Opérations Arithmétiques binaires dans les ensembles *N* et *Z* et indicateurs Z, C, V, N

Opérations arithmétiques binaires dans les ensembles *N* et *Z* et indicateurs Z, C, V, N

Consigne :

1. ***Identifier* les notions/définitions importantes et être capable de les *expliquer*.**
2. ***Refaire les exemples* illustrant ces notions.**
3. ***les appliquer* en faisant les exercices notés exercice à faire.**

Contexte et objectifs :

Comment :

- Représenter les nombres (entiers naturels, entiers) dans un microprocesseur,
- coder en binaire et/ou en hexadécimal sur n bits des entiers naturels et signés,
- Effectuer des opérations d'addition en binaire et/ou en hexadécimal,
- Détecter les dépassements de capacité dans les entiers naturels et signé,
- Interpréter le résultat obtenu en fonction des indicateurs Z, N, C et V,
- Effectuer des opérations de multiplication ou division par une puissance de 2.

1. Système de numération dans une base b :

Tout entier naturel N peut être représenté par une suite de chiffres de la base b : $(a_n, a_{n-1}, \ldots, a_1, a_0)_b$ où les a_i sont les chiffres de la base ($0 < a_i < b$). Ce nombre en notation étendue (ou positionnelle) a pour valeur :

N = $a_n b^n + a_{n-1} b^{n-1} + a_1 b^1 + a_0 b^0 = \sum_{i=0}^{n} a_i b^i$

Avec $a_i \in \{0, 1, \ldots, b-1\}$ **et** $a_n \neq 0$.

***b^n* est le poids attaché à la position du chiffre *a_n*.**

Exemples :

b = 10 - système de numération décimal : les chiffres $a_i \in$ **{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}**
Le nombre décimal 245 est ainsi égal à $2 \times 10^2 + 4 \times 10^1 + 5 \times 10^0$.

b = 16 - système de numération hexadécimal : les chiffres $a_i \in$ **{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}**
Le nombre $(F5)_{16}$ est ainsi égal à $F \times 16^1 + 5 \times 16^0$. Ce nombre est aussi noté 0xF5 ou $(F5)_H$, pour indiquer qu'il est exprimé en hexadécimal.

b = 2 - système de numération binaire : les chiffres binaires (bit = **bi**nary digi**t**) $a_i \in$ **{0, 1}**,
Le nombre $(0101)_2$ est égal à $0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 4 + 1 = 5$.

Le tableau ci-dessous représente **le code binaire naturel** sur 4 bits et sa correspondance en hexadécimal et décimal :

Chiffres HEX.	Valeurs décimales	Équivalence en binaire $2^3 2^2 2^1 2^0$
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

2. Entiers naturels :

Les valeurs représentables des entiers naturels codés sur n bits (**B**inary dig**it**) sont :

Nombres positifs	Représentation sur n bits en base 2
0	0 0 00000000000000000000000000000000 0 0
1	0 0 ..... 0 1
2	0 0 ..... 1 0
2^{n-1}	1 0..... 0 0
2^n -1	1 1 11111111111111111111111111111111 1 1

Exemple : soit un nombre entier naturel codé sur 8 bits (1 octet : n = 8) :

Nombres positifs	Représentation sur 8 bits en base 2
0 = valeur min	0000 0000
1	0000 0001
2	0000 0010
2^{n-1} soit $2^{8-1} = 2^7 = 128$	1000 0000
2^n -1 soit 2^8 -1 =255 = valeur max	1111 1111

En entiers naturels, sur n bits : on peut représenter des entiers $\in$ à $[0,....,2^n - 1]$.

3. Entiers signés:

par convention, on utilise la représentation en complément à 2.

Le bit de rang *n-1* est le bit de poids fort, il représente le bit de signe par convention. Dans la représentation en complément à 2, un nombre négatif est codé par le complément à 2 de sa valeur absolue :

Le complément à 2 (noté C2) :

$\forall$ (b_{n-1} … b_0) € {0, 1}^{n}, C2 (b_{n-1} … b_0) = C1 (b_{n-1} … b_0) + 1.

C1 est le complément à 1 (inversion bit à bit).

