UNIVERSITÉ FRANÇOIS - RABELAIS
CAMPUS BLOIS

UFR Sciences et Techniques
Licence S&T 1ère année

Informatique — UE 102

Architecture des ordinateurs et Algorithmique de base

Jean-Yves Antoine

http://www.info.univ-tours.fr/~antoine/

UNIVERSITÉ FRANÇOIS - RABELAIS
CAMPUS BLOIS

UFR Sciences et Techniques
Licence S&T 1ère année

Informatique UE 102

Chapitre I — Architecture des ordinateurs et conception de circuits logiques

Où comment avoir quelques idées sur la manière dont un ordinateur réalise des traitements puissants en ne connaissant que deux chiffres : 0 et 1 …

OBJECTIFS DU CHAPITRE

I. ARCHITECTURE MATERIELLE DES ORDINATEURS

- comprendre le rôle des différents composants d'un ordinateur
- faire le lien entre le fonctionnement du matériel et l'exécution de programmes

II. LOGIQUE BOOLEENNE ET CONCEPTION DE CIRCUITS

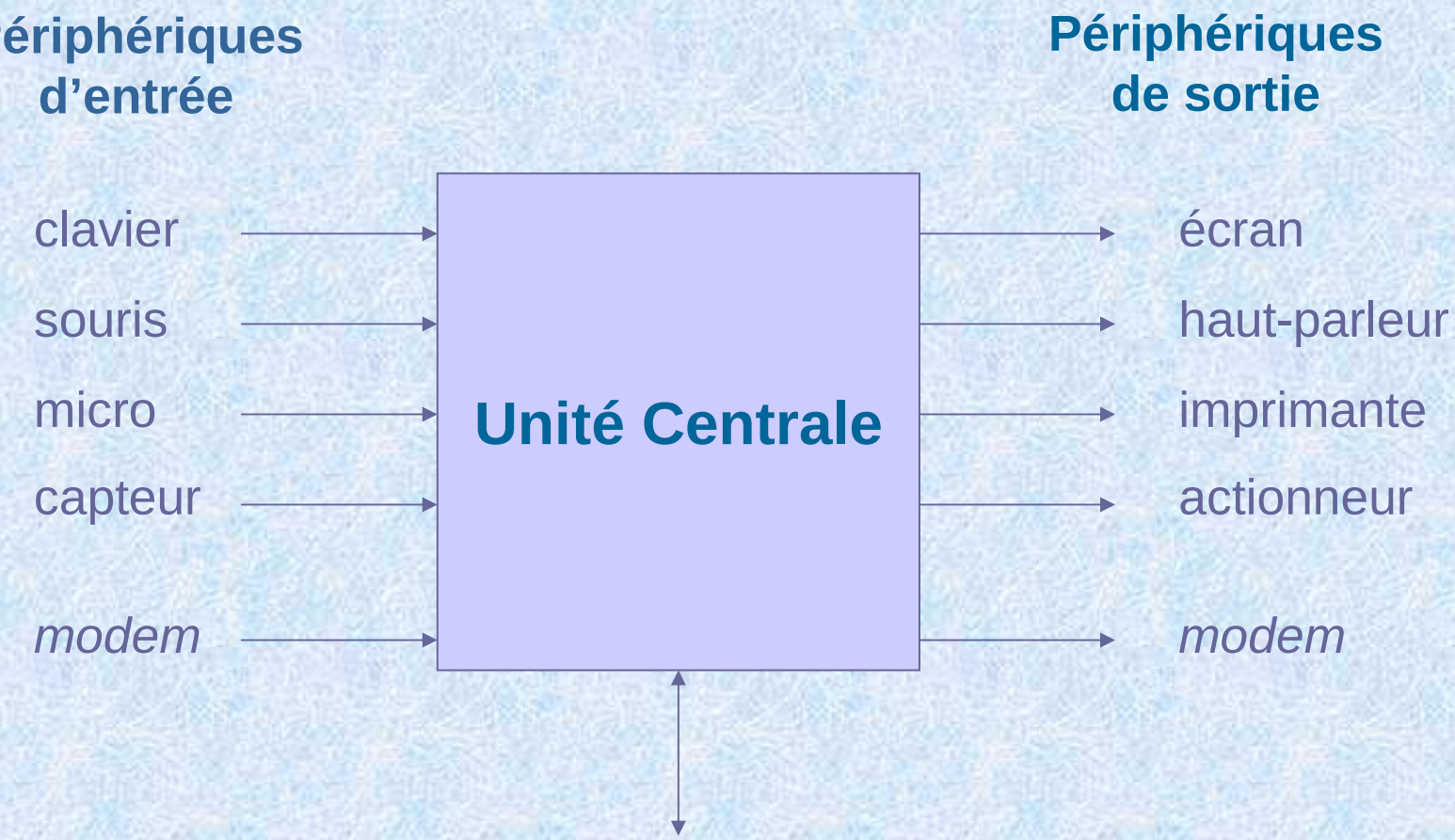
- voir le lien entre calcul logique et réalisation d'opération élémentaires par l'ordinateur
- avoir une idée générale de la conception des circuits logiques qui se trouvent (par millions) dans les puces de nos processeurs

