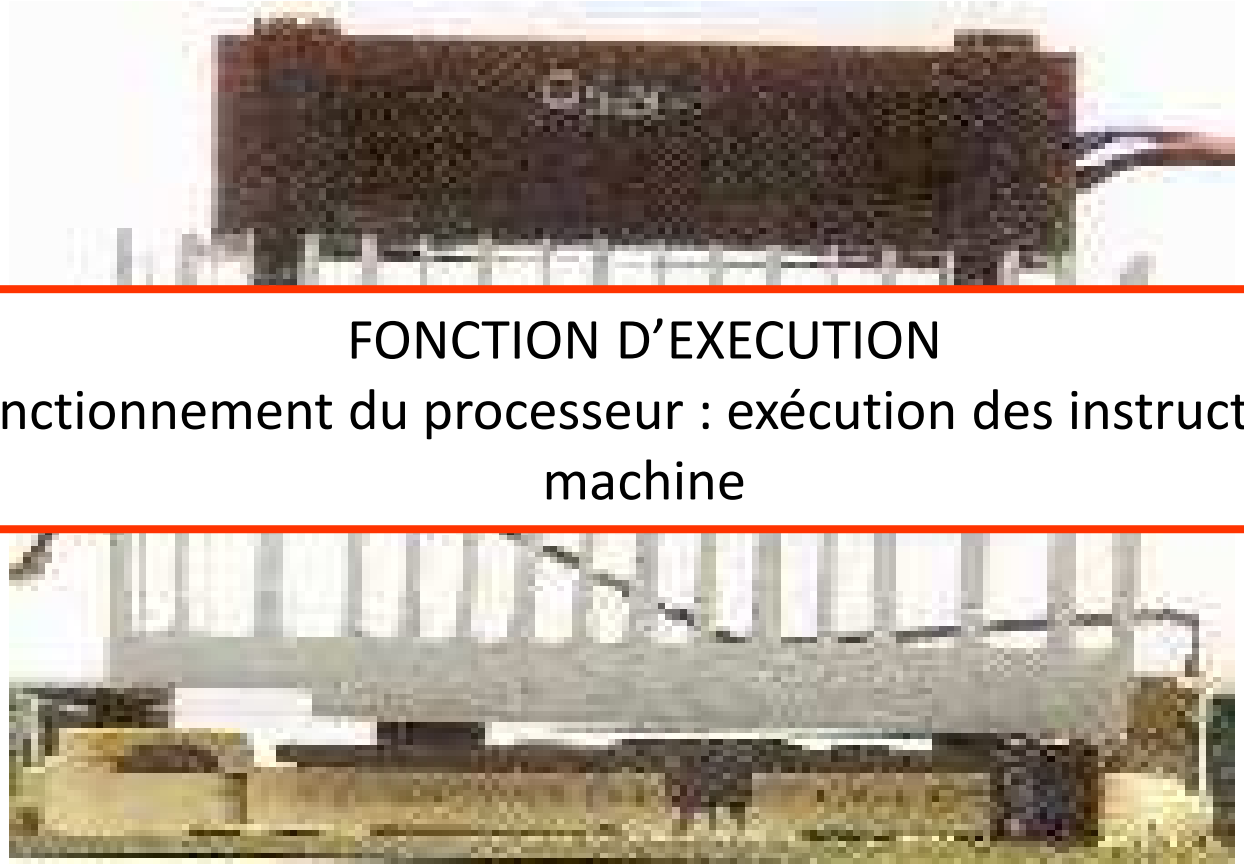
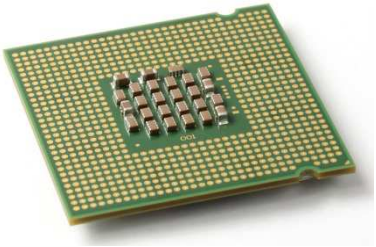


FONCTION D'EXECUTION

Fonctionnement du processeur : exécution des instructions machine

STRUCTURE DU PROCESSEUR

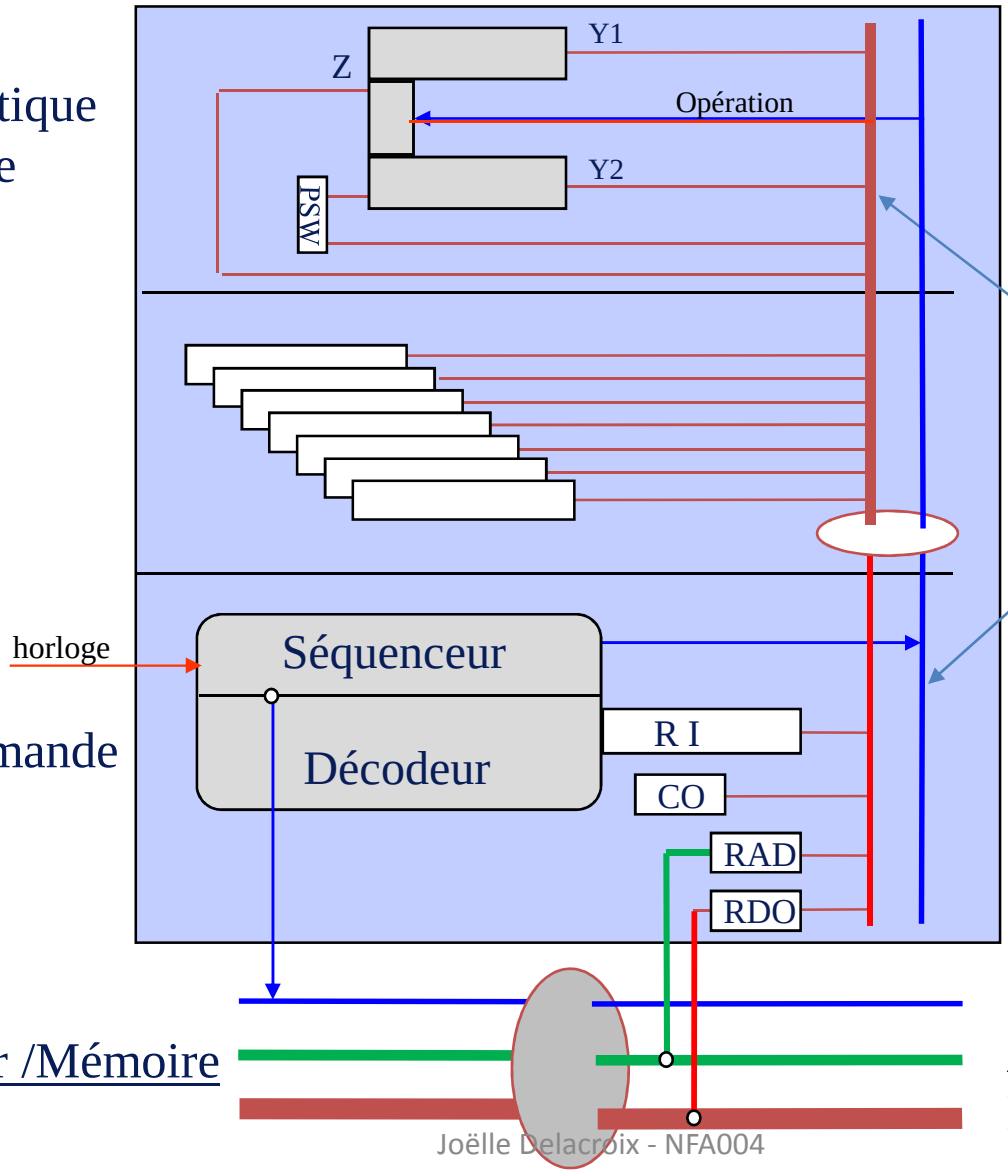
Processeur (Unité Centrale)



Unité Arithmétique et Logique

Registres

Unité de Commande



Bus Interne

Données

Commandes

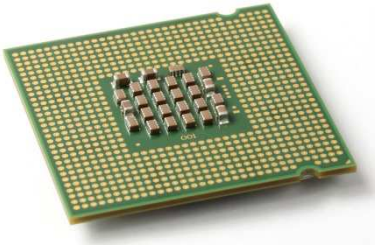
Bus Processeur /Mémoire

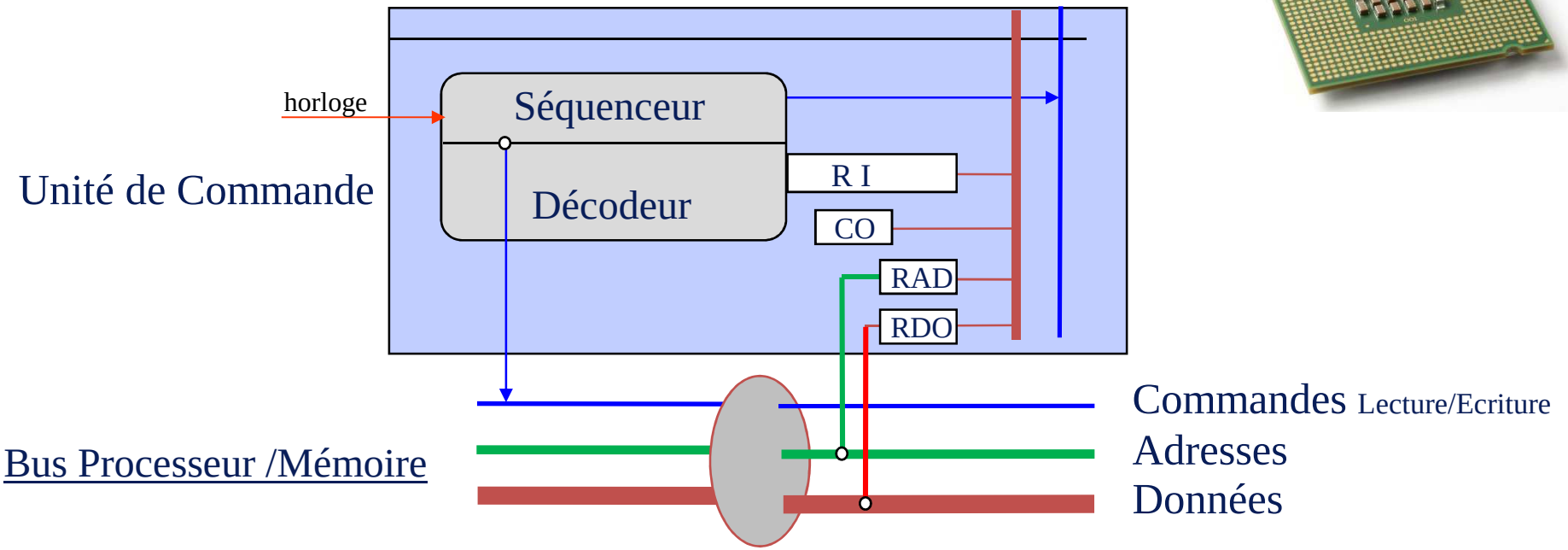
Commandes Lecture/Ecriture

Adresses

Données

Processeur (Unité Centrale)



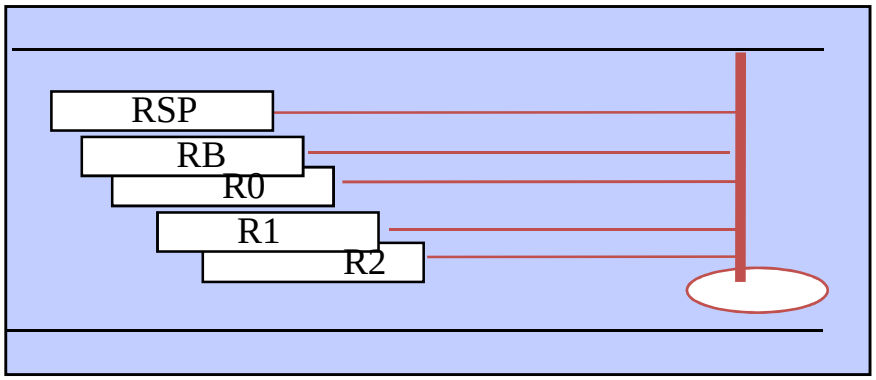


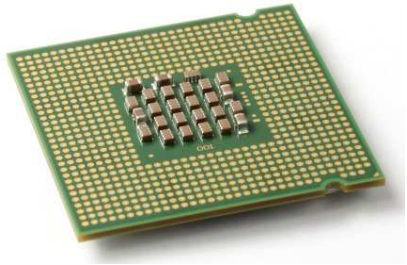
L'unité de commande est chargée de la reconnaissance des instructions et de leur exécution par l'unité de traitement au rythme de l'horloge

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)



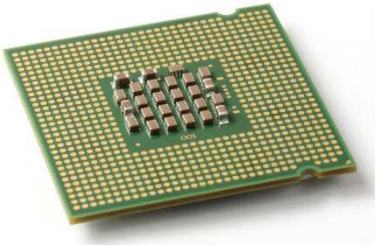


- Le registre est l'entité de base manipulée par le processeur.
- Aucune opération n'est directement réalisée sur les cellules mémoires.

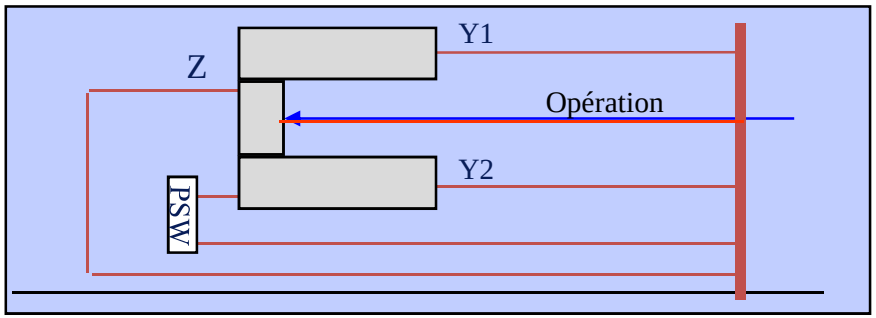
Registres :

- les registres généraux R0, R1, R2
- le registre de pile RSP (Register Stack Pointer)
- les registres d'adressage : RB (registre de base)

Processeur (Unité Centrale)



Unité Arithmétique et Logique



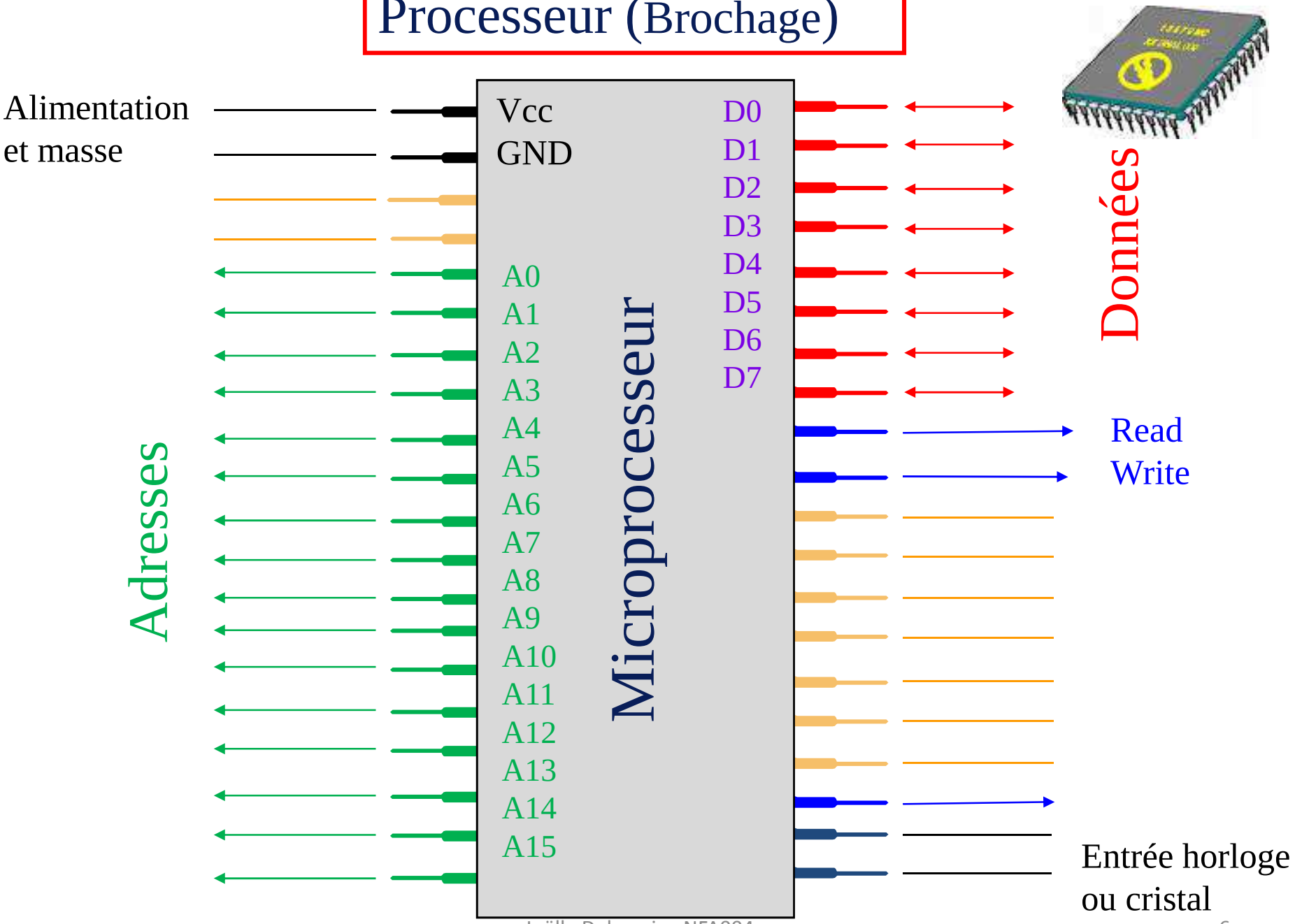
L'unité Arithmétique et Logique (UAL) constitue l'unité d'exécution du processeur. Elle est composée :

