(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 200945190 A1
(43)公開日： 中華民國 98 (2009) 年 11 月 01 日

(21)申請案號：098124440 (22)申請日： 中華民國 92 (2003) 年 09 月 24 日
(51)Int. Cl.： *G06F9/38 (2006.01)*
(30)優先權：2002/09/25 日本 2002-280077
(71)申請人：松下電器產業股份有限公司 (日本) PANASONIC CORPORATION (JP)
日本
(72)發明人：田中哲也 TANAKA, TETSUYA (JP) ；岡林葉月 OKABAYASHI, HAZUKI (JP) ；瓶子岳人 HEISHI, TAKETO (JP) ；小川一 OGAWA, HAJIME (JP) ；鈴木常之 SUZUKI, TSUNEYUKI (JP) ；清原督三 KIYOHARA, TOKUZO (JP) ；田中健 TANAKA, TAKESHI (JP) ；西田英志 NISHIDA, HIDESHI (JP) ；前田昌樹 MAEDA, MASAKI (JP)
(74)代理人：惲軼群；陳文郎
申請實體審查：有 申請專利範圍項數：3 項 圖式數：82 共 131 頁

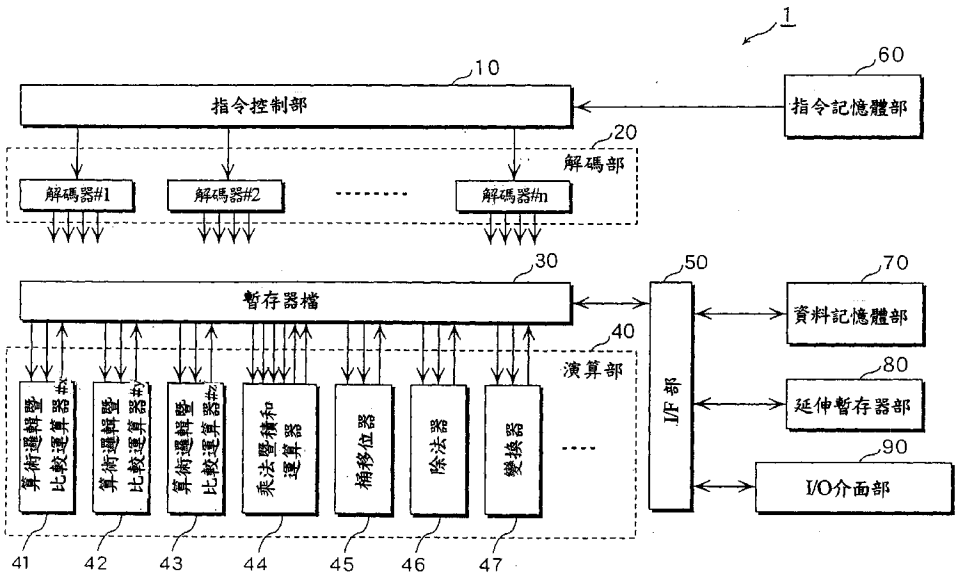
(54)名稱

處理器

PROCESSOR

(57)摘要

本發明之課題係於提供一種執行高機能 SIMD 運算之處理器。本發明之處理器係包含有解碼器 20 及運算部 40 等，解碼部 20 係解讀指令 vcchk 時，運算部 40 等係判定條件旗標暫存器(CFR)32 之向量條件旗標 VC0~VC3(110)全部是否為 0，在全部為 0 時，將條件旗標暫存器(CFR)(32)之條件旗標 C4 及 C5 各設定為 1 及 0；不是時，則將條件旗標 C4 及 C5 各設為 0 及 1。接著，條件旗標 C0~C3 中儲存向量條件旗標 VC0~VC3。



1：處理器
10：指令控制部
20：解碼部
30：暫存器檔
40：運算部
41：算術邏輯暨比較運算器#x
42：算術邏輯暨比較運算器#y
43：算術邏輯暨比較運算器#z
44：積和運算器
45：桶移位器
46：除法器

47：變換器

50：I/F 部

60：指令記憶體部

70：資料記憶體部

80：延伸暫存器部

90：I/O 介面部

(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 200945190 A1
(43)公開日： 中華民國 98 (2009) 年 11 月 01 日

(21)申請案號：098124440 (22)申請日： 中華民國 92 (2003) 年 09 月 24 日
(51)Int. Cl.： *G06F9/38 (2006.01)*
(30)優先權：2002/09/25 日本 2002-280077
(71)申請人：松下電器產業股份有限公司 (日本) PANASONIC CORPORATION (JP)
日本
(72)發明人：田中哲也 TANAKA, TETSUYA (JP) ；岡林葉月 OKABAYASHI, HAZUKI (JP) ；瓶子岳人 HEISHI, TAKETO (JP) ；小川一 OGAWA, HAJIME (JP) ；鈴木常之 SUZUKI, TSUNEYUKI (JP) ；清原督三 KIYOHARA, TOKUZO (JP) ；田中健 TANAKA, TAKESHI (JP) ；西田英志 NISHIDA, HIDESHI (JP) ；前田昌樹 MAEDA, MASAKI (JP)
(74)代理人：惲軼群；陳文郎
申請實體審查：有 申請專利範圍項數：3 項 圖式數：82 共 131 頁

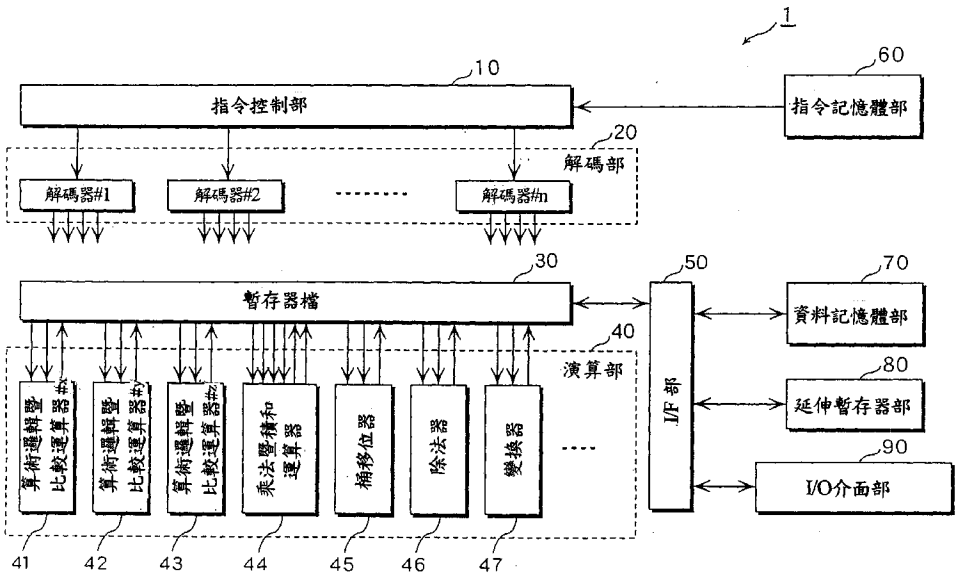
(54)名稱

處理器

PROCESSOR

(57)摘要

本發明之課題係於提供一種執行高機能 SIMD 運算之處理器。本發明之處理器係包含有解碼器 20 及運算部 40 等，解碼部 20 係解讀指令 vcchk 時，運算部 40 等係判定條件旗標暫存器(CFR)32 之向量條件旗標 VC0~VC3(110)全部是否為 0，在全部為 0 時，將條件旗標暫存器(CFR)(32)之條件旗標 C4 及 C5 各設定為 1 及 0；不是時，則將條件旗標 C4 及 C5 各設為 0 及 1。接著，條件旗標 C0~C3 中儲存向量條件旗標 VC0~VC3。



1：處理器
10：指令控制部
20：解碼部
30：暫存器檔
40：運算部
41：算術邏輯暨比較運算器#x
42：算術邏輯暨比較運算器#y
43：算術邏輯暨比較運算器#z
44：積和運算器
45：桶移位器
46：除法器

六、發明說明：

【發明所屬之技術領域】

本發明係有關於DSP及CPU之類的處理器，特別是有關於一種適於進行聲音或影像等信號處理之處理器。

【先前技術】

隨著多媒體技術的進展，以聲音及影像之信號處理等為代表，需要可高速執行媒體處理之處理器。可滿足該需求之習知處理器諸如有支援SIMD(單指令多資料；Single Instruction Multiple Data)型指令之處理器。例如美國英特爾公司之奔騰處理器Pentium(R) /同處理器III /同處理器4之MMX / SSE / SSE2等等。如果為英特爾公司的MMX，是以已儲存於64位元長度之MMX暫存器且以最大8個整數為對象，以1個指令執行同一運算。

惟，按上述習知處理器，尚有未能充分滿足對於媒體處理之各式各樣需求的問題存在。

例如，習知處理器雖能以1個指令執行對於多數資料之運算或以1個指令比較多數資料，卻不能以1個指令評估其等多數比較結果。舉例來說，習知之處理器係執行如此處理，即，以1個指令，對儲存於32位元長度之兩個暫存器之資料群以位元組單位進行比較，且對其結果設定(set)成4個旗標者；但卻不能用1個指令分辨其等4個旗標值全部是否為零者。因此需要一用以讀出4個旗標且分辨全部是否為零之多數指令。這樣的話，例如在每次以4個像素值為單位再與其他像素值進行比較時，勢必需要可評估該比較結果用

之多數指令，便導致指令數量的增加，且使影像處理速度降低。

【發明內容】

在此，本發明係有鑑於如此狀況而所構建者，目的係於提供一種可執行高機能SIMD運算之處理器，及可高速執行數位信號處理且適於多媒體用途之處理器。

為達成上述目的，本發明之第1處理器係一種用以執行SIMD型指令之處理器，其特徵在於包含有：一解讀裝置，係用以解讀指令者；及，一執行裝置，係用以根據前述解讀裝置所進行之解讀結果以執行指令者；前述執行裝置係於藉前述解讀裝置解讀一表示用以判定以多數資料為對象之SIMD型比較指令所得到之比較結果之指令時，針對前述多數資料全部判定是否得到同一比較結果，且產生其結果者。

又，本發明之第2處理器係一種與外部記憶體相連接之處理器，其特徵在於包含有：一暫存器，係用以記憶資料者；一解讀裝置，係用以解讀指令者；及，一執行裝置，係用以根據前述解讀裝置所進行之解讀結果以執行指令者；前述執行裝置係於藉前述解讀裝置解讀一表示用以將暫存器值儲存於外部記憶體之指令時，將由儲存於前述暫存器之4以上的位元組所構成之字資料中，高半字之最低1位元組與低半字之最低1位元組儲存於前述外部記憶體。

又，本發明之第3處理器係一種用以解讀指令且予以執行之處理器，其特徵在於包含有：暫存器，係用以記憶資

料者；一解讀裝置，係用以解讀指令者；及，一執行裝置，係用以根據前述解讀裝置所進行之解讀結果以執行指令者；前述執行裝置係於藉前述解讀裝置解讀一表示用以將前述暫存器之高位數儲存資料之指令時，將前述暫存器中，在不變更除前述高位數外之記憶區之值的狀態下，將前述資料只儲存於前述高位數。

又，本發明之第4處理器係一種用以執行SIMD型指令之處理器，其特徵在於包含有：旗標記憶裝置，係用以儲存旗標者；一解讀裝置，係用以解讀指令者；及，一執行裝置，係用以根據前述解讀裝置所進行之解讀結果以執行指令者；前述執行裝置係於藉前述解讀裝置解讀一以第1暫存器及第2暫存器作為運算元之SIMD型運算指令時，在儲存於前述旗標記憶裝置之第1旗標顯示第1狀態時，只令前述第1暫存器為運算元，執行前述SIMD型運算，而於前述第1旗標顯示第2狀態時，則令前述第1暫存器與第2暫存器為運算元，執行前述SIMD型運算。

又，本發明之第5處理器係一種用以執行SIMD型指令之處理器，其特徵在於包含有：一解讀裝置，係用以解讀指令者；及，一執行裝置，係用以根據前述解讀裝置所進行之解讀結果以執行指令者；前述執行裝置係於藉前述解讀裝置解讀一表示根據多數資料各自之正負號產生各值之SIMD型指令時，產生一資料，以顯示前述多數資料個別為正、零及負中之哪一值者。

又，本發明之第6處理器係一種用以執行SIMD型指令

之處理器，其特徵在於包含有：一參數指定裝置，係用以指定第1參數及第2參數者；一解讀裝置，係用以解讀指令者；及，一執行裝置，係用以根據前述解讀裝置所進行之解讀結果以執行指令者；前述執行裝置係於藉前述解讀裝置解讀表示一用以操作第1資料之指令時，根據前述第1參數對前述第1資料做位元移位，且由所得到之移位資料中，輸出藉前述第2參數界定之單位長度位置之多數單位長度資料者。

又，本發明之第7處理器係一以解讀指令且予以執行之處理器，其特徵在於包含有：一解讀裝置，係用以解讀指令者；及，一執行裝置，係用以根據前述解讀裝置所進行之解讀結果以執行指令者；前述執行裝置係於藉前述解讀裝置解讀一將第1及第2資料作為運算元之加法指令時，在前述第1資料為零或正時，產生一將前述第1資料、前述第2資料與1相加之結果，而在前述第1資料為負時，則產生一將前述第1資料與前述第2資料相加之結果。

又，本發明之第8處理器係一種用以解讀指令且予以執行之處理器，其特徵在於包含有：多數暫存器；一解讀裝置，係用以解讀指令者；及，一執行裝置，係用以根據前述解讀裝置所進行之解讀結果以執行指令者；前述執行裝置係於藉前述解讀裝置解讀一表示用以轉送第1及第2暫存器之值的指令時，將前述第1暫存器之值儲存於第3暫存器，並將前述第2暫存器之值儲存於設在與前述第3暫存器相連續之位置的第4暫存器。

又，本發明之第9處理器係一種用以解讀指令且予以執行之處理器，其特徵在於包含有：一旗標記憶裝置，係用以記憶條件執行指令之述詞所使用之多數旗標者；一解讀裝置，係用以解讀指令者；及，一執行裝置，係用以根據前述解讀裝置所進行之解讀結果以執行指令者；前述執行裝置係於藉前述解讀裝置解讀一用以將前述旗標作為運算元之迴路之分支指令時，分支至迴路之前頭並進行前述旗標之設定者。又，是一種用以解讀指令且予以執行之處理器，其特徵在於包含有：一分支暫存器，係用以儲存分支處位址者；一旗標記憶裝置，係用以記憶條件執行指令之述詞所用之多數旗標者；一解讀裝置，係用以解讀指令者；及，一執行裝置，係用以根據前述解讀裝置所進行之解讀結果以執行指令者；前述執行裝置係於藉解讀裝置解讀一使前述旗標作為運算元且將之儲存於分支暫存器之儲存指令時，將迴路之前頭位址儲存於前述分支暫存器，並進行前述旗標設定。

又，本發明之第10處理器係一種用以執行SIMD型指令之處理器，其特徵在於包含有：一解讀裝置，係用以解讀指令者；及，一執行裝置，係用以根據前述解讀裝置所進行之解讀結果以執行指令者；前述執行裝置係於藉前述解讀裝置解讀一表示令多數資料對為對象後進行差分絕對值和之SIMD型指令時，產生將前述多數資料對各自之差分絕對值相加的值。

又，本發明之第11處理器係一種用以解讀指令且予以

執行之處理器，其特徵在於包含有：一解讀裝置，係用以解讀指令者；及，一執行裝置，係用以根據前述解讀裝置所進行之解讀結果以執行指令者；前述執行裝置係於藉前述解讀裝置解讀一以第1及第2資料為運算元之飽和指令時，在前述第1資料較藉前述第2資料界定之飽和值還大時，產生前述飽和值，而在前述第1資料小於前述飽和值時，則產生前述第1資料。

又，本發明之第12處理器係一種用以解讀指令且予以執行之處理器，其特徵在於包含有：多數暫存器，係n單位長度者；一解讀裝置，係用以解讀指令者；及，一執行裝置，係用以根據前述解讀裝置所進行之解讀結果以執行指令者；前述執行裝置係於藉前述解讀裝置解讀一表示以第1至第3暫存器及1個常數為運算元後以單位長度單位進行選擇之指令時，將自儲存於前述第1及第2暫存器之2n個單位長度資料中藉前述參數選擇之n個單位長度資料儲存於前述第3暫存器。

又，本發明之第13處理器係一種用以執行SIMD型指令之處理器，其特徵在於包含有：一解讀裝置，係用以解讀指令者；及，一執行裝置，係用以根據前述解讀裝置所進行之解讀結果以執行指令者；前述執行裝置係於藉前述解讀裝置解讀SIMD型指令時，藉進行SIMD型運算，產生多數運算結果，且至少針對前述多數運算結果之1個結果進行位元延伸。

又，本發明之第14處理器係一種用以執行SIMD型指令

之處理器，其特徵在於包含有：一旗標記憶裝置，係用以記憶旗標者；一解讀裝置，係用以解讀指令者；及，一執行裝置，係用以根據前述解讀裝置所進行之解讀結果以執行指令者；前述執行裝置係於藉前述解讀裝置解讀一表示對於多數資料對執行SIMD型運算之指令時，對於前述多數資料對個別進行藉前述旗標記憶裝置所記憶之旗標界定之SIMD型運算者。

此外，本發明不只可實現一種可執行具有如此特徵之指令之處理器，還可實現成為一種對於多數資料等運算之運算處理方法，或實現一種含有該特徵之指令的程式。又，如此程式當然亦可透過CD-ROM等記錄媒體或網際網路等傳播媒體予以流通者。

【實施方式】

針對本發明之處理器的架構進行說明。本處理器之指令的並列性高於通常的個人電腦，為一以 AV 媒體系信號處理技術領域為目標而所開發之通用處理器。藉於行動電話、移動式 AV 機器、數位電視、DVD 等用作為共通核心，可以提高軟體再利用性。又，本處理器係以高機能及高成本性能可實現很多媒體處理，進而提供一種以提昇開發效率為目的之高階語言開發環境。

第 1 圖係本處理器之概略方塊圖。本處理器 1 係包含有：指令控制部 10、解碼部 20、暫存器檔 30、運算部 40、I/F 部 50、指令記憶體部 60、資料記憶體部 70、延伸暫存器部 80 及 I/O 介面部 90。運算部 40 係由用以執行 SIMD

型指令之運算的算術邏輯暨比較運算器 41~43、乘法暨積和運算器 44、桶移位器 45、除法器 46 及變換器 47 所構成。乘法暨積和運算器 44 係於不降低位元精確度之狀態下最長可累積到 65 位元。又，乘法暨積和運算器 44 係與算術邏輯暨比較運算器 41~43 同樣，可做 SIMD 型指令之執行者。進而，該處理器 1 係可執行算術邏輯暨比較運算指令最大達三並列者。

第 2 圖係顯示算術邏輯暨比較運算器 41~43 之概略圖。算術邏輯暨比較運算器 41~43 各由 ALU 部 41a、飽和處理部 41b 及旗標部 41c 所構成。ALU 部 41a 係由算術運算器、邏輯運算器、比較器、TST 器所構成。所對應之運算資料之位元寬度則為 8 位元(使用運算器四並列)、16 位元(使用運算器二並列)、32 位元(以全部運算器做 32 位元資料處理)。進而對於算術運算結果，藉旗標部 41c 等裝置進行溢位檢測及條件旗標產生。各運算器、比較器、TST 器之結果係進行算術右移位、以飽和處理部 41b 所進行之飽和、最大及最小值檢測、絕對值產生處理。

第 3 圖係顯示桶移位器 45 之結構的方塊圖。桶移位器 45 係由選擇器 45a、45b、高位元桶移位器 45c、低位元桶移位器 45d 及飽和處理部 45e 所構成，用以執行資料之算術移位(2 之補數體系的移位)或邏輯移位(無符號之移位)。通常是將 32 位元或 64 位元之資料作為輸入或輸出。對於被儲存於暫存器 30a、30b 之被移位資料，以其他暫存器或立即值指定移位量。資料係予以進行左 63 位元至右 63 位

元之算術或邏輯移位，並以輸入位元長度加以輸出。

又，桶移位器 45 係可對於 SIMD 型指令而將 8、16、32、64 位元資料移位。例如可將 8 位元資料移位以四並列處理。

算術移位係指 2 之補數體系移位，係為了加法或減法時之小數點位置的配合、2 乘冪之乘法(2、2 的 2 次方、2 的(－1)次方、2 的(－2)次方倍、...)等所進行的。

第 4 圖係顯示變換器 47 之結構的方塊圖。變換器 47 係由飽和方塊(SAT)47a、BSEQ 方塊 47b、MSKGEN 方塊 47c、VSUMB 方塊 47d、BCNT 方塊 47e 及 IL 方塊 47f 所構成。

飽和方塊(SAT)47a 係用以進行對於輸入資料之飽和處理。藉具有兩個對 32 位元資料進行飽和處理之方塊，以支援二並列之 SIMD 型指令。

BSEQ 方塊 47b 係用以計算由 MSB 連續之 0 或 1。

MSKGEN 方塊 47c 係用以將業經指定之位元區間為 1，除此以外者為 0 後予以輸出。

VSUMB 方塊 47d 係用以將輸入資料區分成所指定之位元寬度，接著輸出其總和。

BCNT 方塊 47e 係用以計算以輸入資料成為 1 之位元的數量。

IL 方塊 47f 係用以將輸入資料區分成所指定之位元寬度，接著輸出各將資料方塊對調後之數值。

第 5 圖係顯示除法器 46 之結構的方塊圖。除法器 46

係用以令被除數為 64 位元、除數為 32 位元，使商及剩餘各以 32 位元輸出。而求出商及剩餘為止需要 34 循環。可處理附帶正負號、無符號、兩者資料者。惟，在被除數與除數中對於符號(正負號)有無的設定是共通的。其他還具有可輸出溢位旗標、0 除法旗標之機能。

第 6 圖係顯示乘法暨積和運算器 44 之結構的方塊圖。乘法暨積和運算器 44 係包含有 2 個 32 位元乘法器(MUL)44a、44b、3 個 64 位元加法器(Adder)44c~44e、選擇器 44f 及飽和處理部(Saturation)44g，以進行下列之乘法、積和運算。

。32x32 位元之帶正負號(signed)乘法、積和、積差運算

。32x32 位元之無符號(unsigned)乘法

。16x16 位元之二並列帶正負號(signed)乘法、積和、積差運算

。32x16 位元之二並列帶正負號(signed)乘法、積和、積差運算

對於整數、固定小數點格式(h1、h2、w1、w2)的資料進行上述運算。又，對於其等運算進行捨進、飽和處理。

第 7 圖係顯示指令控制部 10 之結構的方塊圖。指令控制部 10 係包含有：指令高速緩衝記憶體 10a、位址管理部 10b、指令緩衝器 10c~10e、跳越緩衝器 10f 及旋轉部(rotation)10g，用以進行通常時及分支時的指令供給。藉持有三個 128 位元之指令緩衝器(指令緩衝器 10c~10e)，可對應於最大並列執行數。指令控制部 10 係於分支處理時，在

執行分支前，先將分支處之指令儲存於跳越緩衝器 10f，並事先將分支處位址儲存於後述之 TAR 暫存器(settar 指令)。因此，在分支時，指令控制部 10 使用業已儲存於 TAR 暫存器之分支處位址以及業已儲存於跳越緩衝器 10f 之分支處指令，以便進行分支。

此外，本處理器 1 係具有 VLIW 架構之處理器。在此，VLIW 架構係指用以於一個指令語中儲存有多數指令(載入、儲存、運算、分支等)，並於同時執行全部的指令者。程式設計師係使可並列執行之指令作為一發出群而予以敘述，可將該發出群進行並列處理。在本說明書中，將發出群之區隔以";;"來表示。以下列出標記型態。

(例 1)

```
mov r1, 0x23;;
```

該指令敘述意指：只執行指令 mov 者。

(例 2)

```
mov r1, 0x38
add r0, r1, r2
sub r3, r1, r2;;
```

上述指令敘述意指：以三並列執行指令 mov、add、sub 者。

指令控制部 10 係用以辨識發出群後再送往解碼部 20。解碼部 20 則解析發出群之指令，以控制必要的資源。

其次，針對本處理器 1 所具備之暫存器進行說明。

本處理器 1 之暫存器集係如同下列表 1 所示者。

【表 1】

暫存器名稱	位元寬度	條數	用途
R0~R31	32 位元	32 條	通用暫存器。用於資料記憶體之指標、運算指令中之資料儲存等。
TAR	32 位元	1 條	分支用暫存器。用於分支時之分支位址儲存。
LR	32 位元	1 條	鏈接用暫存器。
SVR	16 位元	2 條	保留用暫存器。保留條件旗標(CFR)與各種模式。
M0~M1 (MH0:ML0~MH1:ML1)	64 位元	2 條	運算用暫存器。用於運算指令中之資料儲存。

又，本處理器 1 之旗標集(以後述之條件旗標暫存器等予以管理之旗標)係如下列表 2 所示者。

【表 2】

暫存器名稱	位元寬度	條數	用途
C0~C7	1 位元	8 條	條件旗標。表示條件成立及條件不成立。
VC0~VC3	1 位元	4 條	媒體處理延伸指令用條件旗標。表示條件成立及條件不成立。
OVS	1 位元	1 條	溢位旗標。檢測運算時之溢位。
CAS	1 位元	1 條	進位旗標。檢測運算時之進位。
BPO	5 位元	1 條	位元位置指定。在遮蔽處理指令時指定成為處理對象之位元位置。
ALN	2 位元	1 條	位元組調正指定。
FXP	1 位元	1 條	固定小數點運算模式
UDR	32 位元	1 條	未定義暫存器

第 8 圖係顯示通用暫存器(R0~R31)30a 之結構圖。通用暫存器(R0~R31)30a 係用以構建成為執行對象之任務的上下文的一部分，儲存資料或位址之 32 位元的暫存器群。此外，通用暫存器 R30 及 R31 係各作為總體指標、堆疊指標，由硬碟予以使用。

第 9 圖係顯示鏈接暫存器(LR)30c 之結構圖。此外，關連著該鏈接暫存器(LR)30c，本處理器 1 亦具有一未圖示之保留暫存器(SVR)。鏈接暫存器(LR)30c 係一用以儲存函數呼叫時之轉回位址之 32 位元暫存器。此外，保留暫存器(SVR)係一用以保留函數呼叫時之條件旗標暫存器的條件旗標(CFR.CF)之 16 位元暫存器。鏈接暫存器(LR)30c 係與後述之分支暫存器(TAR)同樣，亦使用於迴路高速化。低 1 位元係始終讀出 0，而在寫入時則有寫入 0 之必要。

例如執行 call(brl,jmpl)指令後，本處理器 1 係將轉回位址保留於鏈接暫存器(LR)30c，且將條件旗標(CFR.CF)保留於保留暫存器(SVR)。又，執行 jmp 指令後，由鏈接暫存器(LR)30c 取出轉回位址(分支處位址)，回復程式計數器(PC)。進而，執行 ret(jmpr)指令後，由鏈接暫存器(LR)30c 取出分支處位址(轉回位址)，再儲存(回復)於程式計數器(PC)。進而，由保留暫存器(SVR)取出條件旗標，且將其儲存(回復)於條件旗標暫存器(CFR)32 之條件旗標領域 CFR.CF。

第 10 圖係顯示分支暫存器(TAR)30d 之結構圖。分支暫存器(TAR)30d 係一用以儲存分支標的位址之 32 位元暫

存器。其主要是用於迴路高速化。低 1 位元係始終讀出 0，而在寫入時則有寫入 0 之必要。

例如執行了 jmp,jloop 指令時，本處理器 1 係由分支暫存器(TAR)30d 取出分支處位址，再將之儲存於程式計數器(PC)。業已儲存於分支暫存器(TAR)30d 之位址指令係儲存於分支用指令緩衝器時，則分支損失(penalty)變成 0。先將迴路之前頭位址儲存於分支暫存器(TAR)30d，即可使迴路高速化。

第 11 圖係顯示程式狀態暫存器(PSR)31 之結構圖。程式狀態暫存器(PSR)31 係一 32 位元暫存器，構建成一成為執行對象之任務之上下文的一部分，用以儲存如下所示之處理器狀態資訊者。

