

Architecture des ordinateurs

Clement Jonquet

jonquet@lirmm.fr

Polytech' Montpellier

IG3 2011-2012

Objectifs du cours

Comprendre :

- Le fonctionnement des ordinateurs.
- L'organisation interne des machines.
- La représentation des données.
- L'exécution des programmes.
- Les principes d'un langage d'assemblage.

⇒ Voir "ce qui se passe sous le capot"!

⇒ Comprendre les concepts clés pour mieux appréhender la suite des cours d'informatique.

Plan du cours

- 1 Histoire de l'ordinateur
- 2 Présentation générale
- 3 Représentation interne des informations
- 4 Encodage de l'information
- 5 Circuits logiques
- 6 Composants électroniques
- 7 Mémoires
- 8 Unité centrale de traitement
- 9 Superordinateurs et microprocesseurs
- 10 Entrées / sorties
- 11 Assembleur
- 12 Introduction au langage MIPS

Ressources (1/2)



Sur la base des notes de cours de *Christophe Fiorio* (Polytech Montpellier) et plein d'autres.



Architecture et technologie des ordinateurs, *Paolo Zanella & Yves Ligier*, Dunod, 4ème édition, 2005.



Architecture de l'ordinateur, *Andrew Tanenbaum*, Pearson, 5ème édition, 2009.



Technologie des ordinateurs et des réseaux, *Pierre-Alain Goupille*, Dunod, 9ème édition, 2010.

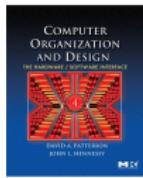
Ressources (2/2)



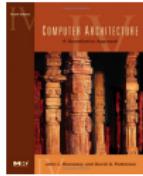
Architecture de l'ordinateur, *Robert Strandh & Irène Durand*, Dunod, 2005.



Architecture des machines et des systèmes informatiques, *Alain Cazez & Joëlle Delacroix*, 3ème édition, Dunod, 2008.



Computer Organization and Design, The Hardware/Software Interface, *David A. Patterson & John L. Hennessy*, Morgan Kaufmann, 4th edition, 2008.



Computer Architecture : A Quantitative Approach, *David A. Patterson & John L. Hennessy*, Morgan Kaufmann, 4th edition, 2009.

Alignement avec Zanella & Ligier, 2005

1. Histoire de l'ordinateur	Chapitre 1
2. Présentation générale	Chapitre 2
3. Représentation interne des informations	Chapitre 3
4. Encodage de l'information	Chapitre 4
5. Circuits logiques	Chapitre 5
6. Composants électroniques	Chapitre 6
7. Mémoires	Chapitre 7
8. Unité centrale de traitement	Chapitre 8
9. Superordinateurs et microprocesseurs	Chapitre 9
10. Entrées / sorties	Chapitre 10
Réseaux (IG4 2nd sem.)	Chapitre 11
Système d'exploitation (IG3/IG4 2nd sem.)	Chapitre 12
11. Assembleur	Chapitre 13
12. Introduction au langage MIPS	-
Génie logiciel (IG4)	Chapitre 14
Structure de données (IG3 Ada)	Chapitre 15
Multimedia (-)	Chapitre 16
Internet (IG5 AIOP)	Chapitre 17

Organisation du cours

- Cours condensé sur le début du semestre (jusqu'à mi-novembre) :
 - ▶ 21h Cours (14 séances de 1h30)
 - ▶ 10,5h TD (7 séances de 1h30)
 - ▶ 3h TP (2 séances de 1h30 à la suite)
- EDT : sur l'intranet Département IG.
- Cours 6 (composants) assuré par Michel Robert ou Lionel Torres (département ERII).
- Cours 9 (superordinateurs) assuré par Thierry Porcher (HPC Project).
- Présence en cours/TD non vérifié. Sauf pour les interventions.

Evaluation

- Examen final + examen de rattrapage.
- La note de rattrapage remplace la note d'examen.
- Questions de cours (5-6 pts).
- Exercice similaire aux TDs/TPs (14-15 pts).
- Support de cours/TD autorisés.

Cours en ligne

Environnement Numérique de Travail (ENT) de l'UM2

http:

//mon.univ-montp2.fr/claroline/course/index.php?cid=M513

Vous y trouverez :

- Description, agenda, annonces.
- Un forum... si vous avez des questions/remarques qui peuvent intéresser les autres.
- Les documents (PDF).
- Les résultats des examens.

Plan du cours

- 1 Histoire de l'ordinateur
- 2 Présentation générale
- 3 Représentation interne des informations
- 4 Encodage de l'information
- 5 Circuits logiques
- 6 Composants électroniques
- 7 Mémoires
- 8 Unité centrale de traitement
- 9 Superordinateurs et microprocesseurs
- 10 Entrées / sorties
- 11 Assembleur
- 12 Introduction au langage MIPS

Besoin de calculer

- L'ordinateur est né du besoin de calculer
 - ▶ toujours plus complexe
 - ▶ toujours plus vite
- ⇒ Automatiser le calcul



Un peu d'histoire...

- XVIIe siècle et avant : les principes fondateurs
- XIXe siècle : les calculateurs
- XXe siècle : théorie de l'information + machine universelle
- 1945 : Architecture de Von Neumann et naissance de l'ordinateur
- ~1950 : 1ère génération : tubes à vides
- ~1960 : 2ème génération : transistors
- ~1970 : 3ème génération : circuits intégrés
- ~1980 : 4ème génération : puces avec des millions de transistors

Fondations

- *John Neper* (1614) : théorie de logarithmes permettant de transformer des multiplications en additions
- *Blaise Pascal* (1642) : première machine à calculer, la Pascaline (principe de roues dentées). Cette machine pouvait additionner et soustraire des nombres de six chiffres et prenait en compte les retenues !
- *Gottfried Leibniz* (1673), mathématicien de génie
 - ▶ Améliore la machine de Pascal en y ajoutant un mécanisme permettant d'automatiser l'exécution répétitive d'additions et de soustraction. La première machine à calculer autorisant les 4 opérations arithmétiques était née.
 - ▶ Système binaire sous sa forme moderne basé sur les deux chiffres 0 et 1 et montra la puissance et la simplicité de l'arithmétique binaire, système utilisé par les ordinateurs actuels.

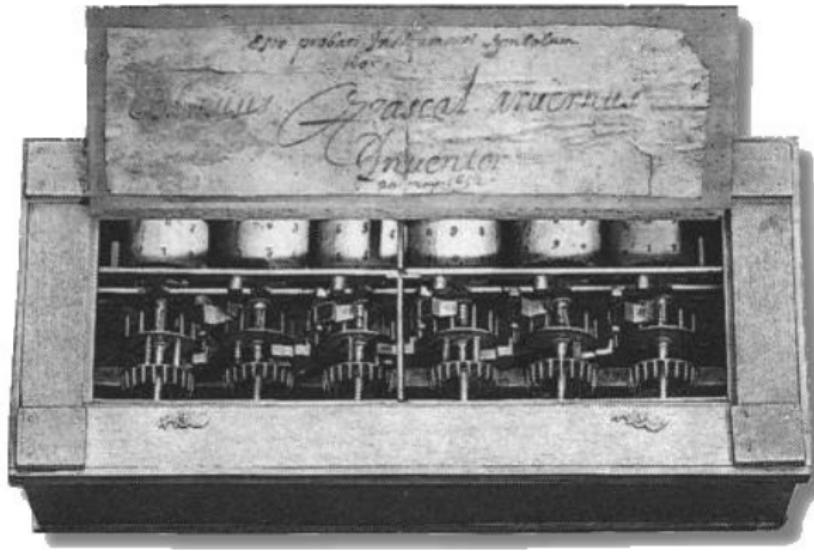


Figure: La Pascaline

Les grandes étapes

- *Jean-Baptiste Falcon* (1728) : commande pour métier à tisser
⇒ Première machine avec un programme externe !
- *Joseph Jacquard* (1805) : carte perforées pour métier à tisser
- *Charles Babbage* (1833) : père de l'ordinateur (rapprochement entre commande externe & machine calculer)
⇒ Réalisation de sa machine analytique avec l'aide *Ada Augusta*, l'ancêtre des ordinateurs.
- *George Boole* (1854) : système de logique symbolique
⇒ Algèbre de boole : fonctions logique décrivant le fonctionnement d'un système le plus simple possible.
- *Herman Hollerith* (1890) : calculateur statistique
⇒ Cartes perforées : premiers supports d'entrée-sortie et premières mémoires de masse.

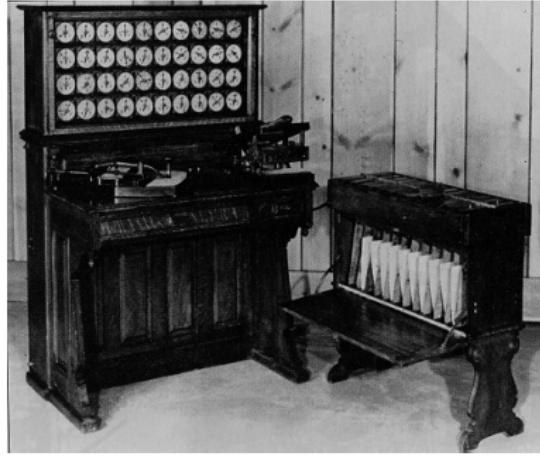


Figure: Herman Hollerith's Tabulating Machine

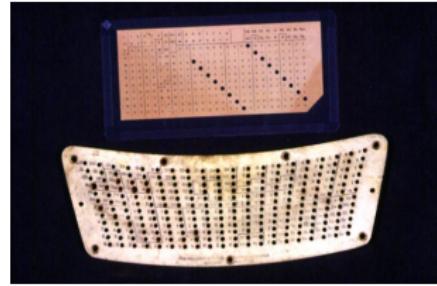


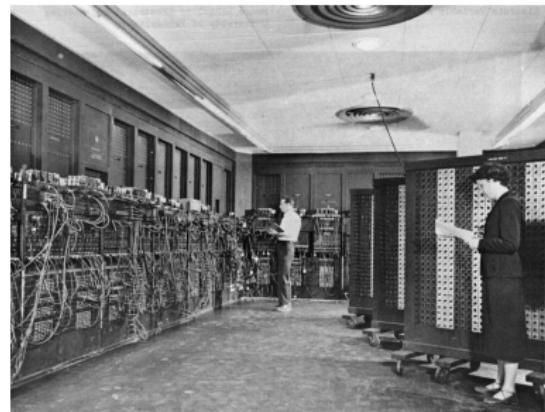
Figure: Carte perforées

Naissance de l'ordinateur

- *Claude Shannon* (1948) : chiffres binaires pour les relations logiques et les calculs logiques et arithmétiques
⇒ Tous calculs peuvent être réalisés avec les 3 opérations logiques de base ET, OU, NON
- *Alan Turing* : machine universelle ou Machine de Turing décrivant un modèle abstrait du fonctionnement des appareils mécaniques de calcul
⇒ Invente les concepts de programmation et de programme
- *John Von Neumann* (1945) : Idée clé : Enregistrer le programme en mémoire
⇒ Architecture de l'ordinateur moderne : l'architecture de Von Neumann

