

# **INTRODUCTION**

## Structure de base d'un ordinateur

# Rôle d'un ordinateur

- Exécuter un programme de traitement sur des données en vue de résoudre un problème.
- Deux aspects :
  - La **structure de l'ordinateur** qui doit être composé d'éléments permettant le stockage, le traitement , la lecture ou l'écriture des données
  - L'expression du problème à résoudre, de sa solution dans un **langage compréhensible par l'ordinateur**

# A quoi sert donc un ordinateur ?

1.



J'ai un **problème** à résoudre !

b

a

Périmètre ?

2.



J'écris une **solution** !



**ALGORITHME**

Périmètre :=  $2a + 2b$

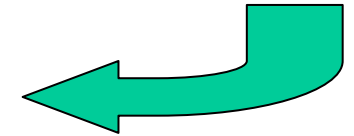


3.

En utilisant un **langage de programmation**, je code la solution pour la faire exécuter par l'ordinateur

→ **PROGRAMME** constitué d'instructions

```
fonction perimetre (a, b : in integer) return
integer is
begin
    perimetre := (2 * a) + (2 * b);
end;
```



# Le codage d'un problème ...

```
fonction perimetre (a, b : in integer) return  
integer is  
begin  
    perimetre := (2 * a) + (2 * b);  
end;
```

Programme en langage de haut niveau  
instructions de haut niveau



Compilateur

Niveau utilisateur

Système d'exploitation

*Gérer et partager le matériel*

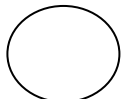
Machine physique "matérielle"

Programme à  
exécuter : instructions machine  
et valeurs **en binaire**

```
01101110111110011  
01111010001011100  
10111011101111111  
00111011110111011  
00111111000111101
```

Mémoire centrale

processeur



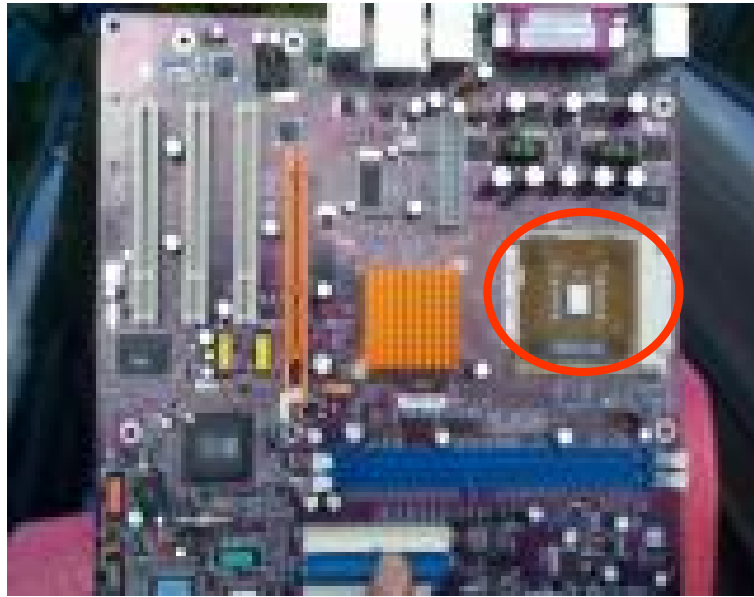
Bus



traduction

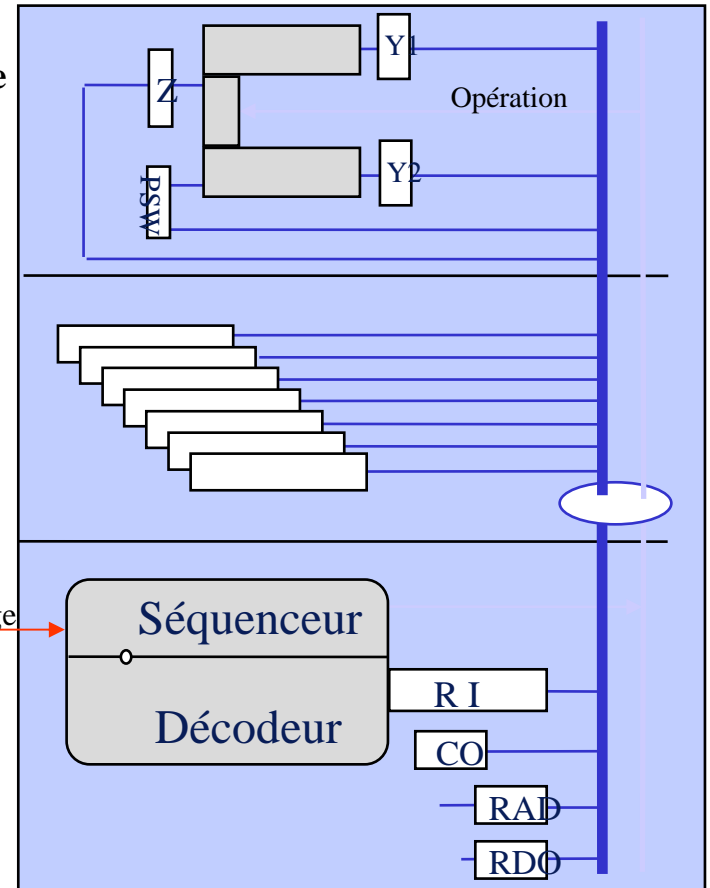


# Composants de l'ordinateur

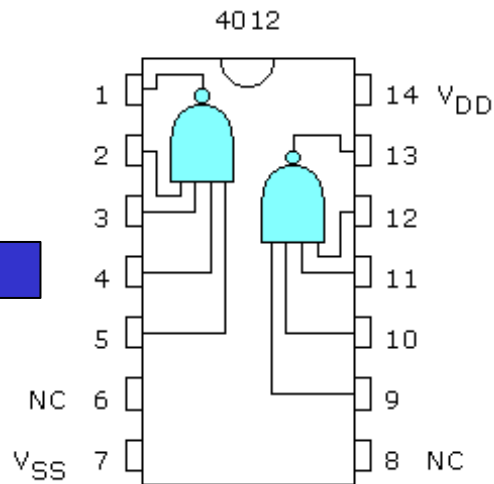


Unité Arithmétique  
et logique

Registres



TRANSISTORS



CIRCUITS LOGIQUES OU INTEGRES (PORTES)

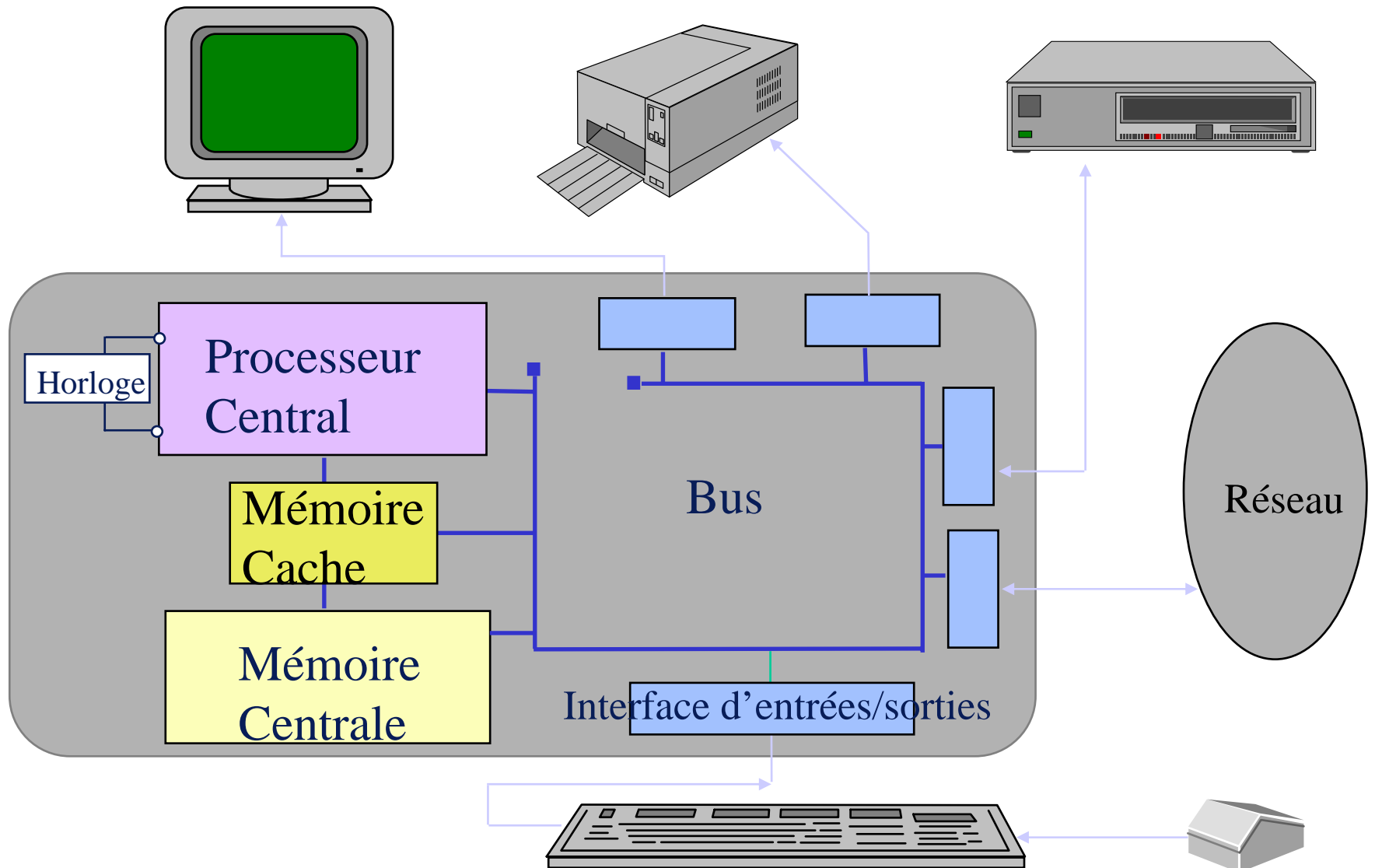
# La représentation des informations sur la machine physique

- La donnée de base manipulée par la machine physique est le **bit** (*Binary Digit*) qui ne peut prendre que deux valeurs : 0 et 1
- Ce 0 et 1 correspondent aux deux niveaux de voltage (0-1 et 2-5 volts) admis pour les signaux électriques issus des composants électroniques (transistors) qui constituent les circuits physiques de la machine
- Toutes les informations (nombres, caractères et instructions) ne peuvent être représentées que par une combinaison de 0 et 1 : **chaîne binaire**. Un **octet** est une chaîne de 8 bits.

# Structure générale de l'ordinateur

- L'ordinateur est composé principalement des éléments suivants :
  - Des éléments permettant la communication entre l'ordinateur et l'être humain : ce sont les **périphériques**.
  - Un élément permettant d'exécuter les instructions d'un programme : c'est le **processeur** (CPU).
  - Des éléments permettant de stocker les données : ce sont les **mémoires** de l'ordinateur.
  - Des éléments permettant aux différents composants (périphériques, processeur, mémoire) de l'ordinateur de communiquer : ce sont les **bus** de l'ordinateur

# Structure générale de l'ordinateur

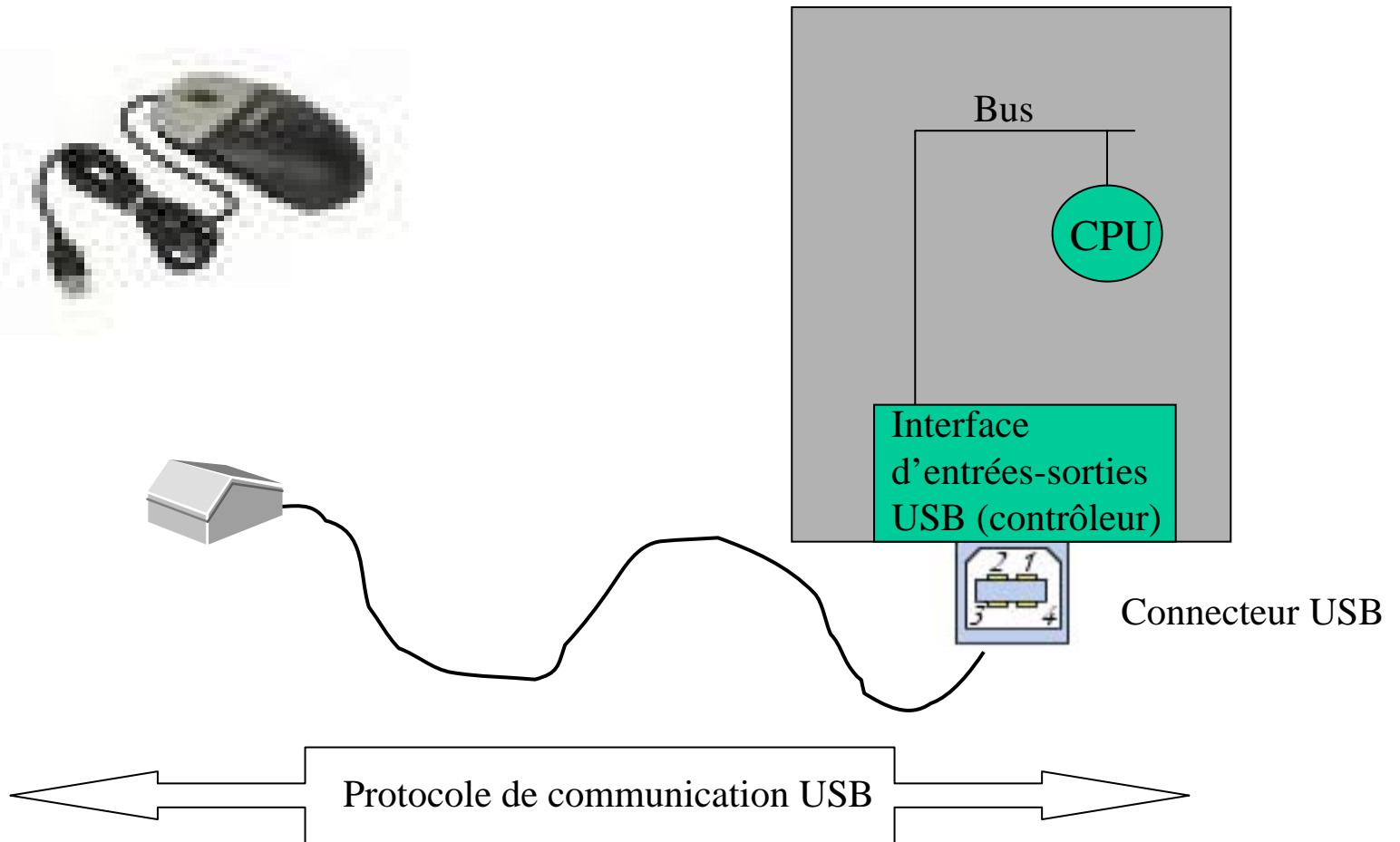




# Périphériques de l'ordinateur

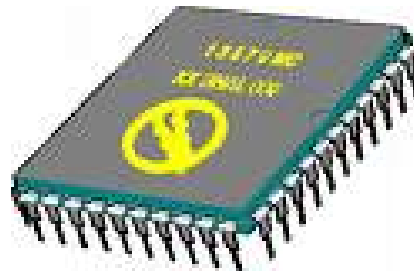
- Un périphérique est un matériel électronique pouvant être raccordé à un ordinateur par l'intermédiaire de l'une de ses **interfaces d'entrée-sortie** (interface série, parallèle, USB, etc.), le plus souvent par l'intermédiaire d'un **connecteur**.
- On distingue habituellement les catégories de périphériques suivantes :
  - **périphériques de sortie**: ce sont des périphériques permettant à l'ordinateur démettre des informations vers l'extérieur, tels qu'un écran, une imprimante..
  - **périphériques de stockage** : il s'agit d'un périphérique capable de stocker les informations de manière permanent (disque dur, CD-ROM, DVD)
  - **périphériques d'entrée** : ce sont des périphériques capables uniquement d'envoyer des informations à l'ordinateur, par exemple la souris, le clavier, etc.

# Périphériques de l'ordinateur : souris USB



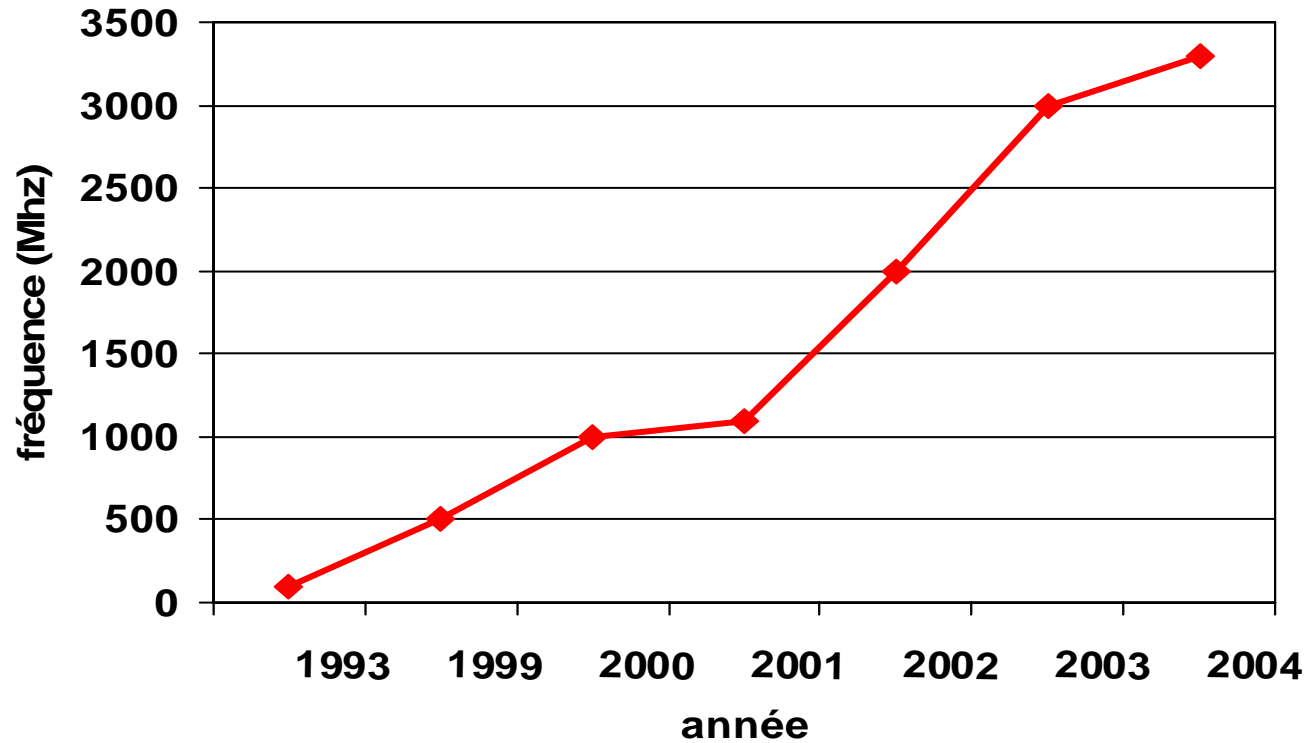
# Le processeur

- Le **processeur** (**CPU**, pour *Central Processing Unit*) est le cerveau de l'ordinateur. Il permet de manipuler, des données et des instructions codées sous forme binaires.
- Le **processeur** est un circuit électronique cadencé au rythme d'une horloge interne qui envoie des impulsions, appelées « **top** ». La **fréquence d'horloge**, correspond nombre d'impulsions par seconde. Elle s'exprime en Hertz (Hz).
- Ordinateur à 200 MHz → l'horloge envoie 200 000 000 de battements par seconde.



Circuits électroniques composés de millions de transistors placés dans un boîtier comportant des connecteurs d'entrée-sortie  
→ **circuit intégré** ou **puce**

# Le processeur : performances



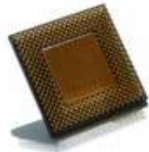
**1971 : 2300 transistors** → **2005 : 230 millions de transistors**

# Les mémoires de l'ordinateur

- Une « **mémoire** » est un composant électronique capable de stocker temporairement des informations.
- Une mémoire est caractérisée par :
  - Sa **capacité**, représentant le volume global d'informations (en bits) que la mémoire peut stocker (par exemple 1 Goctets, soit  $2^{30}$  octets, soit  $2^{30} * 8$  bits).
  - Son **temps d'accès**, correspondant à l'intervalle de temps entre la demande de lecture/écriture et la disponibilité de la donnée.
- L'ordinateur contient différents niveaux de mémoire, organisés selon une **hiérarchie mémoire**.

