

## MOT BINAIRE

- Une information élémentaire binaire est appelée **un bit**
- Un mot binaire est une collection de plusieurs bits:
  - On appelle **octet** un ensemble de 8 bits
  - Les mots binaires peuvent faire 16, 32 ou 64 bits



## NATURE DES MOTS BINAIRES

- Un mot binaire peut représenter:
  - soit un collection d'information binaires indépendantes
  - soit des informations codées (caractères, couleurs, ....)

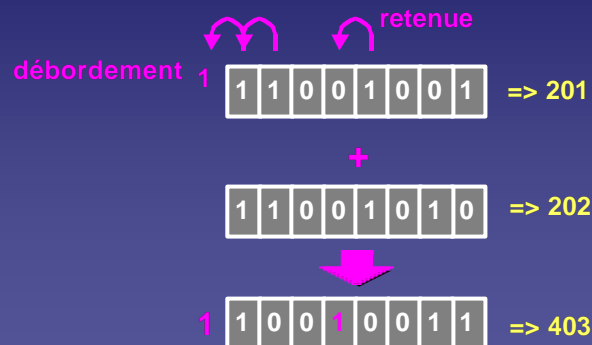
par exemple: le caractère **S** est codé **01010011**

- soit un entier codé en binaire:

0000	=>	0
0001	=>	1
0010	=>	2
0011	=>	3
0100	=>	4
.....		
1110	=>	14
1111	=>	15
- soit une adresse mémoire

## ARITHMETIQUE BINAIRE

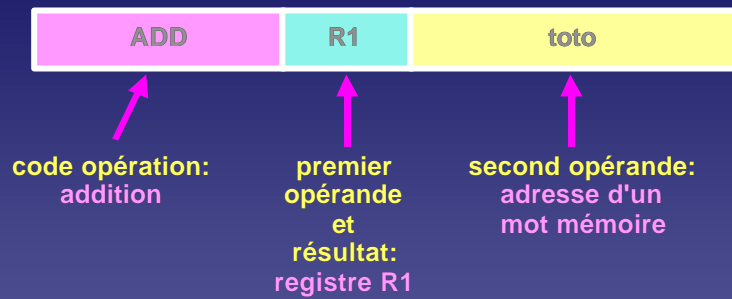
- Il est possible de faire des opérations arithmétiques entre des nombres binaires



## LES INSTRUCTIONS

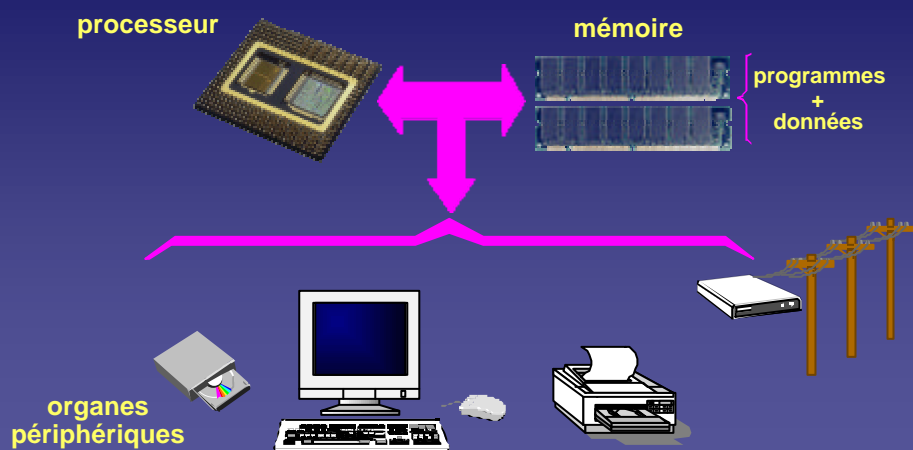
- Le processeur exécute des instructions.
- Celles-ci définissent les opérations élémentaires exécutées par le processeur.
  - ces opérations sont généralement très simples (addition, soustraction, et, ou, non,...)
  - les opérandes sont également spécifiés par chaque instruction. Ils peuvent être:
    - des registres du processeur
    - des positions mémoire

