Architecture de l'ordinateur
et
Systèmes de numération

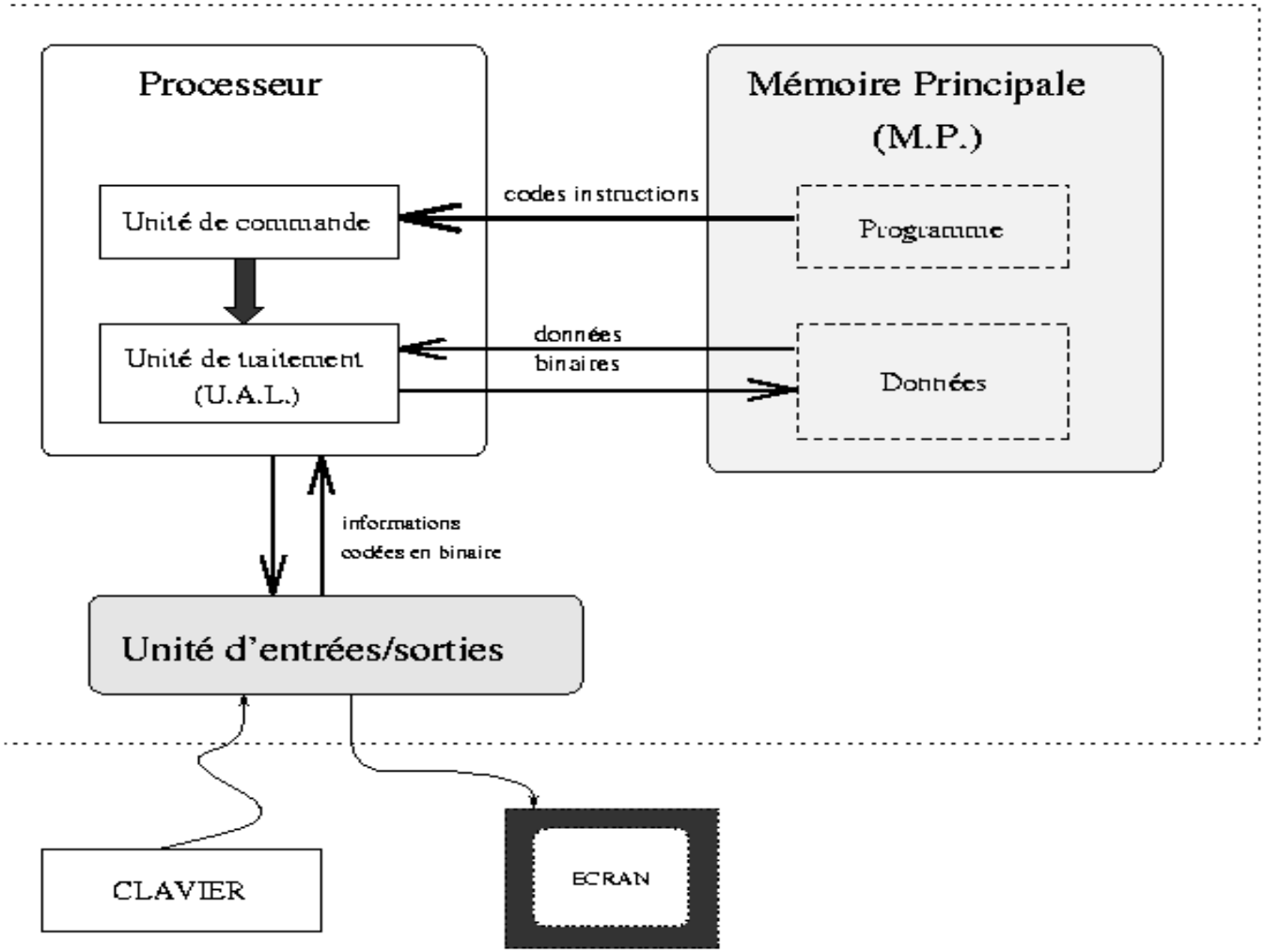
Plan

- Principe de codage du nombres (La mémoire)
- Codage et transcodage de nombres naturels (base 10 en base 2, 8 et 16)
- Opérations arithmétiques sur les nombres binaires
- Codage de nombres relatifs
- Codage de nombres réels
- Exercices d'applications
- Le fonctionnement du processeur : Vue globale

codage de l'information ou la numérisation

- Les informations traitées par un ordinateur peuvent être de différents types : texte, nombres, images, son, vidéo, programme, etc.
- Ces données sont toujours représentées et manipulées par l'ordinateur sous forme binaire, une suite de 0 et de 1 ;
- A quoi consiste les 0 et 1 dans une machine qui ne travaille et qui ne connait que l'électricité ?

