Architecture des ordinateurs

Le but de cette fiche est la découverte du fonctionnement général d'un ordinateur actuel.
En fait, tous les ordinateurs actuels fonctionnent selon un même principe de base. On dit qu'ils ont une **architecture von Neumann**, du nom de l'un de ceux qui en ont décrit le fonctionnement, qui en attribuait lui-même la paternité à Alan Turing.

1. Architecture von Neumann

Nous utiliserons ici le simulateur de machine von Neumann en ligne disponible à l'adresse :

https://apps.lyceum.fr/vnsim/

Voici une description des éléments visibles sur la page :

- À droite se trouve la mémoire : elle contient d'une part le programme et d'autre part quatre variables X, Y, Z et W.
- À gauche se trouve le processeur :
  - le Registre d'Instruction (RI) contient l'instruction en cours d'exécution ;
  - le Compteur de Programme (CP) contient le numéro de la prochaine instruction à exécuter ;
  - l'Accumulateur (ACC) stocke des valeurs en mémoire ;
  - l'Unité Arithmétique et Logique (UAL) effectue les calculs. Elle a deux entrées (en haut) et une sortie qui est redirigée dans l'accumulateur.

Les différents éléments (valeurs en mémoire, numéros des adresses mémoire) utilisés sont transportés (de la mémoire au processeur et vice-versa) par des « bus ».

Exercice 1
Dans le menu « Exemples », ouvrir `ADDITION`.
Ce programme est formé de quatre instructions :

- LOD X : charger (« load ») la valeur de X dans l'accumulateur ACC ;
- ADD Y : ajouter à l'accumulateur la valeur de Y ;
- STO Z : stocker (« store ») le contenu de l'accumulateur dans Z ;
- HLT : s'arrêter (« halt »).

1. Faire glisser la tirette en haut à droite pour qu'elle indique 1000ms (ou davantage) pour ralentir l'exécution.
2. Cliquer alors sur le bouton « Faire un pas » (▶|) et observer ce qui se passe.
   En particulier :
   (a) D'où vient le contenu du registre d'instruction (RI) ?
   (b) D'où viennent les valeurs en entrée et/ou en sortie de l'Unité arithmétique et logique (UAL) ?
   (c) Quelles sont les valeurs du compteur de Programme (CP) et de l'accumulateur (ACC) à l'issue de l'étape exécutée ?
3. Refaire de même que précédemment étape par étape jusqu'à la fin du programme.

Pour recommencer le programme, il faut réinitialiser le compteur de programme (CP) au numéro de cellule du départ (0 ici).

Le programme précédent était très simple : aucune instruction conditionnelle, aucune boucle.
Dans le langage machine, il n'y a pas vraiment d'instruction `if`, et encore moins de `while` ou de `for` comme en Python ou autre langage de haut niveau.
Les instructions qui permettent de faire des boucles et de vérifier des conditions sont le saut (JMP : « jump ») et le saut conditionnel (JMZ : « jump if zero »), qui permettent d'aller à une cellule mémoire quelconque, pour JMZ à la condition que l'accumulateur vaille 0.

Exercice 2
Ouvrir l'exemple EQUAL pour observer le fonctionnement de l'instruction JMZ.

1. Modifier les valeurs de X et/ou Y pour observer le changement de comportement selon que leurs valeurs sont égales ou non.
2. Comment fonctionne en détail ce programme ?

2. Ordinateur Cardiac

Les exercices suivants portent sur un « ordinateur » particulier, le **Cardiac**, créé dans le but d'enseigner, aux étudiants d'informatique à l'Université de Strasbourg, le langage machine. Il en existe une version papier, qui n'est pas vraiment un ordinateur puisque, lorsqu'on l'utilise, c'est à nous de réaliser une partie des opérations (dont une partie du contrôle et les calculs), mais les actions s'y enchaînent comme sur un ordinateur réel.
Nous utiliserons une version en ligne de cet ordinateur, disponible en suivant le lien ci-dessous :

simulateur Cardiac

C'est le même principe que celui vu dans la section précédente, mais sans le schéma et avec des instructions différentes. Par exemple, il n'y a pas d'opérateur de comparaison de nombres.
Selon les architectures, les machines peuvent en effet avoir des jeux d'instructions de base différentes.
D'autre part, le compteur de programme (CP) est ici en fait appelé Compteur ordinal (CO).
La grande différence ici est que la mémoire ne contient que des nombres (heureusement ici pour nous en décimal ; il faut penser que dans la machine ce n'est que du binaire), et donc que les opérations ont un code (de 0 à 9). Si une valeur en cellule mémoire correspond à une instruction, le premier chiffre (à gauche) est le code de l'instruction, et les deux suivants sont le numéro de cellule mémoire associé.
D'autre part, l'ordinateur Cardiac ne manipule pas de nombres négatifs. Ainsi, si le résultat mathématique de la soustraction d'un nombre par un autre est négatif, l'ordinateur retourne la valeur 0 dans l'accumulateur.