位元 SWE：指 VMP(虛擬多處理器；Virtual Multi-Processor)之 LP(邏輯處理器；Logical Processor)切換賦能(enable)。「0」係表示不允許 LP 切換，「1」則表示允許 LP 切換。

位元 FXP：指固定小數點模式。「0」係表示模式 0、「1」則表示模式 1。

位元 IH：指中斷處理旗標，表示處於可遮蔽中斷處理中者。「1」係表示中斷處理中，「0」則表示不是中斷處理中。一發生中斷時則予以自動設定。以 rti 指令，使用於由中斷返回時，分辨出處於另一中斷處理中或處於程式處理中者。

位元 EH：指表示處於錯誤或 NMI 處理中之旗標。「0」係表示不是錯誤/NMI 中斷處理中者，「1」則表示處於錯誤

/NMI 中斷處理中者。在 EH=1 時，產生非同步錯誤或 NMI 時，即予以遮蔽。又，在 VMP 賦能時，則遮蔽 VMP 之板極切換。

位元 PL[1：0]：指特權位準。「00」係特權位準 0，即指處理器抽象(abstraction)位準，「01」係指特權位準 1(不能設定)，「10」係特權位準 2，即指系統程式位準，「11」係特權位準 3，即指用戶程式位準。

位元 LPIE3：係指 LP 固有中斷 3 賦能。「1」係表示允許中斷，「0」則表示不允許中斷。

位元 LPIE2：係指 LP 固有中斷 2 賦能。「1」係表示允許中斷，「0」則表示不允許中斷。

位元 LPIE1：係指 LP 固有中斷 1 賦能。「1」係表示允許中斷，「0」則表示不允許中斷。

位元 LPIE0：係指 LP 固有中斷 0 賦能。「1」係表示允許中斷，「0」則表示不允許中斷。

位元 AEE：係指遺漏調整例外賦能。「1」係表示允許遺漏調整例外，「0」則表示不允許遺漏調整例外。

位元 IE：係指位準中斷賦能。「1」係顯示允許位準中斷，「0」則指不允許位準中斷。

位元 IM[7：0]：係指中斷遮蔽。定義位準 0 至 7，可以每一個位準予以遮蔽。位準 0 則成為最高之位準。只有藉 IM 尚未遮蔽之中斷要求中具最高位準之中斷要求可被處理器 1 受理。一受理中斷要求，低於業經受理之位準以下之位準便自動以硬體進行遮蔽。IM[0]係位準 0 之遮蔽，IM[1]

係位準 1 之遮蔽，IM[2]係位準 2 之遮蔽，IM[3]係位準 3 之遮蔽，IM[4]係位準 4 之遮蔽，IM[5]係位準 5 之遮蔽，IM[6]係位準 6 之遮蔽，IM[7]係位準 7 之遮蔽。

reserved：表示預約位元。始終將 0 讀出。在寫入時，則有寫入 0 之必要。

第 12 圖係顯示條件旗標暫存器(CFR)32 之結構圖。條件旗標暫存器(CFR)32 係構建成部分成為執行對象之任務之上下文之 32 位元暫存器，由條件旗標(condition flag)、執行旗標(運算旗標)、向量條件旗標(vector condition flag)、運算指令用位元位置指定欄、及 SIMD 資料調正資訊欄所構成。

位元 ALN[1:0]：係表示調正(align)模式。設定 valnvc 指令之調整模式。

位元 BPO[4:0]：係表示位元位置。於位元位置指定時必用之指令使用。

位元 VC0~VC3：係向量條件旗標。由靠 LSB 側之位元組或半字按序對應於 VC0，MSB 側則對應於 VC3。

位元 OVS：係指溢位旗標(彙總(summary))。於發生飽和或溢位檢測予以設定。未能測出時，則保持指令執行前之值。清除(clear)係須以軟體進行。

位元 CAS：係指進位(carry)旗標(彙總(summary))。即於以 addc 指令發生進位或以 subc 指令發生借位(borrow)時予以設定。以 addc 指令沒有發生進位或以 subc 指令沒有發生借位時，便保持指令執行前之值。清除則有用軟體進行之

必要。

位元 C0~C7：係指條件旗標。旗標 C7 係使值始終為 1。往旗標 C7 之 FALSE 條件之反映(寫入 0)係予以無視。

reserved：係指預約位元。始終讀出 0。寫入時便有將 0 寫入之必要。

第 13 圖係顯示累加器(M0,M1)30b 之結構圖。該累加器(M0, M1)30b 係構建成部分成為執行對象之任務之上下文，由第 13(a)圖所示之 32 位元暫存器 MH0-MH1(乘除法與積和用暫存器(高 32 位元))及第 13(b)圖所示之 32 位元暫存器 ML0-ML1 乘除法與積和用暫存器(低 32 位元)所構成。

暫存器 MH0-MH1 係使用於以乘法指令儲存於結果之高 32 位元者。在積和指令時則作為累加器之高 32 位元使用。又，處理位元流時，可與通用暫存器組合使用。暫存器 ML0-ML1，於乘法指令時用於儲存結果的低 32 位元者。積和指令時則作為累加器之低 32 位元使用。

第 14 圖係顯示程式計數器(PC)33 之結構圖。該程式計數器(PC)33 係一 32 位元之計數器，構建成部分成為執行對象之任務的上下文，以保持執行中之指令位址。

第 15 圖係顯示 PC 保留用暫存器(IPC)34 之結構圖。該 PC 保留用暫存器(IPC)34 係一構建成部分成為執行對象之任務的上下文之 32 位元暫存器。

第 16 圖係一顯示 PSR 保留用暫存器(IPSR)35 之結構圖。該 PSR 保留用暫存器(IPSR)35 係一 32 位元暫存器，構建成部分成為執行對象之任務之上下文，以保留程式狀

態暫存器(PSR)31 者，對應於程式狀態暫存器(PSR)31 之預約位元之部分始終讀出 0，在寫入時則有將 0 寫入之必要。

其次，針對本處理器 1 之記憶體空間進行說明。在本處理器 1 中，將 4GB 之線性記憶體空間做 32 分割，在 128MB 單位之空間劃分成指令 SRAM(靜態 RAM；Static RAM)及資料 SRAM。以該 128MB 之空間當做 1 方塊，設定欲朝 SAR(SRAM Area Register；SRAM 區域暫存器)存取之方塊。所存取之位址是以 SAR 所設定之空間時，直接對指令 SRAM/資料 SRAM 進行存取，而不是以 SAR 設定之空間時，則對於匯流排控制器(BCU)提出存取要求。BCU 係與晶載記憶體(OCM；on-chip memory)、外部記憶體、外部元件、I/O 埠等相連接，可對於其等元件進行讀寫者。

第 17 圖係顯示本處理器 1 管線(pipe line)動作之時序圖。如本圖所示，本處理器 1 係主要是以指令提取、指令劃分(調度；dispatch)、解碼、執行、寫入等 5 段管線所構造成者。

第 18 圖係用以顯示本處理器 1 中執行指令時各管線動作之時序圖。指令提取級時，存取以程式計數器(PC)33 指定之位址之指令記憶體，將指令轉送至指令緩衝器 10c~10e 等。指令劃分級時，進行諸如對於分支系指令之分支處位址資訊之輸出、輸入暫存器控制信號之輸出、及可變長度指令之劃分，且將指令轉送至指令暫存器(IR)。解碼級時，將 IR 輸入於解碼部 20，且將運算器控制信號、記憶體存取信號予以輸出。執行級時，則執行運算，且將運算結果輸

出於資料記憶體或通用暫存器(R0~R31)30a。在寫入級時，則將轉送資料、運算結果儲存於通用暫存器。

本處理器 1 係藉 VLIW 架構俾可以最高達 3 並列進行上述處理。因此，針對第 18 圖所示之動作，本處理器 1 係以第 19 圖所示之時序並列執行。

其次，針對如上構成之本處理器 1 之指令集進行說明。

以下之表 3 至表 5 係本處理器 1 所執行之指令各按種類而加以分類而成之表。

【表 3】

種　類	運算器	指令執行碼
記憶體轉送指令(載入)	M	ld, ldh, ldhu, ldb, ldbu, ldp, ldhp, ldbp, ldbh, ldbuh, ldbhp, ldbuhp
記憶體轉送指令(儲存)	M	st, sth, stb, stp, sthp, stbp, stbh, stbhp
記憶體轉送指令(其他)	M	dpref, ldstb
外部暫存器轉送指令	M	rd, rde, wt, wte
分支指令	B	br, brl, call, jmp, jmpl, jmpr, ret, jmpf, jloop, setbb, setlr, settar
軟體中斷指令	B	rti, pi0, pi0l, pi1, pi1l, pi2, pi2l, pi3, pi3l, pi4, pi4l, pi5, pi5l, pi6, pi6l, pi7, pi7l, sc0, sc1, sc2, sc3, sc4, sc5, sc6, sc7
VMP/中斷控制指令	B	intd, inte, vmpsleep, vmpsus, vmpswd, vmpswe, vmpwait
算術運算指令	A	abs, absvh, absvw, add, addarvw, addc, addmsk, adds, addsr, addu, addvh, addvw, neg, negvh, negvw, rsub, s1add, s2add, sub, subc, submsk, subs, subvh, subvw, max, min
邏輯運算指令	A	and, andn, or, sethi, xor, not
比較指令	A	cmpCC, cmpCCa, cmpCCn, cmpCCo, tstn, tstna, tstnn, tstno, tstz, tstza, tstzn, tstzo
轉送指令	A	mov, movcf, mvclcas, mvclovs, setlo, vcchk
NOP 指令	A	nop
移位指令 1	S1	asl, aslvh, aslvw, asr, asrvh, asrvw, lsl, lsr, rol, ror
移位指令 2	S2	aslp, aslpvw, asrp, asrpvw, lslp, lsrp

【表 4】

種類	運算器	指令執行碼
抽出指令	S2	ext, extb, extbu, exth, exthu, extr, extru, extu
遮蔽指令	C	msk, mskgen
飽和指令	C	sat12, sat9, satb, satbu, sath, satw
變換指令	C	valn, valn1, valn2, valn3, valnvc1, valnvc2, valnvc3, valnvc4, vhpkb, vhpkh, vhunpkb, vhunpkh, vintlhb, vintlhh, vintllb, vintllh, vlpkb, vlpkbu, vlpkh, vlpkhu, vlunpkb, vlunpkbu, vlunpkh, vlunpkhu, vstovb, vstovh, vunpk1, vunpk2, vxchngh, vexth
位元計數指令	C	bcnt1, bseq, bseq0, bseq1
其他	C	byterev, extw, mskbrvb, mskbrvh, rndvh, movp
乘法指令 1	X1	fmulhh, fmulhhr, fmulhw, fmulhww, hmul, lmul
乘法指令 2	X2	fmulww, mul, mulu
積和指令 1	X1	fmachh, fmachhr, fmachw, fmachww, hmac, lmac
積和指令 2	X2	fmacww, mac
積差指令 1	X1	fmsuhh, fmsuhhr, fmsuhw, fmsuww, hmsu, lmsu
積差指令 2	X2	fmsuww, msu
除法指令	DIV	div, divu
除錯指令	DBGM	dbgm0, dbgm1, dbgm2, dbgm3

【表 5】

種類	運算器	指令執行碼
SIMD 算術運算指令	A	vabshvh, vaddb, vaddh, vaddhvc, vaddhvh, vaddrhvc, vaddsb, vaddsh, vaddsrb, vaddsrh, vasubb, vcchk, vhaddh, vhaddhvh, vhsubh, vhsubhvh, vladdh, vladdhvh, vlsubh, vlsubhvh, vnegb, vnegh, vneghvh, vsaddb, vsaddh, vsgnh, vsrsubb, vsrsubh, vssubb, vssubh, vssbb, vsubh, vsubhvh, vsubsh, vsumh, vsumh2, vsumrh2, vxaddh, vxaddhvh, vxsubh, vxsubhvh, vmaxb, vmaxh, vminb, vminh, vmovt, vsel
SIMD 比較指令	A	vcmpeqb, vcmpeqh, vcmpgeb, vcmpgeh, vcmpgtb, vcmpgth, vcmpleb, vcmpleh, vcmpltb, vcmplth, vcmpneb, vcmpneh, vscmpeqb, vscmpeqh, vscmpgeb, vscmpgeh, vscmpgtb, vscmpgth, vscmpleb, vscmpleh, vscmpltb, vscmplth, vscmpneb, vscmpneh
SIMD 移位指令 1	S1	vaslb, vaslh, vaslvh, vasrb, vasrh, vasrvh, vlslb, vlslh, vlsrb, vlsrh, vrolb, vrolh, vrorb, vrorh
SIMD 移位指令 2	S2	vasl, vaslvw, vasr, vasrvw, vlsl, vlsr
SIMD 飽和指令	C	vsath, vsath12, vsath8, vsath8u, vsath9
SIMD 其他指令	C	vabssumb, vrndvh
SIMD 乘法指令	X2	vfmulh, vfmulhr, vfmulw, vhfmulh, vhfmulhr, vhfmulw, vhmul, vlfmulh, vlfmulhr, vlfmulw, vlmul, vmul, vpfmulhww, vxfmulh, vxfmulhr, vxfmulw, vxmul
SIMD 積和指令	X2	vfmach, vfmachr, vfmacw, vhfmach, vhfmachr, vhfmacw, vhmac, vlfmach, vlfmachr, vlfmacw, vlmac, vmac, vpfmachww, vxfmach, vxfmachr, vxfmacw, vxmac
SIMD 積差指令	X2	vfmsuh, vfmsuw, vhfmsuh, vhfmsuw, vhmsu, vlfmsuh, vlfmsuw, vlmsu, vmsu, vxfmsuh, vxfmsuw, vxmsu

又，表中的「運算器」係指該指令使用之運算器。運算器之略寫符號之意義各如以下說明。即，「A」為ALU指令、「B」為分支指令、「C」為變換指令、「DIV」為除法指令、「DBGM」為除錯指令、「M」為記憶體存取指令、「S1」、「S2」為移位指令、「X1」、「X2」則意指乘法指令。

第20圖係顯示本處理器1所執行之指令格式之圖。

又，圖中之略寫符號之意義如以下說明。即，「P」為述詞(執行條件：指定8個條件旗標C0~C7之任一者)、「OP」為執行碼欄、「R」為暫存器欄、「I」為立即值欄、「D」則意指位移欄。

第21圖~第36圖係用以說明本處理器1所執行之指令的概略機能之圖。亦即，第21圖係用以說明隸屬於種類「ALUadd (加法)類」之指令之圖；第22圖係用以說明隸屬於種類「ALUsub (減法)類」之指令之圖；第23圖係用以說明隸屬於種類「ALU logic(邏輯運算)類等」之指令之圖；第24圖係用以說明隸屬於種類「CMP(比較運算)類」之指令之圖；第25圖係用以說明隸屬於種類「mul(乘法)類」之指令之圖；第26圖係用以說明隸屬於種類「mac(積和運算)類」之指令之圖；第27圖係用以說明隸屬於種類「msu(積差運算)類」之指令之圖；第28圖係用以說明隸屬於種類「MEMld(記憶體讀出)類」之指令之圖；第29圖係用以說明隸屬於種類「MEMstore(記憶體寫出)類」之指令之圖；第30圖係用以說明隸屬於種類「BRA(分支)類」之指令之圖；第31圖係用以說明隸屬於種類「BSasl(算術

桶移位)類等」之指令之圖；第 32 圖係用以說明隸屬於種類「BSlsr(邏輯桶移位)類」之指令之圖；第 33 圖係用以說明隸屬於種類「CNVvaln(算術變換)類」之指令之圖；第 34 圖係用以說明隸屬於種類「CNV(一般變換)類」之指令之圖；第 35 圖係用以說明隸屬於種類「SATvlpk (飽和處理)類」之指令之圖；第 36 圖係用以說明隸屬於種類「ETC(其他)類」之指令之圖。

在其等圖中，項目「SIMD」係指該指令型態(SISD(SINGLE)或 SIMD 之區別)，項目「尺寸」係指成為運算對象之每一個運算元的尺寸，項目「指令」係指該指令之運算碼，項目「運算元」係指該指令之運算元，項目「CFR」係指條件旗標暫存器之變化，項目「PSR」係指處理器狀態暫存器之變化，項目「典型動作」係指動作的概要，項目「運算器」係指所使用之運算器，項目「3116」係指指令尺寸。

其次，針對幾個具有特徵之指令，說明本處理器 1 之動作。此外，指令動作說明所用之各種記號之意義係如下列表 6~表 10 所示。

【表 6】

記號	意　思
X[i]	X 之位元號碼 i
X[i：j]	由 X 之位元號碼 j 至位元號碼 i
X：Y	X 與 Y 之連結
{n{X}}	X 之 n 次重複
sextM(X,N)	將 X 由 N 位元寬度做帶正負號延伸成 M 位元寬度。M 之內定為 32。N 之內定為 X 之所有位元寬度。
uextM(X,N)	將 X 由 N 位元寬度做 0 延伸成 M 位元寬度。M 之內定為 32。N 之內定為 X 之所有位元寬度。
smul(X,Y)	帶正負號乘式 X＊Y
umul(X,Y)	無符號乘式 X＊Y
sdiv(X,Y)	帶正負號之除式之商的整數部 X/Y
smod(X,Y)	與被除數同一帶正負號之餘數
udiv(X,Y)	無符號之除式的商 X/Y
umod(X,Y)	餘數
abs(X)	絕對值

bseq(X,Y)	for (i=0; i<32; i++) {
	if (X[31－i] ! =Y) break;
	}
	result = i;

bcnt(X,Y)	S = 0;
	for (i=0; i<32; i++) {
	if (X[i] ==Y) S++;
	}
	result = S

max(X,Y)	result = (X>Y)? X : Y
min(X,Y)	result = (X<Y)? X : Y
tstz(X,Y)	X & Y ==0
tstn(X,Y)	X & Y ! =0

【表 7】

記號		意　思	
Ra	Ra[31:0]	編號 a 之暫存器	(0 <=a <= 31)
Ra+1	R(a+1)[31:0]	編號 a+1 之暫存器	(0 <=a <= 30)
Rb	Rb[31:0]	編號 b 之暫存器	(0 <=b <= 31)
Rb+1	R(b+1)[31:0]	編號 b+1 之暫存器	(0 <=b <= 30)
Rc	Rc[31:0]	編號 c 之暫存器	(0 <=c <= 31)
Rc+1	R(c+1)[31:0]	編號 c+1 之暫存器	(0 <=c <= 30)
Ra2	Ra2[31:0]	編號 a2 之暫存器	(0 <=a2 <= 15)
Ra2+1	R(a2+1)[31:0]	編號 a2+1 之暫存器	(0 <=a2 <= 14)
Rb2	Rb2[31:0]	編號 b2 之暫存器	(0 <=b2 <= 15)
Rb2+1	R(b2+1)[31:0]	編號 b2+1 之暫存器	(0 <=b2 <= 14)
Rc2	Rc2[31:0]	編號 c2 之暫存器	(0 <=c2 <= 15)
Rc2+1	R(c2+1)[31:0]	編號 c2+1 之暫存器	(0 <=c2 <= 14)
Ra3	Ra3[31:0]	編號 a3 之暫存器	(0 <=a3 <= 7)
Ra3+1	R(a3+1)[31:0]	編號 a3+1 之暫存器	(0 <=a3 <= 6)
Rb3	Rb3[31:0]	編號 b3 之暫存器	(0 <=b3 <= 7)
Rb3+1	R(b3+1)[31:0]	編號 b3+1 之暫存器	(0 <=b3 <= 6)
Rc3	Rc3[31:0]	編號 c3 之暫存器	(0 <=c3 <= 7)
Rc3+1	R(c3+1)[31:0]	編號 c3+1 之暫存器	(0 <=c3 <= 6)
Rx	Rx[31:0]	編號 x 之暫存器	(0 <=x <= 3)

【表 8】

記號	意　思	
+	加法	
−	減法	
&	邏輯積	
|	邏輯和	
!	邏輯否定	
<<	邏輯左移位(算術左位移)	
>>	算術右移位	
>>>	邏輯右移位	
^	互斥邏輯和	
~	邏輯否定	
==	等於(相等)	
!=	不等於(不相等)	
>	大於	帶有正負號(視左邊、右邊之 MSB 帶有正負號)
>=	大於等於	帶有正負號(視左邊、右邊之 MSB 帶有正負號)
>(u)	大於	無符號(視左邊、右邊之 MSB 為無符號)
>=(u)	大於等於	無符號(視左邊、右邊之 MSB 為無符號)
<	小於	帶有正負號(視左邊、右邊之 MSB 帶有正負號)
<=	小於等於	帶有正負號(視左邊、右邊之 MSB 帶有正負號)
<(u)	小於	無符號(視左邊、右邊之 MSB 為無符號)
<=(u)	小於等於	無符號(視左邊、右邊之 MSB 為無符號)

【表 9】

記號	意　思
D(addr)	記憶體內之位址 addr 的雙字資料
W(addr)	記憶體內之位址 addr 的字資料
H(addr)	記憶體內之位址 addr 的半資料
B(addr)	記憶體內之位址 addr 的位元組資料
B(addr, bus_lock)	對記憶體內之位址 addr 的位元組資料進行存取，並同時將使用過之匯流排鎖定。(惟，不能進行鎖定之匯流排便不進行鎖定)
B(addr, bus_unlock)	對記憶體內之位址 addr 的位元組資料進行存取，並同時將使用過之匯流排鎖定予以解除。(惟，不能進行鎖定之匯流排或未進行鎖定之匯流排則將鎖定之解除加以忽視)
EREG(num)	編號 num 之延伸暫存器
EREG_ERR	在前一次之延伸暫存器存取發生錯誤時，為 1 不發生錯誤時，則為 0
<-	結果的寫入
=>	指令之同義字(以組合程式變換)
reg#(Ra)	通用暫存器 Ra 之暫存器編號(5 位元數值)
0x	16 進位之接頭詞
0b	2 進位之接頭詞
tmp	暫時變數
UD	未定義值(依賴實施或動態變化之值)
Dn	位移值(n 為自然數，表示位元數)
In	立即值(n 為自然數，表示位元數)

【表 10】

記　號	意　思
○構文說明	

if(條件){
　　條件成立時執行；
}else{
　　條件不成立時執行；
}	

條件 A 成立時執行，if(條件 A)；※不成立時即不執行	
for(式 1；式 2；式 3)	※與 C 語言同樣
(式 1)？式 2：式 3	※與 C 語言同樣
○用語說明	
針對在說明中所使用之用語下定義。	
整數乘法	用 smul 定義之乘法
固定小數點乘法	經整數運算後，即進行算術左移位。移位量係於 PSR.FXP 為 0 時，是 1 位元，為 1 時，則是 2 位元。
SIMD 運算	straight / cross / high / low / pair 令半字向量資料之高 16 位元為 RH、低 16 位元為 RL。對於 Ra 暫存器與 Rb 暫存器間之運算，將各運算定義如下。
straight	在 RHa 與 RHb、RLa 與 RLb 間進行運算。
cross	在 RHa 與 RLb、RLa 與 RHb 間進行運算。
high	在 RHa 與 RHb、RLa 與 RHb 間進行運算。
low	在 RHa 與 RLb、RLa 與 RLb 間進行運算。
pair	在 RH 與 RHb、RH 與 RLb 間進行運算。(RH 是 32 位元資料)

[指令 vcchk]

指令 vcchk 係一 SIMD 型指令，以判別藉 SIMD 型比較指令(vcmpCCb 等)所得之結果全部是否為 0，且將其結果設定於條件旗標暫存器(CFR)32 者。如下列指令時，

vcchk

如第 37 圖所示，處理器 1 係判定條件旗標暫存器(CFR)32

之向量條件 VC0～VC3(110)全部是否為 0，全部為 0 時，將條件旗標暫存器(CFR)32 之條件旗標 C4 及 C5 各設定為 1 及 0；不是時，則將條件旗標 C4 及 C5 各設定為 0 及 1。接著在條件旗標 C0~C3 儲存向量條件旗標 VC0~VC3。詳細動作係如第 38 圖所示者。

藉該指令，可將 SIMD 型比較指令之結果(尤其是一致與不一致)的取出加速完成。對於檔中的 EOF(End Of File)檢測等有效。

[指令 stbh、stbhp]

指令 stbh 係一用以將儲存於 1 個暫存器中之 2 個位元組資料(儲存於高 16 位元之位元組資料及儲存於低 16 位元之位元組資料)儲存於記憶體等之指令，且為對應於指令 ldbh(朝相反方向轉送資料)之指令。諸如下列指令時，

stbh (Ra),Rb

處理器 1 係藉 I/F 部 50 等，如第 39 圖所示，以暫存器 Ra 所示之位址的記憶區中存放暫存器 Rb 中儲存之 2 個位元組資料(暫存器 Rb 之第 16~23 位元及第 0~7 位元)。詳細動作係如第 40 圖所示者。

指令 stbhp 係一用以將儲存於 2 個暫存器(配對暫存器)中之 4 個位元組資料(儲存於各暫存器之高 16 位元之位元組資料及儲存於低 16 位元之位元組資料)存放於記憶體等之指令，且為對應於指令 ldbhp(朝相反方向轉送資料)之指令。例如下列指令時，

stbhp (Ra),Rb:Rb+1

處理器 1 係藉 I/F 部 50 等，如第 41 圖所示，在暫存器 Ra 所示之位址的記憶區存放暫存器 Rb 及 Rb+1 所儲存之 4 個位元組資料(各暫存器之第 16~23 位元及第 0~7 位元)。詳細動作係如第 42 圖所示者。

藉其等指令，以 16 位元 SIMD 型處理位元組資料時，即不需要做資料的類型變換，可使處理高速化。

[指令 sethi]

指令 sethi 係一不變更暫存器之低 16 位元，且於高 16 位元儲存立即值之指令。諸如下列指令時，

sethi Ra,I16

處理器 1 係如第 43 圖所示，在暫存器 Ra 之高 16 位元儲存 16 位元之立即值(I16)。此時，暫存器 Ra 之低 16 位元不變。詳細動作係如第 44 圖所示者。

藉該指令，與指令「mov Rb,I16」組合時，即可將 32 位元之立即值設定於暫存器者。

[指令 vaddhvc、vaddrhvc]

指令 vaddhvc 係一種 SIMD 型指令，藉向量條件旗標的值切換欲加的來源。例如下列指令時，

vaddhvc Rc,Ra,Rb

處理器 1 係藉運算部 40 等，如第 45 圖所示，以半字向量格式，將暫存器 Ra 的值與暫存器 Ra 或 Rb 的值相加，且將該結果儲存於暫存器 Rc。此時，要加暫存器 Ra 及 Rb 中任一值係按向量條件旗標 VC2 的值而定。具體而言，在向量條件旗標 VC2=1 時，將暫存器 Ra 的值與暫存器 Rb 的值

相加，而 VC2=0 時，則將暫存器 Ra 的值與暫存器 Ra 的值相加。詳細動作係如第 46 圖所示者。

該指令係對於影像處理中之動畫補償有效。對相加結果即暫存器 Rc 之值用 2 相除後的值則為暫存器 Ra 或暫存器 Ra 與 Rb 之平均值，如第 47 圖所示，對動畫補償的半像素(半像素單位的動畫補償)，具有不管是整數像素或半像素都可用同一程式處理之優點。

此外，指令 vaddrhvc 係相當於在上述指令 vaddhvc 的處理上附加上將相加結果捨進之處理者。例如下列指令時，

vaddrhvc Rc,Ra,Rb

處理器 1 係藉算術邏輯暨比較運算器 41 等，如第 48 圖所示，以半字向量格式，將暫存器 Ra 的值與暫存器 Ra 或 Rb 的值相加，進而加上用以捨進處理之 1，將該結果儲存於暫存器 Rc。其他動作係與指令 vaddhvc 同樣。詳細動作係如第 49 圖所示者。

該指令當然亦對於影像處理中之動畫補償有效。

此外，對上述指令 vaddhvc 及指令 vaddrhvc 各自的功能而言，亦可附加 1 位元右移位(用 2 相除之處理)。藉此，處理器可直接算出整數像素或半像素者。

