Systèmes Informatiques et Réseaux

Chapitre 1 : Codage de l'information

2020 - 2021

Sommaire

1. PROGRAMME
2. Introduction générale
3. Codage binaire
4. Système de numération
5. Le codage des nombres
6. Les caractères
7. Le Son et les Images

Sommaire

1 PROGRAMME

2 Introduction générale

3 Codage binaire

4 Système de numération

5 Le codage des nombres

6 Les caractères

7 Le Son et les Images

1ère année

S4. SYSTEMES INFORMATIQUES ET RESEAUX		
CONTENUS		**CAPACITÉS ATTENDUES**
S41. Systèmes d'exploitation et architecture des ordinateurs **Architecture des ordinateurs** - Codage de l'information : bits, octets, codage des nombres, codage ASCII, - Matériel : Unités centrale, processeur, mémoire, entrées/sorties, bus, périphériques, - Périphériques : Supports magnétiques et optiques, écrans, Imprimantes, scanner **Système d'exploitation d'un poste de travail** - Principes des systèmes d'exploitation - Notions de systèmes de fichiers. - Gestion des entrées-sorties. - Gestion des fichiers. - Les interfaces utilisateurs graphiques et textuelles.	**48**	• Décrire la structure et le fonctionnement d'un ordinateur. • Reconnaître les liens entre les différents composants d'un ordinateur. • Décrire le rôle et les principales caractéristiques techniques et fonctionnelles des périphériques • Installer et configurer un périphérique • Expliquer les principes de base du fonctionnement d'un système d'exploitation mono-utilisateur • Utiliser les commandes d'un système d'exploitation mono-utilisateur • Maîtriser l'interface graphique d'un poste de travail

1ère année

S42. Réseaux informatiques **Réseau Internet et services** - Historique, structure et fonctionnement du réseau Internet. - Protocoles du modèle Internet : IP, TCP, UDP. - Service de résolution de noms : DNS. - Service de courrier électronique : SMTP, POP, IMAP, MIME. - Service de pages Web statique et dynamique : HTTP. - Service de transfert de fichiers : FTP.	**16**	• Connaître l'organisation du réseau Internet. • Identifier, décrire et utiliser les différents services Internet.

1ère année

S43. Outils et logiciel bureautiques - Texteur - Tableur - Logiciel de présentation - Logiciels de communication : messagerie électronique, navigation sur réseaux, transfert de fichier - Compression, décompression et conversion de fichiers, prise de contrôle à distance - Échange de données entre logiciels	**48**	• Prendre en main un logiciel à partir d'une documentation technique • Utiliser un texteur, un tableur, un grapheur, un logiciel de présentation assistée et un logiciel de communication ou un logiciel intégré. • Installer un logiciel et le mettre à disposition d'un utilisateur ou d'un groupe d'utilisateurs

2ème année

S4. SYSTEMES INFORMATIQUES ET RESEAUX		
CONTENUS		**CAPACITÉS ATTENDUES**
S41. Systèmes d'exploitation et architecture des ordinateurs **Système d'exploitation multi-utilisateurs et réseau** Gestion des ressources Gestion des utilisateurs	**12**	• Utiliser les interfaces graphiques d'un système d'exploitation multiutilisateurs et/ou réseau.
S42. Réseaux informatiques **Notions de base sur l'architecture des réseaux** - Le modèle OSI - Typologie des réseaux : topologie, protocoles, réseaux locaux, matériel d'interconnexion - Les techniques d'adressage d'un réseau local - Installation et configuration d'un réseau local **Réseau Internet et services** - Installation, configuration, gestion et utilisation d'un serveur Web.	**28**	• Obtenir une culture générale sur l'architecture des réseaux en termes de matériel, interconnexion, topologie et support. • Comprendre le fonctionnement d'un réseau local Ethernet. • Comprendre l'organisation de l'adressage IP. • Installer et configurer un réseau local Ethernet. • Installer et configurer un serveur Web

Sommaire

1. PROGRAMME
2. Introduction générale
3. Codage binaire
4. Système de numération
5. Le codage des nombres
6. Les caractères
7. Le Son et les Images

INFORMATIQUE

C'est la science de traitement automatique de l'information par une machine programmable.

INFORMATIQUE

C'est la science de traitement automatique de l'information par une machine programmable.

Le mot informatique a été créé en 1962 par Philippe Dreyfus. Il s'agit d'un néologisme de la langue française fait de la contraction des deux mots « automatique » et « information »

Traitement

c'est l'ensemble des opérations réalisées sur des informations de base appelées les données pour obtenir de nouveaux informations appelées les résultats

Traitement

c'est l'ensemble des opérations réalisées sur des informations de base appelées les données pour obtenir de nouveaux informations appelées les résultats

Un traitement peut être:

- **Manuel :** :quand il est réalisé manuellement par l'être humain.

Traitement

c'est l'ensemble des opérations réalisées sur des informations de base appelées les données pour obtenir de nouveaux informations appelées les résultats

Un traitement peut être:

- **Manuel :** :quand il est réalisé manuellement par l'être humain.
- **Automatique :** quand il est réalisé par une machine.

Système informatique

Un système informatique est l'ensemble des moyens logiciels (Software) et matériels (hardware) nécessaires pour satisfaire les besoins informatiques des utilisateurs.

Système informatique

Un système informatique est l'ensemble des moyens logiciels (Software) et matériels (hardware) nécessaires pour satisfaire les besoins informatiques des utilisateurs.

- **Matériel :** :désigne tout ce qui a un caractère matériel comme : ordinateur, imprimante, souris ....

Système informatique

Un système informatique est l'ensemble des moyens logiciels (Software) et matériels (hardware) nécessaires pour satisfaire les besoins informatiques des utilisateurs.

- **Matériel :** :désigne tout ce qui a un caractère matériel comme : ordinateur, imprimante, souris ....
- **Logiciel :** c'est l'ensemble des programmes qui commandent le matériel.

Information

c'est toute donnée, tout renseignement capable d'apporter une connaissance et d'être représentée a l'aide d'un code.

Information

c'est toute donnée, tout renseignement capable d'apporter une connaissance et d'être représentée a l'aide d'un code.

L'information est un support de connaissance et de communication.

Information

c'est toute donnée, tout renseignement capable d'apporter une connaissance et d'être représentée a l'aide d'un code.

L'information est un support de connaissance et de communication.

Les informations peuvent prendre plusieurs formes : texte, nombre, image, son, vidéo...

Codage de l'information

L'information peut se présenter sous plusieurs formes, elle est soit:

Codage de l'information

L'information peut se présenter sous plusieurs formes, elle est soit:

- **Analogique :** dans ce cas l'information est considérée comme une fonction continue dans le temps

Codage de l'information

L'information peut se présenter sous plusieurs formes, elle est soit:

- **Analogique :** dans ce cas l'information est considérée comme une fonction continue dans le temps
- **Logique:** dans ce cas l'information ne peut prendre que des états stables dits logiques et symbolisés par deux chiffres (ou symboles) : 0 (absence d'information) et 1(présence de l'information).

Codage de l'information

L'information peut se présenter sous plusieurs formes, elle est soit:

- **Analogique :** dans ce cas l'information est considérée comme une fonction continue dans le temps
- **Logique:** dans ce cas l'information ne peut prendre que des états stables dits logiques et symbolisés par deux chiffres (ou symboles) : 0 (absence d'information) et 1(présence de l'information).
- **Numérique:** il arrive que l'information nécessite d'être représentée de façon plus précise et dans ce cas elle est traduite sous cette forme dont la représentation est une suite binaire.

Codage de l'information

L'information peut se présenter sous plusieurs formes, elle est soit:

- **Analogique :** dans ce cas l'information est considérée comme une fonction continue dans le temps
- **Logique:** dans ce cas l'information ne peut prendre que des états stables dits logiques et symbolisés par deux chiffres (ou symboles) : 0 (absence d'information) et 1(présence de l'information).
- **Numérique:** il arrive que l'information nécessite d'être représentée de façon plus précise et dans ce cas elle est traduite sous cette forme dont la représentation est une suite binaire.

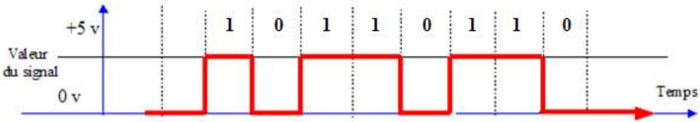
L'information est toujours associée à une grandeur physique (liquide, force, grandeur électrique, pression,. . .)