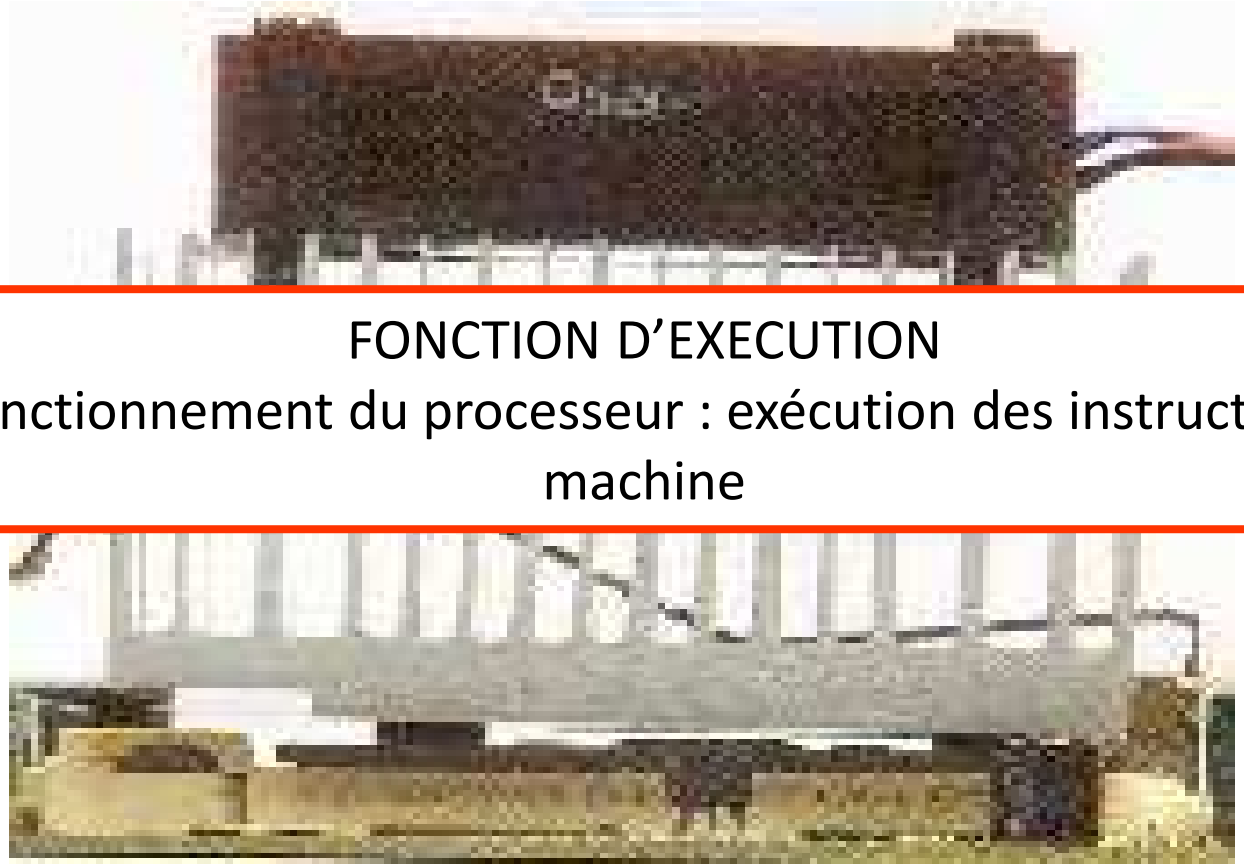
- de l'ensemble des circuits permettant de réaliser les opérations arithmétiques (addition, multiplication, division,…) et les opérations logiques (complément à 2, inverse, OU, ET, …) sur les opérandes Y1 et Y2
- d'un registre d'état PSW qui contient des indicateurs positionnés par le résultat Z des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

S	O	C	Z	

- 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

Processeur (Brochage)





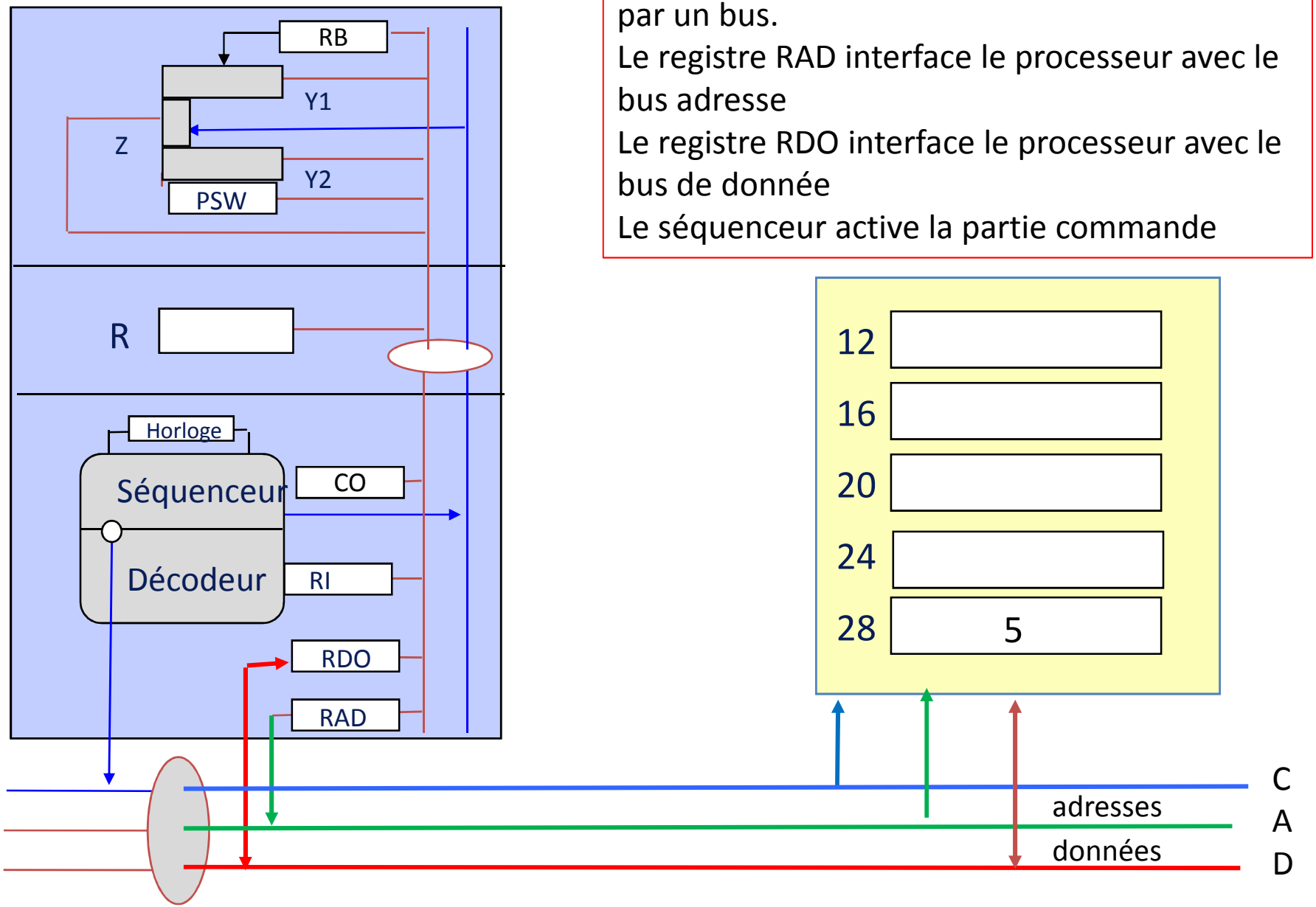
FONCTION D'EXECUTION

Fonctionnement du processeur : exécution des instructions machine

Liaison processeur – Mémoire centrale

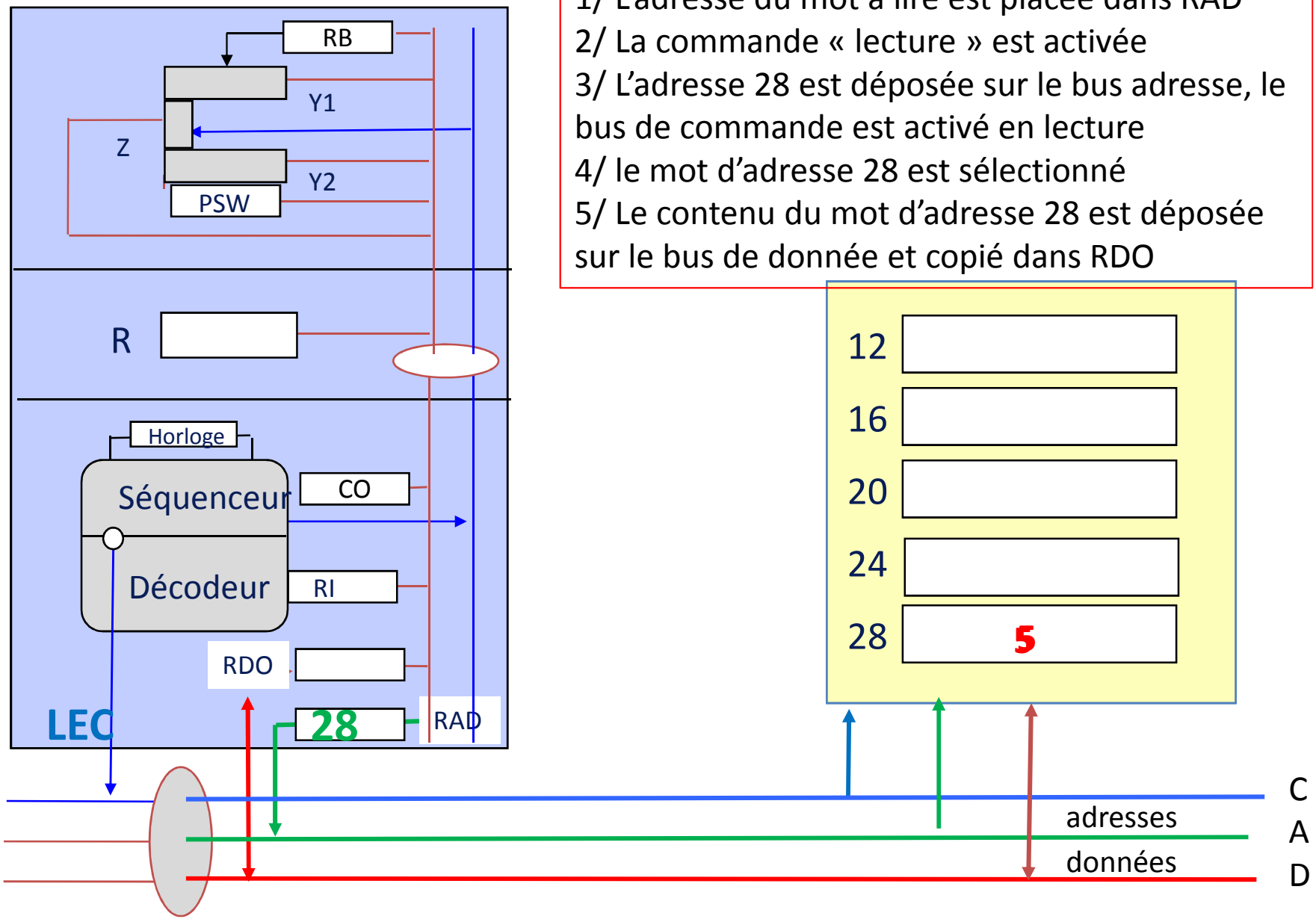
Réaliser une lecture ou une écriture d'un mot mémoire

Le processeur est relié à la mémoire centrale par un bus.
Le registre RAD interface le processeur avec le bus adresse
Le registre RDO interface le processeur avec le bus de donnée
Le séquenceur active la partie commande



