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 ou non les concepts abordés en cours. 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.
Vrai ! Un cache direct doit indexer chaque ligne du cache, tandis qu'un cache 4-associatif doit indexer chaque ensemble de 4 lignes. Le cache 4-associatif aura donc 2 bits d'index de moins que le cache direct.

1.2. Lorsque le cache est plein, tout défaut de cache qui se produit est un « défaut de capacité ».
Faux ! Lorsque le cache est plein, vous pouvez toujours avoir des défauts de première référence (lorsqu'un bloc de données est mis dans le cache pour la première fois) et des défauts de conflit

1.3. Augmenter la taille du cache en ajoutant plus de blocs améliore toujours le taux de réussite.
Faux ! Cela dépend des caractéristiques du programme exécuté. Prenons l'exemple d'un programme qui se compose uniquement d'une boucle qui parcourt un tableau une seule fois avec un pas supérieur à un bloc de cache (par exemple, la taille du bloc de cache est 8 octets, mais nous avons un tableau d'entiers et accédons chaque quatrième élément, donc nous avons un pas de 16 octets). Les échecs de cache seront dans ce cas des défauts de première référence, et il n'y a aucun moyen de réduire le nombre de ces échecs simplement en ajoutant plus de blocs au cache.

2. Comprendre T/I/O

L'accès et la manipulation de données avec les caches repose sur une lecture/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 l'indice de l'octet dans le mot actuel. Dans ce TD, nous nous limitons 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 à utiliser dans cache ?
Puisqu'il s'agit d'un cache direct (c-à-d. 1-associatif), il suffit de déterminer le nombre de lignes de ce cache pour trouver la plage d'indices. En utilisant l'égalité ci-dessus, nous pouvons voir que $nombre\_de\_lignes\_de\_cache = taille\ du\ cache\ /\ taille\ de\ bloc$. Ainsi, notre cache possède $32/8 = 4$ lignes et nous avons ainsi besoin de $\log_2(4) = 2$ bits d'« index » pour différencier les 4 lignes de cache.

Afin de déterminer où se trouvent exactement les bits du champ « Index » dans une adresse mémoire, nous devons calculer le nombre de bits du champ « Offset » et le nombre de bits de l'étiquette « Tag ». Le nombre de bits de l'Offset dépend seulement de la taille du bloc, donc comme nos blocs sont de taille 8 octets, nous avons besoin de $\log_2(8) = 3$ bits pour différencier les 8 octets du bloc, nous utiliserons donc 3 bits pour le champ « Offset ».

Les bits du champ « Offset » sont les moins significatifs dans une adresse mémoire. Les bits du champ « Index » sont l'ensemble des bits les plus significatifs suivants. En désignant le bit 31 comme bit le plus significatif (MSB, à gauche), et le bit 0 comme bit le moins significatif (LSB, à droite), avoir 3 bits d'Offset signifie que les bits du champ « Index » commencent au bit 3, et donc nous utilisons les bits 3 et 4 pour l'indice.

2.2. Quels bits correspondent au champ d'étiquette « Tag » ? quid du champ « Offset » ?
Le champ « Offset » correspond aux 3 bits les moins significatifs, donc en réutilisant la convention de la question précédente, nous utilisons les bits 0, 1 et 2 pour le champ « Offset ». Enfin, l'étiquette (c-à-d. le champ « Tag ») est définie par les bits de poids fort restant : les bits de 5 à 31.

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

Adresse	**T** / **I** / **O**	**S**uccès, **E**chec, **R**emplacement
0x00000004	0 / 0 / 4	E, première référence
0x00000005	0 / 0 / 5	S
0x00000068	3 / 1 / 0	E, première référence
0x000000C8	6 / 1 / 0	R, première référence
0x00000068	3 / 1 / 0	R, conflit
0x000000DD	6 / 3 / 5	E, première référence
0x00000045	2 / 0 / 5	R, première référence
0x00000004	0 / 0 / 4	R, capacité
0x000000C8	6 / 1 / 0	R, capacité

Notez que la distinction E et R ici est purement théorique. Le cache ne se comporte pas différemment dans ces cas.

3. Associativité

Pour minimiser les défauts de cache pour cause de « capacité insuffisante » dans un cache direct, nous pourrions 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 par exemple à des appels répétitifs de fonctions ne seront pas réduits – l'augmentation de 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 une mémoire adressable sur 8 bits. La taille du cache est de 32 octets et la taille d'un bloc 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) (entrée invalide) ou un échec de cache avec remplacement (R).
Comme le cache est 2-associatif, il y a 2 blocs dans une ligne de cache. Les tailles du cache et d'un bloc mémoire sont identiques à l'exemple précédent, nous avons donc 4 blocs. Ainsi, il y'a $4/2 = 2$ lignes de cache. Nous avons besoin de $\log_2(2) = 1$ bit pour différencier les 2 lignes, c-à-d. 1 bit pour le champ « Index ». Comme la taille du bloc est identique à la question précédente, le champ « Offset » occupe donc 3 bits et le reste des bits sont nos bits d'étiquette.

