Chapitre 5 : Vecteurs et GPU

5.1 Une opération vectorielle très fréquente est Y = a*X + Y, appelée DAXPY en double précision. Ecrivez un programme vectoriel (VMIPS) pour effectuer ce calcul.

```
L.D     F0, a
LV      V1, Rx
MULVS.D V2, V1, F0
LV      V3, Ry
ADDVV.D V4, V2, V3
SV      V4, Ry
```

5.2 Vous exécutez le programme suivant sur un VMIPS qui prend un cycle par calcul dans un vecteur et peut chaîner les instructions qui n'ont pas de conflit structurel. Ce VMIPS contient une de chacune des unités matérielles suivantes, (avec la profondeur de pipeline associée), ADD-SUB (6), MUL (7), DIV(20), LOAD-STORE (12). Quel est le temps requis?

```
LV      V1, Rx
MULVS.D V2, V1, F0
LV      V3, Ry
ADDVV.D V4, V2, V3
SV      V4, Ry
```

```
LV      V1, Rx      ; 12 cycles de démarrage
MULVS.D V2, V1, F0  ; chaîné, 7 cycles
                    ; vecteur occupé, terminer 64 cycles
LV      V3, Ry      ; 12 cycles de démarrage
ADDVV.D V4, V2, V3  ; chaîné, 6 cycles
                    ; vecteur occupé, terminer 64 cycles
SV      V4, Ry      ; 12 + 64 cycles
```

12+7+64+12+6+64+12+64=241 cycles, FLOP: 2*64 = 128
1.88 cycle par FLOP

5.3 La mémoire est constituée de 8 bancs entrelacés avec chacun une latence de 12 cycles avant l'arrivée du premier mot, ou de 6 cycles après la fin du dernier accès non consécutif. Quel est le temps requis pour accéder 64 éléments consécutifs? Espacés de 32?

Les demandes d'accès sont envoyées aux 8 bancs consécutivement et les données commencent à arriver après 12 cycles, une par cycle pendant 64

cycles, un total de 76. Dans le second cas, tous les accès tombent dans le même banc mais non consécutifs à 4 mots de distance (32 / 8 = 4 reste 0). Il faut donc 12 cycles + 1 pour le premier accès et les autres arrivent à 6 cycles d'intervalle: 63 * 6, pour un total de 391 cycles.

5.4 Comment peut-on effectuer la réduction associée à un produit scalaire avec les opérations vectorielles du VMIPS?

On peut faire une simple boucle scalaire. Il est normalement plus rapide de faire quelques opérations vectorielles (32 + 32 éléments, 16 + 16...). Le mieux est souvent une combinaison des deux, faire la réduction vectoriellement de 64 à 32, 16 et 8, et ensuite faire une boucle scalaire.