

Introduction à la Programmation

Benoit Donnet
Année Académique 2023 - 2024



Agenda

- Introduction
- Chapitre 1: Bloc, Variable, Instruction Simple
- Chapitre 2: Structures de Contrôle
- Chapitre 3: Méthodologie de Développement
- Chapitre 4: Introduction à la Complexité
- Chapitre 5: Structures de Données
- Chapitre 6: Modularité du Code
- Chapitre 7: Pointeurs
- Chapitre 8: Allocation Dynamique

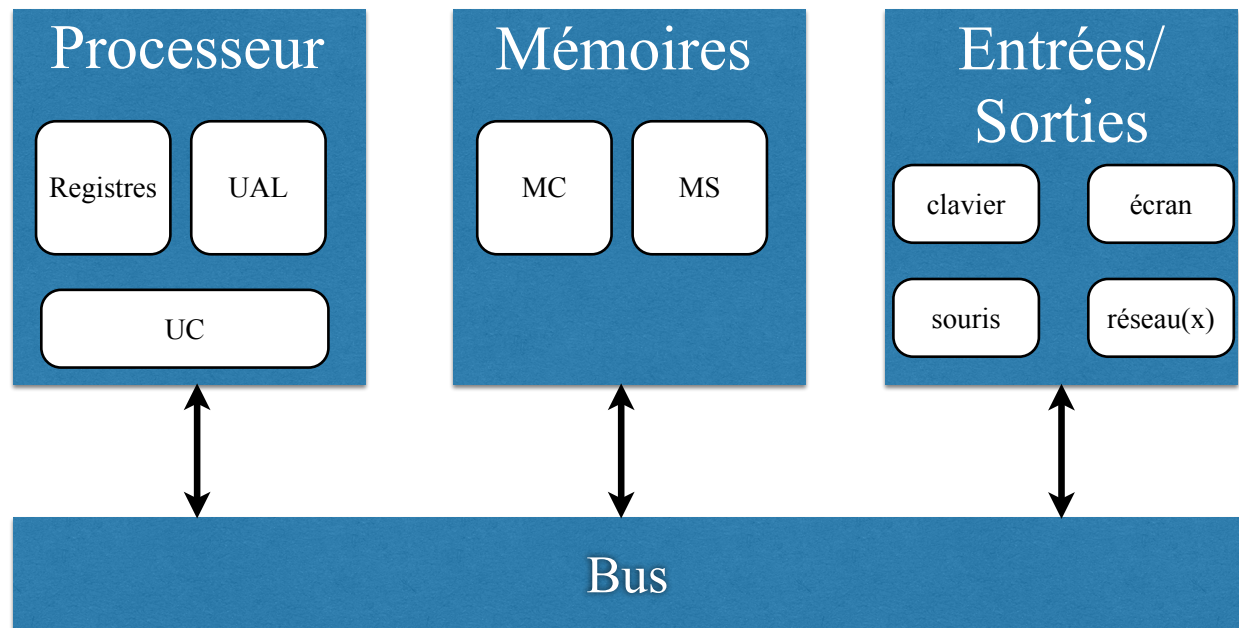
Agenda

- Introduction
 - Organisation d'un Ordinateur
 - Système d'Exploitation
 - Algorithme
 - Codage des Entiers
 - Programme

Agenda

- Introduction
 - Organisation d'un Ordinateur
 - ✓ Schéma Général
 - ✓ Processeur
 - ✓ Mémoire
 - ✓ Entrées/Sorties
 - Système d'Exploitation
 - Algorithme
 - Codage des Entiers
 - Programme

Schéma Général



Processeur

- **Central Processing Unit (CPU)**
- Composé de 3 parties
 - Unité Centrale (UC)
 - Registres
 - Unité Arithmétique et Logique (UAL)



© Intel

Processeur (2)

- **Unité Centrale**

- transfert d'information depuis la mémoire
- décodage et exécution des instructions

- **Registres**

- mémoires très rapides situées dans l'UC
- 3 types de registres:
 - ✓ *registres données*
 - représentent une valeur
 - ✓ *registres adresses*
 - désignent un élément de la mémoire
 - ✓ *registres d'état*
 - représentent une partie de l'état du processeur

Processeur (3)

- **Unité Arithmétique et Logique**

- décodage des fonctions
- opère sur les registres
- opérations arithmétiques de base
 - ✓ +, -, ×, /
- opérations logiques
 - ✓ &, |, >>, <<

- **Performances d'un CPU?**

- liées à la fréquence d'horloge et à la nature du CPU
- mesurées en MIPS
 - ✓ millions of instructions per second

