

B2

COMPILATION & ASSEMBLEUR

PRÉSENTATION

Informations utiles

INFORMATIONS DE CONTACT

- Samir AZZAG
- samir.azzag@ynov.com
- En cas d'urgence :
 - samir_azzag@yahoo.fr

INFORMATIONS IMPORTANTES

- Retards
 - Chaque retard devra être validé par la direction
- Téléphones
 - Aucun appel n'est possible, les téléphones doivent être rangés dans les sacs, sauf utilisation autorisée
- Jeux vidéos et réseaux sociaux
 - La pratique de jeux vidéos pendant les cours donnera lieu à une exclusion du cours
 - L'utilisation des réseaux sociaux est interdite pendant les cours, sauf utilisation autorisée

INFORMATIONS IMPORTANTES

- Niveau sonore
 - La participation au cours est indispensable, mais elle doit se faire avec le niveau sonore le plus bas possible
 - Chaque question ou demande doit être précédée d'une main levée
- Absences
 - Les absences répétées seront sanctionnées
- Matériel indispensable
 - BYOD, votre cerveau et vos softs skills

INFORMATIONS IMPORTANTES

■ Système d'évaluation

- DS/DM : Tout au long du module → 60% de la note finale
- Projet : un mini compilateur C vers x86-64 → 40% de la note finale

INFORMATIONS IMPORTANTES

- Ce cours est une introduction à la compilation et au langage assembleur
- On y explique les techniques et outils utilisés dans les différentes phases d'un compilateur, jusqu'à la production de code assembleur optimisé.
- Un compilateur pour un fragment du langage C vers l'assembleur x86-64 est réalisé en TD
- Les objectifs du cours sont les suivants :
 - Comprendre le fonctionnement et l'intérêt d'un compilateur
 - Comprendre le fonctionnement d'un assembleur
 - Comprendre le fonctionnement interne des composants d'un ordinateur

INFORMATIONS IMPORTANTES

PROGRAMME

- **Rappels**
 - Codage de l'information
 - Les SI et le modèle de Von Neumann
 - Architecture en couches (multi-niveaux)
 - Le Hardware
 - Le Software
 - Algèbre de Boole et fonctions booléennes
 - Circuits logiques
- **Introduction à la compilation et à la théorie des langages**
- **Analyse lexicale et syntaxique**
- **Syntaxe abstraite, sémantique et interprète**
- **Table de symbole et typage**
- **Assembleur et production de code**
- **Allocation de registres**

Ce plan est donné à titre indicatif. Il est pourra être modifié au fil de l'avancement réel du cours.

INFORMATIONS IMPORTANTES

- Ce cours ne « compile » qu'une petite partie de tout ce qui est vu et dit
- Par conséquent, un grand nombre d'informations n'y figurent pas.
- Il ne s'agit que d'un support pour accompagner le cours et non d'un cours détaillé.
- Le contenu qui va suivre doit s'accompagner de vos notes prises pendant les séances de cours

RAPPELS

Codage de l'information

CODAGE DE L'INFORMATION

- Les informations traitées par les ordinateurs sont de différentes natures :
 - Nombres, texte, images, sons, vidéos, programmes...
- Dans un ordinateur, elles sont toujours représentées sous forme binaire(BIT = **B**inary **d**ig**IT**)
 - Une suite de 0 et de 1

00110110 11010101 10100011 00111001 10011000 10101100
00100101 01011010 01110110 11011010 10110111 10101101

CODAGE DE L'INFORMATION

- Définition : le codage de l'information
 - Le codage de l'information permet d'établir une correspondance qui permet sans ambiguïté de passer d'une représentation (dite externe) d'une information à une autre représentation (dite interne : sous forme binaire) de la même information, suivant un ensemble de règles précises.
- Exemple :
 - Le nombre 35 : 35 est la représentation externe du nombre trente cinq
 - La représentation interne de 35 sera une suite de 0 et de 1 : 00100011

CODAGE DE L'INFORMATION

- Conversion de la base 10 vers une base b
 - Méthode 1 : Divisions successives par b
 - Méthode 2 : En se basant sur les tableaux associés aux puissances de la base b

CODAGE DE L'INFORMATION

- Conversion de la base 2 vers une base 10
 - Soit N un nombre représenté en base binaire par : $N = 1010011101_{(2)}$
 - Représentation décimale ?

$$N = 1010011101$$

$$\begin{aligned}N &= 1 \times 2^9 + 0 \times 2^8 + 1 \times 2^7 + 0 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\&= 512 + 0 + 128 + 0 + 0 + 16 + 8 + 4 + 0 + 1 \\&= 669_{(10)}\end{aligned}$$

$$1010011101_{(2)} = 669_{(10)}$$

CODAGE DE L'INFORMATION

- Conversion de la base 8 vers une base 10
 - Soit N un nombre représenté en base octale par : $N = 732_{(8)}$
 - Représentation décimale ?

$$N = 7 \ 3 \ 2$$

$$\begin{aligned}N &= 7 \times 8^2 + 3 \times 8^1 + 2 \times 8^0 \\&= 448 + 24 + 2 \\&= 474_{(10)}\end{aligned}$$

$$732_{(8)} = 474_{(10)}$$

CODAGE DE L'INFORMATION

- Conversion de la base 16 vers une base 10
 - Soit N un nombre représenté en base binaire par : $N = B2A_{(16)}$
 - Représentation décimale ?

$$N = B2A$$

$$\begin{aligned}N &= 11 \times 16^2 + 1 \times 16^1 + 10 \times 16^0 \\&= 2816 + 16 + 10 + \\&= 2858_{(10)}\end{aligned}$$

$$B2A_{(16)} = 2858_{(10)}$$

CODAGE DE L'INFORMATION

Suite binaire	Symbole octale
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

CODAGE DE L'INFORMATION

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

CODAGE DE L'INFORMATION

- Soit N un nombre représenté en base binaire par : $N = 1010011101_{(2)}$
- Représentation Octale ?

$$\begin{aligned}N &= 001 \quad 010 \quad 011 \quad 101 \\&= 1 \quad \quad \quad 2 \quad \quad \quad 3 \quad \quad \quad 5 \\&= 1235_{(8)}\end{aligned}$$

$$1010011101_{(2)} = 1235_{(8)}$$

CODAGE DE L'INFORMATION

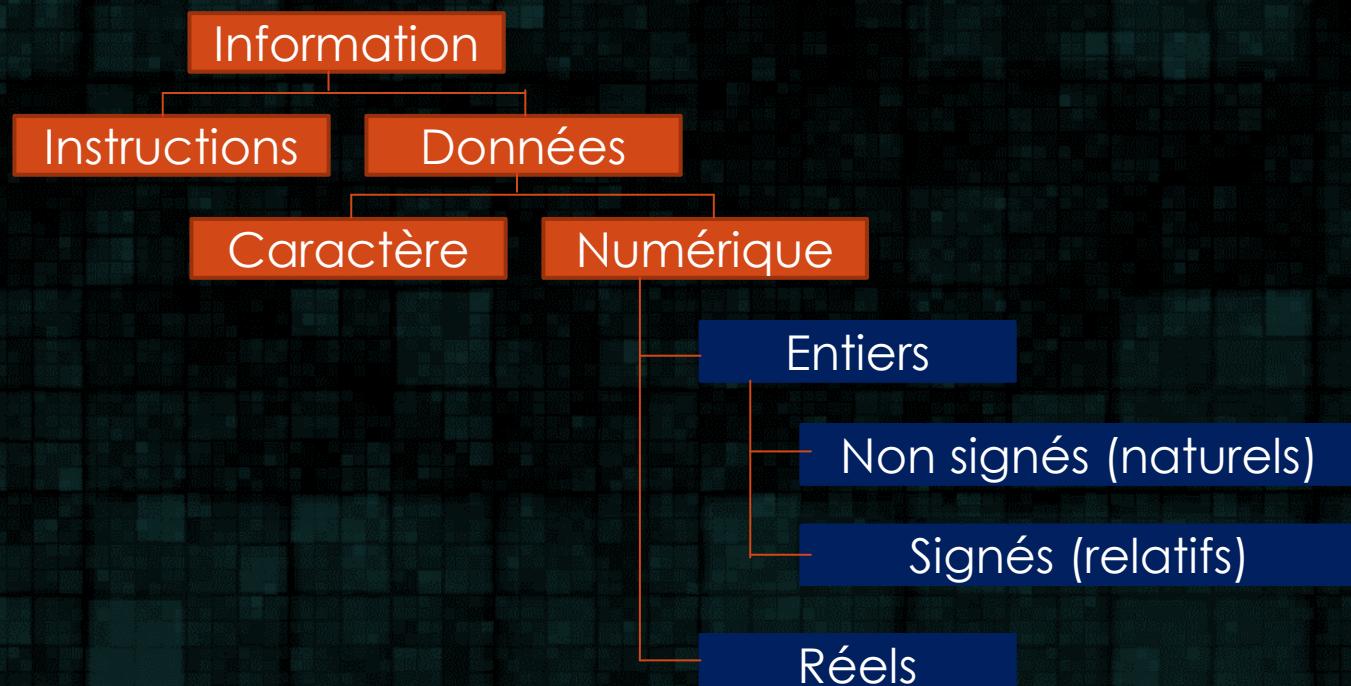
- Soit N un nombre représenté en base binaire par : $N = 1010011101_{(2)}$
- Représentation Hexadécimale ?

$$\begin{aligned}N &= 0010 \quad 1001 \quad 1101 \\&= \quad 2 \quad \quad 9 \quad \quad D \\&= 29D_{(16)}\end{aligned}$$

$$1010011101_{(2)} = 29D_{(16)}$$

CODAGE DE L'INFORMATION

■ Représentation de l'information



CODAGE DE L'INFORMATION

- Codage des entiers naturels :
 - Utilisation du code binaire pur :
 - L'entier naturel (positif ou nul) est représenté en base 2,
 - Les bits sont rangés selon leur poids, on complète à gauche par des 0.
 - Exemple : sur un octet, $10_{(10)}$ se code en binaire pur ?

0 0 0 0 1 0 1 0₍₂₎

CODAGE DE L'INFORMATION

- Codage des entiers naturels :
 - Plage de codage en binaire pur :
 - Codage sur n bits : représentation des nombres de 0 à $2^n - 1$
 - Sur 1 octet (8 bits): codage des nombres de 0 à $2^8 - 1 = 255$
 - sur 2 octets (16 bits): codage des nombres de 0 à $2^{16} - 1 = 65535$
 - sur 4 octets (32 bits) : codage des nombres de 0 à $2^{32} - 1 = 4\ 294\ 967\ 295$

CODAGE DE L'INFORMATION

- Codage des entiers relatifs : binaire signé
 - 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
 - Les autres bits codent la valeur absolue du nombre

CODAGE DE L'INFORMATION

- Exemple : Sur 8 bits, codage des nombres -24 et 128
 - -24 est codé en binaire signé par : $1\ 0011000_{(bs)}$


Signe -
 - 128 ne peut pas être codé en binaire signé sur 8 bits

CODAGE DE L'INFORMATION

■ Exercices

- Coder $100_{(10)}$ et $-100_{(10)}$ en binaire signé sur 8 bits
 - $100_{(10)} = 0110\ 0100_{(bs)}$
 - $-100_{(10)} = 1110\ 0100_{(bs)}$
- Décoder en décimal $1100\ 0111_{(bs)}$ et $0000\ 1111_{(bs)}$
 - $1100\ 0111_{(bs)} = -71_{(10)}$
 - $0000\ 1111_{(bs)} = 15_{(10)}$

CODAGE DE L'INFORMATION

- Codage des entiers relatifs : complément à 1
 - 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

CODAGE DE L'INFORMATION

- Exemple : -24 en complément à 1 sur 8 bits
 - |-24| en binaire pur $00011000_{(2)}$
 - Puis on inverse les bits $11100111_{(\text{c}\grave{\text{a}}1)}$
- Limitation :
 - deux codages différents pour 0 (+0 et -0)
 - Sur 8 bits : $+0 = 00000000_{(\text{c}\grave{\text{a}}1)}$ et $-0 = 11111111_{(\text{c}\grave{\text{a}}1)}$
 - La multiplication et l'addition sont moins évidentes.

