

# Architecture des Systèmes à Processseur

# Sommaire

## 1. Calculateur :

Définition

Fonctions à réaliser

Fonctions de base et éléments actifs

Données, programme, résultats

Principe

Liaison Unité Centrale - Unité d'Echange

Liaison Unité Centrale - Mémoire Centrale

Signaux de contrôle de communication

Notion de BUS

Caractéristiques des BUS

## 2. Instruction :

Définition

Code Opération et Opérande

Modes d'adressage :

Mode d'adressage Immédiat

Mode d'adressage Direct

Mode d'adressage par Registre

Mode d'adressage Implicite

## 3. Structure programmable :

Mémoire Centrale

Pointeur d'Instruction

Registre d'Instruction

Lecture d'une donnée en mémoire

Registre de travail

Unité Arithmétique et Logique

Ecriture d'une donnée en mémoire

Séquence typique de traitement d'une donnée

## 4. Structure matérielle :

Bloc de calcul

Indicateurs

Mémoire Centrale

Bloc de commande

Registres spécialisés supplémentaires

Unité d'Echange

# Sommaire

## 5. Représentation de l'information :

Rappels de numération binaire

Notation hexadécimale

Entiers naturels et retenue

Nombres entiers relatifs

Entiers relatifs et débordement

Code ASCII

## 6. Présentation de SIMCAL :

Structure et IHM

E-S parallèle

E-S série

Temporisateur

Contrôleur d'Interruption

Mémoire Centrale et Entrées-Sorties

## 7. Etapes du développement d'un programme :

Fichiers produits

## 7. Etapes du développement du programme :

Fichiers produits

## 8. Langage d'assemblage de SIMCAL :

Syntaxe

Instructions

Directives

Instructions de transfert

Instructions logiques et arithmétiques

Instructions de branchement

Instructions de contrôle

# Sommaire

## 9. Sous-programme et Pile :

Sous-programme

Pile

Organisation en mémoire

Evolution de la pile

## 10. Entrées-Sorties :

Unité d'Echange

Circuit d'entrée-sortie

Structure d'un circuit d'entrée-sortie

Entrée par test de mot d'état

Sortie par test de mot d'état

Test de mot d'état

## 11. Interruptions :

Principe

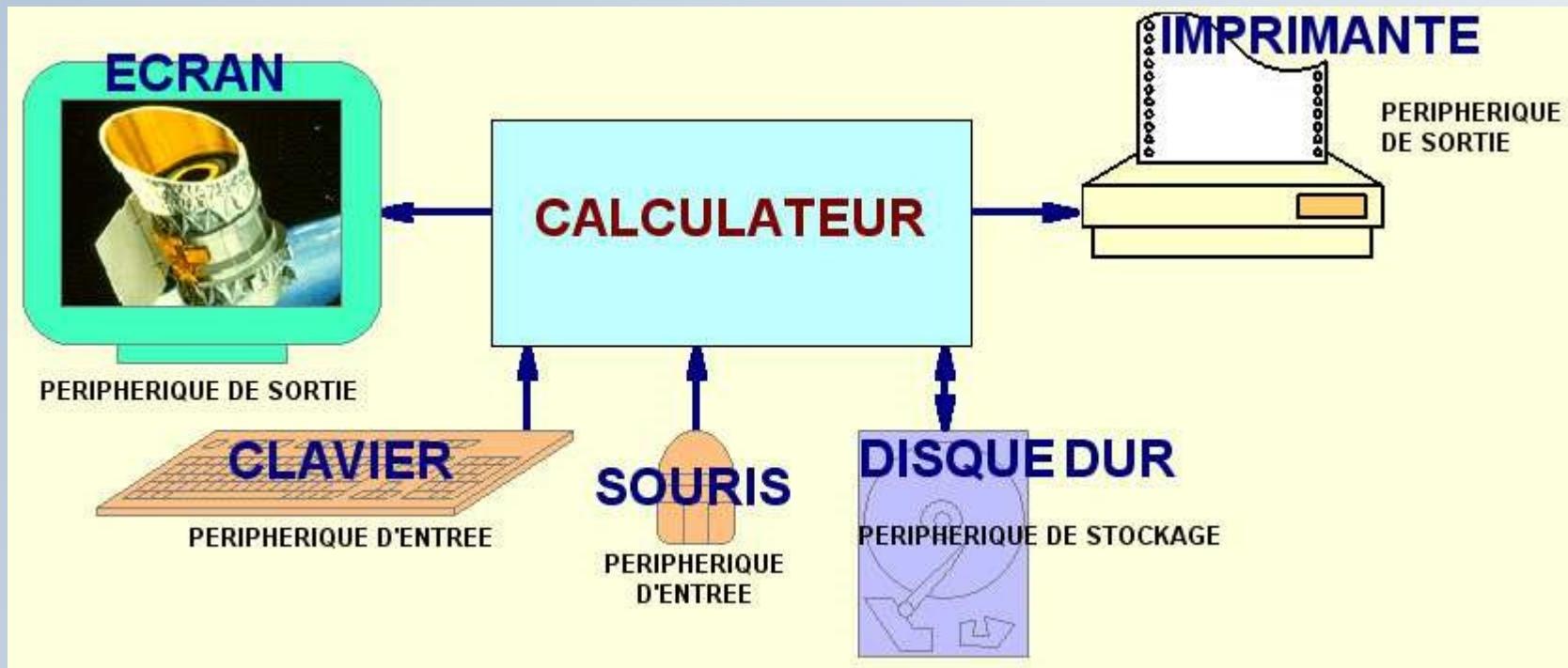
Séquence de traitement

Vectorisation

Contrôleur d'interruption

Entrée-sortie par interruption

# 1. Calculateur : définition



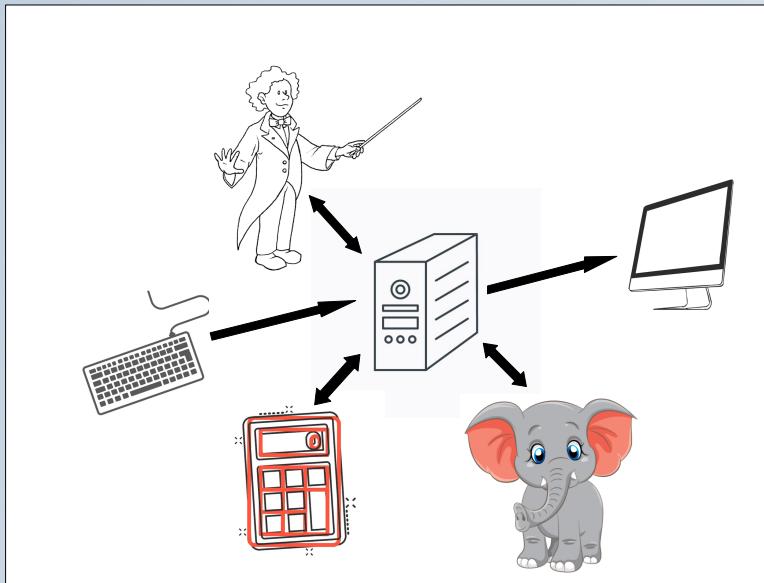
Le calculateur est une machine électronique capable de traiter numériquement des informations.

Exemple de calculateur : l'Ordinateur.

La communication Homme-Calculateur nécessite des PERIPHERIQUES d'entrée-sortie.

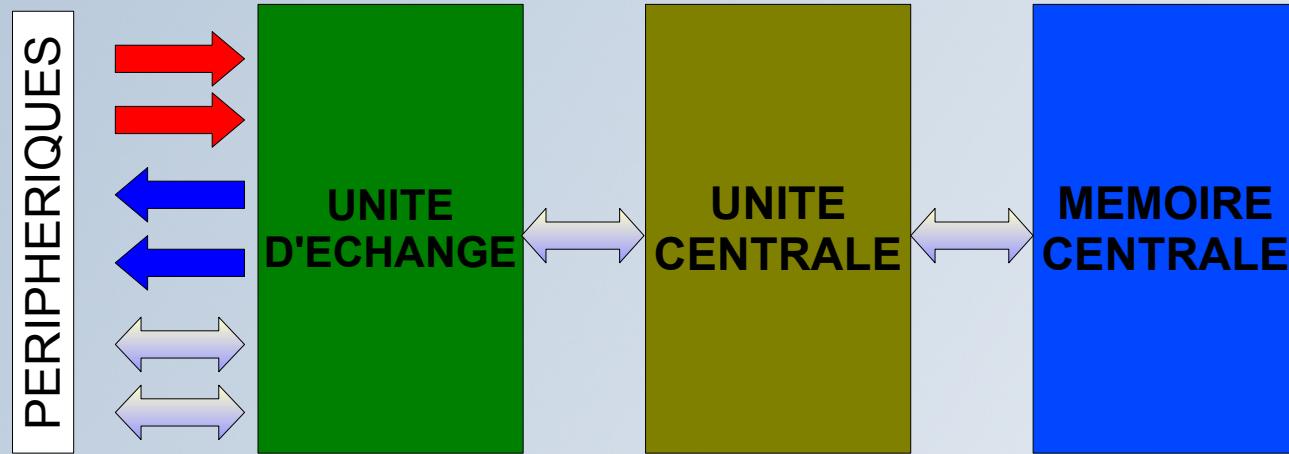
Le calculateur peut également disposer de périphériques de stockage ou mémoires de masse : disque dur, CD, DVD, mémoire flash

# 1. Calculateur : fonctions à réaliser



- Entrée des informations (programme, données) par un **PERIPHERIQUE D'ENTREE** (exemple: clavier)
- Mémorisation des informations : programme + données + résultats par la **MEMOIRE CENTRALE**
- Exécution du calcul grâce à un jeu d'instructions par le **BLOC DE CALCUL**
- Conduite du calcul à l'aide de commandes élémentaires par le **BLOC DE COMMANDE**
- Sortie des informations (résultats) par un **PERIPHERIQUE DE SORTIE** (exemple : écran)

# 1. Calculateur : fonctions de base et éléments actifs



<i>fonction</i>	<i>élément actif</i>	<i>informations</i>
Entrée	Unité d'échange	Données à traiter, programme de traitement (en provenance de périphériques d'entrée)
Mémorisation	Mémoire centrale	Données à traiter, programme de traitement, résultats
Traitement	Unité centrale	Calculs et ordonnancement des opérations
Sortie	Unité d'échange	Résultats des traitements (à destination des périphériques de sortie)

# 1. Calculateur : données, programme, résultats

Donnée à traiter

Programme de traitement

Résultat du traitement

Une Donnée à traiter est une valeur numérique binaire provenant :

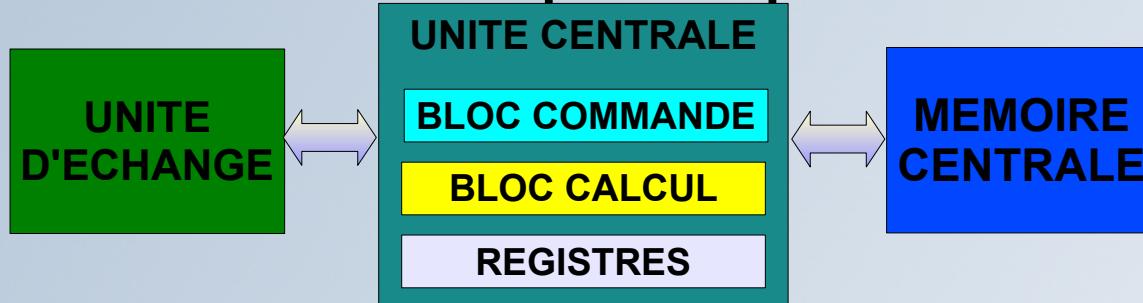
- de l'extérieur via l'Unité d'Echange
- ou
- de la Mémoire Centrale

Le Programme de traitement est une **suite d'instructions**, chaque instruction étant une valeur numérique binaire.

Le résultat d'un traitement est une valeur numérique binaire que l'on peut :

- stocker dans la Mémoire Centrale
- ou
- sortir à destination d'un périphérique
- ou
- considérer comme une nouvelle donnée à traiter

# 1. Calculateur : principe



L'Unité Centrale est constituée de deux blocs :

- un Bloc de Commande qui ordonne les calculs et gère les transferts avec la Mémoire Centrale et l'Unité d'Echange
- un Bloc de Calcul qui effectue les calculs ordonnés par le Bloc de Commande

Des registres, certains spécialisés, d'autres généraux, complètent l'Unité Centrale.

La Mémoire Centrale stocke le programme de traitement (instructions). Elle peut également stocker des données à traiter et des résultats

L'Unité d'Echange regroupe les circuits d'interface nécessaires à la gestion des périphériques. Ces circuits spécialisés adaptent les données à échanger entre Unité Centrale et périphérique :

- vitesse
- format
- type (série/parallèle)
- signaux de dialogue

Une donnée provenant d'un périphérique d'entrée est lue sur un circuit d'entrée par l'Unité Centrale et stockée dans un registre du bloc de calcul.

Une donnée à destination d'un périphérique de sortie est écrite sur le circuit de sortie par l'Unité Centrale à partir du bloc de calcul.

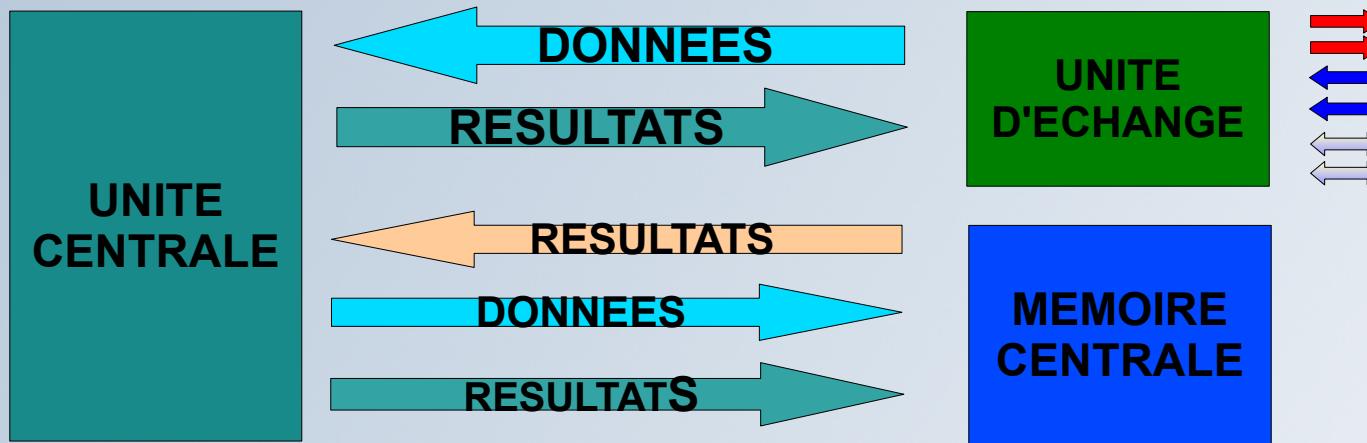
# 1. Calculateur : Liaison Unité Centrale - Unité d'Echange



L'Unité Centrale communique avec l'Unité d'Echange, elle peut :

- LIRE des informations fournies par l'Unité d'Echange:  
=> acquisition de DONNEES A TRAITER
  
- ECRIRE des informations dans l'Unité d'Echange:  
=> sortie de RESULTATS

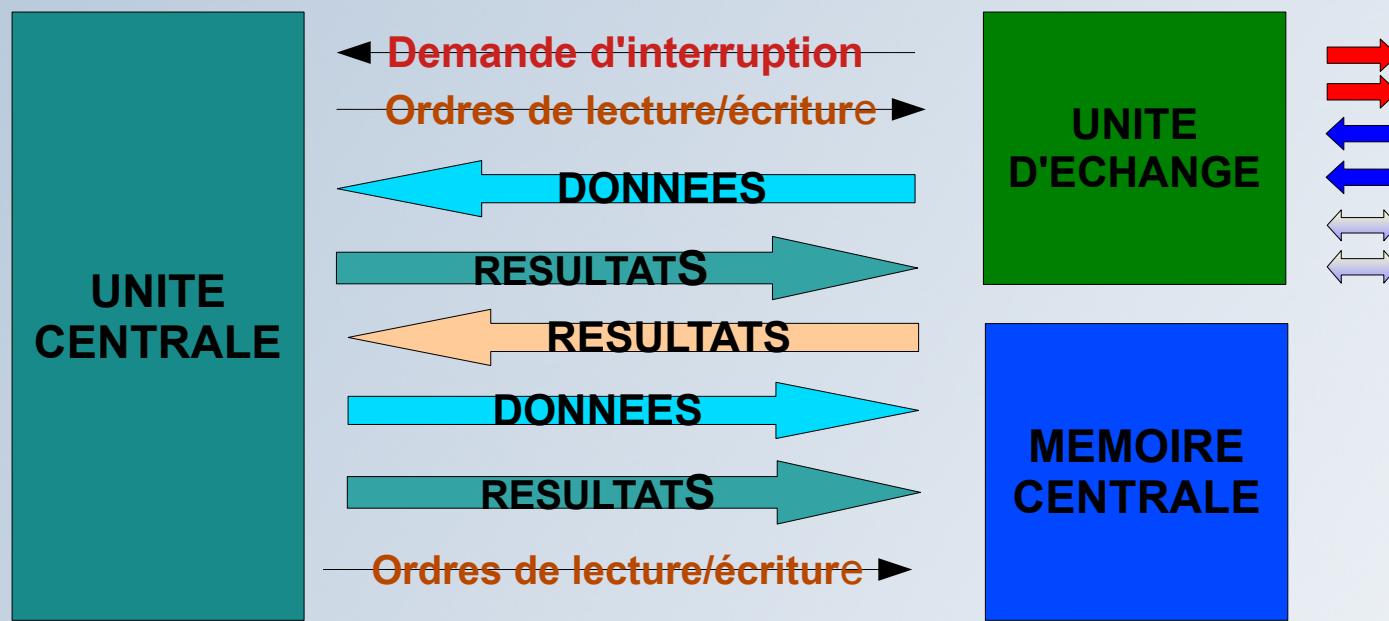
# 1. Calculateur : Liaison Unité Centrale - Mémoire Centrale



L'Unité Centrale communique aussi avec la Mémoire Centrale, elle peut :

- lire des informations fournies par la Mémoire Centrale :  
=> INSTRUCTIONS, DONNEES A TRAITER
- écrire des informations dans la Mémoire Centrale :  
=> stockage de RESULTATS

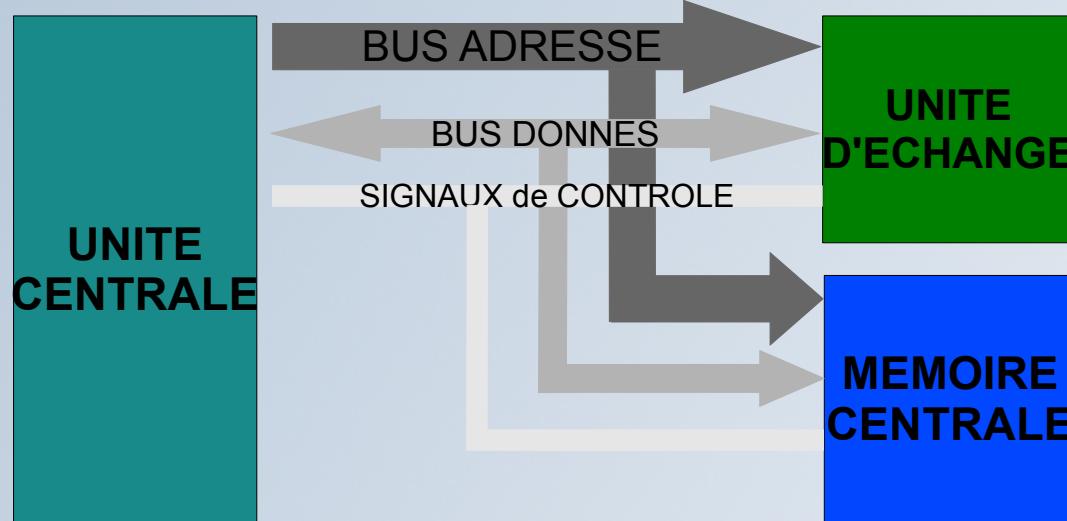
# 1. Calculateur : Signaux de contrôle de communication



Pour communiquer, l'Unité Centrale génère des ordres de lecture ou d'écriture,

- soit de sa propre initiative
- soit en réponse à une interruption demandée par l'Unité d'Echange (et un contrôleur d'interruptions)

# 1. Calculateur : notion de BUS



L'Unité Centrale doit sélectionner l'information à atteindre parmi toutes celles accessibles, pour cela elle doit fournir l'ADRESSE de cette information.