## EXEMPLE D'INSTRUCTION



$R1 \leftarrow R1 + \text{toto}$

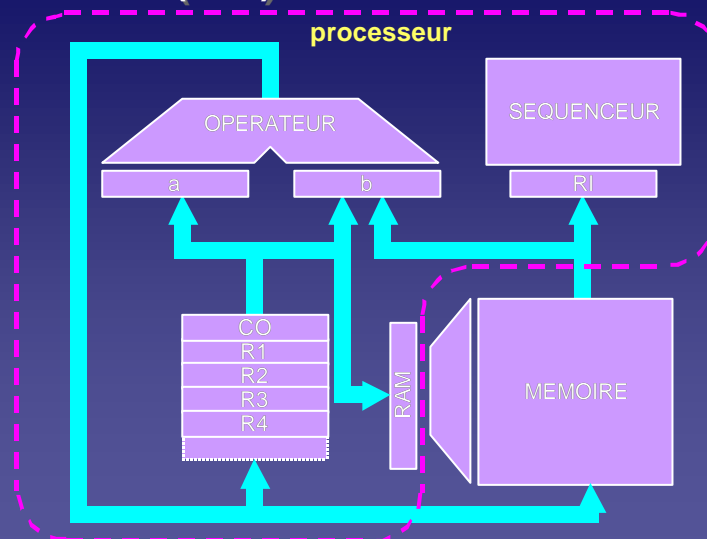
## ECHANGES DANS UN ORDINATEUR



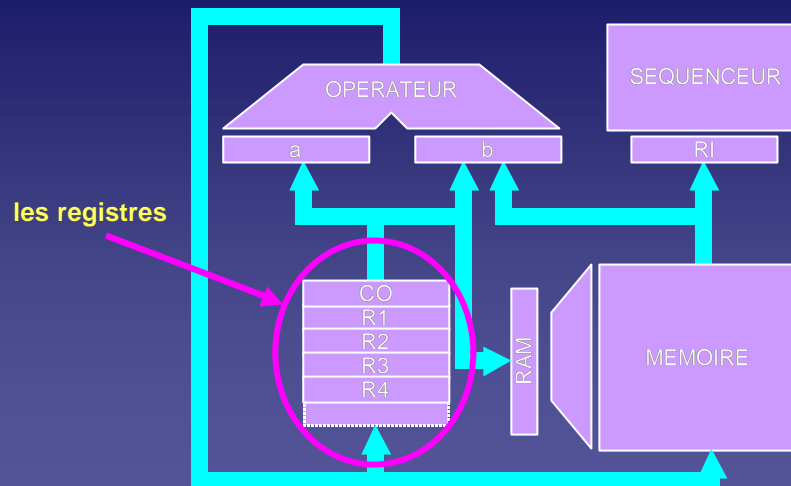
## UN PROGRAMME

- Les programmes sont constitués de suites d'instructions
- Un programme est généralement constitué d'un assez grand nombre d'instructions (quelques centaines de milliers)
- Les instructions qui constituent un programme sont rangées à la suite dans la mémoire