ARCHITECTURE MATERIELLE D'UN ORDINATEUR

TÂCHES D'UN ORDINATEUR

- **traitement de l'information** (numérique, graphique, son…)
- **mémorisation de l'information** (données ou programmes)
- **communication** (Internet ou autre)

ARCHITECTURE MATERIELLE D'UN ORDINATEUR

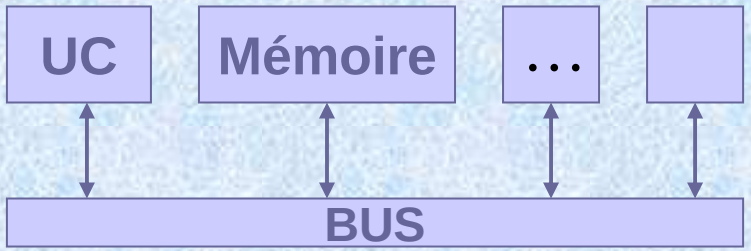


ARCHITECTURE MATERIELLE D'UN ORDINATEUR

UNITE CENTRALE (CPU)

Cerveau et chef d'orchestre du fonctionnement de la machine : c'est le grand … ordinateur de la machine !

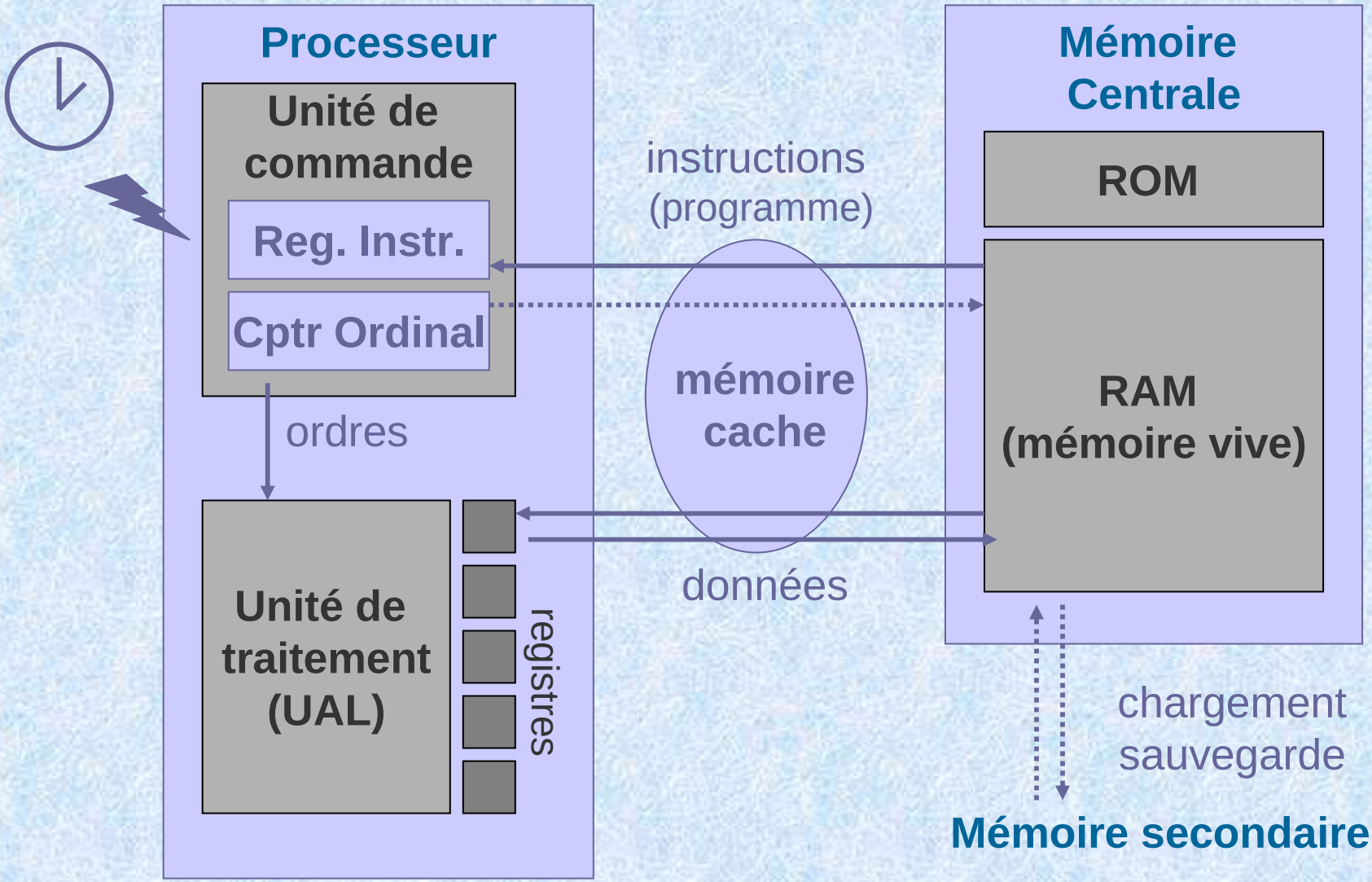
- **processeur** réalisant ou commandant les traitements (plus co-processeurs éventuels pour réaliser des opérations dédiées)
- **mémoire principale** (centrale) : espace de travail du processeur qui y mémorise temporairement les données ou instructions avec lesquelles il travaille
- **communication** interne par **bus** de données ou de contrôle



