Conversions des Entiers

non-signés & signés

Ce document a pour objectif de vous familiariser avec les conversions entre plusieurs bases dans le cas des entiers non-signés et signés.

La plupart des conversions que nous effectuerons seront entre les bases 2, 8, 10, et 16.

Nous verrons que plusieurs notations existent pour représenter les bases. Sans symbole particulier, on considère qu'il s'agit de la base 10 usuelle.

1 Représentation des entiers

Avant de convertir, particulièrement dans des cas concrets, il est nécessaire de connaître une valeur importante : la *taille des mots* manipulés par le processeur. En effet, la taille des bus d'adresse et de données sont fondamentales pour savoir quelles valeurs maximum et minimum sont représentables. Typiquement, un processeur 8 bits ne pourra représenter que 256 valeurs.
Pour comprendre cela, il faut simplement se remémorer les états possibles pour chaque bit, et garder en tête qu'un fil transmet un bit. Les tableaux suivants vous montre tous les états possibles que peuvent prendre les valeurs sur des bus de 1 bit, 2, bits, 3 bits, et 4 bits.

	états	total
1 bit	0 1	2 états
2 bits	0 0 0 1 1 0 1 1	4 états
3 bits	0 0 0 0 0 1 0 1 0 0 1 1 1 0 0 1 0 1 1 1 0 1 1 1	8 états

		total
4 bits	0 0 0 0 0 0 0 1 0 0 1 0 0 0 1 1 0 1 0 0 0 1 0 1 0 1 1 0 0 1 1 1 1 0 0 0 1 0 0 1 1 0 1 0 1 0 1 1 1 1 0 0 1 1 0 1 1 1 1 0 1 1 1 1	16 états

Vous remarquez intuitivement qu'il s'agit de puissances de 2 : $2^1 = 2$, $2^2 = 4$, $2^3 = 8$, $2^4 = 16$. On peut donc déduire le nombre d'états possibles (donc le nombre de valeurs distinctes représentables) avec la formule suivante, où N représente la taille des mots :

$$\textit{Nombre d'états possibles} = 2^N$$

Une fois la taille des mots fixée, on peut en déduire les valeurs minimales et maximales représentables. Il existe néanmoins une distinction fondamentale à connaître en architecture : les entiers signés et entiers non-signés.

— Entiers signés : les entiers sur lesquels le signe est interprété (+ ou −)

— Entiers non signés : les entiers dont le signe n'est pas utilisé (strictement positifs)

On ne fait que manipuler des valeurs où chaque bit sera à 0 ou 1, il s'agit donc d'interpréter chacun des états pour qu'il représente une valeur entière. Chaque « bit » (ou fil/pin du processeur) prendra une puissance de 2, exactement comme pour les dizaines en base 10 :

$531 = 5 \times 10^2 + 3 \times 10^1 + 1 \times 10^0$
$1010 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$

Afin de facilement convertir de la base 2 vers la base 10, vous devez connaître par cœur les 10 premières puissances de 10, voire les 13 premières.

2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}	2^{11}	2^{12}	2^{13}
1	2	4	8	16	32	64	128	256	512	1024	2048	4096	8192

1.1 Entiers non-signés

L'interprétation brut des bits correspond aux entiers non-signés. Ainsi, sur 8 bits, les entiers non-signés correspondent à :

(base 2)	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	*(base 10)*
	128	64	32	16	8	4	2	1	
0000 0000	0	0	0	0	0	0	0	0	0
0000 0001	0	0	0	0	0	0	0	1	1
0000 0010	0	0	0	0	0	0	1	0	2
0000 0011	0	0	0	0	0	0	1	1	3
0000 0100	0	0	0	0	0	1	0	0	4
0000 0101	0	0	0	0	0	1	0	1	5
0000 0110	0	0	0	0	0	1	1	0	6
0000 0111	0	0	0	0	0	1	1	1	7
0000 1000	0	0	0	0	1	0	0	0	8
...		...				...			...
0010 0000	0	0	1	0	0	0	0	0	32
0010 0001	0	0	1	0	0	0	0	1	33
...		...				...			...
0110 0000	0	1	1	0	0	0	0	0	96
0110 0001	0	1	1	0	0	0	0	1	97
0110 0010	0	1	1	0	0	0	1	0	98
0110 0011	0	1	1	0	0	0	1	1	99
0110 0100	0	1	1	0	0	1	0	0	100
...		...				...			...
1000 0000	1	0	0	0	0	0	0	0	128
...		...				...			...
1111 1110	1	1	1	1	1	1	1	0	254
1111 1111	1	1	1	1	1	1	1	1	255

Vous vous rendez compte que représenter 2^8 (256) valeurs dans le cas non-signé correspond à représenter 0 et ses 255 successeurs. Ainsi, pour des mots de taille N, les entiers non-signés auront comme borne minimale 0, et comme borne maximale $2^N - 1$.