Une seule représentation pour 0,

Opérations arithmétiques plus simples (l'addition et la soustraction sont traitées de la même manière) (a – b = a + (–b) = a + C2(b)),

Le complément à 2 du complément à 2 d'un nombre N est le nombre lui-même : C2(C2(N)) = N.

pour un entier codé sur *n* bits :

Le bit de rang *n-1* est appelé bit de signe, c'est le **bit de poids fort : (2^{n-1})**
Le bit de poids faible est le bit de rang 0 : (2^0).
Les valeurs représentables en complément à 2, des entiers codés sur *n* bits sont :

	Valeur	Représentation binaire sur n bits
Valeur minimale	$-(2^{n-1})$	1 0 0 …….. 0 0
	-1	1 1 1 …….. 1 1
	0	0 0 0 …….. 0 0
Valeur maximale	$2^{n-1} - 1$	0 1 1 …….. 1 1

En complément à (la base) 2, sur n bits : on peut représenter des entiers $\in$ à $\left[-2^{n-1},....,2^{n-1}-1\right]$.

Ainsi pour n = 8 bits, les valeurs des entiers sont comprises entre $(-128)_{10}$ et $(127)_{10}$:

	Valeur	Représentation binaire sur 8 bits
Valeur minimale	$-(2^{n-1})$ soit $-(2^{8-1})=-2^7=$ **-128**	1000 0000
	-1	1111 1111
	0	0000 0000
Valeur maximale	2^{n-1}- 1soit $2^{8-1}-1=2^7-1=$ **127**	0111 1111

Exemple : soit un nombre entier *A* codé sur 4 bits = 1001 ce nombre est négatif, le bit de poids fort = 1, c'est le bit de signe.

Le C2 permet de trouver sa valeur absolue :

C2(1001) = C1 (1001) +1 = 0110 +1 = 0111, ce qui signifie que |A| = 7 donc *A* = -7.

Exemple : Pour représenter $(-6)_{10}$ en complément à 2 sur 4 bits (n = 4), la valeur absolue de 6 = 0 1 1 0 = $0*2^3 + 1*2^2 + 1*2^1 + 0*2^0 = 4 + 2$

L'Inversion bit à bit donne : 1 0 0 1 ; On ajoute 1, on obtient : 1 0 1 0

⇨ 1 0 1 0 est la **représentation de $(-6)_{10}$ en complément à 2 sur 4 bits**.

Ainsi $(1111\ 1111)_2$ = 0xFF = $(-1)_{10}$. L'inversion bit à bit donne $(000\ 000)_2$. On ajoute 1, on obtient : $(000\ 0000)_2 + 1 = (0000\ 0001)_2$

Remarque : dans les microprocesseurs, ***la taille n est fixe*** et peut être égale à : 8, 16, 32, 64, 128 bits.

⇨ **Extension du bit de signe**

On peut représenter un nombre de taille n bits sur m bits (m>n), par extension du bit de signe sur (m-n) bits.

Exemple : le nombre entier $(1001)_2$ codé sur 8 bits est : $(1111\ 1001)_2$ obtenu par extension (duplication) sur bit de signe sur 4 bits (8-4). Ce nombre = 0xF7.

A ce stade vous êtes capable de :

- Coder en binaire (et hexadécimal) un entier naturel ou un entier codé sur n bits.
- Donner l'intervalle dans lequel un nombre x codé sur n bits est compris dans les cas :

  $x \in N$ (entiers naturels) et $x \in Z$ (entiers).
- Représenter (trouver) un nombre entier codé sur n bits sur m bits (avec m > n).

4. Opérations sur les entiers naturels_Addition Binaire :

Soit à additionner 2 entiers naturels a et b codés sur n bits et représentés respectivement par les suites suivantes :

- $(a_{n-1}, \ldots, a_1, a_0)$; a_0 est le bit de poids faible (2^0) et a_{n-1} est le bit de poids fort (2^{n-1}),
- et $(b_{n-1}, \ldots, b_1, b_0)$; b_0 est le bit de poids faible (2^0) et b_{n-1} est le bit de poids fort (2^{n-1}).

Principe de l'addition en base 2 : on additionne les bits colonne par colonne en commençant par le bit de poids faible (bit de rang 0).

Remarque importante : Dans les microprocesseurs la valeur du report (retenue) est donnée par le **bit C** appelé **indicateur** de Carry (ou **Flag** en anglais), il est en général mémorisée dans un registre interne du microprocesseur, appelé **registre d'état**.

