



ALF

Représentation des
données

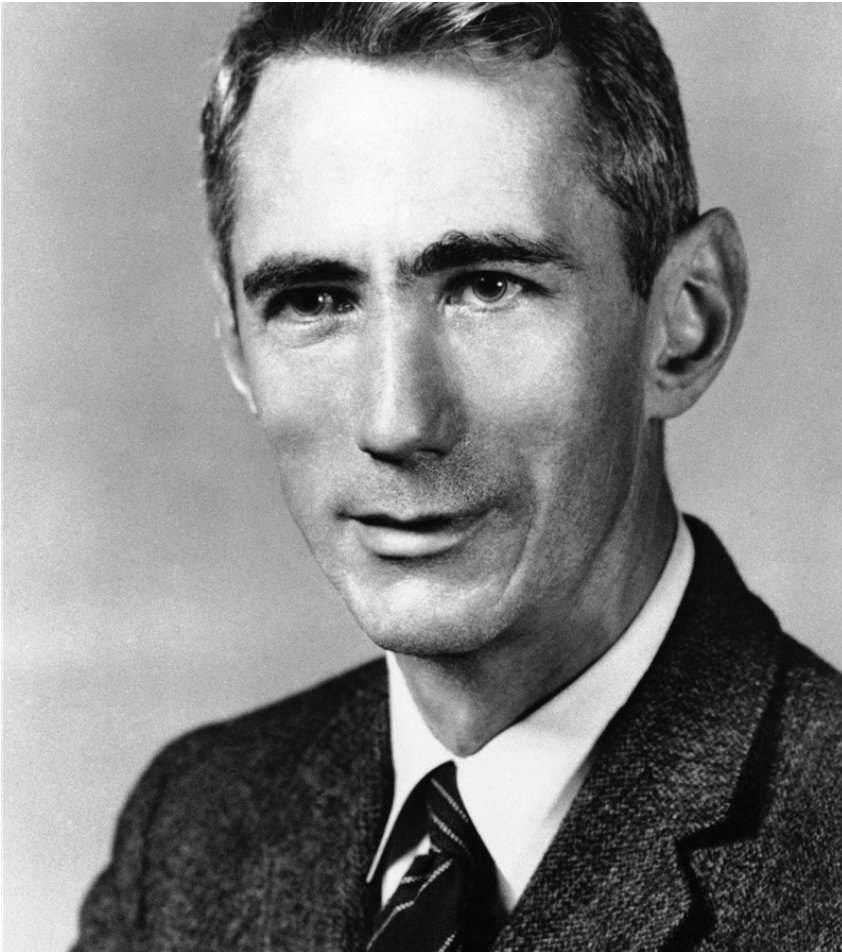
Andrew Tanenbaum, *Modern Operating Systems (4th Edition)*

- Chapitre 1
 - 1.3.1
 - 1.3.2

- CPU
- Mémoire
- Format des données
 - Numéro
 - Numéro avec virgule
 - String
 - Array
 - Struct

