Architecture des microprocesseurs

Mémoire cache du processeur UltraSPARC II

Le processeur UltraSPARC II de chez SUN Microsystems est un processeur RISC 64 bits. Si les données manipulées ont une taille de 64 bits, les instructions sont codées sur 32 bits. Quant aux adresses, leur taille est fixée à 44 bits. Ce processeur possède en interne un cache de données et un cache d'instructions de type différent.

- Le cache de données est un cache de 16 KOctets à correspondance directe. Chaque ligne est composée de 32 octets (ce qui correspond à 4 mots de données).
- Le cache d'instructions est un cache de 16 KOctets, associatif à 2 groupes. Chaque ligne comprend 32 octets (ce qui représentent 8 instructions).

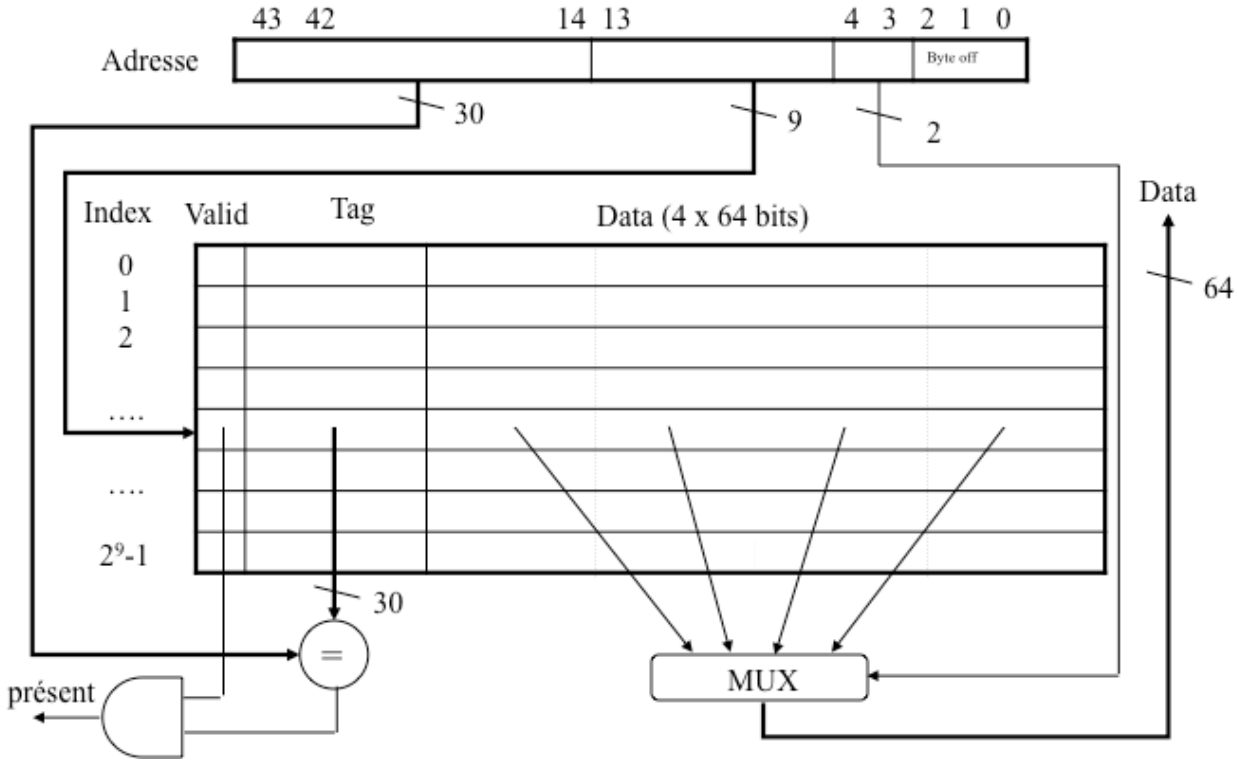
Le système étudié comprend une mémoire principale adressable par octet. Dans cette mémoire sont stockées les données et les instructions.

Partie A : cache de donnée

1) Indiquer la taille totale du cache.