Chaque information en Mémoire Centrale occupe une "case mémoire" située à une ADRESSE dans l'espace de la Mémoire Centrale.

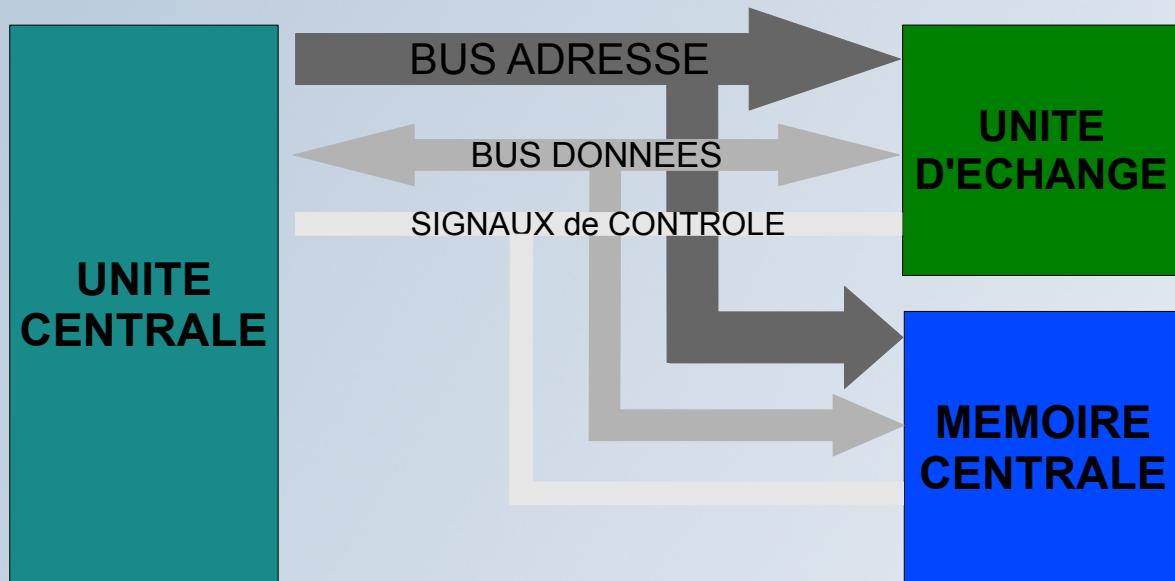
De même, chaque registre d'entrée-sortie occupe une ADRESSE dans l'espace de l'Unité d'Echange. Pour faciliter la communication entre les différents éléments du calculateur les connexions électriques de même nature sont regroupées par "paquet de fils" appelés BUS.

Le BUS de DONNEE permet d'acheminer des données entre Unité Centrale et Mémoire Centrale ainsi qu'entre Unité Centrale et Unité d'Echange.

Le BUS d'ADRESSE permet à l'Unité Centrale de sélectionner la donnée à atteindre par une combinaison binaire, cette combinaison binaire est l'adresse de la donnée.

L'Unité Centrale gère les SIGNAUX de CONTROLE pour donner les ordres nécessaires aux transferts.

# 1. Calculateur : caractéristiques de BUS



**BUS de DONNEES** : bidirectionnel

Largeur (bits en parallèle) : 8, 16, 32 ou 64 bits selon l'unité centrale

**BUS d'ADRESSE** : unidirectionnel

Largeur : 16, 20, 24, 32, 64 bits, selon l'unité centrale, sa largeur détermine l'espace adressable :

$$2^{16} = 65536 \text{ adresses (64 Kilo)}$$

$$2^{20} = 1048576 \text{ adresses (1 Méga)}$$

$$2^{24} = 16 \times 1048576 \text{ adresses (16 Méga)}$$

$$2^{32} = 4 \times 1048576 \times 1048576 \text{ adresses (4 Giga)}$$

## 2. : Instruction : définition

← INSTRUCTION →

0 1 1 0 1 0 1 1 1 1 0 0 1 0 1

Une instruction est une **valeur binaire** qui définit **le type d'opération** à effectuer (exemples: addition, transfert, ET logique).

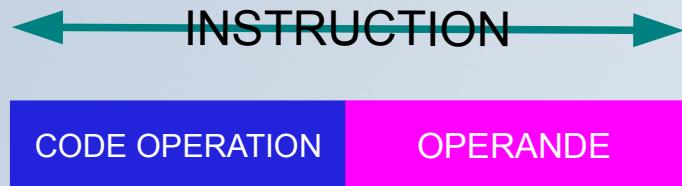
Si l'opération concerne une donnée, l'instruction doit également définir **la donnée**.

La donnée concernée peut être une VARIABLE de la zone de donnée, dans ce cas l'instruction doit indiquer comment atteindre cette donnée, et l'exécution de l'instruction entraînera la lecture ou l'écriture de cette donnée.

Si la donnée concernée est une CONSTANTE, sa valeur peut être codée dans l'instruction.

La méthode permettant d'atteindre une donnée est appelée **mode d'adressage**

## 2. : Instruction : code Opération et Opérande



L'instruction est un code binaire structuré en 2 champs de taille variable :

- **CODE OPERATION**: qui définit le type d'opération à effectuer
- **OPERANDE**: qui définit la donnée concernée par l'opération

## 2. : Instruction : modes d'adressage

Le mode d'adressage détermine la façon d'accéder à une donnée.

Différents modes existent dont les principaux sont :

- **Mode immédiat** : la donnée est immédiatement disponible, dans l'instruction,
- **Mode direct** : la donnée est en mémoire centrale, directement disponible, son adresse mémoire est fournie par l'instruction,
- **Mode registre** : source et destination se trouvent dans un registre,
- **Mode indirect par registre** : la donnée est en mémoire centrale, indirectement disponible, son adresse mémoire est fournie par un registre spécialisé dont l'identité est fournie par l'instruction,
- **Mode implicite** : la donnée est en un lieu précis du processeur, il n'est pas nécessaire d'en donner l'adresse. Concerne également les opérations dépourvues de donnée.

Ces modes peuvent également être utilisés par les instructions de branchement pour atteindre d'autres instructions. Dans le cas des instructions de branchement, il existe également un :

- **Mode relatif** : l'adresse à atteindre est obtenue par addition à l'adresse courante d'un entier relatif.

## 2. : Instruction : modes d'adressage immédiat



Mode IMMEDIAT :

- l'OPERANDE de l'instruction est la donnée concernée par l'opération
- la donnée est dans ce cas une CONSTANTE

## 2. : Instruction : modes d'adressage direct



Mode DIRECT :

- l'OPERANDE est une adresse
- il faudra effectuer une lecture ou écriture à cette adresse
- l'opérande peut être l'adresse d'une donnée à traiter, dans ce cas, celle-ci est généralement une VARIABLE
- l'opérande peut également être l'adresse d'une instruction, cas des branchements (pas dans notre cas car les branchements sont en relatif)

## 2. : Instruction : modes d'adressage indirect par registre



Mode INDIRECT par REGISTRE :

- l'adresse mémoire de la donnée concernée est fournie par un registre spécialisé
- le registre qui contient l'adresse de la donnée est défini par l'OPERANDE

## 2. : Instruction : modes d'adressage par registre



Mode par REGISTRE :

- l'opération se fait de registre à registre (le COP précise souvent le registre destination)
- l'OPERANDE est donc un ou plusieurs registre(s) contenant la (ou les) donnée(s)

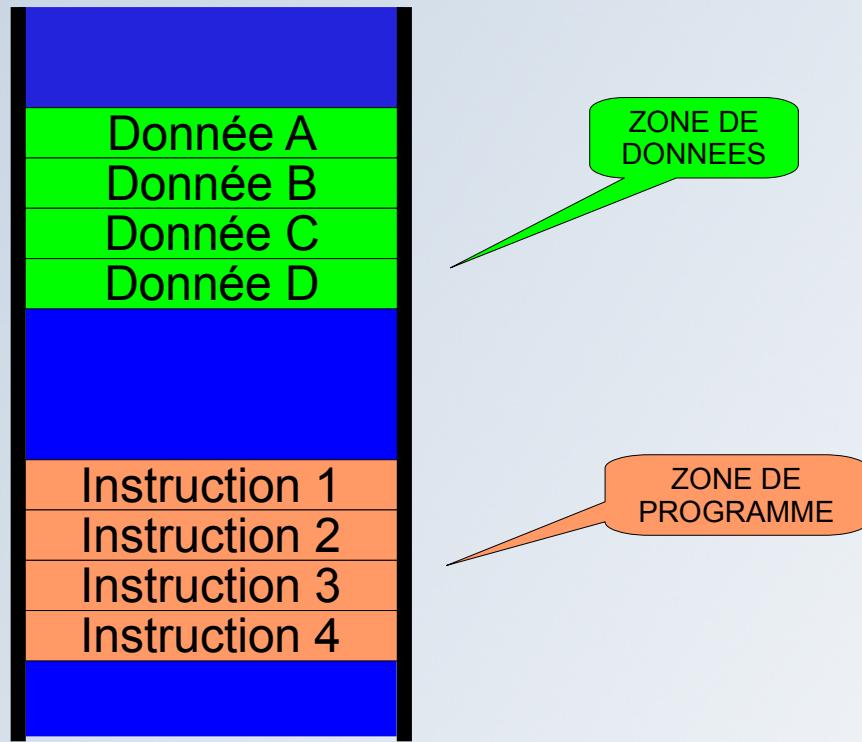
## 2. : Instruction : modes d'adressage implicite



Mode IMPLICITE :

- l'opération ne nécessite pas de donnée
- ou
- la donnée se trouve en un lieu précis du processeur

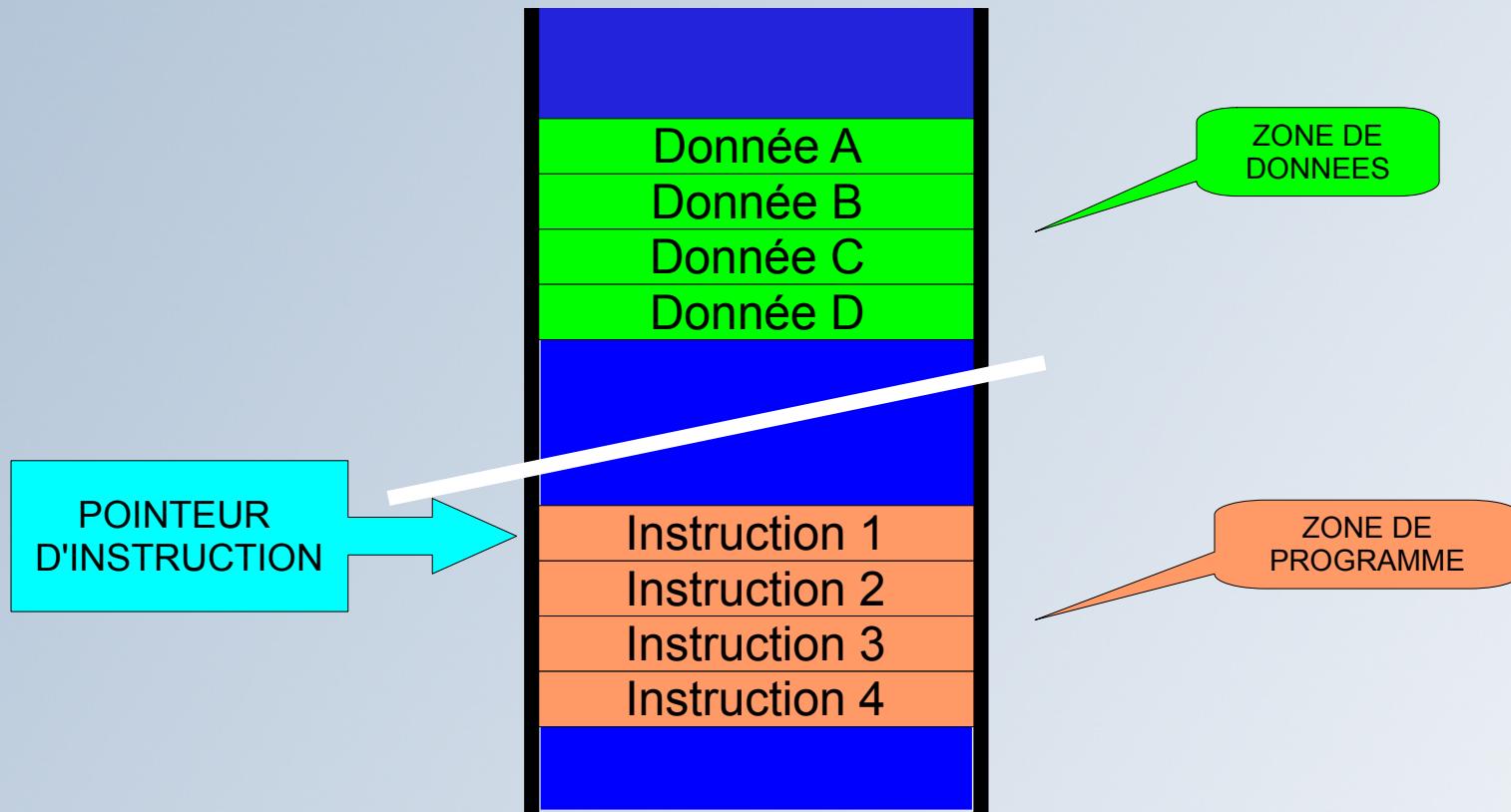
### 3. Structure programmable : Mémoire Centrale



La Mémoire Centrale est organisée en deux zones :

- la zone de DONNEES qui contient les données (variables) à traiter, et qui peut recevoir également les résultats
- la zone de PROGRAMME où se trouvent la suite d'instructions qui composent le programme de traitement (**indispensable**)

### 3. Structure programmable : Pointeur d'Instruction

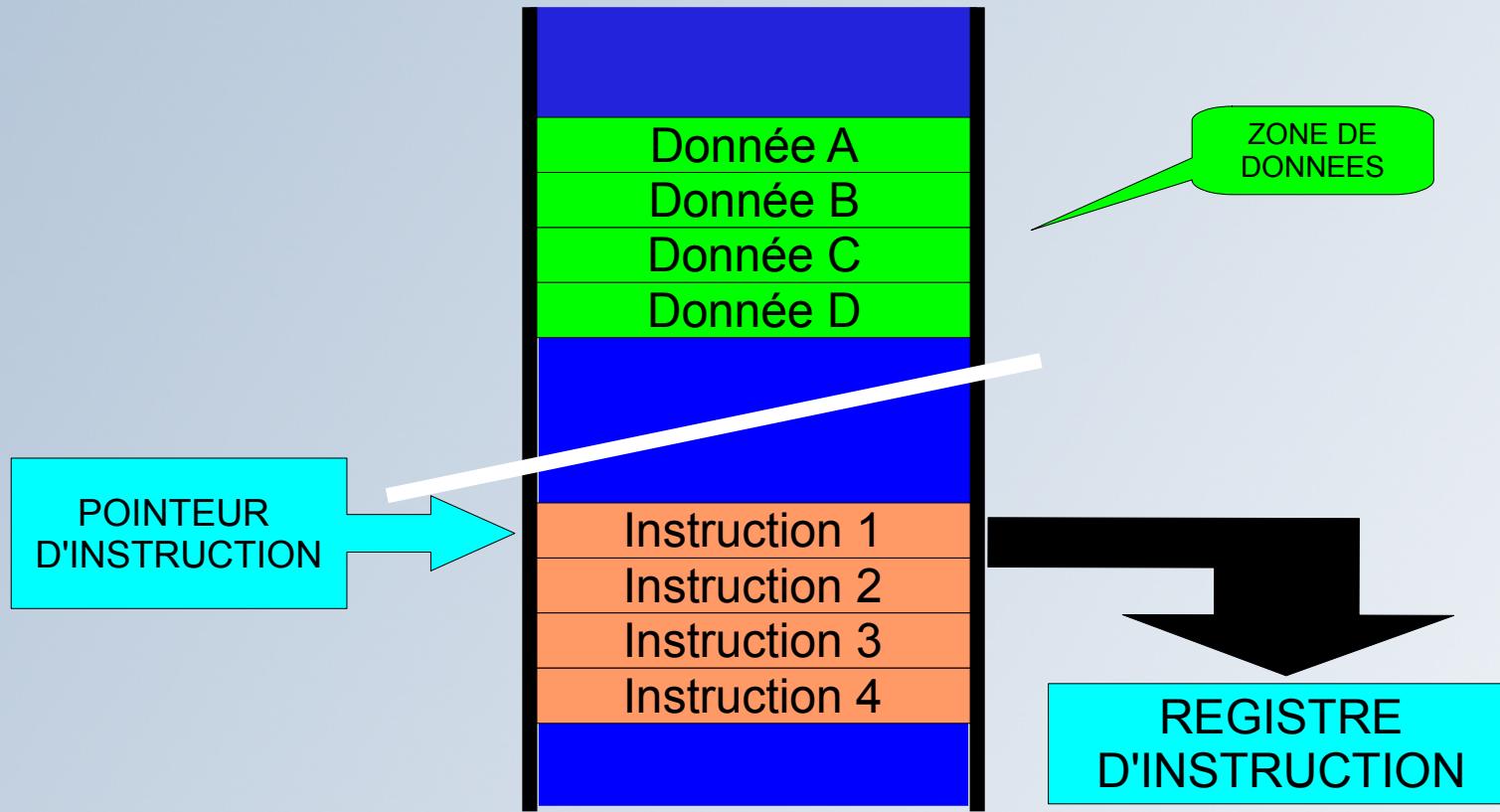


Pour exécuter une instruction, l'Unité Centrale doit la lire en Mémoire Centrale.

