2023

9. Binaire

R2 : Guide SCRAPY

Numéro de projet: **2021-1-FR01-KA220-SCH-000031617**

Co-funded by
the European Union

ECAM EPMI

30/04/2023

Table des matières

Ce projet a été financé avec le soutien de la Commission européenne. Cette publication [communication] reflète uniquement le point de vue de son auteur et la Commission ne peut être tenue responsable de toute utilisation qui pourrait être faite des informations qui y sont contenues.

1. Introduction

Les systèmes numériques sont les méthodes que nous utilisons pour représenter les nombres. Depuis l'école primaire, nous opérons tous principalement dans les limites confortables d'un système numérique en base 10, mais il en existe bien d'autres. Base-2, base-8, base-16, base-20, base... vous comprenez. Il existe une variété infinie de systèmes de nombres de base, mais seuls quelques-uns sont particulièrement importants pour le génie électrique.

Les systèmes numériques les plus populaires ont même leur nom. La base 10, par exemple, est communément appelée système de nombres décimaux. Base-2, dont nous sommes ici pour parler aujourd'hui, porte également le surnom de binaire. Un autre système numérique populaire, la base 16, est appelé hexadécimal.

La base d'un nombre est souvent représentée par un entier en indice après une valeur. Ainsi, dans l'introduction ci-dessus, la première image ferait 10010 quelque chose tandis que la deuxième image ferait 1002 quelque chose. C'est un moyen pratique de spécifier la base d'un nombre lorsqu'il existe un risque d'ambiguïté.

2. Pourquoi le binaire ?

Pourquoi binaire demandez-vous ? Eh bien, pourquoi décimal ? Nous utilisons les décimales depuis toujours et tenons pour acquis la raison pour laquelle nous avons opté pour le système numérique en base 10 pour nos besoins numériques quotidiens. C'est parce que nous avons 10 doigts, ou c'est simplement parce que les Romains l'ont imposé à leurs anciens soumis. Quelle que soit la raison qui y a conduit, les astuces que nous avons apprises en cours de route ont solidifié la place de la base 10 dans nos cœurs ; tout le monde sait compter par dizaines. Nous arrondissons même les grands nombres au multiple de 10 le plus proche. Nous sommes obsédés par 10 !

Les ordinateurs et l'électronique sont limités dans le département des doigts et des orteils. Au niveau le plus bas, ils n'ont que deux façons de représenter l'état de quoi que ce soit : ON ou OFF, haut ou bas, 1 ou 0. Ainsi, tous les appareils électroniques s'appuient sur un système numérique en base 2 pour stocker, manipuler et calculer les nombres. .

La forte dépendance de l'électronique à l'égard des nombres binaires signifie qu'il est important de savoir comment fonctionne le système numérique en base 2. Vous rencontrerez couramment le binaire, ou ses cousins, comme l'hexadécimal, dans tous les programmes informatiques. L'analyse des circuits logiques numériques et autres composants électroniques de très bas niveau nécessite également une utilisation intensive du binaire.

Dans cette leçon, vous découvrirez que tout ce que vous pouvez faire avec un nombre décimal peut également être fait avec un nombre binaire. Certaines opérations peuvent être encore plus faciles à effectuer sur un nombre binaire (bien que d'autres puissent être plus pénibles). Nous couvrirons tout cela et bien plus encore dans cette leçon.

Ce projet a été financé avec le soutien de la Commission européenne. Cette publication [communication] reflète uniquement le point de vue de son auteur et la Commission ne peut être tenue responsable de toute utilisation qui pourrait être faite des informations qui y sont contenues.

3 Comptage et conversion

La base de chaque système numérique est également appelée base. La base d'un nombre décimal est dix et la base d'un nombre binaire est deux. La base détermine le nombre de symboles différents nécessaires pour étoffer un système numérique. Dans notre système de nombres décimaux, nous avons 10 représentations numériques pour des valeurs comprises entre rien et dix : 0, 1, 2, 3, 4, 5, 6, 7, 8 et 9. Chacun de ces symboles représente un élément très spécifique. , valeur standardisée.

En binaire, nous n'avons droit qu'à deux symboles : 0 et 1. Mais en utilisant ces deux symboles, nous pouvons créer n'importe quel nombre possible avec un système décimal.

3.1 Compter en binaire

Vous pouvez compter en décimales à l'infini, même pendant votre sommeil, mais comment compteriez-vous en binaire ? Zéro et un en base deux devraient sembler assez familiers : 0 et 1. À partir de là, les choses deviennent résolument binaires.

