ITI 1500 Hiver 2013

Systèmes Numériques I

Cours

Lundi 11:30 - 13:00 SCS E217

Jeudi 13:00 - 14:30 SCS E217

TUTORIAL Mardi 8 :30-10:00 salle: LMX-221

Professeur : Dr A. Karmouch, Bureau: CBY-A508

Manuel du cours :

- **website:** http://www.site.uottawa.ca/~karmouch/teaching/

- user/ & Password: “elg1500”

- Le livre est **obligatoire** et disponible à la librairie de l’université d’Ottawa.
- **Titre**: Digital Design
- **Auteurs**: Morris Mano & Michael D. Ciletti
- **Edition:** 5ieme Edition
- **Editeur:** Pearson-Prentice Hall

Manuel laboratoire

- ***ITI 1500*** *Manuel de laboratoire* version française (en ligne)
- Il y aura 6 travaux laboratoires
- **Date de début des laboratoires: sera communiquée ultérieurement**

Formation de groupes pour le laboratoire

- **Laboratoires:** Les horaires des laboratoires sont affichés sur le Site Web du cours
- La constitution de groupes sera effectuée dans le laboratoire du 18 Janvier 2013 (lab#0)

  (salle CBY-B302)
- les étudiants qui ne sont pas présents le 13 janvier dans le laboratoire seront affectés aux groupes par le TA (aucun changement n'est possible par la suite)
- → maximum 2 étudiants par groupe

Pondération des notes

- Devoirs 10%
- Laboratoires 15%
- Exam. Mi-session 25%
- Exam. Final 50%

PLAN DU COURS

PRINCIPES FONDAMENTAUX DES SYSTEMES NUMERIQUES

- **Systèmes de numération**

  Notation par valeur de position – Représentation polynomiale

  Système binaire – Système octal – système hexadécimal

  Codes BCD (4,6,8 bits)

- **Les opérations arithmétiques binaires**

  Addition – Soustraction – Multiplication – Division – Compléments

  Nombres binaires signés – Arithmétique des nombres binaires signés

  Arithmétique du complément à deux – Débordement

- **Les codes alphanumériques**

PLAN DU COURS

- **L'algèbre de Boole**

  Fonction logique ET – Fonction Logique OU – Fonction logique NON

  Les identités de Boole – Simplification des fonctions booléennes
- **Les portes logiques fondamentales**

  Porte logique OU – porte logique ET – Porte logique NON
- **Les formes standards**

  Somme des produits – Produit-des-sommes
- **Simplification par la méthode du diagramme de Karnaugh**

  Minimisation du produit des sommes – Fonctions indéterminées
- **Autres portes logiques**

  Porte logique NON ET (NAND) – Porte logique NON OU (NOR)

  Porte logique OU exclusif (XOR) – Porte logique NON exclusif (NXOR)

PLAN DU COURS

- **Les formes standards**
  Somme des produits – Produit-des-sommes
- **Simplification par la méthode du diagramme de Karnaugh**
  Minimisation du produit des sommes – Fonctions indéterminées
- **Autres portes logiques**
  Porte logique NON ET (NAND) – Porte logique NON OU (NOR)
  Porte logique OU exclusif (XOR) – Porte logique NON exclusif (NXOR)
- **Conceptions des réseaux combinatoires**
  Additionneur complet – Comparateur
- **Décodeur, encodeur et multiplexeur**
  Décodeur – Encodeur – Multiplexeur – Démultiplexeur
- **Conception de réseaux séquentiels :**
  Bascules S-R, J-K, D, T, registres, compteurs, diagramme d'états.
- **Révision Générale**

ITI 1500 **Hiver 2012**

Systèmes Numériques I

Cours

Lundi, 16:00 - 17:30 CBYB-012

Mercredi, 14:30 - 16:00 CBYB-012

TUT Mardi 8 :30-10:00 salle : CBY-D103

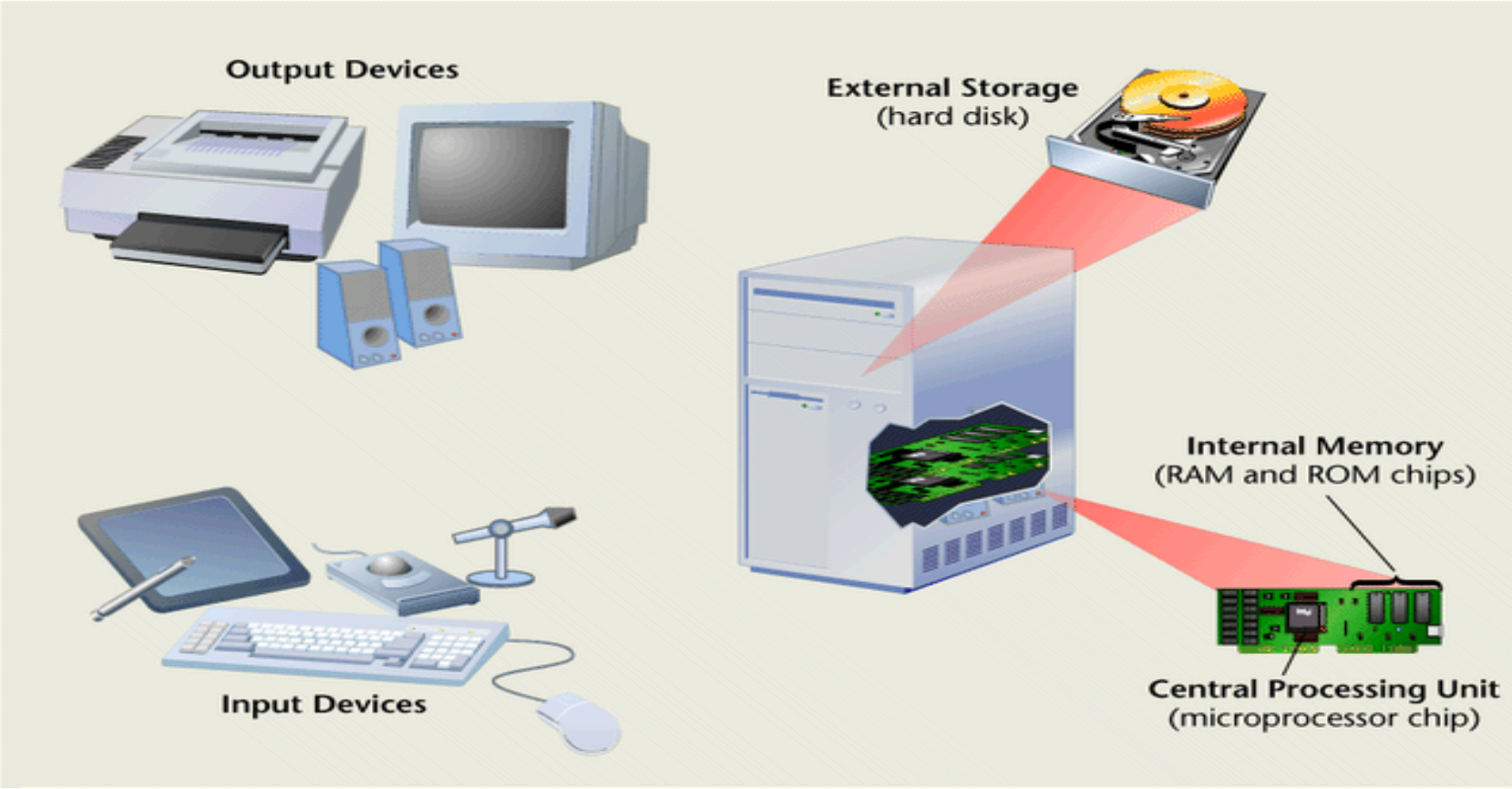
Professeur : Dr A. Karmouch, Bureau: CBY-A508

