M5: Langue – Informatique 2

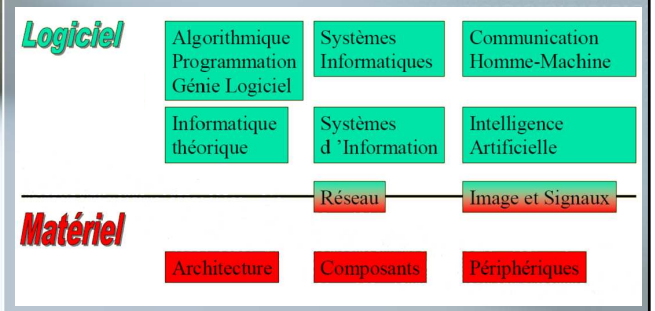
E2: Informatique 2

M. HIMMI
2014

L'Informatique

- 2 grands domaines
  - Le logiciel : programmes
  - Le matériel : ordinateur (électronique)
- 2 points de vue
  - La science de traitement automatique de l'information
  - La science des ordinateurs

Domaines de l'informatique



Science Informatique

- **Science** qui regroupe l'ensemble des théories et des techniques permettant de **traiter de l'information** à l 'aide d'un ordinateur
- **S**ciences et **T**echnologies de l'**I**nformation et de la **C**ommunication

L'ordinateur

- Machine (**calculateur**) commandée par un **programme** enregistré qui permet de traiter des informations en exécutant une **séquence finie d'instructions** (opérations arithmétique et logique)
  - Universel (programmée pour des problèmes différents)
  - Rapide (Millions d'Instructions par seconde; MIPS)
  - Fiable (fonctionnement régulier et sûr)
  - Grande Capacité mémoire

