LU2IN014 : Machine et Représentation

Cours 1 : Algèbre de Boole, Fonctions Booléennes et Représentation des Entiers Naturels

quentin.meunier@lip6.fr

1. Introduction
2. Architecture générale d'un ordinateur
3. Logique booléenne et algèbre de Boole
4. Fonctions Booléennes
5. Représentation des entiers naturels

1 Introduction

2 Architecture générale d'un ordinateur

3 Logique booléenne et algèbre de Boole

4 Fonctions Booléennes

5 Représentation des entiers naturels

Objectif de l'UE Machine et Représentation

- Code LU2IN014 : MAREP ou Machine et représentation
- Objectif : demystifier l'ordinateur via la connaissance de :
  - Représentation des données et des programmes
  - Structure et écriture de programme en langage assembleur
  - Exécution d'un programme binaire

Organisation

- 5 cours, 5 TD et 5 TME
- Debut des TD/TME la semaine 5 (1er Février 2021)

- Note de l'UE ?

Support de cours

- Transparents mis en ligne, une grande partie du cours est faite au tableau
- Enoncés de TD/TME distribués en TD et mis en ligne sur la page de l'UE
- Page de l'UE :
  `https://www-licence.ufr-info-p6.jussieu.fr/lmd/licence/2020/ue/LU2IN014-2021fev/`

1 Introduction

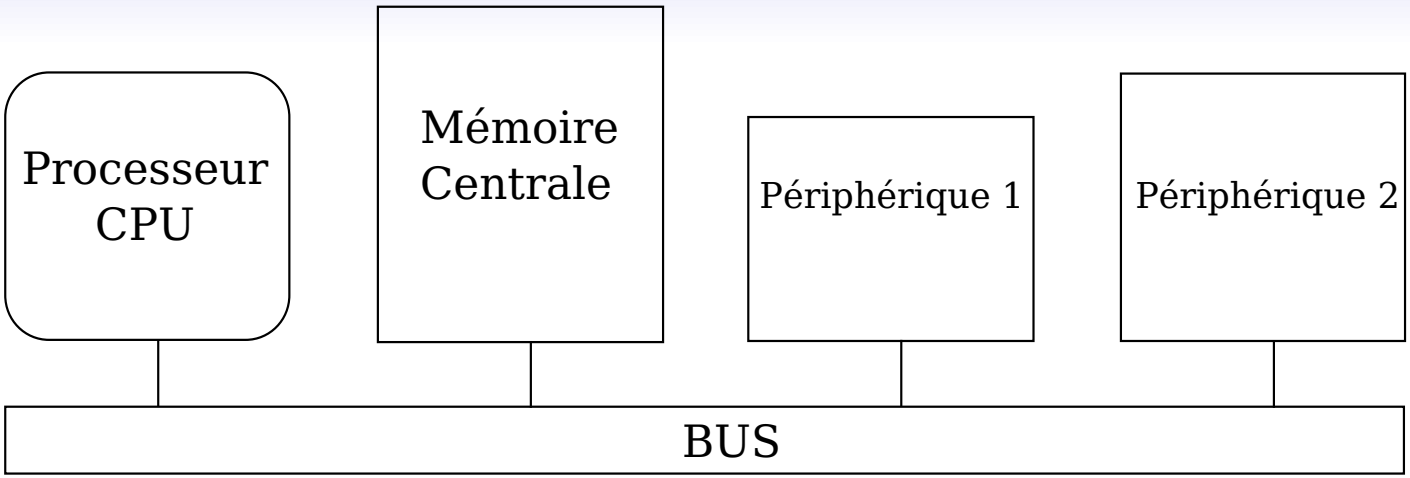
2 Architecture générale d'un ordinateur

3 Logique booléenne et algèbre de Boole

4 Fonctions Booléennes

5 Représentation des entiers naturels

Architecture générale d'un ordinateur



Un ordinateur, dont l'architecture générale est représentée ci-dessus, comprend :

