# TD 6 – Mémoires caches

## 1 Rappels de cours

**Exercice 1 –** Rappeler brièvement le rôle d'une mémoire cache, et citer des instructions en assembleur MIPS dont l'exécution fait intervenir le mécanisme de la mémoire cache.

*Les accès mémoires sont en général localisés en temps et en zone mémoire. La mémoire cache permet d'accélérer les accès multiples à des zones contigües de la mémoire.* `lw` *et* `sw` *font intervenir le mécanisme de la mémoire cache.*

**Exercice 2 –** Rappeler brièvement les différences entre cache directe et cache associative en termes de :
- correspondance entre zone de la mémoire centrale et zone de la mémoire cache,
- translation (adresse en mémoire centrale) $\rightarrow$ (adresse en mémoire cache),
- temps d'accès au cache (temps d'un *cache hit*).

| | *directe* | *associative* |
|---|---|---|
| *correspondance entre zones* | *une ligne est dédiée à une partie de la mémoire toujours la même* | *chaque ligne peut accueillir n'importe quelle partie de la mémoire* |
| *translation d'adresses* | *l'index de la ligne est une partie de l'adresse mémoire* | *il faut chercher dans chaque ligne* |
| *temps d'un cache hit* | *une comparaison d'étiquettes* | *plusieurs comparaisons d'étiquettes (il faut parcourir toute la mémoire cache)* |

**Exercice 3 –** Expliquer ce qu'est une mémoire cache associative à $n$ voies.

*$n$ mémoires caches directes regroupées, $n$ lignes ont le même index. L'accès se fait par l'index de ligne, puis comparaison des étiquettes des $n$ lignes de même index.*

## 2 Adresses et alignement

On considère un cache à $2^p$ lignes stockant chacune $2^m$ octets. Le système qui l'utilise adresse la mémoire sur $32$ bits. On appellera *mot* de la mémoire un ensemble de 4 octets.

**Exercice 4 –** Détailler la translation (adresse en mémoire centrale) $\rightarrow$ (ligne de la mémoire cache) en fonction de $m$ et $p$ si :
- le cache est à correspondance directe.
- le cache est totalement associatif.

Donner notamment le découpage en *offset*, *index* et *tag* dans ces deux cas.

*À chaque ligne, le tag est stocké en plus des octets mémoires.*
*cache à correspondance directe :*
*(adresse mémoire) $\rightarrow$ (tag $32 - p - m$ bits), index (tag $p$ bits), offset (tag $m$ bits)).*
*Étant donné une adresse mémoire, l'index indique l'unique ligne dans laquelle peut se trouver le mot. Si les tags correspondent, l'offset indique l'octet, ou le mot, rédérencé par l'adresse.*

*cache associatif :*
*(adresse mémoire) $\rightarrow$ (tag $32 - m$ bits), offset (tag $m$ bits)).*
*Étant donné une adresse mémoire, toutes les lignes sont parcourues en comparant les tags. Si deux tags correspondent, l'offset indique l'octet, ou le mot, rédérencé par l'adresse.*

**Exercice 5 –** On suppose que $m = 7$ : chaque ligne contient 128 octets de la mémoire centrale. Quelle est, dans ce cas, la longueur (en nombre de bits), de l'offset d'une adresse ? Donner l'adresse en mémoire centrale des premiers mots contenus dans les lignes contenant les mots d'adresses :

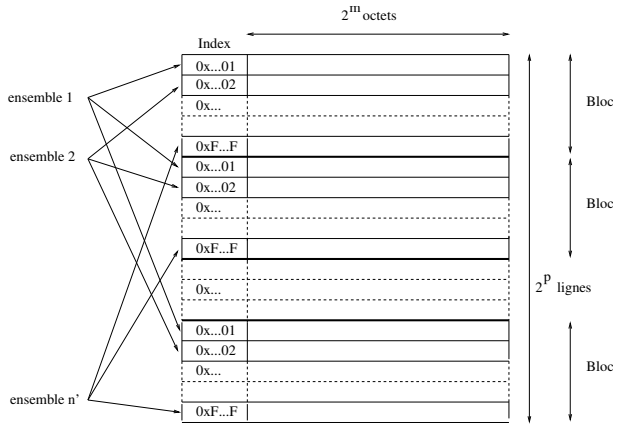


FIGURE 1 – Structuration d'une mémoire cache associative à $n$ voies.

— 0xA23847EC,
— 0x7245E824,
— 0xEEFABCD8.

