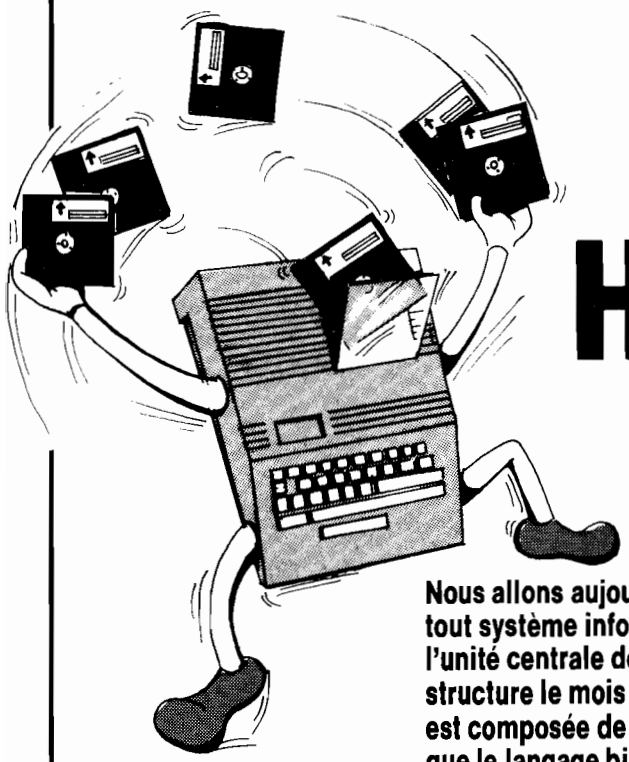
L'ABC DE LA MICRO-INFORMATIQUE



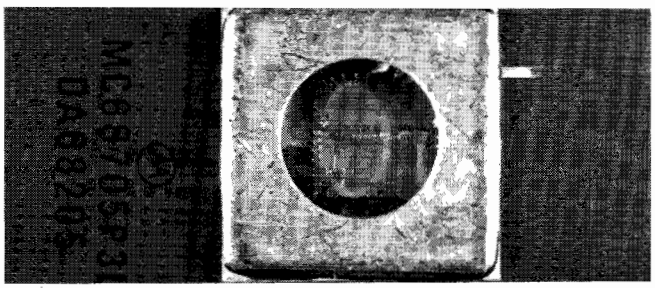
BINAIRE, HEXADECIMAL ET AUTRES MYSTERES

Nous allons aujourd'hui nous intéresser au cœur de tout système informatique avec la présentation de l'unité centrale dont nous avons brièvement évoqué la structure le mois dernier. Comme cette unité centrale est composée de circuits logiques qui ne comprennent que le langage binaire, nous allons commencer par un peu d'arithmétique élémentaire.

Même si vous n'avez pas de grandes connaissances en électronique, vous devez savoir que les circuits logiques ne peuvent manipuler que deux types d'informations : la présence ou l'absence de tension. La valeur exacte de cette tension importe peu et dépend de la technologie des circuits utilisés, ce qui n'est pas notre propos pour l'instant. Par convention, on dit que la présence de tension correspond à un 1 et l'absence de tension à un 0. Nos circuits logiques ne peuvent donc travailler qu'en binaire ou arithmétique à deux états. Nous allons voir qu'une telle arithmétique ne présente pas de difficulté de compréhension majeure une fois que l'on a fait l'effort initial nécessaire.

QU'EST-CE QU'UNE BASE DE NUMÉRATION

Pour savoir convertir un nombre de décimal en binaire et vice versa, il faut savoir ce qu'est une base de numération. Lorsque nous parlons en décimal, nous travaillons en base 10 c'est-à-dire que tous les nombres que nous exprimons peuvent être décomposés en une somme de multiples de puissances de 10. Ainsi le nombre 5 285 est-il 5 fois 1 000 plus 2 fois 100 plus 8 fois 10 plus 5 fois 1. En d'autres termes, il peut s'écrire : 5 x 10 + 2 x 10 + 8 x 10 + 5 x 10. La représentation d'un nombre en binaire exploite le même procédé mais, au lieu d'utiliser 10 comme base, elle utilise 2 puisque nos circuits logiques ne connaissent que deux états. Bien sûr, c'est un peu moins naturel car si les puissances de 10 se calculent aisément (il suffit de mettre autant de 0 derrière le 1 que la puissance exprimée), les puissances de 2 demandent un peu plus d'efforts. En fait, ce qui nous gêne le plus est que nous avons toujours travaillé en base 10 et que c'est devenu une seconde nature. Voici quelques exemples simples. Le nombre binaire 1011 n'est autre que 11 exprimé en décimal ; en effet il est égal à 1 fois 2 puissance 3 plus 0 fois 2 puissance 2 plus 1 fois 2 puissance 1 plus 1 fois 2 puissance 0. Pour vous aider, le tableau de la figure 1 donne les puissances de 2 les plus utiles. Compte tenu de ces explications, le passage de binaire à décimal ou vice versa est très simple à effectuer.