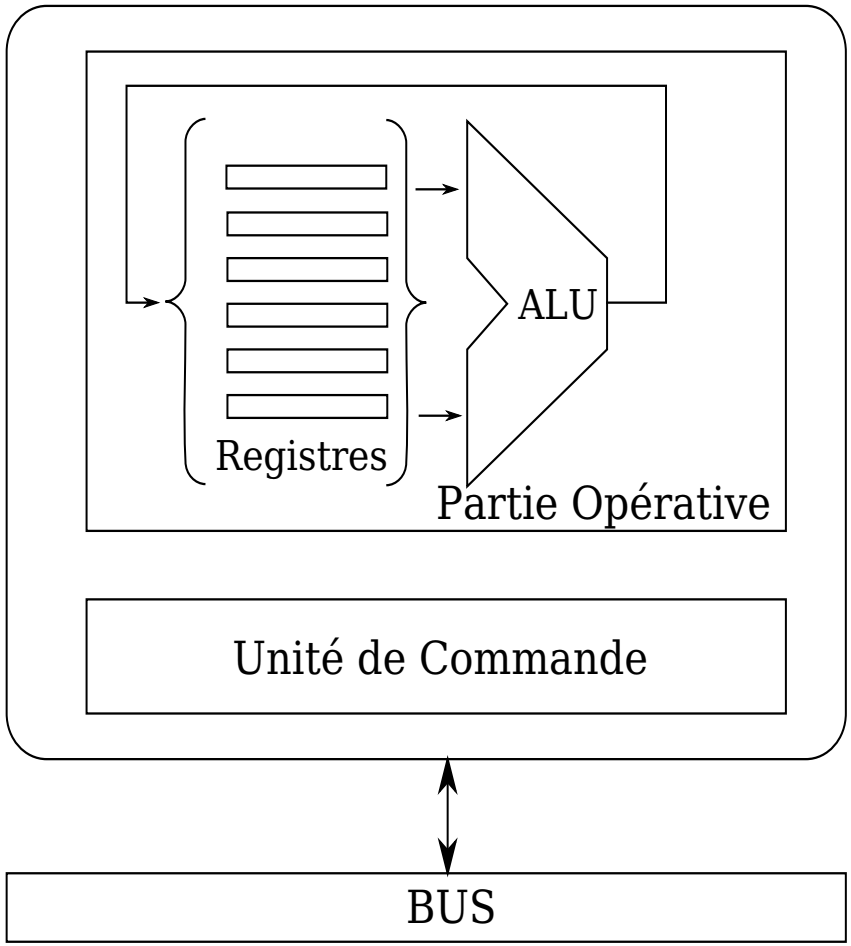
- un processeur (ou CPU)
- une mémoire centrale (ou RAM ou mémoire vive)
- un bus
- des périphériques

Architecture générale d'un ordinateur

- Le **processeur** (ou *CPU*) est l'unité de traitement de l'information (instructions et données). Il exécute des programmes (suite d'instructions qui définissent un traitement à appliquer à des données).
- La **mémoire centrale** (ou *RAM* ou *mémoire vive* ou simplement *mémoire*) est une unité de stockage temporaire des informations nécessaires à l'exécution d'un programme. Externe au processeur, elle stocke en particulier les intructions du programme en cours d'exécution ou à exécuter et les données du programme (nombres, caractères alphanumériques, adresses mémoire, ...).
- Le **bus** est le support physique des transferts d'information entre les différentes unités.
- Les **périphériques** sont des unités connexes permettant de communiquer avec l'ensemble processeur-mémoire : clavier, écran, disque dur, réseau, imprimante/scanner, ...

Architecture d'un processeur séquentiel

- **Unité de commande** (ou *Partie commande*) : analyse les instructions et séquence les actions élémentaires pour leur réalisation
- **Partie opérative** : au service de l'unité de commande, avec outils pour réaliser les actions élémentaires ordonnées par l'unité de commande. Comprend notamment une **unité arithmétique et logique** (*ALU*) et des **registres** internes.



Réalisation physique des éléments d'un ordinateur

- La réalisation physique de ces éléments est électronique (mécanique pour certains périphériques).
- Les grandeurs manipulées au sein des composants sont des tensions électriques qui sont stockées dans des éléments mémorisants ou qui transitent sur des fils et traversent des portes combinant les tensions et réalisant des fonctions logiques.
- On distingue deux niveaux de tension (0V et 1.5V) qui représentent deux valeurs distinctes nommées 0 et 1.
- C'est pourquoi toutes les informations manipulées sont représentées par des mots composés de 0 et de 1.

Des 0 et des 1... La représentation binaire

Toutes les informations manipulées sont représentées par des mots composés de 0 et de 1.

