Université d'Angers — Département Informatique
Faculté des Sciences — L3 Informatique, 07/08
Module : *Architecture des Ordinateurs* — Jean-Michel Richer

Examen
Architecture des Ordinateurs
Aucun document autorisé

Exercice 1 -(7 pts) - On considère le programme C suivant pour lequel n, qui représente la taille maximale des tableaux a et b, est un multiple de 4 :

```
void compute(int n, int *v, int *a, int *b) {
  int i, sum;
  sum = 0;
  for (i=0; i<n; ++i) {
    sum += a[i] + b[i];
  }
  *v = sum / N;
}
```

1. traduire ce sous-programme en assembleur Pentium (on ne donnera que la section de code)
2. le modifier en utilisant les instructions SSE (MOVDQA, MOVDQU, MOVD, PADDD, PXOR, PAND, PSRLDQ, ...) afin de vectoriser le calcul de sum.

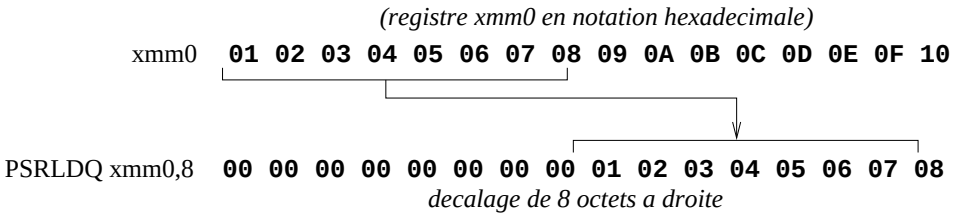
Dans la liste suivante qui indique le comportement de quelques instructions SSE (toutes ne sont pas à utiliser), on notera que :

- **xmmDst** et **xmmSrc** représentent l'un des 8 registres $xmm0$ à $xmm7$
- **mem128** représente un emplacement mémoire dont l'adresse est un multiple de 16
- **mem** représente un emplacement mémoire
- **r32** représente un registre 32 bits
- **imm** représente une constante entière comprise entre 0 et 255

Liste d'instructions SSE

- **MOVDQA xmmDst, xmmSrc/mem128** : charge les données contenues soit dans un registre SSE (xmmSrc) ou un emplacement mémoire vers un registre SSE (xmmDst)
- **MOVDQA xmmDst/mem128, xmmSrc** : charge les données contenues dans un registre SSE (xmmSrc) vers soit un emplacement mémoire ou un registre SSE (xmmDst)
- **MOVDQU xmmDst, xmmSrc/mem** : charge les données contenues soit dans un registre SSE (xmmSrc) ou un emplacement mémoire vers un registre SSE (xmmDst)
- **MOVDQU xmmDst/mem, xmmSrc** : charge les données contenues dans un registre SSE (xmmSrc) vers soit un emplacement mémoire ou un registre SSE (xmmDst)
- **PAND xmmDst, xmmSrc** : réalise un ET logique entre deux registres SSE, le résultat est placé dans xmmDst
- **POR xmmDst, xmmSrc** : réalise un OU logique entre deux registres SSE, le résultat est placé dans xmmDst
- **PXOR xmmDst, xmmSrc** : réalise un OU EXCLUSIF logique entre deux registres SSE, le résultat est placé dans xmmDst

- **MOVD r32, xmmSrc** : déplace les 32 bits de poids le plus faible du registre SSE xmmSrc vers le registre général spécifié
- **PADDD xmmDst, xmmSrc** : réalise l'addition en parallèle des 4 entiers 32 bits stockés dans xmmSrc avec les 4 entiers 32 bits stockés dans xmmDst, le résultat est placé dans xmmDst
- **PSRLDQ xmmDst, imm** : réalise un décalage à droite de imm octets du registre xmmDst



Exercice 2 -(4 pts) - Questions de cours - Expliquez brièvement en une dizaine de lignes (c'est à dire : qu'est ce que c'est, à quoi ça sert, comment ça fonctionne) vous pouvez vous aider d'un schéma :

1. expliquez le principe du pipeline
2. expliquez le principe d'un cache n-way associative

Exercice 3 -(4 pts) - Soit la fonction booléenne suivante :

$$f(A, B, C, D) = (0 - 2, 4, 5, 7, 8, 10, 11, 13 - 15)$$

1. donnez l'expression algébrique de f en fonction de A, B, C, D
2. simplifiez la fonction en utilisant un tableau de Karnaugh

Exercice 4 -(5 pts) Cache - Soit une machine dotée d'un bus d'adresses de 10 bits et d'un bus de données de 8 bits. Cette machine dispose également d'une mémoire cache à accès direct de 8 entrées. On rappelle que pour un cache à accès direct, une donnée située à une adresse a est toujours placée dans la même ligne de cache. On stockera une partie de l'adresse ainsi que la donnée associée à cette adresse.

1. quelle taille mémoire le processeur peut-il adresser ?
2. quelle est la taille du cache ?
3. on accède successivement aux données D_1, D_2, ..., D_{12} dont les adresses, exprimées en binaire, sont :

Adresse	Donnée
0000000000	D_1
0000010100	D_2
0000110110	D_3
0001110000	D_4
0000001100	D_5
0000100100	D_6

Adresse	Donnée
0000111000	D_7
0100110010	D_8
0100111000	D_9
1100110111	D_{10}
0010110011	D_{11}
1100110010	D_{12}

Déterminer l'état du cache (adresses et données stockées dans le cache) lorsque l'on aura accèdé à la dernière donnée D_{12}.