Sous la fenêtre, un micro-ordinateur complet en une seule puce.

Pour passer du binaire au décimal, il suffit de suivre les indications données figure 2 qui, comme vous pouvez le constater, se résument à faire une vulgaire addition. Pour passer de décimal en binaire, c'est à peine plus compliqué puisqu'il suffit de décomposer le nombre décimal en une somme de puissances de 2 en utilisant pour ce faire le tableau de la figure 1 et de respecter ensuite les indications de la figure 3.

Si cela vous rebute un peu, ne soyez pas trop inquiets ; de telles conversions restent d'utilisation assez peu courantes. Il est par contre utile de savoir qu'un chiffre binaire élémentaire s'appelle un bit (contraction de l'américain BInary digiT qui signifie chiffre binaire).

LORSQUE LES BITS FORMENT DES MOTS

Le cœur de toute unité centrale est constitué, comme nous l'avons dit le mois dernier, par un microprocesseur. Ce dernier est un circuit logique qui manipule des nombres représentés en binaire ; nombres dont la taille (en nombre de bits) est fixée une fois pour toutes et est fonction du type de

N	2^N
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1 024
11	2 048
12	4 096
13	8 192
14	16 384
15	32 768
16	65 536
17	131 072
18	262 144
19	524 288
20	1 048 576

Fig. 1. – Table des puissances de 2.

LE NOMBRE BINAIRE B7 B6 B5 B4 B3 B2 B1 B0

EST EGAL A : $B7 \times 2^7 + B6 \times 2^6 + B5 \times 2^5 + B4 \times 2^4 + B3 \times 2^3 + B2 \times 2^2 + B1 \times 2^1 + B0 \times 2^0$

EXEMPLE : 1 0 1 1 0 1 0 1

$= 1 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

$= 128 + 32 + 16 + 4 + 1$

$= 181$

Fig. 2. – Comment passer du binaire au décimal...

1° Décomposer le nombre décimal en une somme de puissances de 2.
2° Ecrire le nombre binaire correspondant en plaçant la décomposition par ordre des puissances de croissances de gauche à droite.

Exemple :

$231 = 128 + 64 + 32 + 4 + 2 + 1$

$231 = 2^7 + 2^6 + 2^5 + 0 \times 2^4 + 0 \times 2^3 + 2^2 + 2^1 + 2^0$

231 = 1 1 1 0 0 1 1 1

Fig. 3. – ... et comment faire l'inverse.

microprocesseur employé. On rencontre ainsi des microprocesseurs 8 bits, 16 bits voire même 32 bits sur les machines les plus puissantes. Sur les micro-ordinateurs grand public et semi-professionnels, on rencontre le plus souvent des microprocesseurs 8 bits et, sur quelques rares machines (Macintosh d'Apple par exemple), des microprocesseurs 16 bits.

La taille des informations manipulées par le microprocesseur porte le nom de mot. On parle ainsi d'un mot de 8 bits, d'un mot de 16 bits, etc. En outre, un mot de 8 bits s'appelle un octet ou byte (prononcez *baillte* et ne confondez pas avec bit).

Si vous faites un petit exercice de conversion, vous vous rendrez très vite compte que sur un octet on peut représenter tout nombre décimal compris entre 0 et 255 (00000000 à 11111111 en binaire). Malgré cette contrainte relativement restrictive, un micro-ordinateur peut réaliser des calculs avec des nombres de tailles quelconques en utilisant des modes de représentation des nombres faisant appel à plusieurs octets. On dit alors qu'il travaille en multiple précision.

Dans la pratique, cette représentation par octet n'est pas très agréable car, si l'on veut écrire les nombres en binaire, cela tient de la place et, si l'on veut parler en décimal, il faut sans cesse être obligé de faire des conversions. Pour palier cela on fait très souvent appel à la notation hexadécimale ou représentation des nombres en base 16.

