TD2 : Entiers naturels et relatifs, addition et soustraction

Objectif(s)

- ✯ Représentation et additions d'entiers naturels et relatifs
- ✯ Réalisation d'un opérateur d'addition

Note pédagogique:
Dans cette UE, la notation binaire (resp. hexadécimale) avec suffixe b (resp. h) indique que le nombre est écrit en base 2 (resp. 16) tandis que la notation avec préfixe `0b` (resp. `0x`) indique la valeur d'un mot binaire.

Un mot binaire a une taille fixe et connue et sa valeur est indépendante d'une valeur numérique : elle précise uniquement la valeur de chacun des bits composant le mot.

Ainsi, un même mot peut être interprété de plusieurs manières différentes (selon codage entiers naturels, entiers relatifs, caractères, instruction mips, etc.). Un nombre, quelque soit sa base d'écriture, n'est pas limité en taille (nombre de chiffres) et ne peut pas "déborder" lors d'une opération.

Exercice(s)

Exercice 1 – Opérations arithmétiques avec des naturels

Question 1
Soit l'addition suivante de deux entiers naturels, quelle doit être la taille minimale du mot stockant le résultat ?

$$\begin{array}{rr} & 5B_h \\ + & 17_h \\ \hline = & \end{array}$$

Solution:

$$\text{On peut passer en base 2 :} \quad \begin{array}{rr} & 00010111_b \\ + & 01011011_b \\ = & 01110010_b \end{array}$$

ou directement en base 16 : $17_h + 5B_h = 72_h$. En binaire, le mot doit contenir au moins 7 bits.

Question 2
Même question pour la somme des entiers naturels 71_h et $B5_h$.

Solution:

$$\begin{array}{rr} & 01110001_b \\ + & 10110101_b \\ = & 100100110_b \end{array}$$

De même en base 16 : $71_h + B5_h = 126_h$: il faut un bit supplémentaire pour la retenue, soit 9 bits.

Question 3
Faites les opérations sur les mots suivants en indiquant s'il y a ou non dépassement de capacité sur entiers naturels. Les deux premières opérations sont sur 8 bits et la dernière sur 16 bits.

0x2B + 0x95 =
0xC8 + 0x6D =
0x8B57 + 0xA34F =

Solution:
0x2B + 0x95 = 0xC0
0xC8 + 0x6D = 0x35 ($C8_h + 6D_h = 135_h$) dépassement de capacité ⇒ le mot résultat interprété comme un entier naturel ne vaut pas le résultat de l'addition des 2 opérandes interprétées comme des entiers naturels (codage entier naturel)
0x8B57 + 0xA34F = 0x2EA6 ($8B57_h + A34F_h = 12EA6_h$) dépassement de capacité ⇒ le mot résultat interprété comme un entier naturel ne vaut pas le résultat de l'addition des 2 opérandes interprétées comme des entiers naturels (codage entier naturel)

Question 4
Enoncez une règle permettant de détecter un dépassement de capacité lors de l'addition d'entiers naturels sur n bits.
Solution:
Cela a été vu en cours.
Pour détecter le dépassement de capacité sur entiers naturels, il suffit de regarder s'il y a une retenue sortante : il y a génération d'une retenue sortante quand l'addition des bits de poids le plus fort (rang $n-1$) génère une retenue. Dans l'exemple précédent, il y a un dépassement de capacité à la deuxième opération car il faut 9 bits pour stocker le résultat, et à la troisième car il faut 17 bits pour stocker le résultat.

Exercice 2 – Construction d'un additionneur n bits

Question 1
Combien faut-il d'entrées et de sorties pour réaliser un étage d'une addition ?
Solution:
3 entrées : a_i, b_i et la retenue entrante de rang i, c_{in_i}. 2 sorties : la somme s_i et la retenue sortante de rang i, c_{out_i}.

Question 2
Donnez la table de vérité d'un additionneur 1 bit.
Solution:
Cet exemple a été fait en cours et donné dans les transparents de cours.

a	b	cin	$cout$	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Question 3
Donnez une expression booléenne pour cout et s.
Solution:
Application du TD1, avec la forme normale disjonctive associée à cout et s :
$s = \overline{a}.\overline{b}.cin + \overline{a}.b.\overline{cin} + a.\overline{b}.\overline{cin} + a.b.cin$

Chercher à simplifier, à utiliser des XOR en factorisant cin et $\overline{cin}$.
$cout = \overline{a}.b.cin + a.\overline{b}.cin + a.b.\overline{cin} + a.b.cin$ Idem, chercher à réduirer l'expression.