Sommaire

1. PROGRAMME
2. Introduction générale
3. Codage binaire
4. Système de numération
5. Le codage des nombres
6. Les caractères
7. Le Son et les Images

Codage de l'information

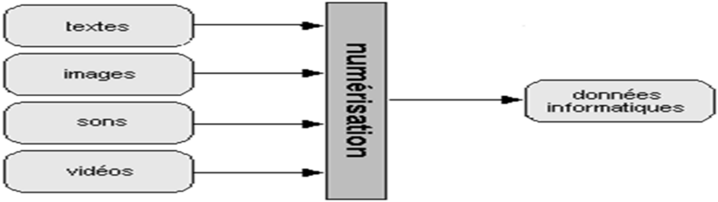
Dans un système informatique, une information est représentée par une suite des 0 et des 1 .Cette forme s'appelle forme binaire



C'est-à-dire que le système informatique ne pourra manipuler, traiter et stocker les informations que sous forme binaire. Ce passage d'une information, d'un langage compréhensible par l'homme, à un langage compréhensible par le système informatique s'appelle codage

Codage de l'information : Définition

Le codage de l'information concerne les moyens de formaliser l'information afin de pouvoir la manipuler, la stocker ou la transmettre. Il ne s'intéresse pas au contenu mais seulement à la forme et à la taille des informations à coder.



La transformation d'un signal analogique en signal numérique est appelée numérisation. La numérisation comporte 3 activités parallèles : l'échantillonnage (en anglais sampling), la quantification et finalement le codage.

OBJECTIF : coder les nombres et les caractères à l'aide de codes binaires.

OBJECTIF : coder les nombres et les caractères à l'aide de codes binaires.

En effet, un ordinateur traite de l'information sous forme numérique binaire.

OBJECTIF : coder les nombres et les caractères à l'aide de codes binaires.

En effet, un ordinateur traite de l'information sous forme numérique binaire.

L'information se caractérise par son contenu (ce qu'elle représente), sa forme (la manière de la formuler), son support (le moyen de la véhiculer).

Le contenu peut concerner des grandeurs numériques (code postal, entier relatif, pixels, etc.), mais aussi des grandeurs analogiques (son, vitesse de rotation, etc.).

On s'intéresse ici à la forme des informations à traiter et à la notion de **codage des nombres et des caractères**.

Un code constitue une correspondance entre des symboles et des objets à désigner.

Un code binaire est une correspondance arbitraire entre un ensemble de **symboles** (0 et 1) et un ensemble d'**objets** (chiffres, lettres, couleurs, etc.).

Norme CEI de 2004 (IEEE 1541)

ATTENTION !

On appelle :

- **bit (b) : bi**nary digi**t**, unité d'information pouvant prendre la valeur 0 ou la valeur 1
- **octet (o) :** un mot binaire de 8 bits.
- **byte (B) :** terme anglo-saxon équivalent à octet

Préfixes binaires (préfixes CEI)

Préfixes binaires (préfixes CEI)		
Nom	Symb.	Puissance de 2
Kibi	Ki	$2^{10} = 1\ 024$
Mébi	Mi	$2^{20} = 1\ 048\ 576$
Gibi	Gi	$2^{30} = 1\ 073\ 741\ 824$
Tébi	Ti	$2^{40} = 1\ 099\ 511\ 627\ 776$
Pébi	Pi	$2^{50} = 1\ 125\ 899\ 906\ 842\ 624$

Préfixes décimaux (préfixes SI)

Préfixes décimaux (préfixes SI)			
Nom	Symb.	Puissance de 10	Err.
Kilo	k	$10^3 = 1\ 000$	2 %
Méga	M	$10^6 = 1\ 000\ 000$	5 %
Giga	G	$10^9 = 1\ 000\ 000\ 000$	7 %
Téra	T	$10^{12} = 1\ 000\ 000\ 000\ 000$	10 %
Péta	P	$10^{15} = 1\ 000\ 000\ 000\ 000\ 000$	13 %

Remarques

Dans ce deuxième tableau, l'erreur indiquée dans la colonne de droite est celle effectuée quand on utilise un préfixe SI à la place d'un préfixe binaire. L'usage de cette norme ne s'est pas encore généralisé.

Remarques

Dans ce deuxième tableau, l'erreur indiquée dans la colonne de droite est celle effectuée quand on utilise un préfixe SI à la place d'un préfixe binaire. L'usage de cette norme ne s'est pas encore généralisé.

Microsoft Windows continue à utiliser à tort les préfixes SI : par exemple 1 ko = 1024 o au lieu de 1000 o. D'autres, comme les systèmes d'exploitation libres de type GNU/Linux, adoptent les préfixes binaires.

Sommaire

1. PROGRAMME
2. Introduction générale
3. Codage binaire
4. Système de numération
   - Base de la numération
   - Changement de bases
5. Le codage des nombres
6. Les caractères
7. Le Son et les Images

Base de la numération

Définition

Une base $B_N = \{C_i\}_{i=1}^{i=N}$ est un système libre de N éléments (chiffres ou caractères) tel que:

$$\forall a \in \mathbb{N}, \exists m \in \mathbb{N} / a = \sum_{j=0}^{m} C_j . N^j \qquad \text{avec } C_j \in \{C_i\}_{i=1}^{N}$$

$$\forall a \in \mathbb{R}^+, \exists \{C_j\}_{j=-\infty}^{\infty} / a = \sum_{j=-\infty}^{+\infty} C_j . N^j \qquad \text{avec } C_j \in \{C_i\}_{i=1}^{N}$$

Les 3 bases les plus utilisées dans les systèmes industriels sont :

Les 3 bases les plus utilisées dans les systèmes industriels sont :

1. Base décimale :

$$B_{10} = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9$$

Les 3 bases les plus utilisées dans les systèmes industriels sont :

1. Base décimale :

$$B_{10} = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9$$

2. Base binaire :

$$B_2 = 0, 1$$

Les chiffres sont alors appelés bit pour **bi**nary digi**t**

Les 3 bases les plus utilisées dans les systèmes industriels sont :

1. Base décimale :

   $$B_{10} = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9$$

2. Base binaire :

   $$B_2 = 0, 1$$

   Les chiffres sont alors appelés bit pour **bi**nary digi**t**

3. Base hexadécimale :

   $$B_{16} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$$

Base décimale B_{10}	Base binaire B_2	Base hexadécimale B_{16}
0	0	0

Base décimale B_{10}	Base binaire B_2	Base hexadécimale B_{16}
0	0	0
1	1	1

Base décimale B_{10}	Base binaire B_2	Base hexadécimale B_{16}
0	0	0
1	1	1
2	10	2

Base décimale B_{10}	Base binaire B_2	Base hexadécimale B_{16}
0	0	0
1	1	1
2	10	2
3	11	3

Base décimale B_{10}	Base binaire B_2	Base hexadécimale B_{16}
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4

Base décimale B_{10}	Base binaire B_2	Base hexadécimale B_{16}
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5

Base décimale B_{10}	Base binaire B_2	Base hexadécimale B_{16}
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6

Base décimale B_{10}	Base binaire B_2	Base hexadécimale B_{16}
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7

Base décimale B_{10}	Base binaire B_2	Base hexadécimale B_{16}
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8

Base décimale B_{10}	Base binaire B_2	Base hexadécimale B_{16}
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9

Base décimale B_{10}	Base binaire B_2	Base hexadécimale B_{16}
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A

Base décimale B_{10}	Base binaire B_2	Base hexadécimale B_{16}
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B

Base décimale B_{10}	Base binaire B_2	Base hexadécimale B_{16}
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C

Base décimale B_{10}	Base binaire B_2	Base hexadécimale B_{16}
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D

Base décimale B_{10}	Base binaire B_2	Base hexadécimale B_{16}
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E

Base décimale B_{10}	Base binaire B_2	Base hexadécimale B_{16}
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Base décimale B_{10}	Base binaire B_2	Base hexadécimale B_{16}
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10

Exemples

Base 10	$(2019)_{10}$	$2.10^3 + 0.10^2 + 1.10^1 + 9.10^0$
Base 2	$(11111100011)_2$	$1.2^{10} + 1.2^9 + 1.2^8 + 1.2^7 + 1.2^6 + 1.2^5$ $+0.2^4 + 0.2^3 + 0.2^2 + 1.2^1 + 1.2^0$
Base Hex	$(7E3)_{16}$	$7.16^2 + 14.16^1 + 3.16^0$

REMARQUE: Pour différencier les bases, on reporte le numéro de la base en indice : $(2019)_{10}$

Passage d'une base B_N à la base décimale B_{10}

En base B_N, chaque chiffre est affecté d'un poids (N^n) avec n le nombre de chiffres à sa droite. Pour obtenir le nombre en base 10, il suffit de faire la somme des chiffres du nombre en base N multipliés par leur poids.

Passage d'une base B_N à la base décimale B_{10}

En base B_N, chaque chiffre est affecté d'un poids (N^n) avec n le nombre de chiffres à sa droite. Pour obtenir le nombre en base 10, il suffit de faire la somme des chiffres du nombre en base N multipliés par leur poids.