- **mot binaire** = un mot formé sur l'alphabet $\{0,1\}$
- **bit** (**b** inary dig**it**) = 0 ou 1
- **octet** = mot binaire composé de 8 bits
- **quartet** = mot binaire composé de 4 bits
- **mot MIPS** (dans la suite *mot*) = mot de 32 bits = un mot de 4 octets

Les traitements réalisés par l'ordinateur sont faits sur la représentation binaire des informations, en calculant des fonctions logiques.

Notion de registre

- Un **registre**, c'est un composant matériel interne au processeur utilisé pour faire des calculs ; en particulier :
  - Sa taille est typiquement de 1 mot ; en mips, tous les registres font 32 bits (= 4 octets)
  - Il est accessible très rapidement en lecture et en écriture
  - On peut effectuer des opérations sur les valeurs qu'il contient et y mettre le résultat des opérations

Attention

- On ne peut pas effectuer des opérations directement sur la mémoire

Fonctionnement d'un programme

D'un point de vue schématique, le fonctionnement d'un programme est le suivant :

- Lecture de données (1 ou 2) en mémoire vers les registres du processeur
  - Lit la mémoire, écrit les registres (chargement ou *load*)
- Opération sur les registres
  - Lit et écrit les registres, utilise l'ALU
- Écriture du résultat en mémoire
  - Lit les registres, écrit en mémoire (écriture ou *store*)

1 Introduction

2 Architecture générale d'un ordinateur

3 Logique booléenne et algèbre de Boole
- Définition
- Opérations logiques
- Algèbre de Boole et règles

4 Fonctions Booléennes

5 Représentation des entiers naturels

Logique booléenne

- Logique booléenne = formalisation des raisonnements basés sur des éléments qui peuvent être soit vrais, soit faux.
- Soit l'alphabet B= {FAUX, VRAI} = {F,V} = {0,1}.
- Ordre sur les éléments de B : $0 < 1$
- **variable booléenne** = une variable pouvant contenir soit vrai, soit faux.
- **fonction booléenne** = une fonction de $B^n \rightarrow B$
- **table de vérité** = énumération ligne à ligne des valeurs prises par une fonction f en fonction de la valeur de ses paramètres

Opérations logiques

- Le complément / NON / NOT et noté par le surlignage est une fonction unaire. Si $a = 0$ alors $\overline{a} = 1$ et si $a = 1$ alors $\overline{a} = 0$
- L'addition / OU / OR et notée $+$ est une fonction binaire et définie par $a + b = max(a, b)$
- La multiplication / ET / AND et notée . est une fonction binaire et définie par $a.b = min(a, b)$.

Algèbre de Boole (1)

- $< B, 0, 1, +, ., - >$ forme une algèbre de Boole car il respecte les axiomes suivant :
- L'addition et la multiplication sont associatives :
  - $\forall x, y, z \in B^3 \ (x + y) + z = x + (y + z)$
  - $\forall x, y, z \in B^3 \ (x.y).z = x.(y.z)$
- L'addition et la multiplication sont commutatives :
  - $\forall x, y \in B^2 \ x + y = y + x$
  - $\forall x, y \in B^2 \ x.y = y.x$
- L'addition et la multiplication sont distributives l'une par rapport à l'autre (ce qui est différent de l'algèbre sur les nombres !) :
  - $\forall x, y, z \in B^3 \ x.(y + z) = x.y + x.z$
  - $\forall x, y, z \in B^3 \ x + (y.z) = (x + y).(x + z)$

Algèbre de Boole (2)

- $< B, 0, 1, +, ., - >$ forme une algèbre de Boole car il respecte les axiomes suivant :
- 0 est l'élément neutre pour l'addition et 1 pour la multiplication :
  - $\forall x \in B \; x + 0 = x$
  - $\forall x \in B \; x.1 = x$
- La somme d'un élément et de son complément est 1 :
  - $\forall x \in B \; x + \overline{x} = 1$
- Le produit d'un élément et de son complément est 0 :
  - $\forall x \in B \; x.\overline{x} = 0$

Autres propriétés

- Théorème de dualité :
  - si $< B, 0, 1, +, ., - >$ est une algèbre de Boole alors $< B, 1, 0, ., +, - >$ en est une également
- Loi de De Morgan :
  - $\overline{x + y} = \overline{x}.\overline{y}$
  - $\overline{x.y} = \overline{x} + \overline{y}$
- Règles de simplification
  - loi d'involution : $\forall x \in B$, $\overline{\overline{x}} = x$
  - éléments absorbants : $\forall x \in B$, $x.0 = 0$ et $x + 1 = 1$
  - idempotence : $\forall x \in B$, $x.x = x$ et $x + x = x$

