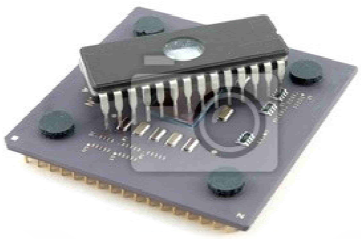
Ministre de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Seddik BENYAHIA - Jijel
Faculté des Sciences & de la Technologie
Département d'Electronique

Support de cours

Systèmes à Microprocesseurs

- Cours & Exercices -



Réalisé par :

Dr. Ammar SOUKKOU

Soukkou.amr@gmail.com

- Manuscrit élaboré selon le programme officiellement agréé et confirmé par le CPNDST.
- ~~Public ciblé :~~
  - Deuxième année Socle Commun (S4).
  - Troisième année Licence Electronique (S5).
  - Master I: Electronique des Systèmes Embarqués.

CONTENU DE LA MATIERE

a. Description de la matière

UE Fondamentale : UEF 3.1.1

Semestre: S5	Crédits	Coefficients	Volume horaire hebdomadaire			Volume Horaire Semestriel (15 semaines)	Travail Complémentaire en Consultation (15 semaines)	Mode d'évaluation	
Intitulé de la matière			**Cours**	**TD**	**TP**			**Contrôle Continu**	**Examen**
Systèmes à Microprocesseurs	**6**	**3**	**3h00**	**1h30**		**67h30**	**82h30**	**40%**	**60%**

b. Objectifs de l'enseignement

Poursuivre l'étude des circuits séquentiels entamés dans le semestre S4. Enseigner à l'étudiant l'architecture, le fonctionnement et la programmation d'un microprocesseur 8 bits, lui faire enfin acquérir les mécanismes de fonctionnement d'un système à microprocesseur (interfaçage, interruptions) ainsi que sa programmation en assembleur.

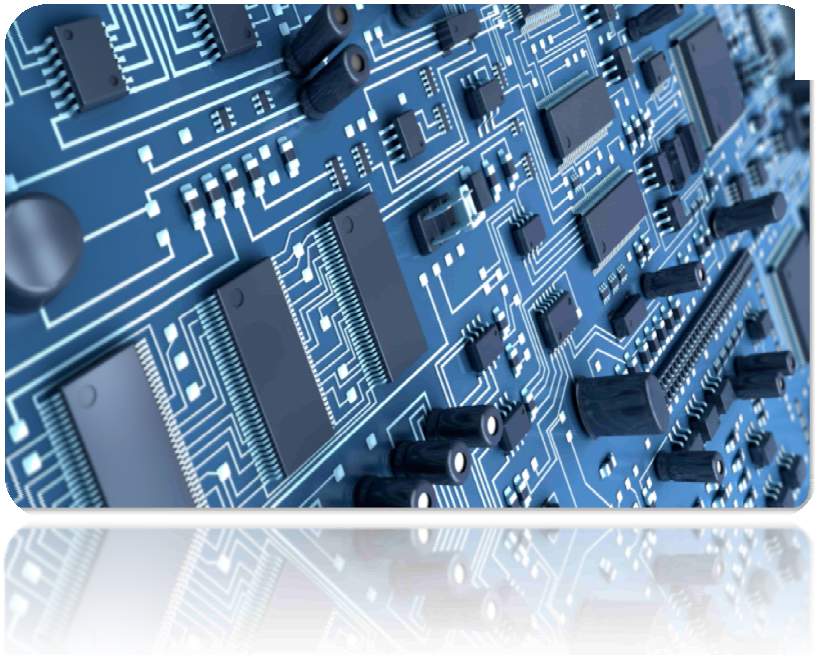
c. Connaissances préalables recommandées

Logique combinatoire et séquentielle.

d. Sommaire

CHAPITRE 1

Les Registres



CHAPITRE 1

LES REGISTRES

1.1. Définition

Un registre est un système séquentiel, permettant la mémorisation d'un ensemble **d'informations** (de **bits**). Il est donc constitué de n bascules, mémorisant chacune un bit, connectées à la même horloge. La figure 1.1 donne un exemple de registre n bits.

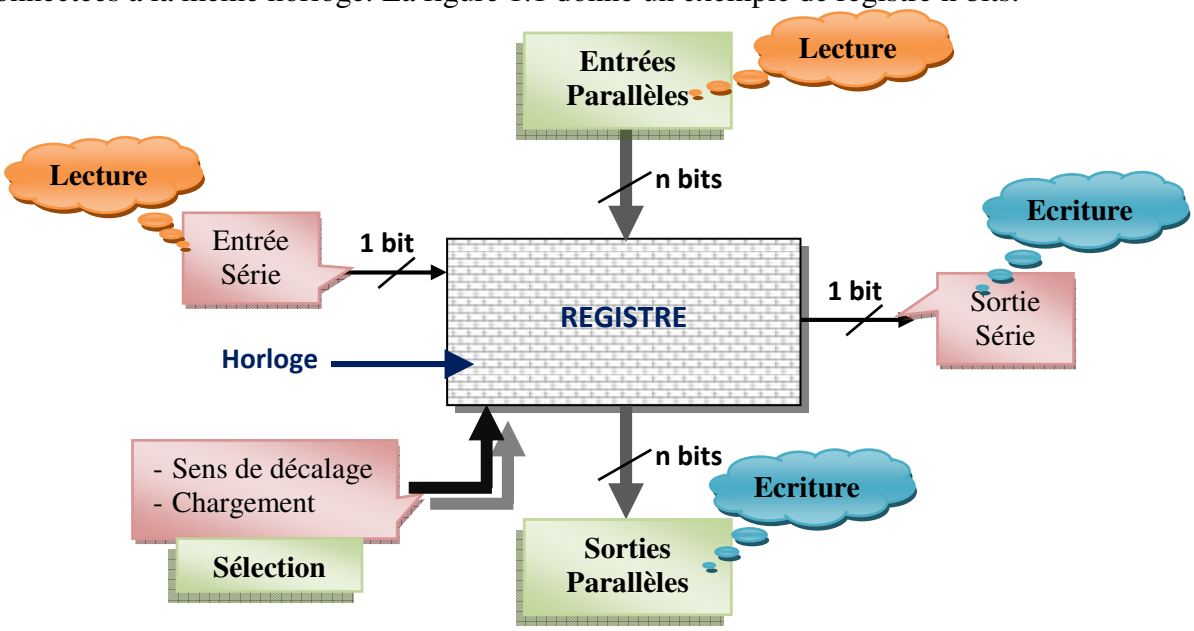


Figure 1.1 : *Structure générale d'un registre.*

1.2. Classement des registres

Il existe plusieurs types de registres dont chacun est bien adapté à un certain type d'application. Donc, la fonction du registre dépend de l'interconnexion entre les entrées et les sorties, comme indiquent les figures 1.2 (a)-(d).

Table 1.1 : *Types des registres.*

Type	Structure	Description	Mode de fonctionnement		
			Ecriture	Lecture	fonction
PIPO	Parallèle - Parallèle	Parallel Input - Parallel Output	Parallèle	Parallèle	Mémorisation
SISO	Série - Série	Serial Input - Serial Output	Série	Série	Décalage
SIPO	Série - Parallèle	Serial Input - Parallel Output	Série	Parallèle	Décalage
PISO	Parallèle - Série	Parallel Input - Serial Output	Parallèle	Série	Décalage

- Un autre type de registres dits ***registres universels*** (registres à entrées série ou parallèles et sorties série ou parallèles).
- **Exemple d'un registre à 04 bits :**

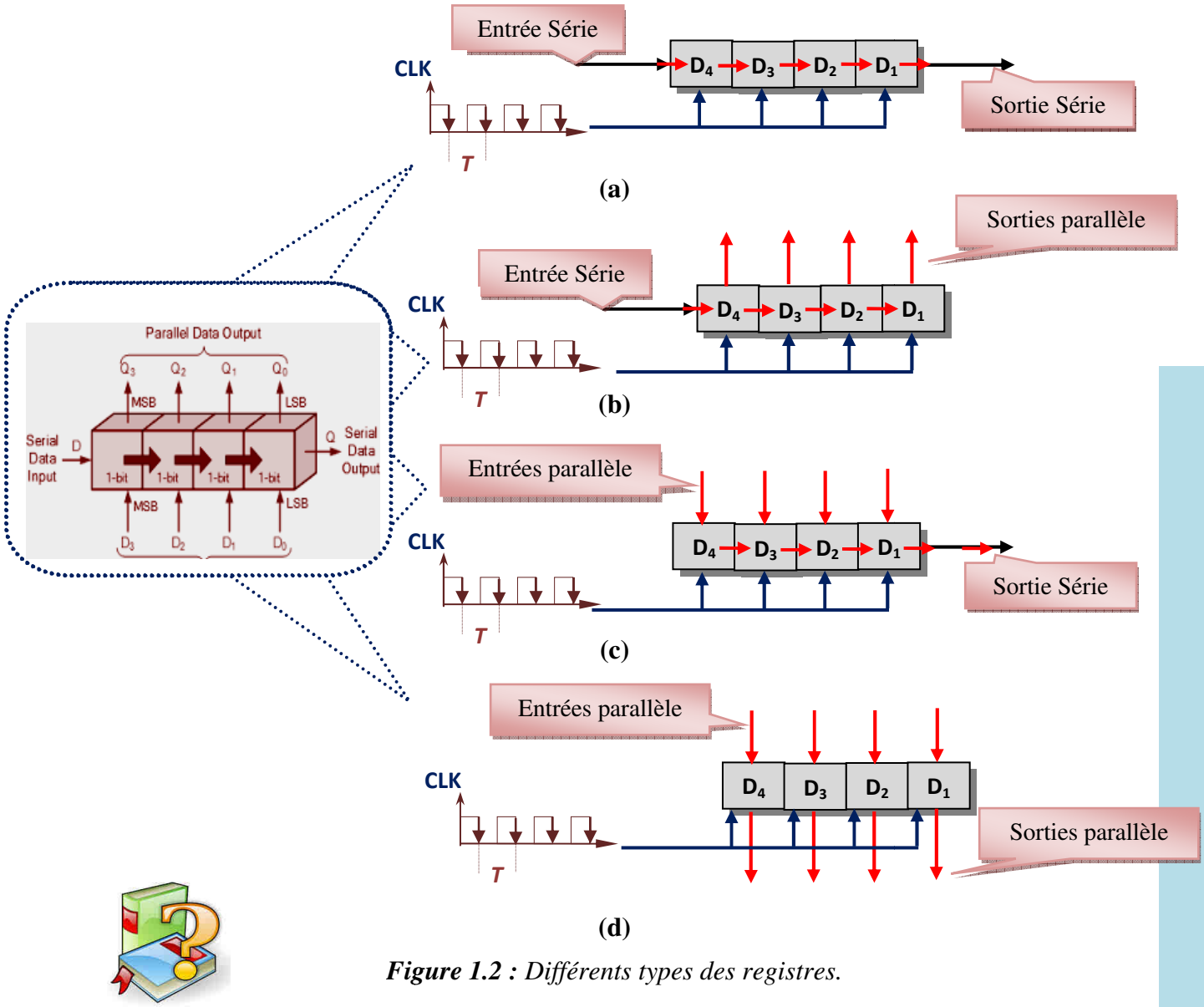


Figure 1.2 : *Différents types des registres.*

Remarques :

- Les bascules dans un registre sont reliées (interconnectées) entre elles soit directement, soit par des opérateurs combinatoires.
- Les opérations sur les bascules est conditionnées par un signal externe : signal de synchronisation (Horloge : **CLK**).