0	1	1	0	1	0	0	1
0×2^7	1×2^6	1×2^5	0×2^4	1×2^3	0×2^2	0×2^1	1×2^0

Passage d'une base B_N à la base décimale B_{10}

En base B_N, chaque chiffre est affecté d'un poids (N^n) avec n le nombre de chiffres à sa droite. Pour obtenir le nombre en base 10, il suffit de faire la somme des chiffres du nombre en base N multipliés par leur poids.

0	1	1	0	1	0	0	1

Poids respectifs $\longrightarrow$ 0×2^7 1×2^6 1×2^5 0×2^4 1×2^3 0×2^2 0×2^1 1×2^0

Passage d'une base B_N à la base décimale B_{10}

En base B_N, chaque chiffre est affecté d'un poids (N^n) avec n le nombre de chiffres à sa droite. Pour obtenir le nombre en base 10, il suffit de faire la somme des chiffres du nombre en base N multipliés par leur poids.

Bit de poids fort (MSB)

	0	1	1	0	1	0	0	1
Poids respectifs $\longrightarrow$	0×2^7	1×2^6	1×2^5	0×2^4	1×2^3	0×2^2	0×2^1	1×2^0

Passage d'une base B_N à la base décimale B_{10}

En base B_N, chaque chiffre est affecté d'un poids (N^n) avec n le nombre de chiffres à sa droite. Pour obtenir le nombre en base 10, il suffit de faire la somme des chiffres du nombre en base N multipliés par leur poids.

Bit de poids fort (MSB) — Bit de poids faible (LSB)

0	1	1	0	1	0	0	1

Poids respectifs $\longrightarrow$ 0×2^7 1×2^6 1×2^5 0×2^4 1×2^3 0×2^2 0×2^1 1×2^0

Passage d'une base B_N à la base décimale B_{10}

En base B_N, chaque chiffre est affecté d'un poids (N^n) avec n le nombre de chiffres à sa droite. Pour obtenir le nombre en base 10, il suffit de faire la somme des chiffres du nombre en base N multipliés par leur poids.

Bit de poids fort (MSB)

Bit de poids faible (LSB)

0	1	1	0	1	0	0	1

Poids respectifs $\longrightarrow$ $0 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

Passage d'une base B_N à la base décimale B_{10}

En base B_N, chaque chiffre est affecté d'un poids (N^n) avec n le nombre de chiffres à sa droite. Pour obtenir le nombre en base 10, il suffit de faire la somme des chiffres du nombre en base N multipliés par leur poids.

Bit de poids fort (MSB) | Bit de poids faible (LSB)

0	1	1	0	1	0	0	1

Poids respectifs $\longrightarrow$ $0 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 105$

Passage de la base décimale B_{10} à une base B_N

La passage de la base décimale B_{10} à une base B_N se fait en recherchant la puissance de N immédiatement inférieure au nombre décimal à convertir, puis la puissance de N immédiatement inférieure au reste, et ainsi de suite jusqu'à un reste nul.

$$a = \sum_{j=0}^{m} a_j . N^j$$

En factorisant par N, on obtient :

$$a = N . \left(\sum_{j=0}^{m-1} a_{j+1} . N^j \right) + a_0$$

Le reste de la division de a par N est égale au coefficient a_0. Par division successive par N, on obtient les autres cœfficients associés aux puissances de B_N:

$$a = N.\left(N.\left(\sum_{j=0}^{m-2} a_{j+2}.N^j\right) + a_1\right) + a_0$$

EXEMPLE : $(105)_{10} \Leftrightarrow (?)_2$

$$\begin{array}{r|l} 105 & 2 \\ \hline 1 & 52 \end{array}$$

$$105 = 52 \times 2 + 1$$

EXEMPLE : $(105)_{10} \Leftrightarrow (?)_2$

$$\begin{array}{r|l}
105 & 2 \\ \hline
1 & \begin{array}{r|l} 52 & 2 \\ \hline 0 & 26 \end{array}
\end{array}$$

$$\begin{aligned}
105 &= 52 \times 2 + 1 \\
52 &= 26 \times 2 + 0
\end{aligned}$$

EXEMPLE : $(105)_{10} \Leftrightarrow (?)_2$

$$\begin{array}{rcl} 105 & = & 52 \times 2 + 1 \\ 52 & = & 26 \times 2 + 0 \\ 26 & = & 13 \times 2 + 0 \end{array}$$

$$\begin{array}{r|l} 105 & 2 \\ \hline 1 & 52 \end{array} \quad \begin{array}{r|l} 52 & 2 \\ \hline 0 & 26 \end{array} \quad \begin{array}{r|l} 26 & 2 \\ \hline 0 & 13 \end{array}$$

EXEMPLE : $(105)_{10} \Leftrightarrow (?)_2$

$$
\begin{aligned}
105 &= 52 \times 2 + 1 \\
52 &= 26 \times 2 + 0 \\
26 &= 13 \times 2 + 0 \\
13 &= 6 \times 2 + 1
\end{aligned}
$$

$$
\begin{array}{r|l}
105 & 2 \\ \hline
1 & 52
\end{array}
\quad
\begin{array}{r|l}
52 & 2 \\ \hline
0 & 26
\end{array}
\quad
\begin{array}{r|l}
26 & 2 \\ \hline
0 & 13
\end{array}
\quad
\begin{array}{r|l}
13 & 2 \\ \hline
1 & 6
\end{array}
$$

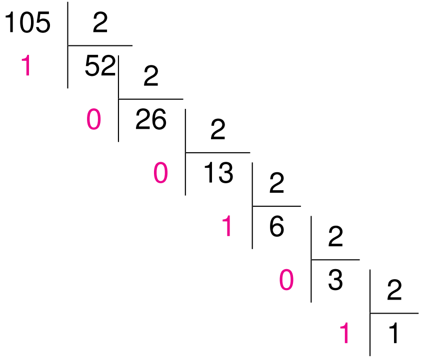
EXEMPLE : $(105)_{10} \Leftrightarrow (?)_2$

$$\begin{array}{rcl}
105 & = & 52 \times 2 + 1 \\
52 & = & 26 \times 2 + 0 \\
26 & = & 13 \times 2 + 0 \\
13 & = & 6 \times 2 + 1 \\
6 & = & 3 \times 2 + 0
\end{array}$$

$$\begin{array}{r|l}
105 & 2 \\ \hline
1 & \begin{array}{r|l}
52 & 2 \\ \hline
0 & \begin{array}{r|l}
26 & 2 \\ \hline
0 & \begin{array}{r|l}
13 & 2 \\ \hline
1 & \begin{array}{r|l}
6 & 2 \\ \hline
0 & 3
\end{array}
\end{array}
\end{array}
\end{array}
\end{array}$$

EXEMPLE : $(105)_{10} \Leftrightarrow (?)_2$

$$\begin{aligned}
105 &= 52 \times 2 + 1 \\
52 &= 26 \times 2 + 0 \\
26 &= 13 \times 2 + 0 \\
13 &= 6 \times 2 + 1 \\
6 &= 3 \times 2 + 0 \\
3 &= 1 \times 2 + 1
\end{aligned}$$



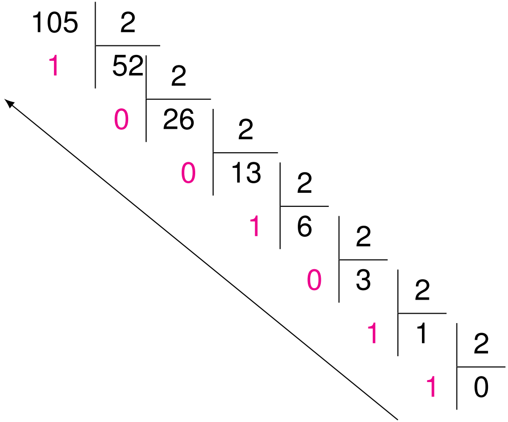
EXEMPLE : $(105)_{10} \Leftrightarrow (?)_2$

$$
\begin{aligned}
105 &= 52 \times 2 + 1 \\
52 &= 26 \times 2 + 0 \\
26 &= 13 \times 2 + 0 \\
13 &= 6 \times 2 + 1 \\
6 &= 3 \times 2 + 0 \\
3 &= 1 \times 2 + 1 \\
1 &= 0 \times 2 + 1
\end{aligned}
$$

$$
\begin{array}{r|l}
105 & 2 \\ \hline
1 & 52 \;|\; 2 \\
0 & 26 \;|\; 2 \\
0 & 13 \;|\; 2 \\
1 & 6 \;|\; 2 \\
0 & 3 \;|\; 2 \\
1 & 1 \;|\; 2 \\
1 & 0
\end{array}
$$

