Architecture des Ordinateurs et Systèmes d'Exploitation

Cours n°1

Structure de base d'un ordinateur
Représentation de l 'information

Ph. Leray

3ème année

Objectifs de l'UV

- Décrire les composants d'un ordinateur (processeur, mémoire, périphériques,...)
- Présenter le principe de fonctionnement d'un ordinateur en décrivant les différentes couches existantes
- Illustrer les liens entre :
  - les langages de programmation de haut niveau,
  - les systèmes d'exploitation,
  - le matériel
- Décrire le fonctionnement d'un système d'exploitation
- Evaluer les performances d'un ordinateur et suivre l'évolution de ses composants

Déroulement de l'UV

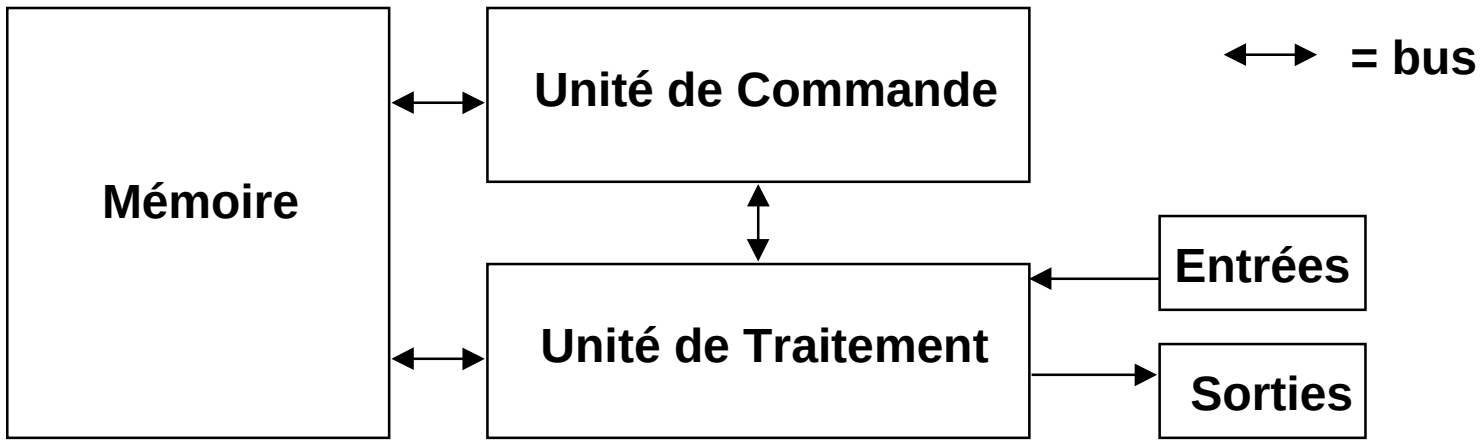
- **Semestre = 15 semaines**
- **Chaque vendredi :**
  - **Cours : 1h30 / semaine (matin)**
  - **Travaux dirigés : 1h30 / semaine (après-midi)**
  - **Travaux pratiques : 1h30 / semaine (après-midi)**
  - **Travaux tutorés : 55' / 2 semaines (matin)**
- **Calendrier :**
  - **septembre - fin octobre : Architecture des Ordinateurs**
  - **novembre - mi décembre : Systèmes d'Exploitation**
  - **fin décembre : Exposés - Présentation de Projets**
  - **janvier : Projet, Examen Final**
- **Notation :**
  - **Examen final : 40 %**
  - **Projet : 30 %**
  - **TD / TP / TT : 30 %**

Un peu d'histoire

- 1623 (Schickard) 1643 (Pascal)
  - Premières machines à calculer : Roues dentées, engrenages
  - Peu de progrès jusqu'au XX siècle : Relais electromagnétiques
- 1940 Jurassic Park (ENIAC 1943-46 : 30 t, 160m^2)
  - Technologie = Tubes à vide (18.000) et commutateurs (6.000)
- 1958 La révolution du transistor
  - Ex: CDC 1604 (25.000 transistors)
- 1965 Les mini-ordinateurs (IBM System/360?)
  - Circuit Intégré = dizaines de transistors sur une puce
- 1975 Les micro-ordinateurs
  - VLSI (Very Large Scale Integration) = dizaines de milliers de transistors sur une puce
  - Ordinateur = outil de bureau (PC=Personnal Computer)

