

Technologie mémoire, mémoire cache

M. Dubacq

S1D 2009

1 Les types de mémoire

Objectif : *Connaître les qualificatifs des mémoires, connaître les technologies existantes*

1. Pour tous les types de mémoire suivants, donnez les qualificatifs applicables ainsi que les ordres de grandeur des paramètres numériques pour la capacité (usuelle), la taille physique, la latence : DRAM, ROM, SRAM, disque dur, disquette HD, DVD, cédérom, bande magnétique. *DRAM : semi-conducteur, volatile, effaçable, 1 Go, 5 cm, 50 ns*
ROM : semi-conducteur, non-volatile, non-effaçable, 100 ko, 1 cm, 100 ns
...à finir

2 Overclocking et mémoire

L'*overclocking* est une technique qui consiste en l'accélération de l'horloge d'un processeur pour obtenir des performances meilleures. Toutefois, la cadence du bus de données, du northbridge et de la mémoire sont toutes déterminées à partir de la cadence du processeur en divisant par un entier. Cet entier est, sur certaines cartes-mères, réglable.

Une RAM est prévue pour fonctionner à 333 MHz. La vitesse du processeur qui va avec est de 3 GHz. En fait, cette RAM est capable de fonctionner sans problèmes jusqu'à 400 MHz, et le processeur peut être accéléré jusqu'à 10% sans défauts (il peut être accéléré uniquement par paliers de 100 MHz).

1. Expliquez pourquoi accélérer modérément la cadence du processeur peut amener à des meilleures performances, et pourquoi l'accélérer énormément peut ne pas fonctionner du tout. *Le processeur continue son cycle d'instruction en fonction de l'horloge, et donc l'exécute plus rapidement. Un cycle d'instruction plus rapide, c'est au total plus d'instructions par seconde. Mais pour que ça marche, il faut que les temps de réaction des circuits électroniques ne soient pas trop grands pour que les communications prévues lors de la conception du chemin de donnée aient le temps de se faire. Si on multiplie par 100 la fréquence, il y a de bonnes chances que les circuits combinatoires n'aient plus le temps de se stabiliser avant la fin du cycle d'horloge.*
2. Quel est le multiplicateur utilisé pour obtenir la cadence nominale? *Le multiplicateur est 3 GHz/333 MHz, soit de 9.*
3. Peut-on réaliser une accélération de la cadence du processeur qui ne crée aucun défaut sans changer le multiplicateur? Que deviendrait la vitesse de la mémoire si on le faisait? *Si on garde le multiplicateur de 9, on peut faire monter le processeur à 3300 MHz, et dans ce cas, la vitesse de la mémoire devient 366 MHz, soit une augmentation de 10% environ (aussi).*
4. Peut-on avoir une accélération du processeur et de la mémoire qui privilégie un peu plus la mémoire? *Si on garde le multiplicateur, non. Mais on peut changer le multiplicateur (par exemple mettre 8). On accélère alors le processeur de 200 MHz, ce qui donne un processeur à 3200 MHz, et une mémoire à 400 MHz. L'amélioration de la mémoire est alors de 20%, et celle du processeur de 7%. Selon qu'on ait souvent besoin de la mémoire ou pas, une solution ou l'autre pourra être un peu plus avantageuse (la deuxième l'est souvent, en pratique).*

3 Mémoire cache

On se place dans le cas d'un processeur qui utilise un adressage de 4 Gio de mémoire, avec 8 kio de mémoire cache. Cette taille de mémoire cache ne changera pas dans tout l'exercice. On utilisera des mots-mémoires de 32 bits.

1. Quelle est la taille d'une adresse (en bits)?

32 bits

2. On utilise les données suivantes :
 - temps d'accès à la mémoire cache : 5 ns,
 - latence d'accès à la mémoire principale : 40 ns ; mémoire,
 - temps de cycle en mode page : 10 ns par mot mémoire ; supplémentaire ;
 Quel est le temps nécessaire pour consulter un mot mémoire en cas de succès de cache? Et pour seulement un octet?

5 ns, pour un mot mémoire ou un octet.

3. Si la taille d'un bloc de mémoire est fixée à 512 bits, en cas de défaut de cache, quel est le temps de pénalité ?

512 bits = 16 mots mémoires, donc $16 \times 10 + 40 = 200$ ns.

4. Même question si la taille d'un bloc de mémoire est fixée à 256 bits.

256 bits = 8 mots mémoires, donc $8 \times 10 + 40 = 120$ ns.

5. On a le choix entre trois modes de construction de cette mémoire : par cache direct de 256 bits par bloc (temps d'accès 3 ns), par cache associatif par ensemble à quatre voies de 256 bits par bloc (temps d'accès 4 ns), par cache associatif mais avec 512 bits par bloc (temps d'accès 5 ns).

