TD AEV

FICHE 7 Mémoire cache

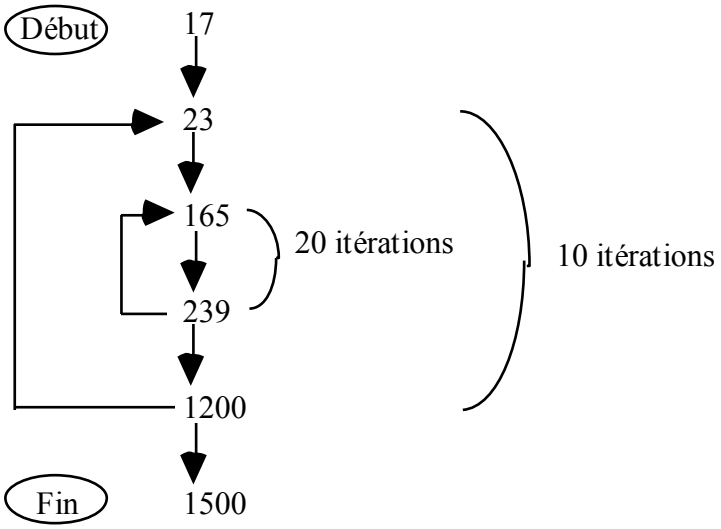
Exercice 1 Adressage

On dispose d'une mémoire principale de 2^{16} octets. Le transfert minimal entre la mémoire et le cache est de 8 octets. On utilise un cache direct mapping de 32 lignes.

1) Définissez le nombre de champs de l'adresse et la taille de chaque champ.
2) Dans quelle ligne se trouvent les mots dont les adresses sont :
   0001000100011011
   1100001100110100
   1101000000011101
   1010101010101010
3) Supposez que l'octet dont l'adresse est 0001 1010 0001 1010 soit rangé dans le cache. Quelles sont les autres adresses rangées dans la même ligne ?
4) Que range-t-on avec la donnée et pourquoi ?
5) Quelle est la taille effective du cache?

Exercice 2 Cache d'instruction

Un programme se compose de deux boucles FOR imbriquées, une petite boucle interne et une plus grande externe. La structure générale du programme est la suivante:



Les adresses mémoires décimales données déterminent l'emplacement des deux boucles ainsi que le début et fin du programme. Tous les emplacements mémoire contiennent des instructions devant être exécutées séquentiellement. Le programme s'exécute sur un processeur avec cache mémoire. Le cache est organisé en direct mapping. La mémoire est de 64K mots, le cache est de 1K mots , il est organisé en bloc de 128 mots.

a)Précisez les différents champs d'une adresse ainsi que leur taille.

b)En ne tenant compte que de l'accès mémoire pour le chargement d'une instruction, donnez le temps de chargement moyen d'une instruction de ce programme. On pose M le temps d'accès à la mémoire d'un bloc et m le temps d'accès au cache d'une instruction

Exercice 3 Rangement mémoire cache

Un processeur adresse une mémoire de 2^{32}mots. On désire utiliser une mémoire cache de 16 Kmots de capacité. Les mots sont de 32 bits. Pour chaque construction donnez le nombre de

champs de l'adresse et la taille de chacun, puis calculez la taille totale du cache en bits (TAG + data + ...)
a) En direct mapping
b) En direct mapping par bloc de 16 mots
c) En Full associative
d) En set-associative par ensemble de 16 mots
e) En set-associative par ensemble de 16 blocs de 8 mots

Exercice 4 Algorithmes de remplacement
Lors d'une exécution d'un programme, on observe la trace des pages suivantes : P = r1, r2, r3, ... rk, rk+1, ... où rk est le numéro de la page qui contient la kième référence du programme.
Exemple

trace	a	b	c	b
défaut	*	*	*	
cache	a	a	c	c
	#	b	b	b

Pour la trace suivante P = abacabdbacd
1)Donner le contenu de la mémoire cache, la présence de défaut de cache pour chaque référence. La mémoire cache contient 2 pages.
a) Pour un algorithme de remplacement FIFO
b) Pour un algorithme de remplacement LRU
c) Est-on loin de l'optimal ?
2) Pour une mémoire cache contenant 3 pages, même question.
3)LRU et FIFO semblent de " bons " algorithmes. L'algorithme MRU Most Recently Used semble intuitivement mauvais. Comparer avec les résultats précédents. Peut-on juger de l'efficacité d'un algorithme sur une seule trace de programme ?

Exercice 5 Exécution
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] = X[k][j]  / ave
}
```

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 F000H (sur 16 bits), la matrice est rangée ligne par ligne, puis colonne par colonne. On répondra aux questions pour ces deux rangements.
Direct mapping
Définir dans quels blocs du cache va 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++)
```