FONCTIONNEMENT SYNCHRONE

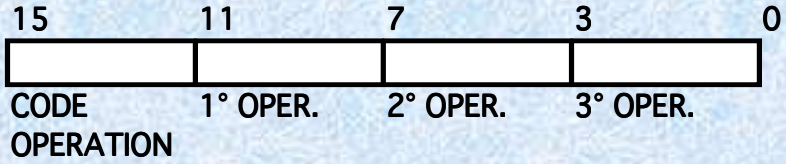
- **horloge** cadençant le travail des composants (signal de synchronisation)
- **processeur** : une instruction élémentaire à chaque coup d'horloge

ARCHITECTURE D'UN ORDINATEUR: UNITE CENTRALE



ARCHITECTURE D'UN ORDINATEUR: UNITE CENTRALE

CODAGE DES INSTRUCTIONS



Codes prédéfinis propres au processeur: langage machine (assembleur)

CYCLE D'EXECUTION D'UNE INSTRUCTION

1. **Chargement instruction** — Recherche en mémoire du code instruction dont l'adresse est dans le CO et rangement dans le registre d'instruction
2. **Incrément CO** — préparation du pas suivant (les instructions sont en première approximation stockées à la suite dans la mémoire)
3. **Décodage de l'instruction** — analyse du code instruction
4. **Préparation de l'exécution de l'instruction**
   - -calcul de l'adresse mémoire des paramètre de l'instruction
   - -chargement des paramètres de la RAM vers les registres
5. **Exécution de l'instruction** avec stockage éventuel des résultats (exécution par l'UAL si calcul, mais autres instructions possibles)

ARCHITECTURE D'UN ORDINATEUR: UNITE CENTRALE

EXEMPLE D'INSTRUCTIONS ELEMENTAIRES

- **Affectations ou recopie de valeur**

  Cela revient à copier (ou à initialiser) des valeurs stockées dans la mémoire ou un registre.

- **Instruction de calcul**

  Opérations arithmétiques ou logiques à 1, 2 ou 3 paramètres : cela revient à charger les valeurs, à faire l'opération (UAL) puis à stocker le résultat dans un registre ou la mémoire RAM.

- **Instructions de branchement** — SI … ALORS … SINON

  Permet de sauter à des parties différentes du programme suivant le contexte de traitement

  Revient uniquement à modifier la valeur du compteur ordinal

ARCHITECTURE ORDINATEUR ET PROGRAMMATION

PROGRAMMATION IMPERATIVE

- **Programmation calquée sur le fonctionnement de l'ordinateur**: suite d'instructions (de haut-niveau) à exécuter à la suite, avec instructions de branchement ⇨ **programmation efficace** en terme de vitesse de calcul
- **Programmation « séquentielle »** parfois éloignée du mode de réflexion humain ⇨ d'autres types de programmation (fonctionnelle, logique, déclarative) peuvent permettre une résolution plus rapide de problèmes … mais une exécution plus lente du programme.
- **Langage impératifs** : Pascal, C, Fortran, Cobol, « noyau » de langages objets tels que C++ ou Java ….

