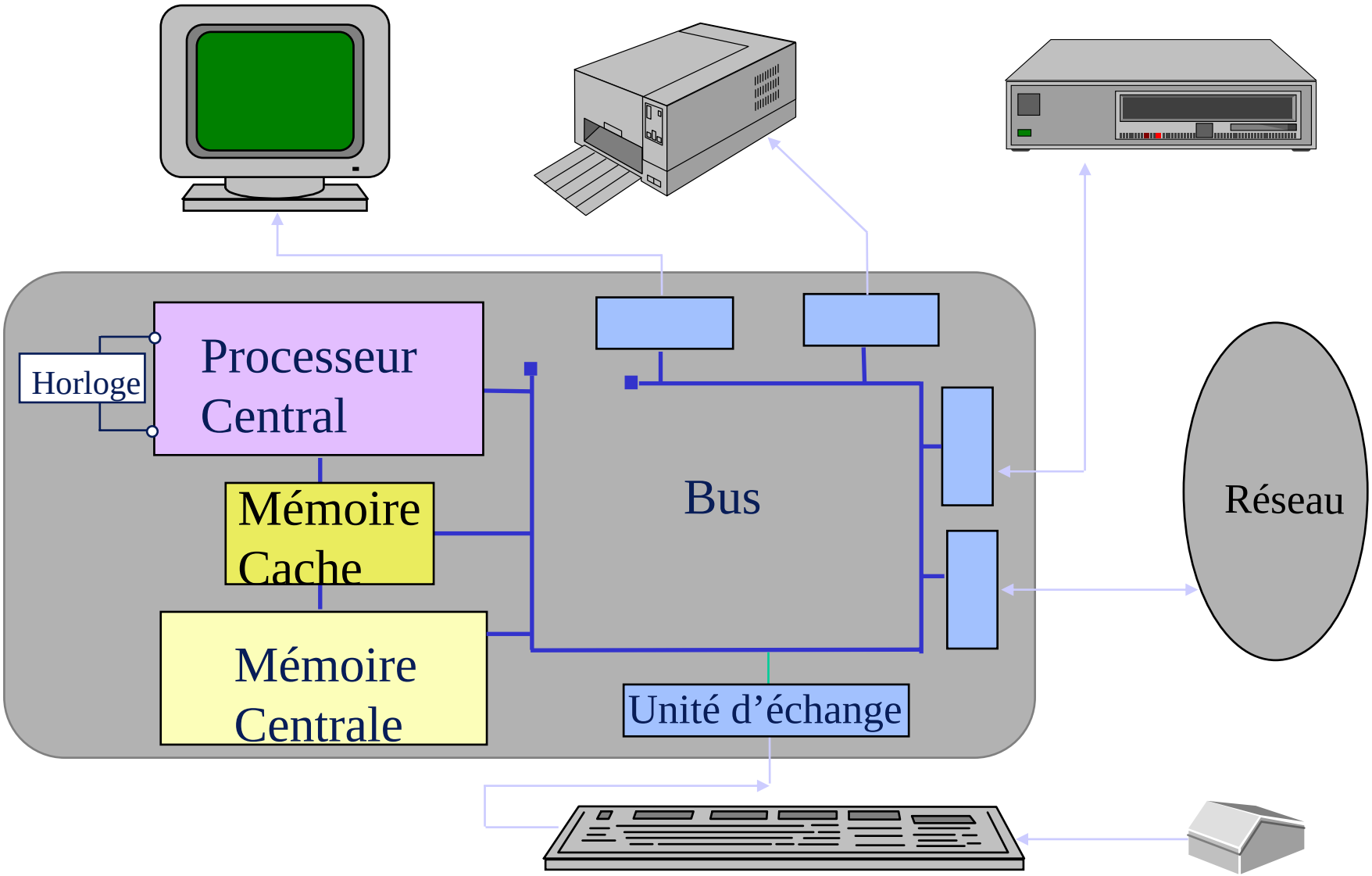
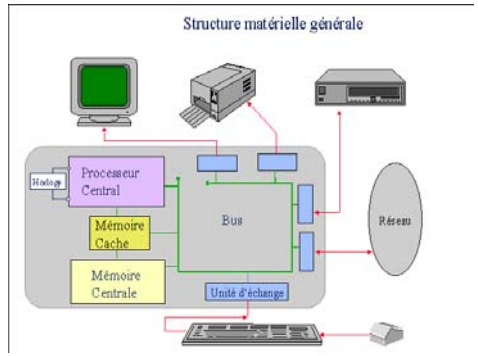
REVISION

Structure générale de la machine physique



Structure générale de la machine physique



• Le processeur est chargé d'exécuter les instructions des programmes placés en mémoire centrale. Le processeur est cadencé par une horloge.

• La mémoire centrale contient les instructions et données des programmes à exécuter. Avec le disque, la mémoire cache, elle constitue un système de hiérarchie de mémoire qui permet de rapprocher la vitesse de la mémoire centrale de celle du processeur.

• Les unités d'échanges (UE) réalisent l'interface entre les périphériques et le processeur.

• Tous les composants de la machine communiquent par l'intermédiaire du bus.

Codage d’un problème

Le codage d'un problème ...

Programme en langage de haut niveau
instructions de haut niveau

Compilateur

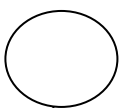
Niveau utilisateur

Machine physique "matérielle"

Programme en langage d'assemblage

Assembleur

processeur



Bus

01101110111110011
01111010001011100
10111101110111111
00111011110111011
00111111000111101

Mémoire centrale

Programme à
exécuter : instructions machine
et valeurs en binaire

traduction

Codage d'un problème

```
Programme exemple
Entier A = 124;
Entier B;
Entier C;
Début
A := A - 12;
Si (A >= 0)
Alors
        B := A;
Sinon
        C : = A;
Finsi
Fin
```

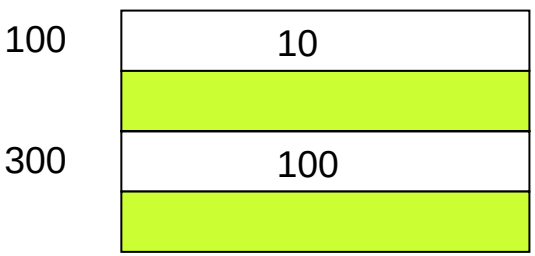

```
A :         DS 1 = 124
B :         DS 1
C :         DS 1
            LOAD D R1 A
            ADD Im R1 -12
            JMPN négatif
            STORE D R1 B
            JMP Fin
Négatif :   STORE D R1 C
Fin         : STOP
```

Langage
Haut Niveau

Langage
Assemblage

Les modes d'adressages : mnémoniques dans le langage d'assemblage

Immédiat Im	le deuxième opérande est la valeur X	LOAD Im R1 300	R1← 300
Direct D Indirect I	le deuxième opérande est un opérande mémoire. X est une adresse	LOAD D R1 300 LOAD I R1 300	R1← (300) = 100 R1←((300)) = 10
Basé B	le deuxième opérande est un opérande mémoire. X est un déplacement à ajouter à la valeur contenue dans un registre du processeur (RB)	RB = 100 LOAD B R1 200	R1 ← (100+200) = 100
Registre/Registre Rg2	Le deuxième opérande est un registre banalisé du processeur	R1 = 10 R2 = 20 ADD Rg2 R1 R2	R1← R1 + R2 = 30
Registre Rg1	Il n'y a pas de deuxième opérande	NEG Rg1 R2	R2 ← - 20



Mémoire centrale

Codage d'un problème

A :	DS 1 = 124	réservation d'un mot initialisé à 124
B :	DS 1	réservation d'un mot
C :	DS 1	réservation d'un mot
	LOAD D R1 A	R1 ← (A) = 124
	ADD Im R1 -12	R1 ← R1 + (-12)
	JMPN négatif	si R1 négatif sauter à négatif sinon continuer (R1 > = 0)
	STORE D R1 B	R1 écrit dans le mot d'adresse B
	JMP Fin	Sauter à fin
Négatif :	STORE D R1 C	(R1 < 0) R1 écrit dans le mot d'adresse C
Fin	: STOP	

Langage
Assemblage

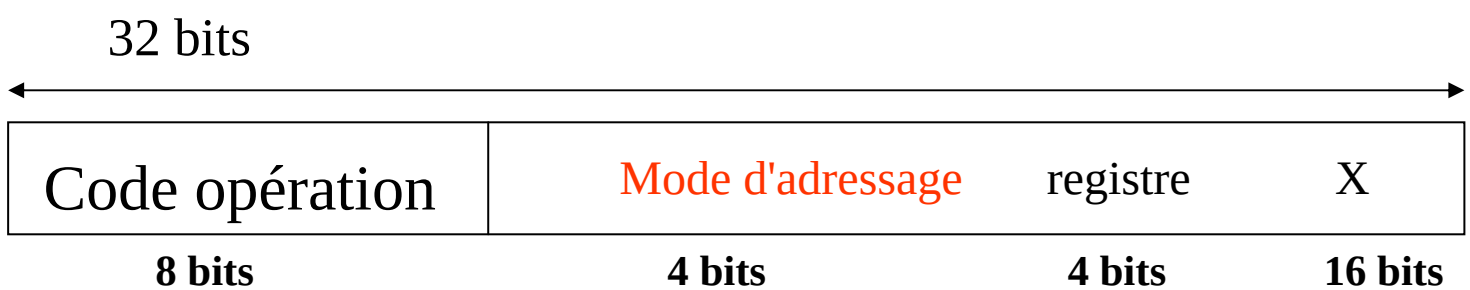
Codage d'un problème

```
A :          DS 1 = 124
B :          DS 1
C :          DS 1
             LOAD D R1 A
             ADD Im R1 -12
             JMPN négatif
             STORE D R1 B
             JMP Fin
Négatif :    STORE D R1 C
Fin          : STOP
```

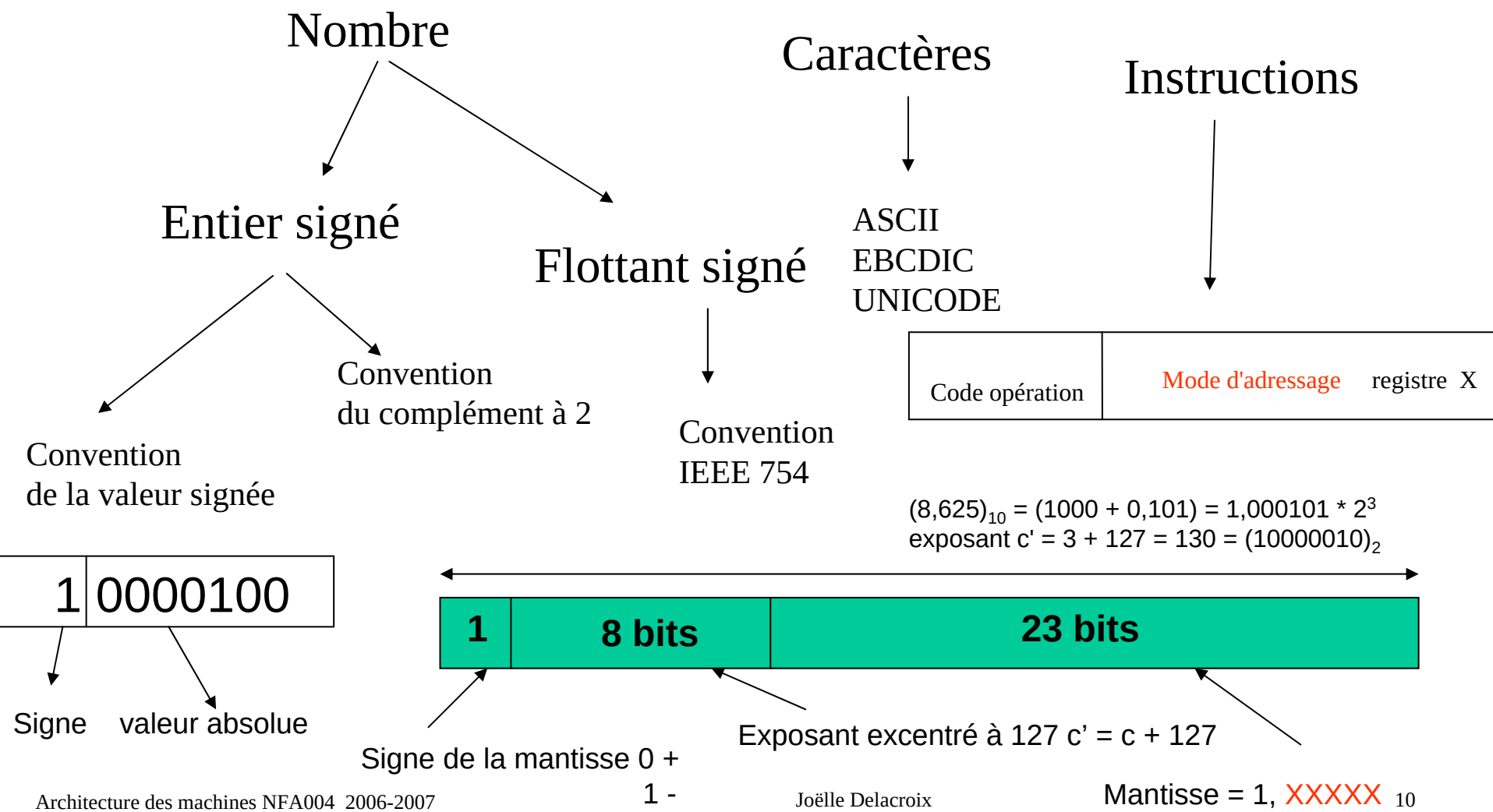
Programme en langage machine

Les entiers sont codés en complément à 2 sur 16 bits
Les adresses sont sur 8 bits