N'oubliez pas que nous n'avons que ces deux chiffres, donc comme nous le faisons en décimal lorsque nous manquons de symboles, nous devons décaler une colonne vers la gauche, ajouter un 1 et tourner tous les chiffres vers la droite pour 0. Donc, après 1, nous obtenons 10, puis 11, puis 100. Commençons à compter...

Décimal	Binaire	...	Décimal	Binaire
0	0		16	10000
1	1		17	10001
2	dix		18	10010
3	11		19	10011
4	100		20	10100
5	101		21	10101
6	110		22	10110
7	111		23	10111
8	1000		24	11000
9	1001		25	11001
dix	1010		26	11010
11	1011		27	11011
12	1100		28	11100
13	1101		29	11101
14	1110		30	11110
15	1111		31	11111

Ce projet a été financé avec le soutien de la Commission européenne. Cette publication [communication] reflète uniquement le point de vue de son auteur et la Commission ne peut être tenue responsable de toute utilisation qui pourrait être faite des informations qui y sont contenues.

Est-ce que cela commence à peindre le tableau ? Examinons comment convertir ces nombres binaires en nombres décimaux.

3.2 Conversion de binaire en décimal

Il n'existe pas une seule façon de convertir un binaire en décimal. Nous présenterons deux méthodes ci-dessous, la méthode la plus « mathématique » et une autre plus visuelle. Nous couvrirons les deux, mais si le premier utilise trop de terminologie laides, passez au second.

Méthode 1

Il existe une fonction pratique que nous pouvons utiliser pour convertir n'importe quel nombre binaire en décimal :

$$a_n 2^n + a_{n-1} 2^{n-1} + \cdots + a_1 2^1 + a_0 2^0$$

Il y a quatre éléments importants dans cette équation :

an, an-1, a1, etc., sont les chiffres d'un nombre. Ce sont les 0 et les 1 que vous connaissez, mais en binaire, ils ne peuvent être que 0 ou 1.

La position d'un chiffre est également importante à observer. La position commence à 0, sur le chiffre le plus à droite ; ce 1 ou 0 est le moins significatif. Chaque chiffre que vous déplacez vers la gauche gagne en signification et augmente également la position de 1.

La longueur d'un nombre binaire est donnée par la valeur de n, en fait, c'est n+1. Par exemple, un nombre binaire comme 101 a une longueur de 3, et quelque chose de plus grand, comme 10011110, a une longueur de 8.

Chaque chiffre est multiplié par un poids : le 2n, 2n-1, 21, etc. Le poids le plus à droite - 20 équivaut à 1, déplacez un chiffre vers la gauche et le poids devient 2, puis 4, 8, 16, 32 , 64, 128, 256,... et ainsi de suite. Les puissances de deux sont d'une grande importance en binaire, elles deviennent vite très familières.

Débarrassons-nous de ces n et exposants et réalisons notre équation de notation positionnelle binaire sur huit positions :

$$a_7 \cdot 128 + a_6 \cdot 64 + a_5 \cdot 32 + a_4 \cdot 16 + a_3 \cdot 8 + a_2 \cdot 4 + a_1 \cdot 2 + a_0 \cdot 1$$

Pour aller plus loin, insérons quelques valeurs pour les chiffres. Et si vous aviez un nombre binaire comme 10011011 ? Cela signifierait (une) valeurs de :

1	0	0	1	1	0	1	1
a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0

Méthode 2

Une autre manière plus visuelle de convertir des nombres binaires en décimaux consiste à commencer par trier chaque 1 et 0 dans une case. Chaque bac a une puissance successive de deux poids, les 1, 2, 4, 8, 16,... auxquels nous sommes habitués. Le réaliser à huit endroits ressemblerait à ceci :

Ce projet a été financé avec le soutien de la Commission européenne. Cette publication [communication] reflète uniquement le point de vue de son auteur et la Commission ne peut être tenue responsable de toute utilisation qui pourrait être faite des informations qui y sont contenues.

128	64	32	16	8	4	2	1

Donc, si nous triions notre nombre binaire 10011011 dans ces bacs, cela ressemblerait à ceci :

128	64	32	16	8	4	2	1
1	0	0	1	1	0	1	1

Pour chaque bac contenant une valeur binaire 0, il suffit de le rayer et de le supprimer.

128	64	32	16	8	4	2	1
1	0	0	1	1	0	1	1

Et puis additionnez les poids restants pour obtenir votre numéro !

