

# Langage machine



Année académique 2021-2022

# Cadre de l'activité d'apprentissage

BINV1070-B : Langage machine

010000100110100101100101011011100111011001100101011011100111010101100101

# Langage Machine

- AcA orientée « Pratique »
  - ✓ Langage machine
- Examen de LM sur machine en janvier
  - ✓ LM vaut pour 100% de la note de l'UE
- En cas d'échec en janvier, représentable en juin et en septembre
- Il est obligatoire pour tous les étudiants de signer tous les examens de janvier

# Cadre du cours

- **Escaliers** de 2h x 12 semaines
  - ✓ La plupart des **notions** seront approfondies au fil des séances
  - ✓ **Régularité**
- Supports de cours
  - ✓ Présentations PowerPoint
  - ✓ Fiches d'exercices
  - ✓ **Prendre des notes manuscrites**





# Atouts et sens du cours

- Pourquoi s'intéresser à ce langage ?

- ✓ Compréhension

- d'un langage dit de bas niveau
- du fonctionnement proche du processeur

- ✓ Abstraction

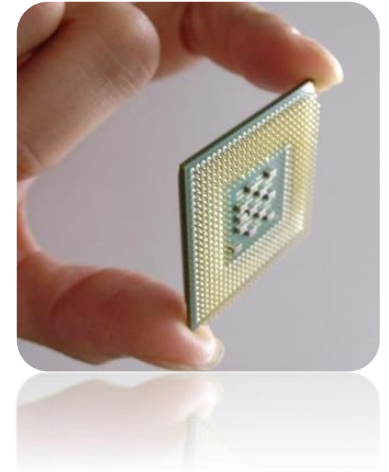
- ✓ Précision



# Plan global du cours

- Des concepts **essentiels** pour l'informaticien

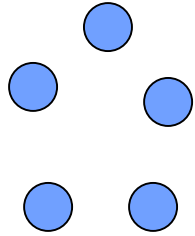
- ✓ Bases **binaire** et **hexadécimal**
- ✓ Fonctionnement orienté programmation dans et autour du **processeur**
- ✓ **Adressage** de valeurs en **mémoire**
- ✓ Langage assembleur **NASM** 32 bits
- ✓ **Algorithmique** fondamentale
  - Affectations, tests, boucles, pile
- ✓ **Hacking** éthique



# Trois bases pour un informaticien

Hexadécimal, binaire et décimal

# Systèmes de numération



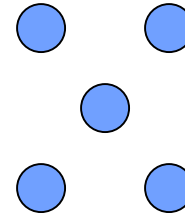
V

101

Cinq

Five

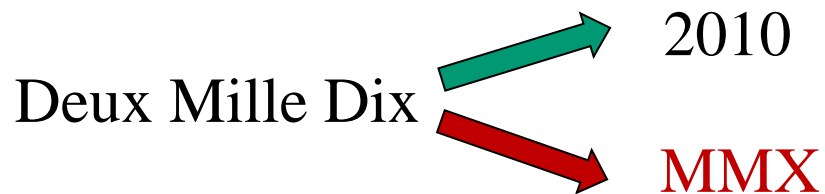
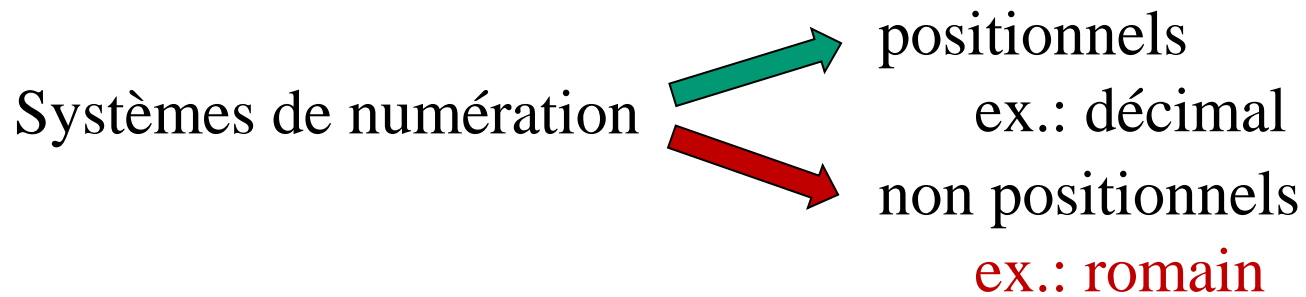
5