1.2.1. Applications des registres

Par le biais des registres, on peut réaliser un certains nombres d'applications particulière en électronique numérique, à savoir :

- Stockage (mémorisation) des informations sous forme binaire.
- Transfert (transmission) des données.
- Conversion des données (parallèle-série, Série- parallèle).
- Opérations arithmétiques et logiques (test, comptage, comparaison, division, …).
- Synchronisation des signaux avec une horloge.

- …

Suite à ces applications offertes par les registres, on peut classer les registres en quatre grandes famille :

- Registres à décalage.
- Registres de mémorisation.
- Compteurs.
- Registres de synchronisation.

Donc, le classement des registres s'effectue à travers deux facteurs principaux :

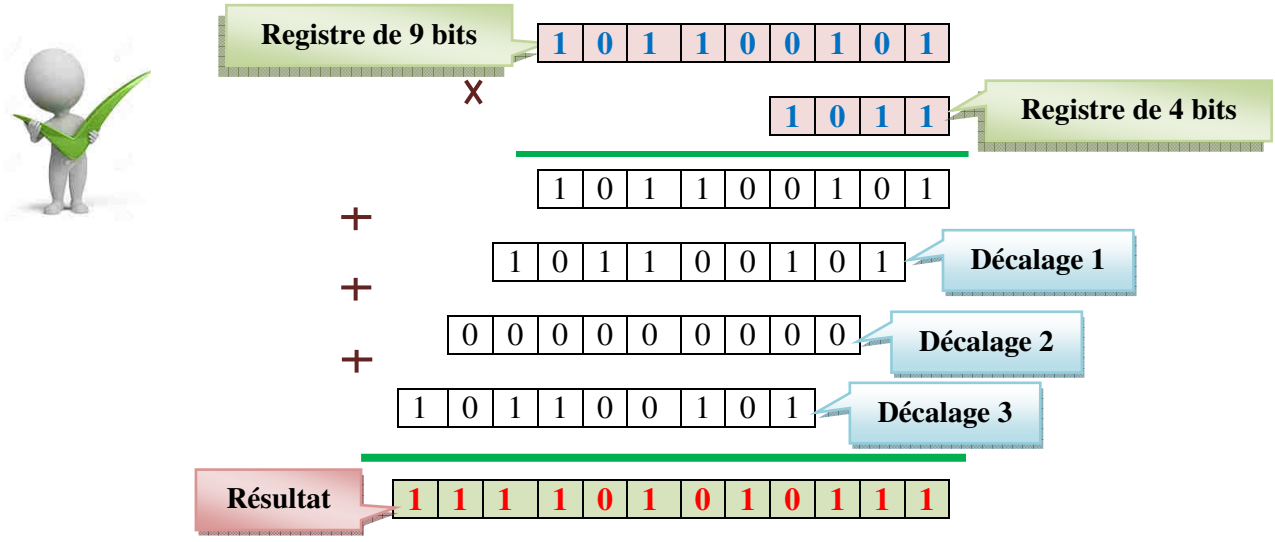
- ***Structurel***, mesuré par la taille du registre, c-à-d, le nombre de bascules.
- ***Fonctionnel***, mesuré par sa fonction (décalage à droite, …), par le type de transfert des données, …

1.3. Registres à décalage

L'opération de décalage consiste à transférer des données entre deux cellules binaires consécutives, selon un sens (de transfert) :

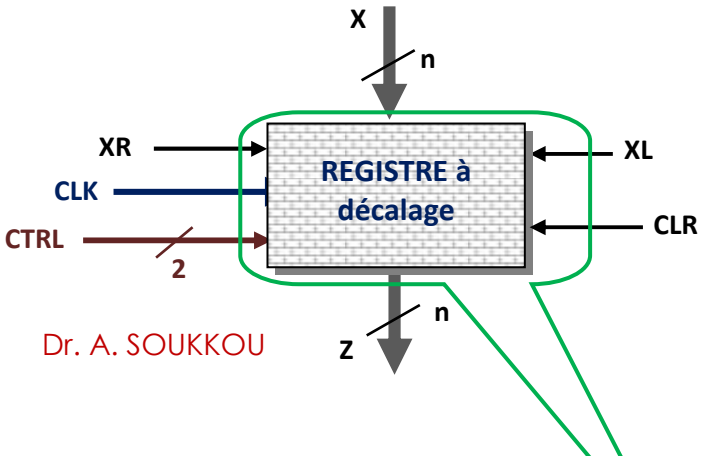
- Bidirectionnel.
- Unidirectionnel (de droite vers la gauche, de gauche vers la droite).

Exemple : Dans la multiplication binaire, il est nécessaire de décaler des bits à gauche ou à droite.



1.3.1. Structure d'un registre universel

La structure d'un registre dépend de sa fonction, c-à-d, le type de chargement : série (introduction des bits d'information l'un après l'autre) ou parallèle (introduction simultanée des bits d'informations). La structure générale d'un registre ainsi que son mode de fonctionnement sont schématisées par la figure 1.3.



Variable	Rôle
X	Entrée de chargement parallèle sur n bits
XR	Entrée série (décalage série à droite)
XL	Entrée série (décalage série à gauche)
CTRL	2 lignes de contrôle (chargement parallèle ou décalage série)
CLR	Entrée d'initialisation (a)synchrone
Z	Sortie sur n bits

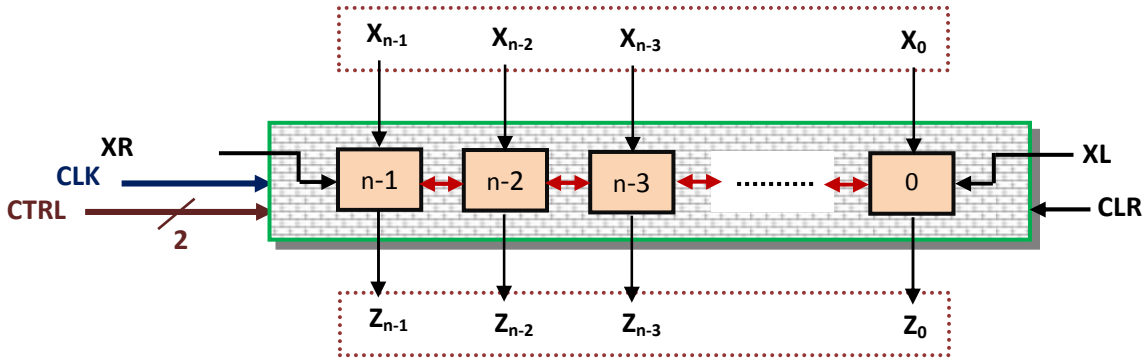


Figure 1.3 : *Schéma bloc et mode de fonctionnement d'un registre universel.*

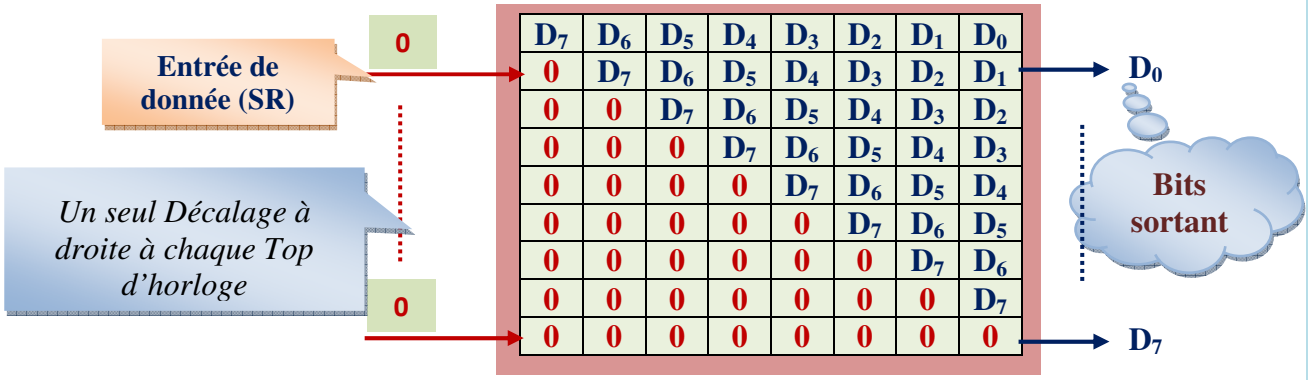
Un registre dit ***universel*** est un registre qui effectue le décalage à droite ou à gauche et un chargement série ou parallèle. Ce type de registre dispose d'entrées de mode de fonctionnement qui définissent le sens de décalage et le type de chargement.

Exemple : Soit un mot de 08 bits (taille de registre = 8), on peut lui faire subir plusieurs types de décalage, à savoir :

- Décalage à droite (SR : Shift Right).
- Décalage à gauche (SL : Shift Left)
- Décalage circulaire (avec rotation)
  - Rotation à gauche.
  - Rotation à droite.

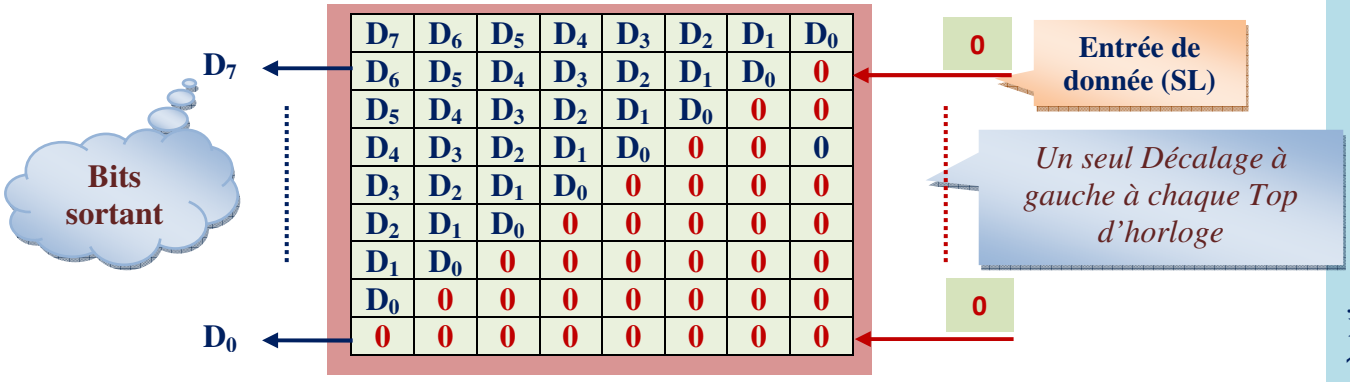
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

DATA



D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	D_7	D_6	D_5	D_4	D_3	D_2	D_1
0	0	D_7	D_6	D_5	D_4	D_3	D_2
0	0	0	D_7	D_6	D_5	D_4	D_3
0	0	0	0	D_7	D_6	D_5	D_4
0	0	0	0	0	D_7	D_6	D_5
0	0	0	0	0	0	D_7	D_6
0	0	0	0	0	0	0	D_7
0	0	0	0	0	0	0	0

(a)



D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
D_6	D_5	D_4	D_3	D_2	D_1	D_0	0
D_5	D_4	D_3	D_2	D_1	D_0	0	0
D_4	D_3	D_2	D_1	D_0	0	0	0
D_3	D_2	D_1	D_0	0	0	0	0
D_2	D_1	D_0	0	0	0	0	0
D_1	D_0	0	0	0	0	0	0
D_0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(b)