Chapitre 1

Le Système Binaire

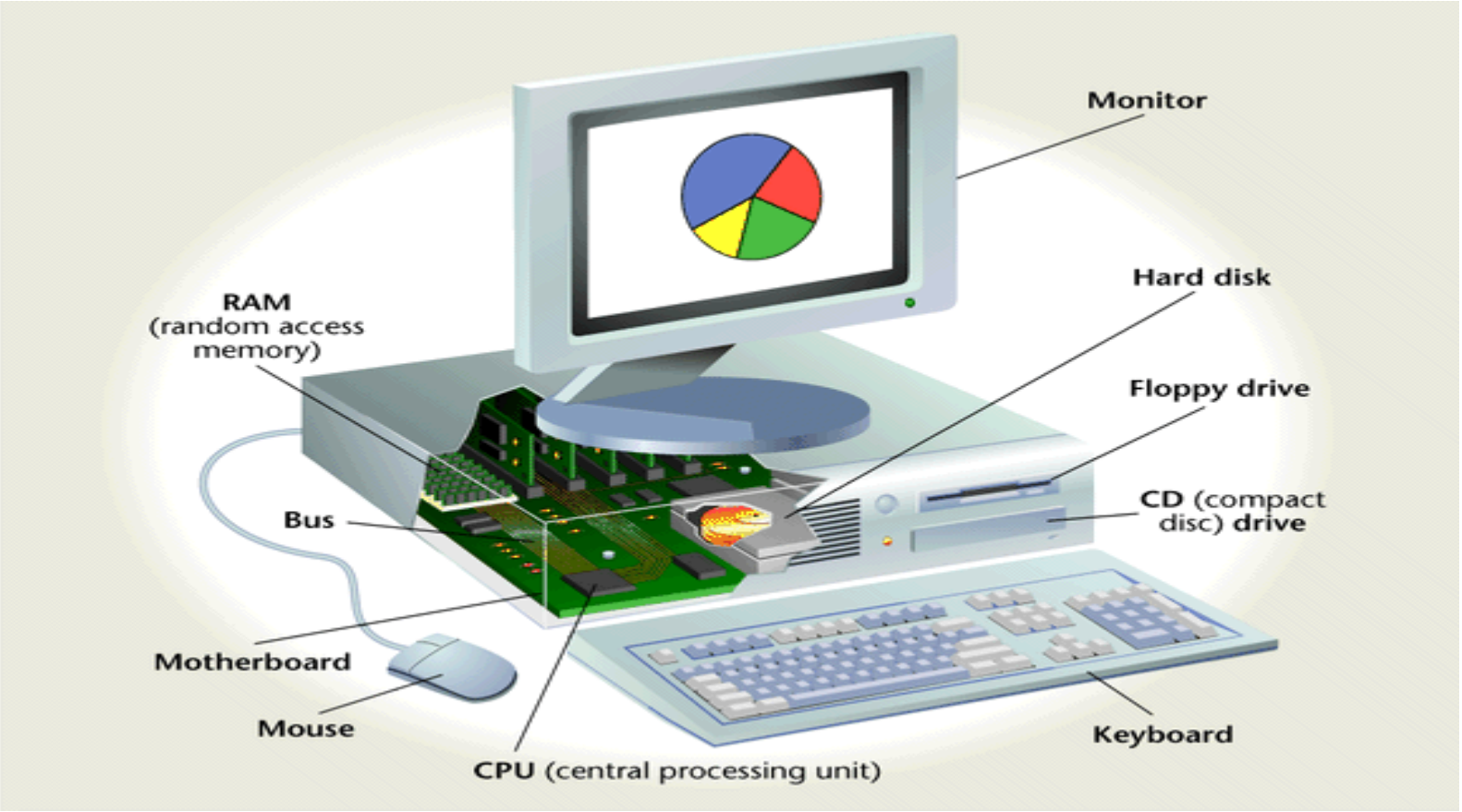
“ Le Tout Numérique”

Composantes de base de l'ordinateur



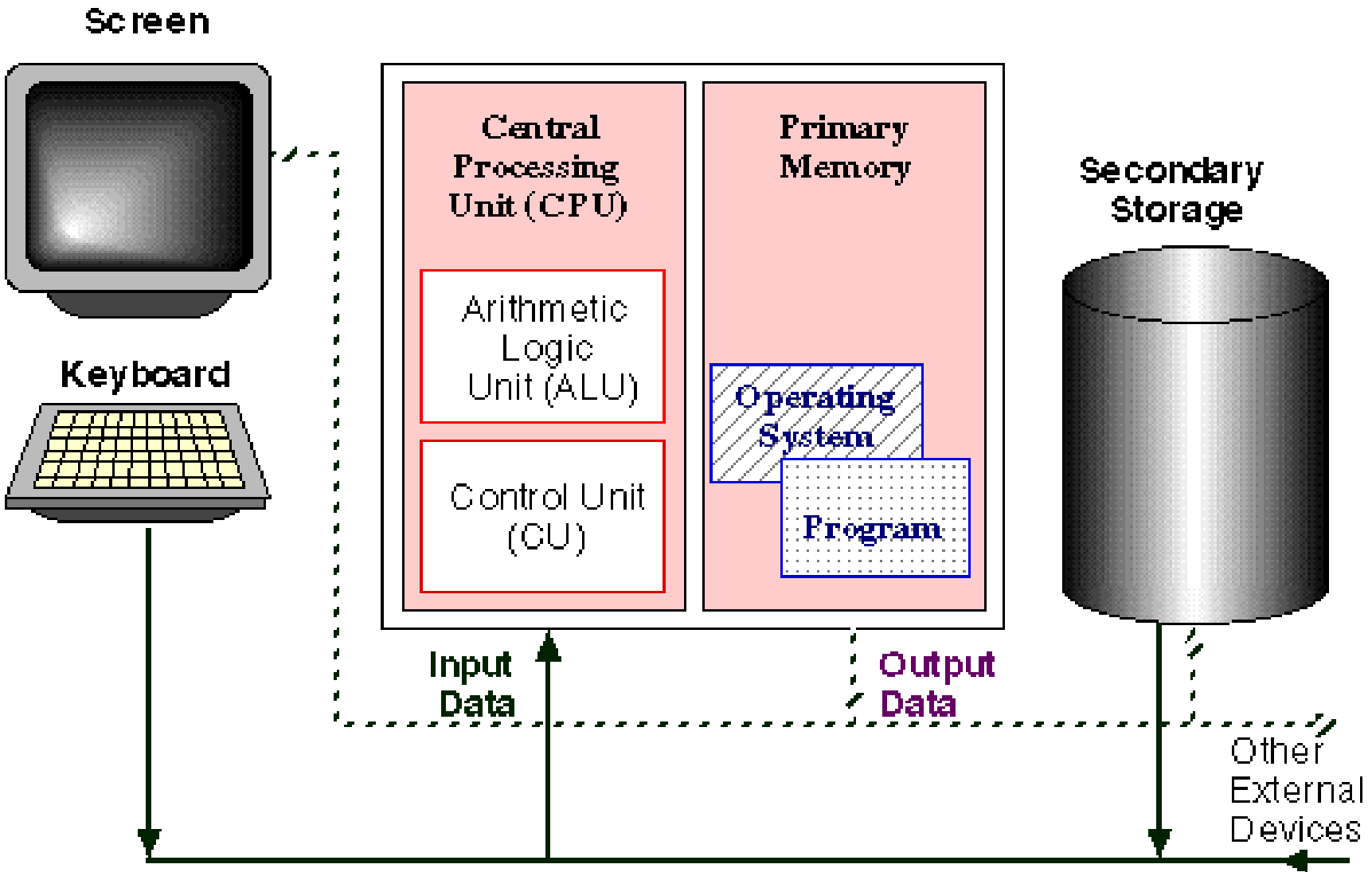
All computers have the same basic components.

