Modèle d'architecture séquentielle

Activité n°2 : Simulateur CPU Von Neumann

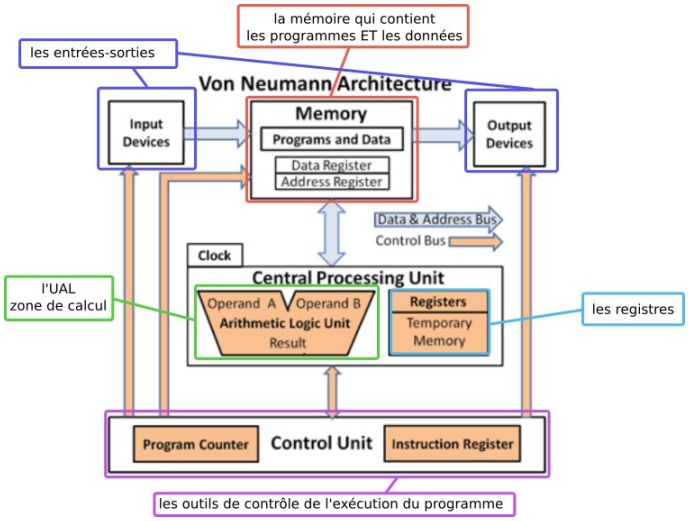
1. Structure des ordinateurs

Un ordinateur est une machine programmable, automatique et universelle :

- **programmable** : la séquence d'opérations exécutée par un ordinateur peut être entièrement spécifiée dans le texte d'un programme,
- **automatique** : un ordinateur peut exécuter un programme sans intervention extérieure (câblage, …),
- **universelle** : un ordinateur peut exécuter tout programme calculable (selon la théorie de Turing) avec le jeu d'instructions câblé dans son processeur.

En 1945, John Von Neumann, mathématicien hongrois exilé aux États-Unis, publie un rapport sur la réalisation du calculateur EDVAC où il propose une architecture permettant d'implémenter une machine universelle, décrite par Alan Turing dans son article fondateur de 1936 sur le problème de l'indécidabilité.

L'architecture de Von Neumann va servir de modèle pour la plupart des ordinateurs de 1945 jusqu'à nos jours, elle se compose de quatre parties distinctes :



✗ L'**Unité Centrale de Traitement** (Central Processing Unit en anglais) ou Processeur est constituée de deux sous-unités :

- L'**Unité de Commande** (Control Unit) :
  - ■ charge la prochaine instruction dont l'adresse mémoire se trouve dans un registre appelé **Compteur de Programme** ou Compteur ordinal (Program Counter ou PC en anglais),
  - ■ la stocke dans le **Registre d'Instruction** (Instruction register),
  - ■ la décode avec le décodeur et commande l'exécution par l'**ALU** avec le séquenceur OU effectuer une opération de branchement (un saut dans le programme), en modifiant le Compteur de Programme, qui par défaut est incrémenté de 1 lors de chaque instruction.

- L'**Unité Arithmétique et Logique** (**ALU** en anglais) :
  - ■ qui réalise des opérations arithmétiques (addition, multiplication, …), logiques (et, ou, …), de comparaisons ou de déplacement de mémoire (copie de ou vers la mémoire).
  - ■ L'ALU stocke les données dans des mémoires d'accès très rapide appelées **registres**.
  - ■ Les opérations sont réalisées par des circuits logiques constituant le **jeu d'instructions du processeur.**

✗ La **mémoire** où sont stockés les **données** et les **programmes** (Von Neumann).

✗ Des **bus** qui sont des fils reliant le **CPU** et la **mémoire** et permettant les échanges de données et d'adresses.