CODAGE DE L'INFORMATION

■ Exercices

- Coder $100_{(10)}$ et $-100_{(10)}$ en complément à 1 sur 8 bits
 - $100_{(10)} = 0110\ 0100 \text{ (Cà1)}$
 - $-100_{(10)} = 1001\ 1011 \text{ (Cà1)}$
- Décoder en décimal $1100\ 0111_{(\text{Cà1})}$ et $0000\ 1111_{(\text{Cà1})}$
 - $11000111_{(\text{Cà1})} = -56_{(10)}$
 - $00001111_{(\text{Cà1})} = 15_{(10)}$

CODAGE DE L'INFORMATION

- Codage des entiers relatifs : complément à 2
 - Aussi appelé Complément Vrai (CV) :
 - Les nombres positifs sont codés de la même manière qu'en binaire pure
 - un nombre négatif est codé en ajoutant la valeur 1 à son complément à 1
 - Le bit le plus significatif est utilisé pour représenter le signe du nombre

CODAGE DE L'INFORMATION

- Exemple : -24 en complément à 2 sur 8 bits
 - 24 est codé par : $00011000_{(2)}$
 - -24 est codé par : $11100111_{(\text{c}\grave{\text{a}}1)}$
 - donc -24 est codé par : $11101000_{(\text{c}\grave{\text{a}}2)}$

CODAGE DE L'INFORMATION

- Codage des entiers relatifs : complément à 2
 - Un seul codage pour 0. Par exemple sur 8 bits :
 - +0 est code par : 0 0 0 0 0 0 0_(cà2)
 - -0 est code par : 1 1 1 1 1 1 1_(cà1)
 - Donc -0 sera représenté par : 0 0 0 0 0 0 0_(cà2)

CODAGE DE L'INFORMATION

- Plage de codage :
 - Avec n bits, on peut coder de $-(2^{n-1})$ à $(2^{n-1}-1)$
 - Sur 1 octet (8 bits), codage des nombres de -128 à 127
 - $+0 = 00000000$ $-0 = 00000000$
 - $+1 = 00000001$ $-1 = 11111111$
 -
 - $+127 = 01111111$ $-128 = 10000000$

CODAGE DE L'INFORMATION

■ Exercices

- Coder $100_{(10)}$ et $-100_{(10)}$ par complément à 2 sur 8 bits
 - $100_{(10)} = 0110\ 0100_{(\text{C}\grave{\text{a}}2)}$
 - $-100_{(10)} = 1001\ 1010_{(\text{C}\grave{\text{a}}2)}$
- Décoder en décimal $1100\ 1001_{(\text{C}\grave{\text{a}}2)}$ et $0110\ 1101_{(\text{C}\grave{\text{a}}2)}$
 - $11001001_{(\text{C}\grave{\text{a}}2)} = -55_{(10)}$
 - $01101101_{(\text{C}\grave{\text{a}}2)} = 109_{(10)}$

CODAGE DE L'INFORMATION

- Les formats de représentation des nombres réels sont :
 - Format à virgule fixe
 - Utilisé par les premières machines
 - Possède une partie 'entière' et une partie 'décimale' séparés par une virgule. La position de la virgule est fixe, d'où le nom.
 - Exemple : $54,25_{(10)}$; $10,001_{(2)}$; A1,F0B $_{(16)}$
 - Conversion de la base b vers la base 10
 - $101,01_{(2)} = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} = 5,25_{(10)}$

CODAGE DE L'INFORMATION

- Le passage de la base 10 à 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 jusqu'à ce que la partie décimale soit nulle ou que le nombre de bits q désiré soit atteint (i.e. que l'on obtienne la précision demandée)
- Exemple 1 : $4,25_{(10)} = ?_{(2)}$ format virgule fixe
 - $4_{(10)} = 100_{(2)}$
 - $0,25 \times 2 = 0,5 \rightarrow 0$ ↓
 - $0,5 \times 2 = 1,0 \rightarrow 1$
 - Donc $4,25_{(10)} = 100,01_{(2)}$

Si le calcul est cyclique ou ne s'arrête pas, on calcule jusqu'à la précision désirée

CODAGE DE L'INFORMATION

- Un problème se pose : l'arrondi en binaire !!
 - Si le premier symbole abandonné est 0, on fait une troncature – on abandonne les symboles suivants (arrondi par défaut)
 - Si le premier symbole abandonné est 1, on ajoute 1 au dernier symbole conservé (arrondi par excès).

$$13,4 = 1101,01100\textcolor{red}{0}1100110\dots \rightarrow 1101,0110$$

Premier symbole abandonné est un zéro

CODAGE DE L'INFORMATION

- Format à virgule flottante (utilisé actuellement sur machine)
 - défini par : $\pm m \cdot b^e$
 - Un signe : + ou -
 - Une mantisse : m (en virgule fixe)
 - Un exposant : e (un entier relatif)
 - Une base : b(2,8,10,16, ...)
 - Exemple : $0,5424 \cdot 10^2$ ₍₁₀₎; $10,1 \cdot 2^{-1}$ ₍₂₎; A0,B4. 16^{-2} ₍₁₆₎

CODAGE DE L'INFORMATION

$$x = \pm M \cdot 2^E$$

- où M est la mantisse (virgule fixe) et E l'exposant (signé).
 - Le codage en base 2, format virgule flottante, revient à coder le signe, la mantisse et l'exposant.
 - Exemple : Codage en base 2, format virgule flottante de (3,25)
 - $3,25_{(10)} = 11,01_{(2)}$ (en virgule fixe)
 - $= 1,101 \cdot 2^1_{(2)}$
 - $= 110,1 \cdot 2^{-1}_{(2)}$
- Un problème se pose : on a différentes manières de représenter E et M → normalisation*

CODAGE DE L'INFORMATION

- Le signe est codé sur 1 bit ayant le poids fort :
 - Le signe - : bit 1
 - Le signe + : bit 0
- Exposant biaisé (Eb)
 - Placé avant la mantisse pour simplifier la comparaison
 - Codé sur p bits et biaisé pour être positif (ajout de $2^{p-1}-1$) (pour 8 bit : $2^{8-1}-1 = 127$)
- Mantisse normalisé(M)
 - Normalisé : virgule est placé après le bit à 1 ayant le poids fort
 - M est codé sur q bits
- Exemple : $11,01 \rightarrow 1,101$ donc M =101

Eb = D + B
D → déplacement algébrique de la virgule
B → décalage ou biais = $2^{p-1}-1$

SM	Eb	M
1 bit	p bits	q bits

CODAGE DE L'INFORMATION

- Notre formule devient alors :

$$x = (-1)^S \cdot 2^{D+B} \cdot (1 + F)$$

- S est le bit de signe et l'on comprend alors pourquoi 0 est positif ($-1^0=1$)
- D+B = Eb (décalage ou exposant biaisé ou biais)
- F est la partie fractionnaire (mantisse)

IEEE 754 simple précision

SM	Eb	M
----	----	---

1 bit

8 bits

23 bits

IEEE 754 double précision

SM	Eb	M
----	----	---

1 bit

11 bits

52 bits

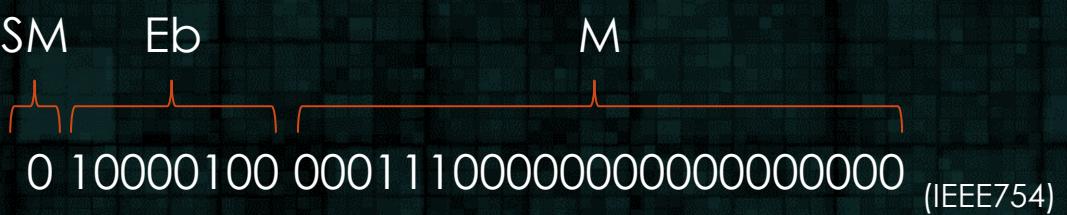
CODAGE DE L'INFORMATION

- Certaines conditions sont toutefois à respecter pour les exposants :
 - L'exposant 00000000 est interdit
 - L'exposant 11111111 est interdit. On s'en sert toutefois pour signaler des erreurs, on appelle alors cette configuration du nombre NaN, ce qui signifie « Not a Number ».

CODAGE DE L'INFORMATION

Conversion du décimal au binaire

- $35,5_{(10)} = ?$ (IEEE 754 simple précision)
- Nombre positif, donc SM = 0
- $35,5 = 100011,1_{(2)}$ (virgule fixe)
- $= 1,000111 \cdot 2^5_{(2)}$ (virgule flottante)
- Exposant : Eb = D + B \rightarrow Eb = 5+127, donc Eb = 132
- $1,M = 1,000111$ donc M = 00011100...



CODAGE DE L'INFORMATION

Conversion du décimal au binaire

- $S = 0$, donc nombre positif
- $E_b = 129$, donc $D = E_b - 127 = 2$
- $1.M = 1,111$



$$1,111 \cdot 2^2_{(2)} = 111,1_{(2)} = 7,5_{(10)}$$

RAPPELS

Arithmétique

ARITHMÉTIQUE

ADDITION BINAIRE ÉLÉMENTAIRE

- Les additions en base 2 s'effectuent comme dans le système décimal, avec la notion de retenue ou *carry* (ici en rouge) en utilisant la table d'addition suivante :

+	0	1
0	0	1
1	1	10

$$\begin{array}{l} 0 + 0 = 0 \\ 0 + 1 = 1 \\ 1 + 0 = 1 \\ 1 + 1 = 10 \\ 1 + 1 + 1 = 11 \end{array}$$

ARITHMÉTIQUE

ADDITION BINAIRE ÉLÉMENTAIRE

+	0	1
0	0	1
1	1	0

$$\begin{array}{r} 1 \\ 47 \\ + 23 \\ \hline = 70 \end{array}$$

$$\begin{array}{r} 111111 \\ 00101111 \\ + 00010111 \\ \hline = 01000110 \end{array}$$

ARITHMÉTIQUE

ADDITION BINAIRE ÉLÉMENTAIRE

$$\begin{array}{r} 01001 \quad (9)_{(10)} \\ + 00100 \quad (4)_{(10)} \\ \hline = 01101 \quad (13)_{(10)} \end{array}$$

$$\begin{array}{r} 101100 \quad (44)_{(10)} \\ + 010001 \quad (17)_{(10)} \\ \hline = 111101 \quad (61)_{(10)} \end{array}$$

$$\begin{array}{r} & 1 & 1 \\ 101011 & (43)_{(10)} \\ + 010011 & (19)_{(10)} \\ \hline = 111110 & (62)_{(10)} \end{array}$$

ARITHMÉTIQUE

SOUstraction BINAIRE ÉLÉMENTAIRE

$$\begin{array}{r} - 1 \quad 9 \quad 5 \\ - 1 \quad 9 \quad 6 \\ \hline = 0 \quad 9 \quad 9 \end{array}$$

$$\begin{array}{r} 19 \\ 9 \\ \hline 9 + 1 \end{array} \longrightarrow 10$$

$$\begin{array}{r} - 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \\ - 0 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \\ \hline = 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \end{array}$$

$$\begin{array}{r} 11 \\ 10 \\ 11 \\ \hline 1 + 1 \end{array} \longrightarrow 10 \quad 3_{(10)} \quad 2_{(10)} \quad 2_{(10)}$$

ARITHMÉTIQUE

ADDITION ET SOUSTRACTION BINAIRE SIGNÉES

- Une soustraction peut toujours, si on rend négatif son second terme, se ramener à une addition, ainsi :

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

- La méthode la plus utilisée pour rendre négatif un nombre binaire est le complément à 2
- C'est cette méthode qui est utilisée par les machines



Remarque : en représentation signée binaire,
le MSB représente le signe (0 si + et 1 si -).
Les nombres signés sont également formatés,
(i.e. qu'ils sont représentés sur un nombre fixe de bits).