## PROCESSEUR (très) SIMPLIFIE



## LES REGISTRES



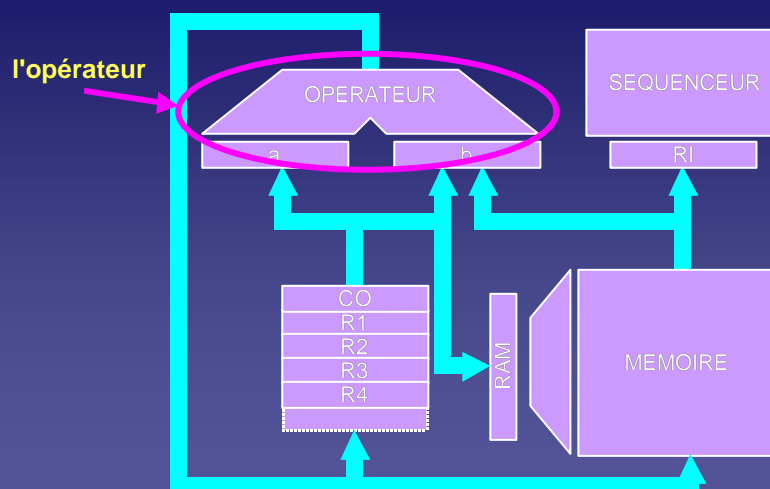
## LES REGISTRES

- Dispositifs électroniques capables de contenir de l'information binaire (généralement des mots de 32 bits)
- En nombre limité
- Servent à stocker des informations de travail
  - variables de traitement
  - compteur ordinal
  - état de la machine

## REGISTRE D'ETAT

- Il existe un registre particulier, qui stocke l'état du processeur.
- Ce registre contient des bits qui décrivent les propriétés du dernier résultat calculé.
  - Signe, nullité, débordement, parité

## L'OPERATEUR



## L'OPERATEUR

### ■ Dispositif électronique capable d'effectuer des opérations binaires arithmétiques et logiques simples entre ses opérandes

- addition, soustraction, incrémentation, décrémentation,
- ET, OU, OU-exclusif, complémentation logique
- décalages, rotations

## OPERATIONS LOGIQUES

### ■ Opérations entre deux opérandes binaires

- **OU** => vaut 1 si, et seulement si, l'un, au moins, des opérandes vaut 1
- **ET** => vaut 0 si, et seulement si, l'un, au moins, des opérandes vaut 0  
(si l'un des opérandes vaut 1 => le résultat vaut l'autre opérande)
- **OU exclusif** => vaut 0 si les deux opérandes sont égaux  
vaut 1 si les deux opérandes sont différents
- Il existe 16 opérations possibles sur deux opérandes

### ■ Opération sur un opérande binaire

- **NON** => vaut 1 si l'opérande vaut 0  
vaut 0 si l'opérande vaut 1

## OPERATIONS LOGIQUES SUR DES MOTS

- Dans un ordinateur les opérations logiques se font bit à bit entre deux mots binaires

