Pràctica 3

Processadors vectorials (II)

1 Introducció

En la pràctica anterior es va introduir l'arquitectura vectorial DLXV. Per a això es van analitzar diverses rutines que mostraven detencions del pipeline per diversos motius. Finalment es va proposar a l'estudiant que implementara el bucle DAXPY per a vectors d'una longitud major que MVL, d'esta manera es va iniciar l'estudiant en les tècniques de programació en màquines vectorials.

En la present pràctica es revisaran diverses estratègies de programació en processadors vectorials amb l'objectiu d'augmentar el rendiment del processador.

2 Tècniques de programació en DLXV. Revisió

En el cas del Bucle DAXPY proposat en la pràctica anterior es va introduir el cas en què la grandària del vector a tractar diferia de la grandària dels vectors de la màquina vectorial. Este problema introduïa la tècnica de seccionament o *strip mining*, per a augmentar el rendiment del processador. Hi ha més casos en què el rendiment de la màquina vectorial es degrada. Vegem alguns d'ells.

2.1 Separació entre els elements d'un vector.

Un problema que ràpidament sorgix és el de l'accés als elements d'un vector que no ocupen posicions consecutives de memòria. Vegem el següent codi per a la multiplicació de matrius:

```
      do 10  i = 1, 100
      do 10  j = 1, 100
             A(i,j) = 0.0
      do 10  k= 1, 100
10           A(i,j) = A(i,j) + B(i,k)*C(k,j)
```

La multiplicació de cada fila de B amb cada columna de C pot ser vectorizada i aplicar la tècnica de seccionament al bucle interior, amb `k` com l'índex variable. La distribució per columnes, usada pel FORTRAN, fa que els elements `B(i,j)` i `B(i+1,j)` siguen adjacents, és a dir, en cada iteració s'accedirà a un element separat de l'anterior per una fila completa de la matriu. En este cas, els elements de `B` que són accedits per les iteracions del bucle interior estan separats per 8 vegades (el nombre de bytes per element) la grandària de la fila, resultant un total de 800 bytes.

En l'exemple anterior, utilitzant una distribució per columnes per a les matrius, la matriu C té una separació o *stride* d'1 (1 doble paraula, és a dir, 8 bytes) i la matriu B té una separació o *stride* de 100 (100 dobles paraules, és a dir, 800 bytes).

Una vegada que els elements desitjats han sigut carregats en el registre vectorial, es té un nou vector amb els seus elements lògicament adjacents. Açò permet a una màquina vector-registre manejar separacions majors que u, crides separacions no unitàries, fent que les operacions de càrrega i emmagatzematge de vectors siguen més generals. Si s'emmagatzemen per files, els elements consecutius de les files són adjacents en memòria, mentres que si s'emmagatzemen per columnes, els elements consecutius en les columnes són adjacents. Açò és extensible a matrius amb `n` dimensions.

La separació en un vector, igual que la direcció de començament del vector, pot ser col·locat en un registre de propòsit general, on és utilitzat per l'operació vectorial. La instrucció `lvws` (lloeu *vector with stride* del DLXV és la utilitzada per a carregar un vector utilitzant una separació (*stride*). Igualment, quan un vector amb separació no unitària serà emmagatzemat, pot utilitzar-se la instrucció `svws` (*store vector with stride*).

En la unitat de memòria poden aparéixer complicacions al suportar separacions majors de u, per tant serà possible sol·licitar accessos al mateix banc a una velocitat superior al temps d'accés de la memòria. Esta situació rep el nom de conflicte **de bancs de memòria** i provoca que cada càrrega o emmagatzematge siga penalitzada en major grau pel temps d'accés de la memòria.

Un conflicte en els bancs de memòria es produïx sempre que es demana al mateix banc un accés abans de que haja completat l'anterior. En els bancs de memòria no ocorreran conflictes si els valors de la separació i del nombre de bancs són cosins entre si i hi ha suficients bancs de memòria per a evitar conflictes en el cas de separació unitària.

2.2 Matrius disperses.

Una matriu dispersa és una matriu en què la majoria dels elements és zero. Estes matrius se solen emmagatzemar d'una manera intel·ligent (amb vectors que emmagatzemen els índexs que no són zero). D'esta manera una suma de vectors dispersos es podria fer de la manera següent:

```
      do 100  i = 1, n
100      A(K(i)) = A(K(i)) + C(M(i))
```

Este codi implementa la suma de `A` i `C`, usant els vectors d'índexs `K` i `M` per a designar els elements que no són zero de `A` i de `C`. (`A` i `C` han de tindre el mateix nombre d'elements que no són zero, en este cas `n`.)