# Les mémoires de l'ordinateur

- L'ordinateur contient différents niveaux de mémoire, organisés selon une **hiérarchie mémoire**.

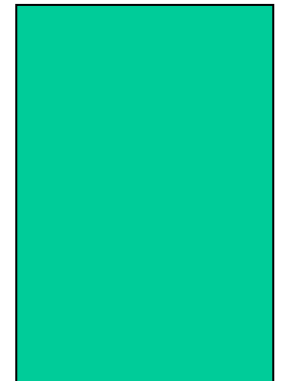
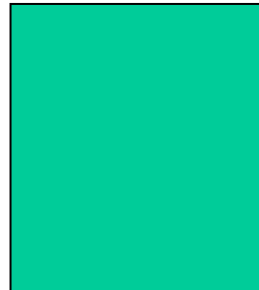


**REGISTRES**  
N bits (32, 64)  
1 nanoseconde

**Mémoires Caches**  
Koctets  
5 nanosecondes

**Mémoires Centrales**  
Goctets  
10 nanosecondes

**Mémoires de masse**  
100 - 200 Goctets  
5 millisecondes



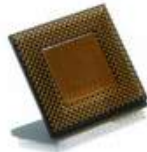
# Les mémoires de l'ordinateur

Mémoires vives : mémoires **volatiles**

Mémoires de masse :  
mémoires **permanentes**

Barrettes mémoire  
SIMM, DIMM...

Plateaux magnétiques

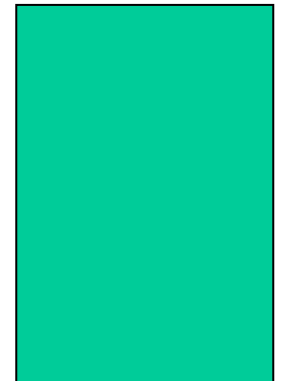
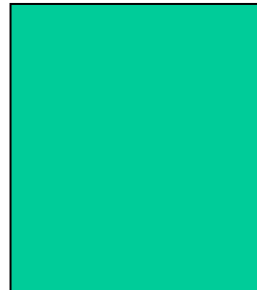


**REGISTRES**  
N bits (32, 64)  
1 nanoseconde

**Mémoires Caches**  
Koctets  
5 nanosecondes

**Mémoires Centrales**  
Goctets  
10 nanosecondes

**Mémoires de masse**  
100 - 200 Goctets  
5 millisecondes



# Les bus de l'ordinateur

- Un « **bus** » est un composant électronique permettant à différents composants de l'ordinateur de s'échanger des informations.
  - Bus système (*Front Side Bus FSB*) permet la communication entre le processeur et la mémoire centrale.
  - Bus d'extension permet aux autres éléments de l'ordinateur de communiquer entre eux.

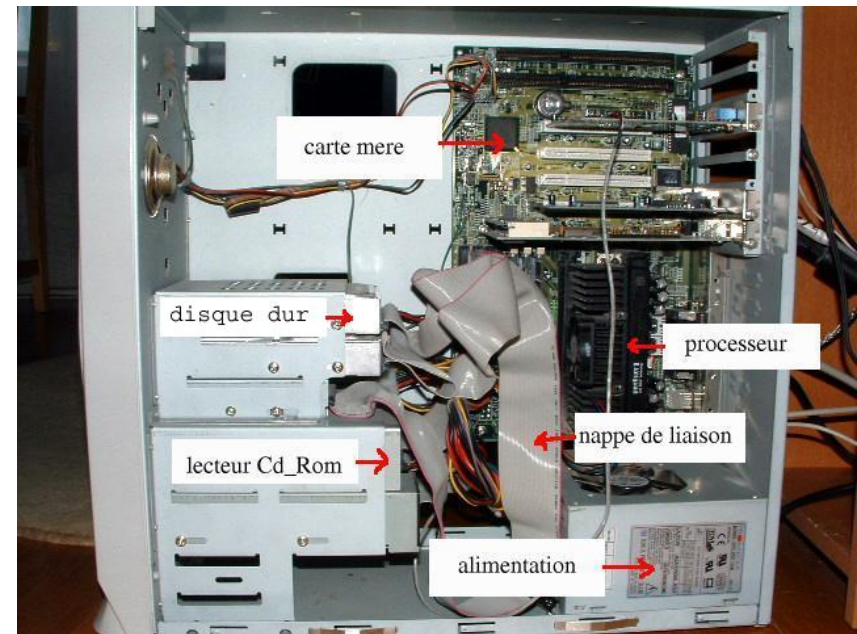


# Structure générale de l'ordinateur

- Le **boîtier** (ou *châssis*) de l'ordinateur est le squelette métallique abritant ses différents composants internes. L'ensemble, boîtier et composants internes, forment **l'unité centrale** de l'ordinateur.

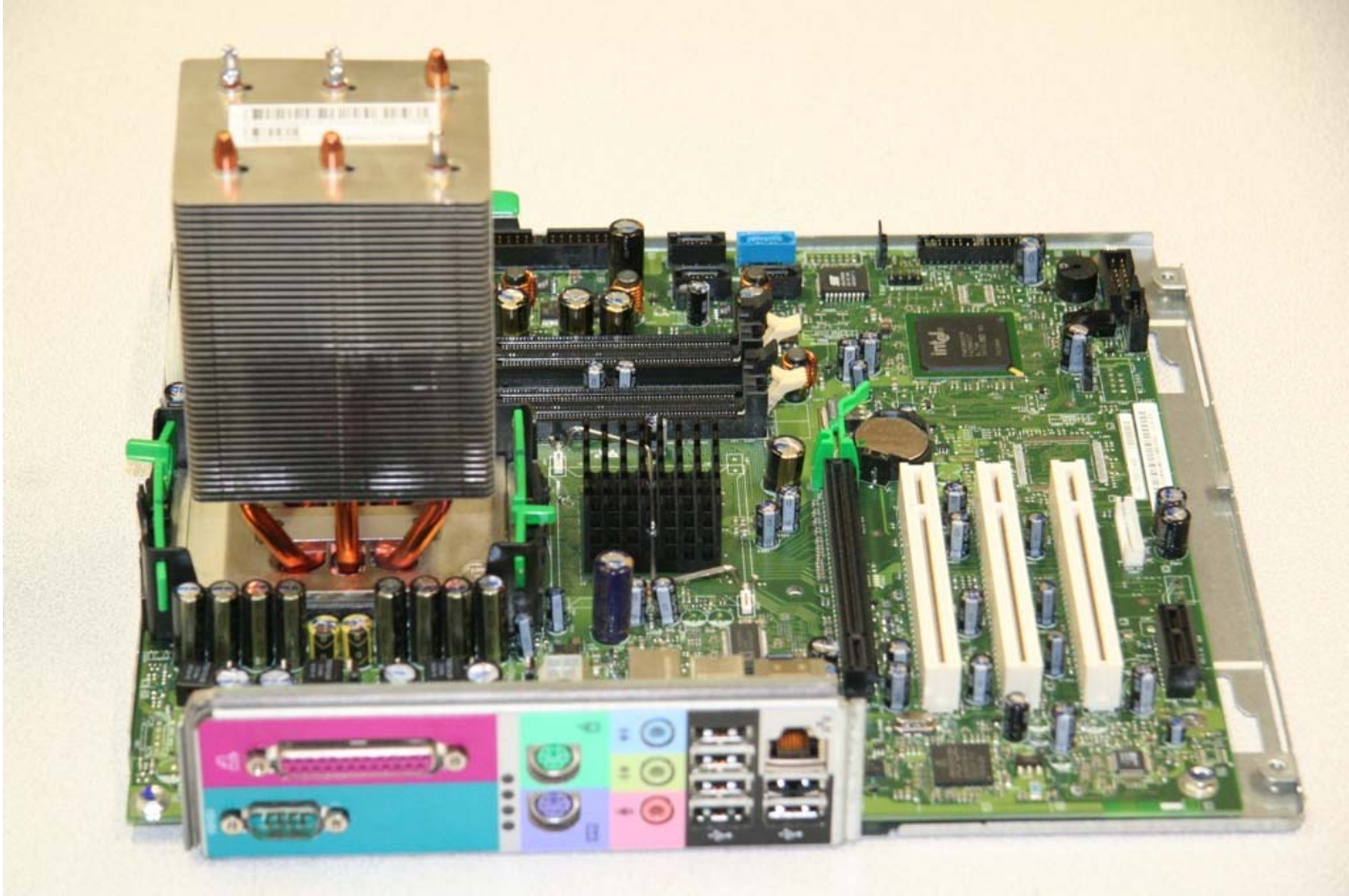
Le boîtier contient :

- La **carte mère** de l'ordinateur;
- Des périphériques de stockage tels que disques, dur, lecteur-graveur DVD.CD-ROM;
- Des cartes d'extensions pour les interfaces d'entrées sorties;
- Un bloc d'alimentation électrique.



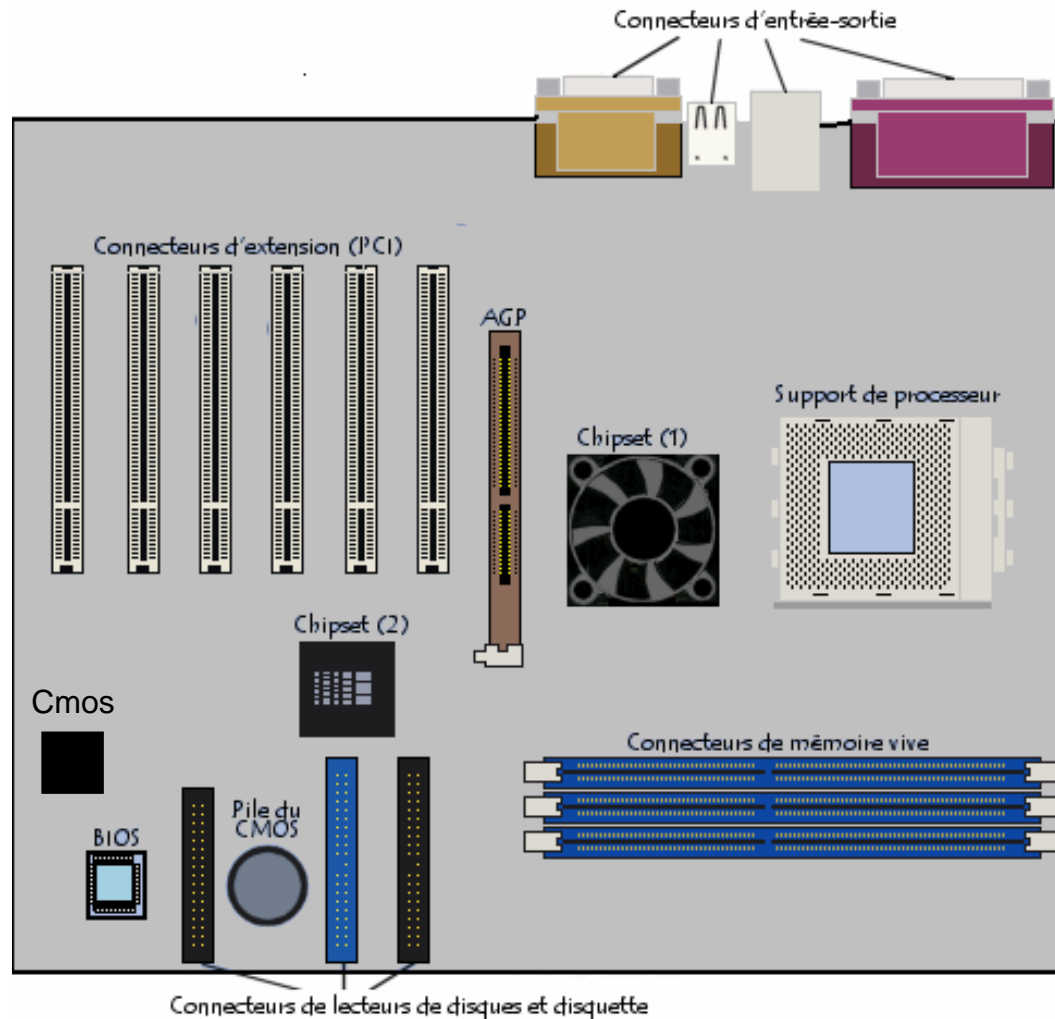
# Structure générale de l'ordinateur

- La **carte mère** de l'ordinateur est le socle permettant la connexion de l'ensemble des éléments essentiels de l'ordinateur.



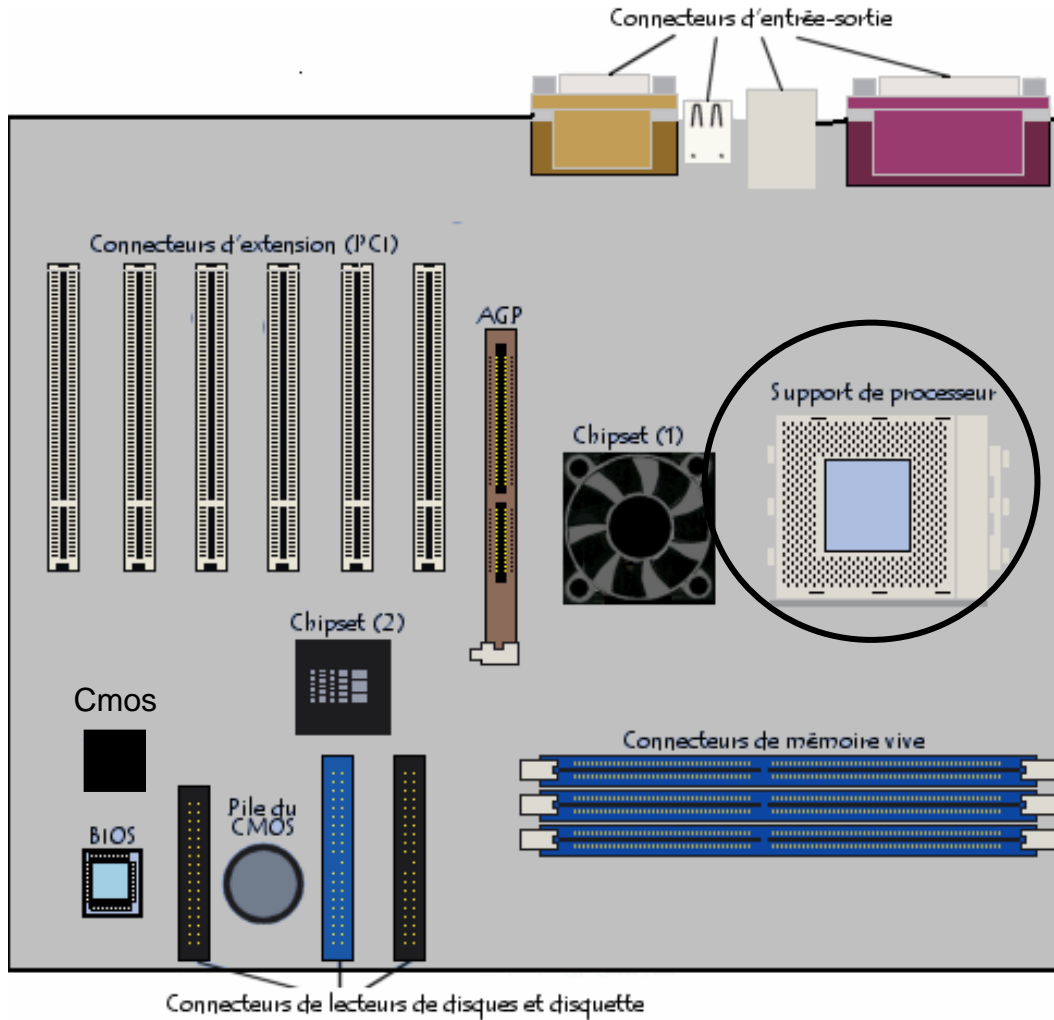
# Structure générale de l'ordinateur

- La **carte mère** de l'ordinateur est le socle permettant la connexion de l'ensemble des éléments essentiels de l'ordinateur.

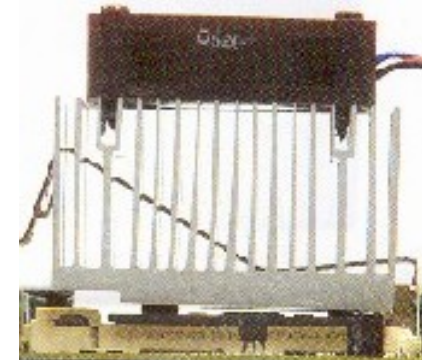


Cette image conçue par Jean François Pillou, tirée de l'encyclopédie informatique Comment ça marche (<http://www.commentcamarche.net>) est mis à disposition sous les termes de la licence Creative Common.

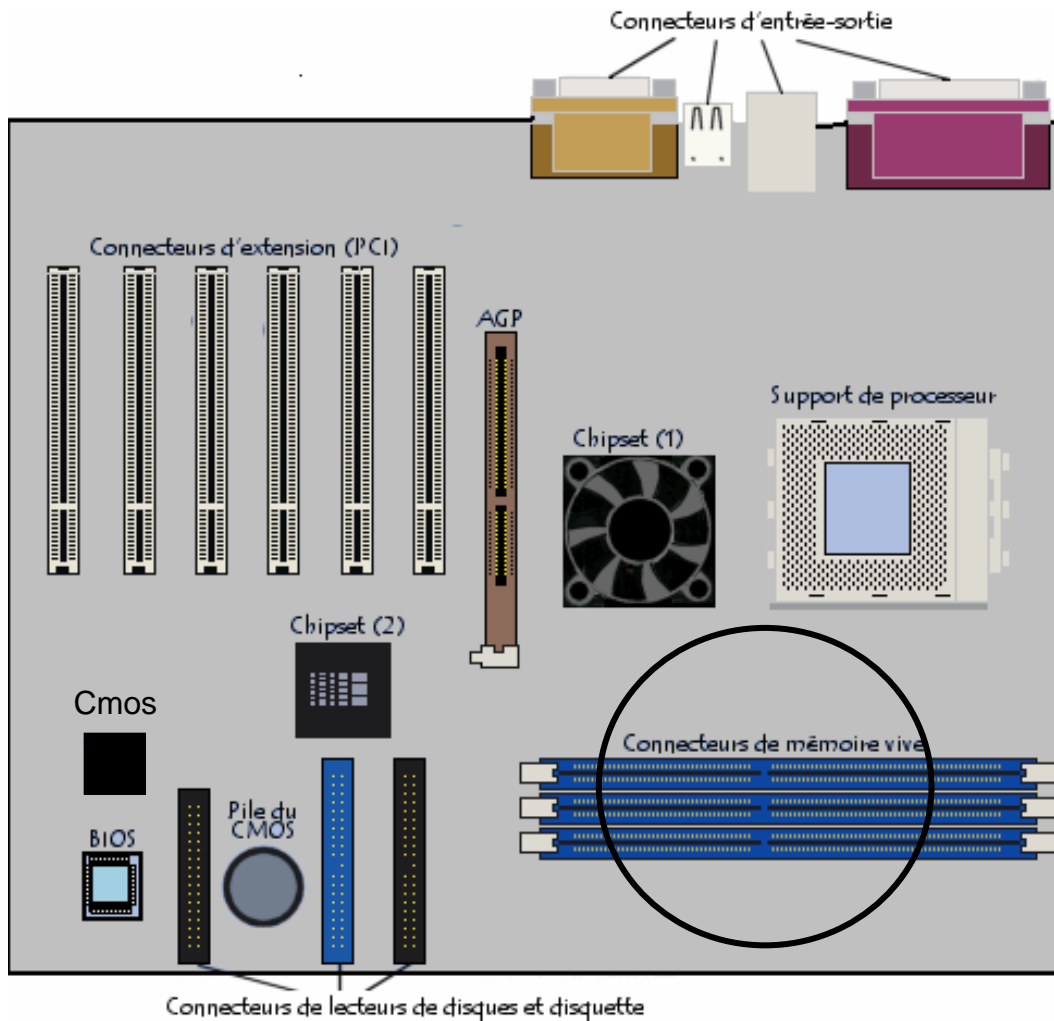
# Structure générale de l'ordinateur



- La **socket** est un connecteur carré sur lequel le processeur vient s'enficher.
- Le processeur est surmonté d'un dissipateur thermique et d'un ventilateur (**ventirad**) qui dissipent la chaleur émise par les composants électroniques.