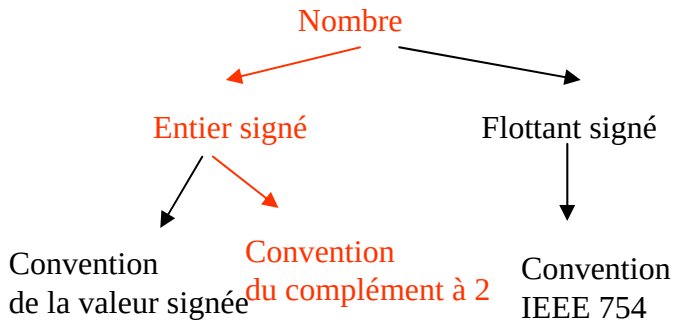
Le format de mes instructions



La représentation des informations sur la machine physique



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 16 bits en complément à 2

$124 = 64 + 32 + 16 + 8 + 4 = 2^6 + 2^5 + 2^4 + 2^3 + 2^2 = 0000000001111100$

Exemple : représenter - 12 sur 16 bits en complément à 2

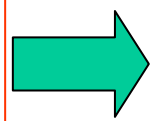
$12 = 8 + 4 = + 2^3 + 2^2 = 0000000000001100$

```
  0000000000001100
  1111111111110011   inversion des bits
+                1   ajout de 1
-------------------
  1111111111110100 = - 12
```

Codage d'un problème

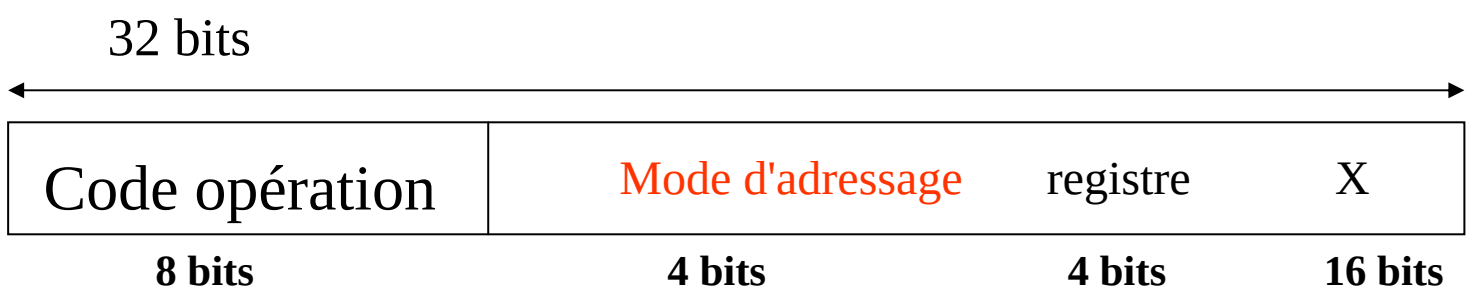
Langage d'assemblage

```
A :          DS 1 = 124
B :          DS 1
C :          DS 1
             LOAD D R1 A
             ADD Im R1 -12
             JMPN négatif
             STORE D R1 B
             JMP Fin
Négatif :    STORE D R1 C
Fin          : STOP
```



Langage machine

```
00  0000000000000000 0000000001111100
04
08
0C  00000000001000100000000000000000
10   00000100000000011111111111110100
14   00001110000000000000000000100000
18   00000001000100010000000000000100
1C  00001011000000000000000000100100
20   00000001000100010000000000001000
24
```



Exécution du programme machine

Exécution du programme machine

Environnement matériel

Processeur et mémoires

vitesse 6-35 ns 70 - 120 ns

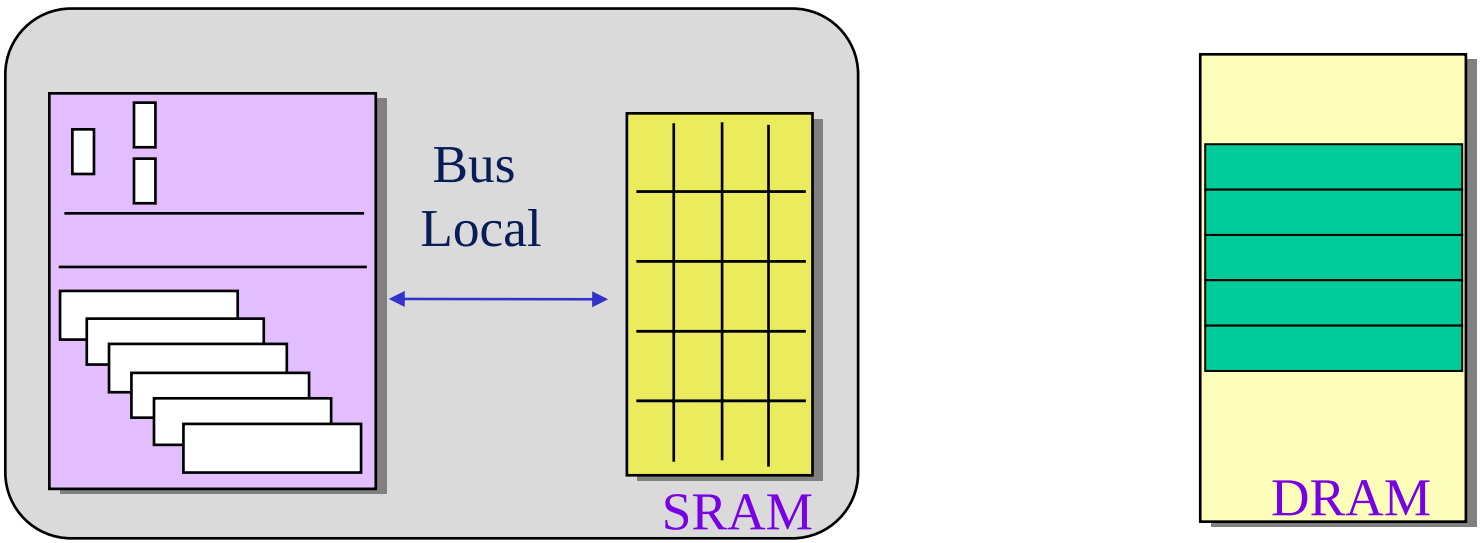
cout Le plus élevé Le moins élevé

capacité La plus petite La plus grande

1. Processeur
Registres

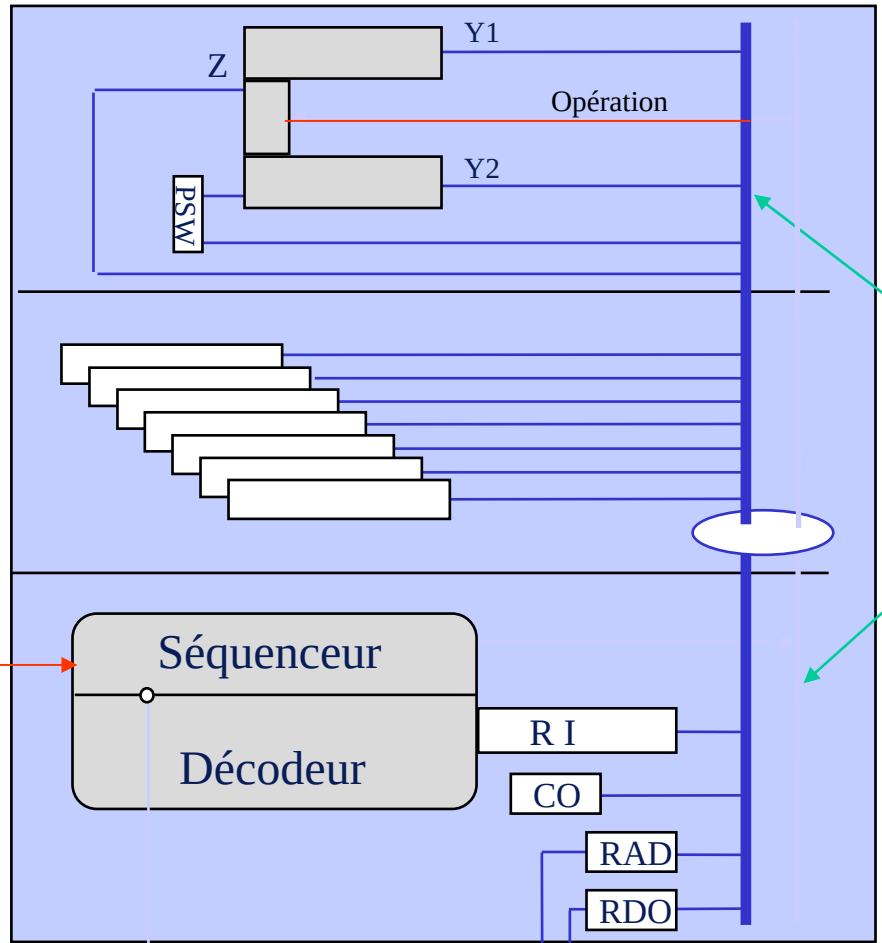
4. Mémoire
Cache

2. Mémoire
centrale



3. Bus système

1. Processeur (Unité Centrale)



Unité Arithmétique et Logique

Registres

Unité de Commande

Bus Interne

Données

Commandes

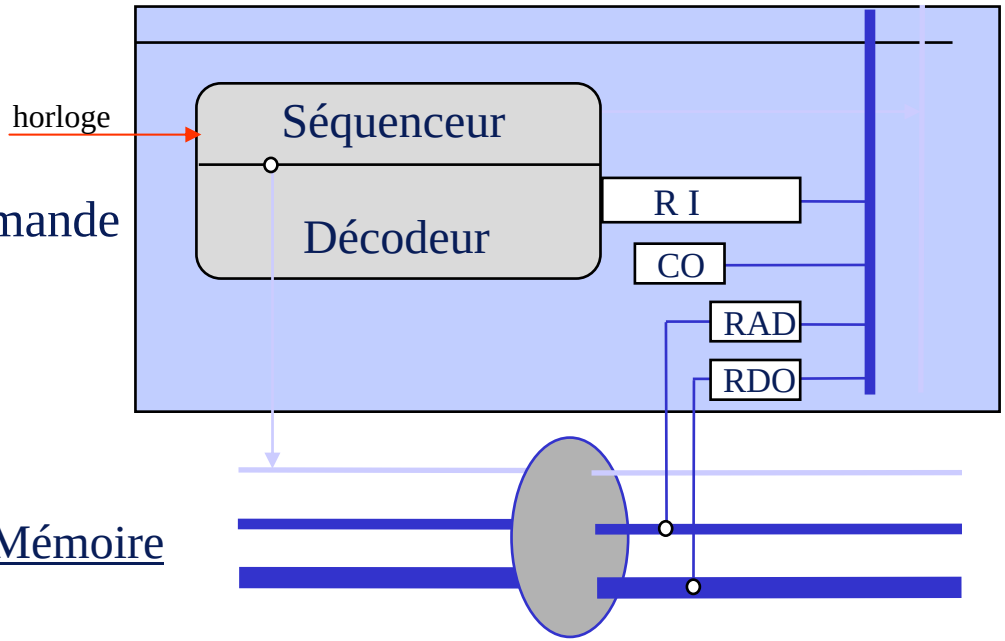
Bus Processeur /Mémoire

Commandes Lecture/Ecriture

Adresses

Données

1. Processeur (Unité Centrale)

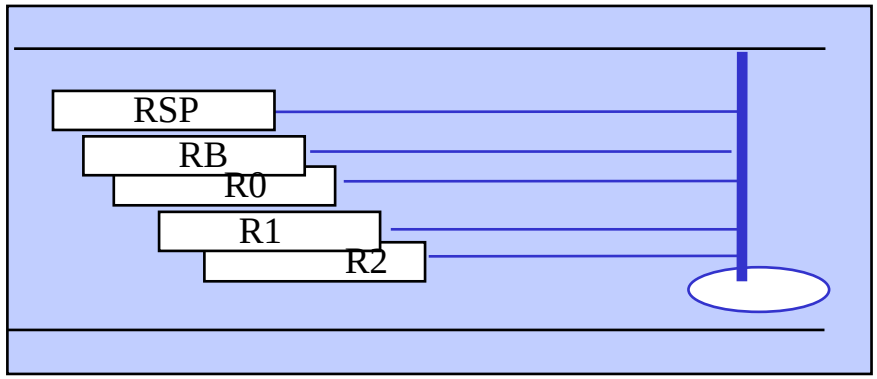


L'unité de commande est chargée de la reconnaissance des instructions et de leur exécution par l'unité de traitement au rythme de l'horloge

