Définition : Informatique

- « Dans son acception courante, l'informatique désigne l'ensemble des sciences et techniques en rapport avec le traitement de l'information.
  Dans le parler populaire, l'informatique peut aussi désigner à tort ce qui se rapporte au matériel informatique (l'électronique), et la bureautique. » (source wikipédia).

- On parle aussi de
  - STIC : sciences et technologies de l'information et de la communication
  - En anglais : Informatics, **Computer science**, Computer engineering, Software engineering, Information Technology.

Données

- Les données sont des biens précieux

- Cycle de vie
  - Production → stockage → exploitation → archivage

- Place de l'informatique
  - D → traitement informatique → D' + valeur ajoutée

- Enjeux
  - Économiques et politiques
  - Ethiques

Représentation et stockage des données

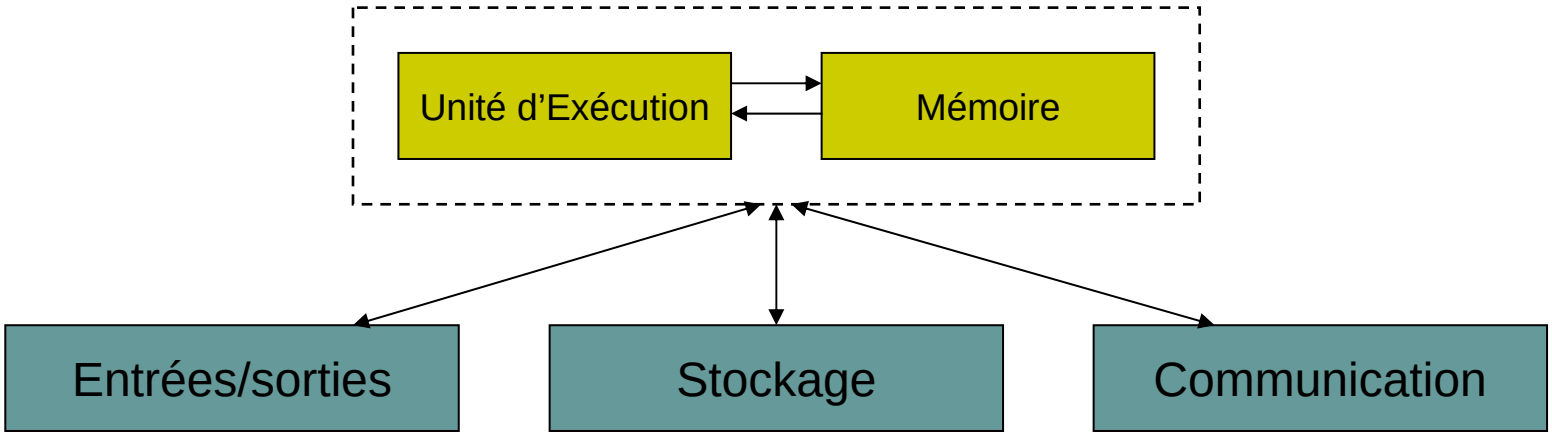
- Sont de toutes sortes, mais doivent être numérisées sous forme de 0 et de 1 (bit = Binary Digit)

- Unités
  - Octet : 8 bits = 256 (2^8) informations (ex : caractères)
  - Kilo Octet (Ko) : 2^{10} octets = 1024 octets
  - Mega Octet (Mo) : 2^{20} octets = 1024 Kilo octets
  - Giga Octet (Go) : 2^{30} octets = 1 073 741 824 octets = 1024 Mo

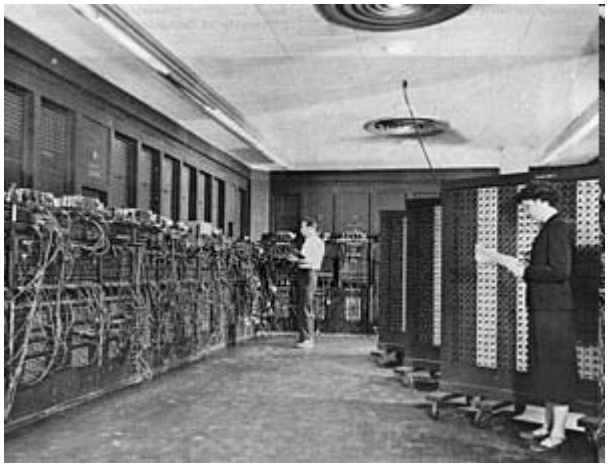
- Supports de stockage et débits réseaux
  - 2000 : Disquette (1,4 Mo), Disque dur (4 Go), Mémoire (128 Mo), modem 56 Kbits/s
  - 2008 : Clé USB (4 Go), Disque dur (320 Go), Mémoire (4 Go), ADSL 10 Mbits/s
  - IPhone : Mémoire flash (4 Go ou 8 Go), Mémoire (32 +16 Mo)

Définition : Ordinateur

- « Un ordinateur est une machine dotée d'une unité de traitement lui permettant d'exécuter des programmes enregistrés manipulant des données sous forme binaire. » (source wikipédia)
- Premiers « ordinateurs » :
  - Pascaline (1642), calculatrice mécanique, horlogerie
  - métiers à tisser Jacquard (1803),
  - Babbage (1835), calcul des tables marines UK
  - Turing (1943), Enigma, décodage msg secrets
  - Von Neumann (1945), ancêtre des machines actuelles programmable