1 Introduction

2 Architecture générale d'un ordinateur

3 Logique booléenne et algèbre de Boole

4 Fonctions Booléennes
- Expression algébrique d'une fonction booléenne
- Représentation schématique des fonctions booléennes
  - Porte logique
  - Circuit logique
- Exemples de circuits logiques

5 Représentation des entiers naturels

Expression algébrique d'une fonction booléenne

- Toute fonction booléenne f peut s'exprimer à partir des constantes 0 et 1, des noms des variables booléennes paramètres de f et des opérations $+$, . et - de l'algèbre de Boole.
- On peut construire une expression algébrique d'une fonction f à partir de sa table de vérité :
  - (1) **Forme normale disjonctive** : construite en réalisant la disjonction (OU) des termes représentant les lignes où f vaut 1. Chaque terme est le produit (ET) des noms de variables de f, complémentés si la contribution de la variable est 0.
  - (2) **Forme normale conjonctive** : produit des termes représentant les lignes où f vaut 0. Chaque terme est la sommes des noms de variable, complémentés si la contribution est 1.
- NB : on peut retrouver la forme normale conjonctive de f à partir de la forme normale disjonctive de $\overline{f}$ en utilisant la loi d'involution $\overline{\overline{f}} = f$.

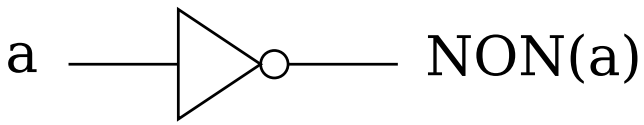
Equivalence de deux fonctions booléennes ou expressions algébriques

- Deux fonctions booléennes sont équivalentes si elles ont la même table de vérité.
- Deux expressions algébriques booléennes sont équivalentes si elles peuvent se réécrire en une même troisième expression.

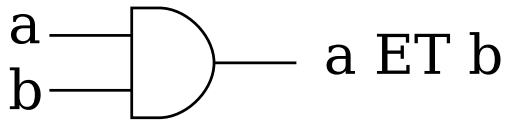
Représentation schématique des fonctions booléennes

- Un circuit est une représentation schématique de l'évaluation d'une fonction booléenne à partir de l'une de ses expressions algébriques.
- Une **porte logique** représentant une fonction booléenne f à n variables est un élément possédant n signaux d'entrée, un signal de sortie et produisant sur le signal de sortie la valeur de f pour la configuration fournie sur les signaux d'entrée.
- Illustration de la représentation schématique des portes ET, OU et NON.

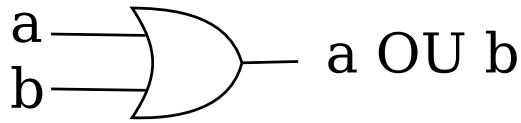
Porte NON



Porte ET



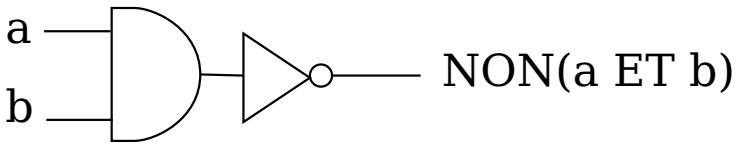
Porte OU



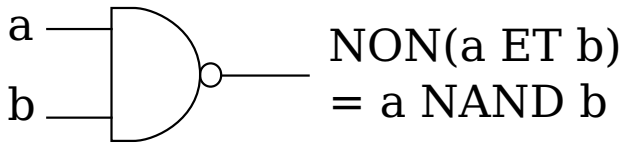
Circuit logique

- Un **circuit logique** est un diagramme orienté composé de signaux d'entrée, de portes logiques, de signaux de sortie et respectant les règles de connectique suivantes :
  - les entrées des portes logiques sont connectées aux signaux d'entrée du circuit ou aux sorties d'autres portes du circuit,
  - les signaux de sortie sont connectés aux sorties de portes du circuit,
  - la composition de fonction $g \circ f$ est représentée par la mise en séquence de f et de g : les signaux de sortie de f sont connectés aux signaux d'entrée de g
- Exemple : $\overline{a.b}$