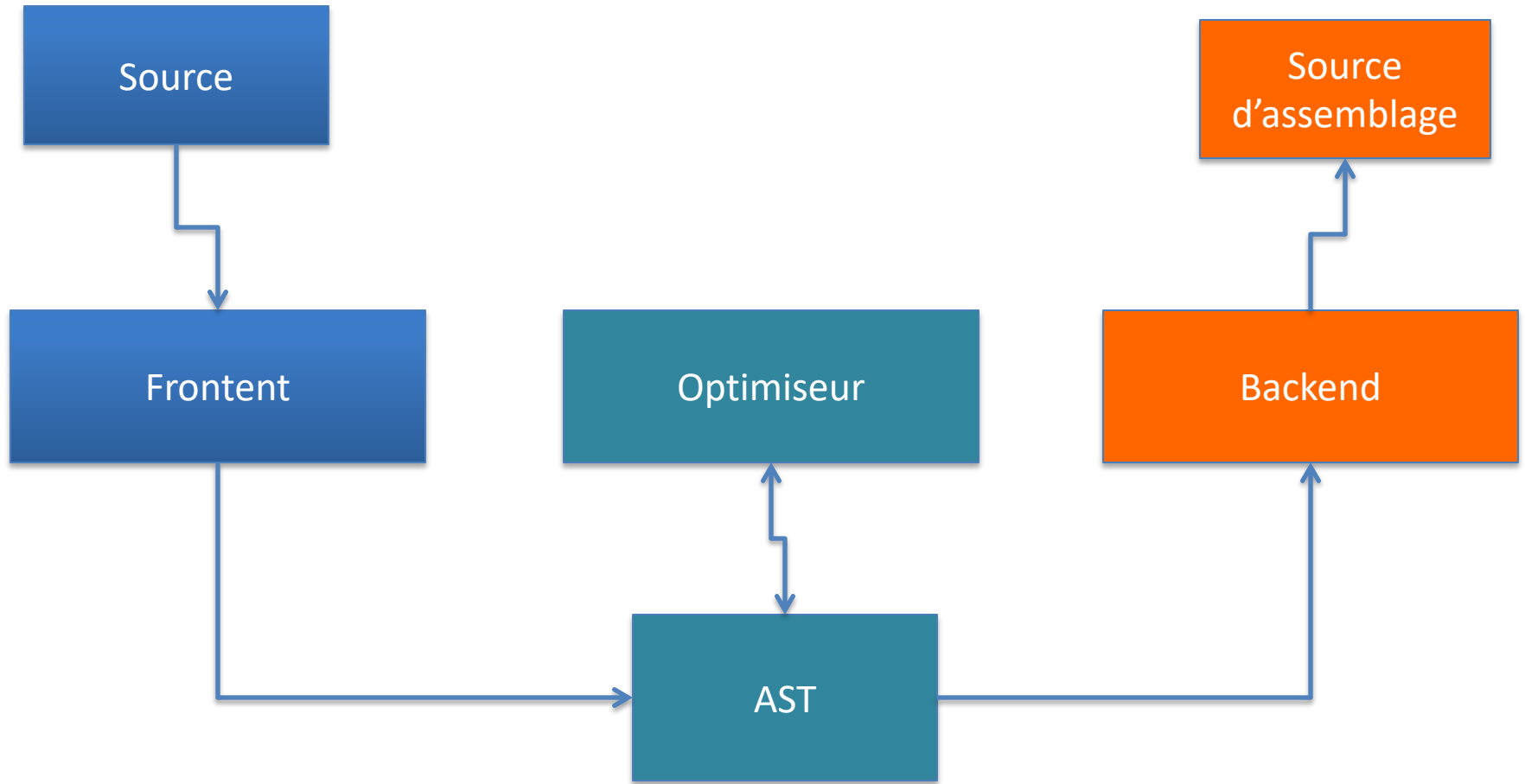


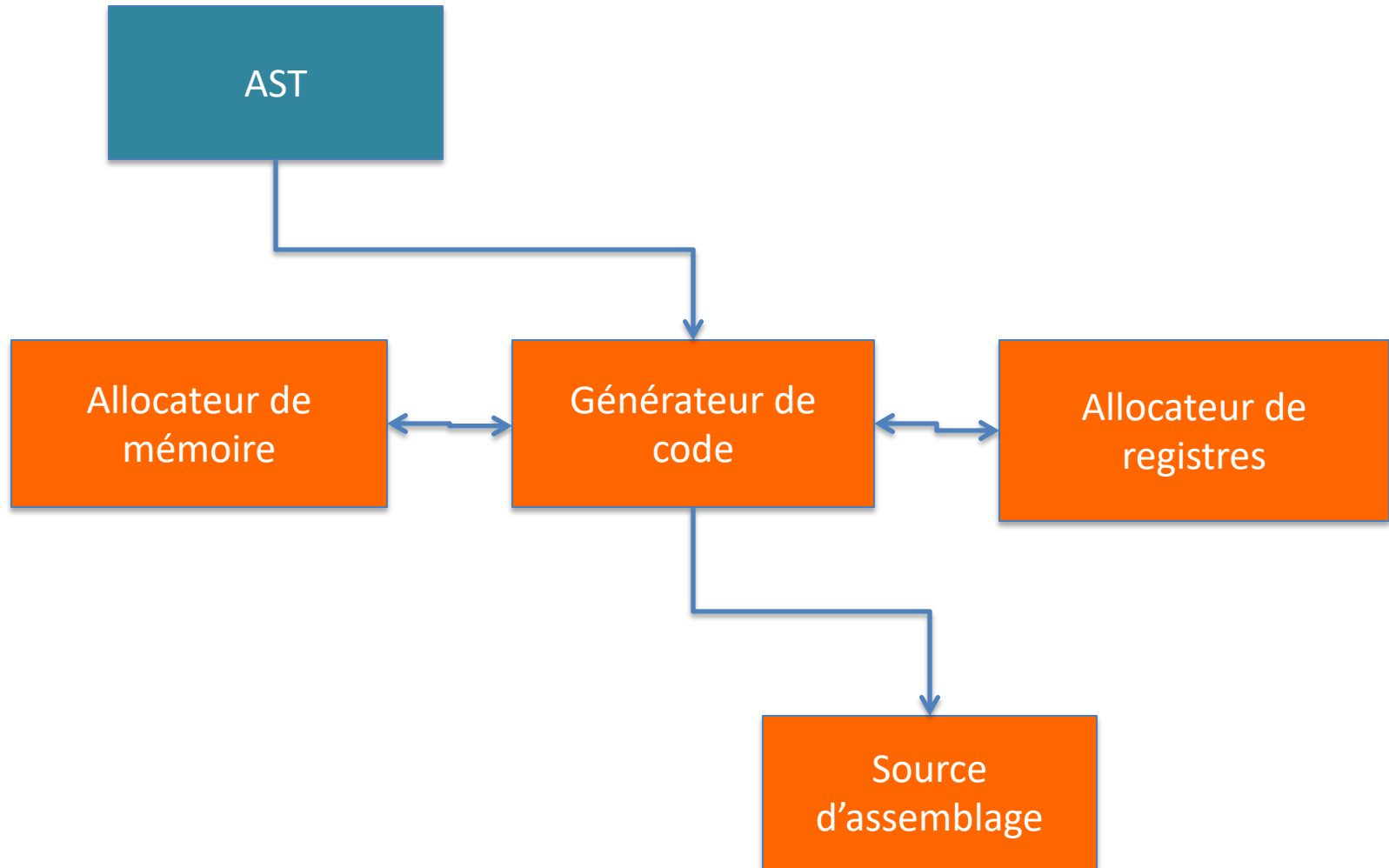