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

1. Processeur (Unité Centrale)



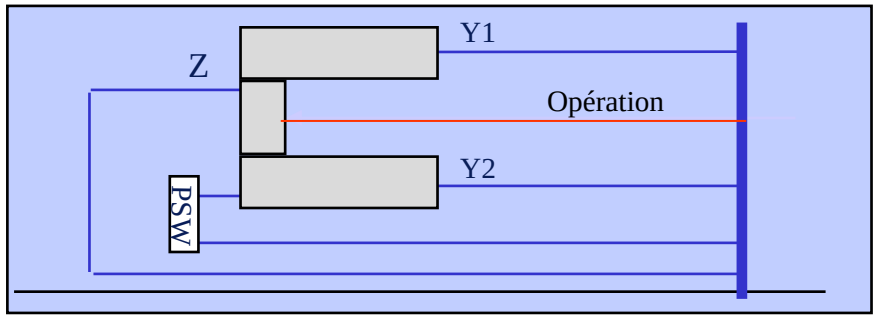
- Le registre est l'entité de base manipulée par le processeur.
- Aucune opération n'est directement réalisée sur les cellules mémoires.

Registres :

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

1. Processeur (Unité Centrale)

Unité Arithmétique et Logique



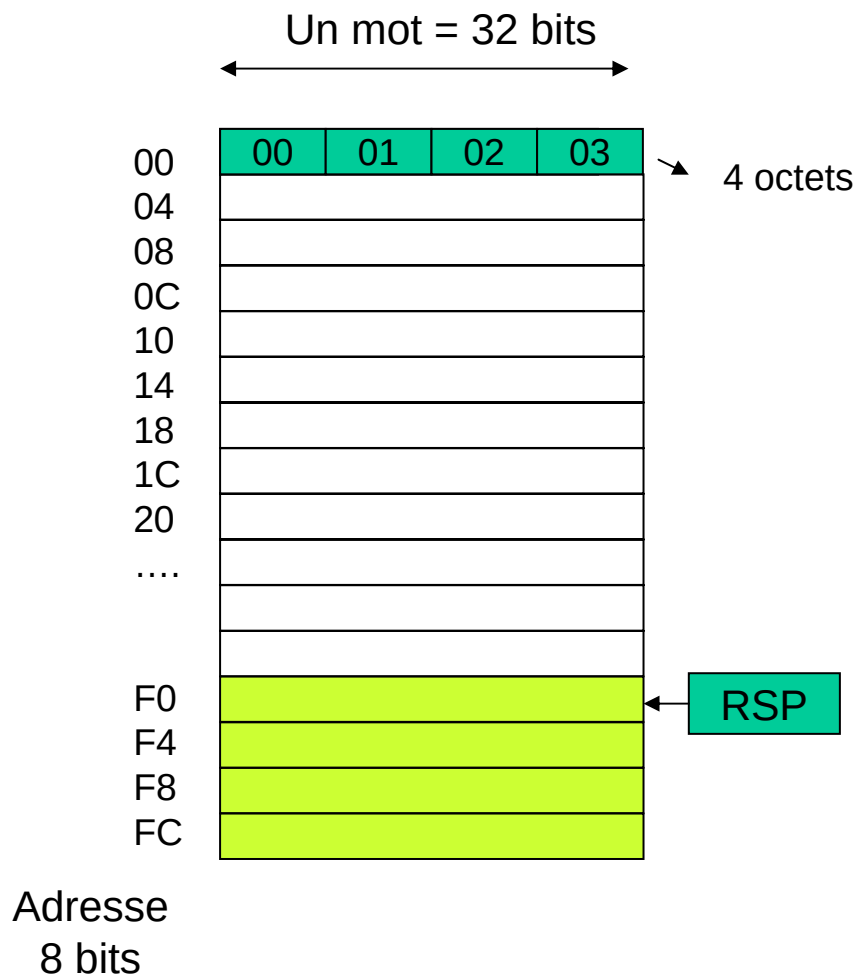
L'unité Arithmétique et Logique (UAL) constitue l'unité d'exécution du processeur.
Elle est composée :

- 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

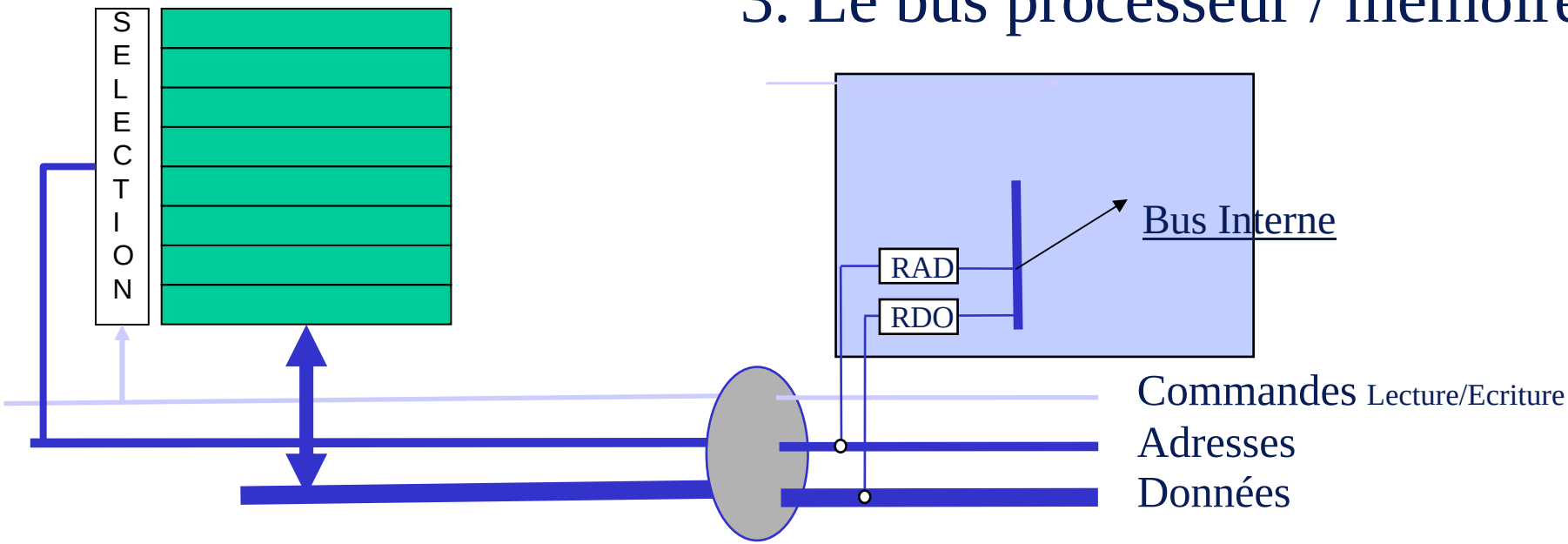
2. La mémoire centrale



Capacité mémoire : 2^8 octets
2^6 mots

- La mémoire centrale est constituée par un ensemble de mots mémoire (32 bits).
- Un mot est constitué par un ensemble d'octets; chaque octet est repéré de manière unique par une adresse.
- Elle s'interface avec le cpu via le bus mémoire-cpu.
- Une zone particulière de la mémoire est gérée comme une structure de pile repérée par le registre RSP du processeur.

3. Le bus processeur / mémoire



Bus Processeur /Mémoire

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.

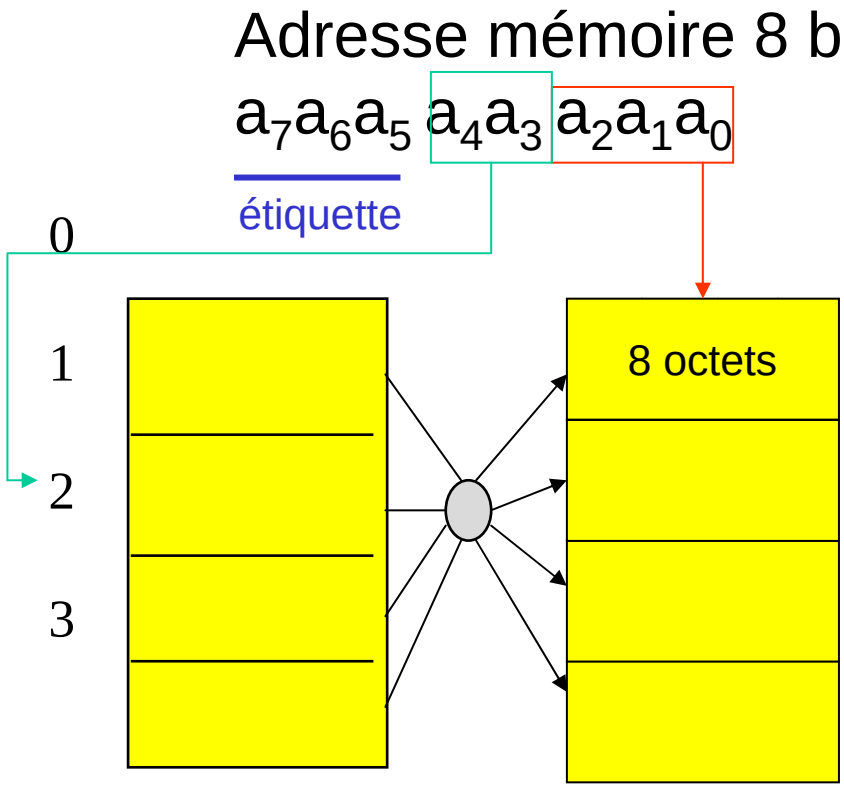
Le bus est composé :

- 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

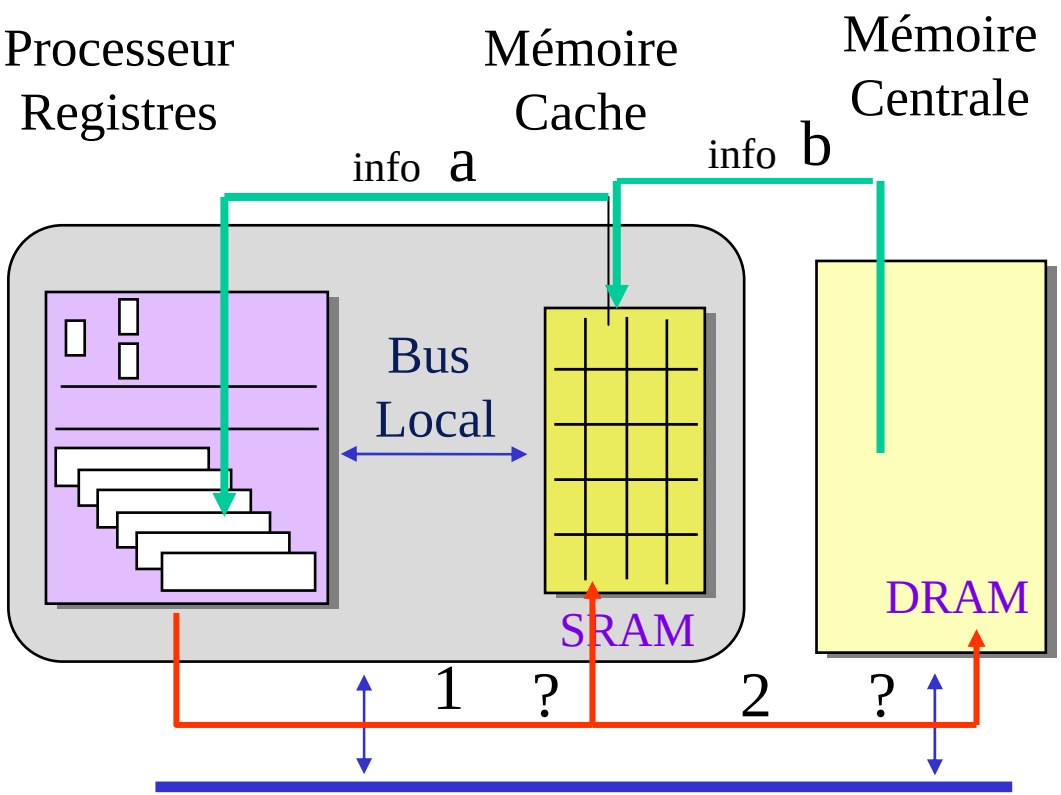
On appelle largeur du bus le nombre de bits que le bus peut transporter en parallèle.