又，亦可定義一兼備上述指令 vaddhvc 及指令 vaddrhvc 之功能的指令。例如，藉條件旗標值，亦可設定用以進行指令 vaddhvc 或指令 vaddhrvc 之動作的新指令。藉此，對於施有或沒有施行捨進處理之任一形態時，都可以同一程式進行處理。

[指令 vsgnh]

指令 vsgnh 係一 SIMD 型指令，藉暫存器之符號(正/負)及零以產生數值者。例如下列指令時，

vsgnh Ra,Rb

處理器 1 係如第 50 圖所示，以半字向量格式，暫存器 Ra 值為正時，將 1 儲存於暫存器 Rb，而為負時，則儲存－1，為 0 時，則儲存 0。詳細動作係如第 51 圖所示。

該指令係於某值為正時，輸出 1；為負時，輸出－1；為零時，則輸出 0，對影像處理中之逆量化有效。尤其藉本處理器 1，可使採用 SIMD 型而難以運算者高速化。

[指令 valnvc1、valnvc2、valnvc3、valnvc4]

指令 valnvc1 係一將資料做位元組調正，且藉向量條件旗標變更所取出之位元組資料之 SIMD 型指令。例如以下指令時，

valnvc1 Rc,Ra,Rb

如第 52 圖所示，處理器 1 係因應條件旗標暫存器(CFR)32 之位元 ALN[1:0]的值，使連結有暫存器 Ra 與暫存器 Rb 之位元列移位，俾行位元組調正，並將對應向量條件旗標 VC0 的值所取出之 4 個位元組資料儲存於暫存器 Rc。具體而言，在向量條件旗標 VC0=0 時，由業經位元組調正後之資料取出 4 個位元組資料 a,a,b,b，且將其等儲存於暫存器 Rc，而在向量條件旗標 VC0=1 時，則取出 4 個位元組資料 a,b,b,c，且將其等儲存於暫存器 Rc。詳細動作係如第 53 圖所示者。

該指令係對影像處理中之動畫補償有效。因此可以半字向量單位，將相加結果暫存器 Rc 的值用 2 相除後所得之值等於 a 及 b，或，(a+b)/2 及(b+c)/2 之值，因此如第 47 圖所示，對動畫補償的半像素(半像素單位的動畫補償)，具有不管是整數像素或半像素都可用同一程式處理之優點。。

又，如第 52 圖所示，指令 valnvc2、valnvc3 及 valnvc4 之基本動作係與上述指令 valnvc1 相同，只是由業經位元組調正之資料取出之位置不同而已。各詳細動作係如第 54 圖、第 55 圖、第 56 圖所示者。藉此，其等指令當然也對於影像處理中之動畫補償有效。

此外，按本發明，調正等單位係不限於位元組，除了位元組外，半字或半位元組等亦可。

[指令 addarvw]

指令 addarvw 係一將兩值相加，在相加對象的一方為正時進而加上 1 之指令。例如以下指令時，

addarvw Rc,Rb,Ra

如第 57 圖所示，處理器 1 係藉算術邏輯暨比較運算器 41 等，暫存器 Ra 值與暫存器 Rb 值相加。此時，暫存器 Ra 值為正時，進一步加上 1。詳細動作係如第 58 圖所示者。

該指令係對於「絕對值捨進(away from zero)」有效。如第 59 圖所示，先將成為絕對值捨進之對象的值儲存於暫存器 Ra，並於暫存器 Rb 儲存相當於較用以進行絕對值捨進之位元還低之位元相當的位元用 1 填入之值。隨即執行上述指令時，將對於暫存器 Ra 值(在此，最高位元為帶正負

號位元，由最高開始小數點位於第 2 位元與第 3 位元間之固定小數點資料)進行絕對值捨進後之結果儲存於暫存器 Rc。在第 58 圖所示之實施形態中，將除了暫存器 Rc 之高 2 位元之外的位元遮蔽，即可對於+0.5，得到+1，而對於－0.5 時，則得到－1，而實現絕對值捨進。因此該指令係對於影像處理中之絕對值捨進有效。

[指令 movp]

指令 movp 係一將任意兩個暫存器值轉送至相連續之兩個暫存器之指令。例如以下指令時，

movp Rc:Rc+1,Ra,Rb

如第 60 圖所示，處理器 1 係藉 I/F 部 50 等，將暫存器 Ra 值轉送至暫存器 Rc，且將暫存器 Rb 值轉送至暫存器 Rc+1。詳細動作係如第 61 圖所示者。

該指令係使獨立互不相干之兩個暫存器的移動以 1 循環進行，因此例如在迴路內部中，可展現減少迴路內循環數的效果。又，按該指令，無須進行暫存器再命名(renaming)(暫存器值破壞)，對進行資料在迴路世代間(疊代(iteration)間)之移動時亦有效。

此外，本發明之運算種類並不限於轉送(mov)者，亦可包括單項運算(neg 等)或二項運算(add)。例如，是指定了任意兩個暫存器(R0,R6)及相連續之兩個暫存器(R2,R3)之加法指令時，可以 1 個指令(1 個循環)執行兩個加法運算「R0+R2→R2」及「R6+R3→R3」。

[指令 jloop、settar]

指令 jloop 係一用以進行迴路中之分支及條件旗標(在此，為述詞)設定之指令。例如以下指令時，

jloop C6,Cm,TAR,Ra

處理器 1 係藉位址管理部 10b 等，(1)在條件旗標 Cm 設定 1，(2)在暫存器 Ra 值小於 0 時，於條件旗標 C6 設定 0，(3)於暫存器 Ra 值加上－1，且儲存於暫存器 Ra，(4)分支於分支暫存器(TAR)30d 所示之位址。在跳越緩衝器 10f(分支用指令緩衝器)未填入分支用指令時，則填入分支處的指令。詳細動作係如第 62 圖所示者。

另一方面，指令 settar 係一用以將分支處位址儲存於分支暫存器(TAR)30d，且進行條件旗標(在此指述詞)設定之指令。例如以下指令時，

settar C6,Cm,D9

處理器 1 係藉位址管理部 10b 等，(1)使程式計數器(PC)33 與位移值(D9)相加之位址儲存於分支暫存器(TAR)30d，(2)提取該位址之指令後再儲存於跳越緩衝器 10f(分支用指令緩衝器)，(3)將條件旗標 C6 設定 1，且將條件旗標 Cm 設定 0。詳細動作係如第 63 圖所示者。

其等指令 jloop 及指令 settar 係對按導言結尾除去型(以下簡稱導結除去(proepi)型)之軟體管線操作之高速化有效之指令，通常成對使用。又，軟體管線操作是一種藉編譯程式之迴路高速化手法之一，將迴路結構變換成導言部、核心部及結尾部，針對核心部，使各疊代(重複)與其前後之疊代相重疊時，可有效率地使多數指令並列執行。

又，導結除去型係指：如第 64 圖所示，令導言部及結尾部為按述詞之條件執行指令時，看到導言部及結尾部後予以除去者。在第 64 圖中，導結除去型 2 級軟體管線操作中，條件旗標 C6 及 C4 係各成為結尾指令(級 2)用、導言指令(級 1)用之述詞。

以下，乃針對其等之指令 jloop 及指令 settar 所具備之旗標轉送功能(條件旗標 Cm 之設定)之意義，一邊與不具旗標轉送功能之通常指令 jloop 及指令 settar 比較，一邊說明。

本實施形態中之上述指令 jloop 及指令 settar 不在處理器 1 之指令集時，即，只有通常指令 jloop 及指令 settar 在指令集中時，條件旗標 Cm 轉送須與指令 jloop 及指令 settar 個別獨立地予以進行。為此，有下列問題產生，即：

(1)增加了與原本迴路執行無關之旗標轉送指令，導至降低導結除去型軟體管線操作之功能；

(2)增加旗標間之資料依賴關係，因旗標間的資料依賴及配置限制等，便使功能降低；

(3)衍生須具有旗標間轉送指令之必要性，即，乃出現指令集中須具有原本不需要之旗標間轉送指令的必要性，導致指令集之位元欄擠壓。

例如，現在對於第 65 圖所示之 C 語言之原始程式採用通常指令 jloop 及指令 settar 時，編譯程式係藉導結除去型之軟體管線操作而產生第 66 圖所示之機器語程式。由該機械語程式之迴路部分(由標號 L00023 迄至指令 jloop)可知，勢必需要一用以設定條件旗標 C4 之指令(指令

cmpeq)，因此對於迴路執行須進行 3 循環。進而，還需要兩個用以設定及重設條件旗標 C4 之指令，便減少導結除去的效果。

對此，本實施形態之指令 jloop 及指令 settar 包括於指令集時，編譯程式含於指令集時，編譯程式係產生第 67 圖所示之機器語程式。從該機器語程式之迴路部分(由標號 L00023 迄至指令 jloop)可知，條件旗標 C4 之設定及重設係個別以指令 jloop 及 settar 進行，因此不需要該原由之特殊指令，迴路執行便只需 2 循環完成。

如此一來，指令「jloop C6,Cm,TAR,Ra」及指令「settar C6,Cm, D9」係具有減少 2 級導結除去型軟體管線操作之執行循環次數之效果。

又，本處理器 1 係不只具有適用於 2 級之軟體管線操作，且可適用於 3 級軟體管線操作之指令「jloop C6,C2:C4,TAR,Ra」及指令「settar C6,C2:C4,D9」。其等指令「jloop C6,C2:C4,TAR, Ra」及指令「settar C6,C2:C4,D9」係相當於上述 2 級用指令「jloop C6,Cm,TAR,Ra」及指令「settar C6,Cm,D9」中之暫存器 Cm 延伸成暫存器 C2、C3 及 C4 者。

亦即，如以下指令時，

jloop C6,C2:C4,TAR,Ra

處理器 1 係藉位址管理部 10b 等，(1)暫存器 Ra 小於 0 時則將 0 設定於條件旗標 C4，(2)將條件旗標 C3 之值轉送至條件旗標 C2，且將條件旗標 C4 之值轉送至條件旗標 C3

與 C6，(3)在暫存器 Ra 加上 1，並儲存於暫存器 Ra，(4)分支於分支暫存器(TAR)30d所示之位址。在跳越緩衝器 10f 中未填入分支處之指令時，則填入分支處的指令。詳細動作係如第 68 圖所示者。

又，如以下指令時，

settar C6,C2:C4,D9

處理器 1 係藉位址管理部 10b 等，(1)將程式計數器(PC)33與位移值(D9)相加後之位址儲存於分支暫存器(TAR)30d，(2)提取該位址之指令後將之儲存於跳越緩衝器 10f(分支用指令緩衝器)，(3)將條件旗標 C4 與 C6 設定為 1，且將條件旗標 C2 與 C3 設定為 0。詳細動作係如第 69 圖所示者。

其等 3 級用指令「jloop C6,C2:C4,TAR,Ra」及指令「settar C6,C2:C4,D9」中之條件旗標的作用係如第 70 圖所示者。如第 70(a)圖所示，按導結除去型 3 級軟體管線操作，條件旗標 C2、C3、C4 個別成為級 3 用、級 2 用、級 1 用之述詞。第 70(b)圖係顯示此時之旗標轉送所進行的執行推移之圖。

顯示其等指令「jloop C6,C2:C4,TAR,Ra」及指令「settar C6,C2:C4,D9」之旗標轉送的意義之程式例係如第 71 至 73 圖所示者。第 71 圖係顯示原始程式例；第 72 圖係顯示使用不具備如此之旗標轉送功能之指令所產生之機器語程式例；第 73 圖係顯示使用本實施形態中具備旗標轉送功能之指令 jloop 及指令 settar 所產生之機器語程式例。比較第 72 圖及第 73 圖後可知，使用本實施形態中具備旗標轉送功能

之指令 jloop 及指令 settar，即不需要 5 個指令，且迴路執行亦減少 1 循環。

此外，針對 4 級以上之軟體管線操作亦同，只要增加述詞用之條件旗標即可。

除了以上具有特徵之指令外，本處理器 1 還可執行上述第 21 圖至第 36 圖中未予以列出之如下列特徵之指令。。

[指令 vsada]

指令 vsada 係一用以算出差分絕對值和之 SIMD 型指令。如以下指令時，

vsada Rc,Ra,Rb,Rx

處理器 1 係藉算術邏輯暨比較運算器 41 等，如第 74 圖所示，對暫存器 Ra 之值與暫存器 Rb 之值間之差分以位元組單位做 SIMD 運算(算出 4 組位元組各自之差分)，取 4 個結果各自之絕對值後相加，且於該結果加上暫存器 Rx 之值後，再將該結果儲存於暫存器 Rc。詳細動作係如第 75(a)圖所示者。

又，處理器 1 對於上述指令 vasada 格式中沒有最後運算元(Rx)之格式的指令亦予以執行。如以下指令時，

vsada Rc,Ra,Rb

處理器 1 係藉算術邏輯暨比較運算器 41 等，對暫存器 Ra 之值與暫存器 Rb 之值間的差分以位元組單位做 SIMD 運算(算出 4 組位元組各自的差分)，取 4 結果各自的絕對值後相加，並將該結果儲存於暫存器 Rc。詳細動作係如第 75(b)圖所示者。

其等指令 vsada 即為指令 vasubb 與指令 vabssumb 之複合指令。指令 vasubb 係一 SIMD 型指令，以位元組單位，對 4 組 SIMD 資料個別相減，將其結果所得到之 4 個碼儲存於條件旗標暫存器。另一方面，指令 vabssumb 係一 SIMD 型指令，按條件旗標暫存器，以位元組單位，對 4 組 SIMD 資料個別做絕對值相加，且將該結果與另一 4 位元組資料相加者。

因此，藉該指令 vsada，與連續使用指令 vasubb 及指令 vabssumb 之形態相比，可以 1 循環算出差分絕對值和，使運算高速化。如此之指令 vsada 係對於影像處理之動畫預測中之差分絕對值和之算出等有效。

此外，依本發明，資料單位係不限於位元組，亦可用於半字或半位元組等等。

[指令 satss,satsu]

指令 satss 係一以任意位置(位數)將附帶正負號之值飽和於附帶正負號之值的指令。如以下指令時，

satss Rc,Ra,Rb

處理器 1 係藉飽和方塊(SAT)47a 等，如第 76 圖所示，暫存器 Ra 之值大於藉暫存器 Rb 界定之飽和值(暫存器 Rb 之 1 的補數)時，將該飽和值儲存於暫存器 Rc，而在暫存器 Ra 之值小於飽和值時，則將暫存器 Ra 之值儲存於暫存器 Rc。詳細動作係如第 77(a)圖所示者。

另一方面，指令 satsu 係一用以任意位置(位數)，將無符號之值飽和於帶有正負號之值之指令。如以下指令時，

satsu Rc,Ra,Rb

處理器 1 係藉飽和方塊(SAT)47a 等，暫存器 Ra 之值大於藉暫存器 Rb 界定之飽和值時，將其飽和值儲存於暫存器 Rc，而在暫存器 Ra 之值小於飽和值時，則將暫存器 Ra 之值儲存於暫存器 Rc。詳細動作係如第 77(b)圖所示者。

藉如此指令 satss 及指令 satsu，可做任意位置之飽和處理。因此在進行編譯程式設計時就無須將飽和位置配合特定位置，使程式設計變得簡單。

[指令 bytesel]

指令 bytesel 係一以位元組單位選擇兩個暫存器中之一值的指令。如以下指令時，

bytesel Rc,Ra,Rb,Rx

處理器 1 係藉運算部 40 等，如第 78 圖所示，根據暫存器 Rx 之值，使暫存器 Ra 及暫存器 Rb 等 8 個位元組資料中任一資料儲存於暫存器 Rc 之動作按暫存器 Rc 之 4 個位元組並列進行。詳細動作係如第 79(a)圖，而暫存器 Rx 與所選擇之位元組資料間之關係則如第 79(b)圖所示者。

又，處理器 1 對如下之格式之指令 bytesel 亦予以執行。即，如以下指令時，

bytesel Rc,Ra,Rb,I12

處理器 1 係藉運算部 40 等，根據 12 位元之立即值，以位元組單位，將暫存器 Ra 及暫存器 Rb 等 8 個位元組資料之任一資料儲存於暫存器 Rc 之動作按暫存器 Rc 之 4 個位元組並列進行。詳細動作係如第 79(c)圖，立即值 I12 與所選

擇之位元組資料間之關係則如第第 79(d)圖所示者。

藉該指令 bytesel，可將位元組資料儲存於暫存器之任意位置，因此可使資料的對調重覆等高速化。又，亦具有可提昇 SIMD 運算適應性之效果。

又，按上述指令「bytesel Rc,Ra,Rb,Rx」，亦可利用暫存器 Rx 之空白位元等，進行指令上述位元組資料各儲存/不儲存於 Rc[31:24]、Rc[23:16]、Rc[15:8]、Rc[7:0]者。藉此，可以位元組單位選擇是否更新暫存器 Rc 之值。

此外，依本發明，資料單位係不限於位元組，亦可為半字或半位元組等。

[延伸 SIMD 運算結果之指令]

本處理器 1 不只可執行如上之指令，還可執行用以進行與 SIMD 運算相關之補助性處理之處理。

例如，處理器 1 係附與某一指令時，如第 80 圖所示，進行將部分 SIMD 運算結果做位元延伸(正負號延伸或零延伸)之補助性處理。第 80 圖係顯示針對儲存於 2 個暫存器之值，以半字單位，對處於直接位置關係或交叉位置關係之資料進行 SIMD 運算之樣態，第 80(a)圖係顯示將所得到之結果的低半字延伸成字之處理；第 80(b)圖則顯示將所得到之結果的高半字延伸成字之處理。

此外，將位於直接位置關係之資料以半字單位進行 SIMD 運算之指令即為諸如指令 vaddh 之類，而對位於交叉位置關係之資料則以半字單位進行 SIMD 運算之指令則為諸如指令 vxaddh 之類。

又，處理器 1 係附與某一指令時，如第 81 圖所示，進行對所有 SIMD 運算結果做位元延伸之補助性處理。第 81 圖係顯示針對儲存於 2 個暫存器之值，以半字單位，對處於直接位置關係或交叉位置關係之資料進行 SIMD 運算之樣態，顯示將所得到之 2 個半字各自延伸成字之處理。

用以延伸如此之 SIMD 運算結果之指令係對於 SIMD 運算後進行帶正負號延伸或零延伸以整合資料尺寸時有效，可使 SIMD 運算與延伸處理以 1 循環執行。

進而，處理器 1 係對於與 SIMD 運算有關聯之補助性指令，亦可執行藉條件旗標等所指定之種類的 SIMD 運算。例如，如第 82 圖所示，處理器 1 係藉條件旗標使第 1 及第 2 運算各指定為「加法」及「減法」時，對於 2 個暫存器之值，以半字單位，對處於直接位置關係或交叉位置關係之兩資料，個別進行加法及減法。

舉例而言，條件旗標 C0 及 C1 為「1,0」時，處理器 1 係藉算術邏輯暨比較運算器 41 等，

(1)將暫存器 Ra 之高半字與暫存器 Rb 之高半字相加，且將其結果儲存於暫存器 Rc 之高半字，與此處理同時，

(2)由暫存器 Ra 之低半字減去暫存器 Rb 之低半字，且將該結果儲存於暫存器 Rc 之低半字。

可指定如此 SIMD 運算種類之指令，對於其運算種類並非為固定而是有賴於其他處理結果而加以決定之處理有效。

此外，本發明係按上述運算(1)及(2)，可適用於不使用

暫存器 Rb 之形態。例如，亦可進行以下處理，即：

(1)將暫存器 Ra 之高半字與暫存器 Ra 之低半字相加，且將該結果儲存於暫存器 Rc 之高半字，與該處理同時，

(2)由暫存器 Ra 之高半字減去暫存器 Ra 之低半字，且將該結果儲存於暫存器 Rc 之低半字。

【發明之效果】

由以上說明可知，本發明之處理器係執行一種 SIMD 型特徵指令，即，辨別藉 SIMD 型比較指令之結果全部是否為 0，將該結果設定於條件旗標。藉此，使 SIMD 型比較指令之結果(尤其是一致及不一致)取出變得快速，便使以多數像素值為單位之比較處理高速化，且檔中之 EOF 檢測等高速化。

又，本發明之處理器係執行一種具有將 1 個暫存器所儲存之 2 個位元組資料(儲存於高 16 位元之位元組資料及儲存於低 16 位元之位元組資料)儲存於記憶體等之特徵的指令。藉此，例如以 16 位元 SIMD 型處理位元組資料時，即不需要資料的格式變換，且使處理高速化。

又，本發明之處理器係執行一具有在不變更暫存器之低 16 位元之狀態下將高 16 位元儲存立即值之特徵指令。藉此，與另一指令「mov Rb,I16」相組合，可將 32 位元立即值設定於暫存器。

又，本發明之處理器係執行具有一藉向量條件旗標之值以切換可加之來源之 SIMD 型特徵指令。藉此，在動畫補償之半像素(半像素單位之動畫補償)中，不管是整數像素及

半像素都可以同一程式進行處理。

又，本發明之處理器係執行一具有藉暫存器之符號(正/負)及零而產生值之 SIMD 型特徵指令。藉此，某值為正時，輸出 1；而為負時，輸出 1，為零時，則輸出 0，因此可使影像處理中之逆量化高速化。

又，本發明之處理器係執行一對資料進行單位長度調正，藉向量條件旗標變更單位長度資料之 SIMD 型特徵指令。藉此，在動畫補償之半像素(半像素單位之動畫補償)中，不管是整數像素及半像素都可以同一程式進行處理。

又，本發明之處理係執行將兩值相加，相加對象的一者為正時，進一步加 1 之特徵指令。藉此，可使影像處理之絕對值捨進處理高速化。

又，本發明之處理器係執行具有將任意兩暫存器之值轉送至相連續之兩暫存器之特徵指令。藉此，可以 1 循環進行呈獨立而互不相干之兩暫存器的移動，例如在迴路內部，可發揮減少迴路內循環數之效果。又，藉該指令，無須進行暫存器再命名(暫存器值的破壞)，對進行資料於迴路世代間(疊代間)之移動時亦有效。

又，本發明之處理器係執行具有進行迴路中之分支及條件旗標(在此為述詞)設定之特徵指令。藉此，可實現藉導言結尾除去型之軟體管線操作之迴路高速化。

又，本發明之處理器係執行具有算出差分絕對值和之 SIMD 型特徵指令。藉此，可使影像處理之動畫預測中之差分絕對值和的算出等處理高速化。

又，本發明之處理器係執行一具有於任意位置(位數)將帶正負號之值飽和於帶正負號之值之特徵指令。藉此，在組合程式設計時無須將飽和位置配合特定的位置，使程式設定變成簡單。

又，本發明之處理器係執行一具有以單位長度單位選擇兩暫存器中之任一值之特徵指令。藉此，可將單位長度資料儲存於暫存器中任意位置，因此可使資料的對調重覆等高速化。又，亦具有可提昇 SIMD 運算之適應性的效果。

又，本發明之處理器係執行具有將 SIMD 運算的結果延伸之特徵指令。藉此，可使在施行 SIMD 運算後再整合資料尺寸之處理以 1 循環執行者。

又，本發明之處理器係亦可執行藉條件旗標等所指定之種類的 SIMD 運算。藉此，可以同一程式實現依據其他處理結果而決定運算種類般之動畫處理。

如上，本發明之處理器係可執行高機能SIMD運算，且高速執行多媒體處理時所需之各種數位信號處理，可使用作為行動電話、移動式AV機器、數位TV、DVD等共通之核心處理器，對於期望高性能、高成本效益之多媒體機器出現的現在而言其實用價值可說是極高者。

【圖式簡單說明】

第1圖係本發明之處理器的概略方塊圖。

第2圖係該處理器之算術邏輯暨比較運算器之概略圖。

第3圖係顯示該處理器之桶移位器結構的方塊圖。

第4圖係顯示該處理器之變換器結構之方塊圖。

第5圖係顯示該處理器之除法器結構之方塊圖。

第6圖係顯示該處理器之除法暨乘積和運算器結構之方塊圖。

第7圖係顯示該處理器之指令控制部結構之方塊圖。

第8圖係顯示該處理器之通用暫存器(R0~R31)結構之方塊圖。

第9圖係顯示該處理器之鏈接暫存器(LR)結構之方塊圖。

第10圖係顯示該處理器之分支暫存器(TAR)結構之方塊圖。

第11圖係顯示該處理器之程式狀態暫存器(PSR)結構之圖。

第12圖係顯示該處理器之條件旗標暫存器(CFR)結構之方塊圖。

第13(a)、13(b)圖係顯示該處理器之累加器(M0,M1)結構之方塊圖。

第14圖係顯示該處理器之程式計數器(PC)結構之方塊圖。

第15圖係顯示該處理器之PC保留用暫存器(IPC)結構之方塊圖。

第16圖係顯示該處理器之PSR保留用暫存器(IPSR)結構之方塊圖。

第17圖係顯示該處理器之管線動作之時序圖。

第18圖係顯示該處理器所進行之指令執行時之各管線

動作之時序圖。

第19圖係顯示該處理器之並列動作之圖。

第20圖係顯示該顯示器執行之指令的格式之圖。

第21圖係用以說明隸屬於種類「ALUadd(加法)類」之指令之圖。

第22圖係用以說明隸屬於種類「ALUsub(減法)類」之指令之圖。

第23圖係用以說明隸屬於種類「ALUlogic(邏輯運算)類等」之指令之圖。

第24圖係用以說明隸屬於種類「CMP(比較運算)類」之指令之圖。

第25圖係用以說明隸屬於種類「mul(乘法)類」之指令之圖。

第26圖係用以說明隸屬於種類「mac(積和運算)類」之指令之圖。

第27圖係用以說明隸屬於種類「msu(積差運算)類」之指令之圖。

第28圖係用以說明隸屬於種類「MEMld(記憶體讀出)類」之指令之圖。

第29圖係用以說明隸屬於種類「MEMstore(記憶體寫出)類」之指令之圖。

第30圖係用以說明隸屬於種類「BRA(分支)類」之指令之圖。

第31圖係用以說明隸屬於種類「BSasl(算術桶移位)類

等」之指令之圖。

第32圖係用以說明隸屬於種類「BSlsr(邏輯桶移位)類」之指令之圖。

第33圖係用以說明隸屬於種類「CNVvaln(算術變換)類」之指令之圖。

第34圖係用以說明隸屬於種類「CNV(一般變換)類」之指令之圖。

第35圖係用以說明隸屬於種類「SATvlpk (飽和處理)類」之指令之圖。

第36圖係用以說明隸屬於種類「ETC(其他)類」之指令之圖。

第37圖係顯示對於指令「vcchk」之處理器的動作之圖。

第38圖係用以說明指令「vcchk」之詳細動作之圖。

第39圖係顯示對於指令「stbh (Ra),Rb」之處理器之動作的圖。

第40圖係用以說明指令「stbh (Ra),Rb」之詳細動作之圖。

第41圖係顯示對於指令「stbhp (Ra),Rb:Rb+1」之處理器的動作之圖。

第42圖係用以說明指令「stbhp (Ra),Rb:Rb+1」之詳細動作之圖。

第43圖係顯示對於指令「sethi Ra,I16」之處理器的動作之圖。

第44圖係用以說明指令「sethi Ra,I16」之詳細動作之

圖。

第45圖係顯示對於指令「vaddhvc Rc,Ra,Rb」之處理器的動作之圖。

第46圖係用以說明指令「vaddhvc Rc,Ra,Rb」之詳細動作之圖。

第47圖係用以說明影像處理中之動畫補償之圖。

第48圖係顯示對於指令「vaddrhvc Rc,Ra,Rb」之處理器的動作之圖。

第49圖係用以說明指令「vaddrhvc Rc,Ra,Rb」之詳細動作之圖。

第50圖係顯示對於指令「vsgnh Ra,Rb」之處理器的動作之圖。

第51圖係用以說明指令「vsgnh Ra,Rb」之詳細動作之圖。

第52圖係顯示對於指令「valnvc1 Rc,Ra,Rb」之處理器的動作之圖。

第53圖係用以說明指令「valnvc1 Rc,Ra,Rb」之詳細動作之圖。

第54圖係用以說明指令「valnvc2 Rc,Ra,Rb」之詳細動作之圖。

第55圖係用以說明指令「valnvc3 Rc,Ra,Rb」之詳細動作之圖。

第56圖係用以說明指令「valnvc4 Rc,Ra,Rb」之詳細動作之圖。

第57圖係顯示對於指令「addarvw Rc,Rb,Ra」之處理器的動作之圖。

第58圖係用以說明指令「addarvw Rc,Rb,Ra」之詳細動作的圖。

第59(a)、(b)圖係顯示「絕對值捨進(由零捨去；away from zero)」的動作之圖。

第60圖係顯示對於指令「movp Rc:Rc+1,Ra,Rb」之處理器的動作之圖。

第61圖係用以說明指令「movp Rc:Rc+1,Ra,Rb」之詳細動作的圖。

第62圖係用以說明指令「jloop C6,Cm,TAR,Ra」之詳細動作的圖。

第63圖係用以說明指令「settar C6,Cm,D9」之詳細動作的圖。

第64圖係顯示導言結尾除去型2級軟體管線操作之圖。

第65圖係顯示C語言之原始程式之列表的圖。

第66圖係顯示使用通常的指令jloop及指令settar所產生之機器語言程式例。

第67圖係顯示使用本實施形態之指令jloop及指令settar所產生之機器語言程式例。

第68圖係用以說明指令「jloop C6,C2:C4,TAR,Ra」之詳細動作之圖。

第69圖係用以說明指令「settar C6,C2:C4,D9」之詳細動作之圖。

第70(a)、(b)圖係顯示導言結尾除去除去型3級軟體管線操作之圖。

第71圖係顯示C語言之原始程式的列表之圖。

第72圖係顯示使用通常之指令jloop及指令settar所產生之機器語言程式之例。

第73圖係顯示使用本實施形態之指令jloop及指令settar所產生之機器語言程式之例。

第74圖係顯示對於指令「vsada Rc,Ra,Rb,Rx」之處理器的動作之圖。

第75(a)圖係用以說明指令「vsada Rc,Ra,Rb,Rx」之詳細動作，第75(b)圖係說明指令「vsada Rc,Ra,Rb」之詳細動作之圖。

第76(a)、(b)圖係係顯示對於指令「satss Rc,Ra,Rb」之處理器的動作之圖。

第77(a)圖係用以說明指令「satss Rc,Ra,Rb」、第77(b)圖係說明指令「satsu Rc,Ra,Rb」之詳細動作的圖。

第78圖係顯示對於指令「bytesel Rc,Ra,Rb,Rx」之處理器的動作之圖。

第79(a)圖係顯示指令「bytesel Rc,Ra,Rb,Rx」之詳細動作；第79(b)圖係係顯示暫存器Rx與所選擇之位元組資料間的關係；第79(c)圖係顯示指令「bytesel Rc,Ra,Rb,I12」之詳細動作；第79(d)圖係顯示立即值I12與所選擇之位元組資料間的關係者。