ENIAC (Electronic Numerical Integrator Analyser and Calculator) -1945

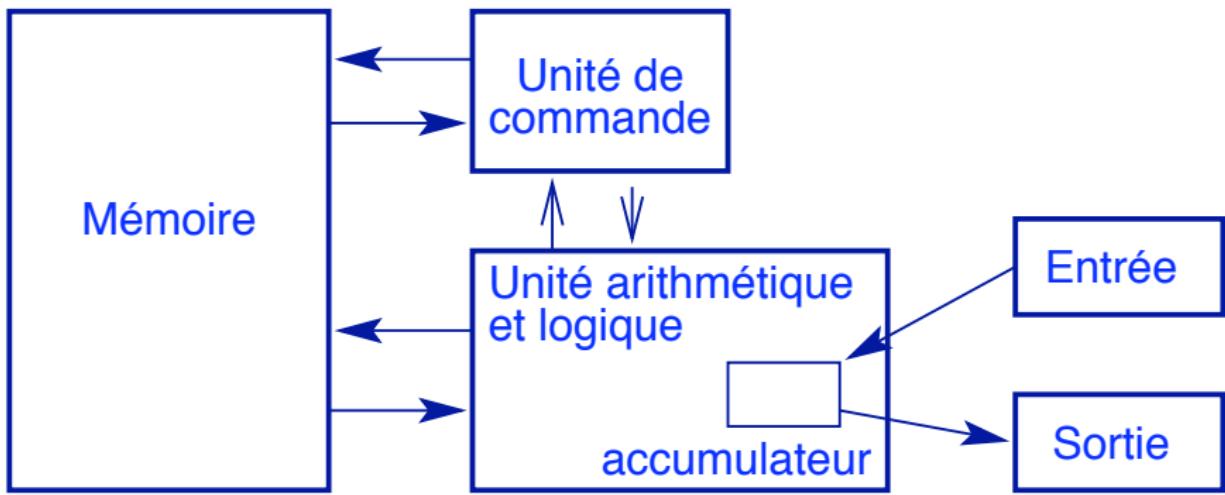
- Technologie des tubes à vide (18000)... 30 tonnes
- Construit à l'Université de Pennsylvanie
- Construit pour être Turing-complet
- Multiplication de 2 nombres de 10 chiffres en 3ms !



Principes de l'ordinateur selon Von Neumann

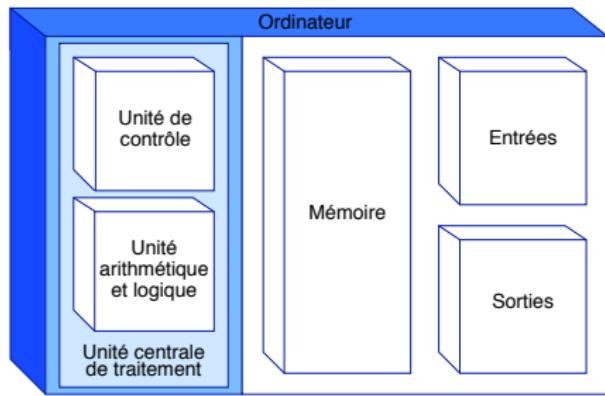
- Machine universelle contrôlée par programme.
- Instructions du programme codées sous forme numérique binaire et enregistrées en mémoire.
- Instruction exécutées normalement en séquence mais pouvant être modifiées par le programme lui-même.
- Existence d'instructions permettant les ruptures de séquences.

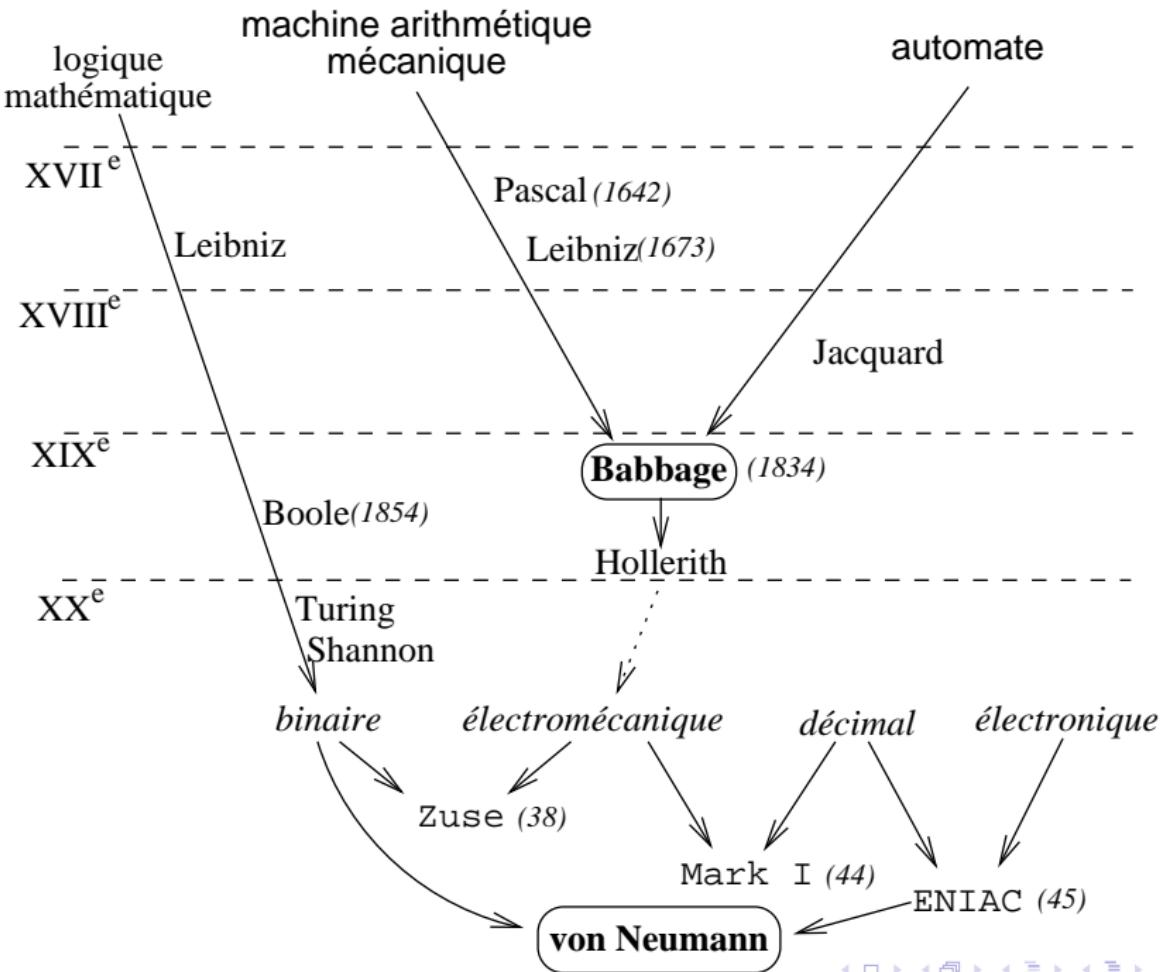
Architecture de Von Neumann



Composants classiques d'un ordinateur

- *la mémoire centrale* qui contient les données et les programmes à exécuter
- *l'unité centrale de traitement* qui exécute les programmes chargés en mémoire
- *les unités d'entrée/sortie* qui permettent le lien et l'échange d'information avec les périphériques (clavier, écran, souris, imprimante, etc.)





Naissance de l'industrie informatique

- ~1950 : 1ère génération : **tubes à vides**
- ~1960 : 2ème génération : **transistors**. Moindre coût, bus unique pour interconnecter les différents composants



- ~1970 : 3ème génération : **circuits intégrés**.
Multi-programmation (plusieurs programmes en mémoire) ⇒
Le principe est simple, dès qu'un programme est en attente
d'une entrée-sortie, l'unité de commande poursuit l'exécution
d'un autre programme.
- ~1980 : 4ème génération : **puces avec des millions de transistors** (Very-large-scale integration (VLSI)) +
multiplication des unités périphériques (stockage, écran,
imprimante, etc.)

1911

2011

Le rapprochement entre trois sociétés américaines donne naissance à la CTRC (Computer Tabulating Recording Company). L'entreprise IBM, l'entreprise horlogère New York et qui compte 1.300 employés, fabrique des instruments de mesure mécanique et des machines à pointer.

1914

Les machines à calculer, souvent baptisées « Watson », participent au jeu télévisé « Jeopardy ! » et bat plusieurs concurrents humains. Capable de comprendre le langage naturel et de répondre à des questions, Watson préfigure le futur de l'informatique et va ouvrir diverses applications dans la santé, la finance ou encore l'énergie.

**1997**

L'animateur « Beep Blue », développé dans les laboratoires du groupe, bat le champion du monde d'échecs, le Russe Garry Kasparov. La machine a effectué 200 millions de coups par seconde.

**1995**

En pleine transformation de son modèle économique, IBM dévoile sa stratégie « e-business », destinée à offrir des solutions informatiques aux entreprises pour pouvoir profiter de la puissance émergente d'Internet et développer leurs affaires.

1981

Big Blue lance son premier PC, commercialisé à 1.500 dollars. Son succès est un précurseur. IBM permet au marché de décoller et devient un nouveau standard pour l'industrie. Le magazine « Time » désignera le PC comme « Man of the Year 1982 ».

**1971**

IBM introduit un nouveau modèle de stockage magnétique de données informatiques, le « floppy disk », qui donnera naissance à la disquette. Facile à utiliser, celle-ci deviendra incontournable avec l'avènement du PC.

1969

Le premier homme marche sur la Lune. La Nasa passe commande à IBM pour la mission Apollo 13 avec IBM, qui fournit les ordinateurs et les ingénieurs nécessaires à sa réussite.



IDE / SOURCE : « LES ECHOS »
PHOTOS : IBM ET AFP

Architecture des ordinateurs

1935

IBM commercialise la première machine à écrire électrique. Le groupe américain sera l'un des principaux acteurs du marché et continuera à fabriquer ces machines jusqu'aux années 1990.