1 0 1 1 0 1 0 1

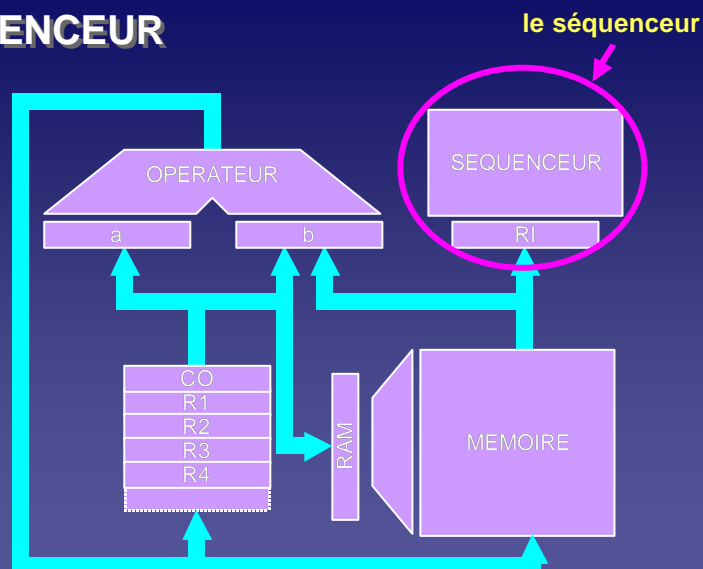
ET

1 1 1 1 0 0 1 1



1 0 1 1 0 0 0 1

## LE SEQUENCEUR





## LE SEQUEUR

### ■ Dispositif électronique qui assure:

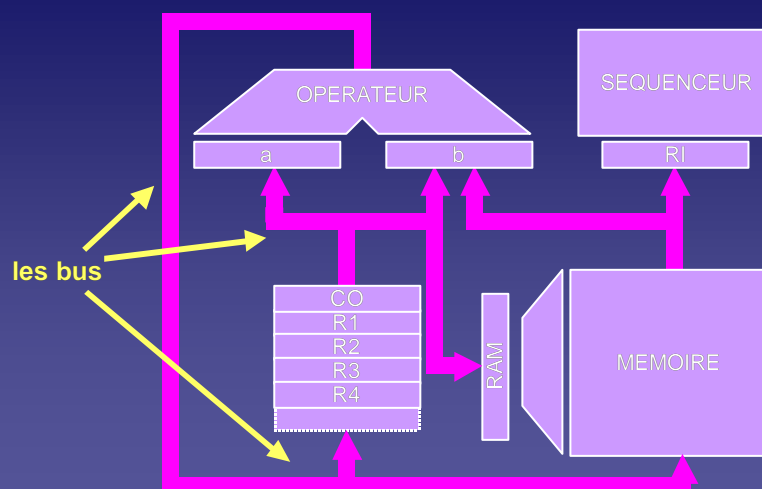
- le décodage de l'instruction courante
- le séquençement de l'exécution de cette instruction

### ■ Une instruction s'exécute en un nombre de pas qui dépend de sa complexité

- Les premiers pas consistent à lire l'instruction courante en mémoire

### ■ L'instruction courante est rangée dans le registre-instruction

## LES BUS



## LES BUS

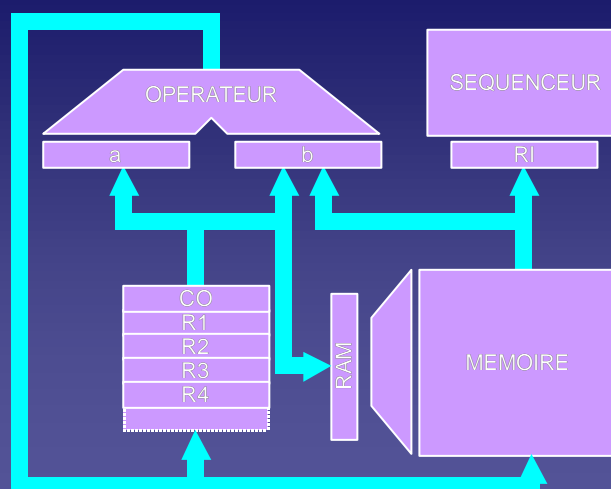
■ **Les bus sont des nappes de fils destinées à véhiculer de l'information entre les différents organes d'un processeur**

- à un instant donné un bus n'a qu'une source
- il peut avoir plusieurs destinations simultanées

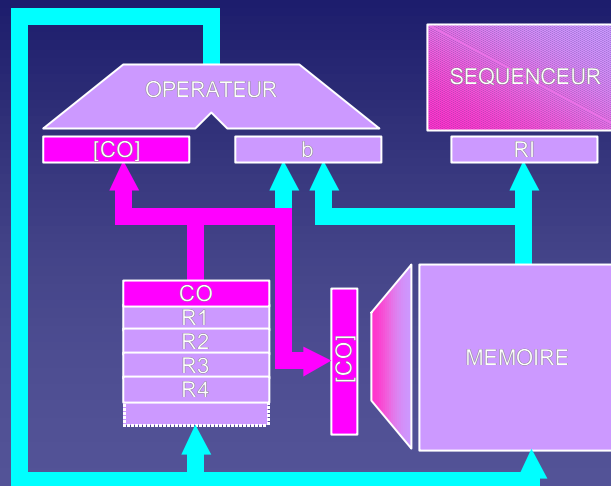
■ **Les différents organes d'un ordinateur sont aussi reliés par des bus**

- ces bus peuvent avoir plusieurs sources (une seule est active à la fois)