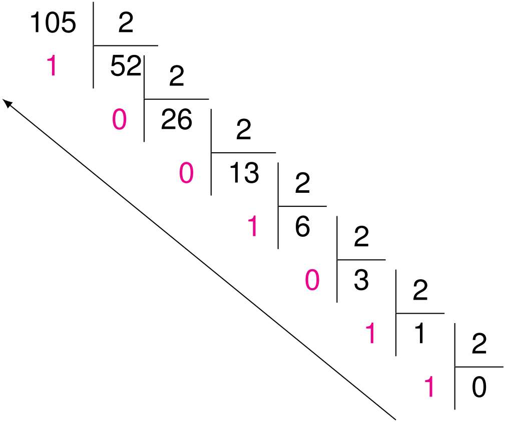
EXEMPLE : $(105)_{10} \Leftrightarrow (?)_2$

$$
\begin{aligned}
105 &= 52 \times 2 + 1\\
52 &= 26 \times 2 + 0\\
26 &= 13 \times 2 + 0\\
13 &= 6 \times 2 + 1\\
6 &= 3 \times 2 + 0\\
3 &= 1 \times 2 + 1\\
1 &= 0 \times 2 + 1
\end{aligned}
$$



EXEMPLE : $(105)_{10} \Leftrightarrow (?)_2$

$$
\begin{aligned}
105 &= 52 \times 2 + 1 \\
52 &= 26 \times 2 + 0 \\
26 &= 13 \times 2 + 0 \\
13 &= 6 \times 2 + 1 \\
6 &= 3 \times 2 + 0 \\
3 &= 1 \times 2 + 1 \\
1 &= 0 \times 2 + 1
\end{aligned}
$$



Ainsi 105 en base 10 s'écrit 1101001 en base 2.

Passage de la base 2^p à la base 2^q

Pour le passage de la base 2^p à la base 2^q, le plus simple est de repasser par la base 2, pour convertir chaque chiffre en son équivalent binaire.

Ainsi chaque chiffre de la base 2^p se code en p chiffres de la base binaire (p bits). En convertissant le nombre binaire par paquets de q chiffres (bits), nous obtenons les chiffres du nombre en base q.

EXEMPLE : $(247)_8 \Leftrightarrow (?)_4$ La base 8 se code sur 3 bits ($8 = 2^3$), et la base 4 se code sur 2 bit ($4 = 2^2$)

$$(A)_8 = 247$$

EXEMPLE : $(247)_8 \Leftrightarrow (?)_4$ La base 8 se code sur 3 bits ($8 = 2^3$), et la base 4 se code sur 2 bit ($4 = 2^2$)

$$\begin{aligned} (A)_8 &= 247 \\ (A)_8 &= \underbrace{2}_{010}\underbrace{4}_{100}\underbrace{7}_{111} \end{aligned}$$

EXEMPLE : $(247)_8 \Leftrightarrow (?)_4$ La base 8 se code sur 3 bits ($8 = 2^3$), et la base 4 se code sur 2 bit ($4 = 2^2$)

$$
\begin{aligned}
(A)_8 &= 247 \\
(A)_8 &= \underbrace{2}_{010}\underbrace{4}_{100}\underbrace{7}_{111} \\
(A)_2 &= 10100111
\end{aligned}
$$

EXEMPLE : $(247)_8 \Leftrightarrow (?)_4$ La base 8 se code sur 3 bits ($8 = 2^3$), et la base 4 se code sur 2 bit ($4 = 2^2$)

$$\begin{aligned}
(A)_8 &= 247 \\
(A)_8 &= \underbrace{2}_{010}\underbrace{4}_{100}\underbrace{7}_{111} \\
(A)_2 &= 10100111 \\
(A)_2 &= \underbrace{10}_{2}\underbrace{10}_{2}\underbrace{01}_{1}\underbrace{11}_{3}
\end{aligned}$$

EXEMPLE : $(247)_8 \Leftrightarrow (?)_4$ La base 8 se code sur 3 bits ($8 = 2^3$), et la base 4 se code sur 2 bit ($4 = 2^2$)

$$
\begin{aligned}
(A)_8 &= 247 \\
(A)_8 &= \underbrace{2}_{010}\underbrace{4}_{100}\underbrace{7}_{111} \\
(A)_2 &= 10100111 \\
(A)_2 &= \underbrace{10}_{2}\underbrace{10}_{2}\underbrace{01}_{1}\underbrace{11}_{3} \\
(A)_4 &= 2213
\end{aligned}
$$

Sommaire

1 PROGRAMME

2 Introduction générale

3 Codage binaire

4 Système de numération

5 Le codage des nombres
- Les nombres entiers naturels
- Les nombres entiers relatifs
- Les nombres rationnels

6 Les caractères

7 Le Son et les Images

Les nombres entiers naturels

DÉFINITION: Codage à champ fixe

Codage sur un nombre de bits n constant (exemple n=8 pour l'octet). La position occupée par le bits (en partant de 0 et de droite), correspond à son poids en puissance de 2.

Il est alors possible de représenter un entier naturel N tel que $0 \leq N \leq 2^n - 1$.

Un entier naturel codé sur un octet est forcément compris entre 0 et 255. Lorsqu'il est codé sur 32 bits, alors il est compris entre 0 et 4294967295.

L'Unité Arithmétique Logique (UAL) du processeur est capable de traiter des mots binaires de 32 bits (ou 64 bits).

Un entier naturel codé sur un octet est forcément compris entre 0 et 255. Lorsqu'il est codé sur 32 bits, alors il est compris entre 0 et 4294967295.

L'Unité Arithmétique Logique (UAL) du processeur est capable de traiter des mots binaires de 32 bits (ou 64 bits).

A la suite d'opérations arithmétiques, des dépassements de capacité (**overflow**) sont possibles. Ils sont en général évités pour les entiers naturels en découpant si nécessaire le mot binaire en plusieurs de taille acceptable par le processeur, le nombre est recomposé par la suite.

Un entier naturel codé sur un octet est forcément compris entre 0 et 255. Lorsqu'il est codé sur 32 bits, alors il est compris entre 0 et 4294967295.

L'Unité Arithmétique Logique (UAL) du processeur est capable de traiter des mots binaires de 32 bits (ou 64 bits).

A la suite d'opérations arithmétiques, des dépassements de capacité (**overflow**) sont possibles. Ils sont en général évités pour les entiers naturels en découpant si nécessaire le mot binaire en plusieurs de taille acceptable par le processeur, le nombre est recomposé par la suite.

Les langages de programmation comme **Python** , peuvent donc traiter des nombres entiers de taille quasiment infinie.

Les nombres entiers relatifs

Signe et valeur absolue (idée 0)

On sacrifie 1 bit pour représenter le signe : $+$ est représenté par `0` et $-$ par `1` .

Les nombres entiers relatifs

Signe et valeur absolue (idée 0)

On sacrifie 1 bit pour représenter le signe : $+$ est représenté par `0` et $-$ par `1` .

On code ainsi avec un mot de n bits les entiers relatifs Z tels que : $-(2^{n-1} - 1) \leq Z \leq 2^{n-1} - 1$.

PROBLÈME:

Les nombres entiers relatifs

Signe et valeur absolue (idée 0)

On sacrifie 1 bit pour représenter le signe : $+$ est représenté par `0` et $-$ par `1` .

On code ainsi avec un mot de n bits les entiers relatifs Z tels que : $-(2^{n-1} - 1) \leq Z \leq 2^{n-1} - 1$.

PROBLÈME:

- on a deux représentations différentes de `0` :

Les nombres entiers relatifs

Signe et valeur absolue (idée 0)

On sacrifie 1 bit pour représenter le signe : + est représenté par `0` et − par `1` .

On code ainsi avec un mot de n bits les entiers relatifs Z tels que : $-(2^{n-1} - 1) \leq Z \leq 2^{n-1} - 1$.

PROBLÈME:

- on a deux représentations différentes de `0` :
  - `00 . . . 0`

Les nombres entiers relatifs

Signe et valeur absolue (idée 0)

On sacrifie 1 bit pour représenter le signe : $+$ est représenté par `0` et $-$ par `1` .

On code ainsi avec un mot de n bits les entiers relatifs Z tels que : $-(2^{n-1}-1) \leq Z \leq 2^{n-1}-1$.

PROBLÈME:

- on a deux représentations différentes de `0` :
  - `00 . . . 0`
  - `10 . . . 0` .

Les nombres entiers relatifs

Signe et valeur absolue (idée 0)

On sacrifie 1 bit pour représenter le signe : $+$ est représenté par `0` et $-$ par `1` .

On code ainsi avec un mot de n bits les entiers relatifs Z tels que : $-(2^{n-1}-1) \leq Z \leq 2^{n-1}-1$.

PROBLÈME:

- on a deux représentations différentes de `0` :
  - `00 . . . 0`
  - `10 . . . 0` .
- Il est difficile d'effectuer des opérations sur les nombres car le bit de signe doit être traité à part.