**1944**

IBM présente son premier calculateur automatisé, l'ASCC. Cet ancêtre de l'ordinateur, installé à l'université de Harvard, est la première machine capable de réaliser de longues opérations. Longue de 15 m et haute de plus de 3 m, elle pèse près de 3 tonnes.

**1948**

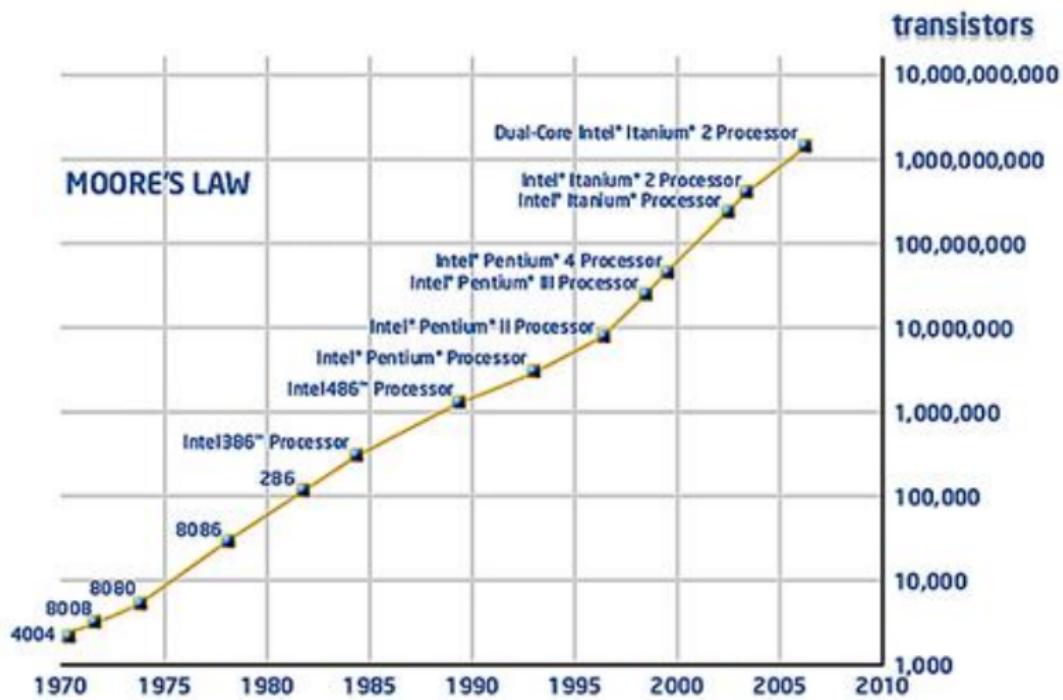
Console de contrôle informatique (SSEC), entièrement électrique et dotée de grandes capacités de mémoire.

**1956**

Les premières machines ayant disposé d'un véritable système d'exploitation sont créées dans les laboratoires du groupe informatique. La technologie de stockage utilisée par IBM deviendra rapidement le standard dans l'industrie.

Loi de Moore (1965)

Nombre de transistors (des microprocesseurs) sur une puce de silicium double tous les deux ans.



Émergence d'une nouvelle discipline

Informatique

La science du traitement rationnel de l'information, considérée comme le support des connaissances dans les domaines scientifiques, économiques et sociaux, notamment à l'aide de machines automatiques.

-

Ensemble des applications de cette science, mettant en oeuvre des matériels (ordinateurs) et des logiciels.

-

(Académie française, 1965).

Aller plus loin...

- <http://www.lamef.bordeaux.ensam.fr/~jlc/ASI/Historique/histo.html>
- <http://histoire.info.online.fr/>
- http://fr.wikipedia.org/wiki/Histoire_de_l'informatique
- Reportage : "les cinglés de l'informatique"

Plan du cours

- 1 Histoire de l'ordinateur
- 2 Présentation générale
- 3 Représentation interne des informations
- 4 Encodage de l'information
- 5 Circuits logiques
- 6 Composants électroniques
- 7 Mémoires
- 8 Unité centrale de traitement
- 9 Superordinateurs et microprocesseurs
- 10 Entrées / sorties
- 11 Assembleur
- 12 Introduction au langage MIPS

Définitions

Ordinateur

Une machine de traitement de l'information (acquérir, conserver, traiter et restituer).

Type d'information & représentation

Valeurs numériques, textes, images, son, vidéos représentés sous forme de données numériques.

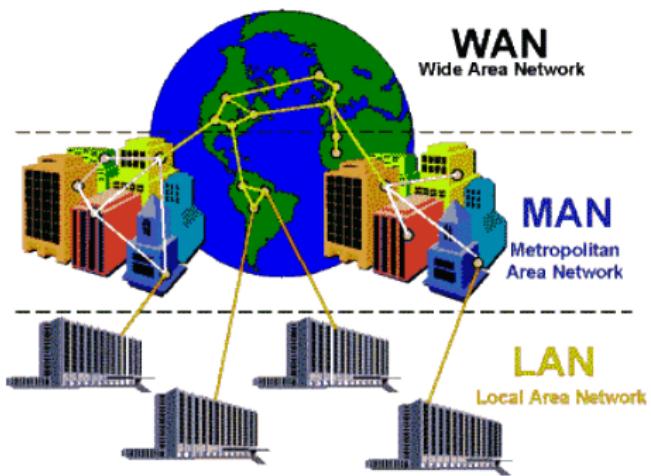
Informatique

Science du traitement de l'information.

Système informatique

Ensemble des moyens logiciels & matériels nécessaires pour satisfaire les besoins informatiques des utilisateurs.

Station de travail & type de réseaux



Évolution des microprocesseurs Intel

Date	Nom	Nombre de transistors	Finesse de gravure (μm)	Fréquence de l'horloge	Largeur des données	MIPS
1971	4004	2 300		108 kHz	4 bits/4 bits bus	
1974	8080	6 000	6	2 MHz	8 bits/8 bits bus	0,64
1979	8088	29 000	3	5 MHz	16 bits/8 bits bus	0,33
1982	80286	134 000	1,5	6 à 16 MHz (20 MHz chez AMD)	16 bits/16 bits bus	1
1985	80386	275 000	1,5	16 à 40 MHz	32 bits/32 bits bus	5
1989	80486	1 200 000	1	16 à 100 MHz	32 bits/32 bits bus	20
1993	Pentium	3 100 000	0,8 à 0,28	60 à 233 MHz	32 bits/64 bits bus	100
1997	Pentium II	7 500 000	0,35 à 0,25	233 à 450 MHz	32 bits/64 bits bus	300
1999	Pentium III	9 500 000	0,25 à 0,13	450 à 1 400 MHz	32 bits/64 bits bus	510
2000	Pentium 4	42 000 000	0,18 à 0,065	1,3 à 3,8 GHz	32 bits/64 bits bus	1 700
2004	Pentium 4D « Prescott »	125 000 000	0,09 à 0,065	2,66 à 3,6 GHz	32 bits/64 bits bus	9 000
2006	Core 2™ Duo	291 000 000	0,065	2,4 GHz (E6600)	64 bits/64 bits bus	22 000
2007	Core 2™ Quad	2*291 000 000	0,065	3 GHz (Q6850)	64 bits/64 bits bus	2*22 000 (?)
2008	Core 2™ Duo (Penryn)	410 000 000	0,045	3,33 GHz (E8600)	64 bits/64 bits bus	~24 200
2008	Core 2™ Quad (Penryn)	2*410 000 000	0,045	3,2 GHz (QX9770)	64 bits/64 bits bus	~2*24 200
2008	Intel Core i7 (Nehalem)	731 000 000	0,045 (2008) 0,032 (2009)	2,66 GHz (Core i7 920) 3,33 GHz (Core i7 Ext. Ed. 975)	64 bits/64 bits bus	?
2009	Intel Core i5/i7 (Lynnfield)	774 000 000	0,045 (2009)	2,66 GHz (Core i5 750) 2,93 GHz (Core i7 870)	64 bits/64 bits bus	?
2010	Intel Core i7 (Gulftown)	1 170 000 000	0,032	3,33 GHz (Core i7 980X)	64 bits/64 bits bus	?

Utilisation des ordinateurs

Programme

Suite d'instructions dans un langage donné, définissant un traitement exécutable par un ordinateur

- programmes systèmes
- programmes d'application

Système d'exploitation

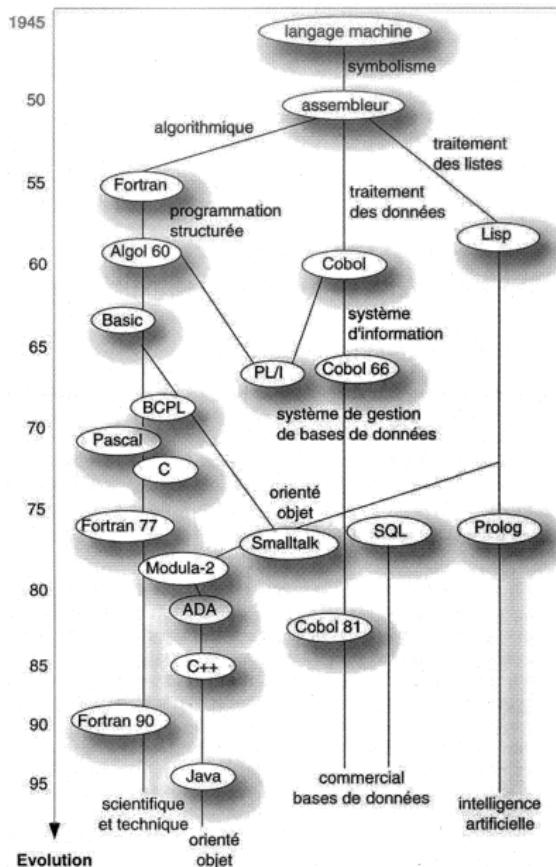
Programme système qui gère les différentes ressources de la machine

Programmation

A partir d'un problème donné, réaliser un programme dont l'exécution apporte une solution satisfaisante au problème posé

