Université Blaise Pascal

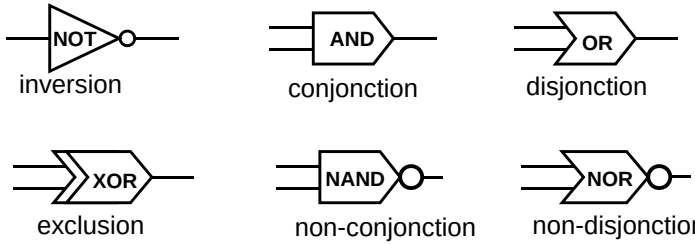
UFR Sciences et Technologies

L1 : Découverte de l'informatique
1ère partie : de l'électronique à l'informatique

(2) Représentation interne des informations

Représentation binaire

- Unités de référence : bit (*binary digit*, **b**) : 0 ou 1 (vrai ou faux)
- correspond à l'organisation microscopique du processeur :



Principales portes logiques (opérateurs booléens)

→ Logique booléenne
→ Synthèse (conception) et analyse (vérification) de circuits intégrés } ch. (4)

- Processeur : ... 8b / 16b / 32b / 64b / 128b ... [bus, mémoires, ...]
- Représentation :
  - des instructions
  - des données
    - alpha-numériques : ASCII, ISO, Unicode (UTF), ...
      ✍ Définir, caractériser, indiquer où trouver les codes :
      EBCDIC, BCD, IEC-10646, ISO pour la langue turc
    - numériques
      → base de numération (binaire, octal, décimal, hexadécimal)

exemple

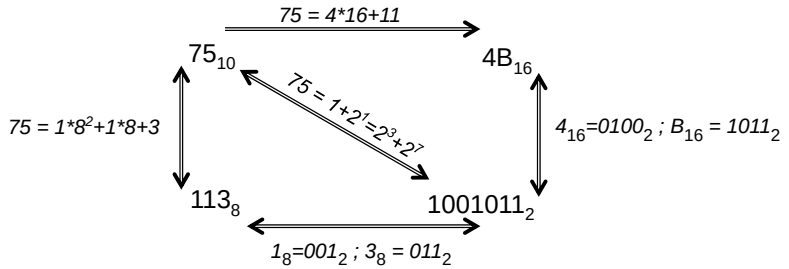
"microcode" 32b :	00100001 00100010
mnémonique ASM* :	ADD reg1 32
interprétation :	ajouter au registre 1 la valeur 32

* langage assembleur

Bases de numération utilisées

- Décimal (base 10)
- Hexadécimal (base 16)
- Octal (base 8)
- Binaire (base 2)

décimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	...
hexadécimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	10	...
octal	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17	20	21	22	23	224	...
binaire	0	1	10	11	100	101	110	111	1000	1001	1010	1011	1100	1101	1110	1111	10000	10001	10010	10011	10100	...



Représentations des données numériques

- nombres entiers naturels
  - BCD
  - binaire
- entiers relatifs
  - Ajout d'un bit de signe
  - Translation de code (*"représentation biaisée"*)
  - Par complément logique : complément à 1
  - Par complément à 2 : [-2n-1 .. +2n-1-1]
- décimaux et fractionnaires
  - → représentation binaire des fractions
- nombres réels
  - ← écriture scientifique des réels

Entiers naturels : code BCD

4 bits utiles pour coder les chiffres (codage alpha-numérique sur 6 bits)

nombre à représenter	0	1	2	3	4	5	6	7	9	codage
valeur du code	0	1	2	3	4	5	6	7	8	alphanum
écriture binaire	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

Addition : gérer le déficit binaire (16-10=6)

addition	résultat attendu	code binaire	résultat binaire	décodage	commentaire
2+3	5	0010+0011	0101	+5	
8+6	14	1000+0110	1110	'E'	dépassement de code à corriger : 'E'-10=4 avec 1 de retenue → 0001 0100

Entiers naturels : code binaire en champ fixe
n bits → $[0..2^n-1]$

Exemple avec un codage sur 4 bits

nombre à représenter	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
écriture binaire	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111

Addition : gérer les dépassement de capacité

addition	résultat attendu	code binaire	résultat binaire	décodage	commentaire
2+3	5	0010+0011	0101	+5	
12+6	18	1100+0110	(1) 0010	'E'	dépassement de capacité : retenue à prendre en compte

🕷 → ☠ : *"il y a une erreur, je m'arrête brutalement", ou*
🕷 → 😐 : *"je donne le résultat sans me préoccuper de son sens", ou*
🕷 → 🏳☹ : *"je signale simplement qu'il y a une erreur", ou*
🕷 → 🏳🚑 : *"je signale l'erreur et je donne le résultat pour ce qu'il vaut"*

Entiers naturels : autres codes

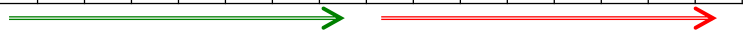
✍ Définition du code, intérêt
- Code excédent-3
- Code 2 dans 5
- Code biquinaire

→ Détection et correction d'erreurs } ch. (3)

Entiers relatifs par ajout de signe

Exemple avec un codage sur 4 bits

nombre à représenter	0	1	2	3	4	5	6	7	**-0**	**-1**	**-2**	**-3**	**-4**	**-5**	**-6**	**-7**
valeur du code	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
écriture binaire	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111