L'élément de l'Unité Centrale qui fournit l'adresse de l'instruction à lire est un compteur appelé **Pointeur d'Instruction (IP)**.

Le Pointeur d'Instruction fait partie du Bloc de Commande de l'Unité Centrale.

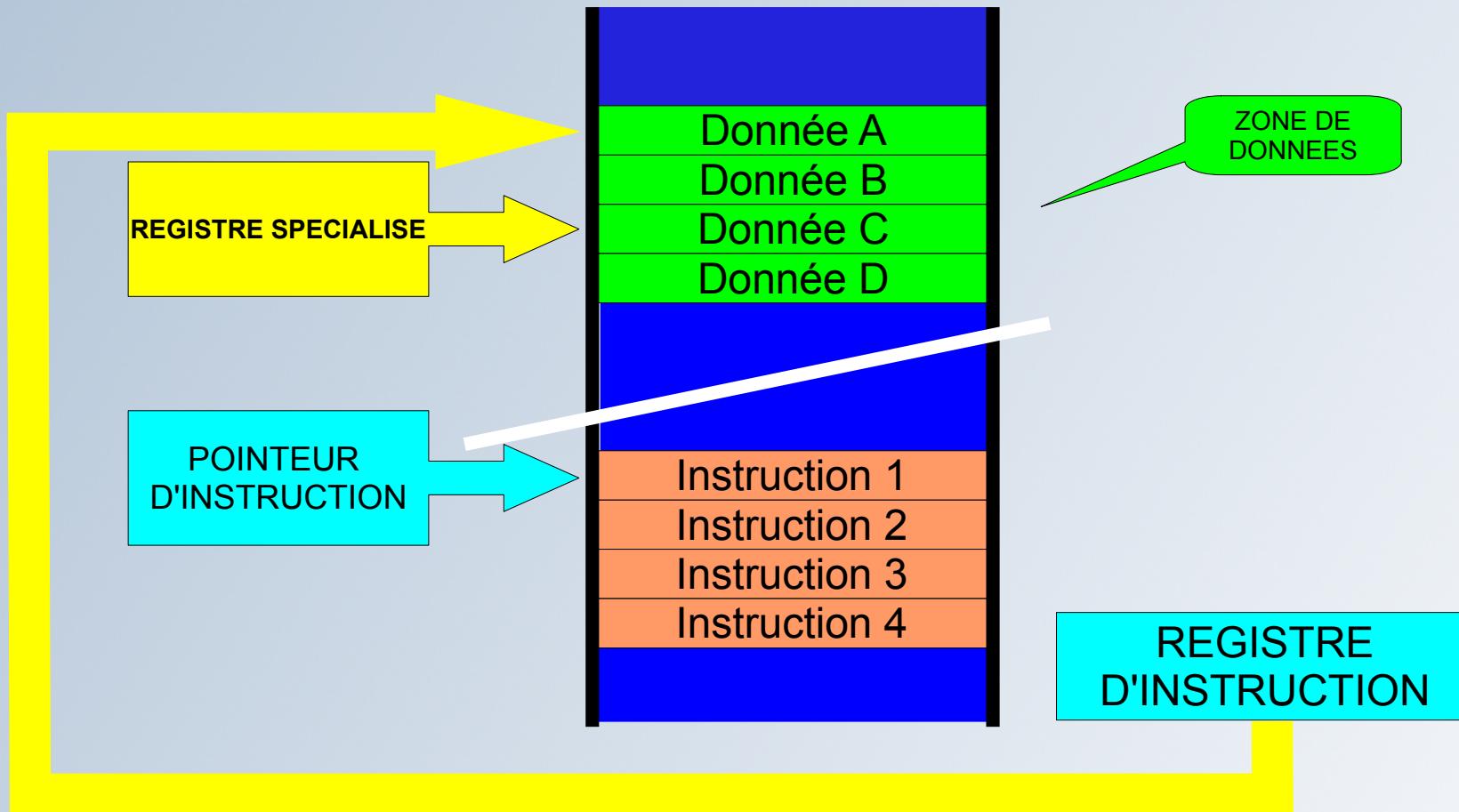
### 3. Structure programmable : Registre d'Instruction



L'instruction lue est chargée dans le **Registre d'Instruction**.

Le Registre d'Instruction fait partie du Bloc de Commande de l'Unité Centrale.

### 3. Structure programmable : lecture d'une donnée en mémoire

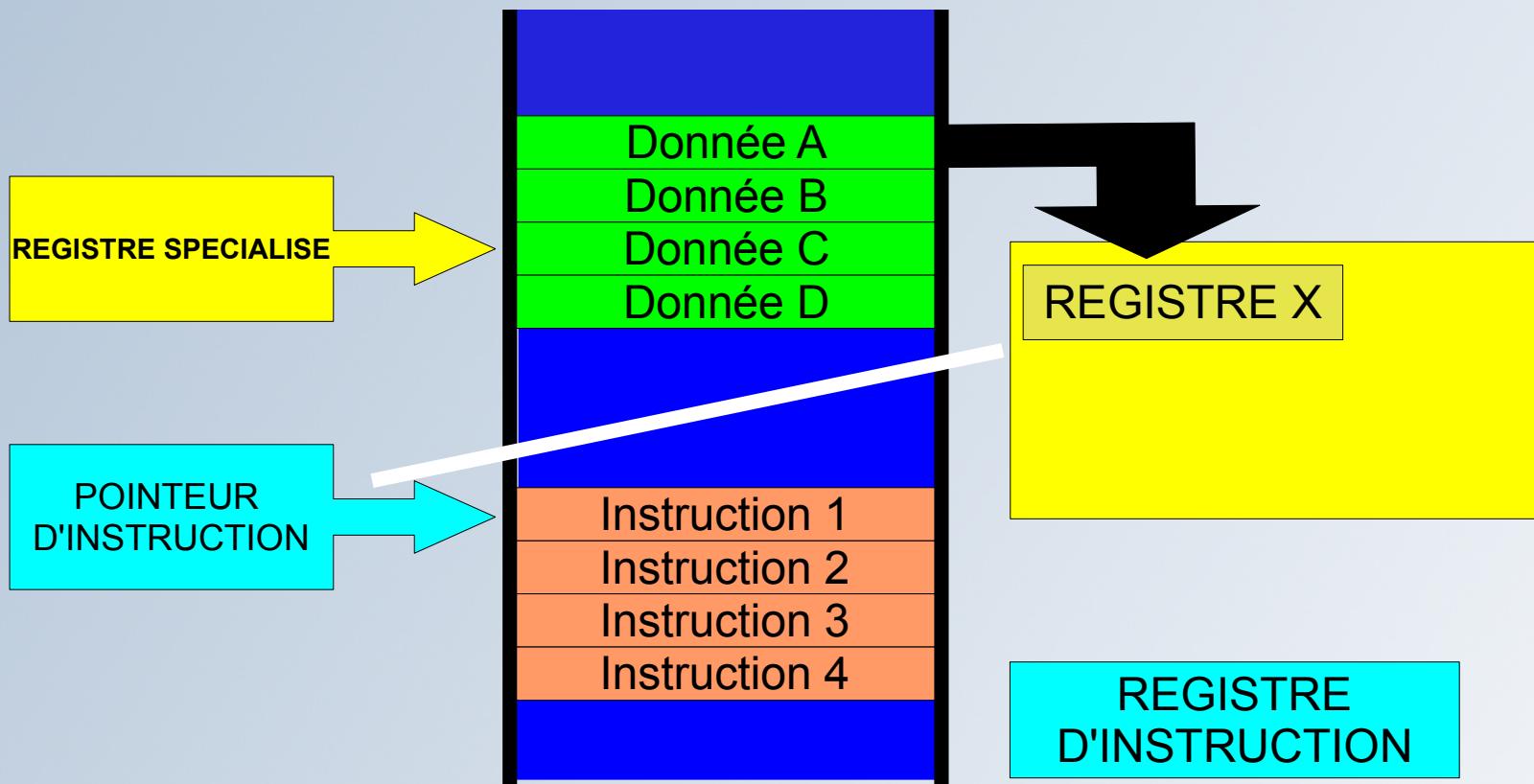


Si l'instruction doit traiter une variable en mémoire, l'Unité Centrale doit lire cette variable.

Pour cela, un **Registre** fournit l'adresse de la variable à traiter. Ce registre peut être :

- le registre d'instruction (**mode direct**)
- un registre spécialisé (**mode indirect par registre**)

### 3. Structure programmable : registre de travail

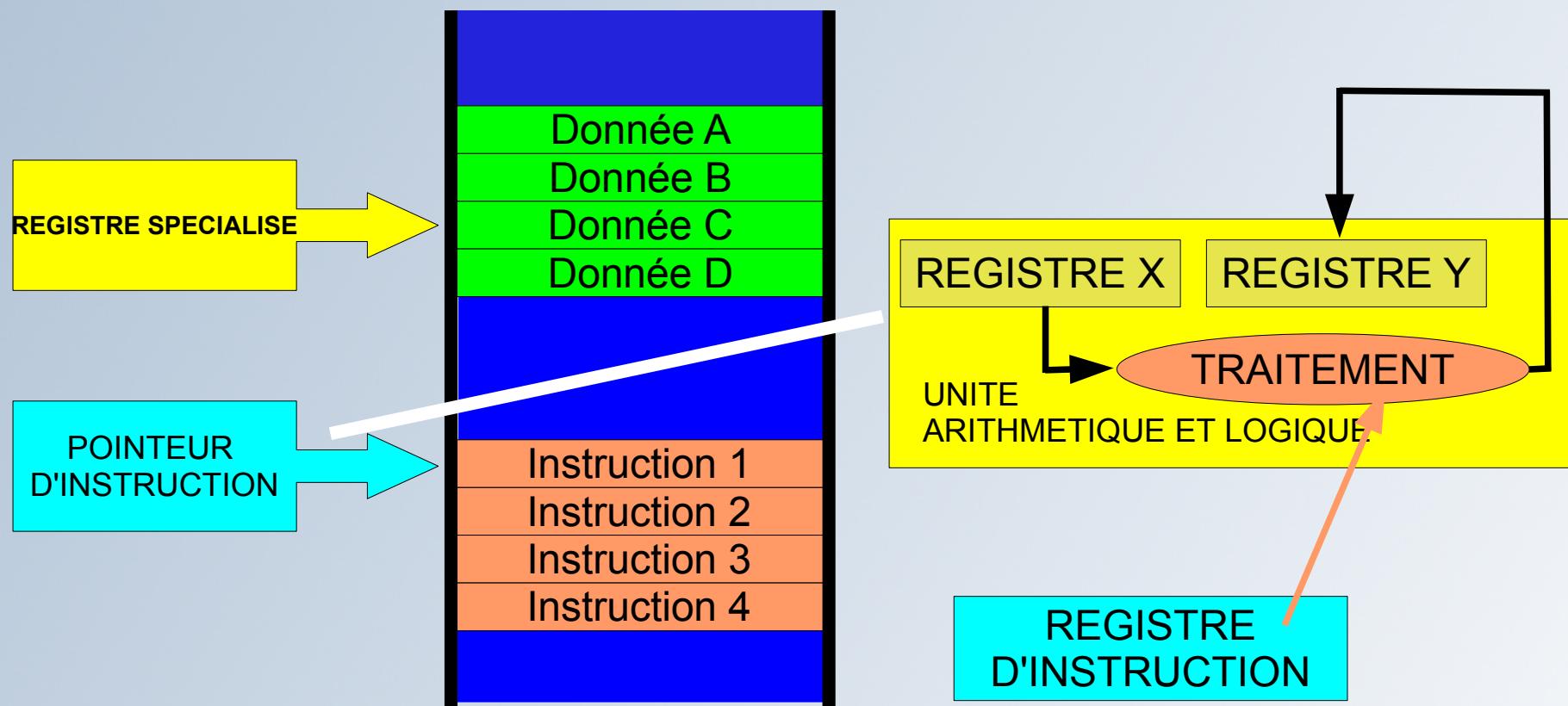


La variable lue est chargée dans un **registre de travail** de l'Unité Centrale, où elle sera traitée.

L'Unité Centrale dispose de plusieurs registres de travail (R0 à R4).

Les registres de travail font partie du Bloc de Calcul de l'Unité Centrale.

### 3. Structure programmable : U.A.L.

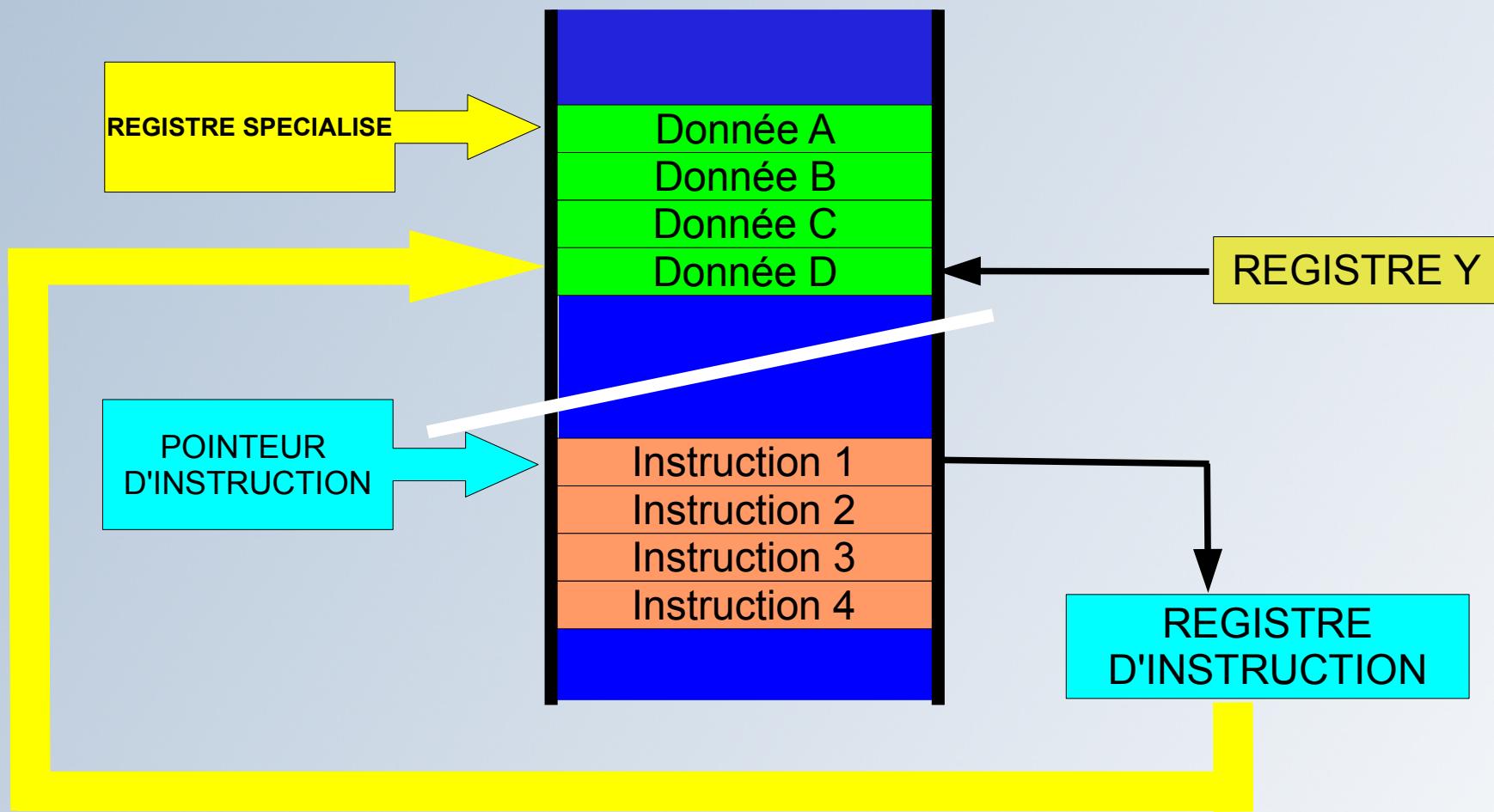


Le contenu du registre d'instruction détermine le type de traitement que doit subir la variable contenue dans le registre de travail.

L'**Unité Arithmétique et Logique (UAL)** effectue le traitement demandé par le Registre Instruction. L'UAL fait partie du Bloc de Calcul de l'Unité Centrale.