Evolution des ordinateurs



1946 : ENIAC

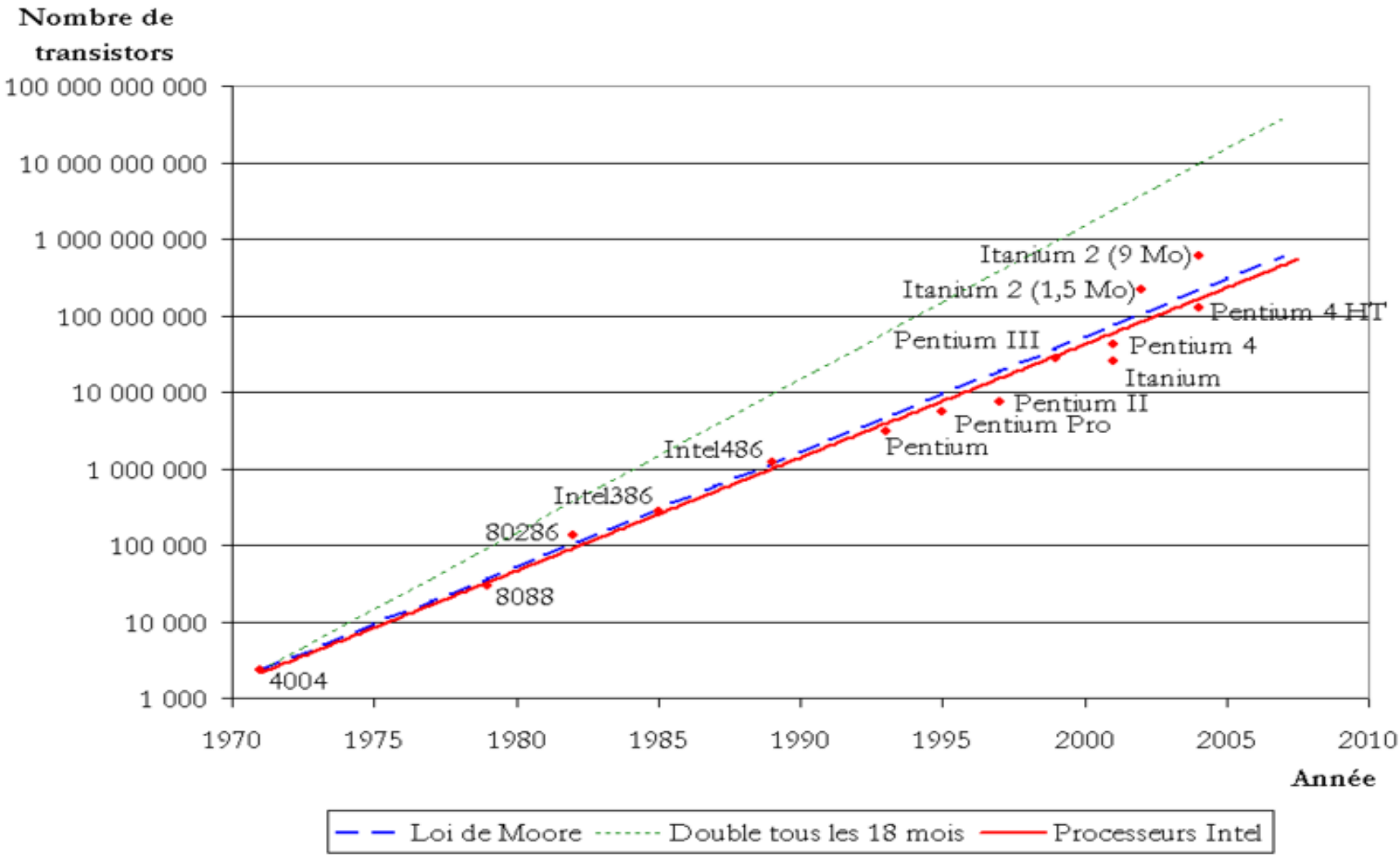
1980 – IBM PC

1982 – Commodore 64

1984 – Macintosh

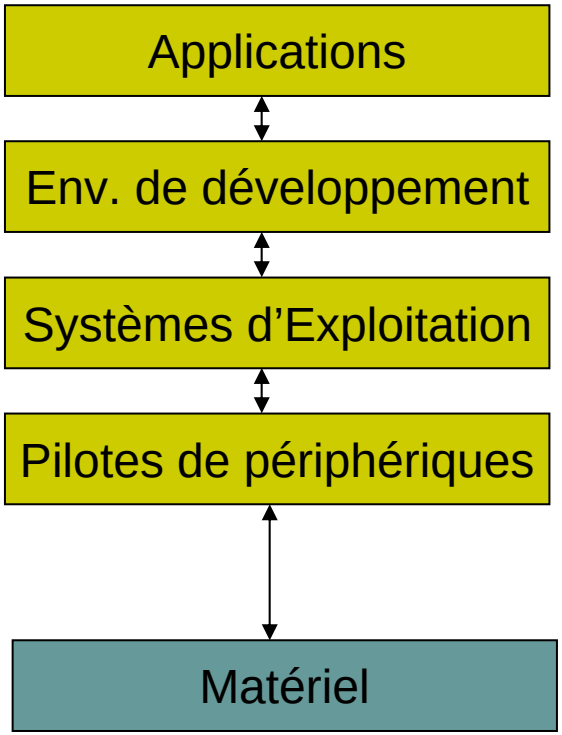
2009

Loi de Moore (doublement tous les 2 ans)



Définition : Logiciel

- Un **logiciel** est un ensemble de programmes qui permet à un ordinateur ou à un système informatique d'assurer une tâche ou une fonction en particulier. (source wikipédia)
- Plusieurs sortes de logiciels, structuration en couches



Logiciels


W

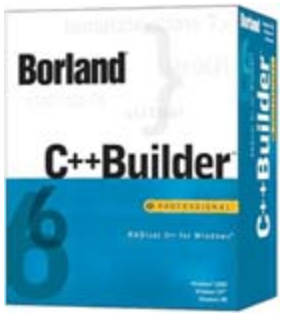

MySQL


MATLAB
7

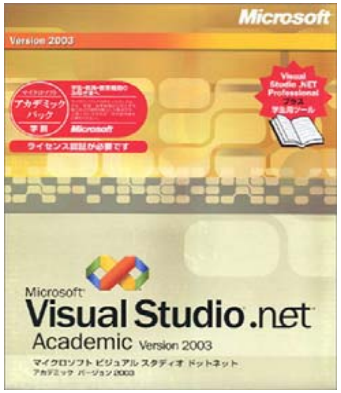

DS
CATIA


skype™


Google
Earth


Borland
C++Builder
6


K DEVELOPING ENVIRONMENT
#include <iostream>
#include <cmath>
class Add
{
private:
int x;
public:
int Addition (void);
};
int main()
{
KDevelop


Microsoft
Version 2003
Microsoft
Visual Studio .net
Academic Version 2003


eclipse
THE ECLIPSE PROJECT
(c) Copyright IBM Corp. and others. 2000, 2003. All rights reserved. Java and all Java-related trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S., other countries, or both.


Windows Vista™


Linux™


X
Mac OS X

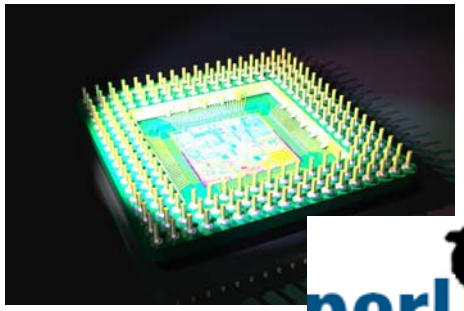
Définitions : Algorithme, Langage

- Un algorithme est un moyen pour un humain de présenter la résolution par calcul d'un problème à une autre personne physique.
  Un algorithme est un énoncé dans un langage défini d'une suite d'opérations permettant de résoudre par calcul un problème.

- Un langage de programmation permet de définir les ensembles d'instructions effectuées par un ordinateur lors de l'exécution d'un programme.
  Les langages de programmation permettent d'implémenter des algorithmes.
  Il existe des dizaines de langages informatiques adaptés, ou non, à des domaines spécifiques

- Langages machine, assembleur, impératifs, fonctionnels, objets...

Langages



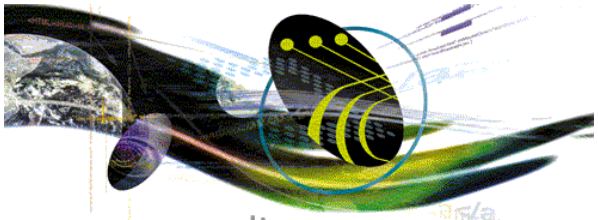
perl

JAVA™ Authorized Java Center

the elePHPant

PHP >> PHP >> php

Sun microsystems

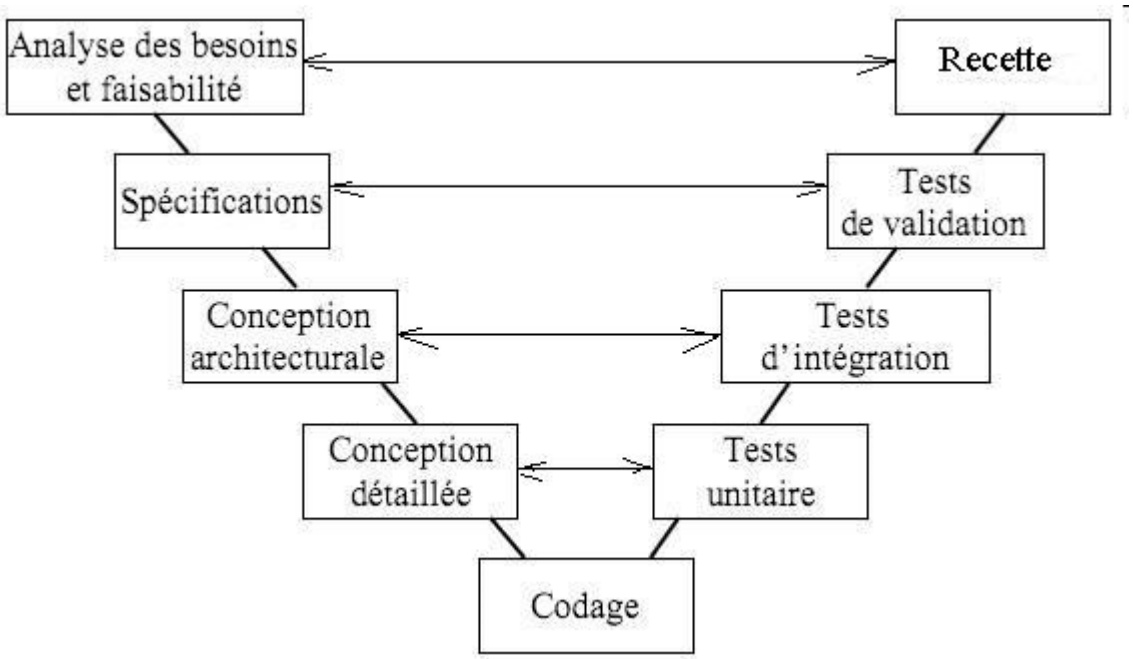


Dreamweaver 2.01

Microsoft® .net™

Génie Logiciel

- Le **génie logiciel** (en anglais : *software engineering*) désigne l'ensemble des méthodes, des techniques et outils concourant à la production d'un logiciel, au-delà de la seule activité de programmation. (source wikipédia)
- Cycle de développement, le plus courant est dit en V



Définition : réseau

- Un **réseau informatique** est un ensemble d'équipements reliés entre eux pour échanger des informations.
- Les protocoles de communication permettent de définir de façon standardisée la manière dont les informations sont échangées entre les équipements du réseau.
- Réseau = équipement + protocoles
- Réseau = débit (en bit/s), latence, gigue, taux de perte

- Quelques chiffres
  - Brest – Paris : aller-retour en 30 ms (13 relais, 500 km)
  - Brest – Sydney : aller-retour en 300 ms (23 relais, 17 000 km)
  - Vitesse du son : 300 m/s = 3 m/10 ms
  - Clignement de paupière : 60 ms
  - Chute d'un objet (h = 1m) : 400 ms

Définition : Informaticien

- « L'informaticien (-ne) exerce un métier de l'informatique.
  La variété des informaticiens reflète d'une part celle des techniques informatiques et d'autre part celle des modes d'organisation du travail informatique.
  Elle s'illustre dans le cadre de la recherche, de la conception de systèmes, de la production et de la gestion, de la maintenance.
  Ces activités peuvent concerner le domaine matériel et/ou le domaine logiciel.» (source wikipédia)