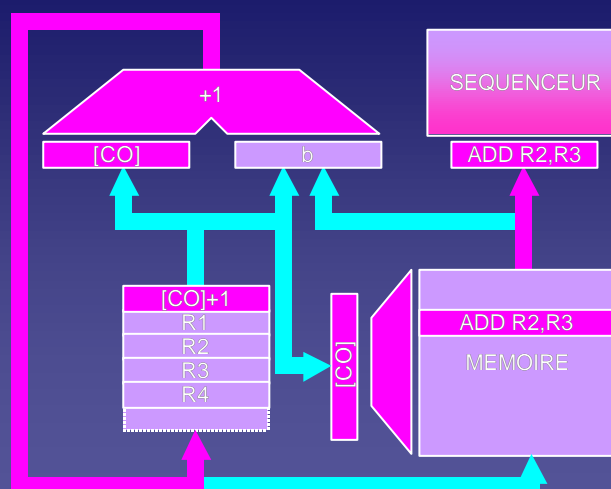
## EXECUTION D'UNE INSTRUCTION



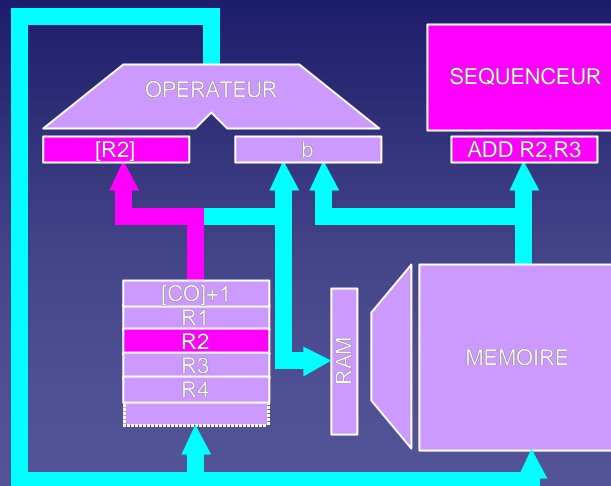
## PREMIER CYCLE



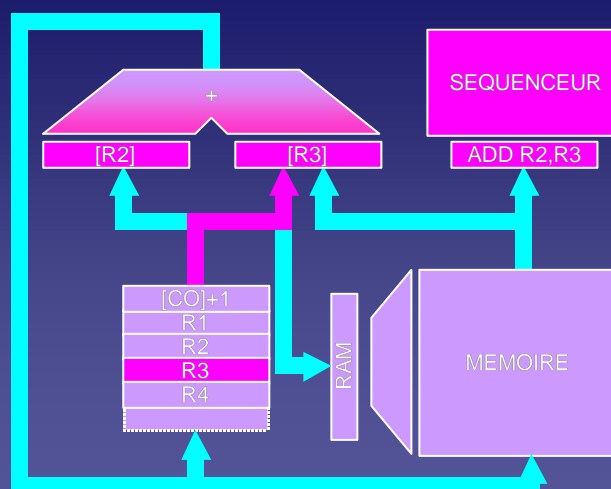
## SECOND CYCLE



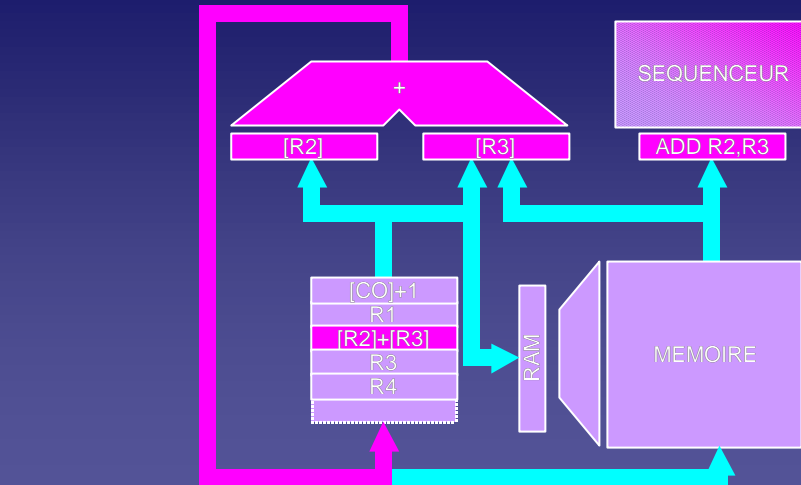
## TROISIEME CYCLE



## QUATRIEME CYCLE



## CINQUIEME CYCLE



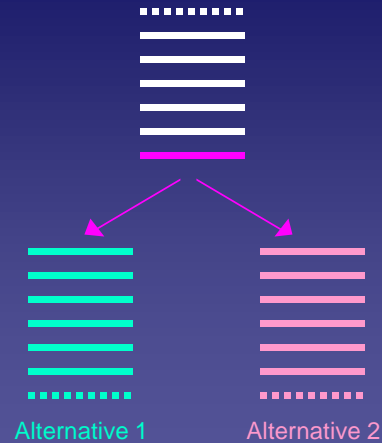
## NOMBRE DE CYCLES D'EXECUTION

- **Le nombre de cycles nécessaires pour exécuter une instruction dépend de la complexité de celle-ci**
  - Les premiers pas consistent à lire l'instruction courante en mémoire
- **Sur les machines modernes, différentes techniques sont utilisées pour accélérer l'exécution des instructions.**