L'HEXADECIMAL

Un mot de 8 bits peut être décomposé en deux groupes de 4 bits. Sur chaque groupe, il est possible de coder les chiffres décimaux de 0 à 9 comme représenté figure 4. Lorsque l'on procède de la sorte, on utilise la notation dite BCD (en américain) ou DCB (en français) qui signifie Décimal Codé en Binaire. Une telle représentation est intéressante car elle permet de coder des nombres de 00 à 99 sur un mot de 8 bits. Malheureusement, elle n'exploite pas à fond toutes les possibilités de ce dernier puisque nous avons vu qu'un tel mot pouvait coder de 0 à 255. Pour ce faire, il est d'usage d'employer la notation hexadécimale qui exploite à fond les deux groupes de 4 bits. En effet, avec 4 bits on dispose de 2^4 combinaisons et on peut donc représenter 16 éléments différents. La représentation BCD fait perdre 6 éléments par groupe de 4 bits.

Comme nous ne disposons de symboles numériques que pour les chiffres de 0 à 9, il faut bien en « inventer » d'autres pour représenter les codes binaires situés au-dessus et présentés figure 4. Les symboles retenus sont tout simplement les lettres de A à F. Ainsi 1010 qui est 10 en décimal classique est A en hexadécimal, et ainsi de suite jusqu'à 1111 qui est 15 en décimal et F en hexadécimal. Vous êtes en droit de vous demander pourquoi on n'utilise pas tout simplement des chiffres classiques pour représenter les valeurs décimales de 10 à 15. La raison est toute simple : si tel était le cas quelle serait la signification du nombre hexadécimal 1511 ? 15 suivi de 11, 15 suivi de 1 suivi de 1, 1 suivi de 5 suivi de 11, 1 suivi de 5 suivi de 1 suivi de 1 ? En utilisant des lettres, notre « 1511 » devient FB ce qui ne peut pas prêter à confusion.

Cette notation hexadécimale est aussi appelée notation en base 16 puisqu'en fait elle revient à décom-

Binaire	DCB	Hexadécimal
0 0 0 0	0	0
0 0 0 1	1	1
0 0 1 0	2	2
0 0 1 1	3	3
0 1 0 0	4	4
0 1 0 1	5	5
0 1 1 0	6	6
0 1 1 1	7	7
1 0 0 0	8	8
1 0 0 1	9	9
1 0 1 0	n'existe pas	A
1 0 1 1	n'existe pas	B
1 1 0 0	n'existe pas	C
1 1 0 1	n'existe pas	D
1 1 1 0	n'existe pas	E
1 1 1 1	n'existe pas	F

Fig. 4. – Binaire, DCB et hexadécimal.

poser les nombres à représenter en fonction des puissances de 16. On passe donc du décimal à l'hexadécimal ou vice versa comme expliqué ci-avant pour le binaire, mais en faisant intervenir cette fois-ci les puissances de 16.

Nous n'avons parlé que de mots de 8 bits pour cet exemple de représentation mais il est bien évident que tout ce que nous avons dit s'applique aussi à des mots de taille plus importante. Ainsi un mot de 16 bits se décompose-t-il en 4 groupe de 4 bits, c'est-à-dire qu'il peut représenter tout nombre compris entre 0000 et FFFF. FFFF n'est autre que 15 × 16 + 15 × 16 + 15 × 16 + 15 × 16 c'est-à-dire 65535 en décimal.

QUE FAIRE D'AUTRE AVEC UN OCTET

Tant que nous parlons de représentation en binaire et de ce que l'on peut faire avec un octet, il nous semble logique d'examiner une autre utilisation très courante d'un tel mot de 8 bits : celle du codage des caractères que l'on rencontre sur tout clavier informatique. En effet, pour représenter dans un micro-ordinateur tous les symboles que l'on rencontre sur un clavier (lettres, chiffres, signes, etc.), un code international a été défini et porte le nom de code ASCII (prononcez ASKI), ce qui signifie American Standard Code for Information Interchange ou encore code américain standard pour l'échange d'informations.

Ce code, qui n'utilise que 7 bits et qui, donc, tient sur un octet, est universellement adopté par tous les constructeurs de micro-ordinateurs à de très rares exceptions près (Sinclair dans le ZX-81 et le Spectrum par exemple). Il permet de coder sur un octet n'importe quel caractère d'un clavier informatique.