MEMOIRE ET PROGRAMMATION

- Déclaration et instanciation des **variables**

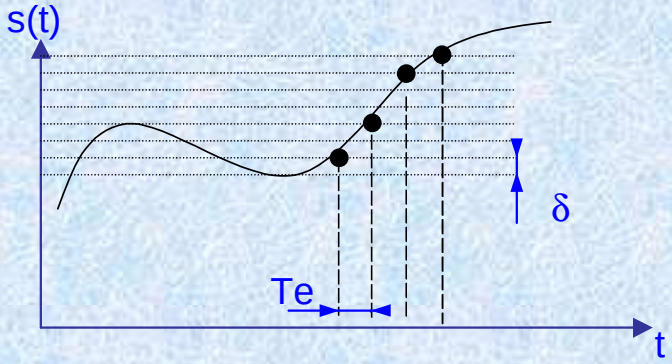
CODAGE BINAIRE DES DONNEES

TYPES DE DONNEES MANIPULEES PAR UN ORDINATEUR

- **Numérique** : Entier (`Integer`) ou Réel (`Float`)

  Données numériques brutes, images (pixels), audio, vidéo, etc…

  Codage **binaire** (0,1)



- **Textuel** : chaîne de caractères

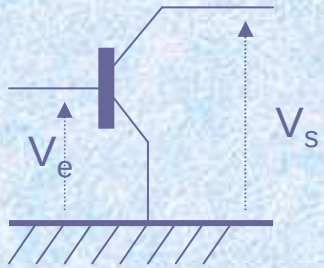
  Entrées / Sortie (clavier, imprimante), documents textuels…

  Codage **numérique** de chaque caractère : **ASCII**, **UniCode.**

CODAGE BINAIRE DES DONNEES

POURQUOI UNE REPRESENTATION BINAIRE ?

Circuits électronique : tension supérieure (1) ou inférieure (0) à une certaine tension dans les transistors.



NUMERATION DANS UNE BASE B

Tout entier s'écrit comme la somme de puissances de la base B :

$N = \sum a_i B^i$ avec $a_i \in \{0, B-1\}$ $\quad <N>_B$ représenté par $<a_N, a_{N-1},\ldots,a_1,a_0>$

Décimal $<1528>_{10} = 1*10^3 + 5*10^2 + 2*10^1 + 8*10^0$

Binaire $<1011>_2 = 1*2^3 + 0*2^2 + 1*2^1 + 1*2^0 = <11>_{10}$

Hexadécimal base 16 : {0, 1,…, 9, A, … F} $\quad <1F>_{16} = <31>_{10}$

CODAGE BINAIRE DES DONNEES

UN PEU DE VOCABULAIRE…

Bit (*Binary digIT*)	0 ou 1
Octet (*Byte*)	bloc de 8 bits : nombres de $<0>_{10}$ à $<255>_{10}$
Mots (*Words*)	mot de 16, 32, 64 bits

CONVERSION BINAIRE - DECIMAL

Binaire → Décimal	sommes des puissances de deux
Décimal → Binaire	recherche des puissances de deux divisions successives par deux

CALCULS ARITHEMATIQUES

- **Addition** et **soustraction** identiques dans n'importe quelle base !
- **Multiplication** et **division** : additions / soustractions successives

CODAGE BINAIRE DES DONNEES

ENTIERS RELATIFS : COMPLEMENT A DEUX

Idée bit de signe (0: positif; 1: négatif) + autres bits pour la valeur absolue

octet en relatif : nombres de $<-127>_{10}$ à $<127>_{10}$

Problèmes deux représentation du zéro (+0 ou –0)

principes d'addition et soustraction non valables

Complément à deux

- Nombres binaires positifs inchangés
- Nombres négatifs — on inverse tous les bits de la valeur absolue (complément à 1) et on ajoute 1

Vérification sur quelques exemples

- Une seule représentation pour 0
- Addition et soustraction conservées

CODAGE BINAIRE DES DONNEES

REELS : VIRGULE FLOTTANTE

Idée approximation $R = s.m.B^e$ avec s bit de signe, m mantisse et e exposant dans la base B

Exemple -10,745 s = 1, m = 10745 et e = -3 en base 10

Flottant la virgule décimale flotte suivant la valeur de l'exposant

Norme IEEE 754 / IEC 60559

	Signe	**Mantisse**	**Exposant**
Simple précision (*float*) : 32 bits	1 bit	8 bits	23 bits
Double précision (*double*) : 64 bits	1 bit	11 bits	52 bits