Intérieur de l'ordinateur

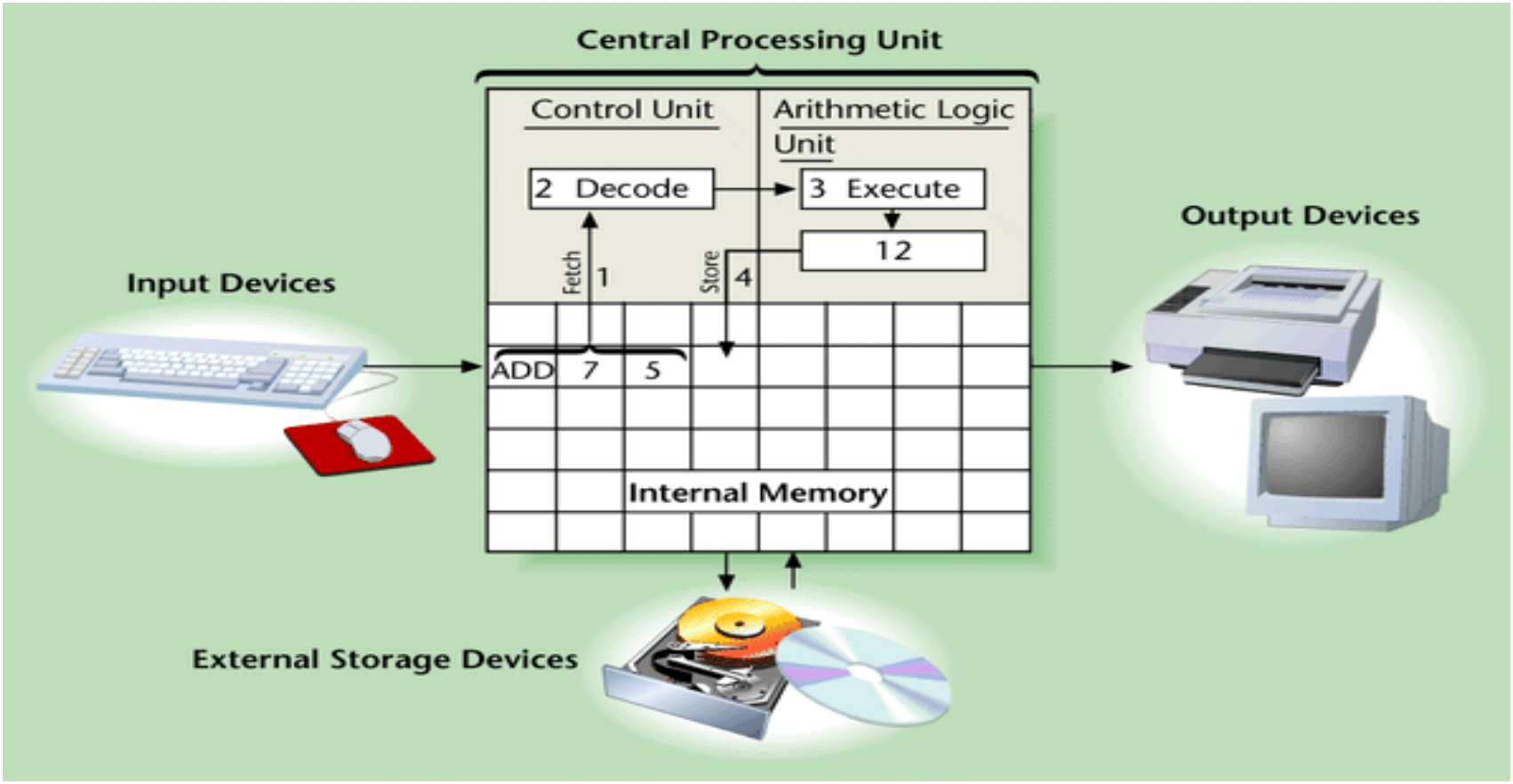


A look inside a computer

Block Diagram of a Digital Computer



Opérations Arithmétiques



What happens inside the CPU in one machine cycle executing the operation 7 + 5

Différents Systèmes de Numération

- Décimal (Arabe): (0,1,2,3,4,5,6,7,8,9):
  Exemple: **452968**
- Octal: (0,1,2,3,4,5,6,7):
  Exemple: **4073**
- Hexadécimal (0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F):
  Exemple: **2BF3**
- Binaire: (0,1):
  Exemple: **1001110001011**

Différents Systèmes de Numération

- Quels types de données sont utilisées par les ordinateurs?
  - a l'intérieure très bas dans l'ordinateur, tout est des 0s et des 1s

- Que peut –t'on faire avec des 0s et des 1s?
  - Les opérations d'algèbre de Boole
  - Ces opérations sont utilisées pour réaliser des circuits logiques

Base dans les systèmes de numération

- Le système décimal de numération utilise la **base 10**. Les valeurs de position sont déterminées en utilisant les puissance de 10

1	6	2	.	3	7	5	Chiffre
100	10	1		1/10	1/100	1/1000	poids

1	6	2	.	3	7	5	chiffre
10^2	10^1	10^0		10^{-1}	10^{-2}	10^{-3}	poids

- Pourquoi Base 10 ?

→ **car 10 chiffres: 0 a 9.**

Base dans les systèmes de numération

- Le système binaire est appelée *binaire* car il utilise la **base 2**. Les valeurs de position sont déterminée en utilisant les puissances de 2.

- Pourquoi Base 2 ?

