

Architecture des ordinateurs

Objectif

Comprendre la structure matérielle d'un ordinateur.
Savoir choisir un ordinateur en fonction d'un besoin.





Architecture des ordinateurs

SOMMAIRE

1. **Généralités**
2. Architecture de base
3. Les mémoires
4. Le microprocesseur
5. Performances d'un microprocesseur
6. Échanges de données



Architecture des ordinateurs

1 Généralités

1.1 Définitions

1.2 Représentation de l'information

1.3 Conversions de bases

1.4 Codage de l'information

1.5 Notion de base de la numérisation



Architecture des ordinateurs

1.1 Définitions

D1- Les systèmes de traitement de l'information(ordinateurs) emploient une technique à deux états (binaire). L'information traitée par ceux-ci doit être traduite en symboles compréhensibles et manipulables par ces systèmes.

D2- Deux types d'informations(données) peuvent être distinguées selon leurs nature et les transformation qu'elles subissent pour être traitées par les systèmes informatiques :

a- Les données discrètes: L'information correspond à un assemblage d'une suite d'éléments indépendants les uns des autres (c'est une suite discontinue de valeurs) et dénombrable (c'est un ensemble fini).

Par exemple : Un texte, qui est un ensemble de lettres (ou de symboles) qui forment des mots.

b- Les données continues ou analogiques: Résultent de la variation continue d'un phénomène naturel.

Par exemple : le son, l'image, la température, la pression, le poids... etc.

D3- Ordinateur : Machine automatique de traitement de l'information Obéit à un programme (binaire) formé par des suites d'opérations arithmétiques et logiques.

D4- Architecture d'un ordinateur: Représente l'organisation des différentes unités de traitement et leurs interconnexions.

Architecture des ordinateurs

1.2 Représentation de l'information

Pour pouvoir traiter ces informations par des équipements informatiques, chaque élément d'information va devoir être substituée par une valeur binaire.

Cette opération porte le nom de **Codage** pour les informations discrètes et **Numérisation** pour les informations continues.

Unité Information numérique = information binaire

n = **1 Bit** (Binary digit)

Représentée par 2 niveaux de tensions physiques (électriques).

Codée par un « 0 » logique ou « 1 » logique.

1 Octet = 1 **byte**(en anglais) = 8 bits

Préfixes binaires conventionnels

| Nom | Symbole | Puissance | ~Déc | Nombre |
|-------|---------|--------------------------------|-----------|----------|
| unité | | $2^0 = 1$ | 10^0 | un |
| kilo | K | $2^{10} = 1024$ | 10^3 | mille |
| mega | M | $2^{20} = 1048576$ | 10^6 | million |
| giga | G | $2^{30} = 1073741824$ | 10^9 | milliard |
| tera | T | $2^{40} = 1099511627776$ | 10^{12} | billion |
| peta | P | $2^{50} = 1125899906842624$ | 10^{15} | billiard |
| exa | E | $2^{60} = 1152921504606846976$ | 10^{18} | trillion |

Architecture des ordinateurs

Systèmes de numération

Le système de numération utilise des symboles appelés chiffres. Le nombre de chiffres utilisés correspond à la base. D'une manière générale, un nombre en base β s'écrit sous cette forme (forme éplandue) :

$$\sum_{i=0}^n (b_i \beta^i) = b_n \beta^n + \dots + b_2 \beta^2 + b_1 \beta^1 + b_0 \beta^0$$

où :

b_i est le chiffre de la base de rang i

β^i est la puissance de la base β d'exposant de rang i

Par convention, l'écriture d'un nombre **N** s'effectue par la juxtaposition de chiffres possédant chacun un poids égal à une puissance entière de sa base de numération. Les chiffres s'écrivent de la gauche vers la droite par valeur décroissante de leur poids. Le poids d'un chiffre dépend donc de son rang et du système de numération adopté :