A l'issue du traitement le résultat peut être chargé dans un **registre de travail différent** ou peut **remplacer la variable initiale**.

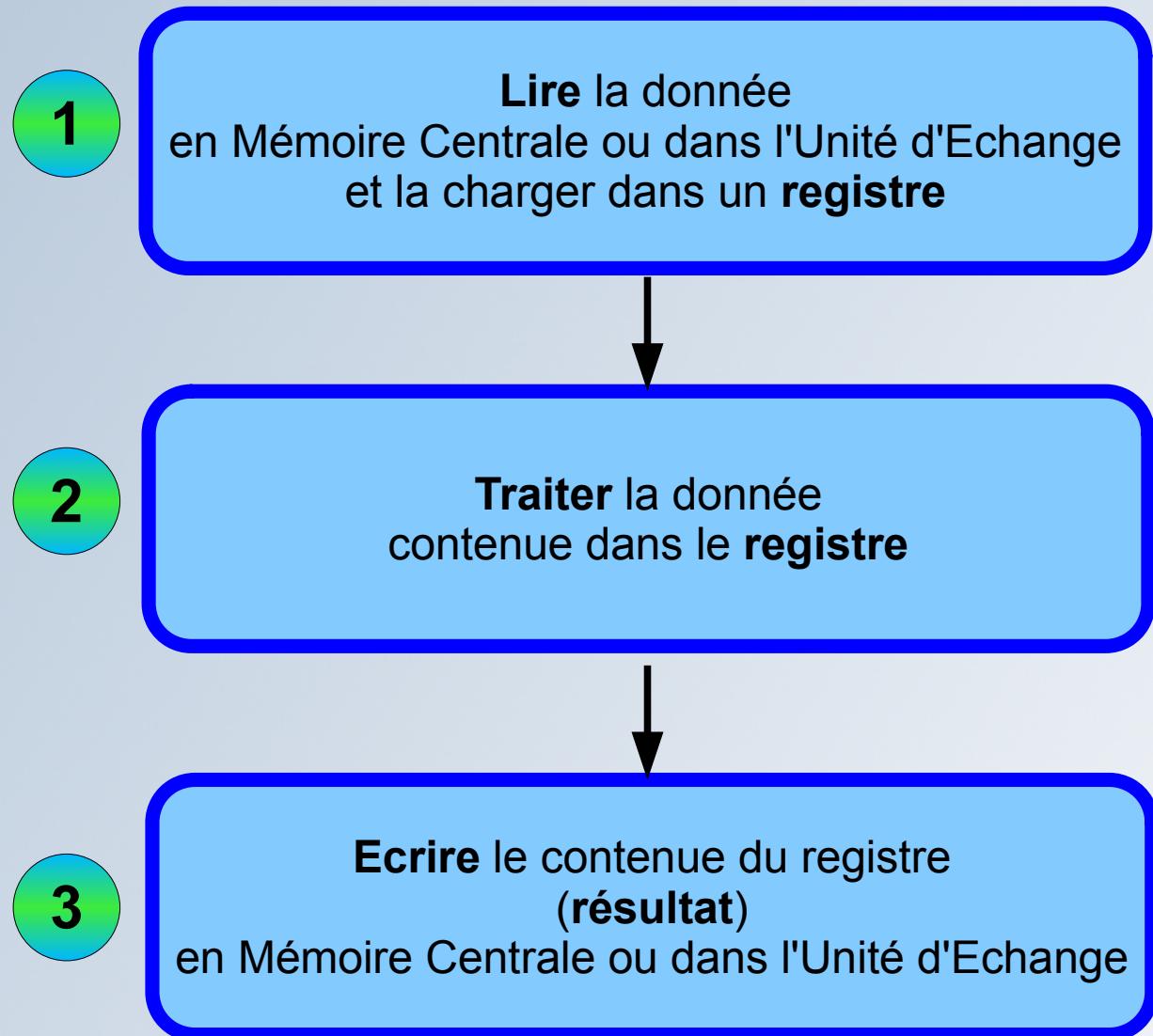
### 3. Structure programmable : écriture d'une donnée en mémoire



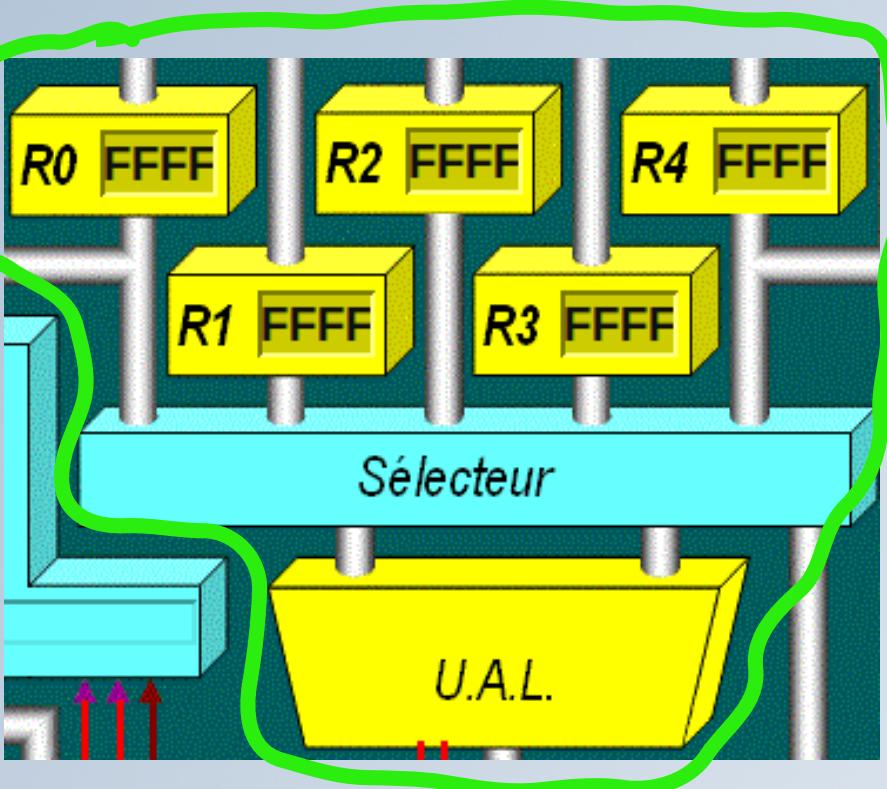
Une instruction peut écrire le contenu d'un registre de travail dans une variable en mémoire. Pour cela, un **Registre** fournit l'adresse de la variable à traiter. Ce registre peut être :

- le registre d'instruction (**mode direct**)
- un registre spécialisé (**mode par registre**)

### 3. Structure programmable : séquence typique de traitement d'une donnée



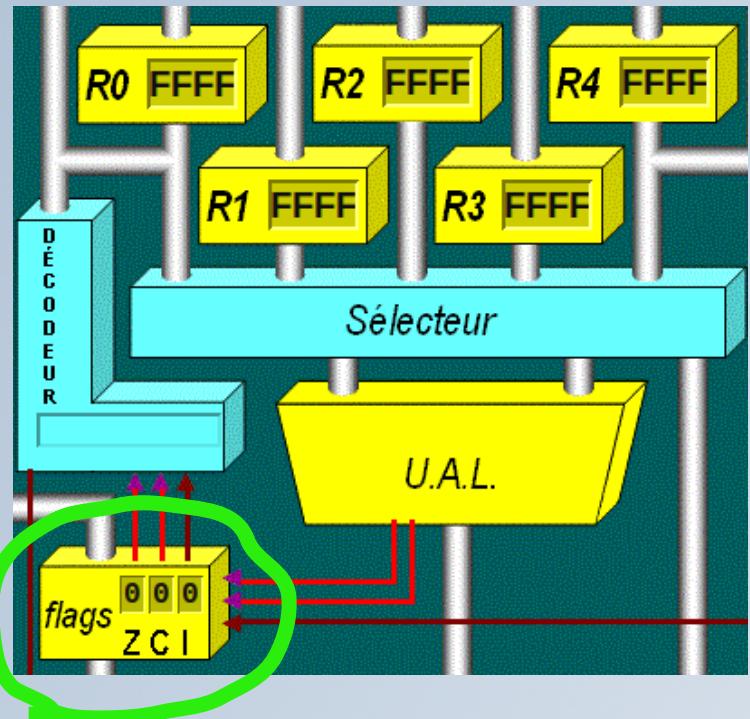
# 4. Structure matérielle : bloc de calcul



Le Bloc de calcul est composé :

- d'une **Unité Arithmétique et Logique (U.A.L.)** circuit combinatoire constitué d'additionneurs, capable d'effectuer entre deux valeurs binaires :
  - des opérations arithmétiques (additions, soustractions)
  - des fonctions logiques bit à bit (ET, OU, ...)
- de registres de travail (**R0, R1, R2, R3, R4**) détenant :
  - avant l'opération, une des deux valeurs d'entrée de l'U.A.L.
  - après l'opération, le résultat de l'opération
- un sélecteur permet le choix d'un ou deux registres d'entrée de l'U.A.L.

# 4. Structure matérielle : Indicateurs



Le Bloc de Calcul comporte également un registre d'**indicateurs (flags)**. Un indicateur est un bit de ce registre, exemples d'indicateurs : **Z** (Zéro) et **C** (Carry, retenue).

Un indicateur est mis à jour (1 ou 0) par l'U.A.L. à l'issue d'une opération arithmétique ou logique.

Un indicateur donne une "indication" sur le résultat de l'opération effectuée :

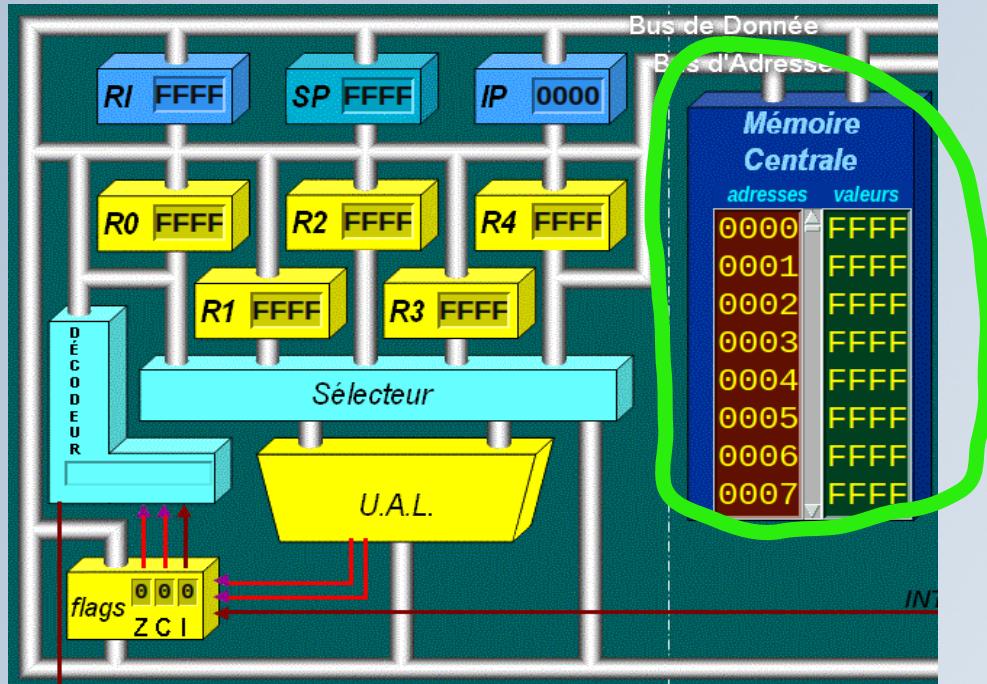
- si le résultat vaut ZERO, Z est mis à 1
- si le résultat est différent de ZERO, Z est mis à 0
- si l'opération a généré une retenue (CARRY), C est mis à 1
- si l'opération n'a pas généré de retenue (CARRY), C est mis à 0

Remarques : Un indicateur est significatif du résultat de la dernière opération effectuée (la plus récente).

Seules les instructions arithmétiques ou logiques affectent les indicateurs.

En plus de Z et C, d'autres indicateurs peuvent exister : Interruption, Parité, Débordement, Signe...

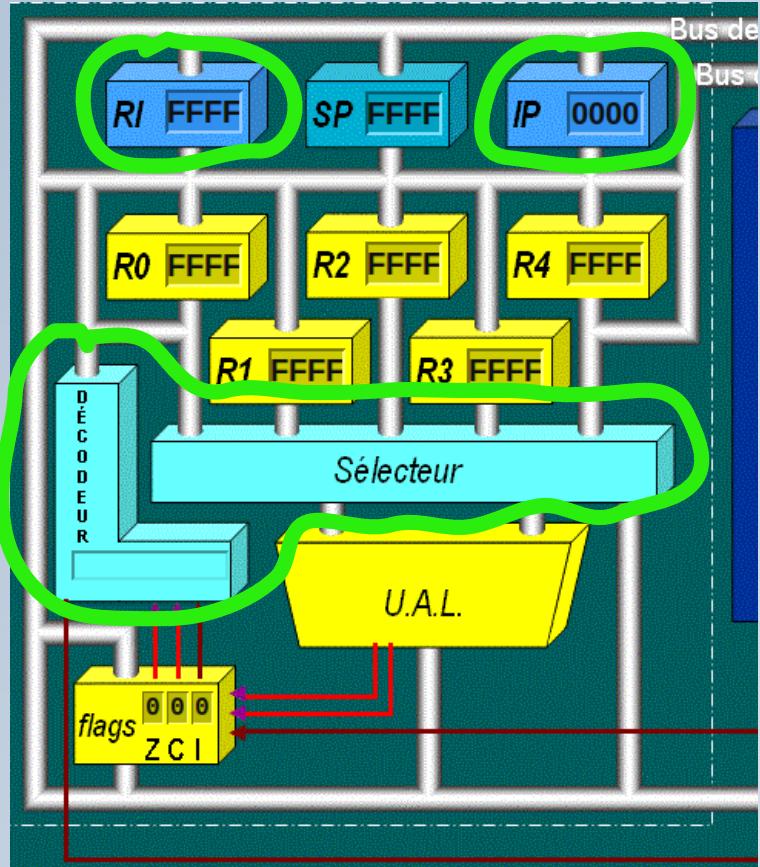
# 4. Structure matérielle : Mémoire Centrale



Deux **bus** relient la Mémoire Centrale à l'Unité Centrale :

- Bus de Donnée
- Bus d'Adresse

# 4. Structure matérielle : Bloc de Commande



Le Bloc de Commande est composé :

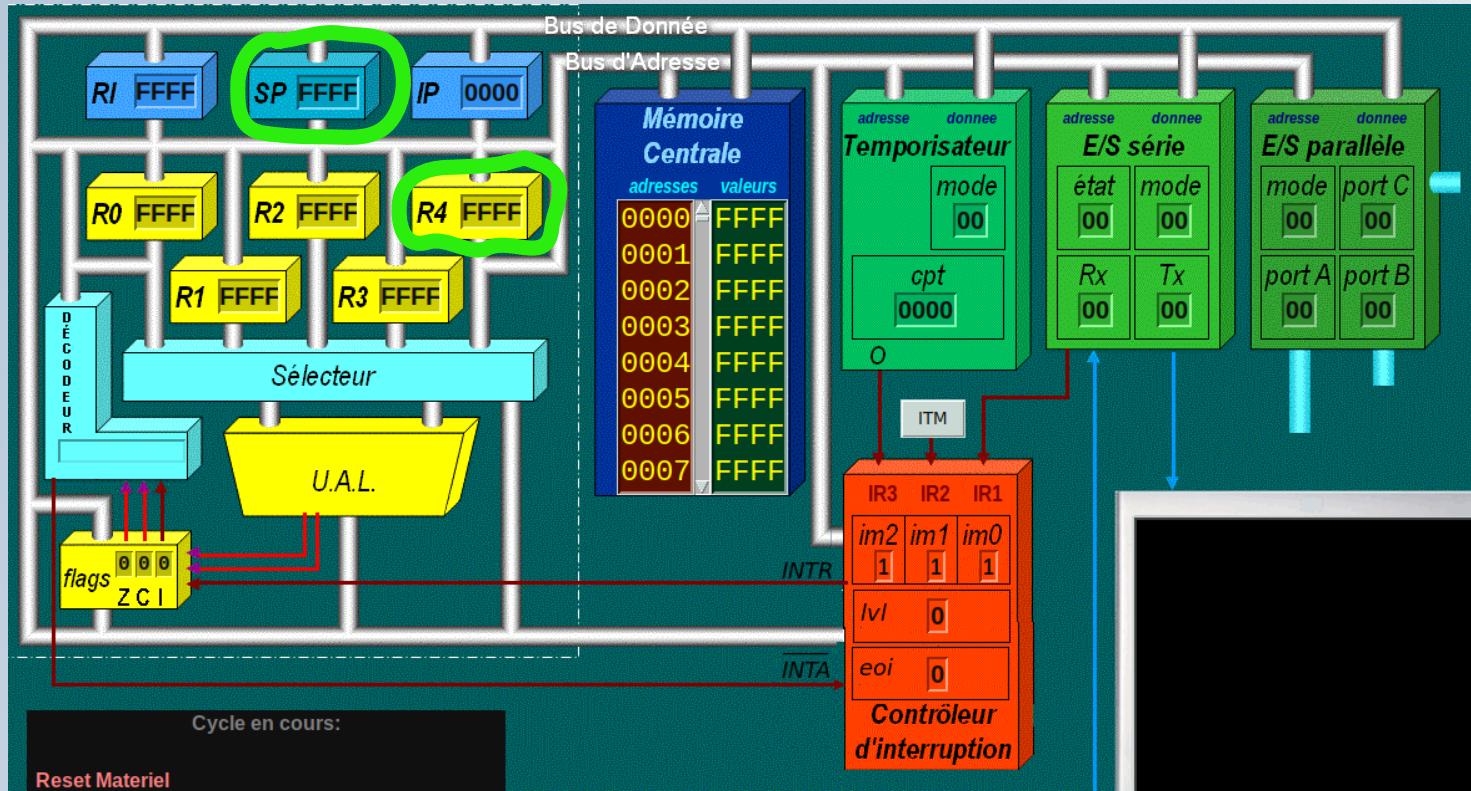
- du **Registre Instruction (RI)** qui reçoit l'instruction lue en Mémoire Centrale à l'adresse fournie par le registre **IP (Pointeur d'Instruction)**.
- d'un **Séquenceur**, activé par le Décodeur, qui génère les séquences de micro-commandes nécessaires à l'exécution de l'instruction décodée.