32 octets par ligne = 4 mots de 64 bits par ligne
16 KO = 16 x 1024 octets = 16384 octets
soit (16384/32) = 512 lignes = 2^9 lignes
Taille d'une ligne = taille des données + 1 bit de validité + TAG
TAG : taille des adresses – 9bits qui indique la ligne – 2 bits qui indiquent la donnée sur la ligne – 3 bits qui représentent la taille de la donnée (64 bits soit 8 octets)
Taille d'une ligne : 32 x 8 + 1 + (44 – 9 – 3 – 2) = 287 bits
Taille du cache : Nbre_ligne* taille_Ligne = 512 * 287 = 17,9 KO

Dessiner l'architecture du cache de données, en précisant tous les paramètres de l'architecture (nombre de lignes, taille des bus, …).



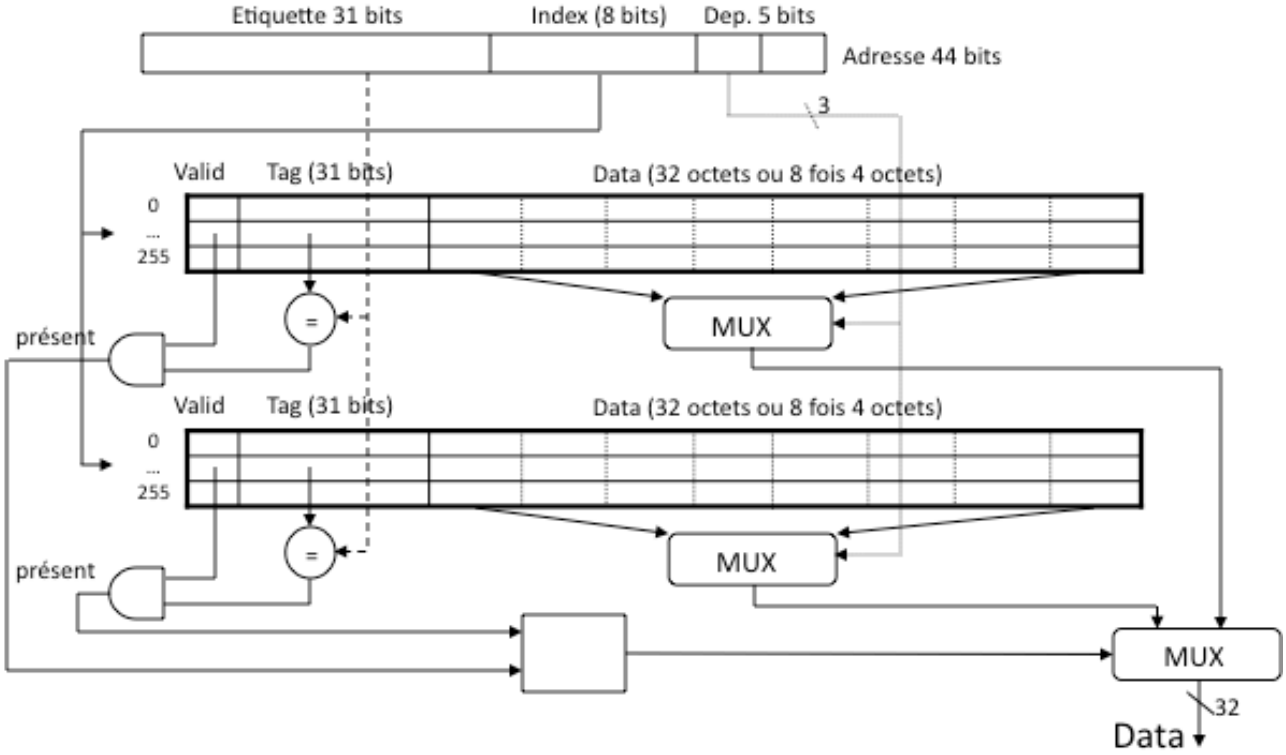
2) On charge dans ce cache le bloc de données (32 octets) dont l'adresse mémoire de la première donnée du bloc est 0x000AB240020. Indiquer la ligne du cache qui reçoit ce bloc et le contenu du champ index de la ligne du cache.

0x000AB240020 On calcule (0x000AB240020 >> 5) % 512 pour trouver la ligne (on omet les 5 bits de poids faible et on garde les 9 bits de poids faible) = ligne 1 0x000AB240020 = 0b0000…0010 0100 00**00 0000 001**0 0000 Le contenu du champ index est donc les 30 bits de poids forts Ce qui donne 0b0000…0010 0100 00 Soit : 0x0002AC90

Partie B : cache d'instructions

3) Indiquer la taille totale du cache.
Dessiner l'architecture du cache d'instructions, en précisant tous les paramètres de l'architecture (nombre de lignes, taille des bus, …).

