Les bases de l'Informatique
Les langages

Thierry Vaira

BTS SN

v1.0 - 12 septembre 2016

Comment « parler » au processeur ?

- Un processeur « parle » le **langage machine** :
  - C'est une suite de bits interprétée par le processeur.
  - C'est le seul langage que le processeur puisse traiter.
  - Il est composé d'**instructions** et de **données** à traiter codées en binaire.
  - Exemple : l'instruction `10110000 01100001` met la valeur hexadécimale 61 dans le registre `AL`. Ici, `10110000` correspond à l'***opcode*** et `01100001` à la **donnée**.
  - Un processeur possède nativement un **jeu d'instructions** composé d'***opcode*** (code opération) et d'opérandes. Il existe des *opcodes* pour faire des opérations arithmétiques, logiques, etc ... Chaque instruction nécessite un certain temps (généralement un nombre de cycles d'horloge) pour s'exécuter.

Comment peut-on « parler » au processeur ?

- Le langage informatique le plus proche du langage machine est l'**assembleur** :
  - ➟ C'est un langage de bas niveau qui représente le langage machine sous une forme lisible par un humain.
  - ➟ Les combinaisons de bits du langage machine sont représentées par des **mnémoniques** (des symboles) faciles à retenir : `ADD` pour l'addition, `MOV` pour la copie de valeurs, etc ...
  - ➟ Exemple : l'instruction binaire `10110000 01100001` sera `movb $0x61,%al` en assembleur.

➟ *Il est en théorie possible de désassembler le code machine d'un programme. En pratique, c'est plus compliqué que cela car il est parfois difficile de savoir si une valeur en mémoire est un opcode ou une donnée. Dans tous les cas, on ne pourra jamais « remonter » plus haut que l'assembleur.*

Comment faire pour « parler » au processeur ?

- Les contraintes sont les suivantes :
  - ➟ L'ordinateur ne sait exécuter qu'un nombre limité d'opérations élémentaires représentées par des instructions codées en binaire
  - ➟ L'ordinateur ne comprend que le langage machine.
- On est donc obliger d'utiliser des « outils » pour traduire un langage simple vers le langage machine.
- On a ensuite créer des langages de programmation évolués (le plus proche possible de l'« humain ») et utilisables sur n'importe quel ordinateur.

➟ *L'arrivée des premiers langages évolués comme le Cobol et le Fortran datent des années 1950.*

Les langages de programmation

- Quel que soit le langage évolué, il faut le **traduire en langage machine**. Il existe des approches différentes :
  - **Assemblage** (assembleur) : traduction du mnémonique (symbole) en *opcode*.
  - **Compilation** (Pascal, C, C++, ...) : traduction de l'ensemble du programme écrit en langage évolué (**code source**) en langage machine (**code objet**). Le programme en langage machine sera ensuite exécuté.
  - **Interprétation** (Javascript, PHP, Python, ...) : traduction de chaque instruction avant de l'exécuter. On nomme souvent ces programmes des **scripts**.
  - **Compilation et Interprétation** (Java) : compilation en un code intermédiaire (*bytecode*) qui n'est pas du code machine mais un code pour une « machine virtuelle ». Ce code est ensuite interprété par un interpréteur (la machine virtuelle).

➠ *Les langages n'ont pas les mêmes performances. On utilise pas les mêmes langages pour faire un OS, un tableur, un site web ou une application pour smartphone.*

Pourquoi faire ?

- Le processeur ne sait que **traiter des données**.
  - ➟ Le traitement est assuré par l'exécution d'instructions simples codées en binaire.
  - ➟ Les données seront n'importe quelle information codée en binaire (une « valeur numérique »).
- On rassemble tout cela à l'intérieur d'un **programme** et finalement :
  - ➟ un programme ne fera que **traiter des données** ...
  - ➟ un ordinateur ne servira qu'à **traiter des données** ...
  - ➟ l'informatique ce n'est que ... **traiter des données** !

Traitons des données I

- Pour manipuler des données, les langages de programmation nous offrent le concept de **variable**.
- Une variable est un **espace de stockage pour un résultat**.
- Une variable est associé à un **symbole** (habituellement un **nom** qui sert d'identifiant) qui renvoie à une **position de la mémoire** (une **adresse**) dont le contenu peut prendre successivement différentes valeurs pendant l'exécution d'un programme.
- De manière générale, les variables ont :
  - un **type** : c'est la convention d'interprétation de la séquence de bits qui constitue la variable. Le type de la variable spécifie aussi sa **taille** (la longueur de cette séquence) soit habituellement 8 bits, 32 bits, 64 bits, ... Cela implique qu'il y a un **domaine de valeurs** (ensemble des valeurs possibles).
  - une **valeur** : c'est la séquence de bits elle même. Cette séquence peut être codée de différrentes façons suivant son type.

Des données numériques

Le caractère '2' :

Caractère '2'

0	0	1	1	0	0	1	0

50 ou 0x32

Code ASCII

L'entier 2 :

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

signe ... 2^3 2^2 2^1 2^0

Le réel 2,0 :

0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

signe **exposant** **mantisse**

La chaîne de caractères "2" :

Caractère '2'

0	0	1	1	0	0	1	0

50 ou 0x32

Code ASCII

Caractère NUL '\0'

0	0	0	0	0	0	0	0

0 ou 0x00

Fin de chaîne

Les types scalaires

- Une **variable scalaire** est destinée par son **type** à contenir une **valeur atomique**.
- Les valeurs atomiques sont :
  - les **booléens**
  - les **nombres entiers**
  - les **nombres à virgule flottante**

Remarque : les **caractères** sont en fait des nombres entiers ! En effet, on est obligé de convertir les caractères sous une forme binaire qui constitue un code. Par exemple, le code ASCII du caractère `'A'` est 65 (`0x41` en hexadécimal).

Les types composés

- Il est possible de regrouper des types scalaires dans des **structures de données**.
- Le type composé de base est le **tableau**.
- Il existe aussi parfois la **table associative** (association clé/valeur).
- La **chaîne de caractères** peut être considérée comme un type composé.
- On verra ensuite des **collections** de données (arbre, liste, ...).

Traitons des données II

- Pour traiter des données, il faut pouvoir les manipuler et faire des opérations sur celles-ci.
- Les langages de programmation nous offrent le concept d'**instructions** dans lesquelles on pourra utiliser des **opérateurs**.
- On rappelle que les instructions que l'ordinateur peut comprendre ne sont pas celles du langage humain. Le processeur sait juste exécuter un nombre limité d'instructions bien définies (son jeu d'instructions).
- Des instructions typiques comprises par un ordinateur sont par exemple :
  - copier le contenu de la case 123 et le placer dans la case 456
  - ajouter le contenu de la case 321 à celui de la case 654
  - placer le résultat dans la case 777
  - si le contenu de la case 999 vaut 0, mettre 0 dans la case 345

L'affectation

- La première des instructions est la possibilité d'**affecter une valeur à une variable**. La plupart des langages proposent l'**opérateur d'affectation =** :