Vous trouverez en figure 5 le tableau officiel du code ASCII sur lequel vous pourrez constater deux choses importantes :

– la présence, dans les deux colonnes de gauche, de groupes de plusieurs lettres. Ces groupes ne correspondent pas à des caractères de claviers mais à des codes de fonctions, normalisés eux aussi, et sur lesquels nous reviendrons ultérieurement dans cette série :

– l'absence des lettres spécifiquement françaises telles que :, é, è, à, ù, ç, etc. La raison de cette absence est simple. Ce code a été défini aux USA, pays qui n'utilise aucune de ces lettres. Des astuces, plus ou moins heureuses, sont donc utilisées sur les micro-ordinateurs « français » pour que nous ayons accès à ces lettres manquantes.

CE QU'IL FAUT RETENIR

Au terme de cette présentation, nous vous avons montré qu'un octet pouvait être utilisé pour représenter des nombres mais aussi n'importe quel caractère d'un clavier informatique.

En manipulant judicieusement ces octets, un micro-ordinateur va donc pouvoir manipuler des nombres et, ainsi, faire des calculs, mais il va aussi pouvoir manipuler des caractères au sens large du terme et, ainsi, du texte de signification quelconque. Nous nous sommes peut-être un peu étendus sur ces aspects mais il faut bien reconnaître qu'il est essentiel de les connaître afin de pouvoir assimiler au mieux le fonctionnement d'un micro-ordinateur.

LE MICRO-PROCESSEUR

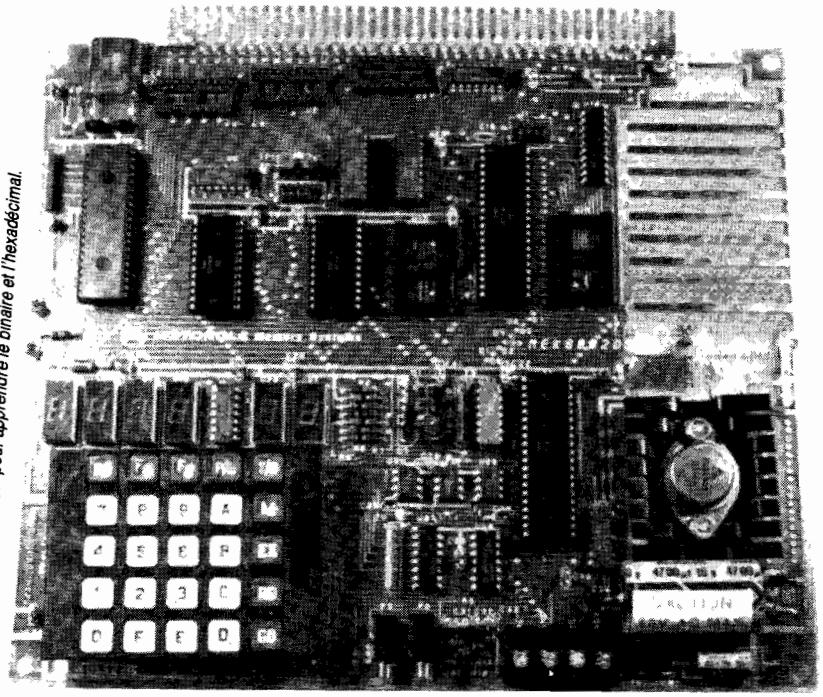
Nous avons vu le mois dernier la structure générale d'une unité centrale avec sa décomposition en quatre sous-ensembles fondamentaux : le microprocesseur, la RAM ou mémoire vive, la ROM ou mémoire morte et le ou les circuits d'interface. Tous ces éléments sont connectés les uns aux autres par un ensemble de lignes qui constituent ce que l'on appelle le bus.

Le microprocesseur est évidemment l'élément fondamental de cette unité centrale puisque c'est lui qui gère tous les échanges d'informations entre les différents éléments. Bien qu'il existe un multitude de microprocesseurs différents, tous respectent la même architecture générale et le même principe de fonctionnement que nous allons commencer à expliciter.

La figure 6 vous propose un synoptique général simplifié d'un micropro-

Bits 7, 6, 5					000	001	010	011	100	101	110	111
Bits 4	3	2	1	Hex 0 / Hex 1	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	'	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	A	LF	SUB	*	:	J	Z	j	z
1	0	1	1	B	VT	ESC	+	;	K	[	k	{
1	1	0	0	C	FF	FS	,	<	L	\	l	¦
1	1	0	1	D	CR	GS	-	=	M	]	m	}
1	1	1	0	E	SO	RS	.	>	N	^	n	~
1	1	1	1	F	SI	US	/	?	O	_	o	DEL

Fig. 5. – Tableau officiel du code ASCII.



