


2021

Simulateur de CPU



L'objectif de cette séance est de découvrir l'assembleur *sans que ce soit trop compliqué*.

La page sur laquelle tu vas travailler est adaptée d'un simulateur de microprocesseur écrit par Peter Higginson et disponible sur <http://www.peterhigginson.co.uk/RISC/>.

Ce CPU fonctionne avec des mots de 16 bits. Chaque instruction (et ses données éventuelles) est donc codée sur 2 octets. Dans la mémoire centrale on a donc regroupé les octets par paquets de deux.

Tu disposes d'un lexique avec quelques commandes de bases à la fin de ce document.

Voici un programme ajoutant 2 nombres

Code assembleur

```
INP R0,2      // Lire un nombre au clavier et le mettre dans R0.
INP R1,2      // Lire un nombre au clavier et le mettre dans R1.
ADD R2,R1,R0  // Mettre R0 + R1 dans R2.
OUT R2,4      // Afficher R2 à l'écran.
HLT           // Stop.
```

1. Écrire ce programme dans la fenêtre *Code assembleur*, sans recopier les commentaires (les // suivis de phrases) puis cliquer sur *ASSEMBLER*. Où voit-on les instructions machine? Quelle est la longueur en octets de ce programme?

[illegible]

Cliquer sur *PAS* pour effectuer la première instruction en mode pas-à-pas. Observer la valeur de PC qui change et entrer la valeur dans la boîte de texte prévue à cet effet. Continuer à exécuter le programme en mode pas-à-pas.

2. Quelle est la plus grande somme que l'on puisse obtenir? Comment expliquer cela?

3. Modifier le code du programme pour qu'il fasse une soustraction.
 Quelle ligne faut-il changer et comment ?

Voici un deuxième programme à lire attentivement. BGE veut dire « *Branch if Greater or Equal* », ce qui peut se traduire ici par « si le résultat de la comparaison précédente indique plus grand ou égal alors va à l'adresse spécifiée ».

Code assembleur

```

    INP R0,2    // Lire un nombre au clavier et mettre dans R0.
    INP R1,2    // Lire un nombre au clavier et mettre dans R1.
    CMP R1,R0   // Comparer R1 à R0.
    BGE plusgrand // Si R1 > R0 aller à pgrand.
    OUT R0,4    // Sinon afficher R0.
    BRA fini    // Et aller à fini.
plusgrand: OUT R1,4 // Afficher R1.
fini:    HLT    // Stop.

```

5. Taper et exécuter ce programme en entrant 2 et 3.
 Qu'affiche le programme ? Dans quels états les flags sont-ils ?

6. Recommencer avec 3 et 2, puis 3 et 3.

Lexique

INP *Rx*, 2 : Saisir une valeur dans Rx
OUT *Rx*, 4 : Afficher Rx à l'écran
MOV *Rx*, *Ry* : dans Rx recopier Ry.
MOV *Rx*, #*val* : dans Rx, recopier la valeur val.
ADD *Rz*, *Ry*, *Rx* : Ajouter Rx et Ry et stocker dans Rz
SUB *Rz*, *Ry*, *Rx* : Retirer Rx à Ry et stocker dans Rz
BEQ *adr* : Si le flag Z est à 1, aller à adr.
BNE *adr* : Si le flag Z est à 0, aller à adr.

Le flag Z est mis à 1 dès qu'une opération donne 0.

Exemple : **INP** *R0*, 2
Exemple : **OUT** *R0*, 4
Exemple : **MOV** *R1*, *R2*
Exemple : **MOV** *R0*, #42
Exemple : **ADD** *R2*, *R1*, *R0*
Exemple : **MUL** *R2*, *R1*, *R0*
Exemple : **BEQ** *fin*
Exemple : **BNE** *fin*