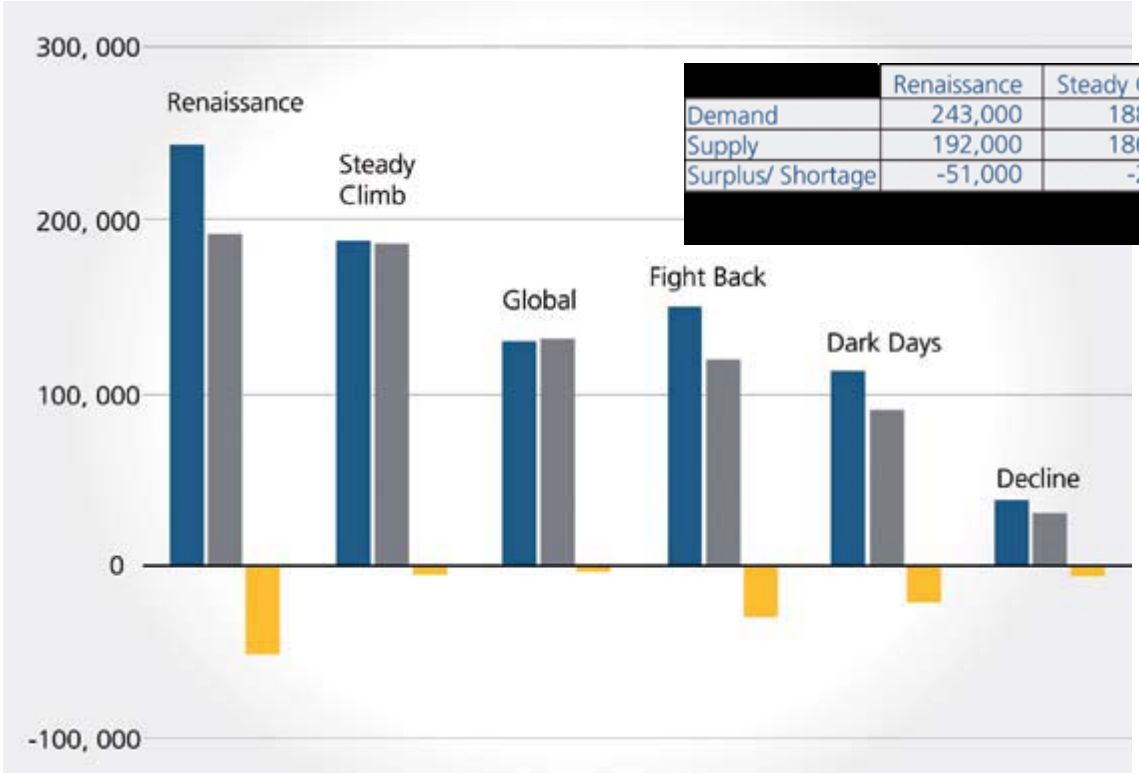
- Formation « standard »
  - Licence : programmation, base de données, réseaux et systèmes, matériels, bases théoriques ➔ vision globale
  - Master : conception de systèmes, spécialisation thématique.

Métier d'informaticien

- Dépend du secteur d'activité, de l'expérience, du niveau de formation, de l'intérêt
- Métiers type :
  - Informaticien « logiciel »
    - développeur puis chef de projet, architecte, resp. d'affaire, consultant...
  - Informaticien « système »
    - Resp. d'applications, admin. systèmes/réseaux puis resp. de sites, ingénieur sécurité…
  - Informaticien « scientifique »
- Type d 'entreprises
  - SSII : sociétés de services en Informatique
  - Services informatique de grande entreprise
  - Laboratoire de recherche

Perspectives d'emploi en 2015

- Etude CEPIS Council of European Professional Informatics Societies Novembre 2007



	Renaissance	Steady Climb	Global	Fight Back	Dark Days	Decline
Demand	243,000	188,500	129,800	150,000	111,000	38,000
Supply	192,000	186,000	131,300	120,000	90,000	30,000
Surplus/ Shortage	-51,000	-2,500	-1,500	-30,000	-21,000	-8,000

3 raisons expliquent ces besoins : croissance du marché, premiers départs en retraite, baisse de l'intérêt des études scientifiques.

Objectifs

- Les ordinateurs permettent de traiter rapidement de nombreuses données dans de nombreux domaines comme
  - Calcul scientifique
  - Ingénierie
  - Bases de données
  - Finance
  - Etc.

- Quelque soit votre futur métier, il y a de grande chances que vous soyez amenés à utiliser des outils informatiques pour faire des calculs (calculatrice, tableur, etc)

- Vous devez donc **connaitre les limites**, **les biais** et **les erreurs** du calcul par ordinateur.

Objectifs

- Plus spécifiquement, une connaissance même succincte du codage permet de créer des programmes plus sûrs :
  - Ariane 501, le 28/02/1996 a explosée car l'accélération horizontale a **dépassé la plage des valeurs autorisées** par le logiciel de vol
    => vidéo
- Que s'est-il passé pour que cette explosion survienne ?
  - Dans un ordinateur, les capacités de stockage sont limitées
  - De manière simplifiée, chaque nombre stocké ne peut pas dépasser certaines valeurs (soit trop grandes soit trop petites).
  - Imaginez un compteur kilométrique avec 3 chiffres : la plus petite valeur est **000** et la plus grande est **999**. Si on **ajoute 1** km **à 999** km on retombe sur **000** ! => C'est le même principe dans un ordinateur.

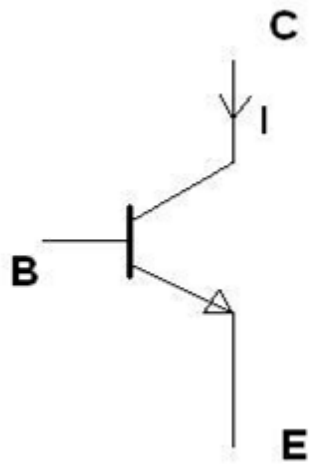
Objectifs

- Ces problèmes sont cruciaux dans presque tous les programmes informatique, que l'on soit programmeur ou simple utilisateur !
  - Imaginez votre compte en banque passer de 99 999 € à 0 € lorsque vous lui ajoutez 1 € !
  - Voir aussi le problème du bug de l'an 2000 (passer de l'an 99 à l'année 00 !).
  - Il peut également s'agir d'un simple problème d'arrondi : 0,8 devient 0,799.

- Dans ce cours nous allons donc étudier la manière dont les **nombres** sont **stockés** dans l'ordinateur afin de **correctement** :
  - **utiliser** les logiciels existant
  - **programmer** nos propres logiciels

Qu'est-ce que le codage ?

- Les ordinateurs ne savent traiter que des 0 et des 1 car ils sont basés sur les transistors :



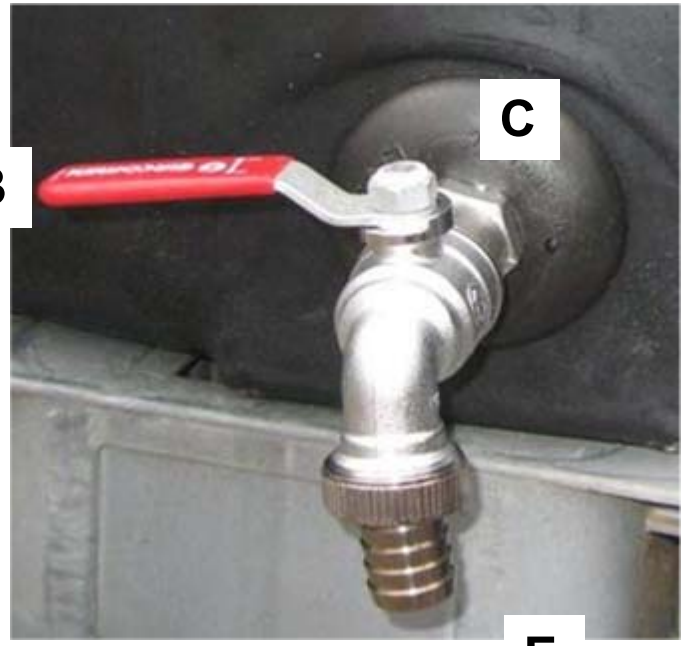
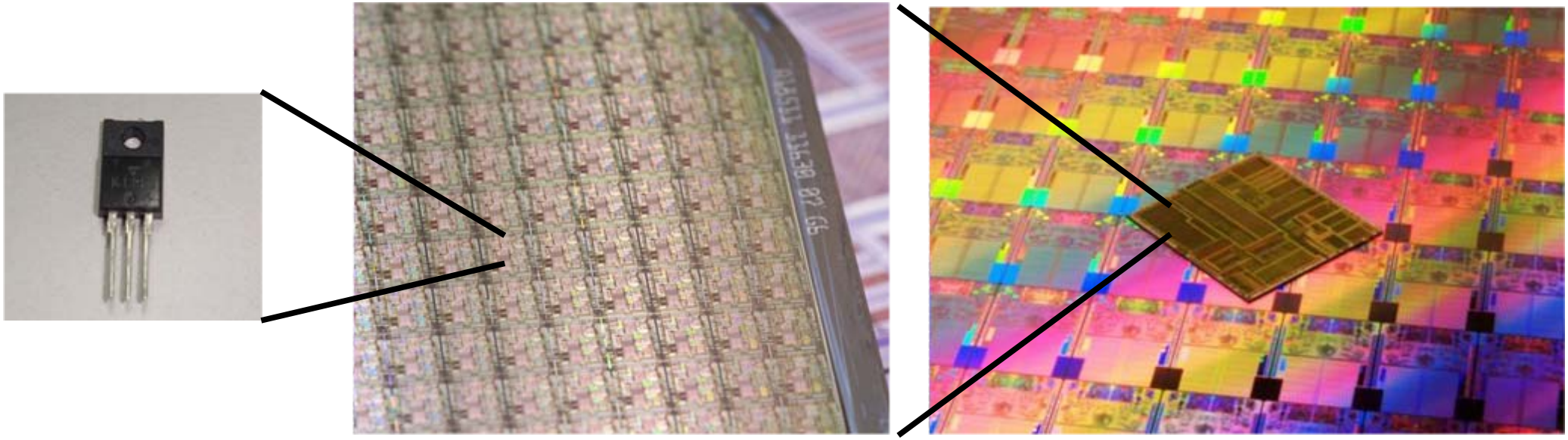


Schéma d'un transistor... dont le fonctionnement est similaire à celui d'un robinet !
Le courant I passe (1) ou ne passe pas (0)... L'eau coule ou ne coule pas.
La tension en B permet de faire passer ou non le courant I... Le levier B permet de faire couler ou non l'eau

Qu'est-ce que le codage ?

