Architecture des Ordinateurs

2ème Année Licence Informatique

Mémoire Cache

Travaux Dirigés #7

1. Contrôle préalable

Cette partie est conçue comme une vérification pour vous permettre de déterminer si vous comprenez les concepts abordés en cours ou non. Veuillez répondre par « Vrai » ou « Faux » aux questions suivantes et inclure une explication :

1.1. Pour la même taille de cache et la même taille de bloc, un cache 4-associatif aura moins de bits d'index qu'un cache direct.

1.2. Lorsque le cache est plein, tout défaut de cache qui se produit est un « défaut de capacité ».

1.3. Augmenter la taille du cache en ajoutant plus de blocs améliore toujours le taux de réussite.

2. Comprendre T/I/O

L'accès et la manipulation de données avec les caches repose sur une décomposition ingénieuse de l'adresse mémoire en trois différents champs de bits :

- **I**ndex – L'indice de la ligne de cache où le bloc mémoire sera placé

$$Nombre\ de\ bits = log_2(Nombre\ de\ lignes\ de\ cache)$$

- **O**ffset – La position de l'octet dans le bloc mémoire

$$Nombre\ de\ bits = log_2(taille\ en\ octet\ du\ bloc\ de\ cache)$$

- **Tag** (étiquette en français) – Utilisé pour distinguer différents blocs de mémoire susceptible d'utiliser la même ligne de cache (c.-à-d. même numéro d'indice).

$$\begin{aligned} Nombre\ de\ bits = {} & Nombre\ de\ bits\ de\ l'adresse\ mémoire \\ & - Nombre\ de\ bits\ du\ champ\ Index \\ & - Nombre\ de\ bits\ du\ champ\ Offset \end{aligned}$$

Ainsi, nous pouvons vérifier l'équation suivante :

$$\begin{aligned} log_2(taille\ en\ octet\ de\ la\ mémoire) = {} & Largeur\ de\ l'adresse\ mémoire \\ = {} & Nombre\ de\ bits\ du\ champ\ \text{Tag} \\ & + Nombre\ de\ bits\ du\ champ\ \text{Index} \\ & + Nombre\ de\ bits\ du\ champ\ \text{Offset} \end{aligned}$$

Une autre égalité utile à retenir est :

$$Taille\ du\ cache = Taille\ du\ bloc \times Nombre\ de\ lignes\ de\ cache$$

Remarque : Dans le cours, le champ « Offset » est décomposé davantage en deux sous parties : Le champ « mot » indiquant la position du mot dans le bloc mémoire et le champ « octet » donnant le numéro d'octet dans le mot actuel. Dans ce TD, nous nous limiterons à la décomposition « Tag / Index / Offset » de l'adresse mémoire.

2.1. Supposons un cache direct avec une capacité de 32 octets et une taille de bloc de 8 octets. Pour une adresse mémoire sur 32 bits, quels bits devons-nous sélectionner pour trouver la ligne dans cache à utiliser ?

2.2. Quels bits correspondent au champ d'étiquette « Tag » ? quid du champ « Offset » ?

2.3. Indiquez pour chacun des accès suivants à la mémoire si c'est un succès de cache (S), un échec de cache (E) ou un échec de cache avec remplacement (R). Indication : dessiner le cache peut vous aider à voir les remplacements plus clairement.

Adresse	**T** / **I** / **O**	**S**uccès, **E**chec, **R**emplacement
0x00000004		
0x00000005		
0x00000068		
0x000000C8		
0x00000068		
0x000000DD		
0x00000045		
0x00000004		
0x000000C8		

3. Associativité

Pour minimiser les défauts de cache pour cause de capacité dans un cache direct, on pourrait augmenter simplement la taille des blocs dans le cache. Cela assurera une meilleure exploitation du principe de localité spatiale, mais le nombre de défauts de cache dus à des appels répétitifs de fonctions par exemple, ne seront pas réduits – augmenter la taille des blocs dans un cache direct n'assure pas une meilleure exploitation du principe de la localité temporelle.

Pour prendre en compte ce principe dans le design du cache, nous permettons d'associer (d'où le nom associativité) à un bloc mémoire plusieurs emplacements possibles dans le cache. Ainsi, un cache est dit N-associatif lorsque chaque bloc de mémoire peut aller dans N emplacements distincts dans le cache. Un cache complètement associatif signifie que chaque bloc de mémoire peut aller n'importe où dans le cache.

