UNIVERSITE DE LA ROCHELLE – LICENCE INFORMATIQUE
3ème année – UE *Architectures avancées-*

TP Architecture 3 :
Simulation d'une hiérarchie mémoire

I – Contexte

La mise en place d'une mémoire cache permet d'augmenter considérablement les performances d'un processeur. En effet, la mémoire cache va tirer profit de la réutilisation des données qui est inhérente à beaucoup d'applications.

Dans ce TP, nous nous intéressons à la simulation d'une hiérarchie mémoire à 1 puis à 2 niveaux. Pour cela nous utiliserons un simulateur de mémoire cache disponible sur Internet et dont la caractéristique principale est de pouvoir simuler autant de niveaux de mémoire cache que désiré. Cet outil se nomme Dinero IV. Il permet, à partir d'une série d'accès mémoire, de déterminer le nombre de défauts de cache et ceci en fonction d'une configuration de mémoire cache données. Il est possible de positionner un grand nombre de paramètres afin de rechercher la configuration optimale pour une application.

La simulation d'une mémoire cache suppose que l'on dispose de la totalité des accès que réalise le processeur. Pour cela, l'outil Dinero IV travaille à partir d'une séquence d'accès à la mémoire, il s'agit simplement de la liste des requêtes mémoires réalisées par le processeur. Pour obtenir cette séquence, il nous faut alors disposer d'un moyen d'extraire l'ensemble des accès mémoire qui sont réalisés par un processeur lors de l'exécution d'un programme. Pour obtenir cette liste d'accès, deux solutions s'offrent à nous :

– On modifie le code C original afin d'y insérer des affichages (*printf*) permettant de suivre l'ensemble des accès mémoires réalisés par le processeur.

– On utilise un simulateur de processeur, par exemple *JSimRisc*, permettant d'extraire l'ensemble des accès mémoires réalisés au cours de l'exécution de l'algorithme.

Pour ce TP, nous utiliserons le simulateur de processeur *JSimRisc*. Nous allons tout d'abord étudier un système disposant d'un seul niveau de mémoire cache, puis nous étudierons le cas d'une hiérarchie mémoire à deux niveaux : cache de niveau 1 et cache de niveau 2

II - Analyse d'une trace d'accès mémoire

Cette partie a pour objectif que de vous familiariser avec l'outil d'analyse. Pour pouvoir faire une simulation de mémoire cache, nous allons utiliser l'outil Dinero IV. Cet outil se présente normalement en ligne de commande, mais pour vous simplifier un peu la manipulation de cet outil, nous vous proposons une interface un peu plus conviviale basée sur une page web dont voici un *snapshot*, figure 1.1. Cette page WEB a également été à disposition par D. Chillet (ENSSAT de Lannion).

Ce simulateur de cache est disponible au travers d'une page web que vous trouverez à l'adresse

http://cairn.enssat.fr/enseignements/Archi/SimuleCache/

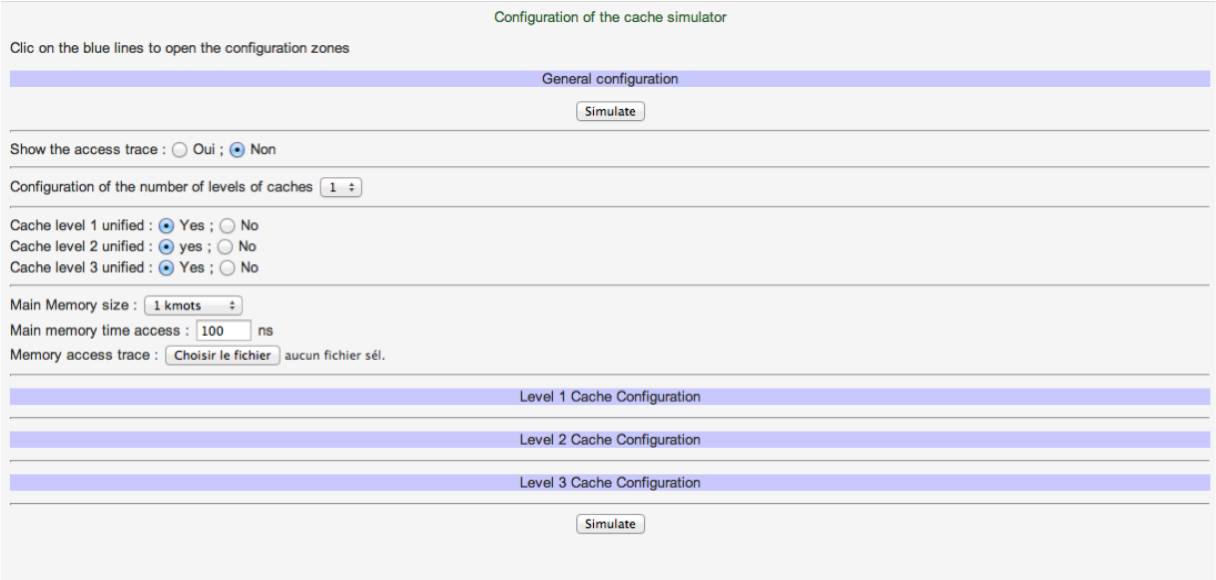

FIGURE 1.1 – Simulateur de mémoire cache avec panneau général de configuration.

Le simulateur de mémoire cache vous demande dans un premier temps de spécifier la configuration générale, c'est à dire le nombre de niveaux de mémoires cache. Il vous est aussi demandé de préciser si ces différents niveaux de mémoire cache sont unifiés ou pas. Une fois la configuration générale définie, il vous faut ensuite spécifier la configuration de chaque niveau de mémoire cache. Pour cela, vous devez cliquer sur le bandeau indiquant « Configuration du cache de niveau X » afin d'ouvrir le panneau de configuration du niveau de mémoire.