Architecture de l'ordinateur



La mémoire

- Types : RAM, DD, Registres, …
- Tout ce qui est enregistré dans la mémoire est de type binaire (Seulement 2 états)
- Pour comprendre le fonctionnement de la mémoire il faut comprendre le fonctionnement de binaire

La mémoire

- Exemple :
- Pour enregistrer le nombre 40 en base 10 dans la mémoire sous forme de nombre binaire : 40 (10) = 101000 (2)
- Ce nombre binaire est constitué de 6 chiffres
- L'unité de mesure : Bit (Binary digIT = chiffre binaire)
- Donc on a besoin de 6 bits pour représenter 101000 en mémoire

La mémoire

- Tout est enregistré au moyens de bascules.
- Une bascule :
  - Chargée = 1
  - Non chargée = 0
- On a besoin de 6 bascules pour enregistrer les 6 bits du nombre binaire : 101000
- L'ordinateur va charger la 4ème et la 6ème bascule en partant de la droite et laisse les autres non chargées

La mémoire

- La mémoire est constituée des emplacements mémoire
- Chaque emplacement mémoire (case mémoire) possède une adresse et contient 8 bascules
- Dans une case mémoire ,on ne peut stocker que 8 bits = 1 octet (Byte)
- Si Donnée < 1 octet → on met 0 à gauche
- Si Donnée > 1 octet → utiliser les emplacements mémoire consécutives (donnée sera représentée sur un mot mémoire)

La mémoire : Unités de mesures

Nombre de bits	Appellation (Fr)	Appellation (En)
2^0=1 bit	bit « unité de base)	Bit
2^2=4 bits	quartet	nibble
2^3=8 bits	Octet (o)	Byte (B)
2^4=16 bits	Mot	word
2^5=32 bits	double mot	double word / dword
2^8=64 bits	quadruple mot	qword
10^3 bits = 1000 bits	Kilobit (Kb ou Kbit)	KiloBit(Kb)
10^3 octets = 8000 bits	Kilo-octet (Ko)	Kilobyte (KB)
1 Mo = 10^3 Ko = 10^6 octets	Mégaoctet (Mo)	Megabyte (MB)
10^9 octets	Gigaoctet (Go)	Gigabyte (GB)
10^{12} octets	Téra-octet (To)	Terabyte (TB)
10^{15} octets	Péta-octet (Po)	Petabyte (PB)
10^{18} octets	Exa-octet (Eo)	Exabyte (EB)
10^{21} octets	Zétta-octet (Zo)	Zettabyte (ZB)
10^{24} octets	Yotta-octet (Yo)	Yottabyte (YB)

Le codage d'une information

- Le codage d'une information consiste à établir une correspondance entre la représentation externe (habituelle) de l'information et sa représentation interne dans la machine, qui est une suite de bits 0 et 1 suivant un ensemble de règles précises.

- *Exemple :* 100011 est la représentation interne de 35

Le codage **de nombres naturels**

- On utilise la base 10 pour représenter les nombres naturels $\mathbb{N}$ = {0, 1, 2, 3, 4, 5, 6, 7, ...}
- Une base : c'est un regroupement de symboles différents qui va nous permettre de représenter des nombres différents. Dans une base b, on utilise b chiffres/alphabets.