Réalisation de NON(a ET b)



Porte NAND

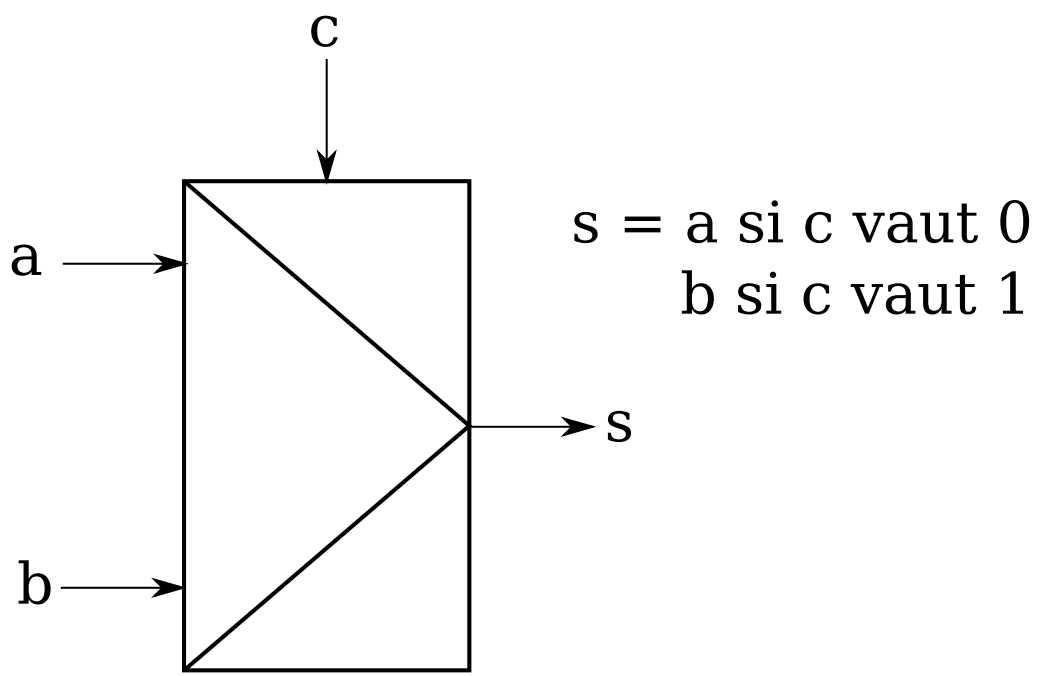


Circuit combinatoire

- **Circuit combinatoire** : c'est un circuit logique dont le graphe orienté est sans cycle. La valeur du signal de sortie ne dépend que des valeurs des signaux d'entrée à l'instant présent (dès qu'une entrée change, la sortie change – modulo le temps de propagation de l'information mais c'est très rapide !).
- **Logisim** : logiciel d'édition et simulation de schémas logiques utilisé en TME. Il permet de représenter une partie simplifiée de l'unité arithmétique et logique d'un processeur.

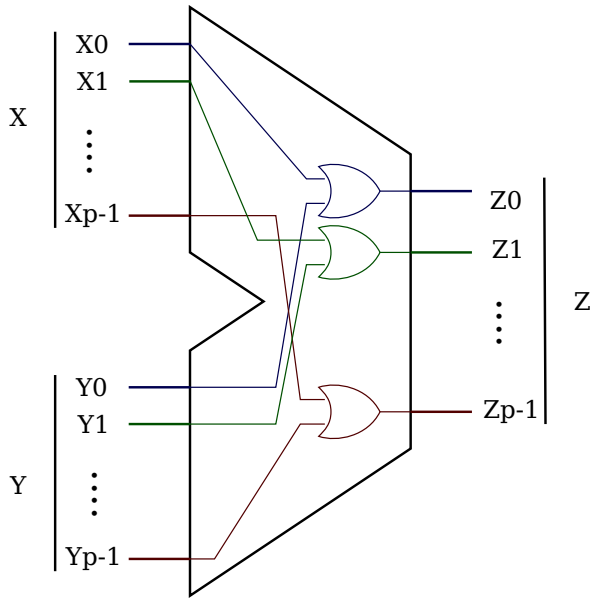
Multiplexeur

- Un **multiplexeur** à deux entrées a et b et une commande c est un circuit :
  - sortie = la valeur de l'entrée a si c vaut 0
  - sortie = la valeur de l'entrée b si c vaut 1
- Un multiplexeur permet de sélectionner une des entrées en fonction de la valeur de la commande, indépendamment de la valeur des entrées