Complément logique (ou complément à 1 - idée 1)

Dans le cas du complément logique, les nombres positifs sont simplement représentés par leur écriture en base 2. Pour les nombres négatifs, on remplace chaque bit à 0 par 1 et vice versa.

EXEMPLE : Représentation de `−3` sur 4 bits

`3` est représenté par `0011` . Donc en complément à 1, `−3` sera représenté par `1100` .

Pour retrouver le nombre à partir de sa représentation en complément à 1 :

- si le bit de gauche est `0` , le nombre est positif et on a sa représentation en binaire

Pour retrouver le nombre à partir de sa représentation en complément à 1 :

- si le bit de gauche est `0` , le nombre est positif et on a sa représentation en binaire
- si le bit de gauche est `1` , alors le nombre est négatif et pour trouver sa valeur absolue, on inverse les bits.

Pour retrouver le nombre à partir de sa représentation en complément à 1 :

- si le bit de gauche est `0` , le nombre est positif et on a sa représentation en binaire
- si le bit de gauche est `1` , alors le nombre est négatif et pour trouver sa valeur absolue, on inverse les bits.

On code ainsi avec un mot de n bits les entiers relatifs Z tels que : $-(2^{n-1} - 1) \leq Z \leq 2^{n-1} - 1$.

EXEMPLE : `6-3=3` avec `6=0110` et `3=0011` .

Calcul en décimal	Calcul en binaire		
6	0110		0010
Facile, on connaît : - 3	+ 1100	$\Rightarrow$	+ 0001
3	1 0010		0011

PROBLÈME: `0` est codé de deux façons différentes :

- `00 . . . 0`
- `11 . . .1`

Complément à 2 - idée 2

Même méthode que précédemment, sauf qu'il faut encore ajouter 1 au résultat pour la représentation des entiers négatifs. Ainsi :

Complément à 2 - idée 2

Même méthode que précédemment, sauf qu'il faut encore ajouter 1 au résultat pour la représentation des entiers négatifs. Ainsi :

- traduire la valeur absolue du nombre négatifs en binaire. Ainsi : sur 4 bits `3` $\rightarrow$ `0011` .

Complément à 2 - idée 2

Même méthode que précédemment, sauf qu'il faut encore ajouter 1 au résultat pour la représentation des entiers négatifs. Ainsi :

- traduire la valeur absolue du nombre négatifs en binaire. Ainsi : sur 4 bits `3` → `0011` .
- prendre le complément logique de nombre binaire obtenu : `0011` → `1100`.

Complément à 2 - idée 2

Même méthode que précédemment, sauf qu'il faut encore ajouter 1 au résultat pour la représentation des entiers négatifs. Ainsi :

- traduire la valeur absolue du nombre négatifs en binaire. Ainsi : sur 4 bits `3` $\rightarrow$ `0011` .
- prendre le complément logique de nombre binaire obtenu : `0011` $\rightarrow$ `1100`.
- ajouter 1 au nombre complémenté : `1100+0001` $\rightarrow$ `1101` .

Complément à 2 - idée 2

Même méthode que précédemment, sauf qu'il faut encore ajouter 1 au résultat pour la représentation des entiers négatifs. Ainsi :

- traduire la valeur absolue du nombre négatifs en binaire. Ainsi : sur 4 bits `3` → `0011` .
- prendre le complément logique de nombre binaire obtenu : `0011` → `1100`.
- ajouter 1 au nombre complémenté : `1100+0001` → `1101` .
- il est possible alors de faire les opérations.

En décimal	En binaire
6	0110
- 3	+ 1101
3	1 0011

REMARQUE: `0` est codé d'une seule façon : `−0` est représenté par `0000` . En effet, `1111+0001=10000` mais on ne retient que les 4 bits de droite.

Pour retrouver le nombre à partir de sa représentation en complément à 2 :

- si le bit de gauche est `0`, le nombre est positif et on a sa représentation en binaire ;

Pour retrouver le nombre à partir de sa représentation en complément à 2 :

- si le bit de gauche est `0`, le nombre est positif et on a sa représentation en binaire ;
- si le bit de gauche est `1` , alors le nombre est négatif et pour trouver sa valeur absolue, on inverse les bits et on ajoute 1.

Pour retrouver le nombre à partir de sa représentation en complément à 2 :

- si le bit de gauche est `0`, le nombre est positif et on a sa représentation en binaire ;
- si le bit de gauche est `1` , alors le nombre est négatif et pour trouver sa valeur absolue, on inverse les bits et on ajoute 1.

Ainsi avec un mot de n bits, on code les entiers relatifs Z tels que : $-2^{n-1} \leq Z \leq 2^{n-1} - 1$.

Pour retrouver le nombre à partir de sa représentation en complément à 2 :

- si le bit de gauche est `0`, le nombre est positif et on a sa représentation en binaire ;
- si le bit de gauche est `1` , alors le nombre est négatif et pour trouver sa valeur absolue, on inverse les bits et on ajoute 1.

Ainsi avec un mot de n bits, on code les entiers relatifs Z tels que : $-2^{n-1} \leq Z \leq 2^{n-1} - 1$.

REMARQUE: on gagne un nombre négatif par rapport au complément à un.

Les nombres rationnels

Nombres à virgule et bases

On peut ajouter une partie fractionnaire à un nombre dans sa représentation en base N en ajoutant des puissances négatives de N. Les chiffres obtenus seront ajoutés à la suite et séparés par une virgule.

Les nombres rationnels

Nombres à virgule et bases

On peut ajouter une partie fractionnaire à un nombre dans sa représentation en base N en ajoutant des puissances négatives de N. Les chiffres obtenus seront ajoutés à la suite et séparés par une virgule.

EXEMPLE :

Les nombres rationnels

Nombres à virgule et bases

On peut ajouter une partie fractionnaire à un nombre dans sa représentation en base N en ajoutant des puissances négatives de N. Les chiffres obtenus seront ajoutés à la suite et séparés par une virgule.

EXEMPLE :

$(63,5)_{(10)}$ est l'écriture en base 10 de $6.10^1 + 3.10^0 + 5.10^{-1}$.

Il n'y a pas forcément de solution exacte dans le codage de la partie fractionnaire d'un nombre décimal et le nombre de puissances négatives dépend des capacités de la machine et de la précision souhaitée.

On code ainsi avec un mot de n bits dont k bits pour la partie fractionnaire, les réels R tels que :

$$-2^{n-1-k} \leq R \leq 2^{n-1-k} - 2^{-k}$$

Il n'y a pas forcément de solution exacte dans le codage de la partie fractionnaire d'un nombre décimal et le nombre de puissances négatives dépend des capacités de la machine et de la précision souhaitée.

On code ainsi avec un mot de n bits dont k bits pour la partie fractionnaire, les réels R tels que :

$$-2^{n-1-k} \leq R \leq 2^{n-1-k} - 2^{-k}$$

Le codage en virgule fixe (**fixed point**) sur n bits, ne permet de représenter qu'un intervalle de 2^n valeurs. Pour un grand nombre d'applications, cet intervalle est trop restreint. La représentation à virgule flottante a été introduite pour répondre à ce besoin et améliorer la précision des calculs.

Les nombres rationnels

Virgule flottante (floating point)

On peut représenter les nombres sous la forme d'un triplet (s, e, m) : $x = s.m.B^e$, avec B la **base**. Dans ce triplet, s est le signe, e est l'exposant et m est la mantisse. Ce triplet est assemblé dans l'ordre signe, exposant, mantisse pour former un nombre.

Les nombres rationnels

Virgule flottante (floating point)

On peut représenter les nombres sous la forme d'un triplet (s, e, m) : $x = s.m.B^e$, avec B la **base**. Dans ce triplet, s est le signe, e est l'exposant et m est la mantisse. Ce triplet est assemblé dans l'ordre signe, exposant, mantisse pour former un nombre.

La norme IEEE[a] 754 définit alors le codage, avec $B = 2$ (nombres codés en binaire), d'un nombre en simple précision sur 32 bits et en double précision . sur 64 bits

a. Institute of Electrical and Electronics Engineers

Le signe

Le signe $s = \pm 1$ est codé sur 1 bit (MSB le bit de poids le plus fort):

- `0` pour positif
- `1` pour négatif

L'exposant

L'exposant e est un entier codé sur 8 bits en simple précision, 11 en double précision.

L'exposant peut être positif ou négatif. Cependant, la représentation habituelle des nombres signés (complément à 2) rendrait la comparaison entre les nombres flottants un peu plus difficile.

L'exposant

L'exposant e est un entier codé sur 8 bits en simple précision, 11 en double précision.

L'exposant peut être positif ou négatif. Cependant, la représentation habituelle des nombres signés (complément à 2) rendrait la comparaison entre les nombres flottants un peu plus difficile.