Le codage **de nombres naturels**

- En décimal, b=10, a_i€ {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
- En binaire, b=2, a_i€ {0, 1}
- En octale, b=8, a_i€ {0, 1, 2, 3, 4, 5, 6, 7},
- En Hexadécimal, b=16,
  $a_i \in$ {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F},

On utilise les 6 premières lettres comme des chiffres ($A_{(16)}=10_{(10)}$, $B_{(16)}=11_{(10)}$, $C_{(16)}=12_{(10)}$, $D_{(16)}=13_{(10)}$, $E_{(16)}=14_{(10)}$, $F_{(16)}=15_{(10)}$),

Le codage **de nombres naturels**

- Autres bases :
- 1 à 9 : unaire (1), binaire (2), trinaire(3), quaternaire (4), quinaire (5), sénaire (6), septénaire (7), octal (8), nonaire (9)
- 10 à 60 : Décimal (10), duodécimal (12), tridécimal (13), hexadécimal (16), vicésimal (20), base 36, sexagésimal (60)

Nombre naturel : De base 10 à base b

- La règle à suivre est les divisions successives :
  - On divise le nombre par la base b,
  - Puis le quotient par la base b,
  - Ainsi de suite jusqu'à l'obtention d'un quotient nul,
  - La suite des restes correspond aux symboles de la base visée.
  - On obtient en premier le chiffre de poids faible et en dernier le chiffre de poids fort.

Nombre naturel : De base b à base 10

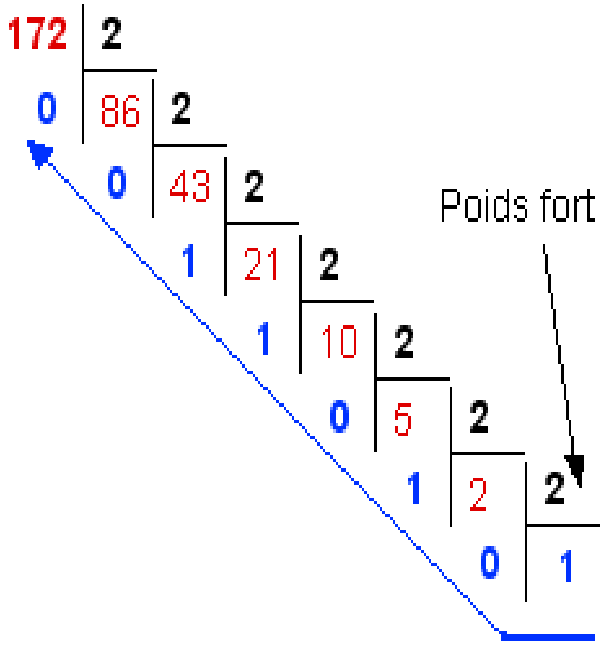
- Soit X = $a_{n-1}\ a_{n-2} \ldots \ldots \ldots a_1 a_{0\,B}$ avec $a_i \in \{0..B-1\}$ une représentation de X en base B sur n chiffres.
- La valeur de X en base 10 est :

$$N_{10} = \sum_{i=0}^{n-1} a_i\, B^i$$

$$N_{10} = a_{n-1} B^{n-1} + \ldots\ldots + a_1 B^1 + a_0\, B^0$$

Nombre naturel : De base 10 à base 2

- Exemple : $(172)_{10}= (?)_2$



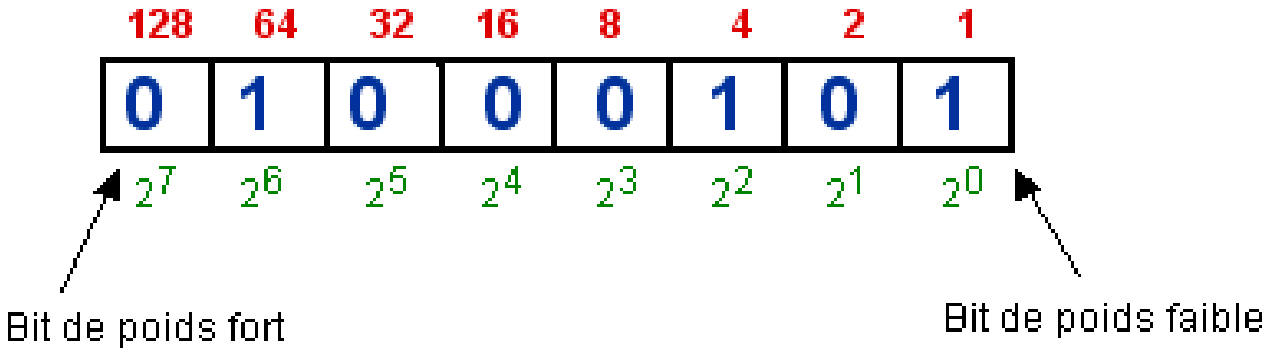
- Donc $(172)_{10}= (10101100)_2$

Vérification :