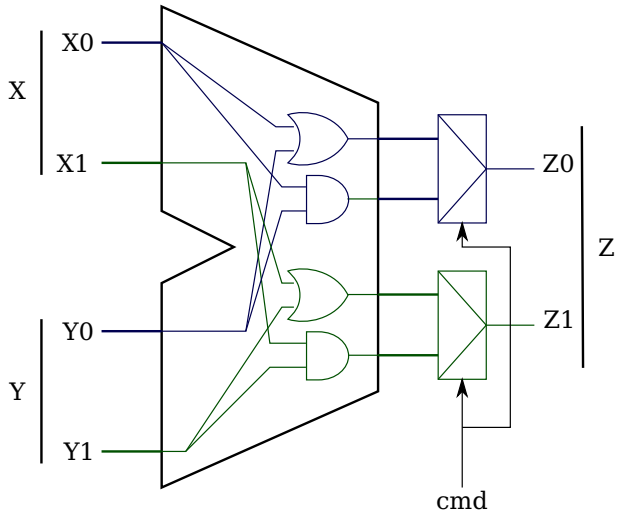
Unité arithmétique et logique

- Existence d'instructions des processeurs réalisant des opérations logiques sur des mots de p bits
- Si f est une fonction booléenne binaire alors sur deux mots de p bits :
  $f(a_{p-1}...a_1a_0, b_{p-1}...b_1b_0) = f(a_{p-1}, b_{p-1})...f(a_1, b_1)f(a_0, b_0)$
  On parle d'opération "bit-à-bit".
- Exemple en MIPS opération logique OU via l'instruction *or $3, $2, $1*



Unité arithmétique et logique

- Existence d'instructions des processeurs pour réaliser des opérations logiques sur des mots de p bits via l'ALU
- Plusieurs opérations logiques précablées : OU, ET, XOR existent et sont précablées
- Instructions MIPS *or $3, $2, $1*, *and $3, $2, $1*,..
- Sélection de l'opération, des opérandes sources et destination via l'unité de commande



Sélection de l'opération OU ou ET en fonction de l'instruction exécutée

La valeur de cmd est déterminée par l'instruction en cours d'exécution :
- Si "or $1, $2, $3" sélection du OU (cmd = 0)
- Si "and $1, $2, $3" sélection du ET (cmd = 1)

1 Introduction

2 Architecture générale d'un ordinateur

3 Logique booléenne et algèbre de Boole

4 Fonctions Booléennes

5 **Représentation des entiers naturels**
- Introduction
- Représentation machine des entiers naturels
  - Taille bornée de la représentation
  - Extension de la représentation
- Changement de base
  - Algorithme de conversion par divisions successives
  - Conversion de la base 2 à la base 16
  - Conversion de la base 16 à la base 2

Représentation des informations en machine

- La représentation binaire est facile à réaliser (2 états d'équilibre) et les opérations fondamentales sont relativement simples à effectuer sous forme de circuit logique
- Différents types d'informations (instructions, données) dans un ordinateur, mais toutes représentées sous forme binaire
- L'information élémentaire = le bit, les informations plus complexes (instruction / données telles que caractère, nombre, ...) = un ensemble de bits

Codage des informations

- Le codage d'une information = correspondance entre la représentation externe de l'information (caractère A ou nombre 36) et sa représentation binaire (suite de bits)
- C'est l'utilisation d'une information qui en détermine le type (décodage appliqué, lieu d'utilisation...)
- Besoin de codage pour les informations traitées par le processeur :
  - Les instructions : codage des instructions, les instructions et leur codage dépendent du (type de) processeur
  - Les données numériques (N, Z, etc.), alphanumériques ou plus complexes (image) : codage avec des normes (complément à deux, ASCII, UTF-8, RGB)

Système de numération

- Un **système de numération** fait correspondre à un nombre N un certain formalisme écrit et oral
- Dans un système de base $B > 1$, les nombres 0, 1, 2, ..., $B - 1$ sont appelés **chiffres**

Expression dans une base B

Tout entier naturel N peut être exprimé comme une somme de multiples de puissance de la base B, les multiples étant des chiffres (donc $< B$)
$N = \sum_i a_i B^i = a_n B^n + a_{n-1} B^{n-1} + ... + a_1 B + a_0$ avec $\forall i, a_i < B$