Mémoire

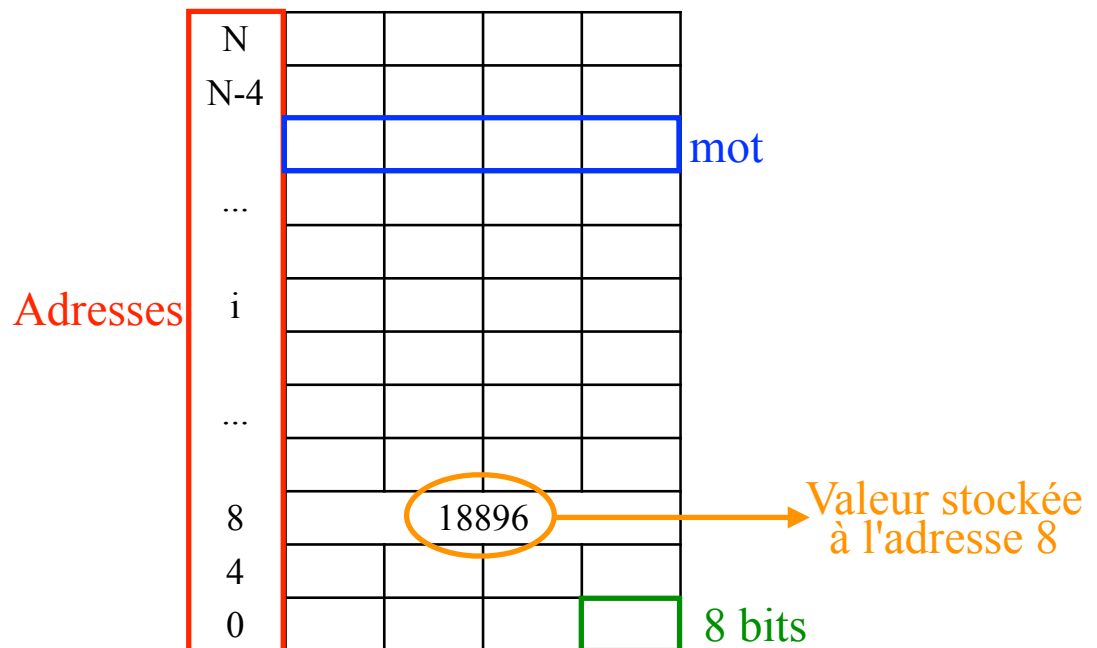
- Il existe deux types de mémoire
 - *mémoire vive*
 - ✓ aka mémoire centrale
 - ✓ RAM
 - *mémoire morte*
 - ✓ aka mémoire secondaire
 - ✓ ROM

Mémoire (2)

- Mémoire Centrale
 - cases numérotées
 - ✓ taille standard d'une case: 8 bits
 - ✓ toute valeur écrite dans une case persiste tant qu'elle n'est pas modifiée et que la machine fonctionne
 - ✓ adresse: numéro de la case
 - ✓ **mot**
 - plus petite unité de mémoire manipulable par l'ordinateur
 - contenu
 - ✓ instructions du programme
 - ✓ données du programme

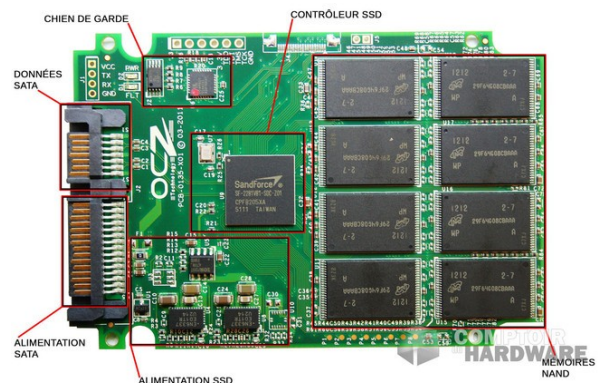
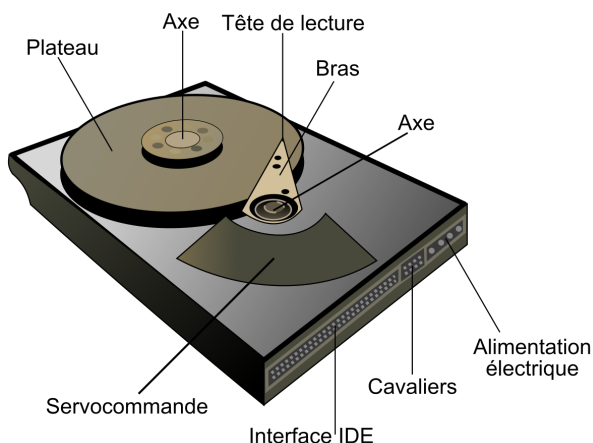
Mémoire (3)

Mémoire Centrale



Mémoire (4)

- Mémoire Secondaire
 - mémoire persistante même quand la machine est éteinte
 - adressage symbolique (par nom) à des groupes de données (fichiers)



Entrées/Sorties

- Permettent d'interagir avec le monde extérieur et l'utilisateur
 - écran
 - clavier
 - disque dur
 - réseau
 - ...