Un kit d'évaluation était idéal pour apprendre le binaire et l'hexadécimal.

cesseur quelconque. Nous y voyons en premier lieu une unité arithmétique et logique (UAL OU ALU selon qu'on parle français ou américain) sur laquelle sont connectés au moins deux registres. Ces derniers sont des éléments logiques capables de contenir un octet (ou un mot de 16 bits sur un microprocesseur 16 bits).

L'ALU, comme son nom l'indique, est capable d'effectuer des opérations arithmétiques (additions, soustractions et parfois multiplication et division) entre les deux nombres contenus dans les registres qui sont reliés.

Cet ALU peut aussi faire des opérations logiques telles que décalages, rotations, ET, OU et plus généralement toute opération logique.

Le deuxième élément fondamental du microprocesseur est le décodeur d'instructions. Il s'agit d'un ensemble logique qui, à partir d'un code binaire reçu, sait déterminer quelle opération doit effectuer le microprocesseur.

Vient ensuite un compteur un peu particulier qui, en français, porte le nom du compteur ordinal et, en américain, celui de *programm counter.* Cet élément est un compteur dont le contenu augmente régulièrement de une ou plusieurs unités et qui peut même brutalement changer de valeur en fonction d'instructions particulières reçues par le microprocesseur. Ce compteur est celui qui permet au microprocesseur de suivre les différentes lignes qui composent un programme et, donc, de l'exécuter.

Un certain nombre de signaux sortent ou rentrent dans notre circuit. On peut les classer en trois groupes principaux : les signaux d'adresses, les signaux de données et les signaux de contrôle. Comme vous pouvez le constater sur le synoptique de la figure 6, le compteur ordinal pilote directement les lignes d'adresses tandis que les lignes de données aboutissent sur les registres connectés à l'ALU mais aussi sur le décodeur d'instructions. Les lignes de contrôle quant à elles rentrent ou sortent d'un bloc logique chargé de contrôler des échanges qui ont lieu sur les bus d'adresses et de données.

CONCLUSION

Le décor relatif à notre unité centrale est planté, nous savons ce que des octets peuvent représenter. Nous sommes donc prêts à voir comment tout cela fonctionne et c'est ce que nous ferons dès le mois prochain.

C. TAVERNIER

(Cette série d'articles a débuté dans notre n° 1733).

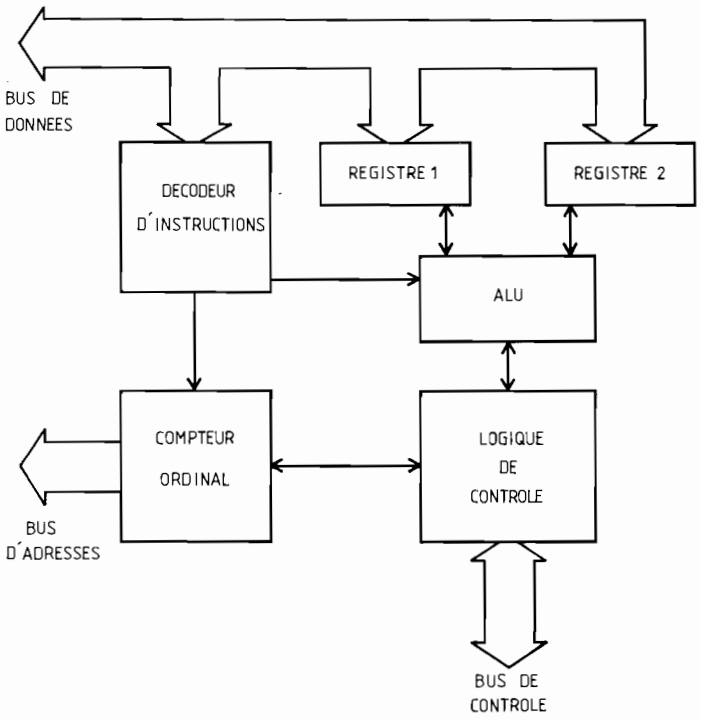


Fig. 6. – Synoptique simplifié d'un microprocesseur quelconque.