Claude Shannon



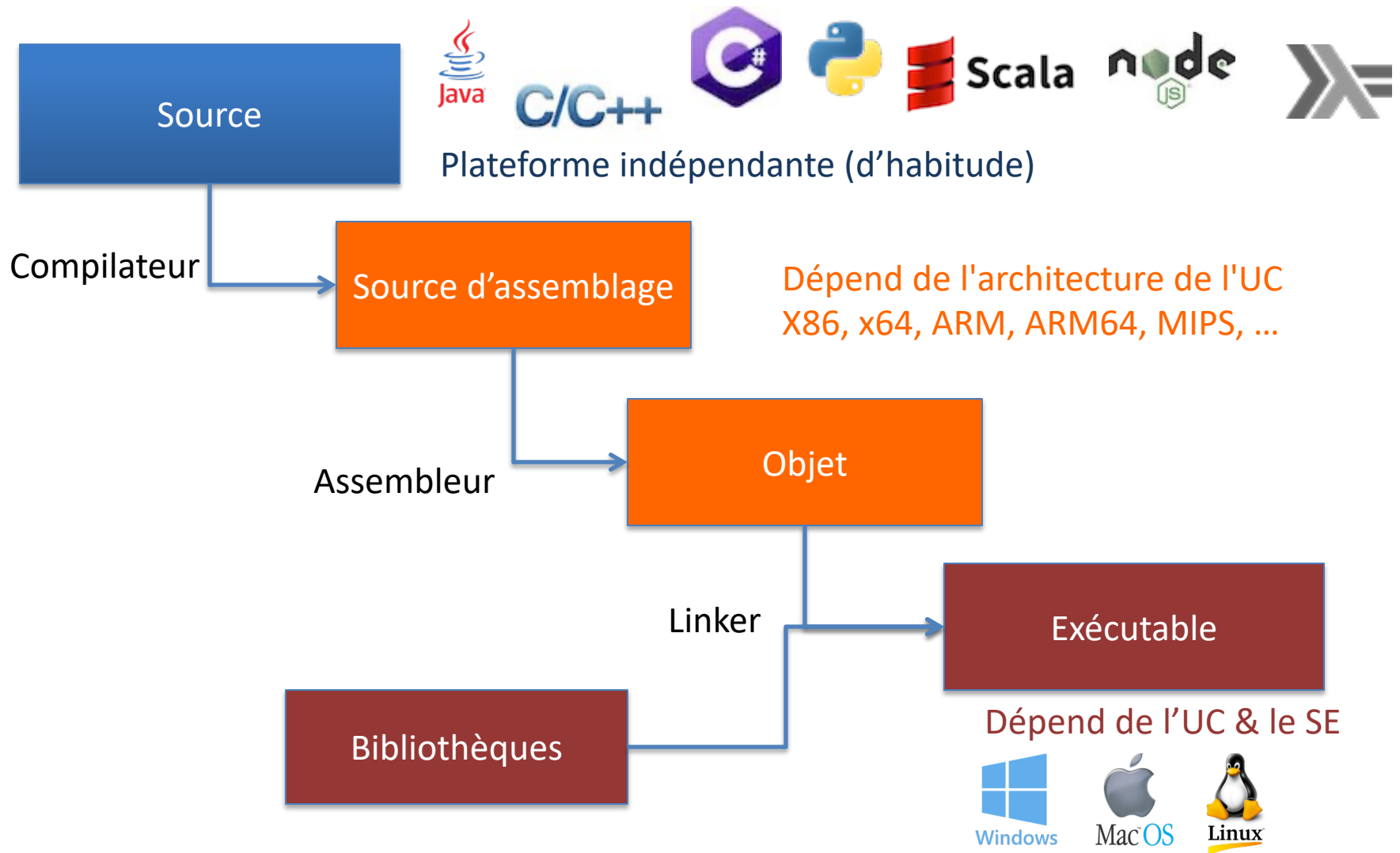
- Américain
- University of Michigan
- MIT
- Théorie d'information