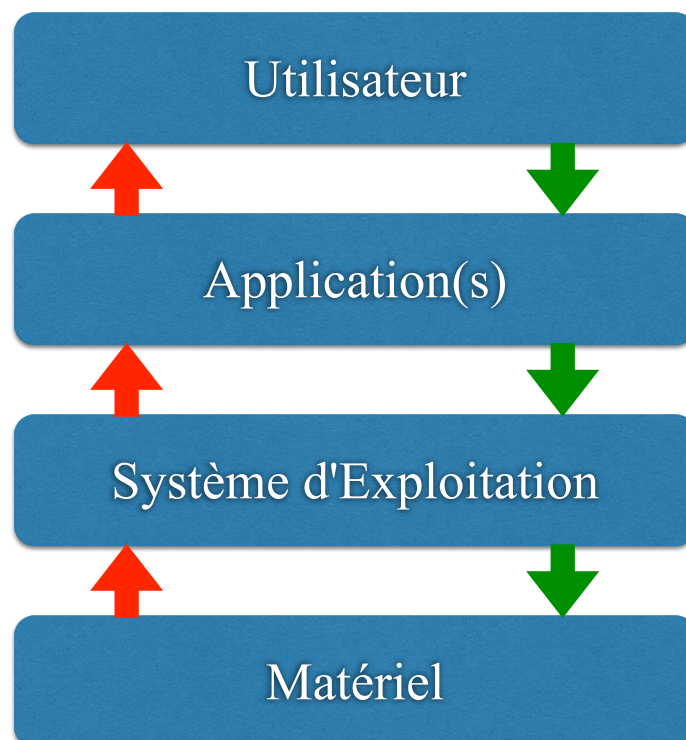
Agenda

- Introduction
 - Organisation d'un Ordinateur
 - Système d'Exploitation
 - Algorithme
 - Codage des Entiers
 - Programme

Systeme d'Exploitation

- L'ordinateur "nu" est totalement inexploitable
- Un ordinateur dispose d'un mécanisme d'initialisation
 - permet de charger un programme particulier
 - système d'exploitation (OS)
- L'OS gère pour l'utilisateur les ressources de la machine et en assure une vision conviviale
- Exemples
 - Windows
 - UNIX
 - Linux
 - BSD
 - Mac OS

Systeme d'Exploitation (2)



Agenda

- Introduction
 - Organisation d'un Ordinateur
 - Système d'Exploitation
 - Algorithme
 - ✓ Définitions
 - ✓ Factorielle
 - ✓ PGCD
 - Codage des Entiers
 - Programme

Définitions

- **Algorithme?**
 - procédure permettant de *résoudre un problème* en décrivant *précisément* et *dans l'ordre* les différentes opérations à effectuer sur base d'un *ensemble de données*
- Exemples
 - recette de cuisine
 - racine carrée d'un nombre positif à une certaine précision près
 - déterminer si un nombre naturel est premier ou non
 - calculer le plus court chemin entre 2 points dans un espace donné
 - ...

Définitions (2)

- Exemples (cont.)
 - extrait d'un dialogue entre un jeune étudiant égaré et une assistante
 - ✓ pourriez-vous m'indiquer le chemin de l'Institut Montefiore?
 - ✓ oui, bien sûr: vous allez tout droit jusqu'au prochain carrefour, vous prenez à gauche au carrefour et, ensuite, la troisième à droite. Vous verrez l'Institut Montefiore juste en face de vous
 - ✓ merci!
- Dans ce dialogue
 - la réponse de l'assistante est la description d'une suite ordonnées d'instructions
 - ✓ tout droit, prendre à gauche, ...
 - manipulant des données
 - ✓ carrefour, rues, ...
 - pour résoudre un problème
 - ✓ aller à l'Institut Montefiore

Définitions (3)

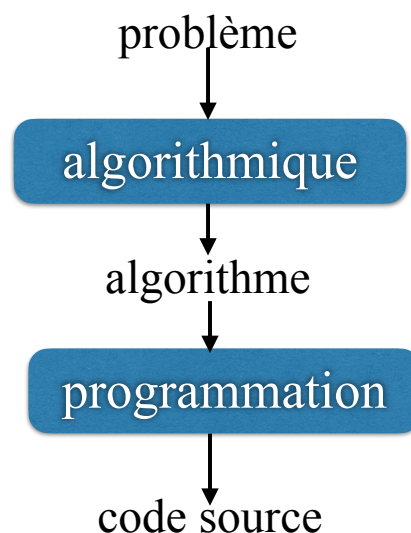
- **Algorithmique**
 - science des algorithmes
- **Validité** d'un algorithme
 - aptitude d'un algorithme à réaliser exactement la tâche pour laquelle il a été conçu
- **Robustesse** d'un algorithme
 - aptitude d'un algorithme à se protéger de conditions anormales d'utilisation
 - cfr. Chap. 5 & 6
- **Réutilisabilité** d'un algorithme
 - aptitude d'un algorithme à être réutilisé pour résoudre des tâches équivalentes à celle pour laquelle il a été conçu
 - cfr. Chap. 6