Par exemple en base 10 le nombre **N1= 7248,5** (sous sa forme résumée), s'écrit sous sa forme éplandue comme suite :

$$N_1 = (7248,5)_{10} = 7 \times 10^3 + 2 \times 10^2 + 4 \times 10^1 + 8 \times 10^0 + 5 \times 10^{-1}$$

↑
chiffre de poids 10^3

↑
chiffre de poids 10^2

↑
chiffre de poids 10^1

↑
chiffre de poids 10^0

↑
chiffre de poids 10^{-1}



Architecture des ordinateurs

1.3 Conversions de bases

Différents systèmes informatiques et langages de programmation peuvent utiliser des bases différentes pour représenter les données. Par exemple, les ordinateurs utilisent la base 2, tandis que les humains utilisent principalement la base 10.

La conversion entre ces bases permet de faire communiquer des systèmes et des programmes qui utilisent des bases différentes.

Conversion Décimal → Binaire

La conversion d'un nombre de la base 10 (décimale) en base 2 (binaire) est un processus visant à représenter ce nombre en utilisant exclusivement les chiffres 0 et 1.

Cette conversion est réalisée grâce à la méthode de la division successive par 2. On commence par diviser le nombre décimal par 2, enregistrant le reste de cette division.

Ensuite, on répète ce processus avec le quotient précédent, enregistrant chaque nouveau reste. On continue ainsi jusqu'à ce que le quotient devienne égal à zéro. La séquence des restes obtenus est ensuite lue de bas en haut pour obtenir la représentation binaire du nombre.

Architecture des ordinateurs

Conversion Décimal → Binaire

Exemple :

On effectue des divisions successives par 2 et on réécrit les restes de bas en haut.

$$(55)_{10} = (110111)_2$$

| dividende | | diviseur | |
|-----------|--|----------|----------|
| ... | | 2 | quotient |
| reste | | | |
| 55 | | 2 | |
| 1 | | 27 | |
| | | 2 | |
| | | 13 | |
| | | 2 | |
| | | 6 | |
| | | 2 | |
| | | 3 | |
| | | 2 | |
| | | 1 | |
| | | 1 | |

Architecture des ordinateurs

Conversion Binaire → Décimal

La conversion de binaire en décimal est un processus permettant de traduire une représentation binaire (système composé de 0 et 1) en une valeur décimale (système décimal). Pour effectuer cette conversion, on attribue des poids croissants de droite à gauche aux chiffres binaires et on additionne ces valeurs pondérées. Chaque chiffre binaire représente une puissance de 2, et en les combinant, on obtient la valeur décimale équivalente

Exemple :

$$\begin{array}{cccccc} (1 & 1 & 0 & 0 & 1)_2 & = & 2^4 + 2^3 + 2^0 = 16 + 8 + 1 \\ 2^4 & 2^3 & 2^2 & 2^1 & 2^0 & & = (25)_{10} \\ \mathbf{16} & \mathbf{8} & \mathbf{4} & \mathbf{2} & \mathbf{1} & & \end{array}$$

Architecture des ordinateurs

Hexadécimal (Base 16)

L'hexadécimal est un système en base 16 utilisant les chiffres de 0 à 9 et les lettres de A à F. Il est plus facile de représenter une valeur à l'aide d'un seul chiffre hexadécimal que de quatre bits binaires.

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

Architecture des ordinateurs

Conversion Hexadécimal → Décimal

La conversion de l'hexadécimal en décimal consiste à traduire un nombre hexadécimal (base 16) en sa valeur équivalente en décimal (base 10). Le processus repose sur l'attribution de valeurs croissantes de 0 à 15 aux chiffres hexadécimaux, puis en multipliant chaque chiffre par la puissance correspondante de 16 et en additionnant les produits.

Exemple :

256 16 1

16^2 16^1 16^0

$$\begin{aligned} (B \ 2 \ 2)_{16} &= B \times 256 + 2 \times 16 + 2 \times 1 \\ &= 11 \times 256 + 32 + 2 \\ &= (2850)_{10} \end{aligned}$$

Conversion Décimal → Hexadécimal

On effectue des divisions successives par 16

Architecture des ordinateurs

Conversion Binaire → Hexadécimal

On regroupe les bits par quartets et on remplace les quartets par leur équivalent hexadécimal.