Question 4
Comment réaliser un additionneur n bits à partir de n additionneurs 1 bit ?
Solution:
Cela a été vu en cours et le schéma donné dans les transparents du cours.
Il faut mettre les additionnneurs en parallèle en chaînant les retenues. La retenue sortante du rang i va sur la retenue entrante du rang $i + 1$. Le rang i correspond au bit de rang i des nombres à additionner. La retenue entrante du rang 0 est mise à 0.

Exercice 3 – Représentation des entiers relatifs

La représentation binaire des entiers relatifs est utilisée sur des écritures de nombres de **longueur donnée** (nombres écrits couramment sur 8, 16, 32 ou 64 bits). Dans une telle écriture on utilise le bit de poids fort (bit le plus à gauche) du nombre pour stocker la valeur du coefficient négatif (-2^{n-1} sur n bits).

Question 1
Les première et troisième colonnes du tableau ci-dessous contiennent tous les mots binaires de 3 bits. Donnez pour chaque mot la valeur décimale de son interprétation selon le codage *entiers relatifs en complément à 2*.

Mot binaire de 3 bits	Valeur en décimal selon codage entiers relatifs	Mot binaire de 3 bits	Valeur en décimal selon codage entiers relatifs
0b000		0b100	
0b001		0b101	
0b010		0b110	
0b011		0b111	

Quel est l'intervalle des valeurs représentables sur un mot de n bits par le codage *entiers relatifs* ?
Quand peut-on avoir un dépassement de capacité lors d'une addition de deux nombres relatifs et quand est-on sur qu'il n'y aura pas de dépassement de capacité ?
Solution:
Rappel : voir le cours pour des exemples de représentation en complément à deux.

Mot binaire de 3 bits	Valeur en décimal selon codage entiers relatifs	Mot binaire de 3 bits	Valeur en décimal selon codage entiers relatifs
0b000	0	0b100	-4
0b001	1	0b101	-3
0b010	2	0b110	-2
0b011	3	0b111	-1

Cela a été vu en cours avec explication dans les transparents de l'intervalle de représention.

Dans cette représentation, le bit de signe est pondéré, ce qui introduit une dissymétrie dans les intervalles des valeurs représentées. Sur n bits, le codage en entiers relatifs par complément à 2 permet de représenter l'intervalle $[-2^{n-1}; 2^{n-1} - 1]$.

On retrouve les problèmes de débordement de capacité lorsque l'on somme deux nombres positifs avec un résultat supérieur à $2^{n-1} - 1$, et symétriquement, lorsque l'on somme deux nombres négatifs avec un résultat inférieur à -2^{n-1}. Si l'on somme deux nombres de signe différent, on reste forcément dans l'intervalle de valeur, on ne peut donc pas avoir de dépassement de capacité.

Question 2

Chacun des mots de 16 bits suivants contient un entier relatif codé en complément à 2. Indiquez si celui-ci est positif ou négatif, puis calculez le mot contenant l'opposé, et écrivez-le en base 2 et en base 16.

Mot de 16 bits Écriture base 16	signe	Mot de 16 bits Écriture base 2	Opposé = CÀ2 Écriture base 2	Opposé = CÀ2 Écriture base 16
0x0B24				
0xABCD				
0xFFFF				
0x5A72				
0x0072				

Solution:

L'opposé d'un nombre N_b vaut $overline N_b + 1$. Cela a été vu en cours.

Mot de 16 bits Écriture base 16	signe	Mot de 16 bits Écriture base 2	Opposé = CÀ2 Écriture base 2	Opposé = CÀ2 Écriture base 16
0x0B24	+	0b0000 1011 0010 0100	0b1111 0100 1101 1100	0xF4DC
0xABCD	-	0b1010 1011 1100 1101	0b0101 0100 0011 0011	0x5433
0xFFFF	-	0b1111 1111 1111 1111	0b0000 0000 0000 0001	0x0001
0x5A72	+	0b0101 1010 0111 0010	0b1010 0101 1000 1110	0xA58E
0x0072	+	0b0000 0000 0111 0010	0b1111 1111 1000 1110	0xFF8E

À noter qu'il existe une façon simple pour calculer le complément à 2 : partir de la droite et recopier tous les bits jusqu'au premier 1 (y compris le premier 1), puis inverser tous les bits à gauche du premier 1.

Exercice 4 – Opérations arithmétiques

Soit un additionneur 8 bits, muni en plus de la sortie résultat sur 8 bits, de deux sorties `Cout` et `Ov`, telles que `Cout` = $\texttt{Cout}_{n-1}$ = $\texttt{Cout}_7$ et `Ov` = $\texttt{Cout}_{n-1} \oplus \texttt{Cout}_{n-2} = \texttt{Cout}_7 \oplus \texttt{Cout}_6$.