*Il faut* $7$ *bits pour indexer les* $128 = 2^7$ *octets.*
— *0xA2384780*,
— *0x7245E800*,
— *0xEEFABC80.*

**Exercice 6 –** On suppose que $m = 7$, que $p = 9$ et que le cache est à correspondance directe. Quelle est, dans ce cas, la longueur (en nombre de bits), de l'index d'une adresse ? Donner :
— l'adresse du premier mot de la première ligne, sachant que le tag de cette ligne est 0xABBA,
— l'adresse du deuxième mot de la deuxième ligne, sachant que le tag de cette ligne est le même que précédemment.

*Il faut* $9$ *octets pour indexer les* $2^9$ *lignes.*
— *0xABBA0000*,
— *0xABBA0084.*

# 3 Mémoire associative à $n$ voies

On considère une mémoire cache associative à $n$ voies ayant $2^p$ lignes, stockant chacune $2^m$ octets. Le système qui l'utilise adresse la mémoire sur $32$ bits. On suppose également que $n$ est une puissance de $2$. On appellera un *ensemble* d'une telle mémoire toutes les lignes de même index. En considérant un tel cache comme un regroupement de $n$ caches directes, on appellera *bloc* un de ces caches directs. La figure 1 illustre ces notions.

**Exercice 7 –** On considère que le cache a une capacité de 32 Ko. Combien de lignes peut-il contenir si $m = 5$, $6$ ou $7$, c'est à dire si les longueurs de ligne sont de 32, 64 ou 128 octets ?

*32 Ko* $\simeq 2^{15}$*o. Le cache peut contenir respectivement* $2^{10} = 1024$ *lignes*, $2^9 = 512$ *lignes*, $2^8 = 256$ *lignes.*

**Exercice 8 –** On considère à présent que le cache a une capacité de 16 Ko, et des lignes contenant $128$ octets (*c.à.d* $m = 7$). Combien d'ensembles contient-elle si elle est associative à $2$, $4$ ou $8$ voies (*c.à.d* $n = 2, 4$ ou $8$) ?

*16 Ko* $\simeq 2^{14}$*o. Nombre total de lignes :* $2^{14-7} = 2^7 = 128$ *lignes. Il y a alors respectivement* $64$, $32$ *ou* $16$ *ensembles.*

**Exercice 9 –** On considère à présent que le cache a une capacité de 64 Ko, des lignes de $128$ octets ($m = 7$) et est associatif à $4$ voies ($n = 4$).
— Combien de lignes et d'ensembles possède le cache ?
— Combien d'entrées sont requises dans le tableau d'étiquettes ?
— Combien de bits d'étiquette sont requis pour chaque entrée dans le tableau d'étiquettes ?

— *64 Ko* $\simeq 2^{16}$*o. Nombre total de lignes :* $2^{16-7} = 2^9 = 512$ *lignes. Il y a alors* $128$ *ensembles.*
— *autant que de lignes :* $512$
— *Une étiquette est codée sur* $32 - 7 - 7 = 18$ *bits.*

**Exercice 10 –** Soit une mémoire cache possédant les mêmes caractéristiques que celle de l'exercice précédent. Pour chacune des adresses mentionnées ci-après, indiquez l'étiquette et le numéro de l'ensemble (qui seront examinés afin de déterminer si l'adresse est contenue dans le cache) ainsi que l'octet référencé dans la ligne.

— 0xABC89987
— 0x32651987
— 0x228945DB
— 0x48569CAC

*Il y a* $128$ *octets par lignes, dont l'offset fait* $7$ *bits. Il y a* $128$ *ensembles de lignes, donc l'index fait* $7$ *octets. L'étiquette est donc composée des* $18$ *premiers bits de l'adresse.*

— *0xABC89987 = 10101011 11000100 01010101 01000111 donc l'étiquette est 10101011 11000100 01, l'index 010101 0 et l'offset 1000111.*
— *0x32651987 = 00110010 01100101 00011001 01000111 donc l'étiquette est 00110010 01100101 00, l'index 011001 0 et l'offset 1000111.*
— *0x228945DB = 00100010 10001001 01000101 11011011 donc l'étiquette est 00100010 10001001 01, l'index 000101 1 et l'offset 1011011.*
— *0x48569CAC = 01001000 01010110 10011100 10101100 donc l'étiquette est 01001000 01010110 10, l'index 011100 1 et l'offset 0101100.*

# 4 Taux de hit et temps d'accès

**Exercice 11 –** Supposons qu'une mémoire cache possède un temps d'accès $T_{hit}$ (latence de cache-hit) de 10 ns et un taux de miss $P_{miss}$ de 5 %. Une modification apportée au cache ferait baisser son taux de miss $P_{miss}$ à 3 % mais ferait monter la latence $T_{hit}$ du cache-hit à 15 ns. Donner une condition sur le temps de traitement $T_{miss}$ d'un miss pour que le cache ainsi modifiée offre de meilleures performances (temps d'accès moyen à la mémoire plus court) ?