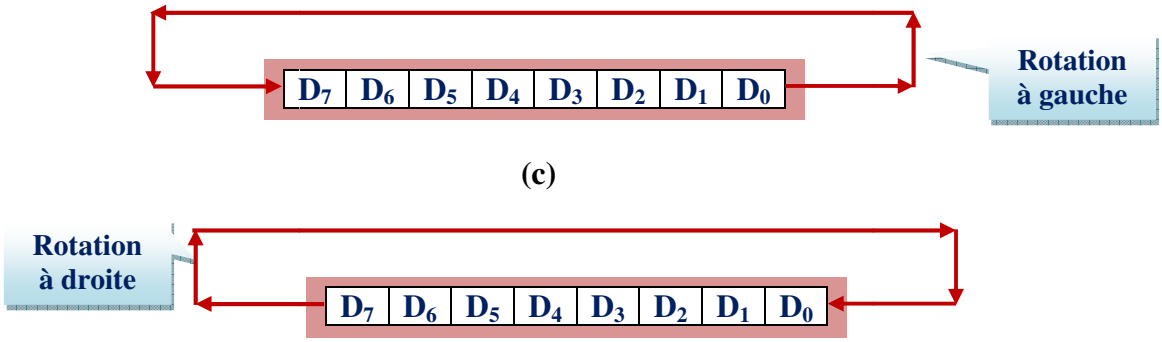
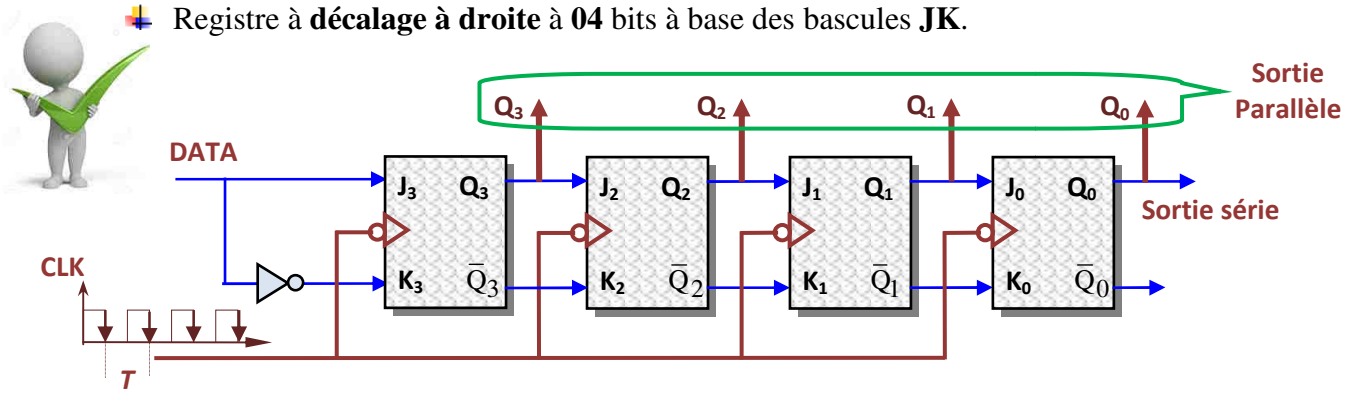


Figure 1.4 : *Différents type de décalage sur un registre de 8 bits.*

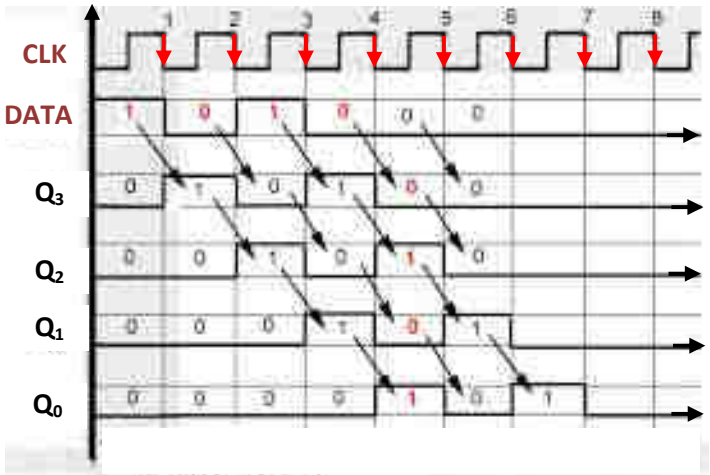
- **Problème :** Le problème consiste à trouver un circuit qui permet de transférer le contenu d'une bascule à son voisin (amont ou aval) à chaque top d'horloge ?

Exemples :

- Registre à **décalage à droite** à **04** bits à base des bascules **JK**.



(a)



DATA	$Q_3Q_2Q_1Q_0$	CLK
	0 0 0 0	Etat initial
1	1 0 0 0	1er Top d'horloge
0	0 1 0 0	2eme Top d'horloge
1	1 0 1 0	3eme Top d'horloge
0	0 1 0 1	4eme Top d'horloge
	0 1 0 1	Etat final

(b)

Figure 1.5 : *Décalage à droite synchrone: (a) Structure du registre. (b) Chronogramme.*

Lors de chaque Top d'horloge, la sortie d'une bascule recopie l'information présente à ses entrées, c-à-d, la sortie de la bascule qui la précède ; $J_i = Q_{i-1}$, $K_i = \bar{Q}_{i-1}$. On parle d'un décalage de l'information. ***On parle du principe du maitre-esclave***.

- Registre de transfert **Parallèle- Parallèle** à 04 bits à base des bascules D.

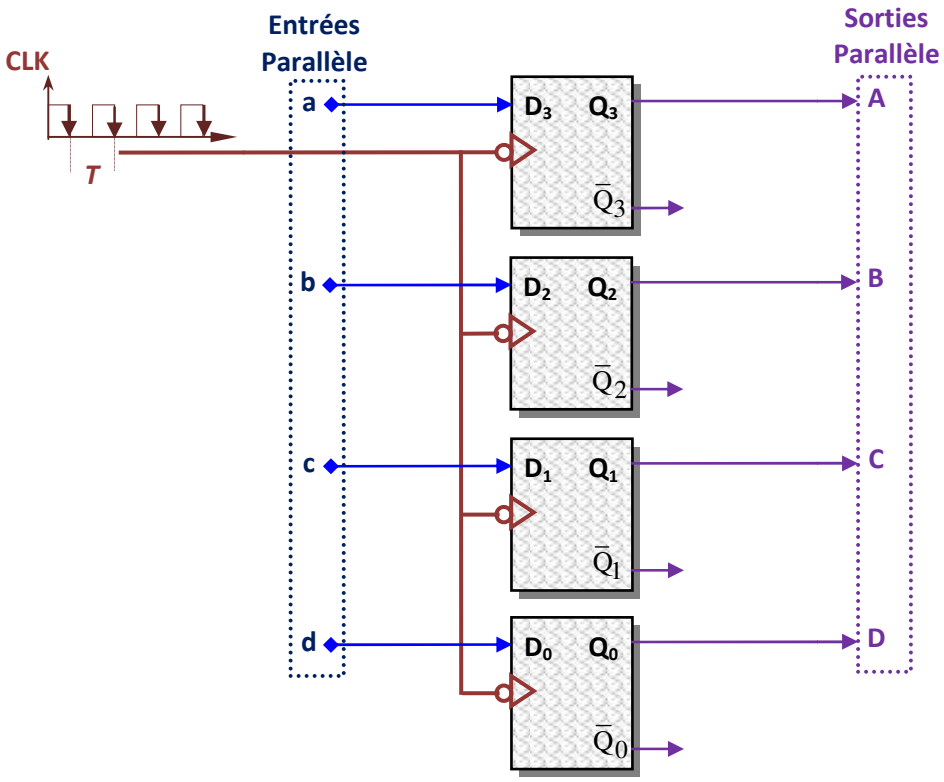
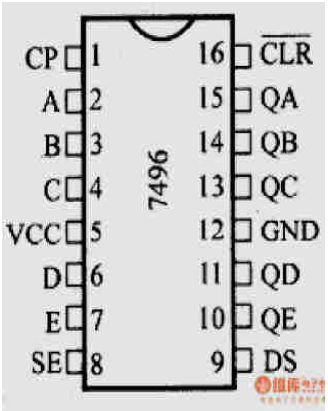


Figure 1.6 : *Transfert parallèle-parallèle à base des bascules D (synchrone).*

- ***Références techniques :*** On peut retrouver les registres à décalage dans les circuits intégrés ci-dessous :
  - 74194 : Registre à décalage à 04 bits à entrée/sortie parallèle ou entrée série à décalage à droite ou à gauche.
  - 7496 : Registre à décalage à 05 bits à entrée/sortie parallèle ou entrée série.
  - 74198 : Registre à décalage à 08 bits à chargement ou série, à décalage à droite ou à gauche et sortie.
  - 74178 : Registre à décalage à chargement parallèle synchrone.
  - …

Exemple : Le circuit intégré **7496** comprenant **5** bascules.



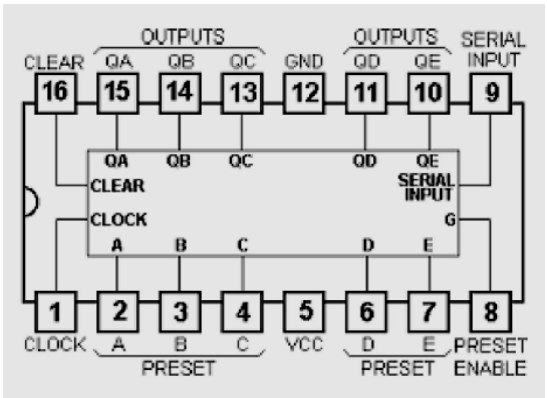


Figure 1.7 : *Brochage du circuit intégré **7496**.*

Remarque : La manipulation la plus fréquente qu'on fait subir aux données conservées dans des bascules ou des registres est le transfert (échange de données d'un registre à un autre).

- Dans les **transferts synchrones** (les plus courants), on utilisera l'**horloge**.
- Dans les **transferts asynchrones**, on utilisera les **entrées de remise à 0 ou 1 asynchrones**.

Conclusion : Parmi les caractéristiques des registres, on peut citer :

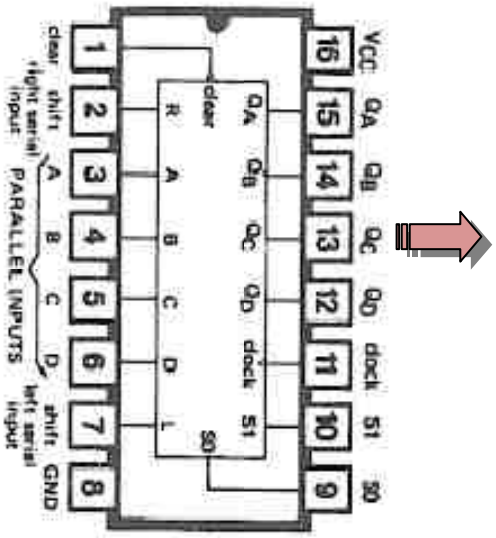
- Taille (nombre de bits qui constituent le registre).
- Types de décalage (à droite, à gauche, les deux directions par une commande de mode (registres bidirectionnels)).
- Deux types de chargement des registres (parallèle, série).

1.4. Registre universel, le 74LS194A

Parmi les registres universels, on trouve le 74194 qui est un registre à décalage universel bidirectionnel à 4 bits synchrone. Il possède une entrée de remise à zéro asynchrone. Son évolution est conditionnée par les fronts de CLK. Il présente 4 modes (S1 et S0) :

- **Mode 0** (**S1S0 = 00**): Aucune opération (Inhibition de l'horloge).
- **Mode 1** (**S1S0 = 01**): Décalage à droite avec entrée série gauche sur SR.
- **Mode 2** (**S1S0 = 10**): Décalage à gauche avec entrée série droite SL.
- **Mode 3** (**S1S0 = 11**): Chargement parallèle des entrées A, B, C et D.