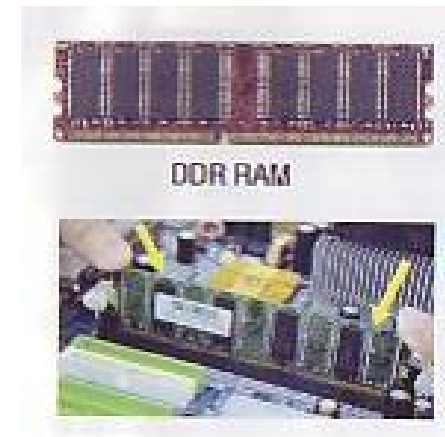


# Structure générale de l'ordinateur

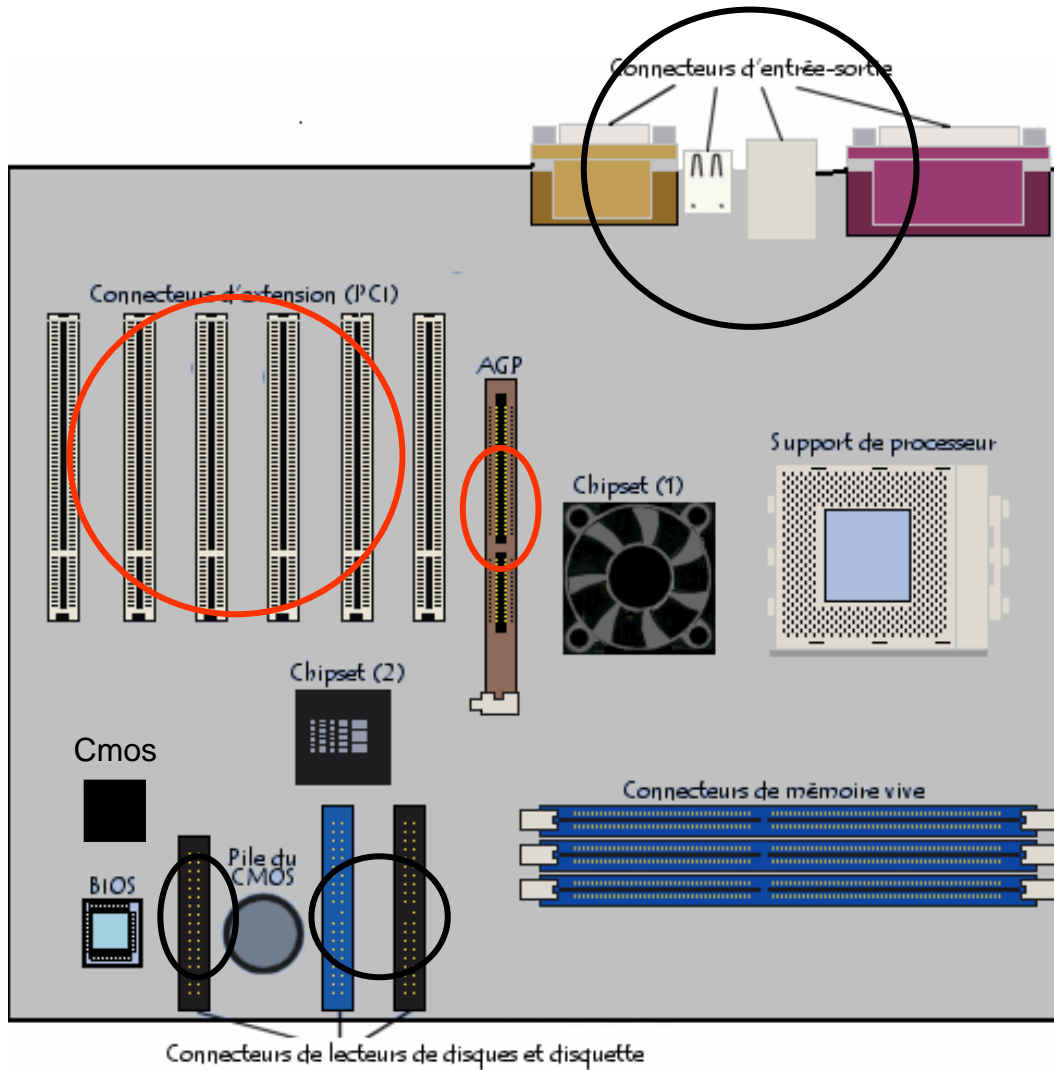


- La mémoire vive se présente sous forme de puces enfilées sur des barrettes qui se branchent sur les connecteurs de la carte mère

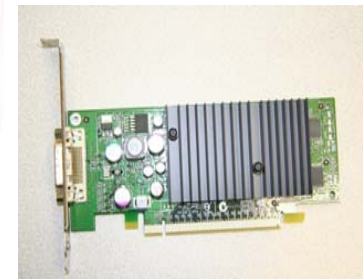
➤ Barrettes DIMM (*Dual Inline Memory Module*) .



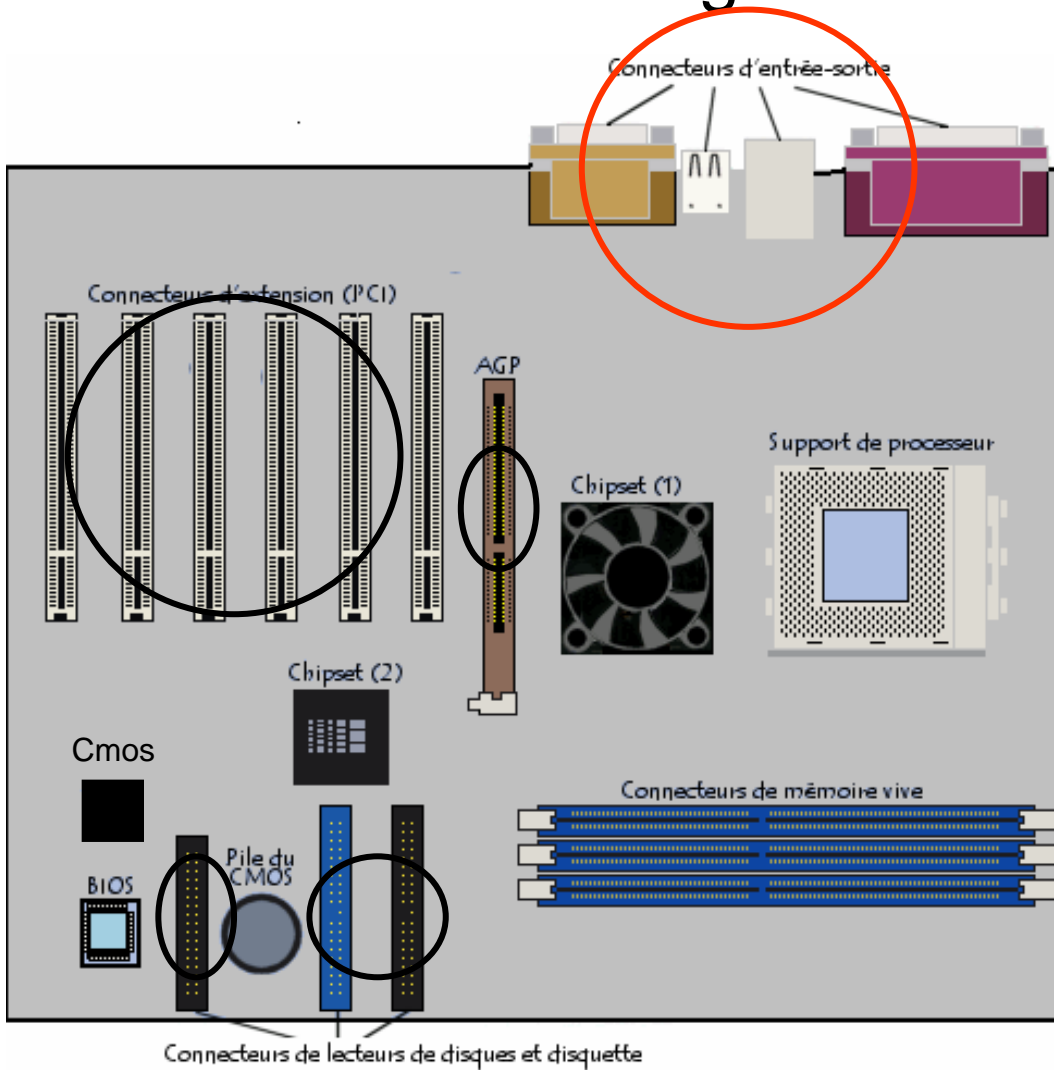
# Structure générale de l'ordinateur



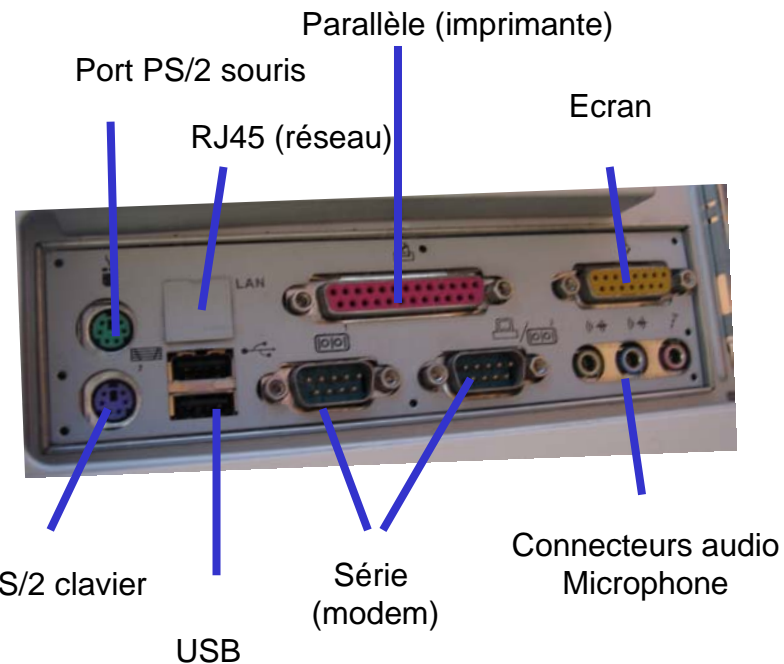
- La carte mère présente un ensemble de connecteurs :
  - Connecteurs d'extension (slots) permettant d'enficher des cartes apportant des fonctionnalités supplémentaires
    - PCI (Peripheral Component Interconnect)
    - AGP (Accelerated Graphic Port) : carte graphique



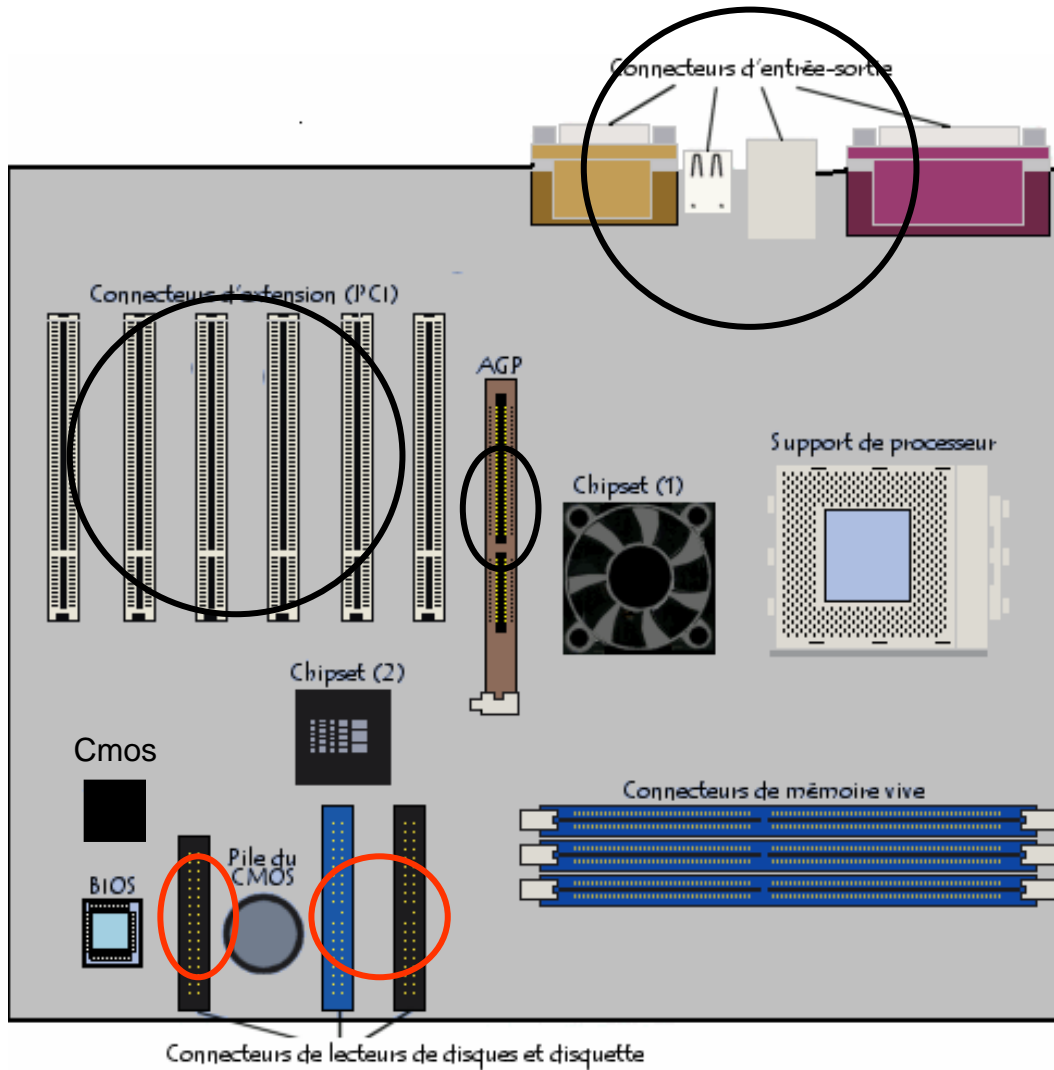
# Structure générale de l'ordinateur



- La carte mère présente un ensemble de connecteurs :
- Connecteurs d'entrées-sorties



# Structure générale de l'ordinateur

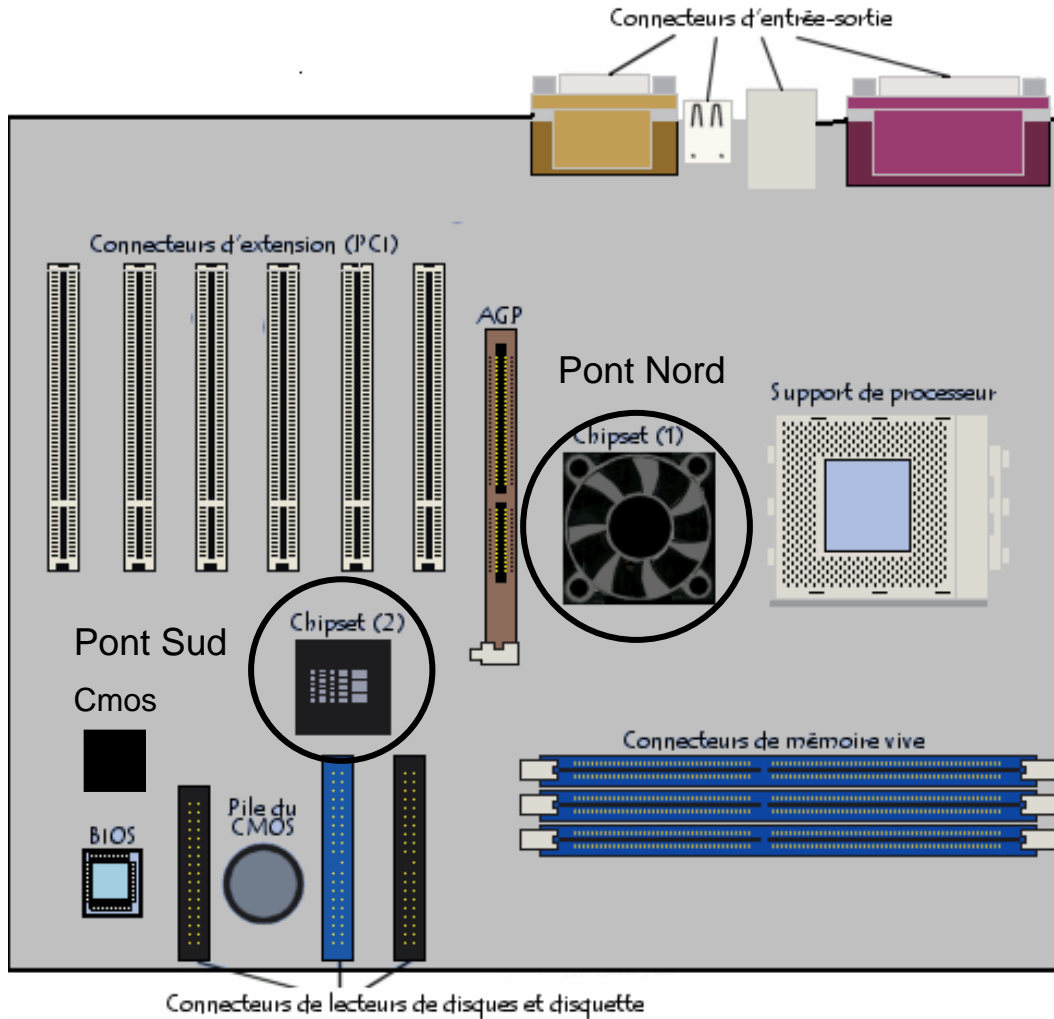


- La carte mère présente un ensemble de connecteurs
- Connecteurs des lecteurs de disques et disquettes reliant les disques de stockage via une **nappe**.



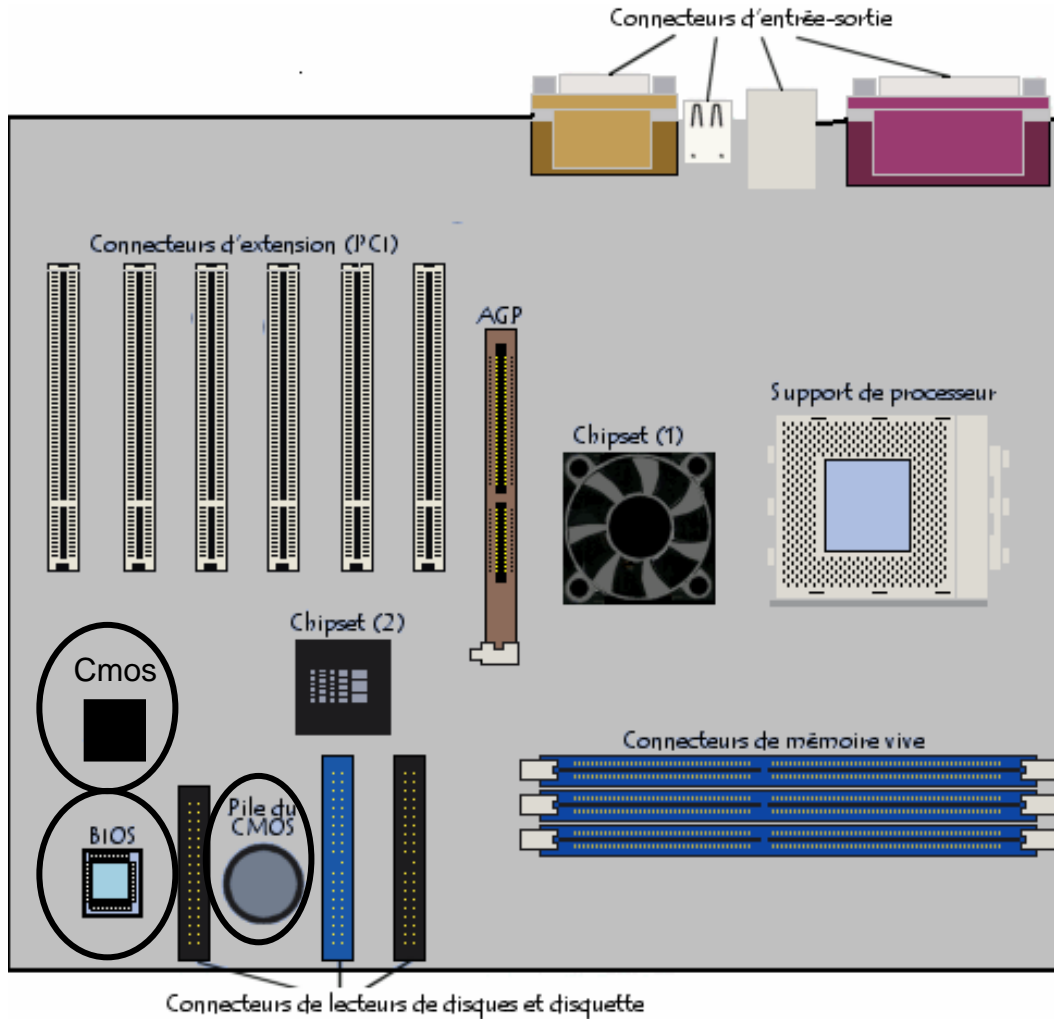


# Structure générale de l'ordinateur



- Le **chipset** est un ensemble de circuits chargés de coordonner les échanges entre les différents éléments de la carte mère (processeur, mémoire, cartes d'extension)
  - **Pont nord** gère notamment les échanges processeur – mémoire centrale
  - **Pont Sud** gère les échanges avec les cartes d'extension et les périphériques d'entrées-sorties
- ☑ *A noter : les deux ponts tendent à s'intégrer en un seul sur les cartes les plus récentes.*

# Structure générale de l'ordinateur



- Deux éléments conservent des données essentielles au démarrage de l'ordinateur :
  - Le **CMOS** alimenté par une pile fichée sur la carte mère contient la configuration matérielle et maintient l'heure de la machine
  - Le **BIOS** contient un programme minimal d'entrées-sorties stocké dans une mémoire non volatile et non modifiable (*Mémoire ROM Read Only Memory*).