- Les transistors sont la base des microprocesseurs



Exemples de microprocesseurs (année – nom – taille gravure – nb transistors – MIPS – Freq):

1974 – Intel 8080	– 6µm	– 6 000	– 0,64	– 2 MHz
2000 – Intel P4	– 0,1 µm	– 42 000 000	– 1 700	– 3,8 GHz
2008 – Intel Core 2 Quad	– 0,045 µm	– 820 000 000	– 48 400	– 3,2 GHz

Qu'est-ce que le codage ?

- Si l'on souhaite mémoriser un nombre tel que 2 ou 3 ou plus, il faut trouver un moyen pour le représenter uniquement avec des 0 et des 1.

- Le codage est donc le moyen qui permet de mémoriser dans l'ordinateur toute sorte de nombre (entiers, décimaux, positifs, négatifs...).

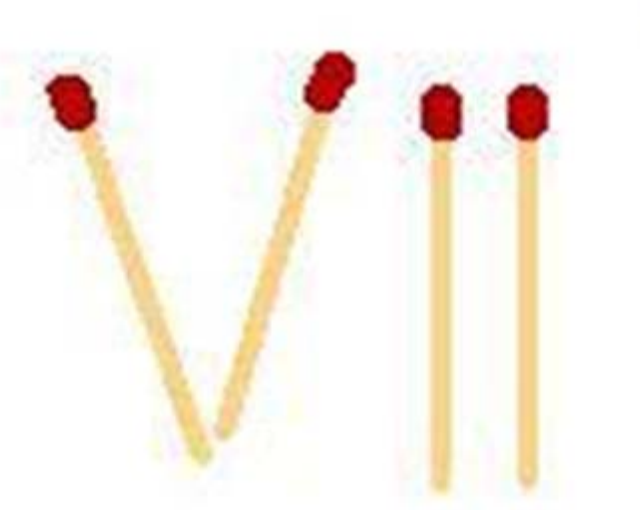
Comment coder ?

- Comment mémoriser toutes sortes de nombres avec uniquement des 0 et des 1 (base 2) ?
  - Une solution simple est la suivante :
    - 0 se code 0
    - 1 se code 1
    - 2 se code 11
    - 3 se code 111
    - 4 se code 1111
    - Etc

  - Cette solution est gourmande en mémoire. Imaginer mémoriser le nombre 300 000 000...

Comment coder ?

- Les chiffres romains...



Oui... mais comment coder l'inclinaison :
0 = pas d'allumette
1 = une alumette droite
2 = une allumette inclinée à droite (oups on dépasse 1)
Pour les calculs (addition...) cela ne va pas être pratique

- Donc on oublie (très vite) cette ”solution” !

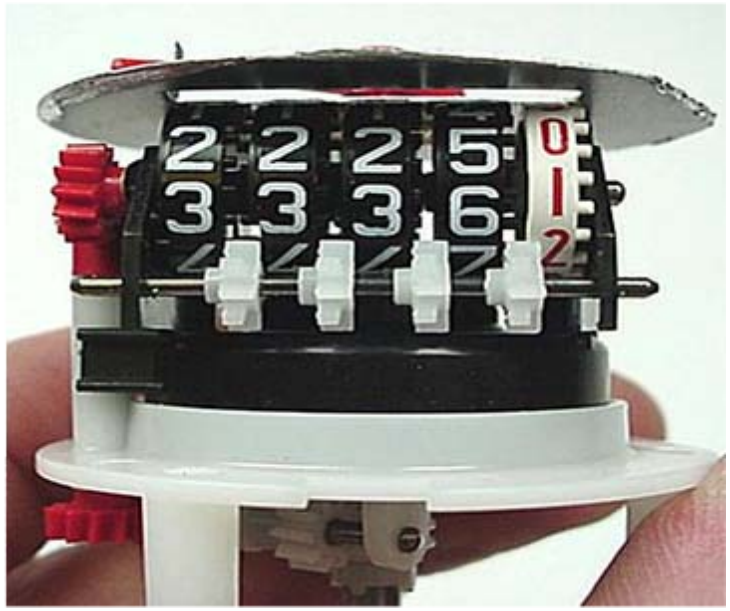
Comment coder ?

- Etudions notre habituelle base à 10 chiffres
  - 0, 1, 2, 3, 4, 5, 6 ,7 ,8 ,9
  - Pour aller au delà de 9, nous avons le même problème qu'en base 2 pour aller au delà de 1 !
  - Pour passer de neuf à ***dix*** dans notre base habituelle, nous ré-utilisons *les mêmes chiffres* en ajoutant un chiffre dit des ”dizaines” **1** et en faisant ”tomber” le 9 à 0

09+1 = **1 0**

Comment coder ?

- On peut facilement concevoir le codage suivant :
  - 0 se code 0000
  - 1 se code 0001
  - 2 se code 0010
  - 3 se code 0011
  - 4 se code 0100
  - 5 se code 0101
  - 6 se code 0110
  - 7 se code 0111
  - 8 se code 1000
  - 9 se code 1001
  - etc



Comment coder ?

- **Exercice** :
  - 10 se code ... **1010**
  - 11 se code ... **1011**
  - 12 se code ... **1100**
  - 15 se code ... **1111**
  - 17 se code ... **10001**
- Exercice : Quelle heure est-il ?

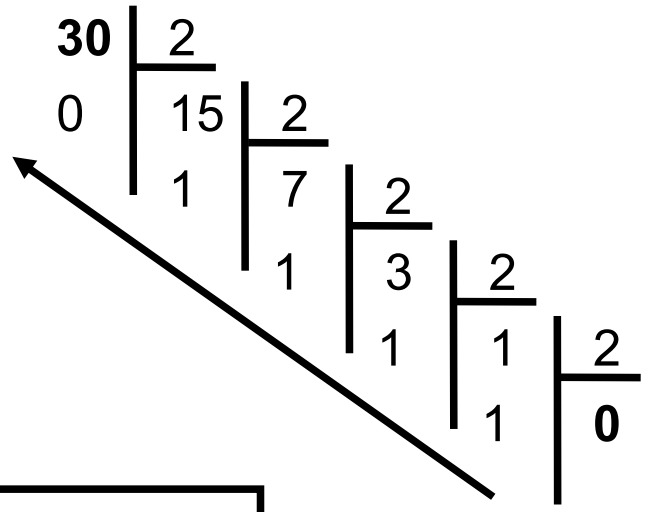
Il est :

9 H
10 m
23 s

- Exercice : coder 300 000 000 **10001111000011010001100000000**

Codage des entiers positifs 10→2

- Divisions par 2 successives
  - Soit le nombre 30 en base **10**. Quelle est son écriture en base **2** ?



Nombre	Diviseur	Résultat	Reste
30	2	15	0
15	2	7	1
7	2	3	1
3	2	1	1
1	2	**0**	1

Réponse : 11110_2

Codage des entiers positifs : 2 →10

- En base 10 le nombre 314 peut s'écrire
  - $314_{10} = 300 + 10 + 4 = 3*10^2 + 1*10^1 + 4*10^0$

- La méthode pour passer d'un nombre binaire (base 2) à un nombre en base 10 est similaire :
  - $10110_2 = 1*2^4 + 0*2^3 + 1*2^2 + 1*2^1 + 0*2^0 = 22_{10}$

Codage de la base 2 à 10

- **Exercice** :
  - Soit les nombres suivant en base 2. Donner leurs valeurs en base 10
    - 0_2 **= 0**
    - 1_2 **= $1.2^0 = 1$**
    - 10_2 **= $1.2^1 = 2$**
    - 111_2 **= $1.2^2 + 1.2^1 + 1.2^0 = 4+2+1 = 7$**
    - 10101_2 **= $1.2^4 + 1.2^2 + 1.2^0 = 16+4+1 = 21$**
    - 101010_2 **= $1.2^5 + 1.2^3 + 1.2^1 = 32+8+2 = 42$**

Généralisation à une base b → 10

- Passage d'une base **b** à la base **10**
  - Soit le nombre **N** en base **b** composé de **n** chiffres :

$$N = (c_n c_{n-1} \ldots c_0)_b$$

  - Sa valeur en base 10 vaut :

$$N = c_n * b^n + c_{n-1} * b^{n-1} + \ldots + c_1 * b^1 + c_0 * b^0$$

Implémentation

- La mémorisation des nombres binaires dans un ordinateur se fait selon des formats bien précis :
  - Un **octet** (**Byte**) correspond à 8 chiffres binaires
    - 00000000 mémorise 0 (le minimum)
    - 11111111 mémorise 255 (le maximum)
    - 00000011 mémorise 3
  - Un **mot** correspond à 2 octets
  - Un **mot long** correspond à 2 mots (4 octets)

- Ne pas oublier de placer les 0 devant le nombre si besoin : le nombre **1010_2** s'implémente **00001010** dans un octet.

