Module 5: Vecteurs et GPU

INF8601: Systèmes Informatiques Parallèles

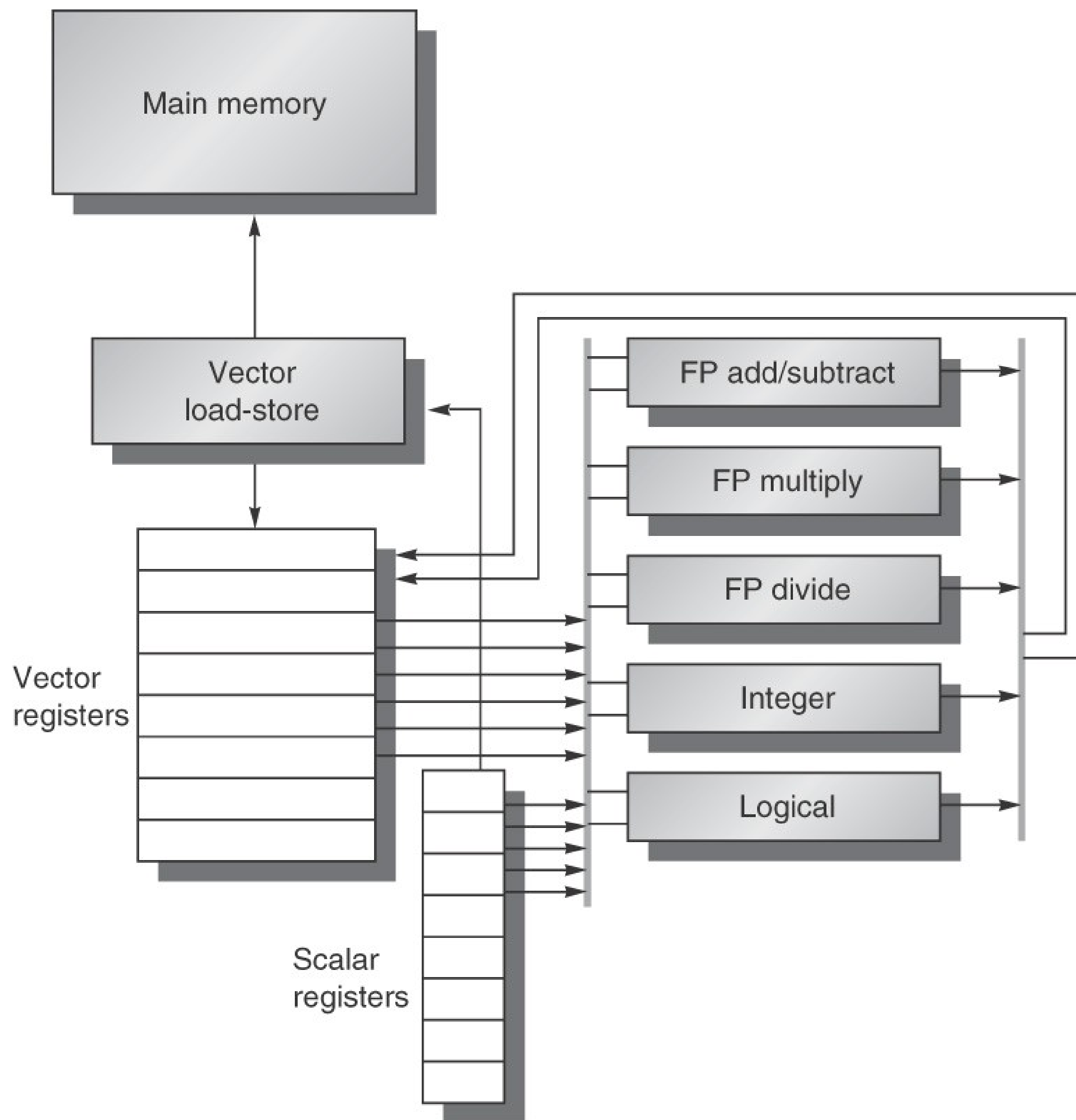
Michel Dagenais

Processeur vectoriel

ÉCOLE POLYTECHNIQUE MONTRÉAL

- Plusieurs registres vectoriels (e.g. 8 registres de 64 éléments de 64 bits) et scalaires.
- Plusieurs ALU (e.g. FP Add/Substract, FP Multiply, FP Divide, Integer), chacun en pipeline (un élément par cycle).
- Chargement et rangement à partir de la mémoire, en pipeline (un élément par cycle), avec un décalage de 1 ou plus entre les éléments, ou par vecteur d'indirection.
- Registre de masque (64 bits) pour activer les opérations ou recevoir le résultat d'une comparaison.
- Registre de longueur qui détermine le nombre d'éléments à traiter (1-64).