*On utilise la formule* $T_{moy} = (T_{hit} \times P_{hit}) + (T_{miss} \times P_{miss})$*. Pour que la modification réduise le temps d'accès moyen, il faut :* $(10 \times 0.95) + (T_{miss} \times 0.05) > (15 \times 0.97) + (T_{miss} \times 0.03)$*, c'est à dire* $(T_{miss} \times 0.05) - (T_{miss} \times 0.03) > (15 \times 0.97) - (10 \times 0.95)$ *donc* $(T_{miss} \times 0.02) > (15 \times 0.97) - (10 \times 0.95)$ *et* $T_{miss} > 0.02 * ((15 \times 0.97) - (10 \times 0.95))$ *d'où* $T_{miss} > 252.2$*ns.*
*En gros, quand* $T_{miss}$ *est assez grand (ici* $T_{miss} > 252.2$*), il vaut mieux baisser la probabilité d'un miss, quitte à augmenter la latence d'un hit (par exemple, augmenter le nombre de voies du cache).*

**Exercice 12 –** Une mémoire cache possède un taux de hit de 95 %, des lignes de 128 octets et une latence de cache-hit de 5 ns. La mémoire principale prend 100 ns pour retourner le premier mot (32 bits) d'une ligne puis 10 ns pour retourner chaque mot suivant.

— Quelle est la valeur de $T_{miss}$ pour ce cache ? Nous supposerons que le cache attend que la ligne soit chargée et réexécute ensuite l'opération mémoire en obtenant alors un cache-hit. Nous négligerons le temps requis pour écrire la ligne dans le cache une fois qu'elle a été extraite de la mémoire principale. Nous supposerons également que le cache prend le même temps pour détecter un miss que pour un cache-hit.

  $128$ *octets font* $32$ *mots. Obtenir* $32$ *mots consécutifs en mémoire prend donc* $100 + 31 \times 10 = 410$*ns.* $T_{miss}$ *nécessite donc une lecture dans le cache (*$5$*ns) pour détecter le miss,* $410$*ns pour remplir la ligne du cache, puis* $5ns$ *pour détecter le cache hit, et retourner le mot, donc* $T_{miss} = 420$*ns.*

— Si le fait de doubler la longueur de ligne du cache permet de réduire le taux de miss à 3 %, le temps d'accès moyen à la mémoire s'en trouvera-t-il réduit ?

  *Si les lignes de chache font* $256$*o, soit 64 mots, il faut* $100 + 63 * 10 = 730$*ns pour extraire de la mémoire centrale une ligne de cache. Le temps* $T_{miss}$ *devient donc* $730$*ns. Le temps moyen d'accès à la mémoire devient donc* $(T_{hit} \times P_{hit}) + (T_{miss} \times P_{miss}) = (5ns \times 0.97) + (740 * 0.03) = 27.05$*ns. Dans le cas de la cache initiale, il est de* $(T_{hit} \times P_{hit}) + (T_{miss} \times P_{miss}) = (5ns \times 0.95) + (420 * 0.05) = 25.75$*. Doubler le nombre de lignes du cache, dans ce cas, augmente le temps moyen d'accès.*

# 5 Échecs obligatoires, de capacité et de conflit

**Exercice 13 –** Un programme accède à $1000000$ adresses mémoire. Lorsqu'il tourne sur un système donné, la mémoire cache obtient un taux de miss de 7 %, dont un quart sont des échecs obligatoires, un autre quart des échecs de capacité et la moitié des échecs de conflits.

— Si la seule modification que vous pouvez apporter au cache consiste à augmenter son degré d'associativité, quel est le nombre maximal de miss que vous pouvez espérer éliminer ?

*Le fait d'augmenter le degré d'associativité permettra de réduire le nombre d'échecs de conflits (des miss qui surviennent parce que les lignes de cache se font concurrence pour des emplacements dans le cache), mais n'affectera pas le nombre d'échecs de capacité (les miss qui surviennent parce qu'un programme référence plus de données que ne peut en contenir le cache). En conséquence, le mieux que nous puissions espérer en augmentant le degré d'associativité du cache est d'éliminer tous les échecs de conflit. Le taux de miss étant de 7 % et le programme opérant* $1000000$ *de références mémoire, le nombre total de miss est de* $70000$*. La moitié d'entre eux sont des échecs de conflit. Le nombre maximale que nous pouvons éliminer en augmentant le degré d'associativité du cache est donc de* $35000$*.*

— Si vous avez la possibilité d'augmenter la taille du cache en plus de son degré d'associativité, quel est le nombre maximal de miss que vous pouvez espérer éliminer ?

*En augmentant la capacité ainsi que le degré d'associativité du cache, nous pouvons éliminer les miss de capacité et les miss de conflit à la fois, soit les trois quarts de l'ensemble des miss, pour un total de* $52500$ *miss.*