Limitations erreurs d'arrondis

CODAGE BINAIRE DES DONNEES

TEXTE : CODE ASCII (American Standard Code for International Interchange)

Idée codage sur un octet de chaque caractère (7 bits ASCII initial)

Exemples caractères chiffres : 0 = $<30>_{16}$ jusque 9 = $<39>_{16}$

lettres majuscules : A= $<41>_{16}$<jusque Z = $<5A>_{16}$

lettres minuscules : a= $<61>_{16}$<jusque z = $<7A>_{16}$

retour à la ligne : $<0A>_{16}$; retour à la ligne = $<0D>_{16}$

Limitation un octet = 256 caractères seulement

Unicode norme internationale de représentation des caractères sur 2 octets (permet de représenter tous les caractères de tous les alphabets principaux + places libres personnalisables)

Les premières valeurs d'Unicode reprennent le code ASCII

Toile : http://www.unicode.org

CODAGE BINAIRE DES DONNEES

ASCII-ANSI : code étendu (8 bits)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	□	□	‚	ƒ	„	…	†	‡	ˆ	‰	Š	‹	Œ	□	□	□
9	□	‘	’	“	”	•	–	—	˜	™	š	›	œ	□	□	Ÿ
A		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
B	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
C	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

UNICODE (16bits): copte

CODAGE BINAIRE DES DONNEES

UNICODE : mandarin (20 988 idéogrammes)

	510	511	512	513	514	515	516	517	518	519	51A	51B	51C	51D	51E	51F
0	儀 5100	儐 5110	儼 5120	儻 5130	兀 5140	児 5150	兠 5160	兰 5170	冀 5180	冒 5190	冠 51A0	冰 51B0	净 51C0	凐 51D0	几 51E0	凰 51F0
1	儁 5101	儇 5111	儘 5121	儱 5131	允 5141	兑 5151	兢 5161	共 5171	冁 5181	冑 5191	冢 51A1	冴 51B1	凍 51C1	凑 51D1	凡 51E1	凱 51F1
2	儂 5102	儒 5112	儢 5122	儲 5132	先 5142	兒 5152	兢 5162	天 5172	冂 5182	冒 5192	冢 51A2	冲 51B2	凌 51C2	凿 51D2	凡 51E2	凧 51F2
3	儃 5103	儓 5113	儻 5123	儳 5133	元 5143	兓 5153	兦 5163	关 5173	冃 5183	冓 5193	冣 51A3	决 51B3	涂 51C3	凍 51D3	凡 51E3	凳 51F3
4	億 5104	儔 5114	儴 5124	儴 5134	兄 5144	兔 5154	兣 5164	兴 5174	冉 5184	冔 5194	冤 51A4	冴 51B4	凄 51C4	凔 51D4	凤 51E4	凴 51F4
5	儅 5105	儕 5115	儥 5125	儵 5135	充 5145	兕 5155	入 5165	兵 5175	内 5185	冕 5195	冥 51A5	况 51B5	凋 51C5	凘 51D5	尻 51E5	凵 51F5
6	儆 5106	儉 5116	儷 5126	儺 5136	兆 5146	兖 5156	兦 5166	其 5176	円 5186	冖 5196	冦 51A6	冶 51B6	准 51C6	準 51D6	处 51E6	凶 51F6
7	儇 5107	儗 5117	儹 5127	儷 5137	兇 5147	兗 5157	内 5167	具 5177	冇 5187	冗 5197	冧 51A7	冷 51B7	凇 51C7	凞 51D7	凧 51E7	凸 51F7

CALCUL BOOLEEN

ALGEBRE DE BOOLE : DEFINITION

- construite sur l'ensemble $\mathcal{B} = \{0,1\}$
- lois de compositions internes addition **+**
  multiplication **.**
- loi de complémentation $0 = 1$ et $\overline{1} = 0$

Sir Boole
(1815-1864)

a	b	a + b	a . b
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

a	$\overline{a}$
0	1
1	0

CALCUL BOOLEEN

- composition des opérations
- règles de priorité : complémentation ⇨multiplication ⇨addition