- ☑ CMOS : Complementary Metal-Oxide Semiconductor
- ☑ BIOS : Basic Input Output System

# Démarrage de l'ordinateur



1. L'utilisateur appuie sur le bouton d'alimentation de l'unité centrale



2. Une fois le courant stabilisé, le processeur démarre et exécute le code du BIOS stocké dans la ROM à une adresse prédéfinie



3. Le BIOS exécute une séquence de vérification des composants (mémoire, vidéo, périphériques de base) (POST : Power-OnSelf Test)

4. Le BIOS accède au CMOS pour lire la configuration matérielle de la machine (date, heure, périphérique de masse contenant le système d'exploitation).

# **LE LANGAGE MACHINE**

## **1. Représentation des informations**

# Le codage d'un problème ...

```
fonction perimetre (a, b : in integer) return  
integer is  
begin  
    perimetre := (2 * a) + (2 * b);  
end;
```

Programme en langage de haut niveau  
instructions de haut niveau



Compilateur

Niveau utilisateur

Système d'exploitation

*Gérer et partager le matériel*

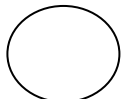
Machine physique "matérielle"

Programme à  
exécuter : instructions machine  
et valeurs **en binaire**

```
01101110111110011  
01111010001011100  
1011101110111111  
00111011110111011  
00111111000111101
```

Mémoire centrale

processeur



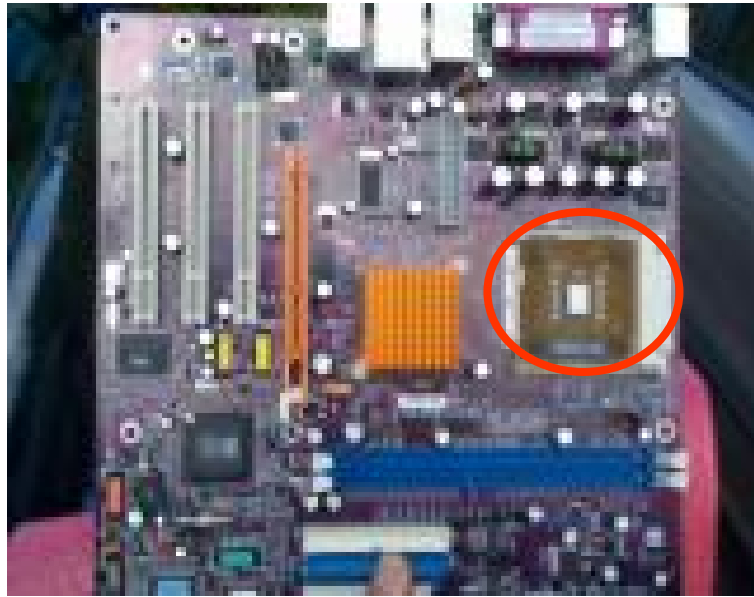
Bus



traduction



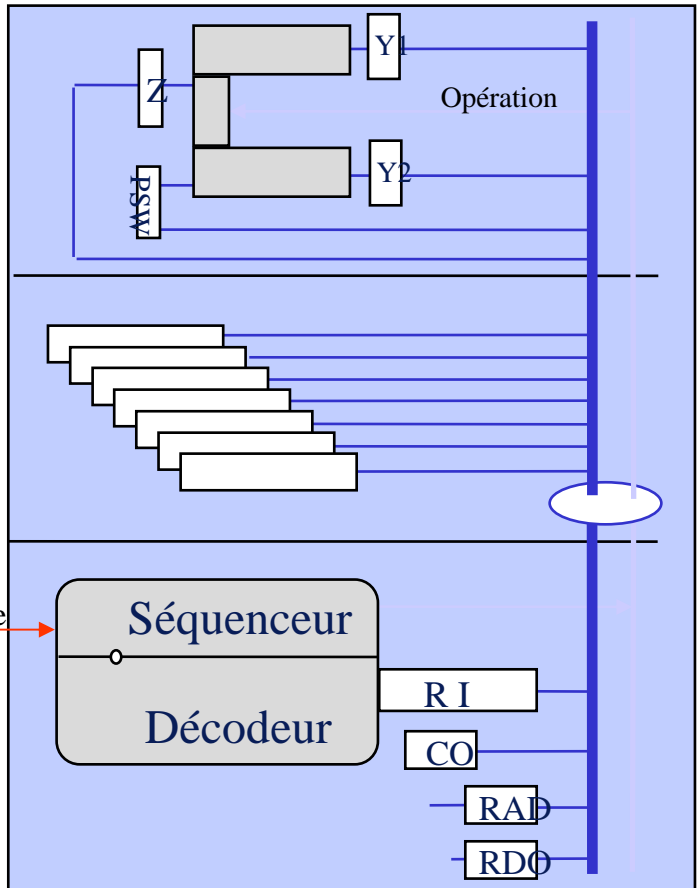
# Composants de l'ordinateur



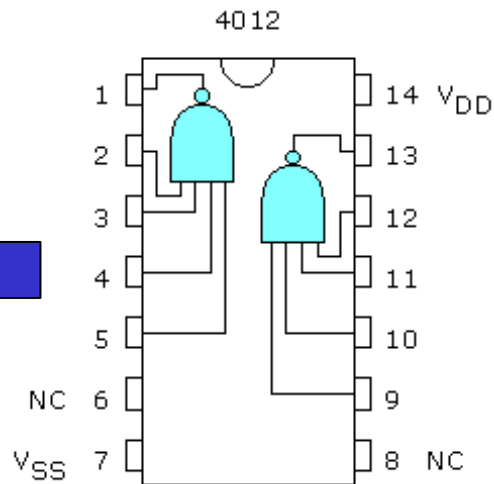
Unité Arithmétique  
et logique

Registres

horloge



TRANSISTORS

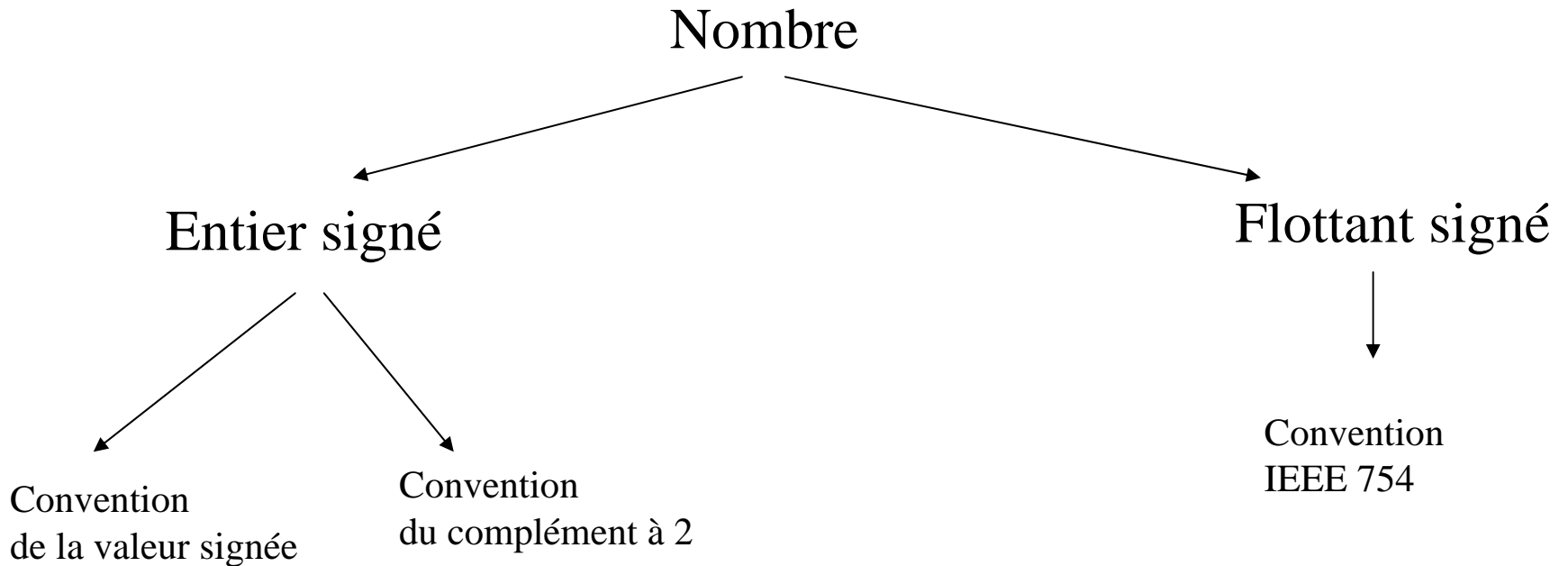


CIRCUITS LOGIQUES OU INTEGRES (PORTES)

# La représentation des informations sur la machine physique

- La donnée de base manipulée par la machine physique est le **bit** (*Binary Digit*) qui ne peut prendre que deux valeurs : 0 et 1
- Ce 0 et 1 correspondent aux deux niveaux de voltage (0-1 et 2-5 volts) admis pour les signaux électriques issus des composants électroniques (transistors) qui constituent les circuits physiques de la machine
- Toutes les informations (nombres, caractères et instructions) ne peuvent être représentées que par une combinaison de 0 et 1 : **chaîne binaire**. Un **octet** est une chaîne de 8 bits.

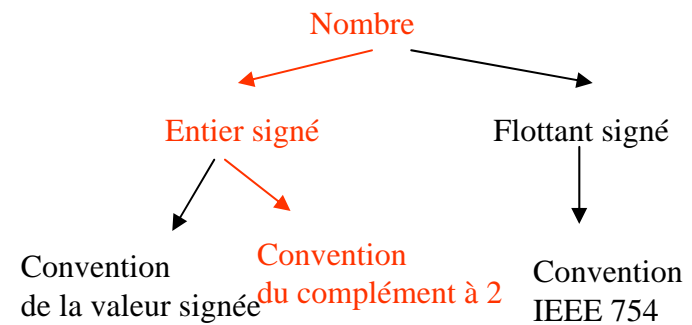
# La représentation des nombres sur la machine physique



La valeur d'un nombre signé est représentée par une **chaîne binaire de taille fixe** (n bits).  
en utilisant une **convention de représentation**



# La représentation des nombres sur la machine physique



Un nombre positif est représenté par son équivalent binaire sur n bits.  
Un nombre négatif est représenté en prenant le complément à 2 de son équivalent positif.

Exemple : représenter + 124 sur 8 bits en complément à 2

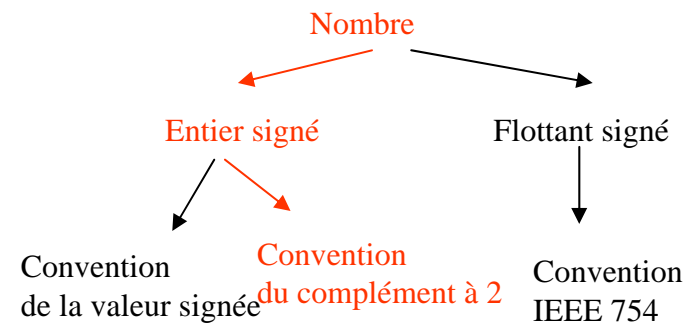
$$124 = 64 + 32 + 16 + 8 + 4 = 2^6 + 2^5 + 2^4 + 2^3 + 2^2 = 01111100$$

Exemple : représenter - 124 sur 8 bits en complément à 2

$$124 = 64 + 32 + 16 + 8 + 4 = 26 + 25 + 24 + 23 + 22 = 01111100$$

	01111100	
	10000011	inversion des bits
+	1	ajout de 1
<hr style="border-top: 1px dashed orange;"/>		
	10000100	= - 124

# La représentation des nombres sur la machine physique



Toute chaîne binaire commençant par un 1 est un nombre négatif : sur 8 bits ce sont les chaînes 11111111 (-1) à 10000000 ( - 128)

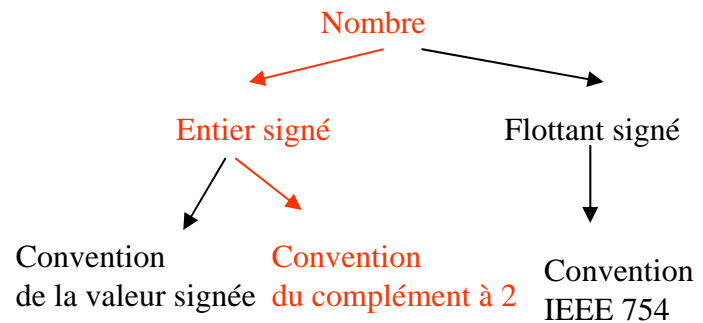
Toute chaîne binaire commençant par un 0 est un nombre positif : sur 8 bits ce sont les chaînes 01111111 (+127) à 00000000 ( + 0)

11111111
00000000
+           1
-----
00000001

11111111
-           1
-----
11111110
00000001

10000000
01111111
+           1
-----
10000000

# L'arithmétique de l'ordinateur



Exemple : additionner + 127 et + 2 avec une représentation en complément à 2 sur 8 bits

127	01111111
+ 2	+ 00000010
-----	
129	10000001

→ ce résultat est un nombre négatif (-127) et non la valeur 129  
le résultat est trop grand pour la machine : il y a **OVERFLOW**

Le résultat est faux !

Éventail des nombres représentables [- 128, 127]

# La représentation des caractères sur la machine physique

	0	1	2	3	4	5	6	7	Poids fort
0	NUL	DLE	sp.	0	█	P		p	0
1	SOH	DC1	!	1	A	Q	a	q	1
2	STX	DC2	█	2	B	R	b	r	2
3	ETX	DC3	█	3	C	S	c	s	3
4	EOT	DC4	█	4	D	T	d	t	4
5	ENQ	NAK	%	5	E	U	e	u	5
6	ACK	SYN	&	6	F	V	f	v	6
7	<b>BEL</b>	ETB	█	7	<b>G</b>	W	g	w	7
8	BS	CAN	(	8	H	X	h	x	8
9	HT	EM	)	9	I	Y	i	y	9
A	LF	SUB	*	:	J	Z	j	z	A
B	VT	ESC	+	;	K	█	k	█	B
C	FF	FS	,	<	L	█	l	█	C
D	CR	GS	-	=	M	█	m	█	D
E	SO	RS	.	>	N	█	n	█	E
F	SI	US	/	?	O	-	o	DEL	F
Poids faible	0	1	2	3	4	5	6	7	

G est codé  
47 hexa  
0100 0111

█ caractère non défini dans la norme (choix nationaux)

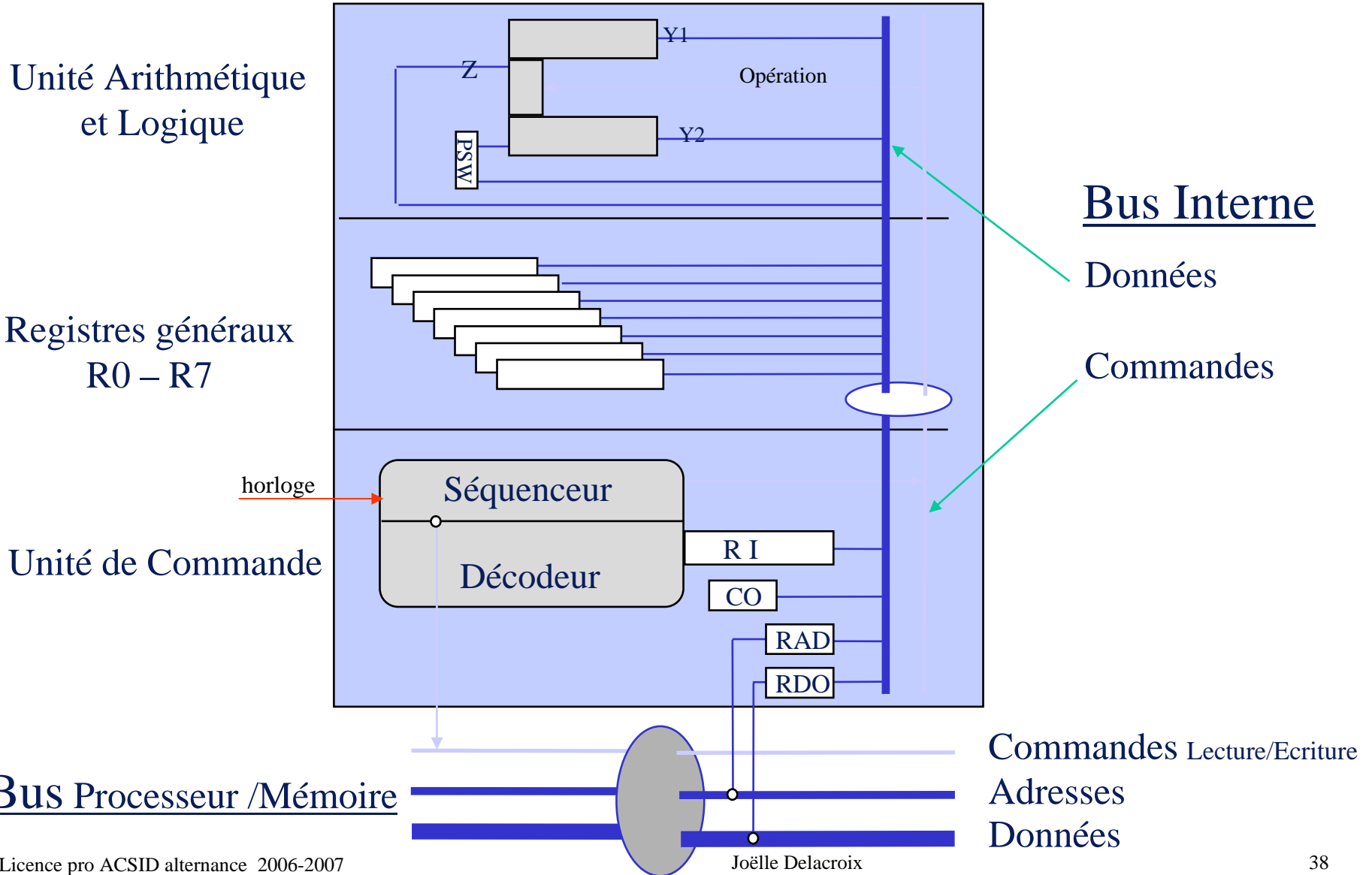
Figure II-6 - Table des codes A.S.C.I.I.