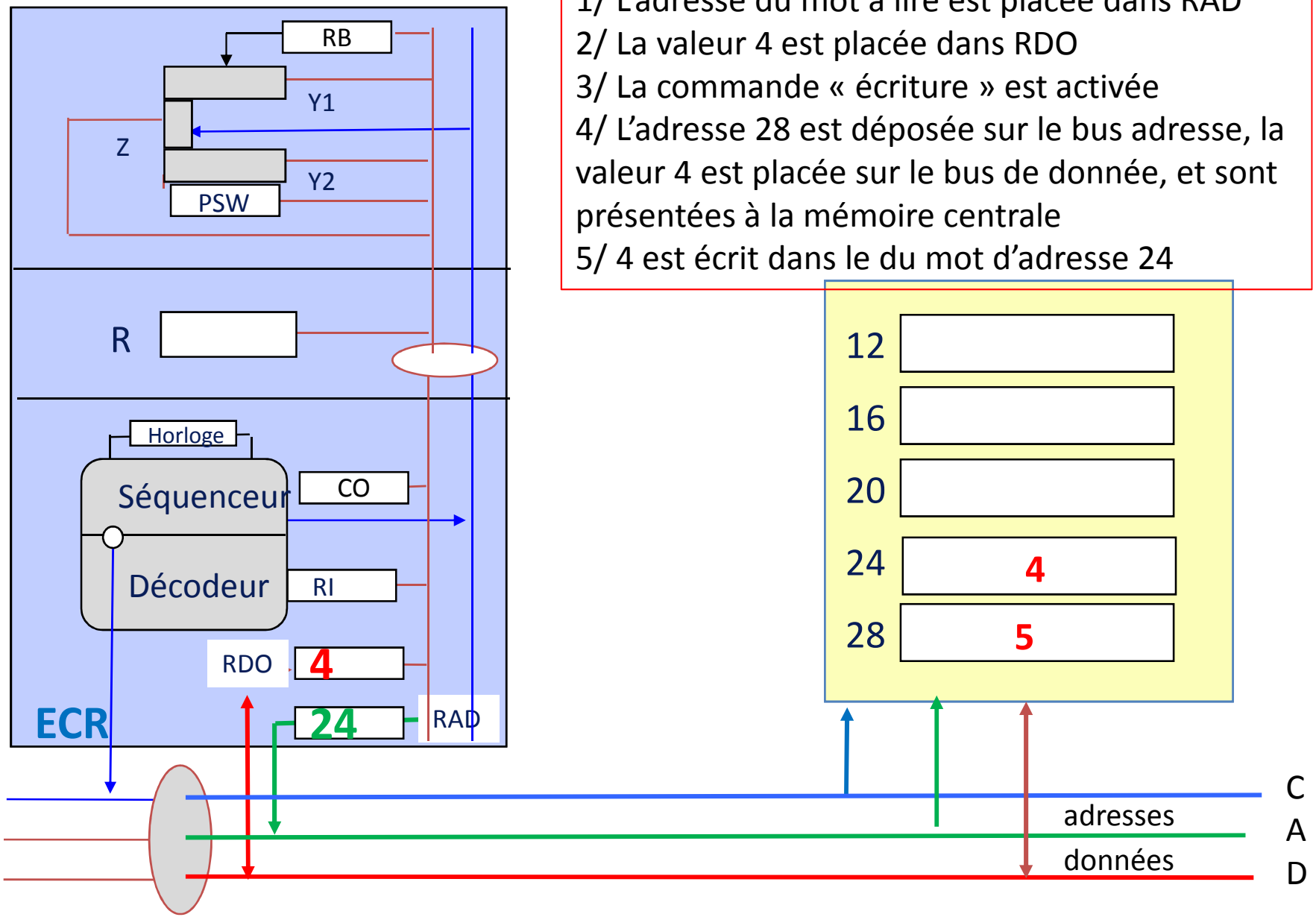
LECTURE

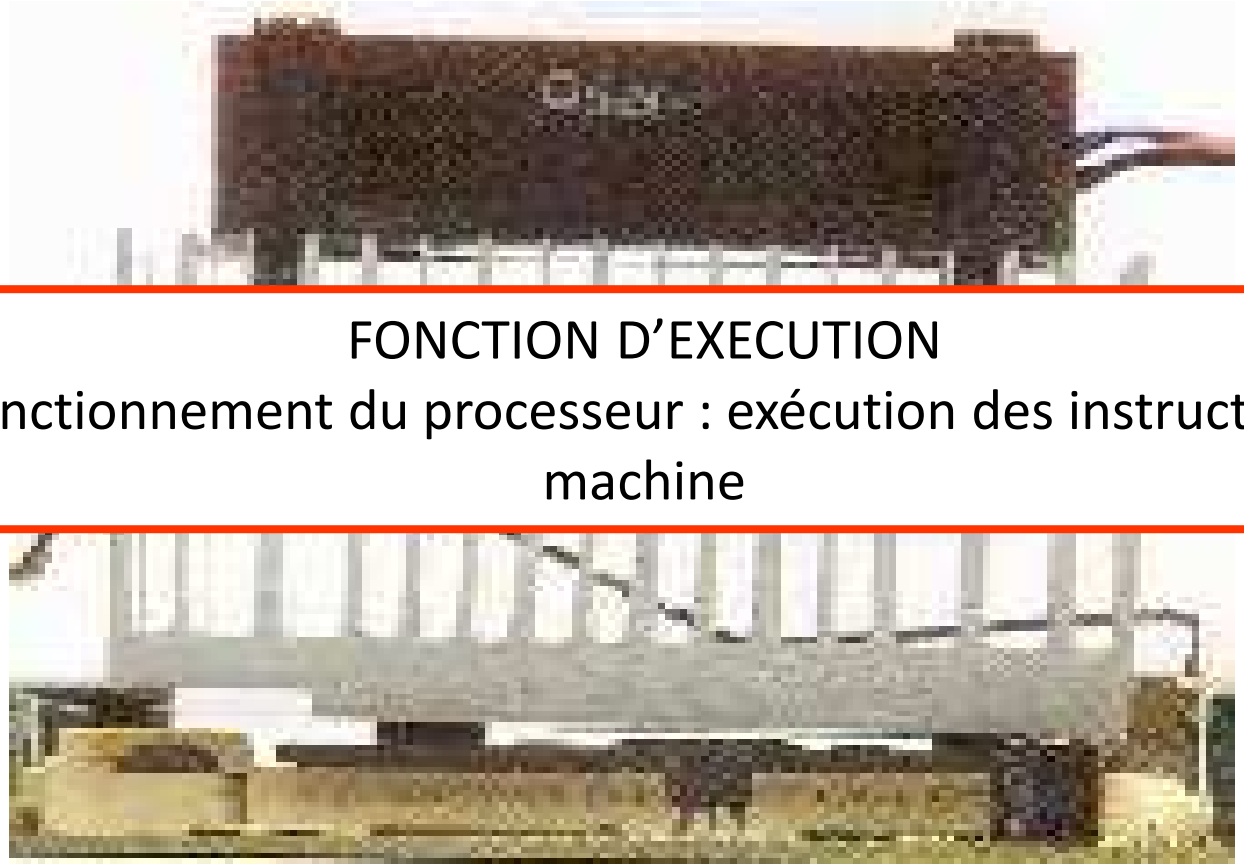
On souhaite Lire le mot d'adresse 28
1/ L'adresse du mot à lire est placée dans RAD
2/ La commande « lecture » est activée
3/ L'adresse 28 est déposée sur le bus adresse, le bus de commande est activé en lecture
4/ le mot d'adresse 28 est sélectionné
5/ Le contenu du mot d'adresse 28 est déposée sur le bus de donnée et copié dans RDO



ECRITURE

On souhaite Ecrire « 4 » dans le mot d'adresse 24
1/ L'adresse du mot à lire est placée dans RAD
2/ La valeur 4 est placée dans RDO
3/ La commande « écriture » est activée
4/ L'adresse 28 est déposée sur le bus adresse, la valeur 4 est placée sur le bus de donnée, et sont présentées à la mémoire centrale
5/ 4 est écrit dans le du mot d'adresse 24



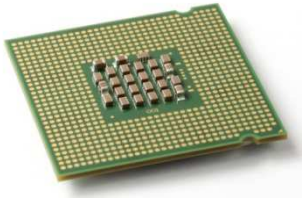


FONCTION D'EXECUTION

Fonctionnement du processeur : exécution des instructions machine

EXECUTION D'UNE INSTRUCTION MACHINE : CHEMIN DE DONNEES DU PROCESSEUR

Exécution des instructions machine

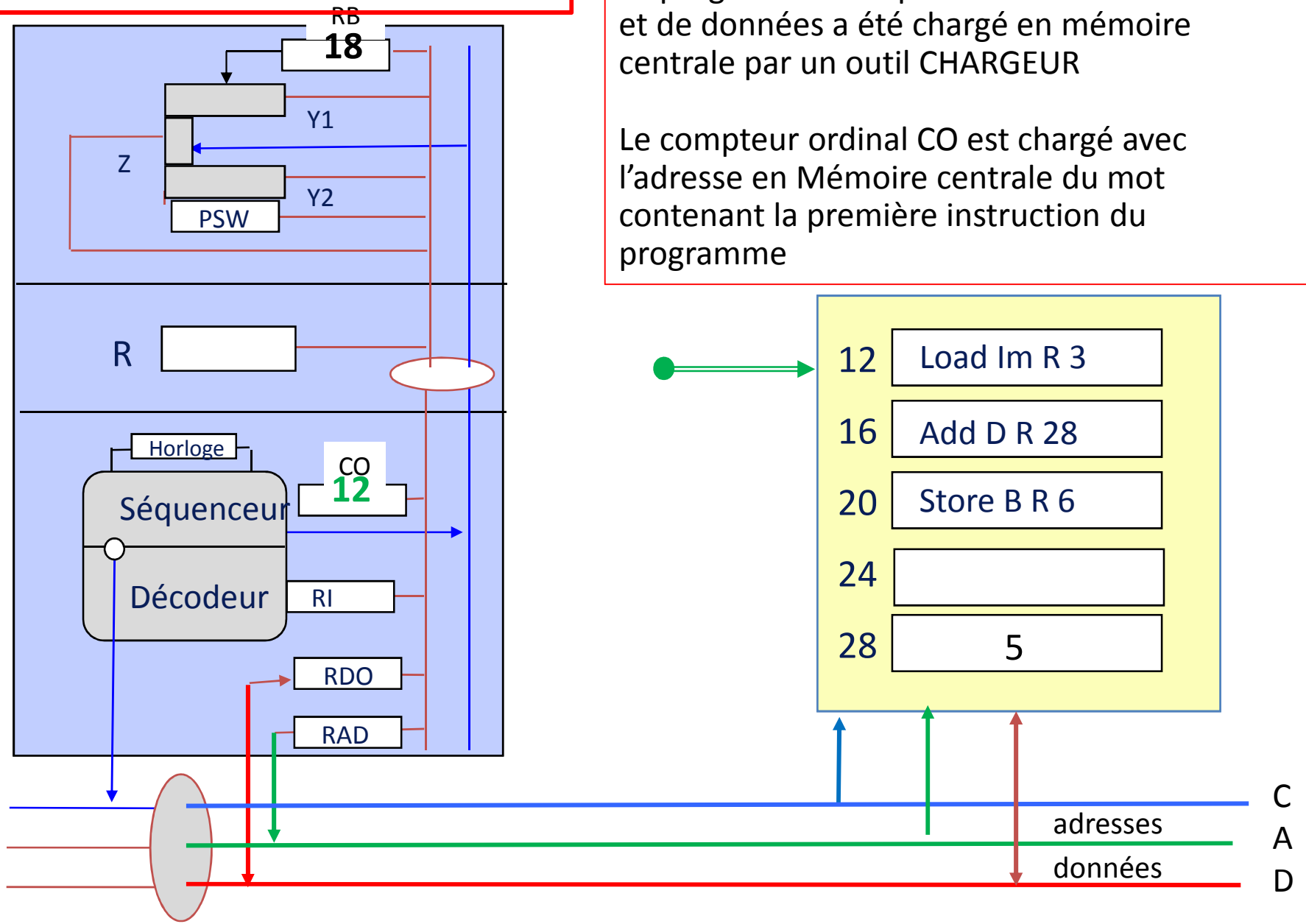


- Le programme composé d'instructions machine et de données a été chargé en mémoire centrale par un outil CHARGEUR
- Le compteur ordinal CO est chargé avec l'adresse en Mémoire centrale du mot contenant la première instruction du programme
- L'exécution du programme s'effectue instruction par instruction, sous le pilotage de l'unité de commande du processeur.
- Le traitement d'une instruction par le processeur se découpe en trois étapes :
  - FETCH : l'instruction est lue en mémoire centrale et copiée dans le registre RI du processeur
  - DECODAGE : l'instruction est reconnue par l'unité de décodage
  - EXECUTION : l'opération correspondant à l'instruction est réalisée

Exécution des instructions machine

Le programme composé d'instructions machine et de données a été chargé en mémoire centrale par un outil CHARGEUR

Le compteur ordinal CO est chargé avec l'adresse en Mémoire centrale du mot contenant la première instruction du programme



Exécution des instructions machine : les trois étapes
Fetch / décodage/ exécution

Début

Lire	la prochaine instruction à exécuter depuis la mémoire et la charger Registre instruction (RI)	Fetch
Modifier	le Compteur Ordinal pour qu'il pointe sur la prochaine instruction à exécuter,	
Décoder	l'instruction qui vient d'être chargée,	Décodage
Charger	les données éventuelles dans les registres internes,	Exécution
Réaliser	l'opération,	

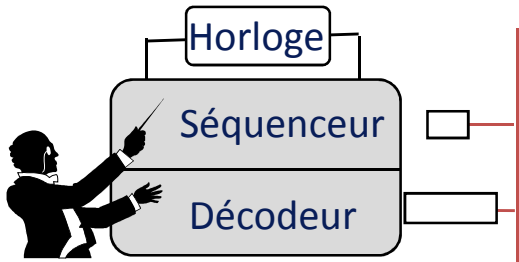
Fin

Exécution des instructions machine : les trois étapes
Fetch / décodage/ exécution

Début

Fetch

Lire	la prochaine instruction à exécuter depuis la mémoire et la charger Registre instruction (RI)
Modifier	le Compteur Ordinal (CO) pour qu'il pointe sur la prochaine instruction à exécuter,



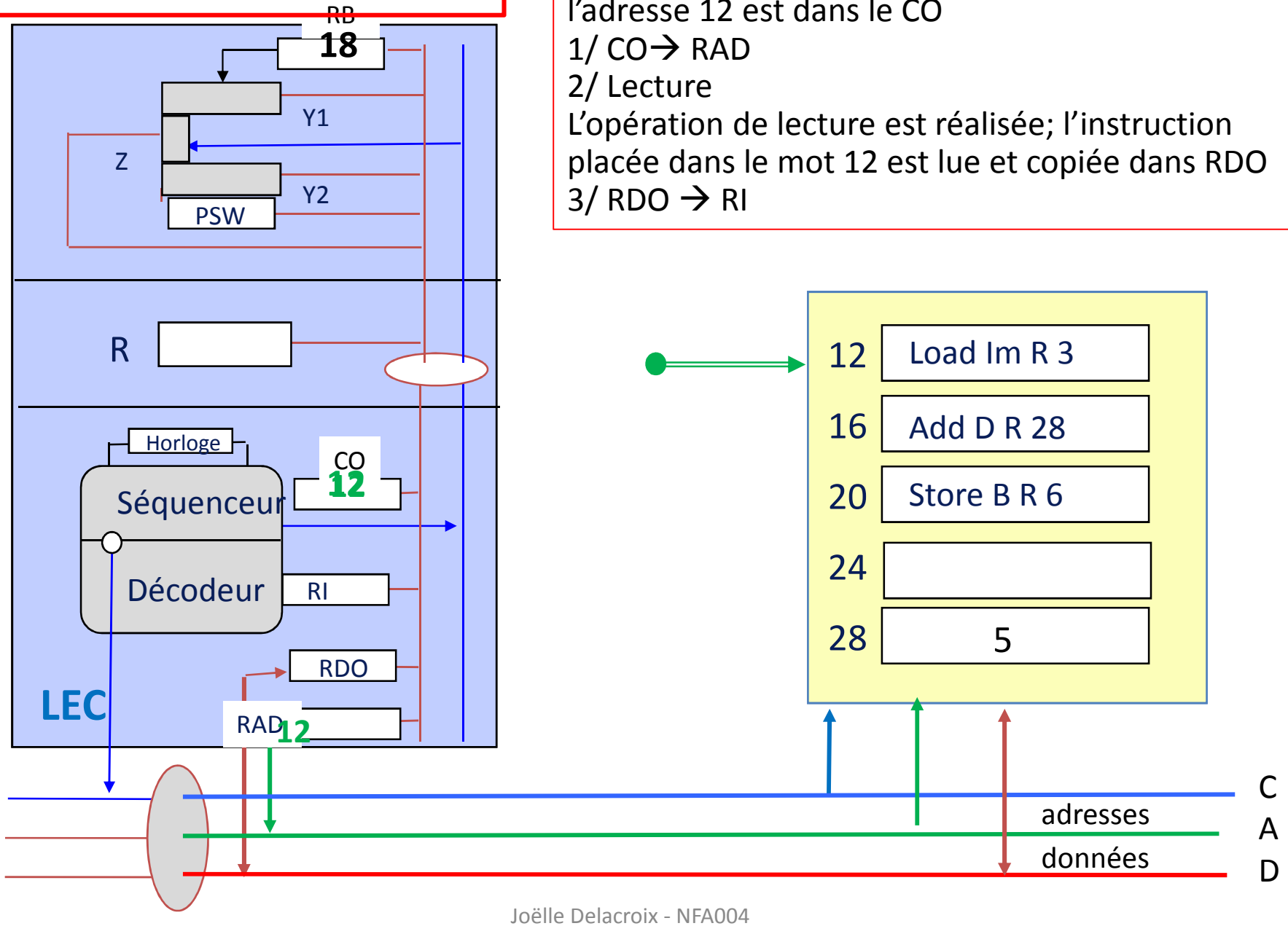
Compteur Ordinal : contient l'adresse de la prochaine instruction a exécuter

Registre Instruction : contient l'instruction en cours d'exécution

Bus

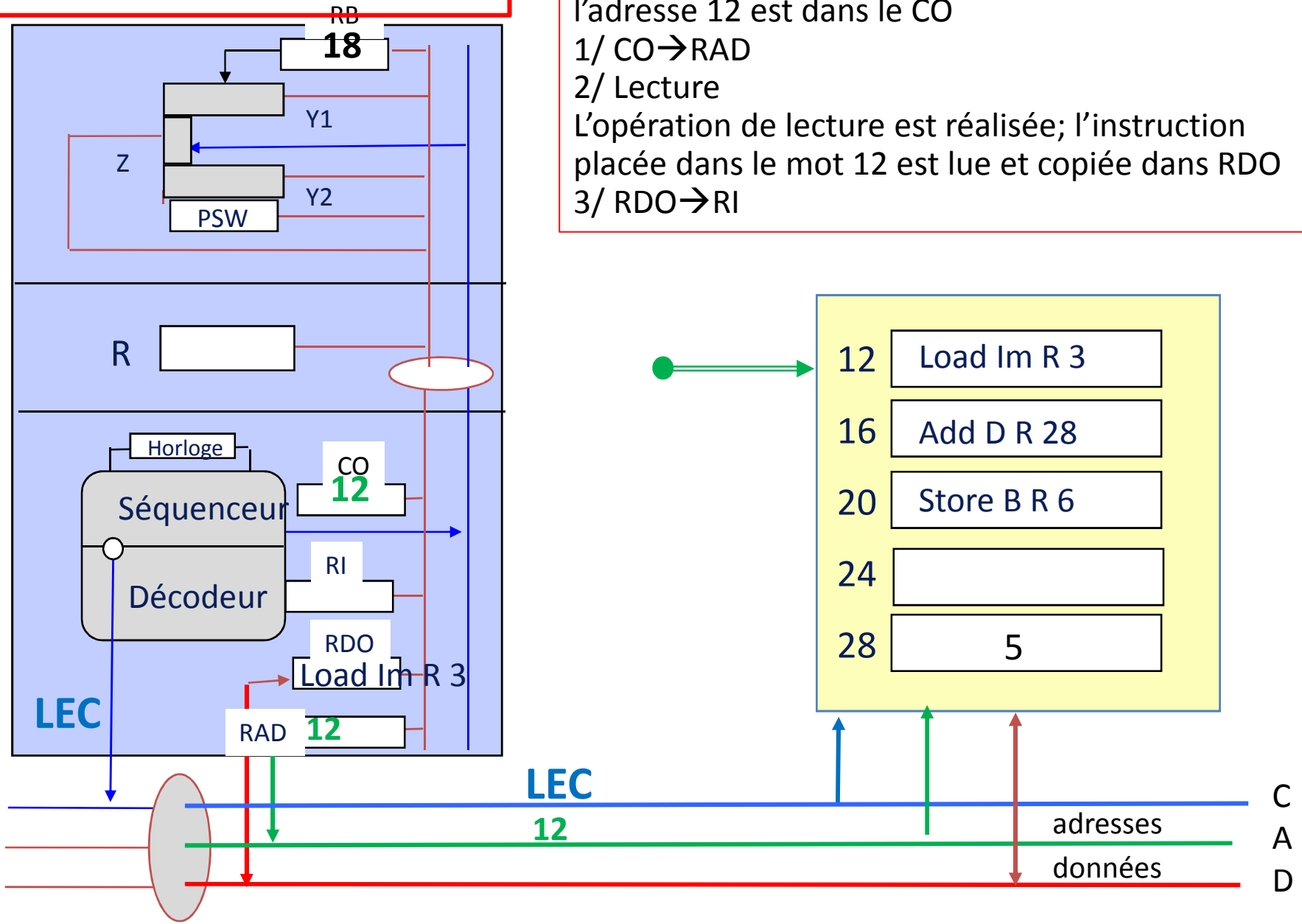
Load Im R 3 – 1/ Fetch

On lit le mot d'adresse 12 qui contient l'instruction; l'adresse 12 est dans le CO
1/ CO→ RAD
2/ Lecture
L'opération de lecture est réalisée; l'instruction placée dans le mot 12 est lue et copiée dans RDO
3/ RDO → RI