$(10101100)_2 = 1 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = 172_{(10)}$

Nombre naturel : De base 10 à base 2

- Description d'un octet :

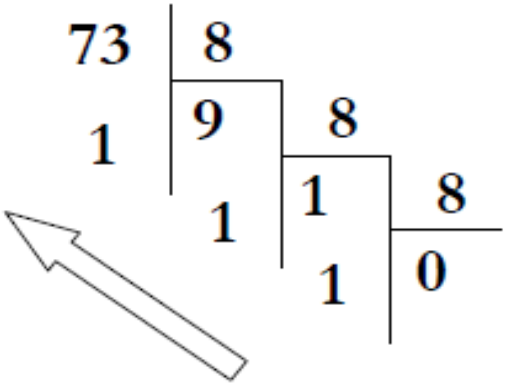


- *Exercice :*

Ecrire sur un octet et sur un double mot le nombre $10_{(10)}$ en code binaire ?

Nombre naturel : De base 10 à base 8

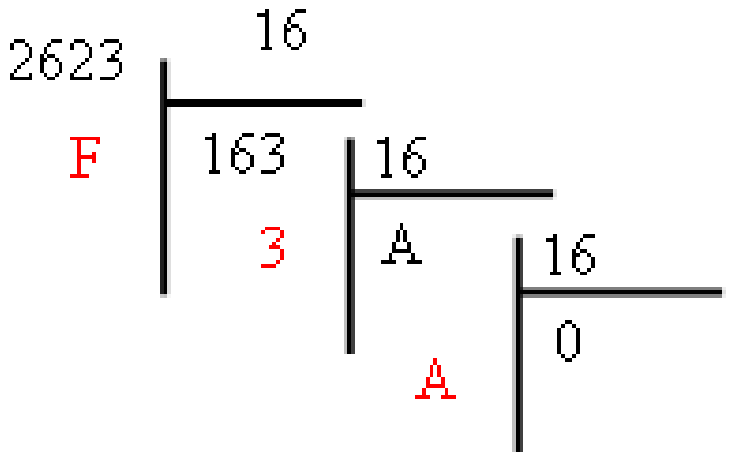
- Soit N le nombre d'étudiants d'une classe représenté en base décimale par : $N = 73_{(10)}$
- Représentation en Octale?



- $73_{(10)} = 111_{(8)}$
- Vérification

Nombre naturel : De base 10 à base 16

- $N_{10} = 2623_{10}$ en Hexadécimal (base 16).



2623 / 16 = 163, il reste 15...

Conversion de binaire vers une base b

- Deux solutions existent pour convertir un nombre binaire vers un nombre en base b :
- **Première solution :**

Convertir vers la base décimale puis convertir ce nombre vers la base b par division euclidienne.

Exemple :

$$10010_{(2)} = ?_{(8)}$$

$$10010_{(2)} = 2^4+2_{(10)}=18_{(10)}=2*8^1+2*8^0{}_{(10)}=22_{(8)}$$

- **Deuxième solution :**

Regroupement de bits et conversion directe : Il suffit de faire regrouper les bits en sous-ensembles de trois bits (base 8) ou quatre bits (base 16) puis remplacer chaque groupe par le symbole correspondant dans cette base.

Conversion de binaire vers une base 8

- regroupement des bits en sous-ensembles de trois bits puis remplacer chaque groupe par le symbole correspondant dans la base 8.
- Exemple :

$N = 001010011101_{(2)}$

Correspondance Octale \Binaire	
Symbole Octale	Suite binaire
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Conversion de binaire vers une base 16

- regroupement des bits en sous-ensembles de quatre bits (quartet) puis remplacer chaque groupe par le symbole correspondant dans la base 16.