ARITHMÉTIQUE

ADDITION ET SOUSTRACTION BINAIRE SIGNÉES

- Soit l'opération suivante : $195 - 96$
 - $195_{(10)} = 11000011_{(2)}$
 - $96_{(10)} = 01100000_{(2)}$
- En représentation signée binaire, +195 doit être représenté sur plus de 8 bits si l'on veut que son bit de signe soit positif. On va donc travailler sur 9 bits pour représenter son signe
 - $+195_{(10)} = 011000011_{(2)}$
 - $+96_{(10)} = 001100000_{(2)}$
 - $-96_{(10)} = 110100000_{(2)}$

$$\begin{array}{r} & \textcolor{green}{1} & \textcolor{green}{1} \\ & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 195 \\ + & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & +(-96) \\ \hline & = & \textcolor{red}{1} & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 99 \end{array}$$

Comme on travail sur 9 bits,
cette retenue est négligée

Signe +

ARITHMÉTIQUE

ADDITION ET SOUSTRACTION BINAIRE SIGNÉES

- L'opération de base des calculateurs électroniques est l'addition (c'est l'UAL qui s'en charge).
- Le résultat d'une soustraction de deux nombres binaires est en fait obtenu par l'addition du premier nombre par le complément à 2 du deuxième.
- Dans le cas d'un automate programmable, les nombres entiers sont stockées dans des mots formatés généralement sur 8, 16 ou 32 bits.
- Le bit de poids le plus fort (MSB) représente le signe (0 pour positif et 1 pour négatif).
- Étudions les différents cas possibles pour l'opération $D = A - B$

ARITHMÉTIQUE

ADDITION ET SOUSTRACTION BINAIRE SIGNÉES

- Cas 1 : deux nombres positifs
 - Exemple : A = +9 et B = +4
 - Sur 5 bits : A = 01001 et B = 00100
 - A + B
 - L'addition est immédiate

$$\begin{array}{r} 01001 \\ + 00100 \\ \hline = 01101 \end{array} \quad \begin{array}{r} + 9 \\ + 4 \\ \hline + 13 \end{array}$$

↑ Bits de signe ↑ Signe +

Remarque : En complément à 2, les nombres doivent toujours avoir le même nombre de bits



ARITHMÉTIQUE

ADDITION ET SOUSTRACTION BINAIRE SIGNÉES

- Cas 2 : un nombre positif et un nombre négatif plus petit

- Exemple : $A = +9$ et $B = -4$
- Sur 5 bits : $A = 01001$ et $B = -(00100)$
- $A + B = A + (-B)$
- $B_{(Cà2)} = 11100$

$$\begin{array}{r} 1 \ 1 \\ 0 \ 1 \ 0 \ 0 \ 1 \\ + \ 1 \ 1 \ 1 \ 0 \ 0 \\ \hline = 1 \ 0 \ 0 \ 1 \ 0 \ 1 \end{array}$$

+ 9
+ (-4)
+ 5

négligé

Signe +

ARITHMÉTIQUE

ADDITION ET SOUSTRACTION BINAIRE SIGNÉES

- Cas 3 : un nombre positif et un nombre négatif plus grand

- Exemple : $A = -9$ et $B = +4$
- Sur 5 bits : $A = -(01001)$ et $B = 00100$
- $-A + B = +(-A) + B$
- $A_{(C_2)} = 10111$

$$\begin{array}{r} & \textcolor{green}{1} \\ & 1 0 1 1 1 \\ + & \textcolor{blue}{0} 0 1 0 0 \\ \hline & = \textcolor{blue}{1} \ 1 0 1 1 \\ & \quad \downarrow \\ & \text{Signe -} \end{array} \quad \begin{array}{l} + (-9) \\ + 4 \\ - 5 \end{array}$$

- Le bit de signe de la somme est négatif, on doit compléter à 2 le résultat : $1011 = 0101_{(C_2)} = 5$, comme le bit de signe est 1, on obtient -5

ARITHMÉTIQUE

ADDITION ET SOUSTRACTION BINAIRE SIGNÉES

- Cas 4 : deux nombres négatifs

- Exemple : $A = -9$ et $B = -4$
- Sur 5 bits : $A = -(01001)$ et $B = -(00100)$
- $-A - B = +(-A) + (-B)$
- $A_{(Cà2)} = 10111$
- $B_{(Cà2)} = 11100$

$$\begin{array}{r} 1 \ 1 \ 1 \\ 1 \ 0 \ 1 \ 1 \ 1 \\ + 1 \ 1 \ 1 \ 0 \ 0 \\ \hline = 1 \ 1 \ 0 \ 0 \ 1 \ 1 \end{array}$$

négligé Signe -

$+ (-9)$
 $+ (-4)$
 -13

- Le bit de signe de la somme est négatif, on doit compléter à 2 le résultat : $0011 = 1101_{(Cà2)} = 13$, comme le bit de signe est 1, on obtient -13

ARITHMÉTIQUE

ADDITION ET SOUSTRACTION BINAIRE SIGNÉES

- Cas 5 : deux nombres égaux et opposés

- Exemple : $A = -9$ et $B = +9$
- Sur 5 bits : $A = -(01001)$ et $B = 01001$
- $-A + B = +(-A) + B$
- $A_{(Ca2)} = 10111$

$$\begin{array}{r} 11111 \\ 10111 \\ + 01001 \\ \hline = 100000 \end{array}$$

négligé Signe -

$+ (-9)$
 $+ (9)$
 $+0$

- Le bit de signe de la somme est positif, on a donc +0

ARITHMÉTIQUE

ADDITION ET SOUSTRACTION BINAIRE SIGNÉES

- L'addition de deux nombres de même signe peut donner lieu à un dépassement de capacité (OverFlow) ! Le résultat obtenu est alors faux!!
- On a un dépassement de capacité quand le bit de signe du résultat est différent de celui des deux nombres additionnés.
 - Le nombre de bits utilisés est insuffisant pour contenir le résultat
 - Autrement dit le résultat dépasse l'intervalle des valeurs sur les n bits utilisés (sur 8 bits, on dépasse l'intervalle -128 et +127)

ARITHMÉTIQUE

ADDITION ET SOUSTRACTION BINAIRE SIGNÉES

$$\begin{array}{r} 1 \textcolor{green}{111111} \\ 10011001 \\ + 10111111 \\ \hline = 101011000 \\ 88 \end{array}$$

$$\begin{array}{r} + (-103) \\ + (-65) \\ \hline = +(-168) \end{array}$$

$$\begin{array}{r} 1 \textcolor{green}{111} \\ 01100111 \\ + 01000001 \\ \hline = 10101000 \\ -88 \end{array}$$

- On est jamais confronté à un dépassement de capacité lorsque les signes des deux nombres sont opposés

ARITHMÉTIQUE

EXERCICES

- On dispose d'une machine travaillant sur des nombres binaires de longueur 8 (8 bits). Faire manuellement ce que l'additionneur de la machine ferait automatiquement, et donner les résultats obtenus en binaire.
Eventuellement, en cas d'erreur, indiquer pourquoi.

$$A = -61 \quad B = -44 \quad C = -72 \quad D = 99 - 35 \quad E = 99 + 35$$

ARITHMÉTIQUE

ADDITION ET SOUSTRACTION BINAIRE : FLOTTANTS

- Soit deux nombres réels N_1 et N_2 tel que

$$N_1 = M_1 \cdot b^{e_1} \quad N_2 = M_2 \cdot b^{e_2}$$

- On veut calculer $N_1 + N_2$?
- Deux cas se présentent :
 - Si $e_1 = e_2$ alors $N_3 = (M_1 + M_2) \cdot b^{e_1}$
 - Si $e_1 \neq e_2$ alors éléver au plus grand exposant et faire l'addition des mantisses et par la suite normalisée la mantisse du résultat.

ARITHMÉTIQUE

ADDITION ET SOUSTRACTION BINAIRE : FLOTTANTS

- Exemple 1
- Soient les nombres A = 1,5 et B = 0,5
- Représentation en IEEE :
 - A = +1,1₍₂₎ (virgule fixe) avec un signe + (donc S = 0)
 - A = +1,1 . 2⁰ (virgule flottante)
 - B = +0,1₍₂₎ (virgule fixe)
 - B = +1,0 . 2⁻¹ (virgule flottante)
 - A > 0 et B > 0 → S_A = 0 et S_B = 0
 - E_{b_A} = 0+127 = 127 E_{b_B} = -1 + 127 = 126
 - A = 0 01111111 10000000000000000000000000 avec 1,M = 1,10000000...
 - B = 0 01111110 00000000000000000000000000 avec 1,M = 1,00000000...

ARITHMÉTIQUE

ADDITION ET SOUSTRACTION BINAIRE : FLOTTANTS

- On ramène les deux nombres au même exposant, le plus grand des deux.
- On décale donc les bits de la mantisse du nombre ayant le plus petit exposant d'autant de bits vers la droite que la différence entre les exposants, sans oublier le 1 avant la virgule.
- Dans l'exemple on veut augmenter de 1 l'exposant du deuxième terme B. La mantisse complète du second nombre passe donc de 1.0000000000... à 0.1000000000... (on supprime le zéro le plus à droite pour rester sur 23 bits).
- On ajoute ensuite les deux mантisses, en tenant compte du signe (attention ici les mантisses ne sont pas exprimées en complément à 2) Dans l'exemple les mантisses sont de même signe on peut donc les ajouter directement :

$$\begin{array}{r} 1,1000000000000000000000000 \\ + 0,1000000000000000000000000 \\ \hline = 10,00000000000000000000000 \end{array} \quad \begin{array}{l} \times 2^0 \\ \times 2^0 \\ \hline \times 2^0 \end{array}$$

ARITHMÉTIQUE

ADDITION ET SOUSTRACTION BINAIRE : FLOTTANTS

- On renormalise ensuite le nombre obtenu.
- Dans l'exemple, le résultat a donc pour exposant 0 et pour mantisse 10.000000000... : on renormalise la mantisse, ce qui augmente l'exposant de 1.
- On a donc le résultat final :
- $0\ 10000000\ 0000000000000000000000000 = +1,0 \cdot 2^1 = +2_{(10)}$

ARITHMÉTIQUE

ADDITION ET SOUSTRACTION BINAIRE : FLOTTANTS

- Exemple 2
- Soient les nombres $A = -1,5$ et $B = -0,5$
- Représentation en IEEE :
 - $A = -1,1_{(2)}$ (virgule fixe) avec un signe + (donc $S = 0$)
 - $A = -1,1 \cdot 2^0$ (virgule flottante)
 - $B = -0,1_{(2)}$ (virgule fixe)
 - $B = -1,0 \cdot 2^{-1}$ (virgule flottante)
 - $A < 0$ et $B < 0 \rightarrow S_A = 1$ et $S_B = 1$
 - $Eb_A = 0+127 = 127 \quad Eb_B = -1 + 127 = 126$
 - $A = 1\ 0111111\ 10000000000000000000000000000000$ avec $1,M = 1,1000000...$
 - $B = 1\ 0111110\ 00000000000000000000000000000000$ avec $1,M = 1,0000000000...$

ARITHMÉTIQUE

ADDITION ET SOUSTRACTION BINAIRE : FLOTTANTS

- On ramène les deux nombres au même exposant, le plus grand des deux
 - $M_A = 1,1000000000000000000000000 \times 2^0$
 - $M_B = 0,1000000000000000000000000 \times 2^0$
- On ajoute ensuite les deux mantisses. Comme les deux mantisses sont de même signe, on les ajoute directement :

$$\begin{array}{r} 1,1000000000000000000000000 \\ + 0,1000000000000000000000000 \\ \hline = 10,00000000000000000000000 \end{array} \quad \begin{array}{l} \times 2^0 \\ \times 2^0 \\ \hline \times 2^0 \end{array}$$

- On renormalise ensuite le nombre obtenu : $1,0000\dots \times 2^1$
- Le signe de la mantisse est 1, donc $A + B = -2$

ARITHMÉTIQUE

ADDITION ET SOUSTRACTION BINAIRE : FLOTTANTS

- Exemple 3
- Soient les nombres A = 1,5 et B = -0,5
- Représentation en IEEE :
 - A = +1,1₍₂₎ (virgule fixe) avec un signe + (donc S = 0)
 - A = +1,1 . 2⁰ (virgule flottante)
 - B = -0,1₍₂₎ (virgule fixe)
 - B = -1,0 . 2⁻¹ (virgule flottante)
 - A > 0 et B < 0 → S_A = 0 et S_B = 1
 - E_b_A = 0+127 = 127 E_b_B = -1 + 127 = 126
 - A = 0 01111111 10000000000000000000000000 avec 1,M = 1,1000000...
 - B = 1 01111110 00000000000000000000000000 avec 1,M = 1,000000000...