Addition : gérer le signe

Entiers relatifs par translation de code (codage biaisé à 2^{n-1}-1)

Exemple avec un codage sur 4 bits (biais à 15)

nombre à représenter	**-8**	**-7**	**-6**	**-5**	**-4**	**-3**	**-2**	**-1**	0	1	2	3	4	5	6	7
valeur du code	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
écriture binaire	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111

Codage de l'entier E sur n bit :
1. vérifier que $E \in [-2^{n-1}..+2^{n-1}-1]$
2. $c=E+2^{n-1}$
3. Ecrire c en binaire

Addition : translater le code obtenu (-2^{n-1})

Complément **à 1** : -p est codé par NON(p)

Exemple avec un codage sur 4 bits

nombre à représenter	**0**	1	2	3	4	5	6	7	**-7**	**-6**	**-5**	**-4**	**-3**	**-2**	**-1**	**-0**
valeur du code	0	1	2	3	4	5	6	7	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
écriture binaire	0000	0001	0010	0011	0100	0101	0110	0111	**1000**	**1001**	**1010**	**1011**	**1100**	**1101**	**1110**	**1111**

addition	résultat attendu	code binaire	résultat binaire	décodage	commentaire
+2+3	+5	0010+0011	0101	+5	
+4+6	+10	0100+0110	1010	**- 5**	dépassement de capacité (intrinsèque)
+6+(-4)	+2	0110+1011	(1)0001	**+1**	erreur et retenue incodable
+4+(-6)	-2	0100+1001	1101	+2	
+6+(-6)	+0	0110+1001	1111	**- 0**	double codage du zéro
(-2)+(-3)	-5	1101+1100	(1)1001	**- 6**	erreur et retenue incodable

→ Remplacer 1111 par 0000 quand nécessaire
→ Ajouter 1 au résultat dès qu'il y a une retenue binaire
→ Codage dans [- 7 .. +7]

Complément **à 2** : -p est codé par 1+NON(p)

n bits → $[-2^{n-1} .. +2^{n-1}-1]$

Exemple avec un codage sur 4 bits

nombre à représenter	**0**	1	2	3	4	5	6	7	**-8**	**-7**	**-6**	**-5**	**-4**	**-3**	**-2**	**-1**
valeur du code	0	1	2	3	4	5	6	7	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
écriture binaire	0000	0001	0010	0011	0100	0101	0110	0111	**1000**	**1001**	**1010**	**1011**	**1100**	**1101**	**1110**	**1111**

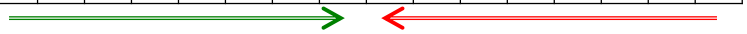
Codage futé en C2 d'un entier relatif E :
1. C est la valeur absolue de E écrite en binaire
2. Si E est négatif, inverser (0/1) tous les bits de C à gauche du 1er 1 en partant de la droite.

Complément **à 2** : -p est codé par 1+NON(p)

n bits → $[-2^{n-1} .. +2^{n-1}-1]$

Exemple avec un codage sur 4 bits

nombre à représenter	**0**	1	2	3	4	5	6	7	**-8**	**-7**	**-6**	**-5**	**-4**	**-3**	**-2**	**-1**
valeur du code	0	1	2	3	4	5	6	7	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
écriture binaire	0000	0001	0010	0011	0100	0101	0110	0111	**1000**	**1001**	**1010**	**1011**	**1100**	**1101**	**1110**	**1111**



addition	résultat attendu	code binaire	résultat binaire	décodage	commentaire
+2+3	+5	0010+0011	0101	+5	
+4+6	+10	0100+0110	1010	**- 5**	dépassement de capacité (intrinsèque)
+6+(-4)	+2	0110+1100	(1) 0010	**+2**	
+4+(-6)	-2	0100+1010	1110	- 2	
+6+(-6)	+0	0110+1010	(1)000	**0**	
(-2)+(-3)	-5	1110+1101	(1) 1011	**- 5**	

→ Les retenues sont ignorées ☺

→ Codage dans [- 8 .. +7]

Nombres réels : "Floats"

- Sur la base de l'écriture scientifique : mantisse*$10^{exposant}$:

  x=m*$\mathbf{2^e}$ avec e entier relatif et ½<=|m|<1

- Codage en champ fixe avec n bits pour la mantisse et p bit pour l'exposant :

  - la mantisse est généralement codé par bit de signe + valeur absolue

  - l'exposant peut être en complément à deux ou translaté (biaisé)

Addition :
- dénormaliser la plus petite valeur
- additionner les mantisses
- renormaliser si nécessaire

Multiplication :
- multiplier les mantisses
- additionner les exposants
- renormaliser si nécessaire

Soustraction : *même principe*

Division : *même principe*

Nombres réels : le standard IEEE-754 (version 1985)

- Simple précision : 32b = mantisse* (1b de signe**, valeur sur 23b)
  et exposant 8b (biaisé à 127)

- Double précision : 64b = mantisse* (1b de signe**, valeur sur 52b)
  et exposant 11b (biaisé à 1023)

- Précision étendue : 80b

* Mantisse avec bit caché : on omet le premier bit qui est forcément à 1

** Deux conventions :
- écriture "naturelle" (bit de signe avec la mantisse)
- écriture IEEE (bit de signe de la mantisse déporté)