第80(a)、(b)圖係顯示對部分SIMD運算結果進行位元

延伸(正負號延伸或零延伸)之形態的圖。

第81圖係顯示將SIMD運算結果全部進行位元延伸之形態的圖。

第82圖係顯示執行藉條件旗標等所指定之種類的SIMD運算之形態的圖。

【主要元件符號說明】

1…處理器

10…指令控制部

10a…指令高速緩衝記憶體

10b…位址管理部

10c~10e…指令緩衝器

10f…跳越緩衝器

10g…旋轉部

20…解碼部

30…暫存器檔

30a…通用暫存器(R0~R31)

30b…累加器(MH,ML)

30c…鏈接暫存器(LR)

30d…分支暫存器(TAR)

31…程式狀態暫存器(PSR)

32…條件旗標暫存器(CFR)

33…程式計數器(PC)

34…PC保留用暫存器(IPC)

35…PSR保留用暫存器(IPSR)

40…運算部

41~43…算術邏輯暨比較運算器

41a…ALU部

41b…飽和處理部

41c…旗標部

44…乘法暨積和運算器

44a、44b…乘法器

44c~44e…加法器

44f…選擇器

44g…飽和處理部

45…桶移位器

45a、45b…選擇器

45c…高位元桶移位器

45d…低位元桶移位器

45e…飽和處理部

46…除法器

47…變換器

47a…SAT方塊

47b…BSEQ方塊

47c…MSKGEN方塊

47d…VSUMB方塊

47e…BCNT方塊

47f…IL方塊

50…I/F部

60…指令記憶體部

70…資料記憶體部

80…延伸暫存器部

90…I/O介面部

200945190

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；惟已有申請案號者請填寫）

※ 申請案號：98144440

※ 申請日：92.9.24　　※IPC 分類：G06F 9/38 (2006.01)

原申請案號：由第 92126392 號申請案分割。

一、發明名稱：(中文/英文)

處理器

PROCESSOR

二、中文發明摘要：

本發明之課題係於提供一種執行高機能SIMD運算之處理器。本發明之處理器係包含有解碼器20及運算部40等，解碼部20係解讀指令vcchk時，運算部40等係判定條件旗標暫存器(CFR)32之向量條件旗標VC0~VC3(110)全部是否為0，在全部為0時，將條件旗標暫存器(CFR)(32)之條件旗標C4及C5各設定為1及0；不是時，則將條件旗標C4及C5各設為0及1。接著，條件旗標C0~C3中儲存向量條件旗標VC0~VC3。

三、英文發明摘要：

A processor according to the present invention includes a decoding unit 20, an operation unit 40 and others. When the decoding unit 20 decodes Instruction vcchk, the operation unit 40 and the like judges whether vector condition flags VC0~VC3 (110) of a condition flag register (CFR) 32 are all zero or not, and (i) sets condition flags C4 and C5 of the condition flag register (CFR) 32 to 1 and 0, respectively, when all of the vector condition flags VC0~VC3 are zero, and (ii) sets the condition flags C4 and C5 to 0 and 1, respectively, when not all the vector condition flags are zero. Then, the vector condition flags VC0~VC3 are stored in the condition flags C0~C3.

七、申請專利範圍：

1. 一種處理器，係用以解讀並執行指令者，其特徵在於包含有：

解讀裝置，係用以解讀指令者；及

執行裝置，係用以根據解讀裝置之解讀結果執行指令者；

前述執行裝置係於已藉前述解讀裝置解讀將第 1 及第 2 資料作為運算元之加法指令時，在前述第 1 資料為零或正時，產生一將前述第 1 資料、前述第 2 資料與 1 相加之結果，而在前述第 1 資料為負時，則產生一將前述第 1 資料與前述第 2 資料相加之結果。

2. 如申請專利範圍第 1 項之處理器，其中前述第 1 資料係一成為絕對值捨進之對象的資料，

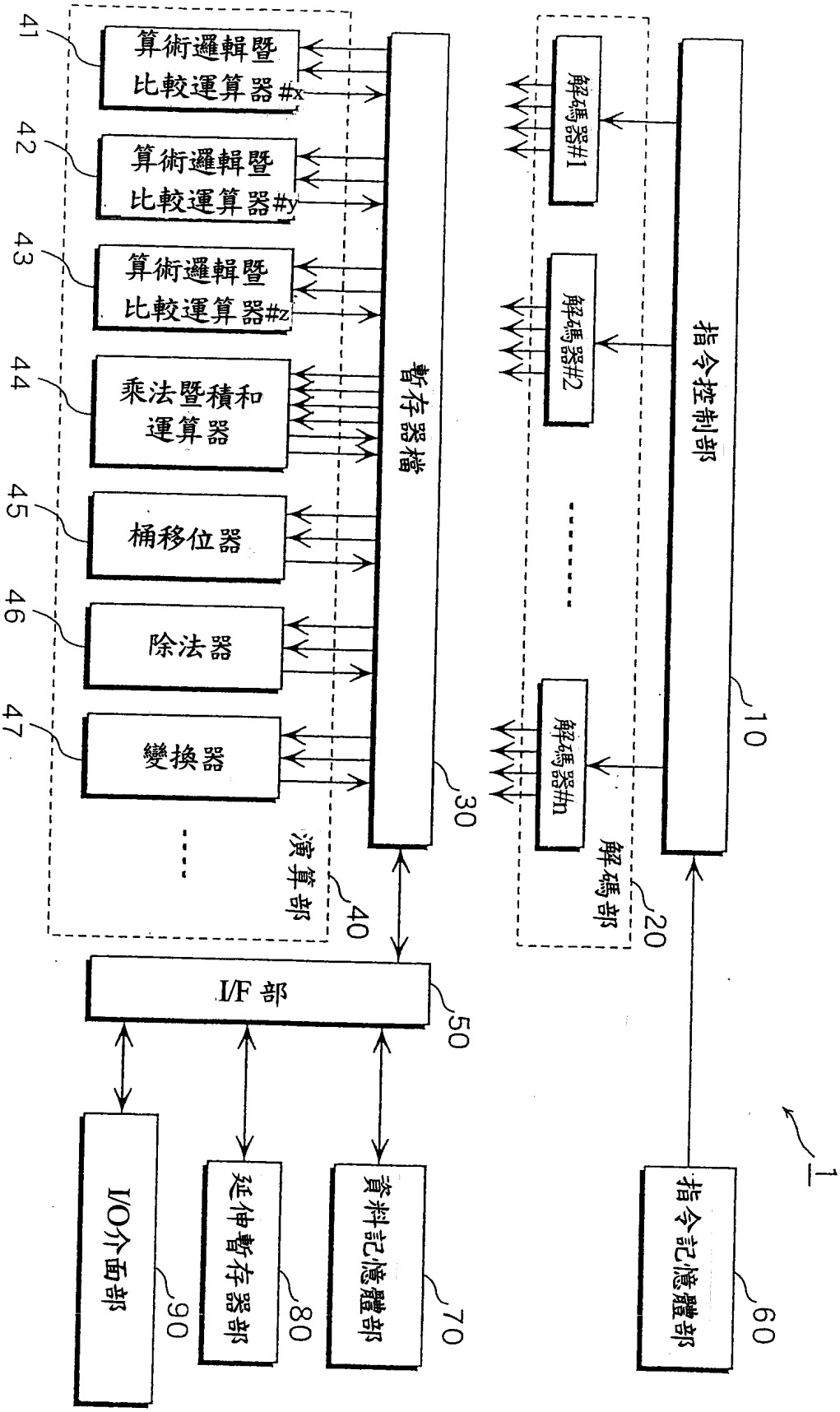
前述第 2 資料係一已指定欲進行絕對值捨進之位數的資料。

3. 如申請專利範圍第 2 項之處理器，其中前述第 2 資料係一將與對前述第 1 資料進行絕對值捨進之位數相對應之位數作為 1，而將其餘位數作為零之值。

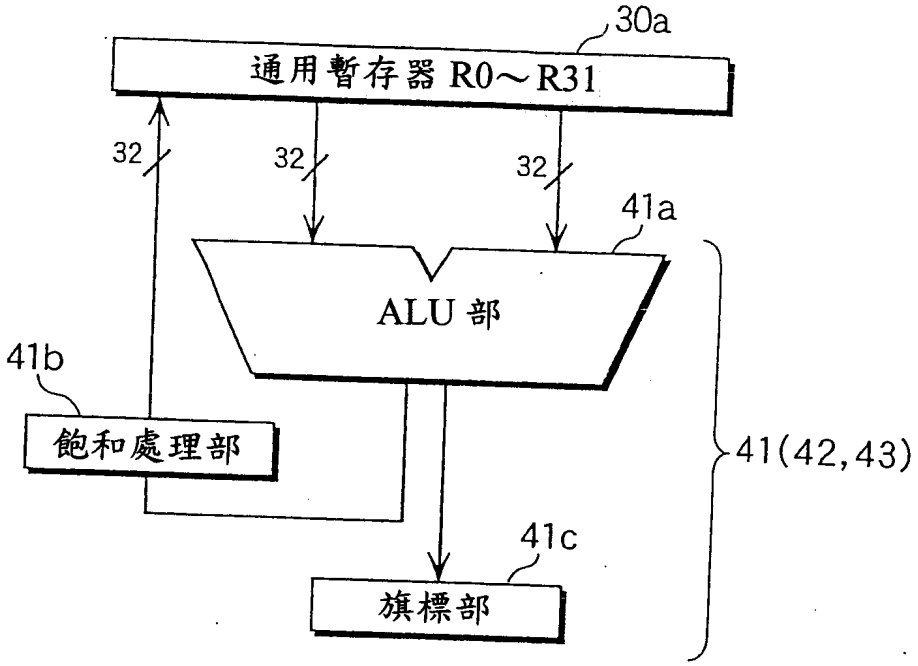
200945190

98144440

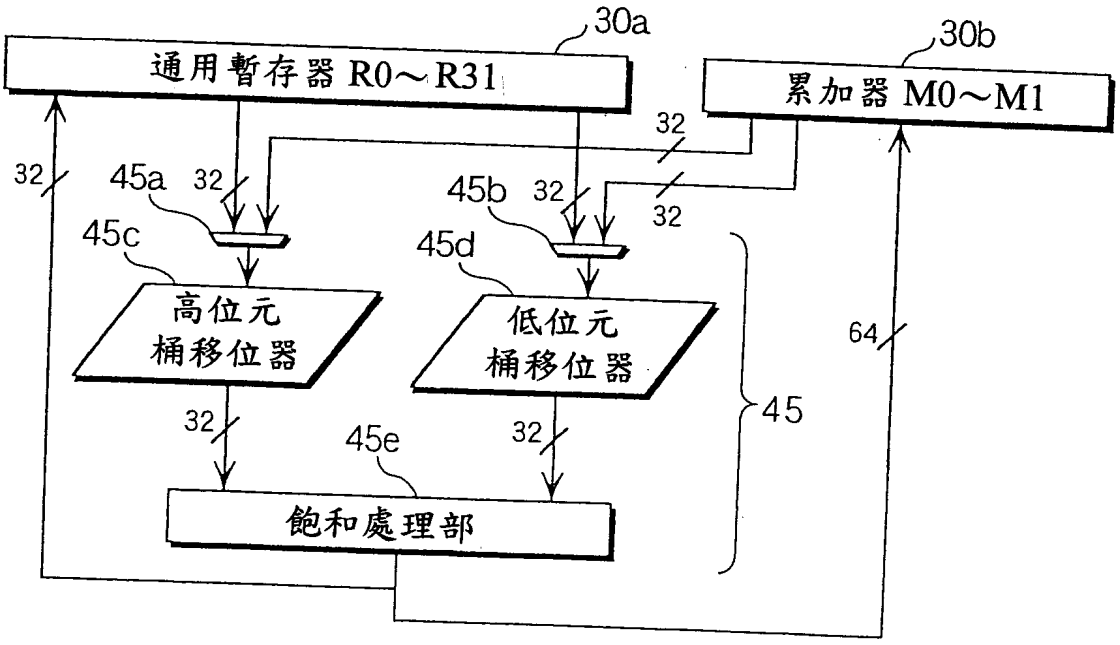
第 1 圖



第 2 圖

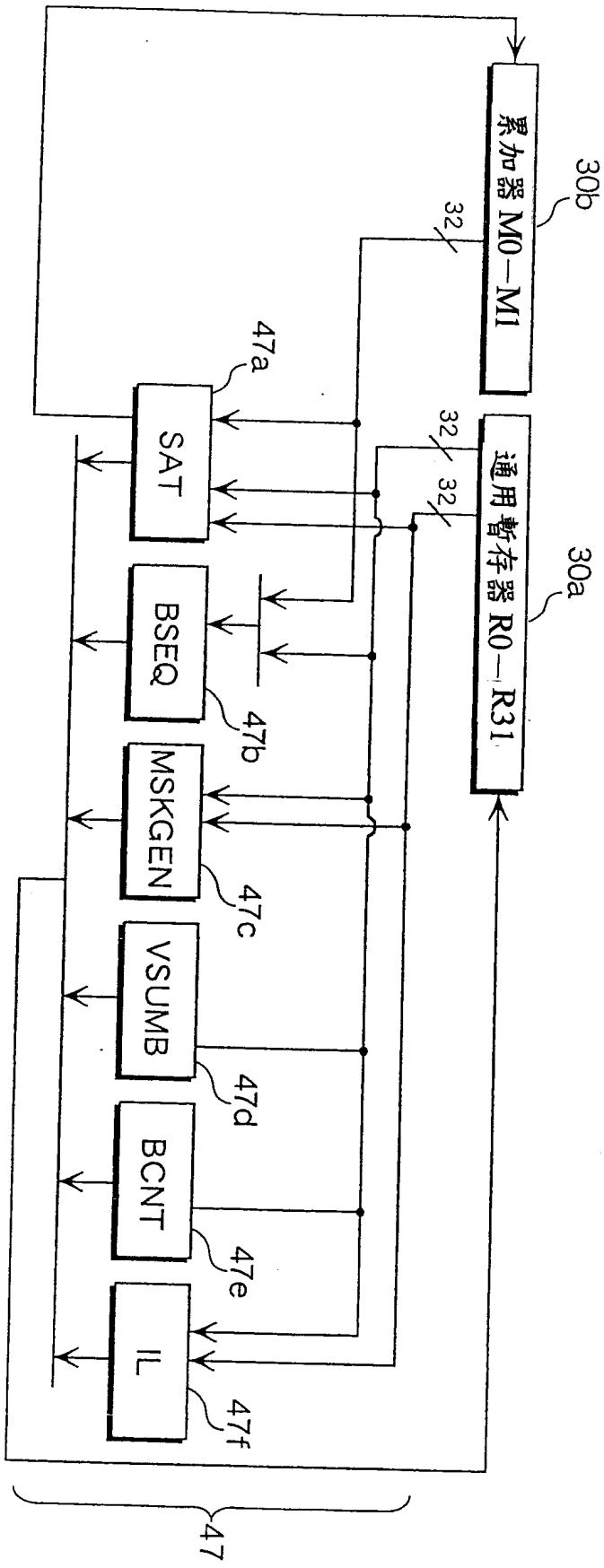

30a
通用暫存器 R0～R31
32
32
32
41a
ALU 部
41b
飽和處理部
41(42,43)
41c
旗標部

第 3 圖

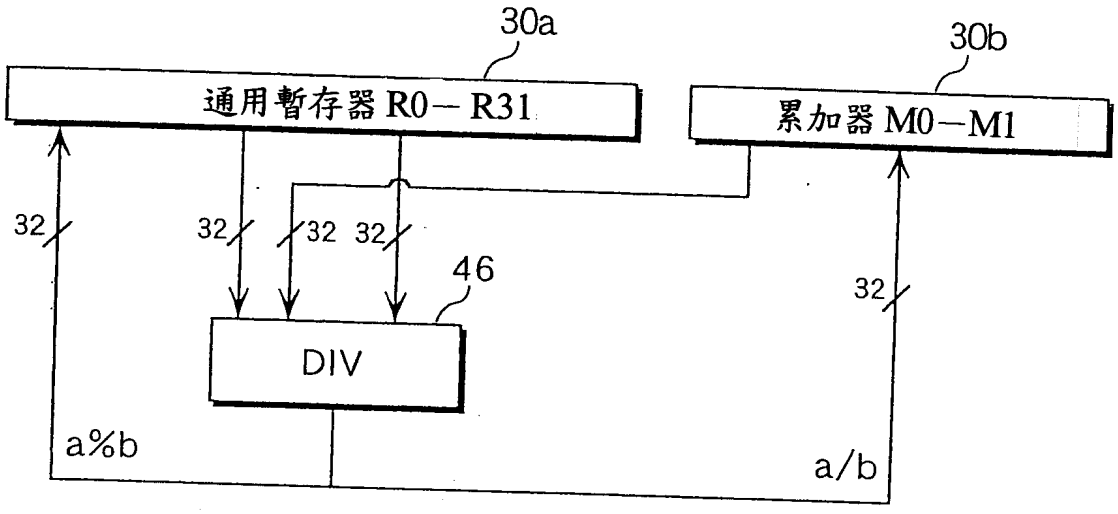

30a
通用暫存器 R0～R31
30b
累加器 M0～M1
32
32
45a
32
45b
32
32
45c
45d
高位元
桶移位器
低位元
桶移位器
64
45
32
45e
32
飽和處理部

200945190

第 4 圖

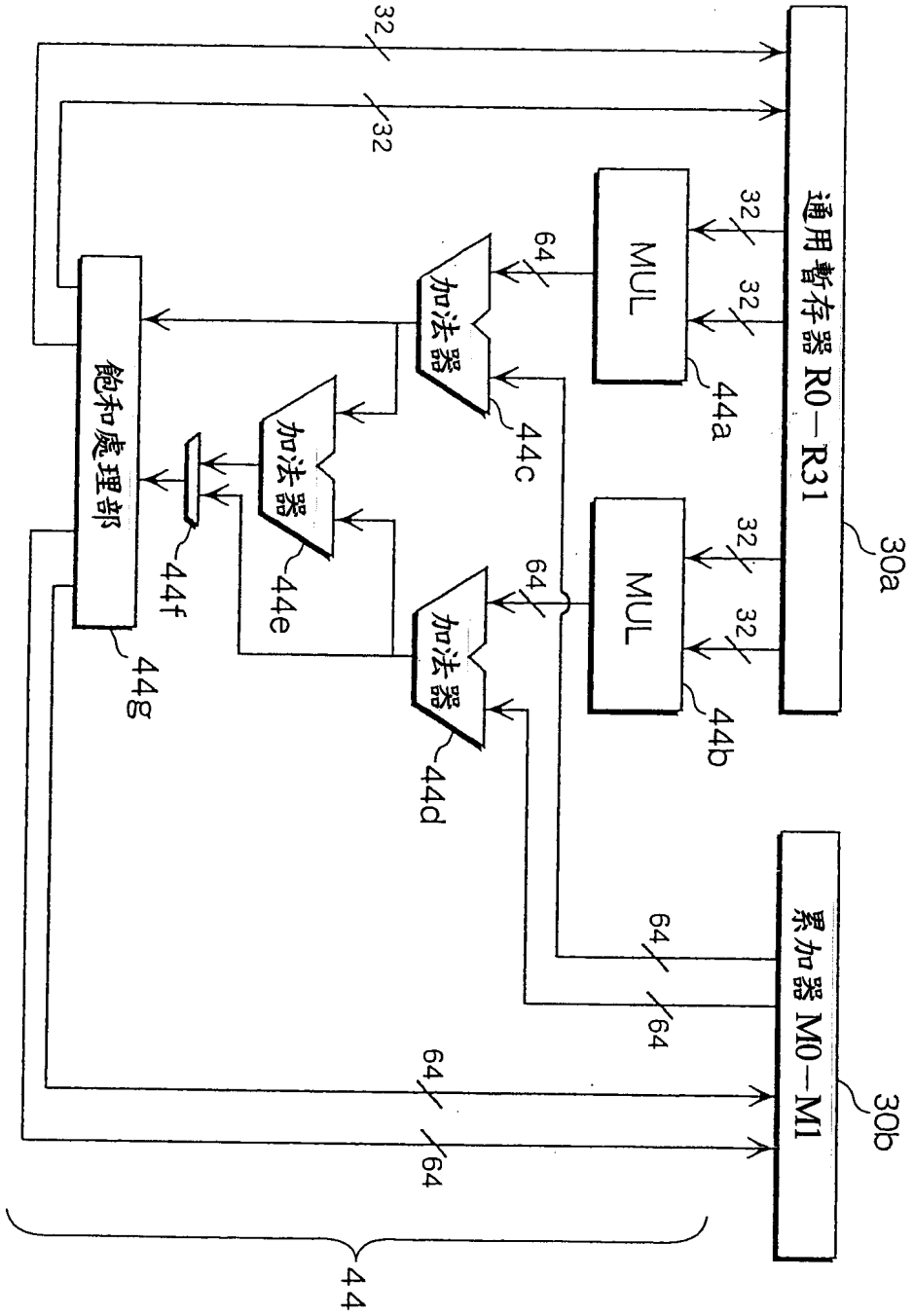


第 5 圖



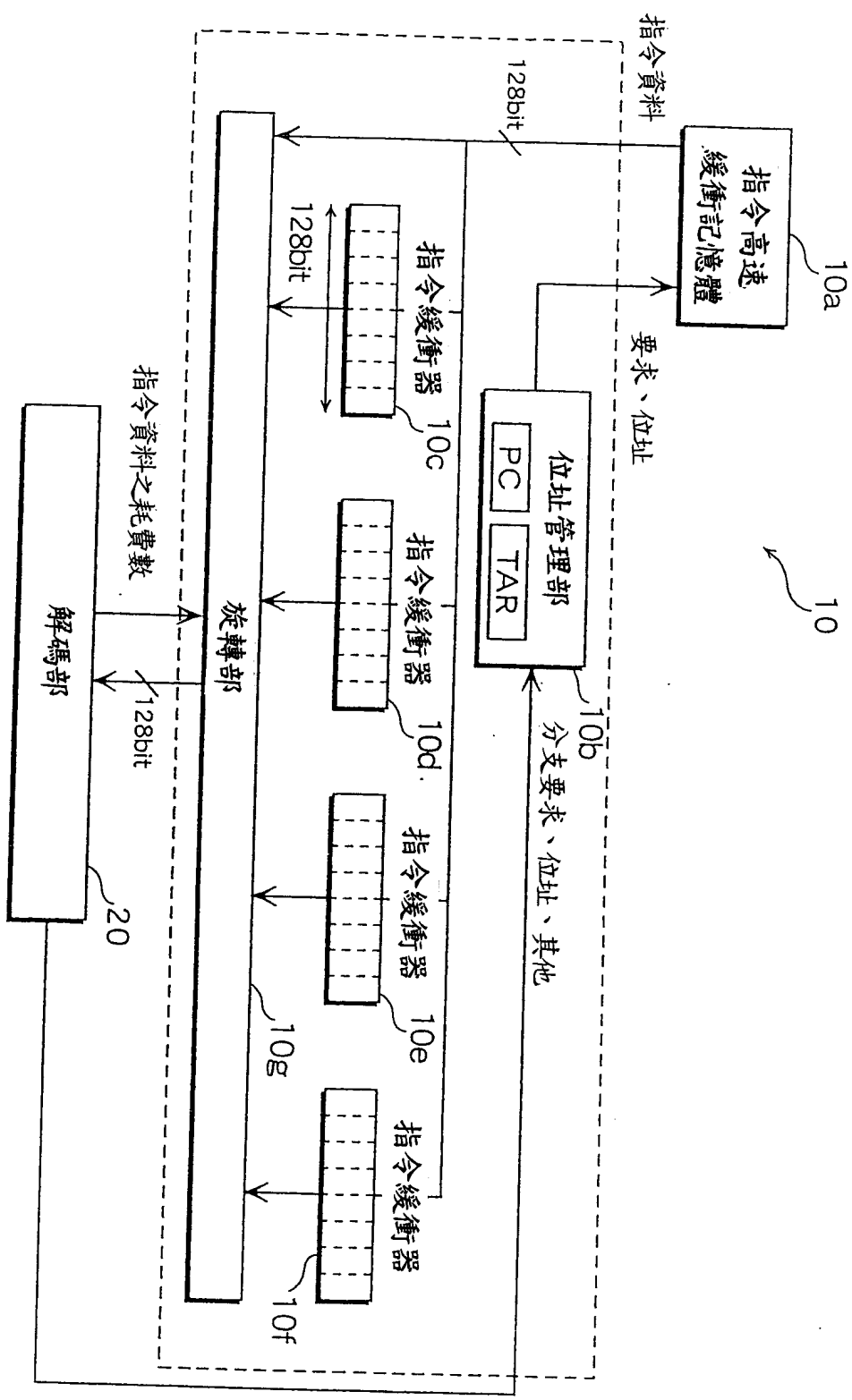
200945190

第6圖

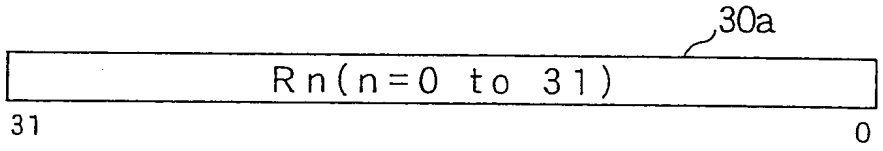


200945190

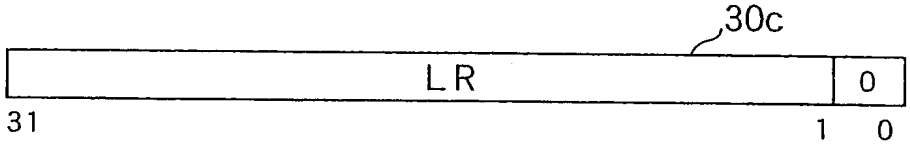
第 7 圖



第 8 圖



第 9 圖



第 10 圖

第 11 圖

31

位元	31	30	29	28	27	26	25	24
位元名稱	保留	SWE	FXP	保留	IH	EH	PL	
位元	23	22	21	20	19	18	17	16
位元名稱	LPIE3	LPIE2	LPIE1	LPIE0	保留	保留	AEE	IE
位元	15	14	13	12	11	10	9	8
位元名稱	保留							
位元	7	6	5	4	3	2	1	0
位元名稱	IM[7:0]							