Définitions (4)

- **Complexité** d'un algorithme
 - le nombre d'instructions élémentaires à exécuter pour réaliser la tâche pour laquelle il a été conçu
 - cfr. Chap. 4
- **Efficacité** d'un algorithme
 - aptitude d'un algorithme à utiliser de manière optimale les ressources du matériel qu'il exécute

Définitions (5)

- La **programmation** a pour rôle de traduire un algorithme dans un langage "compréhensible" par l'ordinateur afin qu'il puisse s'exécuter automatiquement



Factorielle

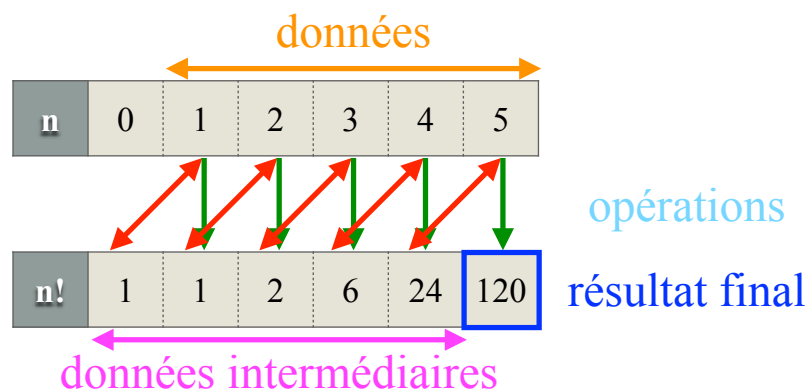
- Fonction factorielle

- $n! = n \times n-1 \times n-2 \times \dots \times 1, n \geq 1$
- $n! = 1, n=0$
- Solution:
 - ✓ calculer explicitement le produit de tous les entiers strictement positifs qui sont inférieurs ou égaux à n
 - ✓ comment faire pour définir
 - les opérations/instructions?
 - les données sur lesquelles travailler?

Factorielle (2)

- Exemple

- $n = 5$



PGCD

- Plus grand commun diviseur (PGCD)
 - $\text{pgcd}(a, b) : \forall a, b \geq 1$, calculer le plus grand nombre entier, appelé *pgcd*, qui divise à la fois a et b
 - Exemples:
 - ✓ $\text{pgcd}(24, 18) = 6$
 - ✓ $\text{pgcd}(10, 10) = 10$
 - ✓ $\text{pgcd}(1, 107) = 1$

PGCD (2)

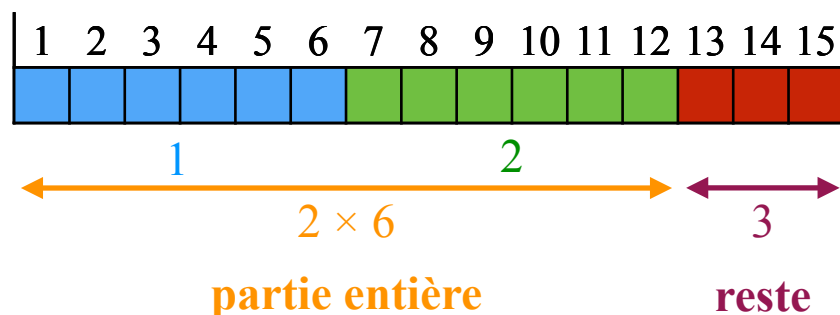
- Solution 1 (*brute force approach*)
 - déduction de l'énoncé
 - ✓ $1 \leq \text{pgcd}(a, b) \leq \min(a, b)$
 - Algorithme
 - ✓ énumérer tous les entiers dans l'intervalle $[1, \min(a, b)]$
 - ✓ tester si ils divisent à la fois a et b
 - ✓ garder le plus grand entier qui satisfait cette condition
 - Ça fonctionne...
 - ✓ ...mais coût prohibitif pour de grandes valeurs de a et b

PGCD (3)

- Solution 2 (*Algorithme d'Euclide*)
 - Observation
 - ✓ $\forall a, b \geq 1: \text{pgcd}(a, b) = \text{pgcd}(b, a)$
 - Si un nombre divise à la fois a et b , alors il divise également leur différence
 - ✓ Dédution: $\forall a, b \geq 1, a \leq b$, on a
 - ✦ $\text{pgcd}(a, b) = \text{pgcd}(a, b-a)$
 - ✦ $\text{pgcd}(a, b) = \text{pgcd}(a, b \% a)$
 - ✓ en considérant que
 - ✦ $\forall n \geq 1, \text{pgcd}(n, 0) = n$
 - ✦ $x \% y$ dénote le reste de la division entière de x par y

PGCD (4)

- Division entière?
 - *division euclidienne*
- Exemple
 - $15 / 6$
 - combien de fois je peux mettre 6 dans 15?