Instructions sur 32 bits => 8 instructions par ligne = 32 octets / ligne 16 KO soit 512 blocs de 8 instructions 2 voies (ensembles) soit 256 lignes par voie = 2^8 Taille d'une ligne : 32 x 8 + 1 + (44 – 5 – 8) = 288 Taille du cache : 256 * 2 * 288 = 147459 = 18 KO



4) On charge un bloc de 8 instructions dans ce cache dont l'adresse de la première instruction est 0x00000AB0220.
Déterminer le numéro de l'ensemble. Indiquer la ou les lignes de cache qui pourront contenir ce bloc. Quel sera alors le contenu du champ index ?

0x00000AB0220
On calcule (0x00000AB0220 >> 5) % 256 pour trouver la ligne (on omet les 5 bits de poids faible et on garde les 8 bits du calcul % 256) = ligne 17 dans les 2 groupes.
0x00000AB0220 = 0b0000 …. 1010 1011 0000 **0010 001**0 0000
Le contenu du champ index est donc les 31 bits de poids forts
soit 0b0000 …. 1010 1011 000
Soit : 0x00000658

Exercice sur l'efficacité de la mémoire cache

Extrait d'un sujet d'exercices de Julien Dusser – IRISA – IFSIC – Université de Rennes 1

On considère un cache de données de 16 Ko à correspondance directe avec des lignes de 64 octets. Soit le morceau de programme suivant :

```
int a[NL][NC];
for (int i = 0; i < NL; i++)
   for (int j = 0; j < NC; j++)
        x = x + a[i][j];
```

Pour toute la suite de l'exercice, on fera les suppositions suivantes. On suppose que les variables i, j, x utilisent des registres. On supposera que l'adresse de début du tableau, &(a[0][0]), commence au début d'une ligne de cache. On rappelle que l'adresse &a[i][j] de l'élément (i, j) du tableau a est égale à &(a[0][0]) + i x NC + j, et on supposera qu'un int fait 4 octets.

Q1) Le taux d'échecs est défini comme le nombre total d'échecs divisé par le nombre total d'accès. En supposant NC et NL grands, quel est le taux d'échecs du cache de données ?

Une ligne de cache de 64 octets contient 16 mots de 4 octets (32 bits).
A chaque chargement du mot en début de ligne de cache, on charge aussi les 15 suivants (exemple : Quand on charge a[0][0] dans le cache, a[0][0] … a[0][15] sont aussi chargés).

Comme le tableau est chargé et utilisé dans le bon sens, une ligne de cache contient des éléments accédés consécutivement. Donc pour un miss de cache, on a ensuite 15 hit. Donc un taux de miss de 1/16.

On considère maintenant le morceau de programme suivant (permutation des boucles i et j) :

```
int a[NL][NC];
for (int j = 0; j < NL; j++)
   for (int i = 0; i < NC; i++)
        x = x + a[i][j];
```

Q2) Quel est le taux d'échecs si NL = NC = 256 ?

Un cache de 16 Ko avec une ligne de cache de 64 octets est composé de 256 lignes. (256 x 64 = 16 Ko).

Si NC = 256, cela signifie qu'on charge une ligne du tableau (256 x 4 octets) dans 16 lignes du cache (16 x 64 = 256 x 4). Donc a[i][j] sera sur la même ligne de cache que a[i + 16][j]. Donc la ligne de cache contenant a[i][j] est évincée par a[i + 16][j] avant d'utiliser a[i][j + 1].

Par conséquent le taux de miss est de 100%

Q3) Quel est le taux d'échecs si NL = 256 et NC = 272 ?

Un cache de 16 Ko avec une ligne de cache de 64 octets est composé de 256 lignes. (256 x 64 = 16 Ko).

Si NC = 272 (= 256 + 16), cela signifie que si a[i][j] est sur la ligne de cache n, alors a[i + 1][j] sera chargée sur la ligne de cache (n + 17) % 256.

Existe t – il une valeur de k telle que l'élément a[i + k][j] évince a[i][j] ?
Pour cela il faut que n = (n + 17 x k) %256. La valeur de k est donc telle que 17 x k est un multiple de 256. La plus petite solution est k = 256.
Comme NL = 256, cela signifie qu'il n'y a pas de k tel que l'élément a[i + k][j] évince a[i][j]. Dans ce cas, quand on charge le premier mot d'une ligne, on charge aussi les 15 suivants. Donc le taux de miss est de 1/16.