Exemples de micro-commandes :

- lecture en mémoire centrale
- écriture en mémoire centrale
- sélection de fonction de l'Unité Arithmétique et Logique
- selection de registre
- chargement de registre

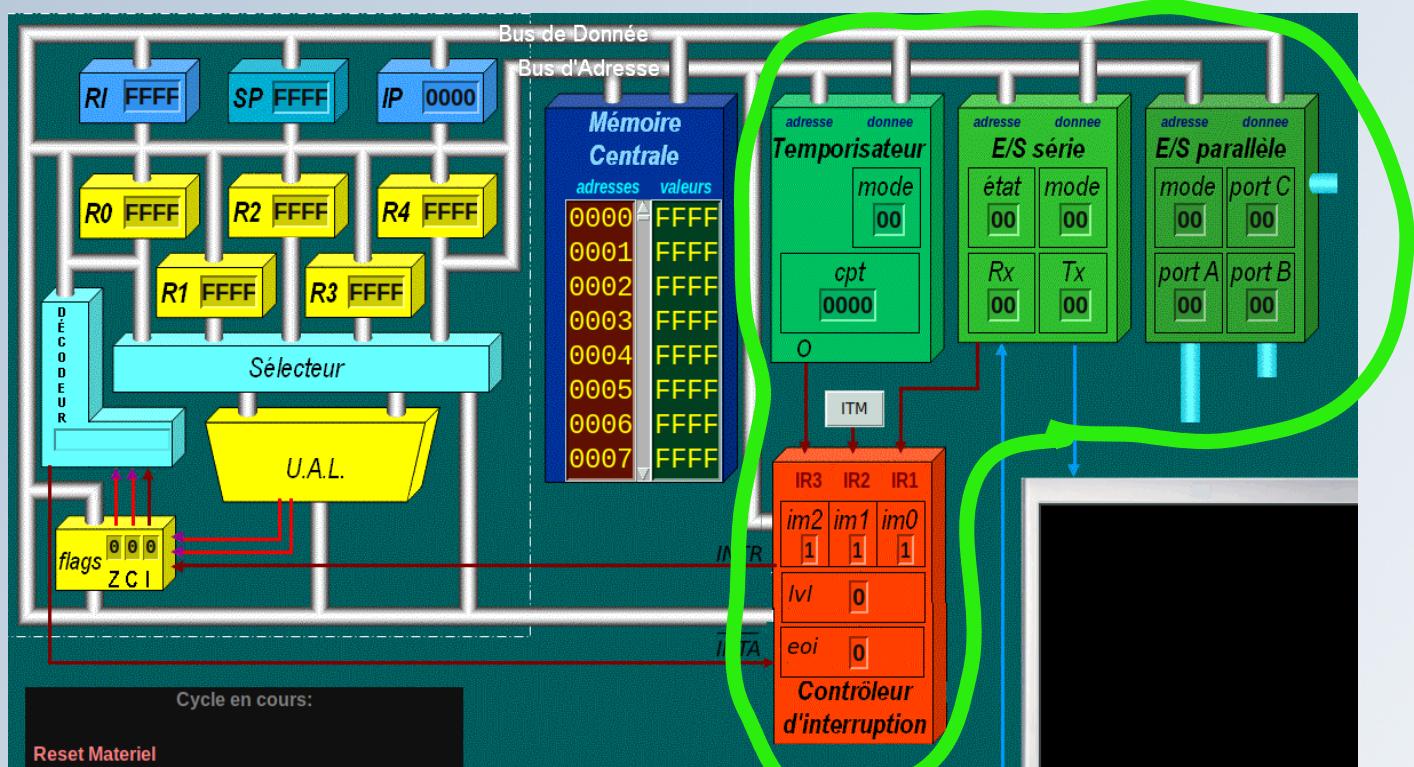
Remarque : Le Séquenceur prend en compte l'état des indicateurs lors de l'exécution

# 4. Structure matérielle : Registres Spécialisés



- Un registre spécialisé complète le calculateur : le **Pointeur de pile (SP)** (Stack Pointer) utilisé pour l'adressage de la zone mémoire appelée **PILE**
- Le registre de travail **R4** est également un registre spécialisé pour le mode d'adressage par registre où il est utilisé comme registre d'adresse.

# 4. Structure matérielle : Unité d'Echange



Elle est composée de circuits d'Entrée-Sortie spécialisés, occupant une partie de l'espace adressable. Exemples :

- Entrée-Sortie parallèle
- Entrée-Sortie série
- Temporisateur
- Contrôleur d'interruption

# 5. Représentation de l'information : rappels de numération binaire

En binaire (base 2) les chiffres sont appelés BIT (contraction de Binary digit)

Ils peuvent prendre 2 valeurs : 0 et 1

Les bits qui composent un nombre sont ordonnés, exemple: **1 1 0 0 1 0 1 1**

Poids fort      Poids faible

Conversion binaire => décimal :

$$\begin{aligned}110101 &= 1 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 \\&= 32 + 16 + 0 + 4 + 0 + 1 \\&= 53\end{aligned}$$

Dans les calculateurs on utilise un format "standard" de 8 bits et des formats multiples de 8.

8 bits = OCTET (BYTE)

16 bits = MOT (WORD)

32 bits = DOUBLE-MOT (DWORD)

64 bits = QUADRUPLE-MOT

L'octet est l'unité d'information dans les calculateurs.

En binaire sont également définis le Kilo-octets, le Méga-octets, le Giga-octets.

KILO =  $2^{10} = 1024$

MEGA =  $2^{20} = 1024 \times 1024 = 1\ 048\ 576$

GIGA =  $2^{30} = 1024 \times 1024 \times 1024 = 1\ 073\ 741\ 824$

# 5. Représentation de l'information : notation hexadécimale

Binaire (base 2)	Décimal (base 10)	Hexadécimal (base 16)
0 0 0 0	0	0
0 0 0 1	1	1
0 0 1 0	2	2
0 0 1 1	3	3
0 1 0 0	4	4
0 1 0 1	5	5
0 1 1 0	6	6
0 1 1 1	7	7
1 0 0 0	8	8
1 0 0 1	9	9
1 0 1 0	10	A
1 0 1 1	11	B
1 1 0 0	12	C
1 1 0 1	13	D
1 1 1 0	14	E
1 1 1 1	15	F

En hexadécimal, la représentation des nombres nécessite 16 symboles :

0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

Représentation d'un entier naturel N sur un octet :

- en binaire : **0000 00 00 ≤ N ≤ 1111 1111**
- en décimal : **0 ≤ N ≤ 255**
- en hexadécimal : **00 ≤ N ≤ FF**

# 5. Représentation de l'information : entiers naturels et retenue

On considère les nombres dans un format déterminé et fixe.

Dans ce format, l'addition de deux nombres peut générer une retenue C (Carry).

Exemple: additions d'entiers naturels dans un format de 8 bits ( N compris entre 0 et 255 en décimal)

$$\begin{array}{r} 10111100 \quad (188) \\ + 01110111 \quad (119) \\ \hline 100110011 \quad (51) \end{array}$$

↓  
résultat faux sur 8 bits  
retenue (C)

$$\begin{array}{r} 00101100 \quad (44) \\ + 10110111 \quad (183) \\ \hline 011100011 \quad (227) \end{array}$$

↓  
résultat exact  
pas de retenue (NC)

Remarque:

La retenue peut être prise en compte pour exprimer le résultat dans un format plus grand que le format du bloc de calcul.

# 5. Représentation de l'information : nombres entiers relatifs

Binaire	Décimal
0 1 1 1	+7
0 1 1 0	+6
0 1 0 1	+5
0 1 0 0	+4
0 0 1 1	+3
0 0 1 0	+2
0 0 0 1	+1
0 0 0 0	0
1 1 1 1	-1
1 1 1 0	-2
1 1 0 1	-3
1 1 0 0	-4
1 0 1 1	-5
1 0 1 0	-6
1 0 0 1	-7
1 0 0 0	-8

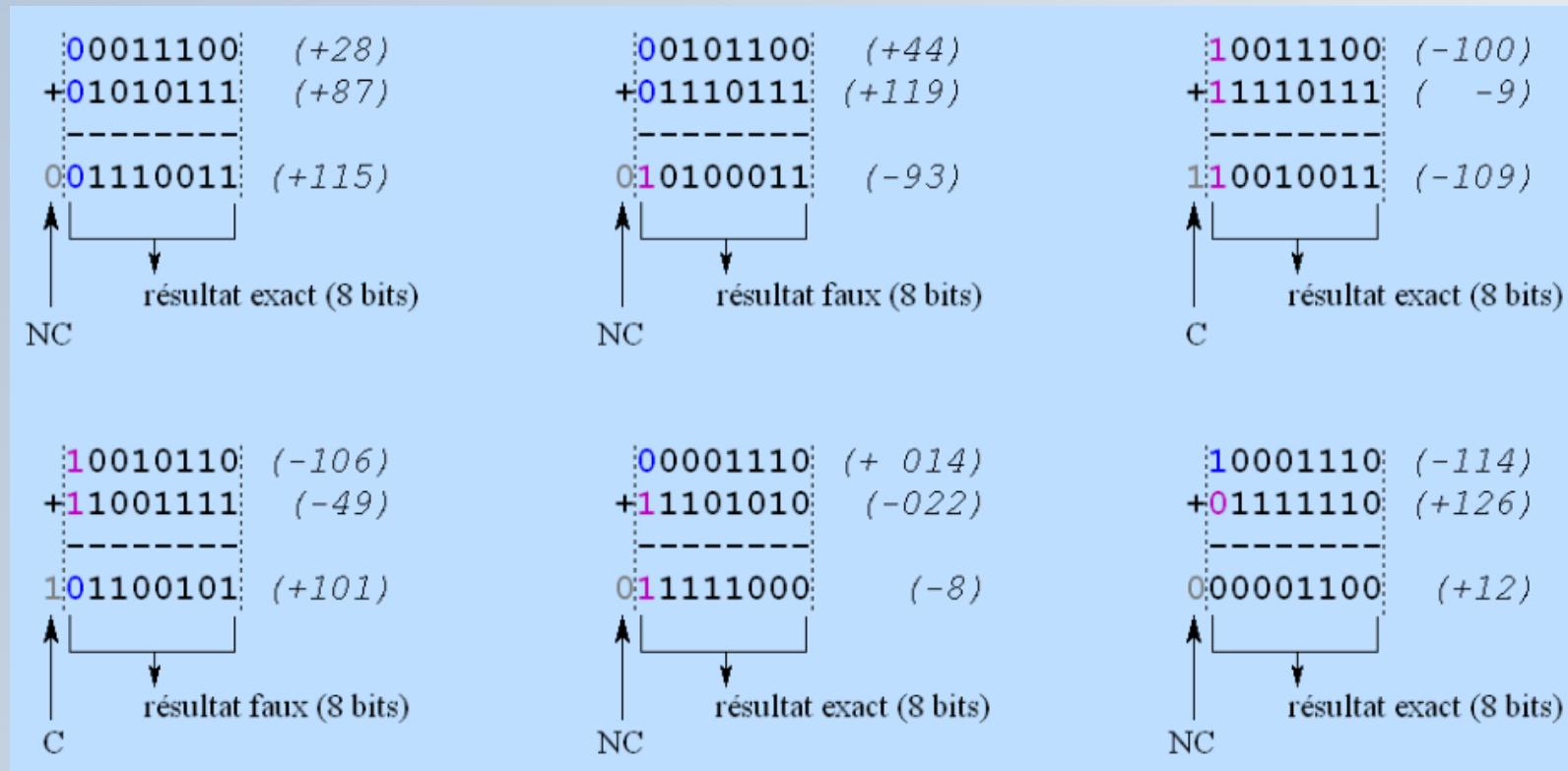
On considère les nombres dans un format déterminé et fixe.

Dans ce format, par convention, les entiers positifs commencent par 0 (bit de poids fort). Pour obtenir un entier négatif on utilise le complément vrai (complément à 2) du positif. Exemple: entiers relatifs dans un format de 4 bits

Remarque:

L'addition de deux entiers relatifs de même signe peut donner un résultat faux dans le format considéré (résultat de signe opposé), on parle alors de débordement (overflow)

# 5. Représentation de l'information : entiers relatifs et débordement



Remarque: L'addition de deux nombres de signe opposé est toujours exacte.

# 5. Représentation de l'information : Code ASCII

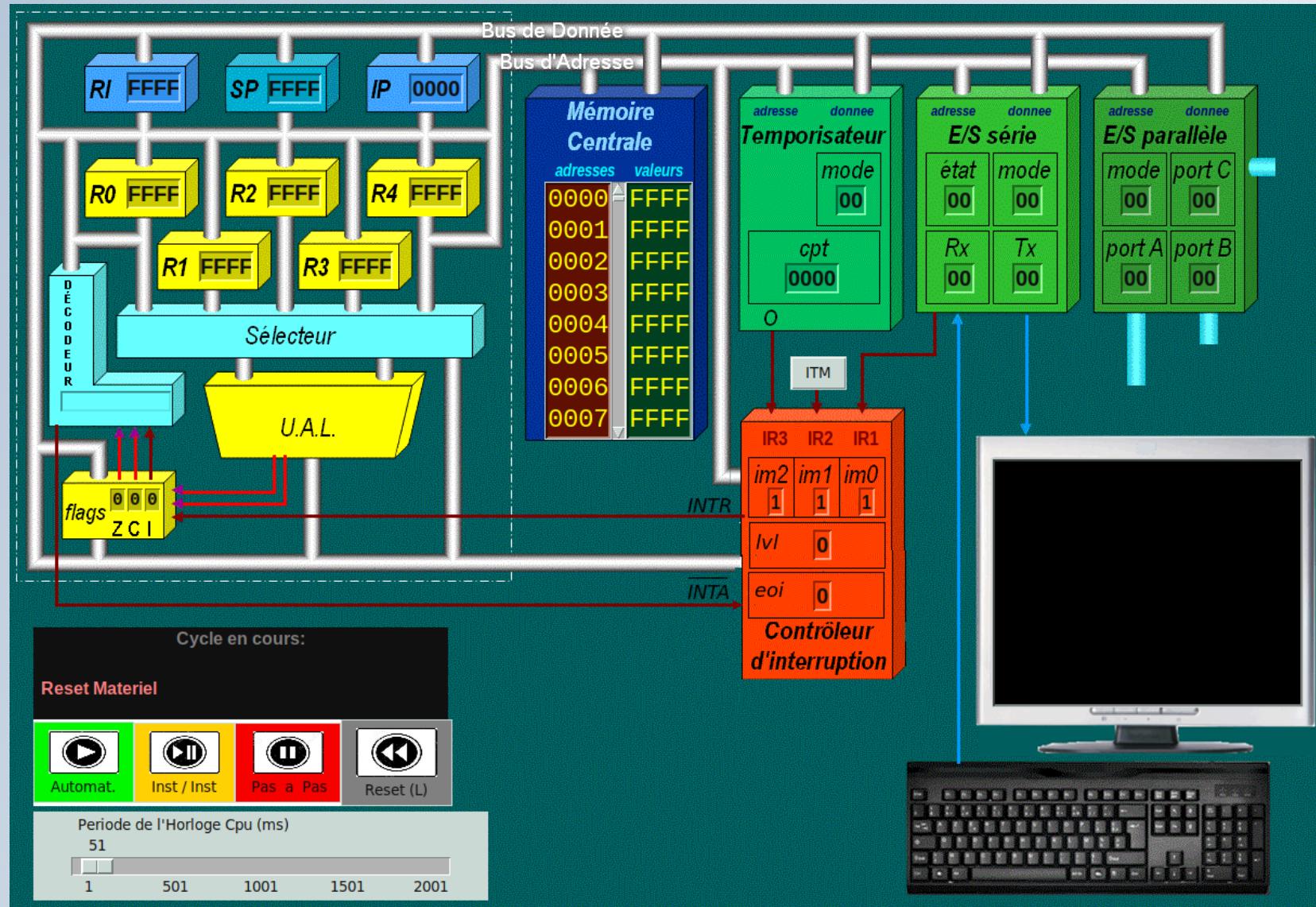
Le code ASCII (American Standard Code for Information Interchange) est un codage normalisé des caractères alphanumériques et signes de ponctuation. Ces codes sont utilisés pour la communication de «texte» entre périphériques et calculateur, ainsi que pour le stockage de «texte» en mémoire.