La description et le schéma de brochage sont données à la figure x.



Broche	Fonction	
DCBA	Entrées parallèles	
SL	Entrées séries gauche	
SR	Entrées séries droite	
S1S0	Sélection du mode de fonctionnement	
	00	Blocage
	01	Décalage à droite
	10	Décalage à gauche
	11	Chargement parallèle
CLR	Remise à zéro asynchrone des sorties	
CLK	Horloge de synchronisation	
$Q_DQ_CQ_BQ_A$	**Sorties**	

Figure 1.8 : *Brochage du circuit intégré* **74LS194A**.

Le tableau 1.2 illustre le fonctionnement détaillé du **registre universel 74LS194A**. un échantillon du chronogramme résume graphiquement la table de vérité de ce registre.

Table 1.2 : *Table de vérité du registre universel 74LS194A.*

INPUTS											OUTPUS			
CLEAR	MODE		CLOCK	SERIAL		PARALLEL				QA	QB	QC	QD
	S1	S0		LEFT	RIGHT	A	B	C	D				
L	X	X	X	X	X	X	X	X	X	L	L	L	L
H	X	X	⌊	X	X	X	X	X	X	QA0	QB0	QC0	QD0
H	H	H	⌈	X	X	a	b	c	d	a	b	c	d
H	L	H	⌈	X	H	X	X	X	X	H	QAn	QBn	QCn
H	L	H	⌈	X	L	X	X	X	X	L	QAn	QBn	QCn
H	H	L	⌈	H	X	X	X	X	X	QBn	QCn	QDn	H
H	H	L	⌈	L	X	X	X	X	X	QBn	QCn	QDn	L
H	L	L	X	X	X	X	X	X	X	QA0	QB0	QC0	QD0

X: Don't Care : Don't Care
a ~ d : The level of steady state input voltage at input A ~ D respectively
QA0 ~ QD0 : No charge
QAn ~ QDn : The level of QA, QB, QC, respectively, before the mst recent positive transition of the clock.

Chronogramme

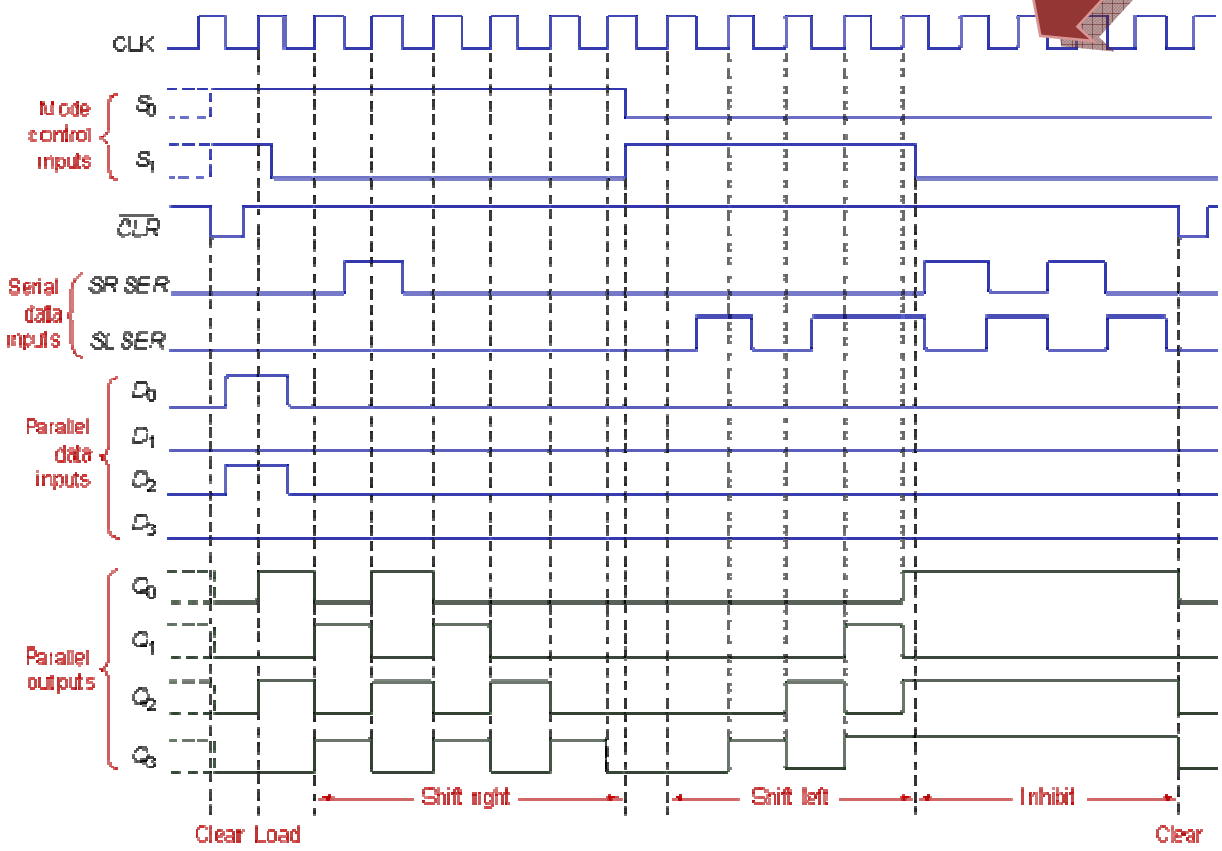


Figure 1.9 : *Echantillon de chronogramme du registre universel 74LS194A.*

Exemple d'application des registres : Transfert de données numériques par liaison série

Le transfert de l'information au sein des systèmes numériques est réalisé sur des mots binaires parallèles. Or lors du transport de l'information sur de longues distances la transmission série est utilisée (coût, fiabilité). La figure x illustre le principe de transmission série des données numériques.

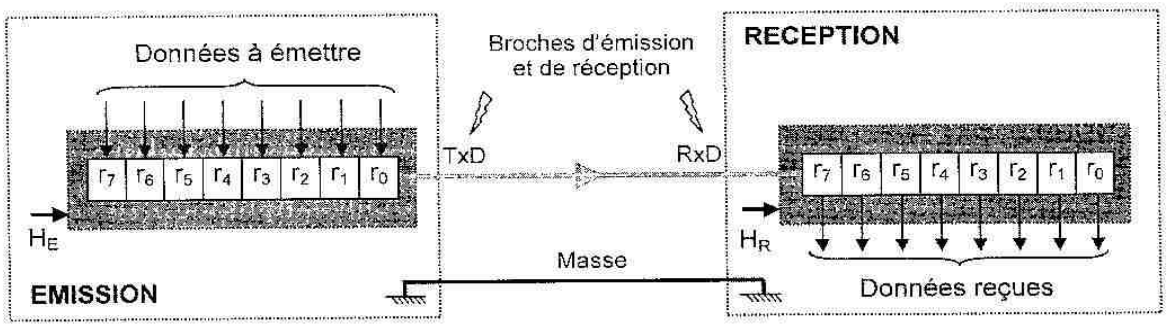


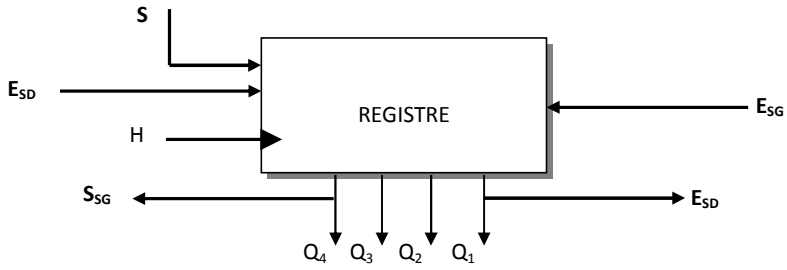
Figure 1.10 : *Principe de transmission série des données numériques.*

1.5. Exercices

EXERCICE 01 :

On veut réaliser un registre à décalage bidirectionnel à entrées série $\mathbf{E_{SD}}$ ou $\mathbf{E_{SG}}$ selon le sens du décalage. Les sorties séries sont $\mathbf{S_{SD}}$ ou $\mathbf{S_{SG}}$, les sorties parallèles sont $\mathbf{Q_1}$, $\mathbf{Q_2}$, $\mathbf{Q_3}$ et $\mathbf{Q_4}$. On utilise quatre bascules **D** à front montants.

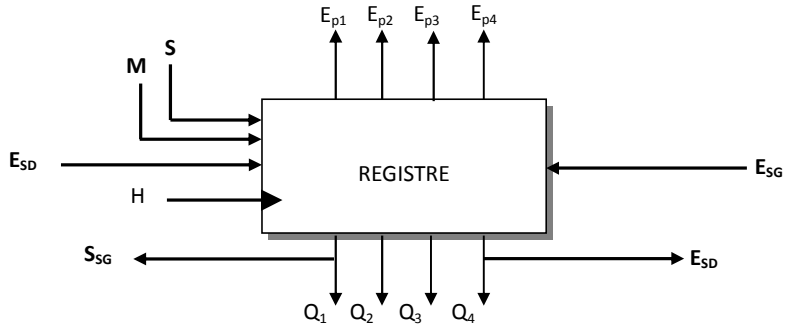
Le sens du décalage est commandé par le signal **S** (1 pour décaler à droite, 0 pour décaler à gauche).



- Donner les équations des signaux d'entrée des bascules D_i (i = 1, 2, 3, 4).
- Donner le schéma du circuit en utilisant des portes NAND en plus des bascules.

EXERCICE 02 :

On veut réaliser un registre dit '***universel***' comme indique cette figure



- Les entrées parallèles sont notées $\mathbf{E_i}$ (i = 1, 2, 3, 4),
- entrées série $\mathbf{E_{SD}}$ et $\mathbf{E_{SG}}$,
- Les sorties parallèles sont $\mathbf{Q_1}$, $\mathbf{Q_2}$, $\mathbf{Q_3}$ et $\mathbf{Q_4}$.

On sélectionne le mode parallèle par **M = 1**, et le mode série par **M = 0**. Le sens de décalage dépend de **M** et du signal de commande **S** (1 pour décaler à droite, 0 pour décaler à gauche).

On réalisera ce registre à l'aide de quatre bascules **D** à front montants et des portes NAND.

EXERCICE 03 :

On désire utiliser le registre à décalage universel 74LS194 pour un chargement parallèle des données et une lecture série de ces données avec décalage vers la droite:

- Remplir le tableau de fonctionnement ci-dessous pour permettre un chargement parallèle du mot binaire 1011 dans le registre 74LS194 ?
- En supposant que le chargement du mot binaire a été effectué, donnez le câblage du 74LS194 pour réaliser un décalage rotatif de la gauche vers la droite?

Entrées								Sorties			
Clear	Mode		CLK	Parallèle				Q_D	Q_C	Q_B	Q_A
	S1	S0		D	C	B	A				
?	?	?	?	?	?	?	?	?	?	?	?

1.5. Les compteurs

Par définition, un compteur est un registre constitué d'un ensemble de bascules interconnectées de façon particulière dont les impulsions (données) d'entrée se propagent pas à pas suivant une loi propre au type de compteur. La figure 1.11 illustre les cinq caractéristiques principales des compteurs.

