Architecture des ordinateurs

Projet assembleur SPARC

Arnaud Giersch, Benoît Meister et Frédéric Vivien

Consignes

1. Ce projet est à faire individuellemnt ou en binôme, à l'exclusion de tout groupe de taille supérieure.
2. Les programmes doivent être écrits directement en assembleur SPARC (et non pas en C puis compilés, ce qui est repérable au premier coup d'œil et vous vaudrait une note nulle).
3. Les programmes doivent être commentés de manière à les rendre facilement compréhensibles (aucun rapport n'est demandé) et présentés de manière lisible.
4. Vos nom, prénom et groupe de TD devront figurer en commentaire au début de *chacun* des fichiers sources.
5. Les sources *compilables* (pas de fichier Word, …) seront à retourner par mail, au plus tard le mercredi 16 janvier (minuit), aux enseignants de TD concernés :
   – Arnaud Giersch <giersch@icps.u-strasbg.fr>
   – Benoît Meister <meister@icps.u-strasbg.fr>
   – Frédéric Vivien <vivien@icps.u-strasbg.fr>

   La date du mercredi 16 janvier ne pourra en aucun cas être dépassée : à partir du lendemain, vous pourrez retirer au secrétariat les annales contenant les corrigés des TDs et TPs et de ce devoir.

1 Fonction modulo

Écrire une fonction calculant le reste de la division entière (fonction *modulo*) de son premier argument par le second. Les arguments sont des entiers positifs, le deuxième est non nul. Vous pourrez utiliser la formule suivante :

$$a \bmod b = a - b \times \left\lfloor \frac{a}{b} \right\rfloor$$

où $\left\lfloor \frac{a}{b} \right\rfloor$ est le résultat de la division entière de a par b.

2 Algorithme d'Euclide

2.1 Version itérative

Écrire un programme prenant en entrée deux nombres entiers positifs a et b, et calculant puis affichant le pgcd de ces deux nombres. Le pgcd sera calculé avec une version itérative de l'algorithme d'Euclide :

```
tant que b ≠ 0
   r := a mod b
   a := b
   b := r
fin tant que
pgcd := a
```

Vous utiliserez la fonction *modulo* écrite à la question 1.

2.2 Version récursive

Même question qu'en 2.1, mais avec une version récursive de l'algorithme :

```
pgcd(a, b) := si b = 0 alors a sinon pgcd(b, a mod b)
```