5

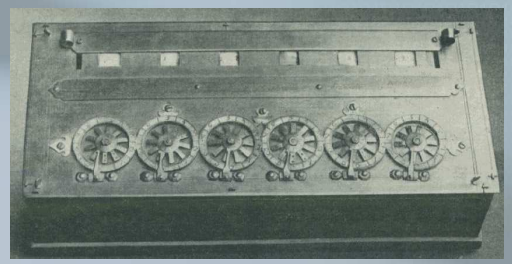
Historique

- Boulier chinois, 700

6

Historique

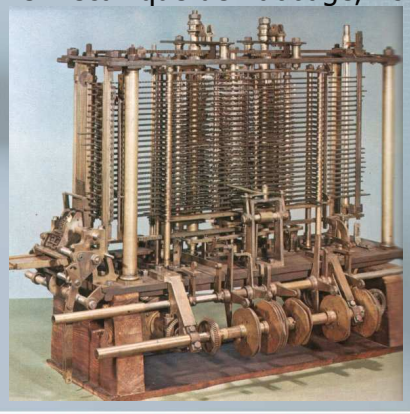
- Machine de Pascal, 1642



7

Historique

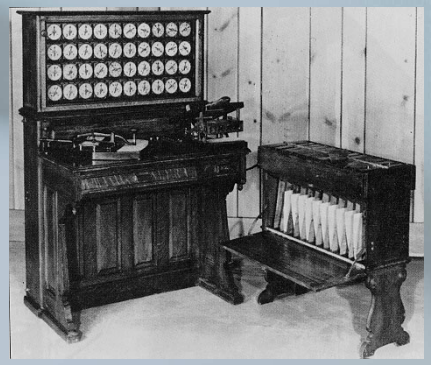
- Mémoire mécanique de Babbage, 1883

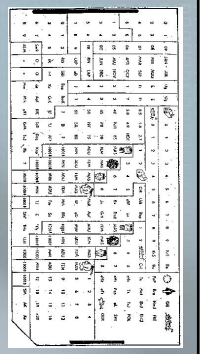


8

Historique

- Machine à carte de Hollerith, 1890

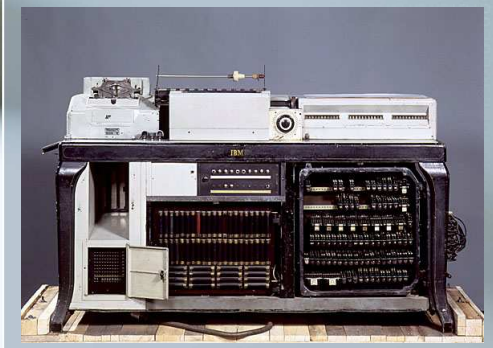


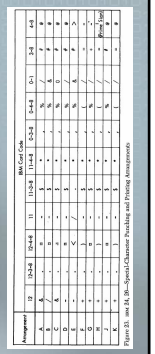


9

Historique

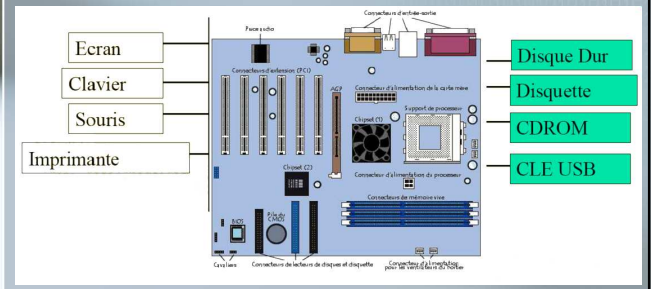
- Machines électroniques IBM, 1940





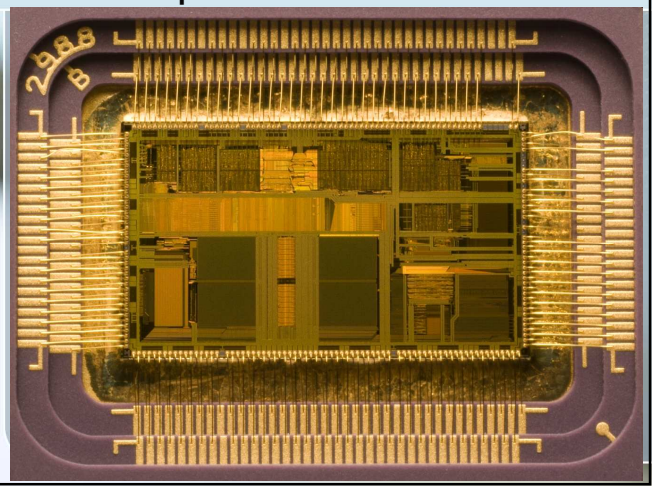
10

Ordinateur de type PC



11

L'unité de traitement
Le microprocesseur



La mémoire centrale

- Ensemble fini de cellules (ou case)
- Chaque case a un numéro : adresse
- Unités de mesure: **bit**, octet (Byte), mot
- Chaque case correspond à un mot

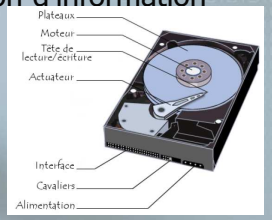
- mémoire morte ROM
  - lecture seule
  - instructions de démarrage
  - Quelques Ko

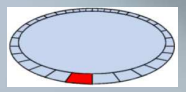
- mémoire vive RAM
  - lecture et écriture
  - volatile
  - taille : centaines de Mo
  - débit de transfert : 2 Go/s
  - contient à la fois le programme à exécuter et ses données(opération de chargement)

13

La mémoire auxiliaire

- Stockage et restitution d'information
- disque dur
  - lecture et écriture
  - taille : centaines de Go
  - débit de transfert : 100 Mo/s
  - logiciels + données





14

La mémoire auxiliaire

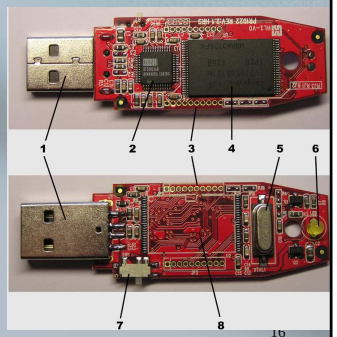
- disquettes
  - lecture et écriture
  - taille : 1,44 Mo
  - débit de transfert : 150 ko/s
  - sauvegarde données (transport)

15

La mémoire auxiliaire

- clés USB
  - lecture et écriture
  - taille : centaines de Mo
  - débit de transfert : 1 Mo/s
  - sauvegarde données (transport)