Load Im R 3 – 1/ Fetch

On lit le mot d'adresse 12 qui contient l'instruction; l'adresse 12 est dans le CO
1/ CO→RAD
2/ Lecture
L'opération de lecture est réalisée; l'instruction placée dans le mot 12 est lue et copiée dans RDO
3/ RDO→RI

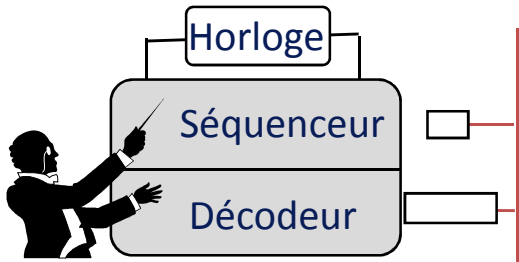


Exécution des instructions machine : les trois étapes
Fetch / décodage/ exécution

Début

Fetch

Lire	la prochaine instruction à exécuter depuis la mémoire et la charger Registre instruction (RI)
Modifier	le Compteur Ordinal (CO) pour qu'il pointe sur la prochaine instruction à exécuter,



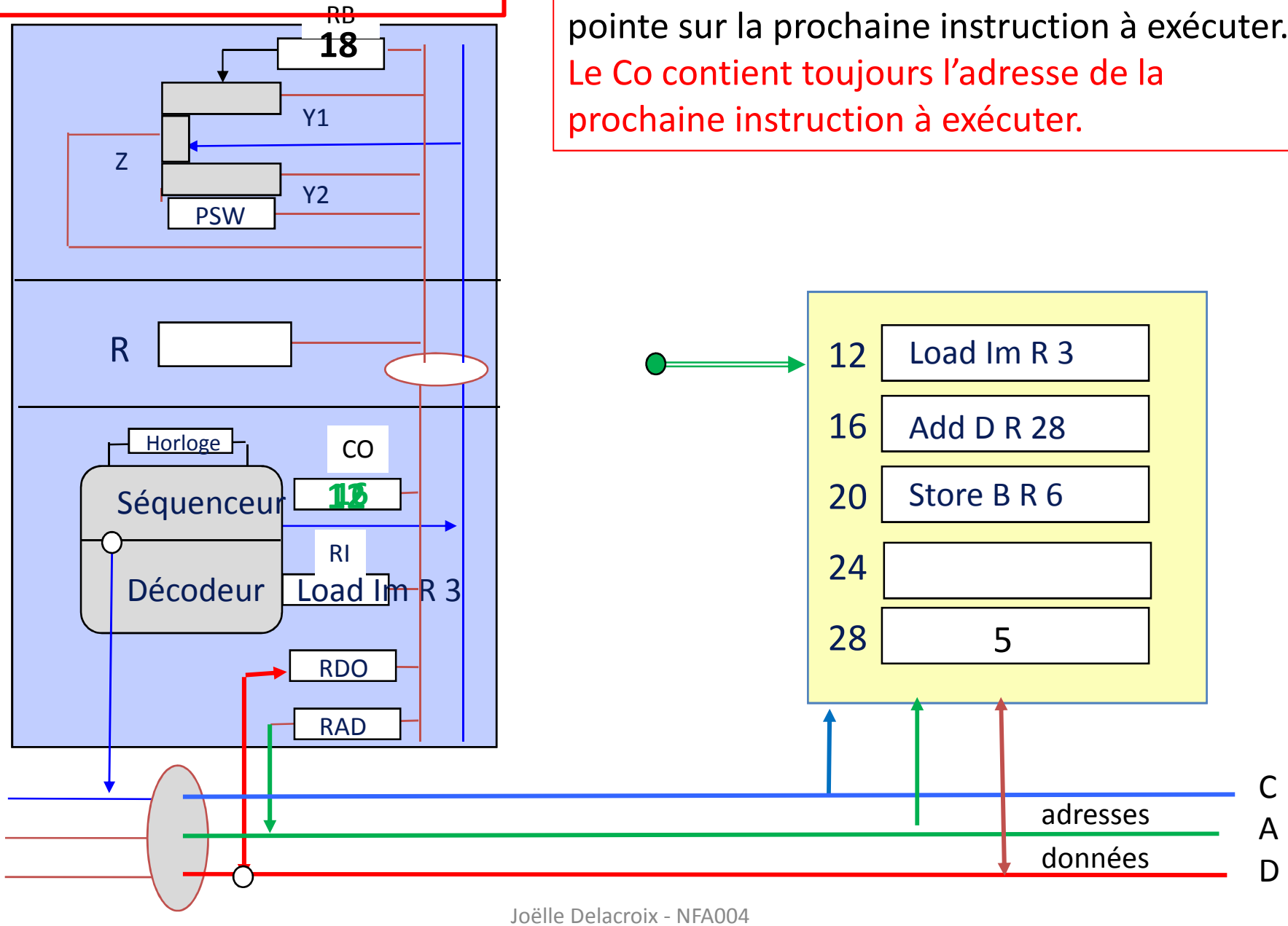
Compteur Ordinal : contient l'adresse de la prochaine instruction a exécuter

Registre Instruction : contient l'instruction en cours d'exécution

Bus

Load Im R 3 – 1/ Fetch

Modifier le Compteur Ordinal (CO) pour qu'il pointe sur la prochaine instruction à exécuter.
Le Co contient toujours l'adresse de la prochaine instruction à exécuter.



Exécution des instructions machine : les trois étapes
Fetch / décodage/ exécution

Début

Lire	la prochaine instruction à exécuter depuis la mémoire et la charger Registre instruction (RI)
Modifier	le Compteur Ordinal pour qu'il pointe sur la prochaine instruction à exécuter,

Fetch

CO → RAD
Lecture
RDO → RI
Incrémentation CO

Exécution des instructions machine : les trois étapes
Fetch / décodage/ exécution

Début

Décoder	l'instruction qui vient d'être chargée,	Décodage
Charger	les données éventuelles dans les registres internes,	Exécution
Réaliser	l'instruction,	

Fin

LOAD Im R 3 : l'adressage est un mode immédiat. Il n'y a pas d'opérandes en mémoire centrale.

Exécution des instructions machine : les trois étapes
Fetch / décodage/ exécution

Début

Décoder l'instruction qui vient d'être chargée,

Décodage

Charger les données éventuelles dans les registres internes,

Réaliser l'instruction,

Exécution

Fin

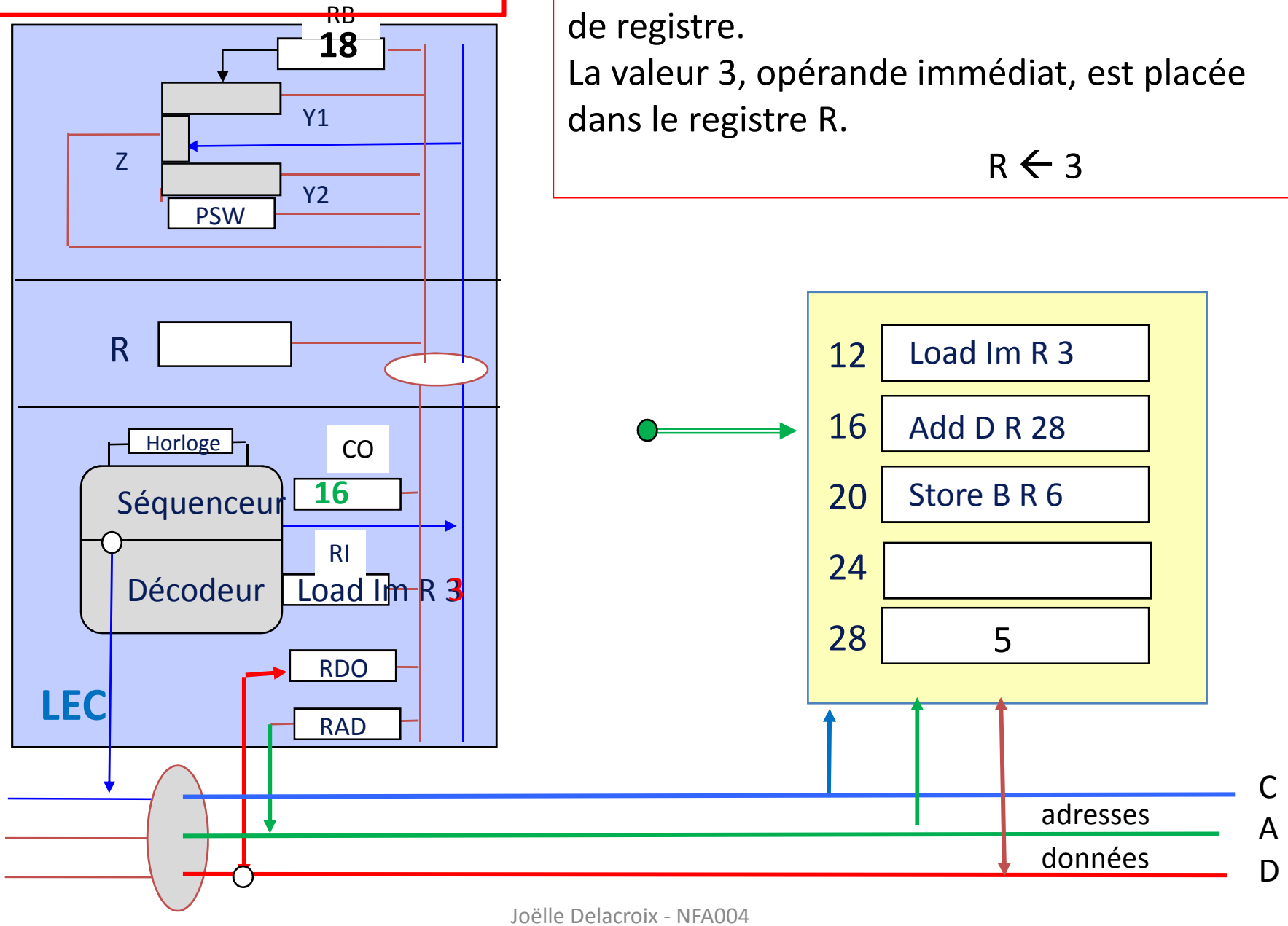
LOAD Im R 3 : l'opération est un chargement de registre.
La valeur 3, opérande immédiat, est placée dans le registre R.

R ← 3

Load Im R 3 – 3/ réaliser

LOAD Im R 3 : l'opération est un chargement de registre.
La valeur 3, opérande immédiat, est placée dans le registre R.

$R \leftarrow 3$



Exécution des instructions machine : les trois étapes
Fetch / décodage/ exécution
LOAD Im R 3

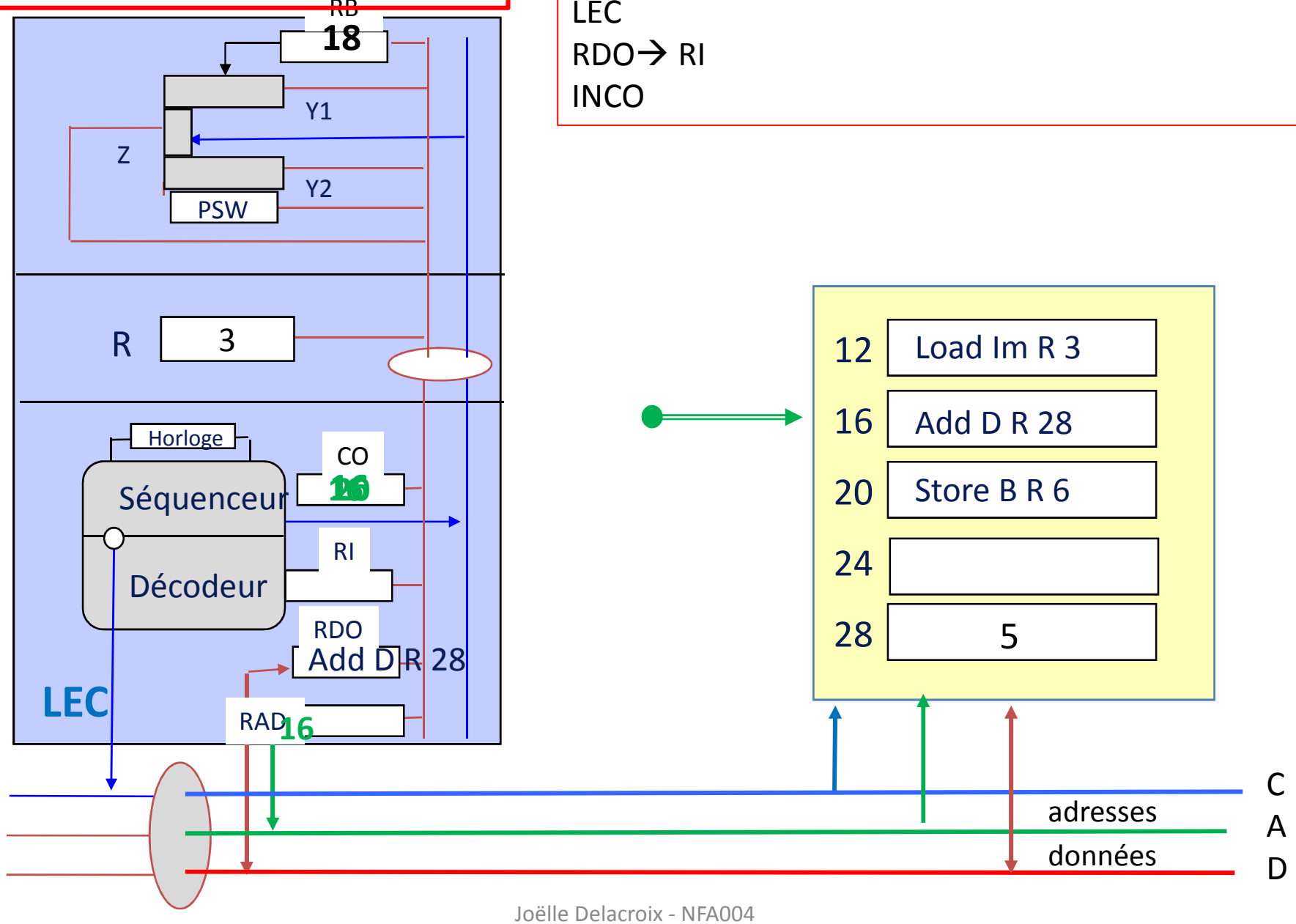
Début

Lire	la prochaine instruction à exécuter depuis la mémoire et la charger Registre instruction (RI)	Fetch CO→ RAD LEC RDO→ RI INCO
Modifier	le Compteur Ordinal pour qu'il pointe sur la prochaine instruction à exécuter,	
Décoder	l'instruction qui vient d'être chargée,	Décodage
Charger	les données éventuelles dans les registres internes,	Exécution 3 → RI
Réaliser	l'opération,	

Fin

ADD D R 28 – 1/ Fetch

CO→ RAD
LEC
RDO→ RI
INCO



Exécution des instructions machine : les trois étapes
Fetch / décodage/ exécution

Début

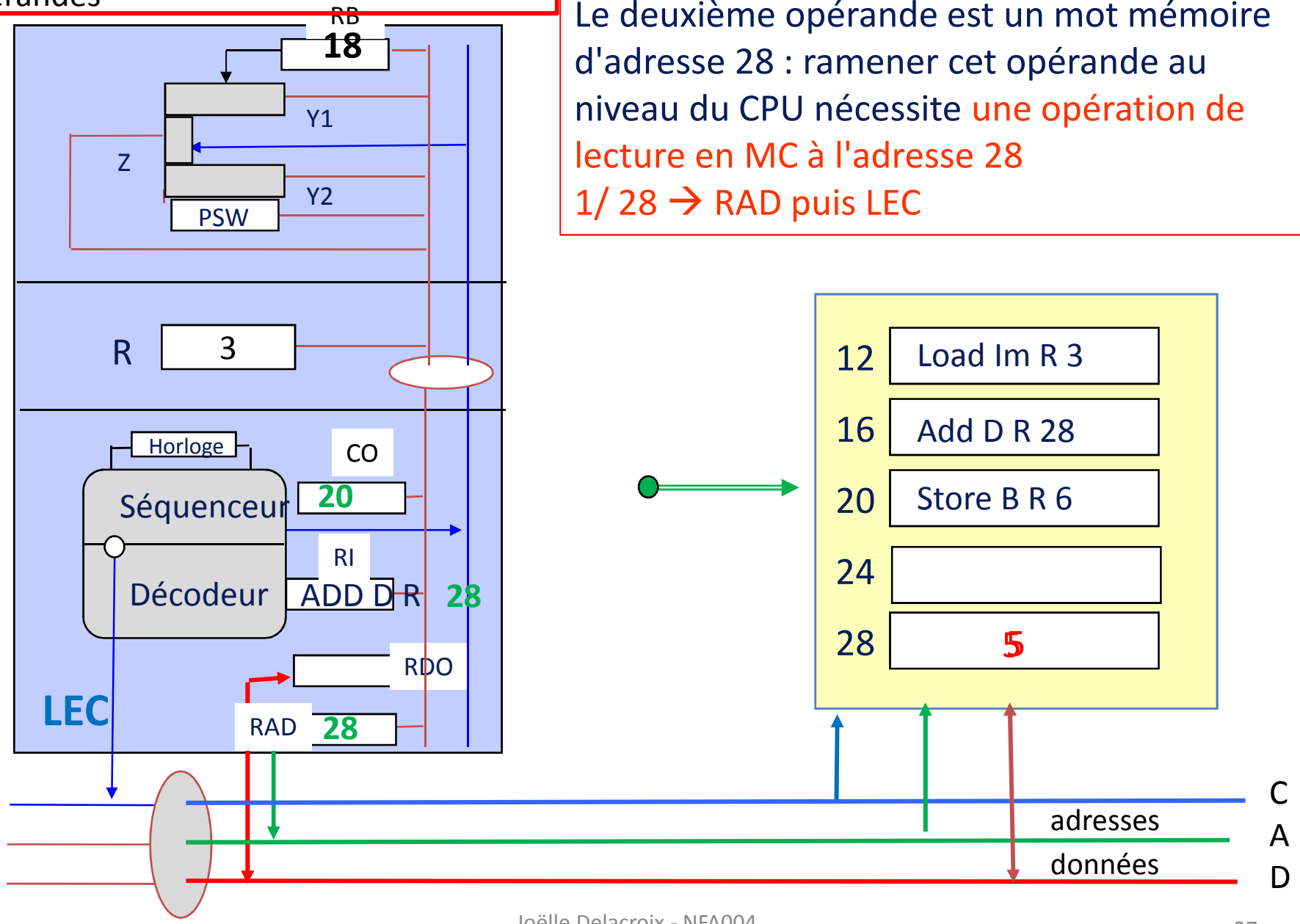
Décoder	l'instruction qui vient d'être chargée,	Décodage
Charger	les données éventuelles dans les registres internes,	Exécution
Réaliser	l'instruction,	

Fin

Add D R 28 : l'adressage est un mode direct. Un premier opérande est le contenu de R. Le deuxième opérande est un mot mémoire d'adresse 28 : ramener cet opérande au niveau du CPU nécessite une opération de lecture en MC à l'adresse 28.