ARITHMÉTIQUE

ADDITION ET SOUSTRACTION BINAIRE : FLOTTANTS

- On ramène les deux nombres au même exposant, le plus grand des deux
 - $M_A = 1,100000000000000000000000 \times 2^0$
 - $M_B = 0,100000000000000000000000 \times 2^0$
- Comme les signes des mantisses sont différents, on garde le signe du plus grand nombre, donc celui de A : signe +
- Le signe de B est < 0, on complémente la mantisse à 2
 - $1,100000000000000000000000 \times 2^0$ (Cà2)
- On ajoute ensuite les deux mantisses

ARITHMÉTIQUE

ADDITION ET SOUSTRACTION BINAIRE : FLOTTANTS

$$\begin{array}{r} 1,100000000000000000000000000000 \\ + 1,100000000000000000000000000000 \\ \hline = 11,0000000000000000000000000000 \\ = 01,0000000000000000000000000000 \end{array} \quad \begin{array}{l} \times 2^0 \\ \times 2^0 \\ \hline \end{array} \quad \begin{array}{l} \times 2^0 \\ \times 2^0 \end{array}$$

Le résultat est négatif, on complémente à nouveau

- On renormalise ensuite le nombre obtenu : 01,0000... $\times 2^0$
- Le signe est positif : $A + B = +1$
- Même chose pour $A = -1,5$ et $B = 0,5$, $A+B = -1$

ARITHMÉTIQUE

ADDITION HEXADÉCIMALES : ENTIERS POSITIFS

A	B	C	D	E	F
10	11	12	13	14	15

■ Cas 1: sans retenue

$$\begin{array}{r} 34\ B\ 5 \\ + 66\ 1\ 4 \\ \hline = 9\ A\ C\ 9 \end{array}$$

$5 + 4 = 9$
 $B + 1 = 11 + 1 = 12 = C$
 $4 + 6 = 10 = A$
 $3 + 6 = 9$

■ Cas 2 : avec la retenue

$$\begin{array}{r} 52\ 6\ 4 \\ + A3\ 2\ E \\ \hline = F5\ 9\ 2 \end{array}$$

1
 $4 + E = 18$ (16 en retenue + 2 au résultat)
 $1 + 6 + 2 = 9$
 $2 + 3 = 5$
 $5 + A = 5 + 10 = 15 = F$

ARITHMÉTIQUE

ADDITION HEXADÉCIMALES : ENTIERS POSITIFS

■ Cas 2 : avec la retenue

$$\begin{array}{r} 1 \quad 1 \\ 4 \ B \ C \ 3 \\ + \ 2 \ A \ \ 2 \ F \\ \hline = \ 7 \ 5 \ \ F \ 2 \end{array}$$

$$3 + F = 18 \text{ (16 en retenue + 2 au résultat)}$$

$$1 + C + 2 = F$$

$$B + A = 11 + 10 = 21 \text{ (16 en retenue + 5 au résultat)}$$

$$1 + 4 + 2 = 7$$

$$\begin{array}{r} 1 \ 1 \ 1 \\ F \ F \ F \ F \\ + \ F \ F \ F \ F \\ \hline = \ 1 \ F \ F \ F \ E \end{array}$$

$$F + F = 15 + 15 = 30 \text{ (16 en retenue + 14 au résultat = E)}$$

$$1 + F + F = 1 + 15 + 15 = 31 \text{ (16 en retenue + 15 au résultat = F)}$$

$$1 + F + F = 1 + 15 + 15 = 31 \text{ (16 en retenue + 15 au résultat = F)}$$

$$1 + F + F = 1 + 15 + 15 = 31 \text{ (16 en retenue + 15 au résultat = F)}$$

1 en retenue à remettre au résultat

ARITHMÉTIQUE

SOUstraction HEXADÉCIMALES : ENTIERS POSITIFS

A	B	C	D	E	F
10	11	12	13	14	15

$$\begin{array}{r} 9 \text{ B } 5 \text{ } 14 \\ - 6 \text{ A } 2 \text{ } 1 \text{ } 9 \\ \hline = 3 \text{ } 12 \text{ } \text{ B} \end{array}$$

$$\begin{aligned} 4 - 9 &= (4 + 16) - 9 = 11 = \text{B} \\ 5 - 3 &= 5 - (2 + 1 \text{ de retenue}) = 2 \\ \text{B} - \text{A} &= 11 - 10 = 1 \\ 9 - 6 &= 3 \end{aligned}$$

$$\begin{array}{r} \text{A } 1\text{ B } 1\text{ C } 1\text{ D} \\ - 2 \text{ } \text{ F } \text{ } \text{ F } \text{ } \text{ F} \\ \hline = 7 \text{ } \text{ B } \text{ } \text{ C } \text{ } \text{ E} \end{array}$$

$$\begin{aligned} \text{D} - \text{F} &= 13 - 15 = -2 = 14 = \text{E} \\ \text{C} - \text{F} &= 12 - 15 = -3 = 13 = \text{C} \\ \text{B} - \text{F} &= 11 - 15 = -4 = 12 = \text{B} \\ \text{A} - (2+1) &= 10 - 3 = 7 \end{aligned}$$

Remarque : Cas où le premier nombre est supérieur au second sinon, il faut passer par le complément à 16



ARITHMÉTIQUE

SOUstraction HEXADÉCIMALES : COMPLÉMENT À 16

A	B	C	D	E	F
10	11	12	13	14	15

$$-9B54_{(16)} = 1001\ 1011\ 0101\ 0100_{(\text{bs})} = 0110\ 0100\ 1010\ 1100_{(\text{C}_2)} = 64AC_{(\text{C}_2\text{à}16)}$$

$$\begin{array}{r} 6\ A\ 2\ 9 \\ + 6\ 4\ A\ C \\ \hline = C\ E\ D\ 5 \end{array} \quad \begin{array}{l} 9 + C = 9 + 12 = 21 (16 \text{ en retenue} + 5 \text{ au résultat}) \\ 1 + 2 + A = 1 + 2 + 10 = 13 = D \\ A + 4 = 10 + 4 = 14 = E \\ 6 + 6 = 12 = C \end{array}$$

$$CED5 = 1100\ 1110\ 1101\ 0101 = 0011\ 0001\ 0010\ 1011 = -312B$$

Signe -, on complémente à nouveau

ARITHMÉTIQUE

SOUstraction HEXADÉCIMALES : COMPLÉMENT À 16

A	B	C	D	E	F
10	11	12	13	14	15

15 15 15 15
9 B 5 4

6 4 A B complément à 1
+ 1

6 4 A C complément à 2

6 A 2 9 $9 + C = 9 + 12 = 21$ (16 en retenue + 5 au résultat)
+ 6 4 A C $1 + 2 + A = 1 + 2 + 10 = 13 = D$

= C E D 5 $A + 4 = 10 + 4 = 14 = E$

$$6 + 6 = 12 = C$$

$$CED5 = (15-C)(15-E)(15-D)(15-5) = 312A + 1 = -312B$$

ARITHMÉTIQUE

MULTIPLICATION BINAIRE : ENTIERS POSITIFS

*	0	1
0	0	0
1	0	1

$$\begin{array}{r} 101101 \\ * \quad 101 \\ \hline = \quad 101101 \\ \quad \quad 000000 \\ \quad \quad 101101 \\ \hline \end{array} \quad \begin{array}{r} 45 \\ \times \quad 5 \\ \hline 225 \end{array}$$

ARITHMÉTIQUE

MULTIPLICATION BINAIRE : ENTIERS SIGNÉS

- Cas de deux nombres positifs :
 - on applique la même méthode que précédemment
- Cas de deux nombres négatifs :
 - On calcul le complément à 2 de chacun des nombres (pour les rendre positifs)
 - On multiplie ces deux nombres positifs comme précédemment
 - Le résultat obtenu est positif, on ne change rien
- Cas où l'un des deux nombres est négatif :
 - On calcul le complément à deux du nombre négatif
 - On multiplie les deux nombres positifs
 - On complémente à deux le résultat, puisqu'il est négatif

ARITHMÉTIQUE

DIVISION BINAIRE : ENTIERS POSITIFS



$$1001100001/101 = 1111001,110011001100\dots$$

ARITHMÉTIQUE

EXERCICES

- Effectuez la division suivante :

- $10001/100$

100 reste 1

- $11001/101$

101 reste 0

- $1111100/1001000111$

0 reste 1111100

- $111111110010/1000011000$

1111 reste 10001010

TRAVAIL À FAIRE

- Vous devez vous renseigner sur l'encodage des jeux de caractères : ASCII, ISO-8859-1, ISO latin-1 .. 9, Unicode
- Les codes détecteurs et correcteurs d'erreurs : contrôle de parité, Code de Hamming, Code Huffman et algorithme de Huffman