A partir de *JSimRisc*, cochez « memory trace : data » si vous souhaitez avoir un état des accès à la mémoire cache pour les données et « memory trace : instructions » des accès à la mémoire cache pour les données. Ce processus permet au simulateur de créer le fichier *pgm.mem* généré à partir du bouton « choisir un fichier » du site dont l'URL : chargement d'un exemple de trace d'accès mémoire au format Dinero IV.

Remarque : Prenez soin de supprimer le fichier *pgm.s.mem* préalablement créé par *JSimRisc*. Ensuite, faites un Reset du processeur, puis cliquez sur Dinero trace : Data et sur Dinero trace : Instruction. Lancez ensuite une exécution, bouton Run, puis dévalidez les deux cases Dinero trace : Data et Instruction. Une fois cela fait, vous disposez du fichier de trace *pgm.s.mem* contenant les accès aux données et aux instructions réalisées par le processeur lors de l'exécution de l'application. Vous pouvez alors fournir ce fichier de trace à l'outil de simulation de mémoire cache.

Configurez la mémoire cache de la façon suivante :

– 1 seul niveau de cache
– cache de niveau 1 unifié
– taille mémoire centrale : 1 kmots
– temps d'accès mémoire centrale : 100ns
– trace des accès, donnez le fichier de trace fourni – taille du cache : 128 mots
– taille des blocs : 16 mots
– temps d'accès à la mémoire cache : 10 ns
– associativité du cache : associatif direct
– politique d'écriture des blocs : write back et write allocate

Dans la liste des exemples présents sur l'archive que vous avez chargée, utiliser le programme *Fibonnacci.s* déjà utilisé dans le TP1. Le simulateur *JSimRisc* sera configuré comme suit :

- 1 seul étage d'exécution
- Utilisation Bypass
- Aucune prédiction de branchement

Question 1 – Suite Fibonnacci

Questionnaire MOODLE

II – Hiérarchie à 1 niveau

L'étude que nous nous proposons de mener concerne la recherche d'une hiérarchie mémoire optimale pour une application de calcul de produit matricielle. Ce calcul est donné par les équations suivantes :

Tous les éléments des matrices sont supposés codés sur 1 mots de 32 bits. Pour la suite on prendra N = 8.

Nous étudierons 2 hiérarchies différentes qui sont :

– **hiérarchie 1** : processeur + mémoire centrale + mémoire cache Caractéristiques de la mémoire centrale :
– temps d'accès : 100 ns ;
– taille :1 kmots;

Caractéristiques de la mémoire cache : – temps d'accès : 10 ns ;
– taille : 128 mots ;
– taille des blocs : 16 mots ;

hiérarchie 2 : processeur + mémoire centrale + mémoire cache Caractéristiques de la mémoire centrale :
– temps d'accès : 150 ns ;
– taille :1 kmots;

Caractéristiques de la mémoire cache : – temps d'accès : 10 ns ;
– taille : 128 mots ;
– taille des blocs : 4 mots ;

Dans un premier temps, nous nous intéressons aux requetés mémoire du processeur pour accéder aux données et uniquement aux données. Nous verrons ensuite le comportement de la hiérarchie mémoire dans le cas où l'on considère aussi les accès au programme.

Question 2 : Accès aux données

Dans un premier temps, on s'intéresse uniquement aux accès aux données.
On imagine que les trois matrices A, B et C sont rangées en mémoire les unes à la suite des autres à partir de l'adresse 0 :
– matrice A stockée de l'adresse 0 à l'adresse $N^2 - 1$;
– matrice B stockée de l'adresse N^2 à l'adresse $2*N^2-1$;
– matrice C stockée de l'adresse $2*N^2$ à l'adresse $3*N^2-1$.

Le listing 1) vous fournit le code C du produit de matrices. Il s'agit donc dans un premier temps d'extraire les requêtes mémoire du processeur lors de l'accès aux structures de données `A[i][k], B[k][j], C[i][j].`

```
#include <stdio.h>
 #define N 8
main() {
int i;
int j;
int k;
for (i=0 ; i< N ; i++) {
for (j=0 ; j<N ; j++) { tmp = 0;
for (k=0; k< N ; k++) {
tmp = tmp + A[i][k] * B[k][j];
} C[i][j] = tmp;
}} }
```

Listing 1 — Code C du produit de matrices

L'algorithme décrit en code C a été compilé et nous avons obtenu le code assembleur fourni dans le listing proposé dans le programme *ProdMat.s* disponible dans les exemples présents dans l'archive de *JSimRisc*.

Questionnaire MOODLE : question 2

On souhaite maintenant travailler sur un cache non unifié. La mémoire cache est alors divisée en deux mémoires cache de même taille, structurée de la même manière, avec la même stratégie. **Seule la hiérarchie 1** est étudiée dans une configuration associative 4 voies avec stratégie LRU.

Questionnaire MOODLE: question 2

Question 3 : Accès aux données et au programme

Si on considère maintenant l'ensemble des accès nécessaires au processeur, il nous faut prendre en compte également l'accès aux instructions.

Pour obtenir la liste des accès mémoire nous allons à nouveau utiliser l'outil *JSimRisc*. En validant la trace des instructions vous obtiendrez la trace des lectures d'instructions et des données réalisées par le processeur.

Questionnaire MOODLE: question 3

On souhaite maintenant travailler sur un cache non unifié. La mémoire cache est alors divisée en deux mémoires cache de même taille, structurée de la même manière, avec la même stratégie. **Seule la hiérarchie 1** est étudiée dans une configuration associative 4 voies avec stratégie LRU.

Questionnaire MOODLE: question 3