Pour régler ce problème, l'exposant est décalé, afin de le stocker sous forme d'un nombre non signé. Ce décalage de $2^{n_e-1}-1$ est appelé biais (n_e représente le nombre de bits de l'exposant) ; il s'agit donc d'une valeur constante une fois que le nombre de bits n_e est fixé.

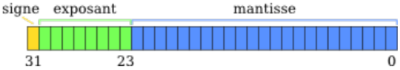
La mantisse

La mantisse m est un nombre à virgule tel que $m \in [1, 2[$, codé sur 23 bits en simple précision, 52 en double précision.

En base binaire, m peut toujours s'écrire $m = 1, \ldots$. Le chiffre 1 n'a pas besoin d'être représenté. Seuls les chiffres après la virgule de la mantisse sont codés en binaire.

IEEE 754 simple précision : sur 32 bits

s : 1 bit ; e : 8 bits ; m : 23 bits



IEEE 754 simple précision : sur 64 bits

s : 1 bit ; e : 11 bits ; m : 52 bits

Exemple

EXEMPLE : : représenter 525,5 au format IEEE 754 simple précision.

- Ecrire 525,5 en base 2 : $(525,5)_{(10)} = (1000001101,1)_{(2)}$

Exemple

EXEMPLE : : représenter 525,5 au format IEEE 754 simple précision.

- Ecrire 525,5 en base 2 : $(525,5)_{(10)} = (1000001101,1)_{(2)}$
- Trouver l'exposant : $(525,5)_{(10)} = (1,0000011011)_{(2)}.2^9$

Exemple

EXEMPLE : : représenter 525,5 au format IEEE 754 simple précision.

- Ecrire 525,5 en base 2 : $(525,5)_{(10)} = (1000001101,1)_{(2)}$
- Trouver l'exposant : $(525,5)_{(10)} = (1,0000011011)_{(2)}.2^9$
- Identifier le triplet (s, e, m) :

Exemple

EXEMPLE : : représenter 525,5 au format IEEE 754 simple précision.

- Ecrire 525,5 en base 2 : $(525,5)_{(10)} = (1000001101,1)_{(2)}$
- Trouver l'exposant : $(525,5)_{(10)} = (1,0000011011)_{(2)}.2^9$
- Identifier le triplet (s, e, m) :
  - $s = 0$ car le nombre est positif

Exemple

EXEMPLE : : représenter 525,5 au format IEEE 754 simple précision.

- Ecrire 525,5 en base 2 : $(525,5)_{(10)} = (1000001101,1)_{(2)}$
- Trouver l'exposant : $(525,5)_{(10)} = (1,0000011011)_{(2)}.2^9$
- Identifier le triplet (s, e, m) :
  - $s = 0$ car le nombre est positif
  - $e_b = e + \underbrace{2^{n_e-1} - 1}_{biais} = 9 + 2^{8-1} - 1 = (136)_{(10)} = (10001000)_{(2)}$ en base 2 sur 8 bits

Exemple

EXEMPLE : : représenter 525,5 au format IEEE 754 simple précision.

- Ecrire 525,5 en base 2 : $(525,5)_{(10)} = (1000001101,1)_{(2)}$
- Trouver l'exposant : $(525,5)_{(10)} = (1,0000011011)_{(2)}.2^9$
- Identifier le triplet (s, e, m) :
  - $s = 0$ car le nombre est positif
  - $e_b = e + \underbrace{2^{n_e-1} - 1}_{biais} = 9 + 2^{8-1} - 1 = (136)_{(10)} = (10001000)_{(2)}$ en base 2 sur 8 bits
  - $m = (00000110110000000000000)_{(2)}$

Exemple

EXEMPLE : : représenter 525,5 au format IEEE 754 simple précision.

- Ecrire 525,5 en base 2 : $(525,5)_{(10)} = (1000001101,1)_{(2)}$
- Trouver l'exposant : $(525,5)_{(10)} = (1,0000011011)_{(2)}.2^9$
- Identifier le triplet (s, e, m) :
  - $s = 0$ car le nombre est positif
  - $e_b = e + \underbrace{2^{n_e-1} - 1}_{biais} = 9 + 2^{8-1} - 1 = (136)_{(10)} = (10001000)_{(2)}$ en base 2 sur 8 bits
  - $m = (00000110110000000000000)_{(2)}$
- Assembler :
  $(525,5)_{(10)} = (0\ 10001000\ 00000110110000000000000)_{(2)}$

Dépassement de capacité et nombres dénormalisés

En arithmétique à virgule flottante, on peut obtenir un résultat valable, ou alors rencontrer un problème de dépassement par valeur supérieure (overflow) lorsque le résultat est trop grand pour pouvoir être représenté, ou par valeur inférieure (underflow) lorsque le résultat est trop petit.

Dépassement de capacité et nombres dénormalisés

En arithmétique à virgule flottante, on peut obtenir un résultat valable, ou alors rencontrer un problème de dépassement par valeur supérieure (overflow) lorsque le résultat est trop grand pour pouvoir être représenté, ou par valeur inférieure (underflow) lorsque le résultat est trop petit.

EXEMPLE : sur 4 bits, que donne 5+4 ?

Certaines représentations sont réservées à la représentation de l'infini ou de codes d'erreurs.

Certaines représentations sont réservées à la représentation de l'infini ou de codes d'erreurs.

Une des parties précédentes décrit comment coder des nombres avec un exposant biaisé e_b strictement compris entre 0 et Val_Max (255 en simple précision et 2047 en double précision). Ainsi, si $e_b \in \{0, Val\_Max\}$, alors on est dans l'un des cas suivants :

Certaines représentations sont réservées à la représentation de l'infini ou de codes d'erreurs.

Une des parties précédentes décrit comment coder des nombres avec un exposant biaisé e_b strictement compris entre 0 et Val_Max (255 en simple précision et 2047 en double précision). Ainsi, si $e_b \in \{0, Val\_Max\}$, alors on est dans l'un des cas suivants :

- $e_b = 0$, $m = 0$: le nombre est ± 0 (représentation dénormalisée)

Certaines représentations sont réservées à la représentation de l'infini ou de codes d'erreurs.

Une des parties précédentes décrit comment coder des nombres avec un exposant biaisé e_b strictement compris entre 0 et Val_Max (255 en simple précision et 2047 en double précision). Ainsi, si $e_b \in \{0, Val\_Max\}$, alors on est dans l'un des cas suivants :

- $e_b = 0$, $m = 0$: le nombre est ± 0 (représentation dénormalisée)
- $e_b = Val\_Max$, $m = 0$, $s = \pm 1$: c'est la représentation de $\pm\infty$

Certaines représentations sont réservées à la représentation de l'infini ou de codes d'erreurs.

Une des parties précédentes décrit comment coder des nombres avec un exposant biaisé e_b strictement compris entre 0 et Val_Max (255 en simple précision et 2047 en double précision). Ainsi, si $e_b \in \{0, Val\_Max\}$, alors on est dans l'un des cas suivants :

- $e_b = 0$, $m = 0$: le nombre est ± 0 (représentation dénormalisée)
- $e_b = Val\_Max$, $m = 0$, $s = \pm 1$: c'est la représentation de $\pm\infty$
- $e_b = Val\_Max$, $m \neq 0$: *NaN* (Not a Number), réservé aux codes d'erreurs (par exemple résultat d'une division par 0)

Certaines représentations sont réservées à la représentation de l'infini ou de codes d'erreurs.

Une des parties précédentes décrit comment coder des nombres avec un exposant biaisé e_b strictement compris entre 0 et Val_Max (255 en simple précision et 2047 en double précision). Ainsi, si $e_b \in \{0, Val\_Max\}$, alors on est dans l'un des cas suivants :

- $e_b = 0$, $m = 0$: le nombre est ± 0 (représentation dénormalisée)
- $e_b = Val\_Max$, $m = 0$, $s = \pm 1$: c'est la représentation de $\pm\infty$
- $e_b = Val\_Max$, $m \neq 0$: NaN (Not a Number), réservé aux codes d'erreurs (par exemple résultat d'une division par 0)
- $e_b = 0$, $m \neq 0$: on dit que la représentation est **dénormalisée** ; dans ce cas, le nombre codé est $x = \pm 0{,}m.B^{(biais-1)}$.

Cette situation arrive lorsqu'un résultat est trop petit pour pouvoir être représenté. Le standard IEEE 754 résout partiellement le problème en autorisant dans ce cas une représentation dénormalisée.

Cette situation arrive lorsqu'un résultat est trop petit pour pouvoir être représenté. Le standard IEEE 754 résout partiellement le problème en autorisant dans ce cas une représentation dénormalisée.

Une représentation dénormalisée est caractérisée par le fait d'avoir un code d'exposant complètement nul, ce qui est interprété comme une indication du fait que le bit de poids fort de la mantisse, implicite, est cette fois à 0 au lieu d'être à 1. De cette façon, le plus petit nombre "exprimable" en simple précision est : $2^{-23} \times 2^{2^{8-1}-1} = 2^{-23-127} = 2^{-150} \approx 10^{-45}$.