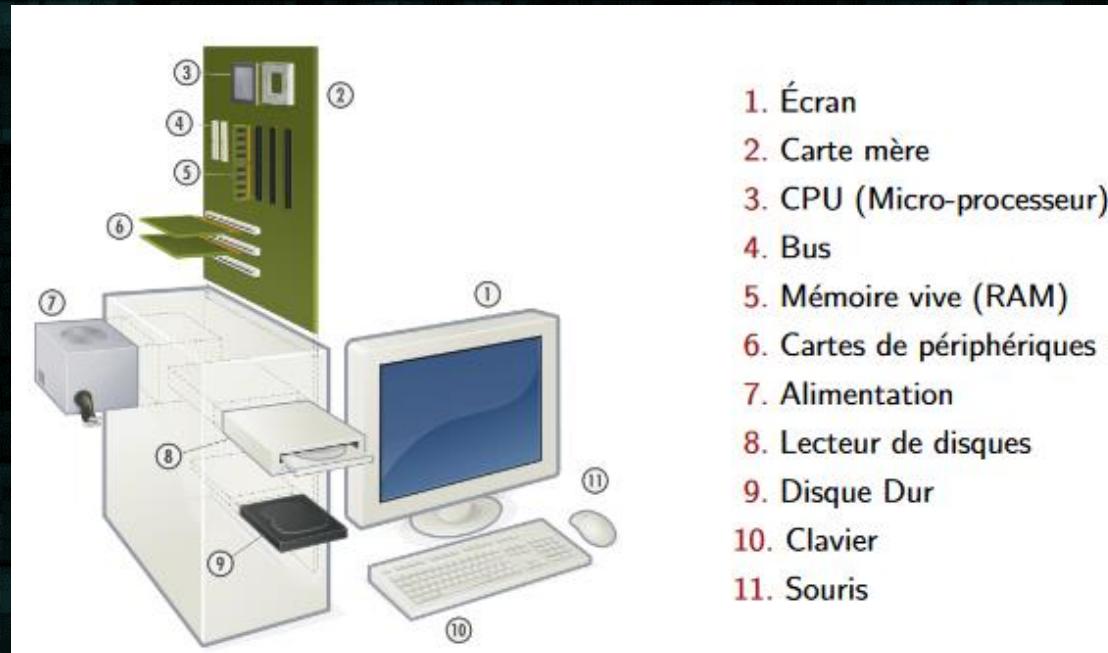
RAPPELS

Structure d'un ordinateur

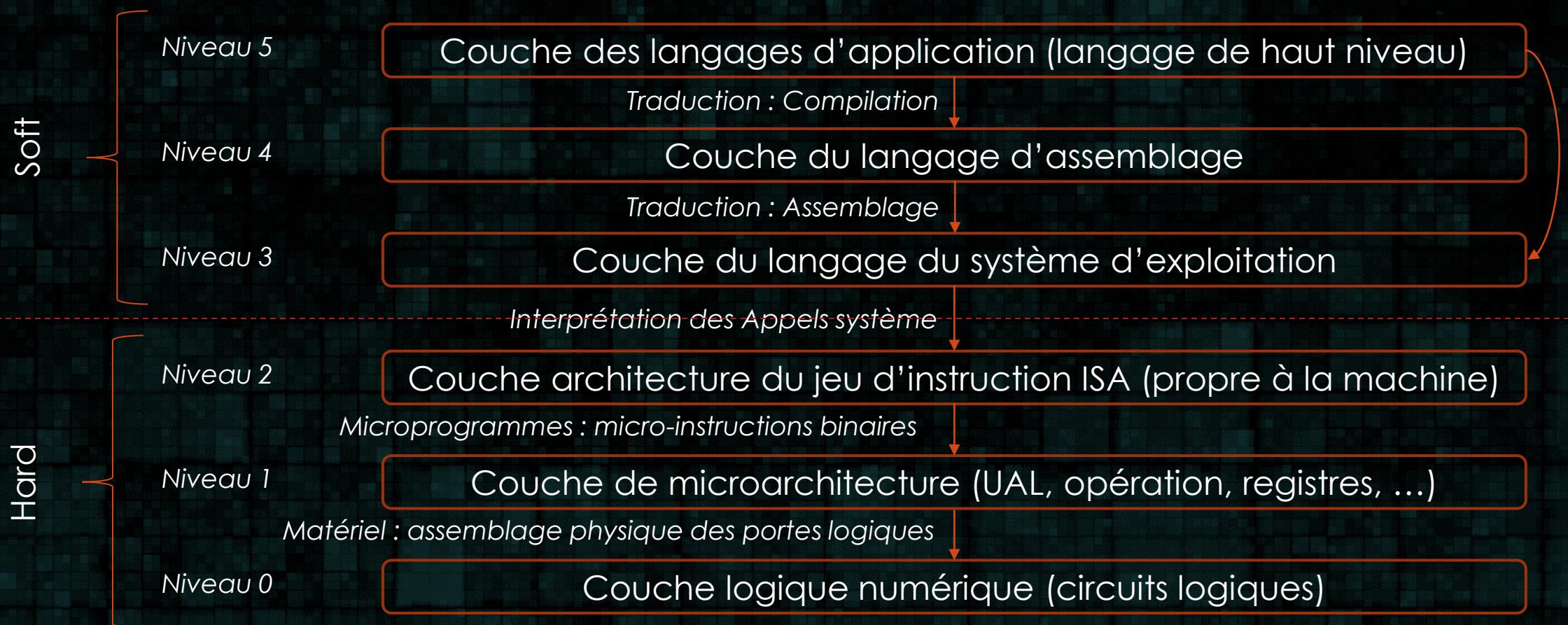
STRUCTURE D'UN ORDINATEUR

- Ordinateur
 - Machine capable de résoudre des problèmes en appliquant des instructions
- Instruction
 - Action à effectuer par l'ordinateur, correspondant à une étape dans un programme
- Programme
 - Suite d'instructions décrivant la façon dont l'ordinateur doit effectuer un travail
- Langage machine
 - Ensemble des instructions exécutables directement par un ordinateur

STRUCTURE D'UN ORDINATEUR

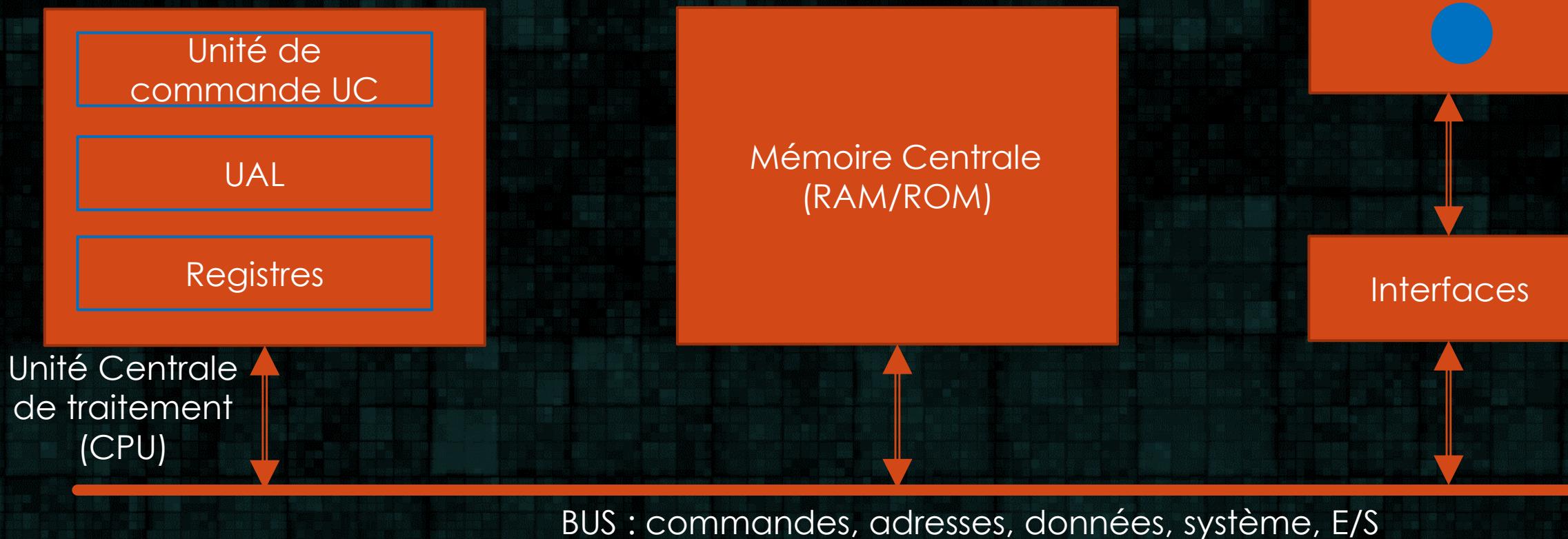


ARCHITECTURE EN COUCHES

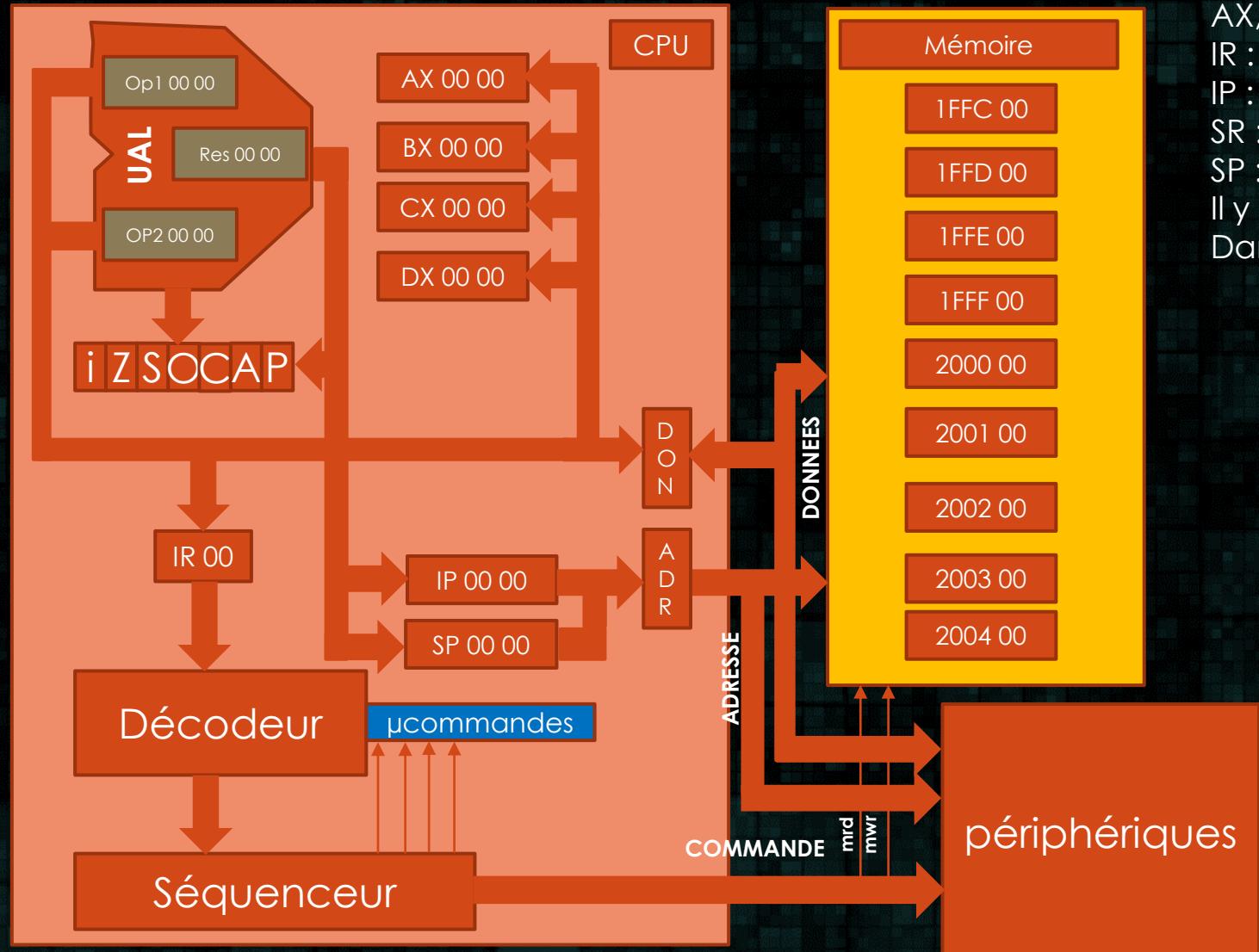


STRUCTURE D'UN ORDINATEUR

■ Le modèle de Von Neumann



Schématisation simplifiée de l'architecture Intel x86



AX, BX, CX, DX : Registres de données, 16bits, R/W

IR : Registre d'instruction, 8bits, R

IP : Compteur Ordinal, 16bits, R/W

SR : Registre d'état, 16bits, R/W

SP : Pointeur de pile, 16bits, R/W

Il y a d'autres types de registres qui n'apparaissent pas dans ce schéma.