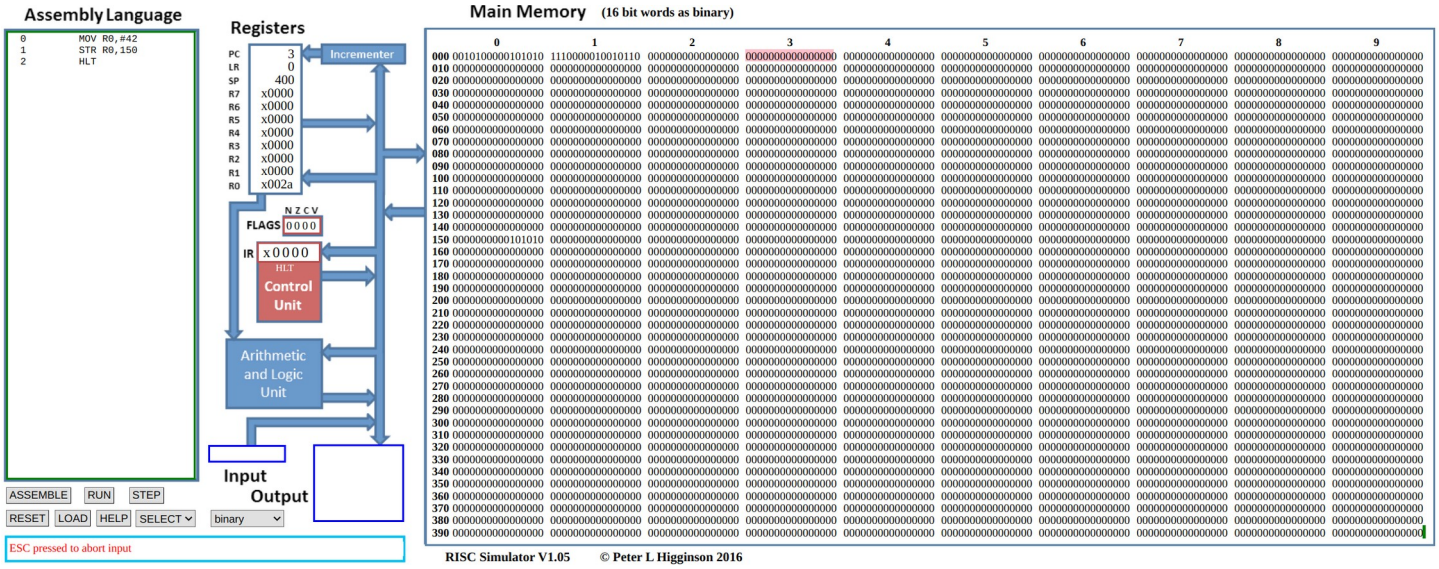
✗ Des dispositifs d'**entrées/sorties** permettant d'échanger avec l'extérieur (lecture ou écriture de données).

Source : http://frederic-junier.org/NSI/premiere/chapitre17/NSI-ArchitectureVonNeumann-Cours2020V2.pdf

2. Simulateur CPU

Nous utiliserons un simulateur en ligne développé par Peter L Higginson, basé sur une architecture de von Neumann. Il est accessible sur le lien suivant https://www.peterhigginson.co.uk/RISC/. La documentation du jeu d'instruction du CPU simulé est accessible au lien http://www.peterhigginson.co.uk/RISC/instruction_set.pdf. Vous dispose également d'un bouton «HELP».

L'utilisation de Chrome est préférable pour accéder à la ressource .



2.1. **Double cliquer** dans la zone « Assembly Language » et **saisir** le code assembleur ci-dessous, **valider** la saisie avec les boutons « submit » puis « Assemble ». **Choisir** le mode « hexadécimal » dans le choix « option ». Que se passe-t-il dans la mémoire aux adressex 000, 001 et 002 ?

```
MOV R0,#42
STR R0,150
HLT
```

.....

.....

.....

2.2. **Exécuter** pas à pas à l'aide du bouton « step ». Que fait ce programme ?

.....

.....

.....

2.3. **Conclure** sur le type d'architecture (Von Neumann ou Harvard)

.....

.....

.....

2.4. La documentation du jeu d'instructions du processeur RISC précise le codage de l'instruction « MOV Rd, #immediate » et « STR Rs, direct » en binaire :

	MOV					Rd			#immediate							
MOV R0,#42	0	0	1	0	1	0	0	0	0	0	1	0	1	0	1	0

	STR				Rd				direct							
STR R0,150	1	1	1	0	0	0	0	0	1	0	0	1	0	1	1	0

Compléter le tableau ci-dessous afin de charger la valeur 62 dans la dernière case mémoire (adresse 399) à l'aide du registre R7. **Modifier** directement la zone mémoire afin de mettre à jour votre programme. **Tester**.

	MOV					Rd			#immediate							
MOV R7,#62	0	0	1	0	1	**1**	**1**	**1**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**0**

	STR				Rd				direct							
STR R7,399	1	1	1	0	**1**	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**1**	**1**	**1**	**1**

2.5. À l'aide de la liste de choix « select », **charger** le programme « add » et **l'exécuter. Préciser** ce que fait ce programme.

.....

.....

.....

2.6. Boucle « pour » : Le programme assembleur est proposé ci-dessous. Le **saisir** et **l'exécuter** afin de **proposer** une version python équivalente.

Version assembleur

```
    INP R1,2
    MOV R0,#0
FOR_LOOP: OUT R0,4
    ADD R0,#1
    CMP R0,R1
    BLT FOR_LOOP
FIN_POUR: HLT
```

Version python

2.7. Boucle « pour » : **Modifier** le programme assembleur de la question 2.5 afin qu'il demande et affecte la borne inférieure de la boucle « pour » dans le registre R2. On cherche à imiter l'instruction python « for R0 in range (R2,R1) ».

Variante : Un raffinement serait de vérifier que R1 > R2 et de n'exécuter la boucle que dans ce cas !!!

2.8. Boucle « pour » : **Modifier** le programme assembleur de la question 2.5 afin qu'il calcule la somme des premiers entiers dans le registre R3, l'affiche en fin de programme et le stocke dans la mémoire à l'adresse 0x100.