Le principe à appliquer pour l'exercice suivant sera le suivant :

1. Copier dans la mémoire et/ou le ruban d'entrée les informations initiales données.
   Il faut ensuite charger le ruban dans la machine (bouton « Charger ruban »).
2. Mettre la valeur du compteur ordinal comme précisé.
3. Exécuter pas à pas les instructions et bien réfléchir au fonctionnement à chaque étape. Pour ce faire, on pourra regarder le fichier contenant le patron du modèle papier, qui indique l'ordre des actions à effectuer à chaque pas, puis plus tard la documentation du Cardiac.

Exercice 3
Initialiser tout d'abord l'ordinateur en suivant les instructions suivantes :

- Sur le ruban d'entrée, placer les valeurs suivantes dans cet ordre :

  ```
  023
  010
  ```

  Puis charger le ruban (« Charger ruban ») ;

- Sur la mémoire centrale, placer le contenu suivant aux numéros de cellules indiqués :

20. 034	23. 635	26. 136
21. 035	24. 628	27. 900
22. 234	25. 436	28. 009

- Enfin, mettre le compteur ordinal sur la valeur 20.

1. Faire fonctionner le Cardiac pas à pas jusqu'à l'arrêt de l'ordinateur. Que fait-il ?
2. Pouvait-on prévoir avant l'exécution si les cellules mémoire entre les adresses 20 et 28 contenaient des instructions ou des données ?
3. Dans ce programme, où sont les instructions et où sont les données ?

Exercice 4
Écrire en **langage machine**, c'est à dire sous la forme de nombres décimaux sur trois caractères devant se trouver à certaines adresses mémoire, un programme Cardiac lisant un nombre sur le ruban d'entrée et imprimant 1 sur le ruban de sortie si ce nombre est supérieur ou égal à 10 ou 0 s'il est strictement inférieur. Pour cela, il faudra utiliser le code d'opération 8 (JAZ). Tester le programme après avoir pensé à mettre une valeur sur le ruban d'entrée et la bonne valeur initiale du compteur ordinal (autrement dit le numéro de cellule du début du programme).

Plutôt que d'utiliser le langage machine, qui n'est pas très parlant pour les humains (à moins peut-être d'y être plongé depuis des années), on utilise un langage intermédiaire, le **langage assembleur** (celui que l'on a vu dans la première partie de l'activité). Pour cela, on donne un nom aux instructions du processeur (voir le jeu d'instructions sur la partie principale du processeur Cardiac). On peut également nommer des lignes (donc en particulier utiliser des noms de variables). Une traduction en langage assembleur du programme exécuté au tout début est :

```
    .at 20
    INP 34
    INP 35
    LDA 34
    ADD 35
    ADD n
    STA 36
    OUT 36
    HRS 00
n:  .word 009
```

ou, en nommant les espaces mémoire utilisés (et en les initialisant, ici à 000) :

```
    .at 20
    INP a
    INP b
    LDA a
    ADD b
    ADD n
    STA x
    OUT x
    HRS 00
n:  .word 009

    .at 34
a:  .word 000
b:  .word 000
x:  .word 000
```

Voir le fichier de documentation de Cardiac pour plus d'informations.

Exercice 5
Traduire en langage assembleur le programme écrit à l'exercice 4.
Tester le programme en le chargeant sur le simulateur.

Exercice 6 (Facultatif)
Écrire un programme en assembleur Cardiac lisant un nombre `n` sur le ruban d'entrée et écrivant la suite des nombres de 1 à `n` sur le ruban de sortie.

Exercice 7 (Facultatif)
Écrire un programme en assembleur Cardiac calculant la multiplication de deux nombres lus sur l'entrée et écrivant le résultat sur la sortie.

Python dispose d'une bibliothèque permettant de traduire un code Python en langage assembleur de l'ordinateur (qui n'est pas le même que le langage Cardiac, bien sûr). Cette bibliothèque est appelée `dis`.

Exemple Copier et exécuter le code Python suivant (attention à bien indenter) :

```
import dis

code = """
x=3
if x<0:
    y=-x
else:
    y=x
"""

dis.dis(code)
```