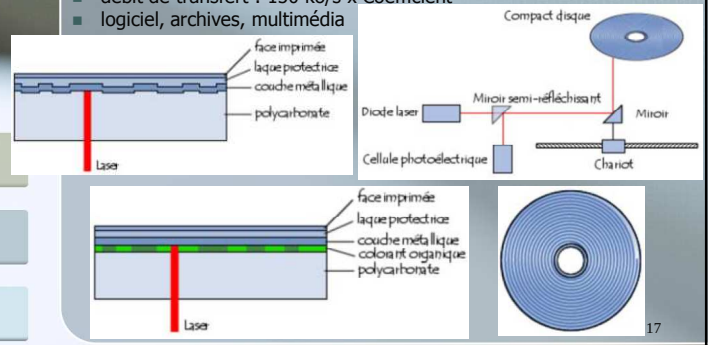
1. Connecteur USB mâle (type A).
2. Contrôleur pour l'USB 2.0
3. JP1 et JP2 pour tests et débogage.
4. Mémoire flash
5. Oscillateur à quartz 12 MHz.
6. DEL pour indiquer l'activité de la clé.
7. Interrupteur (protéger la clé en écriture).
8. Zone prévue pour étendre la capacité sans avoir à créer un autre schéma.



16

La mémoire auxiliaire

- CDROM, DVDROM
  - lecture
  - taille : 700 Mo (CD), 5 Go (DVD)
  - débit de transfert : 150 ko/s x Coefficient
  - logiciel, archives, multimédia



17

Familles d 'ordinateur

- Ordinateurs centraux (MainFrame)
- Ordinateurs Personnels (Personal Computer)
  - Ordinateur de Bureau (Desktop computer)
  - Ordinateur Portable (Laptop, Notebook, …)
- Ordinateurs de poche/ assistants personnels (Personal Digital Assistant)
- Systèmes industriels - temps réel

18

Notation binaire

L'ordinateur manipule exclusivement des informations binaires:

- Ce sont des informations qui n'ont que deux états:

  ouvert – fermé vrai – faux, etc.
- On symbolise une information binaire par 1 et 0
  - Le 1 et le 0 sont des signes pour désigner une information, indépendamment de son support physique.

Avec une telle information binaire, on ne va pas loin:

➢ Utiliser les informations binaires par paquet de 8 bits (octets).

19

Notation binaire

- **Un octet peut servir à coder 256 entités différentes. Exemple:**
  - nombres entiers de 1 à 256
  - nombres entiers de 0 à 255
  - nombres entiers de –127 à +128
  - autre chose qu'un nombre: souvent employé pour du texte
  - …

C'est une affaire de codage !

- Pour des nombres plus grands que 256, des nombres négatifs ou décimaux on utilise plus d'un octet:
  - 2 octets → 256 x 256 = 65 536 possibilités
  - 3 octets → 256 x 256 x 256 = 16 777 216 possibilités
  - …

20

Notation binaire

Définitions

- 8 bits = octet 2^8 =256
- 16 bits = demi-mot 2^{16} =65 536
- 32 bits = mot 2^{32} =4 294 967 296

Remarque $2^{10} = 1024 \approx 10^3$

- 1 Ko = 1 024 octets
- 1 Mo = 1 024 * 1 024 = 1 048 576 octets
- 1 Go = 1 024 * 1 024 * 1 024 = 1 073 741 824 octets

21

Multiples of bytes

SI decimal prefixes			IEC binary prefixes	
Name (Symbol)	**Standard SI**	**Binary usage**	**Name (Symbol)**	**Value**
kilobyte (kB)	10^3	2^{10}	kibibyte (KiB)	2^{10}
megabyte (MB)	10^6	2^{20}	mebibyte (MiB)	2^{20}
gigabyte (GB)	10^9	2^{30}	gibibyte (GiB)	2^{30}
terabyte (TB)	10^{12}	2^{40}	tebibyte (TiB)	2^{40}
petabyte (PB)	10^{15}	2^{50}	pebibyte (PiB)	2^{50}
exabyte (EB)	10^{18}	2^{60}	exbibyte (EiB)	2^{60}
zettabyte (ZB)	10^{21}	2^{70}	zebibyte (ZiB)	2^{70}
Yottabyte (YB)	10^{24}	2^{80}	yobibyte (YiB)	2^{80}

22

Système de numérotation

Qui se rappelle des règles du système de numérotation par position en base 10?