1.2 Entiers signés

Les entiers signés sont simplement une autre interprétation des données où un bit sert à représenter le signe de l'entier (positif ou négatif). Une convention est majoritairement utilisée pour représenter les entiers signés : le premier bit sert à coder le signe (0 correspond à un nombre positif, et 1 à un nombre négatif), et on applique le complément à 2 dans le cas négatif. Ainsi, les entiers signés positifs sont les mêmes que dans le cas des entiers non-signés, mais, il faut d'abord appliquer le complément à 2 à un entier signé négatif pour pouvoir retrouver la valeur absolue binaire.

Appliquer le complément à 2 se résume simplement à appliquer le complément à 1 sur le nombre étudié, puis d'y ajouter 1. Le complément à 1 est l'opération visant à intervertir les 0 par des 1, et les 1 par des 0.
Par exemple, pour appliquer le complément à 2 sur la valeur 1001 :

1 0 0 1

(complément à 1)

0 1 1 0

(ajout de 1)

0 1 1 1

1001 correspond donc au négatif de la valeur absolue binaire 0111 (7), c'est-à-dire à -7.

Ainsi, sur 8 bits, les entiers signés correspondent à :

(base 2)	$+/-$	2^6	2^5	2^4	2^3	2^2	2^1	2^0	*non-signé*	*signé*
	$+/-$	64	32	16	8	4	2	1		
1000 0000	**1**	0	0	0	0	0	0	0	128	-128
1000 0001	**1**	0	0	0	0	0	0	1	129	-127
1000 0010	**1**	0	0	0	0	0	1	0	130	-126
1000 0011	**1**	0	0	0	0	0	1	1	131	-125
...		...				...			...	...
1001 1001	**1**	0	0	1	1	0	0	1	153	-103
1001 1010	**1**	0	0	1	1	0	1	0	154	-102
...		...				...			...	...
1111 1101	**1**	1	1	1	1	1	0	1	253	-3
1111 1110	**1**	1	1	1	1	1	1	0	254	-2
1111 1111	**1**	1	1	1	1	1	1	1	255	-1
0000 0000	**0**	0	0	0	0	0	0	0	0	0
0000 0001	**0**	0	0	0	0	0	0	1	1	1
0000 0010	**0**	0	0	0	0	0	1	0	2	2
0000 0011	**0**	0	0	0	0	0	1	1	3	3
...		...				...			...	...
0001 1001	**0**	0	0	1	1	0	0	1	25	25
0001 1010	**0**	0	0	1	1	0	1	0	26	26
...		...				...			...	...
0111 1100	**0**	1	1	1	1	1	0	0	124	124
0111 1101	**0**	1	1	1	1	1	0	1	125	125
0111 1110	**0**	1	1	1	1	1	1	0	126	126
0111 1111	**0**	1	1	1	1	1	1	1	127	127

Vous vous rendez compte que représenter 2^8 (256) valeurs dans le cas signé correspond à représenter 0 et ses 127 successeurs, ainsi que ses 128 prédécesseurs. Ainsi, pour des mots de taille N, les entiers signés auront comme borne minimale -2^{N-1}, et comme borne maximale $2^{N-1} - 1$.

Pour conclure, chaque valeur binaire peut être interprétée de différentes manières.
Il est extrêmement important de bien comprendre que *l'interprétation* des valeurs va changer les résultats de certaines opérations !
Comparer sur 8 bits 255 et 0 ne donnera pas la même chose si l'on considère les entiers comme signés ou non-signés (0 sera le plus petit en mode non-signé, à l'inverse, ±128 sera considéré comme le plus petit en mode signé sur 8 bits).

1.3 Notations des bases

Plusieurs notations pour représenter les bases existent : celles où l'on explicite par un indice en suffixe indiquant dans quelle base celui-ci a été écrit, ou par un caractère en préfixe du nombre.

— $42_{(10)}$ indique l'on écrit le nombre « 42 » en base 10.
— $0110_{(2)}$ indique que l'on écrit le nombre « 6 » en base 2, c'est-à-dire 0110.