- Code ASCII
- Les codes 0 à 31 sont *caractères de contrôle*. Ils permettent de faire des actions telles que :
  - retour à la ligne (CR)
  - Bip sonore (BEL)
- Les codes 65 à 90 représentent les majuscules
- Les codes 97 à 122 représentent les minuscules

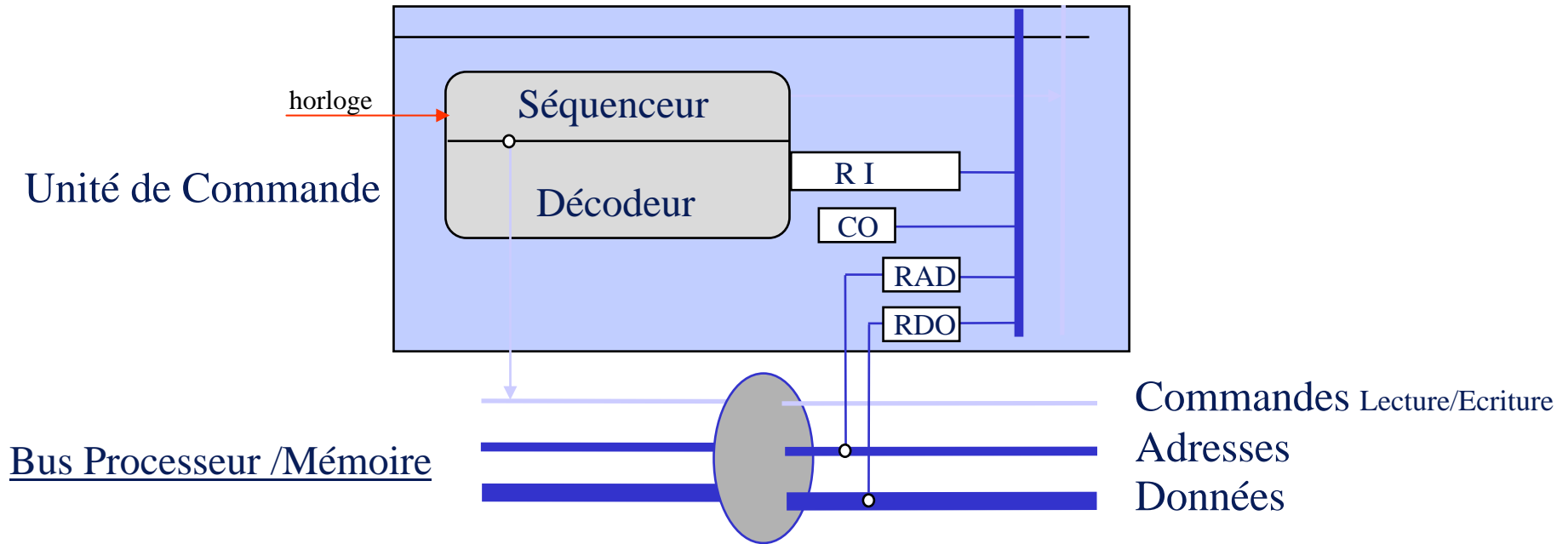
# Fonctionnement du processeur : exécution des instructions

- Structure du processeur et interfaçage avec la mémoire
- Exécution des instructions machines.
  - ↳ Principe de l'exécution d'une instruction
  - ↳ Rupture de séquence : les interruptions

# Processeur (Unité Centrale)



# Processeur (Unité Centrale)



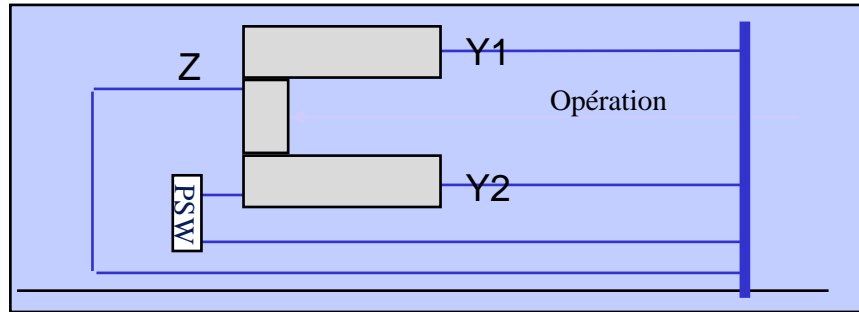
L'unité de commande est chargée de la reconnaissance des instructions et de leur exécution par le processeur au rythme de l'horloge

Les registres :

- **RI** (registre instruction) : contient l'instruction en cours d'exécution
- **CO** (compteur ordinal) : contient l'adresse en MC de la prochaine instruction
- **RAD** (registre adresse) et **RDO** (registre de données) : registres d'interfaçage avec la mémoire centrale

# Processeur (Unité Centrale)

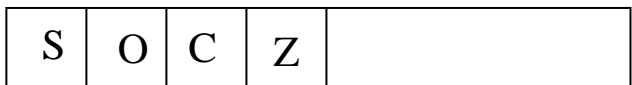
## Unité Arithmétique et Logique



L'unité Arithmétique et Logique (UAL) constitue l'unité d'exécution du processeur.

Elle est composée :

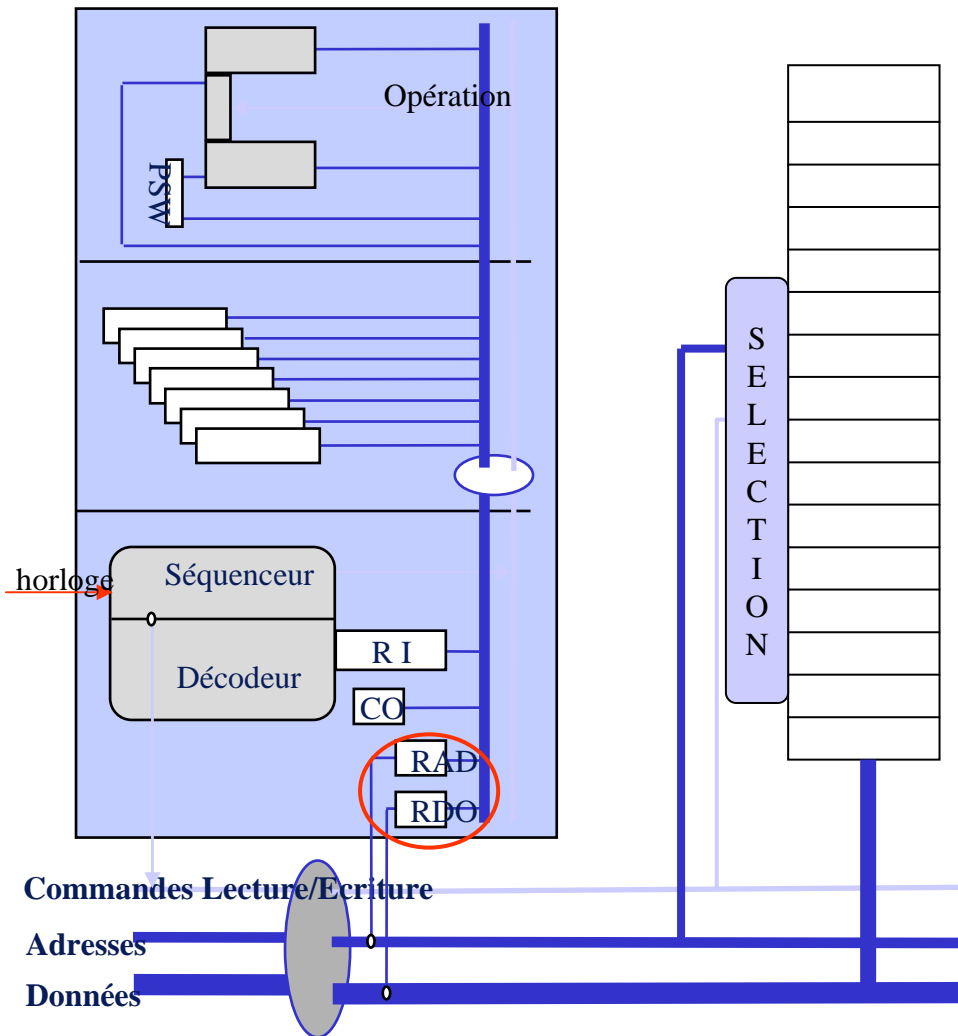
- de l'ensemble des circuits permettant de réaliser les opérations arithmétiques (addition, multiplication, division,...) et les opérations logiques (complément à 2, inverse, OU, ET, ...) sur les opérands Y1 et Y2
- d'un registre d'état PSW qui contient des indicateurs positionnés par le résultat Z des opérations effectuées :



- O : positionné à 1 si Overflow, 0 sinon
- Z : positionné à 1 si résultat opération nul, 0 sinon
- C : positionné à 1 si carry, 0 sinon
- S : positionné à 0 si résultat opération positif, 1 sinon



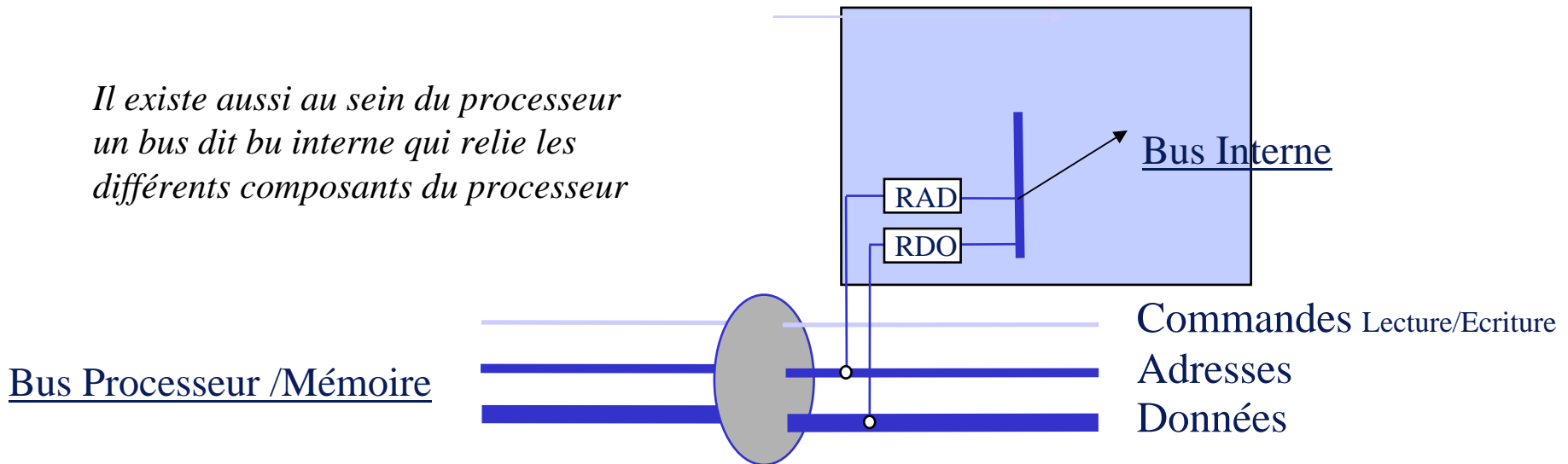
# La mémoire centrale



- La mémoire centrale est constituée par un ensemble de **mots** mémoire.
- Un mot est constitué par un ensemble d'octets.
- Chaque mot est repéré de manière unique par une **adresse**.
- Elle s'interface avec le cpu via les registres RAD et RDO

# Le bus processeur / mémoire

*Il existe aussi au sein du processeur un bus dit bus interne qui relie les différents composants du processeur*



Le bus processeur/mémoire permet la communication entre le processeur et la mémoire centrale. Au niveau du processeur cet interfaçage est mis en œuvre via les registre RAD et RDO.

Le bus est composé :

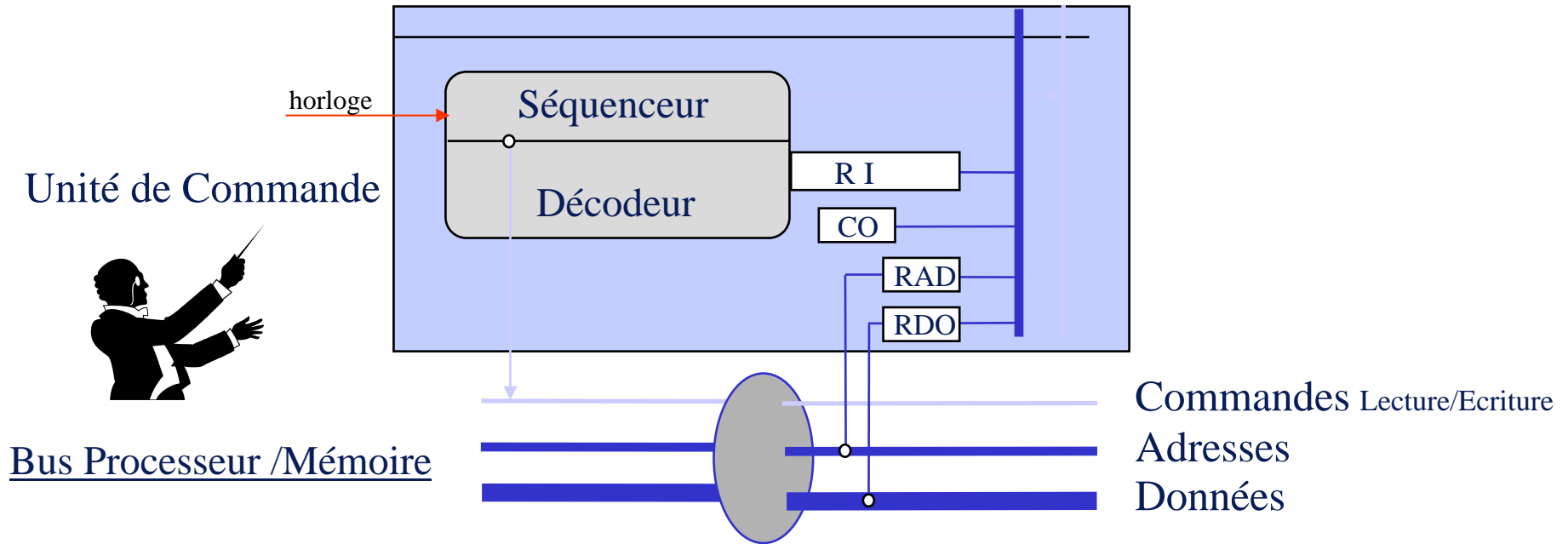
- de **lignes d'adresses** qui transportent l'adresse à laquelle le processeur s'intéresse
- de **lignes de données** qui transportent les mots mémoires
- de **lignes de commandes** qui spécifient le type d'opérations en cours sur le bus

On appelle **largeur du bus** le nombre de bits que le bus peut transporter en parallèle.

# Fonctionnement du processeur : exécution des instructions

- Structure du processeur et interfaçage avec la mémoire
- Exécution des instructions machines.
  - ↳ Principe de l'exécution d'une instruction
  - ↳ Rupture de séquence : les interruptions

# L'Unité de Commande



Unité de Commande



Bus Processeur /Mémoire

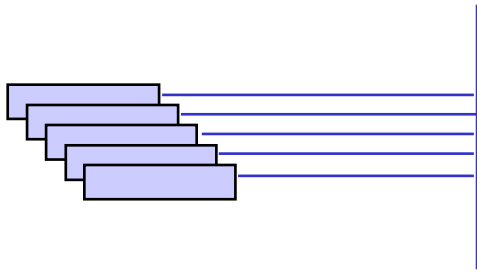
L'unité de commande est chargée de la reconnaissance des instructions et de leur exécution par le processeur au rythme de l'horloge

Ce travail s'effectue pour chaque instruction machine en trois phases :

- Fetch : charger une instruction
- Exécution : réaliser l'instruction

2

Registres



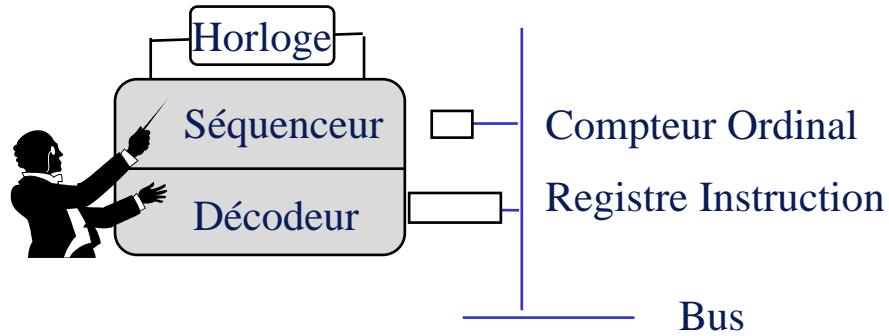
Ouvrir / Fermer les registres

Micro commandes



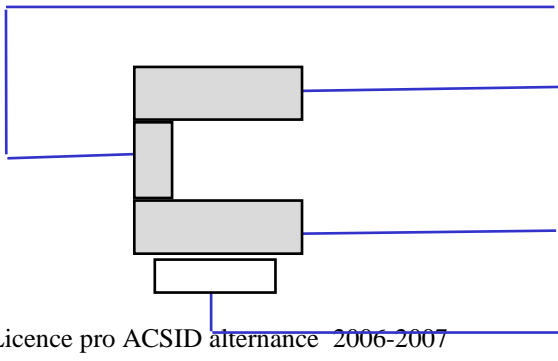
L' unité de commande :

- Charger une instruction : *fetch*
- Exécute l'instruction: *exécution*



Micro commandes

Activer une opération sur les opérandes



UAL

# Les étapes d'exécution d'une instruction (Chargement/Exécution)

Début

Charger

la prochaine instruction à exécuter  
depuis la mémoire dans le  
registre instruction,

Modifier

le Compteur Ordinal pour qu'il pointe  
sur la prochaine  
instruction à exécuter,

Fetch

Charger

les données éventuelles dans les registres  
internes,

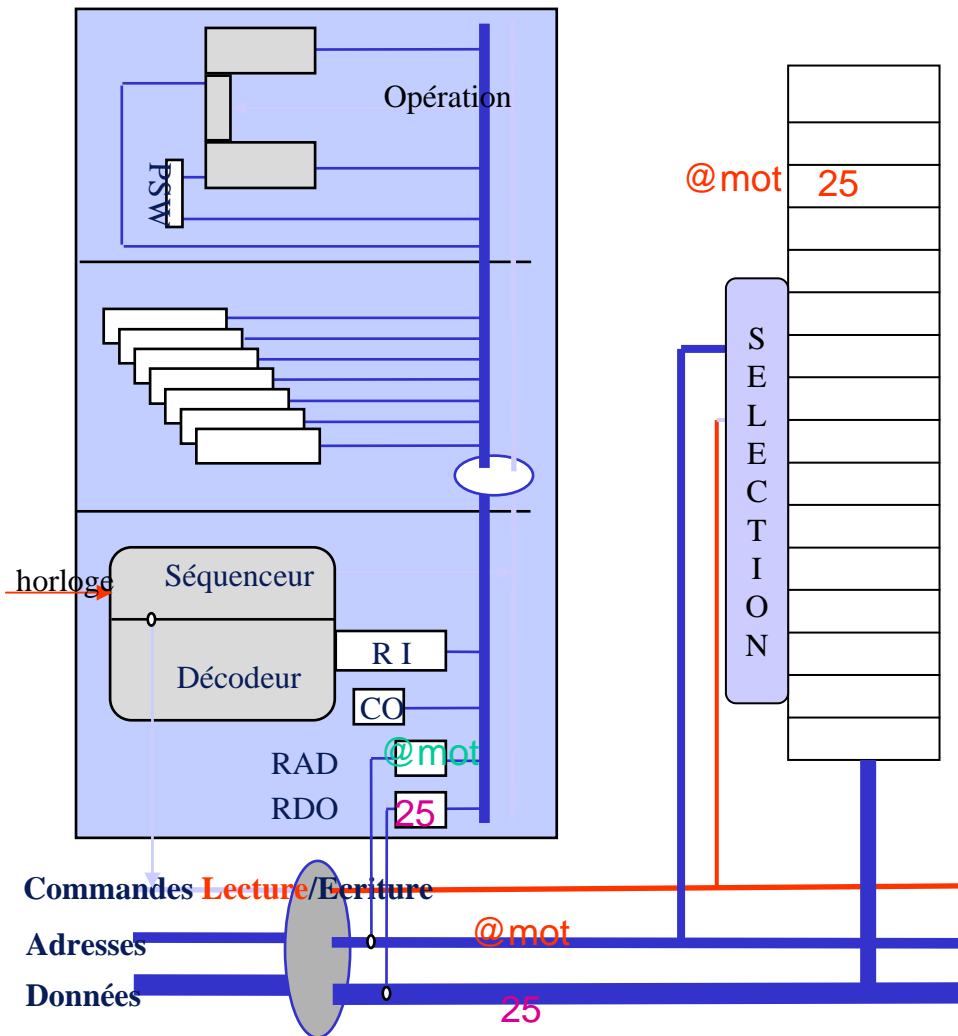
Exécuter

l'instruction,

Exécution

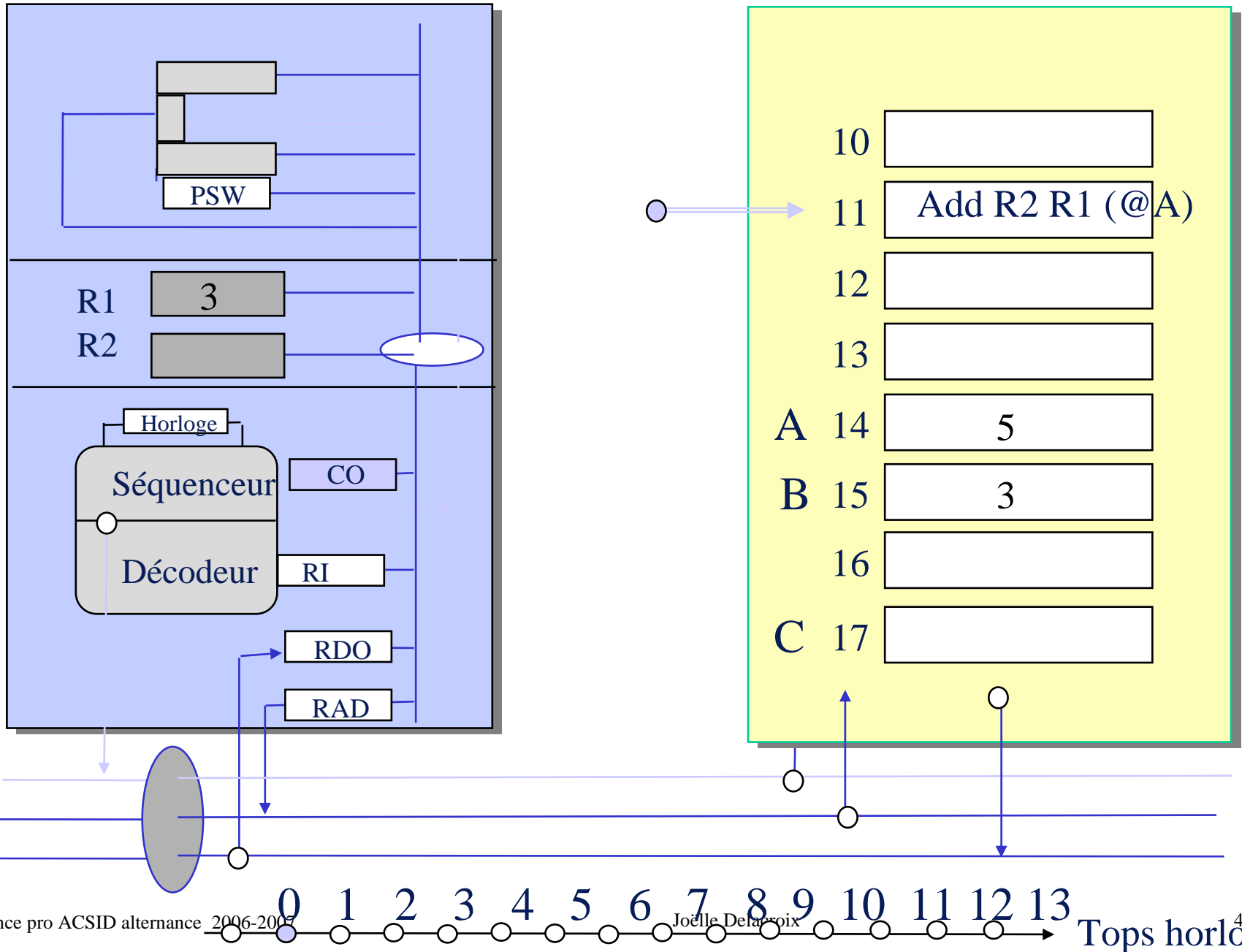
Fin

# Liaison processeur - mémoire centrale



- **Opération de lecture**
  - Adresse du mot à lire --> RAD
  - Activer la commande de lecture
  - le contenu de RAD est déposé sur le bus d'adresses et présenté à la mémoire
  - Le mot lu est déposé sur le bus de données puis copié dans RDO

