TD n° 4 : CACHES

1. Etiquettes et index de cache.

Un processeur a 2 Go de mémoire principe.
Pour les différents caches ci-dessous, on demande
a) Quelle est la décomposition d'une adresse mémoire (figure 1) ? Donner le nombre de bits pour les parties étiquettes, index et adresse dans le bloc.
b) Donner les différentes parties d'une ligne (bloc) de cache (figure 2). Combien y a-t-il de bits pour le contrôle, l'étiquette et la partie donnée ? Quel est le nombre total de bits du cache ? Par rapport à la partie « données » du cache, quel est le surcoût lié aux bits de contrôle et d'étiquette ?

Etiquette	Index	Adresse dans bloc

Figure 1 : décomposition d'une adresse mémoire

Etiquette	ctl	Instructions ou données

Figure 2 : ligne de cache

A) Cache de 2 Mo à correspondance directe et écriture simultanée avec des blocs de 16 octets
B) Cache de 4 Mo à correspondance directe, réécriture et blocs de 32 octets.
C) Cache de 4Mo associatif 4 voies (4 blocs par ensemble), réécriture et blocs de 32 octets.

2. Caches données

On considère une architecture possédant un cache de données de 8K octets organisé en blocs de 32 octets. Les exercices suivants seront traités dans 2 cas : correspondance directe et associativité par ensembles de 2 blocs, avec pseudo-LRU. On considère des tableaux de 4096 flottants simple précision (32 bits), implantés aux adresses suivantes :

X	Y	Z	X1	Y1	X2	Y2
$1\ 0000_H$	$1\ 4000_H$	$1\ 8000_H$	$1\ C000_H$	$1\ E000_H$	$2\ 0000_H$	$2\ 4000_H$

a) Quels sont les éléments des tableaux X et Y qui peuvent occuper le mot 0 du bloc 0 du cache, dans les deux cas : correspondance directe et associativité par ensemble de 2 blocs ?

b) Combien de défauts du cache de données par itération interviennent dans chacune des boucles suivantes, où on suppose que les variables scalaires sont toujours en registre :

b1	b2	b3	b4

For (i=0; i<N; i++)
S += X[i]*Y[i];
	For (i=0; i<N; i++){
	S1 += X1[i]*Y1[i];
	S2 += X2[i]*Y2[i];
	}
		For (i=0; i<N; i++)
		S1.+= X1[i]*Y1[i];
		For (i=0; i<N; i++)
		S2 += X2[i]*Y2[i];
			For (i=0; i<N; i++) {
			S1 += X[i]*Y[i];
			S2 += X[i]*Z[i];
			}

3. Caches instructions

Un processeur a un jeu d'instructions RISC, avec des instructions de longueur fixe d'un mot. Il a un cache instructions de 2 Kmots, avec des blocs de 8 mots. Il utilise la correspondance directe.
Il exécute le programme Figure 3, constitué de deux boucles imbriquées. Les seuls branchements du programme sont les deux branchements de boucle, aux adresses 239 et 1200.
Le temps pour un succès cache est T et un défaut de cache coûte 8T.
a) En négligeant l'effet des défauts de caches pour les données, quel est le temps d'exécution du programme de la Figure 3

b) Reprendre la question précédente en supposant un cache de 1 Kmots avec correspondance directe, puis l'associativité 2 voies (2 blocs par ensemble)

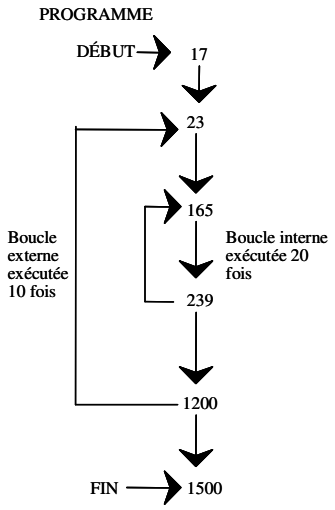

Figure 3: programme considéré

4. Cache données (exercice optionnel)

Soit le programme suivant, qui effectue la normalisation des colonnes d'une matrice X[8][8] : chaque élément de la colonne est divisé par la moyenne des valeurs de cette colonne.

```
float X[8][8], sum, ave;
sum = 0.0;
for (j = 0; j < 8; j++) {
      for (i=0; i<8, i++)
            sum+= X[i][j];
      ave = sum/8;
      for (k=7; k>=0; k--)
            X[k][j] }
```

On suppose que l'on a un cache de 128 octets avec des blocs de 16 octets (soit 8 blocs pour le cache).
L'adresse de X[0][0] est $F000_H$ (sur 16 bits)

Correspondance directe

En supposant la correspondance directe, définir dans quels blocs du cache vont chaque élément de la matrice.
En déduire le nombre de défauts de cache pour l'exécution du programme ?
Quel serait le nombre de défauts de cache en écrivant la seconde boucle interne
for (k=0; k<8, k++)

Correspondance totalement associative

Quel est le nombre de défauts de cache pour le programme initial en utilsant le LRU comme algorithme de remplacement ?

Correspondance associative 2 voies

On suppose maintenant que le cache a 4 ensembles de 2 blocs (associatif 2 voies).
Définir dans quels ensembles vont les éléments de la matrices.
Quel est le nombre de défauts de cache pour le programme initial en utilsant le LRU comme algorithme de remplacement ?