CALCUL BOOLEEN

ALGEBRE DE BOOLE : PROPRIETES

Éléments neutres	$a + 0 = a$	$a \cdot 1 = a$
Commutativité	$a + b = b + a$	$a \cdot b = b \cdot a$
Associativité	$(a + b) + c = a + (b + c)$ $(a \cdot b) \cdot c = a \cdot (b \cdot c)$	
Distributivité	$a + (b \cdot c) = (a + b) \cdot (a + b)$ $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$	
Complémentation	$a \cdot \overline{a} = \mathbf{0}$	$a + \overline{a} = \mathbf{1}$
Idempotence	$a + a = a$	$a \cdot a = a$
Absorption	$a + (a.b) = a$	$a \cdot (a + b) = a$
Th. De De Morgan	$\overline{a+b} = \overline{a} \cdot \overline{b}$	$\overline{a.b} = \overline{a} + \overline{b}$
Th. d'involution	$\overline{\overline{a}} = a$	

CALCUL BOOLEEN ET LOGIQUE

INTERPRETATION LOGIQUE DE L'ALGEBRE DE BOOLE

Valeurs de vérité 0 ≡ *Faux* 1 ≡ *Vrai*

Connecteurs logiques

addition	*OU logique*	(noté ∨)
multiplication	*ET logique*	(noté ∧)
complémentation	*négation*	(notée ¬)

Tables de vérité

P	Q	P ∨ Q	P ∧ Q	P ⇔ Q	P ⇒ Q
F	F	F	F	V	V
F	V	V	F	F	V
V	F	V	F	F	F
V	V	V	V	V	V

P	¬P
V	F
F	V

⇨ **LP : Logique des propositions** (cf. cours de logique en L3)

CALCUL BOOLEEN ET LOGIQUE

FORMULES D'EQUIVALENCE DE LA LP

Double négation	$\neg\neg P \equiv P$
Lois de Morgan	$\neg(P \wedge Q) \equiv \neg P \vee \neg Q$ $\neg(P \vee Q) \equiv \neg P \wedge \neg Q$
Idempotence	$P \vee P \equiv P \wedge P \equiv P$
Commutativité	$P \wedge Q \equiv Q \wedge P$ $\quad$ $P \vee Q \equiv Q \vee P$
Associativité	$(P \wedge Q) \wedge R \equiv P \wedge (Q \wedge R)$ $(P \vee Q) \vee R \equiv P \vee (Q \vee R)$
Contradiction	$P \wedge \neg P \equiv$ **Faux**
Tiers-exclus	$P \vee \neg P \equiv$ **Vrai**
Distributivité	$P \vee (Q \wedge R) \equiv (P \vee Q) \wedge (P \vee R)$ $P \wedge (Q \vee R) \equiv (P \wedge Q) \vee (P \wedge R)$
Absorption	$P \vee (P \wedge Q) \equiv P$ $\quad$ $P \wedge (P \vee Q) \equiv P$

⇨ **Logique des propositions** (cf. cours de logique en L3)

FONCTIONS BOOLEENNES

DEFINITION

On appelle **fonction booléenne** toute application de $\mathcal{B}^n \rightarrow \mathcal{B}$ dont la formule s'écrit uniquement au moyen des opérations élémentaires de l'algèbre de Boole.

Exemple : f(a,b,c) = a + b.c

UTILITE : CONCEPTION DE CIRCUITS

Tout circuit d'un processeur (quelle que soit l'opération à réaliser) peut s'exprimer sous la forme d'une fonction booléenne.

Problème classique

Trouver l'expression d'une fonction booléenne à partir de la définition de la fonction à réaliser et donner le circuit logique correspondant.

- mintermes / maxtermes
- tableaux de Karnaugh

FONCTIONS BOOLEENNES : FORME NORMALE

MINTERME / MAXTERME

Minterme Tout produit des variables de la fonction booléenne ou de leur complément telle que la fonction est égale à 1 quand le produit est égale à 1.

a	b	f(a,b)	
1	1	1	minterme : a . b
1	0	1	minterme : a . $\overline{b}$
0	1	0	maxterme : a + $\overline{b}$
0	0	1	minterme : $\overline{a}$. $\overline{b}$

Maxterme Toute somme des variables de la fonction booléenne ou de leur complément telle que la fonction est égale à 0 quand le maxterme est nul