# Chargement/Exécution





# Les étapes d'exécution d'une instruction (Chargement/Exécution)

Début

Charger

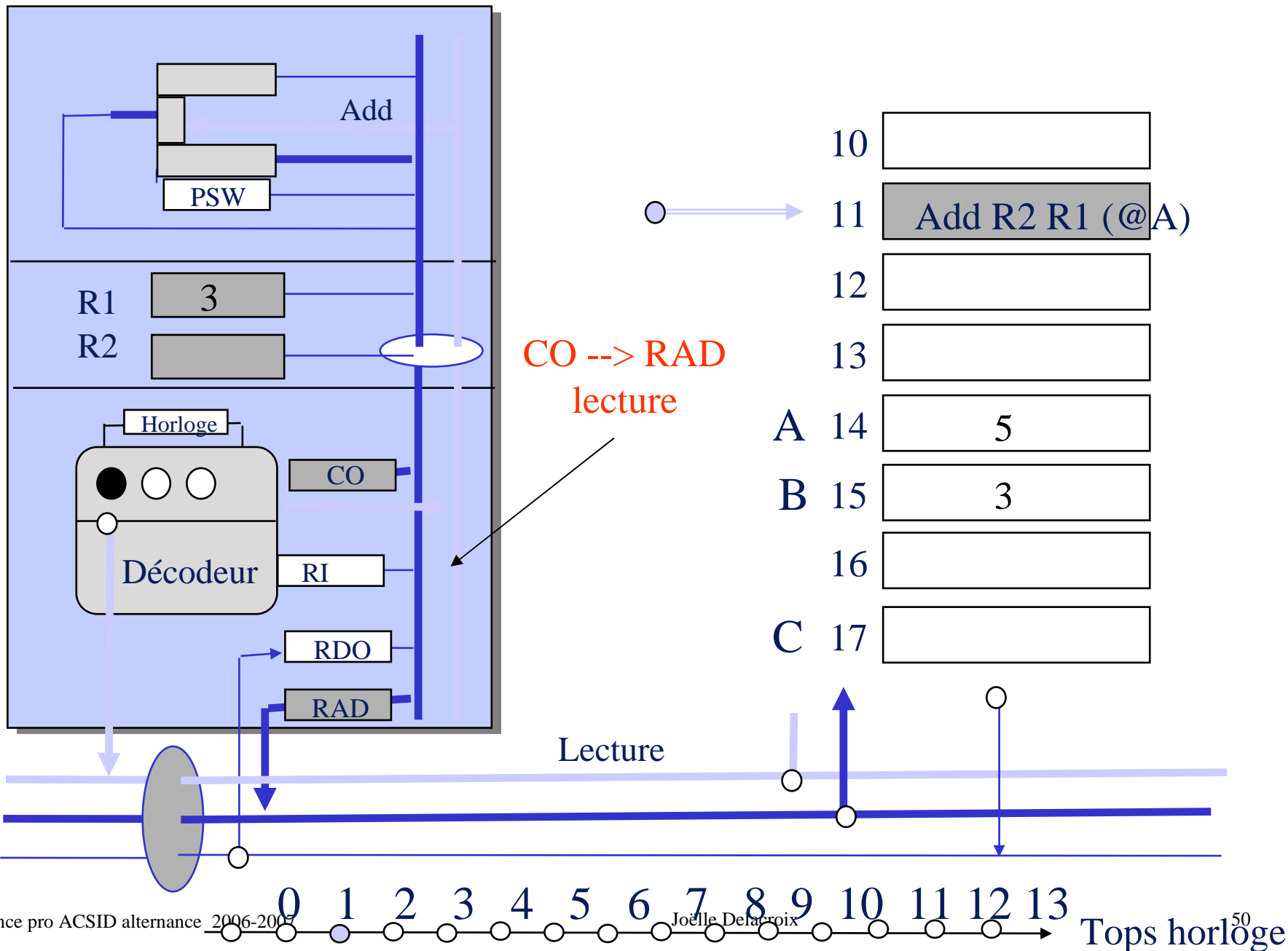
la prochaine instruction à exécuter  
depuis la mémoire dans le  
registre instruction,

Modifier

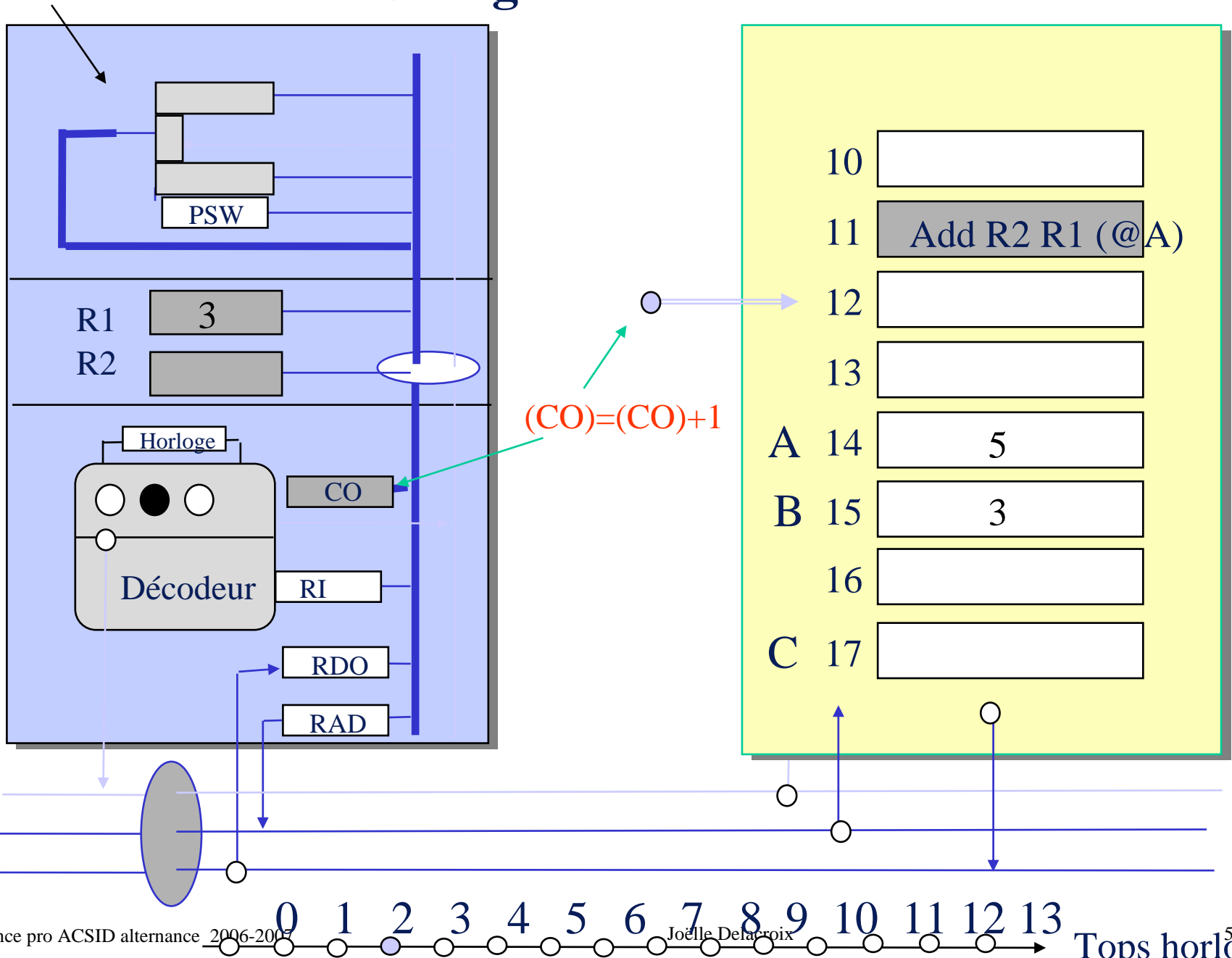
le Compteur Ordinal pour qu'il pointe  
sur la prochaine  
instruction à exécuter,

Fetch

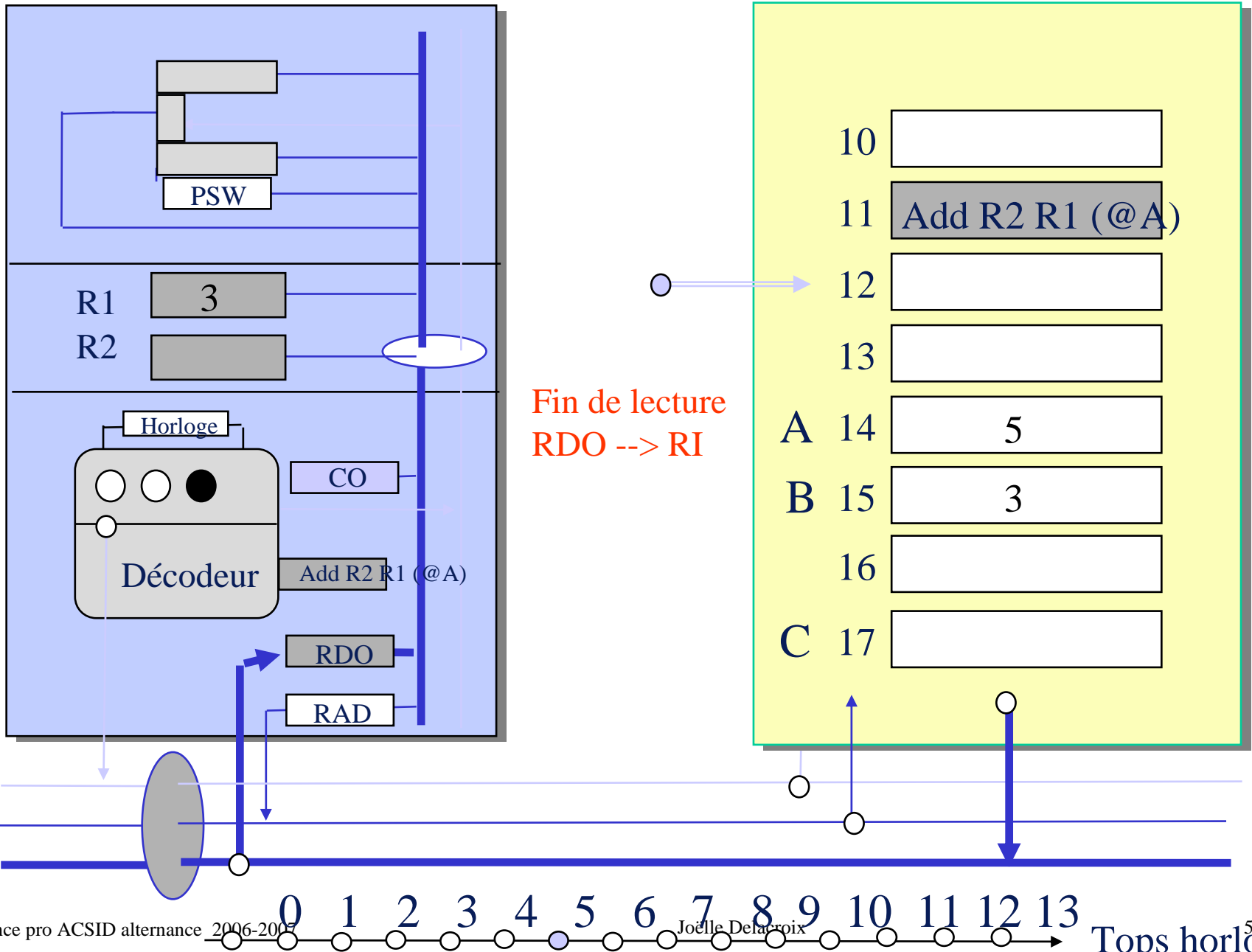
# Chargement Exécution



# Chargement/Exécution



# Chargement/Exécution



# Les étapes d'exécution d'une instruction (Chargement/Exécution)

Début

**Charger** les données éventuelles dans les registres internes,  
**Exécuter** l'instruction,

Exécution

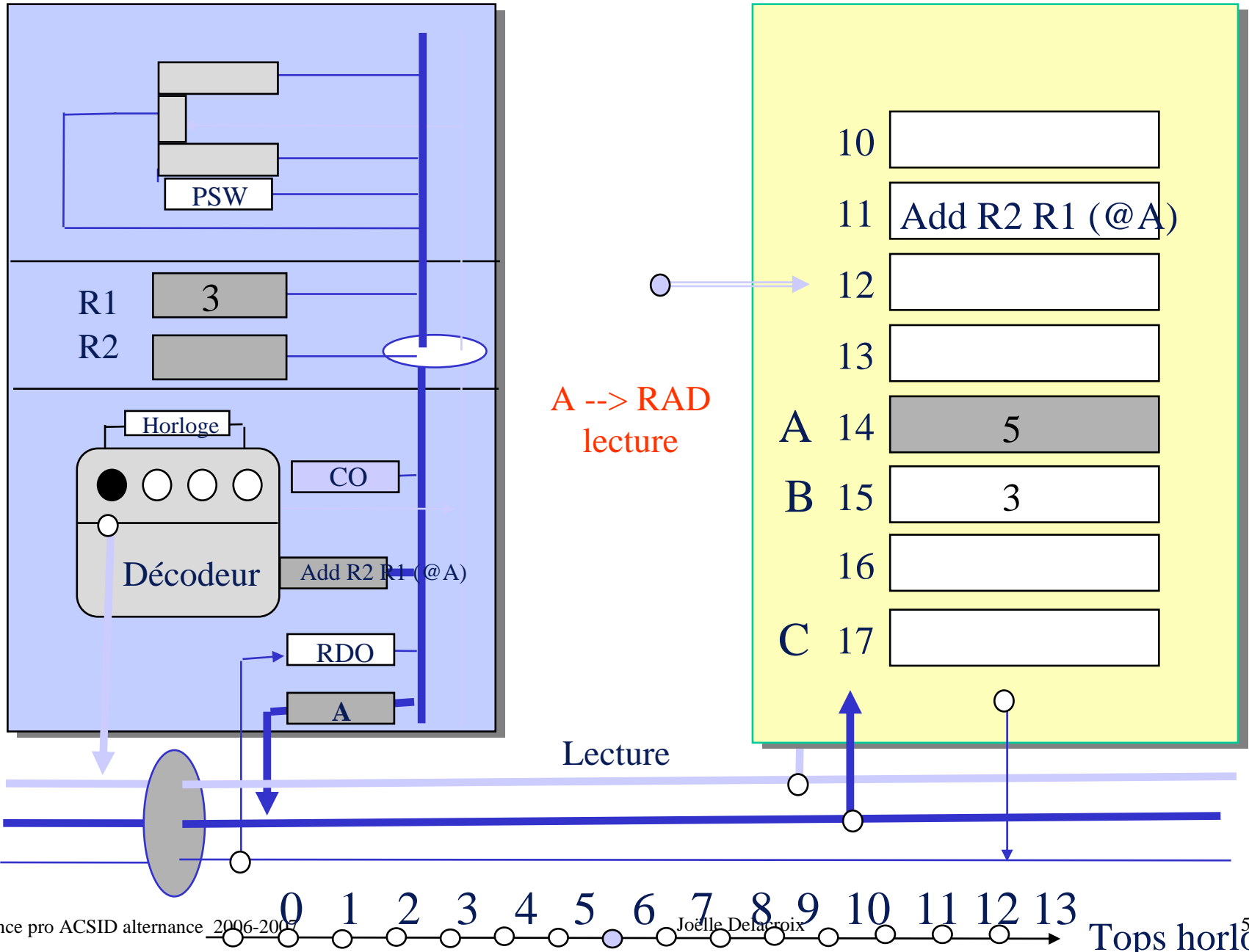
Fin

Add R2 R1 (@A) :  $R2 = (R1) + (@A)$

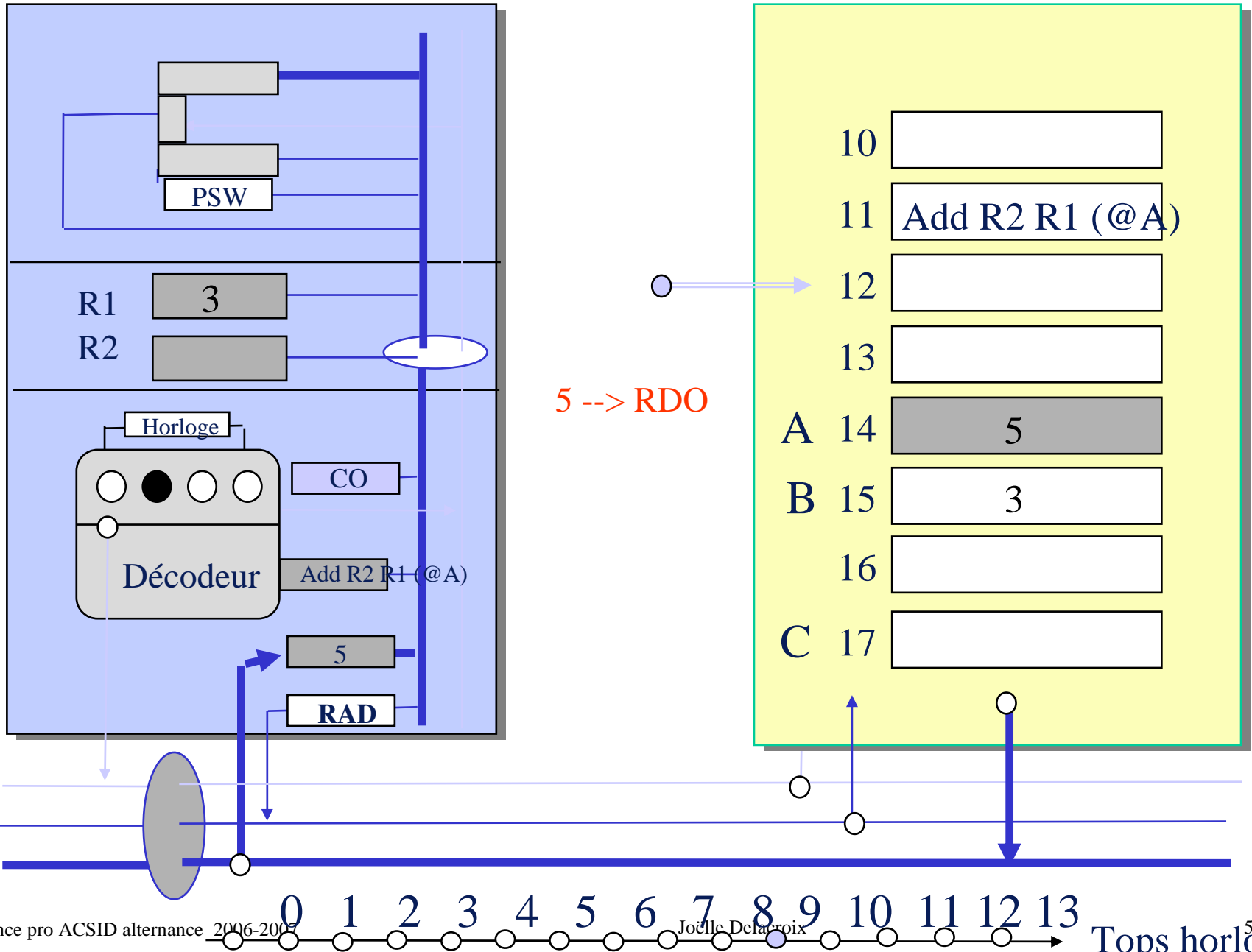
Un premier opérande est le contenu de R1.