Implémentation

- **Exercice** :
  - Implémenter 11001 dans un octet

    00011001
  - Implémenter 11001 dans un mot

    00000000 00011001
  - Implémenter 1011110101111101 dans un mot long

    00000000 00000000 01011110 10111101
  - Implémenter 101010100 dans un octet

    1 **01010100**

    ↑
- Noter le problème du **dépassement** de **capacité**.

Limites de l'implémentation

- Avec **1 bit** il est possible de coder deux nombres: **0 et 1**.
- **2 bits** permettent de coder quatre nombres différents (2*2): **00, 01, 10 et 11**.
- **3 bits** codent 8 nombres (2*2*2) : **000, 001, 010, 011, 100, 101, 110, 111**.
- Pour un groupe de **n bits**, il est possible de représenter **2^n nombres**.

Unités

- les unités habituelles sont les suivantes :
  - Un kilooctet (ko ou kB) = 1024 octets
  - Un Mégaoctet (Mo ou MB) = 1024 Ko = 1 048 576 octets
  - Un Gigaoctet (Go ou GB) = 1024 Mo = 1 073 741 824 octets
  - Un Téraoctet (To) = 1024 Go
  - Un Pétaoctet (Po) = 1024 To

- Attention : kB signifie kiloByte, 1 Byte = 1 octet

NB : depuis 1998, la norme SI stipule qu'1 ko = 1000 octets, 1 Mo = 1000 ko, etc.

Addition des binaires

- Soit deux nombres binaires A et B. L'addition A+B en binaire se fait de manière simulaire que celle en base 10 :

```
    1 1
A =  10011001
B =  01001010
     --------
S =  01100011
```

```
  1   1 1
A =  10011001
B =   11001010
     ---------
S = 101100011
```

Codage des décimaux : 10 → 2

- Codage de la base **10** à la base **2**
  - La partie entière se code comme précédemment
  - La partie décimale se code comme suit :
    - Soit $0{,}125_{10}$ à coder en base **2** :

Nombre	Multiplicateur	Résultat	Partie entière
0,125	2	0,250	0
0,250	2	0,5	0
0,5	2	1,**0**	1

Réponse : $\mathbf{0{,}125_{10}} = \mathbf{0{,}001_{2}}$

Codage des décimaux : 10 → 2

- **Exercice** : coder les nombre décimaux suivants de la base 10 vers la base 2
  - 10,5 = ... **$1010,1_2$**
  - 8,1875 = ... **$1000,0011_2$**
  - 0,8 = ... **$0,1100110011001100...$** **$= 0,\overline{1100}_2$**
  - 0,3 = ... **$0,01001_2$**
- Noter l'impossibilité d'écrire certains décimaux avec un nombre fini de chiffres en base 2.

Généralisation : 10 → b

- Le passage de la base **10** à une base **b** se fait comme pour le passage de la base **10** à la base **2** ***mais*** il faut
  - Pour la **partie entière**
    - remplacer les divisions par 2 par **des divisions par b**
  - Pour la **partie décimale**
    - Remplacer les multiplications par 2 par **des mult. par b**

Codage des décimaux : base 2 → 10

- La méthode pour passer d'un nombre décimal en base 2 à un nombre décimal en base 10 est la suivante :
  - $10110{,}011_2 = 1*2^4 + 0*2^3 + 1*2^2 + 1*2^1 + 0*2^0 + 0*2^{-1} + 1*2^{-2} + 1*2^{-3}$
    $= 22{,}375_{10}$

- Généralisation : passage d'une base **b** à la base **10**
  - Soit le nombre **N** en base **b** composé de **n** chiffres pour sa partie entière et de **m** chiffres pour sa partie décimale :

    $$N = (c_n c_{n-1} \ldots c_0 , c_{-1} c_{-2} \ldots c_{-m})_b$$

  - Sa valeur en base 10 vaut :

    $$N = c_n * b^n + \ldots + c_1 * b^1 + c_0 * b^0 + c_{-1} * b^{-1} + + c_{-2} * b^{-2} + \ldots + + c_{-m} * b^{-m}$$

Implémentation des décimaux – sol. 1

- Implémentation partie entière + partie décimale
  - On peut découper un mot avec 1 octet pour la partie entière et 1 octet pour la partie décimale.
    - Le nombre $10{,}125_{10}$ s'écrit $1010{,}001_2$ et s'implémente :

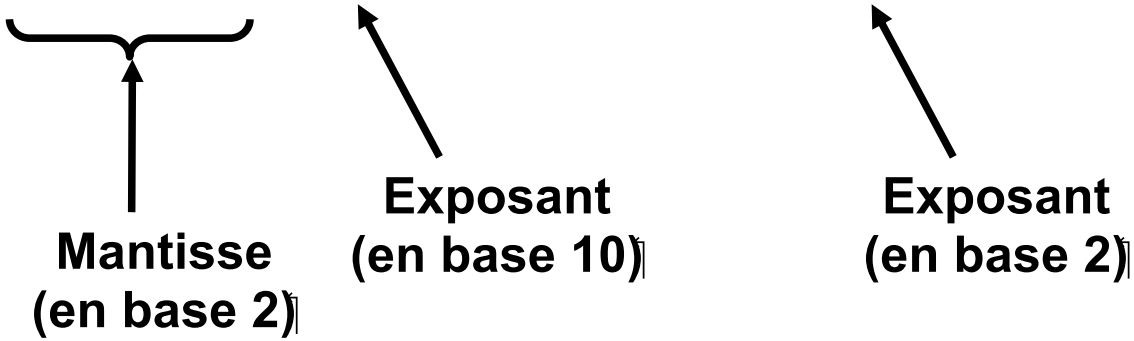
      | 00001010 | 00100000 |

      Noter l'ajout des 0 en **tête** de la partie entière et à la **fin** de la partie décimale.

Implémentation des décimaux – sol. 2

- Implémentation mantisse + exposant
  - Le nombre $10{,}125_{10}$ s'écrit **$1010{,}001_2$** et peut aussi s'écrire sous la forme scientifique (exponentielle normalisée) :

$1{,}010001_2.2^3$ = $1{,}010001_2.2^{11}$



Implémentation des décimaux - sol. 2

- Si l'on prend une mantisse de 12 bits et un exposant de 4 bits (le tout fait 1 mot), le nombre **$1,010001_2.2^{11}$ s'implémente :**

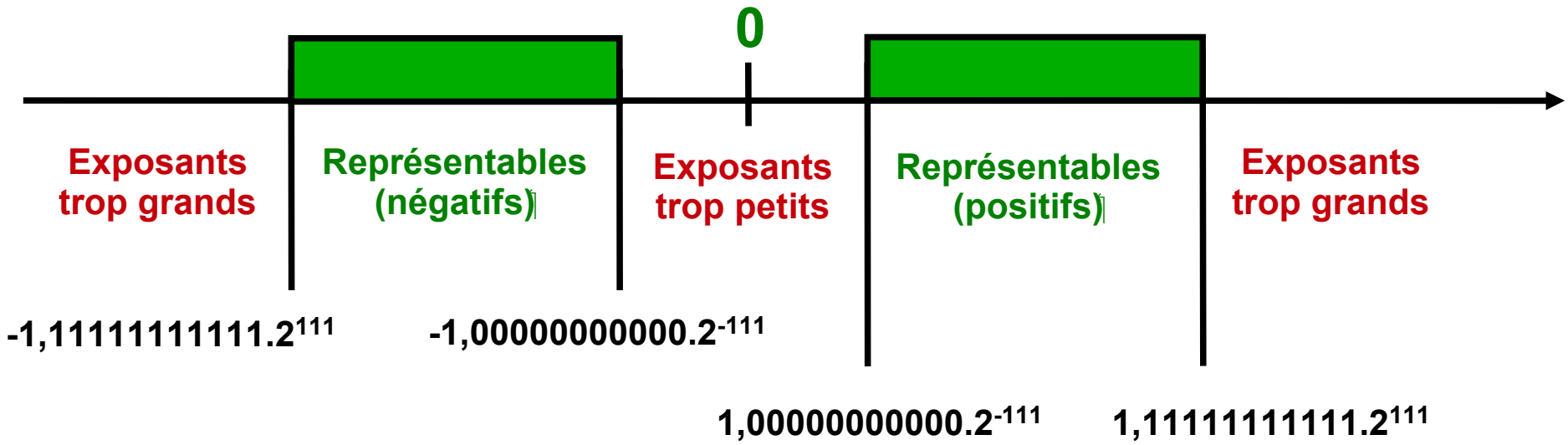
Signe (1bit)	Mantisse (11 bits)	Signe (1bit)	Exposant (3 bits)

|0|01000100000 |0|011 |

- Noter les 2 bits de signe : 0 = positif, 1 = négatif.

