

## TP n°2 : Exécution de programmes C : quelques problèmes.

### Introduction

Les objectifs de ces TP sont les suivants.

- Comprendre la méthodologie de mesure de temps d'exécution de programmes sous Linux, en utilisant le compilateur gcc.
- Observer l'impact de certaines options de compilation. On utilise le compilateur gcc.
- Mettre en évidence un certain de conséquences de l'architecture matérielle sur les résultats obtenus, soit en terme de précision des résultats (partie optionnelle sur les nombres flottants), soit en terme de temps d'exécution (autres exercices) en fonction des options de compilation et/ou des choix de programmation au niveau du code C

Ce TP est à effectuer sur des PC équipés de processeurs Intel Pentium 4.

Les programmes à utiliser sont disponibles dans la page Web du L313

<http://www.lri.fr/~de/ArchiL3-0910.htm>

**Pour chaque exercice, on donnera les temps demandés pour N=10, N=100, N=250, et avec les options de compilation suivantes :**

- **Sans optimisation**
- **Avec l'option de compilation -O3**
- **Avec les options de compilation -O3 -march=pentium4**
- **Avec les options de compilation -O3 -march=pentium4 -mfpmath=sse**

### 1. Calcul de la somme des N premiers entiers

On utilisera les deux procédures suivantes (récursive et itérative) permettant de calculer la somme des N premiers entiers :

```
int sumrec (int x){  
    return (x <1 ? 0: x+sumrec(x-1));}
```

```
int sumite(int x){  
    int i, tmp=0;  
    for (i=1; i<=x; i++) tmp+=i;  
    return tmp;}
```

Soit un programme C qui calcule la somme des N premiers entiers utilisant successivement chacune des procédures.

Donner le temps d'exécution/élément (c'est-à-dire le temps d'exécution total divisé par N)

### 2. Copie de tableaux

Soit un programme C qui effectue la copie d'un tableau N x N dans un autre en utilisant successivement les deux algorithmes suivants :

Version ij

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

Version ji

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

Donner le temps d'exécution/élément (c'est-à-dire le temps d'exécution total divisé par N\*N)

### 3. Création d'un tableau contenant la suite des N premiers entiers

Soient les 3 fonctions suivantes permettant d'écrire la suite des entiers de 0 à N-1 dans un tableau ( $T[i] = i$ ).

```
void foo(int x){
    int i;
    Z[0]=0;
    for (i=1; i<x; i++) Z[i]=Z[i-1]+1;}
```

```
void fou(int x){
    int i;
    Z[0]=0;
    for (i=1; i<x; i++) Z[i]=Z[0]+i;}
```

```
void fov(int x){
    int i;
    for (i=0; i<x; i++) Z[i]=i;}
```

Donner le temps d'exécution/élément (c'est-à-dire le temps d'exécution total divisé par N)

### 4. Exploitation des instructions SSE

On utilise le programme tex5.c. La fonction noSSE effectue la somme du carré de deux vecteurs ; la fonction withSSE effectue la même opération en exploitant le jeu d'instructions SSE en utilisant des intrinsics, c'est-à-dire une expression commode en C d'instruction assembleur.. Donner les temps d'exécution des deux fonctions.

### 5. Optionnel : Calcul en nombres flottants

#### Exercice Opt1.

Soit le programme C suivant :

```
main(){
    int i ;
    float S ;
    S = 1000.0 ;
    for (i=0; i<10000; i++)
        S=S+0.1;
    printf(...
}
```

Quelle est la valeur finale de S ?

Même question en remplaçant les float par des double.

#### Exercice Opt2.

Soit un programme C qui calcule les solutions d'une équation du second degré :  $a.x^2+b.x+c=0$

Exécuter le programme pour les valeurs

a = 0.3

b = 2.1

c = 3.675

en utilisant d'abord des float, puis des double.

Quels sont les résultats obtenus ?

### 6. Annexe : Mesures de temps

### **Outils de mesure**

Les mesures de temps sous Linux utilisent les fonctions suivantes, qui donnent le nombre de cycles d'horloge.

```
double dtime();
long long readTSC ();
long long readTSC ()
{
    long long t;
    asm volatile (".byte 0x0f,0x31" : "=A" (t));
    return t;
}
double dtime()
{
    return (double) readTSC();
}
```

### **Mesure du temps d'exécution**

Pour obtenir les temps d'exécution, on exécute les programmes plusieurs fois. On peut faire la moyenne des temps obtenus en enlevant les valeurs « aberrantes » (très supérieures aux autres). On peut aussi utiliser la plus petite valeur.

```
double t1,t2 ;//déclaration des variables
    t1 = dtime();
//Partie du programme dont on mesure le temps d'exécution.
    t2 = dtime();
    dt = t2-t1; // Nombre de cycles d'horloge processeur
```

### **Documentation du compilateur**

man gcc