L'unité de commande exécute l'algorithme suivant :
répéter

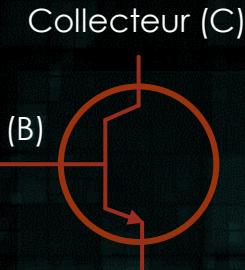
1. charger dans IR l'instruction stockée en MC à l'adresse pointée par le IP;
2. IP:=IP+taille(instruction en IR);
3. décoder (IR) en micro-instructions;
4. localiser en mémoire les données de l'instruction;
5. charger les données;
6. exécuter l'instruction (suite de micro-instructions);
7. stocker les résultats mémoires; jusqu'à l'infini

RAPPELS

Circuits logiques

CIRCUITS LOGIQUES

- Les circuits logiques sont élaborés à partir de composants électroniques : les transistors
- Un transistor peut être vu comme un interrupteur. Il comporte 3 broches
 - Lorsque l'on applique 0V sur la broche Base, le transistor n'est pas conducteur : il est dit **bloqué**.
 - Lorsque l'on applique +5V sur la broche Base, le transistor devient conducteur : il est dit **saturé**.
- Il existe deux types de circuits logiques :
 - Les circuits combinatoires
 - Les circuits séquentiels



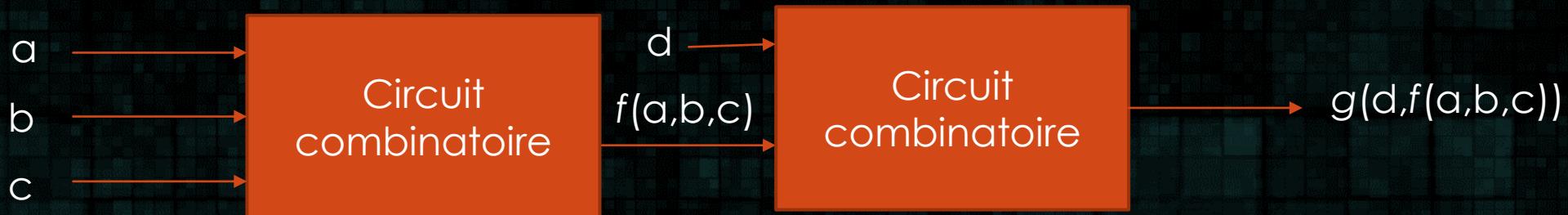
CIRCUITS LOGIQUES

- Les circuits combinatoires :
 - Support théorique → algèbre de Boole
 - Un circuit physique élaboré à partir de composants électroniques
 - Un circuit combinatoire est défini par (ou réalise) une ou plusieurs fonctions logiques booléennes



CIRCUITS LOGIQUES

- Les circuits peuvent être mis en série pour réaliser des compositions de fonction :



CIRCUITS LOGIQUES

- Circuits séquentiels ou à mémoire :
 - Support théorique → FSM (Finite State Machine)
 - Les fonctions de sortie dépendent non seulement de l'état des variables d'entrée mais également de **l'état antérieur de certaines variables de sortie** (propriétés de mémorisation)



CIRCUITS COMBINATOIRES

- Le circuit combinatoire est défini lorsque:
 - Son nombre d'entrées est précisé,
 - Son nombre de sorties est précisé
 - L'état de chaque sortie en fonction des entrées a été précisé
- Ces informations sont fournies grâce à une table de vérité
 - La table de vérité d'une fonction de n variables contient 2^n lignes - états d'entrée

CIRCUITS COMBINATOIRES



i_0	i_1	$f_0(i_0, i_1)$
0	0	
0	1	
1	0	
1	1	

i_1	i_3	i_4	$f_1(i_1, i_3, i_4)$
0	0	0	
0	0	1	
0	1	0	
0	1	1	
...			
1	1	1	

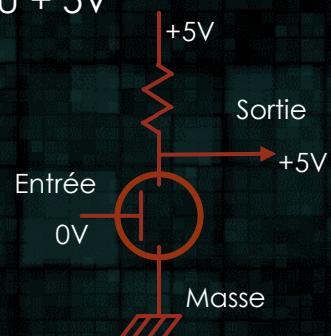
A large orange arrow points from the bottom-left towards the second table, indicating a transition or relationship between the two.

i_0	i_1	i_2	\dots	i_n	$f_0(i_0, i_1)$	$f_1(i_1, i_3, i_4)$	\dots	$f_m(i_9, i_{12}, i_n)$
0	0	0	\dots	0				
0	0	0	\dots	1				
...								
...								
1	1	1	\dots	1				

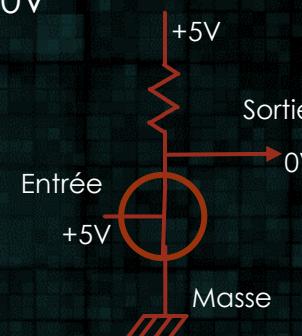
CIRCUITS COMBINATOIRES

- L'opérateur NOT (inverseur) : Une seule entrée et une seule sortie
 - La sortie d'une fonction NON prend l'état 1 si et seulement si son entrée est dans l'état 0

Lorsque l'on applique 0V à l'entrée, le transistor est bloqué et se comporte comme un circuit ouvert. Par conséquent, la sortie apparaît reliée au + 5V



Lorsque l'on applique +5V à l'entrée, le transistor est saturé et se comporte comme un court-circuit. Par conséquent, la sortie apparaît reliée au 0V



Tension électrique	Valeur binaire
0 V	0
+5V	1

CIRCUITS COMBINATOIRES

- L'opérateur NOT (inverseur)
 - *L'inverseur est constitué d'un seul transistor et transforme une donnée binaire en son inverse*

Entrée	Sortie
0	1
1	0



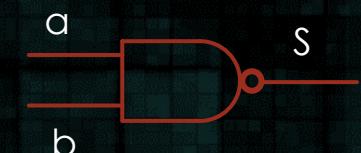
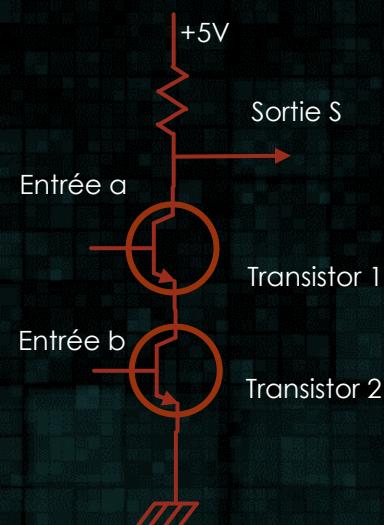
Représentation schématique
d'un inverseur NOT

CIRCUITS COMBINATOIRES

■ L'opérateur NAND : au moins deux entrées

- Les transistors peuvent être associés pour réaliser des opérateurs plus complexes qu'une simple inversion en sortie d'un bit en entrée.
- Ici on utilise deux transistors en série :
- La table de vérité* est la suivante :

Entrée	Sortie
a	S
0	0
0	1
1	0
1	1

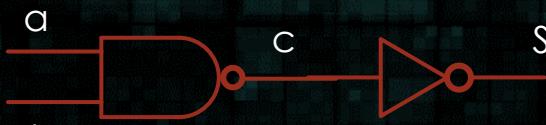


Représentation schématique
d'un NAND

* Une table de vérité résume le fonctionnement d'un composant. Il examine toutes les possibilités binaires en entrée et donne, pour chacune, le résultat en sortie.

CIRCUITS COMBINATOIRES

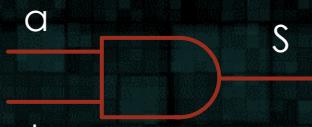
- L'opérateur AND : au moins deux entrées
 - On peut très facilement fabriquer une porte ET en chaînant une porte NAND et un inverseur :



Représentation schématique
d'un AND

Entrée NAND	Sortie NAND	Sortie NOT	
a	b	c	S
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

- La porte AND est tellement utilisée en électronique, quelle possède son propre schéma et sa propre table de vérité (c'est celle-là qu'il faut retenir).



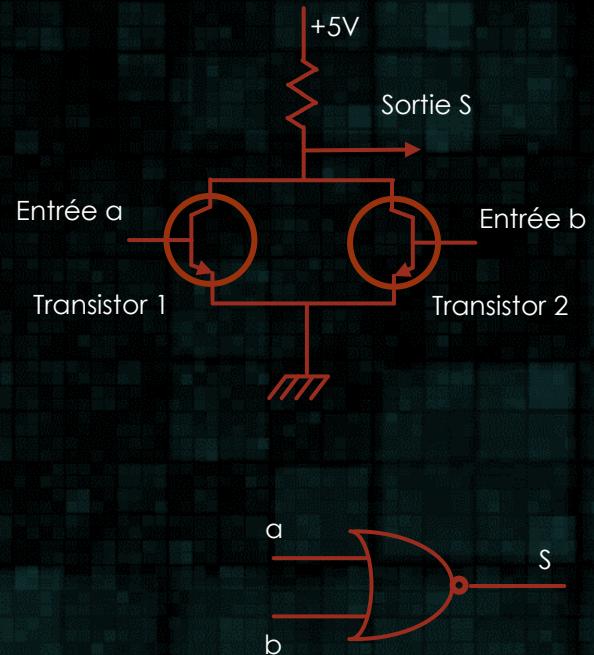
Représentation schématique
d'un AND

Entrée	Sortie	
a	b	S
0	0	0
0	1	0
1	0	0
1	1	1

CIRCUITS COMBINATOIRES

- L'opérateur NOR : au moins deux entrées
 - On met deux transistors en parallèle :

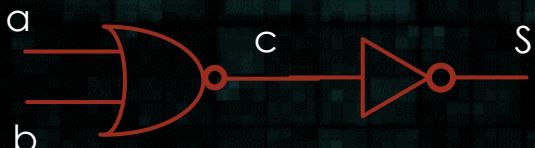
Entrée	Sortie
a	b
0	0
0	1
1	0
1	1



Représentation schématique
d'un NOR

CIRCUITS COMBINATOIRES

- L'opérateur OR : au moins deux entrées
 - Comme pour le ET, on chaîne le NOR avec un NON :



Représentation schématique
d'un OR

Entrée NOR	Sortie NOR	Sortie NOT
a 0	b 0	c 1
0	1	0
1	0	0
1	1	0

- La porte OR est tellement utilisée en électronique, quelle possède son propre schéma et sa propre table de vérité (c'est celle-là qu'il faut retenir).