Pièces de compilation





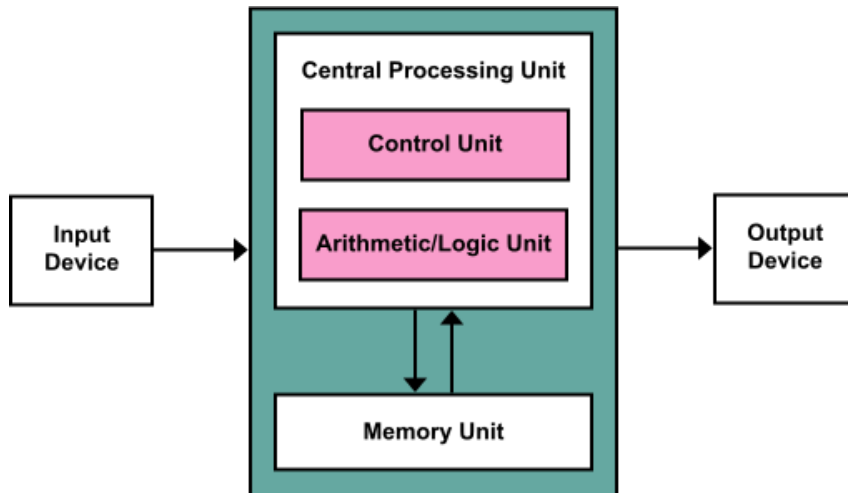
Compilateur



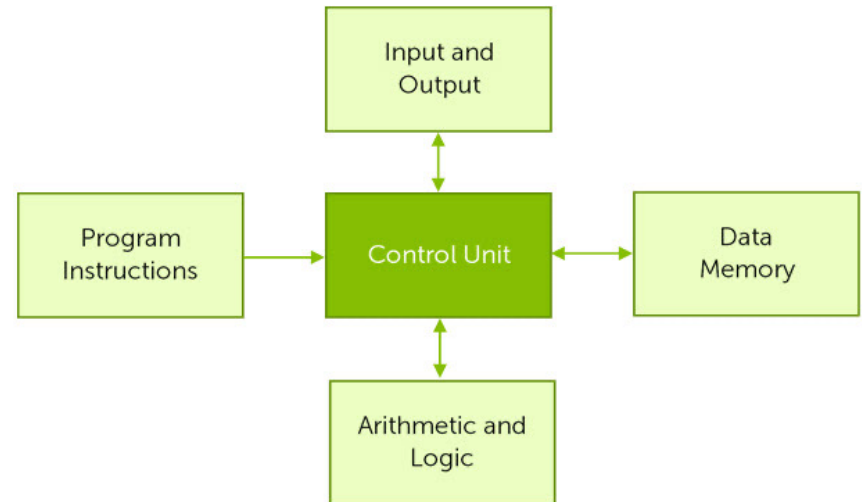
- Architecture
 - Von Neumann
 - Harvard
- Registre
- Nombre de bits
 - 8 / 16 / 32 / 64
- Ensemble d'instructions