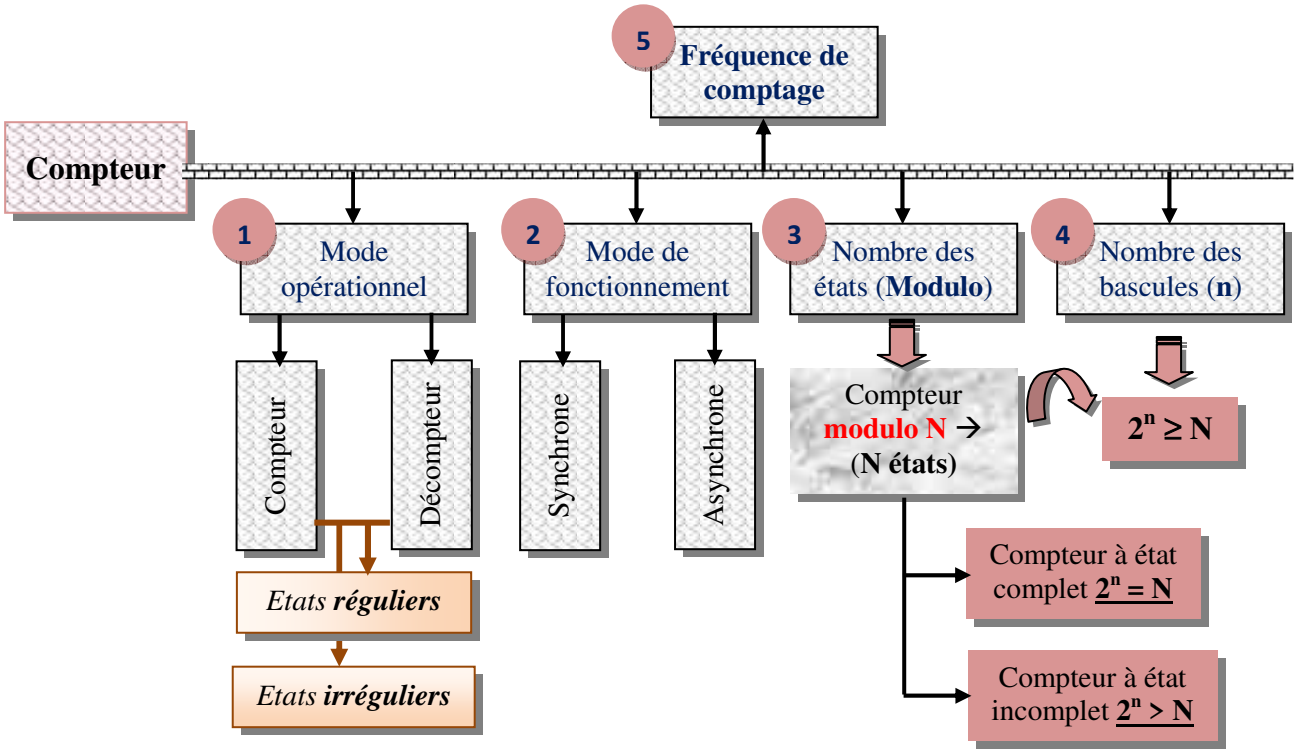


Figure 1.11 : *Caractéristiques principales des compteurs.*

- Le **modulo** d'un compteur est le **nombre des différents états logiques** que les sorties peuvent occuper.
- Un compteur modulo 2^n est constitué de n bascules et peut compter de **0** à 2^n**-1**.

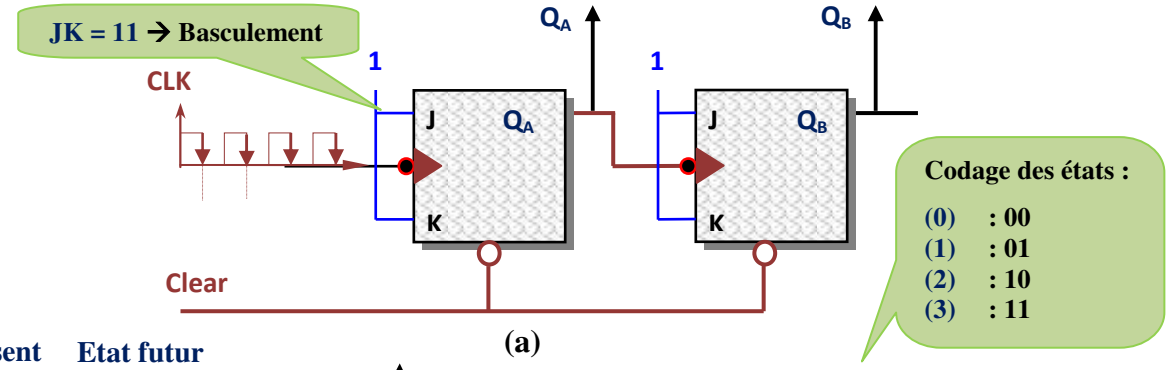
Exemples:

- Compteur modulo 16 (N = 16) ⇒ $2^n \geq 16$ ⇒ **n = 4 bascules.**
- Compteur modulo 10 (N = 10) ⇒ $2^n \geq 10$ ⇒ **n = 4 bascules.**
- Compteur modulo 6 (N = 6) ⇒ $2^n \geq 6$ ⇒ **n = 3 bascules.**
- Compteur modulo 5 (N = 5) ⇒ $2^n \geq 5$ ⇒ **n = 3 bascules.**

1.5.1. Compteurs asynchrones à cycle complet

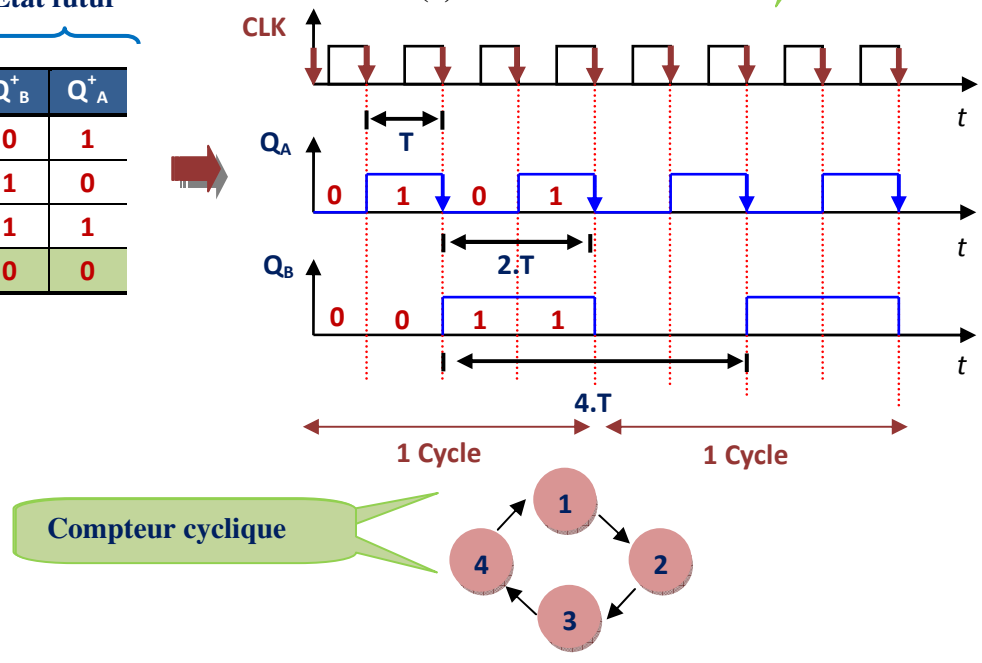
La réalisation d'un compteur asynchrone progressif consiste à mettre en cascade des bascules (assurant la fonction diviseur par 2), à détecter la combinaison de remise à zéro, puis à l'appliquer à l'entrée de forçage à zéro de chaque bascule (**Clear**).

✓ **Compteur asynchrone modulo 4 :** ((**N = 4**) ⇒ $2^n \geq 4$ ⇒ **n = 2** bascules) → *Compteur binaire asynchrone de 2 bits* → *Compteur à état complet.*



(a)

Etat présent		Etat futur	
N	Q_B	Q_A	Q^+_B	Q^+_A
0	0	0	0	1
1	0	1	1	0
2	1	0	1	1
3	1	1	0	0



(b)

Figure 1.12 : *Structure, table des états et chronogramme d'un compteur asynchrone modulo 4.*

- Les 4 états représentés par les nombres **0**, **1**, **2** et **3** en binaires. Ce compteur part de l'état initial (**0 : 00**), compte en binaire pur jusqu'à l'état final (**3 : 11**) en croissant, revient o (0 : 00) et recommence. On parle d'un **compteur progressif**.

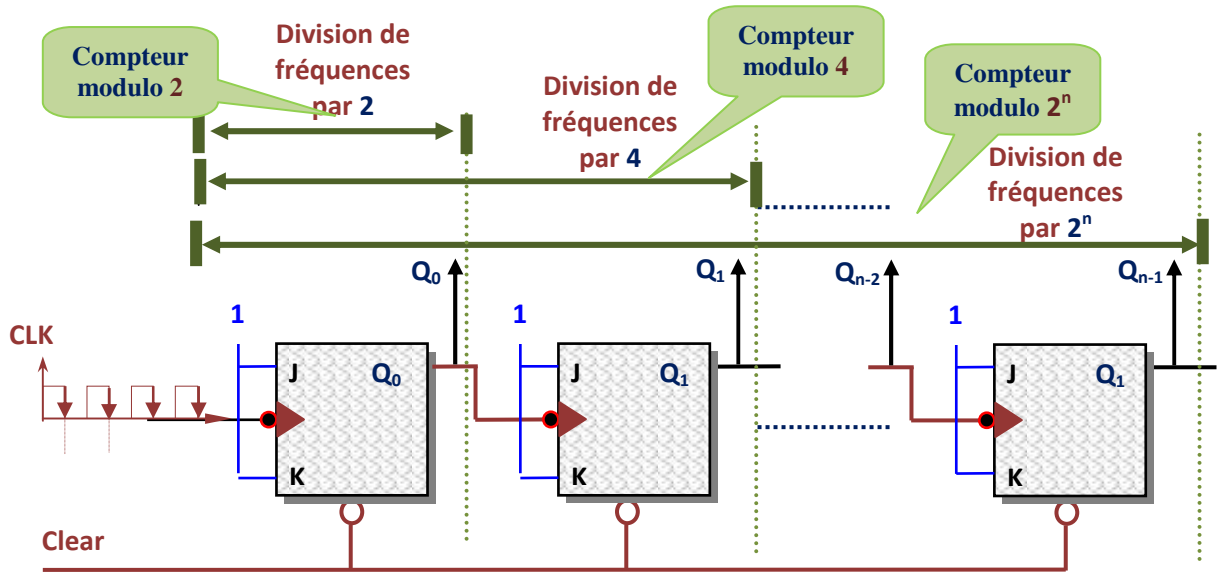


Figure 1.13 : *Diviseur de fréquences.*

Le tableau 1.3 illustre les différentes possibilités offertes par le montage de la figure précédente où les bascules sont interconnectées l'une après l'autres.

Table 1.3 : *Compteurs progressifs.*

Compteur *progressif*	Nombre des états (N)	Nombre de bascules (n)	Etats décimaux	Type de diviseur
Compteur modulo 2	2	1	(0,1)	Diviseur de fréquences par 2
Compteur modulo 4	4	2	(0,1,2,3)	Diviseur de fréquences par 4
Compteur modulo 8	8	3	(0,1,2,3,…,7)	Diviseur de fréquences par 8
Compteur modulo 16	16	4	(0,1,2,…, 15)	Diviseur de fréquences par 16
Compteur modulo 32	32	5	(0,1,2,…,31)	Diviseur de fréquences par 32
Compteur modulo 64	64	6	(0,1,2,…,63)	Diviseur de fréquences par 64
---	---	---	---	---
Compteur modulo N	**N**	**$2^n = N$**	**(0,1,2,…,N-1)**	**Diviseur de fréquences par N**

1.5.2. Compteurs asynchrones à états (cycle) incomplets

Par définition, un compteur à cycle incomplet est un compteur qui n'exploite que quelques états stables parmi les 2^n états offerts par les n bascules.