PGCD (5)

- Exemple (cont.)

- $15 = 2 \times 6 + 3$

- Vocabulaire

- 15

- ✓ *numérateur*

- 2×6

- ✓ *partie entière*

- 2

- ✓ *quotient*

- 6

- ✓ *dénominateur*

- 3

- ✓ *reste*

PGCD (6)

- Généralisation

- $x/y \Rightarrow x = q \times y + r$

- ✓ $0 \leq q, r < y$

- ✓ $q = \lfloor x/y \rfloor$ et $r = x \% y$

- L'opérateur **modulo**, %, permet de donner, directement, le reste de la division entière

PGCD (7)

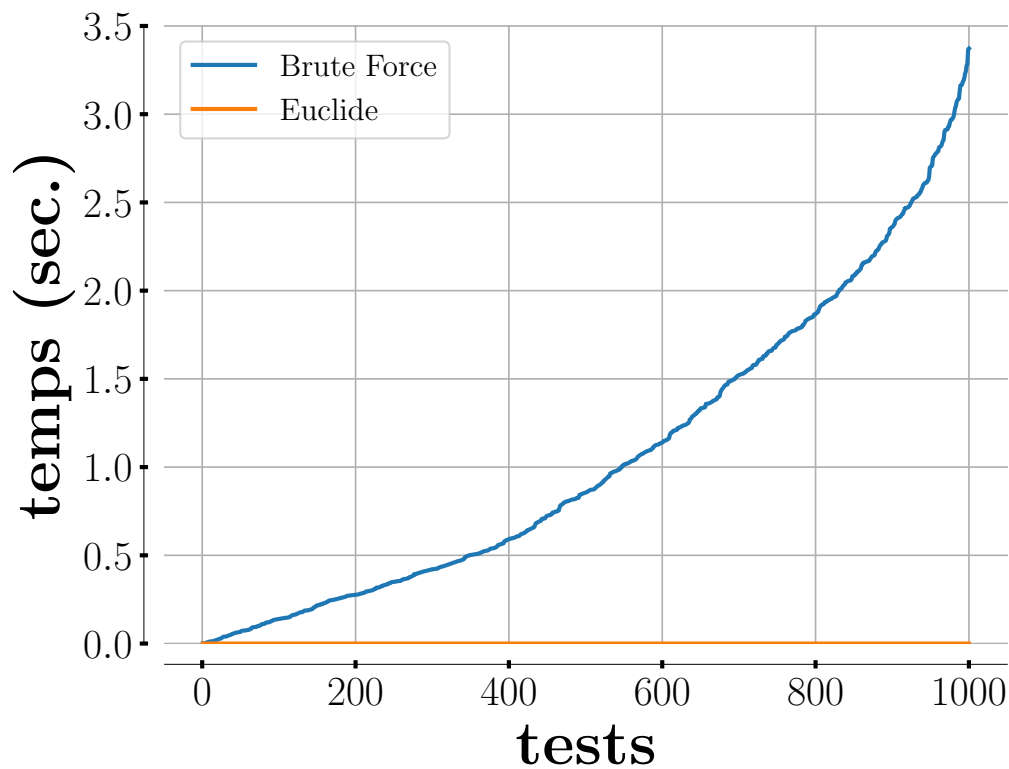
- Algorithme d'Euclide
 $\text{pgcd}(a,b)$
 1. **si** $a > b$
 - ✓ **alors** permuter a et b
 - ✓ **sinon** \emptyset
 2. **Tant que** $a \neq 0$, répéter
 - ✓ remplacer (a, b) par $(b \% a, a)$
 3. **Retourner** la valeur de b
- Illustration: $\text{pgcd}(24, 18)$

Etape	a	b
0	24	18
1	18	24
2	6	18
	0	6
3	0	6

PGCD (8)

- Les algorithmes sont aussi des technologies
 - on peut évaluer leurs performances
- Exemple
 - comparaison des deux solutions (brute force & Euclide) pour le PGCD
- Testbed
 - Intel i7, 2Ghz
 - ✓ quadcore
 - 8Go RAM
 - implémentation en C
 - ✓ $a, b \in [1, 10^9]$
 - ✓ le temps nécessaire pour chaque version est obtenu grâce à `time.h`

PGCD (9)



Agenda

- Introduction
 - Organisation d'un Ordinateur
 - Système d'Exploitation
 - Algorithme
 - Codage des Entiers
 - ✓ Système de Numération
 - ✓ Ordre de Grandeur
 - ✓ Entiers Non Signés
 - ✓ Entiers Signés
 - Programme

Numération

- Un **système de numération** d'un nombre naturel N est constitué
 - d'une **base** $b \in \mathbb{N}$
 - ✓ $|b| > 1$
 - d'un ensemble de b symboles, appelés **chiffres**, compris entre 0 et $b-1$