- on retrouve facilement que: **9562** c'est:

  9 x 1000 + **5** x 100 + **6** x 10 + **2** x 1

  (**9**x 10 x 10 x 10) + (**5** x 10 x 10) + (**6** x 10) + (**2** x 1)

  ou **9** x 10^3 + **5** x 10^2 + **6** x 10^1 + **2** x 10^0

- Deux conséquences:
  - Nous nous servons de dix chiffres, pas un de plus, pas un de moins.
  - La position du chiffre dans un nombre désigne la puissance de dix par laquelle ce chiffre doit être multiplié pour reconstituer le nombre.

23

Système de numérotation

Appliquons ce principe au binaire: (deux chiffres seulement !)

- Pour reconstituer le nombre dans la base décimale:

Prenons un octet : **1 1 0 1 0 0 1 1**

- Ce nombre représente en base dix:

1 x 2^7 + **1** x 2^6 + **0** x 2^5 + **1** x 2^4 + **0** x 2^3 + **0** x 2^2 + **1** x 2^1 + **1** x 2^0

1 x 128 + 1 x 64 + 1 x 16 + 1 x 2 + 1 x 1

128 + 64 + 16 + 2 + 1

211

- Inversement pour un nombre en décimal **186**:

1 x 128 + 0 x 64 + 1 x 32 + 1 x 16 + 1 x 8 + 0 x 4 + 1 x 2 + 0 x 1

1 x 2^7 + 0 x 2^6 + 1 x 2^5 + 1 x 2^4 + 1 x 2^3 + 0 x 2^2 + 1 x 2^1 + 0 x 2^0

1 0 1 1 1 0 1 0

24

Notation hexadécimal

Représenter un octet par une suite de huit bits n'est pas très pratique!

→ On considère un octet comme deux paquets de 4 bits (les quatre de gauche, et les quatre de droite):

- Avec 4 bits nous pouvons coder 2^4=16 nombres différents

→ Choisir de calculer à base seize

16 nombres différents se représentent avec un seul chiffre:

0, 1, 2, …, 9, A, B, C, D, E, F

25

Notation hexadécimal

Prenons un octet au hasard :**1 0 0 1 1 1 1 0**

- Première méthode:
  on passe en décimal : 158
  puis en hexadécimal : **9E**

- Deuxième méthode:
  Divisons **1 0 0 1 1 1 1 0** en :
  1 0 0 1 (gauche) → c'est 8 + 1, donc 9
  1 1 1 0 (droite) → c'est 8 + 4 + 2 donc 14=E
  Le nombre s'écrit donc en hexadécimal : **9E**

Représentation très simple des octets binaire

26

Exercice Numérotation

BINAIRE	OCTAL	DECIMAL	HEXADECIMAL
			11
		11	
	11		
11			
	1		
		74	
			74
	74		
74			
			ABBA
101010101010			
		255	

27

Addition binaire

L'addition en binaire se fait avec les mêmes règles qu'en décimale :

- On commence à additionner les bits de poids faible puis on a des retenues lorsque la somme de deux bits de même poids dépasse la valeur de l'unité la plus grande, cette retenue est reportée sur le bit de poids plus fort suivant...

exemple :

```
  0 1 1 0 1        13
+ 0 1 1 1 0      + 14
-----------      ----
  1 1 0 1 1        27
```

28

Multiplication binaire

- La multiplication se fait en formant un produit partiel pour chaque digit du multiplicateur.
- Lorsque le bit du multiplicateur est nul, le produit partiel est nul, lorsqu'il vaut un, le produit partiel est constitué du multiplicande décalé du nombre de positions égal au poids du bit du multiplicateur.

