Séance 3: Mesure de l'information, codage du son, des images et des vidéos

Q1 – Indiquez les assertions correctes :

- ✔ **1 Kilo octet = 1000 octets**
- ❑ 1 Kibi octet = 1000 octets
- ✔ **1 Kibi octet = 1024 octets**
- ❑ 1 Kibi octet = 1024 bits
- ❑ 1 Ko = 1024 octets
- ✔ **1 Ko = 1000 bits**
- ❑ 1 Kibi Octet = 1000 octet
- ✔ **symbole du kibi est « ki »**
- ✔ **symbole du Gibi est « Gi »**
- ✔ **symbole du kilo est « k »**
- ✔ **ko veut dire « kilo octet »**
- ❑ ko veut dire « **kilo bit** »
- ✔ **kilo = 10^3 = 1000**
- ✔ **méga = 10^6 et géga = 10^9**
- ✔ **Téra = 10^{12}**
- ❑ kibi = kilo
- ✔ **kibi = 2^{10} et ~~gibi = 2^{20}~~**
- ❑ Tibi = 2^{30},

Préfixes binaires (préfixes CEI)

Nom	Symbole	2^{10a} = facteur	a
kibi	Ki	2^{10} = 1 024	1
mébi	Mi	2^{20} = 1 048 576	2
gibi	Gi	2^{30} = 1 073 741 824	3
tébi	Ti	2^{40} = 1 099 511 627 776	4
pébi	Pi	2^{50} = 1 125 899 906 842 624	5
exbi	Ei	2^{60} = 1 152 921 504 606 846 976	6
zébi	Zi	2^{70} = 1 180 591 620 717 411 303 424	7
yobi	Yi	2^{80} = 1 208 925 819 614 629 174 706 176	8

Préfixes décimaux (préfixes SI)

Nom	Symbole	10^{3a} = facteur
kilo	k	10^3 = 1 000
méga	M	10^6 = 1 000 000
giga	G	10^9 = 1 000 000 000
téra	T	10^{12} = 1 000 000 000 000
péta	P	10^{15} = 1 000 000 000 000 000
exa	E	10^{18} = 1 000 000 000 000 000 000
zetta	Z	10^{21} = 1 000 000 000 000 000 000 000
yotta	Y	10^{24} = 1 000 000 000 000 000 000 000 000

Q2 - Calculez en bits les valeurs suivantes

- ❑ 10 octets = ? bits
  1 octet = 8 bits
  ce qui donne : 10 octets = 10 x 8 bits = 80 bits

- ❑ 1,5 Méga octets (ou 1,5 Mo) = ? bits
  1 Méga = 10^6
  1 octet = 8 bits
  1 Méga octets = 10^6x 8 bits
  ce qui donne
  1,5 Mo = 1,5 x 10^6 bits

- ❑ 100 Kibi octets (ou 100 KiO) = ? bits
  1 octet = 8 bits et 1 kibi = 2^{10}
  ce qui donne
  100 kibi octets = 100 x 8 x 2^{10} bits
  = 800 x 2^{10} bits

- ❑ 1 Téra-octets (ou 1 To) = ? bits
  1 Téra = 1T = 10^{12}
  1 octet = 8 bits
  1 Téraoctet = 8 x 10^{12} bits

- ❑ 1 Gibi octets (ou 1 GiO) = ? bits
  1 Gibi = 2^{30}
  1 octet = 8 bits
  1 Gibioctets = 8 x 2^{30} octets

Q3 - Le son est par définition

- ✔ **un signal analogique**
- ❑ un signal numérique
- ✔ **est une vibration mécanique d'un fluide (de l'air notamment), qui se propage sous forme d'*ondes***

Q4 – Indiquez les bonnes réponses :

- ✔ **le format « wav » est un format numérique du son non compressé**

- ❑ le format « **mp3** » est un format numérique du son non compressé
- ✔ **le format « ogg » est un format numérique du son compressé**

Oui, il s'agit d'un format audio et vidéo compresse libre,

- ❑ le format « **mp4** » est un format numérique du son,

Faux, il s'agit d'un format vidéo

La compression audio a pour but de réduire la taille d'un flux audio numérique en vue d'une transmission (contraintes de largeur de bande et de débit) ou d'un stockage (contrainte d'espace de stockage). Conséquence : ça réduire la taille des fichier audio.