Correspondance Hexadécimal \Binaire			
Symbole Hexadécimal	Suite binaire	Symbole Hexadécimal	Suite binaire
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

- Exemple :

$N = 001010011101_{(2)}$

Opérations arithmétiques : Addition

- L'addition en binaire s'effectue de la même manière qu'en décimal en appliquant le principe de la retenue;

  0 + 0 = 0 ,

  0 + 1 = 1 ,

  1 + 0 = 1 ,

  1 + 1 = 0 (avec retenue 1)

En décimal	En binaire
1	1
19	0 0 0 1 0 0 1 1
+ 2	+ 0 0 0 0 0 0 1 0
21	0 0 0 1 0 1 0 1
33	1 1
+ 33	0 0 1 0 0 0 0 1
66	+ 0 0 1 0 0 0 0 1
	0 1 0 0 0 0 1 0

Opérations arithmétiques : Soustraction

- La soustraction en binaire s'effectue de la même manière qu'en décimal en appliquant le principe de la retenue;

  0 - 0 = 0 ,

  0- 1 = 0 (emprunt de 1) ,

  1 - 0 = 1 ,

  1 - 1 = 0

En décimal	En binaire
22 -12 10	0 1 1 0 1 1 0 - 1 1 0 0 0 1 0 1 0
83 - 7 76	0 1 1 1 0 1 0 0 1 1 - 1 1 1 1 0 0 1 1 0 0

Opérations arithmétiques : Produit

- L'addition en binaire s'effectue de la même manière qu'en décimal.

  0 x 0 = 0 ,

  0 x 1 = 0 ,

  1 x 0 = 0 ,

  1 x 1 = 1

En décimal	En binaire
45 x 5 225	101101 x 101 101101 + 0000000 + 10110100 11100001

Opérations arithmétiques : Division

- La division binaire s'effectue à l'aide de soustractions, comme la division décimale, sauf que les digits du quotient ne peuvent être que 1 ou 0.
- Le bit du quotient est 1 si on peut soustraire le diviseur, sinon il est 0.
- Voici un exemple de la division du nombre :

$10010000111_{(2)}$ par $1011_{(2)} = 1101001_{(2)}$

et reste $100_{(2)}$,

C’est-à-dire 1159 / 11 = 105, reste 4.

Opérations arithmétiques : Division

1	0	0	1	0	0	0	0	1	1	1	1	0	1	1				
-	1	0	1	1							0	1	1	0	1	0	0	1
	0	1	1	1	0													
	-	1	0	1	1													
		0	0	1	1	0	0											
			-	1	0	1	1											
				0	0	0	1	1	1	1								
						-	1	0	1	1								
							0	1	0	0								

Exercices d'application 1

Compléter le tableau suivant :

Décimal	Binaire	Octal	Hexadécimal
1	00000001	001	001
10			
	01100100		
			065
		764	