200945190

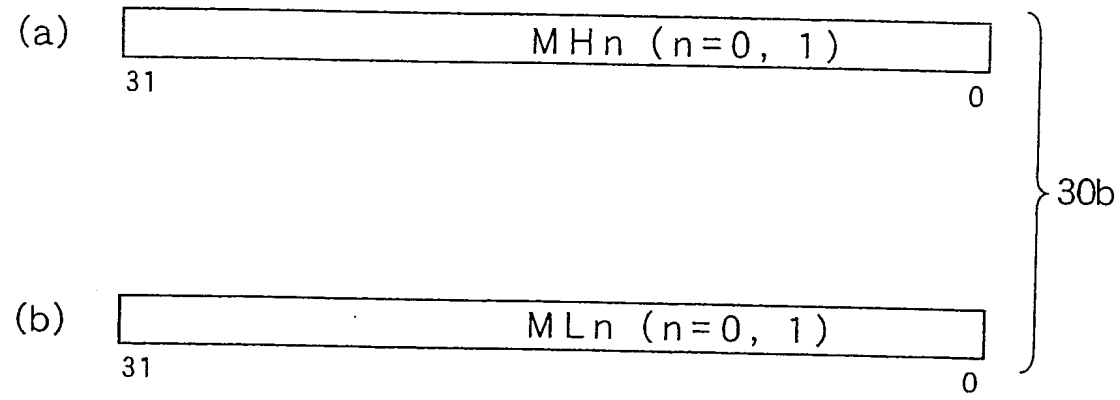
第 12 圖

32

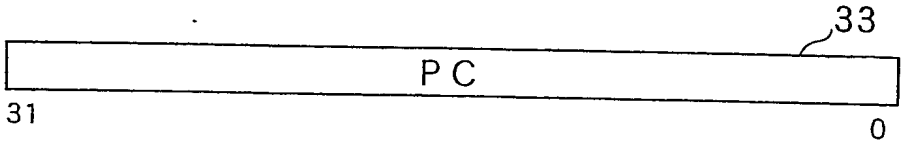
位元	31	30	29	28	27	26	25	24
位元名稱	ALN		保留	BPO				
位元	23	22	21	20	19	18	17	16
位元名稱	保留				VC3	VC2	VC1	VC0
位元	15	14	13	12	11	10	9	8
位元名稱	保留						OVS	CAS
位元	7	6	5	4	3	2	1	0
位元名稱	C7	C6	C5	C4	C3	C2	C1	C0

200945190

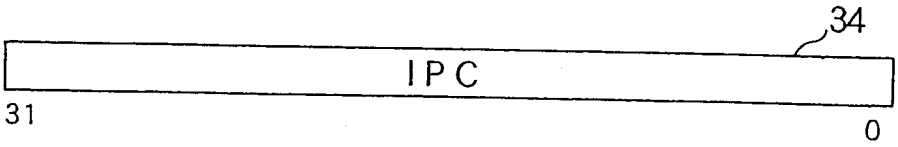
第 13 圖



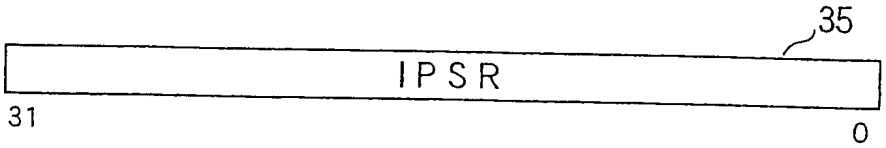
第 14 圖



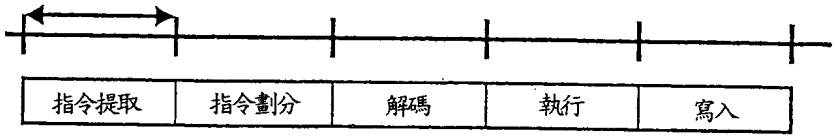
第 15 圖



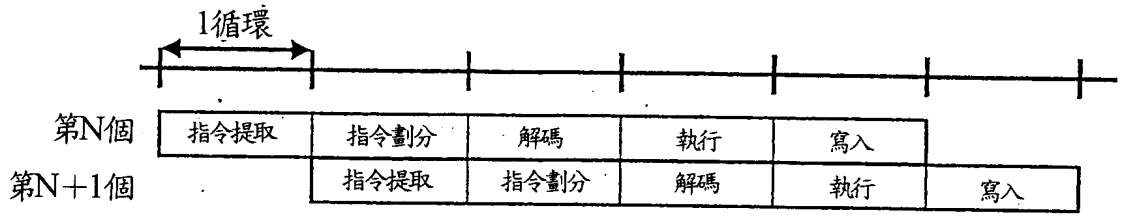
第 16 圖



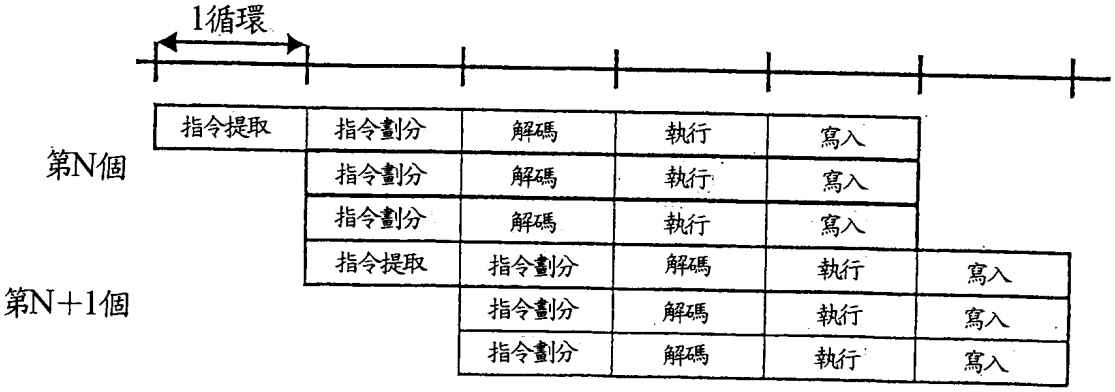
第 17 圖



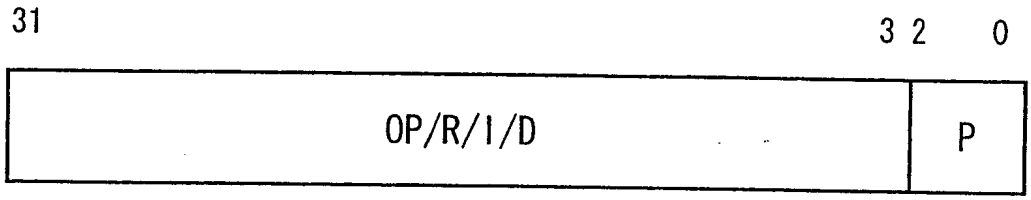
第 18 圖



第 19 圖



第 20 圖



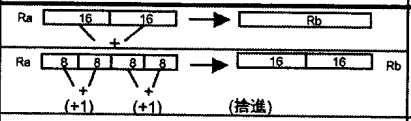
第 21 圖

種類	SIMD	尺寸	指令	運算元	CFR	PSR	典型動作	運算器	31 16
ALU add 系	SINGLE	字	add	Rc,Ra,Rb Rb,Ra,i12s SP,i19s				A	32
				Ra2,Rb2 Rc3,Ra3,Rb3 Ra2,i05s SP,i11s					16
			addu	Rb,GP,i16u Rb,SP,i16u					32
				Ra3,SP,i08u					16
			addc	Rc,Ra,Rb	W:cas,c0:c1		附進位之加法		32
			addvw	Rc,Ra,Rb	W:ovs		附溢位之加法		
			adds	Rc,Ra,Rb			Ra + Rb → >>1 Rc		
			addsr	Rc,Ra,Rb			Ra + Rb + 1 → >>1 Rc		
			s1add	Rc,Ra,Rb			Ra + >>1 Rb → Rc (>>2)		16
				Rc3,Ra3,Rb3					
			s2add	Rc,Ra,Rb					32
				Rc3,Ra3,Rb3					
			addmsk	Rc,Ra,Rb	R:BP0		Ra, Rb, CFR.BP → Rc		16
			addarvw	Rc,Ra,Rb					32
		半字	faddvh	Rc,Ra,Rb	W:ovs				16
	SIMD	半字	vaddh	Rc,Ra,Rb					32
			vaddhvh	Rc,Ra,Rb	W:ovs		Ra, Rb 16 16 → Rc 16 16		
			vsaddh	Rb,Ra,i08s			16 16 +立即值 +立即值 → 16 16 Rb		
			vaddsh	Rc,Ra,Rb			Ra, Rb 16 16 → >>1 >>1 Rc 16 16 (+1) (+1) (經捨進)		
			vaddsrh	Rc,Ra,Rb					
			vaddhvc vaddrhvc	Rc,Ra,Rb Rc,Ra,Rb	R:VC				
			vxaddh	Rc,Ra,Rb			Ra, Rb 16 16 → Rc 16 16		
			vxaddhvh	Rc,Ra,Rb	W:ovs				
			vhaddh	Rc,Ra,Rb			Ra, Rb 16 16 → Rc 16 16		
			vhaddhvh	Rc,Ra,Rb	W:ovs				
			vladdh	Rc,Ra,Rb			Ra, Rb 16 16 → Rc 16 16		
			vladdhvh	Rc,Ra,Rb	W:ovs				
		位元組	vaddb	Rc,Ra,Rb			Ra, Rb 8 8 8 8 → Rc 8 8 8 8 (立即值)		
			vsaddb	Rb,Ra,i08s					
			vaddsb	Rc,Ra,Rb			Ra, Rb 8 8 8 8 → >>1 >>1 Rc 8 8 8 8 (+1)(+1)(+1)(+1) (經捨進)		
			vaddsrb	Rc,Ra,Rb					

第 22 圖

種類	SIMD	尺寸	指令	運算元	CFR	PSR	典型動作	運算器	31 16
ALU sub 系	SINGLE	字	sub	Rc,Rb,Ra Rb2,Ra2 Rc3,Rb3,Ra3				A	32 16
			rsub	Rb,Ra,i08s Ra2,Rb2 Ra2,i04s			立即值 - Ra → Rb (Rb2)		32 16
			subc	Rc,Rb,Ra	W:cas,c0:c1		附進位		
			subvw	Rc,Rb,Ra	W:ovs		附溢位		
			subs	Rc,Rb,Ra			Ra - Rb → >>1 Rc		
			submsk	Rc,Rb,Ra	R:BP0		Ra / Rb: CFR.BP0 → Rc		
		半字	fsubvh	Rc,Rb,Ra					
	SIMD	半字	vsubh	Rc,Rb,Ra			Ra / Rb: 16 16 - → 16 16 Rc		
			vsubhvh	Rc,Rb,Ra	W:ovs				
			vsrsubh	Rb,Ra,i08s			立即值 - 立即值 - 16 16 → 16 16 Rb		
			vsubsh	Rc,Rb,Ra			Ra / Rb: 16 16 - → >>1 >>1 16 16 Rc		
			vxsubh	Rc,Rb,Ra			Ra / Rb: 16 16 - → 16 16 Rc		
			vxsubhvh	Rc,Rb,Ra	W:ovs				
			vhsubh	Rc,Rb,Ra			Ra / Rb: 16 16 - → 16 16 Rc		
			vhsubhvh	Rc,Rb,Ra	W:ovs				
			vlsubh	Rc,Rb,Ra			Ra / Rb: 16 16 - → 16 16 Rc		
			vlsubhvh	Rc,Rb,Ra	W:ovs				
		位元組	vsubb	Rc,Rb,Ra			(立即值) Ra / Rb: 8 8 8 8 - → 8 8 8 8 Rc		
			vsrsubb	Rb,Ra,i08s					
			vasubb	Rc,Rb,Ra	R:VC				

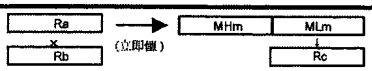
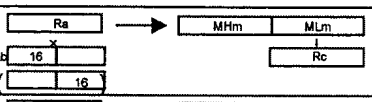
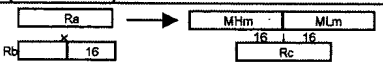
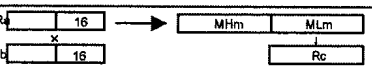
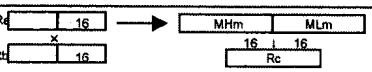
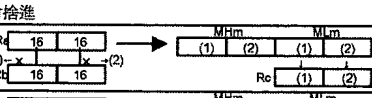
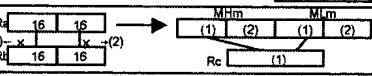
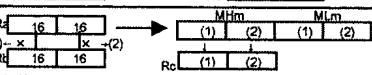
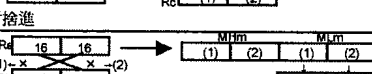
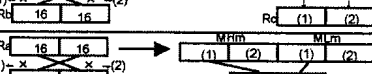
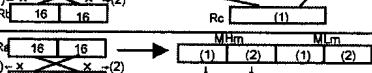
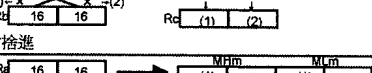
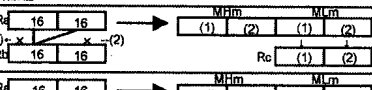
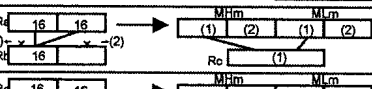
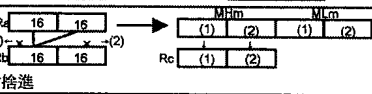
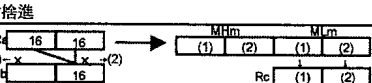
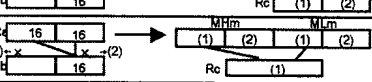
第 23 圖

種類	SIMD	尺寸	指令	運算元	CFR	PSR	典型動作	運算器	31 16
ALU logic 系	SINGLE	字	and	Rc,Ra,Rb Rb,Ra,i08u Ra2,Rb2			邏輯積	A	32 16 32 16 32 16
			andn	Rc,Ra,Rb Rb,Ra,i08u Ra2,Rb2					
			or	Rc,Ra,Rb Rb,Ra,i08u Ra2,Rb2			邏輯和		
			xor	Rc,Ra,Rb Rb,Ra,i08u Ra2,Rb2			互斥邏輯和		
ALU mov 系	SINGLE	字	mov	Rb,Reg32 Reg32,Rb Rb2,Reg16 Reg16,Rb2 Ra2,Rb Ra,i16s Ra2,i08s			Reg32 = TAR LR SVR PSR CFR MH0 MH1 ML0 ML1 EPSR IPC IPSR PC EPC PSR0 PSR1 PSR2 PSR3 CFR0 CFR1 CFR2 CFR3 Reg16 = TAR LR MH0 MH1	A	32 16 32 16 32 16 32
			movp	Rc:Rc+1,Ra,Rb			Rc<-Ra; Rc+1<-Rb;		
			movcf	Ck,Cj,Cm,Cn			Ci<-Cj; Cm<-Cn;		
			mvclovs	Cm:Cm+1	W:ovs		Cm:Cm+1<-CFR.OVS:-CFR.OVS; CFR.OVS<-0;		
			mvclcas	Cm:Cm+1	W:cas		Cm:Cm+1<-CFR.CAS:-CFR.CAS; CFR.CAS<-0;		
			sethi	Ra,i16s					
ALU max min 系	SINGLE	字	max	Rc,Ra,Rb	W:c0:c1		Rc <- max(Ra,Rb)	A	32
			min	Rc,Ra,Rb	W:c0:c1		Rc <- min(Ra,Rb)		
	SIMD	半字	vmaxh	Rc,Ra,Rb					
			vminh	Rc,Ra,Rb					
		位元組	vmaxb	Rc,Ra,Rb					
			vminb	Rc,Ra,Rb					
ALU abs 系	SINGLE		abs	Rb,Ra			絕對值	A	32
			absvw	Rb,Ra	W:ovs		附溢位		
			fabsvh	Rb,Ra	W:ovs				
	SIMD	半字	vabshvh	Rb,Ra	W:ovs				
ALU neg 系	SIN GLE	字	negvw	Rb,Ra	W:ovs			A	32
		半字	fnegvh	Rb,Ra	W:ovs				
	SIMD		vneghvh	Rb,Ra	W:ovs				
ALU sum 系	SIMD	半字	vsumh	Rb,Ra				A	32
			vsumh2	Rb,Ra					
			vsumrh2	Rb,Ra					
		位元組	vabssumb	Rc,Ra,Rb				C	
ALU 其他	SIN GLE		fmdvh	Rb,Ra	W:ovs		捨進	C	32
	SI MD		vfmdvh	Rb,Mn	W:ovs				
			vsel	Rc,Ra,Rb	R:VC			A	
			vsgnh	Rb,Ra				C	

第 24 圖

種類	SIMD	尺寸	指令	運算元	CFR	PSR	典型動作	運算器	31 16
CMP	SINGLE		cmpCCn	Cm,Ra,Rb,Cn Cm,Ra,i05s,Cn Cm:Cm+1,Ra,Rb,Cn Cm:Cm+1,Ra,i05s,Cn	W:CF		CC = eq, ne, gt, ge, gtu, geu, le, lt, leu, leu Cm <- result & Cn; (Cm+1 <- ~result & Cn;)	A	32
			cmpCCa	Cm:Cm+1,Ra,Rb,Cn Cm:Cm+1,Ra,i05s,Cn	W:CF		Cm <- result & Cn; Cm+1 <- ~(result & Cn);		
			cmpCCo	Cm:Cm+1,Ra,Rb,Cn Cm:Cm+1,Ra,i05s,Cn	W:CF		Cm <- result | Cn; Cm+1 <- ~(result | Cn);		
			cmpCC	C6,Ra2,Rb2 C6,Ra2,i04s	W:CF		CC = eq, ne, gt, ge, le, lt C6 <- result		16
			tstzn	Cm,Ra,Rb,Cn Cm,Ra,i05u,Cn Cm:Cm+1,Ra,Rb,Cn Cm:Cm+1,Ra,i05u,Cn	W:CF		Cm <- (Ra & Rb == 0) & Cn; (Cm+1 <- ~(Ra & Rb ==0) & Cn;)		32
			tstza	Cm:Cm+1,Ra,Rb,Cn Cm:Cm+1,Ra,i05u,Cn	W:CF		Cm <- (Ra & Rb == 0) & Cn; Cm+1 <- ~((Ra & Rb == 0) & Cn);		
			tstzo	Cm:Cm+1,Ra,Rb,Cn Cm:Cm+1,Ra,i05u,Cn	W:CF		Cm <- (Ra & Rb == 0) | Cn; Cm+1 <- ~((Ra & Rb == 0) | Cn);		
			tstnn	Cm,Ra,Rb,Cn Cm,Ra,i05u,Cn Cm:Cm+1,Ra,Rb,Cn Cm:Cm+1,Ra,i05u,Cn	W:CF		Cm <- (Ra & Rb != 0) & Cn; (Cm+1 <- ~(Ra & Rb != 0) & Cn;)		
			tstna	Cm:Cm+1,Ra,Rb,Cn Cm:Cm+1,Ra,i05u,Cn	W:CF		Cm <- (Ra & Rb != 0) & Cn; Cm+1 <- ~((Ra & Rb != 0) & Cn);		
			tstno	Cm:Cm+1,Ra,Rb,Cn Cm:Cm+1,Ra,i05u,Cn	W:CF		Cm <- (Ra & Rb != 0) | Cn; Cm+1 <- ~((Ra & Rb != 0) | Cn);		
			tstz	C6,Ra2,Rb2 C6,Ra2,i04u	W:CF		C6 <- (Ra2&Rb2 == 0)		16
			tstn	C6,Ra2,Rb2 C6,Ra2,i04u	W:CF		C6 <- (Ra2&Rb2 != 0)		
	SIMD	半字	vcmpCCh vscmpCCh	Ra,Rb Ra,i05s	W:CF		CC = eq, ne, gt, le, ge, lt		32
		位元組	vcmpCCb vscmpCCb	Ra,Rb Ra,i05s	W:CF		CC = eq, ne, gt, le ge, lt		

第 25 圖

種類	SIMD	尺寸	指令	運算元	CFR	PSR	典型動作	運算器	31 16
mul 系	SINGLE	字 × 字	mul	Mm,Rc,Ra,Rb Mm,Rb,Ra,i08s				X2	32
			mulu	Mm,Rc,Ra,Rb Mm,Rb,Ra,i08s			無符號乘法		
			fmulww	Mm,Rc,Ra,Rb		fxp	固定小數點運算		
		字 × 半字	hmul lmul	Mm,Rc,Ra,Rb Mm,Rc,Ra,Rb				X1	
			fmulhww	Mm,Rc,Ra,Rb		fxp			
		半字 × 半字	fmulhw	Mm,Rc,Ra,Rb		fxp			
			fmulhh fmulhhr	Mm,Rc,Ra,Rb Mm,Rc,Ra,Rb		fxp fxp	 附捨進		
	SIMD	半 字	vmul	Mm,Rc,Ra,Rb				X2	
			vfmulw	Mm,Rc,Ra,Rb		fxp			
			vfmulh vfmulhr	Mm,Rc,Ra,Rb Mm,Rc,Ra,Rb		fxp fxp	 附捨進		
		×	vxmul	Mm,Rc,Ra,Rb					
			vxfmulw	Mm,Rc,Ra,Rb		fxp		X2	
			vxfmulh vxfmulhr	Mm,Rc,Ra,Rb Mm,Rc,Ra,Rb		fxp fxp	 附捨進		
		半 字	vhmul	Mm,Rc,Ra,Rb					
			vhfmulw	Mm,Rc,Ra,Rb		fxp			
			vhfmulh vhfmulhr	Mm,Rc,Ra,Rb Mm,Rc,Ra,Rb		fxp fxp	 附捨進		
			vlmul	Mm,Rc,Ra,Rb					
			vlfmulw	Mm,Rc,Ra,Rb		fxp			
			vlfmulh vlfmulhr	Mm,Rc,Ra,Rb Mm,Rc,Ra,Rb		fxp fxp	 附捨進		
		字 × 半字	vpfmulhww	Mm,Rc:Rc+1,Ra,Rb Mm,Rc:Rc+1,Ra,Rb		fxp fxp			

第 26 圖

種類	SIMD	尺寸	指令	運算元	CFR	PSR	典型動作	運算器	31 16
mac系	SINGLE	字×字	mac	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx			使用mul之積和運算	X2	32
			fmacww	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	使用fmulww之積和運算		
		字×半字	hmac	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx			使用hmul之積和運算	X1	16
			lmac	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx			使用lmul 之積和運算		
			fmachww	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	使用fmulhww之積和運算		32
		半字×半字	fmachw	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	使用fmulhw之積和運算		16
			fmachh	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	使用fmulhh之積和運算		32
			fmachhr	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	附捨進		
	SIMD	半 字 × 半 字	vmac	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx			使用vmul之積和運算	X2	16
			vfmacw	Mm,Rc,Ra,Rb,Mn		fxp	使用vfmulw之積和運算		32
			vxmac	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx			使用vxmul之積和運算		16
			vxfmacw	Mm,Rc,Ra,Rb,Mn		fxp	使用vxfmulw之積和運算		32
			vxfmach	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	使用vxfmulh之積和運算		
			vxfmachr	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	附捨進		
			vhmac	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx			使用vhmul之積和運算		
			vhfmacw	Mm,Rc,Ra,Rb,Mn		fxp	使用vhfmulw之積和運算		
			vhfmach	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	使用vhfmulh之積和運算		
			vhfmachr	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx					
			vlmac	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx			使用vlmul之積和運算		32
			vlfmacw	Mm,Rc,Ra,Rb,Mn		fxp	使用vlfmulw之積和運算		
			vfmach	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	使用vfmulh之積和運算		
			vfmachr	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	附捨進		
			vlfmach	Mm,Rc,Ra,Rb,Mn Mm,Rc,Ra,Rb,Mn		fxp	使用vlfmulh之積和運算		
			vlfmachr	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	附捨進		
		字×半字	vpfmachww	Mm,Rc:Rc+1,Ra,Rb,Mn		fxp			

第 27 圖

種類	SIMD	尺寸	指令	運算元	CFR	PSR	典型動作	運算器	31 16
msu 系	SINGLE	字 × 字	msu	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx			使用mul之積差運算	X2	32
			fmsuww	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	使用fmulww之積差運算		
		字 × 半字	hmsu	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx			使用hmul之積差運算	X1	
			lmsu	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx			使用lmul之積差運算		
			fmsuhww	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	使用fmulhww之積差運算		
		半字 × 半字	fmsuhw	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	使用fmulhw之積差運算		
			fmsuhh	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	使用fmulhh之積差運算		
			fmsuhhr	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	附捨進		
		半 字 × 半 字	vmsu	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx			使用vmul之積差運算	X2	
			vfmsuw	Mm,Rc,Ra,Rb,Mn		fxp	使用vfmul之積差運算		
			vfmsuh	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	使用vfmulh之積差運算		
			vxmsu	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx			使用vxmul之積差運算		
			vxfmsuw	Mm,Rc,Ra,Rb,Mn		fxp	使用vxfmulw之積差運算		
			vxfmsuh	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	使用vxfmulh之積差運算		
			vhmsu	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx			使用vhmul之積差運算		
			vhfmsuw	Mm,Rc,Ra,Rb,Mn		fxp	使用vhfmulw之積差運算		
			vhfmsuh	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	使用vhfmulh之積差運算		
			vlmsu	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx			使用vlmul之積差運算		
			vlfmsuw	Mm,Rc,Ra,Rb,Mn		fxp	使用vlfmulw之積差運算		
			vlfmsuh	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	使用vlfmulh之積差運算		