Le deuxième opérande est un mot mémoire d'adresse A : ramener cet opérande au niveau du CPU nécessite **une opération de lecture en MC à l'adresse A.**

# Chargement/Exécution



# Chargement/Exécution



# Les étapes d'exécution d'une instruction (Chargement/Exécution)

Début

Charger les données éventuelles dans les registres internes,  
Exécuter l'instruction,

Exécution

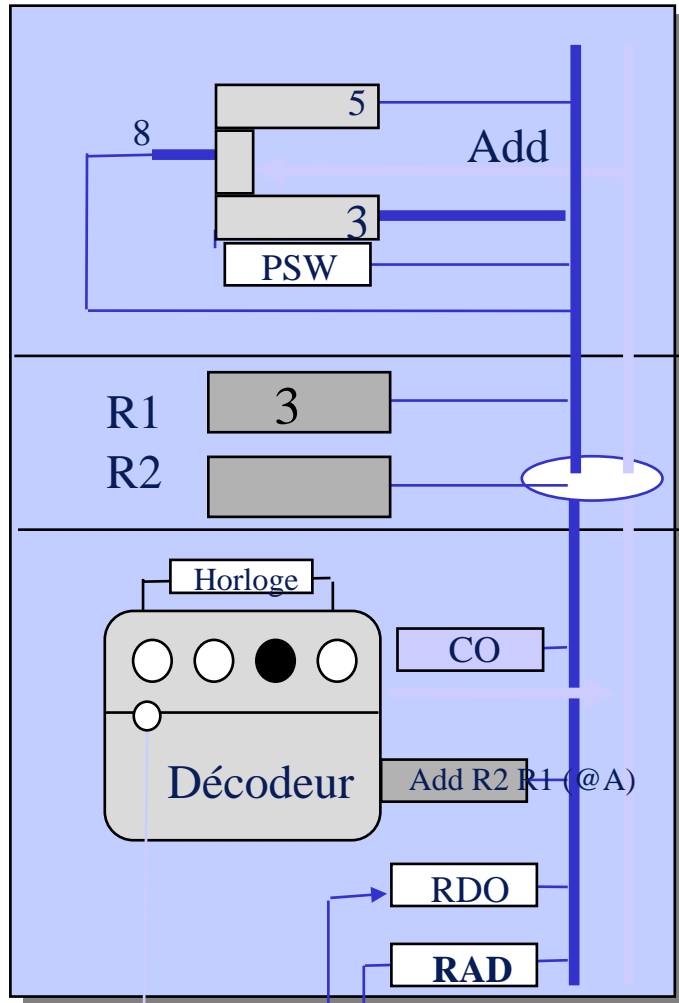
Fin

Add R2 R1 (@A) :  $R2 = (R1) + (@A)$

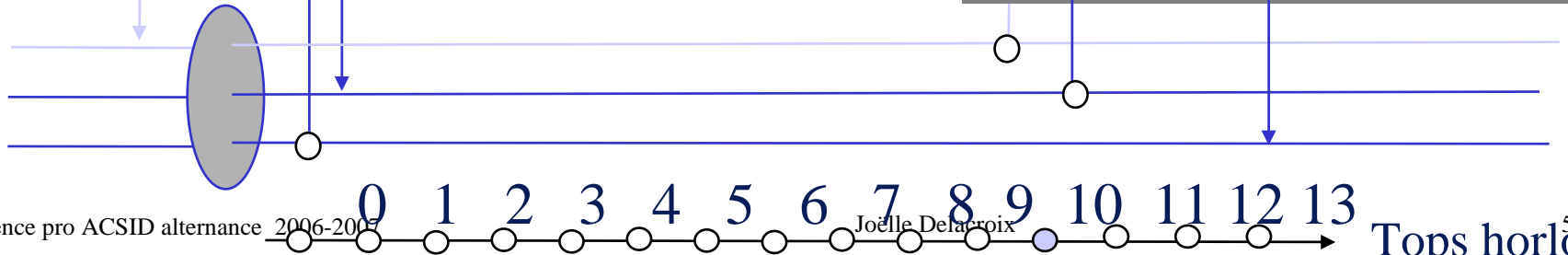
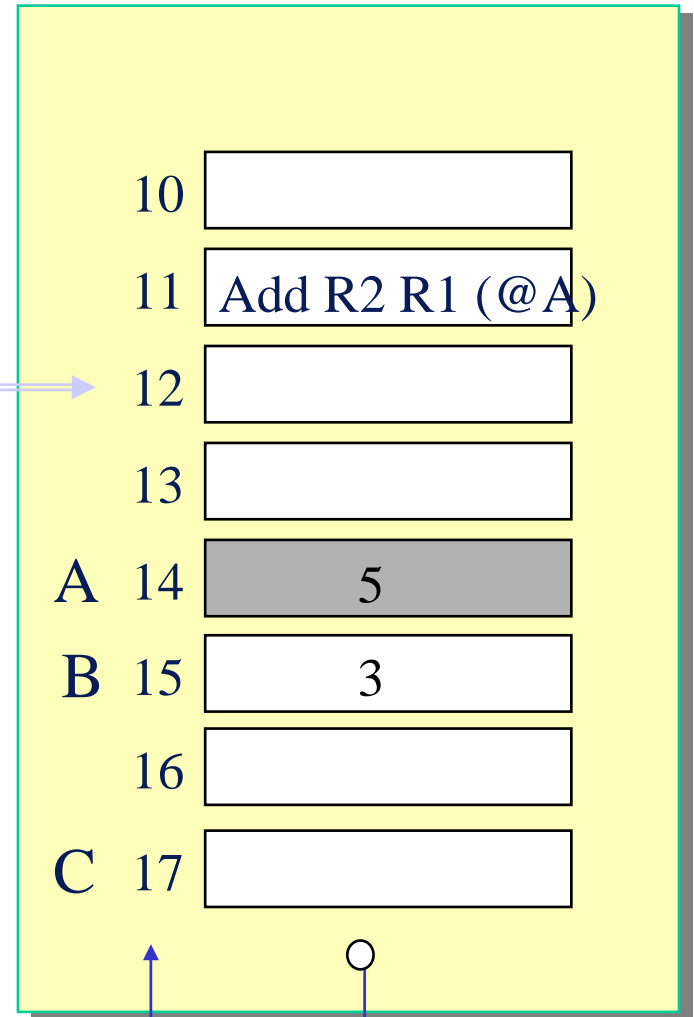
L'opération est une addition : il faut amener les deux opérandes à l'UAL et activer le circuit d'addition. Puis stocker le résultat dans R2.



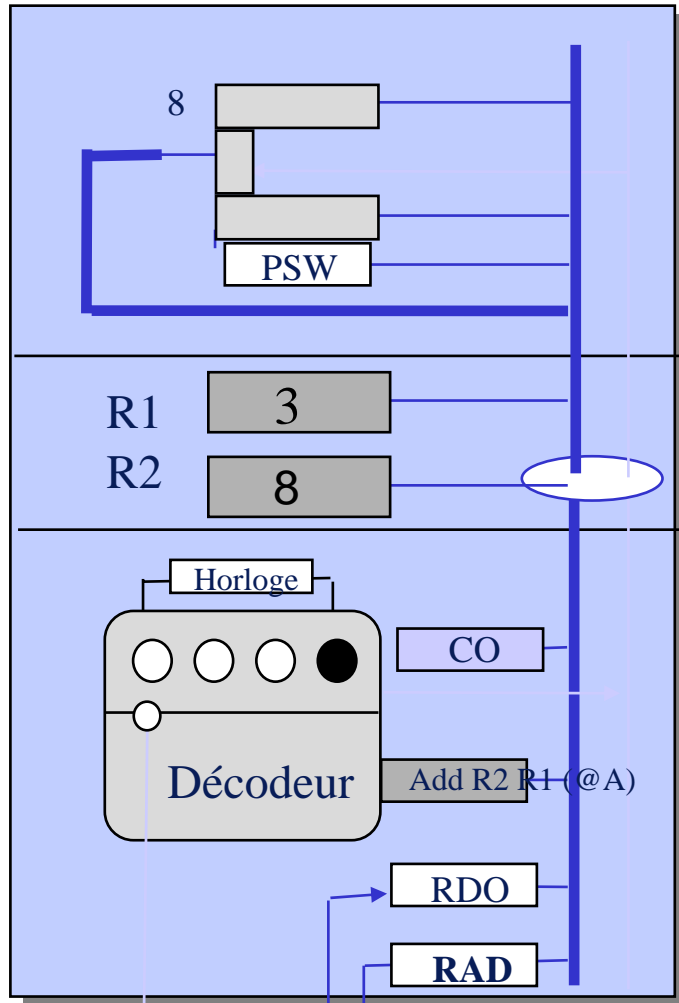
# Chargement/Exécution



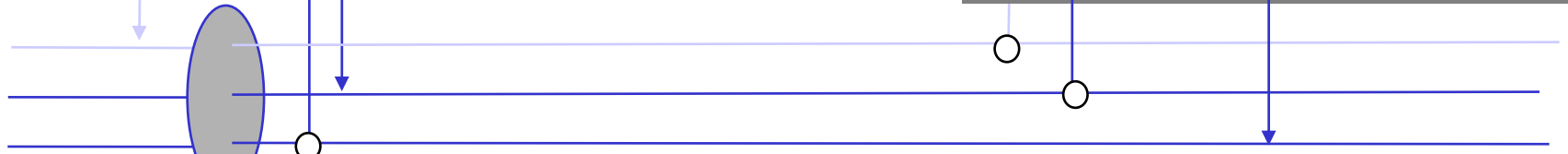
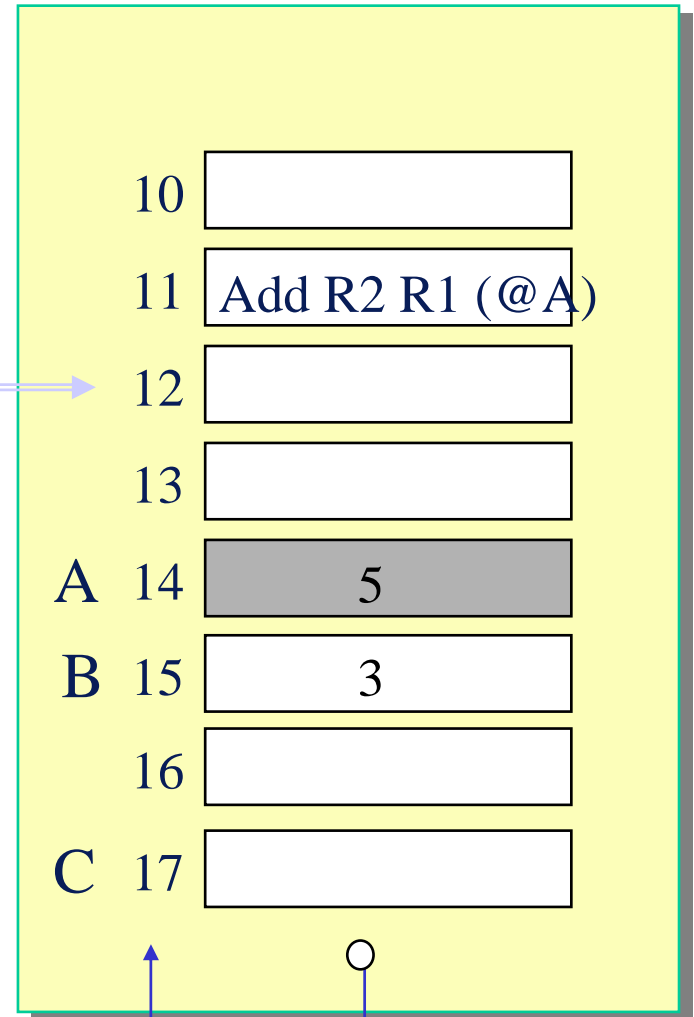
RDO --> Ual  
 R1 --> ual  
 ADD  
 la sortie de l'ual  
 devient égale à 8



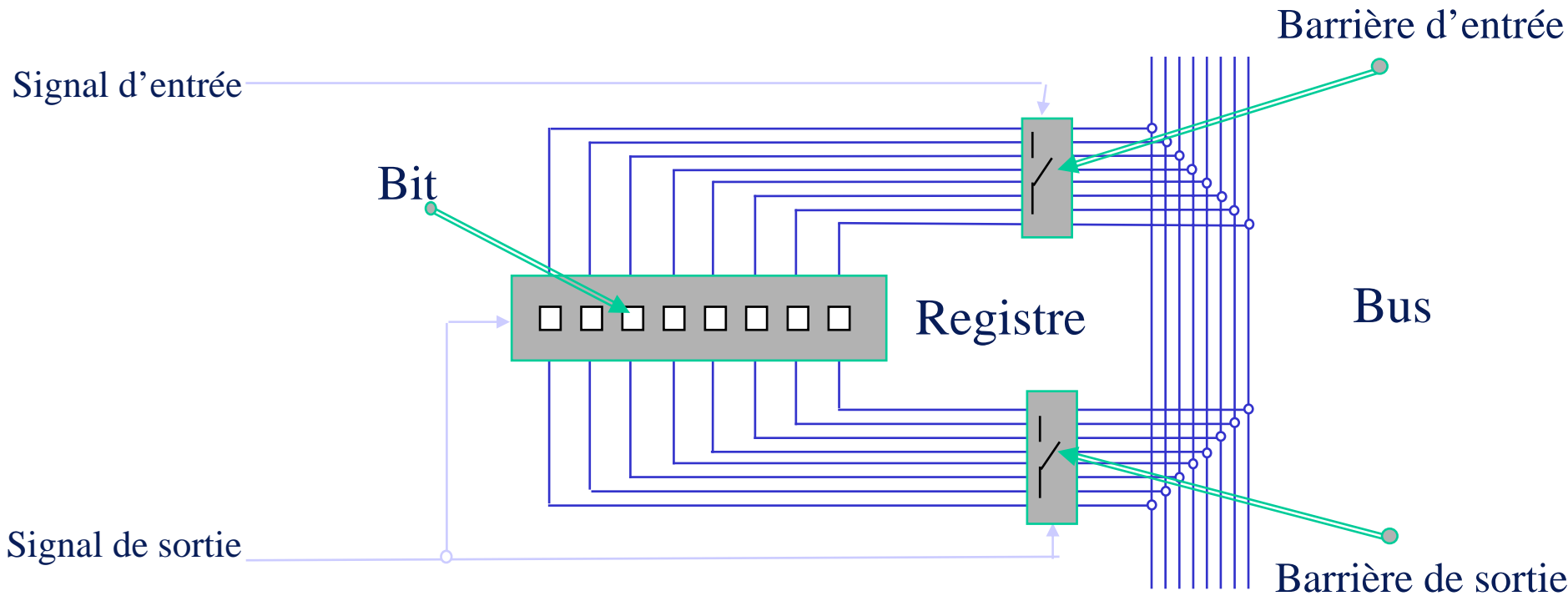
# Chargement/Exécution



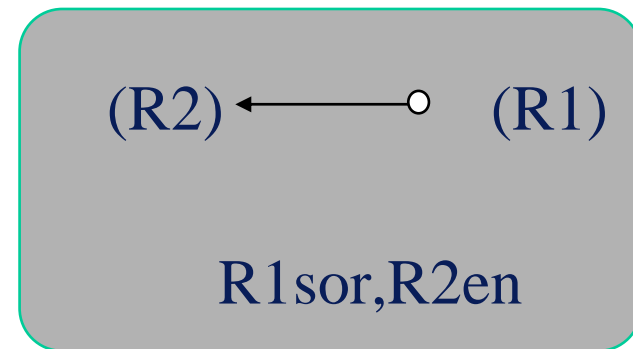
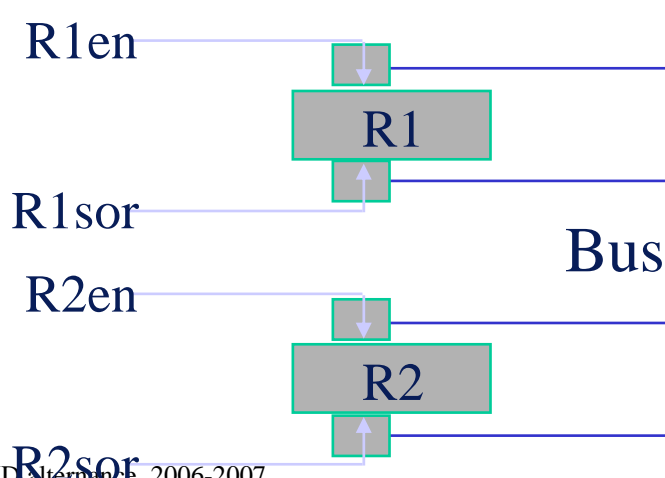
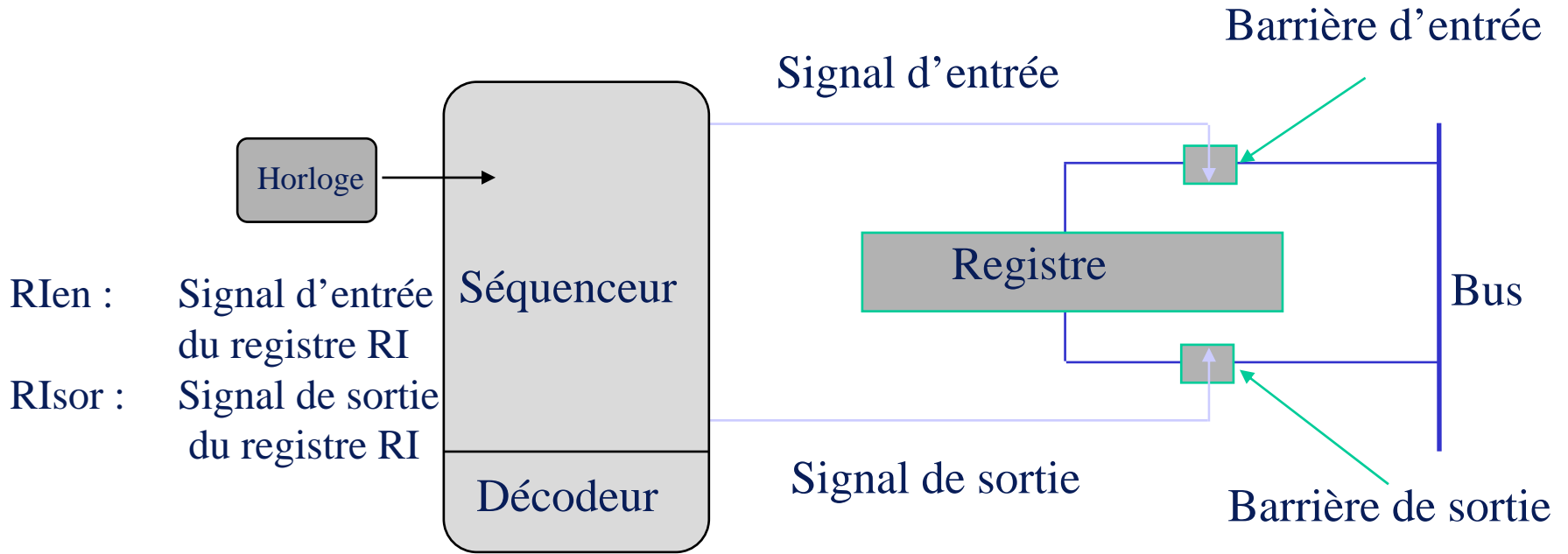
Sortie Ual --> R2