Vijf

8 représentations différentes du nombre 5  
8 systèmes différents de numération





Un système de numération positionnel  
est caractérisé par sa base et  
par le nombre de symboles

**Une base est la valeur attribuée au symbole 10**

# Les bases...

- Décimale
  - ✓  $10_{10} = 10_{10}$
- Binaire
  - ✓  $10_2 = 2_{10}$
- Hexadécimale
  - ✓  $10_{16} = 16_{10}$

**Une base est la valeur attribuée au symbole 10**

# Compter...

- Selon la base décimale
  - ✓ 0,1,2,3,4,5,6,7,8,9,10,11,12,13,...
- Selon la base binaire
  - ✓ 0,1,10,11,100,101,110,111,1000,1001,1010,1011,1100,1110,1111,10000,...
- Selon la base hexadécimale
  - ✓ 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F,10,11,12,13,14,15,16,17,18,19,1A,1B,1C,1D,1E,1F,20,21,22,23,24,25,26,27,28,29,2A,2B,...

# Dans la base hexadécimale

- ...
- $A_{16} = 10_{10}$
- $B_{16} = 11_{10}$
- $C_{16} = 12_{10}$
- $D_{16} = 13_{10}$
- $E_{16} = 14_{10}$
- $F_{16} = 15_{10}$
- $10_{16} = 16_{10}$
- ...



□ Système décimal

base :  $10_{10}$

symboles : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

$$4693 = 4*10^3 + 6*10^2 + 9*10^1 + 3*10^0$$

□ Système hexadécimal

base :  $10_{16}$  ( $= 16_{10}$ )

symboles : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,  
A, B, C, D, E, F

$$\begin{aligned} A3C_{16} &= A_{16}*10_{16}^2 + 3*10_{16}^1 + C*10_{16}^0 \\ &= 10*16^2 + 3*16^1 + 12*16^0 = 2620 \end{aligned}$$

□ Système binaire

base :  $10_2$  ( $= 2_{10}$ )

symboles : 0, 1

$$\begin{aligned} 101_2 &= 1_2*10_2^2 + 0_2*10_2^1 + 1_2*10_2^0 \\ &= 1*2^2 + 0*2^1 + 1*2^0 = 5 \end{aligned}$$

# A méditer...

- *There are 10 kinds of people : those who understand binary, and those who don't.*



# Introduction à l'IDE SASM

Environnement de développement  
« intégré »

# L'IDE SASM

- Simple Open Source IDE
  - ✓ pour **NASM**, MASM, GAS, FASM
  - ✓ x86 (**32 bits**) ou x64 (64 bits)
- <https://dman95.github.io/SASM/english.html>
- **Windows** ou Linux
- **Pour Mac**  
<https://sites.google.com/a/brianrhall.net/www/rss/installingsasmonamac>
  - ✓ **Ne fonctionne pas sur MacOS High Sierra**



# L'IDE SASM

- Démonstration de l'IDE...
  - ✓ Semaine.1.Théorie.1.asm
  - ✓ Semaine.1.Théorie.1.exe
- .exe exécutable  
en fenêtre de commandes  
sous Windows...
  - ✓ Dans l'explorateur Windows, sélectionner l'URL et ajouter CMD devant l'URL pour appuyer sur la touche *Enter* pour ouvrir une fenêtre de commandes dans ce répertoire



Memory

Variable or expression	Value	Type
Add variable...	Smart	d Array size <input type="checkbox"/> Address

Semaine.1.Théorie.1.asm

```

1 %INCLUDE "io.inc" ; procédures d'input/ouput clavier et écran, voir l'aide (
2
3 SECTION .bss ; section pour déclarer des données non initialisées
4
5 SECTION .data ; section pour déclarer des données initialisées
6 message db 'Institut Paul Lambin',0
7
8 SECTION .text ; section pour écrire du code en langage assembleur
9 ; cette section .text contiendra des instructions écrites
10 GLOBAL CMAIN ; le libellé de début de la programmation dans l'IDE SASM
11
12 CMAIN:
13     mov     ebp,esp; pour debugging fonctionnel dans l'IDE SASM
14
15 ; le libellé CMAIN: est le point d'entrée de notre programme
16 ; écrivez votre instructions en langage assembleur NASM 32 bits
17 ; à partir de la ligne ci-dessous
18     PRINT_STRING message
19
20
21 ; fin correcte de CMAIN dans l'IDE SASM
22     xor     eax,eax
23     ret