- La notation condensée de l'entier naturel N dans la base B est $a_n a_{n-1} ... a_1 a_0{}_B$
- Cette notation est pondérée : chaque position correspond à une puissance de la base B et il y a un chiffre pour chaque position (éventuellement 0)

Notations pour les systèmes de base

Notation avec indice

- b pour la base 2/le binaire : 111_b
- d pour la base 10/le décimal : 111_d
- h pour la base 16/l'hexadécimal : 111_h
- Par défaut, sans indice, il s'agit de la base 10

Représentation en base 2

- Les informations manipulées par un ordinateur étant des mots binaires, les nombres entiers sont représentés en base 2 et les chiffres sont 0 et 1

Interprétation des entiers naturels représentés en binaire

$$(a_{n-1}...a_1a_0)_b = N_d = (\sum_{i=0}^{n-1} a_i 2^i)_d$$

Exemples

$N_d = 110011_b = (1 * 2^5 + 1 * 2^4 + 0 * 2^3 + 0 * 2^2 + 1 * 2^1 + 1 * 2^0)_d$
$N_d = (2^5 + 2^4 + 2^1 + 2^0)_d = 51_d$

- Nombres entiers exprimés en binaire $\Rightarrow$ grand nombre de bits
- Représentation en base 16/en hexadécimal préférée car conversion depuis/vers binaire simple et notation plus dense

Représentation en hexadécimal

- En base 16, on utilise les symboles 0, 1, ..., 8, 9, A, B, C, D, E, F pour les 16 chiffres, avec la correspondance suivante :

Hexadécimal	Décimal	Binaire
0_h	0	0000_b
1_h	1	0001_b
2_h	2	0010_b
3_h	3	0011_b
4_h	4	0100_b
5_h	5	0101_b
6_h	6	0110_b
7_h	7	0111_b

Hexadécimal	Décimal	Binaire
8_h	8	1000_b
9_h	9	1001_b
A_h	10	1010_b
B_h	11	1011_b
C_h	12	1100_b
D_h	13	1101_b
E_h	14	1110_b
F_h	15	1111_b

Interprétation d'un naturel représenté en hexadécimal

$N_d = (a_{n-1}a_{n-2}a_1a_0)_h$
$N_d = (a_{n-1}16^{n-1} + a_{n-2}16^{n-2} + ... + a_1 16 + a_0)_d$

- Il y a quatre fois moins de symboles que dans la notation en binaire
- Attention $1001_h \neq 1001_b$!

Représentation des valeurs des mots

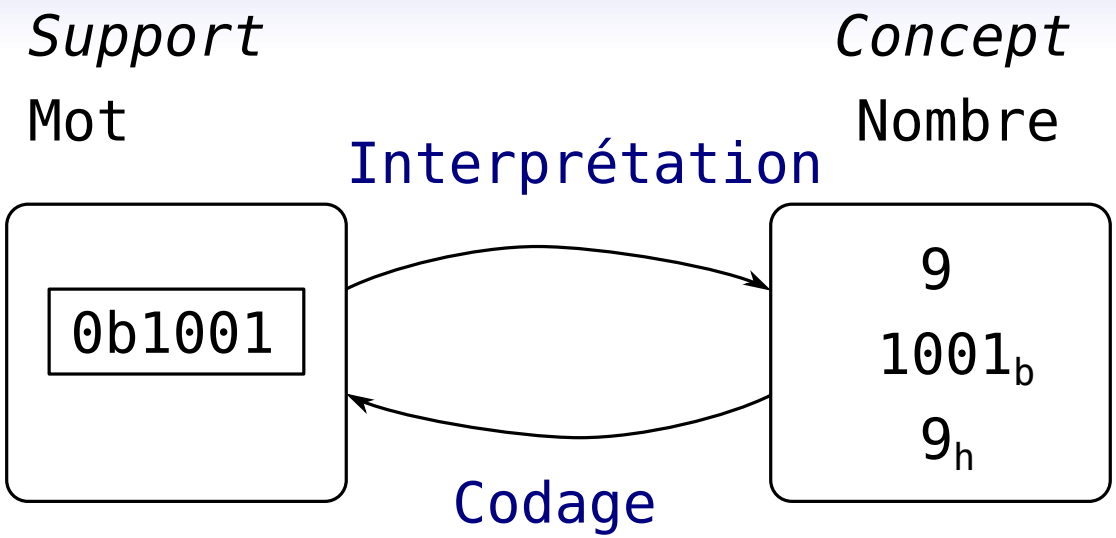
- Un mot peut être interprété de **différentes façons**, pas uniquement comme un nombre non signé
- $\Rightarrow$ Pour décrire la valeur d'un mot – par exemple le contenu d'un registre – on utilise une notation **avec préfixe** :
  - Le préfixe $0x$ pour représentation hexadécimale : $0x11$, $0x0445$, $0x1234ABCD$
  - Le préfixe $0b$ pour la représentation binaire : $0b1110$, $0b00010011$
- Cette notation décrit le contenu du mot **sans l'interpréter** :
  - Ne dit pas s'il s'agit d'un nombre positif, d'un nombre négatif, d'un caractère, d'une instruction...
  - Décrit juste la valeur des bits 1 à 1, du bit de **poids faible** (le plus à droite) au bit de **poids fort** (le plus à gauche)
- Les 0 en tête sont généralement écrits pour indiquer la taille du mot
- La taille de représentation (taille du mot) doit être connue : c'est normalement toujours le cas