Architecture de base

- **Idée de Von Neuman (1945), peu de changement depuis :**



  - **Mémoire = stocke les informations et les programmes**
  - **UC = cherche les instructions en mémoire, les reconnait et indique à l'UT ce qu'elle doit effectuer**
  - **UT = assure les opérations élémentaires que demande l'UC (opérations logiques, arithmétiques, ...)**
  - **Bus = transfère les informations**

Fonctionnement en couches

Langage de base = 0|1

➜ **(0) Couche physique (le matériel)**

Avec les composants, on peut faire des choses plus évoluées = micro-instruction

➜ **(1) Couche microprogrammée**

Avec les micro-instructions, on peut inventer de nouvelles instructions plus simples = langage machine

➜ **(2) Couche "machine"**

+ Organisation de la mémoire, de l'exécution des programmes

➜ **(3) Couche du système d'exploitation**

Apparition d'un langage "compréhensible" par l'utilisateur mais pas par la machine (Traduction vers (2) (3) = "assembleur")

➜ **(4) Couche du langage d'assemblage**

Création de langages plus évolues (Pascal, C, ...)

➜ **(5) Couche des langages d'application**

Représentation de l'Information

- **Une information peut être représentée de plusieurs façons :**

  ex : le chiffre quatre : 4 100 IV

- **Représentation des nombres**

  Représentation usuelle = base 10 symboles = { 0,1,2,3,...9 }

  $1999 = 1 * 10^3 + 9 * 10^2 + 9 * 10^1 + 9$

  Représentations courantes en Informatique = bases 2, 8, 16.

  base 2 = binaire (représentation interne) un bit : { 0,1 }

  base 8 = octal { 0,1,2,3,...7 }

  base 16 = héxadécimal { 0,1,2,3,...9,A,B,C,D,E,F }

Un peu d'Arithmétique binaire

- **Table d'addition :**

0	0	1	1
+ 0	+ 1	+ 0	+ 1
Somme 0	1	1	0
Retenue 0	0	0	1

- **Addition de 2 nombres binaires :**

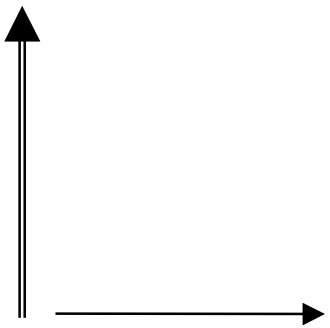
```
Ret :      111
        10100101
   +        10111
  ----------------
        10111100
```

Représentation des entiers naturels

- **Passage de la base 10 vers la base B:**

  divisions successives par B

  22 / 2 = 11 (reste 0)
  11 / 2 = 5 (reste 1)
  5 / 2 = 2 (reste 1)
  2 / 2 = 1 (reste 0)
  1 / 2 = 0 (reste 1)



  $(22)_{10} = (10110)_2$

- **Passage de la base B à la base 10 :**

  $(X)_B = (x_{n-1}x_{n-2}...x_1x_0)_B = (x_{n-1} * B^{n-1} + x_{n-2} * B^{n-2} + .. + x_1 * B + x_0)_{10}$

  ex: $(10110)_2 = 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 1 * 2^1 + 0 = 16 + 4 + 2 = (22)_{10}$

Représentation des réels

(Virgule fixe)

- **Même méthode pour la partie entière**
- **On garde la virgule**
- **La partie décimale se décompose en puissances négatives de 2**

- **Ex : $(0,75)_{10}$ = ?**

  0,75 * 2 = 1,5 (on garde 1, reste 0,5)

  0,5 * 2 = 1,0 (on garde 1 , reste 0 : terminé)

  ➜ $(0,75)_{10} = (0,11)_2 = 1 * 2^{-1} + 1 * 2^{-2}$ (= 0,5 + 0,25)

- **Ex : $(0,65)_{10}$ = ?**

  0,65 * 2 = 1,3 (1)

  0,3 * 2 = 0,6 (0)

  0,6 * 2 = 1,2 (1)

  0,2 * 2 = 0,4 (0)

  0,4 * 2 = 0,8 (0)

  0,8 * 2 = 1,6 (1)

  0,6 ...