3.3 Conversion du décimal au binaire

Tout comme pour passer du binaire au décimal, il existe plusieurs façons de convertir le décimal en binaire. Le premier utilise la division et les restes, et le second utilise la soustraction. Essayez les deux et restez fidèle à celui avec lequel vous êtes à l'aise !

Méthode 1

Il n'est pas aussi simple de convertir un nombre décimal en binaire. Cette conversion nécessite de diviser à plusieurs reprises le nombre décimal par 2, jusqu'à ce que vous le réduisiez à zéro. Chaque fois que vous divisez, le reste de la division devient un chiffre du nombre binaire que vous créez.

Vous ne vous souvenez plus comment faire les restes ? Si cela fait longtemps, rappelez-vous que, puisque nous divisons par deux, si le dividende est pair, le reste sera 0 ; un dividende impair signifie un reste de 1.

Par exemple, pour convertir 155 en binaire, vous devez suivre ce processus :

155 ÷ 2 = 77 R 1 (C'est le chiffre le plus à droite, 1ère position)
77 ÷ 2 = 38 R 1 (2e position)
38 ÷ 2 = 19 R 0 (3ème position)
19 ÷ 2 = 9 R 1
9 ÷ 2 = 4 R 1
4 ÷ 2 = 2 R 0
2 ÷ 2 = 1 R 0
1 ÷ 2 = 0 R 1 (8ème position)

Le premier reste est le chiffre le moins significatif (le plus à droite), alors lisez de haut en bas pour étoffer notre nombre binaire de droite à gauche : 10011011. Faites correspondre le avec l'exemple ci-dessus... c'est un bingo !

Méthode 2

Ce projet a été financé avec le soutien de la Commission européenne. Cette publication [communication] reflète uniquement le point de vue de son auteur et la Commission ne peut être tenue responsable de toute utilisation qui pourrait être faite des informations qui y sont contenues.

Si diviser et trouver des restes n'est pas votre truc, il existe peut-être une méthode plus simple pour convertir un nombre décimal en binaire. Commencez par trouver la plus grande puissance de deux qui est encore plus petite que votre nombre décimal et soustrayez-la de la décimale. Ensuite, continuez à soustraire par la plus grande puissance de deux possible jusqu'à atteindre zéro. Chaque position de poids qui a été soustraite obtient un chiffre binaire à 1 ; les chiffres qui n'ont pas été soustraits obtiennent un 0.

En reprenant notre exemple, 155 peut être soustrait par 128, ce qui donne 27 :

155 - 128 = 27

128	64	32	16	8	4	2	1
1							

Notre nouveau nombre, 27, ne peut être soustrait ni de 64 ni de 32. Ces deux positions obtiennent un 0. Nous pouvons soustraire de 16, ce qui donne 11.

27 - 16 = 11

128	64	32	16	8	4	2	1
1	0	0	1				

Et 8 soustrait de 11, ce qui donne 3. Après cela, plus de chance avec 4.

11 - 8 = 3

128	64	32	16	8	4	2	1
1	0	0	1	1	0		

Notre 3 peut être soustrait de 2, produisant 1. Et enfin, le 1 soustrait de 1 pour faire 0.

3 - 2 = 1
1 - 1 = 0

128	64	32	16	8	4	2	1
1	0	0	1	1	0	1	1

Nous avons un nombre binaire !

Bits, quartets et octets

En discutant de la marque d'un nombre binaire, nous avons brièvement abordé la longueur du nombre. La longueur d'un nombre binaire correspond au nombre de 1 et de 0 qu'il contient.

4 longueurs de nombres binaires courantes

Les valeurs binaires sont souvent regroupées en une longueur commune de 1 et de 0, ce nombre de chiffres est appelé la longueur d'un nombre. Les longueurs de bits courantes des nombres binaires incluent les bits, les quartets et les octets (vous avez encore faim ?). Chaque 1 ou 0 dans un nombre binaire est appelé un bit. À partir de là, un groupe de 4 bits est appelé un quartet, et 8 bits forment un octet.

Ce projet a été financé avec le soutien de la Commission européenne. Cette publication [communication] reflète uniquement le point de vue de son auteur et la Commission ne peut être tenue responsable de toute utilisation qui pourrait être faite des informations qui y sont contenues.

Les octets sont un mot à la mode assez courant lorsque l'on travaille en binaire. Les processeurs sont tous conçus pour fonctionner avec une longueur définie de bits, qui est généralement un multiple d'un octet : 8, 16, 32, 64, etc.

