Architecture des ordinateurs

(X31I050)

Frédéric Goualard

Laboratoire d'Informatique de Nantes-Atlantique, UMR CNRS 6241
Bureau 112, bât. 11
Frederic.Goualard@univ-nantes.fr

Aspects pratiques 1

- Ressources et planning sur madoc
- *Syllabus* : description de tous les aspects pratiques (**lisez-le !**)
- Fascicules présentant les notions vues en cours

Aspects pratiques 2

- 10 cours magistraux d'1h20
  - Présentation des points à étudier (« *Notions abordées* »)
  - 7 quiz sur madoc pour la compréhension de chaque partie
- 12 séances de travaux dirigés d'1h20
  - Faire les exercices de TD **avant**
  - Mini cours interactif sur les points difficiles relevés
  - 2 évaluations sur table des parties précédentes
- 9 séances de travaux pratiques
  (dont 1h20 d'évaluation : projet + assembleur MIPS)
  - C/C++
  - Circuits logiques (logisim)
  - Assembleur MIPS (MARS)

Travail personnel **absolument** indispensable en amont des séances

Planning

Architecture des ordinateurs

2022/2023

Tous groupes sauf alternants MIAGe

Attention: le planning est susceptible de changer au cours du semestre et de varier en fonction des groupes.

v. 1, 2022-09-01 *10.5 CMs, 12 TDs, 9 TPs*	**Travail à faire** **(Exercices de TD et quiz)**				
(36) 05/09 – 09/09		CM1 Représentation de l'information	CM2 Représentation de l'information		
(37) 12/09 – 16/09	Ex. 1.1-1.5, 1.7, 1.8, 1.9, 1.11, 1.12	CM3 Représentation de l'information Performances	CM4 Logique & circuits combinatoires	TD 1.1 Représentation de l'information	
(38) 19/09 – 23/09	Ex. 1.10, 1.14, 1.15, Ex. 2.1, 2.5 Quiz "Représentation de l'information" Quiz "Performances"	CM5 Circuits séquentiels	TD 1.2 Représentation de l'information		
(39) 26/09 – 30/09	Ex. 3.1, 3.2, 3.4 Quiz "Logique & circuits combinatoires"	TP 1.1 Représentation de l'information	TD 2 Performances		
(40) 03/10 – 07/10	Ex. 3.6	TP 1.2 Représentation de l'information	TD 3.1 Circuits combinatoires		
(41) 10/10 – 14/10	Ex. 4.1, 4.2	TP 2 Circuits combinatoires	CM6 Circuits séquentiels	CC 1 Rep. Info + Performances TD 3.2 Circuits combinatoires	
(42) 17/10 – 21/10	Ex. 4.3, 4.4 Quiz "Circuits séquentiels"	TP 3.1 Circuits séquentiels	CM7 Assembleur MIPS		TD 4.1 Circuits séquentiels
(43) 24/10 – 28/10	Ex. 5.1, 5.4 Quiz "Assembleur MIPS"	TP 3.2 Circuits séquentiels	TD 4.2 Circuits séquentiels		
(44) 31/10 – 04/11	Distribution du sujet de mini-projet				
(45) 07/11 – 11/11	Ex. 5.5, 5.7	CM8 Assembleur MIPS Instructions et chemins de données	TD 5.1 Assembleur MIPS	TP Mini-projet	CM9 Instructions et chemins de données
(46) 14/11 – 18/11	Ex. 5.6, 5.9, 5.11 Quiz "Instructions et chemins de données"	TP 4.1 Assembleur MIPS	CM10 Pipelines Caches	TD 5.2 Assembleur MIPS	*CM 11* *(40 minutes)*
(47) 21/11 – 25/11	Ex. 6.1, 6.3 Quiz "Pipelines et caches"	TP 4.2 Assembleur MIPS	CC 2 Circuits combinatoires & séquentiels TD 5.3 Assembleur MIPS		
(48) 28/11 – 02/12	Ex. 7.1, 7.3, 7.4, 7.5	TP CCTP MIPS	TD 6 Instructions & chemins de données		
(49) 05/12 – 09/12		TD 7 Pipelines et caches			
(50) 12/12 – 16/12					

Objectifs du cours et évaluation

- Objectifs du cours
  - Connaître la représentation interne des données/instructions en binaire
  - Comprendre les mécanismes de fonctionnement d'un processeur
  - Comprendre les interactions entre processeur et périphériques (mémoire, unités de stockage de masse, E/S,...)
  - Appréhender l'impact de l'architecture d'un ordinateur sur les performances de programmes écrits dans des langages de haut niveau
- Évaluation

Contrôle continu	2 contrôles « sur table »	25%	50%
	Exercice de TP	10%	
	Projet de TP	15%	
Examen			50%

- Gestion des absences
  - Voir le syllabus

Motivations (1)

Pourquoi un module sur l'architecture des ordinateurs ?