Exercices d'application 1

100111
+ 100101
+ 001101
+ 110000
+ 111111

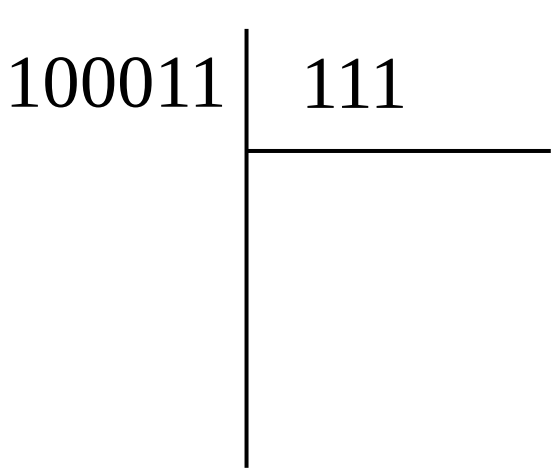
1BD
+ 789
+ DEF

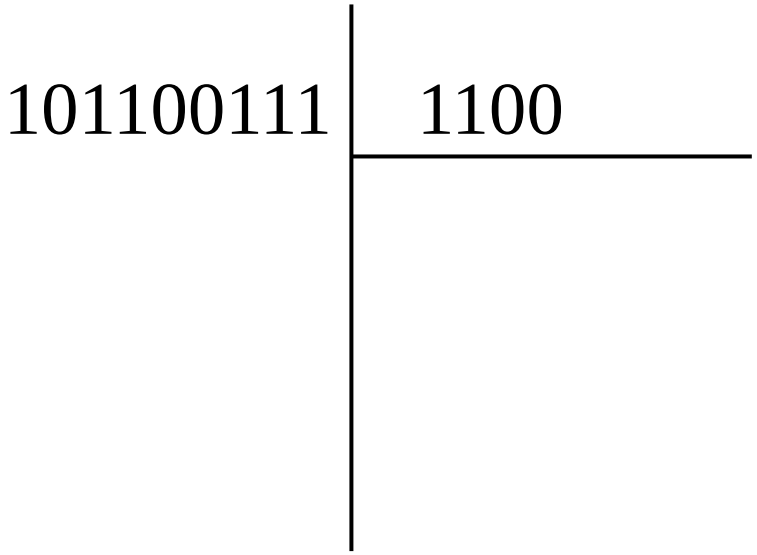
10111111
- 00001011
- 10010101

A9B1
x F310

11011
x 01001

Exercices d’application 1





Exercices d'application 1

- Ecrire un algorithme nomme CONVERSION2_10 permettant de saisir un entier exprimé en base binaire.
  Ensuite, il le convertit vers la base 10 puis affiche le résultat obtenu.

Transcodage : Relatifs - binaire

Les nombres relatifs sont l'ensemble des nombres positifs et négatifs « **Z** ».

Exemples :

- Un nombre positif : (+10) = 10 (le signe '+' n'est pas obligatoire à mettre)
- Un nombre négatif : (-18) (le signe '-' est obligatoire à mettre : sans ce signe ce nombre est positif)
- Un nombre à la fois positif et négatif = 0 (c'est le seul nombre à être positif et négatif)

Transcodage : Relatifs - binaire

Il existe au moins trois façons pour coder un nombre relatif:

1. Code binaire signé (signe et valeur absolue)
2. Code complément à 1
3. Code complément à 2 (Utilisé sur ordinateur)

Code binaire signé

Le bit le plus significatif (de poids le plus fort) est utilisé pour représenter le signe du nombre :

- Si le bit le plus fort = 1 alors nombre négatif
- Si le bit le plus fort = 0 alors nombre positif
- Les autres bits codent la valeur absolue du nombre

NB :

- **LSB**: Least Significant Bit (Bit de plus faible poids)
- **MSB**: Most Significant Bit (Bit de plus fort poids)

Code binaire signé

Exemple :

Sur 8 bits, codage des nombres -24 et -128 en (bs = binaire signé) :

-24 est codé en binaire signé par :

$$1\ 0\ 0\ 1\ 1\ 0\ 0\ 0_{(bs)}$$

-128 hors limite ➔ nécessite 9 bits au minimum

Code binaire signé

Etendu de codage :

Avec n bits, on code tous les nombres entre $-(2^{n-1}-1)$ et $(2^{n-1}-1)$

$^{n-1}$: on élimine un bit pour le signe

Exemple :

Avec 4 bits : 1 bit de signe et 3 bits pour coder tous les nombres entre -7 et +7

Limitations du binaire signé:

Deux représentations du zéro : + 0 et - 0

Exemple : Sur 4 bits : $+0 = 0000_{(bs)}$, $-0 = 1000_{(bs)}$

Code binaire signé

Exercice :

- Coder 100 et -100 en binaire signé sur 8 bits

$100_{(10)} = (01100100)_{(bs)}$

$-100_{(10)} = (11100100)_{(bs)}$

- Décoder en décimal $(11000111)_{(bs)}$ et $(00001111)_{(bs)}$

$(11000111)_{(bs)} = -71_{(10)}$

$(00001111)_{(bs)} = 15_{(10)}$

Complément à un (C1)

- Aussi appelé Complément Logique (CL) ou Complément Restreint (CR) :
  - Les nombres positifs sont codés de la même façon qu'en binaire pure.
  - Un nombre négatif est codé en inversant chaque bit de la représentation de sa valeur absolue
- Le bit le plus significatif est utilisé pour représenter le signe du nombre :
  - Si le bit le plus fort = 1 alors nombre négatif
  - Si le bit le plus fort = 0 alors nombre positif

Complément à un (C1)

- *Exemple :*

-24 en complément à 1 sur 8 bits :

