TP N° 1 : Simulation de caches direct puis associatif

L'objectif de ce TP est de simuler l'utilisation de caches de données direct et associatif en faisant varier les différents paramètres : nombre de lignes de cache, nombre d'entrées par ligne,…. La programmation se fera en langage JAVA.

Pour tester les différents caches on utilisera les fichiers fournis suivants :

- `matrice10.txt` : c'est un fichier constitué de lignes formés d'une adresse (en base 10) et du mode d'accès (Lecture=R et Ecriture=W). Ce fichier est issu d'un programme effectuant des addition et multiplication matricielle sur des matrices 10x10
- `matrice100.txt` : comme le fichier précédent mais sur des matrices 100x100
- `alea10.txt` : ce fichier contient des adresses aléatoires avec des modes d'accès aléatoires. La taille du fichier est identique à celle de `matrice10.txt`.

Exercice 1: Cache direct

1. Écrire un programme de simulation de cache **direct** de taille variable permettant de lire un fichier d'adresses (au format décrit en introduction) et de simuler la mise en cache des données désignées par chaque ligne.

   Le cache simulé aura les caractéristiques suivantes :

   - Taille d'un bloc = 32 octets
   - Nombre de lignes du cache = 2^n avec `n` entier >0

2. Le simulateur de cache générera les informations suivantes :

   - Nombre de succès
   - Nombre d'échecs
   - Temps moyen d'accès sachant qu'un accès au cache se fait en 5ns et un accès mémoire se fait en 50ns

3. Tester le programme précédent avec le fichier `matrice10.txt` et pour `n=2,3,4,5,6,7`. Déterminer la valeur de `n` la plus intéressante

4. Tester le programme précédent avec le fichier `alea10.txt`. Comment expliquer le résultat ?

Exercice 2 : Cache associatif

1. Écrire un simulateur de cache **associatif** de taille variable et au nombre d'entrées variable permettant de lire un fichier d'adresses (au format décrit en introduction) et de simuler la mise en cache des données désignées par chaque ligne.

   Le cache simulé aura les caractéristiques suivantes :

   - Taille d'un bloc = 32 octets
   - Nombre de lignes du cache = 2^n avec `n` entier >0
   - Nombre d'entrées du cache = `nbEntrees` avec `nbEntrees` entier >0

   L'algorithme de remplacement du cache est l'algorithme LRU

   Le simulateur de cache générera les informations suivantes :

   - Nombre de succès

- Nombre d'échecs
- Temps moyen d'accès sachant qu'un accès au cache se fait en 5ns et un accès mémoire se fait en 50ns

2. Tester le programme précédent avec le fichier `matrice10.txt` et pour `n=3, 4, 5` et `nbEntrees=2,4`. Que concluez-vous ?
3. Tester le programme précédent avec le fichier `matrice100.txt` et pour `n=7,8` et `nbEntrees=2,4`. Que concluez-vous ?

Exercice 3 : write back ou write through (optionnel)

L'objectif ici est de modifier programme de simulation de cache pour tester la différence de performances entre les politiques d'écriture du cache en mémoire **write back** et **write through**.

Pour comparer les performances de ces 2 politiques vous considérerez un cache associatif possédant 64 lignes et 2 entrées par ligne.

Vous déterminerez pour chaque fichier `matrice10.txt` et `matrice100.txt` le nombre d'écritures effectuées en mémoire centrale. Que concluez-vous ?