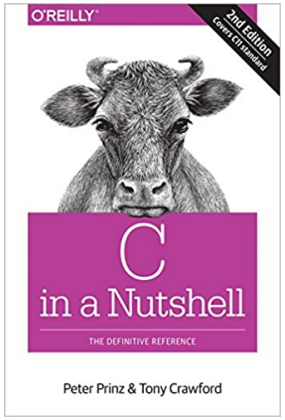
Exemple de la boîte de vitesses d'une voiture : si on connaît son principe de fonctionnement, on sait mieux utiliser le véhicule :

- Prise directe
- Démarrage en seconde sur glace

Motivation (2)

C in a Nutshell, 2e édition :

```
long double factorial(unsigned int n)
{
        long double f = 1;
        while (n > 1) {
                f *= n--;
        }
        return f;
}
```

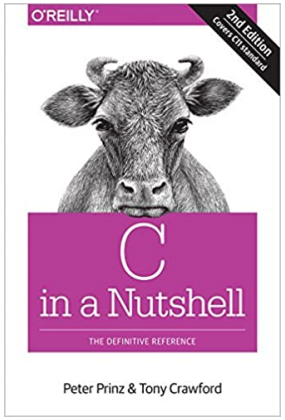


*« Although the factorial of an integer is always an integer, the function uses the type **long double** in order to accommodate very large results. »*

Motivation (2)

C in a Nutshell, 2e édition :

```
long double factorial(unsigned int n)
{
        long double f = 1;
        while (n > 1) {
                f *= n--;
        }
        return f;
}
```



> *« Although the factorial of an integer is always an integer, the function uses the type **long double** in order to accommodate very large results. »*

Problème avec `long double` :

- Partie fractionnaire de 64 bits
- Au-delà de 2^{65} : tous les entiers ne sont pas représentables
- Type `uint64_t` : calcul jusqu'à $n = 20$
- Type `long double` : calcul jusqu'à $n = 25$ (`factorial(26)` faux)

Motivations (3)

```cpp
#include <iostream>

using namespace std;

int main(void) {
  unsigned int matSZ;
  cout << "Taille de la matrice ? ";
  cin >> matSZ;
  int *mat = new int[matSZ*matSZ*sizeof(int)];
  unsigned int ligne, colonne;
  cout << "Position de l'élément ? ";
  cin >> ligne >> colonne;
  int val;
  cout << "Valeur de l'élément ? ";
  cin >> val;
  if (ligne >= matSZ || colonne >= matSZ) {
    cout << "Position invalide" << endl;
  } else {
    mat[ligne*matSZ+colonne] = val;
  }
}
```

```java
import java.util.Scanner;

public class scan {
  public static void main(String[] args)
{
    Scanner in = new Scanner(System.in);
    System.out.printf("Enter i: ");
    int i = in.nextInt();
    int[] M = new int[i*i*4];
    M[5] = 3;
  }
}
```

Motivations (3)

```cpp
#include <iostream>

using namespace std;

int main(void) {
  unsigned int matSZ;
  cout << "Taille de la matrice ? ";
  cin >> matSZ;
  int *mat = new int[matSZ*matSZ*sizeof(int)];
  unsigned int ligne, colonne;
  cout << "Position de l'élément ? ";
  cin >> ligne >> colonne;
  int val;
  cout << "Valeur de l'élément ? ";
  cin >> val;
  if (ligne >= matSZ || colonne >= matSZ) {
    cout << "Position invalide" << endl;
  } else {
    mat[ligne*matSZ+colonne] = val;
  }
}
```

```java
import java.util.Scanner;

public class scan {
  public static void main(String[] args)
{
    Scanner in = new Scanner(System.in);
    System.out.printf("Enter i: ");
    int i = in.nextInt();
    int[] M = new int[i*i*4];
    M[5] = 3;
  }
}
```

- Si `matSZ==`2^{31}`==2147483648` , allocation de 0 octet dans le tas (`new int[size_t]` et `sizeof(size_t)==8`)
- Écrasement possible d'autres variables dans le tas, même avec une saisie contrôlée (exception en Java)