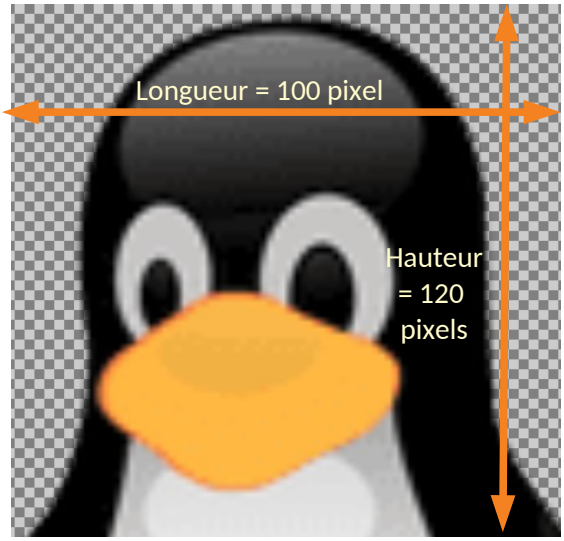
Q5 – Pourquoi on a crée des codages du son, de l'image et de la vidéo **compressés** au lieu de les garder bruts (sans compression) ?

- **C'est pour réduire le volume de stockage des fichiers audio et vidéo.**
- **C'est aussi pour rendre plus fluide la transmission de ces fichiers, notamment sur internet**

Q6 – Pour calculer la définition d'une image, on utiliser deux valeurs :

- ☐ **nombre de pixels sur la longueur (qu'on peut qualifiée aussi de largeur) qui correspond au nombre de colonnes**
- ☐ **nombre de pixel sur la hauteur qui correspond au nombre de lignes**

Exemple



Nous donnera un définition de:
100 pixel en longueur (nombre de colonnes) x 120 pixel en hauteur (nombre de lignes) = 12000 pixels

Q7 – Que veut-on dire par profondeur d'une image :

La profondeur de couleurs d'une image (et par abus de langage profondeur d'une image) décrit le nombre de bits utilisés pour représenter la couleur d'un pixel.

Q8 – Lorsqu'on vous dit qu'une image est codée en RVB que veut dire ces lettres :

- ☐ **R : Rouge**
- ☐ **V : Vert**
- ☐ **B : Bleu**

Q9 – Sur combien de bits code-t-on ma couleur en « ***True color*** » ? Indiquez le nombre de bits par couleur

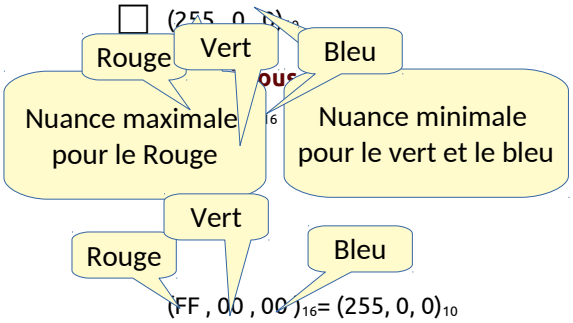
La couleur d'un pixels d'une image « *True color* » est codée au total sur 24 bits avec 8 bits pour chacune des trois couleurs de base : 8 pour le rouge, 8 pour le vert et 8 pour le bleu

Q10 : En supposant que vous codez en ***True color*** (***RVB***), indiquez les couleurs représentées par les codes suivants :

Il faut savoir qu'en « *true color* », on code chaque couleur sur 8 bits.

- **La valeur 0 veut dire aucune nuance de cette couleur (ce qui va vers l'absence totale de cette couleur)**
- **la valeur 255 veut dire nuance maximale, ce qui veut dire qu'on a présence très forte de cette couleur.**