Architecture

Von Neumann

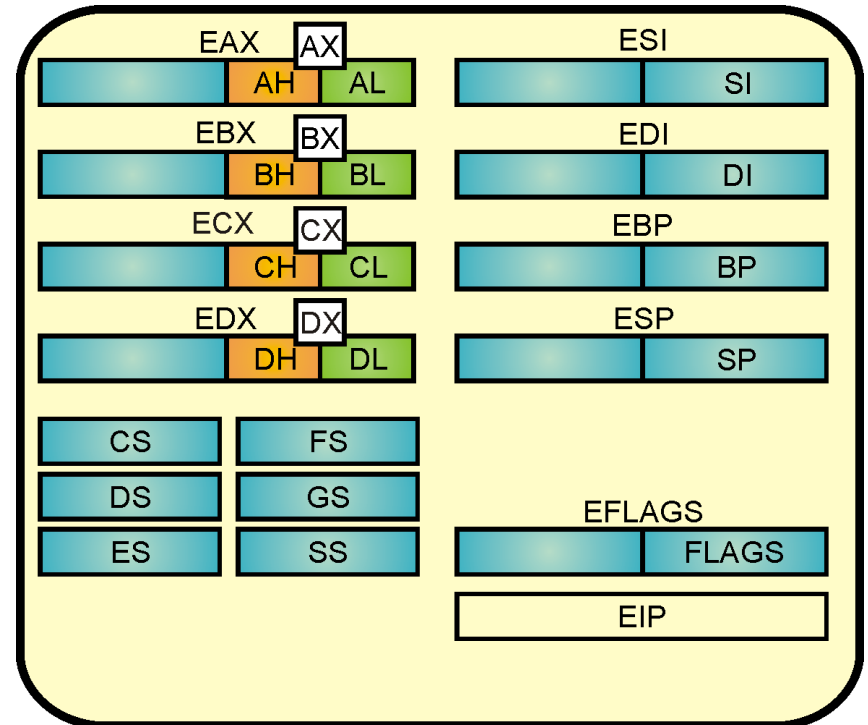


Harvard



Registre

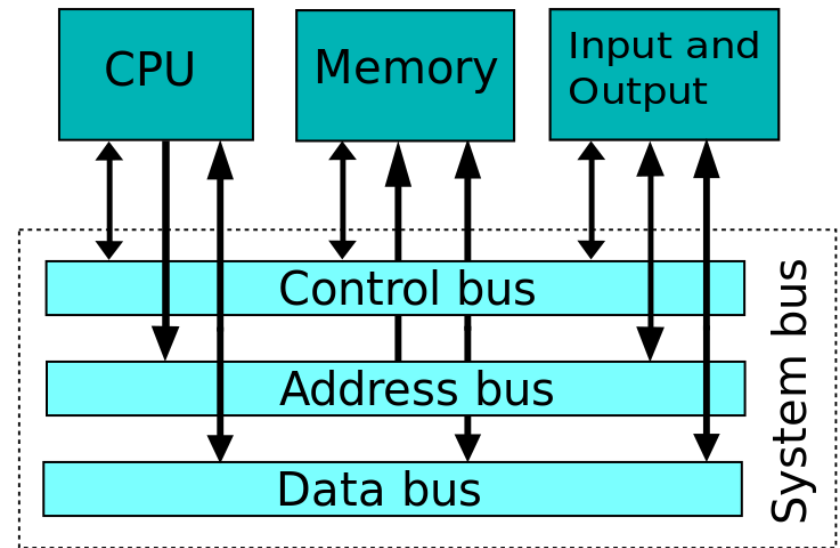
- « *Variables à l'intérieur* » du CPU
- Spécialise
- Utilisation général



Registres Intel 386

Nombre de bits

- Mot
 - *word size*
- Dimension de registre
- Largeur du bus de données
 - adresse (pointer)
 - donnees
- rétro compatible



Reduced Instruction Set

- Petit nombre d'instructions
- Utilise Moins de puissance
- Le programmeur a plus de souplesse
- Écrivez plus de code (ASM)

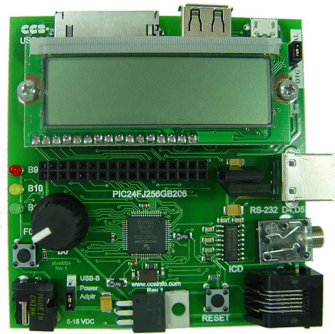
Complex Instruction Set

- Un grand nombre d'instructions
- Utilise Plus de puissance
- Moins de flexibilité de programmation
- Écrivez moins de code