Add D R 28 – 2/ rechercher les opérandes

Add D R 28 : l'adressage est un mode direct. Le deuxième opérande est un mot mémoire d'adresse 28 : ramener cet opérande au niveau du CPU nécessite une opération de lecture en MC à l'adresse 28
1/ 28 → RAD puis LEC



Exécution des instructions machine : les trois étapes
Fetch / décodage/ exécution

Début

Décoder l'instruction qui vient d'être chargée, — Décodage

Charger les données éventuelles dans les registres internes,

Réaliser l'instruction, — Exécution

Fin

Add D R 28 : 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 R.

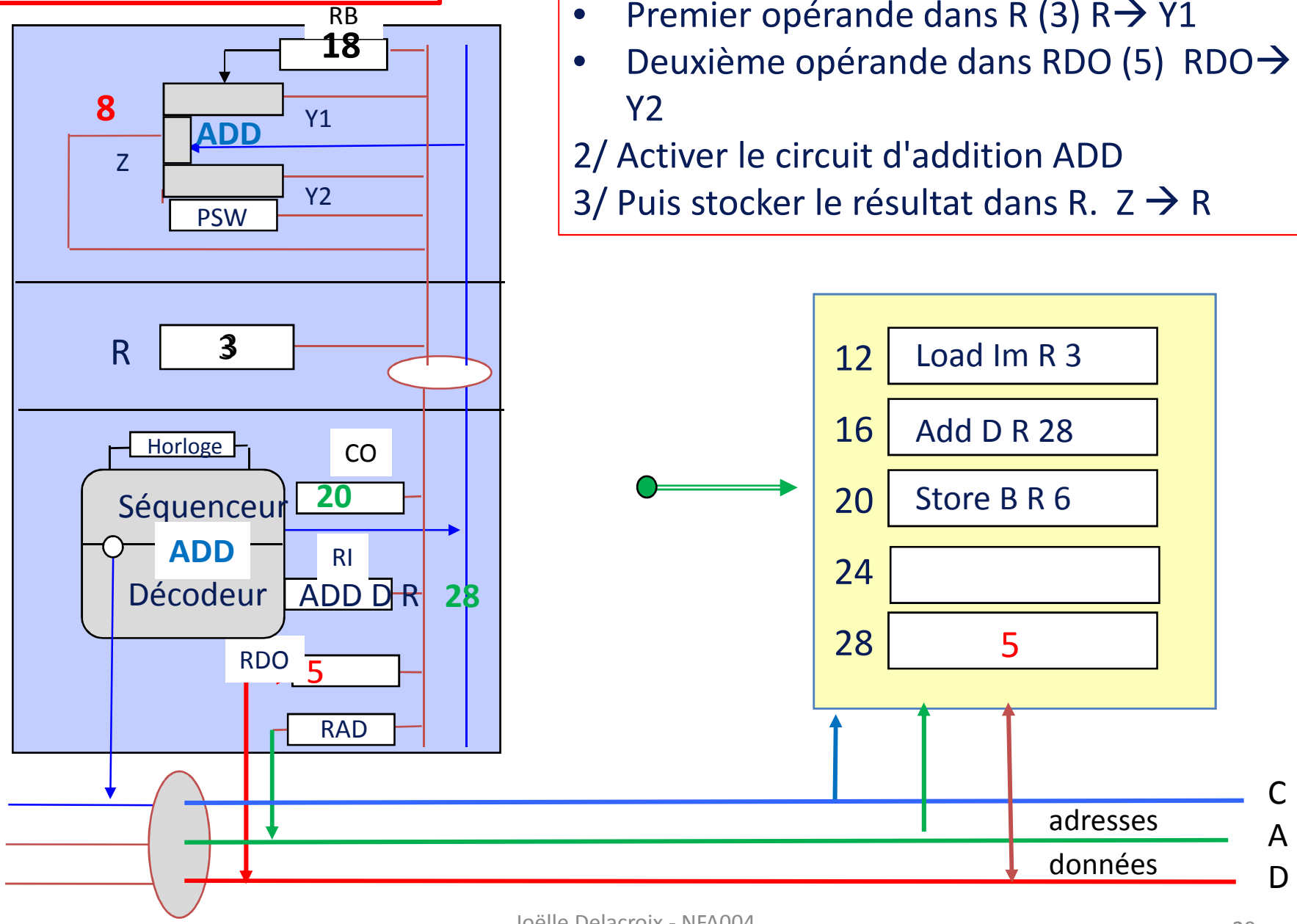
- Premier opérande dans R (3)
- Deuxième opérande dans RDO (5)

Add D R 28 – 3/ Réalisation

1/ Il faut amener les deux opérandes à l'UAL.
- Premier opérande dans R (3) R→ Y1
- Deuxième opérande dans RDO (5) RDO→ Y2

2/ Activer le circuit d'addition ADD

3/ Puis stocker le résultat dans R. Z → R



Exécution des instructions machine : les trois étapes
Fetch / décodage/ exécution
ADD D R 28

Début

Lire	la prochaine instruction à exécuter depuis la mémoire et la charger Registre instruction (RI)	Fetch CO→ RAD LEC RDO→ RI INCO
Modifier	le Compteur Ordinal pour qu'il pointe sur la prochaine instruction à exécuter,	
Décoder	l'instruction qui vient d'être chargée,	Décodage
Charger	les données éventuelles dans les registres internes,	Exécution 28 → RAD LEC R→ Y1 RDO→ Y2 ADD Z→ R
Réaliser	l'opération,	

Fin

Exécution des instructions machine : les trois étapes
Fetch / décodage/ exécution
STORE B R 6

Début

Décoder	l'instruction qui vient d'être chargée,	Décodage
Charger	les données éventuelles dans les registres internes,	Exécution
Réaliser	l'instruction,	

Fin

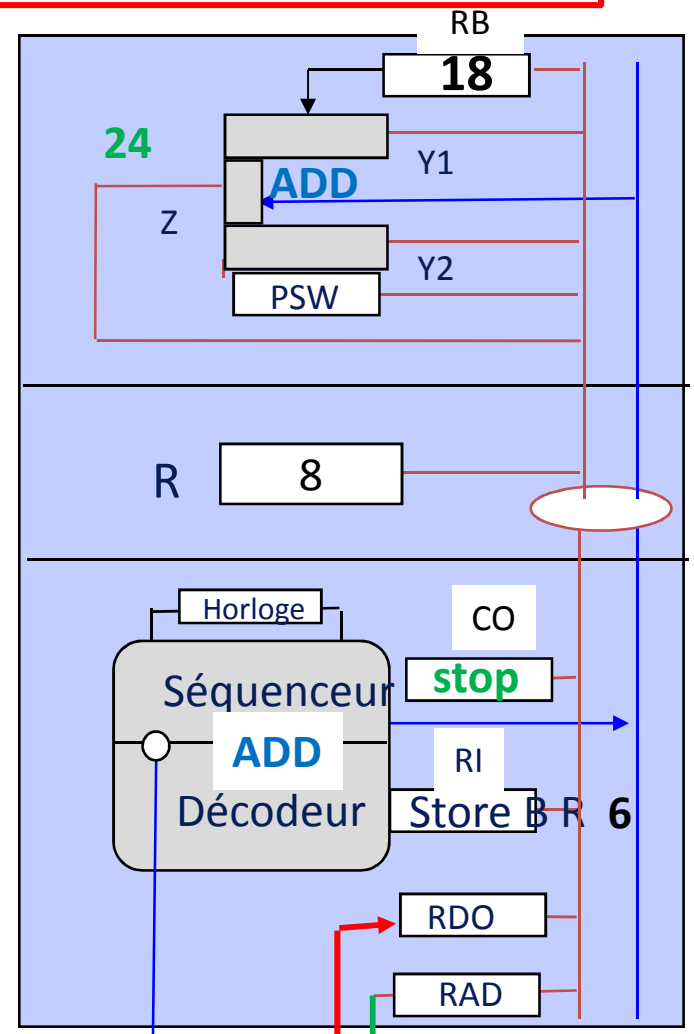
Store B R 6 : l'adressage est un mode basé. On écrit le contenu de R dans le mot mémoire d'adresse C = (RB) + 6

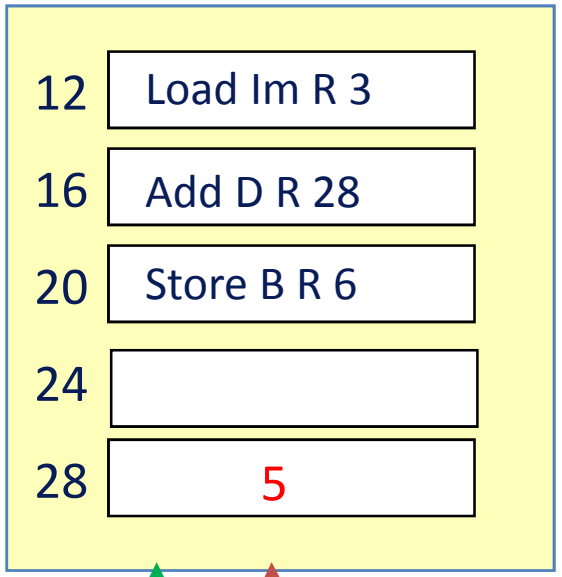
STORE B R 6– 3/ Réalisation

On écrit le contenu de R
dans le mot mémoire d'adresse C = (RB) + 6
1/ Calcul de (RB) + 6

RB→Y1
6 → Y2
ADD

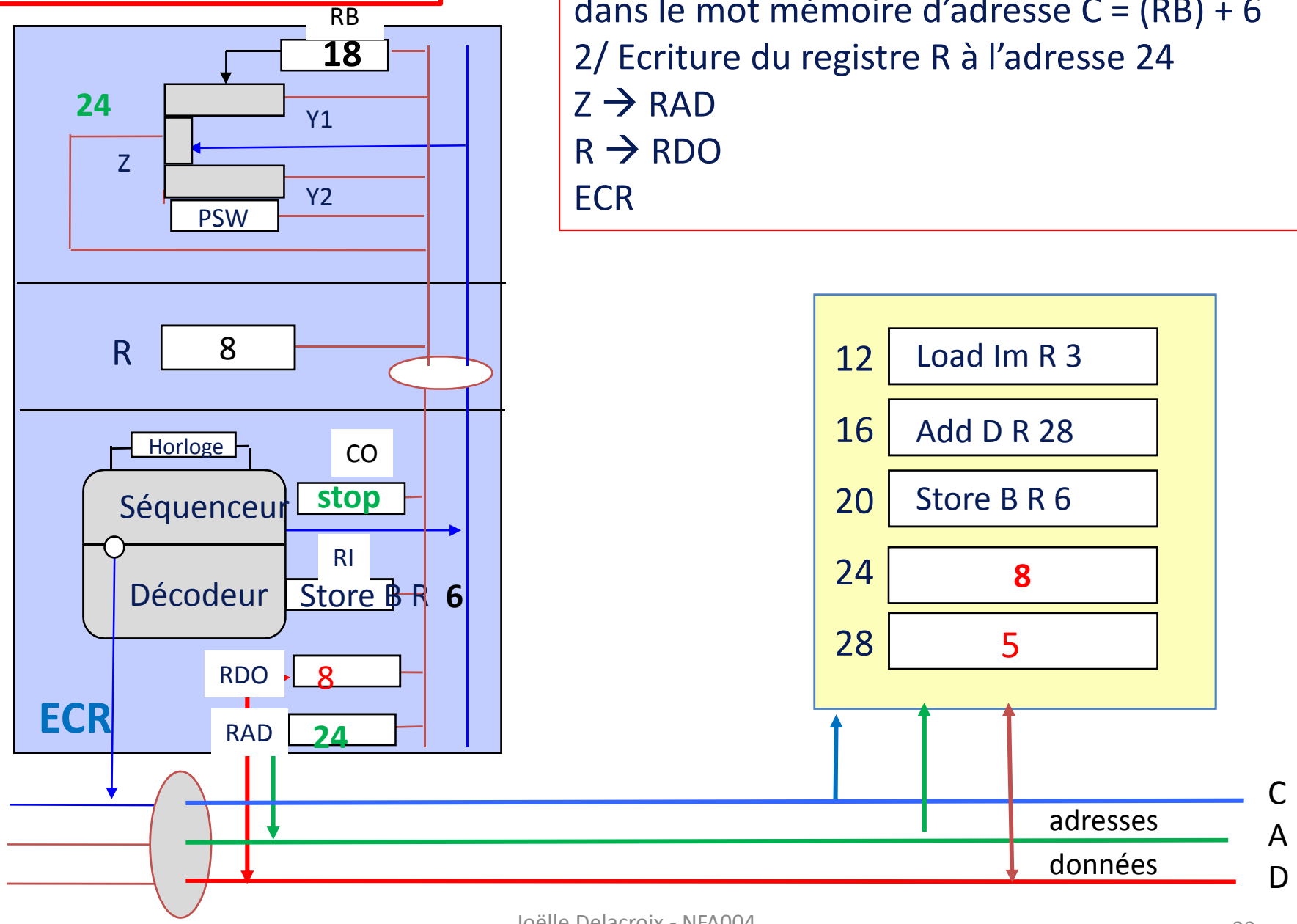
La sortie Z contient l'adresse d'écriture





STORE B R 6– 3/ Réalisation

On écrit le contenu de R
dans le mot mémoire d'adresse C = (RB) + 6
2/ Ecriture du registre R à l'adresse 24
Z → RAD
R → RDO
ECR