第 28 圖

種類	SIMD	尺寸		指令	運算元	CFR	PSR	典型動作	運算器	31 16
MEM ld 系	SINGLE	字		ld	Rb,(Ra,d10u) Rb,(GP,d13u) Rb,(SP,d13u) Rb,(Ra+)i10s			暫存器 32 ← 記憶體 32	M	32
					Rb2,(Ra2) Rb2,(Ra2,d05u) Rb2,(GP,d06u) Rb2,(SP,d06u) Rb2,(Ra2+)					16
		半字		ldh	Rb,(Ra,d09u) Rb,(GP,d12u) Rb,(SP,d12u) Rb,(Ra+)i09s			32 ← 16		32
					Rb2,(Ra2) Rb3,(Ra3,d04u) Rb2,(GP,d05u) Rb2,(SP,d05u) Rb2,(Ra2+)					16
				ldhu	Rb,(Ra,d09u) Rb,(GP,d12u) Rb,(SP,d12u) Rb,(Ra+)i09s					32
		位元組		ldb	Rb,(Ra,d08u) Rb,(GP,d11u) Rb,(SP,d11u) Rb,(Ra+)i08s			暫存器 32 ← 記憶體 8		
				ldbu	Rb,(Ra,d08u) Rb,(GP,d11u) Rb,(SP,d11u) Rb,(Ra+)i08s					
		位元組 -> 半字		ldbh ldbuh	Rb,(Ra+)i07s Rb,(Ra+)i07s			16 16 ← 8 8		
		配對	字	ldp	Rb:Rb+1,(Ra,d11u) LR:SVR,(Ra,d11u) TAR:UDR,(Ra,d11u) Rb:Rb+1,(GP,d14u) LR:SVR,(GP,d14u) TAR:UDR,(GP,d14u) Rb:Rb+1,(SP,d14u) LR:SVR,(SP,d14u) TAR:UDR,(SP,d14u) Rb:Rb+1,(Ra+)i11s			32/32 ← 32/32		
					Rb:Rb+1,(SP,d07u) LR:SVR,(SP,d07u) Rb2:Re2,(Ra2+)					16
			半字	ldhp	Rb:Rb+1,(Ra,d10u) Rb:Rb+1,(Ra+)i10s			32/32 ← 16 16		32
					Rb2:Re2,(Ra2+)					16
			位元組	ldbp	Rb:Rb+1,(Ra,d09u) Rb:Rb+1,(Ra+)i09s			32/32 ← 8 8		32
			位元組 -> 半字	ldbhp ldbuhp	Rb:Rb+1,(Ra+)i07s Rb:Rb+1,(Ra+)i07s			16 16 / 16 16 ← 8 8 8 8		

第 29 圖

種類	SIMD	尺寸		指令	運算元	CFR	PSR	典型動作	運算器	31 16
MEM store 系	SINGLE	字		st	(Ra,d10u),Rb (GP,d13u),Rb (SP,d13u),Rb (Ra+)i10s,Rb			暫存器 32 → 記憶體 32		32
					(Ra2),Rb2 (Ra2,d05u),Rb2 (GP,d06u),Rb2 (SP,d06u),Rb2 (Ra2+),Rb2					16
		半字		sth	(Ra,d09u),Rb (GP,d12u),Rb (SP,d12u),Rb (Ra+)i09s,Rb			16 → 16	M	32
					(Ra2),Rb2 (Ra2,d04u),Rb2 (GP,d05u),Rb2 (SP,d05u),Rb2 (Ra2+),Rb2					16
		位元組		stb	(Ra,d08u),Rb (GP,d11u),Rb (SP,d11u),Rb (Ra+)i08s,Rb			8 → 8		32
		位元組 -> 半字		stbh	(Ra+)i07s,Rb			8 8 → 16		
		配對	字	stp	(Ra,d11u),Rb:Rb+1 (Ra,d11u),LR:SVR (Ra,d11u),TAR:UDR (GP,d14u),Rb:Rb+1 (GP,d14u),LR:SVR (GP,d14u),TAR:UDR (SP,d14u),Rb:Rb+1 (SP,d14u),LR:SVR (SP,d14u),TAR:UDR (Ra+)i11s,Rb:Rb+1			32 32 → 32 32		
					(SP,d07u),Rb:Re (SP,d07u),LR:SVR (Ra2+),Rb2:Re2					16
			半字	sthp	(Ra,d10u),Rb:Rb+1 (Ra+)i10s,Rb:Rb+1			16 16 → 32		32
					(Ra2+),Rb2:Re2					16
			位元組	stbp	(Ra,d09u),Rb:Rb+1 (Ra+)i09s,Rb:Rb+1			8 8 → 16		32
			位元組 -> 半字	stbhp	(Ra+)i07s,Rb:Rb+1			8 8 8 8 → 32		

第 30 圖

種類	SIMD	尺寸	指令	運算元	CFR	PSR	典型動作	運算器	31 16
BRA			setlr	d09s C5,d09s			設定LR 將由LR提取之指令儲存於分支用緩衝器	B	32
			settar	d09s C6,d09s C6,C2:C4,d09s C6,Cm,d09s C6,C4,d09s	W:c6 W:c2:c4,c6 W:c6,cm W:c6		設定TAR 將由TAR提取之指令儲存於分支用緩衝器		16
			setbb	LR TAR			將由LR提取之指令儲存於分支用緩衝器 將由TAR提取之指令儲存於分支用緩衝器		32
			jloop	C5,LR,Ra,i08s C6,TAR,Ra,i08s C6,C2:C4,TAR,Ra,i08 C6,Cm,TAR,Ra,i08s C6,TAR,Ra2 C6,C2:C4,TAR,Ra2 C6,Cm,TAR,Ra2	W:c5 W:c6 W:c2:c4,c6 W:c6,cm W:c6 W:c2:c4,c6 W:c6		只有述詞 [c5] 只有述詞 [c6]		32
			jmp	TAR LR					16
			jmpl	TAR LR	R:CF				
			jmpf	TAR LR Cm,TAR C6,C2:C4,TAR					
			jmpr	LR					
			br	d20s d09s			只有述詞 [c6][c7]		32 16
			brl	d20s d09s	R:CF				32 16
			rti			W:PSR R:eh			

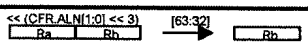
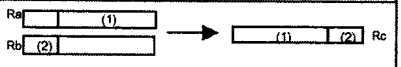
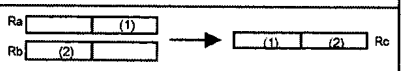
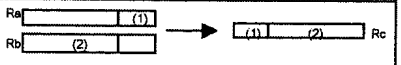
第 31 圖

種類	SIMD	尺寸	指令	運算元	CFR	PSR	典型動作	運算器	31 16
BS asl 系	SINGLE	字	asl	Rc,Ra,Rb Rb,Ra,i05u Ra2,i04u			左移位 <<Rb[0:4] Ra → Rc	S1	32 16
			faslvw	Rc,Ra,Rb Rb,Ra,i05u Rc,Ra,Rb Rb,Ra,i05u	W:ovs		<<Rb[0:4] 裡飽和 Ra → Rc		
		配對字	aslp	Mm,Ra,Mn,Rb Mm,Rb,Mn,i06u Mm,Rc,MHn,Ra,Rb Mm,Rb,MHn,Ra,i06u			<<Ra[0:4] MHm MLm → MHm MLm ↓ Rb	S2	
			faslpvw	Mm,Ra,Mn,Rb Mm,Rb,Mn,i06u	W:ovs		<<Rb[0:4] 裡飽和 MHm MLm → MHm MLm ↓ Rb		
	SIMD	字	vasl	Mm,Ra,Mn,Rb Mm,Rb,Mn,i05u			MHm MLm → MHm MLm (1)<<Ra[0:4] (2)<<Ra[0:4] ↓ Rb	S2	32
			vfaslvw	Mm,Ra,Mn,Rb Mm,Rb,Mn,i05u	W:ovs		MHm MLm → MHm MLm (1)<<Ra[0:4] (2)<<Ra[0:4] ↓ Rb		
		半字	vaslh	Rc,Ra,Rb Rb,Ra,i04u			<<Rb[0:3] <<Rb[0:3] Ra (1) (2) → (1) (2) Rc	S1	
			vfaslvh	Rc,Ra,Rb Rb,Ra,i04u	W:ovs		<<Rb[0:3] <<Rb[0:3] Ra (1) (2) → (1) (2) Rc 裡飽和		
		位元組	vaslb	Rc,Ra,Rb Rb,Ra,i03u			<<Rb[0:2] <<Rb[0:2] Ra (1) (2) (3) (4) → (1) (2) (3) (4) Rc <<Rb[0:2] <<Rb[0:2]		
BS asr 系	SINGLE	字	asr	Rc,Ra,Rb Rb,Ra,i05u Ra2,i04u			算術右移位 >>Rb[0:4] Ra → Rc	S1	32 16
		配對字	asrp	Mm,Ra,Mn,Rb Mm,Rb,Mn,i06u Mm,Rc,MHn,Ra,Rb Mm,Rb,MHn,Ra,i06u			>>Ra[0:4] MHm MLm → MHm MLm ↓ Rb	S2	32
	SIMD	字	vasr	Mm,Ra,Mn,Rb Mm,Rb,Mn,i05u			MHm MLm → MHm MLm (1)>>Ra[0:4] (2)>>Ra[0:4] ↓ Rb		
		半字	vasrh	Rc,Ra,Rb Rb,Ra,i04u			>>Rb[0:3] >>Rb[0:3] Ra (1) (2) → (1) (2) Rc	S1	
		位元組	vasrb	Rc,Ra,Rb Rb,Ra,i03u			>>Rb[0:2] >>Rb[0:2] Ra (1) (2) (3) (4) → (1) (2) (3) (4) Rc >>Rb[0:2] >>Rb[0:2]		

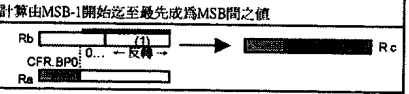
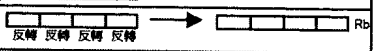
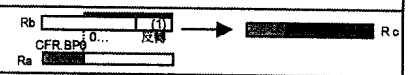
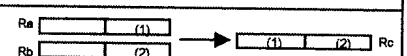
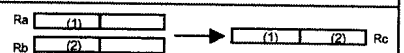
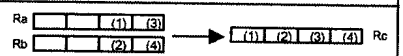
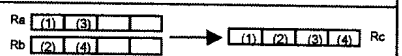
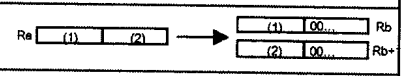
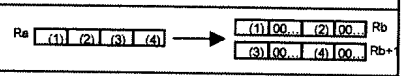
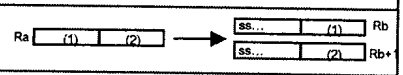
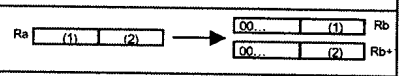
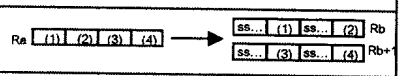
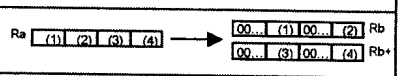
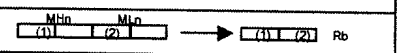
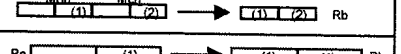
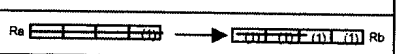
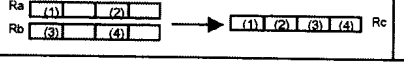
第 32 圖

種類	SIMD	尺寸	指令	運算元	CFR	PSR	典型動作	運算器	31 16
BS lsr 系	SINGLE	字	lsr	Rc,Ra,Rb Rb,Ra,i05u			邏輯右移位 >>Rb[0:4] Ra → Rc	S1	32
		配對字	lsrp	Mm,Ra,Mn,Rb Mm,Rb,Mn,i06u Mm,Rc,MHn,Ra,Rb Mm,Rb,MHn,Ra,i06u			>>>Ra[0:4] MHm MLm → MHm MLm Rb	S2	
	SIMD	字	vlsr	Mm,Ra,Mn,Rb Mm,Rb,Mn,i05u			MHm MLm → MHm MLm (1)>>>Ra[0:4] (2)>>>Ra[0:4] Rb		
		半字	vlsrh	Rc,Ra,Rb Rb,Ra,i04u			>>>Rb[0:3] >>>Rb[0:3] Ra (1) (2) → (1) (2) Rc	S1	
		位元組	vlsrb	Rc,Ra,Rb Rb,Ra,i03u			>>>Rb[0:2] >>>Rb[0:2] Ra (1) (2) (3) (4) → (1) (2) (3) (4) Rc >>>Rb[0:2] >>>Rb[0:2]		
BS rotate 系	SINGLE	字	rol	Rc,Ra,Rb Rb,Ra,i05u			Ra → Rc <<Rb[4:0]	S1	32
	SIMD	半字	vrolh	Rc,Ra,Rb Rb,Ra,i04u					
		位元組	vrolb	Rc,Ra,Rb Rb,Ra,i03u					
BS ext 系	SINGLE	字	extw	Mm,Rb,Ra			Ra → s... Ra Rc	C	32
		半字	exth	Ra2			Ra2 (1) → s... (1) Ra2	S2	16
			exthu	Ra2			Ra2 (1) → 0... (1) Ra2		
		位元組	extb	Ra2			Ra2 (1) → s... (1) Ra2		
			extbu	Ra2			Ra2 (1) → 0... (1) Ra2		
	SIMD	半字	vexth	Mm,Rb,Ra			(1) (2) → s... s... (1) (2) Ra Rb	C	32

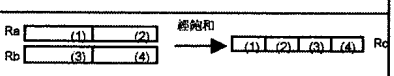
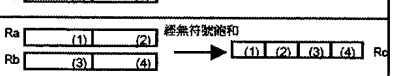
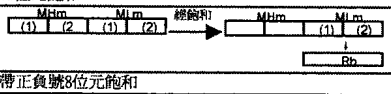
第 33 圖

種類	SIMD	尺寸	指令	運算元	CFR	PSR	典型動作	運算器	31 16
CNV valn 系	S I M D		valn	Rc,Ra,Rb	R:aln[1:0]			C	32
			valn1	Rc,Ra,Rb					
			valn2	Rc,Ra,Rb					
			valn3	Rc,Ra,Rb					
			valnvc1	Rc,Ra,Rb	R:VC0				
			valnvc2	Rc,Ra,Rb	R:VC0				
			valnvc3	Rc,Ra,Rb	R:VC0				
			valnvc4	Rc,Ra,Rb	R:VC0				

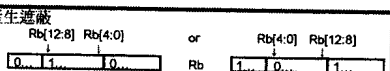
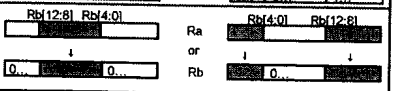
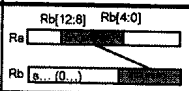
第 34 圖

種類	SIMD	尺寸	指令	運算元	CFR	PSR	典型動作	運算器	31 16
CNV	SINGLE		bcnt1	Rb,Ra			計算1的數目	C	32
			bseq0 bseq1 bseq	Rb,Ra Rb,Ra Rb,Ra			計算由MSB開始迄至最先成爲0間之值 計算由MSB開始迄至最先成爲1間之值 計算由MSB-1開始迄至最先成爲MSB間之值		
			mskbrvh	Rc,Ra,Rb	R:BP0				
			byterev	Rb,Ra					
			mskbrvb	Rc,Ra,Rb	R:BP0				
	SIMD	半字	vintllh	Rc,Ra,Rb					
			vintlhh	Rc,Ra,Rb					
		位元組	vintllb	Rc,Ra,Rb					
			vintlhb	Rc,Ra,Rb					
		半字	vhunpkh	Rb:Rb+1,Ra					
		位元組	vhunpkb	Rb:Rb+1,Ra					
		半字	vlunpkh	Rb:Rb+1,Ra					
			vlunpkhu	Rb:Rb+1,Ra					
		位元組	vlunpkb	Rb:Rb+1,Ra					
			vlunpkbu	Rb:Rb+1,Ra					
		半字	vunpk1	Rb,Mn					
			vunpk2	Rb,Mn					
			vstovh	Rb,Ra					
		位元組	vstovb	Rb,Ra					
			vhpkb	Rc,Ra,Rb					

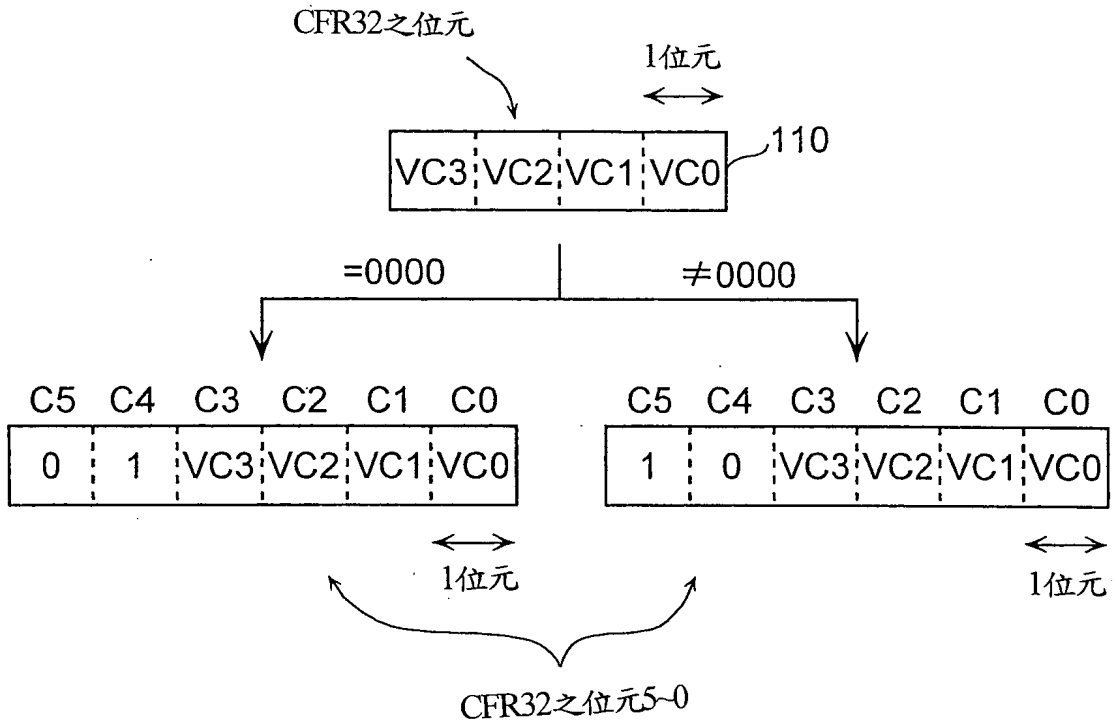
第 35 圖

種類	SIMD	尺寸	指令	運算元	CFR	PSR	典型動作	運算器	31 16
SAT vlpk 系	S I M D	字->半字	vlpkh	Rc,Ra,Rb				C	32
			vlpkhu	Rc,Ra,Rb					
		半字->位元組	vlpkb	Rc,Ra,Rb					
			vlpkbu	Rc,Ra,Rb					
SAT sat 系	S I N G L E	字	satw	Mm,Rb,Mn			字飽和	C	32
			sath	Rb,Ra			半字飽和		
			satb	Rb,Ra			位元組飽和		
			satbu	Rb,Ra			無符號位元組飽和		
			sat9	Rb,Ra			9位元飽和		
			sat12	Rb,Ra			12位元飽和		
	S I M D	半字	vsath	Mm,Rb,Mn					
			vsath8	Rb,Ra			帶正負號8位元飽和		
			vsath8u	Rb,Ra			無符號 8位元飽和		
			vsath9	Rb,Ra			9 位元飽和		
			vsath12	Rb,Ra			12 位元飽和		

第 36 圖

種類	SIMD	尺寸	指令	運算元	CFR	PSR	典型動作	運算器	31 16
MSK			mskgen	Rc,Rb Rb,I05U,i05u			產生遮蔽 	S2	32
			msk	Rc,Ra,Rb Rb,Ra,I05U,i05u					
EXTR			extr extru	Rc,Ra,Rb Rb,Ra,I05U,i05u Rc,Ra,Rb Rb,Ra,I05U,i05u			 經帶正負號延伸 (不經正負號延伸)	S2	32
DIV			div divu	MHm,Rc,MHn,Ra,Rb MHm,Rc,MHn,Ra,Rb	W:ovs		除法	DIV	32
ETC			piNl			W:ih,ie,fie,pl R:PSR	軟體中斷 N=0～7	B	32
			piN			W:ih,ie,pl R:PSR	軟體中斷 N=0～7		16
			scN			W:ih,ie,pl R:PSR	系統呼叫 N=0～7		
			ldstb	Rb,(Ra)			載入匯流排鎖定	M	32
			rd	Rb,(Ra) Rb,(d11u) Rb2,(Ra2)		R;eee	外部暫存器讀入		16
			wt	(Ra),Rb (d11u),Rb (Ra2),Rb2		R:eee	外部暫存器寫入		32
			dpref	(Ra,d11u)			預擷取		16
			dbgmN	i18u			N=0～3	DBGM	32
			vcchk		W:CF R:VC		VC旗標檢查	B	
			vmpsw vmpsw	LR			VMP切換		
			vmpintd1 vmpintd2 vmpintd3			W:ie W:ie	禁止VMP切換		
			vmpinte1 vmpinte2 vmpinte3			W:ie	允許VMP切換		
			nop				無運算	A	16

第 37 圖


CFR32之位元
1位元
VC3 VC2 VC1 VC0
110
=0000
≠0000
C5 C4 C3 C2 C1 C0
0 1 VC3 VC2 VC1 VC0
C5 C4 C3 C2 C1 C0
1 0 VC3 VC2 VC1 VC0
1位元
1位元
CFR32之位元5~0

第 38 圖

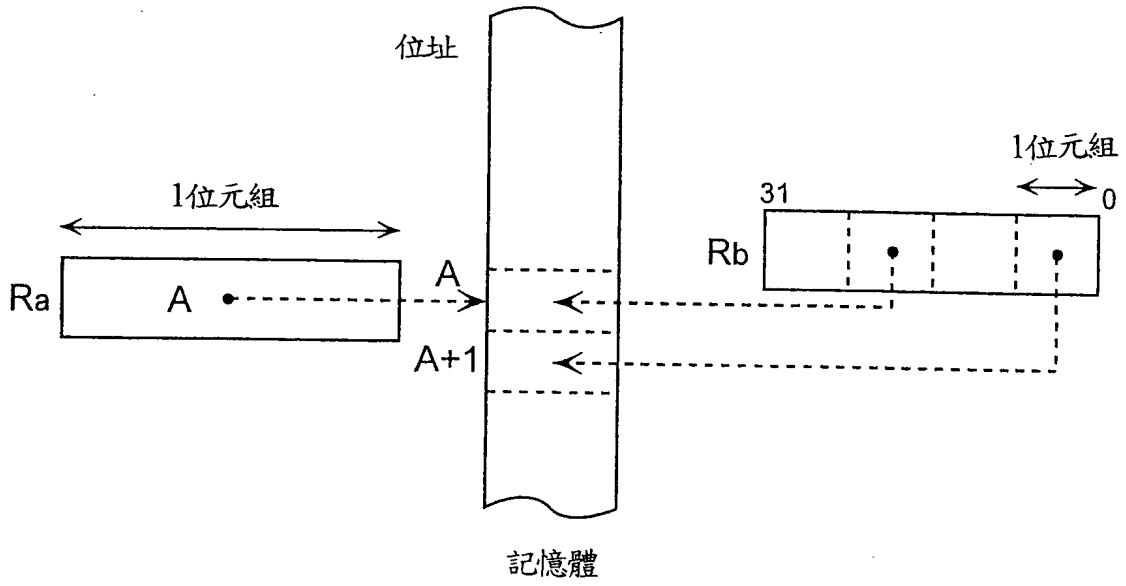
vcchk

動作	if (VC3:VC2:VC1:VC0==0b0000) {
	C4:C5 <- 1:0;
	} else {
	C4:C5 <- 0:1;
	}
	C3:C2:C1:C0 <- VC3:VC2:VC1:VC0;
	
	判定VC0～VC3是否全爲0，並將判定結果輸出於 C4 及 C5。將 VC0～VC3 轉送至 C0～C3。

組合程式助憶	格式	影響旗標	受影響之旗標
Vcchk	16bit	C0,C1,C2,C3,C4,C5	VC0,VC1,VC2,VC3

備考

第 39 圖



第 40 圖

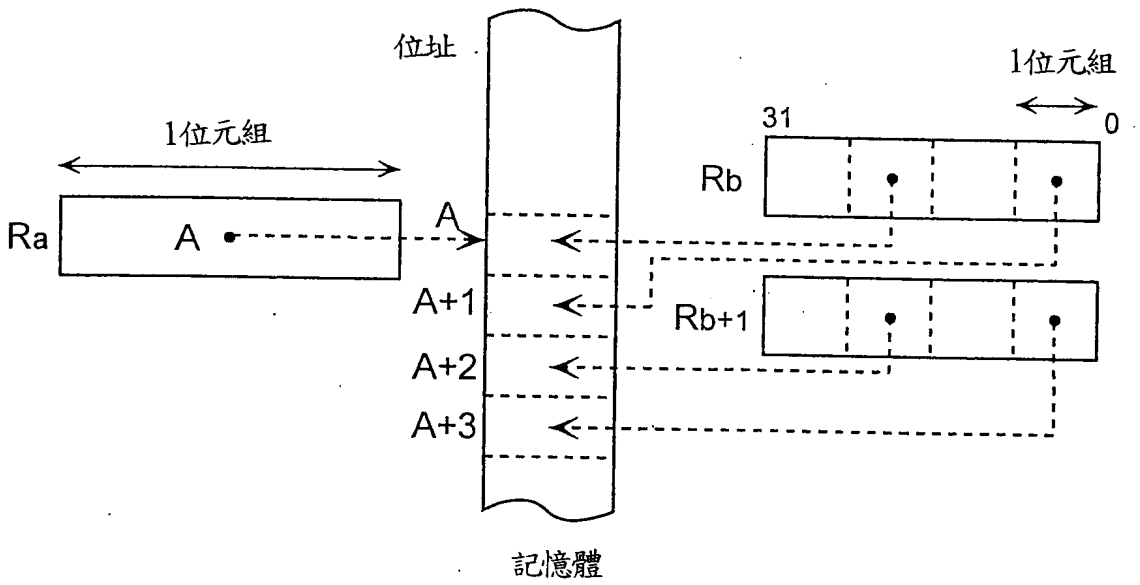
stbh (Ra),Rb

動作	**=> stbh (Ra+)0,Rb** **H(Ra) <- Rb[23:16]:Rb[7:0];** 在 **Ra** 所示之位址上，儲存業已存放於 **Rb** 之兩個位元組資料。而兩個位元組資料係存放於**Rb**之**16~23**位元及**0~7**位元。

組合程式助憶	格式	影響旗標	受影響之旗標
stbh (Ra),Rb	32bit 同義字	–	PSR.AEE

備考
當 Ra 未調整爲 2 位元組時，則產生一遺漏調整例外。

第 41 圖



第 42 圖

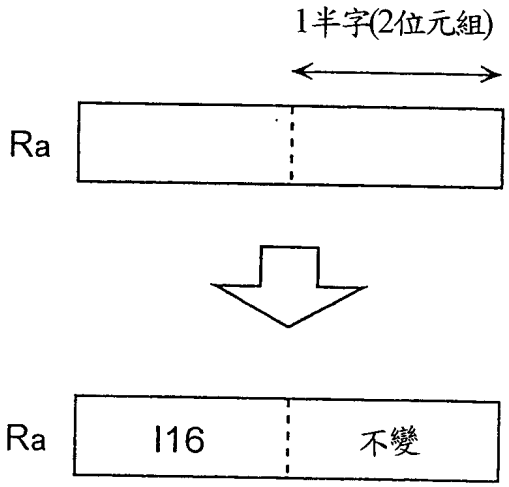
stbhp (Ra),Rb:Rb+1

動作	=> stbhp (Ra+)0,Rb:Rb+1 W(Ra) <- Rb[23:16]:Rb[7:0]:Rb+1[23:16]:Rb+1[7:0]; 在 Ra 所示之位址上，儲存業已存放於 Rb及Rb+1 共四個位元組資料。而四個位元組資料係存放於Rb及Rb+1之16~23位元及0~7位元。