Pour obtenir un compteur asynchrone à états incomplets (**M # 2^n**), où **N** est le nombre des états et **n** est le nombre des bascules, On prend la même structure que le compteur modulo **2^n**, sauf l'application du signal **Clear** (pour toutes les bascules du montage) lorsque le compteur arrive à l'état **M**.

✓ **Compteurs asynchrones modulo 10**

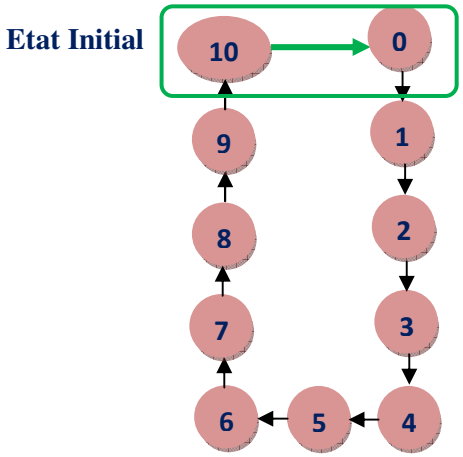
Pour obtenir un compteur asynchrone modulo 10 (**11** états : 0,1,2, …, 10), on prend la même structure que le montage du compteur modulo **16** (**16 = 2^4**), puis on applique le signal externe **Clear = 1010** = (**10**) pour remise à l'état initial (**0**) = **0000** toutes les bascules, comme indique le tableau ci-dessous.

Table 1.4 : *Tables des états d'un compteur asynchrone modulo 10.*

Compteurs asynchrones modulo **16**

Compteurs asynchrones modulo **10**

Etat décimal	Etat présent $Q_DQ_CQ_BQ_A$	Etat futur $Q^+_D\,Q^+_C\,Q^+_B\,Q^+_A$	
0	0000	0001	Etat Initial
1	0001	0010	
2	0010	0011	
3	0011	0100	
4	0100	0101	
5	0101	0110	
6	0110	0111	
7	0111	1000	
8	1000	1001	
9	1001	1010	
10	1010	0000	Etat final
11	1011	1100	
12	1100	1101	
13	1101	1110	
14	1110	1111	
15	1111	000	



$$Clear = Q_D\bar{Q}_CQ_B\bar{Q}_A$$

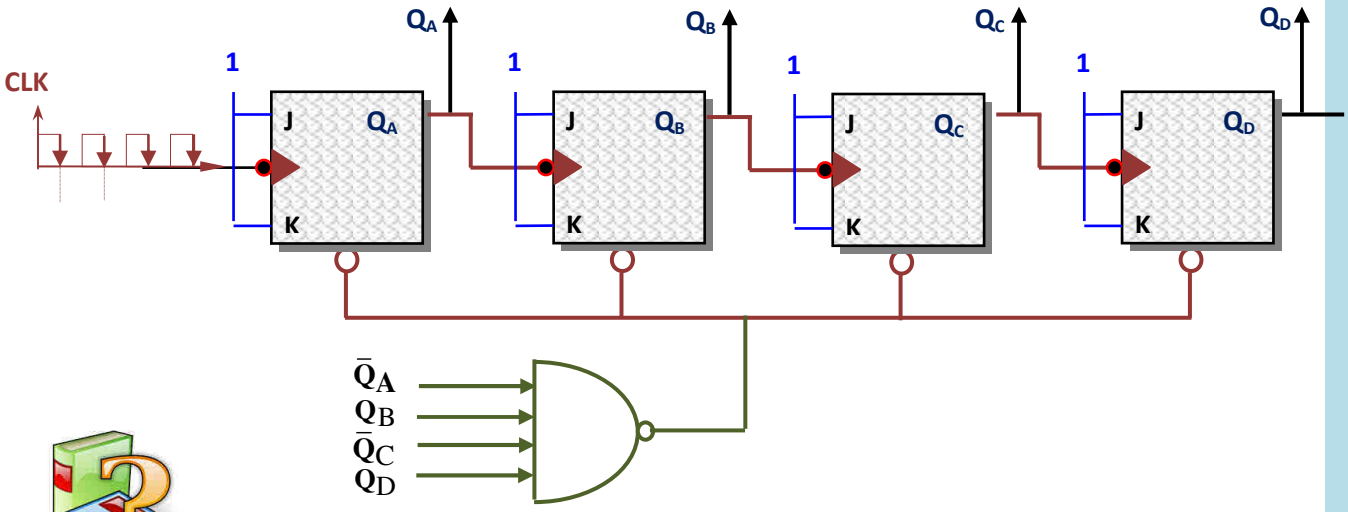


Figure 1.14 : *Compteurs asynchrones modulo 10 (**Compteur self-stopping**).*

Remarque : Pour la réalisation des compteurs ***self-stopping*** , deux méthodes sont offertes :

- Utiliser des bascules possédant les entrées asynchrones (**Clear**, **Preset**) pour bloquer les états indésirables (voir exemple précédent) ou
- Exploiter les tables de transitions (**compteurs synchrones**).

Pour la réalisation de Décompteurs asynchrones (**Compteur régressifs**), on garde la même structure de compteurs asynchrones, sauf la synchronisation s'effectue par **les sorties complémentaires** des bascules, comme indique la figure ci-dessous.

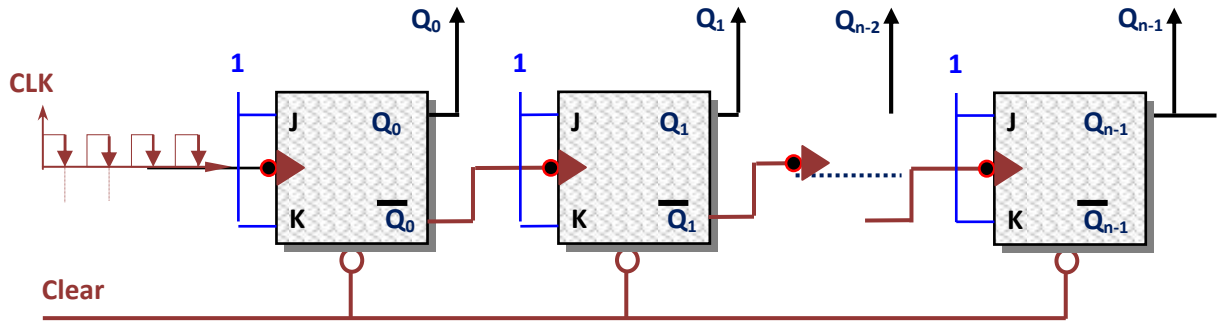


*Figure 1.15 : Décompteurs asynchrones (**Compteur régressifs**).*

1.5.3. Compteurs synchrones

Dans ce type de compteurs, le changement des états des bascules s'effectue simultanément.

a. Compteurs synchrones à cycle complet : $N = 2^n$

Exemple : Compteur modulo 8 synchrone

N = 8 ⇒ n = 3 Bascules ($Q_CQ_BQ_A$) sont nécessaires pour réaliser ce compteur

Table 1.5 : Tables des états d'un compteur synchrone modulo 8.

Etat présent		Etat futur	Entrées des bascules JK						Entrées des bascules D		
Etat décimal	$Q_CQ_BQ_A$	$Q^+_C\,Q^+_B\,Q^+_A$	C		B		A		BCA		
			J_C	K_C	J_B	J_A	D_C	D_B	D_C	D_B	D_A
0	000	001	0	x	0	1	1	x	0	0	1
1	001	010	0	x	1	x	x	1	0	1	0
2	010	011	0	x	x	1	1	x	0	1	1
3	011	100	1	x	x	x	x	1	1	0	0
4	100	101	x	0	0	1	1	x	1	0	1
5	101	110	x	0	1	x	x	1	1	1	0
6	110	111	x	0	x	1	1	x	1	1	1
7	111	000	x	1	x	x	x	1	0	0	0

Les entrées J et K des bascules sont obtenues par utilisation de la tables de KARNAUGH, avec :

$$J_A = K_A = 1$$
$$J_B = K_B = Q_A$$
$$J_C = K_C = Q_A \,.\, Q_B$$

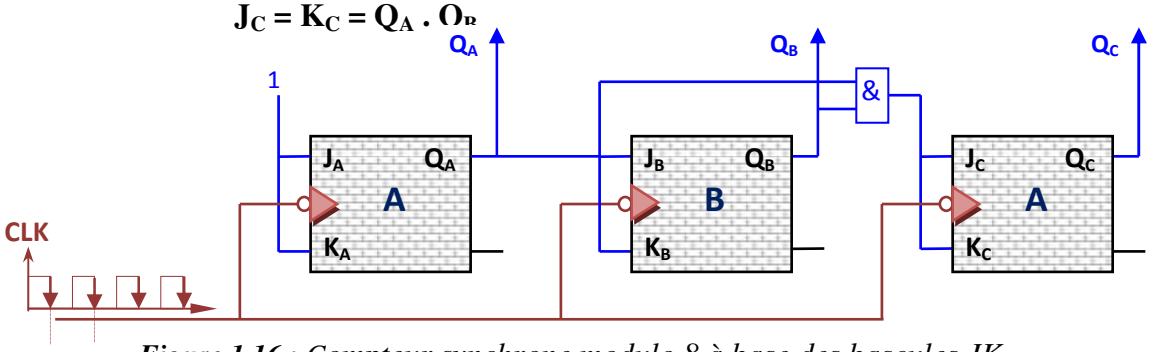


Figure 1.16 : Compteur synchrone modulo 8 à base des bascules JK.

De la même manière, on peut construire ce compteur à base des bascules D, où Les entrées D des bascules sont obtenues par utilisation de la tables de KARNAUGH, avec :

$$D_A = \bar{Q}_A$$
$$D_B = Q_A \oplus Q_B$$
$$D_C = Q_C \oplus (Q_B \cdot Q_A)$$

b. Compteurs synchrones à cycle incomplet : $2^n \geq N$

Exemple : Compteur modulo 6 synchrone

N = 6 ⇒ n = 3 Bascules ($Q_C Q_B Q_A$) sont nécessaires pour réaliser ce compteur.

Table 1.6 : *Tables des états d'un compteur synchrone modulo 8.*

	Etat présent	Etat futur	Entrées des bascules JK							Entrées des bascules D		
Etat décimal	$Q_C Q_B Q_A$	$Q^+_C Q^+_B Q^+_A$	C		B		A			BCA		
			J_C	K_C	J_B	J_A	D_C	D_B		D_C	D_B	D_A
0	000	001	0	x	0	1	1	x		0	0	1
1	001	010	0	x	1	x	x	1		0	1	0
2	010	011	0	x	x	1	1	x		0	1	1
3	011	100	1	x	x	x	x	1		1	0	0
4	100	101	x	0	0	1	1	x		1	0	1
5	101	000	x	0	1	x	x	1		1	1	0
6	110	xxx	x	x	x	x	x	x		x	x	x
7	111	xxx	x	x	x	x	x	x		x	x	x

Les entrées D des bascules sont obtenues par utilisation de la tables de KARNAUGH, avec :

$$D_A = \bar{Q}_A$$
$$D_B = \bar{Q}_A \cdot Q_B + Q_A \cdot \bar{Q}_B \cdot Q_C$$
$$D_C = Q_A \cdot Q_B + \bar{Q}_A \cdot Q_C$$

- ***Références techniques :*** on peut retrouver les compteurs dans les circuits intégrés comme indique le tableau ci-dessous.