  - |-24|en binaire pur = 0 0 0 1 1 0 0 0 $_{(2)}$
  - Puis on inverse les bits : 1 1 1 0 0 1 1 1 $_{(cà1)}$

- ***Limitation :***
  - Deux codages différents pour 0 (+0 et -0)
  - Sur 8 bits :
    +0=0 0 0 0 0 0 0 0$_{(cà1)}$ et -0=1 1 1 1 1 1 1 1$_{(cà1)}$

Complément à un (C1)

- *Exercices :*
- Coder 100 et -100 par complément à 1 (C1) sur 8 bits

  $100_{(10)} = (01100100)_{(C1)}$

  $-100_{(10)} = (10011011)_{(C1)}$

- Décoder en décimal $(11000111)_{(C1)}$ et $(00001111)_{(C1)}$

  $(11000111)_{(C1)} = -56_{(10)}$

  $(00001111)_{(C1)} = 15_{(10)}$

Complément à deux (C2)

Aussi appelé Complément Vrai (CV) :

- Les nombres positifs sont codes de la même manière qu'en binaire pure.
- Un nombre négatif est code en ajoutant la valeur 1 à son complément à 1

- Le bit le plus significatif est utilisé pour représenter le signe du nombre

Exemple :

-24 en complément à 2 sur 8 bits ?

24 est codé par : $0\ 0\ 0\ 1\ 1\ 0\ 0\ 0_{(2)}$

-24 : $1\ 1\ 1\ 0\ 0\ 1\ 1\ 1_{(C1)}$

Enfin -24 est codé par : $1\ 1\ 1\ 0\ 1\ 0\ 0\ 0_{(C2)}$

Complément à deux (C2)

- Un seul codage pour 0 = $00000000_{(C2)}$

  Par exemple sur 8 bits :

  +0 est code par $00000000_{(C2)}$

  -0 est code par $11111111_{(C1)}$,donc -0 sera représenté par $00000000_{(C2)}$
- ***Etendu de codage :***
  - Avec n bits, on peut coder de $-(2^{n-1})$ a $(2^{n-1}-1)$
  - Sur 1 octet (8 bits), codage des nombres de -128 à 127

+0	= 00000000	-0	=00000000
+1	= 00000001	-1	=11111111
…		…	
+127	= 01111111	-128	=10000000

Complément à deux (C2)

Exercice :

- Coder $100_{(10)}$ et $-100_{(10)}$ par complément à 2 sur 8 bits

$$100_{(10)} = 01100100_{(C2)}$$
$$-100_{(10)} = 10011100_{(C2)}$$

- Décoder en décimal $11001001_{(C2)}$ et $01101101_{(C2)}$

$$11001001_{(C2)} = -55_{(10)}$$
$$01101101_{(C2)} = 109_{(10)}$$

Codage des nombres réels en base 2

- Les formats de représentations des nombres réels sont :

1. Format virgule fixe

   Utilisé par les premières machines, possède une partie ‘entière’ et une partie ‘décimale’ sépares par une virgule.

   La position de la virgule est fixe d’où le nom.

   Exemple : $54,25_{(10)}$; $10,001_{(2)}$; A1, $F0B_{(16)}$

1. Format virgule flottante (utilisé actuellement sur machine) défini par : ± m . b ^{e}
   - Un signe + ou –
   - Une mantisse m (en virgule fixe)
   - Un exposant e (un entier relatif)
   - Une base b (2, 8, 10, 16, …)

- Exemple : $0,5425.10^{2}_{(10)}$; $10,1.2^{-1}_{(2)}$; $A0,B4.16^{-2}_{(16)}$

Codage des nombres réels en base 2 :

De base 10 à la base 2

- Le passage de la base 10 a la base 2 est défini par :
  - Partie entière est codée sur p bits (division successive par 2)
  - Partie décimale est codée sur q bits en multipliant par 2 successivement (On note seulement la partie entière obtenue) jusqu'à ce que la partie décimale soit nulle ou le nombre de bits q est atteint.