EXPRESSION SOUS FORME NORMALE

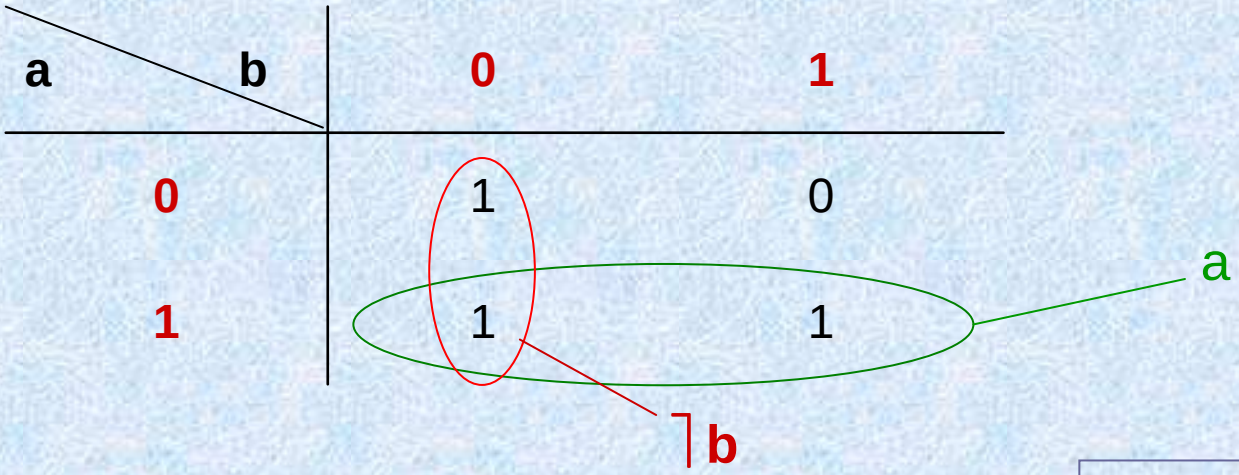
f.n.disjonctive	somme des mintermes de la fonction
f.n. conjonctive	produit des maxtermes de la fonction

Exemple **fnd** f(a,b) =

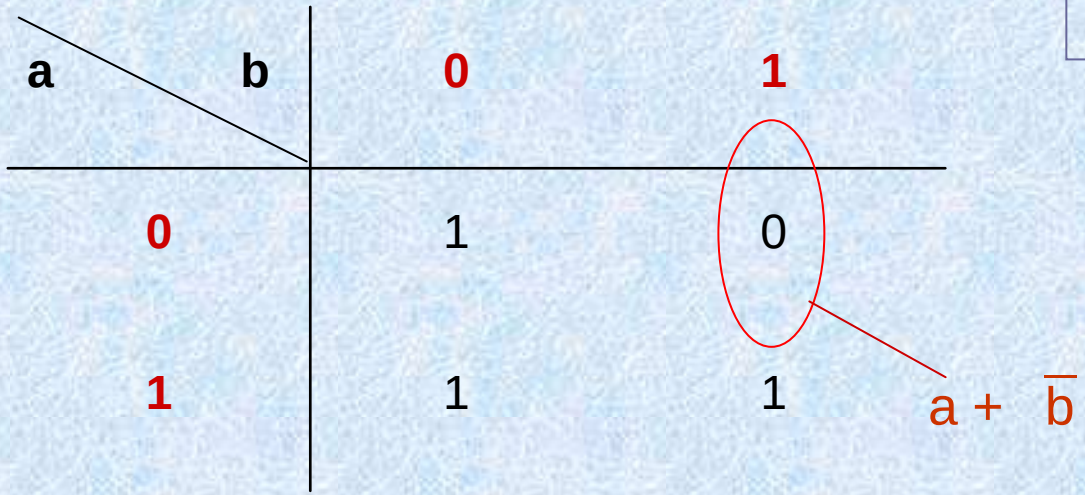
fnc f(a,b) =

SIMPLIFICATION DES FONCTIONS BOOLEENNES

TABLEAU DE KARNAUGH À DEUX DIMENSIONS



fnc ≡ fnd ≡ $a + \overline{b}$



SIMPLIFICATION DES FONCTIONS BOOLEENNES

TABLEAU DE KARNAUGH À TROIS DIMENSIONS

$f \equiv (\ \overline{b}.\ \overline{c}\ ) + a$

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

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

$f \equiv (a+\ \overline{b})\ .\ (a+\ \overline{c})$

CONCEPTION DE CIRCUITS

PORTES NAND ET PORTES NOR

Principe Il est possible d'exprimer toutes les fonctions booléennes à l'aide de l'une des deux fonctions NAND (NON-ET) ou NOR (NON-OU).

$\text{NAND}(a,b) = \overline{a.b}$ et $\text{NAND}(a) = \overline{a}$

$\text{NOR}(a,b) = \overline{a + b}$ et $\text{NOR}(a) = \overline{a}$

Intérêt Une seule porte logique à réaliser pour tout le circuit.