  → **car 2 chiffres: 0 et 1.**

Représentation des Nombres

Il y a deux notations possibles pour représenter un nombre dans un système donné:

1- Notation Valeur de Position

2- Natation Polynomiale

Notation valeur de position

$$N = (a_{n-1}a_{n-2} \ldots a_1a_0 \,.\, a_{-1}a_{-2} \ldots a_{-m})_r$$

Ou

. = point de la base

r = la base

n = nombre de chiffres de la partie entière

m = le nombre de chiffres de la partie fractionnaire

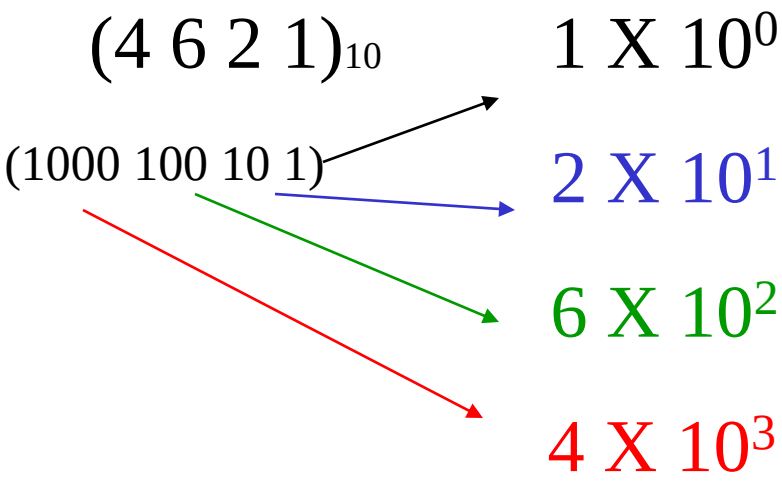
a_{n-1} = Le chiffre le plus significatif

a_{-m} = Le chiffre le moins significatif

Notation valeur de position

Système de Numération Décimal

- Le système de numération décimal est un nombre à valeur de position
- Exemple:



Notation valeur de position

Le système binaire

- Exemple de nombre binaires & les valeurs de positions.

$$\underline{1}\ \underline{0}\ \underline{1}\ \underline{0}\ \underline{1}\ \underline{0}\ \underline{1}$$

$$2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$$

1	1	0	1	.	0	1	Chiffre binaire, ou bits
2^3	2^2	2^1	2^0		2^{-1}	2^{-2}	Poids (en base 2)

Binary Digit → BIT

Notation Polynomiale

$$N = a_{n-1} \times r^{n-1} + a_{n-2} \times r^{n-2} + \ldots + a_0 \times r^0 + a_{-1} \times r^{-1} \ldots + a_{-m} \times r^{-m}$$

$$= \sum_{i=-m}^{n-1} a_i r^i$$

Position (N) ***Polynomiale (N)***

$$N = (651.45)_{10} = 6 \times 10^2 + 5 \times 10^1 + 1 \times 10^0$$

$$+ 4 \times 10^{-1} + 5 \times 10^{-2}$$

Nous allons étudier trois systèmes de numération

- Binaire
- Octal
- Hexadecimal

Le système binaire

Chiffre (bits) = {0, 1}

Position Polynomiale

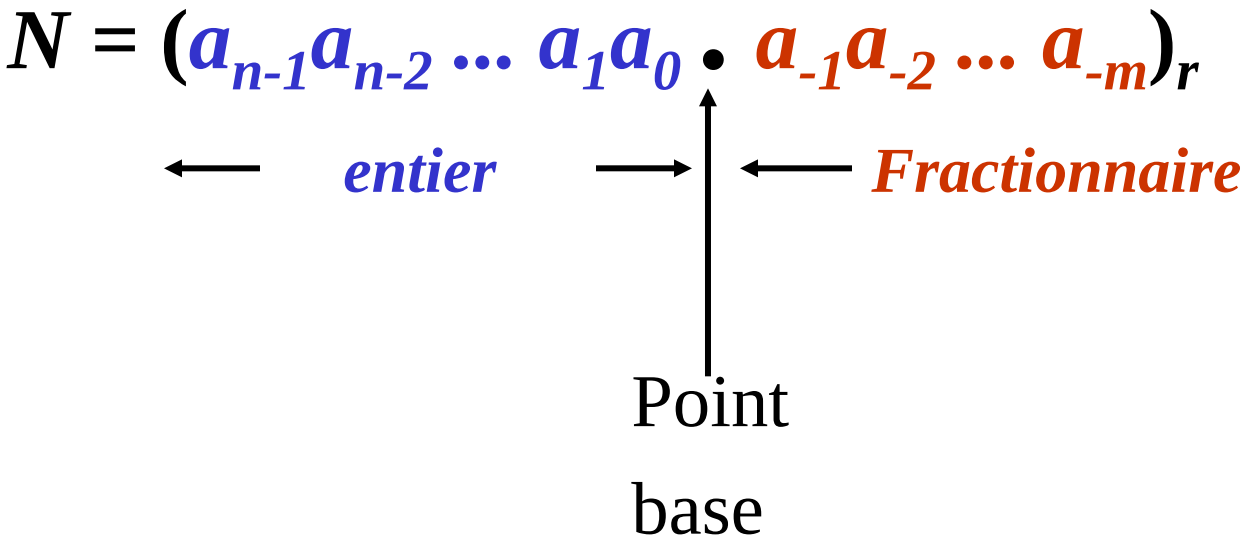
– $(11010.11)_2 = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2}$

– **1 K (kilo) = 2^{10} = 1,024**

– **1M (mega) = 2^{20} = 1,048,576**

– **1G (giga) = 2^{30} = 1,073,741,824**

Conversion décimal vers le binaire

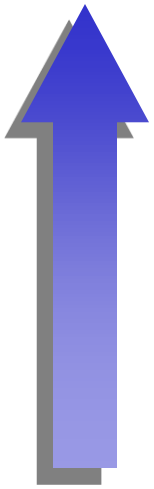


→ **La *partie entière* et la *partie fractionnaire* sont converties de deux manières différentes**

Conversion de la partie entière

- Continuer a diviser par 2 jusqu' à ce que le quotient est 0. Collecter ensuite les restes *dans l'ordre inverse*.
- Exemple: $(162)_{10}$.

162 / 2	= 81	reste 0
81 / 2	= 40	reste 1
40 / 2	= 20	reste 0
20 / 2	= 10	reste 0
10 / 2	= 5	reste 0
5 / 2	= 2	reste 1
2 / 2	= 1	reste 0
1 / 2	= 0	reste 1



- Alors $(162)_{10} = (10100010)_2$

Conversion de la partie Fractionnaire

→ Continuer à multiplier la partie fractionnaire par 2 jusqu'a obtenir zéro. Collecter les parties entières (dans ordre en avant).

– Cependant obtenir zéro peut ne pas se produire!