4. Mémoire cache

Le processeur intègre un cache à correspondance directe à 4 lignes de 2 mots mémoires chacun.
La politique d'écriture du cache est une politique immédiate



Mémoire cache : principe



1. L'info cherchée est-elle dans le cache ?
OUI / Succès (a) : ramener l'info dans le processeur
NON / Défaut (2) : chercher l'info dans la mémoire centrale

2. L'info est-elle en mémoire centrale ?
OUI / Succès (b) : ramener l'info dans le cache , puis dans le processeur (a)
NON / Défaut

CONTEXTE

- 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

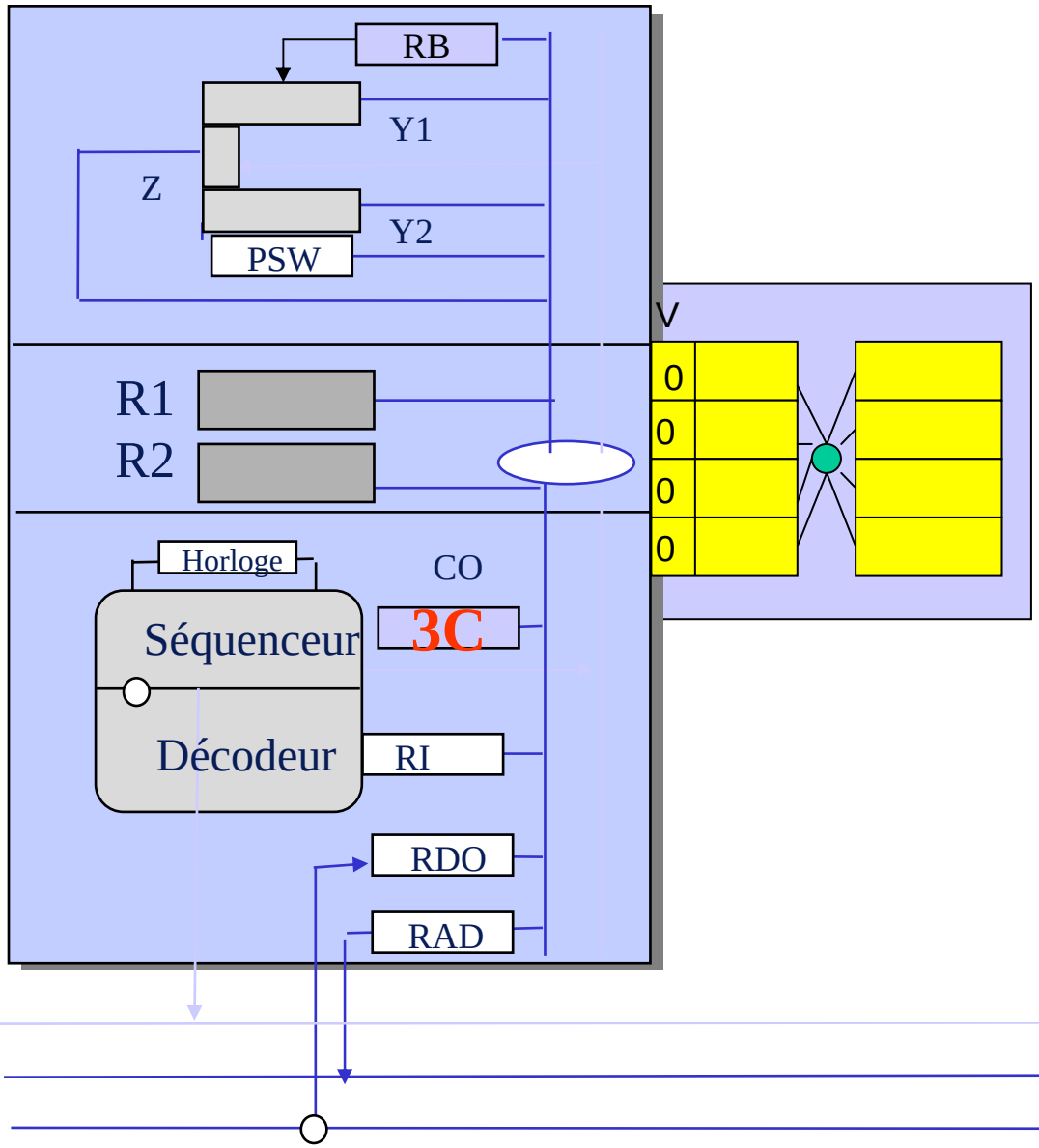
- L'exécution du programme s'effectue instruction par instruction, sous le pilotage de l'unité de commande du processeur.

- 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 du programme machine

Un exemple d’exécution

CONTEXTE



30	A : 124
34	B :
38	C :
3C	LOAD D R1 A
40	ADD Im R1 -12
44	JMPN négatif
48	STORE D R1 B
4C	JMP Fin
50	Négatif : STORE D R1 C
54	Fin : STOP

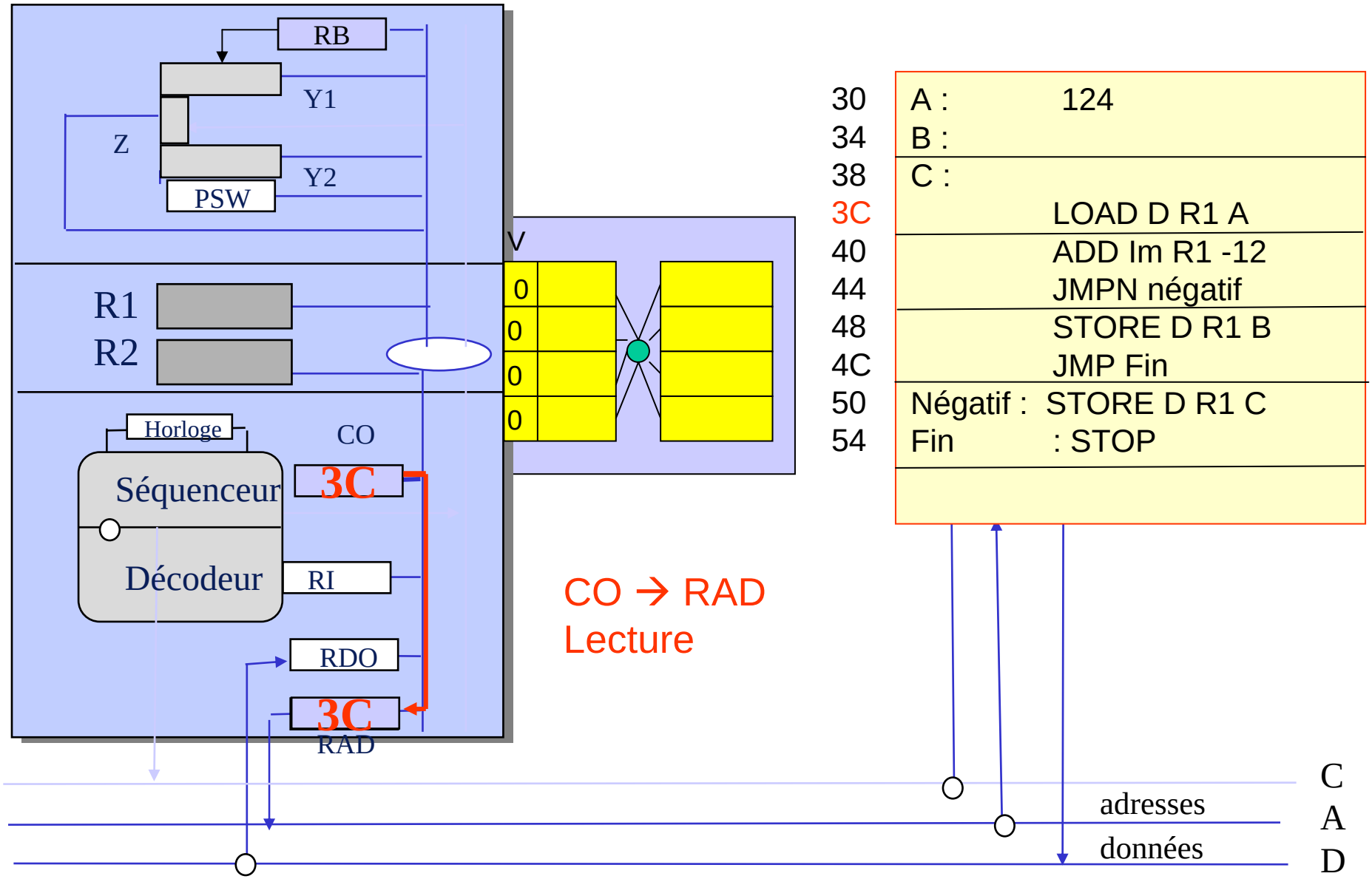
Les étapes d'exécution d'une instruction (Chargement/Décodage/Exécution)

Début

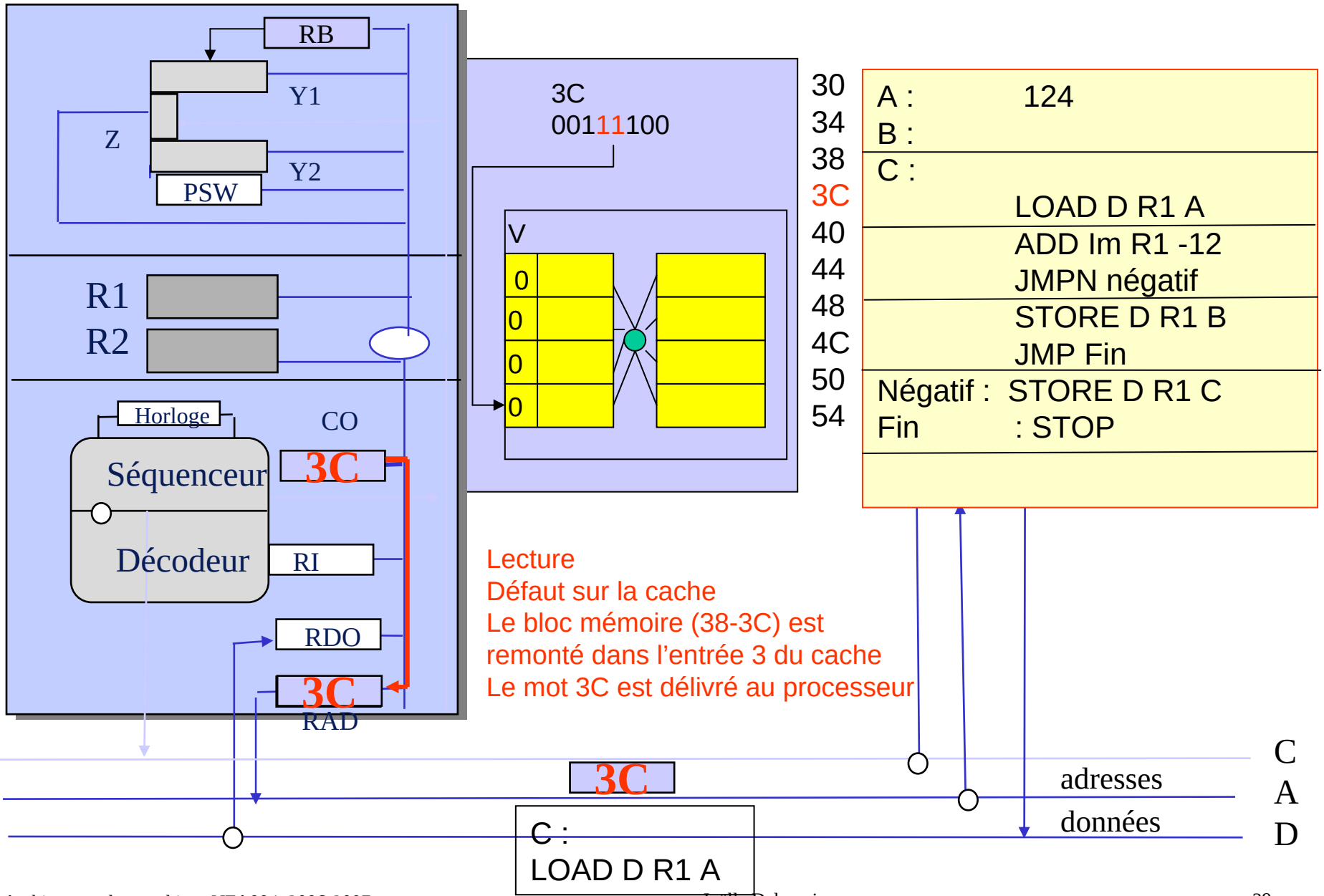
Charger	la prochaine instruction à exécuter depuis la mémoire dans le registre instruction (RI),	Fetch
Modifier	le Compteur Ordinal pour qu'il pointe sur la prochaine instruction à exécuter,	
Décoder	l'instruction qui vient d'être chargée,	Décodage
Charger	les données éventuelles dans les registres internes,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