Parmi les caractères servant de préfixe, on retrouvera : % 0o $

— « %00101111 » le pourcentage indique que l'on manipule un nombre binaire (ici 47)
— « 0o42 » le zéro suivi d'un O minuscule indiquent que l'on manipule un nombre octal (ici 34)
— « $15AB » le dollar indique l'on manipule un nombre hexadécimal (ici 5547)
— Un nombre sans préfixe est considéré par défaut comme un nombre décimal.

2 Conversions base 2

2.1 Base 2 vers 10

La conversion de la base 2 non-signée vers la base 10 implique simplement d'utiliser la décomposition en puissances de 2. On lit chaque chiffre, et on le multiplie par la puissance de 2 associée.

$$\begin{aligned}
\%\,1010\,0110 &= 1\times2^7 + 0\times2^6 + 1\times2^5 + 0\times2^4 + 0\times2^3 + 1\times2^2 + 1\times2^1 + 0\times2^0 \\
&= 1\times128 + 0\times64 + 1\times32 + 0\times16 + 0\times8 + 1\times4 + 1\times2 + 0\times1 \\
&= 128 + 0 + 32 + 0 + 0 + 4 + 2 + 0 \\
&= 166
\end{aligned}$$

Ainsi, dans le cas d'un entier non-signé, %1010 0110 correspond à 166.

Pour les entiers signés, il suffit de regarder le bit de poids fort (c'est-à-dire le chiffre portant la plus grande valeur, autrement dit, le chiffre le plus à gauche) pour constater qu'il s'agit d'un 0 (nombre positif) ou d'un 1 (nombre négatif). Si le nombre commence par un 0, et est donc positif, il suffit d'appliquer l'algorithme pour convertir les nombres non-signés. Si le nombre commence par un 1, et est donc négatif, il faut donc retirer le bit de poids fort, puis appliquer le complément à 2 avant de convertir avec l'algorithme classique de conversion.

%**1**010 0110
(nombre négatif)
% 010 0110
(complément à 1)
% 101 1001
(ajout de 1)

$$\begin{aligned}
\%\ 101\ 1010 &= 1\times2^6 + 0\times2^5 + 1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0 \\
&= 1\times64 + 0\times32 + 1\times16 + 1\times8 + 0\times4 + 1\times2 + 0\times1 \\
&= 64 + 0 + 16 + 8 + 0 + 2 + 0 \\
&= 90
\end{aligned}$$

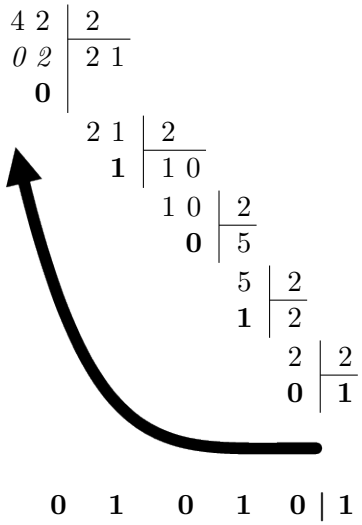
(négatif)

$$-90$$

Ainsi, dans le cas d'un entier signé, %1010 0110 correspond à −90.

2.2 Base 10 vers 2

Pour convertir de la base 10 vers la base 2 (également appelée notation binaire), il suffit d'effectuer des divisions successives jusqu'à obtenir 1. On reporte ensuite chaque reste, ainsi que le dernier quotient, en bas des divisions successives. Et enfin, on inverse l'ordre de lecture des chiffres reportés plus bas.



Ce qui donne :

0 1 0 1 0 1

On inverse ensuite l'ordre de lecture en prenant les chiffres depuis la droite (on peut directement lire les résultats des divisions depuis le dernier quotient jusqu'au premier reste en suivant la flèche) :

1 0 1 0 1 0

Et on obtient ainsi 42 en binaire, c'est-à-dire : %101010

Pour convertir les nombres négatifs, on applique le même algorithme de conversion vers la base 2, puis on calcule le complément à 2 (c'est-à-dire faire le complément à 1, puis ajouter 1 au résultat).

