Université de Lausanne
Sections ESC et IMM

Informatique I
Introduction à la programmation

Modalités générales

Cours et exercices

- Cours:

  Mardi, 10h15 à 12h, Amphipôle D

- Exercices (répartis en trois groupes):

  De A à E: mardi, 13h15 - 15h, Amphipôle 140-146

  De F à M: mardi, 15h15 - 17h, Amphipôle 140-146

  De N à Z: mercredi, 8h15 - 10h, Amphipôle 140-146

- Site web: http://moodle2.unil.ch/course/view.php?id=826

- Responsables du cours:
  - Prof. Alesandro Villa: Alessandro.Villa@unil.ch
  - Prof. Jérémie Cabessa: Jeremie.Cabessa@unil.ch

- Assistants:
  - Vincent Buntinx: vbuntinx@romandie.com
  - Alberto Antonioni: Alberto.Antonioni@unil.ch
  - David Correia Saaverdra: David.CorreiaSaavedra@unil.ch
  - Nicolas Gailly: nikkolasg@gmail.com

Modalités d'examen

- Test intermédiaire (mi-semestre)
  - Ecrit
  - Facultatif
  - Bonus:
    - 0.5 pt si la note est entre 4.5 et 5.4
    - 1.0 pt si la note est entre 5.5 et 6
- Examen (session janvier 2013)
  - Ecrit
  - Obligatoire
- Examen de rattrapage (session août 2013)
  - Ecrit ou oral (suivant le nombre d'inscrits)