Code ASCII :

	0	1	2	3	4	5	6	7
0	<i>NUL</i>	<i>DLE</i>	<i>SPACE</i>	0	@	P	`	p
1	<i>SOH</i>	<i>DC1</i>	!	1	A	Q	a	q
2	<i>STX</i>	<i>DC2</i>	"	2	B	R	b	r
3	<i>ETX</i>	<i>DC3</i>	#	3	C	S	c	s
4	<i>EOT</i>	<i>DC4</i>	\$	4	D	T	d	t
5	<i>ENQ</i>	<i>NAK</i>	%	5	E	U	e	u
6	<i>ACK</i>	<i>SYN</i>	&	6	F	V	f	v
7	<i>BEL</i>	<i>ETB</i>	'	7	G	W	g	w
8	<i>BS</i>	<i>CAN</i>	(	8	H	X	h	x
9	<i>HT/TAB</i>	<i>EM</i>	)	9	I	Y	i	y
A	<i>LF</i>	<i>SUB</i>	*	:	J	Z	j	z
B	<i>VT</i>	<i>ESC</i>	+	;	K	[	k	{
C	<i>FF</i>	<i>FS</i>	,	<	L	\	l	
D	<i>CR</i>	<i>GS</i>	-	=	M	]	m	}
E	<i>SO</i>	<i>RS/HOME</i>	.	>	N	^	n	~
F	<i>SI</i>	<i>US</i>	/	?	O	-	o	<i>DEL</i>

Exemple : la lettre **N** est à l'intersection de la colonne **4** et ligne **E**, son code est **4E** (hexadécimal),

# 6. Présentation de SIMCAL (SIMulateur de CALculateur : structure et IHM)

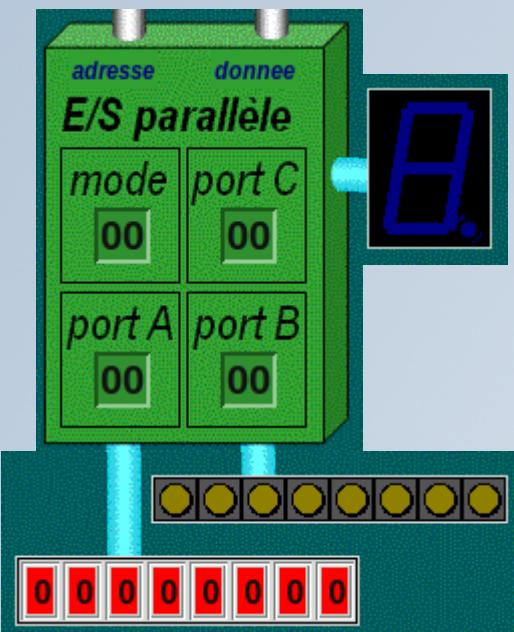


# 6. Présentation de SIMCAL (SIMulateur de CALculateur) : structure et IHM

SIMCAL est un simulateur de calculateur écrit en Python pour ordinateur, compatible PC, constitué :

- D'une Unité Centrale 16 bits :
  - Données sur 16 bits,
  - Adresses sur 16 bits.
- D'une Unité d'Echange 8 bits constituée :
  - D'un circuit d'entrée/sortie série (entrée = clavier réel du PC, sortie écran simulé),
  - D'un circuit d'E/S parallèles, 3 ports (entrée = interrupteurs simulés, sortie = leds ou afficheur simulés).
- D'un temporisateur
- D'une contrôleur d'interruption à 3 entrées :
  - appui sur un bouton poussoir simulé (ITM),
  - Réception d'un code ASCII, du clavier simulé,
  - Fin de cycle du temporisateur
- D'un jeu d'instructions : 30 instructions de type RISC à taille fixe de 16 bits

# 6. Présentation de SIMCAL (SIMulateur de CALculateur) : E/S parallèles



Le circuit d'Entée-Sortie parallèle est constitué de 4 registres :

- **Port A** : registre 8 bits, à configurer en entrée ou sortie, adresse FF0
- **Port B** : registre 8 bits, à configurer en entrée ou sortie, adresse FF1
- **Port C** : registre 8 bits, à configurer en entrée ou sortie, adresse FF2
- **Mode** : registre 8 bits, pour configuration des ports, adresse FF3

Modes:

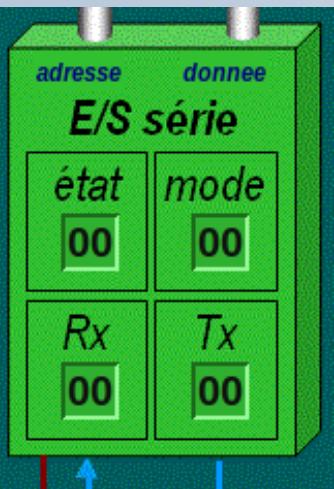
mode	port A	port B	port C
0	entrée	entrée	entrée
1	entrée	entrée	sortie
2	entrée	sortie	entrée
3	entrée	sortie	sortie
4	sortie	entrée	entrée
5	sortie	entrée	sortie
6	sortie	sortie	entrée
7	sortie	sortie	sortie

# 6. Présentation de SIMCAL (SIMulateur de CALculateur) : E/S série

Le circuit d'Entrée-Sortie série est constitué de 4 registres :

- **Rx** : registre Récepteur 8 bits, reçoit le code ASCII (octet) provenant du clavier du PC, adresse FF4
  - **Tx** : registre Transmetteur 8 bits, envoie son contenu à l'écran simulé, adresse FF4
  - **Mode** : registre 8 bits, pour configurer le circuit, adresse FF5
  - **Etat** : registre 8 bits, indicateur d'état du circuit, adresse FF5
- Le circuit peut, dans certains modes, générer une demande d'**interruption** à la **réception** d'un octet. Modes:

mode	Interruption	Transmission	Réception
0	non	non	non
1	non	non	oui
2	non	oui	non
3	non	oui	oui
5	oui	non	oui
7	oui	oui	oui



Dans le registre d'état (8 bits), les 3 bits de poids faibles sont significatifs :

0 0 0 0 0 ER TV RP

RP = 1 = Récepteur plein = nouvelle donnée reçue à lire

RP = 0 = Récepteur vide = précédente donnée reçue lue

TV = 1 = Transmetteur vide = transmission précédente terminé, transmetteur disponible

TV = 0 = Transmetteur occupé = transmission en cours, transmetteur non disponible

ER = 1 = Erreur de réception = écrasement dans le récepteur d'une donnée non lue par une nouvelle donnée

ER = 0 = Pas d'erreur de réception

**NB :** La lecture du registre d'état remet à zéro le bit ER, la lecture du récepteur remet à zéro le bit RP

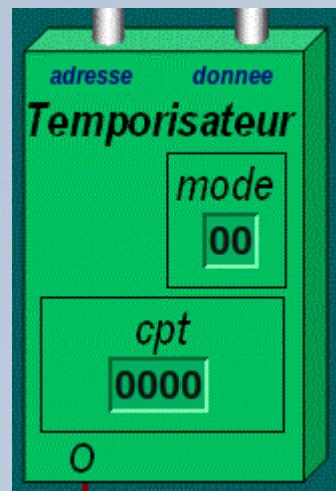
# 6. Présentation de SIMCAL (SIMulateur de CALculateur) : temporisateur

Le circuit Temporisateur est constitué de 2 registres:

- **Mode**: registre 8 bits, pour configuration du décompteur, adresse FF6
- **cpt**: décompteur 16 bits, valeur initiale de décomptage, adresse FF7

Modes:

mode	Décomptage
0	non, arrêt
1	oui, apériodique
2	oui, périodique



- En mode apériodique, le circuit effectue un cycle de décomptage à partir de la valeur N, valeur initiale chargée dans le compteur, jusqu'à 0. En fin de cycle (arrivé à 0) le circuit génère une demande d'interruption.
- En mode périodique, le circuit répète sans fin des cycles de décomptage à partir de la valeur N, valeur initiale chargée dans le compteur jusqu'à 0. Arrivé à 0 le circuit génère une demande d'interruption et recommence un cycle à partir de N

NB: Le décomptage commence dès le chargement de la valeur N initiale  
La fréquence de l'horloge du décompteur est d'environ 5 Hz (selon PC).

# 6. Présentation de SIMCAL (SIMulateur de CALculateur) : contrôleur d'interruptions

Le circuit Contrôleur d'Interruption est constitué de 2 registres:

- **Masque** : registre 8 bits, pour masquer ou démasquer (interdire / autoriser) les entrées de demande d'interruption, adresse FF8
- **FIN (EOI)** : registre "Fin d'INterruption" (End Of Interrupt), pour gerer l'imbrication (éventuelle) de plusieurs interruptions, adresse FF9.

Dans le registre de masque, les 3 bits de poids faibles sont significatifs :

- $imN = 0$  = Entrée démasquée = demande propagée
- $imN = 1$  = Entrée masquée = demande bloquée

0	0	0	0	0	im2	im1	im0
---	---	---	---	---	-----	-----	-----

Le circuit dispose de 3 entrées (IR3, IR2, IR1) permettant le traitement de 3 sources d'interruptions.

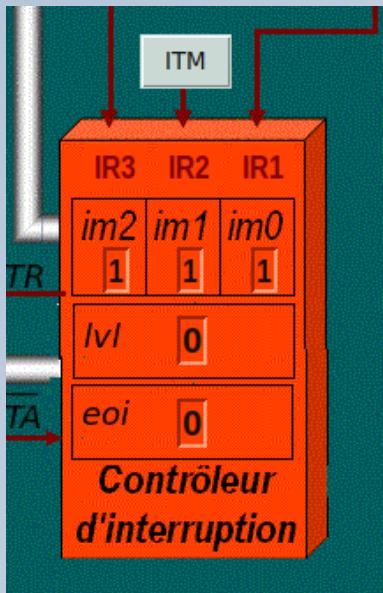
Un niveau de priorité est affecté à chaque entrée, l'entrée IR3 est la plus prioritaire :

- IR3 est activée par le temporisateur, en fin de décomptage (passage à 0)
- IR2 est activée par le bouton poussoir ITM (bouton simulé, à cliquer)
- IR1 est activée par la réception d'une donnée sur le circuit d'E-S série, si le mode de ce dernier le permet

Gestion des priorités : en cas de demandes simultanées, la plus prioritaire est prise en compte, les autres sont mises en attente de la fin de celle en cours.

L'écriture de la valeur 0 dans le registre FIN indique au contrôleur que l'interruption en cours est terminée, libérant la(les) demande(s) en attente.

Tant que le registre FIN n'a pas reçu 0, toute demande de priorité inférieure ou égale à celle en cours reste en attente.



# 7. Présentation de SIMCAL (SIMulateur de CALculateur) : mémoire centrale et Entrées/Sorties

Les Entrées-Sorties sont "mappées" en mémoire. Elles occupent donc une zone d'adresses prise sur l'espace mémoire.

## Mémoire Centrale

SIMCAL dispose d'une mémoire vive de 512 (1024, 2048 ou 4080) mots de 16 bits (taille mémoire configurable).

Cet espace mémoire occupe les adresses 0 à 1FF (3FF, 7FF, FEF).

Quelle que soit la taille de la mémoire centrale, l'adressage de l'espace mémoire utilise un format de 16 bits.

## Espace d'entrée-sortie

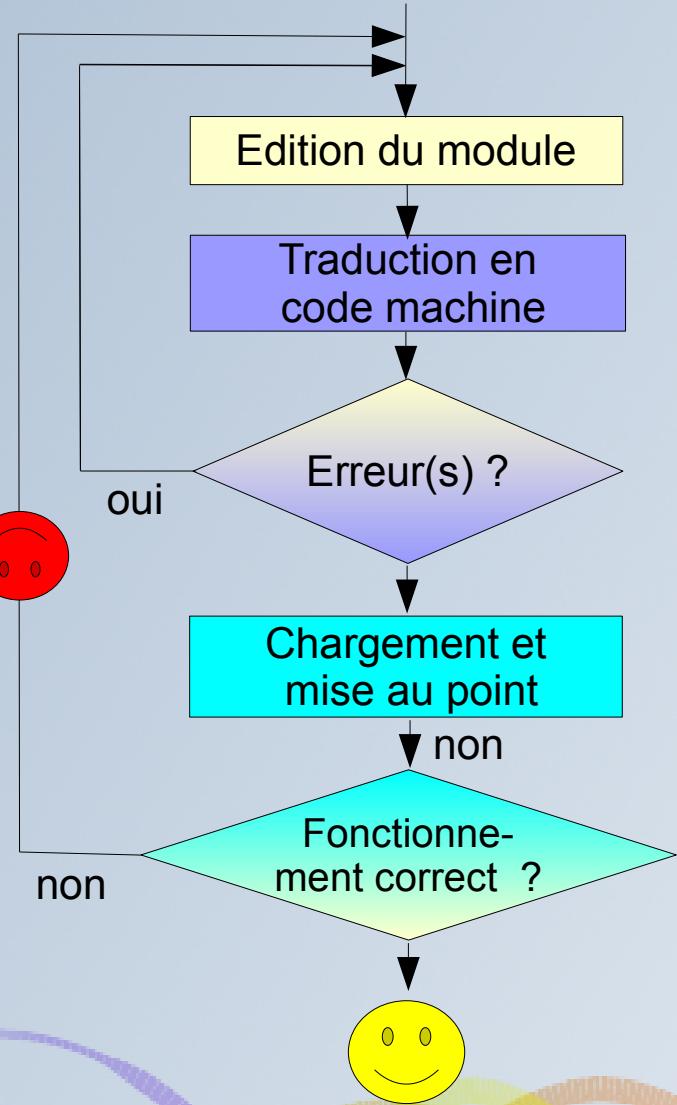
L'espace d'entrée-sortie de SIMCAL occupe les adresses FF0 à FF9 (adresses au format 16 bits)

NB: Les registres 8 bits des circuits d'entrée-sortie reçoivent (ou fournissent) les 8 bits de poids faibles du bus de données 16 bits.

Répartition des E/S :

adresse	registre	circuit
FF0	port A	E-S parallèle
FF1	port B	
FF2	port C	
FF3	mode	
FF4	Rx / Tx	E-S série
FF5	état / mode	
FF6	mode	Temporisateur
FF7	décompteur	
FF8	masque	Contrôleur
FF9	FIN (EOI)	d'Interruption

# 6. Etapes de développement d'un programme : fichiers produits



Etape	Outil logiciel	Fichiers produits	Extension des noms de fichiers
Edition du module	Editeur	Source	.asm
Traduction en code machine	Assembleur AsmCal	Objet et listing	.sim et .lst
Chargement, essai et mise au point	Simulateur Simcal		

# 8. Langage d'assemblage de SIMCAL : syntaxe

SIMCAL est associé à un assembleur : **AsmCAL** (AsmCAL est insensible à la casse). Un programme source est constitué de lignes d'instructions et de lignes de directives. Une ligne de programme source peut également être constituée d'un commentaire uniquement.

Les nombres peuvent être exprimés en décimal ou en hexadécimal à l'aide du préfixe **0x**. Une ligne d'instruction ne peut comporter qu'une instruction et doit être formatée comme suit : **ETIQUETTE: Séparateur1 INSTRUCTION Séparateur2 ;COMMENTAIRE**

Champ	Description
ETIQUETTE	Facultative. Commence par une lettre et se termine par « : » (sans espace)
Séparateur1	Espace(s) ou tabulation(s) *. Facultatif s'il n'y a pas d'étiquette
INSTRUCTION	Code Opération et Opérande séparés pas espace(s) ou tabulation(s)
Séparateur2	Espace(s) ou tabulation(s) *
COMMENTAIRES	Facultatif. Doit commencer par ' ;'

\* Pour une mise en page optimale du listing, utiliser de préférence la tabulation comme séparateur

# 8. Langage d'assemblage de SIMCAL : instructions

## Instructions de transfert

- valeur => registre
- registre => registre
- registre => mémoire
- mémoire => registre
- registre => entrée-sortie
- entrée-sortie => registre

## Instructions arithmétiques et logiques

- ET logique
- Addition
- Soustraction
- Incrémentation
- Décrémentation
- Décalages
- Rotations
- Complément

## Instructions de branchement

- Inconditionnelles
- Conditionnelles
- Interruptions logicielles

## Instruction de contrôle

- Arrêt

# 8. Langage d'assemblage de SIMCAL : directives

Quatre directives : EQU, ORG, VAR, CST

**EQU** Directive d'équivalence. Etablit une équivalence entre un symbole et une valeur.

**ORG** Directive d'origine. Précise l'adresse mémoire d'implantation des codes qui suivent. Facultatif, par défaut adresse 0.

**VAR** Directive de définition de variable **16 bits**. Réserve un ou plusieurs mots de mémoire et y associe un nom de variable.

Exemple:

*Réultat: VAR ? ; un mot nommé Réultat*

*Notes: VAR 10? ; tableau de 10 mots nommé Notes*

**CST** Directive de définition de constante **16 bits**. Affecte un ou plusieurs mots de mémoire et y associe un nom de constante

Exemple:

*Toto: CST 0X20 ; le mot Toto de valeur 20 hexa.*

*Titi: CST 10,20,0X30 ; le tableau Titi de 3 mots*

*Text: CST 'blaBLA' ; le tableau Text de 6 codes ascii étendu 16 bits*

# 8. Langage d'assemblage de SIMCAL : instructions de transfert

LDR R0,N	Immédiat	$R0 \leftarrow N$	<b>R0 uniquement</b>
LDR Rd,[N]	Direct	$Rd \leftarrow [N]$	mot à l'adresse N $0 \leq d \leq 4$
STR [N],Rs		$[N] \leftarrow Rs$	mot à l'adresse N $0 \leq s \leq 4$
LDR Rd,Rs LDR SP,Rs LDR FL,Rs	Registre	$Rd \leftarrow Rs$	$0 \leq d \leq 4 \quad 0 \leq s \leq 4$
LDR Rd,[R4]	Indirect par registre	$Rd \leftarrow [R4]$	Mot à l'adresse fournie par R4 (uniquement) $0 \leq d \leq 4$
STR [R4],Rs	Indirect par registre	$[R4] \leftarrow Rs$	Mot à l'adresse fournie par R4 (uniquement) $0 \leq d \leq 3$
POP Rd POP FL	Registre	$Rd \leftarrow [SP]$ $SP \leftarrow SP+1$ $FL \leftarrow [SP]$ $SP \leftarrow SP+1$	Mot à l'adresse fournie par SP $0 \leq d \leq 4$
PUSH Rs PUSH FL		$SP \leftarrow SP-1$ $[SP] \leftarrow Rs$ $SP \leftarrow SP-1$ $[SP] \leftarrow FL$	Mot à l'adresse fournie par SP $0 \leq s \leq 4$

# 8. Langage d'assemblage de SIMCAL : instructions arithmétiques et logiques

Mnémonique	adressage	résultat	notes
ADD Rd,RxRy	Registre	Rd $\leftarrow$ Rx + Ry	$0 \leq d \leq 4$ $0 \leq x \leq 4$ $0 \leq y \leq 4$ $x \neq y$
SUB Rd,RxRy		Rd $\leftarrow$ Rx - Ry	$0 \leq d \leq 4$ $0 \leq x \leq 4$ $0 \leq y \leq 4$ $x \neq y$
INC Rx		Rd $\leftarrow$ Rx + 1	$0 \leq x \leq 4$
DEC Rx		Rd $\leftarrow$ Rx - 1	$0 \leq x \leq 4$
AND Rd,RxRy	Registre	Rd $\leftarrow$ Rx ET Ry	$0 \leq d \leq 4$ $0 \leq x \leq 4$ $0 \leq y \leq 4$ $x \neq y$
OR Rd,RxRy		Rd $\leftarrow$ Rx OU Ry	$0 \leq d \leq 4$ $0 \leq x \leq 4$ $0 \leq y \leq 4$ $x \neq y$
NOT Rx		Rx $\leftarrow$ $\sim$ Rx	complément à 1 $0 \leq x \leq 4$