La somme s de a + b peut être supérieure à la valeur maximale représentable sur n bits, c'est-à-dire à 2^{n-1} et n'est donc pas nécessairement représentable sur *n* bits.

La partie représentable de la somme est s= (a+b) mod 2^n, s est représentée en convention standard par la suite (s_{n-1}, …, s_1, s_0) sur n bits.

Les s_i sont obtenus à partir des a_i et b_i, en appliquant l'opération d'addition binaire bit à bit de droite à gauche en reportant (on dit aussi en propageant) les retenues éventuelles.

La retenue r_i prise en compte au rang i, est engendrée au rang (i-1) pour i > 0, la retenue initiale $r_0 = 0$.

La dernière retenue r_n codée sur un bit est engendrée au rang n - 1:

Cette retenue est appelé report, en anglais **Carry et noté C**.

Si la dernière retenue $r_n = 0$, la somme s est représentable sur n bits.

Si la dernière retenue $r_n = 1$, la somme s n'est représentable sur n bits on dit qu'il y a **débordement**.

Le débordement dans l'addition des entiers naturels est indiqué par l'indicateur C = 1. *On a un dépassement de capacité sur les entiers naturels*.

Exemple : Addition binaire sur 8 bits

r = report = retenue	1	1 0 0 0 1 1 1 0
a		1 1 0 0 0 1 0 1
b		1 1 1 0 0 0 1 1
a+b		1 0 1 0 1 0 0 0

$r_n = 1$ (ici : n = 8) → C = 1. Le résultat de l'addition précédente est s = a + b = 10101000 et C=1 dépassement de capacité sur les entiers naturels.

Exercice à faire : faire l'opération 0000 0001 + 1111 1111, donner le résultat en binaire et sur 8 bits et l'état de l'indicateur C.

.....
.....
.....
.....
.....
.....
.....

Exemple : l'addition Hexadécimale (***cette représentation permet de condenser l'écriture en binaire***) sur 8 bits de $(1100\ 0101)_2 + (1110\ 0011)_2$, donne le résultat ci-dessous :

r	1	0x00
a		0xC5
b		0xE3
a+b		0xA8

La notation 0x indique que le nombre est codé en Hexadécimale.

On effectue l'opération comme suit :

5 + 3 = 8 (rang 16^0) et C + E (pour le rand 16^1)

soit 12 + 14 = 26 = 16 + 10 ce que donne A en hexadécimal et la retenue C = 1.

Le résultat s= a +b = 0xA8 et C=1 dépassement de capacité sur les entiers naturels.

Soit $10 \times 16^1 + 8 \times 16^0 = 168$;

comme C=1, le résultat est $168 + 16^2 = 168 + 256 = 424$.

Vérification : En base 10 :

a = 197, (c'est à dire $a = 2^0 + 2^2 + 2^6 + 2^7$), et b = 227, la somme s = a +b = 424.

Exercice à faire : Si on effectue la somme s = a +b avec a et b codés sur 16 bits. Quel est l'état de C ? Justifiez votre réponse.

.....
.....
.....
.....
.....
.....
.....
.....

5. Opérations sur les entiers_Addition en code complément à 2 (CC2) :

Soit 2 nombres a et b représentés en code complément à 2, le tableau ci-dessous donne les différents cas possibles de l'addition de 2 nombres entiers a et b en CC2 sur n bits et l'état de l'indicateur V (dépassement de capacité, oVerflow).

V = 1 si le résultat de la somme de a et b ne peut pas être représenté en CC2 sur n bits.

a, b sont	r_{n-1}	r_n	s_{n-1}	V	Si le résultat de l'addition est :
> 0	0	0	0	0	≥ 0 s est représentable sur n bits
	1	0	**1**	**1**	**< 0 s est non représentable sur n bits**
De Signe ≠	0	0	1	0	s est toujours représentable sur n bits
	1	1	0	0	
< 0	0	1	**0**	**1**	**≥ 0 s est non représentable sur n bits**
	1	1	1	0	< 0 s est présentable sur n bits

r_{n-1} : est le report de l'addition des bits de rang n-2

r_n : est le dernier report, il représente le report de l'addition des bits de rang n-1

s_{n-1} : est le bit de poids fort de la somme, **c'est-à-dire le bit de signe de la somme.**

- Le débordement dans le cas de l'addition standard est indiqué par l'indicateur C(arry)=1.
- Le débordement dans le cas de l'addition en CC2 est indiqué par l'indicateur V=1 (oVerflow).