```

Input



Output

Institut Paul Lambin

Registers

Register	Hex	Info
eax	0x1	1
ecx	0x4018c0	4200640
edx	0x0	0
ebx	0x7ffde000	2147344384
esp	0x28ff1c	0x28ff1c
ebp	0x28ff1c	0x28ff1c
esi	0x40126c	4199020
edi	0x40126c	4199020
eip	0x401403	0x401403 <main+115>
eflags	0x206	[ PF IF ]
cs	0x23	35
ss	0x2b	43
ds	0x2b	43
es	0x2b	43
fs	0x53	83
gs	0x2b	43

[09:27:34] Build started...  
 [09:27:34] Built successfully.  
 [09:27:34] Debugging started...

GDB command:

☐ Print

Perform

# Librairie SASM d'entrées/sorties

- Les **lectures** (au clavier) et **écritures** (à l'écran) sont des opérations complexes
- Une **librairie** permet de simplifier cela
- **"io.inc"** *macro library for NASM*
  - ✓ Documentation dans l'aide (touche F1)
- **PRINT\_STRING** *data*
  - ✓ *Print null-terminated text string*
  - ✓ *data* : string constant or name of variable
  - ✓ Ex. de variable initialisée :  
message db 'Institut Paul Lambin',0

# Introduction au processeur

Architecture x86 32 bits



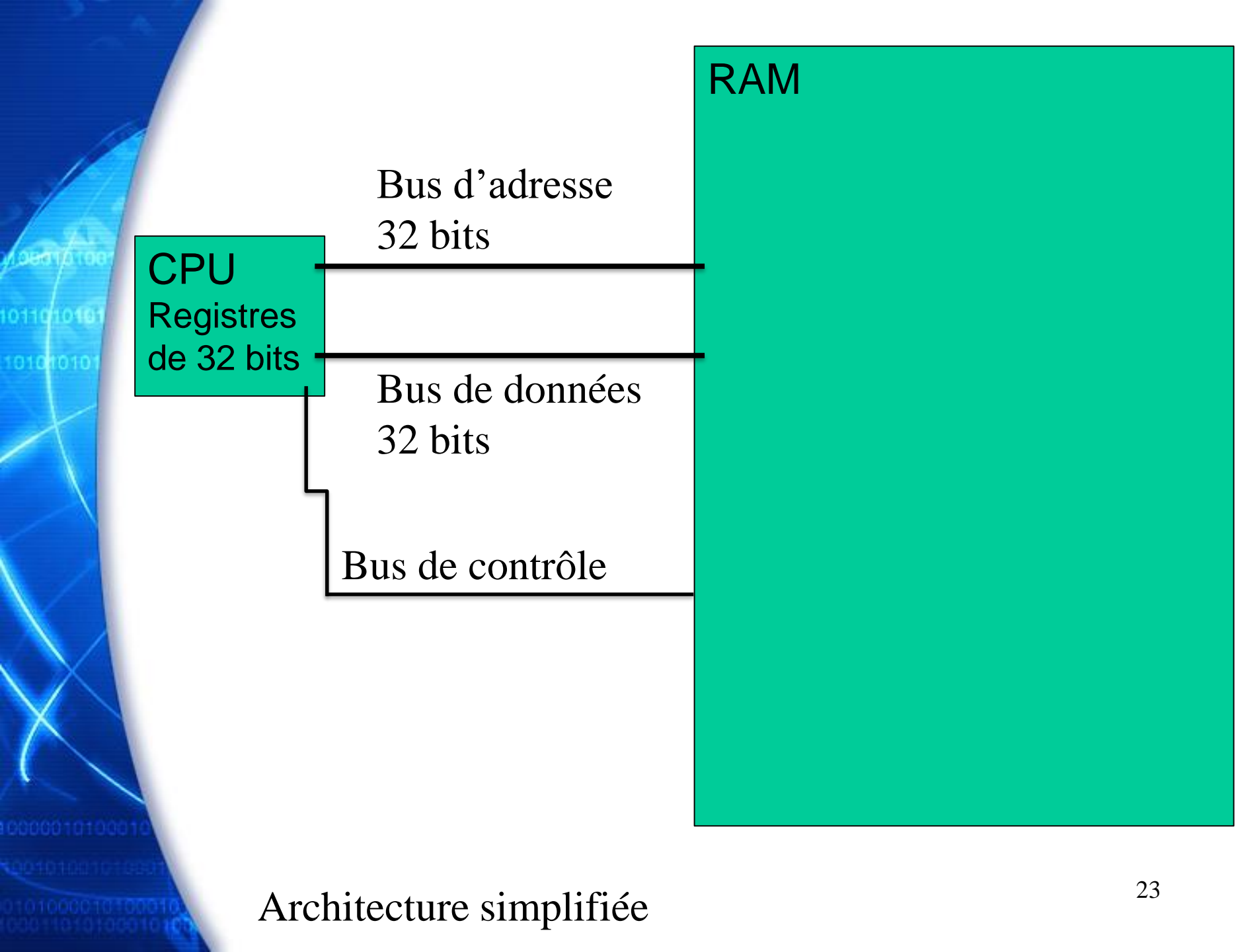
# Théorie d'un processeur 32 bits

- La plupart des notions abordées ci-après seront **approfondies** plus tard
- Nous reviendrons donc sur ce sujet **au fil des séances selon nos besoins** pour les exercices

# Composants principaux

- Le **Processeur** (*CPU*) contient
  - ✓ Des **registres**
  - ✓ Une UAL : Unité Arithmétique et Logique (*ALU*)
  - ✓ Une UC : Unité de Commande (*CU*)
  - ✓ Une horloge qui donne le rythme par seconde
- La **Mémoire**