Résumer:

Longueur	Nom	Exemple
1	Peu	0
4	Grignoter	1011
8	Octet	10110101

Motest un autre mot à la mode qui est lancé de temps en temps. Le mot sonne beaucoup moins délicieux et beaucoup plus ambigu. La longueur d'un mot dépend généralement de l'architecture d'un processeur. Cela peut être 16 bits, 32, 64 ou même plus.

5. Remplissage avec des zéros non significatifs

Vous pouvez voir des valeurs binaires représentées en octets (ou plus), même si la création d'un nombre de 8 bits nécessite l'ajout de zéros non significatifs. Les zéros non significatifs sont un ou plusieurs 0 ajoutés à gauche du bit de poids fort dans un nombre. Généralement, vous ne voyez pas de zéros non significatifs dans un nombre décimal : 007 ne vous dit pas plus sur la valeur du nombre 7 (il pourrait dire autre chose).

Les zéros non significatifs ne sont pas obligatoires sur les valeurs binaires, mais ils aident à présenter des informations sur la longueur en bits d'un nombre. Par exemple, vous pouvez voir le chiffre 1 imprimé comme 00000001, juste pour vous indiquer que nous travaillons dans le domaine d'un octet. Les deux nombres représentent la même valeur, cependant, le nombre précédé de sept 0 ajoute des informations sur la longueur en bits d'une valeur.

6 opérateurs au niveau du bit

Il existe plusieurs façons de manipuler les valeurs binaires. Tout comme avec les nombres décimaux, vous pouvez effectuer des opérations mathématiques standard – addition, soustraction, multiplication et division – sur des valeurs binaires (que nous aborderons à la page suivante). Vous pouvez également manipuler des bits individuels d'une valeur binaire à l'aide d'opérateurs au niveau du bit.

Les opérateurs au niveau du bit exécutent des fonctions bit par bit sur un ou deux nombres binaires complets. Ils font appel à une logique booléenne opérant sur un groupe de symboles binaires. Ces opérateurs bit à bit sont largement utilisés en électronique et en programmation.

Ce projet a été financé avec le soutien de la Commission européenne. Cette publication [communication] reflète uniquement le point de vue de son auteur et la Commission ne peut être tenue responsable de toute utilisation qui pourrait être faite des informations qui y sont contenues.

7 Complément (PAS)

Le complément d'une valeur binaire revient à trouver l'exact opposé de tout ce qui la concerne. La fonction complément examine un nombre et transforme chaque 1 en 0 et chaque 0 devient un 1. L'opérateur complément est également appelé NOT.

Par exemple, pour trouver le complément de 10110101 :

PAS 10110101 (décimal 181)
-------- =
01001010 (décimal 74)

NOT est le seul opérateur au niveau du bit qui n'opère que sur une seule valeur binaire.

8 OU

OU prend deux nombres et produit leur union. Voici le processus pour OU deux nombres binaires ensemble : alignez chaque nombre pour que les bits correspondent, puis comparez chacun de leurs bits qui partagent une position. Pour chaque comparaison de bits, si l'un ou les deux bits sont 1, la valeur du résultat à cette position de bit est 1. Si les deux valeurs ont un 0 à cette position, le résultat obtient également un 0 à cette position.

Les quatre combinaisons OU possibles et leur résultat sont :

- 0 OU 0 = 0
- 0 OU 1 = 1
- 1 OU 0 = 1
- 1 OU 1 = 1

Par exemple, pour trouver le 10011010 OU 01000110, alignez chacun des nombres bit par bit. Si l'un ou les deux nombres ont un 1 dans une colonne, la valeur du résultat a également un 1 :

10011010
OU 01000110
-------- =
11011110

Considérez l'opération OU comme une addition binaire, sans report. 0 plus 0 font 0, mais 1 plus n'importe quoi valent 1.

9 ET

AND prend deux nombres et produit leur conjonction. AND ne produira un 1 que si les deux valeurs sur lesquelles il opère sont également 1.

Le processus consistant à associer ET à deux valeurs binaires est similaire à celui de OU. Alignez chaque nombre pour que les bits correspondent, puis comparez chacun de leurs bits qui partagent une position. Pour chaque comparaison de bits, si l'un ou les deux bits

Ce projet a été financé avec le soutien de la Commission européenne. Cette publication [communication] reflète uniquement le point de vue de son auteur et la Commission ne peut être tenue responsable de toute utilisation qui pourrait être faite des informations qui y sont contenues.

sont 0, la valeur du résultat à cette position de bit est 0. Si les deux valeurs ont un 1 à cette position, le résultat obtient également un 1 à cette position.