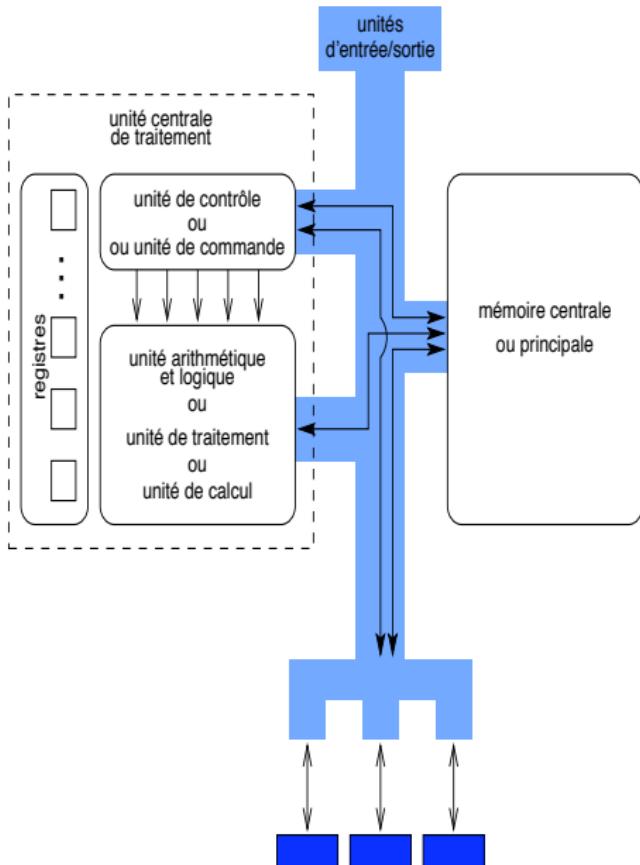
- langages de programmation (machine, assembleur, évolués)

Évolution schématique des langages



Exécution d'un programme

- ① Chargement du programme et des données depuis un périphérique dans la mémoire centrale
- ② Chargement séquentiel des instructions du programme de la mémoire centrale dans l'unité de contrôle
- ③ Analyse par l'unité de contrôle de l'instruction et passage à l'UAL pour traitement
- ④ Traitement de l'instruction par l'UAL avec éventuellement appel à la mémoire ou aux unités d'entrée-sortie.



Mémoire centrale (1/2)

- Elle peut contenir les données et des programmes
- Unité élémentaire d'information : bit (binary digit) = 0 ou 1.
- Octet = 8 bits.
- Mot mémoire = regroupement d'octets (unité d'accès de base à la mémoire et unité de base de traitement).



Comment peut-on alors coder des nombres, des caractères, des instructions avec seulement 2 caractères ?

- Utiliser plusieurs bits pour coder une information
- ⇒ Méthode de codage e.g., ASCII, ISO 8859-1, UTF-8

Mémoire centrale (2/2)

Mot-mémoire

La mémoire est constituée de cellules. Chaque cellule correspond à un *mot-mémoire*. La longueur de ce mot constitue une caractéristique importante de l'architecture d'un ordinateur :

- chaque mot possède sa propre adresse (sa position dans la mémoire)
 - c'est l'unité de base de traitement (taille des instructions)
-
- La capacité d'une mémoire s'exprime en fonction du nombre de mots-mémoire ainsi que du nombre de bits par mot. Mais en général seulement la taille en octet est compté.
 - Unité de mesure de la capacité de mémoire
 - ▶ Kilo (Ko) = 2^{10} =1024 octets
 - ▶ Méga (Mo) = 2^{20} =1048576 octets
 - ▶ Giga (Go) = 2^{30} =1073741824 octets
 - ▶ Tera (To) = 2^{40} =1099511627776 octets

Registres

Registre

Un registre est une cellule de mémoire ayant une fonction particulière. Il existe 2 types de registres en mémoire centrale :

- **registre adresse** (RA), qui contient l'adresse d'un mot mémoire
- **registre mot** (RM), qui contient le contenu d'un mot mémoire

Si la mémoire comporte 256 mots de 32 bits, le registre d'adresse doit avoir $\log_2(256) = \log_2(2^8) = 8$ bits tandis que le registre mot doit avoir 32 bits.

Ces registres sont utilisés pour exécuter les 2 opérations élémentaires en mémoire :

- **lecture** : le registre d'adresse contient l'adresse du mot à lire qui est copié dans le registre mot.
- **écriture** : le registre d'adresse contient l'adresse du mot dans lequel le contenu du registre mot va être écrit.

Unité centrale de traitement

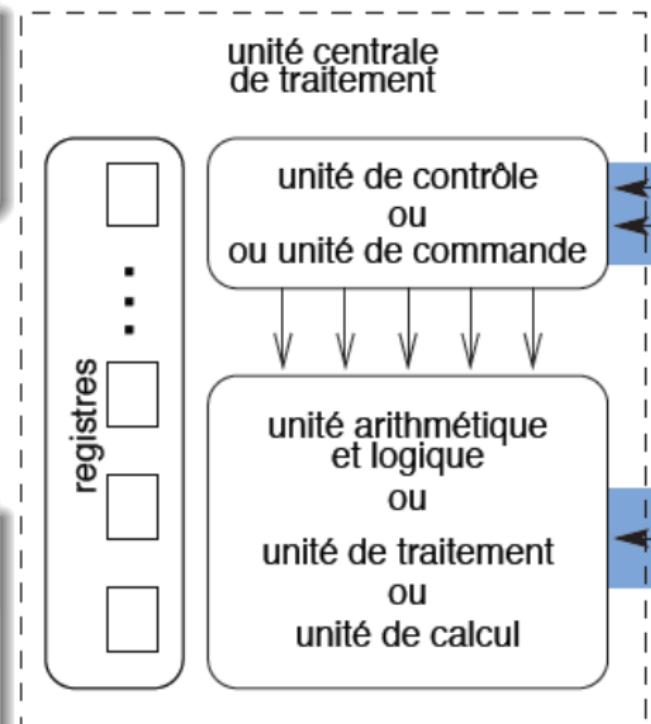
Unité de commande

Prends les instructions en mémoire, les décode et les passe à l'UAL en fonction des cycles horloges.



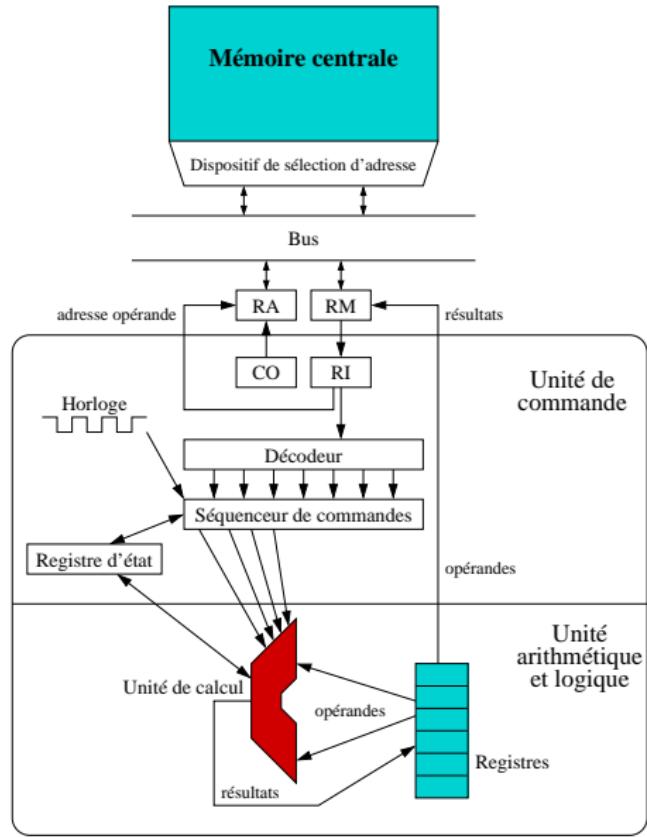
Unité Arithmétique et Logique (UAL)

Réalise effectivement les opérations arithmétiques (+, -, *, /) et logiques (NOT, AND, OR, XOR).



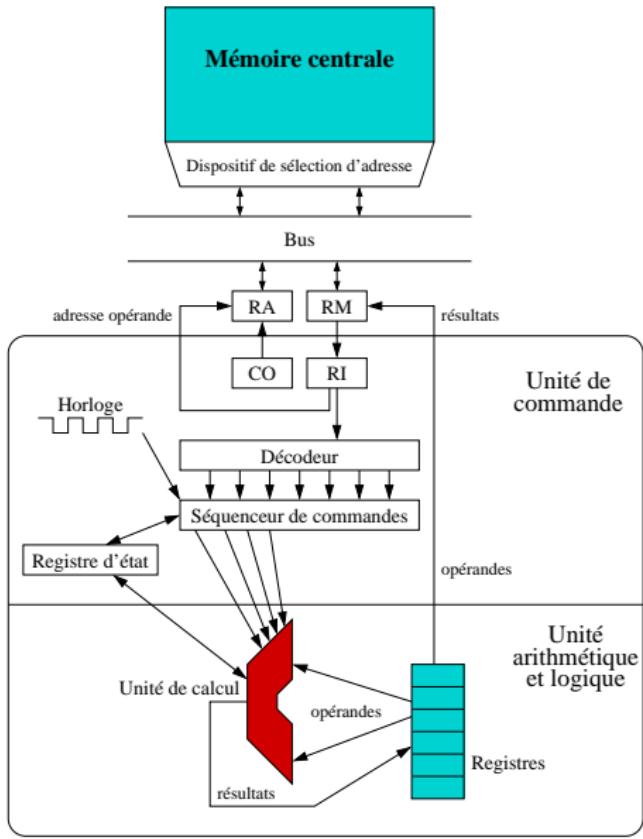
Unité de commande

- **Registre d'instruction (RI)** :
contient l'instruction (opération + opérande) en cours d'exécution
- **Compteur ordinal (CO)** :
adresse de la prochaine instruction à exécuter
- **Décodeur** : décode les instructions
- **Séquenceur** : active les circuits nécessaires de l'UAL
- **Horloge** : rythme l'enchaînement des commandes (externe à l'unité de commande)



Exécution d'une instruction

- ① Chargement de la prochaine instruction à exécuter depuis la mémoire jusqu' dans le RI.
- ② Modification du CO.
- ③ Décodage de l'instruction (opérateur).
- ④ Localisation dans la mémoire des données (opérande) utilisées par l'instruction.
- ⑤ Chargement des données dans les registres internes de l'unité centrale.
- ⑥ Exécution de l'instruction.
- ⑦ Stockage des résultats.
- ⑧ Retour à la première étape.



Entrées-sorties et périphériques

Un ordinateur a besoin d'échanger de l'information avec l'environnement extérieur. Ainsi il lui faut par exemple charger le programme et les données avec lesquels il va travailler, mais aussi communiquer avec l'utilisateur, visualiser des résultats.