# 8. Langage d'assemblage de SIMCAL : décalages

Mnémonique	adressage	résultat	notes
SHL Rx,N	Registre	$Rx \leftarrow Rx \ll N$	Décalage à gauche de N bits, introduction de 0 $0 \leq x \leq 4 \quad 0 \leq N \leq 15$
SHR Rx,N		$Rx \leftarrow Rx \gg N$	Décalage à droite de N bits, introduction de 0 $0 \leq x \leq 4 \quad 0 \leq N \leq 15$
ROL Rx,N		$Rx \leftarrow Rx \gg N$	Rotation à gauche de N bits $0 \leq x \leq 4 \quad 0 \leq N \leq 15$
ROR Rx,N		$Rx \leftarrow Rx \gg N$	Rotation à droite de N bits $0 \leq x \leq 4 \quad 0 \leq N \leq 15$

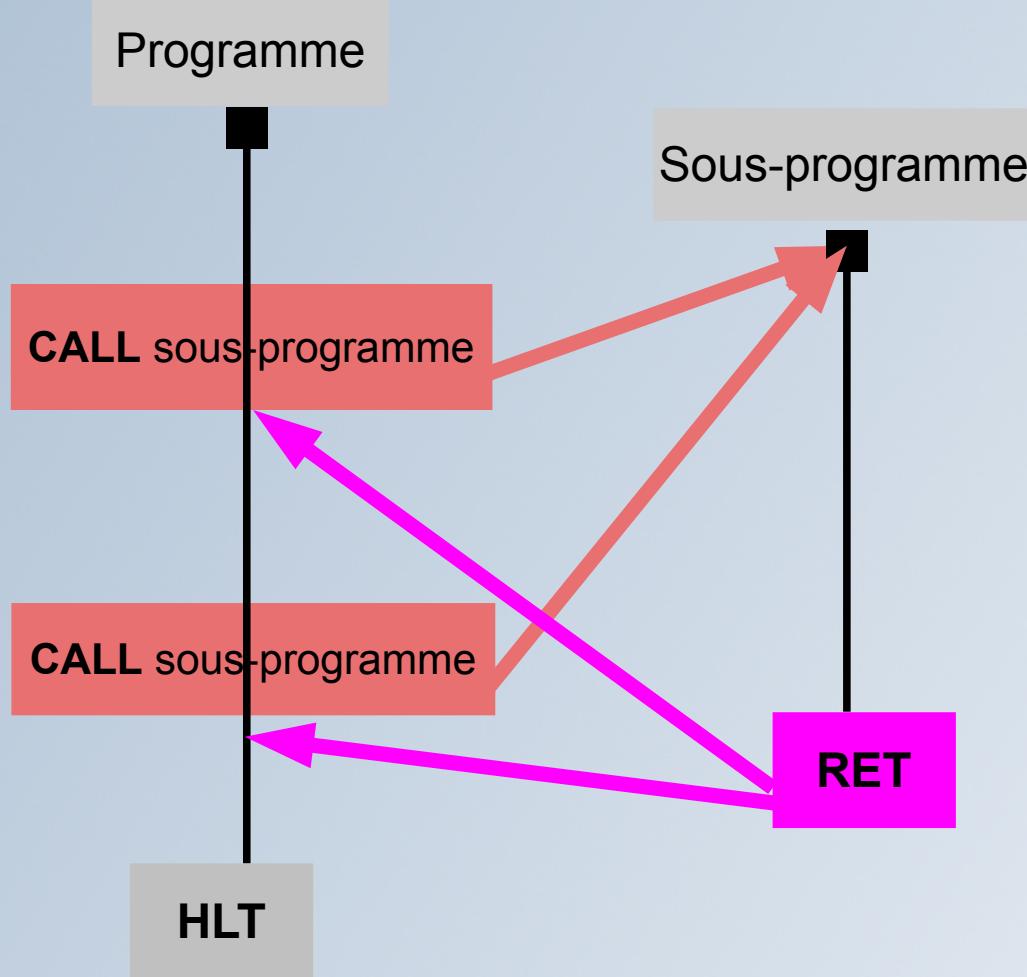
# 8. Langage d'assemblage de SIMCAL instructions de branchement

mnémonique	adressage	résultat	notes
JMP N	Relatif	$IP \leftarrow N$	Branchement limité à IP +511/- 512
JNC N		$IP \leftarrow N$ si Carry=0 (Jump si Not Carry)	Branchement conditionnel limité à IP +/- 256
JC N		$IP \leftarrow N$ si Carry=1 (Jump si Carry)	Branchement conditionnel limité à IP +/- 256
JNZ N		$IP \leftarrow N$ si Zero=0 (Jump si Not Zero)	Branchement conditionnel limité à IP +/- 256
JZ N		$IP \leftarrow N$ si Zero=1 (Jump si Zero)	Branchement conditionnel limité à IP +/- 256
CALL N		$SP \leftarrow SP-1$ $[SP] \leftarrow IP$ puis $IP \leftarrow N$	Branchement Limité à IP +511/-512
INT N	Indirect par mémoire	$SP \leftarrow SP-1$ puis $[SP] \leftarrow FL$ $SP \leftarrow SP-1$ puis $IP \leftarrow [N]$ Flag I $\leftarrow 0$	Interruption logicielle Vecteur N $\leq 255$ Branchement à l'adresse contenue dans le vecteur N
RET	implicite	$IP \leftarrow [SP]$ $SP \leftarrow SP+1$	Retour après CALL
IRET		$IP \leftarrow [SP]$ puis $SP \leftarrow SP+1$ $FL \leftarrow [SP]$ puis $SP \leftarrow SP+1$	Retour d'interruption

# 8. Langage d'assemblage de SIMCAL : instructions de contrôle

Mnémonique	adressage	résultat	notes
NOP	Implicite		No Opération
HLT			Arrêt d'exécution

# 9. Sous-programme et pile : sous-programme



## Sous-programme:

- Bloc d'instructions exécuté à la demande d'un programme "principal"
- Le programme "appelant" se branche au sous-programme par une instruction d'appel : CALL adresse\_du\_sous-programme
- Le sous-programme se termine par une instruction de retour au programme appelant : RET
- L'instruction RET n'a pas d'opérande, l'adresse de retour est implicitement définie
- Lors de l'appel l'adresse de retour est sauvegardée en mémoire dans une zone appelée PILE pointée par le registre pointeur de pile SP
- L'adresse de retour est celle de l'instruction qui suit l'instruction CALL

# 9. Sous-programme et pile : pile

## Pile :

- Zone mémoire destinée à l'empilement d'informations, du fond (adresse la plus grande) vers le sommet (adresse la plus petite).
- La gestion de la pile est partagée entre le programmeur et le calculateur :
  - le programmeur choisit l'emplacement et la taille de la pile, en fonction des disponibilités de la mémoire,
  - le programmeur initialise en début de programme le pointeur de pile SP à l'adresse correspondant au fond de pile,
  - l'Unité Centrale met à jour le pointeur de pile en fonction des empilements/déempilements d'information de façon à ce qu'il pointe toujours sur le sommet de la pile.
- Les informations empilées sont :
  - les adresses de retour aux programmes appelants,
  - des données "utilisateurs".

# 9. Sous-programme et pile : organisation en mémoire

(exemple)

Zone  
Données

Zone  
Programme

PILE

adresses	contenu
	...
0010	Donnée A
0011	Donnée B
0012	Donnée C
	...
0020	Instruction m Instruction n Instruction o <b>RET</b>
	...
0030	Instruction1 Instruction2 CALL 0020 Instruction4 ...
	<b>HLT</b>
	...
0070	sommet fond
	...

Sous-  
Programme

Programme  
Principal

Pointeur de pile (SP)

# 9. Sous-programme et pile : évolution de la pile

Programme principal		
Adr		
0020	LDR	R0,0x70
0021	LDR	SP,R0
0022	LDR	R0,1
0023	CALL	sprog1
<b>0024</b>	STR	[0x10],R0
0025	HLT	

sprog1		
Adr		
0030	LDR	R1,R0
0031	CALL	sprog2
<b>0032</b>	LDR	R0,R2
0033	RET	

sprog2		
Adr		
0040	ADD	R2,R0R1
0041	SHL	R2,1
0042	RET	

Avant appel à sprog1	Après appel sprog1 Avant appel sprog2	Après appel sprog2 Avant retour sprog2	Après retour sprog2 Avant retour de sprog1	Après retour de sprog1
----------------------	--	---	---	------------------------

P  
I  
L  
E

006A	
006B	
006C	
006D	
006E	
006F	
<b>SP -&gt;</b>	fond

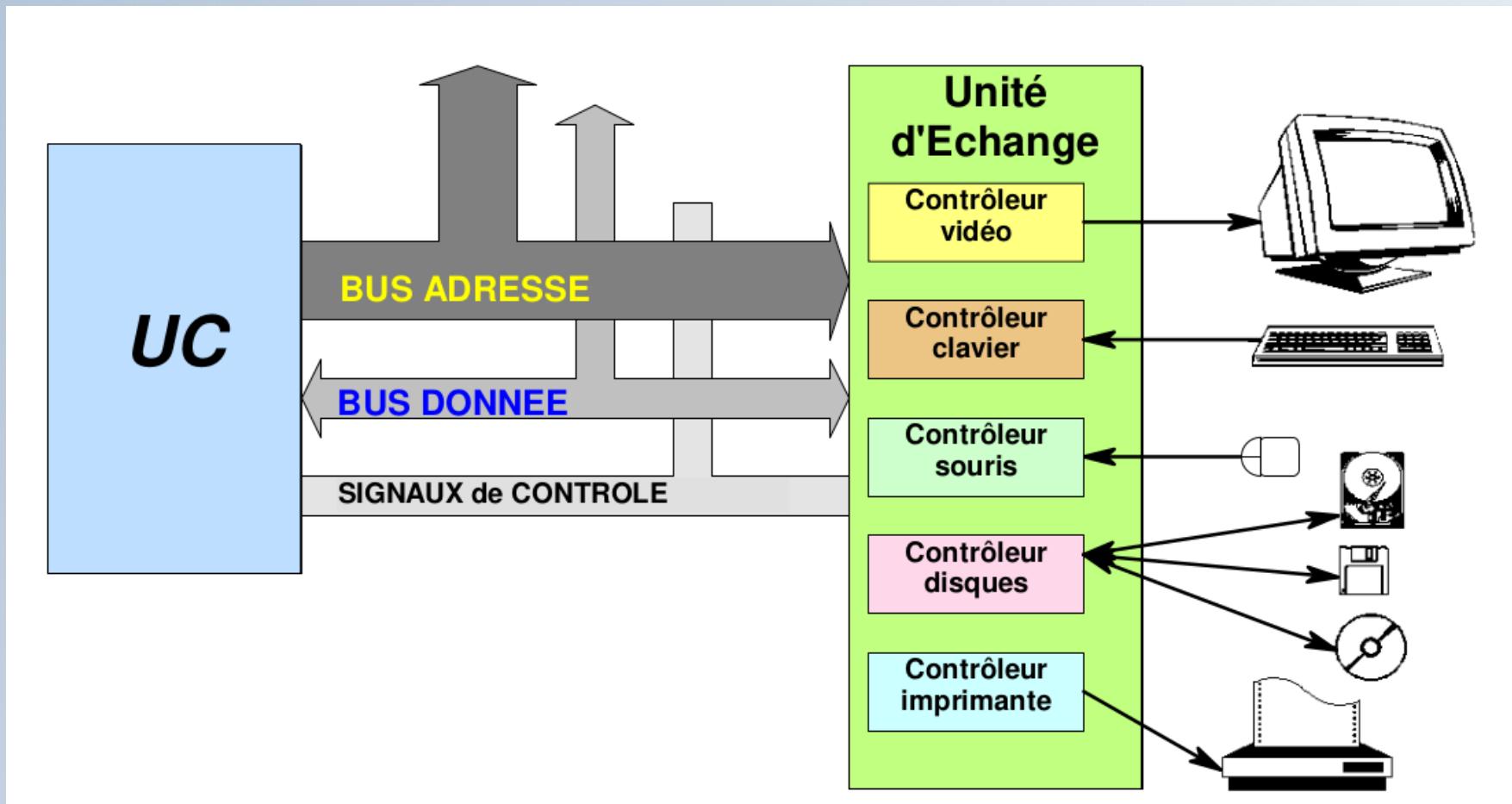
006A	
006B	
006C	
006D	
006E	
<b>SP -&gt;</b>	<b>0024</b>
0070	fond

006A	
006B	
006C	
006D	
<b>SP -&gt;</b>	<b>0032</b>
006F	0024
0070	fond

006A	
006B	
006C	
006D	
006E	0032
<b>SP -&gt;</b>	<b>0024</b>
0070	fond

006A	
006B	
006C	
006D	
006E	0032
006F	0024
<b>SP -&gt;</b>	<b>fond</b>

# 10. Entrées-Sorties : Unité d'échange



L'Unité d'Echange est constituée de circuits spécialisés, contrôleurs d'Entrée et/ou Sortie. Chaque circuit constitue une interface entre l'Unité Centrale et un périphérique;

# 10. Entrées-Sorties : Circuits d'Entrées-Sorties

Circuits d'Entrée-Sortie :

- Circuits "périphériques" spécialisés

- Principaux types:

- E/S parallèle (Ports parallèle)

Ces circuits permettent la communication avec un périphérique, les données étant échangées en parallèle, généralement 8 bits, avec ou sans dialogue.

- E/S série (Ports série)

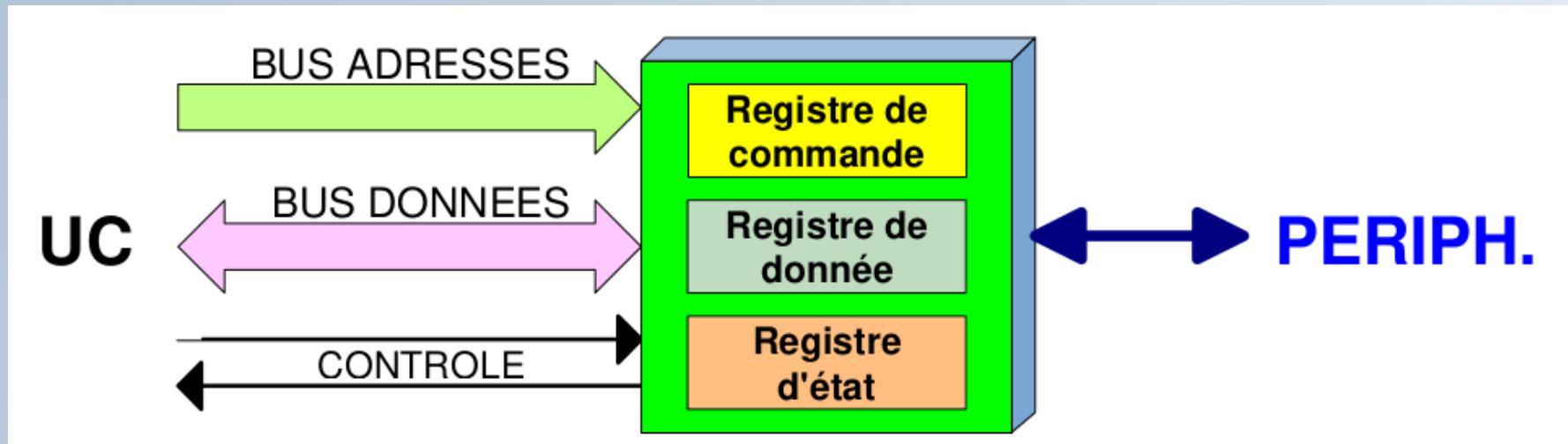
Ces circuits permettent la communication avec un périphérique, les données étant échangées en série, de façon synchrone ou asynchrone. Ces circuits se chargent de la "sérialisation" (émission) et de la "désérialisation" (réception) des données.

- Ces circuits sont programmables et proposent généralement plusieurs modes de fonctionnement. Ils nécessitent donc une initialisation avant utilisation afin de déterminer le mode de fonctionnement et les paramètres de la communication.

- Chaque circuit d'Entrée-Sortie occupe plusieurs adresses dans l'espace adressable par l'Unité Centrale.

- Chaque adresse occupée est celle d'un registre du circuit.