Exemple :

$$\begin{array}{ccccccc} 8 & 4 & 2 & 1 & 8 & 4 & 2 & 1 & 8 & 4 & 2 & 1 \\ (& 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 &)_2 \\ (& B & & & 2 & & & & 2 & & & &)_{16} \end{array}$$

Conversion Hexadécimal → Binaire

On effectue l'opération inverse



Architecture des ordinateurs

1.4 Codage de l'information

Définition

Le codage est l'opération qui consiste à faire correspondre à chaque symbole d'un alphabet une représentation binaire unique (mot code).

On distingue deux types de codes : Les codes de **longueur fixe** et les codes de **longueur variable**.

Les informations directement traitées par un ordinateurs sont des :

Entiers : naturels et relatifs

Flottants : nombres réels

Caractères : symboles

Instructions : Opération élémentaire exécutée par un ordinateur

Architecture des ordinateurs

A- Codage de longueur fixe

Code des nombres :

Chaque symbole est codé par un nombre de bits fixe, appelé longueur du code.

Quelques exemples de codes:

Code des entiers naturels : **BCD Binary Coded Décimal**

Chaque chiffre de la base 10 est remplacé par son équivalent binaire sur 4 bits

| Décimal | Binaire sur 4 bits | | | |
|---------|--------------------|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 |
| 6 | 0 | 1 | 1 | 0 |
| 7 | 0 | 1 | 1 | 1 |
| 8 | 1 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 |

Code des entiers relatifs : **Code Complément à 2 (c a 2)** :(voir TD)

$$\text{Soit } (-x)_{\text{ca2}} = \overline{x} + 1$$

Le bit de poids fort = bit de signe

Code des nombres réels : Normes IEEE754

Architecture des ordinateurs

Code des caractères alphanumériques (Table ASCII) :

Le code ASCII (American Standard Code for Information Inter-change)

Les caractères représentables en ASCII sont les lettres majuscules et minuscules, les chiffres ainsi que les symboles mathématiques et de ponctuation.(Voir TD)