Implémentation des décimaux – sol. 2

- Mantisse + exposant, borne sup et borne inf :



Conclusion

- Pourquoi apprendre le codage ?
  - Éviter les pièges du calcul sur ordinateur (que l'on soit utilisateur ou programmeur)
- Qu'est ce que le codage ?
  - On code car les ordinateurs ne traitent que le binaire
- Comment coder ?
  - Passage d'une base dans une autre
  - Implémentation dans un format spécifique

Portes logiques

- Objectifs du cours
- Algèbre de Boole
- Circuits logiques
- Passage d'une formule logique à un circuit
- Réduction d'un circuit
- Conclusion & perspectives

Objectifs du cours

- En plus de stocker des nombres, les ordinateurs **effectuent des opérations** sur ces nombres :
  - Il peut s'agir d'opérations communes (addition, multiplication...)
  - Il peut s'agir également d'opérations logiques (A et B, A et B ou C, non A, si A est vrai alors D est vrai...)
- Le cours développe comment ces opérations sont réalisables dans un ordinateur qui ne sait traiter que des 0 et des 1.

Algèbre de Boole

- Grâce à l'algèbre de Boole (mathématicien britannique du XIXème) il est possible d'utiliser uniquement
  - 2 valeurs (**vrai** et **faux**)
  - 3 opérateurs
    - **ET** (noté au choix : ∧, .)
    - **OU** (noté au choix : v, +)
    - **NON** (noté au choix: ~, ¬)

    pour réaliser des opérations logiques.

Algèbre de Boole

- Propriétés de base de l'algèbre de Boole :
  - Associativité $a \vee (b \vee c) = (a \vee b) \vee c$
    $a \wedge (b \wedge c) = (a \wedge b) \wedge c$
  - Commutativité $a \vee b = b \vee a$
    $b \wedge a = a \wedge b$
  - Distributivité $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$
    $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$
  - Absorbtion : $a \vee(a \wedge b) = a$ et $a \wedge (a \vee b) = a$
  - Complémentarité $a \wedge \sim a = 0$ et $a \vee \sim a = 1$

Algèbre de Boole

- Autres propriétés :
  - Idempotence : $a \vee a = a$ et $a \wedge a = a$
  - Eléments neutres : $a \wedge 1 = a$ et $a \vee 0 = a$
  - Eléments absorbants : $a \wedge 0 = 0$ et $a \vee 1 = 1$
  - Complémentarité : $\sim 0 = 1$ et $\sim 1 = 0$
  - Involution : $\sim\sim a = a$
  - Lois de Morgan : $\sim(a \vee b) = \sim a \wedge \sim b$ et $\sim(a \wedge b) = \sim a \vee \sim b$
  - Redondance : $(a \wedge b) \vee (\sim a \wedge c) = (a \wedge b) \vee (\sim a \wedge c) \vee (b \wedge c)$

- Par convention,
  - NON est prioritaire sur ET
  - ET est prioritaire sur OU

Algèbre de Boole

- Tables de vérité

A	B	A.B
0	0	**0**
0	1	**0**
1	0	**0**

Table de vérité de l'opérateur ET.
Se lit comme suit :
0.0=0
0.1=0
1.0=0
1.1=1

A	B	A+B
0	0	**0**
0	1	**1**
1	0	**1**

Table de vérité de l'opérateur OU.
Se lit comme suit :
0+0=0
0+1=1
1+0=1
1+1=1

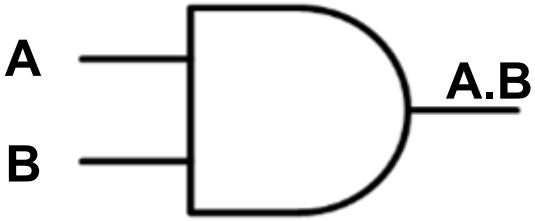
A	~A
0	**1**
1	**0**

Table de vérité de l'opérateur NON.
Se lit comme suit :
~0=1
~1=0

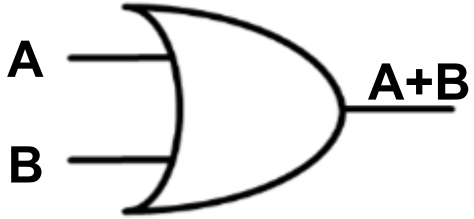
Circuits logiques

- En 1938, Claude Shannon utilisa l'algèbre de Boole pour faire les **circuits** de commutation téléphonique.
- Actuellement les portes logiques sont au coeur de tout microprocesseur.

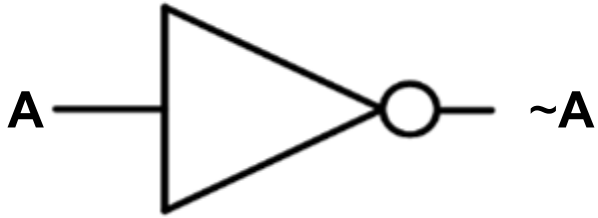
- Les portes courantes sont :



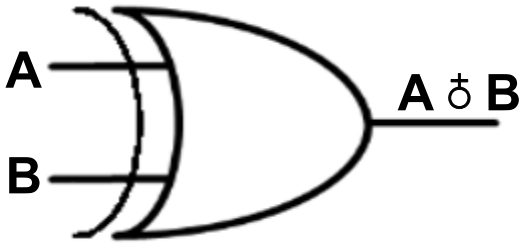
Porte ET



Porte OU



Porte NON

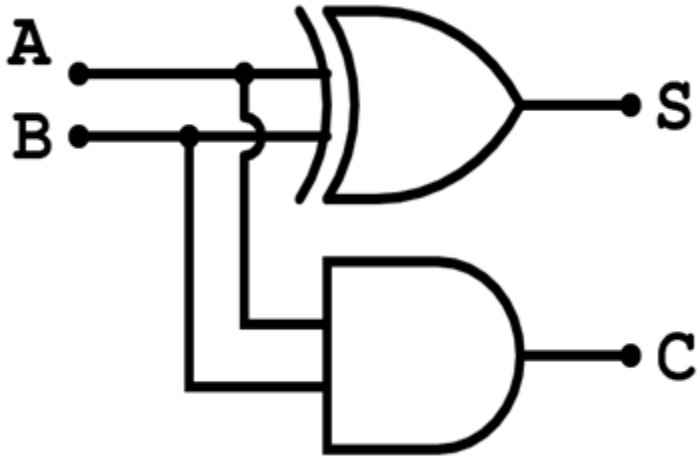


Porte OU Exclusif

Circuits logiques

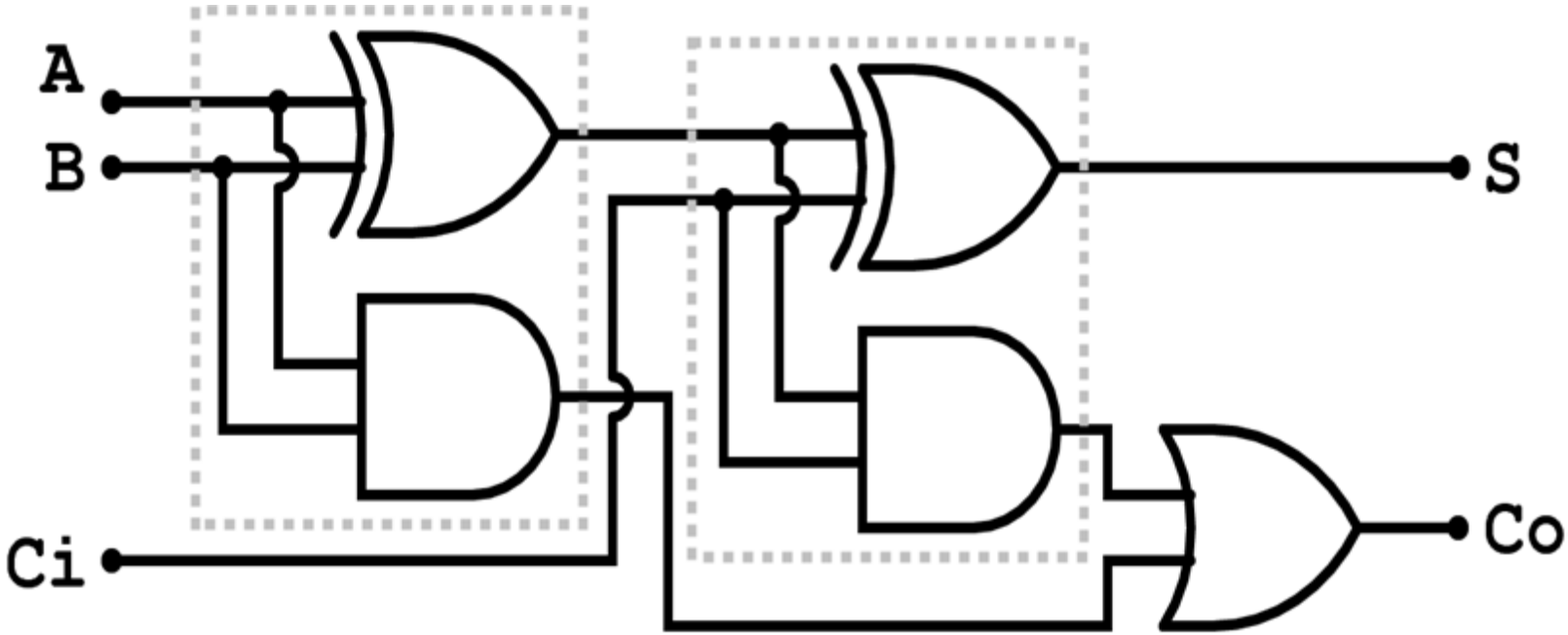
- Exemple : demi-additionneur
  Soit A et B deux nombres binaires à 1 seul bit
  Soit S la somme de A+B et C la retenue