☐ (255, 0, 0)$_{10}$

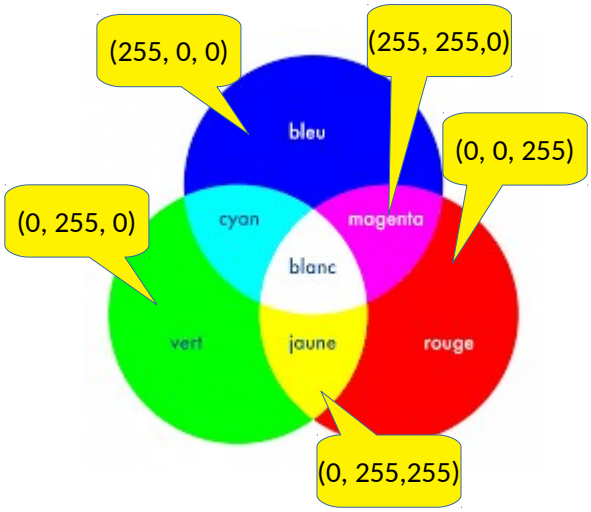


(FF , 00 , 00)$_{16}$= (255, 0, 0)$_{10}$

Ce qui nous donne le rouge

☐ (255, 255, 255)$_{10}$

Nuances maximales pour les trois couleurs Rouge/Vert et Bleu (RVB) ce qui nous donne par juxtaposition additive le blanc.



- ☐ $(0, 255, 0)_{10}$: **Vert foncé**
- ☐ $(0, 0, 255)_{10}$: **Bleu foncé**
- ☐ $(0, 255, 255)_{10}$: **Vert et bleu nous donne du jaune**
- ☐ $(0, 0, 0)_{10}$: **Aucune nuance pour les 3 couleurs nous donne du noir**
- ☐ $(128, 128, 128)_{10}$: **Nuances moyennes pour les 3 couleurs nous donne du gris (moyen ni foncé ni clair))**

Je rappel que lorsque vous avez la même dose (nuance) de couleurs pour les 3 couleurs de base (RVB) vous aurez du gris. Si cette dose (nuance) est proche de 0 alors on se rapprochera du noir (gris très foncé) alors que cette nuance est proche de 255 alors on sa rapprochera du blanc (gris très clair)

Q11 – En supposant que vous avez une définition d'une image 600x400, calculez le poids (capacité mémoire) de cette image si sa profondeur est codée :

- ❑ en « ***True color*» *RVB*** (24 bits) : .....
- ❑ sur 8 bits (256 couleurs) : .....
- ❑ en noir et blanc : .....

La définition de notre image est égale au nombre de ses pixels en colonne fois le nombre de ses pixel en ligne, c'est à dire 600x400 = 240000 pixels :

Je rappel que le poids d'une image est mesuré en calculant le nombre de bits (d'octets) qu'elle occupe en mémoire, ce qui est égale à la définition de l'image fois sa profondeur (nombre de bits codant la couleur du pixel)

- **En « *True color*» *RVB* (24 bits) , Chaque pixel occupera 24 bits. Ce qui donne le poids de cette image égale à :**

  240000 x 24 bits = 5760000 bits

- **Pour une profondeur de couleur de 8 bits, on aura le poids de notre image égale à :**

  240000 x 8 bits = 1920000 bits

- **En noir et blanc, notre image occupera l'espace suivant :**

  240000 x 1 = 240000

Résumé :

Définition de notre image :	**600x400 = 240000 pixels**
Profondeur de couleur	**Poids de l'image**
«true color»: 24 bits	**Poids = 24x240000 bits = 5760000 bits = 5760000 /8 octets = 720000 octets = 720 Ko = 0,72 Mo**
Couleur sur 8 bits	**Poids = 8x240000 bits = 1 920 000 bits = 1920000/8 octets = 240 000 octets = 240 Ko = 0,24 Mo**
Noir et blanc sur 1 bit	**Poids = 1x240000 bits = 240 000 bits = 240000/8 octets = 30000 octets = 30 Ko = 0,03 Mo**

Q12 – On considère qu'on a une animation lorsqu'on fait défiler combien d'image par secondes ?

c'est à partir de 10 images par seconde qu'on commence à tromper la perception visuelle de l'être humain. On peut considérer donc qu'on commence à voir une animation à partir de cette vitesse de défilement d'images

Q13 – Lorsqu'on code une image, il est souhaitable de faire défiler le maximum d'images par secondes au moins 40 images par seconde pour avoir une très bonne qualité de l'animation ?