- Construction et représentation d'un naturel N

$$\begin{aligned} N &= a_{n-1} \times b^{n-1} + a_{n-2} \times b^{n-2} + \dots + a_1 \times b^1 + a_0 \times b^0 \\ &= \sum_{i=0}^{n-1} a_i \times b^i \\ &= (a_{n-1}a_{n-2}\dots a_1a_0)_b \\ &\text{avec } a_i \in \{0, 1, \dots, b-1\} \end{aligned}$$

Numération (2)

- Système de numération les plus pertinents
 - **décimal**
 - ✓ base 10
 - ✓ $b = \{0, 1, \dots, 9\}$
 - ✓ exemple: $(77)_{10}$
 - **binaire**
 - ✓ base 2
 - ✓ $b = \{0, 1\}$
 - ✓ exemple: $(1001101)_2$
 - **octal**
 - ✓ base 8
 - ✓ $b = \{0, 1, \dots, 7\}$
 - ✓ exemple: $(115)_8$
 - **hexadécimal**
 - ✓ base 16
 - ✓ $b = \{0, 1, \dots, 9, A, \dots, F\}$
 - ✓ exemple: $(4D)_{16}$

Ordre de Grandeur

- La représentation d'une donnée en mémoire se fait de façon binaire
 - {0, 1}
 - **bit**
 - ✓ unité élémentaire d'information
- Un ordinateur ne lit que des suites binaires
- Exemple

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

- Quantité:
 - 8 bits = 1 byte (1B)
 - normalisation IEC 80000-13:2008

Ordre de Grandeur (2)

Décimale			Binaire		
Valeur	Métrique	Abbréviation	Valeur	Métrique	Abbréviation
1000	kilobyte	kB	1024	kibibyte	KiB
1000 ²	megabyte	MB	1024 ²	mebibyte	MiB
1000 ³	gigabyte	GB	1024 ³	gibibyte	GiB
1000 ⁴	terabyte	TB	1024 ⁴	tebibyte	TiB
1000 ⁵	petabyte	PB	1024 ⁵	pebibyte	PiB
1000 ⁶	exabyte	EB	1024 ⁶	exbibyte	EiB
1000 ⁷	zettabyte	ZB	1024 ⁷	zebibyte	ZiB
1000 ⁸	yottabyte	YB	1024 ⁸	yobibyte	YiB

Entiers Non Signés

- Représentation binaire des entiers positifs (i.e., naturels)
 - tout naturel $X \in [0, 2^n-1]$ peut être représenté par une suite unique de bits $x_0, x_1, x_2, \dots, x_{n-1}$ telle que

$$X = \sum_{i=0}^{n-1} x_i \times 2^i$$

Entiers Non Signés (2)

- La base 2 est un standard de représentation des entiers
 - beaucoup de processeurs représentent les entiers en 32 bits ($n=32$) ou 64 bits ($n=64$)
- Conversion d'un entier X de base 10 vers base 2
 - algorithme de conversion
 1. rechercher la plus grande puissance de $2 \leq X$
 2. soustraire cette puissance de 2 à X
 3. recommencer à 1. avec le résultat obtenu jusqu'à 0
 4. X s'écrit sous la forme d'une somme de puissance de 2

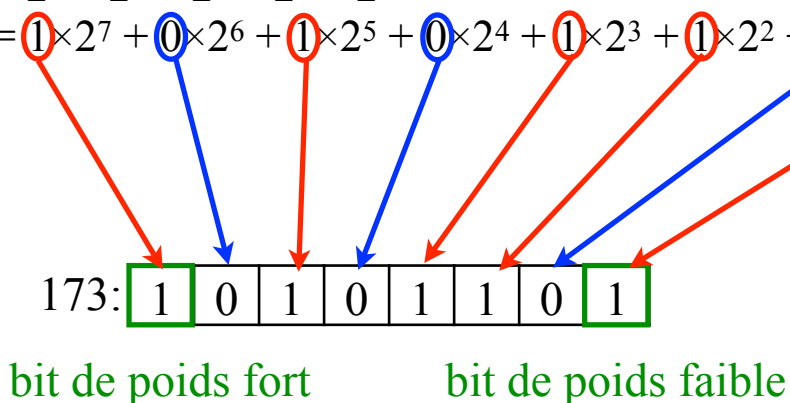
Entiers Non Signés (3)

- Exemple
 - représentation de 173 en base 2, sur 8 bits ($n=8$)

Etape	Valeur	Puissance	Soustraction
1	173	$2^7 = 128$	$173 - 128 = 45$
2	45	$2^5 = 32$	$45 - 32 = 13$
3	13	$2^3 = 8$	$13 - 8 = 5$
4	5	$2^2 = 4$	$5 - 4 = 1$
5	1	$2^0 = 1$	$1 - 1 = 0$

Entiers Non Signés (4)

