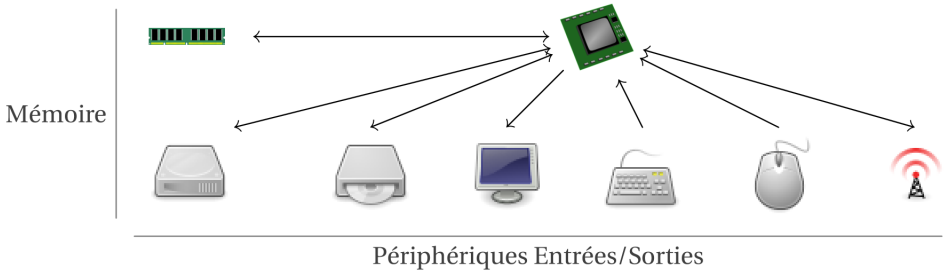
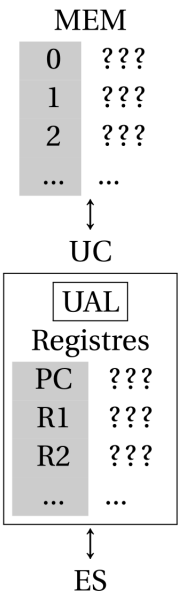
27 Architecture d'un ordinateur

Remarque. On présente quelques modèles qui permettent de décrire et comprendre le fonctionnement d'un ordinateur et nous évitent de rentrer dans les détails techniques. □



◇ *L'architecture de von Neumann* est un modèle abstrait pour décrire le fonctionnement d'un ordinateur auquel se conforment la plupart des ordinateurs actuels à quelques adaptations près. Dans ce modèle, l'ordinateur est décrit par 4 composantes :

- La *mémoire* (MEM), dans laquelle se trouvent *à la fois* les programmes à exécuter et les données (ce point est fondamental). Les différents emplacements de la mémoire sont repérés par leur *adresse* ;
- L'*unité de contrôle* (UC) est chargée de lire des intructions dans la mémoire et de les réaliser ;
- L'*unité arithmétique et logique* (UAL) est en particulier chargée de réaliser les opérations de calcul ;
- Les *dispositifs d'entrées-sorties* (ES) regroupent les différents périphériques (que l'on ne distingue pas dans ce modèle).



Les *instructions* réalisées par l'UC peuvent être :

- Des intructions de lecture ou d'écriture en mémoire ;
- Des opérations à effectuer (qui seront réalisées par l'UAL) ;
- Des instructions d'échange de données avec les périphériques.

L'UC dispose d'un certain nombre d'emplacements mémoire qui lui sont spécifiques et que l'on appelle *registres*.

Exemple. Imaginons que l'UC possède 3 registres notés R1, R2 et PC (pour *program counter*, ce registre contient l'adresse mémoire de la prochaine instruction à exécuter). Les instructions réalisables par l'UC sont représentées par un *opcode* (un nombre) mais on leur donne aussi un *nom* pour plus de clarté. On peut par exemple imaginer que l'UC comprend les instructions suivantes :

Opcode	Nom	Description
1	READ_R1 a	Recopie le contenu de l'adresse mémoire a dans le registre R1
2	WRITE_R1 a	Recopie le contenu du registre R1 à l'adresse mémoire a
3	INPUT x	Lis un nombre au clavier et l'écrit dans le registre Rx ($x = 1$ ou $x = 2$)
4	PRINT x	Affiche à l'écran le contenu du registre Rx ($x = 1$ ou $x = 2$)
5	ADD x	Calcule la somme des registres R1 et R2 et met le résultat dans le registre Rx ($x = 1$ ou $x = 2$)
6	R1Z_JUMP a	Si le contenu du registre R1 est nul, la prochaine instruction à exécuter est celle située à l'adresse a

On voit que chaque instruction est constitué par 2 nombres : l'*opcode* de l'instruction et une adresse (ou un numéro de registre). Lorsque l'on passe à l'instruction suivante, le registre PC doit donc être augmenté de 2 *sauf* lorsque l'instruction est R1Z_JUMP a et le contenu de R1 est nul : dans ce cas, la nouvelle valeur de PC est a.

◇ De ce point de vue, un programme apparait comme étant une suite de nombres qui représentent des codes d'instructions ou des adresses : c'est ce que l'on appelle le *langage machine*. Il est quasiment impossible d'écrire directement un tel programme, on utilise donc un *assembleur* : son rôle est de traduire un programme décrit symboliquement par les noms des instructions et des variables en langage machine.

◇ Écrire en assembleur est également fastidieux, on a donc inventé des langages de plus haut niveau (Pascal, C, etc.) et des *compilateurs* qui se chargent de traduire les programmes écrits dans ces langages en assembleur (qui est ensuite traduit en langage machine).

◇ Le *système d'exploitation* : c'est le premier programme qui est lancé lorsque l'ordinateur est allumé (les exemples les plus courants : Windows, OS X et GNU/Linux). Son rôle :

- Lancer les autres programmes et gérer éventuellement leur exécution simultanée (en fait, à tour de rôle) ;
- Permettre aux programmes d'accéder aux périphériques de manière simple ;
- Gérer les disques durs, en particulier leur organisation en répertoires et fichiers.

Tous les autres programmes vont être exécutés « à l'intérieur » du système d'exploitation. Suivant les cas, il est possible que d'autres fonctions soient attribuées au système d'exploitation (gérer le système de fenêtrage, les différents utilisateurs, etc.).

Remarque. Un programme produit par un compilateur est lié à un type particulier d'ordinateur mais aussi au système d'exploitation. Les langages *interprétés* comme PYTHON ne sont pas compilés mais ils tournent grâce à un *interpréteur* (l'interpréteur est spécifique à un type d'ordinateur et un système d'exploitation). □