Représentation schématique
d'un OR

Entrée	Sortie
a 0	b 0
0	1
1	0
1	1

CIRCUITS COMBINATOIRES

- L'opérateur XOR : au moins deux entrées
 - Dernier opérateur logique, le XOR est fréquemment utilisé en informatique.
 - Voici ses caractéristiques :

Entrée	Sortie
a	b
0	0
0	1
1	0
1	1



Représentation schématique
d'un XOR

CIRCUITS COMBINATOIRES

- Toute fonction logique peut être réalisée à l'aide des portes élémentaires
- La réalisation d'une fonction booléenne se fait en trois étapes :
 - Écrire l'équation de la fonction à partir de sa table de vérité
 - Simplifier l'équation (diminuer le nombre d'opérateurs et donc de portes logiques)
 - Réaliser l'équation à l'aide des portes disponibles

CIRCUITS COMBINATOIRES

- À partir de la table de vérité, nous pouvons avoir deux formes analytiques, dénommées formes canoniques :
 - Somme canonique de produits SOP = disjonctions de conjonctions
 - Produit canonique de sommes POS = conjonctions de disjonctions
- **Définition** : une fonction est dite sous forme canonique SOP ou POS si et seulement si dans chaque terme de la fonction figure toute les variables de la fonction

CIRCUITS COMBINATOIRES

- Exemple de SOP :

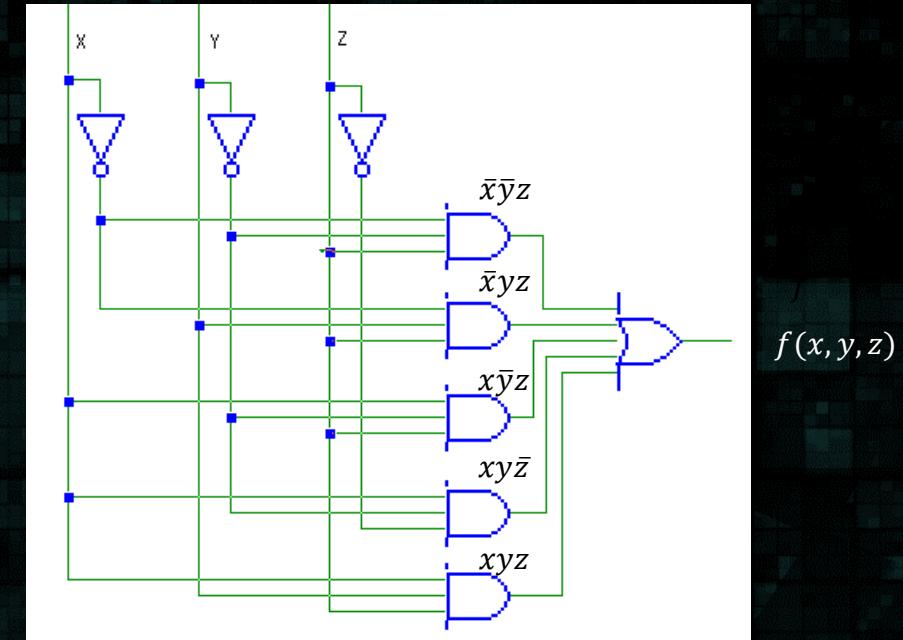
- soient les 3 variables x, y et z et $f(x,y,z) = x.y + z$, alors la table de vérité associée est :

La fonction regroupe 5 min-termes

Pour la ligne 2 $f(0,0,1)=1$, le min-terme est $(\bar{x} \cdot \bar{y} \cdot z)$

f sous sa forme SOP est donc :

$$f(x,y,z) = \bar{x}\bar{y}z + \bar{x}yz + x\bar{y}z + xy\bar{z} + xyz = \sum_{min}(1,3,5,6,7)$$



x	y	z	f = x.y+z
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

CIRCUITS COMBINATOIRES

- Exemple de POS :

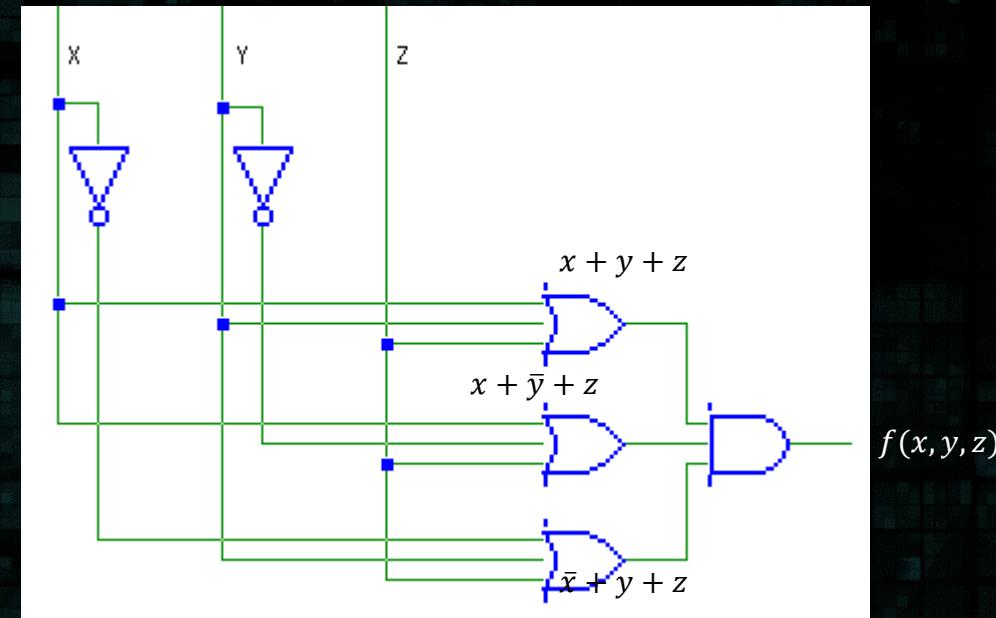
- soient les 3 variables x, y et z et $f(x,y,z) = x \cdot y + z$, alors la table de vérité associée est :

La fonction regroupe 3 max-termes

Pour la ligne 3 $f(0,1,0)=0$, le min-terme est $(x + \bar{y} + z)$

f sous sa forme POS est donc :

$$f(x,y,z) = (x + y + z)(x + \bar{y} + z)(\bar{x} + y + z) = \prod_{max} (0,2,4)$$



x	y	z	$f = x \cdot y + z$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

CIRCUITS COMBINATOIRES

- Soucis majeurs des concepteurs
 - Réduire le nombre de portes nécessaires à la réalisation des systèmes
 - Minimiser le coût en nombre de boîtiers
 - La consommation électrique
 - Minimiser la complexité
 - Créer un système équivalent avec certains paramètres optimisés

ALGÈBRE DE BOOLE

- Résumé des règles de calcul de l'algèbre de Boole à connaître :

Constantes

$$a + 0 = a$$
$$a + 1 = 1$$

$$a \cdot 0 = 0$$
$$a \cdot 1 = a$$

Idempotence

$$a + a = a$$

$$a \cdot a = a$$

Complémentation

$$a + \bar{a} = 1$$

$$a \cdot \bar{a} = 0$$

Commutativité

$$a + b = b + a$$

$$a \cdot b = b \cdot a$$

Distributivité

$$a + (bc) = (a + b)(a + c)$$
$$a(b + c) = (ab) + (ac)$$

Associativité

$$a + (b + c) = (a + b) + c = a + b + c$$
$$a(bc) = (ab)c = abc$$

Lois de De Morgan

$$\overline{ab} = \bar{a} + \bar{b}$$

$$\overline{a + b} = \bar{a}\bar{b}$$

L'implication

$$a \rightarrow b = \bar{a} + b$$

Loi d'absorption

$$a + (ab) = a$$
$$a(a + b) = a$$

Autres relations

$$\bar{\bar{a}} = a$$

$$(a + b)(a + \bar{b}) = a$$

XOR

$$a \oplus b = (a + b)\overline{(ab)} = (ab) + (b\bar{a}) = \overline{(ab)} + \overline{(a\bar{b})} = (a + b)(\bar{a} + \bar{b}) = \bar{a}b + a\bar{b}$$

ALGÈBRE DE BOOLE

- Les tableaux de Karnaugh :
 - La table est un diagramme formé de carrés dont le nombre est égale à 2^n , n représentant le nombre de variables de la fonction logique à simplifier
 - Chaque carré représente un min-terme marqué par « 1 » (les « 0 » de la table représentent les max-termes) de la fonction booléenne

ALGÈBRE DE BOOLE

- On représente un tableau à 2 dimensions
- Chaque dimension concerne une ou 2 variables
- Le passage d'une colonne à une colonne adjacente ou d'une ligne à une ligne adjacente modifie la valeur d'une seule variable
- Le tableau se referme sur lui-même : la colonne la plus à gauche est voisine de la colonne la plus à droite, idem pour les lignes du haut et du bas
 - Pour les 2 colonnes (2 lignes) extrêmes, là aussi, une seule variable doit changer de valeur entre ces 2 colonnes (lignes)
- Une case du tableau contient une valeur booléenne, déterminée à partir de la table de vérité et des valeurs des variables

	ab	00	01	11	10
cd					
00					
01					
11					
10					

ALGÈBRE DE BOOLE

■ Méthodologie

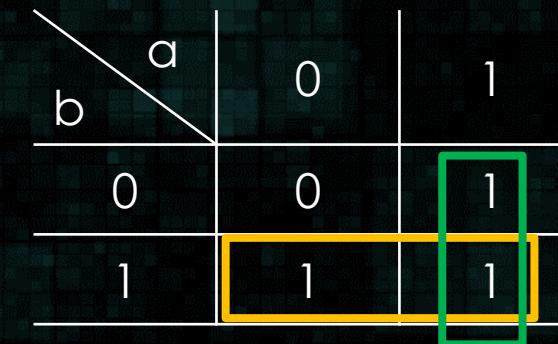
- Regroupement en blocs rectangulaires des bits à 1 adjacents
 - Tous les bits à 1 du tableau doivent être englobés dans au moins un bloc (un bloc à une taille de 1, 2, 4, 8 ... bits)
 - On doit créer les blocs les plus gros possibles
 - Un bit à 1 peut appartenir à plusieurs blocs si cela permet de créer des blocs plus gros
- A chaque bloc correspond un terme formé comme suit
 - Si une variable dans le bloc change de valeur (valeurs 0 et 1 pour des cases différentes), on ne la prend pas en compte
 - On ne conserve que les variables qui ne varient pas.
 - Si une variable a reste à 1 : on note a , si reste à 0 : on note \bar{a}
 - Le terme logique du bloc correspond au ET de ces variables qui ne changent pas
- La fonction logique simplifiée est le OU de tous les termes des blocs trouvés = SOP

		ab	00	01	11	10
		cd	00	01	11	10
00	01	00	1	0	1	1
		01	1	1	0	0
11	10	11	1	0	0	0
		10	1	1	1	0

ALGÈBRE DE BOOLE

- Exemple avec 2 variables :

a	b	f(a,b)
0	0	0
0	1	1
1	0	1
1	1	1



- 2 groupes de 2 bits adjacents :

- Pour le vertical : on a toujours a = 1 donc cela donne le terme a
- Pour l'horizontal : idem mais avec b

$$f(a, b) = a + b$$

ALGÈBRE DE BOOLE

- Exemple avec 3 variables :

a	b	c	f(a,b,c)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Bloc le plus petit :

a passe de 0 à 1, on ne la prendra pas en compte

b reste à 0 et c reste à 1

Donne le terme $\bar{b}c$

$$f(a, b, c) = a + \bar{b}c$$



Bloc le plus gros :

a reste à 1

b passe de 0 à 1

c passe de 0 à 1

On ne conserve que les variables qui ne changent pas, on a donc le terme a

ALGÈBRE DE BOOLE

- Exemple avec 4 variables :

3 blocs

8 cases : d

4 cases : $\bar{b}\bar{c}$

2 cases : $\bar{a}bc$

	ab	00	01	11	10
cd					
00	1	0	0	1	
01	1	1	1	1	
11	1	1	1	1	
10	0	1	0	0	

The Karnaugh map shows minterms 1 at (00,00), (01,01), (11,01), (10,10), (00,11), and (10,11). A green L-shaped bracket covers (00,00), (01,00), and (10,00). A yellow horizontal bracket covers (01,01), (01,11), and (10,11). An orange vertical bracket covers (11,01) and (10,01).

$$f(a, b, c, d) = d + \bar{b}\bar{c} + \bar{a}bc$$

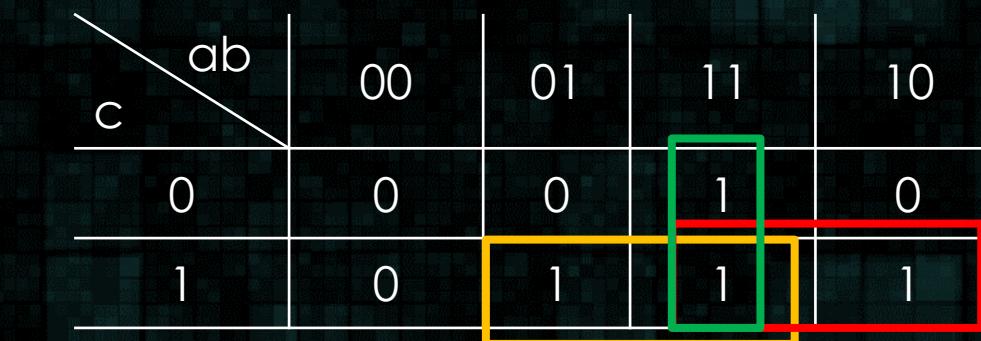
ALGÈBRE DE BOOLE

- Synthèse de construction d'un circuit logique
 - Identifier les entrées et les sorties
 - Identifier le rôle de différents éléments :
 - À quoi sert le circuit ?
 - Qu'obtient-on en sortie ?
 - Quel rôle jouent les entrées ?
 - Construire la table (les tables) de vérité.
 - Identifier chaque fonction à partir de la table de vérité
 - Simplifier chaque fonction
 - Dessiner le schéma du circuit.

