Nombre et ordinateur

1 Rappel sur la base 2

8 bits donne 2^8 combinaisons

$2^8 = 256$: 0 (0) … 255 (1111 1111)

Donner le nombre de valeurs et les deux limites valeurs minimale et maximale avec un nombre de 4 bits de long, puis de 10 bits, puis de 12 bits.

Écrire les 16 premiers chiffres en base 16 sur 4 bits de long.

2 Les préfixes binaires normalisés (source Wikipédia)

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

A connaître :

$2^8 = 256$ $2^{10} = 1024$ $2^{12} = 4096$

3 Les formats de données numériques dans un ordinateur

Les données numériques dans un ordinateur sont disponibles avec plusieurs types possibles. Après observation du tableau ci-dessous représentant les types disponibles en langage c répondre aux questions suivantes :

- C'est quoi un type ?

- Qu'est-ce qui différencie un type dans la mémoire de l'ordinateur ?

Type de donnée	Signification	Taille (en octets)	Plage de valeurs acceptée
char	Caractère	1	-128 à 127
unsigned char	Caractère non signé	1	0 à 255
short int	Entier court	2	-32 768 à 32 767
unsigned short int	Entier court non signé	2	0 à 65 535
int	Entier	2 (sur processeur 16 bits) 4 (sur processeur 32 bits)	-32 768 à 32 767 -2 147 483 648 à 2 147 483 647
unsigned int	Entier non signé	2 (sur processeur 16 bits) 4 (sur processeur 32 bits)	0 à 65 535 0 à 4 294 967 295
long int	Entier long	4	-2 147 483 648 à 2 147 483 647
unsigned long int	Entier long non signé	4	0 à 4 294 967 295
float	Flottant (réel)	4	$3.4*10^{-38}$ à $3.4*10^{38}$
double	Flottant double	8	$1.7*10^{-308}$ à $1.7*10^{308}$
long double	Flottant double long	10	$3.4*10^{-4932}$ à $3.4*10^{4932}$

4 La représentation des nombres entiers relatifs

Cette représentation est réalisée en prenant le complément à 2 ou complément vrai (CV) du nombre entier positif dont on souhaite connaitre la représentation négative.

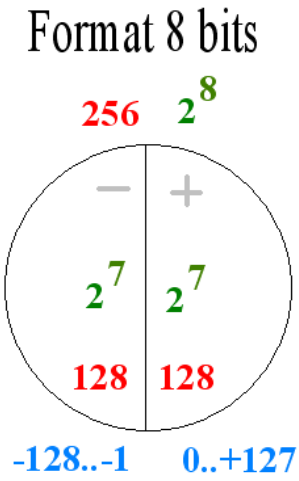
ATTENTION avec un format déterminé on IGNORE tout ce qui dépasse du format.

➪ Dans cette représentation le bit de poids fort égal à 1 indique un nombre négatif.

➪ Avec cette représentation soustraire un nombre entier revient à additionner sa représentation en complément vrai.

➪ Quand on est en présence d'un nombre négatif pour connaitre sa valeur absolue il suffit de calculer le complément vrai de ce nombre.

Exemple pour un format de 8 bits pour déterminer le nombre de combinaisons possibles en positif et négatif :



- Donner les valeurs possibles pour un format de 4 bits.
- Donner les valeurs possibles pour un format de 16 bits.

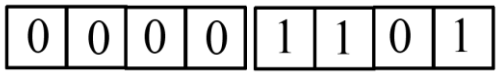
Détermination du complément à 2 d'un nombre

Par exemple -13 sur un format de 8 bits. (Le codage disponible va de -128 à +127)

-13 = CV(13)

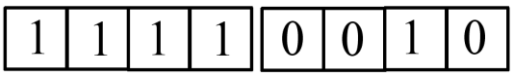
Étapes du calcul à suivre impérativement

1. Codage de 13 dans le format imposé.

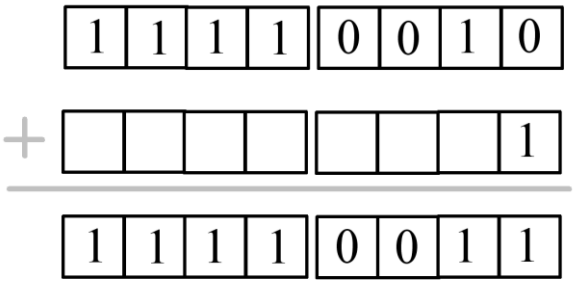


Puis calcul du complément vrai CV de 13

2. Calculer le complément restreint (CR) ou complément à 1



3. Ajouter 1 et ignorer tout ce qui dépasse le format



Donc la représentation de -13 dans un format de 8 bits est égale à **1111 0011 soit $F3**