Processeur vectoriel

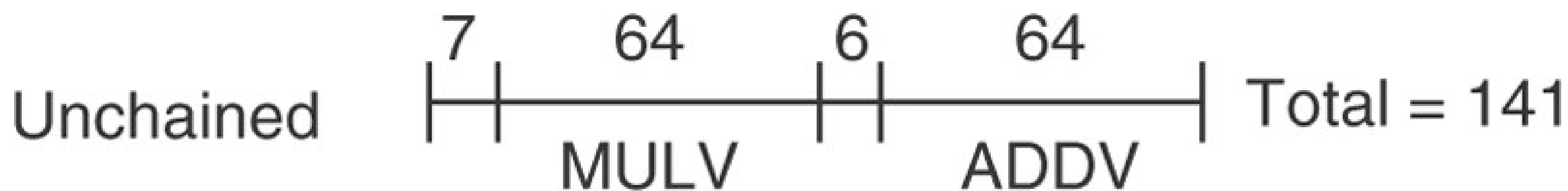


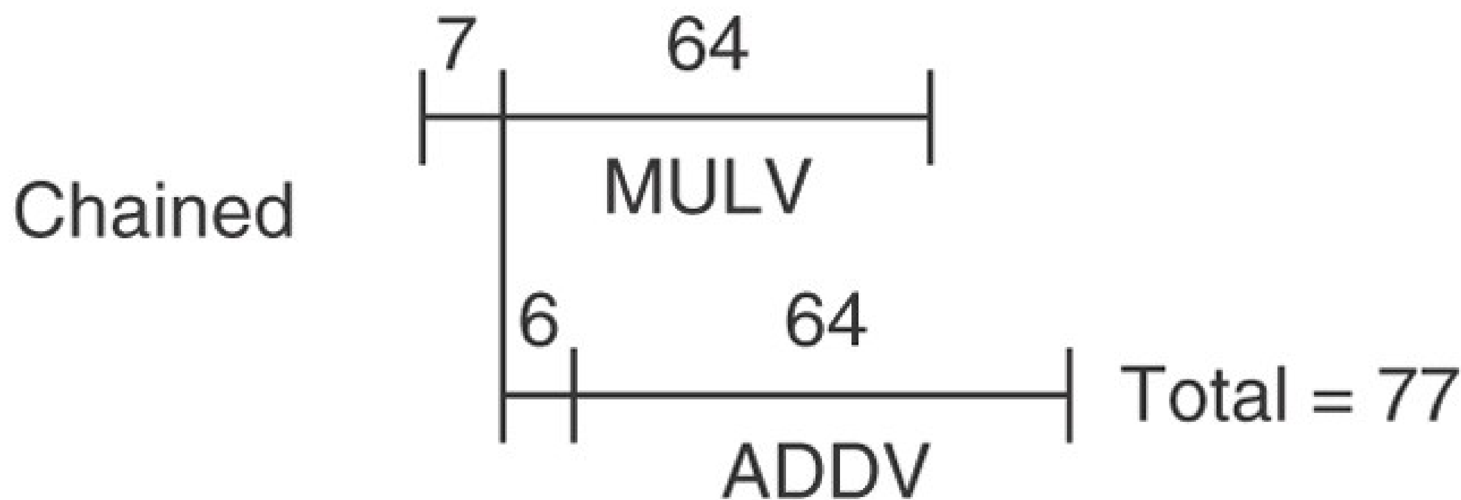
© 2007 Elsevier, Inc. All rights reserved.

Caractéristiques

- Réduction du nombre d'instructions exécutées (instruction vectorielle versus boucle).
- Plusieurs instructions vectorielles peuvent s'exécuter presqu'en même temps par chaînage si elles sont indépendantes ou si le résultat d'une est directement pris par l'autre au cycle suivant avec le matérial adéquat.
- Possibilité de 2 ou 4 voies pour chaque ALU, opérant sur 2 ou 4 éléments en parallèle.
- Attention à nb_elements / nb_voies versus profondeur du pipeline.
- Goûlot d'étranglement vers la mémoire?

Caractéristiques





© 2007 Elsevier, Inc. All rights reserved.

Répertoire d'instructions VMIPS

Instruction	Operands	Function
ADDV.D ADDVS.D	V1,V2,V3 V1,V2,F0	Add elements of V2 and V3, then put each result in V1. Add F0 to each element of V2, then put each result in V1.
SUBV.D SUBVS.D SUBSV.D	V1,V2,V3 V1,V2,F0 V1,F0,V2	Subtract elements of V3 from V2, then put each result in V1. Subtract F0 from elements of V2, then put each result in V1. Subtract elements of V2 from F0, then put each result in V1.
MULV.D MULVS.D	V1,V2,V3 V1,V2,F0	Multiply elements of V2 and V3, then put each result in V1. Multiply each element of V2 by F0, then put each result in V1.
DIVV.D DIVVS.D DIVSV.D	V1,V2,V3 V1,V2,F0 V1,F0,V2	Divide elements of V2 by V3, then put each result in V1. Divide elements of V2 by F0, then put each result in V1. Divide F0 by elements of V2, then put each result in V1.
LV	V1,R1	Load vector register V1 from memory starting at address R1.
SV	R1,V1	Store vector register V1 into memory starting at address R1.
LVWS	V1,(R1,R2)	Load V1 from address at R1 with stride in R2, i.e., R1+i × R2.
SVWS	(R1,R2),V1	Store V1 from address at R1 with stride in R2, i.e., R1+i × R2.
LVI	V1,(R1+V2)	Load V1 with vector whose elements are at R1+V2(i), i.e., V2 is an index.
SVI	(R1+V2),V1	Store V1 to vector whose elements are at R1+V2(i), i.e., V2 is an index.
CVI	V1,R1	Create an index vector by storing the values 0, 1 × R1, 2 × R1,...,63 × R1 into V1.
S--V.D S--VS.D	V1,V2 V1,F0	Compare the elements (EQ, NE, GT, LT, GE, LE) in V1 and V2. If condition is true, put a 1 in the corresponding bit vector; otherwise put 0. Put resulting bit vector in vector-mask register (VM). The instruction S--VS.D performs the same compare but using a scalar value as one operand.
POP	R1,VM	Count the 1s in the vector-mask register and store count in R1.
CVM		Set the vector-mask register to all 1s.
MTC1 MFC1	VLR,R1 R1,VLR	Move contents of R1 to the vector-length register. Move the contents of the vector-length register to R1.
MVTM MVFM	VM,F0 F0,VM	Move contents of F0 to the vector-mask register. Move contents of vector-mask register to F0.