- ❑ ~~Vrai~~
- ❑ **Faux : C'est tout à fait faux : l'être humain n'est pas capable d'identifier une image qu'on insère parmi d'autre images défilées à plus de 25 images par secondes. Donc il est souvent inutile d'aller au delà de cette vitesse.**

Séance 4 : Codages S+VA, C1 C2

Q14 – Codage des entiers :

Si le codage est en binaire naturel (dit aussi binaire pure ou entier non signé), indiquez l'intervalle des valeurs pouvant être représentées sur n bits.

Donnez L'étendue des valeurs du codage S+VA, C1 et C2:

Codages	Indiquez l'étendue des valeurs si le codage est sur 3 bits (bit de signe compris)
S+VA	$[-(2^{3-1}-1), (2^{3-1}-1)] = [-3, +3]$
C1	$[-(2^{3-1}-1), (2^{3-1}-1)] = [-3, +3]$
C2	$[-2^{3-1}, (2^{3-1}-1)] = [-4, +3]$

Codages	Étendue des valeurs si le codage est sur n bits (bit de signe compris)
S+VA	$[-(2^{n-1}-1), (2^{n-1}-1)]$
C1	$[-(2^{n-1}-1), (2^{n-1}-1)]$
C2	$[-2^{n-1}, (2^{n-1}-1)]$

Q15 – Codage S+VA, C1 et C2 avantage et inconvénients

Complétez le tableau suivant :

	Avantages	Inconvénients
S+VA	Représentation naturelle et simple	Calculs non évidents double représentation du zéro
C1	Représentation des nombres négatifs relativement simple calculs possibles	double représentation du zéro problème de performance du à l'addition de la retenus
C2	Représentation des nombres négatifs plus compliquée calculs possibles	Meilleure performance (on additionne pas la retenue)

Q16 – Donnez sur 8 bits, en S+VA, C1 et C2 le codage des nombres suivants :

$(42)_{10}$	$(0\ 0101010)_{SVA}$
$(42)_{10}$	$(0\ 0101010)_{C1}$
$(42)_{10}$	$(0\ 0101010)_{C2}$
$(-42)_{10}$	$(1\ 0101010)_{SVA}$
$(-42)_{10}$	$(1\ 1010101)_{C1}$
$(-42)_{10}$	$(1\ 1010110)_{C2}$

Q17 – Calcul arithmétique :

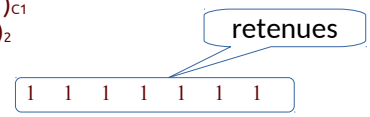
Effectuez les calculs suivants

28+ (–63) en C1 sur 8 bits :

$(28)_{10} = (\mathbf{0}\ 0011100)_2$
$(63)_{10} = (\mathbf{0}\ 0111111)_2$
$(-63)_{10} = (\mathbf{1}\ 1000000)_{C1}$
ce qui donne

	$(28)_{10}$	(**0** 0 0 1 1 1 0 0)$_{C1}$
-	$(63)_{10}$	(**1** 1 0 0 0 0 0 0)$_{C1}$
=	$(-35)_{10}$	(**1** 1 0 1 1 1 0 0)$_{C1}$

63 + (–28) en C1 sur 8 bits :
$(28)_{10} = (\mathbf{0}\ 0011100)_2$
$(-28)_{10} = (\mathbf{1}\ 1100011)_{C1}$
$(63)_{10} = (\mathbf{0}\ 0111111)_2$
ce qui donne

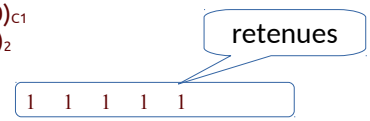


	$+(63)_{10}$	(**0** 0 1 1 1 1 1 1)$_{C1}$
	$-(28)_{10}$	(**1** 1 1 0 0 0 1 1)$_{C1}$
=	$(+35)_{10}$	(**0** 0 1 0 0 0 1 0)$_{C1}$

28+ (–63) en C2 sur 8 bits
$(28)_{10} = (\mathbf{0}\ 0011100)_2$
$(63)_{10} = (\mathbf{0}\ 0111111)_2$
$(-63)_{10} = (\mathbf{1}\ 1000001)_{C1}$
ce qui donne

	$+(28)_{10}$	(**0** 0 0 1 1 1 0 0)$_{C1}$
	$-(63)_{10}$	(**1** 1 0 0 0 0 0 1)$_{C1}$
=	$(-35)_{10}$	(**1** 1 0 1 1 1 0 1)$_{C1}$