Unités d'entrées-sorties

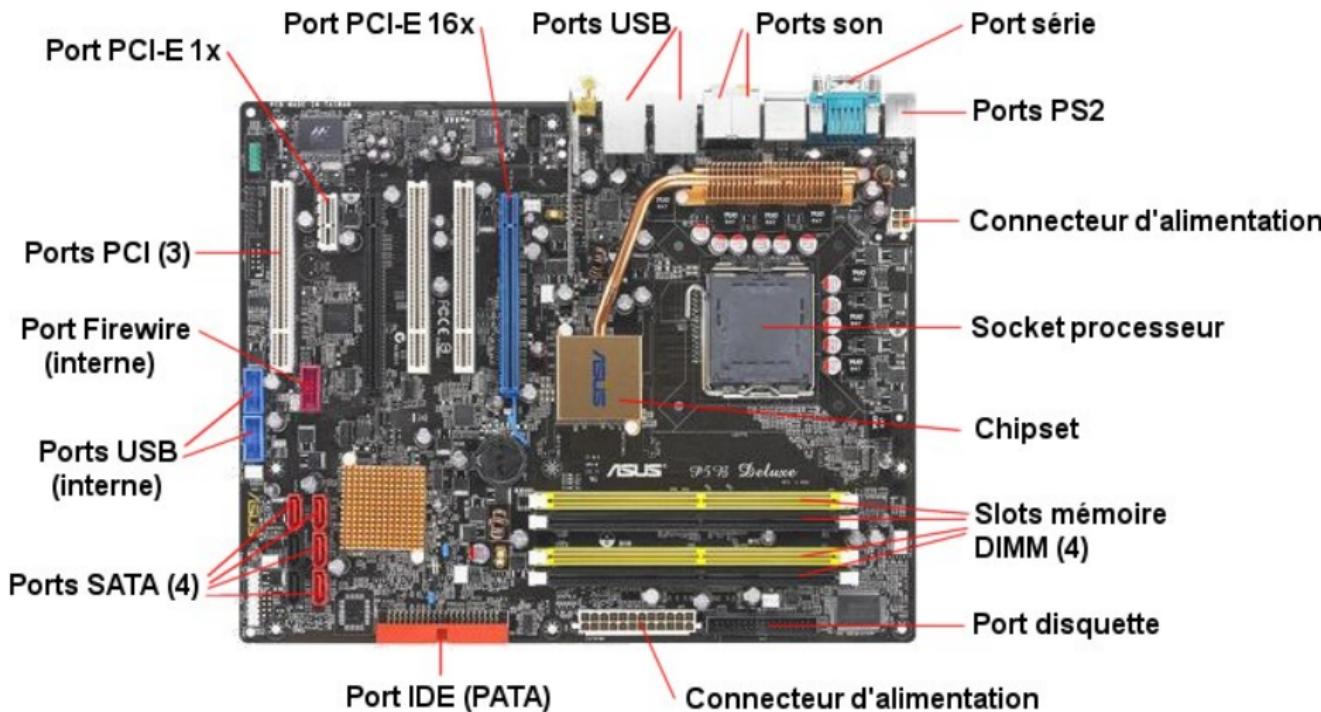
Transfèrent les informations entre l'unité centrale et les unités périphériques.

Unités périphériques

Unités d'échange de données avec le monde extérieur (écran, clavier, souris, imprimante, modem) et mémoires auxiliaires (disques) qui permettent de stocker de façon permanente. Chaque périphérique est associé à un contrôleur.

Le **bus** est l'ensemble des lignes de liaison qui assurent les communications entre les différents composants de l'ordinateur.

A quoi cela ressemble en vrai ?



2 règles à retenir

Tous les ordinateurs se composent principalement de millions voire de milliards de transistors qui ne peuvent effectuer que des opérations élémentaires très rapidement.

Un ordinateur est une machine qui doit être programmée, c'est-a-dire qu'il faut prévoir absolument tout ce qu'il doit faire et le lui expliquer précisément ! C'est le rôle des informaticiens.

vu sur YATAHONGA.com



Plan du cours

- 1 Histoire de l'ordinateur
- 2 Présentation générale
- 3 Représentation interne des informations**
- 4 Encodage de l'information
- 5 Circuits logiques
- 6 Composants électroniques
- 7 Mémoires
- 8 Unité centrale de traitement
- 9 Superordinateurs et microprocesseurs
- 10 Entrées / sorties
- 11 Assembleur
- 12 Introduction au langage MIPS

Différents types d'informations

- Instructions
- Données
 - ▶ Nombres (entiers, flottants)
 - ▶ Images
 - ▶ Vidéos
 - ▶ Sons
 - ▶ etc.



Toujours représentées sous forme binaire (0 ou 1) à l'aide de **bits**.

Codage

Fonction établissant une correspondance entre la représentation externe de l'information (e.g., A, 36, une image, un son) et sa représentation interne qui est une suite de bits (e.g., 01000001, 100100, ...).

Système binaire

Un système de numération utilisant la base 2. Toutes les informations sont codées avec des 0 et des 1.

- 1 bits : 2^1 possibilités = 0, 1
- 2 bits : 2^2 possibilités = 00, 01, 10, 11
- 3 bits : 2^3 possibilités = 000, 001, 010, 011, 100, 101, 110, 111
- n bits : 2^n possibilités
- Un mot = un ensemble de bit avec un poids $2^n, 2^{n-1} \dots 2^1, 2^0$

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	1	0	0	1	1	0	0

Avantage

- Facile à réaliser techniquement. En électronique ces 2 états corresponde à l'existence ou non d'une tension (+5V=1 et 0V=0).
- Opérations fondamentales faciles à effectuer (circuits logiques)
- Arithmétique binaire peut être réalisée à partir de la logique symbolique

- **Instructions**

- ▶ Le codage dépend du processeur
- ▶ Décodage par l'unité de commande
- ▶ Nombre limité d'instructions –
processeur CISC/RISC
(Complex/Reduced Instruction-Set
Computer)



- **Données**

- ▶ non numériques (codage assez simple car aucune opération arithmétique ou logique ne sera appliquée sur ces données, une table de correspondance suffit)
- ▶ numériques (codage complexe qui doit faciliter la mise en place de circuits réalisant les opérations arithmétiques)

Données non numériques

Afin de faciliter les échanges entre machines, des codages binaires normalisés ont été établis (BCD, ASCII, Unicode, etc.)

- Nombre variable de bits, 6, 7, 8 16, 32
- Certains bit sont réservés au "contrôle" ou à la "correction" des autres

Caractères ASCII (American Standard Code for Information Interchange)

- 7 bits (\Rightarrow 128 caractères)
- Pas de caractères accentués
- 1 bit supplémentaire utilisé pour le contrôle de parité

0 : nul	1 : soh	2 : stx	3 : etx	4 : eot	5 : enq	6 : ack	7 : bel
8 : bs	9 : ht	10 : nl	11 : vt	12 : np	13 : cr	14 : so	15 : si
16 : dle	17 : dc1	18 : dc2	19 : dc3	20 : dc4	21 : nak	22 : syn	23 : etb
24 : can	25 : em	26 : sub	27 : esc	28 : fs	29 : gs	30 : rs	31 : us
32 : sp	33 : !	34 : "	35 : #	36 : \$	37 : %	38 : &	39 : '
40 : (41 :)	42 : *	43 : +	44 : ,	45 : -	46 : .	47 : /
48 : 0	49 : 1	50 : 2	51 : 3	52 : 4	53 : 5	54 : 6	55 : 7
56 : 8	57 : 9	58 : :	59 : ;	60 : <	61 : =	62 : >	63 : ?
64 : @	65 : A	66 : B	67 : C	68 : D	69 : E	70 : F	71 : G
72 : H	73 : I	74 : J	75 : K	76 : L	77 : M	78 : N	79 : O
80 : P	81 : Q	82 : R	83 : S	84 : T	85 : U	86 : V	87 : W
88 : X	89 : Y	90 : Z	91 : [92 : \	93 :]	94 : ^	95 : _
96 : `	97 : a	98 : b	99 : c	100 : d	101 : e	102 : f	103 : g
104 : h	105 : i	106 : j	107 : k	108 : l	109 : m	110 : n	111 : o
112 : p	113 : q	114 : r	115 : s	116 : t	117 : u	118 : v	119 : w
120 : x	121 : y	122 : z	123 : {	124 :	125 : }	126 : `	127 : del

Caractères Unicode

- 1 à 4 octets (\Rightarrow 1114112 caractères)
- Codage unique quelque soit la plateforme, le logiciel, la langue.
- Normalisé ISO/IEC 10646 (UTF-8, UTF-16, UTF-32)