  ➜ $(0,65)_{10} = (0,10\underline{1001})_2$

Codage des nombres

Codage = représentation avec un nombre limité de bits.

- **Problème de « dépassement de capacité »**

  Exemple pour les entiers naturels :

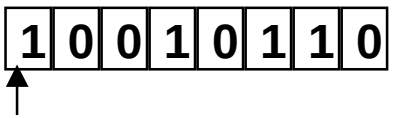
Représentation			Codage sur 8 bits	
	10101010	(170)		1 0 1 0 1 0 1 0
+	11000000	(192)	+	1 1 0 0 0 0 0 0
	101101010	(362)	?	1 0 1 1 0 1 0 1 0

  ➜ 362 est un entier trop grand pour être codé sur 8 bits

Codage des entiers relatifs 1/3

- **Problème = représenter le signe**

- **Première méthode : signe ("+" = 0, "-" = 1) + valeur absolue**
  - **ex: codage de $(-22)_{10}$ sur 8 bits (1 bit de signe + 7 bits)**

    = 

  - **Avantage : facile à lire pour l'utilisateur**
  - **Problèmes :**
    - » **2 représentations possibles pour 0 (+0 et -0)**
    - » **comment additionner 2 nombres de signe différent ?**

Codage des entiers relatifs 2/3

- **Seconde méthode : complément à 2**

  technique de passage de X à -X (en complément à 2) =

  - » **on part de X :**

    +22

0	0	0	1	0	1	1	0

    +1

0	0	0	0	0	0	0	1

  - » **on complémente : (0 devient 1, et inversement)**

  - » **on ajoute 1 :**

1	1	1	0	1	0	0	1

1	1	1	1	1	1	1	0

    -22

1	1	1	0	1	0	1	0

    -1

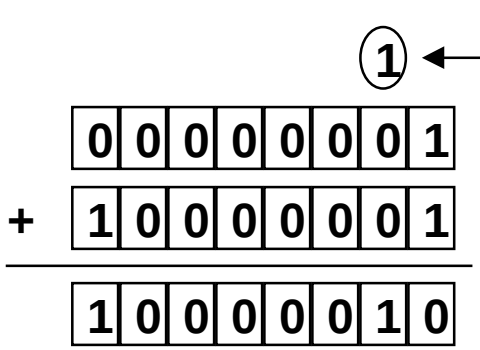
1	1	1	1	1	1	1	1

  - **Inconvénient : plus difficile à interpréter**
  - **Avantages :**
    - » **un seul codage pour 0**
    - » **pas de problème d'addition**

Codage des entiers relatifs 3/3

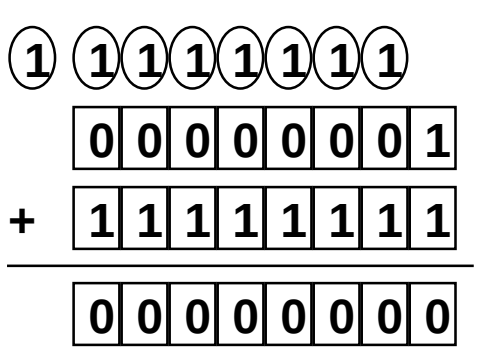
1 + (-1) = ??