63 + (–28) en C2 sur 8 bits
$(28)_{10} = (\mathbf{0}\ 0011100)_2$
$(-28)_{10} = (\mathbf{1}\ 1100100)_{C1}$
$(63)_{10} = (\mathbf{0}\ 0111111)_2$
ce qui donne



	$+(63)_{10}$	(**0** 0 1 1 1 1 1 1)$_{C2}$
	$-(28)_{10}$	(**1** 1 1 0 0 1 0 0)$_{C2}$
=	$(+35)_{10}$	(**0** 0 1 0 0 0 1 1)$_{C2}$

63 + 96 en C2 sur 7 bits

Les valeurs que l'on peut représenter en C2 sur 7 bits sont dans l'intervalle :

$[-2^{7-1}, (2^{7-1}-1)]= [-2^6, (2^6-1)]=[-64, +64]$

On voit bien que la valeur 63 est dans cet intervalle, mais ce n'est pas le cas pour la valeur 96. On déduit qu'il est impossible de faire ce calcul en utilisant uniquement 7 bits. On doit rajouter un bit supplémentaire au moins pour faire ce calcul. Faisons le sur 8 bits

			1	1					
	$(63)_{10}$	(	**0**	0	1	1	1	1	1	$)_{C2}$
+	$(96)_{10}$	(	**0**	1	1	0	0	0	0	$)_{C2}$
=	$+(159)_{10}$	(	**1**	0	0	1	1	1	1	$)_{C2}$

On voit que sur 8 bits le résultats obtenu est négatif ce qui témoigne d'un débordement de capacité car on avait additionné 2 nombre positifs.

L'intervalle des valeurs représentables sur 8 bits en C2 est : $[-2^{8-1}, (2^{8-1}-1)]= [-2^7, (2^7-1)]=[-128, +127]$

On voit bien que la valeur obtenue (qui est 159) est en dehors de cet intervalle et c'est la raison pour laquelle nous avons obtenue la situation de débordement de capacité lors du calcul de 63 + 96.

Séance 5 : Codages des réels (virgule fixe, virgule flottante)

Q18 – Codage en virgule fixe :

Sur **9 bits** dont un bit de signe et 5 bits pour la partie entière et 3 bits pour la partie décimale, donnez les représentations des nombres suivants :

$(12,4)_{10}$	(……)$_{S+VA}$
$(-12,4)_{10}$	(……)$_{S+VA}$
$(AC,3)_{16}$	(……)$_{S+VA}$

Réponse :

$(12,4)_{10}$ = (?)$_{S-VA}$

Calculons la partie entière en binaire : $12 = (1100)_2$

Calculons la partie décimale en binaire :

0,4 x 2 = 0, 8 donc $0,4 = (0,0 \ldots..)_2$

0, 8x2 = 1,6 donc $0,4 = (0,01 \ldots..)_2$

0, 6x2 = 1,2 donc $0,4 = (0,011 \ldots..)_2$

0, 2x2 = 0,4 donc $0,4 = (0,0110 \ldots..)_2$

0, 4x2 = 1,6 ici on revient à 0,4 donc nous auront une série de chiffres (0110) qui va se répéter indéfiniment

donc $0,4 = (0, \overline{0110})_2$

Représentons maintenant notre nombre sur 9 bits comprenant un bit de signe, 5 bits pour la partie entière et 3 bits pour la partie décimale :

$(12,4)_{10}$ = (**0** | 0 1 1 0 0 , 0 1 1)$_{SVA}$

Signe positif — Partie entière sur 5 bits — Partie décimale sur 3 bits

Remarquez que la partie décimale du nombre binaire (complément à 2) que nous avons trouvé comporte une infinité de chiffres. Cela veut dire que les 3 bits de la partie décimale, dans notre représentation nous oblige à accepter une approximation de notre nombre.