- On a donc
 - $173 = 2^7 + 2^5 + 2^3 + 2^2 + 2^0$
 - $173 = 1 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$



Entiers Non Signés (5)

- Conversion base 2 \rightarrow base 10

- il suffit de réutiliser la formule $X = \sum_{i=0}^{n-1} x_i \times 2^i$

- Exemple

- $(11001110)_2$

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

$$X = 1 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$$

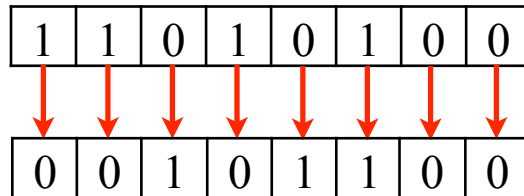
$$X = 206$$

Entiers Signés

- Pour les entiers signés, le bit de poids fort indique le signe
 - 0: positif
 - 1: négatif
- La représentation binaire d'un entier signé se fait via le complément à deux
- Entier signé base 10 \rightarrow base 2
 - Algorithme (avec exemple pour -7)
 1. transformer la valeur absolue du nombre en binaire (cfr. Slide 53)
 - ♦ $7 = 00000111$
 2. inversion des bits
 - ♦ $00000111 \rightarrow 11111000$
 3. ajouter 1
 - ♦ $00000001 + 11111000 = 11111001$

Entiers Signés (2)

- Entier signé base 2 \rightarrow base 10
 - Algorithme (avec exemple pour $(11010100)_2$)
 1. retrouver le complément à 2



2. calculer la somme des exposants avec $X = \sum_{i=0}^{n-1} x'_i \times 2^i$

$$X = 0 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$$

$$X = -44$$

Agenda

- Introduction
 - Organisation d'un Ordinateur
 - Système d'Exploitation
 - Algorithme
 - Codage des Entiers
 - Programme
 - ✓ 3 Etapes
 - ✓ Code Source
 - ✓ Compilation

3 Etapes

- 3 étapes pour un programme
 - lire les données en entrées
 - effectuer des calculs
 - écrire les données en sortie



Code Source

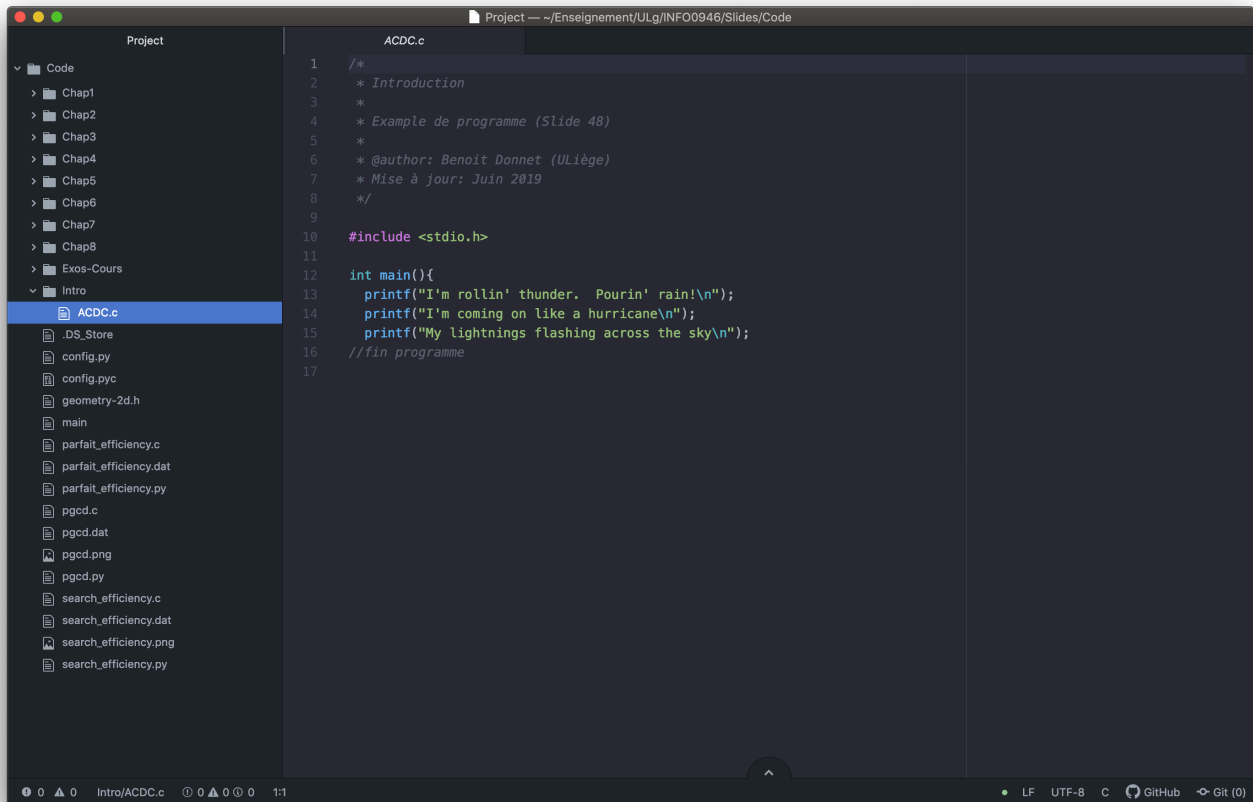
```
/*  
 * Introduction  
 *  
 * Example de programme (Slide 48)  
 *  
 * @author: Benoit Donnet (ULiege)  
 * Mise à jour: Juin 2019  
 */
```