Exercices

- Déterminer le codage de +35 et -35 format 8 bits
- Déterminer le codage de -98 et -9 format 8 bits
- Calculer en format 8 bits 35 - 9 et -35 - (-98)
- Déterminer en format 8 bits ce que représente 1110 0010

5 Les types principaux dans trois langages de programmation

Type de données	Taille en octets	Plage de valeurs acceptée	c	pascal	python
Entier sur un octet	1	-128 +127	char	Shortint	NON
Entier positif sur un octet	1	0 255	unsigned char	Byte	Voir chaine de caractère
Entier court (16 bits)	2	-32768 +32767	shortint	Integer	NON
Entier long (32 bits)	4	-2147483648 +2147483647	Long int	Longint	OUI
Réel sur 32 bits	4	$3.4\ 10^{-38}$ $-3.4\ 10^{38}$	float	Single	NON
Réel sur 64 bits	8	$1.7\ 10^{-308}$ $-1.7\ 10^{308}$	double	Double	OUI

Le codage du format réel fera l'objet d'une fiche spécialisée.

6 Attention aux formats binaires dans les ordinateurs[1]

Comment sont écrites les données dans les mémoires des ordinateurs, ce qui peut rendre des échanges de données problématiques entre ordinateurs travaillant dans des représentations différentes dans le cas de l'échange de format binaire. (Dans ce cas l'échange en ASCII est préférable).

BIG ENDIAN

Quand certains ordinateurs enregistrent un entier sur 32 bits en mémoire, par exemple `0xA0B70708` en notation hexadécimale, ils l'enregistrent dans des octets dans l'ordre qui suit : `A0 B7 07 08`, pour une structure de mémoire fondée sur une unité atomique de 1 octet et un incrément d'adresse de 1 octet. Ainsi, l'octet de poids le plus fort (ici `A0`) est enregistré à l'adresse mémoire la plus petite, l'octet de poids inférieur (ici `B7`) est enregistré à l'adresse mémoire suivante et ainsi de suite.

	0	1	2	3	
...	A0	B7	07	08	...

Pour une structure de mémoire ou un protocole de communication fondé sur une unité atomique de 2 octets, avec un incrément d'adresse de 1 octet, l'enregistrement dans des octets sera `A0B7 0708`. L'unité atomique de poids le plus fort (ici `A0B7`) est enregistrée à l'adresse mémoire la plus petite.

	0	1	2	3	
...	A0	B7	07	08	...

Les architectures qui respectent cette règle sont dites ***big-endian*** ou **gros-boutistes** ou **mot de poids fort en tête**, par exemple les processeurs Motorola 68000, les SPARC (Sun Microsystems) ou encore les System/370 (IBM).

De plus, tous les protocoles TCP/IP communiquent en *big-endian*[9]. Il en va de même pour le protocole PCI Express.

LITTLE ENDIAN

Les autres ordinateurs enregistrent `0xA0B70708` dans l'ordre suivant : `08 07 B7 A0` (pour une structure de mémoire fondée sur une unité atomique de 1 octet et d'un incrément d'adresse de 1 octet), c'est-à-dire avec l'octet de poids le plus faible en premier. De telles architectures sont dites ***little-endian*** ou **petit-boutistes** ou **mot de poids faible en tête**. Par exemple, les processeurs x86 ont une architecture petit-boutiste. Celle-ci, au prix d'une moindre lisibilité du code machine par le programmeur, simplifiait la circuiterie de décodage d'adresses courtes et longues en 1975 [réf. souhaitée], quand un 8086 avait 29 000 transistors. Elle est d'influence pratiquement nulle en 2016, aussi bien en temps d'exécution (masqué) que d'encombrement (un Haswell typique comporte 1,4 *milliard* de transistors).

	0	1	2	3	
...	08	07	B7	A0	...

Pour une structure de mémoire ou un protocole de communication fondé sur une unité atomique de 2 octets, avec un incrément d'adresse de 1 octet, l'enregistrement dans des octets sera `0708 A0B7`. L'unité atomique de poids le plus faible (ici `0708`) est enregistré à l'adresse mémoire la plus petite.

	0	1	2	3	
...	07	08	A0	B7	...

BI-ENDIAN

[1] Source Wikipédia