Caractères spéciaux

Chiffres

Lettres

20/11/24

| Dec | Hx | Oct | Char | Dec | Hx | Oct | Html | Chr | Dec | Hx | Oct | Html | Chr | Dec | Hx | Oct | Html | Chr |
|-----|----|-----|------------------------------------|-----|----|-----|-------|--------------|-----|----|-----|-------|----------|-----|----|-----|--------|------------|
| 0 | 0 | 000 | NUL (null) | 32 | 20 | 040 | | Space | 64 | 40 | 100 | @ | @ | 96 | 60 | 140 | ` | ` |
| 1 | 1 | 001 | SOH (start of heading) | 33 | 21 | 041 | ! | ! | 65 | 41 | 101 | A | A | 97 | 61 | 141 | a | a |
| 2 | 2 | 002 | STX (start of text) | 34 | 22 | 042 | " | " | 66 | 42 | 102 | B | B | 98 | 62 | 142 | b | b |
| 3 | 3 | 003 | ETX (end of text) | 35 | 23 | 043 | # | # | 67 | 43 | 103 | C | C | 99 | 63 | 143 | c | c |
| 4 | 4 | 004 | EOT (end of transmission) | 36 | 24 | 044 | $ | \$ | 68 | 44 | 104 | D | D | 100 | 64 | 144 | d | d |
| 5 | 5 | 005 | ENQ (enquiry) | 37 | 25 | 045 | % | % | 69 | 45 | 105 | E | E | 101 | 65 | 145 | e | e |
| 6 | 6 | 006 | ACK (acknowledge) | 38 | 26 | 046 | & | & | 70 | 46 | 106 | F | F | 102 | 66 | 146 | f | f |
| 7 | 7 | 007 | BEL (bell) | 39 | 27 | 047 | ' | ' | 71 | 47 | 107 | G | G | 103 | 67 | 147 | g | g |
| 8 | 8 | 010 | BS (backspace) | 40 | 28 | 050 | (| (| 72 | 48 | 110 | H | H | 104 | 68 | 150 | h | h |
| 9 | 9 | 011 | TAB (horizontal tab) | 41 | 29 | 051 |) |) | 73 | 49 | 111 | I | I | 105 | 69 | 151 | i | i |
| 10 | A | 012 | LF (NL line feed, new line) | 42 | 2A | 052 | * | * | 74 | 4A | 112 | J | J | 106 | 6A | 152 | j | j |
| 11 | B | 013 | VT (vertical tab) | 43 | 2B | 053 | + | + | 75 | 4B | 113 | K | K | 107 | 6B | 153 | k | k |
| 12 | C | 014 | FF (NP form feed, new page) | 44 | 2C | 054 | , | , | 76 | 4C | 114 | L | L | 108 | 6C | 154 | l | l |
| 13 | D | 015 | CR (carriage return) | 45 | 2D | 055 | - | - | 77 | 4D | 115 | M | M | 109 | 6D | 155 | m | m |
| 14 | E | 016 | SO (shift out) | 46 | 2E | 056 | . | . | 78 | 4E | 116 | N | N | 110 | 6E | 156 | n | n |
| 15 | F | 017 | SI (shift in) | 47 | 2F | 057 | / | / | 79 | 4F | 117 | O | O | 111 | 6F | 157 | o | o |
| 16 | 10 | 020 | DLE (data link escape) | 48 | 30 | 060 | 0 | 0 | 80 | 50 | 120 | P | P | 112 | 70 | 160 | p | p |
| 17 | 11 | 021 | DC1 (device control 1) | 49 | 31 | 061 | 1 | 1 | 81 | 51 | 121 | Q | Q | 113 | 71 | 161 | q | q |
| 18 | 12 | 022 | DC2 (device control 2) | 50 | 32 | 062 | 2 | 2 | 82 | 52 | 122 | R | R | 114 | 72 | 162 | r | r |
| 19 | 13 | 023 | DC3 (device control 3) | 51 | 33 | 063 | 3 | 3 | 83 | 53 | 123 | S | S | 115 | 73 | 163 | s | s |
| 20 | 14 | 024 | DC4 (device control 4) | 52 | 34 | 064 | 4 | 4 | 84 | 54 | 124 | T | T | 116 | 74 | 164 | t | t |
| 21 | 15 | 025 | NAK (negative acknowledge) | 53 | 35 | 065 | 5 | 5 | 85 | 55 | 125 | U | U | 117 | 75 | 165 | u | u |
| 22 | 16 | 026 | SYN (synchronous idle) | 54 | 36 | 066 | 6 | 6 | 86 | 56 | 126 | V | V | 118 | 76 | 166 | v | v |
| 23 | 17 | 027 | ETB (end of trans. block) | 55 | 37 | 067 | 7 | 7 | 87 | 57 | 127 | W | W | 119 | 77 | 167 | w | w |
| 24 | 18 | 030 | CAN (cancel) | 56 | 38 | 070 | 8 | 8 | 88 | 58 | 130 | X | X | 120 | 78 | 170 | x | x |
| 25 | 19 | 031 | EM (end of medium) | 57 | 39 | 071 | 9 | 9 | 89 | 59 | 131 | Y | Y | 121 | 79 | 171 | y | y |
| 26 | 1A | 032 | SUB (substitute) | 58 | 3A | 072 | : | : | 90 | 5A | 132 | Z | Z | 122 | 7A | 172 | z | z |
| 27 | 1B | 033 | ESC (escape) | 59 | 3B | 073 | ; | ; | 91 | 5B | 133 | [| [| 123 | 7B | 173 | { | { |
| 28 | 1C | 034 | FS (file separator) | 60 | 3C | 074 | < | < | 92 | 5C | 134 | \ | \ | 124 | 7C | 174 | | | |
| 29 | 1D | 035 | GS (group separator) | 61 | 3D | 075 | = | = | 93 | 5D | 135 |] |] | 125 | 7D | 175 | } | } |
| 30 | 1E | 036 | RS (record separator) | 62 | 3E | 076 | > | > | 94 | 5E | 136 | ^ | ^ | 126 | 7E | 176 | ~ | ~ |
| 31 | 1F | 037 | US (unit separator) | 63 | 3F | 077 | ? | ? | 95 | 5F | 137 | _ | _ | 127 | 7F | 177 | | DEL |