Chapitre 0: Introduction

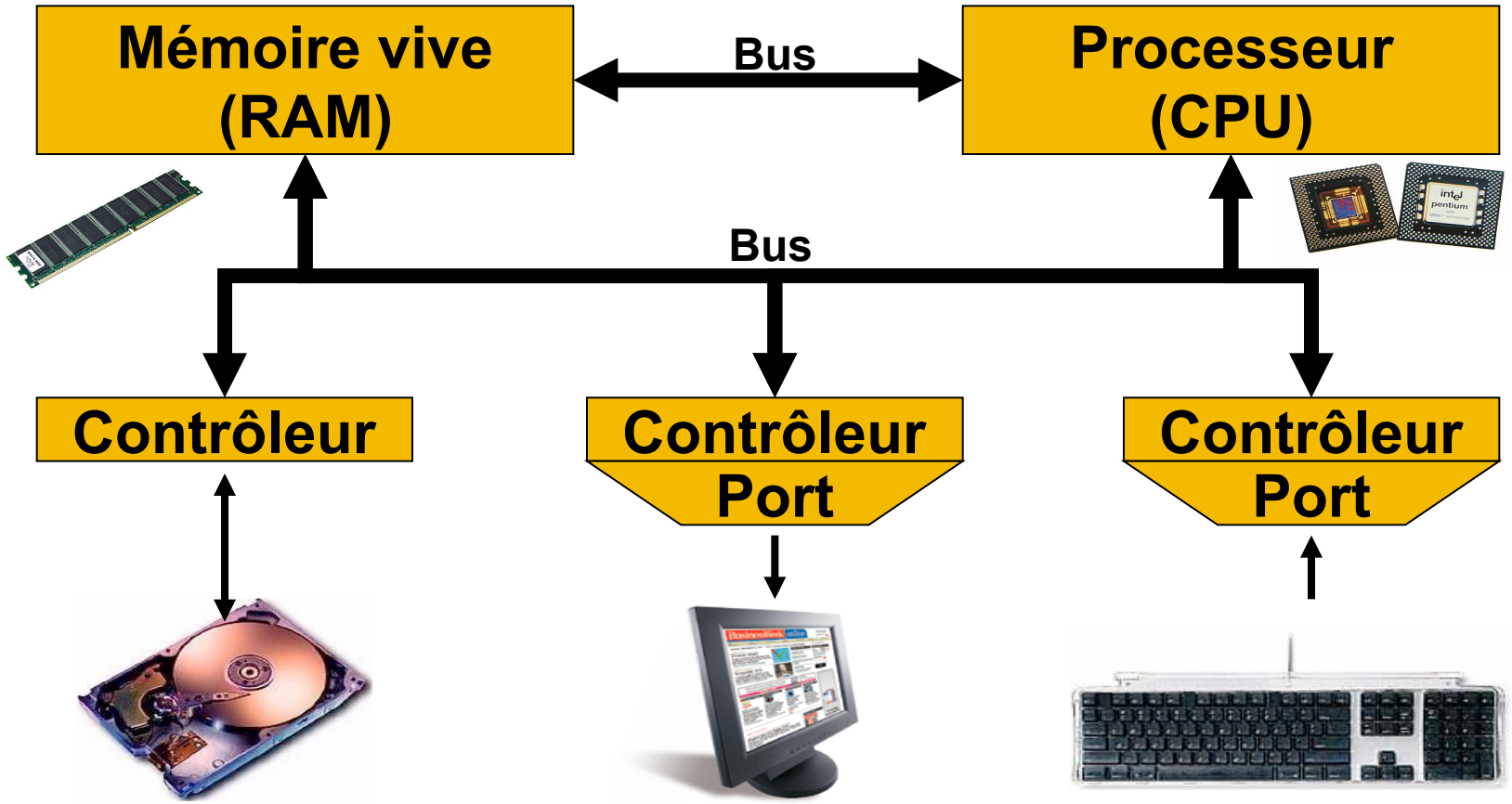
0.1 L'ordinateur personnel

Un ordinateur est un “organisme” constitué de deux types de composants:

- Hardware (éléments matériels)
  - processeur principal
  - coprocesseurs (graphique, arithmétique, etc.)
  - périphériques (clavier, écran, souris, etc.)
  - boîtier (titane, fer, plastique, etc.)
  - alimentation (transformateur, batteries, etc.)
  - Etc.

- Software (éléments non-matériels)
  - système d'exploitation
  - Programmes
  - virus
  - etc.

0.2 L'intérieur de l'ordinateur



0.3 Les systèmes d'exploitation

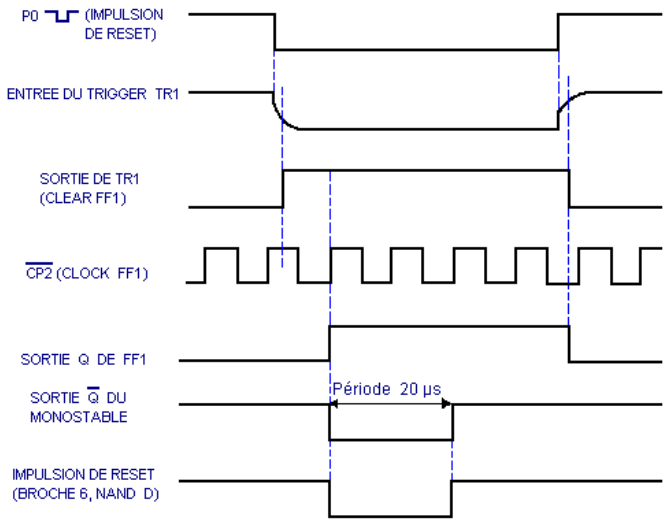
Un **système d'exploitation** est un programme qui permet d'accéder aux *ressources physiques* de l'ordinateur (clavier, écran, souris, disque dur, mémoire, etc.) et aux *ressources logiques* (les fichiers) [**file manager**]

En 2012 il y a trois familles de systèmes d'exploitation qui dominent le marché:

- **Windows** (Windows 7, Windows 8, Vista XP, 2000, etc.) par Microsoft
- **Mac OS** (OS X, OS 9.x) par Apple
- **Linux**, logiciel dans le domaine public, avec des présentations différentes pour l'usager
  - KDE
  - Gnome
  - ...

0.4 Numérisation des données

- Un ordinateur ne fonctionne qu'à partir de signaux électriques binaires (potentiels électriques min ou max).



- Ainsi, pour être traitée par l'ordinateur, toute information doit être **codée** sous forme numérique binaire
- En fait, tout alphabet fini peut être encodé en binaire.
- De même, tout instruction peut être encodée en binaire.

- Pour fonctionner, un ordinateur ne fait rien d'autre que de décoder continuellement le flux d'informations binaires qu'il reçoit.

- **Langage naturel:**
  - Alphabet latin: A, B, C, D, …

encodage

- **Langage binaire:**
  - Alphabet binaire: 0, 1
  - Un chiffre binaire est appelé **BIT** (**Bi**nary digi**T**)

0.5 Représentation en binaire

- Nombre décimaux
- Nombre binaires

0	$0 = 0 \times 2^0$
1	$1 = 1 \times 2^0$
2	$10 = 1 \times 2^1 + 0 \times 2^0$
3	$11 = 1 \times 2^1 + 1 \times 2^0$
4	$100 = 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$
5	$101 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$
…	…

0.6 L'octet

- Un **octet (byte)** et un paquet de 8 bits
  - Exemples: 01101010, 11100101, 01010001, …
- La mémoire d'un ordinateur se mesure par le nombre d'octets disponibles pour stocker de l'information codée en binaire
  - 1 Kilobyte (KB) = 2^{10} Bytes (= 1'024 Bytes)
  - 1 Megabyte (MB) = 2^{20} Bytes (= 1'048'576 Bytes)
  - 1 Gigabyte (GB) = 2^{30} Bytes (= 1'073'741'824 B)
  - 1 Terabyte (TB) = 2^{40} Bytes

0.7 Encodage du texte en binaire

- On utilise généralement **un octet** pour représenter chaque symbole.
- Un octet = 8 bits, on a donc $2^8 = 256$ octets différents.
- On a donc 256 symboles différents qui peuvent être encodés par un octet.