Par exemple pour convertir -5 en base 2 sur 8 bits, on calcule tout d'abord l'équivalent de 5 en binaire :

0 0 0 0 0 1 0 1

Puis on calcule son complément à 1 (vous remarquerez que le bit codant le signe passe de 0 à 1, indiquant donc maintenant un nombre négatif) :

1 1 1 1 1 0 1 0

Et enfin on lui ajoute 1 :

1 1 1 1 1 0 1 1

Et on obtient ainsi -5 en binaire sur 8 bits, c'est-à-dire : % 1111 1011

Pour convertir un nombre négatif en binaire, vous pouvez chercher son ordre de grandeur en base 2 pour déduire combien de chiffres seront nécessaires pour le représenter. Ensuite, il suffit d'étendre le nombre avec des 1 devant si la taille des entiers est plus grande (pour les entiers positifs ou non signés, il suffit d'étendre le nombre avec des 0 devant, comme en base 10).

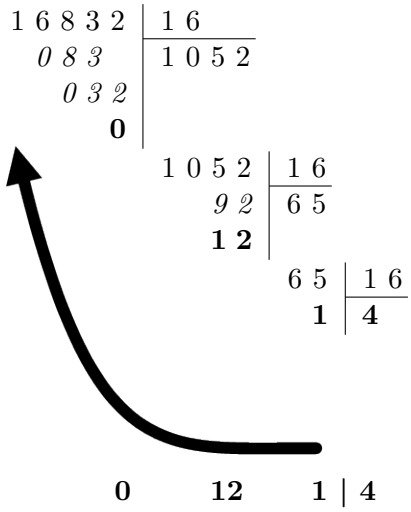
3 Conversions base 16

La base 16 (également appelée notation hexadécimale), contrairement à la base 2, étend le nombre de symboles pouvant représenter des données. Là où la base 2 ne s'appuie que sur deux symboles/deux états possibles, la base 16 s'appuie sur seize symboles/seize états possibles. Vous devez connaître ces 16 symboles et leurs équivalences en base 10.

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

3.1 Base 10 vers 16

Pour convertir de la base 10 vers la base 16, il suffit d'effectuer des divisions successives jusqu'à obtenir 1. On reporte ensuite chaque reste, ainsi que le dernier quotient, en bas des divisions successives. Et enfin, on inverse l'ordre de lecture des chiffres reportés plus bas. La différence avec les divisions par 2 est que lorsque l'on obtient un reste ou un quotient supérieur ou égal à 10, alors on utilise les symboles supplémentaires pour représenter ces valeurs



Ce qui donne :

0 12 1 4

Puis avec la conversion hexadécimale :

0 C 1 4

On inverse ensuite l'ordre de lecture en prenant les chiffres depuis la droite (on pourrait directement lire les résultats des divisions depuis le dernier quotient jusqu'au premier reste, comme l'indique la flèche) :

4 1 C 0

Et on obtient ainsi 16832 en hexadécimal, c'est-à-dire : $41C0

Vous constatez que l'algorithme de conversion fonctionne pour plusieurs bases, y compris plus grandes et plus petites que le décimal. En réalité, cette méthode est l'exacte inverse de la décomposition en puissance de 2 ou 10 (on cherche à obtenir le reste de chaque puissance).

3.2 Base 2 vers 16

Parmi les conversions, vous devez également parfaitement savoir convertir les données binaires en hexadécimal (et inversement). La méthode de conversion est extrêmement simple, beaucoup plus que les précédentes, car les éléments de la base 16 sont des multiples de la base 2.

Pour convertir des valeurs binaires en valeurs hexadécimales, il suffit simplement de prendre des paquets de 4 bits, et les convertir en un symbole associé. En effet, sur 4 bits, on peut représenter 16 états, c'est-à-dire que l'on peut utiliser les symboles hexadécimaux pour les écrire.

$$\%\,1010\;0110 \quad = \quad 10\;\;6 \quad = \quad \$\,\mathrm{A6}$$

On peut directement résumer les valeurs ainsi :

base 16	*base 2*
0	0000
1	0001
2	0010
3	0011

base 16	*base 2*
4	0100
5	0101
6	0110
7	0111

base 16	*base 2*
8	1000
9	1001
A	1010
B	1011

base 16	*base 2*
C	1100
D	1101
E	1110
F	1111

Ainsi, que l'entier interprété soit signé ou non-signé, il suffit simplement de transformer les paquets de 4 bits.