Cependant, il faut remarquer que plus le nombre représenté est petit, moins sa mantisse comportera de bits significatifs.

Cependant, il faut remarquer que plus le nombre représenté est petit, moins sa mantisse comportera de bits significatifs.

La norme IEEE spécifie quatre modes d'arrondi :

Cependant, il faut remarquer que plus le nombre représenté est petit, moins sa mantisse comportera de bits significatifs.

La norme IEEE spécifie quatre modes d'arrondi :

- au plus près (si le nombre est entre deux, il est arrondi à la valeur la plus proche avec un bit de poids faible à 0 (mode d'arrondi par défaut).

Cependant, il faut remarquer que plus le nombre représenté est petit, moins sa mantisse comportera de bits significatifs.

La norme IEEE spécifie quatre modes d'arrondi :

- au plus près (si le nombre est entre deux, il est arrondi à la valeur la plus proche avec un bit de poids faible à 0 (mode d'arrondi par défaut).
- vers 0

Cependant, il faut remarquer que plus le nombre représenté est petit, moins sa mantisse comportera de bits significatifs.

La norme IEEE spécifie quatre modes d'arrondi :

- au plus près (si le nombre est entre deux, il est arrondi à la valeur la plus proche avec un bit de poids faible à 0 (mode d'arrondi par défaut).
- vers 0
- vers $+\infty$

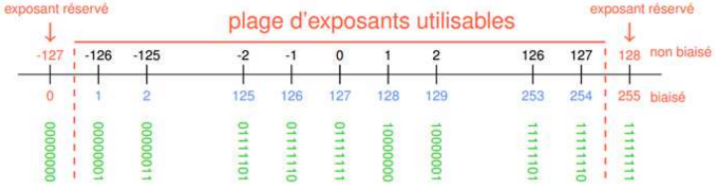
Cependant, il faut remarquer que plus le nombre représenté est petit, moins sa mantisse comportera de bits significatifs.

La norme IEEE spécifie quatre modes d'arrondi :

- au plus près (si le nombre est entre deux, il est arrondi à la valeur la plus proche avec un bit de poids faible à 0 (mode d'arrondi par défaut).
- vers 0
- vers $+\infty$
- vers $-\infty$

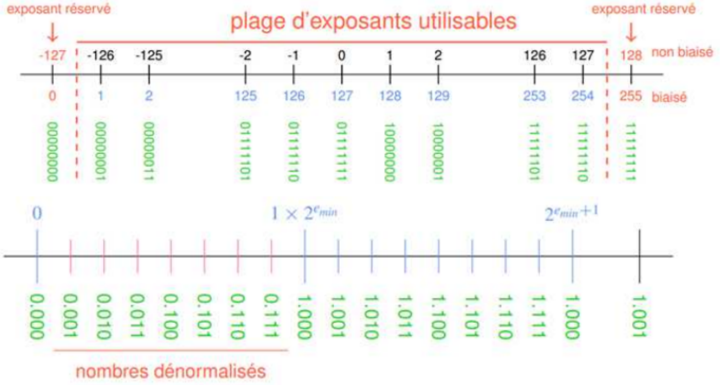
Résumé

En simple précision



Résumé

En simple précision



Résumé

Tableau récapitulatif

	IEEE 754 simple précision	IEEE 754 double précision
Exposant	-126 à 127	-1022 à 1023
Mantisse	1 à presque 2 ($2-2^{-23}$)	1 à presque 2 ($2-2^{-52}$)
Plus petit nombre normalisé	2^{-126}	2^{-1022}
Plus grand nombre normalisé	Presque 2^{128}	Presque 2^{1024}
Plus petit nombre dénormalisé	2^{-149} ($0,000\ldots1\times2^{-126}$)	2^{-1074} ($0,000\ldots1\times2^{-1022}$)
Intervalle utile	Approximativement 10^{-38} à 10^{38}	Approximativement 10^{-308} à 10^{308}

Sommaire

1. PROGRAMME
2. Introduction générale
3. Codage binaire
4. Système de numération
5. Le codage des nombres
6. Les caractères
   - Les caractères alphanumériques
   - Le code ASCII
   - L'Unicode
7. Le Son et les Images

Les caractères alphanumériques

Le nombre de caractères à coder est grands. On estime qu'actuellement, il y a plus de 245000 caractères assignés, 93 écritures différentes.

Les caractères alphanumériques

Le nombre de caractères à coder est grands. On estime qu'actuellement, il y a plus de 245000 caractères assignés, 93 écritures différentes.

Les caractères alphanumériques comprennent les caractères alphabétiques (de A à Z en alphabet latin), ainsi que les caractères numériques comprenant les chiffres 0 à 9 mais aussi les signes – ou $+$.

Les caractères alphanumériques

Le nombre de caractères à coder est grands. On estime qu'actuellement, il y a plus de 245000 caractères assignés, 93 écritures différentes.

Les caractères alphanumériques comprennent les caractères alphabétiques (de A à Z en alphabet latin), ainsi que les caractères numériques comprenant les chiffres 0 à 9 mais aussi les signes – ou $+$.

Il y a aussi bien d'autres caractères : ponctuation, touches clavier (espace, entrée, ...), ...

Le code ASCII

Le code ASCII (American Standard Code for Information Interchange) a été standardisé en 1963.

Le code ASCII

Le code ASCII (American Standard Code for Information Interchange) a été standardisé en 1963.

Il utilise un octet pour encoder 128 caractères : 33 caractères de contrôle, 94 caractères imprimables (lettres, chiffres, . . .), l'espace. Le 8$^{\text{ème}}$ bit reste à 0.

Le code ASCII

Le code ASCII (American Standard Code for Information Interchange) a été standardisé en 1963.

Il utilise un octet pour encoder 128 caractères : 33 caractères de contrôle, 94 caractères imprimables (lettres, chiffres, . . .), l'espace. Le $8^{\text{ème}}$ bit reste à 0.

Beaucoup de pages de codes étendent l'ASCII en utilisant le $8^{\text{ème}}$ bit pour définir des caractères numérotés de 128 à 255. La norme ISO/CEI 8859 fournit des extensions pour diverses langues.

Hex	Dec	Char		Hex	Dec	Char	Hex	Dec	Char	Hex	Dec	Char
0x00	0	NULL	null	0x20	32	Space	0x40	64	@	0x60	96	`
0x01	1	SOH	Start of heading	0x21	33	!	0x41	65	A	0x61	97	a
0x02	2	STX	Start of text	0x22	34	"	0x42	66	B	0x62	98	b
0x03	3	ETX	End of text	0x23	35	#	0x43	67	C	0x63	99	c
0x04	4	EOT	End of transmission	0x24	36	$	0x44	68	D	0x64	100	d
0x05	5	ENQ	Enquiry	0x25	37	%	0x45	69	E	0x65	101	e
0x06	6	ACK	Acknowledge	0x26	38	&	0x46	70	F	0x66	102	f
0x07	7	BELL	Bell	0x27	39	'	0x47	71	G	0x67	103	g
0x08	8	BS	Backspace	0x28	40	(	0x48	72	H	0x68	104	h
0x09	9	TAB	Horizontal tab	0x29	41	)	0x49	73	I	0x69	105	i
0x0A	10	LF	New line	0x2A	42	*	0x4A	74	J	0x6A	106	j
0x0B	11	VT	Vertical tab	0x2B	43	+	0x4B	75	K	0x6B	107	k
0x0C	12	FF	Form Feed	0x2C	44	,	0x4C	76	L	0x6C	108	l
0x0D	13	CR	Carriage return	0x2D	45	-	0x4D	77	M	0x6D	109	m
0x0E	14	SO	Shift out	0x2E	46	.	0x4E	78	N	0x6E	110	n
0x0F	15	SI	Shift in	0x2F	47	/	0x4F	79	O	0x6F	111	o
0x10	16	DLE	Data link escape	0x30	48	0	0x50	80	P	0x70	112	p
0x11	17	DC1	Device control 1	0x31	49	1	0x51	81	Q	0x71	113	q
0x12	18	DC2	Device control 2	0x32	50	2	0x52	82	R	0x72	114	r
0x13	19	DC3	Device control 3	0x33	51	3	0x53	83	S	0x73	115	s
0x14	20	DC4	Device control 4	0x34	52	4	0x54	84	T	0x74	116	t
0x15	21	NAK	Negative ack	0x35	53	5	0x55	85	U	0x75	117	u
0x16	22	SYN	Synchronous idle	0x36	54	6	0x56	86	V	0x76	118	v
0x17	23	ETB	End transmission block	0x37	55	7	0x57	87	W	0x77	119	w
0x18	24	CAN	Cancel	0x38	56	8	0x58	88	X	0x78	120	x
0x19	25	EM	End of medium	0x39	57	9	0x59	89	Y	0x79	121	y
0x1A	26	SUB	Substitute	0x3A	58	:	0x5A	90	Z	0x7A	122	z
0x1B	27	FSC	Escape	0x3B	59	;	0x5B	91	[	0x7B	123	{
0x1C	28	FS	File separator	0x3C	60	<	0x5C	92	\	0x7C	124	|
0x1D	29	GS	Group separator	0x3D	61	=	0x5D	93	]	0x7D	125	}
0x1E	30	RS	Record separator	0x3E	62	>	0x5E	94	^	0x7E	126	~
0x1F	31	US	Unit separator	0x3F	63	?	0x5F	95	_	0x7F	127	DEL

Latin-1

Le code ASCII ne permet d'encoder que l'alphabet latin, sans accents.

