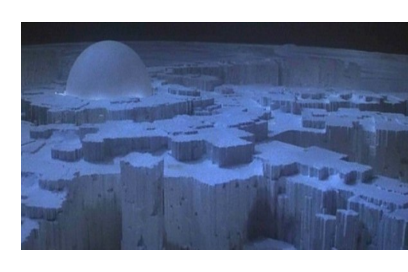

Présentation :

Un robot envoyé sur la planète Krypton à la recherche de Kryptonite doit envoyer vers la terre des informations sous forme de données numériques par ondes radio.

Ces données, du type « nombres à virgule flottante », sont codées selon le principe de la norme IEEE754, mais sur un format de 16 bits, soit deux octets (dans le but d'économiser de la bande passante par rapport au type officiel « flottant simple précision» sur 32 bits).

Dans tout ce document, les couleurs suivantes seront utilisées :

vert	:	hexadécimal
rouge	:	binaire
bleu	:	entier décimal
noir	:	flottant décimal

Le format retenu est :

- 1 bit de **signe**
- 4 bit d'**exposant biaisé**
- 11 bits de **mantisse**

Signe	Exposant biaisé				Mantisse										
S	E_3	E_2	E_1	E_0	M_{10}	M_9	M_8	M_7	M_6	M_5	M_4	M_3	M_2	M_1	M_0
Octet 1								Octet2							

Par exemple(*), le code 9A40 (en hexadécimal), soit 1001101001000000 (en binaire), représentera le nombre -0,080078125 (en décimal).

(*) : La démonstration de cet exemple sera donnée ci-après.

Calcul du biais

Démontrez (à partir de la documentation de la norme IEEE754) que le biais est égal à **7** :

biais = $2^{4-1} - 1 = 2^3 - 1 = 8 - 1 = \mathbf{7}$ car l'exposant biaisé est sur 4 bits

Convertir en décimal les flottants normalisés suivants :

Nombre 1 : 0x9A40 (exemple)

En binaire :

Signe	Exposant biaisé				Mantisse										
1	0	0	1	1	0	1	0	0	1	0	0	0	0	0	0
Octet 1								Octet2							

Signe : ☐ positif ☒ négatif

Exposant débiaisé en décimal : $2^{3-7} = 2^{-4}$

Mantisse normalisée : $1,01001000000 = 2^0 + 2^{-2} + 2^{-5}$

Résultat : $-(2^0 + 2^{-2} + 2^{-5}) \times 2^{-4} = \mathbf{-0,080078125}$

Nombre 2 : 0x2E00 (à compléter)

En binaire :

Signe	Exposant biaisé				Mantisse										
0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0
Octet 1								Octet2							

Signe : ☒ positif ☐ négatif

Exposant débiaisé en décimal : $2^{5-7} = 2^{-2}$

Mantisse normalisée : $1,11000000000 = 2^0 + 2^{-1} + 2^{-2}$

Résultat : $(2^0 + 2^{-1} + 2^{-2}) \times 2^{-2} = \mathbf{0,4375}$

Nombre 3 : 0xF000 (à compléter)

En binaire :

Signe	Exposant biaisé				Mantisse										
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
Octet 1								Octet2							

Signe : ☐ positif ☒ négatif

Exposant débiaisé en décimal : $2^{14-7} = 2^7$

Mantisse normalisée : $1,00000000000 = 2^0$

Résultat : $-(2^0) \times 2^7 = \mathbf{-128}$

Nombre 4 : 0xDC01 (à compléter)

En binaire :

Signe	Exposant biaisé				Mantisse										
1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	1
Octet 1								Octet2							

Signe : ☐ positif ☒ négatif

Exposant débiaisé en décimal : $2^{11-7} = 2^4$

Mantisse normalisée : $1,10000000001 = 2^0 + 2^{-1} + 2^{-11}$

Résultat : $(2^0 + 2^{-1} + 2^{-11}) \times 2^4 = \mathbf{-24.0078125}$

Nombre 5 : 0x0D00 (à compléter)

En binaire :

Signe	Exposant biaisé				Mantisse										
0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0
Octet 1								Octet2							

Signe : ☒ positif ☐ négatif

Exposant débiaisé en décimal : $2^{1-7} = 2^{-6}$

Mantisse normalisée : $1,10100000000 = 2^0 + 2^{-1} + 2^{-3}$

Résultat : $(2^0 + 2^{-1} + 2^{-3}) \times 2^{-6} = \mathbf{0,025390625}$

Nombre 6 : 0x60C4 (à compléter)

En binaire :

Signe	Exposant biaisé				Mantisse										
0	1	1	0	0	0	0	0	1	1	0	0	0	1	0	0
Octet 1								Octet2							

Signe : ☒ positif ☐ négatif

Exposant débiaisé en décimal : $2^{12-7} = 2^5$

Mantisse normalisée : $1,00011000100 = 2^0 + 2^{-4} + 2^{-5} + 2^{-9}$

Résultat : $(2^0 + 2^{-4} + 2^{-5} + 2^{-9}) \times 2^5 = \mathbf{35,0625}$

Questions diverses :

Donnez les deux manières de coder le nombre zéro (en hexadécimal) :

+0 ⇨ 0x0000

-0 ⇨ 0x8000

Donnez le code (en hexadécimal) de +2 (décimal) :

+2 ⇨ 0x4000

Donnez les codes des infinis :

+∞ ⇨ 0x7800

-∞ ⇨ 0xF800

À quoi correspondent ces trois codes : 0x7F00 0xF83A 0xFFFF ?

Réponse :

Ce sont des codes représentant des « nan » (not a number)

Quel est le plus petit nombre positif « codable » (hormis zéro) ?

- En hexadécimal : 0001
- En décimal : 7,62939453125e-06
- C'est un nombre : ☐ normalisé ☒ dénormalisé

Quel est le plus grand nombre positif « codable » ?

- En hexadécimal : 77FF
- En décimal : 255,9375

C'est un nombre : ☒ normalisé ☐ dénormalisé

Progression des nombres en virgule flottante :

	N1		N2		N2 - N1	N2 / N1
	hexa	flottant	hexa	flottant	flottant	flottant
Petits	1800	0,0625	1801	0,062530517578125	≃ 0,000031	≃ 1,00049
Moyens	3A00	1,25	3A01	1,25048828125	≃ 0,00049	≃ 1,00039
Grands	7100	144	7101	144,0625	≃ 0,063	≃ 1,00043

Dans le tableau ci-dessus , on a calculé la différence et le produit de deux flottants « successifs », pour des nombres petits, moyens et grands.

Qu'observe-t-on ?

N2-N1 augmente fortement avec la valeur des flottants, tandis que N2/N1 est à peu près constant

La **suite** des flottants codés avec cette méthode est-elle :

☐ approximativement arithmétique ☒ approximativement géométrique ☐ autre

Dans quel(s) cas doit-on utiliser des **flottants** plutôt que des **entiers** (aidez-vous du graphique ci-dessous) ?

On utilise les flottants lorsque l'on a besoin d'une précision (ou incertitude) relative constante, autant sur les petits nombres que sur les grands.

Les entiers ont, quant à eux, une précision (ou incertitude) absolue constante (d'une unité).