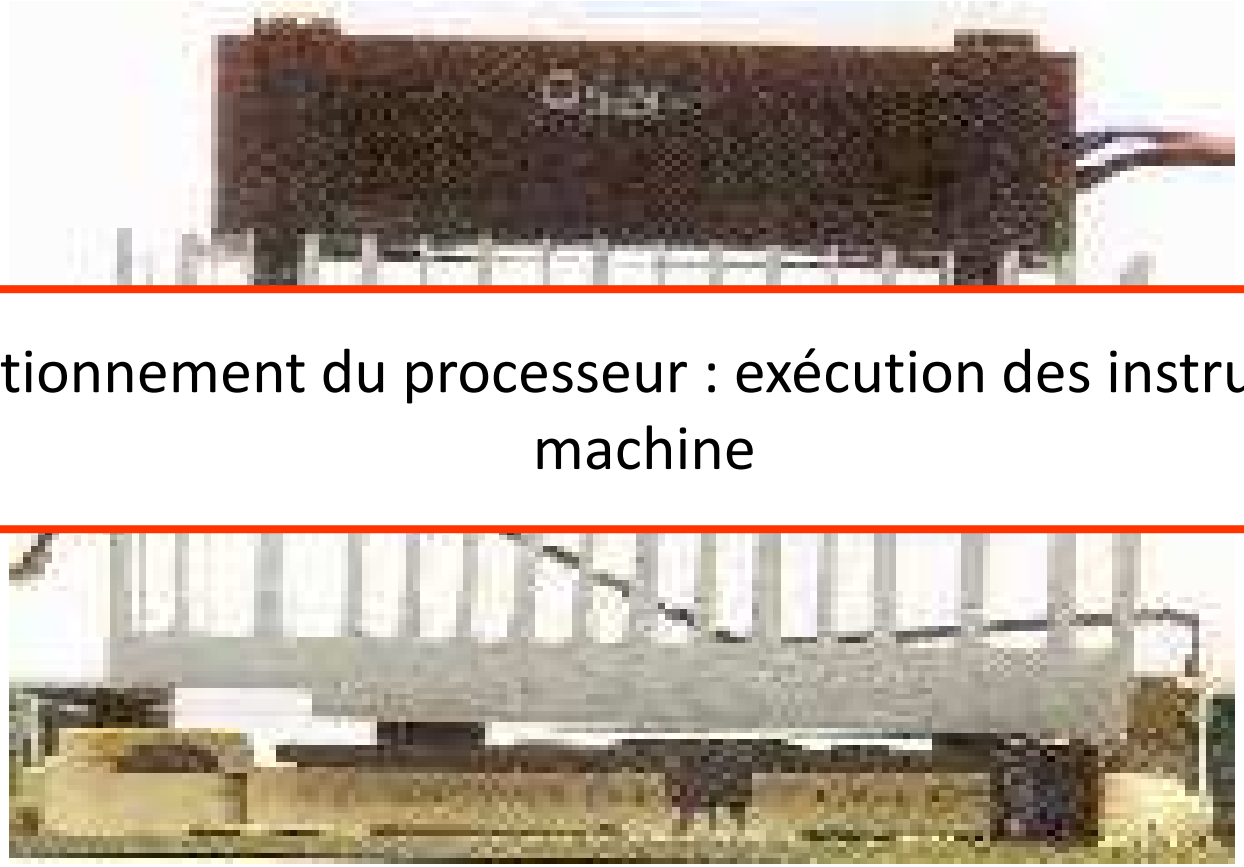


Exécution des instructions machine : les trois étapes
Fetch / décodage/ exécution
STORE B R 6

Début

Lire	la prochaine instruction à exécuter depuis la mémoire et la charger Registre instruction (RI)	Fetch CO→ RAD LEC RDO→ RI INCO
Modifier	le Compteur Ordinal pour qu'il pointe sur la prochaine instruction à exécuter,	
Décoder	l'instruction qui vient d'être chargée,	Décodage
Charger	les données éventuelles dans les registres internes,	Exécution RB→Y1 6→ Y2 ADD Z→ RAD R→ RDO ECR
Réaliser	l'opération,	

Fin

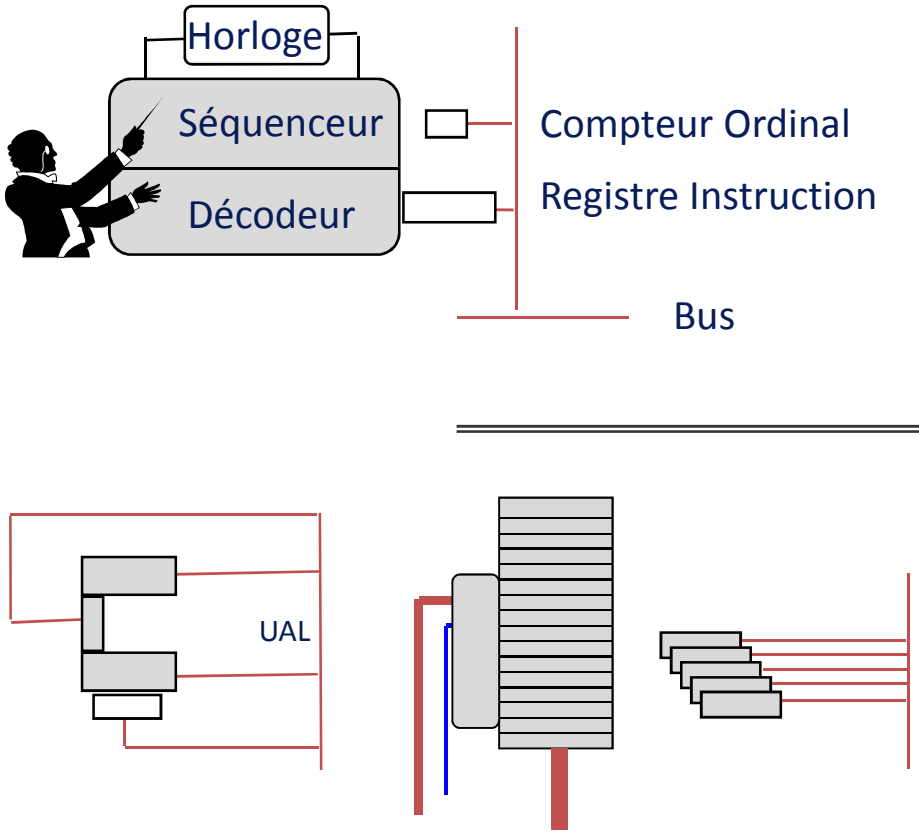


Fonctionnement du processeur : exécution des instructions machine

EXECUTION D'UNE INSTRUCTION MACHINE : LES MICROCOMMANDES DU PROCESSEUR

Exécution d'une instruction machine : les micro-commandes du processeur

L'unité de commande (séquenceur) génère, en fonction des tops horloge, des micro-commandes pour piloter les registres, l'Ual et la mémoire centrale



L' unité de commande :

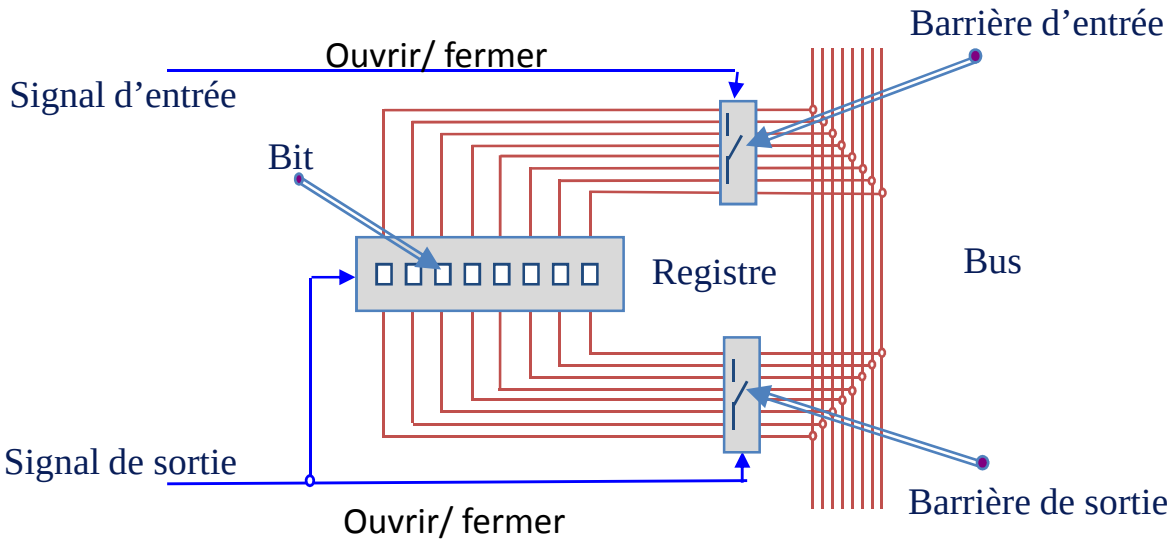
- Charger une instruction : fetch
- Décoder l'instruction : décodage
- Exécuter l'instruction: exécution

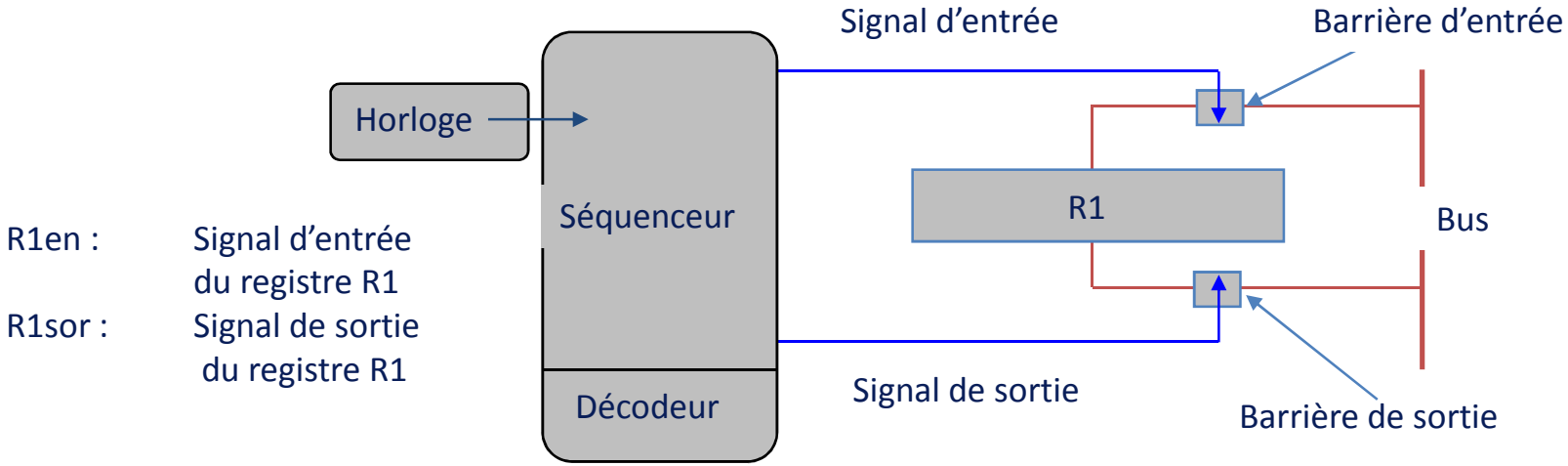
Micro commandes

Activer une opération sur les opérandes
Activer une opération sur la mémoire
Ouvrir / Fermer les registres

Exécution d'une instruction machine : les micro-commandes du processeur

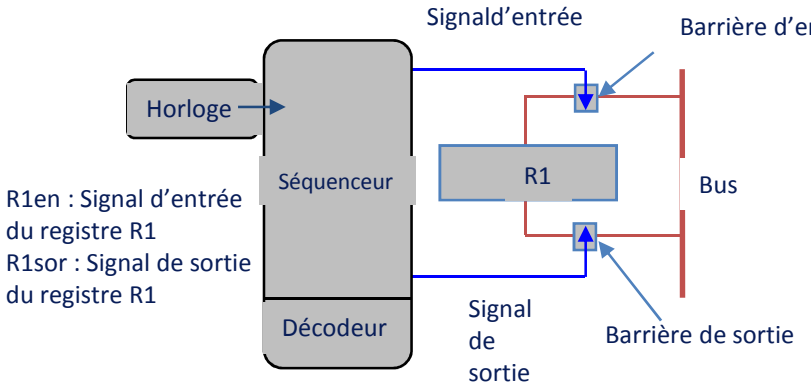
Registre (principe de fonctionnement)

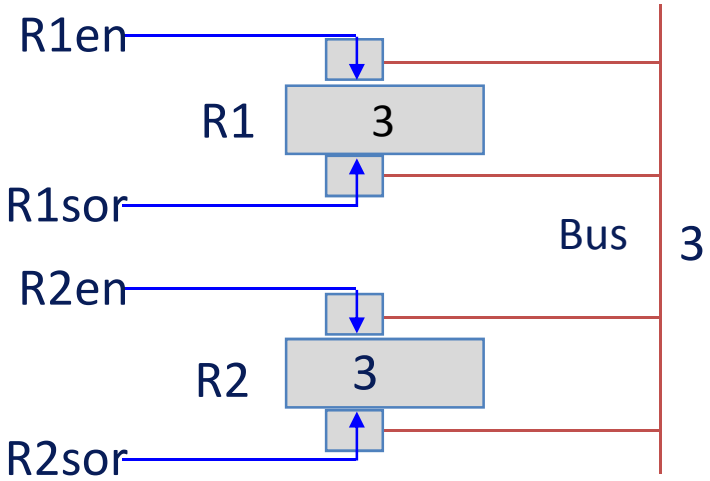




Exécution d'une instruction machine : les micro-commandes du processeur

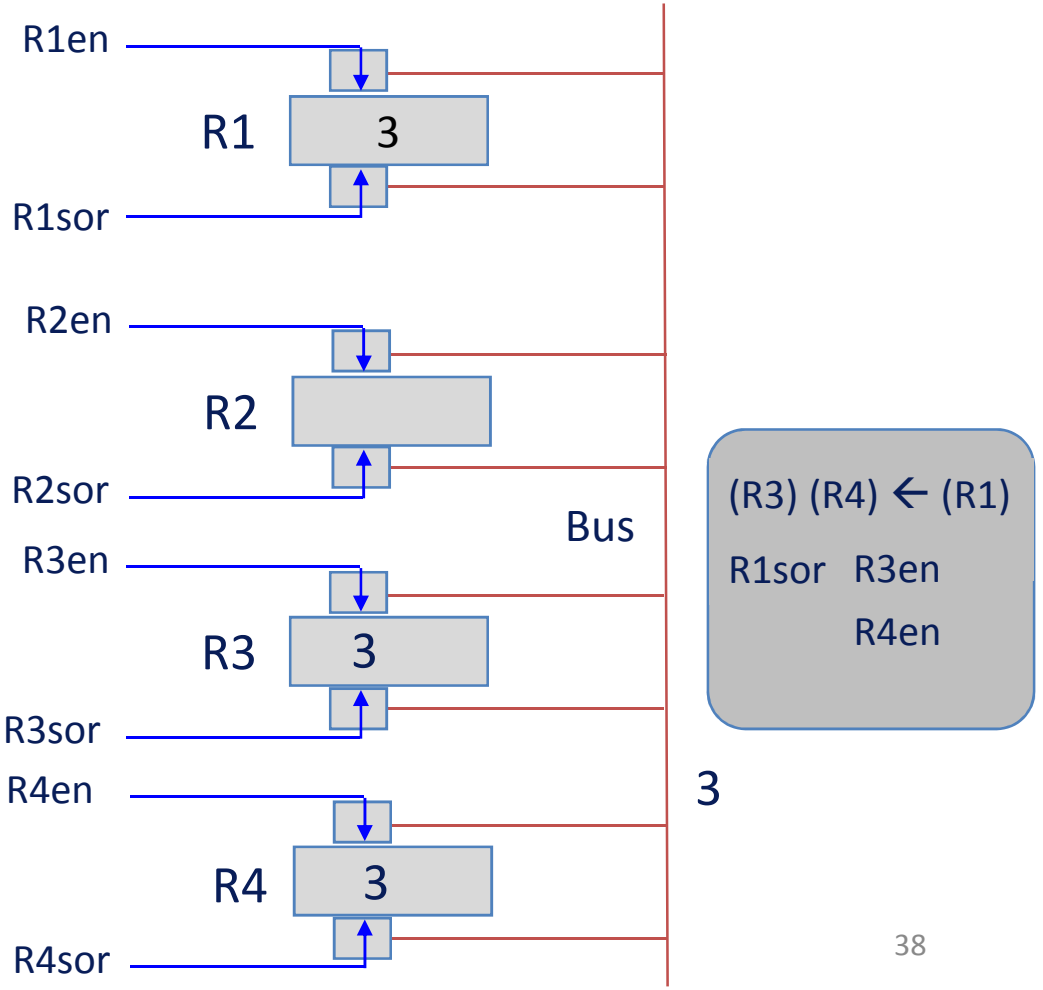
Registre (principe de fonctionnement)





(R2) ← (R1)

R1sor R2en



(R3) (R4) ← (R1)