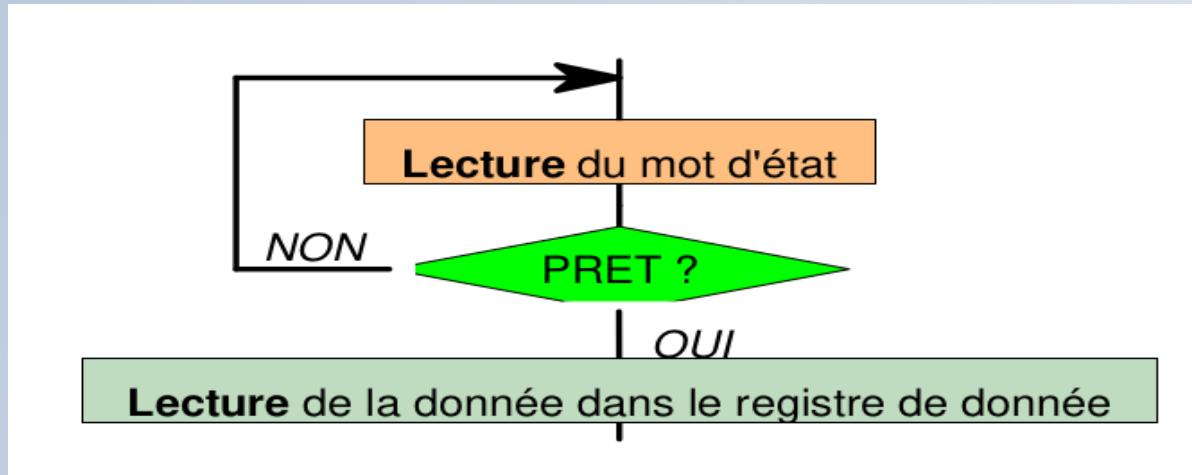
# 10. Entrées-Sorties : structure d'un circuit d'Entrée-Sortie



Structure simplifiée d'un circuit d'Entrée-Sortie:

- Registre de commande:
  - L'Unité Centrale écrit dans ce registre un mot de commande qui initialise le circuit (mode, paramètres,...)
  - Le contenu de ce registre est appelé MOT de COMMANDE
- Registre de donnée:
  - L'Unité Centrale y écrit les données à destination du périphérique
  - L'Unité Centrale y lit les données en provenance du périphérique
- Registre d'état:
  - L'Unité Centrale y lit une information concernant l'état du circuit voire du périphérique lui-même (prêt/non prêt)
  - En général, chaque bit de ce registre est significatif d'un état particulier
  - Le contenu de ce registre est appelé MOT d'ETAT

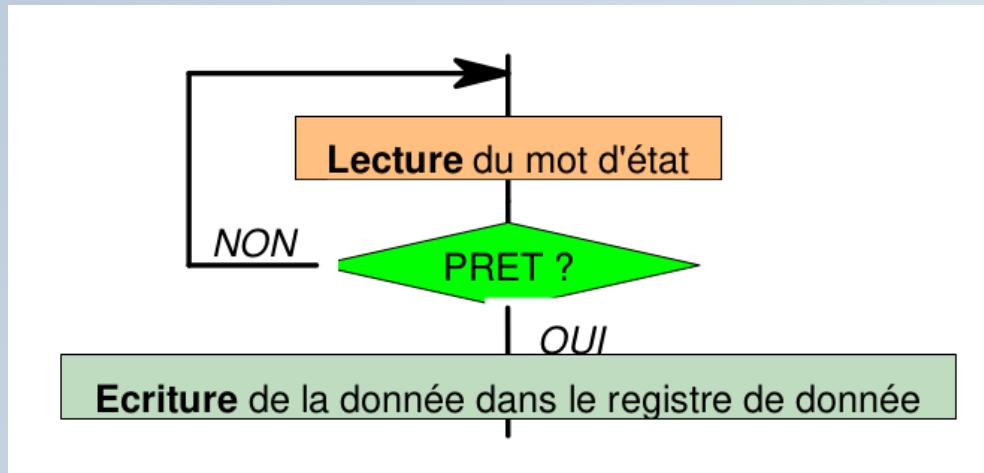
# 10. Entrées-Sorties : entrée par test de mot d'état



## ENTREE par test du MOT d'ETAT

- Ce test se fait dans une boucle:
  - on lit le mot d'état (contenu du registre d'état)
  - on teste le mot d'état ; il indique si le circuit est prêt (donnée reçue) pour un transfert de donnée
    - si le circuit n'est pas prêt on boucle sur la lecture du registre d'état
    - quand le circuit est prêt on sort de la boucle et on lit la donnée reçue (provenant du périphérique)

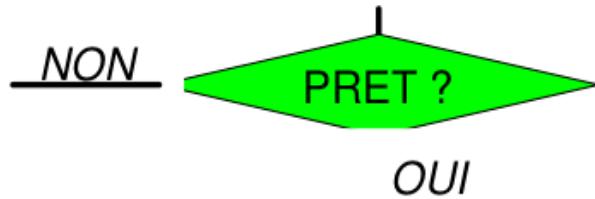
# 10. Entrées-Sorties : sortie par test de mot d'état



## SORTIE par test du MOT d'ETAT

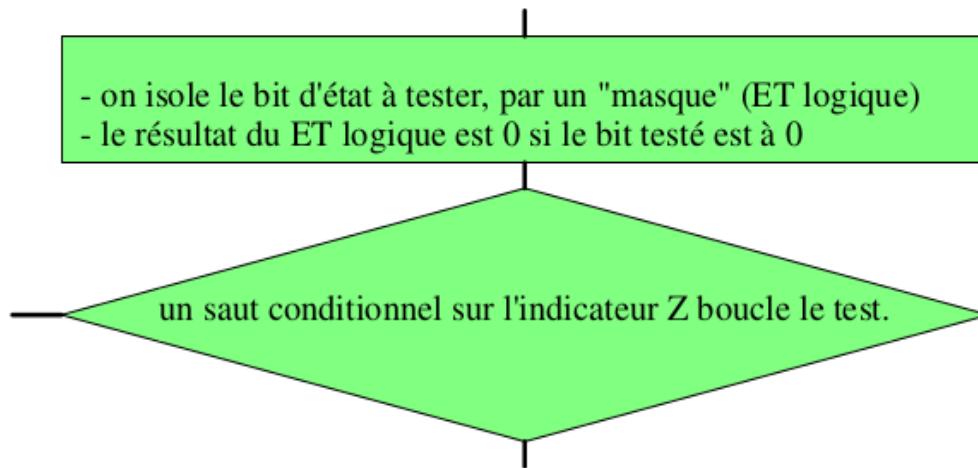
- Ce test se fait dans une boucle :
  - on lit le mot d'état (contenu du registre d'état)
  - on teste le mot d'état ; il indique si le circuit est prêt pour un transfert (sortie) de donnée
  - si le circuit n'est pas prêt on boucle sur la lecture du registre d'état
  - quand le circuit est prêt on sort de la boucle et on écrit la donnée à sortir (à destination du périphérique)

# 10. Entrées-Sorties : test de mot d'état



## Détail du test de MOT d'ETAT

- Un mot d'état peut être constitué de plusieurs bit significatifs permettant de connaître l'état du circuit d'Entrée/ Sortie.
- En général chaque bit significatif indique un état particulier du circuit (exemple, prêt pour une sortie de donnée, donnée reçue, erreur)

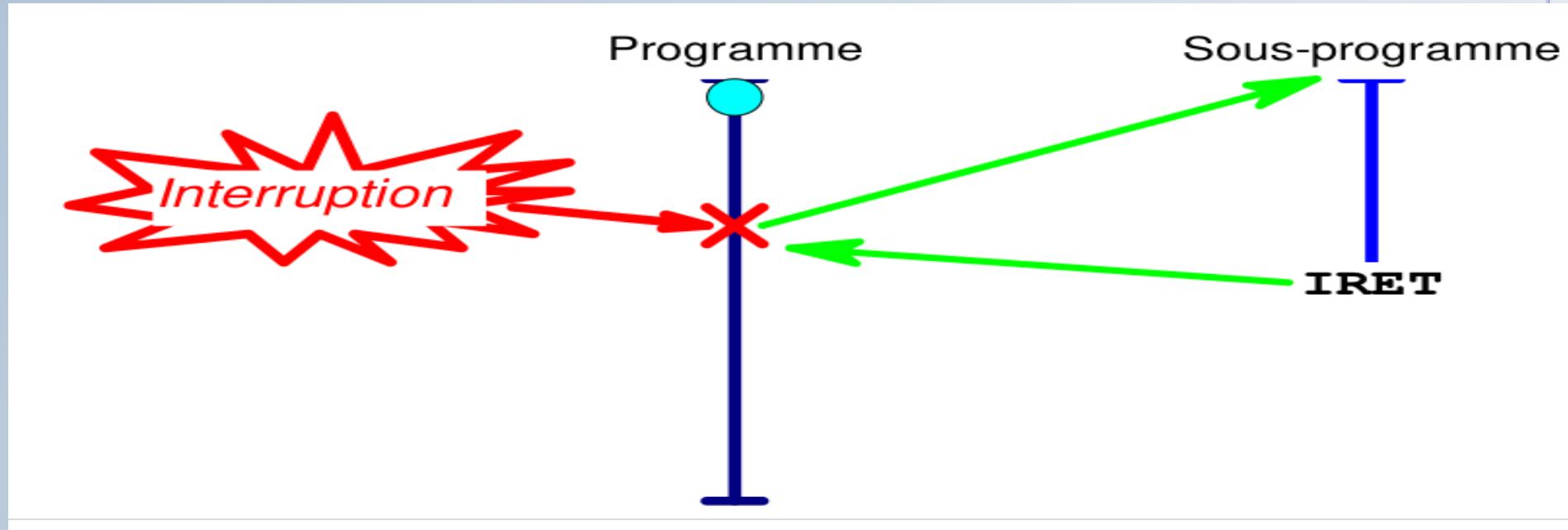


Exemple: pour tester le bit 3 (b3) d'un octet:

b7 b6 b5 b4 b3 b2 b1 b0

ET 0 0 0 0 1 0 0 0

# 11. Interruptions : principe



## Interruption:

- Possibilité d'interrompre l'exécution d'un programme pour exécuter un sous-programme qui se termine par un retour au programme interrompu
- L'interruption peut être provoquée par :
  - un événement matériel externe: signal électrique sur une entrée de contrôle de l'Unité Centrale
  - un événement interne "grave": tentative de division par zéro par exemple
  - une instruction particulière de demande d'interruption (interruption logicielle programmée)
- Le sous-programme d'interruption se termine par une instruction de retour **IRET**

# 11. Interruptions : chronologie de traitement

Séquence de traitement d'une interruption:

0) Pour qu'une demande d'interruption puisse être traitée il faut que l'Unité Centrale soit autorisée à prendre en compte les demandes.

Cela dépend de l'indicateur **I** qui est :

- mis à 1 pour autoriser la prise en compte de la demande
- mis à 0 pour interdire la prise en compte de la demande

1) L'Unité Centrale termine l'exécution de l'instruction en cours

2) L'Unité Centrale signale qu'elle accepte la demande d'interruption

3) Le "demandeur" fournit son "identité" (numéro) à l'Unité Centrale. Ce numéro est appelé **niveau** d'interruption

4) L'Unité Centrale sauvegarde en pile les indicateurs (flags)

5) L'Unité Centrale sauvegarde en pile l'adresse de retour (**IP**)

6) Le **niveau** permet à l'Unité Centrale de lire l'adresse du sous-programme à exécuter, dans un tableau en mémoire appelé Table des Vecteurs d'interruption. Ce tableau détient les adresses des sous-programmes d'interruption, rangés par ordre de niveau croissant

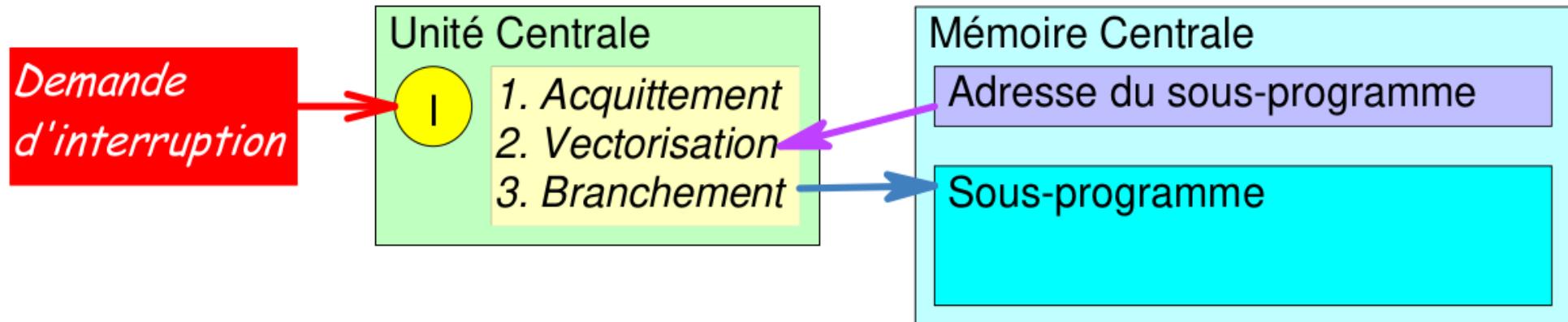
7) L'Unité Centrale remet à zéro l'indicateur **I**, interdisant ainsi toute nouvelle demande

8) L'Unité Centrale se branche à l'adresse du sous-programme et l'exécute

9) L'exécution de l'instruction **IRET** qui termine le sous-programme récupère dans la pile l'adresse de retour , récupère dans la pile les indicateurs et restaure le registre (flags) réautorisant ainsi les demandes futures (**I** remis à 1)

10) L'Unité Centrale retourne au programme interrompu

# 11. Interruptions : vectorisation



Exemple de table de vecteurs:

	adresse donnée
0001	adresse du sous-programme de niveau 1
0002	adresse du sous-programme de niveau 2
0003	adresse du sous-programme de niveau 3
0004	adresse du sous-programme de niveau 4

Pour lire l'adresse d'un sous-programme, l'Unité Centrale utilise le niveau pour adresser la table des vecteurs.

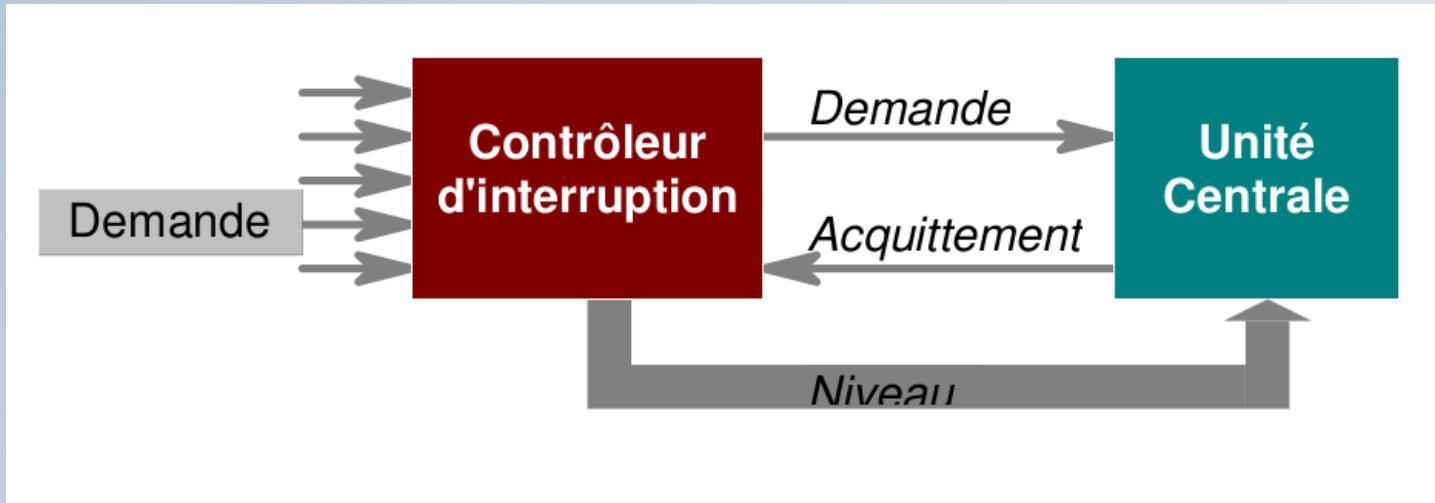
Remarques: Par défaut un sous-programme d'interruption ne peut pas être interrompu (**I=0**).

Pour qu'un sous-programme d'interruption soit lui-même interruptible, il faut y remettre **I à 1**.

Pour fournir le niveau du "demandeur", on utilise un circuit périphérique spécialisé: le **Contrôleur d'Interruption**.



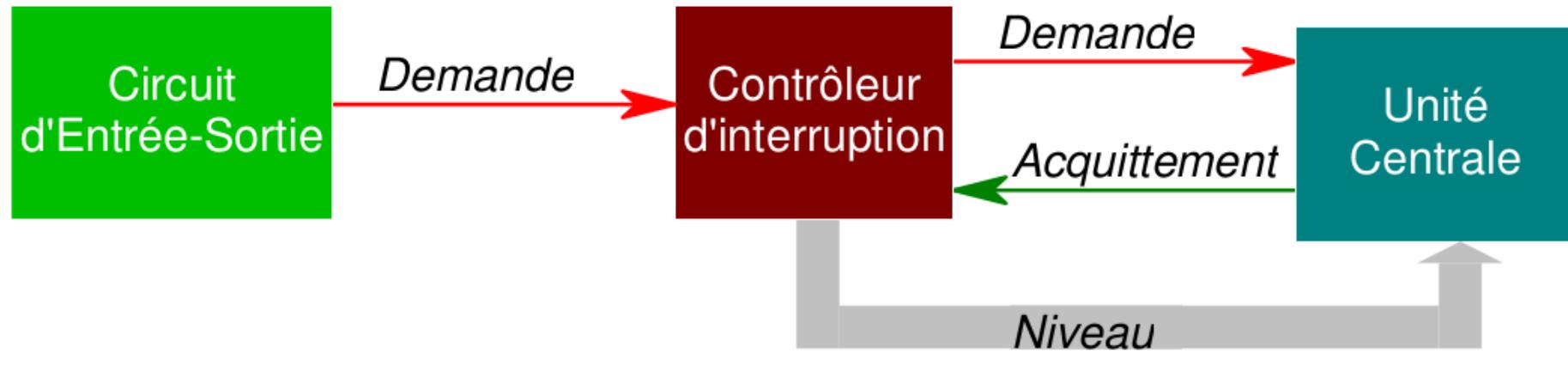
# 11. Interruptions : contrôleur d'interruptions



Contrôleur d'interruption :

- Circuit périphérique capable de gérer plusieurs entrées de demande d'interruption.
- Il prend en compte la demande d'interruption, la transmet à l'Unité Centrale et fournit le niveau de cette demande.
- A chaque entrée est préaffecté un NIVEAU qui sera communiqué à l'Unité Centrale en réponse au signal d'acquittement indiquant l'acceptation de la demande d'interruption

# 11. Interruptions : entrée/sortie par interruption



Transfert par interruption :

- Mode de transfert E-S opposé au transfert par test de mot d'état, sans attente "passive" de l'Unité Centrale.
- Le circuit d'E-S fait une demande d'interruption quand il est "PRET" pour un transfert :
  - Entrée : Lorsque le circuit d'E-S reçoit une donnée provenant du périphérique, il fait une demande d'interruption. Le sous-programme d'interruption associé à cet événement se charge de lire la donnée reçue,
  - Sortie : Lorsque le circuit d'E-S est disponible pour un transfert (sortie précédente terminée) il fait une demande d'interruption. Si l'Unité Centrale n'a pas de donnée à envoyer au périphérique la demande peut être ignorée, sinon le sous-programme d'interruption écrit la donnée à destination du périphérique dans le circuit d'E-S.