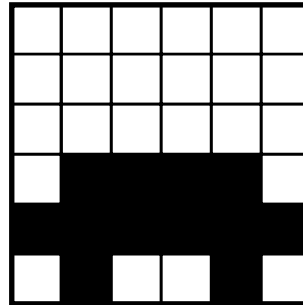
Architecture des ordinateurs

Code d'une image matricielle

Image matricielle = matrice de points élémentaires

= **P**icture **É**lément = pixel

Chaque pixel est codé en binaire sur un certains nombre de bits



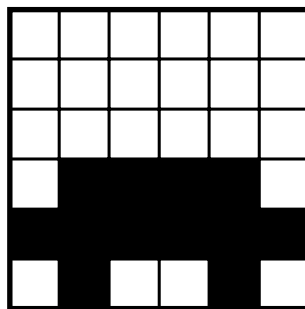
La **définition** d'une image est le nombre de pixels (de points) constituant l'image, c'est-à-dire le nombre de colonnes de l'image que multiplie son nombre de ligne.

La **résolution**, exprimée en DPI(Dot Per Inch ou pixels par pouce).
C'est le rapport entre le nombre de pixels de l'image et la taille réelle de sa représentation sur un support physique (scanner ou imprimante).

Architecture des ordinateurs

Image noir et blanc :

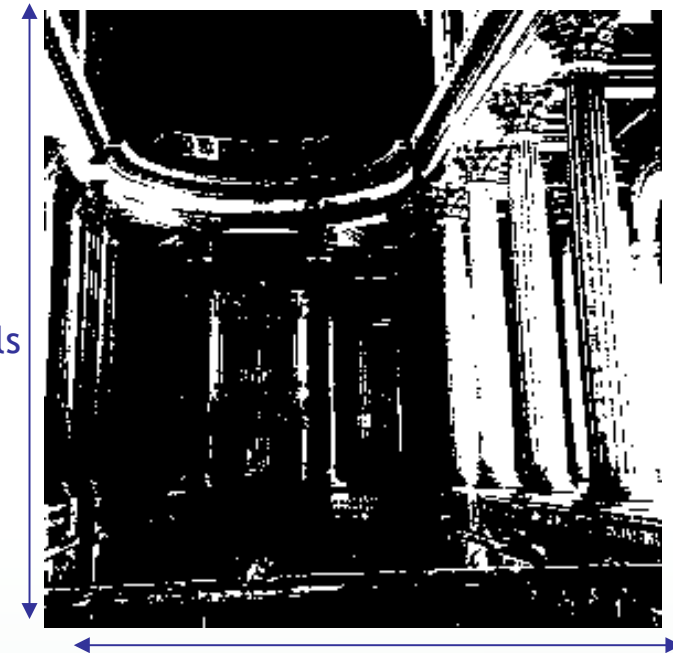
Chaque pixel est codé sur 1 bit : 0 = blanc 1 = noir



| | | | | | |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |

6 x 6 = 36 pixels = 36 bits

303 pixels



303 pixels

303 x 303 x 1 bit = **91809 bits**

Architecture des ordinateurs

Image à plusieurs niveaux de gris

Chaque pixel est codé sur plusieurs bits

Si on code sur 8 bits = 1 octet = 1 pixel

303 pixels



303 pixels

$303 \times 303 \times 1 \text{ octet} = 91809 \text{ octets}$
 $= 734\,472 \text{ bits}$