Vectoriser une boucle

- Découper la boucle en morceaux de 64 éléments + un morceau de moins de 64 éléments.
- Trouver les opérations vectorielles.
- Gérer les registres vectoriels.
- Remplacer les opérations conditionnelles par des masques de contrôle des opérations vectorielles.
- Fonctionne bien pour des boucles simples d'algèbre vectorielle.

Exemple: découper en boucles de 64

ÉCOLE POLYTECHNIQUE MONTRÉAL

```
for(i = 0; I < n; i++) Y[i] = a * X[i] + Y[i];
```

```
low = 0;
vl = (n % 64)

for(j = 0; j <= (n / 64); j++) {
  for(i = low; i < (low + vl); i++) {
    Y[i] = a * X[i] + Y[i];
  }
  low = low + vl;
  vl = 64;
}
```

Exemple: conditions

ÉCOLE POLYTECHNIQUE MONTRÉAL

```
for(i = 0; i < 64; i++) {
  if(X[i] != 0) X[i] = X[i] - Y[i];
}
```

```
LV       V1, Rx
LV       V2, Ry
L.D      F0, #0
SNEVS.D V1, F0
SUBVV.D V1, V1, V2
SV       V1, Rx
```

Instructions multimédia

- Un peu de vectorisation opportuniste, extensions Intel x86.
- MMX en 1996, registre FP 64 bits réutilisé pour 8 opérations de 8 bits ou 4 opérations de 16 bits.
- SSE en 1999, 2001, 2004, 2007, registres 128 bits pour 16 opérations de 8 bits, 8 de 16 bits, 4 de 32 bits, 2 de 64 bits, 4 de 32 bits FP, 2 de 64 bits FP.
- AVX en 2010, registres 256 bits, opérations sur 4 éléments FP double précision (64 bits) ou sur plus d'élements de 8, 16 ou 32 bits.

Les processeurs graphiques

ÉCOLE POLYTECHNIQUE MONTRÉAL

- Puissants et peu dispendieux, super vecteur à rabais?
- Section de code parallélisable (grid) décomposée en bloc de fils (thread block).
- Chaque bloc est exécuté sur un processeur SIMD du GPU.
- Un processeur SIMD possède plusieurs (e.g. 32) ALU (thread processor) et gère plusieurs fils simultanément.
- Le processeur SIMD choisit la prochaine instruction du prochain fil prêt et l'exécute sur ses 32 ALU.

La mémoire sur les GPU

ÉCOLE POLYTECHNIQUE MONTRÉAL

- Mémoire privée: pour chaque ALU (résultats intermédiaires).
- Mémoire locale: mémoire locale à chaque processeur SIMD.
- Mémoire GPU: mémoire accessible à tous les processeurs SIMD de même qu'au processeur hôte.
- Un circuit filtre les accès mémoire pour regrouper les accès à des adresses consécutives et les faire en pipeline.

Les instructions PTX

ÉCOLE POLYTECHNIQUE MONTRÉAL

- Programmes en PTX (NVidia), convertis en code machine pour la carte spécifique au moment de l'exécution.
- Opérations arithmétiques (s32, u32, f32, s64, u64, f64), transcendentales (sqrt, sin...), logiques, accès mémoire, opérations atomiques, contrôle (branch, call, ret, bar, exit).
- Toutes les instructions peuvent être conditionnelles et dépendre d'un bit d'activation dans un registre prédicat.
- Lorsque certains ALU font une branche IF, les autres sont inactifs et sont réactivés pour le ELSE.

Discussion sur les GPU

ÉCOLE POLYTECHNIQUE MONTRÉAL

- Une instruction vectorielle avec un élément par ALU, plutôt qu'une instruction vectorielle par ALU pendant 64 cycles en pipeline.
- Les nombreux fils (comme le Hyperthreading) compensent pour la latence d'accès mémoire.
- Modèle de programmation assez contraignant, plus difficile à optimiser.
- Peu d'outils pour analyser la performance ou déboguer.
- Matériel puissant et peu dispendieux.