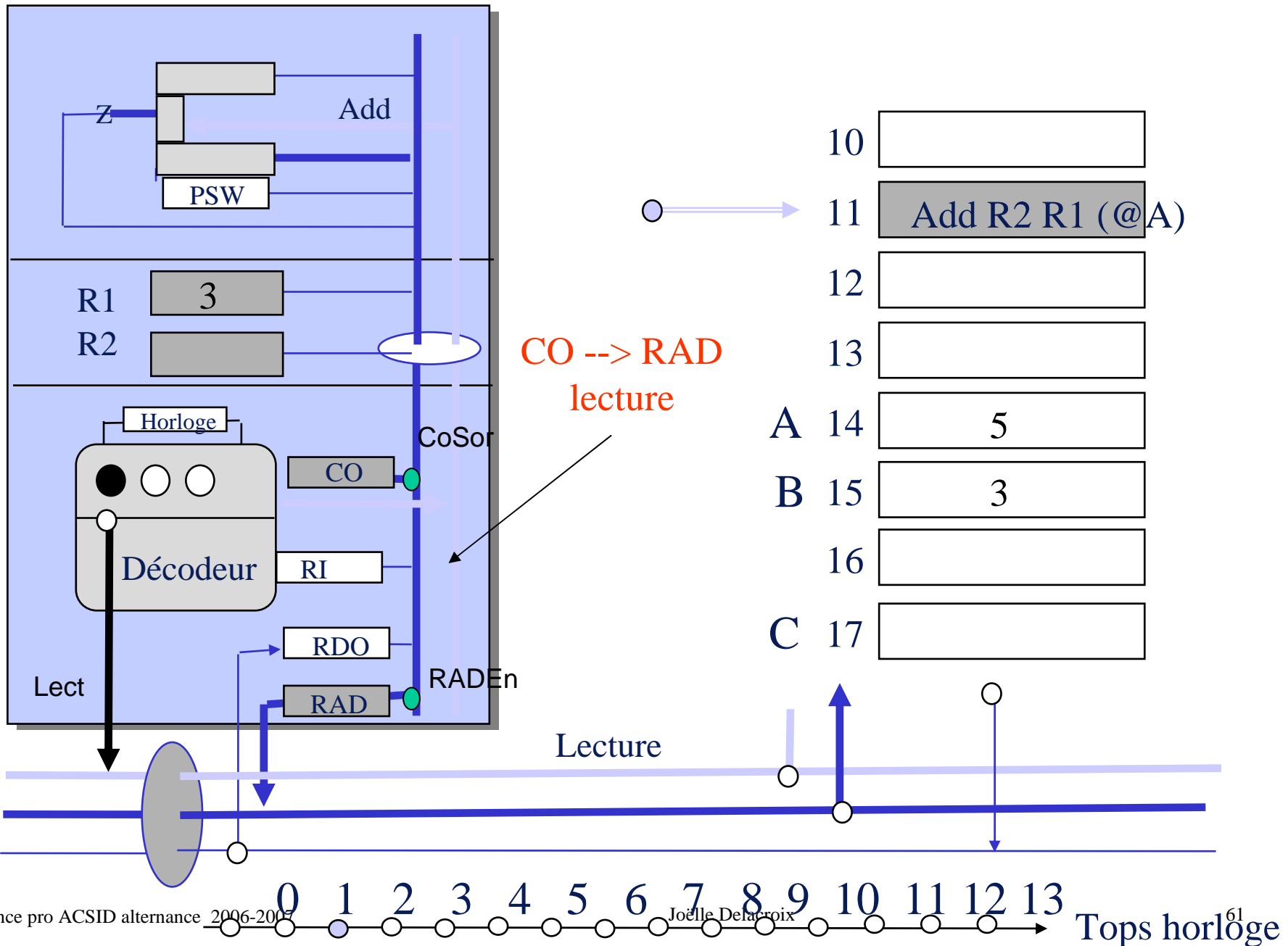
# Registre (principe de fonctionnement)



# Exécution : Registre (principe de fonctionnement)



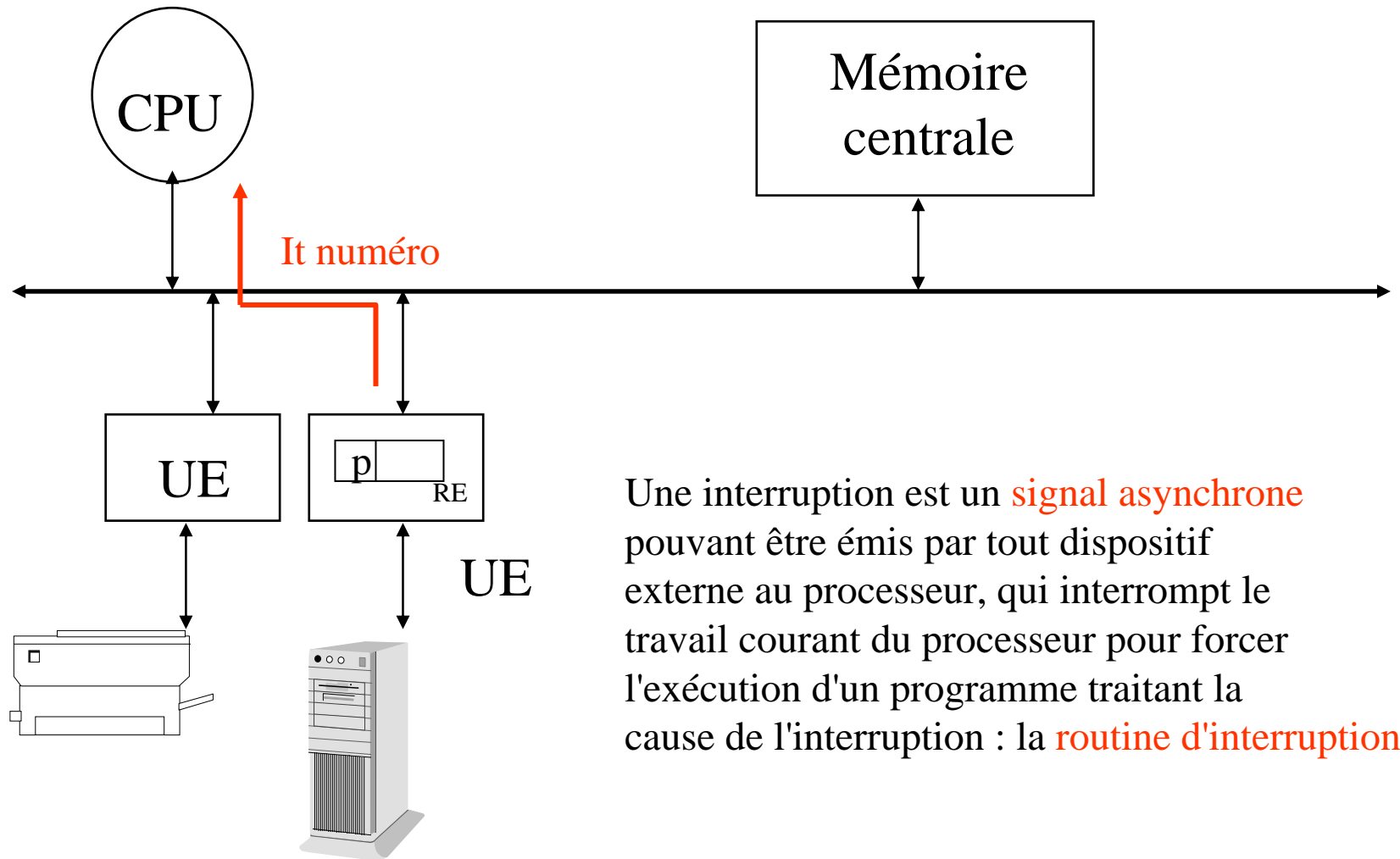
# CoSor, RADen, Lect, AttenteMémoire



# Fonctionnement du processeur : exécution des instructions

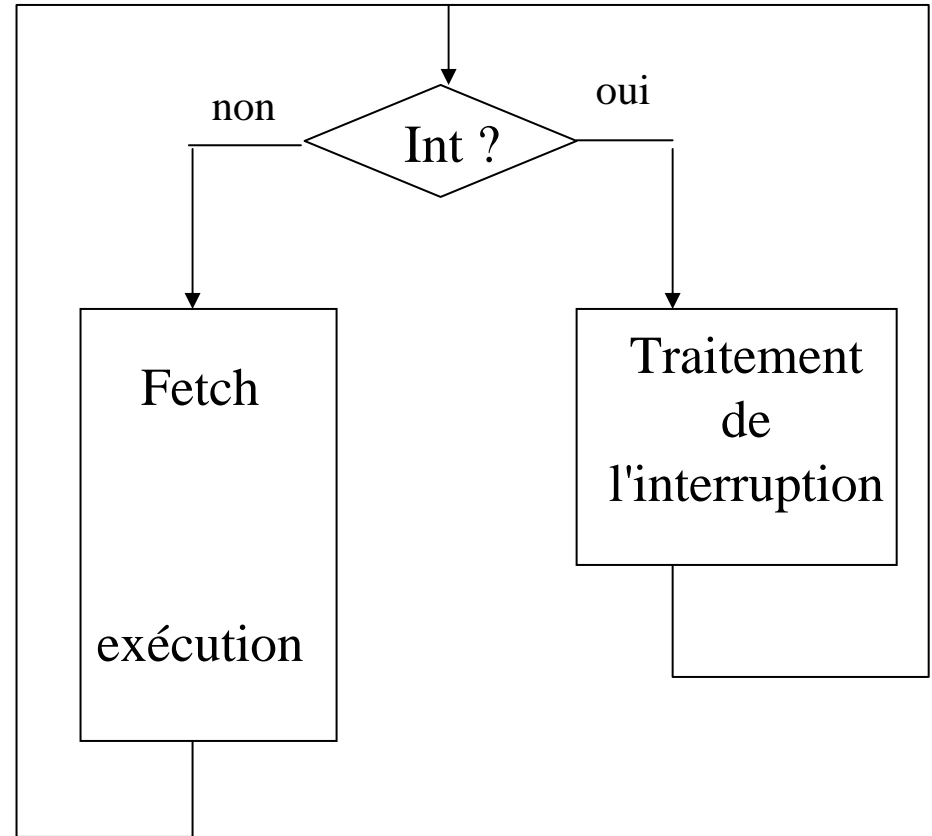
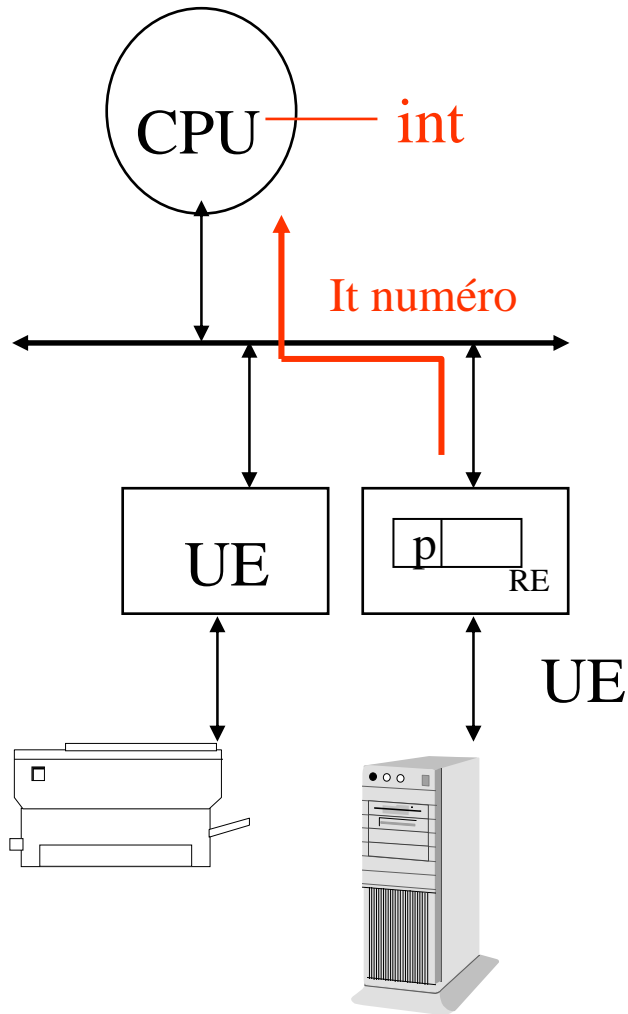
- Rappels : structure du processeur et interfaçage avec la mémoire
  - ↳ Exécution des instructions machines.
    - ↳ Principe de l'exécution d'une instruction
    - ↳ Rupture de séquence : les interruptions

# Les interruptions



Une interruption est un **signal asynchrone** pouvant être émis par tout dispositif externe au processeur, qui interrompt le travail courant du processeur pour forcer l'exécution d'un programme traitant la cause de l'interruption : la **routine d'interruption**

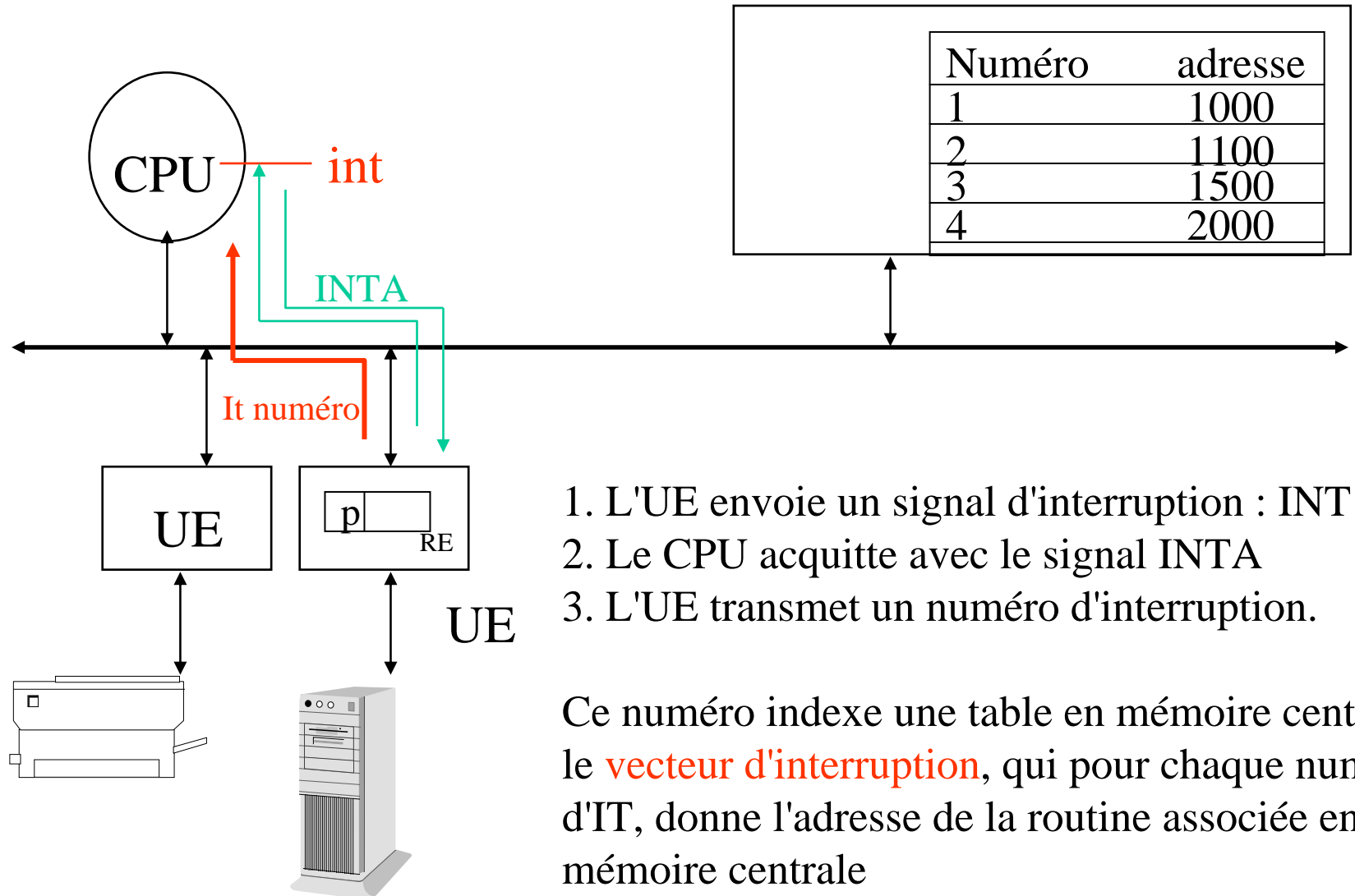
# Les interruptions





# Les interruptions

## Mémoire centrale



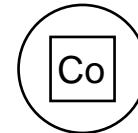
# Les Interruptions

Table des vecteurs  
d'interruptions

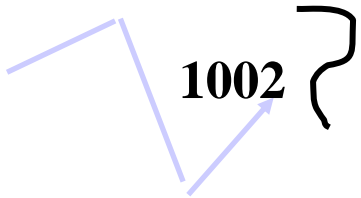
It 3	0017
Sous traitant It3	
Programme	

0017

1002



Programme



1002



Latence



Sous-Traitant d'IT n°3

Co ← 0017

0017

Traitement It

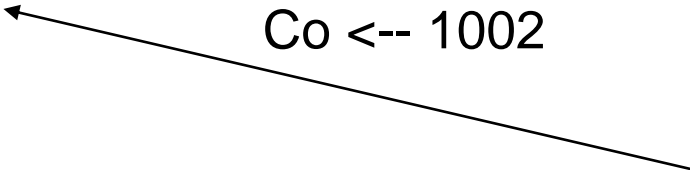


RTI

Interruption  
n°3

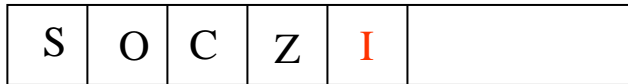
1002

Co ← 1002



# Le masquage des interruptions

## Registre PSW du processeur



I= 1, It masquée

I= 0, It démasquée

- O : positionné à 1 si Overflow, 0 sinon
- Z : positionné à 1 si résultat opération nul, 0 sinon
- C : positionné à 1 si carry, 0 sinon
- S : positionné à 0 si résultat opération positif, 1 sinon

- Une interruption masquée n'ait pas prise en compte par le processeur lorsque celle-ci survient.
- EI : Enable Interrupt
  - les interruptions sont démasquées
- DI : Desable Interrupt
  - les interruptions sont masquées

# Hiérarchisation des interruptions

- Le processeur dispose d'une seule broche INT pour recevoir les interruptions.
- Que faire :
  - Si deux interruptions arrivent en même temps ?
  - Si une interruption survient durant le traitement d'une autre interruption ?
- On introduit une notion de priorité (niveau) entre interruptions.
  - Exemple : 8 niveaux

Niveau 0 1 2 3 4 5 6 7


Plus forte priorité

Plus faible priorité

# Hiérarchisation des interruptions

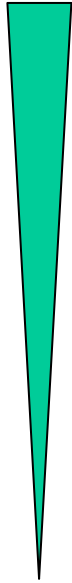
- Le processeur dispose d'une seule broche INT pour recevoir les interruptions.
- Que faire :
  - Si deux interruptions arrivent en même temps ?  
☞ On traite d'abord la plus prioritaire
  - Si une interruption survient durant le traitement d'une autre interruption ?  
☞ Elle interrompt le traitement en cours seulement si elle est de priorité supérieure
- On introduit une notion de priorité (niveau) entre interruptions.
  - Exemple : 8 niveaux

Niveau 0 1 2 3 4 5 6 7

Plus forte priorité  Plus faible priorité

# Hiérarchisation des interruptions

Arrivée IT 5, 6



Traitement IT5,  
IT 6 en attente

Arrivée IT 7  
IT 6, 7 en attente

Arrivée IT 2  
On interrompt  
Traitement IT5



Traitement IT2,  
IT 6, 7 , 5 en attente



Fin Traitement IT2,  
Reprise traitement IT 5

Fin Traitement IT2,  
Traitement IT 6, puis IT 7