$(-12,4)_{10}$ = (?)$_{S-VA}$

Lors du calcul précédent nous avons calculé (12,4) en C2 :

$(12,4)_{10}$ = (**0** | 0 1 1 0 0 , 0 1 1)$_{SVA}$

Pour trouver $(-12,4)_{10}$, il suffit d'inverser le bit de signe de la valeur précédente ce qui donne :

$(-12,4)_{10}$ = (**1** | 0 1 1 0 0 , 0 1 1)$_{SVA}$

$(AC,3)_{16}$ = (?)$_{S-VA}$

Si on se réfère à ce que nous avons appris lors que chapitre sur les systèmes de numération, notamment la conversion de la base 16 vers la base 2 on trouveras ceci :

$(AC,3)_{16}$ **= $(10101100, 0011)_2$**

Comme vous le constatez, le nombre trouvé nécessite au moins 8 bits pour la partie entière et 4 pour la partie décimale. On déduit donc que l'on ne pourra pas représenter ce nombre en S+VA sur 9 bits dont 5 pour la partie entière et 3 pour la partie décimale. Il faut au moins prévoir :

- 1 bits de signe
- 8 bits pour la partie entière
- et 4 bits pour la partie décimale

donc au total 1 + 8 + 4 = 13 bits ce qui nous donne ceci :

(**0** | 1 0 1 0 1 1 0 0 , 0 0 1 1)$_{SVA}$

+ — A — C — 3

Signe positif — Partie entière sur 8 bits — Partie décimale sur 4 bits

$(12,4)_{10}$	(……..)$_{C1}$
$(-12,4)_{10}$	(……..)$_{C1}$
$(AC,3)_{16}$	(……..)$_{C1}$

Réponse :

$(12,4)_{10}$ = (**0** 0 1 1 0 0 , 0 1 1)$_{C1}$

$(-12,4)_{10}$ = (**1** 1 0 0 1 1 , 1 0 0)$_{C1}$

Pour coder $(AC,3)_{16}$, c'est comme précédemment (pour la représentation en S+VA) 9 bits ne suffisent. Il nous fallait au moins 13 bits. D'ailleurs, étant donné qu'il s'agit d'un nombre positif, nous avons exactement le même codage en S+VA, en C1 et en C2 :

$AC,3)_{16}$=

(**0** 1 0 1 0 1 1 0 0 , 0 0 1 1)$_{C1}$

Signe positif — Partie entière sur 8 bits — Partie décimale sur 4 bits

$(12,4)_{10}$	(……..)$_{C2}$
$(-12,4)_{10}$	(……..)$_{C2}$
$(AC,3)_{16}$	(……..)$_{C2}$

Réponse :

$(12,4)_{10}$ = (**0** 0 1 1 0 0 , 0 1 1)$_{C2}$

$(-12,4)_{10}$ = (**1** 1 0 0 1 1 , 1 0 1)$_{C2}$

Pour coder $(AC,3)_{16}$, c'est comme précédemment (pour la représentation en S+VA) 9 bits ne suffisent. Il nous fallait au moins 13 bits. D'ailleurs, étant donné qu'il s'agit d'un nombre positif, nous avons exactement le même codage en S+VA, en C1 et en C2 :

$AC,3)_{16}$=

(**0** 1 0 1 0 1 1 0 0 , 0 0 1 1)$_{C2}$

Signe positif — Partie entière sur 8 bits — Partie décimale sur 4 bits

Q19 – Codage en virgule fixe (calcul):

Sur **9 bits** dont un bit de signe et 5 bits pour la partie entière et 3 bits pour la partie décimale, en utilisant la représentation en C2, faire le calcul suivants : **$(12,4)_{10}$ + $(-25,25)_{10}$**

Réponse : Vous faites le calcul en faisant abstraction de la virgule :

La valeur (12,4)10 nous l'avons déjà trouvé en C2 (voir question précédente :

$(12,4)_{10}$ (**0** 0 1 1 0 0 , 0 1 1)$_{C2}$

Cherchons maintenant le codage de $(-25,25)_{10}$ en C2 :

$(25,25)_{10}$ **0** 1 1 0 0 1 , 0 1 0)$_{C2}$

On déduit donc $(-25,25)_{10}$

$(-25,25)_{10}$ **1** 0 0 1 1 0 , 1 1 0)$_{C2}$

Posons notre calcul avec les codages trouvés :

			1	1		1		1			
$(12,4)_{10}$	(	**0**	0	1	1	0	0	,	0	1	1)$_{C2}$
$(-25,25)_{10}$	(	**1**	0	0	1	1	0	,	1	1	0)$_{C2}$
–12,85	(	**1**	1	0	0	1	1		0	0	1)$_{C2}$