組合程式助憶	格式	影響旗標	受影響之旗標
stbhp (Ra),Rb:Rb+1	32bit 同義字	—	PSR.AEE

備考
當 Ra 未調整爲 4 位元組時，則產生一遺漏調整例外。

第 43 圖

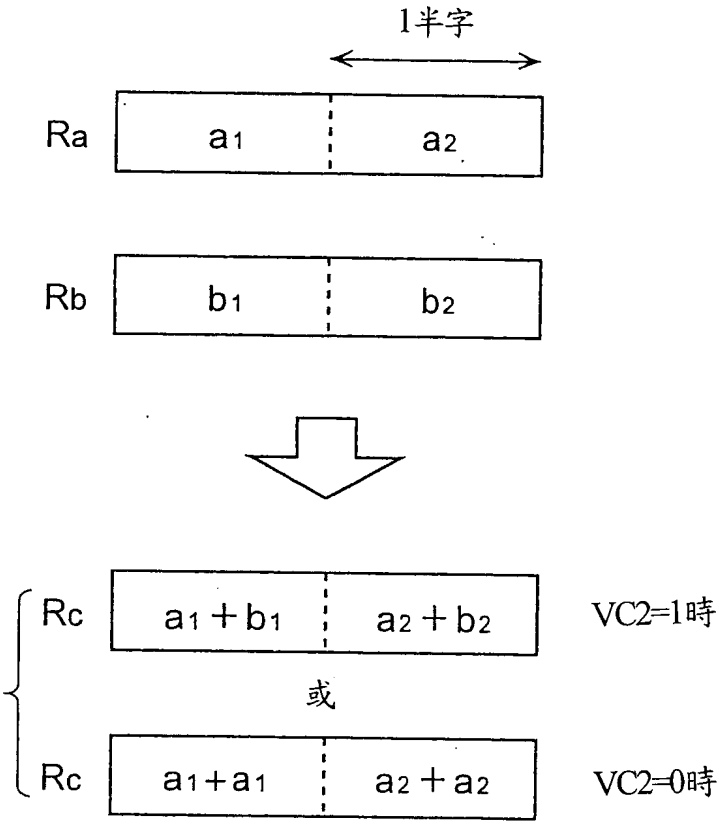


第 44 圖

sethi Ra,I16

動作	**Ra <- (uext(I16) << 16)+uext(Ra[15:0]);** **在Ra之高16位元儲存立即值(I16)。對Ra之低16位元不予影響。**		
組合程式助憶	格式	影響旗標	受影響之旗標
sethi Ra,I16	32bit	—	—
備考			
I16 是一無符號值。			

第 45 圖



第 46 圖

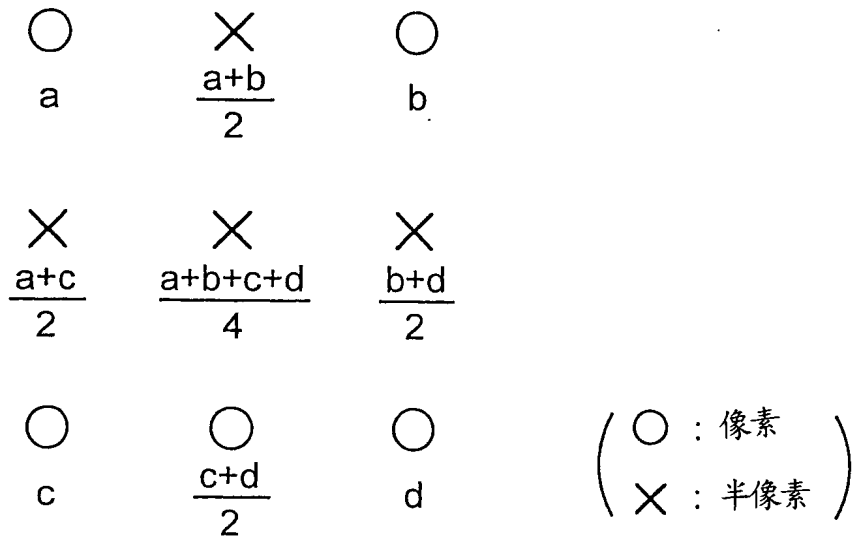
vaddhvc Rc,Ra,Rb

動作	**Rc[31:16] <- Ra[31:16] + Ra[31:16], if (VC2 == 0);** **Rc[31:16] <- Ra[31:16] + Rb[31:16], if (VC2 == 1);** **Rc[15: 0] <- Ra[15: 0] + Ra[15: 0], if (VC2 == 0);** **Rc[15: 0] <- Ra[15: 0] + Rb[15: 0], if (VC2 == 1);** 使用於影像處理之動畫補償。 以半字向量格式處理每一暫存器。 將**Ra**與**Ra**或**Rb**相加。藉**VC2**以決定要與**Ra**或**Rb**相加。

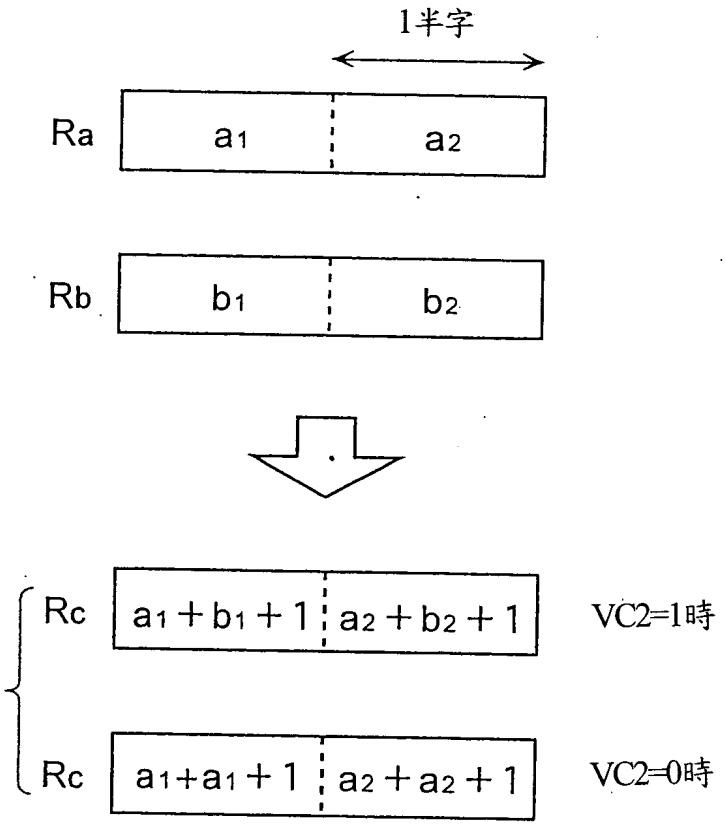
組合程式助憶	格式	影響旗標	受影響之旗標
vaddhvc Rc,Ra,Rb	32bit	–	VC2

備考

第 47 圖



第 48 圖



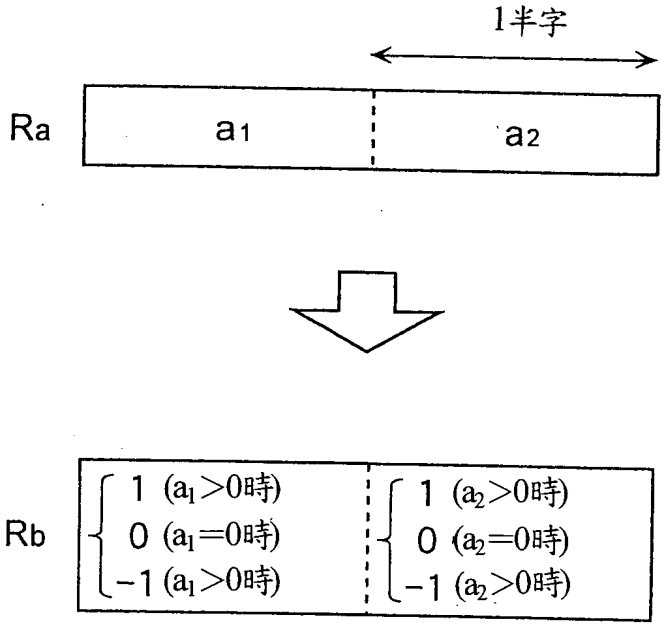
第 49 圖

vaddrhvc Rc,Ra,Rb

動作	**Rc[31:16] <- Ra[31:16] + Ra[31:16] + sext16(1), if (VC2 == 0);** **Rc[31:16] <- Ra[31:16] + Rb[31:16] + sext16(1), if (VC2 == 1);** **Rc[15: 0] <- Ra[15: 0] + Ra[15: 0] + sext16(1), if (VC2 == 0);** **Rc[15: 0] <- Ra[15: 0] + Rb[15: 0] + sext16(1), if (VC2 == 1);** 以半字向量格式處理每一暫存器。 將**Ra**與**Ra**或**Rb**相加，進而爲了捨進而加上**1**。 藉**VC2**以決定要與**Ra**或**Rb**相加。

組合程式助憶	格式	影響旗標	受影響之旗標
vaddrhvc Rc,Ra,Rb	32bit	—	VC2
備考			

第 50 圖



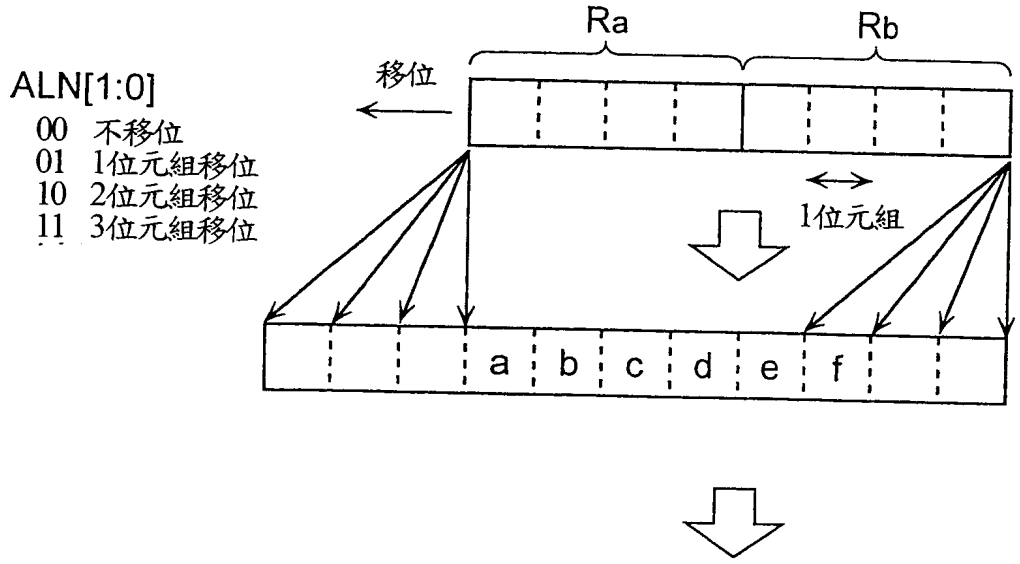
第 51 圖

vsgnh Rb,Ra

動作	**Rb[31:16] <- 0x0001, if (Ra[31:16] > 0);** **Rb[31:16] <- 0x0000, if (Ra[31:16] == 0);** **Rb[31:16] <- 0xffff, if (Ra[31:16] < 0);** **Rb[15: 0] <- 0x0001, if (Ra[15: 0] > 0);** **Rb[15: 0] <- 0x0000, if (Ra[15: 0] == 0);** **Rb[15: 0] <- 0xffff, if (Ra[15: 0] < 0);** 使用於影像處理之逆量子化。 以半字向量格式處理每一暫存器。 **Ra**爲正時，輸出**1**，而**Ra**爲負時則輸出 **-1**， 又爲**0**時，則輸出**0**。

組合程式助憶	格式	影響旗標	受影響之旗標
vsgnh Rb,Ra	32bit	–	–
備考			

第 52 圖



指令	結果	
valnvc1	Rc a a b b	(VC0=0時)
	Rc a b b c	(VC0=1時)
valnvc2	Rc c c d d	(VC0=0時)
	Rc c d d e	(VC0=1時)
valnvc3	Rc a a b b	(VC0=0時)
	Rc a c b d	(VC0=1時)
valnvc4	Rc c c d d	(VC0=0時)
	Rc c e d f	(VC0=1時)

第 53 圖

valnvc1 Rc,Ra,Rb

動作	**tmp [87: 0] <- Ra:Rb << (CFR.ALN[1:0] << 3);** **tmp2[31: 0] <- tmp[63:32];** **Rc <- tmp2[31:24]:tmp2[31:24]:tmp2[23:16]:tmp2[23:16],** **if (VC0 == 0);** **Rc <- tmp2[31:24]:tmp2[23:16]:tmp2[23:16]:tmp2[15: 8],** **if (VC0 == 1);** 使用於影像處理之動畫補償。 按**CFR.ALN**執行位元組調正，藉**VC0**值變更所取出之位元組資料。

組合程式助憶	格式	影響旗標	受影響之旗標
valnvc1 Rc,Ra,Rb	32bit	—	CFR.ALN,VC0
備考			

第 54 圖

valnvc2 Rc,Ra,Rb

動作	**tmp [87: 0] <- Ra:Rb << (CFR.ALN[1:0] << 3);** **tmp2[31: 0] <- tmp[47:16];** **Rc <- tmp2[31:24]:tmp2[31:24]:tmp2[23:16]:tmp2[23:16],** **if (VC0 == 0);** **Rc <- tmp2[31:24]:tmp2[23:16]:tmp2[23:16]:tmp2[15: 8],** **if (VC0 == 1);** 使用於影像處理之動畫補償。 按**CFR.ALN**執行位元組調正，並藉**VC0**值變更所取出之位元組資料。

組合程式助憶	格式	影響旗標	受影響之旗標
valnvc2 Rc,Ra,Rb	32bit	–	CFR.ALN,VC0
備考			

第 55 圖

valnvc3 Rc,Ra,Rb

動作	**tmp [87: 0] <- Ra:Rb << (CFR.ALN[1:0] << 3);** **tmp2[31: 0] <- tmp[63:32];** **Rc <- tmp2[31:24]:tmp2[31:24]:tmp2[23:16]:tmp2[23:16],** **if (VC0 == 0);** **Rc <- tmp2[31:24]:tmp2[15: 8]:tmp2[23:16]:tmp2[7: 0],** **if (VC0 == 1);** 使用於影像處理之動畫補償。 按**CFR.ALN**執行位元組調正，並藉**VC0**值變更所取出之位元組資料。

組合程式助憶	格式	影響旗標	受影響之旗標
valnvc3 Rc,Ra,Rb	32bit	—	CFR.ALN,VC0

備考

第 56 圖

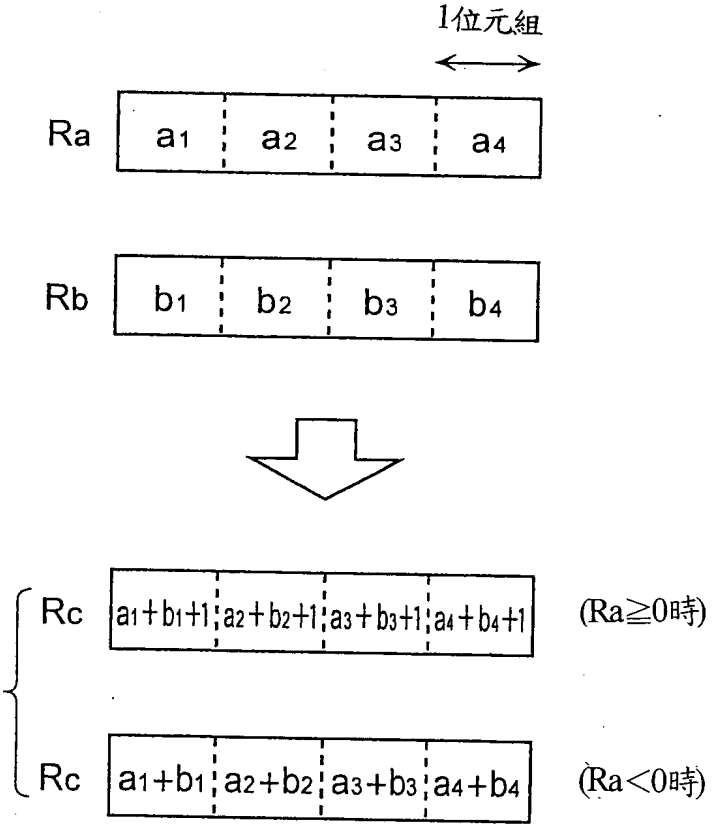
valnvc4 Rc,Ra,Rb

動作	**tmp [87: 0] <- Ra:Rb << (CFR.ALN[1:0] << 3);** **tmp2[31: 0] <- tmp[47:16];** **Rc <- tmp2[31:24]:tmp2[31:24]:tmp2[23:16]:tmp2[23:16],** **if (VC0 == 0);** **Rc <- tmp2[31:24]:tmp2[15: 8]:tmp2[23:16]:tmp2[7: 0],** **if (VC0 == 1);** 使用於影像處理之動畫補償。 按**CFR.ALN**執行位元組調正，並藉**VC0**值變更所取出之位元組資料。

組合程式助憶	格式	影響旗標	受影響之旗標
valnvc4 Rc,Ra,Rb	32bit	–	CFR.ALN,VC0

備考
當 CFR.ALN[1:0]==3時，tmp2[7: 0] 是 0x00。

第 57 圖


1位元組
Ra
a1
a2
a3
a4
Rb
b1
b2
b3
b4
Rc
a1+b1+1
a2+b2+1
a3+b3+1
a4+b4+1
(Ra≧0時)
Rc
a1+b1
a2+b2
a3+b3
a4+b4
(Ra<0時)

第 58 圖

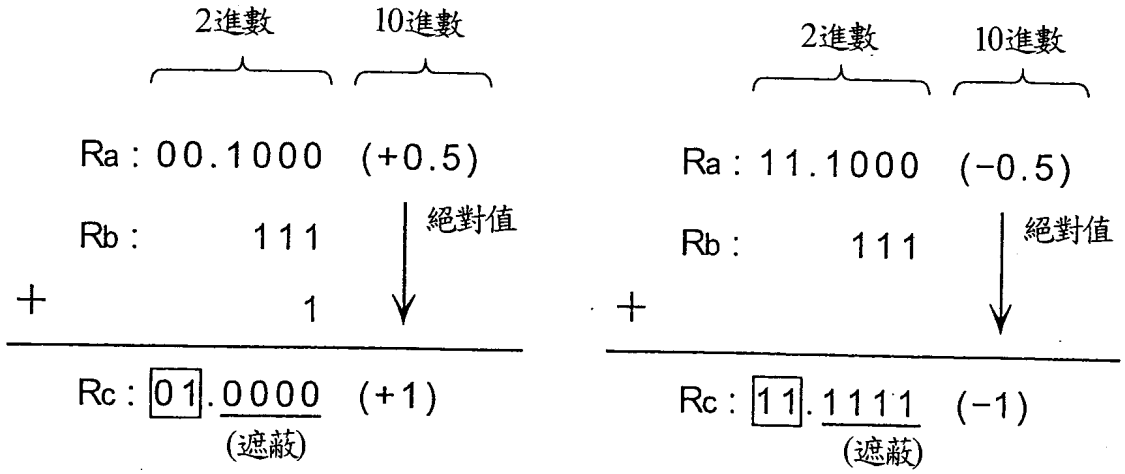
addarvw Rc,Ra,Rb

動作	**tmp[32:0] <- sext33(Ra) + sext33(Rb) + uext33(~(Ra[31:31]));** **Rc <- tmp[31:0];** **Rc <- 0x7fff_ffff, CFR.OVS <- 1,** **if (tmp[32:0] > 0x0_7fff_ffff[32:0]);** **Rc <- 0x8000_0000, CFR.OVS <- 1,** **if (tmp[32:0] < 0x1_8000_0000[32:0]);** 進行影像處理之絕對值捨進。 將 **Ra** 與 **Rb**相加。 **Ra** 是正值時，進而加 **1**。

組合程式助憶	格式	影響旗標	受影響之旗標
addarvw Rc,Ra,Rb	32bit	CFR.OVS	–

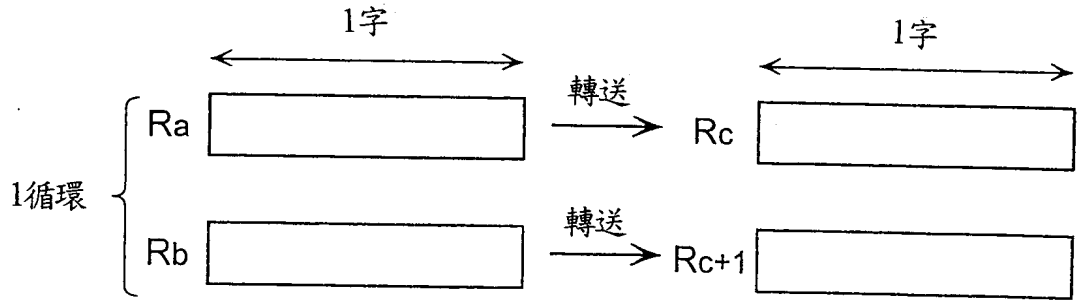
備考
進行絕對值捨進 (away from zero)，在Rb上較存放欲做捨進之位元還低之位元用1填入之值。 ex. Rb <- 0x0000_7fff (捨進成 MSB 16 位元時)

第 59 圖



(a) Ra≧0時　　(b) Ra<0時

第 60 圖



第 61 圖

movp Rc:Rc+1,Ra,Rb

動作	**Rc <- Ra;** **Rc+1 <- Rb;** **將Ra轉送至Rc 且將Rb轉送至Rc+1。**

組合程式助憶	格式	影響旗標	受影響之旗標
movp Rc:Rc+1,Ra,Rb	32bit	—	—

備考
令Rc爲偶數。

第 62 圖

jloop C6,Cm,TAR,Ra

動作	**=>jloop C6,Cm,TAR,Ra,-1** **PC <- TAR; C6 <- (Ra >= 0)? 1:0; Cm <- 1;** **Ra <- Ra - sext(1);** 於迴路使用。 進行以下之處理。 **(1)** 在**Cm**設定**1**。 **(2)** 在**Ra**加上 **-1**，並將該結果儲存於**Ra**。當**Ra**較**0**還小時，則於**C6**設定**0**。 **(3)** 分支於**TAR**所示之位址。在分支用指令緩衝器未填充分支處之指令時，則填充分支處之指令。

組合程式助憶	格式	影響旗標	受影響之旗標
jloop C6,Cm,TAR,Ra	32bit 同義字	Cm,C6	—

備考
※ Cm = C6 時之動作爲未定義。

第 63 圖

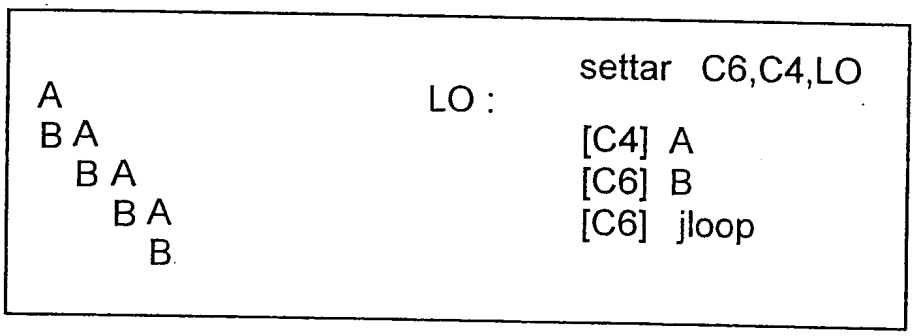
settar C6,Cm,D9

動作	**TAR <- PC + (sext(D9[8:1]) <<1);** **C6 <- 1; Cm <- 0;** 進行以下之處理。 **(1) 將PC值與位移值(D9)相加之位址儲存於TAR。** **(2) 提取該位址之指令後，再儲存於分支用指令緩衝器。** **(3) 將C6設定於1，並將Cm設定於0。**

組合程式助憶	格式	影響旗標	受影響之旗標
settar C6,Cm,D9	32bit	Cm,C6	—

備考
D9係帶正負號值，請令低 1 位元爲 0。 Cm = C6 時之動作爲未定義。

第 64 圖



第 65 圖

```
int func2(int a, int b, int c)
{
        int i;
        int t;

        for (i = 0; i < 100; i++){
                y[i] = x[i] + i;
                t += x[i];
        }
        return t;
}
```

第 66 圖

```
        mov     r4,0
        ld      r6,(gp,_x$ - .MN.gptop)
        cmpne   C4,gp,gp        // 需要該指令
        ;;
        mov     r1,98
        settar  C6,L00023
        ld      r5,(gp,_y$ - .MN.gptop)
        ;;
L00023                          //3cycle/iterartion
[C4]    add     r2,r3,r4
[C4]    add     r0,r3,r0
[C6]    ld      r3,(r6+)
        ;;
[C4]    add     r4,r4,1
[C4]    st      (r5+),r2
        ;;
        cmpeq   C4,gp,gp        // 需要該指令
[C6]    jloop   C6,tar,r1,-1
        ;;
        ret
        ;;
```

```
        mov     r4,0
        ld      r6,(gp,_x$ - .MN.gptop)
        ;;
        mov     r1,98
        settar  C6,C4,L00023    //C4 重設亦同時進行
        ld      r5,(gp,_y$ - .MN.gptop)
        ;;
L00023                                  //2cycle/iterartion
[C4]    add     r2,r3,r4
[C4]    add     r0,r3,r0
[C6]    ld      r3,(r6+)
        ;;
[C4]    add     r4,r4,1
[C4]    st      (r5+),r2
[C6]    jloop   C6,C4,tar,r1,-1//C4 重設亦同時進行
        ;;
        ret
        ;;
```

第 68 圖

jloop C6,C2:C4,TAR,Ra

動作	**=> jloop C6,C2:C4,TAR,Ra,-1** **PC <- TAR;** **C2 <- C3, C3 <- C4, C6 <- C4;** **C4 <- (Ra >= 0)? 1 : 0;** **Ra <- Ra - sext(1);** 於迴路使用。 進行以下之處理。 **(1)** 將**C3**轉送至**C2**，且將**C4**轉送至**C3**及**C6**。 **(2)** 在**Ra**加上 **-1**，並將該結果儲存於**Ra**。當**Ra**較**0**還小時，則於**C4**設定**0**。 **(3)** 分支於**TAR**所示之位址。在分支用指令緩衝器未填充分支處之指令時，則填充分支處之指令。

組合程式助憶	格式	影響旗標	受影響之旗標
jloop C6,C2:C4,TAR,Ra	32bit 同義字	C2,C4,C6,C3	—
備考			

第 69 圖

settar C6,C2:C4,D9

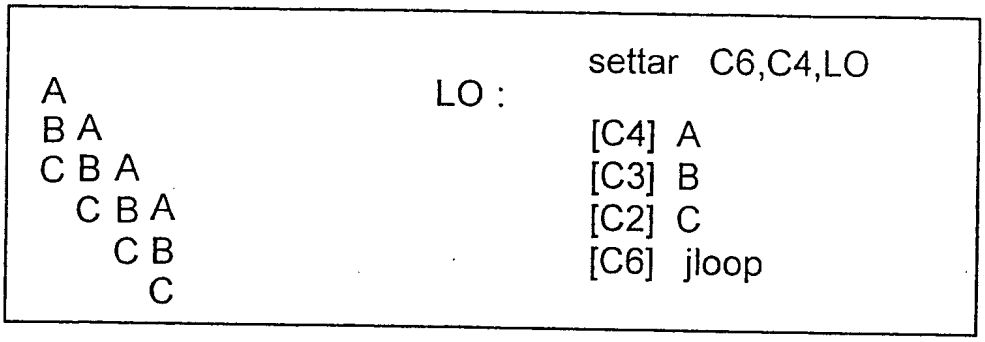
動作	**TAR <- PC + (sext(D9[8:1]) <<1);** **C2 <- 0; C3 <- 0; C4 <- 1, C6 <- 1;** 進行以下之處理。 **(1)** 將**PC**值與位移值**(D9)**相加之位址儲存於**TAR**。 **(2)** 提取該位址之指令後，再儲存於分支用指令緩衝器。 **(3)** 將**C4**及**C6**設定於**1**，並將**C2**及**C3**設定於**0**。