Fin

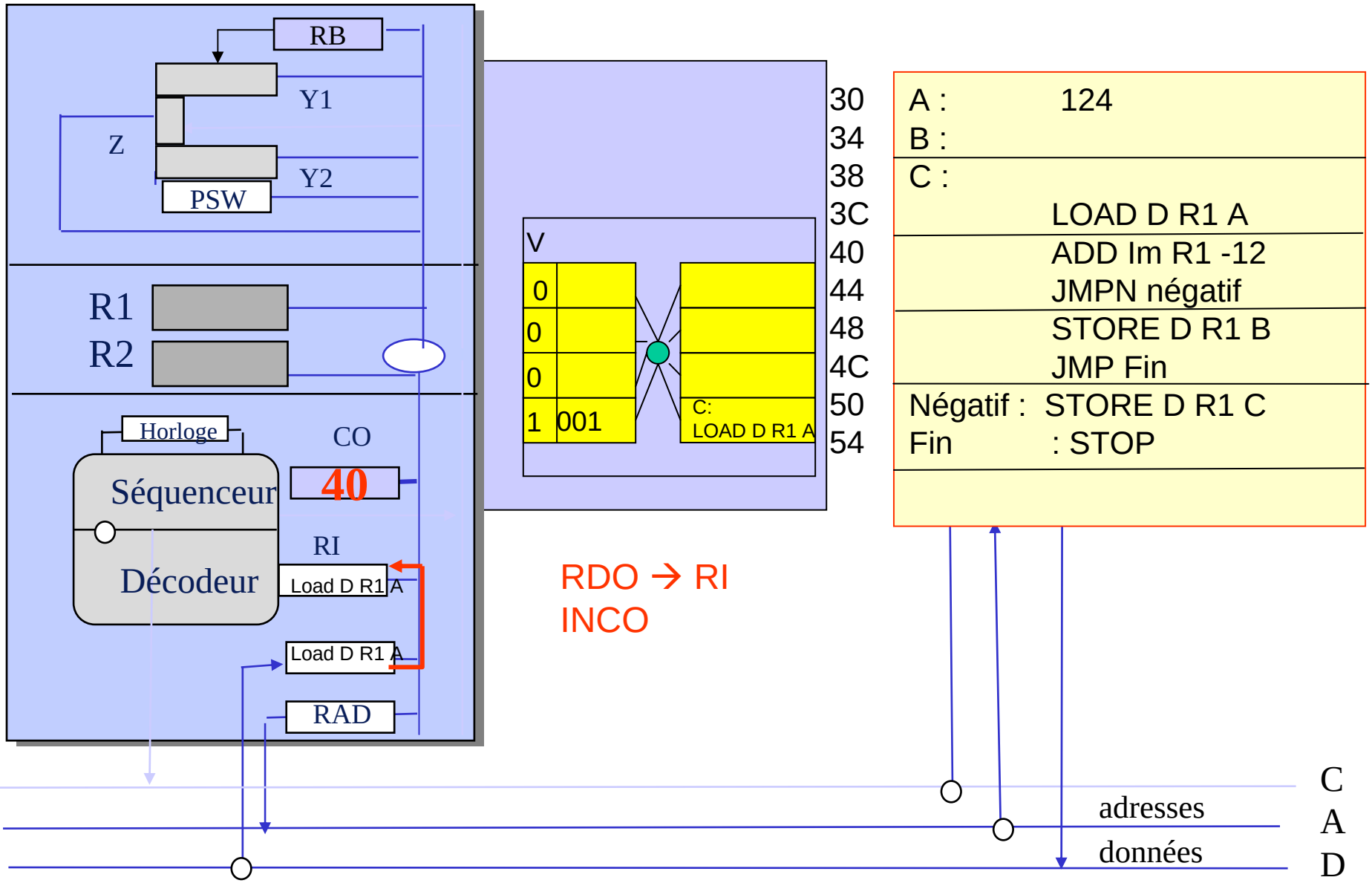
Exécution des instructions



Exécution des instructions



Exécution des instructions



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

Début

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

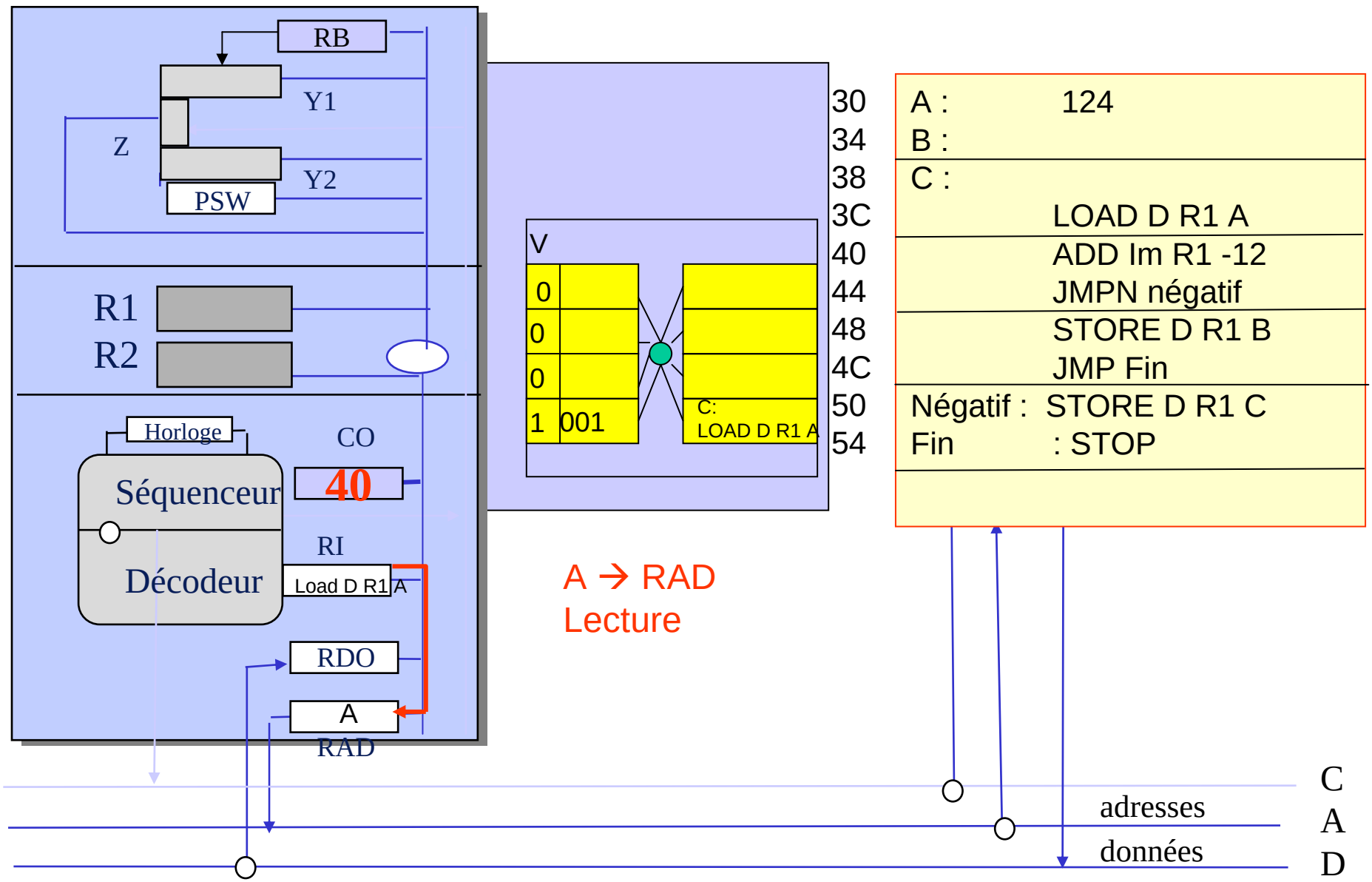
Charger les données éventuelles dans les registres internes,

Exécuter l'instruction, — Exécution

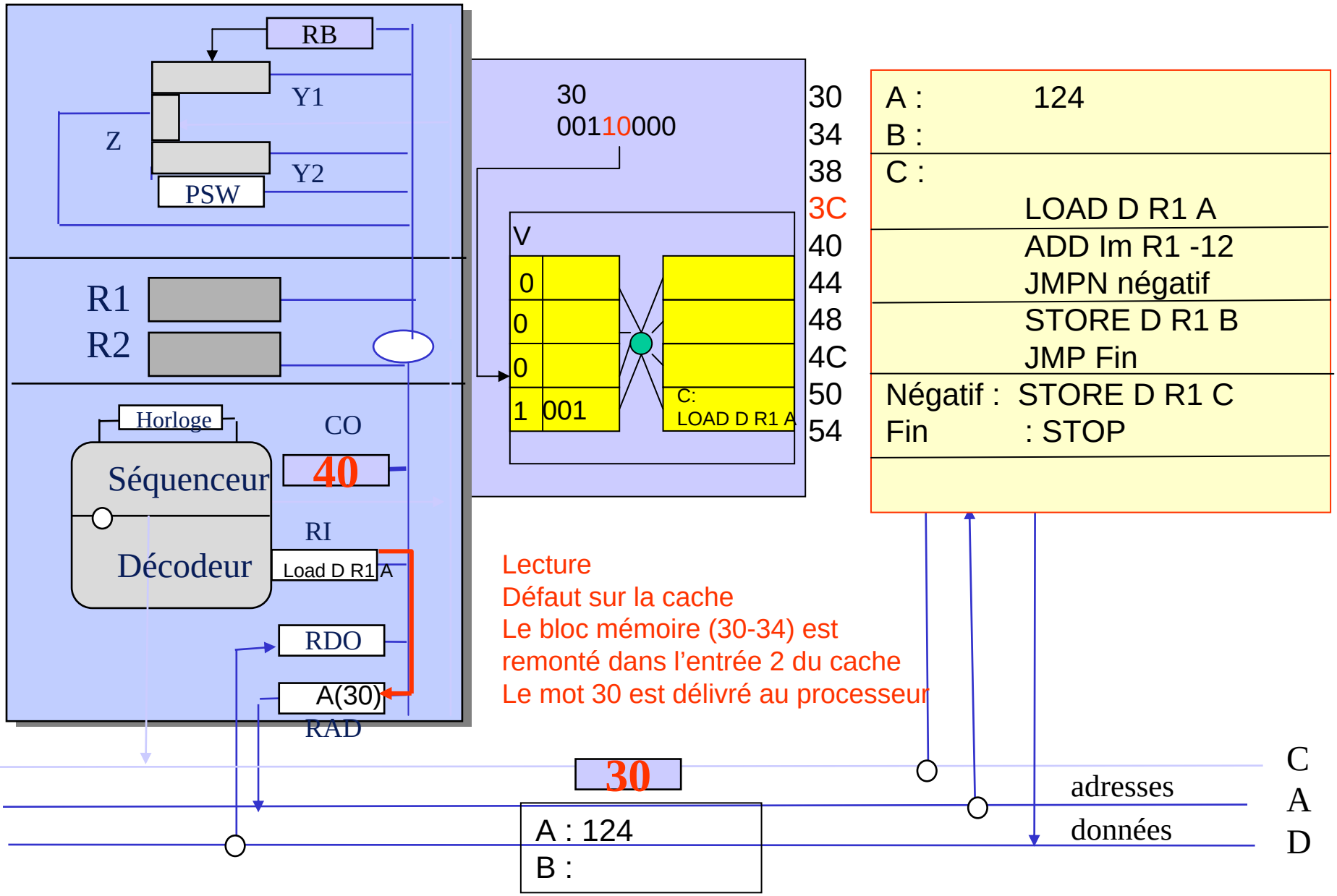
Fin

LOAD D R1 A : l'adressage est un mode direct. Il faut lire l'opérande d'adresse A en mémoire centrale