  - ✓ RAM (*Random Access Memory*)
- Le **Bus de données** (*Databus*)

permet le transfert des données entre le processeur, la mémoire, et les composants externes.



# Les bus...

- **Le bus de données**, permet la circulation des données
- **Le bus d'adresse**, permet de pointer la valeur à aller chercher en mémoire
- **Le bus de contrôle**, permet de contrôler le type de l'opération sur le bus, par ex. *Read* ou *Write*



# Représentation des informations

- Toute information est dans l'ordinateur une suite de 0 et de 1
- 0 ou 1
  - ✓ 1 chiffre binaire = 1 *binary digit* = 1 bit
- 1 octet contient 8 bits
  - ✓ L'octet est la plus petite entité adressable
  - ✓ Schématisé 

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

 ou 

A	5
---	---
- Toute information est représentée sur un nombre limité et fixé d'octets

# Les registres de 32 bits

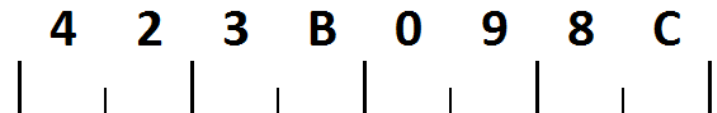
- **EAX** (Accumulateur)
- **EBX** (Base)
- **ECX** (Compteur)
- **EDX** (Donnée)

31	15	7	0	15	0
EAX	AX (AH)	AX (AL)		CS	
EBX	BX (BH)	BX (BL)		DS	
ECX	CX (CH)	CX (CL)		SS	
EDX	DX (DH)	DX (DL)		ES	
ESI	SI			FS	
EDI	DI			GS	
EBP	BP				
ESP	SP				
EFLAGS	FLAGS				
EIP	IP				

- Affectation d'une valeur dans le registre EAX :

✓ **MOV EAX, 0x423B098C**

✓ Schématisé comme suit



- Page à étudier (de manière progressive)  
sur Wikipédia : [https://fr.wikibooks.org/wiki/Programmation\\_Assembleur/x86/Registres](https://fr.wikibooks.org/wiki/Programmation_Assembleur/x86/Registres)

✓ Y revenir souvent !