Chap 2 - Architecture des ordinateurs
Démonstration

IFT6800

11 septembre 2015

Changement de base

Décimal → Binaire

Pour transformer un nombre dans sa forme décimale vers sa forme binaire, on divise (avec reste) le nombre par deux tant qu'on n'obtient pas 0. Le i^e bit à partir de la droite est le reste de la i^e division.

$$
\begin{aligned}
577/2 = 288 \text{ reste } 1 &\rightarrow 1 \\
288/2 = 144 \text{ reste } 0 &\rightarrow 01 \\
144/2 = 72 \text{ reste } 0 &\rightarrow 001 \\
72/2 = 36 \text{ reste } 0 &\rightarrow 0001 \\
36/2 = 18 \text{ reste } 0 &\rightarrow 00001 \\
18/2 = 9 \text{ reste } 0 &\rightarrow 000001 \\
9/2 = 4 \text{ reste } 1 &\rightarrow 1000001 \\
4/2 = 2 \text{ reste } 0 &\rightarrow 01000001 \\
2/2 = 1 \text{ reste } 0 &\rightarrow 001000001 \\
1/2 = 0 \text{ reste } 1 &\rightarrow 1001000001
\end{aligned}
$$

On peut vérifier notre réponse en additionnant les puissances de deux aux indices où le registre est à 1.

$$
\begin{aligned}
(1001000001)_2 &= 1 \times 2^9 + 0 \times 2^8 + 0 \times 2^7 + 1 \times 2^6 + \\
&\quad 0 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 0 \times 2^3 + \\
&\quad 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\
&= 2^9 + 2^6 + 2^0 = 512 + 64 + 1 = 577
\end{aligned}
$$

Complément à deux

Soit x un entier non-négatif. On écrit $(x)_b$ pour signifier "la représentation binaire de x".

Un registre de n bits peut contenir 2^n mots binaires différents. Si on choisit une représentation non-signée, alors ces mots binaires représentent des entiers de 0 à $2^n - 1$. Si on choisit une représentation signée du type complément à deux, les mots binaires représentent des entiers entre $-(2^{n-1})$ et $2^{n-1} - 1$.

Soit un nombre entier x quelconque. Pour lui donner une représentation en complément à deux, on écrit $|x|$ sous sa forme binaire. Si $x < 0$, on le complémente et ajoute un.

Par exemple, sur un registres de 3 bits, on pourra écrire des nombres entiers entre -2^{3-1} et $2^{3-1} - 1$, i.e. entre -4 et 3. L'interprétation en entier non-signé et signé avec le complément à deux est la suivante :

Mot	Entier non-signé	Entier signé par le complément à 2
000	0	0
001	1	1
010	2	2
011	3	3
100	4	-4
101	5	-3
110	6	-2
111	7	-1

Pour trouver la représentation du nombre -3, on fait les calculs suivants :

$$\begin{aligned}(3)_b &= 011 \\ &\to 100 \quad \text{(on le complémente)} \\ &\to 101 \quad \text{(on ajoute 1)}\end{aligned}$$

Remarque 1. *La représentation en complément à deux est cohérente avec l'identité* $0 = -0$

$$\begin{aligned}(0)_b &= 000 \\ &\to 111 \quad \text{(on le complémente)} \\ &\to 1000 \quad \text{(on ajoute 1)} \\ &\to 000 \quad \text{(le registre n'a que 3 bits)}\end{aligned}$$

On a bien à la fin 000 qui est égal à $(0)_b$ et à $(-0)_b$.

Remarque 2. *La représentation en complément à deux est cohérente avec l'identité* $-(-x) = x$.

Par exemple $-(-2) = 2$:

$$\begin{aligned}(-2)_b &= 110 \\ &\to 001 \quad \text{(on le complémente)} \\ &\to 010 \quad \text{(on ajoute 1)}\end{aligned}$$

On a bien à la fin 010 qui est égal à $(2)_b$.

Remarque 3. *L'addition dans la représentation en complément à deux est identique à l'addition en représentation binaire pour nombres non-négatifs.*

Par exemple $(-1) + 2 = 1$:

$$\begin{aligned}(-1)_b &= 111 \\ &\to 1001 \quad \text{(on additionne 010)} \\ &\to 001 \quad \text{(le registre n'a que 3 bits)}\end{aligned}$$

On a bien à la fin 001 qui est égal à $(1)_b$.

Explications mathématiques (Supplément)

Si on se restreint aux registres de n bits, "complémenter et ajouter 1" est équivalente à effectuer le calcul $2^n - x$:

$$\begin{aligned} 1111\ldots1 - (x)_b + 1 &= 2^{n-1} + 2^{n-2} + \ldots 2 + 1 - x + 1 \\ &= 2^n - x \end{aligned}$$

On peut maintenant faire une preuve rigoureuse de nos remarque 2 et 3.

Remarque. *La représentation en complément à deux est cohérente avec l'identité* $-(-x) = x$.