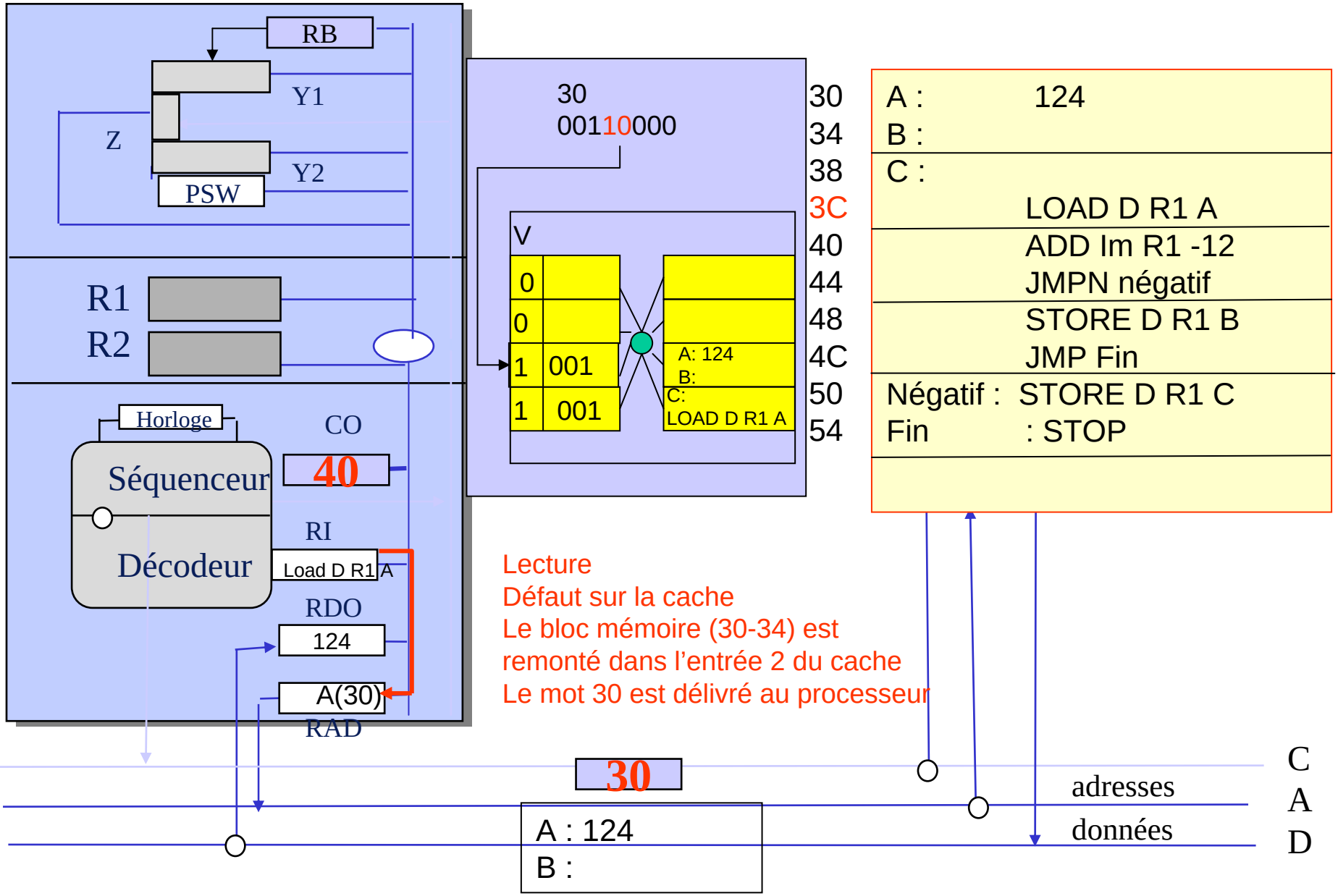
Exécution des instructions



Exécution des instructions



Exécution des instructions



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

Début

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

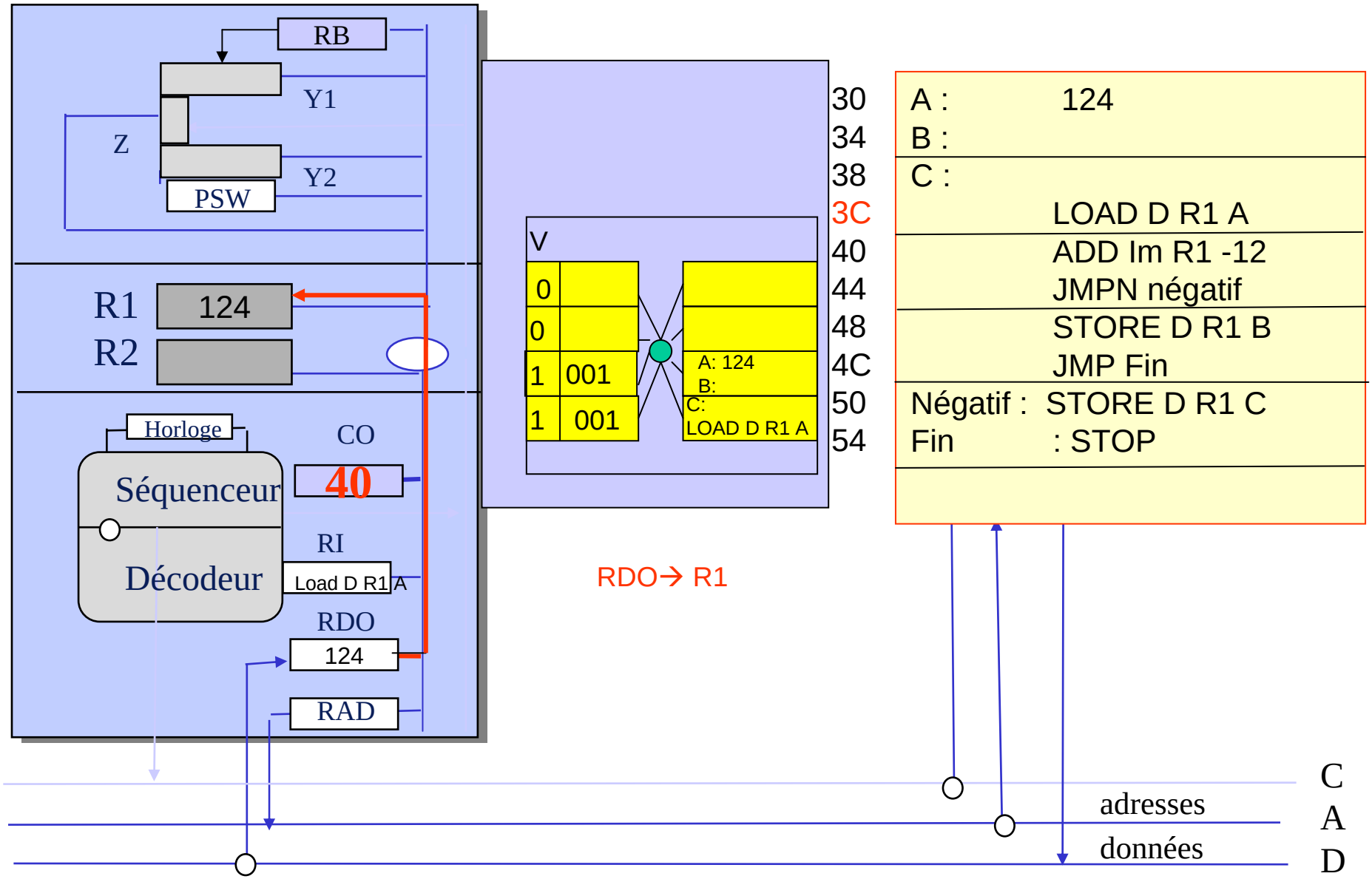
Charger les données éventuelles dans les registres internes,
Exécuter l'instruction, — Exécution

Fin

LOAD D R1 A : l'opération est un chargement de registre.
La valeur 124 est placée dans le registre R1.