3.3 Base 16 vers 2

La conversion dans l'autre sens fonctionne strictement de la même manière : on prend chaque caractère hexadécimal, et on le transforme en son équivalent binaire en conservant l'ordre des chiffres.

$$\begin{array}{lcccccl}
\$\,\mathrm{BA10} & = & 11 & 10 & 1 & 0 & = \%\,1011\;1010\;0001\;0000 \\
\$\,\mathrm{ABCD} & = & 10 & 11 & 12 & 13 & = \%\,1010\;1011\;1100\;1101 \\
\$\,\mathrm{DEAD} & = & 13 & 14 & 10 & 13 & = \%\,1101\;1110\;1010\;1101 \\
\$\,\mathrm{BEEF} & = & 11 & 14 & 14 & 15 & = \%\,1011\;1110\;1110\;1111
\end{array}$$

3.4 Base 16 vers 10

Pour convertir de la base 16 vers la base 10 dans le cas d'un entier non-signé, il suffit de multiplier chaque valeur par la puissance de 16 associée.

$$\begin{array}{lcl}
\$\,\mathrm{AD42} & = & A \times 16^3 \;+\; D \times 16^2 \;+\; 4 \times 16^1 \;+\; 2 \times 16^0 \\
& = & 10 \times 4096 \;+\; 14 \times 256 \;+\; 4 \times 16 \;+\; 2 \times 1 \\
& = & 40960 \;+\; 3584 \;+\; 64 \;+\; 2 \\
& = & 44610
\end{array}$$

Ainsi, dans le cas d'un entier non-signé, $\$\,\mathrm{AD42}$ correspond à 44610.

Pour convertir un nombre signé, il faut d'abord tester le bit de poids fort pour vérifier s'il est à 0 ou 1. Vous pouvez visuellement savoir cela en regardant la valeur la plus à gauche : si cette valeur est supérieure ou égale à 8, alors le bit de poids fort sera à 1, donc le nombre sera négatif.
Dans tous les cas, vous devrez effectuer un complément à 2 sur la valeur binaire. Pour cette raison, dans le cas des entiers signés dont la valeur la plus à gauche est supérieure ou égale à 8, il vaut mieux convertir le nombre hexadécimal en binaire pour pouvoir faire toutes les opérations suivantes et en déduire le nombre décimal négatif.

4 Conversions base 8

La base 8 (également appelée notation octale) est rarement utilisée, mais il arrive qu'elle le soit. Vous devez donc savoir convertir la base 8 pour ces quelques situations. Néanmoins, nous nous contenterons des quelques cas faciles.

Comme vous l'avez compris, la base 8 est constituée de chiffres allant de 0 à 7. Ainsi, 0o 10 correspond à 8, et 0o 21 à 17.

4.1 Base 10 vers 8

Pour convertir de la base 10 vers la base 8, on effectue des divisions successives par 8 jusqu'à obtenir 1.
Nous ne montrerons pas d'exemple étant donné que vous maîtrisez maintenant la technique.

4.2 Base 8 vers 10

Pour convertir de la base 8 vers la base 10, il suffit de multiplier par les puissances de 8.

$$\begin{aligned} 0o\,4321 &= 4 \times 8^3 + 3 \times 8^2 + 2 \times 8^1 + 1 \times 8^0 \\ &= 4 \times 512 + 3 \times 64 + 2 \times 16 + 1 \times 1 \\ &= 2048 + 192 + 32 + 1 \\ &= 2273 \end{aligned}$$

4.3 Base 2 vers 8

Convertir de la base 2 vers de l'octal est extrêmement simple, car la technique est proche de celle de l'hexadécimal. Au lieu de prendre des paquets de 4 bits (représentants 16 états), il suffit de prendre des paquets de 3 bits (représentants 8 états).

$$\%\,0111\ 1010\ 0110 = \%\,011\ 110\ 100\ 110 = 3\ 6\ 4\ 6 = 0o\,3646$$

Ainsi, % 0111 1010 0110 correspond à 0o 3646, c'est-à-dire 1958 en décimal.

4.4 Base 8 vers 2

On peut très facilement convertir la base 8 vers la base 2 en exécutant la même démarche précédente inversée : on remplace chaque chiffre par son équivalent binaire.

$$\begin{aligned} 0o\,1337 &= 1\ 3\ 3\ 7 = \%\,001\ 011\ 011\ 111 = \%\,0010\ 1101\ 1111 \\ 0o\,4321 &= 4\ 3\ 2\ 1 = \%\,100\ 011\ 010\ 001 = \%\,1000\ 1101\ 0001 \\ 0o\,7065 &= 7\ 0\ 6\ 5 = \%\,111\ 000\ 110\ 101 = \%\,1110\ 0011\ 0101 \end{aligned}$$

Ce document et ses illustrations ont été réalisés par Fabrice BOISSIER en novembre 2022
(dernière mise à jour en septembre 2023)