Réalisation d'un circuit suivant une logique NAND ou NOR

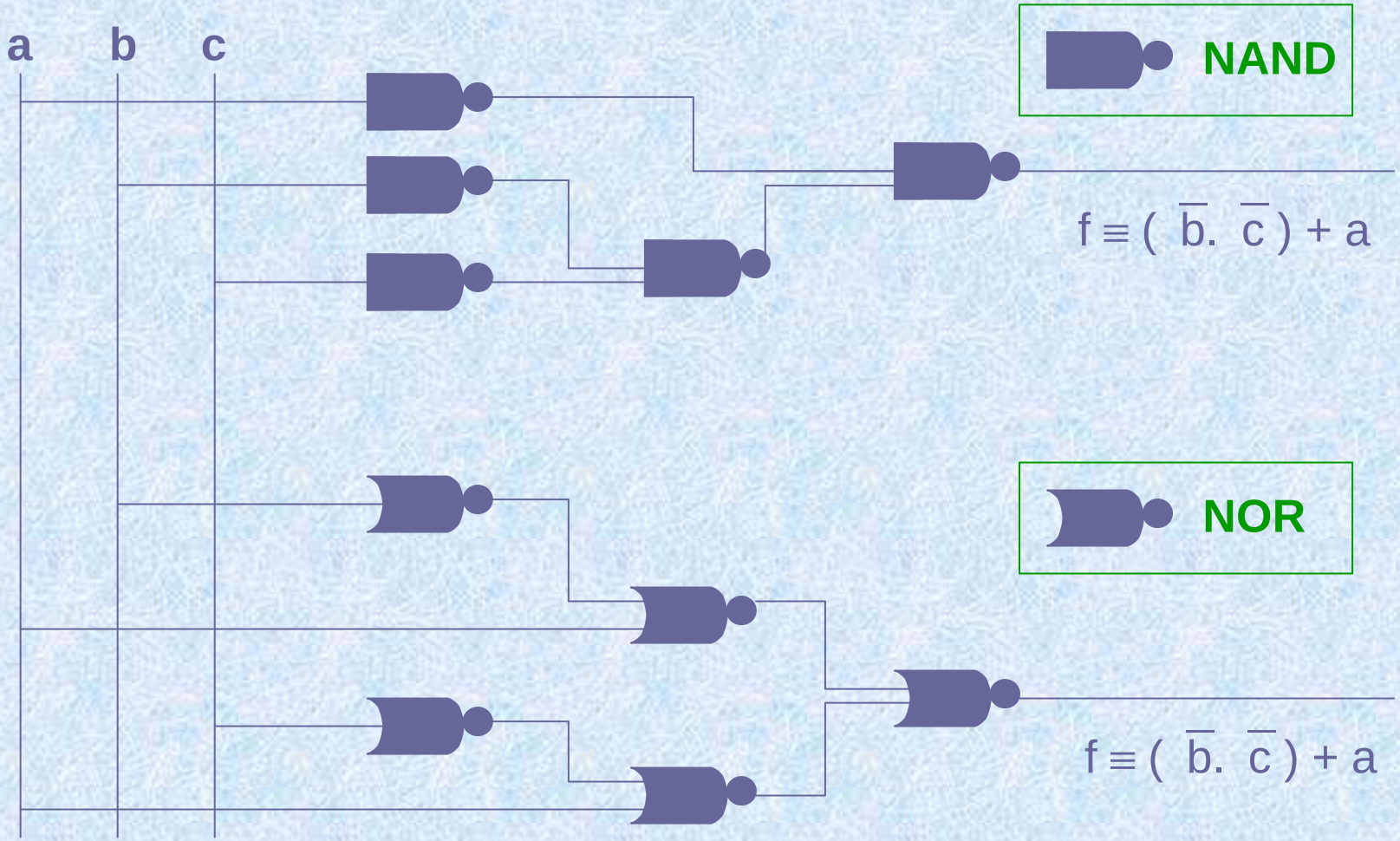
1) Partir de l'expression sous forme normale (disjonctive pour NAND, conjonctive pour NOR) simplifiée par la méthode des tableaux de Karnaugh

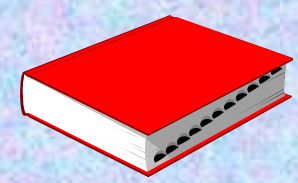
2) Tout exprimer sous forme de NAND (resp. NOR) par complémentation successive des sommes (resp. produits) en appliquant les théorèmes de De Morgan et d'involution.

Exemple : $a + b = \overline{a} + \overline{b}$

CONCEPTION DE CIRCUITS

EXEMPLE





Bibliographie

Ouvrages généraux

Tannenbaum…

Cours sur la Toile

Supports du cours : www.sir.blois.univ-tours.fr/~antoine/enseignement/pascal