–Exemple: $(0.375)_{10}$

0.375 x 2 = 0.750
0.750 x 2 = 1.500
0.500 x 2 = 1.000

- alors, $(.375)_{10} = (.011)_2$

et $(162.375)_{10} = (10100010.011)_2$

Comprendre la méthode de conversion

- Cette méthode peut être utilisée pour convertir un nombre décimal vers tout autre système
- Par exemple 162.375 a convertir du décimal vers décimal: Pour la partie entière on a

```
162 / 10    = 16     reste 2
 16 / 10    = 1      reste 6
  1 / 10    = 0      reste 1
```

- Chaque division va "enlever" le chiffre le plus a droite (le reste) du nombre. Le quotient représente les chiffres restants dans le nombre.

Comprendre la méthode de conversion

Pour la partie Fractionnaire

$$0.375 \times 10 = 3.750$$
$$0.750 \times 10 = 7.500$$
$$0.500 \times 10 = 5.000$$

• Chaque multiplication va “enlever” le chiffre le plus a gauche (la partie entière) du nombre. La fraction représente les chiffres restant

Conversion binaire vers décimal

- Pour convertir les nombres binaire, ou base 2 vers décimal, on obtient d'abord la notation polynomiale du nombre, ensuite on fait la somme de tous les produits du polynomiale
  - Exemple: $(1101.01)_2$

1	1	0	1	.	0	1	Chiffres binaire ou bits
2^3	2^2	2^1	2^0		2^{-1}	2^{-2}	Poids (en base 10)

- La valeur décimale est:

$$(1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) + (0 \times 2^{-1}) + (1 \times 2^{-2}) =$$
$$8 + 4 + 0 + 1 + 0 + 0.25 = (13.25)_{10}$$

Le système binaire est difficile à manipuler

01010100000111100000011110000000111110
00000011101001010101000001110001010101
01010101010101010101010101010101010101
10101010101010101010101010101010101010

C’est pourquoi on a développée les systèmes Octal et Hexadécimal

le système Octal

–chiffres = {0, 1, 2, 3, 4, 5, 6, 7}

Position Polynomiale

$-(127.4)_8 = 1 \times 8^2 + 2 \times 8^1 + 7 \times 8^0 + 4 \times 8^{-1}$

• Octal (base 8) comprend des chiffres entre 0 a 7. Puisque $8 = 2^3$, un chiffre octal nécessite 3 positions binaires.

Conversion Décimal vers Octal

→**Partie entière:** diviser par 8 jusqu'a ce que quotient est 0. Collecter les restes *dans l'ordre inverse.*

→ **Partie Fractionnaire:** continuer a multiplier la partie fractionnaire par 8 jusqu'a obtenir zéro. Collecter les parties entière (dans l'ordre en avant).

Même chose que pour la conversion binaire

Conversion octal vers décimal

• Pour convertir le nombre octal, ou base 8 vers décimal, on obtient d'abord la notation polynomiale du nombre, ensuite on fait la somme de tous les produits.

Exemple

$$(127.4)_8 = 1 \times 8^2 + 2 \times 8^1 + 7 \times 8^0 + 4 \times 8^{-1} = (87.5)_{10}$$

Conversion Binaire vers Octal

• pour convertir le binaire vers l'octal, former des groupes de 3 bits à partir du point binaire. Ajouter des zéros à chaque bout si nécessaire. Ensuite convertir chaque groupe de bits en son équivalent octal.

•Exemple

$(10110100.001011)_2 = (010\ 110\ 100\ .\ 001\ 011\ )_2$

$= (\ 2\ \ 6\ \ 4\ .\ \ 1\ \ 3\ )_8$

Conversion Octal vers binaire

- Pour convertir un nombre octal vers le binaire on remplace chaque chiffre octal par son équivalent de séquence de 3-bit binaires.

- Exemple

$$261.35_8 = (2\ 6\ 1\ .\ 3\ 5)_8$$
$$= (010\ 110\ 001\ .\ 011\ 101)_2$$

Système Hexadécimal

-chiffres = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9**,**A, B, C, D, E, F**}**

Position Polynomiale

- $(B65F)_{16} = 11 \times 16^3 + 6 \times 16^2 + 5 \times 16^1 + 15 \times 16^0$

• *Hexadécimal* (base 16) chiffres sont 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. puisque $16 = 2^4$, un chiffre hexa est équivalent a 4 position binaires.

•Il est plus simple de travailler avec des nombres comme B5 au lieu de 10110101.

Conversion décimal vers Hexadécimal

→ **Partie entière:** diviser by 16 jusqu'a ce que quotient est 0. Collecter les restes *dans l'ordre inverse*.

→ **Partie Fractionnaire:** continuer a multiplier la partie fractionnaire par 16 jusqu'a obtenir zéro. Collecter les parties entière (dans ordre en avant).

Même principe que pour la conversion binaire

Conversion Hexadécimal vers Décimal

- Pour convertir les nombres hexa, ou base 16 vers décimal, on obtient d'abord la notation polynomiale du nombre, ensuite fait la somme de tous les produits

$$(\mathbf{B65F})_{16} = 11 \times 16^3 + 6 \times 16^2 + 5 \times 16^1 + 15 \times 16^0 = (46{,}687)_{10}$$

Conversion hexa vers le binaire

- Pour convertir un nombre hexa vers le binaire on remplace chaque chiffre hexa par son équivalent de séquence binaire de 4-bits.
- Exemple

261.35_{16} = (2 6 1 . 3 5)$_{16}$

= (0010 0110 0001 . 0011 0101)$_2$

Conversion Binaire vers Hexadécimal

•Pour convertir le binaire vers hexa, former des groupes de 4 bits à partir du point binaire. Ajouter des zéros à chaque bout si nécessaire. Ensuite convertir chaque groupe de bits en son équivalent hexa.

•Exemple

$10110100.001011_2 = (1011\ 0100\ .\ 0010\ 1100)_2$

$= (B\ \ 4\ .\ 2\ \ C)_{16}$

Decimal	Binary	Octal	Hex
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Les opérations Arithmétiques Binaires

Addition Binaire

	0	0	1	1
	+0	+ 1	+0	+1
	--	--	--	--
Somme	0	1	1	10

(somme de 0 et retenue de 1)

- Exemples:

1 0 0 1	0 0 0 1	1 1 0 0
+ 0 1 1 0	+ 1 0 0 1	+ 0 1 0 1
1 1 1 1	1 0 1 0	1 | 0 0 0 1

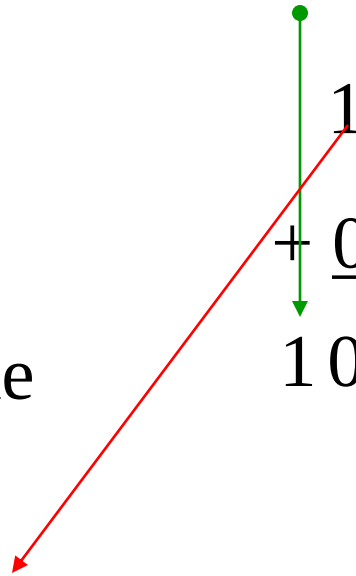
Exemple d'Addition Binaire

Retenue 1 1 1 1 0 1

1 0 1 1 0 1

\+ 0 1 1 1 0 1

Somme 1 0 0 1 0 1 0



Vérifier vos résultats

$(45)_{10}$

\+ $(29)_{10}$

= 74

$1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

$= 32 + 0 + 8 + 4 + 1 = 45$

Addition de grand nombres binaires

- *Exemple* de grand nombres binaires:

```
   1 0 1 0   0 0 1 1   1 0 1 1   0 0 0 1
 + 0 1 1 1   0 1 0 0   0 0 0 1   1 0 0 1
-----------------------------------------
 1 0 0 0 1   0 1 1 1   1 1 0 0   1 0 1 0
   ┘
```