組合程式助憶	格式	影響旗標	受影響之旗標
settar C6,C2:C4,D9	32bit	C2,C4,C6,C3	—

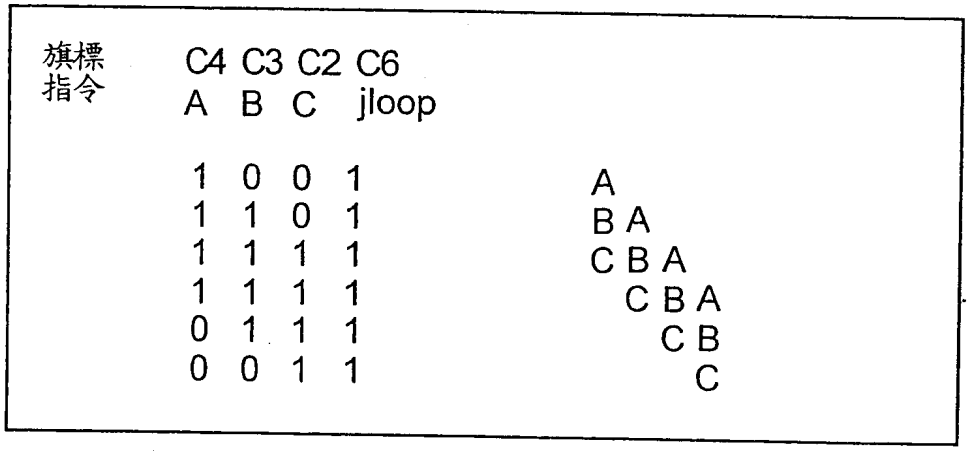
備考
D9係帶正負號值，請令低 1 位元爲 0。

第 70 圖

(a)



(b)



第 71 圖

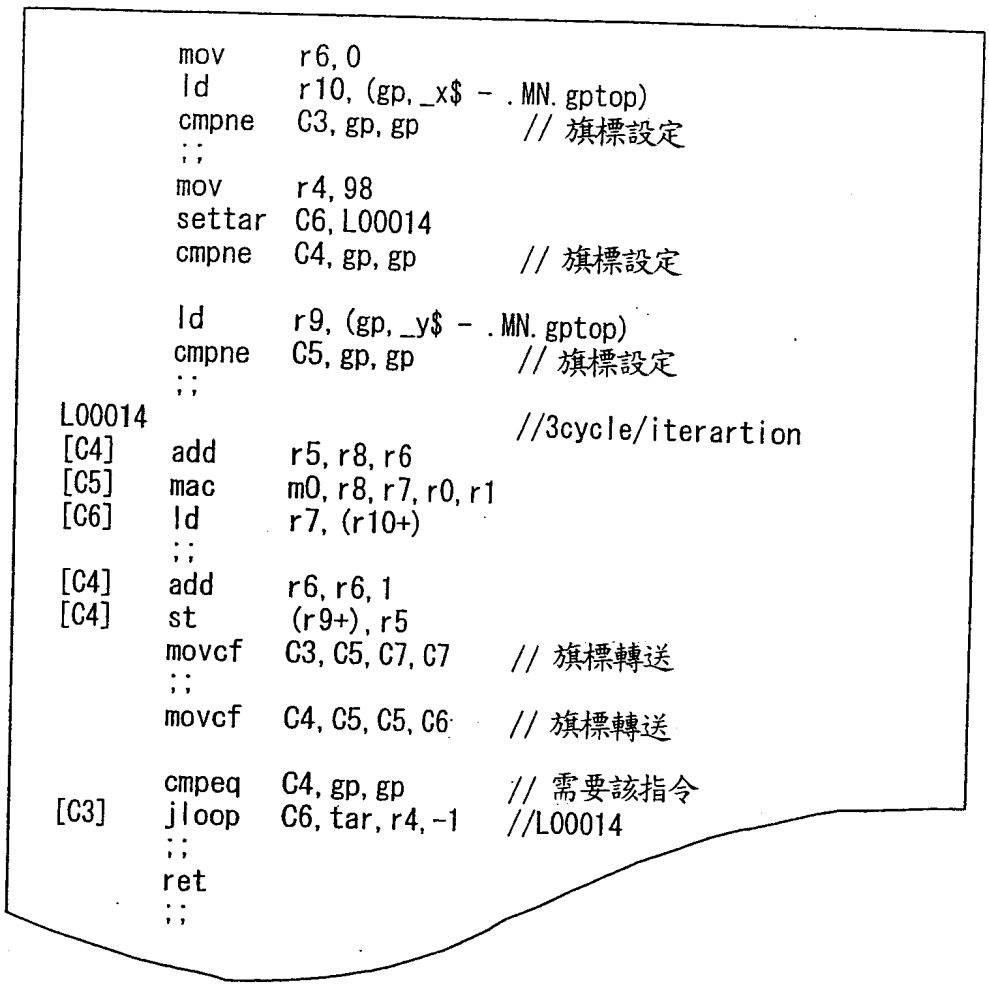
```
int x[100], y[100];

int func(int a, int b, int c)
[
        int i:

        for (i = 0; i <100; i++){
                y[i] = a * x[i] + b + i:
        }
        return t;
}
```

200945190

第 72 圖



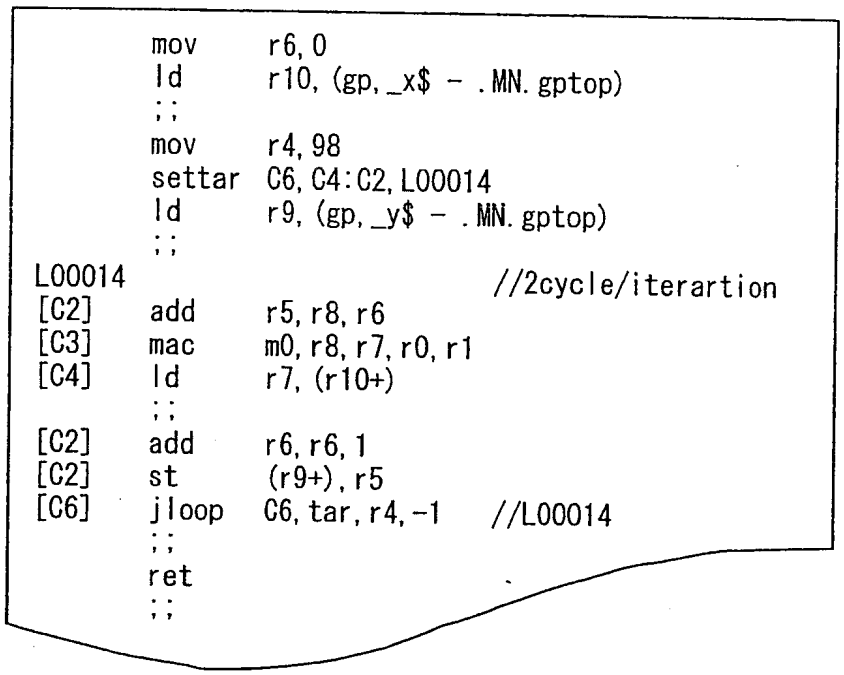
```
        mov     r6,0
        ld      r10,(gp,_x$ - .MN.gptop)
        cmpne   C3,gp,gp        // 旗標設定
        ;;
        mov     r4,98
        settar  C6,L00014
        cmpne   C4,gp,gp        // 旗標設定

        ld      r9,(gp,_y$ - .MN.gptop)
        cmpne   C5,gp,gp        // 旗標設定
        ;;
L00014                          //3cycle/iterartion
[C4]    add     r5,r8,r6
[C5]    mac     m0,r8,r7,r0,r1
[C6]    ld      r7,(r10+)
        ;;
[C4]    add     r6,r6,1
[C4]    st      (r9+),r5
        movcf   C3,C5,C7,C7     // 旗標轉送
        ;;
        movcf   C4,C5,C5,C6     // 旗標轉送

        cmpeq   C4,gp,gp        // 需要該指令
[C3]    jloop   C6,tar,r4,-1    //L00014
        ;;
        ret
        ;;
```

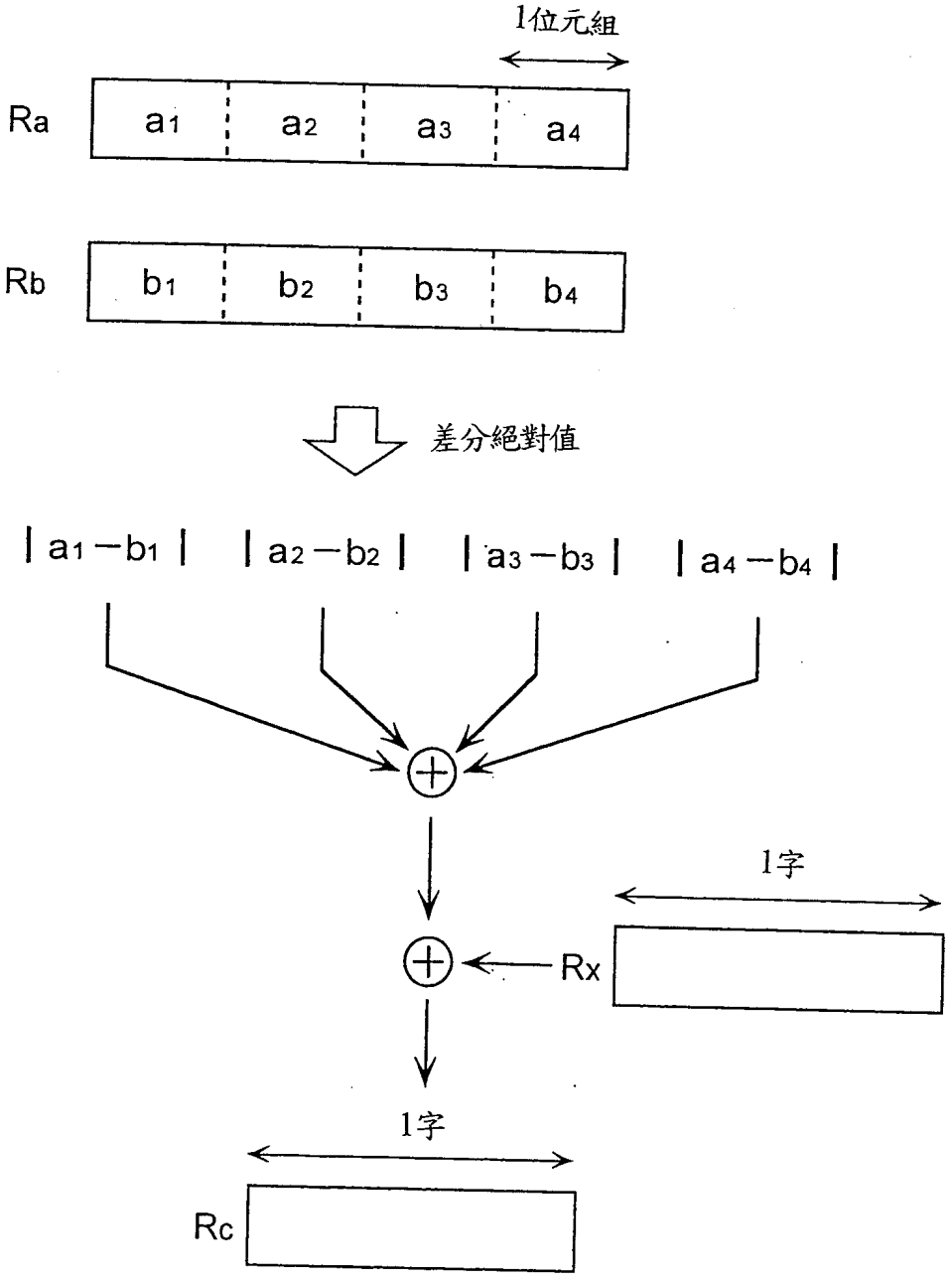
200945190

第 73 圖



```
        mov     r6,0
        ld      r10,(gp,_x$ - .MN.gptop)
        ;;
        mov     r4,98
        settar  C6,C4:C2,L00014
        ld      r9,(gp,_y$ - .MN.gptop)
        ;;
L00014                                  //2cycle/iterartion
[C2]    add     r5,r8,r6
[C3]    mac     m0,r8,r7,r0,r1
[C4]    ld      r7,(r10+)
        ;;
[C2]    add     r6,r6,1
[C2]    st      (r9+),r5
[C6]    jloop   C6,tar,r4,-1    //L00014
        ;;
        ret
        ;;
```

第 74 圖


1位元組
Ra
a1
a2
a3
a4
Rb
b1
b2
b3
b4
差分絕對值
| a1 − b1 |
| a2 − b2 |
| a3 − b3 |
| a4 − b4 |
1字
Rx
1字
Rc

第 75 圖

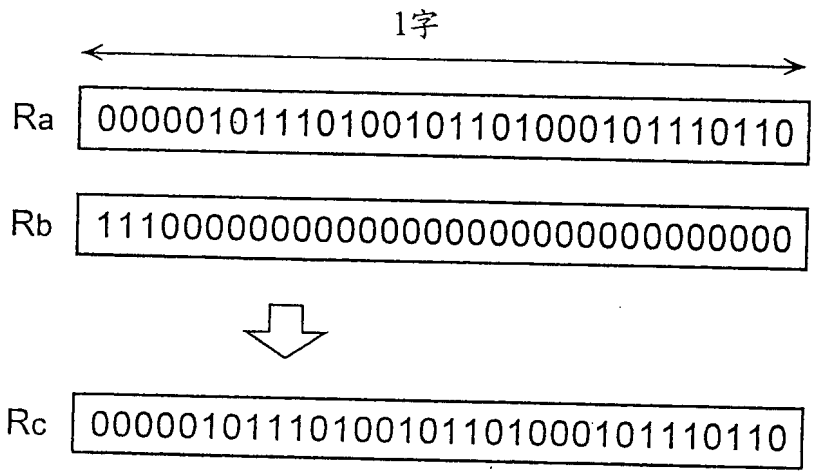
(a)

組合程式助憶	vsada Rc,Ra,Rb,Rx
動作	Rc <- Rx + sext(abs(uext9[Rb[31:24]]-uext9(Ra[31:24])) + abs (uext9[Rb[23:16]] – uext9(Ra[23:16])) + abs (uext9[Rb[15: 8]] – uext9(Ra[15: 8])) + abs (uext9[Rb[7: 0]] – uext9(Ra[7: 0]))

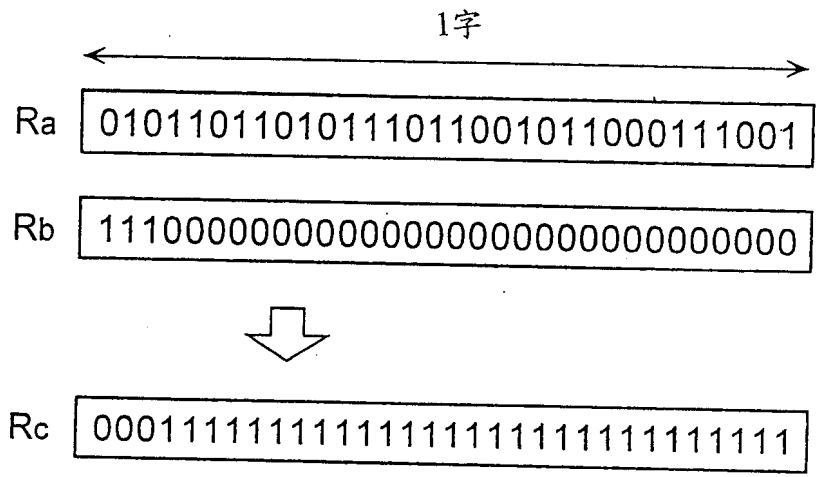
(b)

組合程式助憶	vsada Rc,Ra,Rb
動作	Rc <- sext(abs(uext9[Rb[31:24]]-uext9(Ra[31:24])) + abs (uext9[Rb[23:16]] – uext9(Ra[23:16])) + abs (uext9[Rb[15: 8]] – uext9(Ra[15: 8])) + abs (uext9[Rb[7: 0]] – uext9(Ra[7: 0]))

第 76 圖



(a) Ra小於飽和值時



(b) Ra大於飽和值時

200945190

第 77 圖

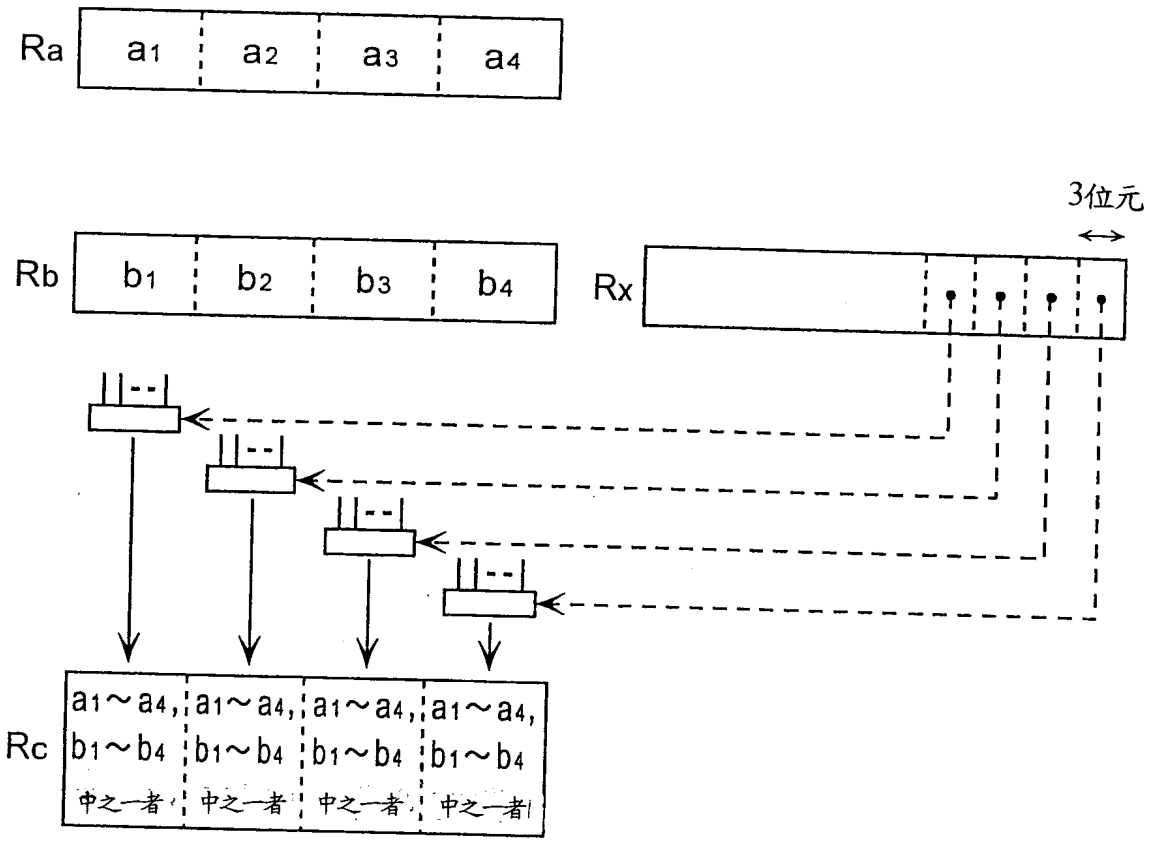
(a)

組合程式助憶	satss Rc,Ra,Rb
動作	tmp <- (sext(Ra[31:31] & Rb) ^ (Ra & Rb); Rc <- Ra; Rc <- Rb ^ (~sext(Ra[31:31])), if(tmp != 0x0000_0000);

(b)

組合程式助憶	satsu Rc,Ra,Rb
動作	tmp <- (Ra & Rb); Rc <- Ra; Rc <- (~Rb) ^ (~sext(Ra[31:31])), if(tmp != 0x0000_0000);

第 78 圖



(a)

組合程式助憶	bytesel Rc,Ra,Rb,Rx
動作	Rc[31:24] <- ((Ra:Rb) >>> (Rx[11:9] << 3))[7:0]; Rc[23:16] <- ((Ra:Rb) >>> (Rx[8: 6] << 3))[7:0]; Rc[15:8] <- ((Ra:Rb) >>> (Rx[5: 3] << 3))[7:0]; Rc[7: 0] <- ((Ra:Rb) >>> (Rx[2: 0] << 3))[7:0];

(b)

Rx[n+2:n]	選擇對象
000	選擇Ra之第1位元組 (Ra:Rb[63:56])
001	選擇Ra之第2位元組 (Ra:Rb[55:48])
010	選擇Ra之第3位元組 (Ra:Rb[47:40])
011	選擇Ra之第4位元組 (Ra:Rb[39:32])
100	選擇Rb之第1位元組 (Ra:Rb[31:24])
101	選擇Rb之第2位元組 (Ra:Rb[23:16])
110	選擇Rb之第3位元組 (Ra:Rb[15:8])
111	選擇Rb之第4位元組 (Ra:Rb[7:0])

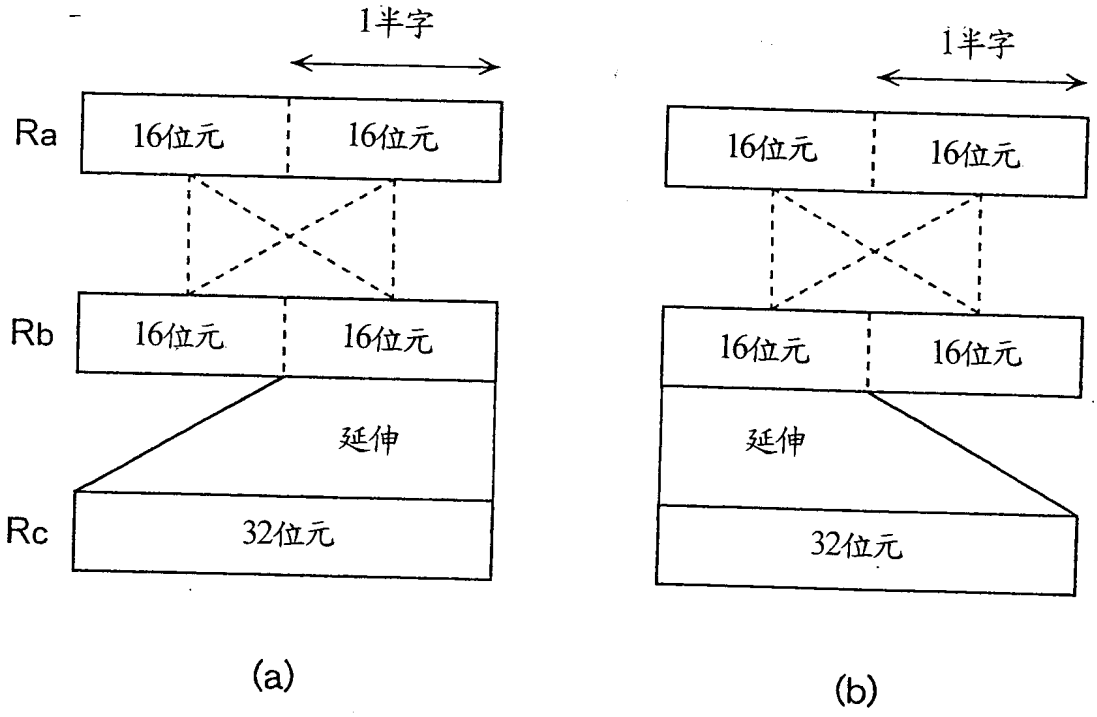
(c)

組合程式助憶	bytesel Rc,Ra,Rb,I12
動作	Rc[31:24] <- ((Ra:Rb) >>> (I12[11:9] << 3))[7:0]; Rc[23:16] <- ((Ra:Rb) >>> (I12[8: 6] << 3))[7:0]; Rc[15:8] <- ((Ra:Rb) >>> (I12[5: 3] << 3))[7:0]; Rc[7: 0] <- ((Ra:Rb) >>> (I12[2: 0] << 3))[7:0];

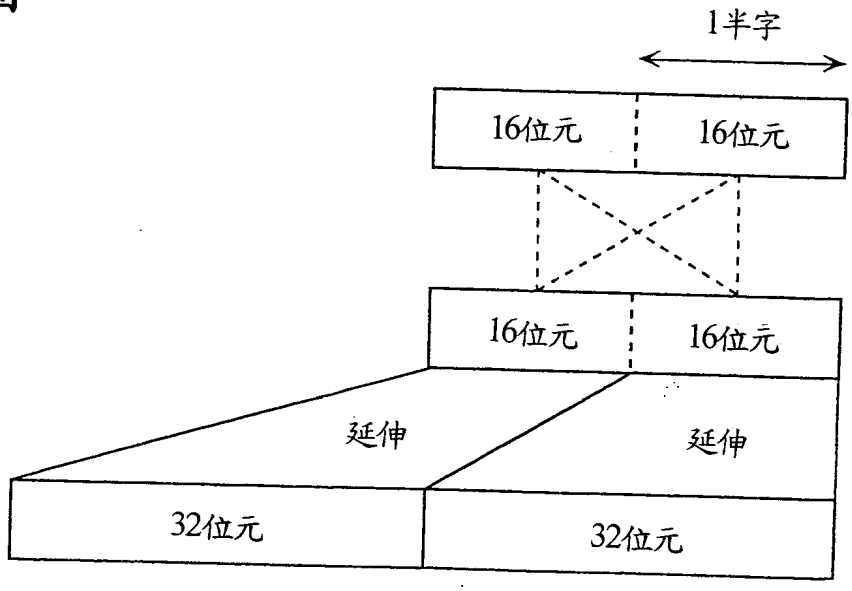
(d)

I12[n+2:n]	選擇對象
000	選擇Ra之第1位元組 (Ra:Rb[63:56])
001	選擇Ra之第2位元組 (Ra:Rb[55:48])
010	選擇Ra之第3位元組 (Ra:Rb[47:40])
011	選擇Ra之第4位元組 (Ra:Rb[39:32])
100	選擇Rb之第1位元組 (Ra:Rb[31:24])
101	選擇Rb之第2位元組 (Ra:Rb[23:16])
110	選擇Rb之第3位元組 (Ra:Rb[15:8])
111	選擇Rb之第4位元組 (Ra:Rb[7:0])

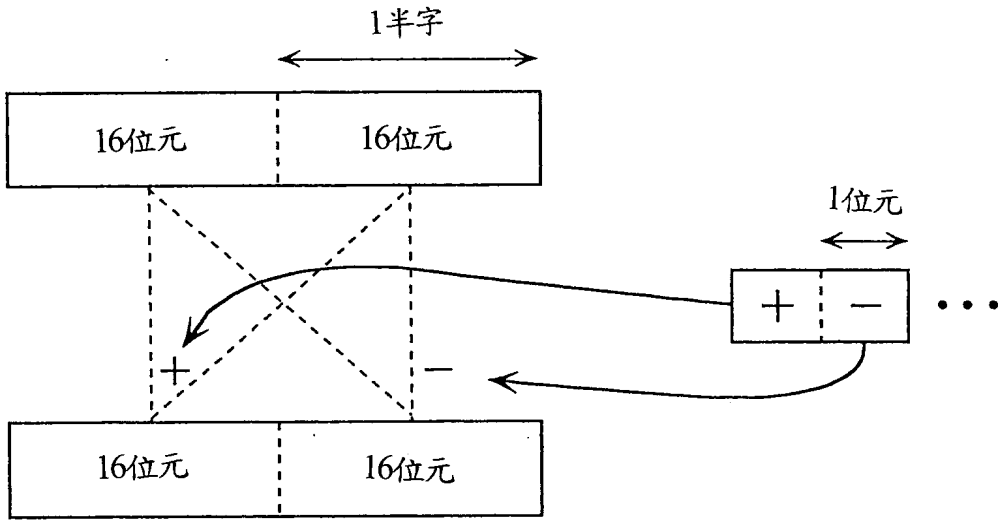
第 80 圖


1半字
Ra
16位元
16位元
Rb
16位元
16位元
延伸
Rc
32位元
(a)
1半字
16位元
16位元
16位元
16位元
延伸
32位元
(b)

第 81 圖


1半字
16位元
16位元
16位元
16位元
延伸
延伸
32位元
32位元

第 82 圖



四、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：

1…處理器
10…指令控制部
20…解碼部
30…暫存器檔
40…運算部
41…算術邏輯暨比較運算器#x
42…算術邏輯暨比較運算器#y
43…算術邏輯暨比較運算器#z
44…積和運算器
45…桶移位器
46…除法器
47…變換器
50…I/F部
60…指令記憶體部
70…資料記憶體部
80…延伸暫存器部
90…I/O介面部

五、本案若有化學式時，請揭示最能顯示發明特徵的化學式：