  - On arrive à exécuter plusieurs instructions par cycle (Le Pentium 4 exécute en moyenne 2,5 instructions par cycle!)

## PRISES DE DECISIONS DANS UN PROGRAMME

### ■ Il existe des instructions qui permettent de prendre des décisions

- Suivant la valeur d'un résultat, ou la comparaison de deux variables, on choisit la séquence d'instructions qui constitue la suite du programme



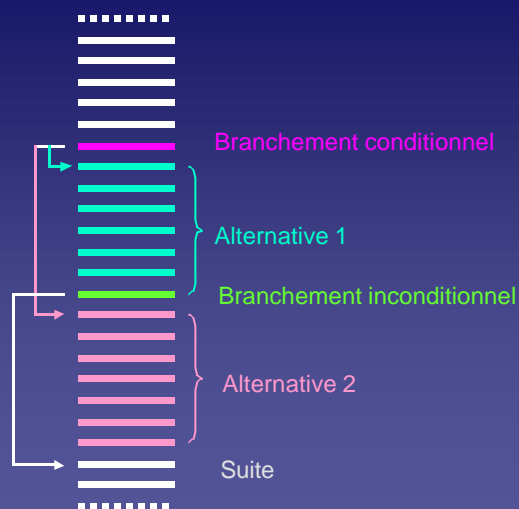
## BRANCHEMENT CONDITIONNEL

### ■ Par commodité, une instruction de branchement conditionnel détermine si la suite du programme est en séquence ou à une adresse donnée.

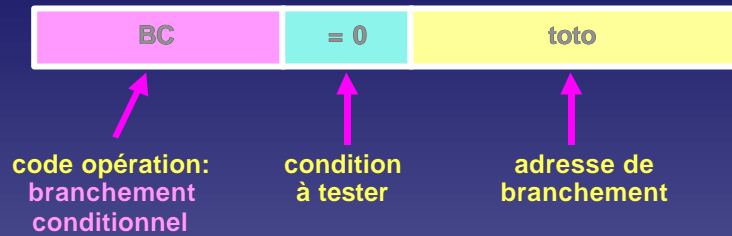
### ■ Le branchement conditionnel:

- soit teste des conditions préalablement calculées
- soit compare des opérandes

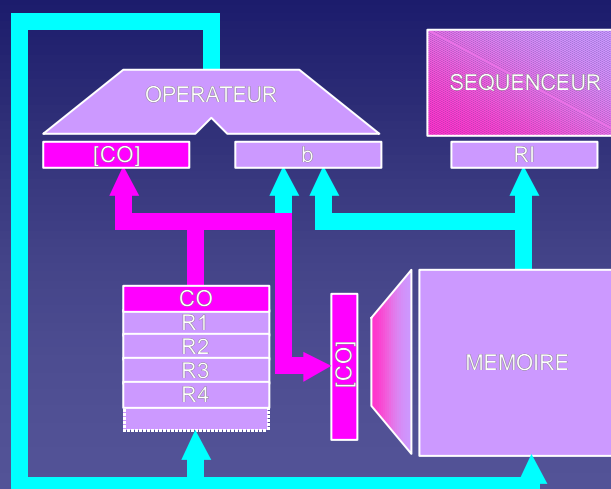
### ■ Un branchement inconditionnel permet de rejoindre les alternatives



## EXEMPLE DE BRANCHEMENT CONDITIONNEL



## PREMIER CYCLE



The diagram illustrates the control unit architecture with the following components and connections:

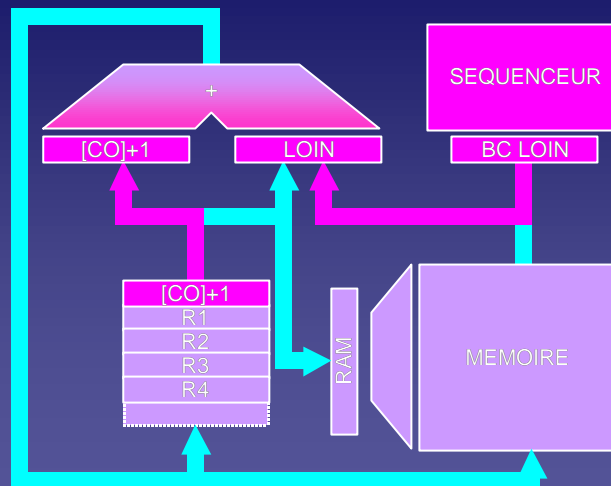
- SEQUENCEUR**: The main control unit, containing a **BC LOIN** (remote binary counter) and **MEMOIRE** (memory).
- BC LOIN**: A binary counter that receives control signals and provides feedback to the **SEQUENCEUR**.
- MEMOIRE**: A memory unit that stores control signals and provides feedback to the **SEQUENCEUR**.
- Control Signals**: Represented by red arrows, these signals flow from the **SEQUENCEUR** to the **BC LOIN** and **MEMOIRE**, and from the **BC LOIN** and **MEMOIRE** back to the **SEQUENCEUR**.
- Data Path**: Represented by a blue arrow, it flows from the **SEQUENCEUR** to the **BC LOIN** and **MEMOIRE**, and from the **BC LOIN** and **MEMOIRE** back to the **SEQUENCEUR**.
- Arithmetic Unit**: A trapezoidal block labeled **+1** (incrementer) that receives control signals and provides feedback to the **SEQUENCEUR**.
- Registers**: A vertical stack of registers labeled **[CO]+1**, **R1**, **R2**, **R3**, and **R4**. These registers receive control signals and provide feedback to the **SEQUENCEUR**.
- Control Signals**: Represented by red arrows, these signals flow from the **SEQUENCEUR** to the **BC LOIN** and **MEMOIRE**, and from the **BC LOIN** and **MEMOIRE** back to the **SEQUENCEUR**.
- Data Path**: Represented by a blue arrow, it flows from the **SEQUENCEUR** to the **BC LOIN** and **MEMOIRE**, and from the **BC LOIN** and **MEMOIRE** back to the **SEQUENCEUR**.

Diagram illustrating the architecture of a micro-ordinateur (microcomputer). The components and their connections are as follows:

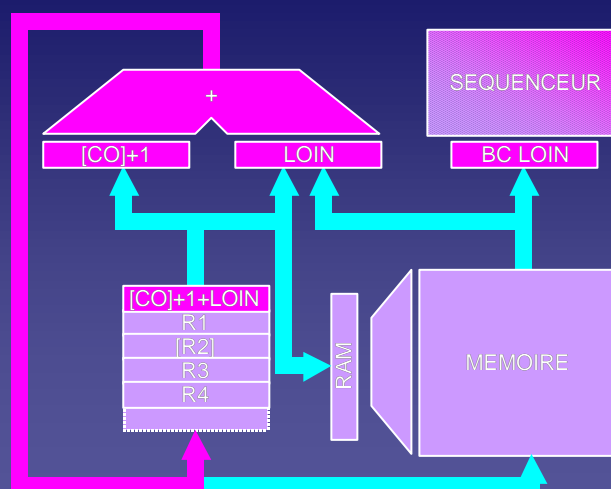
- OPERATEUR** (Operator) is connected to **CO+1** (Control) and **b** (bus).
- SEQUENCEUR** (Sequencer) is connected to **BC LOIN** (Far BC) and **b**.
- MEMOIRE** (Memory) is connected to **CO+1** and **b**.
- CO+1** (Control) is connected to **b** and **BC LOIN**.
- b** (bus) is connected to **SEQUENCEUR** and **BC LOIN**.



## TROISIEME CYCLE (si condition vraie)



## QUATRIEME CYCLE



## MODES D'ADRESSAGE

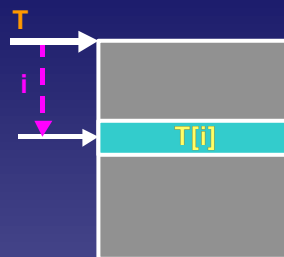
■ Pour faciliter l'utilisation des différentes structures de données, il existe différentes manières d'adresser des informations en mémoire.

- accès à des éléments de tableaux
- accès via des pointeurs
- accès aux éléments de structures

## ACCES A DES ELEMENTS DE TABLEAU

■ L'accès aux éléments d'un tableau se fait à l'aide d'un **index**

- on part de l'adresse du tableau
- à laquelle on ajoute le contenu d'un registre d'index
- le contenu du registre d'index peut être auto-incrémenté ou auto-décrémenté
- le registre d'index peut être général ou spécialisé
- dans l'instruction, la désignation de l'adresse comporte celle du registre d'index



## ACCES VIA UN POINTEUR

### ■ L'accès à un élément via un pointeur nécessite:

- L'utilisation d'une variable auxiliaire (appelée pointeur) qui contient l'adresse de l'élément visé
- Ce pointeur apparaît comme un registre dit **de base**
- Ce registre de base peut être général ou spécialisé

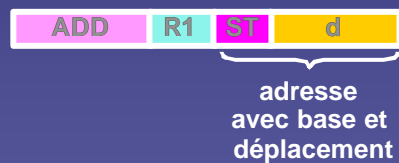
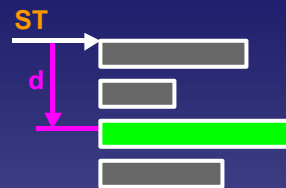
$V12 := V12 \text{ xor } B^{\wedge}$



## ACCES A UN ELEMENT DE STRUCTURE

### ■ L'accès à un élément de structure suppose:

- l'accès à la structure (par exemple via un pointeur)
- la sélection d'un élément grâce à un **déplacement**



## FORME GENERALE D'UNE ADRESSE

### ■ Suivant les machines, les adresses mémoires contiennent:

- un ou plusieurs registres pouvant jouer le rôle d'index et de base (ils peuvent être auto-incrémentés ou auto-décrémentés)
- une valeur immédiate pouvant jouer le rôle d'adresse ou de déplacement

### ■ L'adresse finale est obtenue en ajoutant les contenu des registres de base et d'index ainsi que la valeur immédiate

## GESTION DE LA MEMOIRE

### ■ Comme la mémoire est très vaste, elle peut contenir plusieurs programmes à la fois.

- cela signifie qu'un programme donné peut s'exécuter quelque soit son emplacement en mémoire.
- il ne doit donc pas contenir d'adresses mémoire absolues, mais des adresses **relatives** (à son début).
- Il existe donc des registres spéciaux (dits de **segment**) qui repèrent le début des programmes et qui sont chargés au moment de l'exécution d'un programme.