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	NUL 0000	STX 0001	SOT 0002	ETX 0003	EOT 0004	ENQ 0005	ACK 0006	BEL 0007	BS 0008	HT 0009	LF 000A	VT 000B	FF 000C	CR 000D	SO 000E	SI 000F
10	DLE 0010	DC1 0011	DC2 0012	DC3 0013	DC4 0014	NAK 0015	SYN 0016	ETB 0017	CAN 0018	EM 0019	SUB 001A	ESC 001B	FS 001C	GS 001D	RS 001E	US 001F
20	SP 0020	! 0021	" 0022	# 0023	\$ 0024	% 0025	& 0026	' 0027	(0028) 0029	* 002A	+ 002B	,002C	- 002D	.002E	/
30	Ø 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	:	;	< 003C	= 003D	> 003E	? 003F
40	ø 0040	À 0041	Á 0042	Ç 0043	Ð 0044	È 0045	Í 0046	Ó 0047	Ù 0048	Ñ 0049	Ї 004A	Ќ 004B	Ӆ 004C	Ӎ 004D	Ӎ 004E	Ӯ 004F
50	܂ 0050	܃ 0051	܄ 0052	܅ 0053	܆ 0054	܇ 0055	܈ 0056	܉ 0057	܊ 0058	܋ 0059	܌ 005A	܍ 005B	܎ 005C	܏ 005D	ܐ 005E	ܑ 005F
60	܂ 0060	܃ 0061	܄ 0062	܅ 0063	܆ 0064	܇ 0065	܈ 0066	܉ 0067	܊ 0068	܋ 0069	܌ 006A	܍ 006B	܎ 006C	܏ 006D	ܐ 006E	ܑ 006F
70	܂ 0070	܃ 0071	܄ 0072	܅ 0073	܆ 0074	܇ 0075	܈ 0076	܉ 0077	܊ 0078	܋ 0079	܌ 007A	܍ 007B	܎ 007C	܏ 007D	ܐ 007E	ܑ 007F
80	܂ 20AC	܃ 201A	܄ 201E	܅ 201F	܆ 2026	܇ 2020	܈ 2021	܉ 2030	܊ 2033	܋ 203A	܌ 204A	܍ 205A	܎ 206A	܏ 207A	ܐ 208A	ܑ 209A
90	܂ 2018	܃ 2019	܄ 201C	܅ 201D	܆ 2022	܇ 2023	܈ 2014	܉ 2122	܊ 203A	܋ 204A	܌ 205A	܍ 206A	܎ 207A	܏ 208A	ܐ 209A	ܑ 209B
A0	܂ 00A0	܃ 00A2	܄ 00A3	܅ 00A4	܆ 00A6	܇ 00A7	܈ 00D8	܉ 00A3	܊ 0156	܋ 00AB	܌ 00AC	܍ 00AD	܎ 00AE	܏ 00C6	ܐ 00C7	
B0	܂ 00B0	܃ 00B1	܄ 00B2	܅ 00B3	܆ 00B4	܇ 00B5	܈ 00B6	܉ 00B7	܊ 00F8	܋ 00B3	܌ 0157	܍ 00BB	܎ 00BC	܏ 00BD	ܐ 00BE	ܑ 00E6
C0	܂ 0104	܃ 012E	܄ 0100	܅ 0106	܆ 00C4	܇ 00C5	܈ 0116	܉ 0112	܊ 00C9	܋ 0179	܌ 0116	܍ 0122	܎ 0136	܏ 012A	ܐ 013B	
D0	܂ 0160	܃ 0143	܄ 0145	܅ 00D3	܆ 014C	܇ 00D5	܈ 00D6	܉ 00D7	܊ 0141	܋ 015A	܌ 016A	܍ 00DC	܎ 011B	܏ 011D	ܐ 00DF	
E0	܂ 0105	܃ 012F	܄ 0101	܅ 0107	܆ 00E4	܇ 00E5	܈ 0113	܉ 010D	܊ 00E3	܋ 017A	܌ 0117	܍ 0123	܎ 0137	܏ 012B	ܐ 013C	
F0	܂ 0161	܃ 0144	܄ 0146	܅ 00F3	܆ 014D	܇ 00F5	܈ 00F6	܉ 00F7	܊ 0142	܋ 015B	܌ 016B	܍ 00FC	܎ 011C	܏ 011E	ܐ 02D3	

Données numériques

- Nombres entiers positifs ou nuls : 0 ; 1 ; 315
- Nombres entiers négatifs : -1 ; -1255
- Nombres fractionnaires : 3,1415 ; -0,5

Un algorithme de codage réalise la conversion en binaire. Les opérations arithmétiques (+, -, *, /) sont effectuées en arithmétique binaire.

Entiers positifs ou nuls

Tout nombre entier positif (de n chiffres a_i) peut être représenté, en base b , par une expression de la forme :

$$x = a_{n-1} * b^{n-1} + a_{n-2} * b^{n-2} + \dots + a_1 * b^1 + a_0 * b^0$$

335 en base 10

$$335 = 3 * 10^2 + 3 * 10^1 + 5 * 10^0$$

$$\Rightarrow 335$$

335 en base 2

$$335 = 1 * 2^8 + 0 * 2^7 + 1 * 2^6 + 0 * 2^5 + 0 * 2^4 + 1 * 2^3 + 1 * 2^2 + 1 * 2^1 + 1 * 2^0$$

$$\Rightarrow 101001111$$

Changement de base : binaire → décimal

Additionner les puissances de 2 correspondants aux bits de valeur 1.

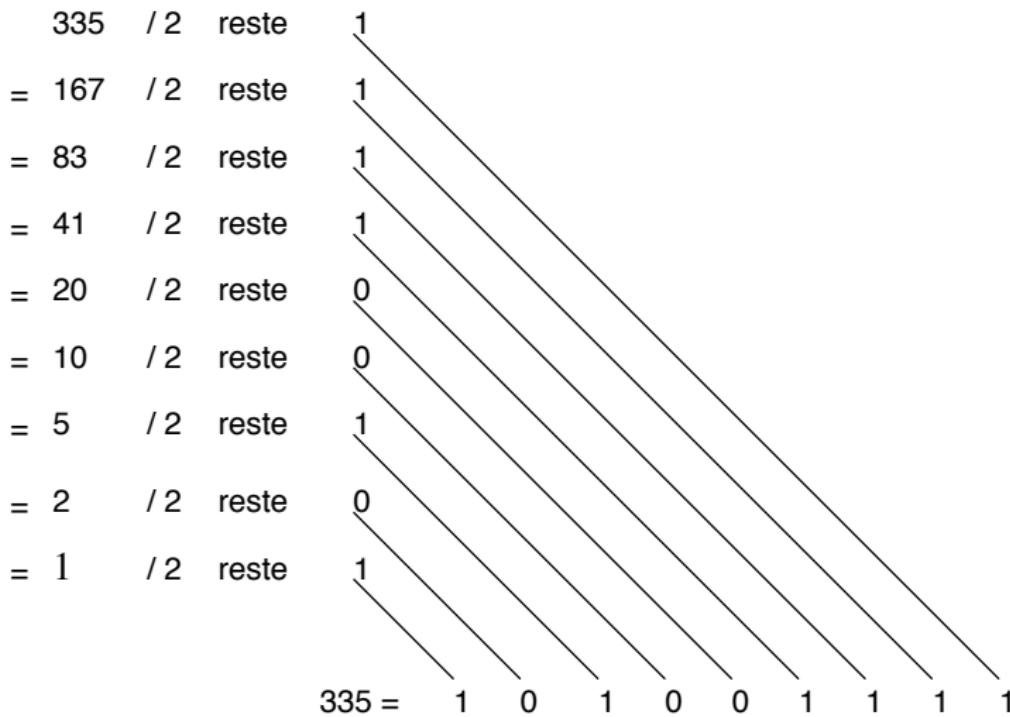
$$101001111 = 2^8 + 2^6 + 2^3 + 2^2 + 2^1 + 2^0 = 256 + 64 + 8 + 4 + 2 + 1 = 335$$

Attention au poids !

- **bit de poids faible** : le bit ayant la moindre valeur (i.e., celui de droite)
- **bit de poids fort** : le bit ayant la plus grande valeur (i.e., celui de gauche)

Changement de base : décimal → binaire

Quotient



Correspondance entre les systèmes les plus utilisés

décimal	binaire	octal	hexadécimal
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10

Addition de nombres binaires

222+17 en base 2

$$\begin{array}{r} & 1 & 1 & ^10 & 1 & 1 & 1 & 1 & 0 \\ + & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ \hline & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \end{array}$$

=> 239

222+199 en base 2

$$\begin{array}{r} & 1 & ^11 & 1 & ^10 & ^11 & ^11 & ^11 & 1 & 0 \\ + & & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ \hline & (1) & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{array}$$

=> 421

=> Attention au bit de retenue (*Carry*)

Soustraction de nombres binaires

8-1 en base 2

$$\begin{array}{r} 1_0 \quad 0_1 \quad 0_1 \quad 1_0 \\ - \quad 0 \quad 0 \quad 0 \quad 1 \\ \hline 0 \quad 1 \quad 1 \quad 1 \end{array}$$

=> 7

13-6 en base 2

$$\begin{array}{r} 1_0 \quad 11_0 \quad 1_0 \quad 1 \\ - \quad 0 \quad 1 \quad 1 \quad 0 \\ \hline 0 \quad 1 \quad 1 \quad 1 \end{array}$$

=> 7

Multiplication et division de nombres binaires

7*5 en base 2

$$\begin{array}{r} & & 1 & 1 & 1 \\ * & & 1 & 0 & 1 \\ \hline & & 1 & 1 & 1 \\ 1 & 1 & 10 & 0 & 0 & 0 \\ & 1 & 1 & 1 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 1 & 1 \end{array}$$

=> 35

356/4 en base 2

$$\begin{array}{r} 1 0 1 1 0 0 1 0 0 \\ / \qquad \qquad \qquad 1 0 0 \\ \hline 1 0 1 1 0 0 1 \end{array}$$

=> 89

Champ fixe

L'entier maximal pouvant être codé dépendra du nombre de bits que l'on réserve pour coder un nombre. En général les entiers sont codés sur un mot e.g., pour un ordinateur 32 bits : $2^{32} - 1 = 4\,294\,967\,295$.

Dépassement de capacité

Lorsque par exemple le résultat d'une opération sur des nombres produit un nombre plus grand que la taille du mot prévu pour représenter ces nombres (e.g., bit de retenue).

Entiers négatifs

Plusieurs façons de représenter un nombre négatif :

- Valeur absolue signée
- Complément à 1 (ou logique)
- Complément à 2 (ou arithmétique)

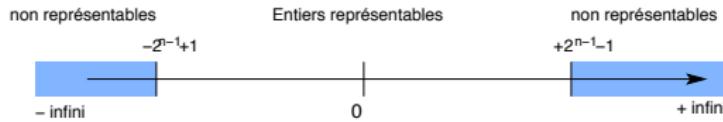
Avantages & inconvénients ?



Valeur absolue signée

Les nombres sont codés comme des entiers positifs mais on sacrifie un bit (celui de poids fort) pour coder le signe :

- Bit $n - 1$ pour le signe (signe $+ = 0$, signe $- = 1$)
- Bits $n - 2 \dots 0$ pour la valeur absolue
- e.g., $6 = 0110 / -6 = 1110$



Avantage

- Symétrie : autant de négatifs que de positifs

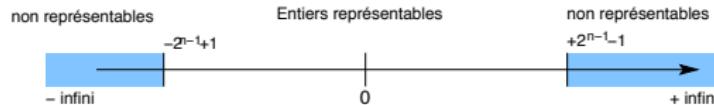
Inconvénient

- 2 représentations du 0 : 0000000 et 10000000
- Bit de signe doit être traité de façon particulière => opérations arithmétiques non aisées

Complément à 1

Les nombres positifs sont codés comme précédemment, les négatifs sont obtenus en remplaçant tous les bits à 1 par 0 et vice-versa.