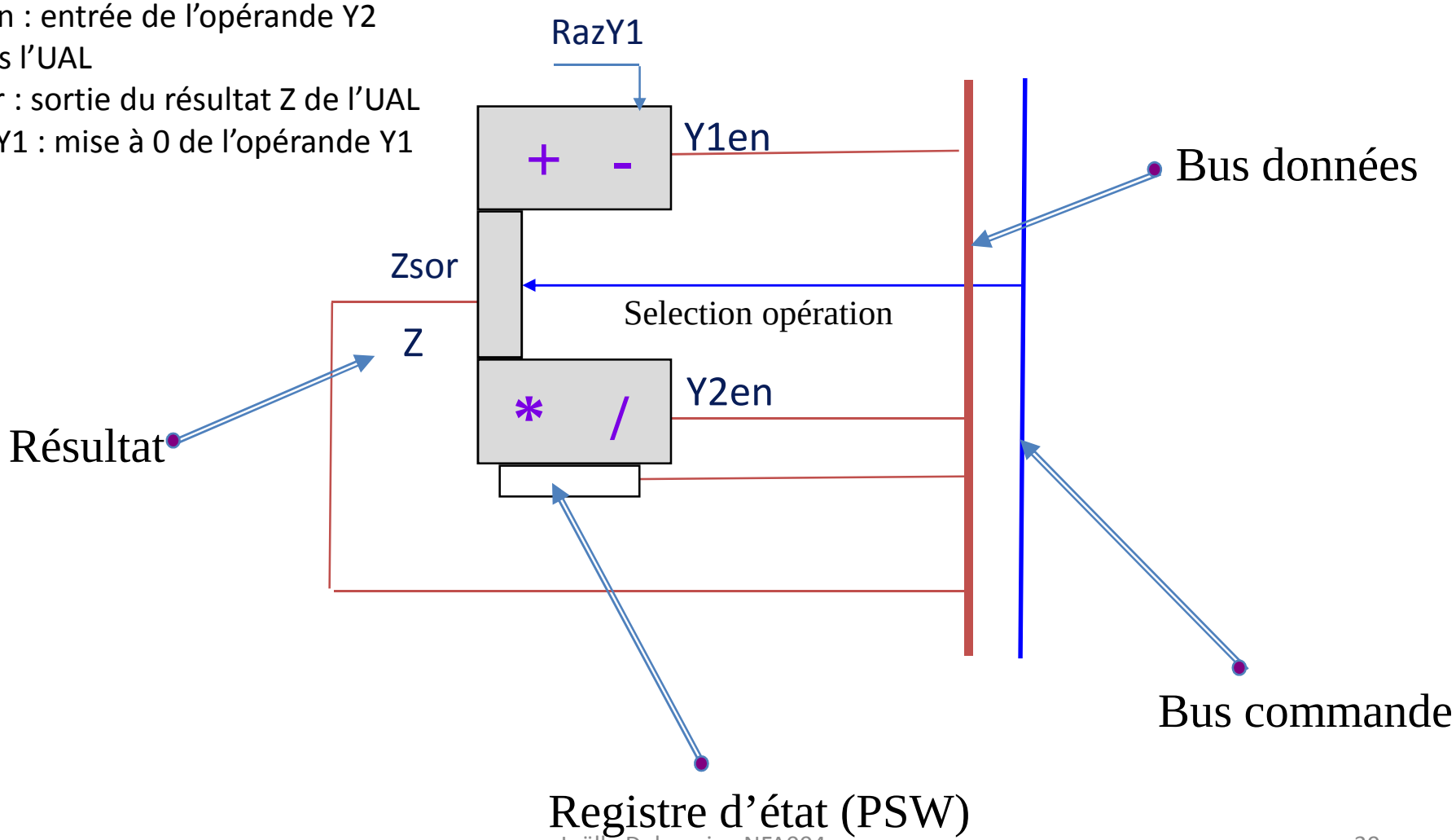
R1sor R3en

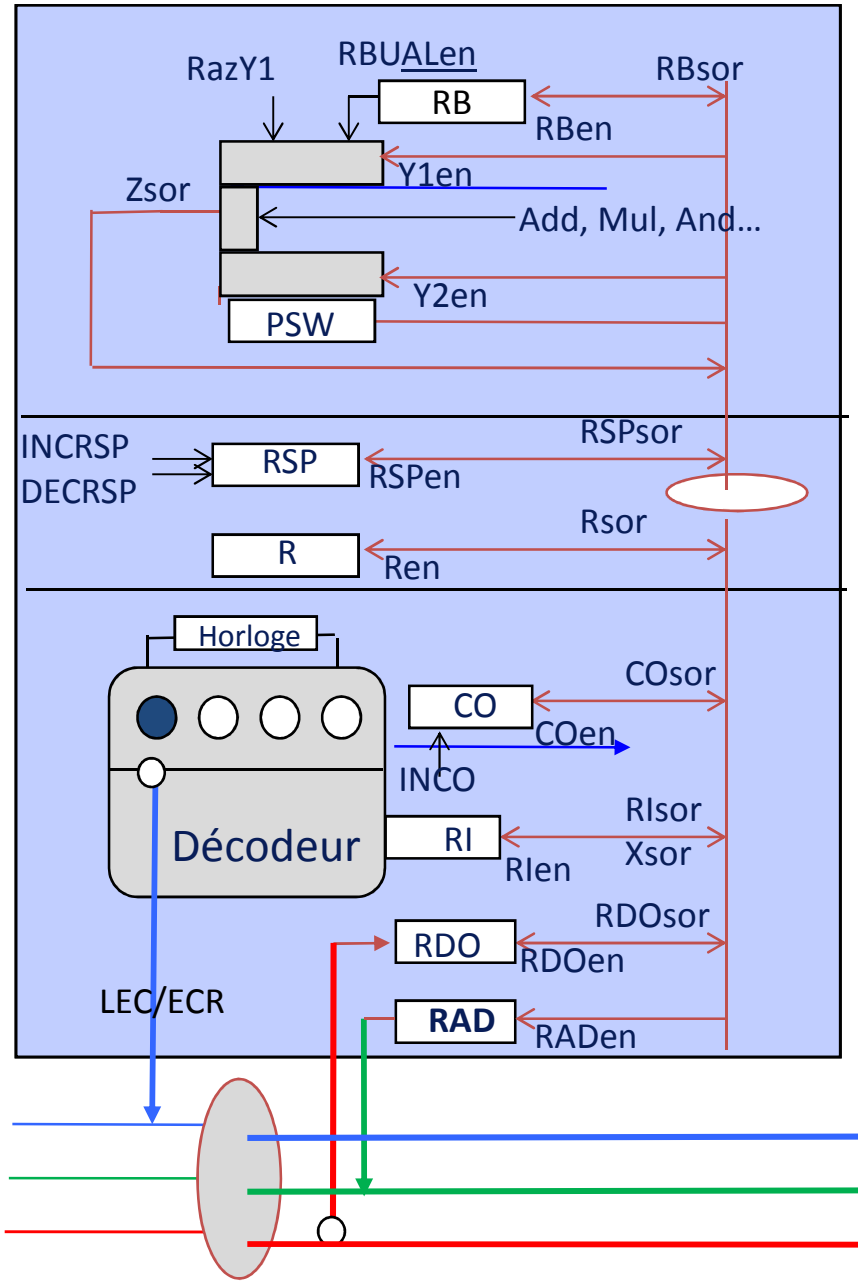
R4en

Exécution d'une instruction machine : les micro-commandes du processeur

UAL (principe de fonctionnement

Y1en : entrée de l'opérande Y1 dans l'UAL
Y2en : entrée de l'opérande Y2 dans l'UAL
Zsor : sortie du résultat Z de l'UAL
RazY1 : mise à 0 de l'opérande Y1

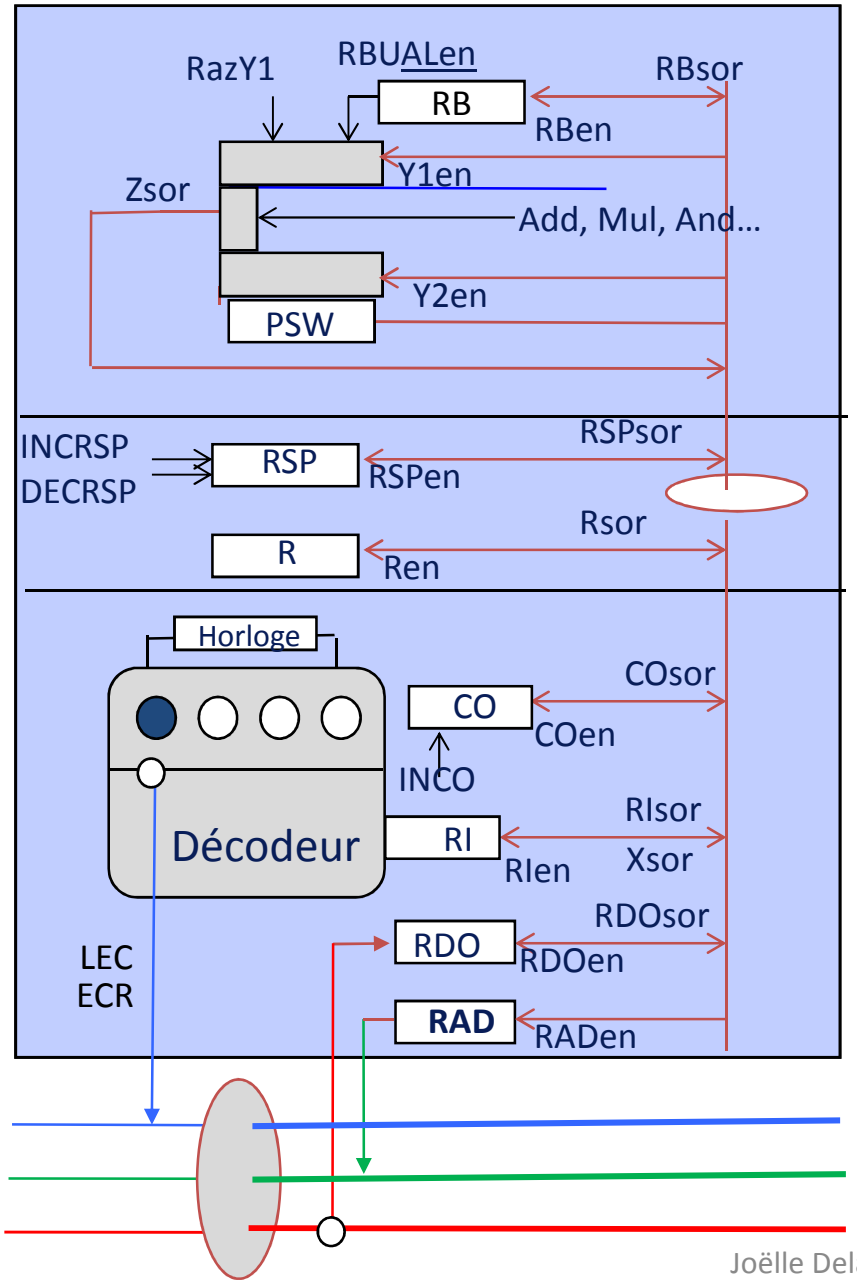




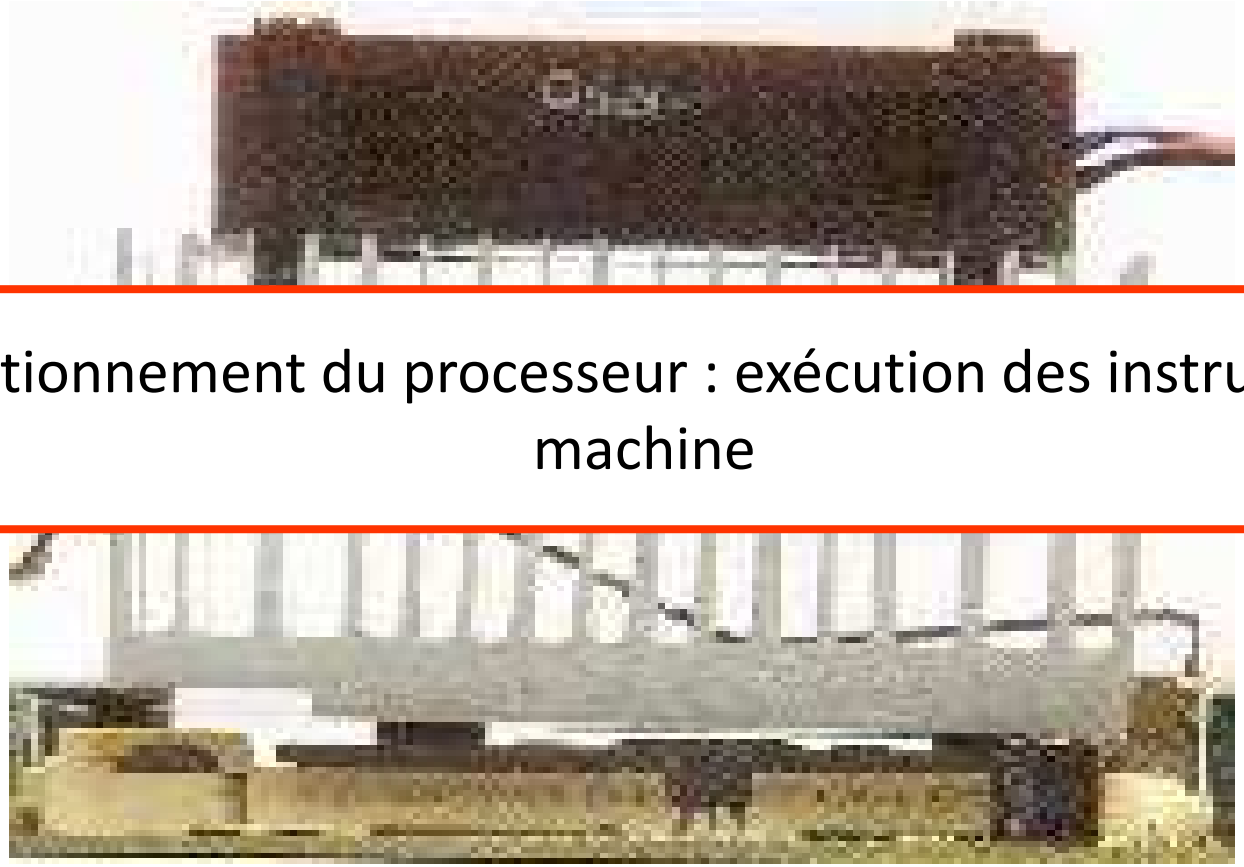
Microcommandes

RADen	Entrée dans RAD
RDOen, RDOsor	Entrée dans RDO, Sortie de RDO
RIen RIsor, Xsor	Entrée dans RI, Sortie de RI Sortie partie basse (X) de RI
COen, COsor INCO	Entrée dans CO, Sortie de CO Incrémentation du CO
Ren, Rsor	Entrée dans R, Sortie de R
RSPen,RSPsor INCRSP DECRSP	Entrée dans RSP, Sortie de RSP Incrémentation RSP Décrémentation RSP
Y1en, Y2en	Entrée Opérandes Y1, Y2 dans UAL
Zsor	Sortie du résultat Z de l'UAL
RBen, RBsor RBUALen	Entrée dans RB, Sortie de RB Entrée de RB dans UAL
RazY1	Mise à 0 de l'opérande Y1
LEC/ECR	Lecture/écriture vers MC
Add, And, Mul, Or, Xor	Opération sur UAL

Microcommandes



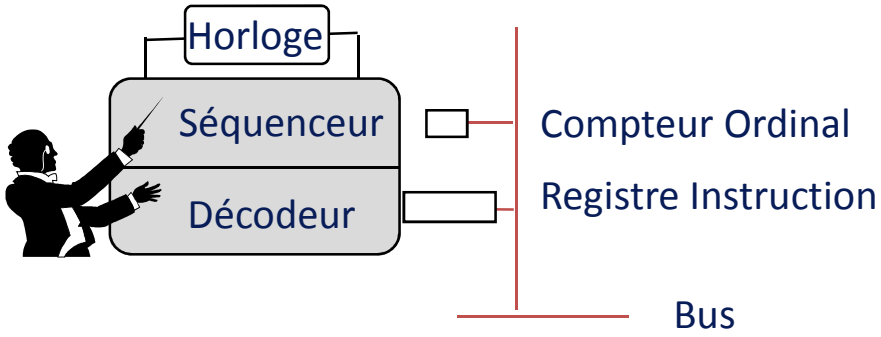
LOAD Im R 3	**Co→ RAD**	**COsor, RADen**
Fetch	**lecture**	**LEC**
	RDO→ RI	**RDOsor, RIen**
	Incrément CO	**INCO**
Exécution	**X (3) → R**	**Xsor, Ren**
ADD D R 28	**Co→ RAD**	**COsor, RADen**
Fetch	**lecture**	**LEC**
	RDO→ RI	**RDOsor, RIen**
	Incrément CO	**INCO**
Exécution	**X→ RAD**	**Xsor, RADen**
	lecture	**LEC**
	RDO → Y1	**RDOsor, Y1en**
	R → Y2	**Rsor, Y2en**
	Addition	**Add**
	Z → R	**Zsor, Ren**
STORE B R 6	**Co→ RAD**	**COsor, RADen**
Fetch	**lecture**	**LEC**
	RDO→ RI	**RDOsor, RIen**
	Incrément CO	**INCO**
Exécution	**X→ Y2**	**Xsor, Y2en**
	RB→UAL	**RBUALen**
	Addition	**Add**
	Z→RAD	**Zsor, RADen**
	R→ RDO	**Rsor, RDOen**
	Ecriture	**ECR**



Fonctionnement du processeur : exécution des instructions machine

EXECUTION D'UNE INSTRUCTION MACHINE : LE CADENCEMENT DE L'HORLOGE MICROPROGRAMMATION, SEQUENCEUR

Exécution d'une instruction machine : micro-commandes et micro-instructions



L'activation des micro-commandes composant une instruction machine s'effectue au rythme de l'horloge du processeur

➢ A chaque top horloge, le séquenceur active un sous-ensemble de micro-commandes composant une instruction

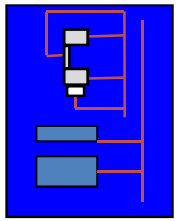
Processeur F (horloge) = 2 GHZ ; P top horloge = 0,5 ns = durée du cycle processeur

LOAD Im R 3 **Fetch**	**Co→ RAD** **lecture** **RDO→ RI** **Incrément CO**	**COsor, RAD en LEC**	TOP horloge 1
		RDOsor, RIen **INCO**	TOP horloge 2
Exécution	**X (3) → R**	**Xsor, Ren**	TOP horloge 3

