

TD5 : Programmation assembleur

Le simulateur d'assembleur est disponible à l'adresse [https://perso.liris.cnrs.fr/t\(pierron/archi2025/simproc.py](https://perso.liris.cnrs.fr/t(pierron/archi2025/simproc.py).

Utilisation

Le code assembleur doit être placé dans le fichier `prog.asm`. Les entrées seront écrites dans `entrees.txt`. Chaque instruction `IN Ri` lira un caractère ASCII du fichier, et le stockera dans R_i . Si la fin du fichier est atteinte, R_i contiendra 255. `READ Ri` et `PRINT Ri` sont des macros permettant de lire/écrire directement un entier.

Après avoir exécuté le simulateur `simproc.py`, le fichier `sorties.txt` contiendra la sortie de votre programme. Chaque instruction `OUT Ri` écrit dans `sortie.txt` le caractère dont le code ASCII est contenu dans R_i .

Le jeu d'instruction utilisables est le suivant :

- ADD, SUB, MUL, DIV, MOD, OR, AND, XOR, SL, SR (et leurs variantes immédiates)
- LD, ST
- IN, OUT, READ, PRINT
- JMP, CALL, RET, JEQU, JNEQ, JSUP, JINF, JSEQ, JIEQ
- STOP (qui arrête la simulation)

Exercice 1

Que fait le code présent dans le fichier [https://perso.liris.cnrs.fr/t\(pierron/archi2025/mystere.asm](https://perso.liris.cnrs.fr/t(pierron/archi2025/mystere.asm) ?

Exercice 2

Recoder l'instruction `PRINT`. Autrement dit, écrire un programme qui affiche l'entier contenu dans R_0 sur `out`, sans utiliser `PRINT`.

Exercice 3

Recoder l'instruction `MUL`. Vous utiliserez l'algorithme que vous connaissez (normalement) depuis le primaire.

Exercice 4

Écrire un programme affichant le maximum d'un tableau d'entiers.

Exercice 5

Écrire un programme qui trie un tableau d'entiers.

Exercice 6

Écrire un programme qui lit un entier n et affiche un code de Gray sur n bits.

Exercice 7

Écrire un programme qui lit un entier n et affiche une solution au problème des tours de Hanoï¹ à n disques. Votre programme écrira $2^n - 1$ lignes de la forme xy où $x, y \in \{1, 2, 3\}$ représentant le mouvement du premier disque de la tour x vers la tour y .

Exercice 8

Traduire le code C suivant en assembleur.

```
1 int somme(int n){  
2     if (n==0) {  
3         return 0;  
4     } else {  
5         return n+somme(n-1);  
6     }  
}
```

1. Traduire ce code en assembleur.
2. Cette fonction est-elle récursive terminale ?
3. Écrire une version récursive terminale de cette fonction et l'implémenter en assembleur. *On pourra utiliser un accumulateur.*
4. Dérécursiver cette fonction.

1. https://fr.wikipedia.org/wiki/Tours_de_Hano%C3%AF