Exemple :

```
     0 1 0 1          5      multiplicande
 x   0 0 1 0      x 2        multiplicateur
---------------   -----
     0 0 0 0        1 0
   0 1 0 1
 0 0 0 0
---------------
 0 0 1 0 1 0
```


Codage des valeurs numériques

- Les entiers naturels peuvent être directement codés en binaire.
- Les autres types de valeurs nécessitent **un codage**

Codage des valeurs numériques

Codage des entiers négatifs (entiers signés)

- Ce codage doit répondre à trois critères:
  - Les nombres négatifs doivent pouvoir être distingués des positifs.
  - La somme d'un nombre et de son opposé est nulle.
  - L'opposé de l'opposé d'un nombre est égal à ce nombre

Codage des valeurs numériques

Codage des entiers négatifs (entiers relatifs)

- Au moins 3 façons pour coder:
  - binaire signé (signe et valeur absolue)
  - complément à 1
  - code complément à 2

Codage des valeurs numériques: entiers relatifs

binaire signé

Le bit le plus significatif est réservé pour le signe:

nombre négatif → le bit le plus fort = 1

nombre positif → le bit le plus fort = 0

La valeur absolue est codée en binaire naturel

Exemple sur 8 bits:

-24 est codé en binaire signé par: 1 0 0 1 1 0 0 0

24 est codé en binaire signé par: 0 0 0 1 1 0 0 0

-128 est impossible à codé sur 8 bits, elle nécessite minimum 9 bits

33

Codage des valeurs numériques: entiers relatifs

binaire signé

- **Etendu de codage :**

  Avec 4 bits : entre -7 et +7

  Avec n bits, entre $-(2^{n-1}-1)$ et $(2^{n-1}-1)$

- **Limitations:**

  Deux représentations du zéro : + 0 et – 0

  Multiplication et addition pas évidentes.

34

Codage des valeurs numériques: entiers relatifs

Complément à 1

- les nombres positifs sont codés de la même façon qu'en binaire pure.
- un nombre négatif est codé en inversant chaque bit de la représentation de sa valeur absolue
- Le bit le plus significatif est utilisé pour coder le signe du nombre :
  - si le bit le plus fort = 1 alors nombre négatif
  - si le bit le plus fort = 0 alors nombre positif

Exemple sur 8 bits:

24 est codé par: 0 0 0 1 1 0 0 0

-24 est codé par: 1 1 1 0 0 1 1 1

35

Codage des valeurs numériques: entiers relatifs

Complément à 1

- **Limitations:**
  - Deux représentations du zéro : + 0 et – 0

    Sur 8 bits : +0=0 0 0 0 0 0 0 0

    -0=1 1 1 1 1 1 1 1
  - Multiplication et addition pas évidentes.

36

Codage des valeurs numériques: entiers relatifs

Complément à 2.

- Les nombres positifs sont représentés en binaire simple
- Les nombres négatifs sont obtenus de la manière suivante:
  - On inverse les bits de l'écriture binaire de sa valeur absolue (complément à un) puis,
  - On ajoute 1 au résultat (les dépassements sont ignorés).
- Exemple:
  l'opposé de 00011110
  est égal à 11100001+1 = 11100010.
- **Avec ce système, tous les nombres négatifs commencent par un 1 (*digit de gauche*)**

37

Codage des valeurs numériques

- Pour vérifier que la somme d'un nombre et de son opposé est nulle, il faut d'abord savoir faire une addition!
- L'addition se pose exactement comme en base 10, avec des retenues (1+1=10, je pose 0 et je retiens 1...).
- Il y a évidemment une astuce: le résultat de l'addition s'écrit lui aussi sur un octet, la dernière retenue n'apparaîtra donc pas dans ce résultat !

38

Calcul binaire (exercices)

1. Vérifiez que la somme de deux opposés est nulle en additionnant 00011110 et 11100010, ou 10001100 et 01110100
2. L'opposé de l'opposé d'un nombre est égal à ce nombre: en utilisant la méthode décrite, prenez l'opposé de 11100010, puis celui de 01110100, et vérifiez que le premier est 00011110 et le second 10001100.
3. Vérifiez que la somme de 11100010 (-30) et 01110100 (116) se lit bien 86, et que celle de 10001100 et 00011110 se lit bien -86.

39

Calcul binaire (exercices)

4. Quel est le plus grand nombre positif que l'on peut écrire dans un octet, avec ce codage?
5. Comment s'écrit son opposé?
6. Prendre l'opposé de 10000000. Que constate-t-on?
7. Ajouter 10000000 et 01111111. Que vaut le résultat ? Quelle valeur faut-il attribuer à 10000000, avec ce codage?
8. Combien de valeurs numériques peut-on écrire dans un octet, avec ce codage? Comparer au binaire ordinaire.

