

Logiciel de Base

TPL2 : implantation de tris en C et assembleur

ENSIMAG 1A

Année scolaire 2009–2010



1 Présentation du TP

Le but de ce TP est d'implanter plusieurs versions d'un tri itératif simple travaillant sur des tableaux d'entiers.

On vous fournit sur EnsiWiki une archive contenant les sources de départ. Les fichiers `tri_c.c`, `tri_asm.s` et `tri_asm_opt.s` contiennent les squelettes des fonctions que vous devez écrire dans ce TP. Le fichier `tp2.c` contient le programme principal, ainsi que des fonctions qui vont vous servir à tester vos implantations.

Vous pouvez compiler votre TP avec la commande :

```
gcc -m32 -Wall -Wextra -O3 -g -std=c99 -o tp2 tp2.c tri_c.c tri_asm.s tri_asm_opt.s
```

Le programme principal supporte plusieurs options d'exécution que vous pouvez lister en tapant `./tp2 --aide` qui affiche `./tp2 [<taille >= 2>] [--trace] [--verif]` :

- le premier paramètre à passer au programme est la taille du tableau : elle doit être d'au moins 2 éléments (pour éviter de saturer la mémoire de la machine, ne dépassez pas une taille de 100 millions d'éléments) ;

- l’option `--trace` permet d’afficher le contenu du tableau avant et après les différents tris : cette option sert à vérifier visuellement que votre tri fonctionne et n’est évidemment utile qu’avec des tableaux de petites tailles ;
- l’option `--verif` permet de vérifier que le résultat de vos fonctions de tri est bien correct (en comparant le tableau trié par vos fonctions avec le même tableau trié par une fonction de tri de la bibliothèque C) : cette option est utile pour vérifier que votre code est correct sur des tableaux de grande taille (mais attention, elle est très lente à l’exécution).

Tous ces paramètres peuvent être omis ou combinés car ils ont tous des valeurs par défaut (taille : 10, trace : non et verif : non). Vous êtes libres de rajouter d’autres options si vous en avez besoin. Par défaut, le tableau initial est rempli avec des entiers tirés aléatoirement entre 0 et taille - 1.

2 Le tri du nain de jardin

2.1 Principe du tri

On va travailler dans ce TP sur un tri itératif appelé le tri du nain de jardin. Son principe est très simple : un nain de jardin cherche à trier des pots de fleurs par taille croissante. Il regarde le pot devant lui : s’il est plus petit que le pot à sa droite, le nain avance d’un pas vers la droite (s’il n’est pas arrivé à la fin de la file de pots). Si le pot devant lui est plus grand que le pot à sa droite, le nain échange les deux pots, et recule d’un pas vers la gauche (s’il n’est pas revenu au début de la file de pots).

2.2 Implantation en C

On demande d’abord d’implanter une version en C de ce tri dans la fonction `void trier_c(int tab[], unsigned taille)` qui prend en argument le tableau d’entiers signés à trier ainsi que sa taille. Cette fonction doit être écrite dans le fichier `tri_c.c`.

On vous demande d’écrire du code clair et commenté. Une implantation directe de ce tri s’écrit en moins de 10 lignes de C. Les fonctions inutilement complexes ou confuses seront sanctionnées.

2.3 Implantation en assembleur

Vous devez ensuite traduire la fonction `void trier_c` que vous avez écrit vers de l’assembleur Pentium. Vous écrirez pour cela une fonction assembleur dont le prototype C serait `void trier_asm(int tab[], unsigned taille)`. Cette fonction doit être écrite dans le fichier `tri_asm.s`.

On vous demande de faire une traduction systématique, c’est à dire :

- d’ajouter en commentaire dans votre fichier `.s` chaque ligne de la fonction C, suivie des instructions assembleur qui lui correspondent ;
- d’utiliser les directives `.equ` ou `.set` pour définir les constantes utilisées lors des déplacements relatifs à `%ebp` ;
- de stocker dans la pile les variables locales et les paramètres de la fonction ;
- de ne pas chercher à optimiser le code que vous écrivez.

Par exemple, le fragment de C :

```
int x, y;
x = 5;
y = x + 4;
```

se traduira en assembleur par :

```
...
/* int x */
.equ x, -4
/* int y */
.equ y, -8
...
/* x = 5 */
movl $5, x(%ebp)
/* y = x + 4 */
movl x(%ebp), %eax
addl $4, %eax
movl %eax, y(%ebp)
```

La notation tiendra compte significativement du respect de ces consignes.

2.4 Optimisation de la version assembleur

On demande enfin d'écrire en assembleur une fonction dont le prototype C serait `trier_asm_opt` et qui correspond à la version optimisée de la fonction `tri_asm`. Cette fonction sera écrite dans le fichier `tri_asm_opt.s`

Vous devez appliquer un type d'optimisation simple consistant à minimiser le nombre d'accès mémoire à l'intérieur des boucles de votre programme. Pour cela, vous utiliserez au mieux les registres du Pentium pour y stocker les variables locales et paramètres dont vous souhaitez accélérer la gestion.

Vous devez respecter les conventions de gestion des registres de GCC. Plus précisément, on rappelle que `%eax`, `%ecx` et `%edx` sont des registres *scratch* (qui peuvent donc être utilisés librement), mais que `%ebx`, `%esi` et `%edi` ne le sont pas (et doivent donc être sauvegardés avant utilisation).

3 Comparaison des performances

Après avoir implanté les différentes versions du tri du nain de jardin, vous devez réaliser des tests de performances de ces différentes versions. Le programme principal `tp2.c` fourni contient déjà tout le code nécessaire à ces évaluations (pour peu que vous ayez respecté les conventions de nommage des fonctions externes).

La bibliothèque C contient différents algorithmes de tri, dont le plus connu (le *quicksort*) est implanté dans la fonction `qsort`. Cette fonction servira de tri de référence pour l'évaluation des performances de vos implantations.

Le *quicksort* est un tri dont la complexité est en $O(n \log(n))$, alors que le tri du nain de jardin a une complexité en $O(n^2)$. Cependant, il existe une configuration du tableau

initial pour lequel le tri implanté dans ce TP a une complexité en $O(n)$. On vous demande de trouver ce cas particulier et de modifier (très légèrement) le programme principal pour permettre la génération d'un tableau dans cette configuration initiale, afin de vérifier que le tri du nain de jardin est alors significativement plus rapide que le *quicksort*. Vous pouvez aussi implanter la réciproque de cette configuration optimale et vérifier que le temps d'exécution du tri du nain de jardin devient très long.

4 Règles du jeu

On vous demande de rendre vos sources (que les `.c` et `.s`, pas les binaires), ainsi qu'un fichier en texte brut dans lequel vous noterez les tests que vous avez effectués, les performances obtenues (copiez-collez simplement les traces d'exécution) ainsi que vos commentaires et conclusions.

Vous pouvez travailler en binôme ou en monôme : le fait d'être seul n'entraînera pas de bonus dans la notation.

Enfin, on rappelle que la fraude est sanctionnée sévèrement pour tous les TP et projets, comme précisé dans la charte d'utilisation du matériel informatique¹. Les enseignants se réservent le droit d'utiliser des logiciels de détection automatique pour localiser les ressemblances suspectes entre les fichiers rendus. Toute fraude avérée sera sanctionnée par un 0/20 au TP, sans possibilité de rattrapage. La récidive entraînera l'application des sanctions administratives prévues dans le règlement des études.

Le TP doit être rendu sur Teide au plus tard le vendredi 7 mai à 18h00.

1. http://intranet.ensimag.fr/teide/Charte_contre_la_fraude.php