Exemple :

$4,25_{(10)} = ?_{(2)}$ [Le résultat en format virgule fixe]

$4_{(10)} = 100_{(2)}$

$0,25 \times 2 = \underline{0},5 \rightarrow 0$

$0,5 \times 2 = \underline{1},0 \rightarrow 1$

Donc $4,25_{(10)} = 100,\underline{01}_{(2)}$

Codage des nombres réels en base 2 :

De base 10 à la base 2

Exercice : Coder en binaire $28{,}8625_{(10)}$, $7{,}875_{(10)}$ et $5{,}3_{(10)}$ avec p = 8 et q = 8 ?

- Conversion de 28 : $(11100)_2$
- Conversion de 0,8625 :

0,8625 x 2 = 1,725 = **1** + 0,725
0,725 x 2 = 1,45 = **1** + 0,45
0,45 x 2 = 0,9 = **0** + 0,9
0,9 x 2 = 1,8 = **1** + 0,8
0,8 x 2 = 1,6 = **1** + 0,6
0,6 x 2 = 1,2 = **1** + 0,2
0,2 x 2 = 0,4 = **0** + 0,4
0,4 x 2 = 0,8 = **0** + 0,8 …

Sens d'écriture de gauche à droite

➔28,8625 peut être représenté par $(11100,\mathbf{11011100}...)_2$

Codage des nombres réels en virgule fixe : De base b à la base 10

- Etant donné une base b, un nombre x est représenté, en format virgule fixe, par :

$$x = a^{n-1}a^{n-2}\ldots a^{1}a^{0}, a^{-1}a^{-2}\ldots a^{-p}{}_{(b)}$$

a^{n-1} est le chiffre de poids fort (MSB)

a^{-p} est le chiffre de poids faible (LSB)

n est le nombre de chiffre avant la virgule

p est le nombre de chiffre après la virgule

- La valeur de x en base 10 est : $x = \sum_{i=-p}^{n-1} a_i \, b^i$ (10)

Codage des nombres réels en virgule fixe : De base b à la base 10

Exemples :

- Décimale

$128,75 = 1 \times 10^2 + 2 \times 10^1 + 8 \times 1 + 7 \times 10^{-1} + 5 \times 10^{-2}$

- Binaire

$101,01_{(2)} = 1.2^2 + 0.2^1 + 1.2^0 + 0.2^{-1} + 1.2^{-2} = 1 \times 4 + 1 + 0,25 = 5,25_{(10)}$

- Hexadécimale

$AE,1F = 10 \times 16^1 + 14 \times 16^0 + 1 \times 16^{-1} + 15 \times 16^{-2} = 160 + 14 + 0,0625 + 0,5859375 = 174,12109375_{(10)}$

Exercice : Algorithme de transcodage

- Ecrire un algorithme qui permet de saisir un réel n écrit en base 2 contenant 8 chiffres en partie entière et 8 chiffres en partie réelle, puis convertir ce nombre en base 10 et afficher la valeur trouvée.

Processeur

- C’est un ensemble de composants électroniques = UAL + {registres}
- Le registre : composé de plusieurs bascules permettant de stocker des informations (instructions à faire, données programme, adresses mémoire,…)
- Celui qui réfléchit dans l’ordinateur

Processeur

- L’ordinateur peut réfléchir grâce aux portes logiques
- Une porte logique : c’est un composant électronique qui ne permet le passage de l’électricité que sous dans certaines conditions
- Exemple : les portes logiques ET, OU, Ou exclusif et NON

Processeur

- Plusieurs portes forment un circuit qui peut réaliser des opérations
- Une opération : Combinaison des portes logiques, appelée « circuit combinatoire logique » ou simplement « circuit logique »
- L'UAL est un ensemble de circuits logiques
- On envoi l'information à l'UAL sous la forme de courant électrique pour faire le calcul
- Le courant sortant de circuit logique indique le résultat

Bus

- Permet d'acheminer les données entre les composants du l'ordinateur
- Le bus : c'est un ensemble de fils électriques fins
- 1 fil peut transporter un bit
- Fil chargé = 1
- Fil non chargé = 0
- Si le nombre de fils augmente, le nombre de bits transporté augmente
- On parle de largeur de bus = nombre de fils

Résumé

- Dans un ordinateur, tout est électricité : Mémorisation, réflexion, Transfert de données, Saisie, Affichage, Ecoute, …