- Il existe une **table de correspondance** pour 127 valeurs: la table ASCII

(American Standard Code for Information Interchange).

36 $	37 %	38 _	39 '	40 (	41)	42 *	43 +	44 ,	45 -	46 .	47 /
48 0	49 1	50 2	51 3	52 4	53 5	54 6	55 7	56 8	57 9	58 :	59 ;
60 <	61 =	62 >	63 ?	64 @	65 A	66 B	67 C	68 D	69 E	70 F	71 G
72 H	73 I	74 J	75 K	76 L	77 M	78 N	79 O	80 P	81 Q	82 R	83 S
84 T	85 U	86 V	87 W	88 X	89 Y	90 Z	91 [	92 \	93]	94 ^	95 _
96 `	97 a	98 b	99 c	100 d	101 e	102 f	103 g	104 h	105 i	106 j	107 k

Chapitre 1: Programmation

1.1 Démarche générale

La programmation d'un ordinateur consiste à « expliquer » en détails à une machine ce qu'elle doit faire, en sachant d'emblée qu'elle ne peut pas véritablement « comprendre » un langage humain, mais seulement effectuer un traitement automatique sur des séquences de caractères.

Un programme n'est rien d'autre qu'une suite d'instructions encodées qui respectent de manière très stricte un ensemble de conventions fixées à l'avance que l'on appelle langage informatique.
L'ordinateur est pourvu d'un mécanisme qui décode ces instructions en associant à chaque « mot » du langage une action précise.

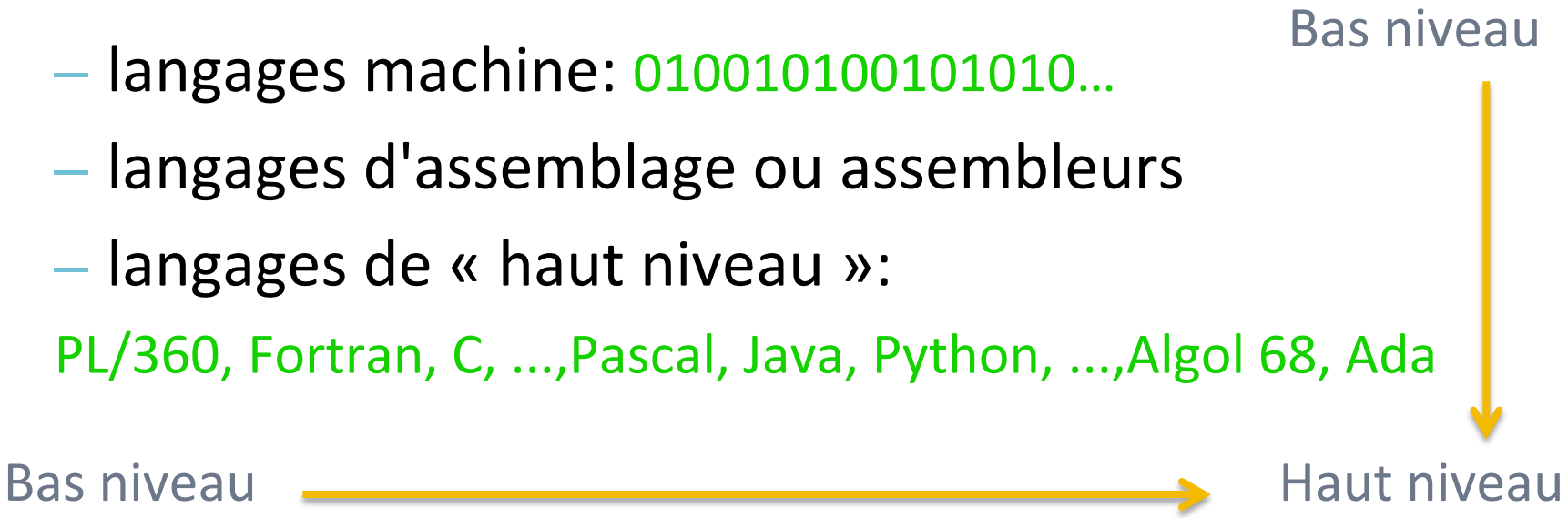
Un algorithme est une suite finie et non-ambiguë d'opérations ou d'instructions permettant de résoudre un problème donné. L'activité de programmation consiste à écrire des algorithmes pour résoudre des problèmes trop complexes pour être traités de tête. Certains problèmes (et même la plupart) sont impossibles à résoudre de manière algorithmique!

