# UNIVERSITÉ PARIS SACLAY
# Master E3A – SETI

# T1 Architectures avancées
# TD2 Hiérarchie mémoire

## 1 Étiquettes et index de cache

Un processeur a 2 Go de mémoire principale.
On considérera les les cas suivant :

1. Cache de 2 Mo à correspondance directe et écriture simultanée avec des lignes de 16 octets
2. Cache de 4 Mo à correspondance directe, réécriture et lignes de 32 octets.
3. Cache de 4 Mo associatif 4 voies (4 lignes par ensemble), réécriture et lignes de 32 octets.

Pour ces différents caches, on demande :

**1.1** Quelle est la décomposition d'une adresse mémoire (figure 1) ? Donner le nombre de bits pour les parties étiquettes, index et déplacement dans la ligne.


FIGURE 1 – Décomposition d'une adresse mémoire

**1.2** 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 données ? 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 ?

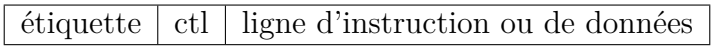


FIGURE 2 – Ligne de cache

## 2 Caches données

On considère une architecture possédant un cache de données de 8K octets organisé en lignes de 32 octets. Les exercices suivants seront traités dans 2 cas : correspondance directe et associativité par ensembles de 2 lignes, 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 |
|---|---|---|---|---|---|---|
| 0x10000 | 0x14000 | 0x18000 | 0x1C000 | 0x20000 | 0x24000 | 0x28000 |

**2.1** Quels sont les éléments des tableaux X et Y qui peuvent occuper le mot 0 de la ligne 0 du cache ?

**2.2** Combien de défauts de cache de données par itération interviennent dans chacune des boucles $b_1 \cdots b_4$ ci-dessous ? On suppose que les variables scalaires sont toujours en registre.

$b_1$

```
for(i=0;i<N;i++)
 S+=X[i]*Y[i];
```

$b_2$

```
for(i=0;i<N;i++){
  S1+=X1[i]*Y1[i];
  S2+=X2[i]*Y2[i];
}
```

$b_3$

```
for(i=0;i<N;i++)
  S1+=X1[i]*Y1[i];
for (i=0;i<N;i++)
  S2+=X2[i]*Y2[i];
```

$b_4$

```
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 lignes de 8 mots. Il utilise la correspondance directe. Il exécute le code de la 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 $8t$.

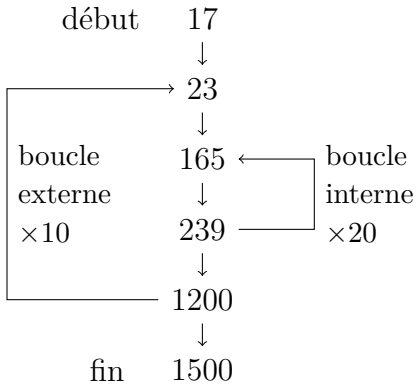


FIGURE 3 – Programme

**3.1** En négligeant l'effet des défauts de caches pour les données, quel est le temps d'exécution du programme de la Figure 3.

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

## 4 Mémoire virtuelle et TLB

Soit le code suivant de multiplication de matrices.

```
1  float a[1024][1024], b[1024][1024], c[1024][1024];
2  multiply()
3  {
4    int i, j, k;
5    for(i = 0; i < 1024; i++)
6      for(j = 0; j < 1024; j++){
7        c[i][j]=0.0;
8        for(k = 0; k < 1024; k++)
9          c[i][j] += a[i][k] * b[k][j];
10     }
11 }
```

On suppose que le code binaire pour exécuter cette fonction tient dans une page de 4 Ko et que la pile tient elle aussi dans une page.

**4.1** Quel est le nombre de défauts de TLB avec un TLB de 8 entrées utilisant LRU comme politique de remplacement ?.
NB : Les pages pour le code binaire et pour la pile seront en permanence en mémoire, et les entrées correspondantes du TLB seront donc constamment utilisées. 6 entrées du TLB restent disponibles pour l'exécution du programme.

**4.2** Reprendre la question précédente en utilisant l'algorithme ikj (optionnel)