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.

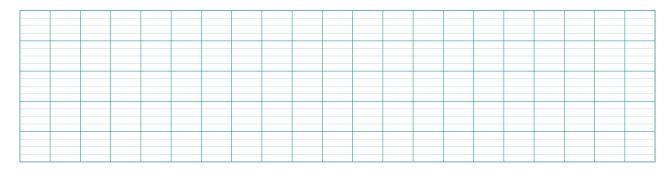
Tu disposes d'un lexique avec quelques commandes de bases à la fin de ce document.

Voici un programme ajoutant 2 nombres

Code assembleur

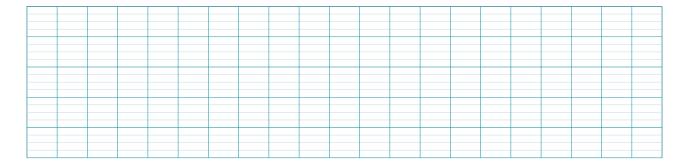
```
INP R0,2 // Lire un nombre au clavier et le mettre dans R0.
INP R1,2 // Lire un nombre au clavier et le mettre dans R1.
ADD R2,R1,R0 // Mettre R0 + R1 dans R2.
OUT R2,4 // Afficher R2 à l'écran.
HLT // Stop.
```

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?

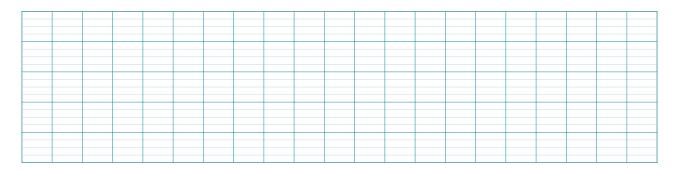


Cliquer sur *PAS* pour effectuer la première instruction en mode pas-à-pas. Observer la valeur de PC qui change et entrer la valeur dans la boîte de texte prévue à cet effet. Continuer à exécuter le programme en mode pas-à-pas.

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?

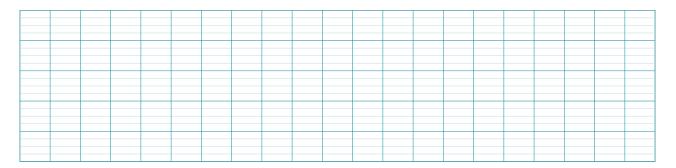


Voici un deuxième programme à lire attentivement. BGE veut dire « *Branch if Greater or Equal* », ce qui peut se traduire ici par « si le résultat de la comparaison précédente indique plus grand ou égal alors va à l'adresse spécifiée ».

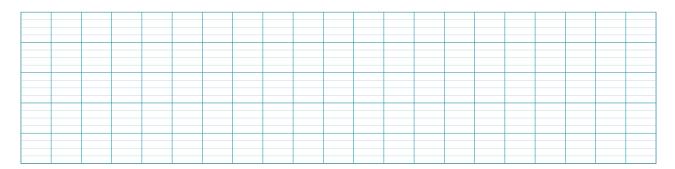
Code assembleur

```
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.
```

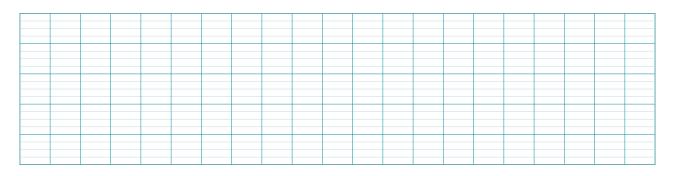
5. Taper et exécuter ce programme en entrant 2 et 3. Qu'affiche le programme? Dans quels états les flags sont-ils?



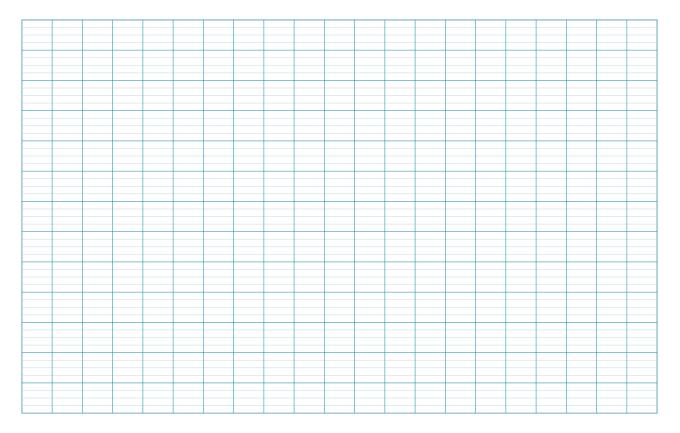
6. Recommencer avec 3 et 2, puis 3 et 3.



7. Émettre une hypothèse sur la signification des flags.



8. Écrire un petit programme qui demande un entier *n* au clavier, le stocke dans R0, puis calcule dans R1 la somme de tous les entiers de 1 à *n* : il suffit de mettre R1 à 0 puis créer une boucle : on ajoute R0 à R1 et on enlève 1 à R0. Si R0 n'est pas nul on boucle, sinon on affiche le résultat et on s'arrête.



Lexique

INP	Rx,2	: Saisir une valeur dans Rx	Exemple: INP	R0,2
OUT	Rx,4	: Afficher Rx à l'écran	Exemple : OUT	R0,4
MOV	Rx,Ry	: dans Rx recopier Ry.	Exemple : MOV	R1,R2
MOV	Rx,#val	: dans Rx, recopier la valeur val.	Exemple : MOV	R0,#42
ADD	Rz,Ry,Rx	: Ajouter Rx et Ry et stocker dans Rz	Exemple : ADD	R2,R1,R0
SUB	Rz,Ry,Rx	: Retirer Rx à Ry et stocker dans Rz	Exemple : MUL	R2,R1,R0
BEQ	adr	: Si le flag Z est à 1, aller à adr.	Exemple : BEQ	fin
BNE	adr	: Si le flag Z est à 0, aller à adr.	Exemple : BNE	fin

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