Remarque : C=1 n'implique pas que V soit égal à 1.

Exemple :

r = report	0 0 0 0 1 1 1 0		1 0 0 0 1 1 1 0		1 0 0 0 1 1 1 0		0 0 0 0 1 1 1 0
a	1 0 0 0 0 1 0 1		0 1 0 0 0 1 0 1		1 1 0 0 0 1 0 1		0 0 0 0 0 1 0 1
b	1 0 1 0 0 0 1 1		0 1 1 0 0 0 1 1		0 1 1 0 0 0 1 1		0 0 1 0 0 0 1 1
a+b	0 0 1 0 1 0 0 0		1 0 1 0 1 0 0 0		0 0 1 0 1 0 0 0		0 0 1 0 1 0 0 0
	V=C=1 **débordement** $r_n=1$; $r_{n-1}=0$		**V=1 et C=0** **débordement** $r_n=0$; $r_{n-1}=1$		**V=0 et C=1** correct $r_n=1$; $r_{n-1}=1$		**V=C=0** correct $r_n=0$; $r_{n-1}=0$

Cet exemple vérifie que si $r_n = r_n\text{-}1$ *alors il n'y a pas de débordement et la somme est représentable en CC2 sur 8 bits.*

Exercice à faire : illustrer les cas précédents par des exemples d'opérations en binaire sur 16 bits.

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

Soustraction par addition en CC2 :

On peut calculer la soustraction de a - b en effectuant l'opération :

A + C1(b) + 1 = a + C2(b).

A ce stade vous êtes capable de :

A ce stade vous êtes capable de :

- poser et effectuer une opération d'addition en binaire ou en hexadécimal de 2 nombres x et y codés sur n bits. x, y ∈ N (entiers naturels) et x, y ∈ Z (entiers).
- Donner le résultat sur n bits et l'état des indicateurs Z, N, C et V.
- Expliquer dans quels cas V = 1
- Trouver le résultat correct dans le cas d'un débordement C = 1 (entiers naturels)
- Trouver le résultat correct dans le cas d'un débordement sur les entiers, V = 1
- Calculer la soustraction de a – b en effectuant l'opération a + C2(b).

6. Multiplication par une puissance de 2 (2^k) :

La multiplication d'un entier a codé sur n bits par la suite de bits (a_{n-1}, …, a_0) par 2^k (k > 0) est obtenue par un **décalage de cette suite de bits de k positions vers la gauche**.

Les k bits de plus faible poids sont remplacés par des 0.

Si le résultat n'est représentable, un débordement est généré.

Exemple : soit l'opération 10*4 = 40. Cette opération réalisée sur 8 bits donne :

10 = 0000 1010 ; $4 = 2^2$

0000 1010 décalé à gauche de 2 postions donne : 0010 1000 soit $2^3 + 2^5 = 8 + 32 = 40$.

Exercice à faire : à quoi correspond l'opération 16*3.

.....
.....
.....
.....
.....
.....
.....
.....

7. Division par une puissance de 2 (2^k) :

La division d'un entier a codé sur n bits par la suite de bits (a_{n-1}, …, a_0) par 2^k (k > 0) est obtenu par un **décalage de cette suite de bits de k positions vers la droite**.

Cas 1 : le bit de signe a_{n-1} est recopié dans les k bits de poids fort. Ce type **de décalage est dit arithmétique**.

Cas 2 : Dans le **décalage dit logique**, les k positions de poids fort sont remplies de 0.

Exemple : soit à réaliser l'opération 20 / 2 = 10. On a 20 = 16 +4 = 0001 0100 ;

0001 0100 décalé à droite d'une position donne 0000 1010 soit 10.

Exercice à faire : à quoi correspond l'opération 30 / 2 = 15 :

.....
.....
.....
.....
.....
.....

A ce stade vous êtes capable de :

- Effectuer des multiplications ou divisions par puissance de 2 en appliquant des opérations de décalage de bits.