"Signe + valeur absolue"



L'algorithme d'addition n 'est pas adapté à ce codage

Complément à 2



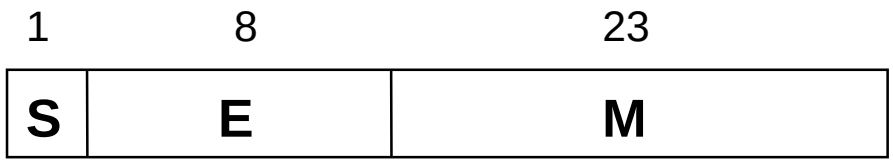
L'algorithme d'addition pour les entiers naturels marche aussi pour le codage en complément à 2

Retenues

Codage des réels (IEEE 754) 1/2

(Virgule flottante)

- **Codage sur 32 bits (simple), 64 (double) ou 80 (étendue)**
- **inspiré de la notation scientifique (ex : $+1{,}05 \times 10^{-6}$)**
- **Ex en simple précision :**



$X = (-1)^S * 1{,}M * 2^{E-127}$ **avec $0 < E < 255$**

ex: **$X = (-1{,}5)_{10}$** **(S=1; M=0,5; E=127)**

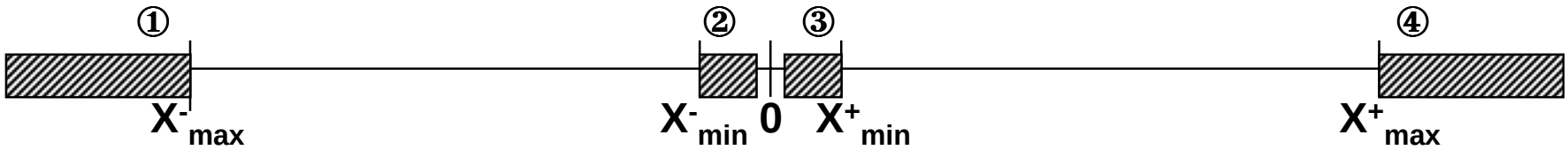
X = | **1** | **01111111** | **1000…000** |

Codage des réels (IEEE 754) 2/2

Cas particuliers :

- ±∞ (M=0, E=255) (résultat d'un div/0,...)
- 0 (M=0, E=0) (impossible d'écrire 0 sous la forme $1,M*2^{E-127}$)
- NaN (M≠0, E=255) (le résultat n'est pas un nombre : ∞ / ∞ , …)

Les dépassements de capacité :



- (1) débordement supérieur négatif
- (2) débordement inférieur négatif
- (3) débordement inférieur positif
- (4) débordement supérieur positif

Codage des caractères (code ASCII) 1/2

- **Idée = associer un nombre à un caractère**

- **Codage ASCII (American Standard Code for Information Exchange) =**

  - **7 bits (donc 128 possibilités)**
  - **Les caractères 0 (00) à 31 (1F) sont des caractères spéciaux**
    - **saut de ligne, tabulation, beep ...**
  - **Les caractères 32 (20) à 127 (7F) représentent :**
    - **les minuscules**
    - **les majuscules**
    - **les signes de ponctuation …**

Codage des caractères (code ASCII) 2/2

Table des codes ASCII (hexadécimal)

00 NUL	01 SOH	02 STX	03 ETX	04 EOT	05 ENQ	06 ACK	07 BEL
08 BS	09 HT	0A NL	0B VT	0C NP	0D CR	0E SO	0F SI
10 DLE	11 DC1	12 DC2	13 DC3	14 DC4	15 NAK	16 SYN	17 ETB
18 CAN	19 EM	1A SUB	1B ESC	1C FS	1D GS	1E RS	1F US
20 SP	21 !	22 "	23 #	24 $	25 %	26 &	27 '
28 (	29)	2A *	2B +	2C ,	2D -	2E .	2F /
30 0	31 1	32 2	33 3	34 4	35 5	36 6	37 7
38 8	39 9	3A :	3B ;	3C <	3D =	3E >	3F ?
40 @	41 A	42 B	43 C	44 D	45 E	46 F	47 G
48 H	49 I	4A J	4B K	4C L	4D M	4E N	4F O
50 P	51 Q	52 R	53 S	54 T	55 U	56 V	57 W
58 X	59 Y	5A Z	5B [	5C \	5D]	5E ^	5F _
60 `	61 a	62 b	63 c	64 d	65 e	66 f	67 g
68 h	69 i	6A j	6B k	6C l	6D m	6E n	6F o
70 p	71 q	72 r	73 s	74 t	75 u	76 v	77 w
78 x	79 y	7A z	7B {	7C |	7D }	7E ~	7F DEL