Adresse	T / I / O			Succès, Echec, Remplacement
0b0000 0100	0	0	4	E, première référence
0b0000 0101	0	0	5	S,
0b0110 1000	6	1	0	E, première référence
0b1100 1000	12	1	0	E, première référence
0b0110 1000	6	1	0	S,
0b1101 1101	13	1	5	R, première référence
0b0100 0101	4	0	5	E, première référence
0b0000 0100	0	0	4	S,
0b1100 1000	12	1	0	R, capacité

Encore une fois, les bits 0, 1 et 2 de l'adresse définissent les bits du champ « Offset », le champ « Index » est défini en revanche par un seul bit (le bit 3), les bits 4 à 7 étant les bits d'étiquette.

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

$\frac{3\ succés}{9\ accès} = \frac{1}{3}$ taux de réussite

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. Nous pouvons réduire les échecs de première référence en ayant des lignes de cache plus longues (blocs plus gros). Nous pouvons également précharger des blocs au préalable en utilisant un circuit spécial pour prédire 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 déjà chargé en cache est éjecté de ce dernier car un autre bloc avec le même numéro d'indice est requis par le programme. 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 ?
$\log_2(1Mo) = \log_2(2^{20}) = 20$ bits.

6.2 Donnez la décomposition « T / I / O » de l'adresse ?
Offset = $\log_2(1Ko) = \log_2(2^{10}) = 10$ bits.

Index = $\log_2(\frac{16\,Ko}{1\,Ko}) = \log_2(16) = 4$ bits.

Tag = $20 - 4 - 10 = 6$ bits

6.3 Calculez le taux de réussite pour la ligne 5 du code C
Les éléments du tableau d'entiers A sont accédés avec un pas de $128 \times \text{sizeof(int)} = 128 \times 4 = 512$ octets. Comme l'adresse de début de A coïncide avec le début d'un bloc (bloc n°64 dans ce cas), Cela signifie que chaque deux accès au tableau A se font à un même bloc mémoire. Le premier des accès dans chaque bloc est de ce fait un défaut de cache de première référence, mais le second accès sera un succès. Ainsi, nous aurons un taux de réussite de 50%.

6.4 Calculez le taux de réussite pour la ligne 8 du code C
La taille du tableau A est $8192 \times \text{sizeof(int)} = 2^{15} = 32\,Ko$. C'est deux fois la taille de notre cache ! Cela signifie qu'à la fin de la première boucle, nous allons avoir la seconde moitié du tableau A stockée en cache. Cependant, comme la seconde boucle commence à nouveau depuis le début de A, nous ne pourrons réutiliser aucune des données de cache importées dans la première boucle et devons recommencer depuis le début. Ainsi, notre taux de réussite sera le même que celui de la ligne 5 puisque nous accédons à la mémoire exactement de la même manière dans la seconde boucle. C.-à-d. un taux de réussite de 50%.

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 (c.-à-d., nombre d'accès RAM) divisé par le nombre total d'accès à l'ensemble du **système de caches**.
- **Taux local d'échec** : Le nombre d'accès manqués par le cache de niveau *n* divisé par le nombre total d'accès à **ce niveau** de cache.

6.1 Dans un système de caches sur deux niveaux (c.-à-d. caches de niveau 1 et 2), il y eu 20 échecs sur un nombre total d'accès de 100. Quel est le taux global d'échec ?
$\frac{20}{100} = 20\%$ taux d'échec global.

6.2 Si le 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 ?
$\frac{20}{50\% \times 100} = 40\%$ taux d'échec local. Rappelons que le cache de niveau $n+1$ est accédé seulement dans le cas où le cache de niveau n subit un échec. Donc, si le cache L1 a un taux d'échec de 50% cela signifie que nous accédons au cache L2 50 fois.

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 ?
$\frac{taux\ global\ d'échec}{taux\ local\ d'échec\ de\ L1} = \frac{5\%}{20\%} = \frac{1}{4} = 25\%$

6.4 Donnez le Temps d'Accès Mémoire Moyen (AMAT) du système
$AMAT = 2 + 20\% \times 15 + 5\% \times 100 = 10\ cycles$, en utilisant les taux globaux d'échec. Alternativement, $AMAT = 2 + 20\% \times (15 + 25\% \times 100) = 10\ cycles$.

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 devrait avoir ?
Soit S = *temps d'accès succès* du cache L3. En utilisant l'équation AMAT, nous avons

$$2 + 20\% \times \left(15 + 25\% \times (S + 30\% \times 100)\right) \leq 8$$

Résoudre l'inégalité ci-dessus donne : $S \leq 30\ cycles$. Donc, le plus grand temps de réponse du cache L3 est de 30 cycles.