Soustraction Binaire

	0	0	1	1
	- 0	- 1	- 0	- 1
	--	--	--	--
difference	0	11	1	0

(Diff. de 1 RETENUE DE 1)

0 10100

101011

- 01010 1

0 0 10110

Multiplication binaire

	0	0	1	1
	x 0	x 1	x 0	x 1
	--	--	--	--
Produit	0	0	0	1

—— A 0 010000.010

x B 0 001000.010

0000000000
0010000010
0000000000
0000000000
0000000000
0000000000
0010000010

1000110.000100

Division Binaire

	0		1	
÷	1	÷	1	
=	0	=	1	

Dividende

10000 010		1000 010	Diviseur
1000 010		1.1..	Quotient
01000 0000			
1000010			

Addition de trois chiffres binaires

x	y	retenue avant	Somme	Retenue après
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Compléments dans les systèmes de numération

•Compléments sont utilisés dans les ordinateurs pour simplifier l'opération de soustraction et les manipulations logiques.

• Il y a deux types de compléments pour chaque **base « r »** d'un système:

- ***complément à la base « r »***

Ex. ***Complément à la base 10***

Complément à base 2

- ***complément à la base moins un → (r-1)***

Ex. ***base 10 → 9***

Base 2 → 1

Complément à la base

$$[N]_r = r^n - (N)_r$$

n est le nombre de chiffre dans $(N)_r$

Exemple

- **le complément a 2 de $(N)_2 = (101001)_2$**

Donne :

$[N]_2 = 2^6 - (101001)_2 = (1000000)_2 - (101001)_2 = (010111)_2$

- **Complément 10 de $(N)_{10} = (72092)_{10}$**

Donne: $[N]_{10} = (100000)_{10} - (72092)_{10} = (27908)_{10}$.

Complément à 2

. Peut être obtenu directement à partir du nombre comme suit:

1- copier chaque bit du nombre à partir du bit le plus à droite (le bit le moins significatif)vers le bit le plus à gauche (le bit plus significatif) jusqu' à ce que le premier 1 est copié.

2- après que le premier 1 est copie remplacer chaque bit restant par un 1 si le bit est 0 ou par 0 si le bit est 1.

(a) $[1010100]_2 = 2^7 - (1010100)_2 = (10000000)_2 - (1010100)_2 = (\mathbf{0101100})_2$

(b) $[101001]_2 = 2^6 - (101001)_2 = (1000000)_2 - (101001)_2 = (\mathbf{010111})_2$

(a) **1 0 1 0 1 0 0** (b) **1 0 1 0 0 1**

2 's → **0 1 0 1 1 0 0** **0 1 0 1 1 1**

Complément à la base moins un

$$[N]_{r-1} = (r^n - 1)_r - (N)_r$$

- complement 9 de $[546700]_9$ = **999999 – 546700 = 453299**
- complement à 1 de [1011000]

  = $(10000000 - 1)_2 - (1011000)_2 = (0100111)_2$

Complément à 1

• peut être obtenu directement à partir du nombre en remplaçant chaque 0 par un 1 et chaque 1 par un 0.

[1011000] = $(10000000 - 1)_2 - (1011000)_2$

= **$(0100111)_2$**

1 0 1 1 0 0 0

complément à 1 → **0 1 0 0 1 1 1**

Soustraction avec le complément à 2

- **le complément à 2 est utilisé pour réduire la soustraction à une addition.**

$$A - B = A + (-B)$$

(additionner le complément à 2 de B à A)

- Complément 2 a les propriétés suivantes pour le signe « - »

A + (-A) = 0 **A + 2's{A} = 0**

- (- A) = A **10's{ 10's{A}} = A**

A - B = A + (-B) **A - B = A + 2's{B}**

Soustraction avec le complément à 2

Exemples:

A= 1010100 B= 1000011

- complément à 2

A - B = A+ [B]= (1010100) + (0111101) = (0010001)

$$\begin{array}{rr} & 1010100 \\ + & 0111101 \\ \hline & \cancel{1}\ 0010001 \end{array}$$

Ne pas tenir compte de la retenue

Soustraction avec Complément à 1

Exemples:

A= 1010100 B= 1000011

- complément à 1

A - B = A+ [B]= (1010100) + (0111100) = (0010001)

1010100

\+ 0111100

Retenue 1 0010000

A ajouter + 1

0010001

Soustraction avec Complements 10& 9

$(72)_{10}$ - $\mathbf{(32)_{10}}$ = $\mathbf{(40)_{10}}$

Complement 10

[32]= 10^2 **– (32)$_{10}$ = (68)$_{10}$**

(72)$_{10}$ + (68)$_{10}$ = ~~**1**~~ **(40)$_{10}$**

Complement 9

[32]= (10^2 -1**) – (32)$_{10}$ = (67)$_{10}$**

$(72)_{10}$ + **(67)$_{10}$ = (1 + 39)$_{10}$ = (40)$_{10}$**

Les nombres Binaires Signés

• rappel: les systèmes numériques sont construits avec des composants à deux états: 0 et 1.

•Les seuls états sont "1" et "0". Il n'y a pas d'état correspondant à "-" (signe moins)

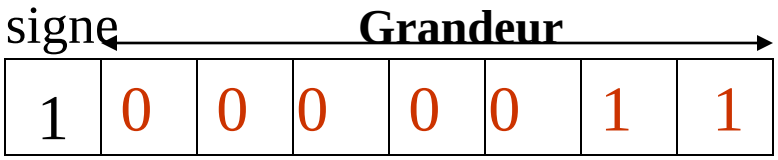
→à cause des limitations « matériel » l'ordinateur utilise le bit le plus à gauche du nombre pour représenter le signe:

-- "0" indique un nombre positif,

-- alors que un "1" indique un nombre négatif

Représentation Signe Grandeur

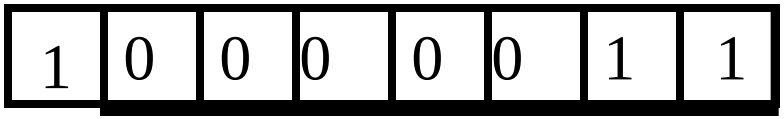
→En utilisant 8 bits pour représenter un nombre binaire:



-3 = 10000011 = 1/ (bit signe) 0000011

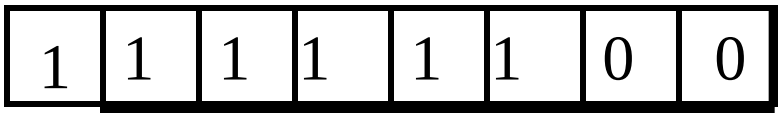
Représentation complément signés

(a) Représentation Signe Grandeur



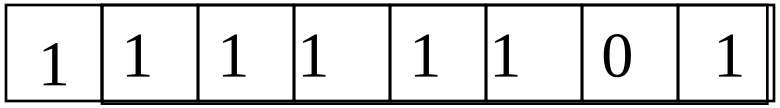
-3 = 10000011 = 1/ (signe bit) 0000011

(b) Représentation complément à 1 signé



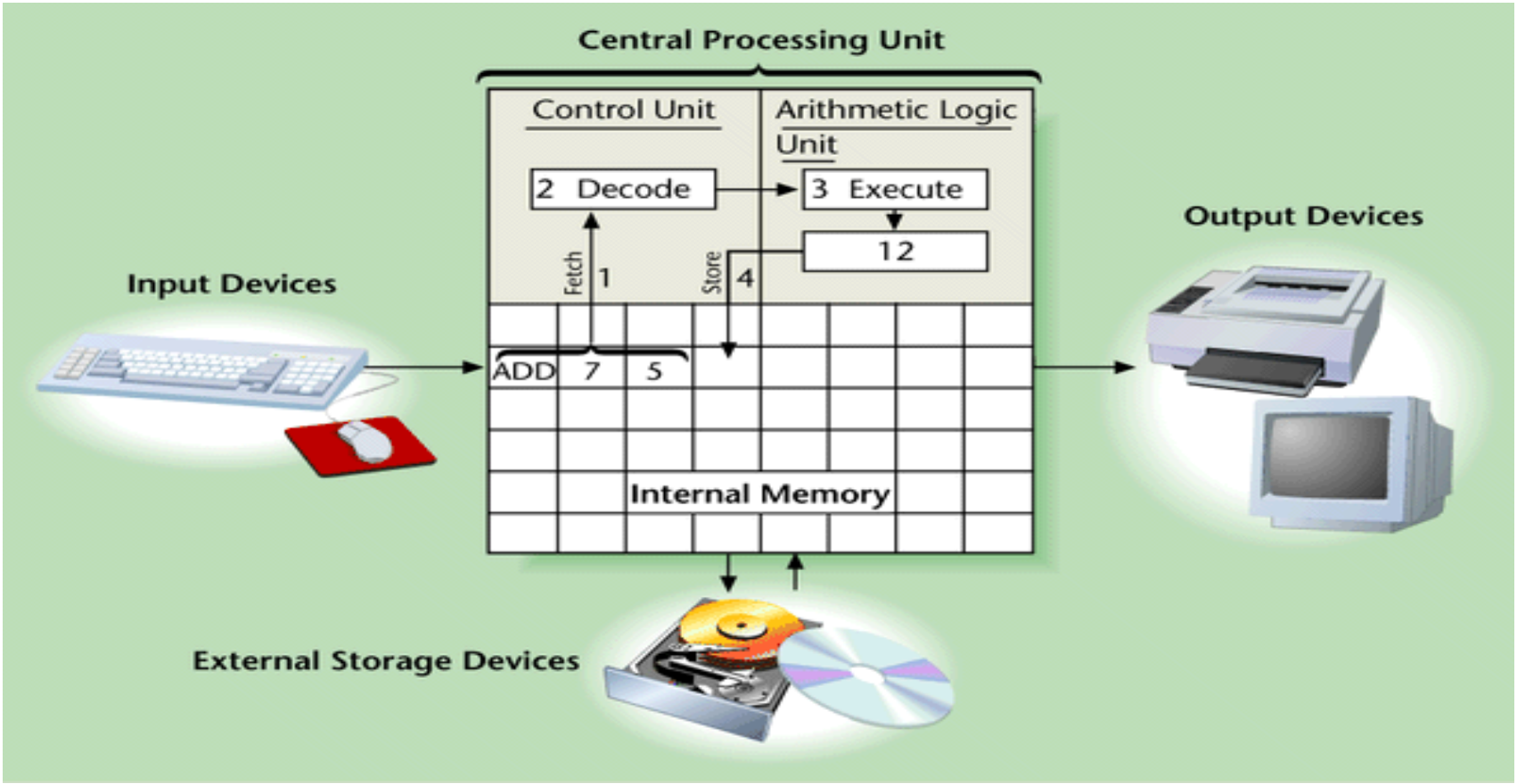
-3 = 10000011 = 1/ (signe bit) 1111100

(c) Représentation complément à 2 signé



-3 = 10000011 = 1/ (signe bit) 1111101

Opérations Arithmétiques



What happens inside the CPU in one machine cycle executing the operation 7 + 5

Registres de Longueur Fixe

- Les ordinateurs ont des registres de longueur fixe
- Cela veut dire que les nombres sont représentés sur des un nombre fixe de bits :
  - Les premier microprocessors avait 4-bits
  - Intel 8080 et 6502 (Apple II) chips était de 8-bits
  - Intel 8088 (IBM PC) et Motorola 68000 (Mac) sont 16-bits
  - Pentium chips et PowerPC chips sont 32-bits

Débordement durant opération d'addition

- le registre de longueur fixes peut représenter des nombres de certain grandeur seulement

→Pour le registre à 4-bits la grandeur des entiers positifs est de 0 - 15

→Pour le registre à 8-bits la grandeur des entiers positifs est de 0 – 255

→ Lors de l'addition de nombre positifs, ***le débordement se produit*** *quand* la somme des deux nombres se situe à l'extérieur de la grandeur du registre

Débordement -Compléments Signés

- Lors que les nombres sont représentés en compléments signés une retenue de « 1 » qui se dégage de l'addition des deux bits les plus significatifs **N 'EST PAS** un indicateur de débordement. Ex

3	00011
+ (-3)	+11101
= 0	= 00000, retenue de "1"(complément a 2)

Débordement -Compléments Signés

•Pour nombres compléments signés le débordement se produit lorsque

→ ***L' addition de deux nombre positives donne un nombre négatif***

OU → ***L' addition de deux nombre négatifs donne un nombre positif***

Exemples de débordement

- registre à 6-bits :complément a 2 signé

  + 17 = 010001

  + 16 = +010000

  =100001

  100001 = - (11111) = -(31)$_{10}$ au lieu de **+ (33)$_{10}$**

- On refait la même chose avec registre 7-bits

  + 17 = 0 010001

  + 16 = +0 010000

  = 0100001

  0100001 = + 33 Pas de débordement

Les codes binaires

- Pour représenter l'information comme une chaîne alpha-numérique de caractères.
- ***Décimal Codé Binaire (DCB ou BCD)***
  - Utilisé pour représenter les chiffres décimaux de 0 - 9.
  - 4 bits sont utilisées.
  - A chaque position est associée un poids
  - Les poids sont: 8, 4, 2, et 1 à partir du plus significatif au moins significatif (appelé code 8-4-2-1).