1.2 Langages de programmation

Un **langage formel** (ou artificiel vs naturel) est un langage qui au bout du compte est compris par un ordinateur.

- langages machine: 010010100101010…
- langages d'assemblage ou assembleurs
- langages de « haut niveau »:

PL/360, Fortran, C, ...,Pascal, Java, Python, ...,Algol 68, Ada



Langages fonctionnels vs langages impératif

- Programmation fonctionnelle:
  - Démarche: résoudre un problème, c'est calculer une fonction (en général, cette fonction fait appel à d'autres fonctions)
  - Lisp, Haskell, …
- Programmation impérative:
  - Démarche: on dit, étape par étape, ce qu'il faut faire => algorithme
  - C, C++, Java, Python, …

Exemple: Déterminer si un nombre est premier

- Démarche fonctionnelle:

```
divisible(n,d) =df ∃ q | n = q*d
premier(n) =df ¬ divisible(n,x) ∀ x ∈ [2..n-1]
```

- Démarche impérative:

```
premier := Vrai
x := 2
tant_que premier et x ≤ n-1 faire
      si n mod x = 0 alors premier := Faux
      sinon x := x + 1
```

Langages spécialisés vs. langages universels

- Langages spécialisés:
  - programmation logique: Prolog, …
  - base de données: SQL, ...
  - pages Web: Perl, PHP, HTML, ...
  - Etc.
- Langages universels:
  - C, C++, Java, Python, …

Langages orientés objets

- Programmation orientée objet: Il s'agit d'une *méthode de programmation* que nous étudierons dans le cadre de ce cours.
- Langages intégrant ce concept dès l'origine:
  - Smalltalk, Eiffel, Java, Python, ...
- Langages ayant évolués vers ce concept:
  - C ⇒ C++ ou Objective C
  - Pascal ⇒ Delphi
  - Basic ⇒ Visual Basic
  - etc.

1.3 Interprétation et compilation

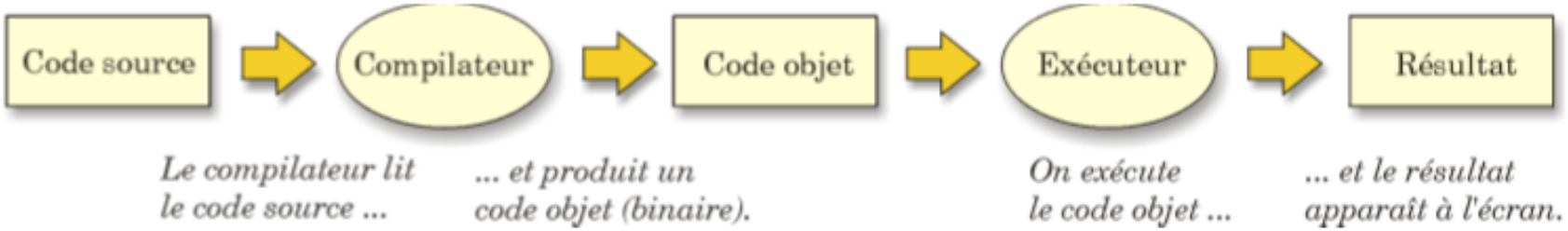
- Tout programme écrit dans un langage de programmation doit d'abord être **traduit** en code binaire afin de pouvoir être compris et exécuté par l'ordinateur.
- Il existe deux techniques principales pour effectuer cette traduction:
- l'**interprétation** et la **compilation**

- **L'intérprétation**:

  Un logiciel, appelé interpréteur, traduit au fur et à mesure chaque ligne du programme source en instructions binaires qui sont ensuite directement exécutées.

- **La compilation:**

  Un logociel, appelé compilateur, traduit la totalité du programme en une fois et produit un nouveau code appelé programme objet. Le programme objet peut ensuite être conservé et exécuté sans avoir besoin d'être retraduit à chaque fois.



1.4 Python

- Créé au début des années 90 par Guido Van Rossum;
- Distribué sous forme de logiciel libre;
- Disponible pour de nombreux systèmes d'exploitation, notamment Linux, Mac OS et Windows;
- Langage de très haut niveau, universel, impératif et orienté objet.