Démonstration. Appliquer deux fois une négation revient à ne rien faire du tout puisque $2^n - (2^n - x) = 2^n - 2^n + x = x$. □

Remarque. *L'addition dans la représentation en complément à deux est identique à l'addition en représentation binaire pour nombres non-négatifs.*

Démonstration. Soit x, y les deux registres. On divise en 4 cas :

— $x, y \geq 0$: l'addition est la même.

— $x, y < 0$:

$$2^n - |x| + 2^n - |y| = 2^{n+1} - |x| - |y| \to -|x| - |y| = x + y$$

— Si $x \geq 0$ et $y < 0$:

$$x + 2^n - |y| = 2^n - (|-x + y|) \to x - |y| = x + y$$

— Si $x < 0$ et $y \geq 0$:

$$2^n - |x| + y = 2^n - (|x - y|) \to -|x| + y = x + y$$

□

Attention, on peut facilement dépasser le registre. On peut additionner x et y à condition que $x + y$ reste entre -2^{n-1} et $2^{n-1} - 1$ inclusivement.

Détermination du signe

Si on suppose qu'un mot du registre est un entier signé selon le complément à deux, comment peut-on déterminer le signe de l'entier qu'il représente ?

En inspectant le tableau plus haut, on observe que l'entier est négatif si et seulement si le premier bit du registre est 1. Peut-on généraliser cette

observation ?

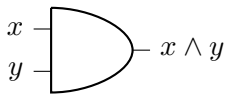
Oui ! Les mots binaires qui représentent un entier signé du type complément à deux prennent des valeurs non-négatives 0 à $2^{n-1} - 1$. Les nombres entre 2^{n-1} et 2^n représentent les nombres négatifs. Ces nombres ont un 1 à gauche dans leur représentation binaire.

Circuits Booléens et arithmétique booléenne

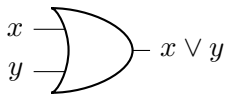
Rappel

N'importe quel calcul sur des registres booléens peut être fait à partir des trois portes booléennes suivantes :

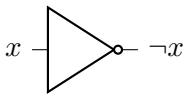
Le ET booléen :



Le OU booléen



Le NON booléen



L'ensemble constitué de ces trois portes est dit *universel*.

Ou excusif

La table de vérité du OUX (noté $\oplus$) est

x	y	$x \oplus y$
1	1	0
1	0	1
0	1	1
0	0	0

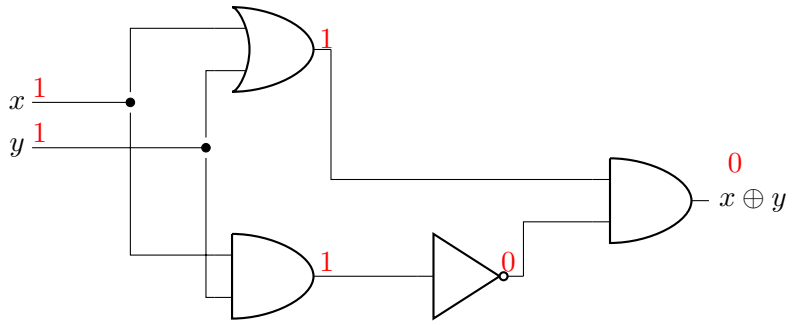
En mots, $x \oplus y$ est 1 si et seulement si soit x, soit y est 1, mais pas les deux !

On peut interpréter ces valeurs comme 1 signifie “vrai” et 0 signifie “faux”. On utilise les valeurs booléeens 0 et 1 puisque c’est ce que l’ALU fait dans le CPU.

On observe : $x \oplus y = (x \vee y) \wedge \neg(x \wedge y)$. Comment peut-on en être sûrs ? En exhibant la table de vérité.

x	y	$x \vee y$	$x \wedge y$	$\neg(x \wedge y)$	$(x \vee y) \wedge \neg(x \wedge y)$
1	1	1	1	0	0
1	0	1	0	1	1
0	1	1	0	1	1
0	0	0	0	1	0

On inspecte les tables de vérités et observons que $x \oplus y$ a les mêmes valeurs de vérité que $(x \vee y) \wedge \neg(x \wedge y)$ pour tous booléens x et y. La deuxième expression est complètement exprimée en “et”, “ou” et “non”. Il ne nous reste alors qu’à traduire cette expression sous forme de circuit à l’aide de porte dans l’ensemble universel.



NB : En rouge est l'exécution du circuit sur entrée $x = 1$ et $y = 1$.

Circuit pour déterminer si > 0

Soit un registre représentant un entier signé selon le complément à deux. Bâtissons un circuit qui retourne 1 si et seulement si le registre est strictement positif. On se restreint aux registres de 8 bits.

On décrit le circuit à partir des opérations booléennes universelles. Sur entrée $x_1, \ldots, x_8$ où $x_i \in \{0, 1\}$, retourner 0 si et seulement si $x_1 = 1$ ou si $x_2 = x_3 = \cdots = x_8 = 0$:

$$\neg x_1 \wedge (x_2 \vee x_3 \vee x_4 \vee x_5 \vee x_6 \vee x_7 \vee x_8)$$