A	B	S=A+B	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



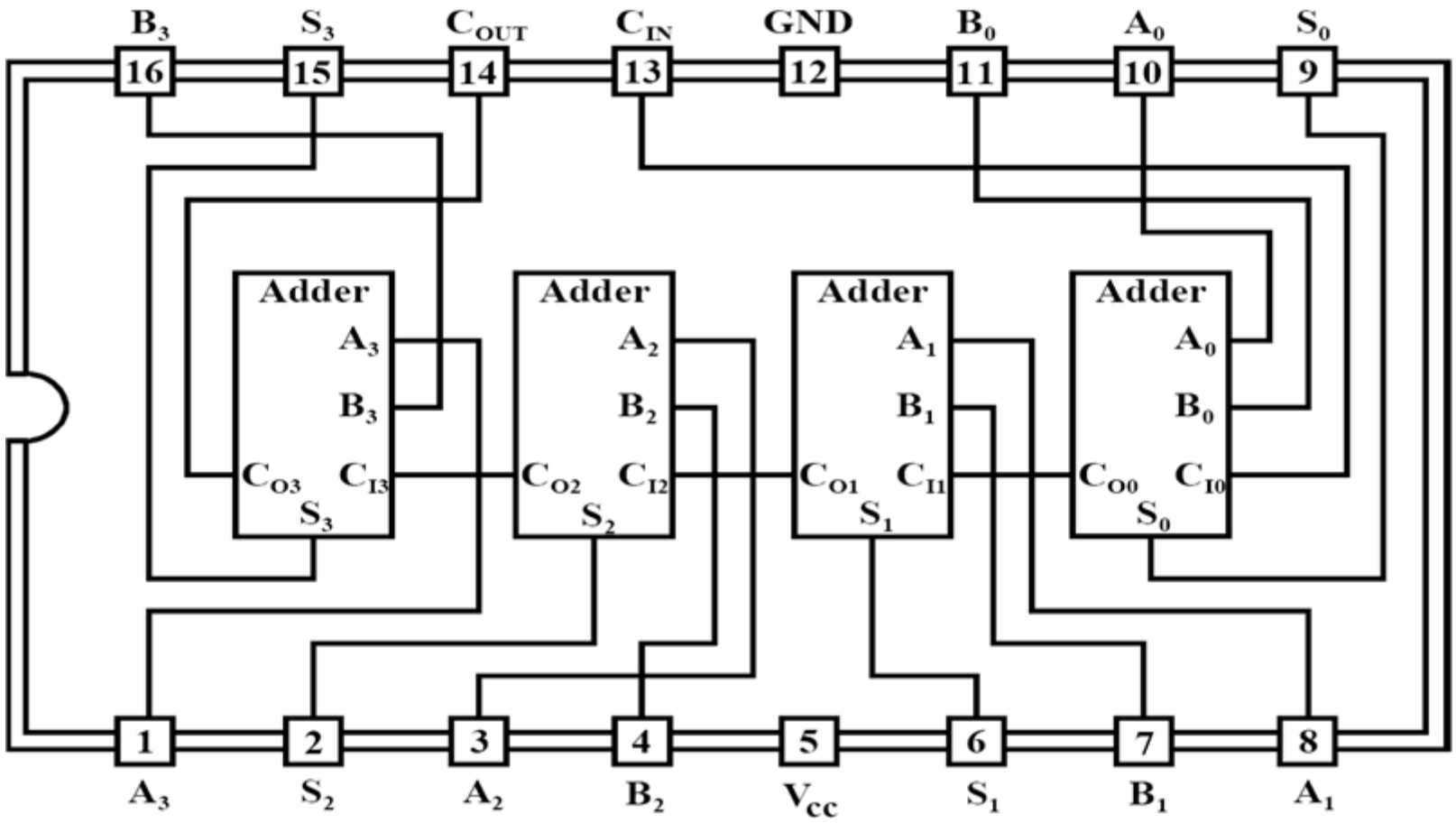
Circuits logiques

- Additionneur complet : $(A+B)+C_i = S+C_0$
  - Deux demi-additionneurs et une porte OU



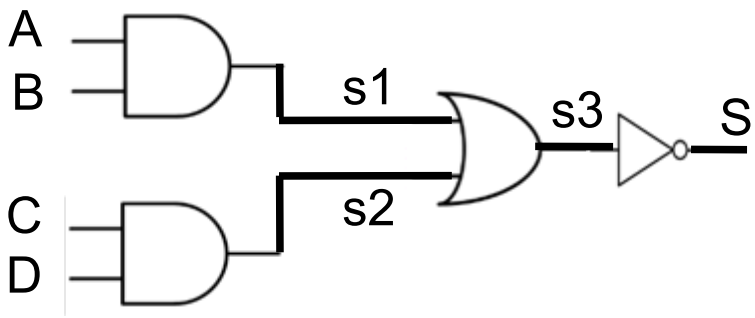
Circuits logiques

- Additionneur 4 bits



Circuits logiques

- Autre exemple



s1=A.B s2=C.D s3=s1+s2 S=~s3

D'où S = ~(A.B + C.D)=(~A+~B).(~C+~D)

De manière systématique, la table de vérité donne (avant simplification) : **S =**
~A.~B.~C.~D + ~A.~B.~C.D + ~A.~B.C.~D +
~A.B.~C.~D + ~A.B.~C.D + ~A.B.C.D +
A.~B.~C.~D + A.~B.~C.D + A.~B.C.~D

A	B	C	D	s1	s2	s3	S
0	0	0	0	0	0	0	**1**
0	0	0	1	0	0	0	**1**
0	0	1	0	0	0	0	**1**
0	0	1	1	0	**1**	**1**	0
0	1	0	0	0	0	0	**1**
0	1	0	1	0	0	0	**1**
0	1	1	0	0	0	0	**1**
0	1	1	1	0	**1**	**1**	0
1	0	0	0	0	0	0	**1**
1	0	0	1	0	0	0	**1**
1	0	1	0	0	0	0	**1**
1	0	1	1	0	**1**	**1**	0
1	1	0	0	**1**	0	**1**	0
1	1	0	1	**1**	0	**1**	0
1	1	1	0	**1**	0	**1**	0
1	1	1	1	**1**	**1**	**1**	0

Circuits logiques

- Simplification de l'expression de S =
  - ~A.~B.~C.~D + ~A.~B.~C.D + ~A.~B.C.~D +
    ~A.B.~C.~D + ~A.B.~C.D + ~A.B.C.~D +
    A.~B.~C.~D + A.~B.~C.D + A.~B.C.~D
  - ~A.~B.~C.(~D+D) + A.~B~.C.(~D+D) + ~A.B.~C.(~D+D) +
    ~A.B.C.~D + ~A.~B.C.~D + A.~B.C.~D
  - ~A.~B.~C + A.~B.~C + ~A.B.~C + ~A.B.C.~D + ~A.~B.C.~D + A.~B.C.~D
  - ~C(~A.~B + A.~B + ~A.B) + ~D.(~A.B.C + ~A.~B.C + A.~B.C)
  - ~C(~A.(~B + B) + A.~B) + ~D.C(~A.~B + ~A.B + A.~B)
  - ~C(~A + A.~B) + ~D.C(~A + A.~B)
  - (~A + A.~ B)(~C + ~D.C)
  - (~A + ~B)(~C + ~D)
  - Il existe des techniques pour simplifier : tableaux de Carnot

Conclusion

- Pourquoi les circuits logiques ?
  - Effectuer des opérations logiques
- Comment ?
  - Algèbre de Boole et les portes logiques
- Exemples de circuits
  - Tables de vérité et leurs simplifications

Plan

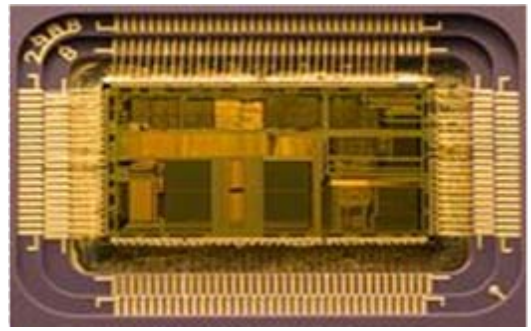
- Qu'est-ce qu'un micro-processeur (µ-processeur) ?
- Contenu d'un µ-processeur
- Programmation d'un µ-processeur
- Exemples de programme assembleur
- Conclusion générale

Qu'est-ce qu'un µ-processeur

- C'est un processeur dont les composants ont été miniaturisés pour tenir sur un seul circuit intégré.

