CS361

Mémoires Caches

Préambule :

Considérons l'architecture suivante : Mémoire principale de 1Giga-octets, Une mémoire Cache 1 Mega octets, le cache est composé de bloc de 256 octets.

Nous considérerons les structures de cache suivantes : (1) Correspondance directe, (2) Totalement associatif, (3) Associatif par ensemble (4blocs)

1. Donner la structure de l'adresse pour chacune des configurations
2. Donner le nombre de blocs dans cette mémoire cache
3. donner le nombre d'ensembles pour chacune des architectures associatives par ensemble.

Exercice 1 : Mémoire Cache

Un cache possède une capacité de 64 Ko, des blocs de 128 octets et un degré d'associativité de 4. Le système contenant le cache utilise des adresses de 32 bits.

1. Combien de blocs et d'ensembles possède le cache ? Combien d'entrées sont requises dans le tableau d'étiquettes (tags) ? Combien de bits d'étiquettes (tags) sont requis pour chaque entrée dans le tableau d'étiquettes (tags) ?

2. Pour chacune des adresses mentionnées ci-après, indiquez le numéro de l'ensemble qui sera examiné afin de déterminer si l'adresse est contenue dans le cache et celui de l'octet référencé dans la ligne de cache : 0xABC89987, 0x32651987, 0x228945DB, 0x48569CAC

Exercice 2 :

Le programme C ci-dessous exécute la multiplication de deux matrices A et B contenant chacune 16 éléments et stocke le résultat dans la matrice C. Les entiers (int) sont codés sur 32 bits et l'adressage se fait à l'octet (i.e. pour un mot de 32 bits on utilise 4 adresses).

```
int A[4][4];
int B[4][4];
int C[4][4];
int i, j, k, acc;

...

for (i = 0; i < 4; i++)
{
    for (j = 0; j < 4; j++)
    {
        acc = 0;
        for (k = 0, k < 4; k++)
        {
            acc = acc + A[i][k] * B[k][j];
        }
        C[i][j] = acc;
    }
}
```

Figure 1: code C multiplication de deux matrices

Les trois matrices A, B et C ont des adresses virtuelles comme illustré par la figure 2 ci-dessous.

A	A[0][0]	B	B[0][0]	C	C[0][0]
A+4	A[0][1]	B+4	B[0][1]	C+4	C[0][1]
A+8	A[0][2]	B+8	B[0][2]	C+8	C[0][2]
A+12	A[0][3]	B+12	B[0][3]	C+12	C[0][3]
A+16	A[1][0]	B+16	B[1][0]	C+16	C[1][0]
A+20	A[1][1]	B+20	B[1][1]	C+20	C[1][1]
A+24	A[1][2]	B+24	B[1][2]	C+24	C[1][2]
A+28	A[1][3]	B+28	B[1][3]	C+28	C[1][3]
A+32	A[2][0]	B+32	B[2][0]	C+32	C[2][0]
A+36	A[2][1]	B+36	B[2][1]	C+36	C[2][1]
A+40	A[2][2]	B+40	B[2][2]	C+40	C[2][2]
A+44	A[2][3]	B+44	B[2][3]	C+44	C[2][3]
A+48	A[3][0]	B+48	B[3][0]	C+48	C[3][0]
A+52	A[3][1]	B+52	B[3][1]	C+52	C[3][1]
A+46	A[3][2]	B+46	B[3][2]	C+46	C[3][2]
A+60	A[3][3]	B+60	B[3][3]	C+60	C[3][3]

Figure 2: Adresses virtuelles des matrices A, B et C (offset donné en décimal)

L'architecture que l'on considère ici comporte un cache pour les données et un cache pour les instructions. La mémoire cache est adressée par les adresses physiques, i.e. les adresses virtuelles sont traduites avant d'accéder en cache.

La mémoire virtuelle est basée sur des pages de 4kbytes et les adresses physiques ont une largeur de 32 bits.

1. Quelles sont les accès en mémoire de donnée fait par ce code ?
2. On considère un cache à correspondance directe d'une taille de 256 bytes (octets) avec des blocs de 16 bytes (octets). Dessiner la structure de ce cache en précisant comment est décomposée l'adresse physique pour accéder à une donnée.
3. Expliquer pourquoi il est possible de déduire la séquence d'accès au cache sans connaître le contenu de la table des pages.
4. On suppose que A=0x400, B=0x440 et C=0x480. Quel est le taux d'échec du cache lors du calcul de C ?on considère que le cache est vide avant le premier « for » du programme.
5. Maintenant on suppose que A=0x400, B=0x800 et C=0x480. Quel est le taux d'échec du cache lors du calcul de C ? Le cache est vide avant le premier accès mémoire.
6. Donner la structure la plus simple pour une mémoire cache de 256 bytes qui donne toujours le même taux de miss que celui calculé en 4 pour toutes les adresses possible des matrices A, B et C.