Ainsi, par exemple, l'ISO 8859-1, aussi appelée Latin-1, étend l'ASCII avec les caractères accentués utiles aux langues originaires d'Europe occidentale comme le français ou l'allemand. C'est le cas de ce cours, codé Latin-1.

Latin-1

ISO-8859-1

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1x	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2x	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL
8x	PAD	HOP	BPH	NBH	IND	NEL	SSA	ESA	HTS	HTJ	VTS	PLD	PLU	RI	SS2	SS3
9x	DCS	PU1	PU2	STS	CCH	MW	SPA	EPA	SOS	SGCI	SCI	CSI	ST	OSC	PM	APC
Ax	NBSP	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
Bx	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
Cx	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
Dx	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
Ex	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
Fx	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

L'Unicode

La première norme date de 1991, la dernière (6.1) de 2012.

Unicode (Universal Character Set Transformation Format) code les caractères sous forme de séquences de 1 à 4 **codets** d'un octet chacun. La norme Unicode définit entre autres un ensemble (ou répertoire) de caractères. Chaque caractère est repéré dans cet ensemble par un index entier aussi appelé **point de code**. Par exemple le caractère €(euro) est le $8365^{\text{ème}}$ caractère du répertoire Unicode, son index, ou point de code, est donc 8364 (on commence à compter à partir de 0).

Le répertoire Unicode peut contenir plus d'un million de caractères, ce qui est bien trop grand pour être codé par un seul octet. La norme Unicode définit donc des méthodes standardisées pour coder et stocker cet index sous forme de séquence d'octets : UTF-8 est l'une d'entre elles, avec UTF-16, UTF-32 et leurs différentes variantes.

La principale caractéristique d'UTF-8 est qu'elle est rétro-compatible avec la norme ASCII, c'est-à-dire que tout caractère ASCII se code en UTF-8 sous forme d'un unique octet, identique au code ASCII. Par exemple *A* (A majuscule) a pour code ASCII 65 et se code en UTF-8 par l'octet 65. Chaque caractère dont le point de code est supérieur à 127 (caractère non ASCII) se code sur 2 à 4 octets. Le caractère € (euro) se code par exemple sur 3 octets : 226, 130, et 172.

Type	Caractère	Point de code (Hex)	Valeur scalaire		Codage UTF-8	
			Base 10	binaire	binaire	Base 16
Contr.	[NUL]	U+0000	0	*0000000*	**0***0000000*	00
	[US]	U+001F	0	*00*11111	**0***00*11111	1F
Texte	[SP]	U+0020	32	*0*100000	**0***0*100000	20
	A	U+0041	65	1000001	**0**1000001	41
	˜	U+007E	126	1111110	**0**1111110	7E
Contr.	[DEL]	U+007F	127	1111111	**0**1111111	7F
	[PAD]	U+0080	128	*000*10 000000	**110***000*10 **10**000000	C2 80
	[APC]	U+009F	159	*000*10 011111	**110***000*10 **10**011111	C2 9F
Texte	[NBSP]	U+00A0	160	*000*10 100000	**110***000*10 **10**100000	C2 A0
	é	U+00E9	233	*000*10 101001	**110***000*10 **10**101001	C3 A9
	??	U+07FF	2047	11111 111111	**110**11111 **10**111111	DF BF
	?!	U+0800	2048	*0000* 100000 000000	**1110***0000* **10**100000 **10**000000	E0 A0 80
	€	U+20AC	8364	*00*10 000010 101100	**1110***00*10 **10**000010 **10**101100	E2 82 AC
		U+D7FF	55295	1101 011111 111111	**1110**1101 **10**111111 **10**111111	ED 9F BF

Plus d'information sur le codage d'un caractère à l'adresse suivante, en remplaçant xxxx par le code U-xxxx :

http://www.fileformat.info/info/unicode/char/xxxx/index.htm

On y trouve également les commandes ou raccourcis pour obtenir les caractères dans différents environnement :

- Microsoft *Windows*® : Alt+xxxx

Sommaire

1. PROGRAMME
2. Introduction générale
3. Codage binaire
4. Système de numération
5. Le codage des nombres
6. Les caractères
7. Le Son et les Images
   - Le Son
   - Les Images

Codage de l'information : Le Son

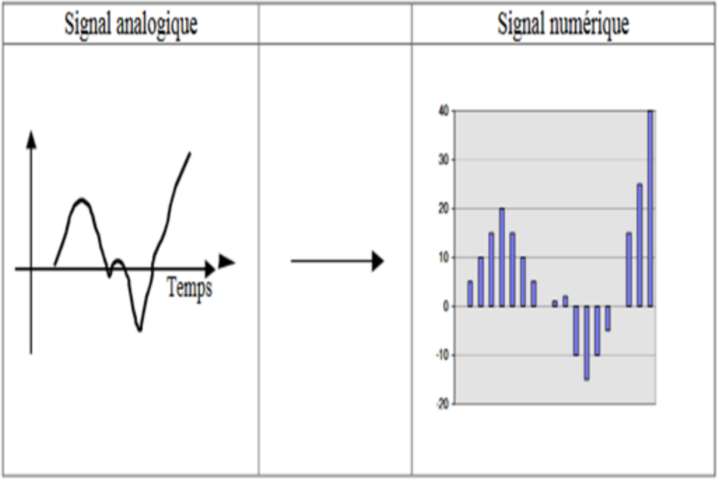
Objectif : on numérise ce type de signal

1-Échantillonnage : Pour pouvoir représenter un son sur un ordinateur, il faut arriver à le convertir en valeurs numériques, car celui - ci ne sait travailler que sur ce type de valeurs. Il s'agit donc de relever des petits échantillons de son (ce qui revient à relever des différences de pression) à des intervalles de temps précis. On appelle cette action l'échantillonnage ou la numérisation du son

2-Quantification : elle consiste à attribuer une valeur à une grandeur physique, prise dans un ensemble fini de valeurs, souvent dans le but de numériser une information analogique pour la traiter par ordinateur.

Codage de l'information : Le Son

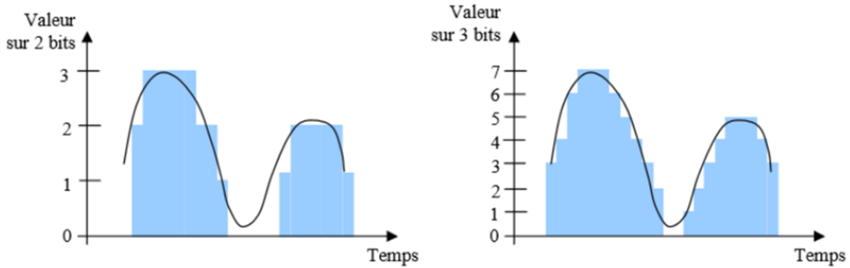
OBJECTIF : on numérise ce type de signal



Codage de l'information : Le Son

OBJECTIF : on numérise ce type de signal

Si on code sur plus de bits, la numérisation s'affine :



Codage de l'information : Les Images

Une image est décomposée en pixels (correspond à un échantillon) et à chaque pixel est associé un code couleur.

La qualité de l'image va donc dépendre :
- Du nombre de pixel contenu dans l'image.
- Du nombre de bits utilisé pour coder la couleur de chaque pixel.

Codage de l'information : Les Images

Le codage RVB (ou RGB) 24bits :

On peut obtenir n'importe quelle couleur en utilisant une combinaison des couleurs rouges vertes et bleu. Le code RVB va coder le niveau d'intensité de chacune de ces couleurs sur un octet (256 niveau d'intensité par couleur). Le premier octet représentant le niveau de rouge, le deuxième octet le niveau de vert et le troisième octet le niveau de bleu.

Quelques couleurs et leur codage RVB :

rouge	255.0.0	vert	0.255.0	bleu	0.0.255
magenta	255.0.255	cyan	0.255.255	jaune	255.255.0
noir	0.0.0	blanc	255.255.255	gris	128.128.128
marron	88.41.0	orange	237.127.16	violet	102.0.153