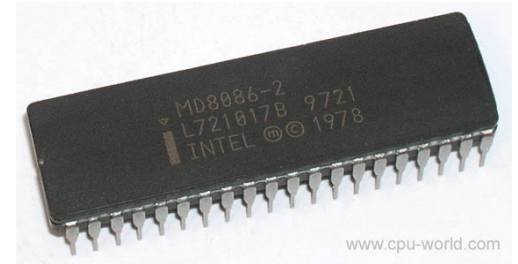
Exemple



Arduino (ATmega328)
Harvard
8 bits



Pic24
Harvard
16 bits



Intel 8086 (reference)
Von Neumann
16 bits

Exemple



GForce (GPGPU)
Harvard (modified)
192 bits



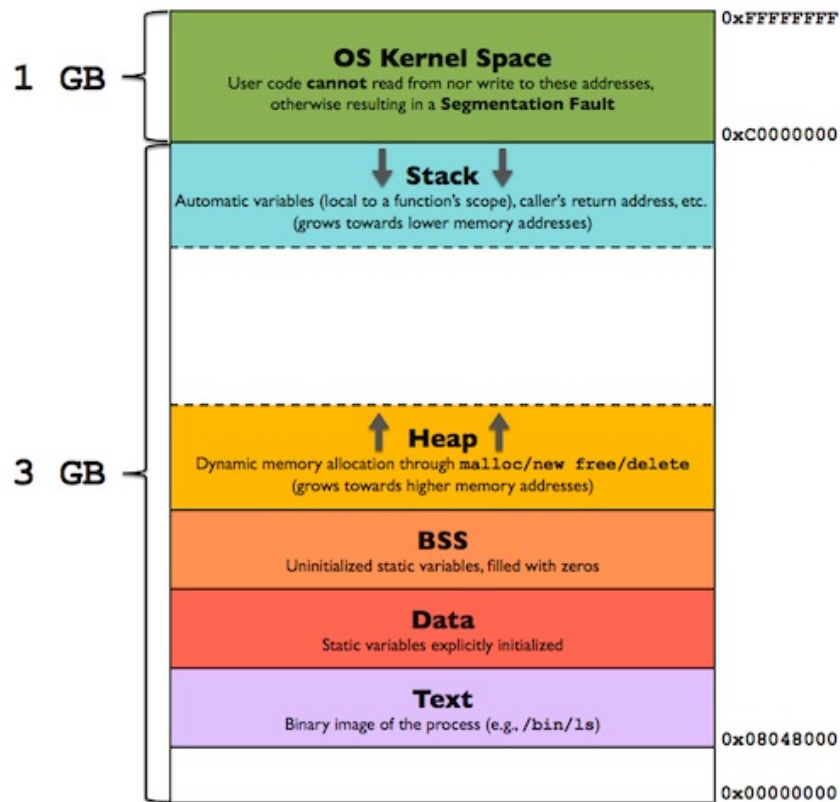
Snapdragon (ARM)
Von Neumann
32 / 64 bits



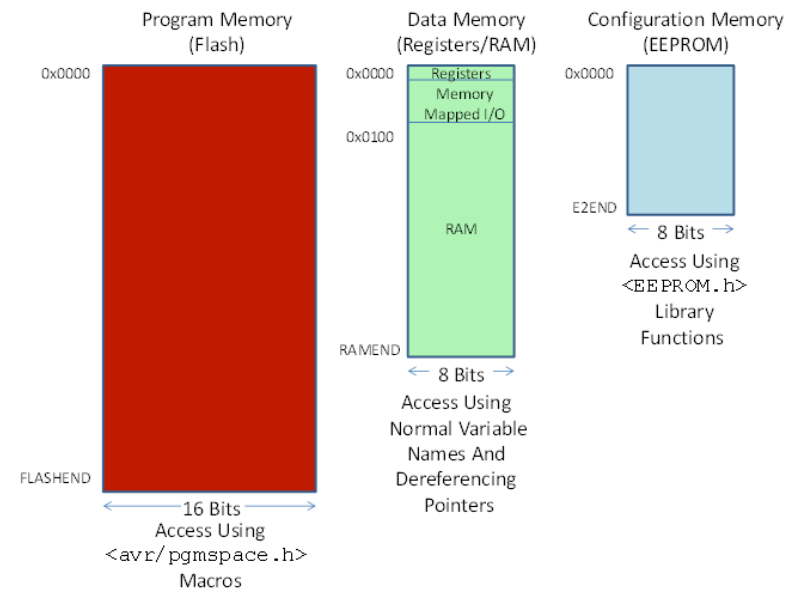
Intel Core i7
Von Neumann
32 / 64 bits

Memoire

Von Neumann



Harvard



- Simple
 - Numéro
 - Numéro avec virgule
 - Caractère
 - Booléen
 - String
 - Pointer
- Compose
 - Struct
 - Array

- **char** 8 bits
- **short** 16 bits
- **int** mot (32 / 64 bits)
- **long** 32 bits

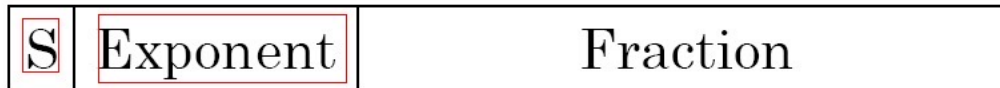
Numéro avec virgule

- **IEEE 754**
- **float** 32 bits
 - floating point
- **double** 64 bits
 - double precision

IEEE FLOATING-POINT FORMAT

single: 8 bits
double: 11 bits

single: 23 bits
double: 52 bits



$$x = (-1)^S \times (1 + \text{Fraction}) \times 2^{(\text{Exponent} - \text{Bias})}$$