Question 1

Soit l'addition binaire suivante : $0b01101001 + 0b10000000 = 0b11101001$. Quelle opération réalise-t-on en décimal

dans le cas où les opérandes et le résultat sont interprétés comme des entiers naturels ? Le résultat théorique de l'opération est-il codable sur 8 bits ? Quelles sont les valeurs des bits `Cout` et `Ov` ? Même question dans le cas où les opérandes et le résultat sont interprétés comme des entiers relatifs en complément à 2.

Solution:

Si les opérandes et le résultat sont des entiers naturels (l'additionneur n'a pas de moyen de le savoir), on effectue l'opération $105 + 128 = 233$. Le résultat de l'additionneur est donc bien le résultat théorique de l'addition. En effet, il est bien codable sur 8 bits car il appartient à l'intervalle $[0; 255]$. Les bits `Cout` et `Ov` valent tous les deux 0.
Si les opérandes et le résultat sont des entiers relatifs, on effectue l'opération $105 + (-128) = -23$. Encore une fois, le résultat est conforme au résultat théorique, car il est bien codable sur 8 bits : il appartient à l'intervalle $[-128; 127]$.

Par ailleurs, le fait que le bit `Cout` vaille 0 est à lier avec le fait que le résultat sur entiers naturels est valide, et le fait que le bit `Ov` vaille 0 est à lier avec le fait que le résultat sur entiers relatifs est valide.

Question 2

Soit l'opération $0xE2 + 0x9C$. Calculer le résultat. Quelle opération effectue-t-on si les opérandes sont interprétés des entiers naturels ? Le résultat est-il valide ? Mêmes questions si les opérandes sont vus comme des relatifs. Quelles sont les valeurs des bits `Cout` et `Ov` ?

Solution:

Le résultat de l'opération est 0x7E (sur 8 bits). En naturel, on effectue $226 + 156 = 126$: le résultat est invalide. Cela est visible au niveau de l'additionneur car le bit `Cout` est à 1. En entiers relatifs, on effectue l'opération $-30 + (-100) = 126$. Le résultat est invalide car le résultat théorique (-130) n'est non codable sur 8 bits. On remarque que le bit `Ov` est à 1.

Question 3

Soit l'opération sur entiers naturels $136 + 250$. Quels sont les valeurs des mots en entrée et en sortie de l'additionneur (en binaire ou en hexa) ? Le résultat est-il valide, et pourquoi ? Si maintenant les mêmes valeurs en entrée de l'additionneur sont interprétées comme des entiers relatifs, que vaut le résultat ? Est-il valide ?

Solution:

Au niveau de l'additionneur, on effectue l'opération $0x88+0xFA = 0x82$. Le résultat est invalide car non codable sur 8 bits (bit `Cout` à 1). Si ces entrées sont considérées comme des entiers relatifs, on effectue l'opération $-120+(-6) = -126$. Ce résultat est en revanche valide car il est égal au résultat théorique, qui appartient à l'intervalle des valeurs représentables (bit `Ov` à 0).

Question 4

Trouver des valeurs d'opérandes en entrée de l'additionneur qui produisent un résultat valide s'ils sont interprétés comme entiers naturels et invalide s'ils sont interprétés comme entiers relatifs.

Solution:

Il faut que `Cout` = 0 (pas de dépassement dans le cas d'une interprétation comme entiers naturels), mais qu'il y ait une retenue au rang $n - 2$ (pour qu'il y ait un dépassement lors d'un interprétation comme entiers relatifs). Les 2 bits de poids fort des nombres doivent donc être 0 pour que `Cout` = 0.

En entier naturels, il faut donc deux nombres inférieurs à 128 qui produisent un résultat supérieur ou égal à 128 pour avoir une retenur au rang $n-2$ (ex : 70 + 70 = 140, soit 01000110 + 01000110 = 10001100). En relatif, on ajoute 2 nombres positifs et on obtient un résultat négatif (en l'occurrence 70 + 70 = -116). On remarque que le bit `Cout` vaut 0 et que le bit `Ov` vaut 1.

Exercice 5 – Extension d'entiers naturels et relatifs

Question 1

Soit le mot hexadécimal 0xFF sur 8 bits. Donnez sa valeur en décimal dans le cas d'un codage entier naturel et dans

le cas d'un codage entier relatif (en complément à 2). Étendez ce nombre sur 32 bits. Obtenez-vous la même chose si l'on considère que c'est un entier naturel ou un entier relatif ? En déduire les règles d'extension d'un entier naturel et d'un entier relatif.

Solution:

Entier naturel : 0xFF vaut 255 et sur 32 bits 0x000000FF

Entier relatif : 0xFF vaut -1 et sur 32 bits 0xFFFFFFFF

Noter la différence pour l'extension car celle-ci réapparaitra en langage d'assemblage : l'extension d'un entier naturel se fait en ajoutant des 0 sur les bits supplémentaires alors que pour un entier relatif on étend le bit de signe sur les bits de poids forts supplémentaires.