Commentaires

`#include <stdio.h>` dérive de compilation

`int main(){` ordonnancement bloc d'instructions instruction
`printf("I'm rollin' thunder. Pourin' rain!\n");`
`printf("I'm coming on like a hurricane\n");`
`printf("My lightnings flashing across the sky!\n");`
`}//fin programme`

Code Source (2)



```
1  /*
2  * Introduction
3  *
4  * Exemple de programme (Slide 48)
5  *
6  * @author: Benoit Donnet (ULiège)
7  * Mise à jour: Juin 2019
8  */
9
10 #include <stdio.h>
11
12 int main(){
13     printf("I'm rollin' thunder. Pourin' rain!\n");
14     printf("I'm coming on like a hurricane\n");
15     printf("My lightnings flashing across the sky\n");
16     //fin programme
17 }
```

Code Source (3)

- Comment exécuter le programme?
 - l'ordinateur ne comprend pas les **langages de programmation**
 - ✓ langage informatique permettant à un humain d'écrire un code source qui sera analysé par un ordinateur
- Nécessité de transformer le programme en quelque chose de compréhensible par l'ordinateur
 - *compilation*

Compilation

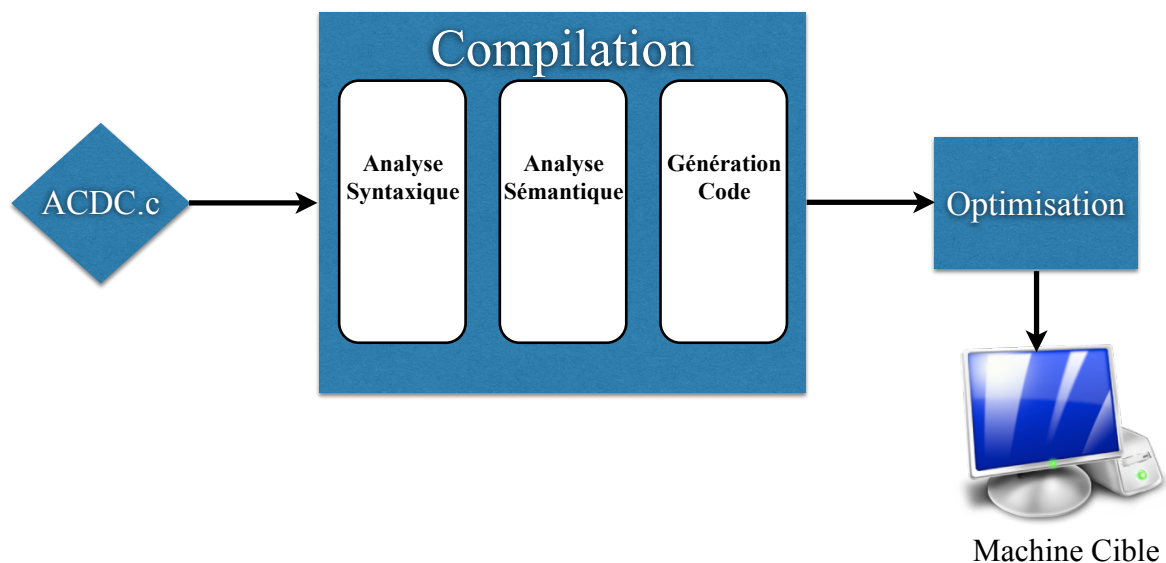
- **Compilation**

- transforme un programme écrit dans un langage de haut niveau (e.g., C) en un ensemble d'instructions exécutables par la machine
- vérifie aussi
 - ✓ la *syntaxe*
 - ✦ principes et règles permettant de construire des phrases
 - ✦ façon dont les mots-clés du langage se combinent pour construire un programme
 - ✓ la *sémantique*
 - ✦ donne un sens au programme

- **Compilateur**

- programme qui effectue le processus de compilation
 - ✓ en C, il s'agit de `gcc`

Compilation (2)



Compilation (3)

- Exemple de compilation et exécution

Exécutable
Compilateur Source

```
gibson:~ benoit$ gcc -o ACDC ACDC.c
gibson:~ benoit$ ./ACDC
I'm rollin' thunder.  Pourin' rain!
I'm coming on like a hurricane
My lightnings flashing across the sky!
gibson:~ benoit$
```