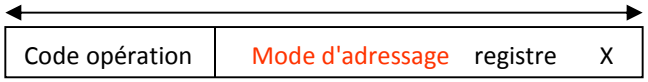
Exécution d'une instruction machine : micro-commandes et micro-instructions

Le concepteur de processeur

Définit l'architecture de son processeur



Définit le jeu d'instructions du processeur



Choisit des conventions de représentations des nombres, caractères

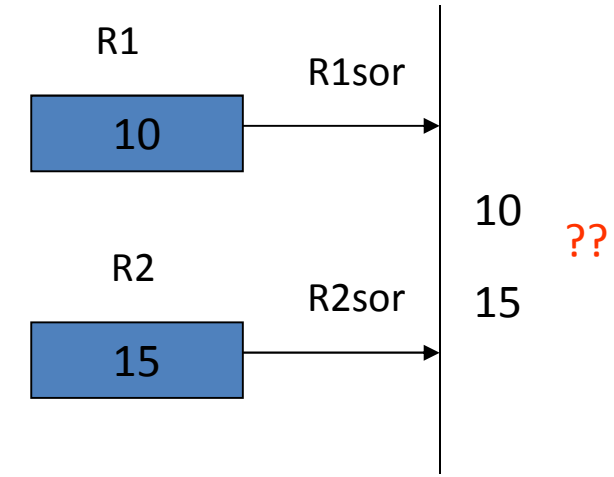
Complément à 2, ASCII, IEEE754

Construit le circuit séquenceur

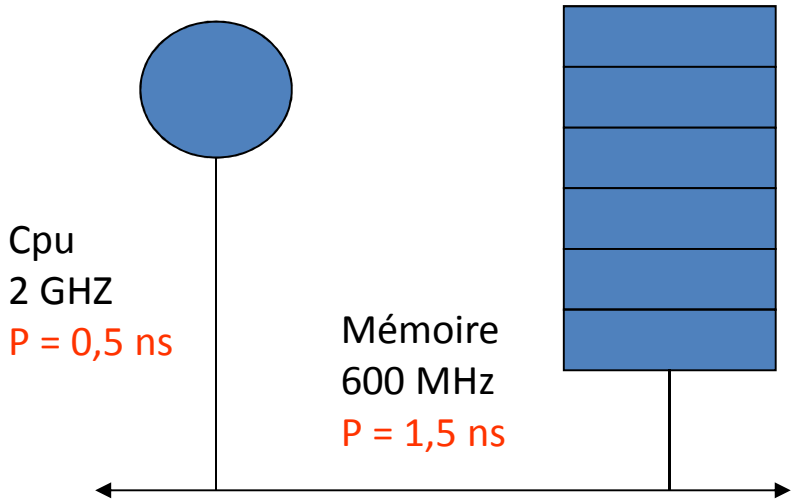
Il programme l'activation des micro-commandes en fonction des tops horloge.
Il cherche à minimiser le temps d'exécution de chaque instruction, donc le nombre de tops horloge nécessaire pour exécuter chaque instruction

Exécution d'une instruction machine : micro-commandes et micro-instructions

Le concepteur du processeur programme l'activation des micro-commandes en fonction des tops horloge.
Il cherche à minimiser le temps d'exécution de chaque instruction, donc le nombre de tops horloge nécessaire pour exécuter chaque instruction



Deux registres ne peuvent ouvrir leur porte de sortie sur un même cycle



La mémoire a un cycle 3 fois plus long que celui du processeur : le processeur attend la mémoire centrale durant deux cycles processeur

LOAD Im R 3 **Fetch** **Exécution**	**Co→ RAD** **lecture** **RDO→ RI** **Incrément CO** **X (3) → R**	**µI1 : COsor, RADen, LEC** **µI2 : RDOsor, RIen, INCO** **µI3 : Xsor, Ren**	**1 cycle** **Attente mémoire (2 cycles)** **1 cycle** **1 cycle** **5 cycles**
ADD D R 28 **Fetch** **Exécution**	**Co→ RAD** **lecture** **RDO→ RI** **Incrément CO** **X→ RAD** **lecture** **RDO → Y1** **R → Y2** **Addition** **Z → R**	**µI1 : COsor, RADen, LEC** **µI2 : RDOsor, RIen, INCO** **µI3 : Xsor, RADen, LEC** **µI4 : RDOsor, Y1en** **µI5 : Rsor, Y2en, Add** **µI6 : Zsor, Ren**	**1 cycle** **Attente mémoire (2 cycles)** **1 cycle** **1 cycle** **Attente mémoire (2 cycles)** **1 cycle** **1 cycle** **1 cycle** **10 cycles**
STORE B R 6 **Fetch** **Exécution**	**Co→ RAD** **lecture** **RDO→ RI** **Incrément CO** **X→ Y2** **RB→UAL** **Addition** **Z→RAD** **R→ RDO** **Ecriture**	**µI1 : COsor, RADen, LEC** **µI2 : RDOsor, RIen, INCO** **µI3 : Xsor, Y2en, RBUALen, Add** **µI4 : Zsor, RADen** **µI5 : Rsor, RDOen, ECR**	**1 cycle** **Attente mémoire (2 cycles)** **1 cycle** **1 cycle** **1 cycle** **1 cycle** **Attente mémoire (2 cycles)** **9 cycles**

Exécution d'une instruction machine : micro-commandes et micro-instructions
Niveaux de programmation

```
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

Machine physique "matérielle"

processeur

COsor, RADen, LEC
RDOsor, RIen
R1sor, Y1en
Xsor, Y2en, Mul
Zsor, R1en

Séquenceur : microprogrammation

Bus

```
MUL Im R1 2
MUL Im R2 2
ADD Rg2 R1 R2
```

Programme en langage d'assemblage

Assembleur

```
011011100000….10010
001110111101….11011
001111110001….11101
```

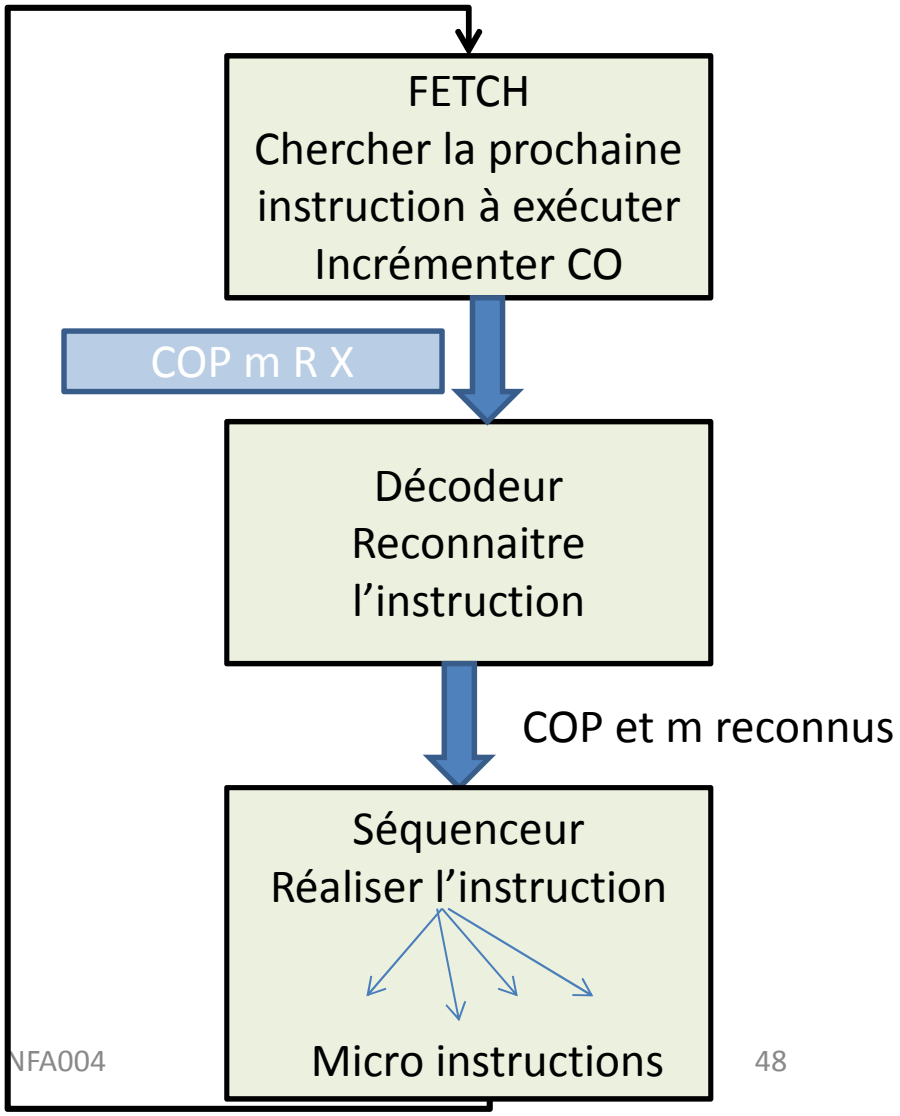
Programme en langage machine

Mémoire centrale

traduction

Exécution d'une instruction machine : micro-commandes et micro-instructions
Niveaux de programmation
Le séquenceur

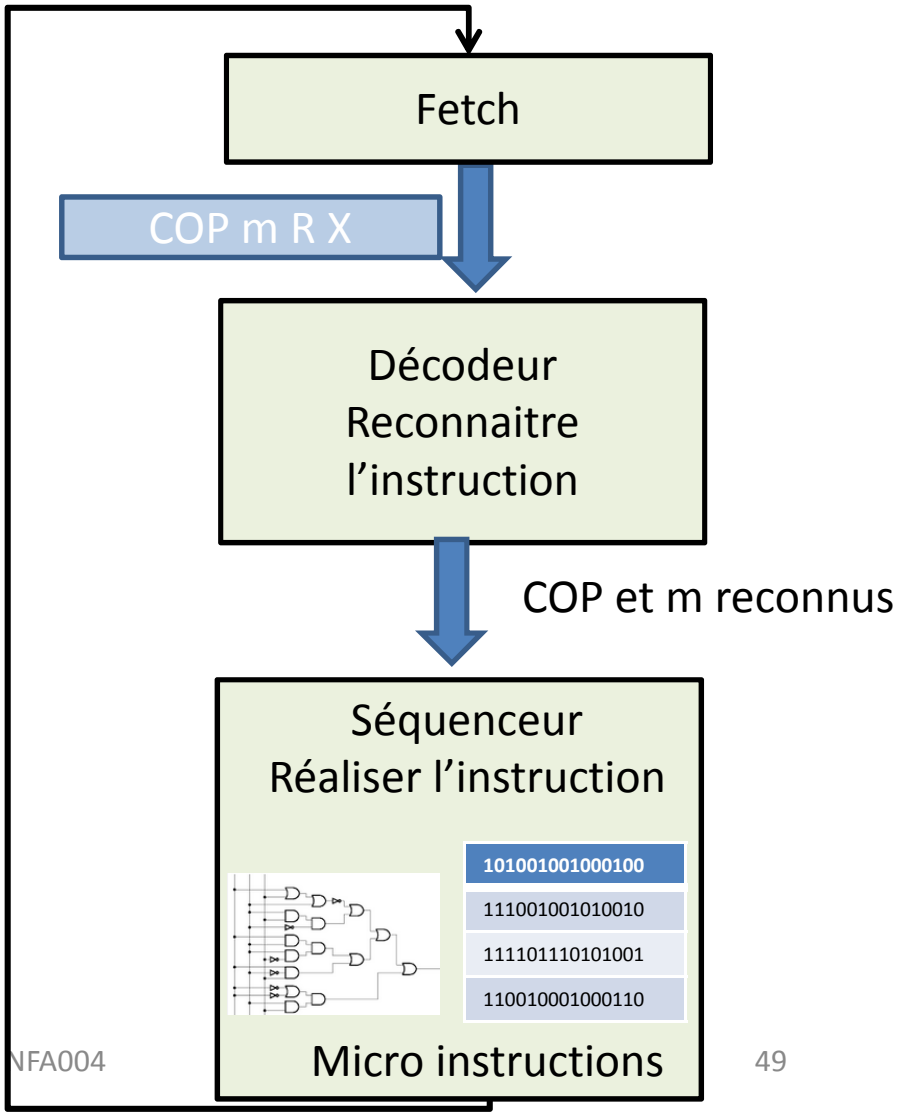
- Le séquenceur est un automate :
  - Il reçoit des informations du décodeur
  - Il produit les signaux de commandes contrôlant les différentes unités
- Séquenceur cablé :
- Séquenceur microprogrammé



Exécution d'une instruction machine : micro-commandes et micro-instructions
Niveaux de programmation
Le séquenceur

- Séquenceur cablé:
  - Un circuit composé de portes logiques
  - Un sous-circuit par instruction activé selon le code envoyé par le décodeur

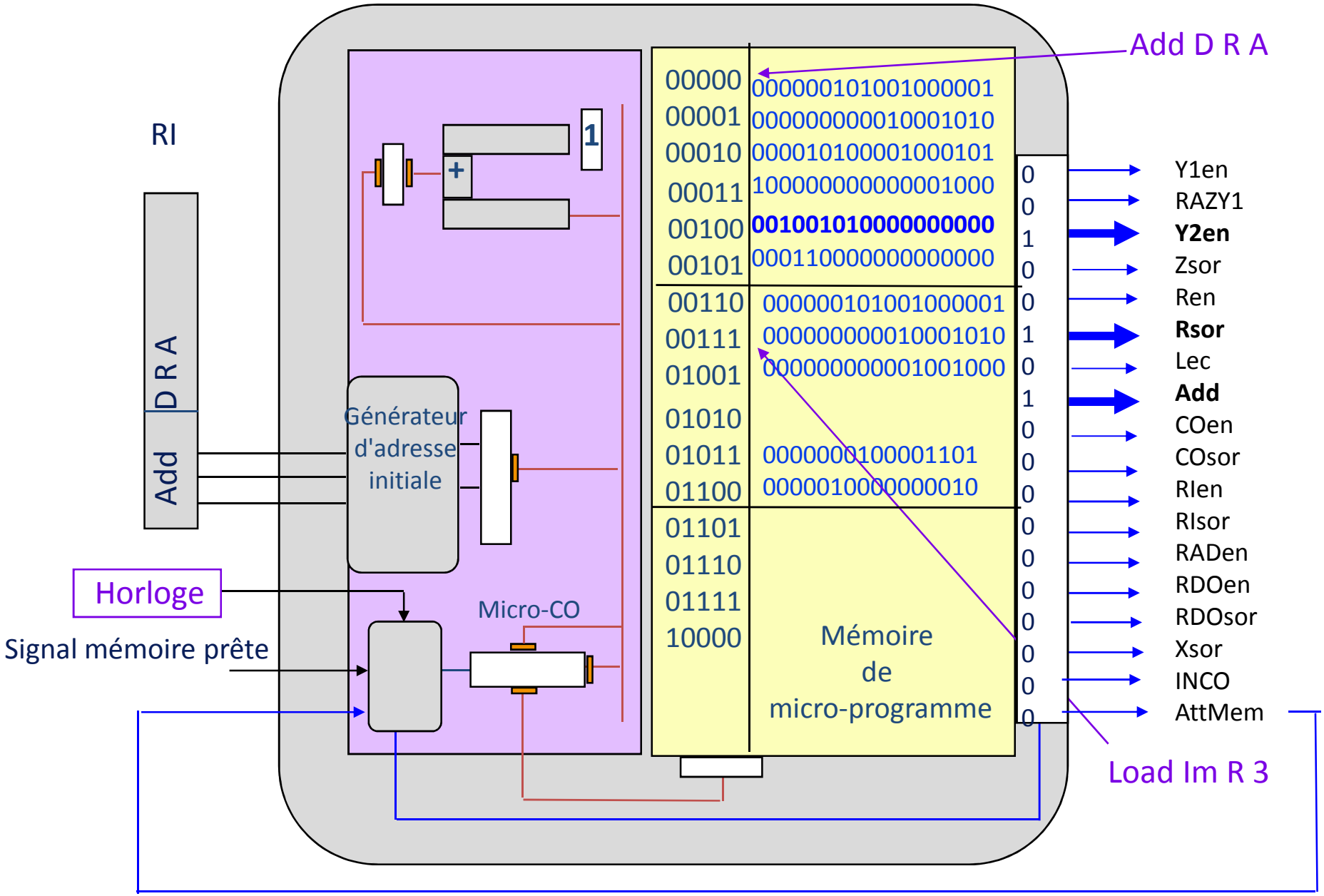
- Séquenceur cablé:
  - Une mémoire non volatile (ROM) contient des micro-programmes composés de micro-instructions
  - Un sous-programme par instruction activé selon le code envoyé par le décodeur



Exécution d'une instruction machine : micro-commandes et micro-instructions
Niveaux de programmation
Le séquenceur micro programmé

- Il est constitué :
  - d'une mémoire qui contient les micro-programmes de chaque instruction machine;
  - d'un micro-compteur ordinal qui pointe sur la micro-instruction à exécuter.

- Une micro-instruction (horizontale):
  - est une chaine binaire dont la taille est égale au nombre de micro-commandes disponibles;
  - Un bit à 0 indique que la micro-commande ne doit pas être activée; un bit à 1 indique que la micro-commande doit être activée au cours du cycle processeur.



Séquenceur micro-programmé