

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

```
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?

[illegible]

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

[illegible]

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?

[illegible]

[illegible][illegible]

This image shows a full page of blank graph paper. The grid consists of small squares formed by thin, light blue lines. There are 20 columns and 20 rows of squares, creating a uniform background for drawing or writing. The margins are consistent on all sides.

Lexique

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

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`

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