- Un processeur est une Unité Centrale de Traitement
  - qui interprète les instructions et
  - traite les données d'un programme

- On retrouve
  - des données codées (nombres, caractères) et
  - des portes logiques



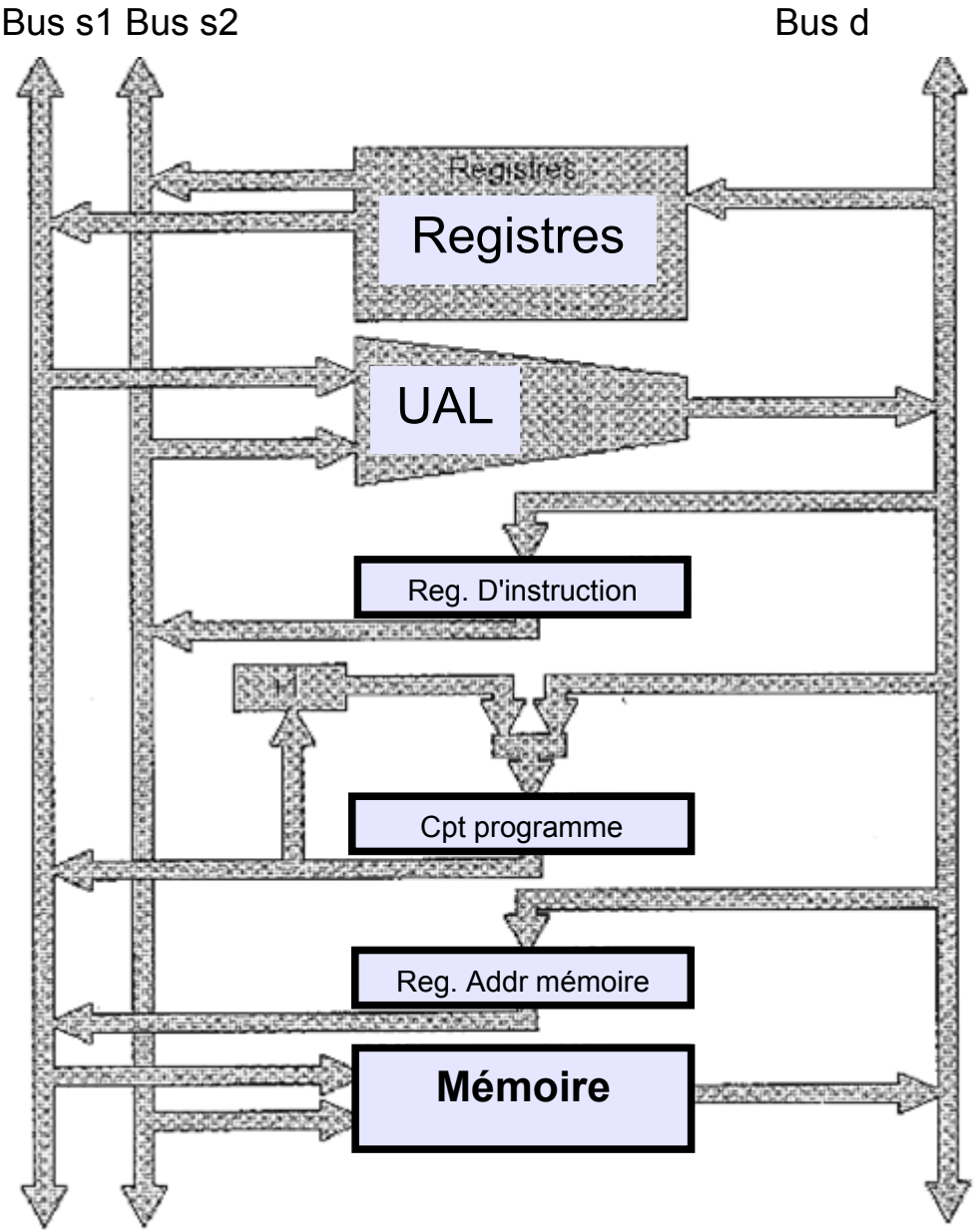
Intel 486-DX2 à 66MHz

Contenu d'un µ-processeur

- Une UAL (Unité Arithmétique et Logique)
- Des registres
  - Instruction (contient l'instruction en cours de traitement)
  - Compteur de programme (adresse de l'instruction en cours)
  - Accumulateurs (stockage de données traitées)
  - Adresse (prochaine adresse contenant l'information à lire par l'UAL)
- Une horloge (synchronise les actions)
- L'unité d'Entrées / Sorties
  - Notamment des bus permettant l'accès à la mémoire vive

Contenu d'un µ-processeur

- Exemple un processeur RISC
  (Reduced Instruction Set Computer
  processeur à jeu d'instructions réduit)



Programmation d'un µ-Processeur

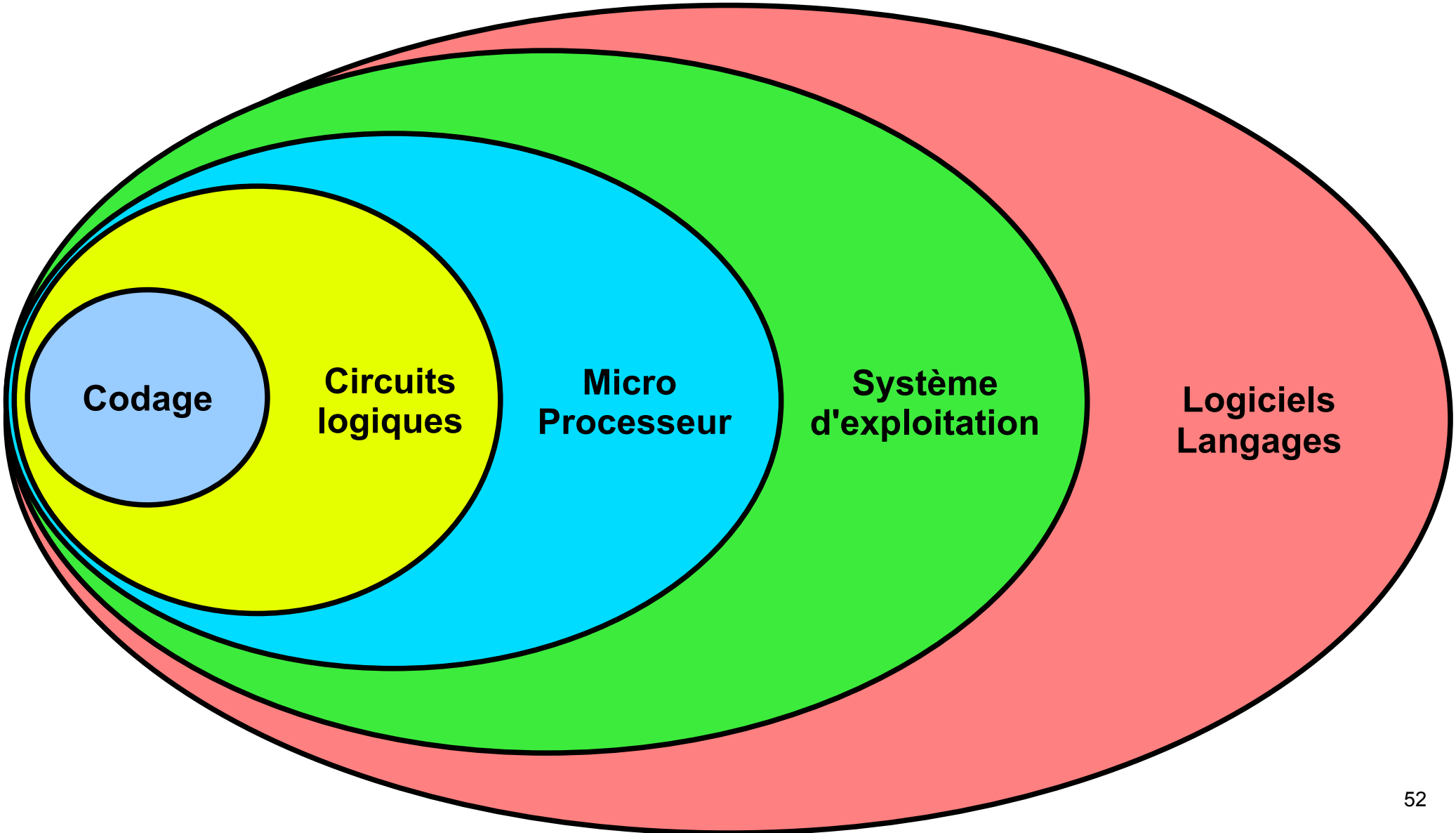
- Un microprocesseur traite un ensemble d'instructions relativement simples.
  Par exemple :
  - add (pour faire l'addition)
  - sub (soustraction)
  - and / or (opérations logiques)
  - st (store = mémorisation d'une valeur)
  - ro (rotation des bits : rol (left) *et* ror (right))
  - b (branchement / saut, conditionnel ou non)

Exemple de programme

- L'assembleur est le langage de prédilection pour programmer *directement* un µ-processeur
  - Exemple 1
    - addi r1, r0, 15 (ajouter r0=0 à 15 et mettre le résultat dans r1)
    - addi r1, r1, 20 (ajouter r1=15 à 20 et mettre le résultat dans r1 soit 35)
  - Exemple 2
    - addi r1, r0, 15
    - or r4, r0, r1
    - st (r1+0), r2
    - ldx r3, (r0+r1)
    - addi r1, r1, 1
    - addi. r4, r4, -3
    - add r2, r3, r2
    - stx (r0+r1), r2
    - add r2, r0, r3
    - bgt -7

Calcul la suite de Fibonnacci :
1,2,3,5,8,13,21
=> voir en TD

Conclusion


Codage
Circuits logiques
Micro Processeur
Système d'exploitation
Logiciels Langages