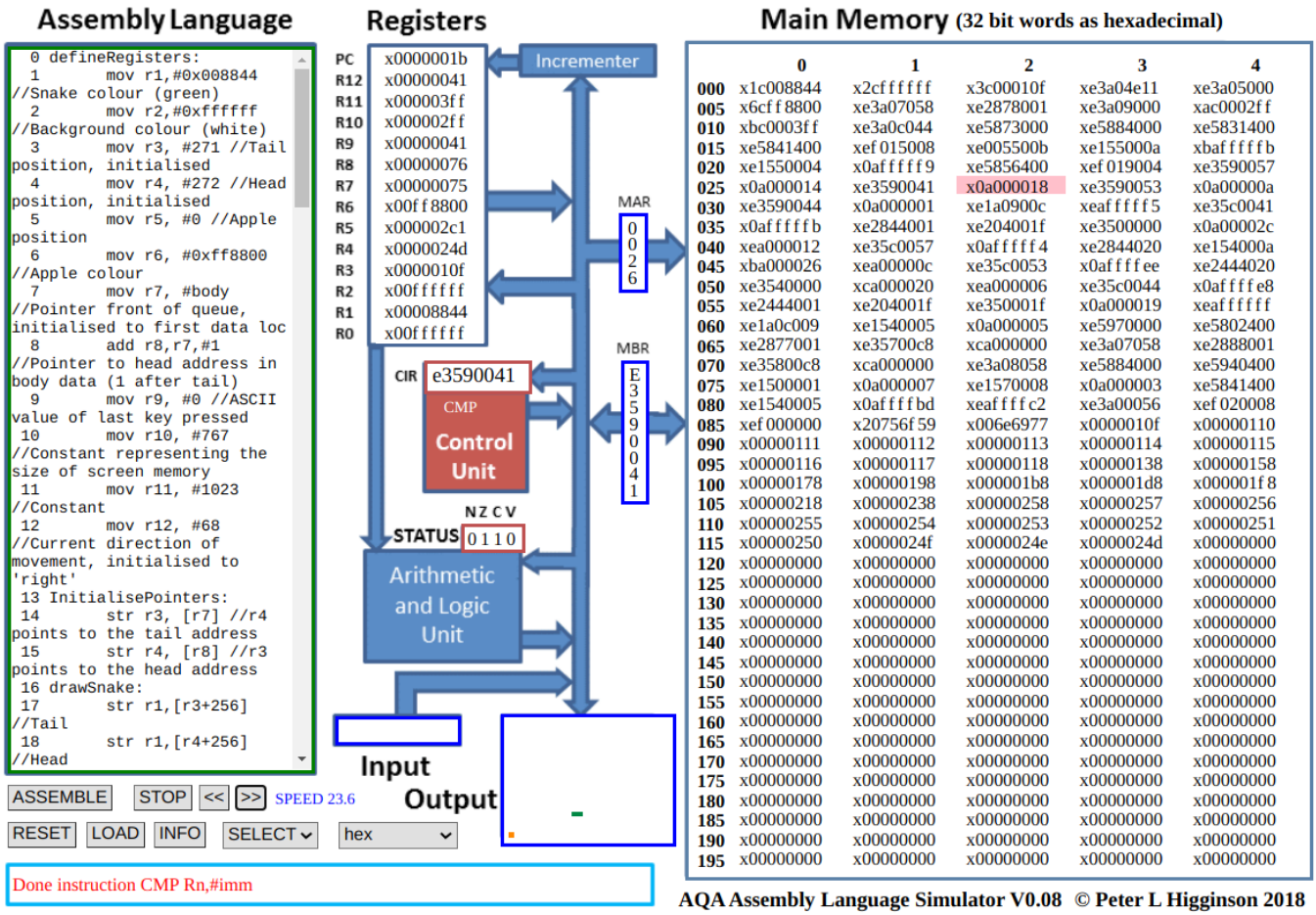
2.9. Boucle «inconnu» : Le programme assembleur est proposé ci-dessous. Le **saisir** et **l'exécuter** afin de **proposer** une version python équivalente. Afin de valider la bonne structure algorithmique réalisée, vous pouvez modifier la première instruction MOV R0,#5 par MOV R0,#0

Version assembleur	*Version python*

MOV R0,#5
Loop_Begin: CMP R0,#0
BLS Loop_End
OUT R0,4
SUB R0,#1
BRA Loop_Begin
Loop_End: HLT	

2.10. Richard Pawson, co-auteur de l'émulateur en ligne, a développé le jeu « snake » pour celui-ci. Il s'appuie sur l'émulateur simplifié disponible sur https://www.peterhigginson.co.uk/AQA/ et le code assembleur commenté est disponible sur son github, (et dans les fichiers de TP)

https://raw.githubusercontent.com/richardpawson/education/master/Snake/Snake/snake.txt



Lancer le simulateur simplifié AQA à partir du lien ci-dessus, **charger** le fichier assembleur snake.txt à l'aide du boutton « LOAD », **lancer** la simulation et **régler** la vitesse au maximum à l'aide du bouton « >> ». Les déplacements sont obtenus à l'aide des touches w (Monter), s (Descendre), a (Gauche) et s (Droite). Faites-vous plaisir ...