- Bit $n - 1$ pour le signe (signe $+ = 0$, signe $- = 1$)
- Bits $n - 2 \dots 0$ pour les positifs et leurs compléments
- $6 = 0110 / -6 = 1001$



Avantage

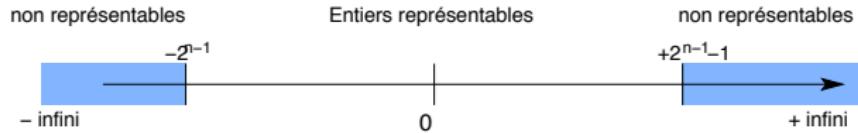
- Symétrie : autant de négatifs que de positifs
- Une soustraction se réduit à l'addition de son complément

Inconvénient

- 2 représentations du 0 : 0000000 et 1111111
- Bit de retenue à reporter lors de l'addition

Complément à 2

- On ajoute 1 au complément à 1.
- $6 = 0110 / -6 = 1001 + 1 = 1010$



Avantage

- Une seule représentation du zéro
- Une soustraction se réduit à l'addition de son complément
- Pas de report du bit de retenue pour l'addition

Inconvénient

- Dissymétrie : plus de négatifs que de positifs

Exemple : représentation d'entiers signés sur 16 bits

16 bits $\Rightarrow 2^{16} = 65\,536 = 2 \times 32\,768$ valeurs possibles

décimal	valeur absolue et signe	complément à 2	complément à 1
+32767	0111...1...1111	0111...1...1111	0111...1...1111
+32766	0111...1...1110	0111...1...1110	0111...1...1110
...
+1	0000...0...0001	0000...0...0001	0000...0...0001
+0	0000...0...0000	0000...0...0000	0000...0...0000
-0	1000...0...0000	-----	1111...1...1111
-1	1000...0...0001	1111...1...1111	1111...1...1110
...
-32766	1111...1...1110	1000...0...0010	1000...0...0001
-32767	1111...1...1111	1000...0...0001	1000...0...0000
-32768	-----	1000...0...0000	-----

Exemple : soustraction de nombres sur 4 bits

décimal	signe+val. absolue	comp. à 1	comp. à 2
+7	0111	0111	0111
-6	+1110	+1001	+1010
—	—	—	—
+1	?101	10000 ↓ 1	10001 ↓ 0001
		0001	
	(a)	(b)	(c)

- (a) est la plus facile à lire, mais le bit de signe doit être traité à part ;
- (b) on effectue l'addition du complément, y compris le bit de signe, avec report de la retenue ;
- (c) on effectue une addition, y compris le bit de signe, mais sans report de la retenue.

Opérations arithmétiques avec les nombres signés en complément à 2 (1/4)

Addition

4 cas sont possibles :

- **Les deux nombres sont positifs** : addition binaire classique.
- **Le nombre positif est plus grand que le nombre négatif** : addition binaire classique et on 'oublie' la dernière retenue (à gauche). La somme est positive.
- **Le nombre négatif est plus grand que le nombre positif** : addition binaire classique, la somme est négative et représentée directement dans le système complément à 2.
- **Les deux nombres sont négatifs** : addition binaire classique et on oublie la dernière retenue (à gauche). La somme est négative.

Opérations arithmétiques avec les nombres signés en complément à 2 (2/4)

Soustraction

La soustraction est considérée comme un cas particulier de l'addition :

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

On prend donc le système complément à deux pour représenter les négatifs, et on effectue une addition.

Opérations arithmétiques avec les nombres signés en complément à 2 (3/4)

Multiplication

Les deux nombres doivent être représentés dans une forme sans complément (i.e., valeur absolue). On effectue la multiplication et on décide du signe du résultat :

- opérandes sont de mêmes signes : le résultat est positif
- opérandes signes différents : le résultat est négatifs, on le représente avec son complément à 2

Opérations arithmétiques avec les nombres signés en complément à 2 (3/4)

Division

Les deux nombres doivent être représentés dans une forme sans complément (i.e., valeur absolue) :

- ① Déterminer si le dividende et le diviseur sont de mêmes signes ou de signes différents. Ceci va déterminer le signe du quotient ; Initialiser le quotient à zéro.
- ② Soustraire le diviseur du dividende en utilisant l'addition avec complément à deux pour obtenir le premier reste partiel ; Incrémenter le quotient de 1. Si le reste partiel est positif aller à l'étape trois. Si le reste partiel est zéro ou négatif la division est terminée.
- ③ Soustraire le diviseur du reste partiel et incrémenter le quotient de 1. Si le résultat est positif, répéter l'opération pour le reste partiel trouvé. Si le résultat est zéro ou négatif la division est terminée.

Nombres fractionnaires

Représentation de la virgule :

- **virgule fixe** : nombre fixe de chiffres après la virgule ;
Les bits à gauche (resp. à droite) de la virgule représentent la partie entière (resp. la partie décimale "binaire") du nombre et correspondent à des puissances de 2 (resp. l'inverse des puissances de 2).
- **virgule flottante** : la position de la virgule n'est pas fixe. Ces nombres sont des approximations de nombres réels.

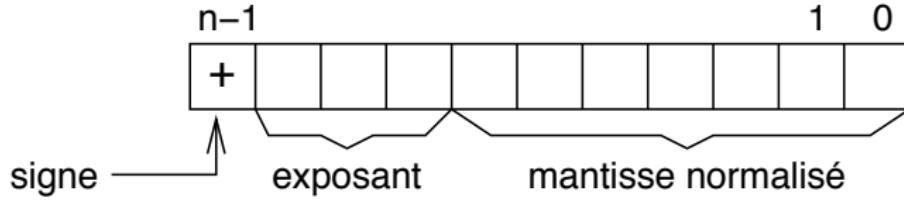
Représentation en virgule flottante

Représentation d'un nombre sous la forme d'un produit de 2 facteurs

$$N = M \times B^E$$

- B : base
 - M : mantisse (nombre purement fractionnaire i.e., 0,xxx)
 - E : exposant
- e.g., $0,23643 \times 10^3 = 236,43$
 - exposant et mantisse peuvent être signés

La représentation classique consiste à coder l'exposant en représentation *biaisé* (ou en *excèdent*) sur k bits et la mantisse en valeur absolue signée sur $(n - k - 1)$ bits.



Codage de l'exposant

- Taille de l'exposant bornée.
- Codage par excédent n : on décale l'exposant on lui ajoutant n .
=> Pas d'exposant négatifs
=> Facilite les opérations de tri (pas besoin de conversion au décimal pour trier)
- e.g., 3 bits : 111(+3), 110(+2), 101(+1), 100(0), 011(-1), 010(-2), 001(-3), 000(-4)
- Si la taille de l'exposant augmente alors l'intervalle des valeurs possibles représentables grandit.

Codage de la mantisse

- Taille de la mantisse bornée.
- Changement de base (décimal → binaire) obtenu par multiplications successives par 2.
- Si cela ne converge pas vers 1 alors il n'y a pas de représentation exacte de ce nombre, on tronque alors suivant la taille de la mantisse.
- Si la taille de la mantisse augmente, la précision des valeurs possibles représentables grandit.

Conversion de 0.375

$$0.375 \times 2 = 0.75 ; 0.75 \times 2 = 1.5 ; 0.5 \times 2 = 1.0$$

$0.375 \rightarrow 0.011$ en binaire ($0.25 + 0.125$)

soit 0.11×2^{-1} en forme normalisée

Opérations arithmétiques en virgule flottante

- Il est toujours possible de revenir à des opérations arithmétiques sur les nombres entiers :
 - ▶ **multiplication** : additionner les exposants, multiplier les mantisses et re-normaliser le résultat
 - ▶ **division** : soustraire les exposants, diviser les mantisses et re-normaliser le résultat
 - ▶ **addition** : de-normaliser la plus petite valeur d'exposant, additionner les mantisses, re-normaliser le résultat
 - ▶ **soustraction** : de-normaliser la plus petite valeur d'exposant, soustraire les mantisses, re-normaliser le résultat
- Il peut être nécessaire d'arrondir la mantisse (i.e., perte de précision).
- Dépassement de capacité et sous-passement de capacité peuvent se produire si l'exposant devient trop grand ou trop petit.

Norme IEEE 754

Objectif : harmoniser les représentations en virgule flottante et définir le comportement en cas d'exception (dépassement, sous-passement)

- Bit de signe S (signe $+ = 0$, signe $- = 1$)
- Mantisse normalisée en base 2 avec un bit caché
- Exposant codé en excédent $2^{n-1} - 1$ ($n =$ nombre de bit de l'exposant)

Définition

$$x = (-1)^S \times (1, M) \times 2^{E-(2^{n-1}-1)}$$

Valeurs particulières :

- $E=0$ et $M=0 \Rightarrow +0$
- $E=2^n - 1$ et $M=0 \Rightarrow +\infty$
- $E=2^n - 1$ et $M \neq 0 \Rightarrow \text{NaN (Not a Number)}$

Précision des flottants (norme IEEE 754)

	simple précision 32 bits	double précision 64 bits
bit de signe	1	1
exposant	8	11
mantisso	23	52
codage de l'exposant	excédent 127	excédent 1023
variation de l'exp.	-126 à +127	-1022 à +1023
plus petit nombre	2^{-126}	2^{-1022}
plus grand nombre	environ 2^{+128}	environ 2^{+1024}
échelle des nombres décimaux	environ 10^{-38} à 10^{+38}	environ 10^{-308} à 10^{+308}

Exemple

-0,75 en simple précision

$$-0,75 \times 2 = 1,5 ; 0,5 \times 2 = 1,0$$

$$-0,75 \rightarrow 0,11 \text{ en binaire } (0,50 + 0,25)$$

soit $1,1 \times 2^{-1}$ en forme normalisée

$$\Rightarrow (-1)^S \times (1+, 1000\ 0000\ 0000\ 0000\ 0000) \times 2^{126-127}$$

En simple précision :

1 01111110 10000000000000000000000000000000

1 10000001 01000000000000000000000000000000 en simple précision

signe = 1

exposant = 129

mantisso = $1 \times 2^{-2} = 1/4 = 0.25$

$$\Rightarrow (-1)^S \times (1 + .mantisso) \times 2^{E-127} = (-1)^1 \times (1 + 0.25) \times 2^{129-127}$$

$$= -1 \times 1.25 \times 2^2$$

$$= -5$$

Autres représentations binaires

- **Décimaux codés binaire (BCD)** : Chaque chiffre d'un nombre est codé individuellement en son équivalent binaire sur 4 bits.
- e.g., $15 = 0001'0101$
- **Code excès 3** : BCD+3 à chaque chiffre
- **Code 2 dans 5**
- **Code biquinaire**