Architecture des ordinateurs

Image couleur 24 bits

Le code **R****V****B** = Rouge, Vert, Bleu

Chaque couleur est codée sur 8 bits

La couleur du pixel est l'association des 3 couleurs

Chaque pixel est codé sur 3 octets soit 24 bits (true color)

303 pixels



303 pixels

$$\begin{aligned} 303 \times 303 \times 3 \text{ octets} &= 275\,424 \text{ octets} \\ &= 2\,203\,392 \text{ bits} \end{aligned}$$

20/11/24



Architecture des ordinateurs

B- Codage de longueur variable (La compression)

Je vais vous expliquer le code à longueur variable a travers un exemple concret :

Le Codage de Huffman :

Le codage de Huffman ou codage d'entropie est une méthode de compression de données qui permet de réduire la longueur du codage d'un alphabet. Il substitue à un code de longueur fixe un code de longueur variable.

Le principe est très simple : les caractères dans un texte qui apparaissent le plus souvent doivent être codés en un minimum d'espace possible (l'unité de traitement est ramenée au bit). Prenons l'exemple d'un fichier texte, chaque lettre en code ASCII est représentée sur un octet (8 bits). Le but est de recoder les données qui ont une fréquence apparition dans un texte très faible sur une longueur binaire supérieure à la moyenne, et de recoder les données très fréquentes sur une longueur binaire très courte.

Par exemple, la lettre Z dans un texte en français sera moins présent qu'un E. Le Z pourrait être codé sur 8 bits tandis que le E sur 3 bits. au final on aura un gain de bits.

Architecture des ordinateurs

1.5 Notion de base de la numérisation

Définition

La numérisation consiste à transformer des données continues (analogiques) en données numériques utilisables par les systèmes informatiques.

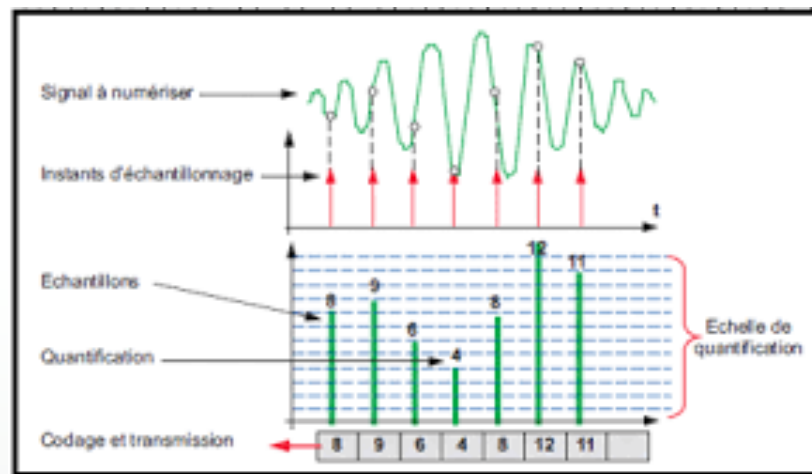
C'est à dire transformer une suite continue de valeurs en une suite discrète et finie

Théorème de Shannon et Nyquist :

Un signal analogique peut être reconstruit à partir des échantillons numérisés si la fréquence d'échantillonnage est égale à au moins deux fois la fréquence maximale du signal à échantillonner.

$$F_{ech} \geq 2f_{max}$$

Pour se faire, on prélève, a des instants significatifs, un échantillon (**échantillonnage**) du signal et on indique la valeur de son amplitude par rapport a une échelle finie (**quantification**). Une fois quantifiée, chaque valeur obtenue sera codifiée (**Codage**). voir le schéma ci-dessous :



Architecture des ordinateurs

*Du simple binaire naît
l'univers numérique, où
tout peut être imaginé et
créé.*

You can do anything!
"Just get up and do it, follow your dreams"