40

Calcul binaire (exercices)

- Quelle est la valeur décimale des suites binaires (1010, 10010110 et 1011010011101001), si elles sont codées en : binaire pur, Binaire signé, Complément à 1, Complément à 2
- Coder les nombres +20 et -15 Sur 4, 8 et 16 bits, en : Binaire pur, Binaire signé, Complément à 1 et Complément à 2
- Calculer 20-15 sur 8 et 16 bits en Complément à 2

Codage d'un nombre réel

- **Format virgule fixe:** (premières machines)
  - Constitué d'une partie entière et d'une partie décimale séparées par une virgule.
  - La position de la virgule est fixe

Exemple:

$514{,}25_{(10)}$; $101{,}001_{(2)}$; $A1B{,}F0B_{(16)}$

- **Format virgule flottante:**
  - défini par : ± m.be avec
    - Le signe + ou –
    - La mantisse m (en virgule fixe)
    - L'exposant e (un entier relatif)

Codage d'un nombre réel

- La norme IEEE définit la façon de coder un nombre réel. Cette norme se propose de coder le nombre sur 32 bits et définit trois composantes :
  - **le signe est représenté par un seul bit, le bit de poids fort (celui le plus à gauche)**
  - **l'exposant est codé sur les 8 bits consécutifs au signe**
  - **la mantisse (les bits situés après la virgule) sur les 23 bits restants**

seeeeeeeemmmmmmmmmmmmmmmmmmmmmmm

  - le **s** représente le bit relatif au signe
  - les **e** représentent les bits relatifs à l'exposant
  - les **m** représentent les bits relatifs à la mantisse

Codage d'un nombre réel

- Pour le réel en double précision (64 bits) celle norme devient:
  - **1 bit de signe de la mantisse**
  - **11 bits pour l'exposant**
  - **52 bits pour la mantisse**

Codage d'un nombre réel

Certaines conditions sont respecter pour les exposants :

- l'exposant 00000000 est interdit
- l'exposant 11111111 est interdit.
- Il faut rajouter 127 (01111111) à l'exposant pour une conversion de décimal vers un nombre réel binaire. Les exposants peuvent ainsi aller de -254 à 255

La formule d'expression des nombres réels est:

(-1)^S * 2^(E - 127) * (1 + F)

où:

- **S est le bit de signe** et l'on comprend alors pourquoi 0 est positif (-1^0=1).
- **E est l'exposant** auquel on doit bien ajouter 127 pour obtenir son équivalent codé.
- **F est la partie fractionnaire**, la seule que l'on exprime et qui est ajoutée à 1 pour effectuer le calcul.

45

Codage d'un nombre réel

Exemple: Soit à coder la valeur 525,5

- 525,5 est positif donc le 1er bit sera 0.
- 525.5 en base 2 est 1000001101,1
- En normalisant 1,0000011011*2^9
- On ajoute 127 à 9 =136 = **1000 1000** en base 2
- La mantisse est composée de la partie décimale de 525,5 en base 2 normalisée, c'est-à-dire **0000 0110 11**.
- La mantisse doit occuper 23 bits on ajoute des zéros pour compléter: **0000 0110 1100 0000 0000 000**

La représentation de 525,5 avec la norme IEEE est donc:

0 1000 1000 0000 0110 1100 0000 0000 000

0100 0100 0000 0011 0110 0000 0000 0000

(**44 03 60 00** en hexadécimal)

46

Codage d'un nombre réel

Exemple: Soit à coder la valeur -0.625

- Le bit **s** vaut 1 car 0,625 est négatif .
- 0,625 s'écrit en base 2: 0,101
- En normalisant 1.01 x 2^{-1}
- On ajoute 127 à -1 =126 = **111 1110** en base 2
- La mantisse est **0100 0000 0000 0000 0000 000** (seuls les chiffres après la virgule sont représentés, le nombre entier étant toujours égal à 1) .

La représentation de -0.625 avec la norme IEEE est donc:

1 0111 1110 0100 0000 0000 0000 0000 000

1011 1111 0010 0000 0000 0000 0000 0000

(**FF 20 00 00** en hexadécimal)

47