Motivation (4)

```
#include <iostream>
#include <cstring>

using namespace std;

int main(void)
{
  struct {
         char mot_de_passe[15];
         bool mdp_ok = false;
  } data;

  cout << "Mot de passe ? ";
  cin >> data.mot_de_passe;


  if (!strcmp(data.mot_de_passe, "Sesame")) {
    data.mdp_ok = true;
  }

  // [...]

  if (data.mdp_ok) {
    cout << "Mot de passe correct" << endl;
  } else {
    cout << "Mot de passe incorrect" << endl;
  }
}
```

```
Mot de passe ? avoine
Mot de passe incorrect

Mot de passe ? Sesame
Mot de passe correct
```

Motivation (4)

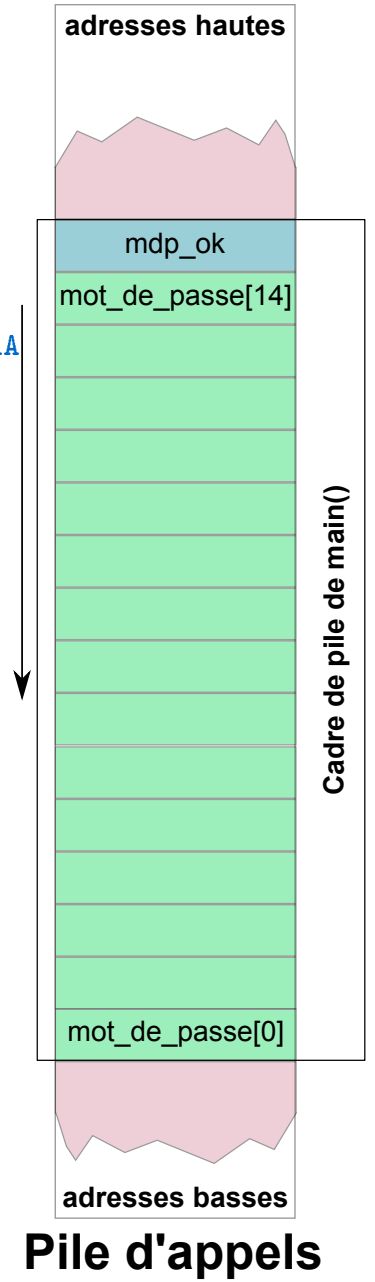
```
#include <iostream>
#include <cstring>

using namespace std;

int main(void)
{
  struct {
         char mot_de_passe[15];
         bool mdp_ok = false;
  } data;

  cout << "Mot de passe ? ";
  cin >> data.mot_de_passe;


  if (!strcmp(data.mot_de_passe, "Sesame")) {
    data.mdp_ok = true;
  }

  // [...]

  if (data.mdp_ok) {
    cout << "Mot de passe correct" << endl;
  } else {
    cout << "Mot de passe incorrect" << endl;
  }
}
```

```
Mot de passe ? avoine
Mot de passe incorrect

Mot de passe ? Sesame
Mot de passe correct

Mot de passe ? AAAAAAAAAAAAAAA
Mot de passe correct
```

Motivation (4)

```
#include <iostream>
#include <cstring>

using namespace std;

int main(void)
{
  struct {
         char mot_de_passe[15];
         bool mdp_ok = false;
  } data;

  cout << "Mot de passe ? ";
  cin >> data.mot_de_passe;


  if (!strcmp(data.mot_de_passe, "Sesame")) {
    data.mdp_ok = true;
  }

  // [...]

  if (data.mdp_ok) {
    cout << "Mot de passe correct" << endl;
  } else {
    cout << "Mot de passe incorrect" << endl;
  }
}
```

```
Mot de passe ? avoine
Mot de passe incorrect

Mot de passe ? Sesame
Mot de passe correct

Mot de passe ? AAAAAAAAAAAAAAAA
Mot de passe correct
```



Pile d'appels

Problème : Buffer overflow

Références

- John L. Hennessy and David A. Patterson. *Computer Architecture : A Quantitative Approach*. Morgan Kaufmann, September 2006
- Noam Nisan and Shimon Schocken. *The Elements of Computing Systems : Building a Modern Computer from First Principles*. MIT Press, illustrated edition, 2008
- Daniel Page. *Practical Introduction to Computer Architecture*. Texts in Computer Science. Springer London, 2009
- Charles Petzold. *CODE : The Hidden Language of Computer Hardware and Software*. Microsoft Press, 2000

Plan du cours

1. Représentation de l'information
2. Performances
3. Logique, circuits combinatoires et circuits séquentiels
4. Architecture du processeur MIPS
   (gestion des instructions, chemins de données)
5. Assembleur MIPS
6. Pipelines et caches