Des expériences montrent que pour le genre d'applications que l'on va utiliser sur cette machine, dans la première solution, on a un taux de succès cache de 75%, dans la deuxième un taux de succès de 80% et dans la troisième un taux de succès de 87%.

Quel est la solution la plus intéressante ? Justifiez en exhibant le temps moyen d'accès à la mémoire dans les trois cas.

Premier cas : $5 + 0,25 \times 120 = 35$ ns. Deuxième cas : $5 + 0,2 \times 120 = 29$ ns. Troisième cas : $5 + 0,13 \times 200 = 31$ ns. Il est donc meilleur de prendre la deuxième solution.

6. Dans le cas de la première solution, décrivez comment on décompose une adresse (nombre de bits en particulier). Dites aussi combien de blocs différents peuvent résider en mémoire cache.

La sélection octets s'étale sur 256 bits soit 32 octets, soit 5 bits pour sélectionner un octet. Il peut y avoir jusqu'à $8192/32 = 256$ blocs différents en mémoire, qui forment 8 bits d'index cache. Les $32 - 8 - 5 = 19$ bits restants servent à identifier le bloc.

7. Dans le cas de la deuxième solution, décrivez comment on décompose une adresse (nombre de bits en particulier). Dites aussi combien de blocs différents peuvent résider en mémoire cache, ainsi que le nombre de blocs par voie.

La sélection octet s'étale toujours sur 5 bits, mais il n'y a plus que 64 index caches différents, car ils sont répartis en quatre voies (soit bien 256 blocs au total). Donc, il y a 6 bits d'index cache et $32 - 6 - 5 = 21$ bits d'identifiant de bloc.

8. Dans le cas de la troisième solution, dites comment se décompose une adresse.

6 bits d'index cache, et donc $32 - 6 = 26$ bits d'identifiant de bloc.

9. On se place dans le cadre de la troisième solution (cache associatif). On veut accéder aux mots-mémoires qui vont de l'adresse $0x40000000$ à l'adresse $0x40003FC$ (inclus). Dans l'hypothèse la plus pessimiste possible, dites combien il y a de défauts de cache en expliquant pourquoi.

Lecture de $4 \times 16 \times 4$ mots = 256 mots = 32 blocs consécutifs, qui commencent à un début de bloc. Donc, il y a au pire 32 défauts de cache.

10. Quelles sont les politiques possibles d'écriture en cache en cas de succès ?

On peut faire de l'écriture immédiate ou de l'écriture différée, c'est-à-dire au moment du remplacement dans la mémoire cache du bloc.

4 Gestion mémoire de matrices

Une matrice est représentée en C par un tableau de taille N^2 de `float` (c'est-à-dire des nombres codés selon le standard IEEE 754).

Dans l'ordinateur que nous examinons, les blocs de mémoire cache de données (cache associatif) sont de 256 mots de 4 octets, avec une taille totale de mémoire cache de 2 Mo. On veut faire une multiplication de matrices.

Rappel : le coefficient c_{ij} d'une matrice $C = A \times B$ est

$$c_{ij} = \sum_{1 \leq k \leq N} a_{ik} b_{kj}.$$

Le temps de calcul d'un produit de matrice est donc le temps de N^3 multiplications et N^3 additions, ainsi que quelques opérations qui prennent assez peu de temps par rapport au reste (tests de fin de boucle).

1. Quelle est la quantité de mémoire nécessaire pour stocker une matrice de taille 4×4 ? Et une matrice de taille $N \times N$ en général? *16 float à stocker, soit 64 octets. Et $4N^2$ octets en général.*
2. On veut calculer $C = A^2$. À chaque étape de calcul, on réaffiche complètement les deux matrices (A et C, même si C n'est pas encore calculée complètement) . Jusqu'à une certaine taille de matrice, on constate que le temps de calcul ressemble à $t = k \times N^3$, avec un certain k qui dépend de la vitesse du processeur. Mais à partir d'une certaine taille, on constate un très fort ralentissement. Comment peut-on l'expliquer? Quelle est cette taille? *On doit mettre en mémoire la matrice d'origine complète et la matrice résultat. Quand il n'y a plus de place en mémoire cache, les calculs nécessitent d'accéder à des données qui sont en mémoire principale, évacuant des données qui vont resservir un peu après, et donc revenir dans la mémoire cache. Cet effet de ping-pong peut faire perdre beaucoup de temps. La taille à partir de laquelle se déclenche cet effet est quand la mémoire est pleine. $M = 2 \times 4 \times N^2 = 2^21$, donc $N^2 = 2^18$ octets. Donc $N = 2^9$.*
3. On constate que l'affichage d'une matrice entre la taille 8 et la taille 9 est assez important. Quelle peut-être l'explication? *La taille 8 correspond à la taille d'un bloc de la mémoire cache. Lorsqu'on dépasse cette taille, même s'il n'y a pas beaucoup de cases mémoires en plus, il faut charger un autre bloc en mémoire.*