<http://blog.csdn.net/xiabodan>

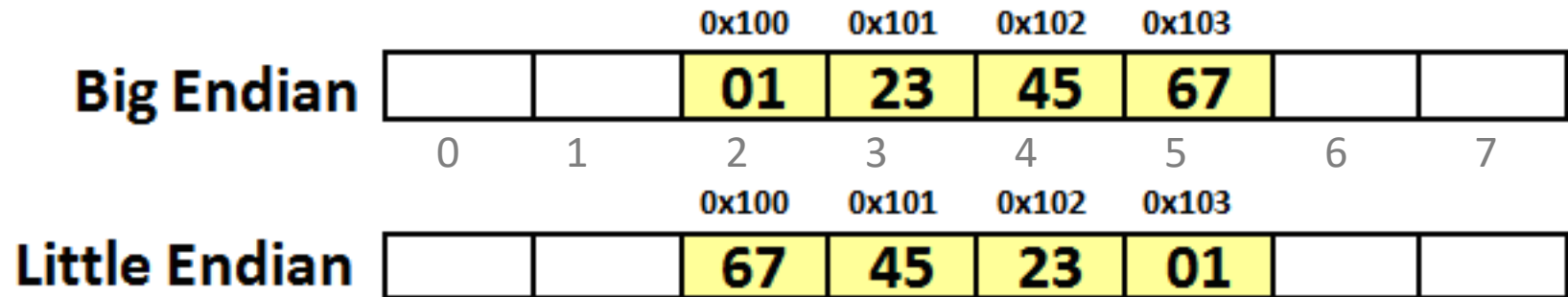
- Exponent: excess representation: actual exponent + Bias
 - Ensures exponent is unsigned
 - Single precision: Bias = 127;
 - Double precision: Bias = 1023



Nombre d'octets multiples

Numéro 0x01234567

Bytes 4



Adresse 0x0

Adresse 0xFFFFF...

Character

- C/C++
 - **char** 8 bits (ASCII)
- Java
 - **char** 16 bits (Unicode)

- C/C++
 - **char** 8 bits
 - TRUE != 0
 - FALSE - 0
- Java
 - **boolean** 8 - 16 bits (pas standard)
 - true
 - false
- Optimisation
 - fusionner plus boolean

String

- C style
- Pascal
- Java

String - C Style

char s[N]

N



s	t	r	i	n	g	\0	...
---	---	---	---	---	---	----	-----

115	116	114	105	110	103	0	...
-----	-----	-----	-----	-----	-----	---	-----

String - Pascal

s:string

256



\5	s	t	r	i	n	g	...
----	---	---	---	---	---	---	-----

5	115	116	114	105	110	103	...
---	-----	-----	-----	-----	-----	-----	-----

String - Java

String s

value		s		t		r		i		n		g	...
length	0	0	0	5									

value	0	11	0	11	0	11	0	10	0	11	0	10	...
length	0	0	0	5		4		5		0		3	

- C/C++, Pascal
 - mot (adresse)
- Java
 - numéro (référence)

Struct

```
{  
    short s;  
    float f;  
}
```



- Type de données
- Longueur
 - Une dimension
 - Multiple dimensions
- Bits pour les données * longueur
- **long l[35][35] – 64 * 35 * 35 bits**

Row / Column Major

1	2	3
4	5	6
7	8	9



row-major

1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---

1	2	3
4	5	6
7	8	9



column-major

1	4	7	2	5	8	3	6	9
---	---	---	---	---	---	---	---	---

- Les données sont transférées dans les tailles des mots
- Adresse
 - 0 .. 3
 - 4 ... 7
 - 8 ... 11
 - ...

Alignement

```
int n;  
char c;  
long l[2];  
char s[6];
```

20	s	s		
16	s	s	s	s
12	l	l	l	l
8	l	l	l	l
4	c			
0	n	n	n	n

- CPU
- Mémoire
- Format des données
 - Numéro
 - Numéro avec virgule
 - String
 - Array
 - Struct

Questions

