Architecture des ordinateurs

L3 Informatique 2013-2014

TD/TP 6 - Assembleur : les bases

Points abordés

- Écrire des programmes en assembleur
- Comprendre les instructions
- Utiliser ddd
- Manipuler la mémoire en assembleur
- Représenter des données

Exercice 1. (Suite de Syracuse)
La suite de Syracuse est une suite dont le premier terme u_0 est arbitraire et dont le terme u_{n+1} s'obtient en fonction de u_n comme suit :

- $u_{n+1} = \frac{u_n}{2}$ si u_n est paire,
- $u_{n+1} = 3u_n + 1$ sinon.

La *conjecture de Collatz*, formulée dans les années 1930 et toujours ouverte, énonce que quelque soit le nombre u_0 que l'on fixe, il existe un entier n tel que $u_n = 1$.

1. Écrire un code assembleur qui place une valeur dans eax (représentant u_0), calcule les termes successifs $u_1, u_2, \ldots$ dans eax et met dans ebx le plus petit n tel que $u_n = 1$.
2. Écrire un code C qui fait de même. Ouvrez-le avec ddd (pensez à le compiler avec l'option -g), observez le code assembleur produit et comparez-le au votre.
3. Comparez les temps de calculs de vos deux programmes (pour voir la différence, modifiez vos programmes pour qu'ils effectuent le calcul sur beaucoup de valeurs).

Exercice 2. (Caractères)
Lorsque l'on réserve une zone mémoire en octets par la commande `db`, on peut l'initialiser par des caractères (qui sont interprétés par NASM comme leurs codes ASCII). Par exemple, les directives `db 65` et `db 'A'` sont équivalentes. Cela permet d'initialiser une zone mémoire réservée par une chaine de caractères/

Écrivez un programme en assembleur qui déclare un message (dans la section data) et le met en majuscules. Faites-en sorte que les caractères qui ne sont pas des lettres minuscules ne soient pas modifiés par votre programme.

Vous avez jusqu'ici vu comment réserver des zones mémoires en les initialisant (avec db,dw,dd). Si vous voulez réserver beaucoup de mémoire d'un coup, cela peut s'avérer fastidieux. Il existe des directives permettant de réserver un certain nombre de cases mémoire, sans les initialiser. Ces directives sont resb, resw et resd et elles réservent respectivement des octets, des mots (4 octets) ou des doubles (8 octets).
Par exemple, `toto : resw 100` réserve 200 octets et [toto] pointe vers le premier octet.

Exercice 3. (Renverser un tableau)
Écrivez un programme en assembleur qui renverse un tableau de n cases d'un octet, c'est à dire qui met la case $n-1$ dans la case 0, la case $n-2$ dans la case 1, etc... Votre programme ne devra pas utiliser de second tableau pour effectuer l'opération. Pensez à initialiser votre tableau pour constater facilement que votre programme fonctionne. Modifiez ensuite votre programme pour qu'il renverse un tableau de mots.

Exercice 4. (Crible d'Ératosthène)
Le crible d'Ératosthène est une méthode rapide pour déterminer quels sont les entier premiers plus petit que n. Il consiste à initialiser un tableau tab de n cases en mettant toutes ses valeur à 1, sauf les cases 0 et 1 qui sont mises à 0. Ensuite, pour chaque case $i \geq 2$, si $tab[i]$ vaut 0, on ne fait rien et s'il vaut 1, on met à 0 les cases $k \times i$ avec $k > 1$ et $k \times i < n$. Programmez le crible d'Ératosthène en assembleur.

Exercice 5. (Liste Chaînée)
Écrivez un programme qui manipule une liste chaînée de couples d'entiers. Pour cela :
- Commencez par réserver en mémoire un tableau de mots et initialisez-le à 0.
- Implémentez ensuite votre liste chaînée dans ce tableau. Vous pouvez par exemple grouper les mots par 4, les 2 premiers codant votre couple d'entiers, le troisième contenant un pointeur sur la case précédente et le dernier un pointeur sur la case suivante.

Faites en sorte que votre programme puisse éliminer une case de la liste, en créer une et échanger 2 cases.