Codage entiers naturels

Codage entiers naturels

- Pour encoder un entier naturel sur un mot, on utilise la représentation binaire du nombre
- Le bit de poids faible (bit n°0) correspond à 2^0, le bit de poids fort (bit n° $N-1$) à 2^{N-1}
- Ce codage s'appelle le codage **entiers naturels**

Codage entiers naturels



Exemple

- Sur 4 bits, le mot 0b1001 s'interprète comme la valeur 9 selon le codage **entiers naturels**, et $9 = 1001_b$
- Néanmoins, ce sont **deux choses différentes** : le même mot pourrait être interprété d'une façon différente, en utilisant un autre codage
- Le nombre 3 se code 0b0011 sur un mot de 4 bits

Représentation en machine : taille bornée

Intervalle de représentation

- Sur p symboles en base B, représentation possible des entiers naturels compris dans l'intervalle $[0, B^p - 1]$
- Sur un mot de n bits, représentation possible de l'intervalle [0, 2^n - 1] avec le codage **entiers naturels**

Exemples

- Sur 3 chiffres en décimal, intervalle [0, 999]
- Sur 3 symboles en hexadécimal, intervalle [0, 4095 = FFF_h]
- Sur 8 bits, intervalle [0, 255 = 11111111_b]

Extension de la représentation d'un entier naturel

Extension de p à n bits

- Soit un mot de p bits contenant une valeur v, interprété selon le codage entier naturel par le nombre d
- Pour que le mot v' de n bits ($n > p$) encode également le nombre d, il faut :
  - Copier les bits de poids faible de v dans v'
  - Mettre les bits de poids fort restant à 0

Extension de 4 à 8 bits

- 3 sur 4 bits se code 0b0011 $\rightarrow$ 3 sur 8 bits se code 0b0000 0011
- 9 sur 4 bits se code 0b1001 $\rightarrow$ 9 sur 8 bits se code 0b0000 1001

Changement de base

Comment passer d'une base B_1 à une base B_2 ?

- Algorithmes de conversion
  - Par divisions successives
  - Par tableau de puissance
- Correspondance simple entre certaines bases (2 vers 16, 16 vers 2)

Algorithme de conversion par divisions successives

Conversion de la base 10 à une base $B > 1$ pour un nombre N donné possible par division successive en s'apercevant que le symbole des unités est le reste de la division euclidienne par B.

Algorithme

$i \leftarrow 0$
$Q \leftarrow 1$
while $Q > 0$ **do**
  $(Q, R) \leftarrow \frac{N}{B}$
  $a_i \leftarrow R$
  $N \leftarrow Q$
  $i \leftarrow i + 1$
end while
$a_i \leftarrow 0$
Return $a_{j, j \in [0,i]}$

Exemple de conversion par divisions successives

Conversion de 25_d en binaire

Conversion de la base 2 à la base 16

Conversion base 2 → base 16

- Séparer le nombre binaire en quartet (paquet de 4 bits) en partant de la droite
- Convertir chaque quartet en son symbole hexadécimal

Exemple

Conversion de la base 16 à la base 2

Conversion base 16 $\rightarrow$ base 2

- Convertir chaque symbole hexadécimal en quartet/mot de 4 bits

Exemple