Les codes binaires

– DCB Codes:

0→ 0000 1→ 0001 2→ 0010

3→ 0011 4→0100 5→ 0101

6→ 0110 7→0111 8→ 1000

9→ 1001

– Utilisée pour coder les nombres en sortie pour affichage

–***Exemple***: $(9750)_{10} = (1001011101010000)_{BCD}$

Les codes binaires : ASCII

- ***ASCII*** (American Standard Code for Information Interchange) (voir table 1.7 du livre)
  - Code le plus utilisé pour représenter le codage caractères
  - ***Exemple***: représentation en code du mot

'Digital'

Caractère	Code binaire	code Hexadécimal
D	1000100	44
i	1101001	69
g	1100111	67
i	1101001	69
t	1110100	74
a	1100001	61
l	1101100	6C

Exercices pratiques

Exemples: Complément à 2' Signés

	Bit-signe	
$(9)_{10}$	0	1001
$+(6)_{10}$	0	0110
	0	1111
$(9)_{10}$	0	1001
$-(6)_{10}$	1	1010
	0	0011
$(6)_{10}$	0	0110
$-(9)_{10}$	1	0111
	1	1101

Exemples: Complément à 1’ Signés

	Bit-signe	
$(9)_{10}$	0	1001
$+(6)_{10}$	0	0110
	0	1111
$(9)_{10}$	0	1001
$-(6)_{10}$	1	1001
1	0	0010 = (0010) + (0001) = (0011)
$(6)_{10}$	0	0110
$-(9)_{10}$	1	0110
	1	1 1 0 0

Exercices

Question 1-

(a) Convertir le nombre binaire suivant en (i) Octal, (ii) Décimal, (iii) hexadécimal

10101101.10110

(b) Convertir A= 16.25 et B = 8.25 en binaire utiliser 7 bits pour représenter la partie entière et 3 bits pour représenter la partie fractionnaire ensuite effectuer les opérations suivantes

I) C= A + B
ii) D = A – B
iii) E= A x B
vi) F = A B

Note: calculer C et D
(a) en utilisant le binaire non-signés et sans compléments
(b) en utilisant le complément à 2 **signé**

(c) Convertir le nombre suivant en (i) Décimal, (ii) Octal, (iii) binaire

(FD8.C2B)16

Exercices

Solutions

10101101.10110

i) Octal→ (010 101 101.101 100)

(2 5 5 . 5 4)$_8$

iii) Hexa → (1010 1101.1011 0000)2

(A D . B 0)16

ii) Décimal

(10101101.10110)$_2$ = $1 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2}\ 1 \times 2^{-3} + 1 \times 2^{-4} + 0 \times 2^{-5}$

= (173.6875)$_{10}$

Conversion de la partie entière

- Conversion de $(16)_{10}$.

```
16 / 2   = 8    reste 0
 8 / 2   = 4    reste 0
 4 / 2   = 2    reste 0
 2 / 2   = 1    reste 0
 1 / 2   = 0    reste 1
```

- Alors $(16)_{10} = (10000)_2$

Conversion de la partie Fractionnaire

–Conversion de: $(0.25)_{10}$

$0.25 \times 2 = 0.50$
$0.50 \times 2 = 1.00$ ↓

•Donc, $(.25)_{10} = (.01)_2$

et $(16.25)_{10} = (10000.01)_2$

En procède de la même manière pour B = 8.25 ce qui donne

$(8.25)_{10} = (1000.01)_2$

→ Représentation sur 7 bits et 3 bits

A = $(16.25)_{10}$ = (0010000.010)

B = $(8.25)_{10}$ = (0001000.010)

Exemples: Binaire Non Signé

A	0010000.010
+ B	0001000.010
	0 011000.100

A	0010000.010
- B	0001000.010
	0001000.000

Complément à 2' Signés

A	0 010000.010
+ B	0 001000.010
	0 011000.100

A	0 010000.010
+ (-B)	1 110111.110
	0 001000.000

Multiplication

A 0 010000.010

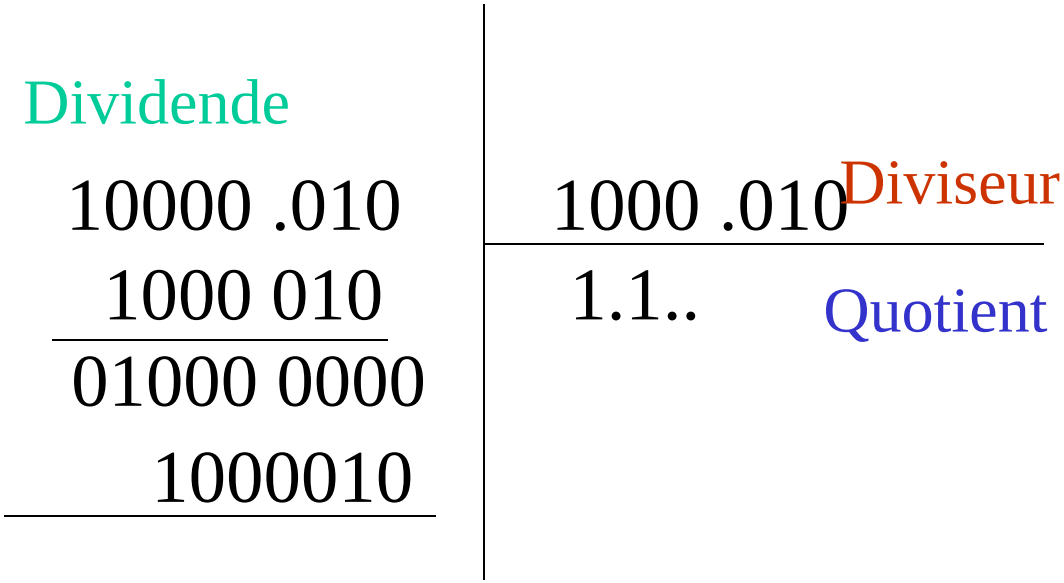
x B 0 001000.010

0000000000
0010000010
0000000000
0000000000
0000000000
0000000000
0010000010

1000110.000100

Division

A $\div$ B



Question 1: (20 points)

Convertir A = $(00010010.0101)_{DCB}$ and B= $(2.25)_{10}$ en binaire en utilisant 8 bits pour la partie entiere et 3 pour la partie fractionnaire, y compris le bit du signe. Effectuer les opérations suivantes

(i) C = -A - B avec le complément a 2 signé
(ii) D = – A + B avec le complément a 1 signé
(iii) E= A – B avec le complément a 9.

A= $(12.50)_{10}$ B= $(2.25)_{10}$

Faire: $(12.50)_{10}$ - $(2.25)_{10}$ en complement à 9

----complément à 9 de (2.25)

Première étape est de représenter les A et B sur le même nombre de positions ensuite faire le complément à 9 de (02.25)

$[02.25]_9 = (10^4 - 1)_{10} - (02.25)_{10} = (99.99 - 02.25)_{10} = (97.74)_{10}$

E = A- B = A+ $[B]_9$ =

$$\begin{array}{rr} & 12.50 \\ + & 97.74 \\ \hline (\text{retenue})\ 1 & 10\ .2\ 4 \\ & \quad + 1 \\ \hline & 10.\ \ 25 \end{array}$$