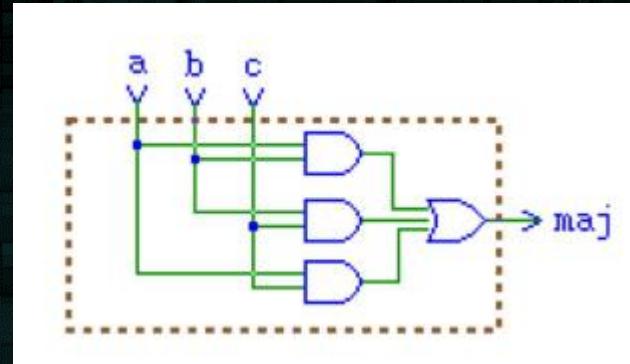
ALGÈBRE DE BOOLE

- La fonction « majorité » qui associe au triplet de booléens a, b, c la valeur 1 si il y a une majorité de 1 en entrée, et 0 sinon

a	b	c	f(a,b,c)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



$$f = ab + bc + ac$$



ALGÈBRE DE BOOLE

- Et pour la fonction majorité avec un nombre paire d'entrées ?

a	b	c	d	s ₀	s ₁
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	0	0
0	0	1	1	1	0
0	1	0	0	0	0
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	1	0	1
1	0	0	0	0	1
1	0	0	1	1	0
1	0	1	0	1	0
1	0	1	1	0	1
1	1	0	0	1	0
1	1	0	1	0	1
1	1	1	0	0	1
1	1	1	1	0	1

EXERCICES DE MISE EN PRATIQUE

- Donnez la fonction simplifiée pour chacun des tableaux suivants :

cd \ ab	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	0	1	1	0
10	0	1	1	0

cd \ ab	00	01	11	10
00	1	0	0	1
01	0	1	1	0
11	0	1	1	0
10	1	0	0	1

cd \ ab	00	01	11	10
00	1	0	0	1
01	1	1	1	1
11	1	1	0	0
10	0	0	0	0

cd \ ab	00	01	11	10
00	0	0	1	0
01	1	0	1	1
11	1	1	1	1
10	0	0	1	0

cd \ ab	00	01	11	10
00	0	1	1	0
01	1	0	0	1
11	1	0	0	1
10	0	1	1	0

cd \ ab	00	01	11	10
00	0	1	0	1
01	1	0	1	1
11	0	1	0	1
10	1	1	1	1

EXERCICES DE MISE EN PRATIQUE

- Donnez la fonction simplifiée pour chacun des tableaux suivants :

$\backslash ab$	00	01	11	10
cd	1	1	1	1
00	1	1	1	1
01	1	1	1	1
11	0	1	1	0
10	0	1	1	0

$$f = b + \bar{c}$$

$\backslash ab$	00	01	11	10
cd	0	0	1	0
00	0	0	1	0
01	1	0	1	1
11	1	1	1	1
10	0	0	1	0

$\backslash ab$	00	01	11	10
cd	1	0	0	1
00	1	0	0	1
01	0	1	1	0
11	0	1	1	0
10	1	0	0	1

$\backslash ab$	00	01	11	10
cd	1	0	0	1
00	1	0	0	1
01	1	1	1	1
11	1	1	0	0
10	0	0	0	0

$\backslash ab$	00	01	11	10
cd	0	1	1	0
00	0	1	1	0
01	1	0	0	1
11	1	0	0	1
10	0	1	1	0

$\backslash ab$	00	01	11	10
cd	0	1	0	1
00	0	1	0	1
01	1	0	1	1
11	0	1	0	1
10	1	1	1	1

EXERCICES DE MISE EN PRATIQUE

- Donnez la fonction simplifiée pour chacun des tableaux suivants :

ab \ cd	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	0	1	1	0
10	0	1	1	0

$$f = b + \bar{c}$$

ab \ cd	00	01	11	10
00	0	0	1	0
01	1	0	1	1
11	1	1	1	1
10	0	0	1	0

ab \ cd	00	01	11	10
00	1	0	0	1
01	0	1	1	0
11	0	1	1	0
10	1	0	0	1

$$f = \bar{b}\bar{d} + bd$$

ab \ cd	00	01	11	10
00	0	1	1	0
01	1	0	0	1
11	1	0	0	1
10	0	1	1	0

ab \ cd	00	01	11	10
00	1	0	0	1
01	1	1	1	1
11	1	1	0	0
10	0	0	0	0

ab \ cd	00	01	11	10
00	0	1	0	1
01	1	0	1	1
11	0	1	0	1
10	1	1	1	1

EXERCICES DE MISE EN PRATIQUE

- Donnez la fonction simplifiée pour chacun des tableaux suivants :

ab \ cd	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	0	1	1	0
10	0	1	1	0

$$f = b + \bar{c}$$

ab \ cd	00	01	11	10
00	1	0	0	1
01	0	1	1	0
11	0	1	1	0
10	1	0	0	1

$$f = \bar{b}\bar{d} + bd$$

ab \ cd	00	01	11	10
00	1	0	0	1
01	1	1	1	1
11	1	1	0	0
10	0	0	0	0

$$f = d\bar{c} + \bar{a}d + \bar{b}\bar{c}$$

ab \ cd	00	01	11	10
00	0	0	1	0
01	1	0	1	1
11	1	1	1	1
10	0	0	1	0

ab \ cd	00	01	11	10
00	0	1	1	0
01	1	0	0	1
11	1	0	0	1
10	0	1	1	0

ab \ cd	00	01	11	10
00	0	1	0	1
01	1	0	1	1
11	0	1	0	1
10	1	1	1	1

EXERCICES DE MISE EN PRATIQUE

- Donnez la fonction simplifiée pour chacun des tableaux suivants :

ab \ cd	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	0	1	1	0
10	0	1	1	0

$$f = b + \bar{c}$$

ab \ cd	00	01	11	10
00	1	0	0	1
01	0	1	1	0
11	0	1	1	0
10	1	0	0	1

$$f = \bar{b}\bar{d} + bd$$

ab \ cd	00	01	11	10
00	1	0	0	1
01	1	1	1	1
11	1	1	0	0
10	0	0	0	0

$$f = d\bar{c} + \bar{a}d + \bar{b}\bar{c}$$

ab \ cd	00	01	11	10
00	0	0	1	0
01	1	0	1	1
11	1	1	1	1
10	0	0	1	0

$$f = ab + cd + \bar{b}d$$

ab \ cd	00	01	11	10
00	0	1	1	0
01	1	0	0	1
11	1	0	0	1
10	0	1	1	0

ab \ cd	00	01	11	10
00	0	1	0	1
01	1	0	1	1
11	0	1	0	1
10	1	1	1	1

EXERCICES DE MISE EN PRATIQUE

- Donnez la fonction simplifiée pour chacun des tableaux suivants :

ab \ cd	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	0	1	1	0
10	0	1	1	0

$$f = b + \bar{c}$$

ab \ cd	00	01	11	10
00	1	0	0	1
01	0	1	1	0
11	0	1	1	0
10	1	0	0	1

$$f = \bar{b}\bar{d} + bd$$

ab \ cd	00	01	11	10
00	1	0	0	1
01	1	1	1	1
11	1	1	0	0
10	0	0	0	0

$$f = d\bar{c} + \bar{a}d + \bar{b}\bar{c}$$

ab \ cd	00	01	11	10
00	0	0	1	0
01	1	0	1	1
11	1	1	1	1
10	0	0	1	0

$$f = ab + cd + \bar{b}d$$

ab \ cd	00	01	11	10
00	0	1	1	0
01	1	0	0	1
11	1	0	0	1
10	0	1	1	0

$$f = bd + \bar{b}d$$

ab \ cd	00	01	11	10
00	0	1	0	1
01	1	0	1	1
11	0	1	0	1
10	1	1	1	1

EXERCICES DE MISE EN PRATIQUE

- Donnez la fonction simplifiée pour chacun des tableaux suivants :

ab \ cd	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	0	1	1	0
10	0	1	1	0

$$f = b + \bar{c}$$

ab \ cd	00	01	11	10
00	1	0	0	1
01	0	1	1	0
11	0	1	1	0
10	1	0	0	1

$$f = \bar{b}\bar{d} + bd$$

ab \ cd	00	01	11	10
00	1	0	0	1
01	1	1	1	1
11	1	1	0	0
10	0	0	0	0

$$f = d\bar{c} + \bar{a}d + \bar{b}\bar{c}$$

ab \ cd	00	01	11	10
00	0	0	1	0
01	1	0	1	1
11	1	1	1	1
10	0	0	1	0

$$f = ab + cd + \bar{b}d$$

ab \ cd	00	01	11	10
00	0	1	1	0
01	1	0	0	1
11	1	0	0	1
10	0	1	1	0

$$f = bd + \bar{b}d$$

ab \ cd	00	01	11	10
00	0	1	0	1
01	1	0	1	1
11	0	1	0	1
10	1	1	1	1

$$f = ab + cd + \bar{a}bc + a\bar{c}d + \bar{a}bd + \bar{b}cd$$

ALGÈBRE DE BOOLE

- Cas particulier : fonctions booléennes incomplètement définies
 - Il existe des fonctions booléennes pour lesquelles il n'y a pas de valeurs associées à certains termes produits
 - Une fonction booléenne est dite incomplète si au moins une combinaison de variables est indéterminée
 - Ceux-ci ne sont jamais "sélectionnés" et la valeur qui leur est associée peut être indifféremment 0 ou 1 en fonction de nos besoins
 - Par convention, on note « ω » ou encore plus simplement « d » (don't care)

ALGÈBRE DE BOOLE

- Exemple simple :

- On veut trier des objets selon le poids P de la manière suivante :
 - $x=0$ si $P < 400g$ et $x=1$ si $P \geq 400g$
 - $y=0$ si $P \leq 500g$ et $y=1$ si $P > 500g$
 - Les objets qui seront acceptés doivent avoir un poids $400g \leq P \leq 500g$

a	b	$f(a,b)$
0	0	0
0	1	d
1	0	1
1	1	0

$$f(a,b) = 1 \text{ si } 400g \leq P \leq 500g$$

$f(0,1)$ est non définie car $P < 400g$ et $P \geq 500g$ en même temps

a	b	0	1
0	0	0	d
1	1	1	0

$$f(a,b) = \bar{a}b + \bar{b}a = a \oplus b$$

ou

a	b	0	1
0	0	0	d
1	1	1	0

$$f(a,b) = \bar{a}b$$