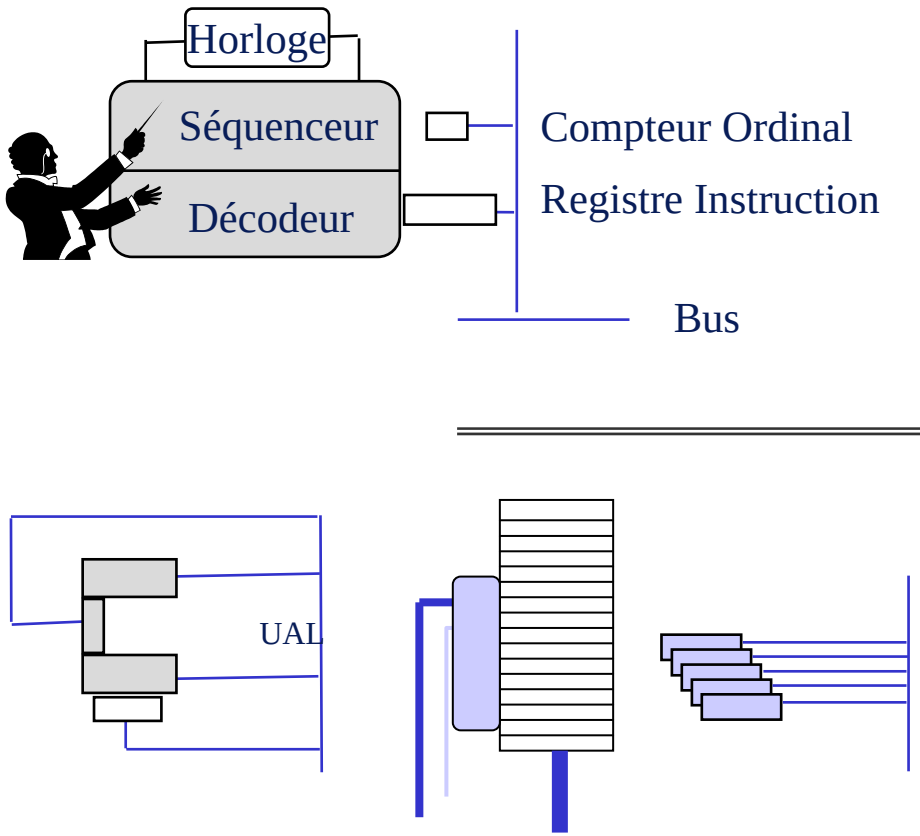
R1 ← 124

Exécution des instructions



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



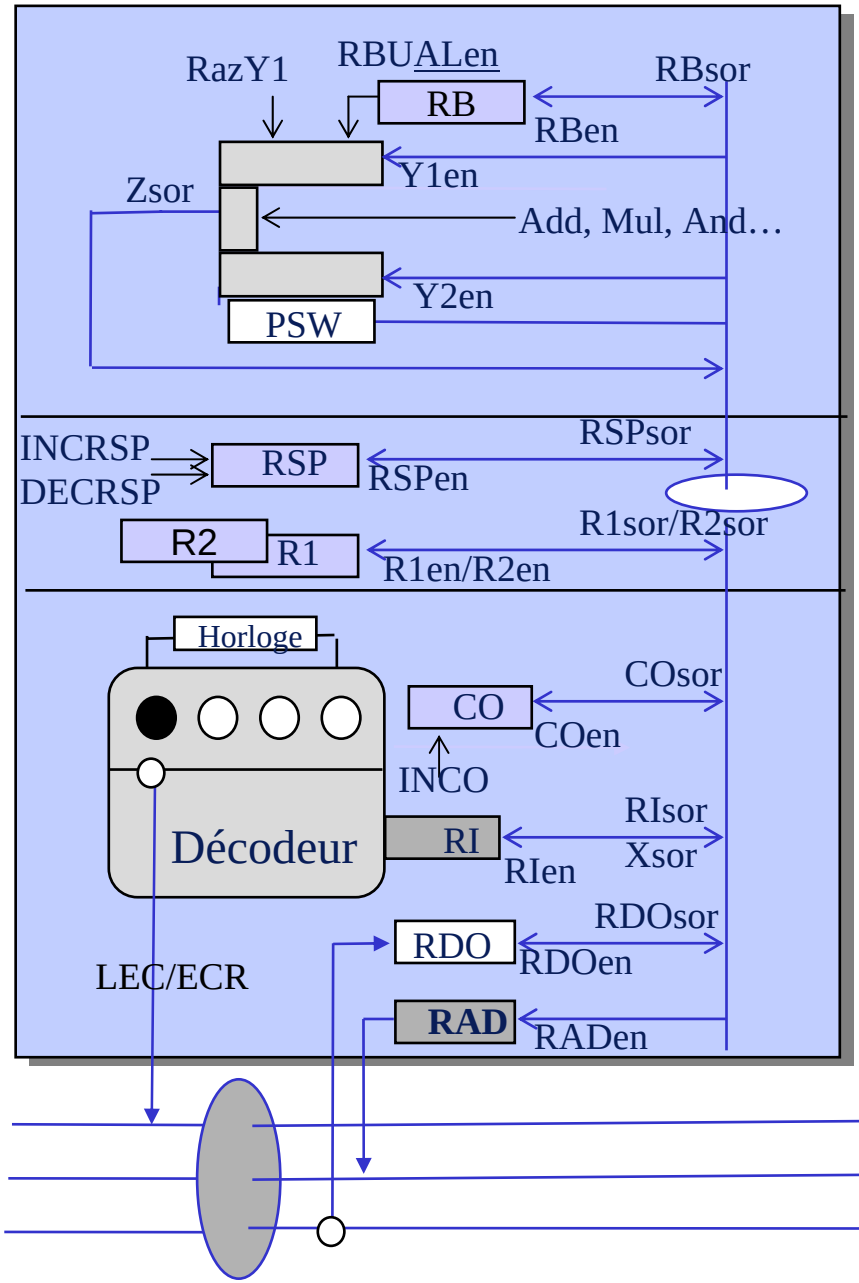
L' unité de commande :

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

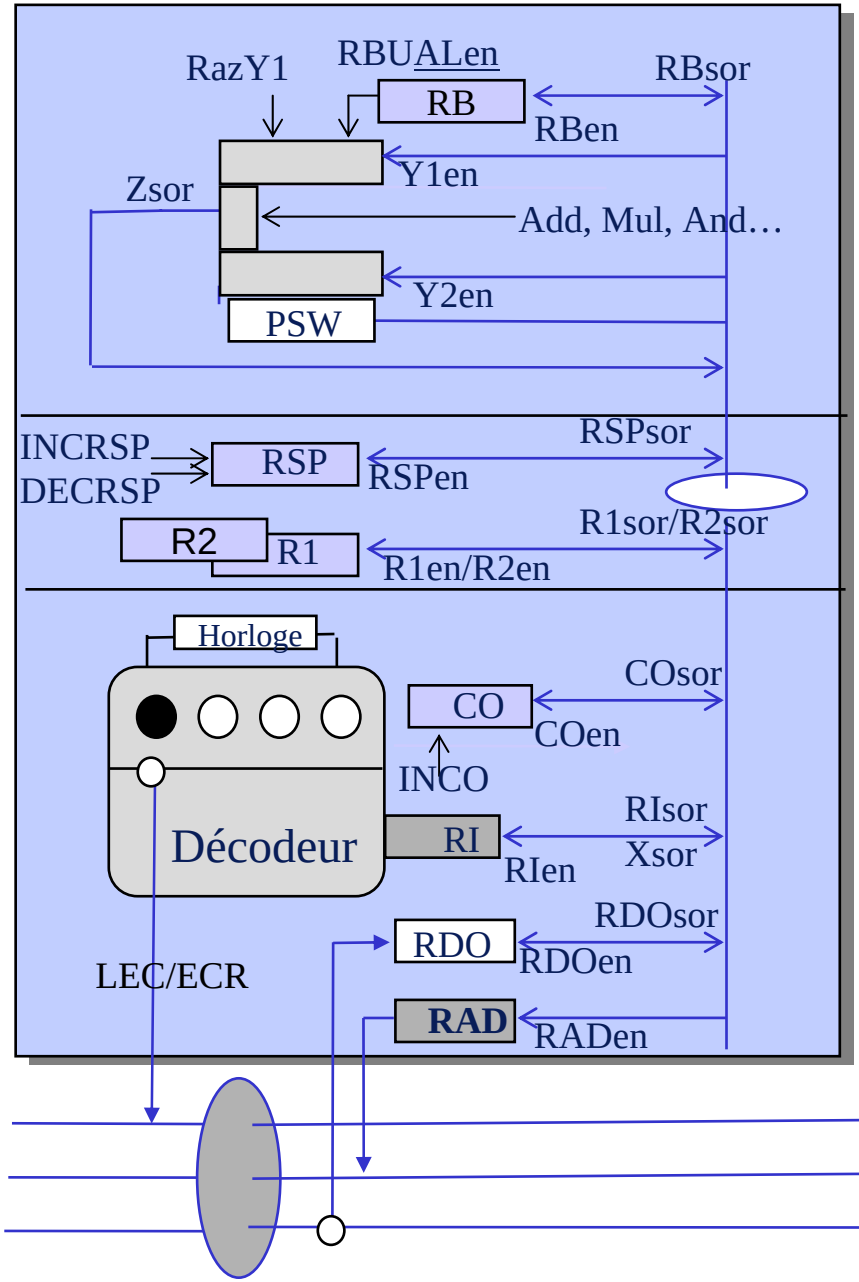
Micro commandes

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

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
R1en, R1sor R2en, R2sor	Entrée dans R1, Sortie de R1 Entrée dans R2, Sortie de R2
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



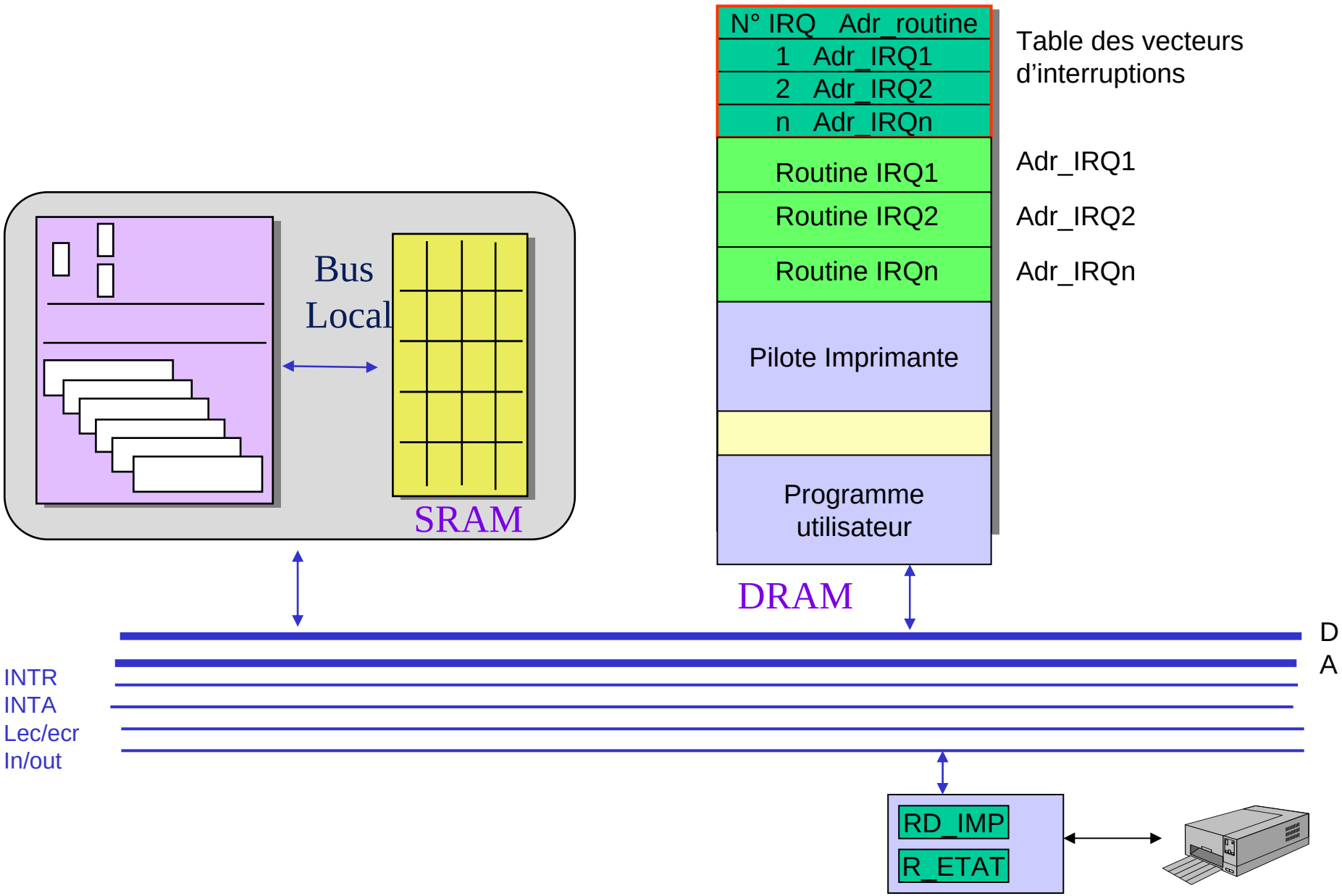
Load D R1 A	
Fetch	
CO -> RAD	COsor, RADen
Lecture	LEC
RDO→ RI	RDOsor, RIen
Incrément CO	INCO
Exécution	
A→ RAD	Xsor, RADen
Lecture	LEC
RDO→ R1	RDOsor, R1en