Un mecanisme per a suportar les matrius disperses es basa en les operacions **d'expansió-comprensió** (*scatter-gather*) utilitzant un **vector d'índexs**. Una operació de compressió (*gather*) presa un vector d'índexs i busca el vector els elements del qual estan en les direccions donades per la direcció base més els desplaçaments presents en el vector d'índexs. El resultat és un vector no dispers en el registre vectorial. Després d'operar sobre eixos elements d'una forma densa, el vector dispers pot ser emmagatzemat en forma expandida per mitjà d'un emmagatzematge distribuït (*gather*) utilitzant el mateix vector d'índexs.

Esta tècnica permet que el codi amb matrius disperses puga ser executat de forma vectorial. El codi font descrit anteriorment no podria ser vectoritzat automàticament per un compilador pel fet que el compilador no pot saber que els elements de `K` són valors distints i que, per tant, no hi ha dependències. En el seu lloc, una directiva de l'ensamblador podria comunicar al compilador que pot vectoritzar el bucle.

Les instruccions `lvi` (Carregar vector indexat) i `svi` (Emmagatzemar vector indexat) proporcionen eixes operacions en el DLXV. La instrucció `cvi` (Crear vector d'índexs) del DLXV crega un vector d'índexs a partir d'una separació (m) on els valors del vector d'índexs seran: 0,m,2m,..., 63m.

3 Experimentant amb el simulador xdlxvsim

Exercici 1:

Realitzar un programa de multiplicació de matrius emmagatzemades **per columnes en memòria** (com en Fortran). Les matrius seran de 6x6 nombres de punt flotant de doble precisió (8 bytes) amb estos valors:

$$\begin{pmatrix} 10 & 11 & 12 & 13 & 14 & 15 \\ 20 & 21 & 22 & 23 & 24 & 25 \\ 30 & 31 & 32 & 33 & 34 & 35 \\ 40 & 41 & 42 & 43 & 44 & 45 \\ 50 & 51 & 52 & 53 & 54 & 55 \\ 60 & 61 & 62 & 63 & 64 & 65 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 7 & 8 & 9 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 & 8 & 9 \\ 9 & 8 & 7 & 6 & 5 & 4 \\ 3 & 2 & 1 & 9 & 8 & 7 \\ 6 & 5 & 4 & 3 & 2 & 1 \end{pmatrix} = \begin{pmatrix} 384 & 375 & 366 & 384 & 375 & 366 \\ 684 & 675 & 666 & 684 & 675 & 666 \\ 984 & 975 & 966 & 984 & 975 & 966 \\ 1284 & 1275 & 1266 & 1284 & 1275 & 1266 \\ 1584 & 1575 & 1566 & 1584 & 1575 & 1566 \\ 1884 & 1875 & 1866 & 1884 & 1875 & 1866 \end{pmatrix}$$

Es recorda que el DLXV no té instruccions per a sumar tots els elements d'un mateix vector. També es recorda que accedir a elements consecutius en memòria és més eficient que si els elements estan separats.

Exercici 2:

A continuació es pretén repassar els conceptes exposats al principi de la memòria i observar les parades i rendiments de bucles que utilitzen estes tècniques de programació en màquines vectorials. En primer lloc executa en el mode pas a pas el bucle següent:

```
;Exemple Vectorial 4
;Accés a matrius disperses
        .data 0
        .global a
a:      .double 1, 0, 0, 0, 0, 0, 0, 0, 0, 0
        .double 2, 0, 0, 0, 0, 0, 0, 0, 0, 0
        .double 3, 0, 0, 0, 0, 0, 0, 0, 0, 0
        .double 4, 0, 0, 0, 0, 0, 0, 0, 0, 0
        .double 5, 0, 0, 0, 0, 0, 0, 0, 0, 0
        .double 6, 0, 0, 0, 0, 0, 0, 0, 0, 0
        .double 7, 0, 0, 0, 0, 0, 0, 0, 0, 0
        .double 8, 0, 0, 0, 0, 0, 0, 0, 0, 0
        .double 9, 0, 0, 0, 0, 0, 0, 0, 0, 0
        .double 10, 0, 0, 0, 0, 0, 0, 0, 0, 0
        .global k
k:      .double 0, 80, 160, 240, 320, 400, 480, 560, 640, 720
        .global c
c:      .double 0, 10, 0, 0, 0, 0, 0, 0, 0, 0
        .double 0, 10, 0, 0, 0, 0, 0, 0, 0, 0
        .double 0, 10, 0, 0, 0, 0, 0, 0, 0, 0
        .double 0, 10, 0, 0, 0, 0, 0, 0, 0, 0
        .double 0, 10, 0, 0, 0, 0, 0, 0, 0, 0
        .double 0, 10, 0, 0, 0, 0, 0, 0, 0, 0
        .double 0, 10, 0, 0, 0, 0, 0, 0, 0, 0
        .double 0, 10, 0, 0, 0, 0, 0, 0, 0, 0
        .double 0, 10, 0, 0, 0, 0, 0, 0, 0, 0
        .double 0, 10, 0, 0, 0, 0, 0, 0, 0, 0
        .global m
m:      .double 8, 88, 168, 248, 328, 408, 488, 568, 648, 728

inici:  add     r10, r0, k      ;r10 conté la direcció del vector K
        add     r11, r0, a      ;r11 conté la direcció del vector A
        add     r12, r0, m      ;r12 conté la direcció del vector M
        add     r13, r0, c      ;r13 conté la direcció del vector C
        add     r1, r0, #10     ;10 -> r1
        movi2s  vlr, r1         ;10 -> vlr
        lv      v1, r10         ;Càrrega vector K en V1
```

```
        lvi     v2, r11, v1     ;Càrrega vector A(K(i))
        lv      v3, r12         ;Càrrega vector M
        lvi     v4, r13, v3     ;Càrrega vector C(M(i))
        addv    v2, v2, v4      ;Suma vectorial
        svi     r11, v1, v2     ;Emmagatzema A(K(i))
        trap    #0              ;Fi del programa vectorial
```

En el present exemple s'utilitza l'accés a matrius disperses comentat anteriorment. Observa les detencions que es produïxen i justifica el per què d'estes detencions. Com es podrien evitar? Estudia la utilització del vector d'índexs per a l'agrupament/dispersió dels elements corresponents.

Exercici 3:

A continuació es proposa que l'estudiant execute el següent codi vectorial on s'usen tècniques d'execució condicional per mitjà d'agrupament i dispersió.

```
;Exemple Vectorial 5
;Execució condicional d'instruccions per mitjà de scatter/gather

        .data 0
        .global a
a:      .double 10, 0, 10, 0, 10, 0, 10, 0, 10, 0
        .double 10, 0, 10, 0, 10, 0, 10, 0, 10, 0
        .global b
b:      .double 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
        .double 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
        .global zero
zero:   .double 0

inici:  add     r10, r0, a      ;r10 conté la direcció del vector A
        add     r11, r0, b      ;r11 conté la direcció del vector B
        add     r1, r0, #20     ;20 -> r1
        movi2s  vlr, r1         ;20 -> vlr
        add     r2, r0, #8      ;8 -> r2
        lv      v1, r10         ;Càrrega vector A en V1
        ld      f0, zero        ;Càrrega un zero en coma flotant en F0
        snesv   f0, v1          ;Posa VM a 1 sí V1(i) diferent de F0
        cvi     v2, r2          ;Crega vector d'índexs en V2
        pop     r1, vm          ;Compte el numere d'uns de VM
        movi2s  vlr, r1         ;Càrrega el vector de longitud vectorial
        cvm                     ;Posa VM tot a 1
        lvi     v3, r10, v2     ;Càrrega els elements de A diferents de zero
        lvi     v4, r11, v2     ;Càrrega els elements de B corresponents
        subv    v3, v3, v4      ;Resta vectorial
        svi     r10, v2, v3     ;Emmagatzema el resultat en A
        trap    #0              ;Fi del programa vectorial
```

Estudia novament el codi amb atenció. Justifica novament les detencions que es produïxen i proposa un mètode per a evitar les detencions de memòria.

4 Treball a realitzar

En la memòria sobre processadors vectorials, tant de la pràctica 3 com de la 4, s'adjuntarà una anàlisi de temps d'execució de les rutines d'exemple i de les realitzades per l'estudiant, així com una anàlisi de riscos i rendiments de l'arquitectura vectorial del DLXV per a cada cas.