Les quatre combinaisons ET possibles et leur résultat sont :

- 0 ET 0 = 0
- 0 ET 1 = 0
- 1 ET 0 = 0
- 1 ET 1 = 1

Par exemple, pour trouver la valeur de 10011010 ET 01000110, commencez par aligner chaque valeur. Le résultat de chaque position de bit ne sera que 1 si les deux bits de cette colonne sont également 1.

10011010
ET 01000110
-------- =
00000010

Considérez ET comme une multiplication. Chaque fois que vous multipliez par 0, le résultat sera également 0.

10XOR

XOR est le OU exclusif. XOR se comporte comme un OU normal, sauf qu'il ne produira un 1 que si l'un ou l'autre nombre a un 1 dans cette position de bit.

Les quatre combinaisons XOR possibles et leur résultat sont :

- 0 XOR 0 = 0
- 0 XOR 1 = 1
- 1 XOR 0 = 1
- 1 XOR 1 = 0

Par exemple, pour trouver le résultat de 10011010 XOR 01000110 :

10011010
XOR01000110
-------- =
11011100

Notez le 2ème bit, un 0 résultant de deux 1 XOR ensemble.

Ce projet a été financé avec le soutien de la Commission européenne. Cette publication [communication] reflète uniquement le point de vue de son auteur et la Commission ne peut être tenue responsable de toute utilisation qui pourrait être faite des informations qui y sont contenues.

11 Décalages de 11 bits

Les décalages de bits ne sont pas nécessairement un opérateur au niveau du bit comme ceux répertoriés ci-dessus, mais ils constituent un outil pratique pour manipuler une seule valeur binaire.

Un décalage de bits comporte deux éléments : la direction et la quantité de bits à décaler. Vous pouvez décaler un nombre vers la gauche ou vers la droite, et vous pouvez le décaler d'un bit ou de plusieurs bits.

Lors du déplacement vers la droite, un ou plusieurs des bits les moins significatifs (sur le côté droit du nombre) sont simplement coupés et déplacés vers le rien infini. Des zéros non significatifs peuvent être ajoutés pour conserver la même longueur de bits.

Par exemple, en décalant 10011010 vers la droite de deux bits :

DROITE-SHIFT-2 10011010 (décimal 154)
-------- =
00100110 (décimal 38)

Le déplacement vers la gauche ajoute pousse tous les bits vers le côté le plus significatif (le côté gauche) du nombre. Pour chaque décalage, un zéro est ajouté à la position du bit le moins significatif.

Par exemple, en décalant 10011010 d'un bit vers la gauche :

GAUCHE-SHIFT-1 10011010 (décimal 154)
-------- =
100110100 (décimal 308)

Ce simple décalage de bits remplit une fonction mathématique compliquée. Les décalages de n bits vers la gauche multiplient un nombre par 2n (voyez comment le dernier exemple a multiplié l'entrée par deux ?), tandis qu'un décalage de bits vers la droite fera une division entière par 2n. Le déplacement vers la droite pour diviser peut devenir étrange : toutes les fractions produites par la division par décalage seront coupées, c'est pourquoi 154 décalé deux fois vers la droite est égal à 38 au lieu de 154/4 = 38,5. Les décalages de bits peuvent constituer un moyen extrêmement rapide de diviser ou de multiplier par 2, 4, 8, etc.

12Conclusion

Le binaire est la pierre angulaire de tous les calculs, calculs et opérations en électronique. Il y a donc de nombreux endroits où aller à partir d'ici.

Maintenant que vous pouvez convertir entre décimal et binaire, vous pouvez appliquer ces connaissances pour comprendre comment les caractères sont codés de manière universelle : ASCII

Ou vous pouvez appliquer vos nouvelles connaissances brillantes aux circuits et circuits intégrés de bas niveau :

Ce projet a été financé avec le soutien de la Commission européenne. Cette publication [communication] reflète uniquement le point de vue de son auteur et la Commission ne peut être tenue responsable de toute utilisation qui pourrait être faite des informations qui y sont contenues.

- Logique numérique
- Registres à décalage

Vous pouvez également voir comment le binaire joue un rôle important dans ces protocoles de communication :

- Communication série
- Interface périphérique série
- I2C

Ce projet a été financé avec le soutien de la Commission européenne. Cette publication [communication] reflète uniquement le point de vue de son auteur et la Commission ne peut être tenue responsable de toute utilisation qui pourrait être faite des informations qui y sont contenues.