Pour un cache N-associatif, nous avons la règle :

$$N \times Nombre\ de\ lignes\ de\ cache = Nombre\ de\ blocs\ du\ cache$$

5.1 Soit un cache 2-associatif avec une politique de remplacement LRU et un mémoire adressable sur 8 bits. La taille du cache est de 32 octets et la taille des blocs de cache est de 8 octets. Indiquez pour chacun des accès suivants à la mémoire si c'est un succès de cache (S), un échec de cache (E) ou un échec de cache avec remplacement (R).

Adresse	T / I / O	Succès, Echec, Remplacement
0b0000 0100		
0b0000 0101		
0b0110 1000		
0b1100 1000		
0b0110 1000		
0b1101 1101		
0b0100 0101		
0b0000 0100		
0b1100 1000		

5.2 Quel est le taux de réussite de nos accès ci-dessus ?

4. Les trois causes de défauts de Cache

Réexaminez les questions 2.3 et 3.1 et classez chaque défaut de cache comme l'un des 3 types d'échecs décrits ci-dessous :

- **Défauts de première référence (compulsory misses)** : Un échec qui doit se produire quand un bloc mémoire est référencé pour la première fois. On peut réduire les échecs de première référence en ayant des lignes de cache plus longues (blocs plus gros). On peut également précharger des blocs au préalable en utilisant un circuit spécial qui essaie de deviner les prochains blocs susceptibles d'être requis.
- **Défauts de capacité (capacity misses)** : Ces défauts sont dû au fait que le cache ne peut pas contenir tous les blocs référencés pendant l'exécution du programme. Le nombre de ces défauts peut être réduit en augmentant la taille du cache.
- **Défauts de conflit (conflict misses)** : Ces défauts interviennent en plus des deux précédents types. Un bloc a pu être chargé puis enlevé du cache car d'autres blocs avec le même indice ont été chargés. Le nombre de ces défauts peut être réduit en augmentant l'associativité du cache.

5. Analyse de code

Soit le code en C ci-dessous, s'exécutant sur un système équipé de 1 Mo de mémoire et un cache direct de 16 Ko organisé en blocs de 1Ko.

```
1  #define NUM_INTS 8192     // 2^13
2  int A[NUM_INTS];          // A est alloué à l'adresse 0x10000
3  int i, total = 0;
4  for (i = 0; i < NUM_INTS; i += 128) {
5     A[i] = i;
6  }
7  for (i = 0; i < NUM_INTS; i += 128) {
8     total += A[i];
9  }
```

6.1 Quelle est la taille en bits de l'adresse mémoire de ce système ?

6.2 Donnez la décomposition « T / I / O » de l'adresse ?

6.3 Calculez le taux de réussite pour la ligne 5 du code C

6.4 Calculez le taux de réussite pour la ligne 8 du code C

6. Temps d'Accès Mémoire Moyen

Le Temps d'Accès Mémoire Moyen (*anglais* : Average Memory Access Time – AMAT) est défini par la règle suivante :

$$\text{AMAT} = \textit{temps d'accès succès} + \textit{taux d'échec} \times \textit{pénalité d'échec}$$

temps d'accès succès = temps d'accès à une donnée résidant dans le cache
taux d'échec = nombre de défaut de cache / nombre d'accès cache.

Pour un système de caches en hiérarchie, il existe deux mesures de taux d'échec pour chaque niveau de cache :

- **Taux global d'échec** : Le nombre d'accès manqués à ce niveau divisé par le nombre total d'accès **au système de cache**.
- **Taux local d'échec** : Le nombre d'accès manqués à ce niveau divisé par le nombre total d'accès à **ce niveau de cache**.

6.1 Dans un cache L2 (c.-à-d. de niveau 2), il y eu 20 échecs sur un nombre total d'accès de 100. Quel est le taux global d'échec ?

6.2 Si un cache L1 (c.-à-d. de niveau 1) a un taux d'échec de 50%, quel est le taux d'échec local du cache L2 ?

Pour les questions suivantes, considérez un système ayant les caractéristiques suivantes :

- Un cache L1 avec un *temps d'accès succès* égale à 2 cycles d'horloge et un taux local d'échec de 20%.
- Un cache L2 avec un *temps d'accès succès* de 15 cycles et un taux global d'échec de 5%.
- Une mémoire principale avec un temps d'accès de 100 cycles d'horloge.

6.3 Quel est le taux local d'échec du cache L2 ?

6.4 Donnez le Temps d'Accès Mémoire Moyen (AMAT) du système

6.5 Nous aimerions réduire l'AMAT du système à 8 cycles d'horloge ou moins en ajoutant un cache de niveau 3 (L3). Si le cache L3 a un taux d'échec local de 30%, quel est le plus grand temps de réponse que ce cache peut avoir ?