Table 1.7 : *Compteurs en circuits intégrés.*

Circuit intégré	Fonction
7490	Compteur binaire asynchrone modulo 10
7492	Compteur binaire asynchrone modulo 12
7493	Compteur binaire asynchrone modulo 16
74163	Compteur binaire synchrone modulo 16
74160	Compteur binaire synchrone modulo 10

Exemple : Le circuit intégré **74163** comprenant **04** bascules du type JK .

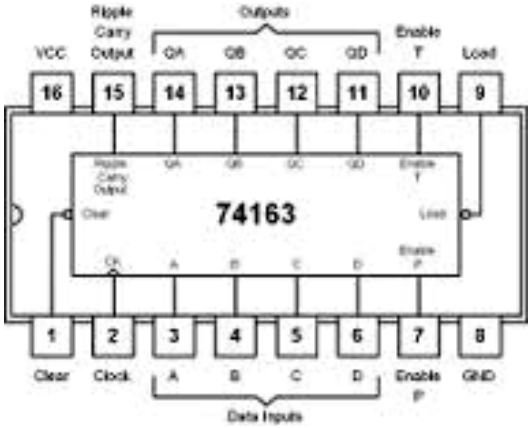


Figure 1.17 : *Brochage du circuit intégré **74163**.*

1.6. Compteurs spécifiques : Les compteurs à registres à décalage

Un compteur à registre à décalage est un registre dont la sortie série est réacheminée à son entrée série pour produire des séquences spéciales. On rencontre deux types :

- Compteurs à séquences irrégulières.
- Compteur Johnson.
- Compteur en anneau.

1.6.1. Compteurs à séquences irrégulières

Ce type de compteur permet de produire une séquence binaire irrégulière illustrée par un digramme d'état quelconque.

Exemple : Créer un compteur permettant de produire la séquence binaire irrégulière illustrée par le diagramme d'état ci-dessous, en utilisant les bascules JK.

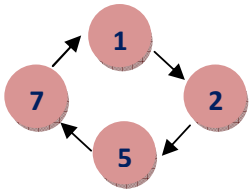


Table 1.8 : *Tables des états d'un compteur synchrone à séquences irrégulières.*

	Etat présent	Etat futur	Entrées des bascules JK					
Etat décimal	$Q_C Q_B Q_A$	$Q^+_C Q^+_B Q^+_A$	C		B		A	
			J_C	K_C	J_B	J_A	D_C	D_B
0	000	xxx	x	x	x	x	x	x
1	001	010	0	x	1	x	x	1
2	010	101	0	x	x	1	1	x
3	011	xxx	x	x	x	x	x	x
4	100	xxx	x	x	x	x	x	x
5	101	111	x	0	1	x	x	1
6	110	xxx	x	x	x	x	x	x
7	111	001	x	0	1	x	x	1

Les entrées J et K des bascules sont obtenues par utilisation de la tables de KARNAUGH, avec :

$$J_A = 1$$
$$K_A = \overline{Q}_C$$
$$J_B = K_B = 1$$
$$J_C = K_C = Q_B$$

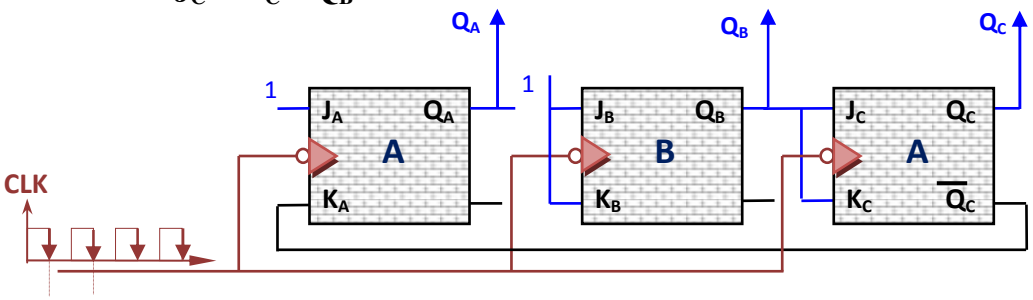


Figure 1.18 : *Compteur synchrone à séquences irrégulières à base des bascules JK.*

1.6.2. Compteur Johnson

Compteur Johnson est un compteur modulo **N = 2*n**, où n est le nombre des bascules, dont le complément de la sortie de la dernière bascule est retro-couplée vers l'entrée de la première bascule.

Exemple : Compteur Johnson à 04 bits → 2*4 = 8 états, comme indique la table ci-dessous.

Table 1.9 : *Tables des états d'un compteur Johnson à 04 bits.*

	Etats
Top d'horloge	$Q_AQ_BQ_CQ_D$
0	0000
1	1000
2	1100
3	1110
4	1111
5	0111
6	0011
7	0001

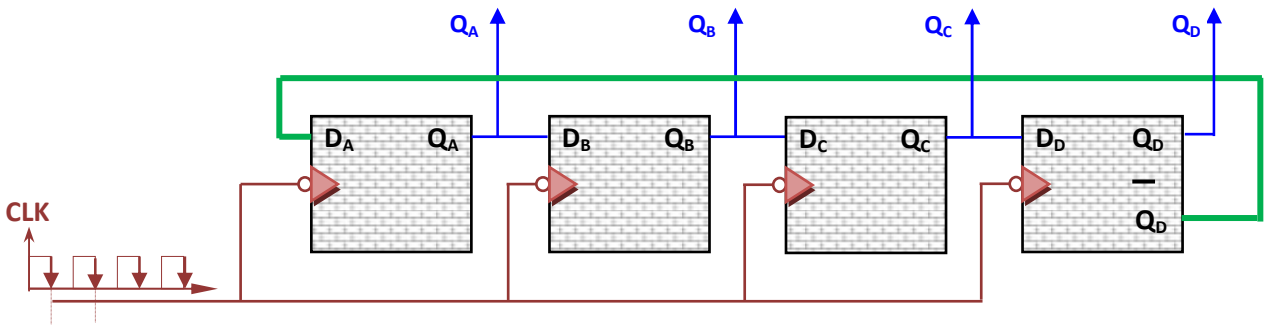


Figure 1.19 : *Compteur Johnson à 04 bits à base des bascules D.*

1.6.3. Compteur en anneau

Compteur en anneau est un compteur, où chaque état est représenté par une bascule, dont la sortie de la dernière bascule est retro-couplée vers l'entrée de la première bascule.

Exemple : Compteur en anneau à 04 bits → 4 états, comme indique-la table ci-dessous.

Table 1.10 : *Tables des états d'un compteur en anneau à 04 états.*

	Etats
Top d'horloge	$Q_AQ_BQ_CQ_D$
0	**1000**
1	0100
2	0010
3	0001

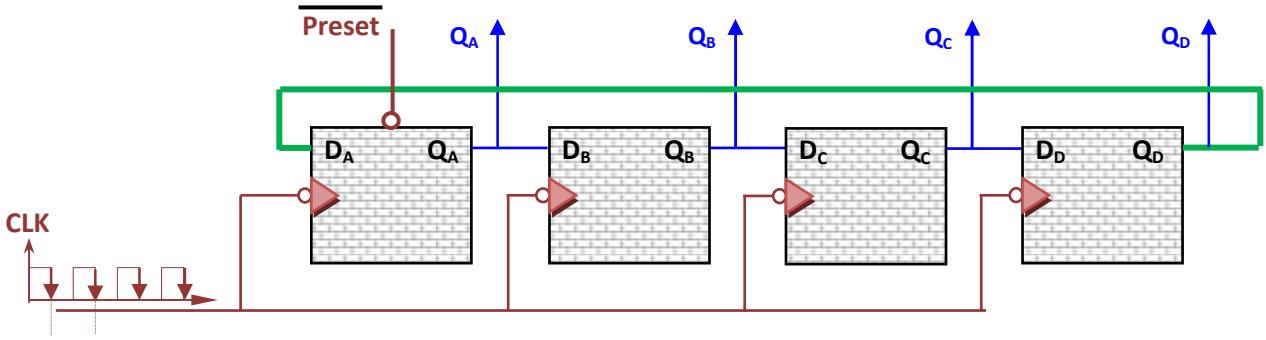


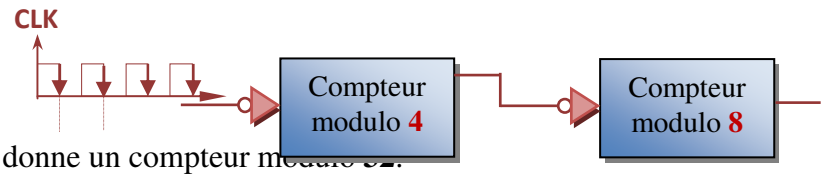
Figure 1.20 : *Compteur en anneau à 04 états à base des bascules D.*

Problème : ***Comment obtenir un compteur ayant un modulo plus élevé ?***

Solution : La mise en cascade des compteurs permet d'obtenir un modulo plis élevé. Le modulo total des compteurs montés en cascade est égal au produit des modulos individuels.

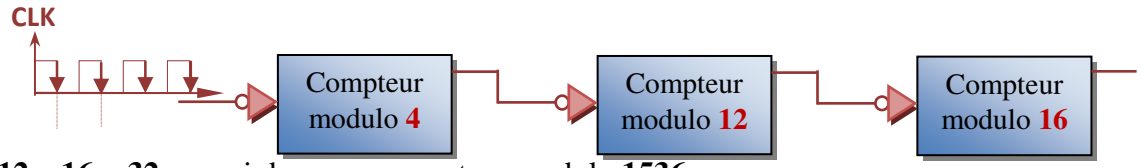
Exemples :

- ✓ **Compteur modulo 32**



4 x 8 = 32, ce qui donne un compteur modulo **32**.

- ✓ **Compteur modulo 1536**



4 x 12 x 16 = 32, ce qui donne un compteur modulo **1536**.

1.6.4. Exercices

EXERCICE 01 :

- Effectuer la synthèse d'un compteur synchrone modulo 6 à l'aide de bascules JK fonctionnant sur fronts descendants.
- Réaliser de la même manière un compteur asynchrone modulo 5.
- Réaliser un compteur asynchrone avec des bascules JK fonctionnant sur fronts montants réalisant le cycle suivant :

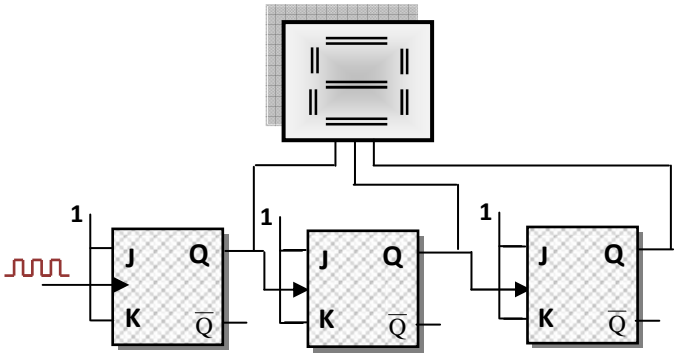
0, 1, 2, 3, 7, 8, 9, 10, 15, 0, …

EXERCICE 02 :

Pour afficher l'état de tels compteurs, prenons l'affichage à sept segments à décodeur incorporé. Quelle est la fréquence maximale de l'utilisation.

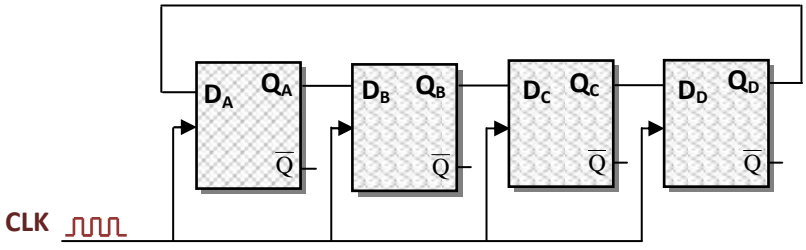
T_P = 50 ns, Le temps de propagation dans une bascule.