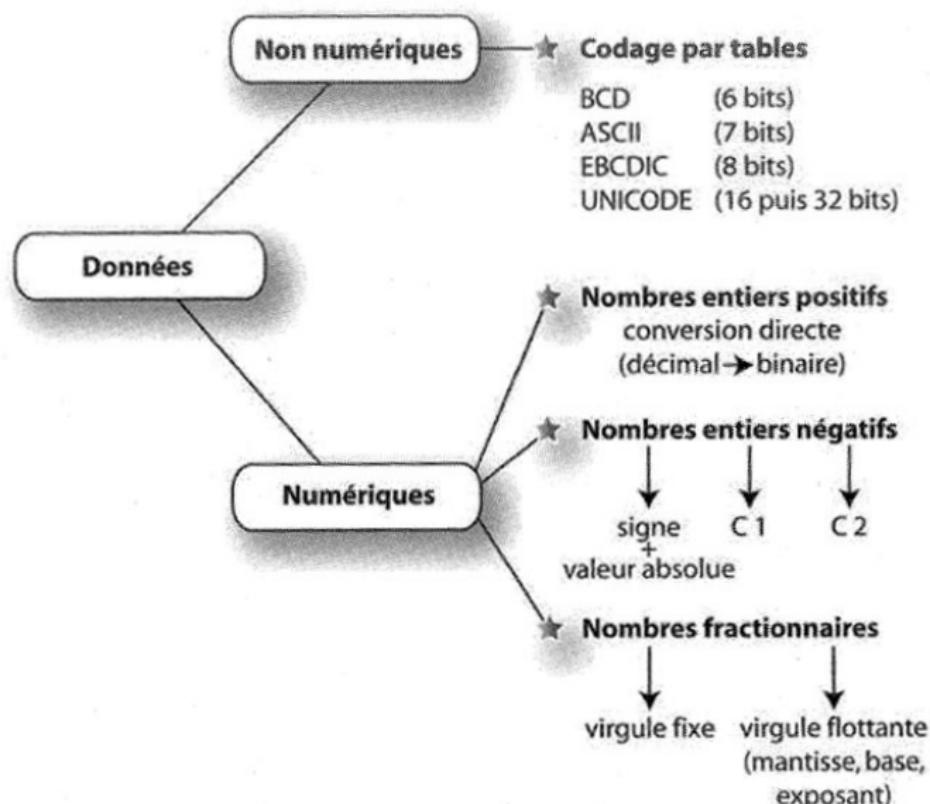
Avantage

- Opérations d'entrées / sorties plus faciles

Inconvénient

- Opérations arithmétiques compliquées

Récapitulatif des différentes représentations relatives aux données



Représentation des primitives d'un langage (e.g., Java)

Primitive	Signification	Taille (en octets)	Plage de valeurs acceptée
char	Caractère	2	valeur du jeu de caractères Unicode (65000 caractères possibles)
byte	Entier très court	1	-128 à 127
short	Entier court	2	-32768 à 32767
int	Entier	4	-2 147 483 648 à 2 147 483 647
long	Entier long	8	-9223372036854775808 à 9223372036854775807
float	flottant (réel)	4	-1.4*10 ⁻⁴⁵ à 3.4*10 ³⁸
double	flottant double	8	4.9*10 ⁻³²⁴ à 1.7*10 ³⁰⁸
boolean	booléen	1	0 ou 1 (en réalité, toute autre valeur que 0 est considérée égale à 1)

Plan du cours

- 1 Histoire de l'ordinateur
- 2 Présentation générale
- 3 Représentation interne des informations
- 4 **Encodage de l'information**
- 5 Circuits logiques
- 6 Composants électroniques
- 7 Mémoires
- 8 Unité centrale de traitement
- 9 Superordinateurs et microprocesseurs
- 10 Entrées / sorties
- 11 Assembleur
- 12 Introduction au langage MIPS

Encodage de l'information

Objectif

Utilisation des codes pour représenter l'information afin de résoudre 3 types de problèmes :



- Assurer l'**intégrité** de l'information
(détection et correction d'erreurs)



- Minimiser la **taille** de l'information (compression),



- Garantir la **sécurité** de l'information (encryptage/chiffrement).

Nous allons voir quelques exemples de codage.

Codes détecteurs et correcteurs d'erreurs

Une information peut subir des modifications involontaires lors de sa transmission ou lors de son stockage en mémoire.

⇒ Utiliser des codes permettant de **détecter** ou même de **corriger** les erreurs.

⇒ Utiliser des bits supplémentaires (de contrôle) à ceux nécessaires pour coder l'information.

- Codes autovérificateurs (e.g., contrôle de parité),
- Codes autocorrecteurs (e.g., double parité, hamming, codes polynomiaux).

Contrôle de parité

- Code autovérificateur le plus simple.
- A un mot de taille m , on ajoute 1 bit de parité.
- "parité paire" : la valeur du bit de parité est à 1 si le nombre de bit à 1 du mot $m+1$ est pair.

Exemple en parité paire : 7+1 bits

Valide :

1	1	0	0	1	1	0	0
0	1	0	0	1	1	0	1

Erreur :

0	1	0	0	1	1	0	0
---	---	---	---	---	---	---	---

Si un bit est changé par erreur la parité n'est plus vérifiée. L'erreur est détectée (mais pas corrigée).

⇒ Retransmission de l'information.

Contrôle de double parité

- Code obtenu en effectuant un double contrôle de parité.
- A un mot de taille m , on ajoute 1 bit de parité transversal.
- Après une série de mot, on ajoute 1 mot de parité (longitudinal).

Exemple en double parité impaire paire : 7+1 bits et série de 4 mots.

	No de bit							bit de	contrôle
	1	2	3	4	5	6	7	parité	transversal
1. car.= 1	0	1	1	1	0	0	1	0	← faux
2. " = 9	0	1	1	1	0	0	1	1	OK
3. " = 6	0	1	1	0	1	1	0	1	OK
4. " = 8	0	1	1	1	0	0	0	0	OK

bit de	parité	1	1	1	1	0	0	1	1
contrôle longitudinal		OK	OK	OK	↑	OK	OK	OK	faux

Si un bit est changé, l'erreur est détectée et corrigée.

Objectif

Diminuer le nombre de bits utilisés pour le stockage et la transmission des informations. Les algorithmes de compression se caractérisent par les facteurs suivants :

- le taux de compression,
- la qualité de compression (avec ou sans perte d'information),
- le temps de compression.

Codage de Huffman

- Algorithme de compression sans perte, très connu.
- Réduit le nombre de bits utilisés pour représenter les caractères les plus fréquent,
- Augmente le nombre de bits utilisés pour représenter les caractères peu fréquent,
- Un arbre binaire donne le codage pour chaque caractère.

Exemple

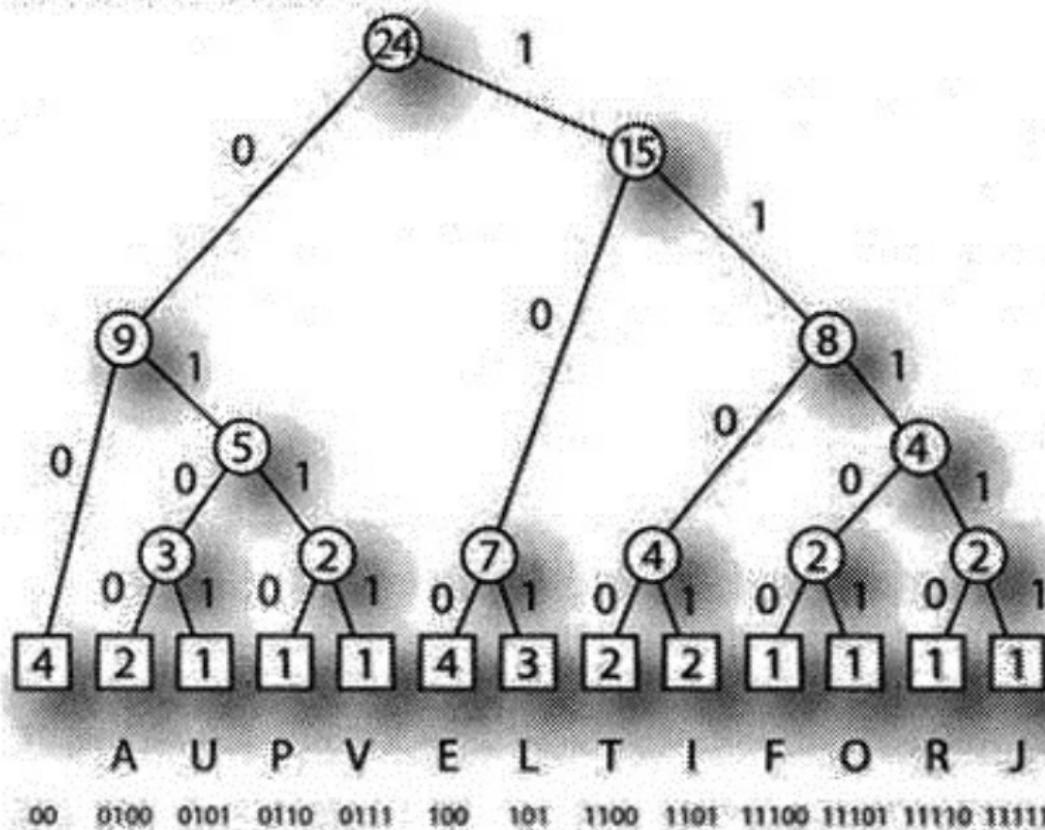
Phrase

La jolie petite fleur va

Fréquence des lettres

E	L	A	T	I	P	R	U	F	O	V	J
4	4	3	2	2	1	1	1	1	1	1	1

Huffman : Construction de l'arbre binaire



Autres algorithmes de compression

Lempel-Ziv-Welch (LZW)

Réversible. Utilise des caractères additionnels pour représenter des chaînes de caractères récurrentes. Similaire à Huffman mais pour des "mots" et sans lecture préalable du message complet.

Run Length Encoding (RLE)

Réversible. Supprime les séquences de caractères identiques et on les remplace par un caractère spécial.

Joint Photographic Expert Group (JPEG)

Irréversible. Image divisée en carré puis représentation de chaque carré par une somme de fonctions cosinus, de fréquence et d'amplitudes appropriées.

Ondelettes

Irréversible. Diviser un signal en ondelettes (sous-signaux) ne contenant que les détails à une échelle donnée.

Encryptage

La cryptologie existe depuis plus de 2000 ans ! \Rightarrow Transformer un message sous une forme secrète impossible à déchiffrer sans une clé.

- cryptographie : concevoir des algorithmes d'encryptage,
- cryptoanalyse : casser ces algorithmes.

Exemples

- Data Encryption Standard (DES) - système à clé privée développé dans les années 70.
- Rivest, Shamir, Adelman (RSA) - système à clé publique basé sur la réduction de nombres premiers. Le plus utilisé.
- Pretty Good Privacy (PGP) - système mixte.