Entrées-sorties et interruptions

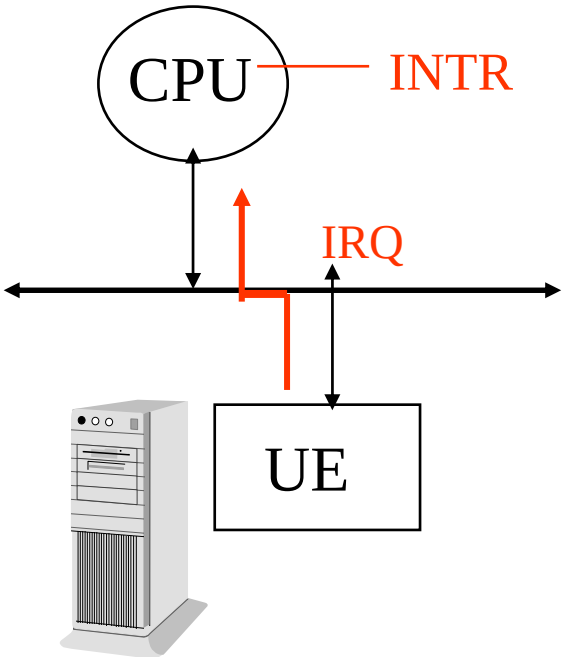
Contexte

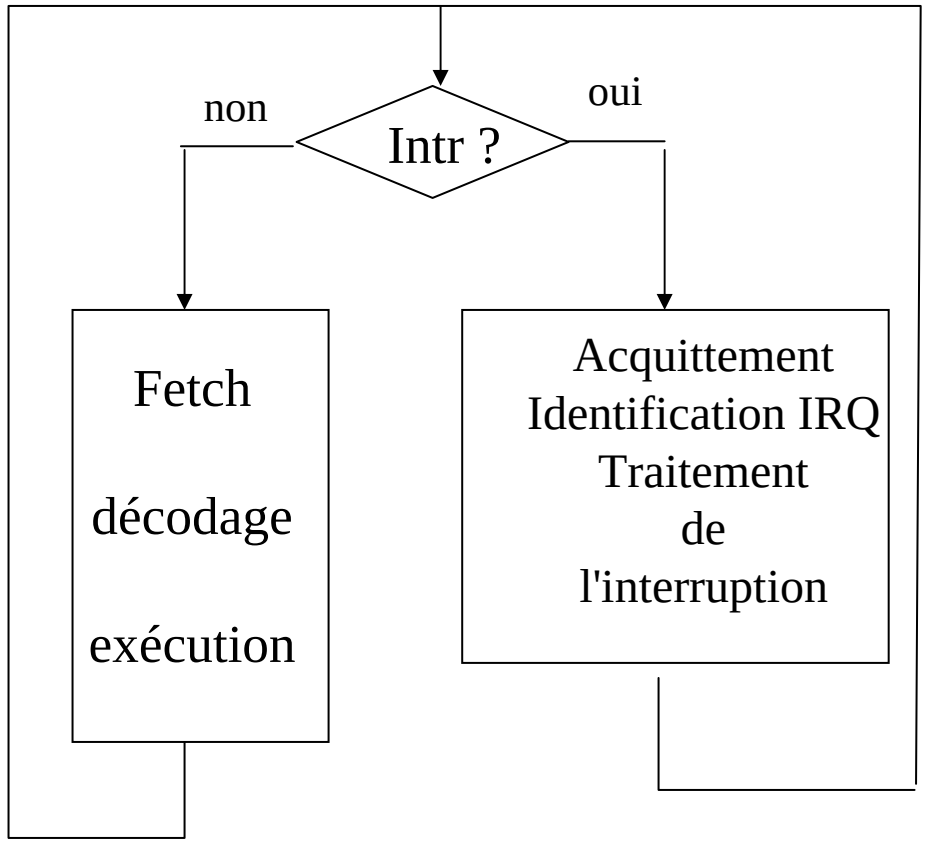
Le programme demande l'impression de la donnée A

```
Programme exemple
Entier A = 124;
Entier B;
Entier C
Début
A := A - 12;
Imprimer (A);
Si (A > 0)
Alors
            B := A;
Sinon
            C : = A;
Finsi
Fin
```


N° IRQ Adr_routine
1 Adr_IRQ1
2 Adr_IRQ2
n Adr_IRQn
Table des vecteurs d'interruptions
Routine IRQ1
Adr_IRQ1
Routine IRQ2
Adr_IRQ2
Routine IRQn
Adr_IRQn
Pilote Imprimante
Programme utilisateur
Bus Local
SRAM
DRAM
D
A
INTR
INTA
Lec/ecr
In/out
RD_IMP
R_ETAT

Mécanisme des interruptions

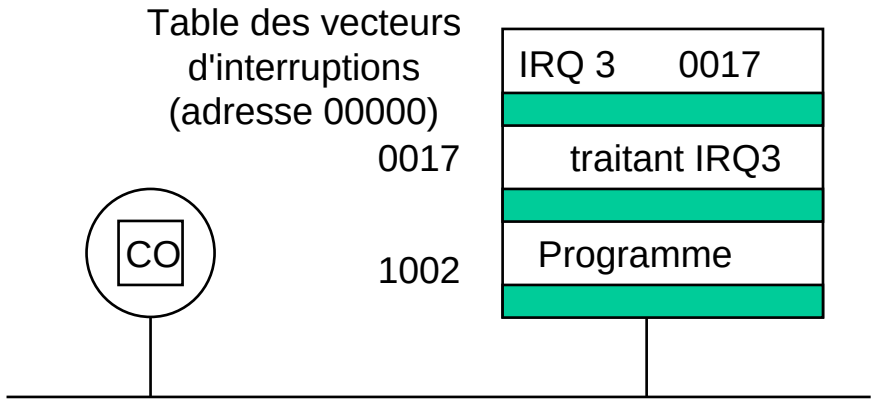


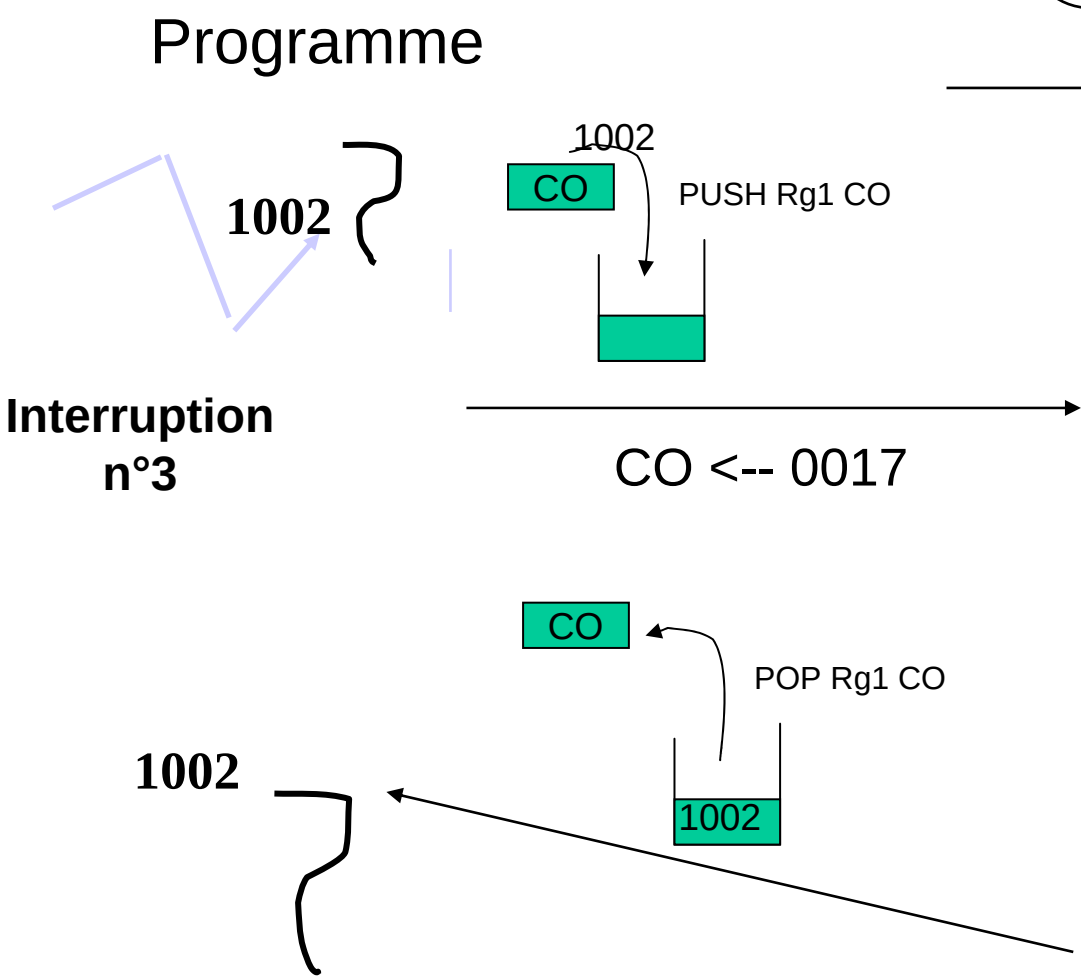


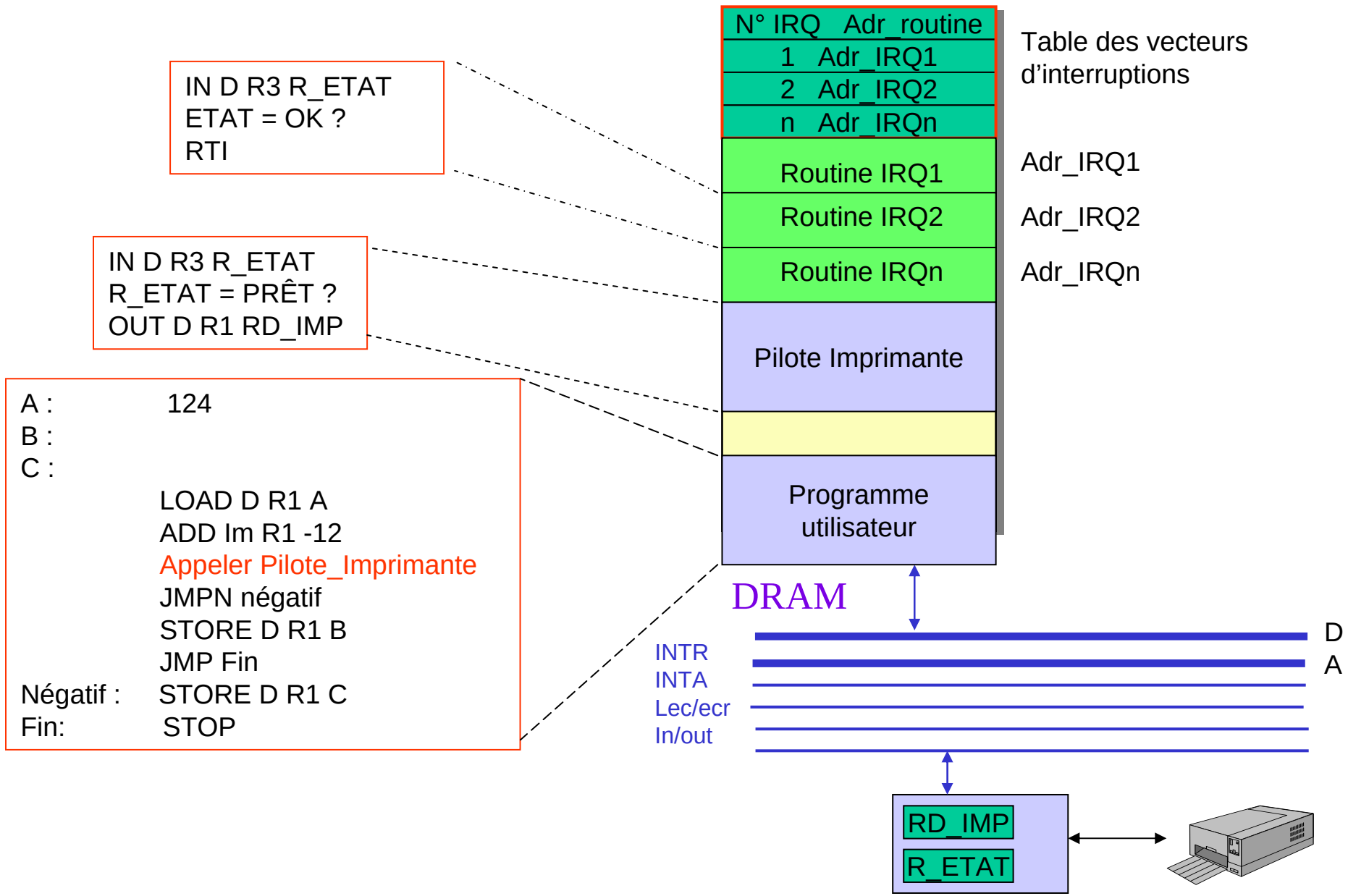
Un périphérique signale un événement au processeur en émettant une interruption matérielle.

Le processeur teste sa ligne d'entrée d'interruption (INTR) avant de commencer le traitement de l'instruction suivante du programme qu'il exécute.

Les Interruptions






IN D R3 R_ETAT
ETAT = OK ?
RTI
IN D R3 R_ETAT
R_ETAT = PRÊT ?
OUT D R1 RD_IMP
A : 124
B :
C :
LOAD D R1 A
ADD Im R1 -12
Appeler Pilote_Imprimante
JMPN négatif
STORE D R1 B
JMP Fin
Négatif : STORE D R1 C
Fin: STOP
N° IRQ Adr_routine
1 Adr_IRQ1
2 Adr_IRQ2
n Adr_IRQn
Table des vecteurs d'interruptions
Routine IRQ1
Routine IRQ2
Routine IRQn
Adr_IRQ1
Adr_IRQ2
Adr_IRQn
Pilote Imprimante
Programme utilisateur
DRAM
D
A
INTR
INTA
Lec/ecr
In/out
RD_IMP
R_ETAT

Le mode avec interruptions