Université de Valenciennes
ISTV – Master Informatique 1 – 2017_2018
TP : **Conception d'un Simulateur de Mémoire Cache**

Proposé par: Smaïl NIAR

I. Introduction

Dans ce TP, il s'agit de concevoir un simulateur de mémoire cache en langage C, C++ ou Java.
Ce TP peut être réalisé en binôme (deux étudiants), la note finale du TP prendra en compte le travail réalisé et le nombre d'étudiants qui ont (effectivement) participé à la réalisation du TP.
Le but de ce TP est d'écrire un programme capable de simuler le fonctionnement d'une mémoire cache.
Vous devez remettre à votre enseignant votre compte rendu de TP, contenant :

(a) Les noms des personnes qui ont participé aux projets
(b) Votre programme écrit en C, C++ ou Java qui sera de la forme ***nom_Code à mettre dans Moodle***
(c) La valeur de la variable **code = Σ(codes Ascii de la première lettre des noms du binôme)**
(d) Les résultats expérimentaux (sous formes de tableaux et de courbes)
(e) Les réponses aux questions

Dernier délai pour remettre les résultats : J+15 (15 jours)

II. Les données du problème

Votre programme doit prendre comme paramètres (fournis lors de l'exécution) :

- La taille totale de la cache en octets (noté cs : cache size)
- La taille d'un bloc en octets (noté bs : bloc size)
- Le degrés d'associativité du cache (noté assoc)

Dans ce TP, on considère que les adresses font références uniquement à des mots de 32 bits (4 octets).. Par conséquent cs et bs sont multiples de 4.
Par ailleurs, cs donne la taille totale du cache pour les données uniquement. L'espace pris pour stocker les étiquettes, le bit de validation, bit de bloc modifié (M), ..Etc., n'est pas pris en compte dans bs.
Il faut noter que si assoc=1, le cache est « à accès direct » appelé Direct mapped cache ou DMC, alors que si **assoc = cs/bs**, le cache est totalement associatif.
Votre simulateur lit un fichier texte qui contient la trace d'un programme de référence. Ce programme contient les références mémoires (une par ligne) faite par un programme. Ce fichier est appelé la trace mémoire du programme. Chaque ligne du fichier commence soit par la lettre R ou par W, suivant qu'il s'agisse d'une lecture ou d'une écriture de données. Les traces d'exécution de 3 programmes sont aussi sur Moodle. Il s'agit du programme de multiplication de matrice 10*10, Multiplication de matrices 100*100, et le programme de tri par fusion sur un vecteur de 200 éléments.
Votre cache ne va pas contenir des données (ce qu'elle est sensée faire), mais uniquement les adresses des données qui doivent normalement s'y trouver. En effet comme il s'agit de compter le nombre de défauts de la mémoire cache, il n'est pas nécessaire de savoir la valeur des données.
En fonction de la valeur de **Code / nombre de possibilités**
Vous aurez à implémenter la gestion des écritures :

0) par la technique de l'écriture directe (Write Through **WT**) , on écrit en m^me temps en mémoire et en mémoire cache. Il n'est pas nécessaire de faire des sauvegardes.
1) par l'écriture différée (Write Back **WB)**. Dans ce cas, l'écriture en mémoire centrale est réalisée uniquement lorsque le bloc est supprimé du cache.

Pour le remplacement des blocs, vous aurez aussi à implémenter l'une des méthodes suivantes :

0) FIFO : Le premier bloc entrée premier sortie. On utilise un compteur pour indiquer, le bloc le plus ancien. La valeur du compteur est stockée avec le bloc.
1) LRU : Le moins récemment référencé est supprimé. Dans ce cas, on choisi de supprimer le bloc qui a été le moins référencé depuis qu'il a été chargé. On utilise un compteur qu'on incrémente à chaque accès.

2) NRU : On utilise in bit (R) pour indique est ce que le bloc a été récemment accédé ou non. En cas de défaut, on élimine le premier bloc qui a sont bit R=0. Toutes les 1000 références on remet tous les bits R à 0.

3) Aléatoire

Vous devez mettre la formule qui calcule « code » en grand au niveau de la première page de votre rapport.

Exemple : pour Niar et Adam, on aura :

'N'+'A' = 65 + 78 = 143, Gestion Ecriture = 143/2 = 1 : WB

Remplacement des blocs = 143/4 = 3 : Random

L 'appel au votre programme sera :

% simCachecode cs bs assoc trace.txt

Exemple : simCache143 4096 64 4 mergeort.txt

Exécution du programme de Niar et Adam avec cs=4096 bs=64 assoc=4 sur la trace mergesort.txt

Pour chacun des paragraphes suivant et pour chaque expérience, votre rapport devra comporter :

- Les paramètres de la cache : cs, bs, assoc
- Le nombre total de lectures et d'écritures réalisés
- Le nombre de défauts en lecture et le nombre défaut en écriture
- Si les écritures sont gérées par WB, le nombre de blocs vidés (purgés ou écrits) en mémoires (dirty write)

III. Effet de la taille d'un bloc bs sur les performance

Dans ce paragraphe on utilise une mémoire cache à deux ensembles (2-way set associative cache). La taille totale est de 2Kbytes (512 mots).

On définit **la pénalité d'un défaut** = (12 + bs/8) cycles.

Voici comment est calculé le nombre de cycles pris par la mémoire :

Pour la méthodes WB, **Cycles perdu par la mémoire** = nombre de cycles dues à la pénalité des défauts * (nombre défauts lecture + nombre défauts écriture + nombre de lignes supprimées du cache)

Pour la WT, **Cycles perdu par la mémoire** = [nombre de cycles dues à la pénalité des défauts * (nombre défauts lecture + nombre défauts écriture) + 12 * nombre d'écritures] .

- Donnez la valeur **Cycles perdu par la mémoire**, pour toutes les tailles de blocs bs de 8 à la taille maximale possible des 3 traces d'exécution. Quelles est la valeur bs optimale pour chaqu'une des trace ?

IV. Effet du degrés d'associativité

Pour mergeSort supposant que le nombre d'instructions mémoires « sw » et « lw » représentent 40% du nombre total d'instructions. Calculez le **temps d'exécution du programme** pour les deux cas suivants :

- Il n'y a pas de cycles perdus par la mémoire (taille cache infinie) et chaque instruction s'exécute en 1 cycle = 10 nsec.
- La taille du cache n'est pas infinie, mais elle (cs) vaut 2048 octets et bs vaut 64 octets, le temps d'un cycle est donné par :10 + $\log_2$ (assoc) nsec et la pénalité due à un défaut est de 20 cycles. Donnez le temps d'exécution pour toutes les valeurs possibles du paramètre « assoc » de 1 jusqu'à une certaine valeur maximale que vous avez à préciser.