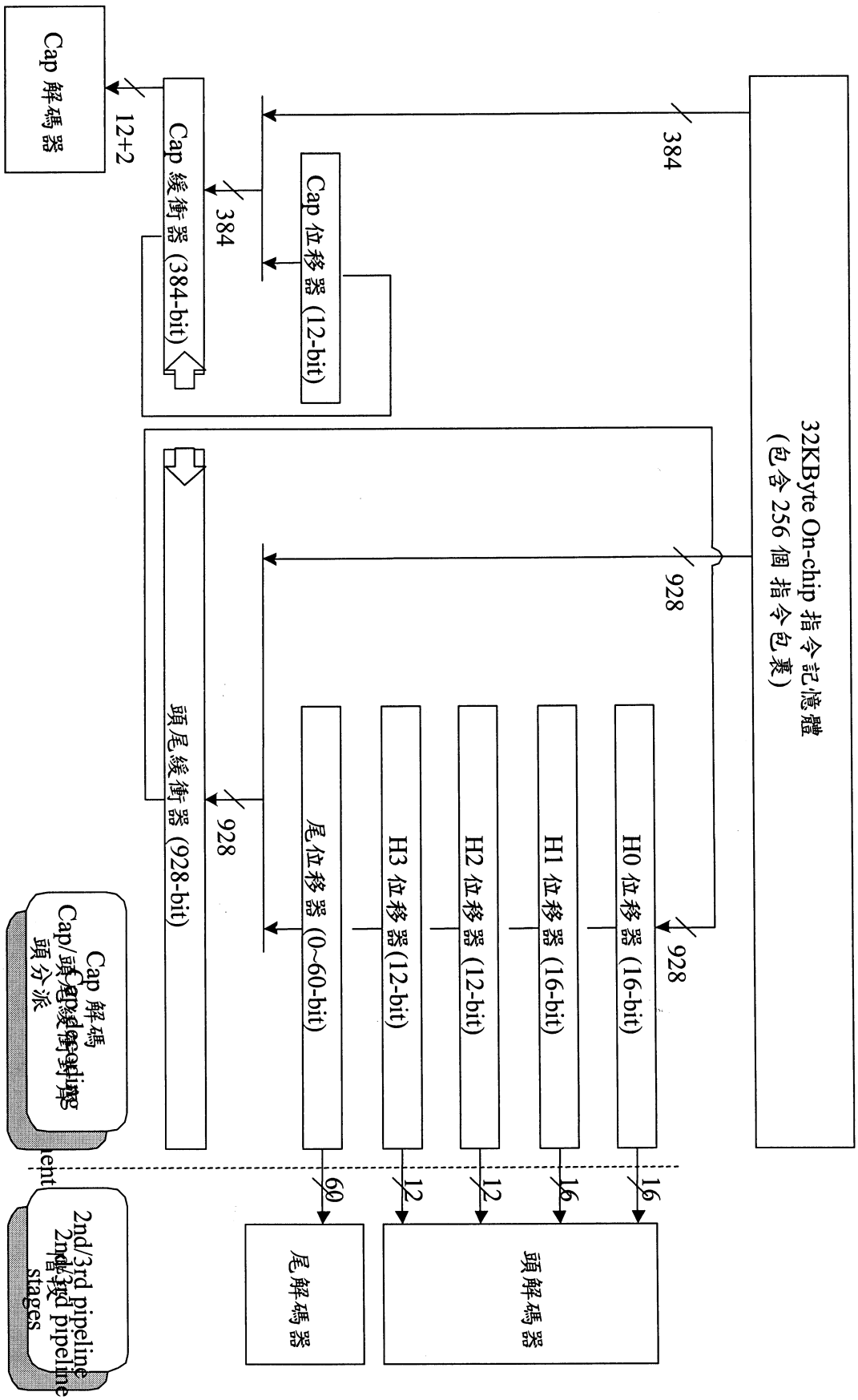

12-bit
指令封包
Cap
T3
T2
H3
H2
1,024-bit 指令包裹
2-bit
4-bit
4-bit
2-bit
00
有效
尾長度
偏移量
00: DSP 指令
10: RPT 指令
01: 控制 / 數量指令
11: 包裹結束
e.g. 00 NOP; NOP; ADD D0,AC0,AC1; ADD D0,AC0,AC1;
00
0011
0010
00
資訊帽
00
00
0000
0001
H2
1000
T2
00
00
0000
0001
H3
1000
T3

第九圖

I266238


資訊帽 (12-bit)
尾 (4~20-bit)
TRAP
00
00
0000
0001
syscall #
VLIW 格式
J
01
00
包裹 #
頁 #
索引
頭指引者
JR
01
10
Rs
BNEZ
01
11
包裹偏移量
Ring
Rs
索引
頭指引者
RPT
10
重複 #
instr #

第十圖


32KByte On-chip 指令記憶體
(包含 256 個 指令包裹)
384
928
Cap 位移器 (12-bit)
Cap 緩衝器 (384-bit)
384
Cap 解碼器
12+2
928
H0 位移器 (16-bit)
H1 位移器 (16-bit)
H2 位移器 (12-bit)
H3 位移器 (12-bit)
尾 位移器 (0~60-bit)
頭尾 緩衝器 (928-bit)
928
16
16
12
12
60
頭 解碼器
尾 解碼器
Cap 解碼
Cap/頭尾緩衝對準
頭尾分派
2nd/3rd pipeline
2nd/3rd pipeline
stages

第十一圖