T_d = 100 ns, Le temps de décodage.



EXERCICE 03 :

- Etant donné le compteur en anneau de la figure ci dessous. Etablir sa table de transitions et montrer qu'il possède quatre états distincts.
- Représenter l'allure des signaux Q_A, Q_B, Q_C et Q_D en fonction du temps (CLK : signal d'horloge de forme carré et de période *T*). En supposant que l'état initial $(Q_AQ_BQ_CQ_D)$= (1000).

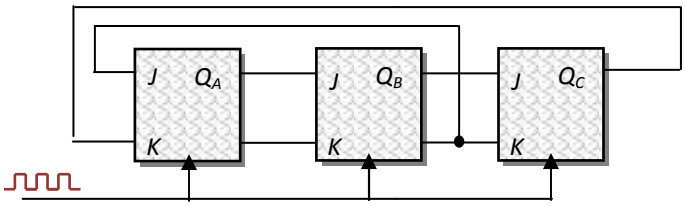


- Sachant que les bascules D utilisées possèdent des entrées de mise à 1 (Preset) et de remise à 0 (Clear). Comment peut-on démarrer ce compteur à partir de l'état $(Q_AQ_BQ_CQ_D)$= (0010).
- Pour un compteur en Anneau modulo N, donner le nombre de bascules nécessaires. Comparer ce nombre avec celui obtenu pour un compteur binaire normal.

- Que devient ce compteur lorsque la sortie $\overline{Q_D}$ est reliée à l'entrée D de la première bascule. Représenter l'allure des signaux Q_A, Q_B, Q_C et Q_D et CLK à partir de l'état initial $(Q_AQ_BQ_CQ_D)= (0000)$.

EXERCICE 04 :

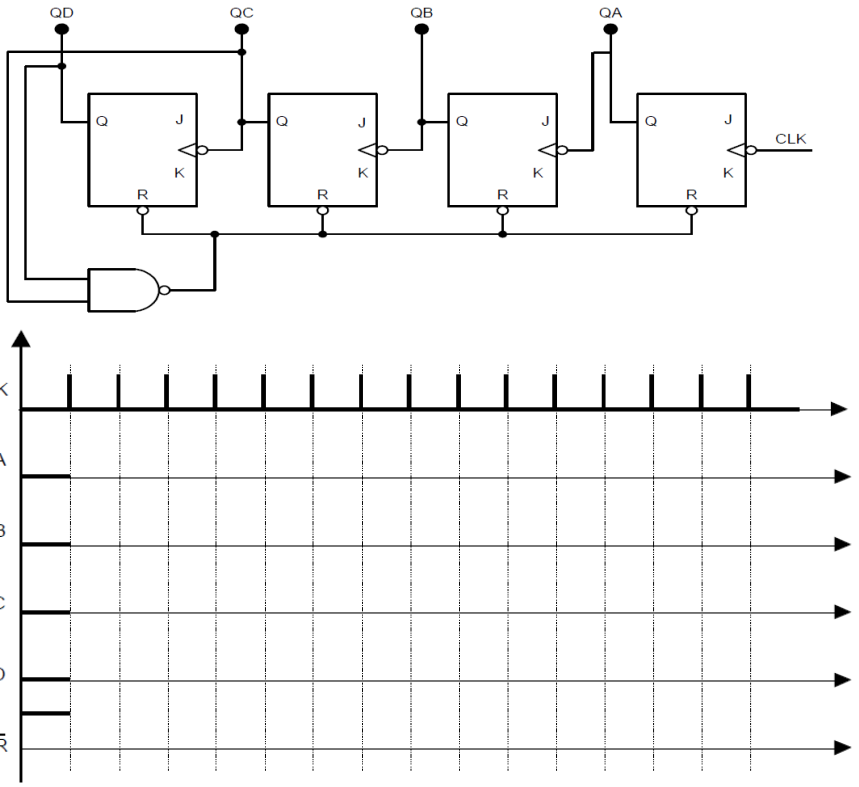
Soit le compteur suivant :



- Donner la séquence de ces états à partir de l'état 101.
- Même question à partir de l'état 000.

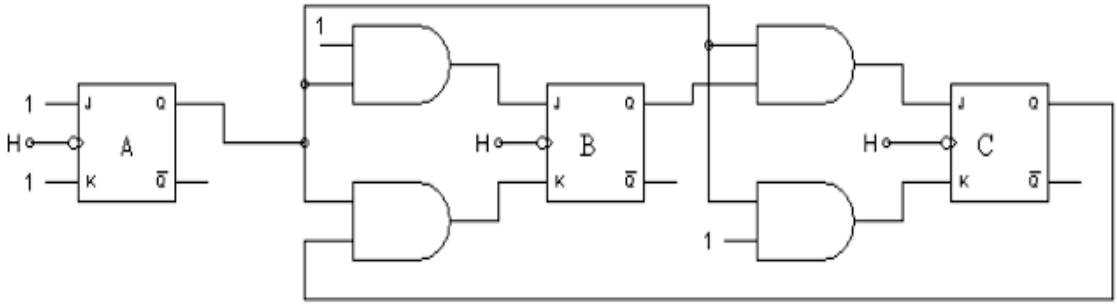
EXERCICE 05 :

Dessinez les formes d'onde demandées suite à l'analyse de la figure suivante :



EXERCICE 06 :

Soit le montage suivant :

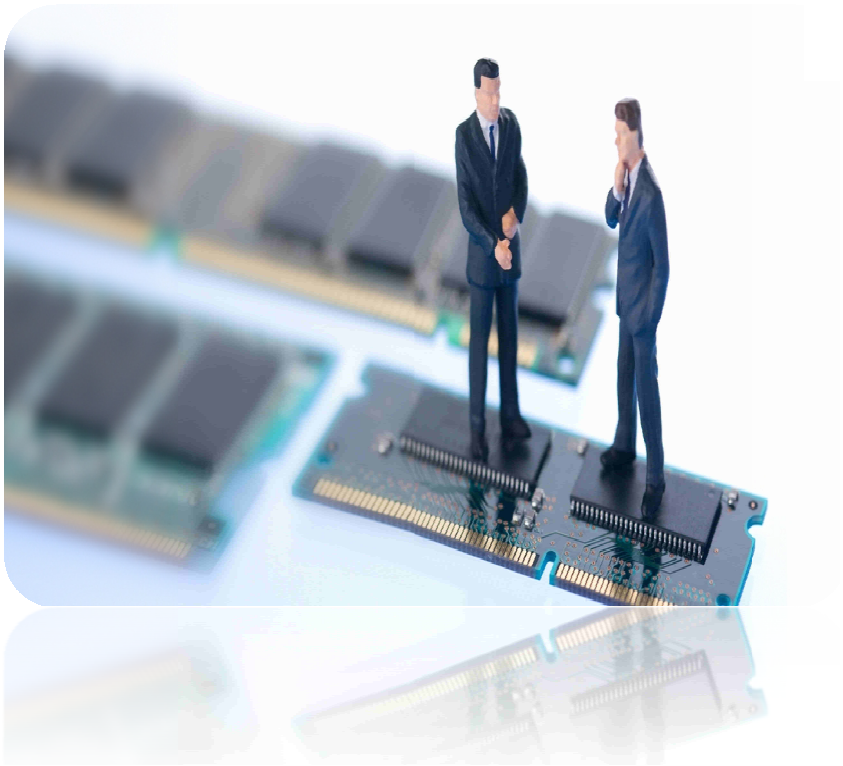


- Donnez les équations des entrées J et K des 3 bascules.

- On suppose que le compteur par de l'état $Q_C Q_B Q_A$ = 000. Tracez les chronogrammes de l'horloge H et des sorties Q_A, Q_B et Q_C.
- Déterminez le modulo de ce compteur, la fréquence fI (I = A, B ou C) et le rapport cyclique aI pour Q_A, Q_B et Q_C,

CHAPITRE **2**

Les mémoires à semi-conducteurs



CHAPITRE 2

LES MEMOIRES A SEMI-CONDUCTEURS

2.1. Définitions

Une mémoire est un dispositif (***Electronique*** – ***Magnétique***) capable d'**enregistrer,** de **conserver 'emmagasiner'** puis de **restituer** une **information** binaire (à la demande). L'unité d'information (**bit**, **octet**, etc.) s'appelle '**point mémoire**' ou '**cellule**'.

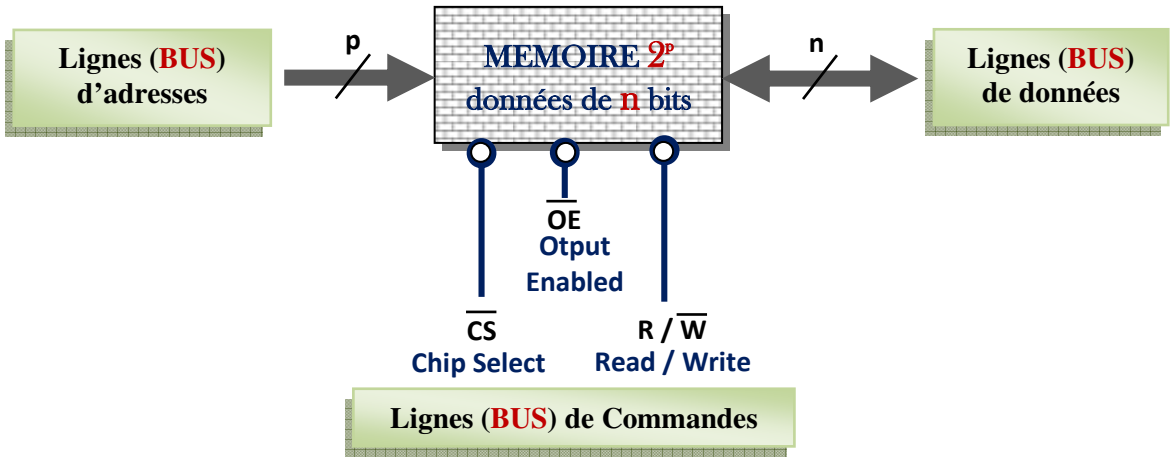


Figure 2.1 : *Schéma fonctionnel d'une mémoire.*

- ✓ **L'adresse** fournie par le bus d'adresses est le mot binaire de **p bits** qui permet de **localiser** la donnée.
- ✓ **La donnée** de **n bits** entre (**écriture**) et sort (**lecture**) par le bus de données qui est bidirectionnel : deux sens possibles, en liaison avec le signal R/W.
- ✓ La mémoire peut stocker $\mathbf{2^p}$ **données** de **n bits** chacune.
- ✓ Le signal **CS** permet la sélection du circuit.
- ✓ Le signal OE qui permet de mettre en haute impédance (validation des données de sorties).

La mémoire est un arrangement linéaire d'un ensemble de cases mémoires. Chaque case à une taille mesurée par le nombre de **bits** (**registre mémoire**). A chaque case est associé un nombre propre: **Son adresse**. Donc, la capacité mémoire est le nombre de cases mémoires, exprimée par le **Kilo** (case). Généralement, une **case = 1 octet (8 bits), 2 octets (16 bits), ...**.

Exemple: 1 case = 1 octet = 1 byte.

- 1 Kilo Octet (**KO**) = 2^{10} = 1024 octets.
- 1 Méga Octet (**MO**) = 1 KO x 1 KO = 2^{20} octets.
- 1 Giga Octet (**GO**) = 1 MO x 1 KO = 2^{30} octets.
- 1 Téra Octet (**TO**) = 1 GO x 1 KO = 2^{40} octets.

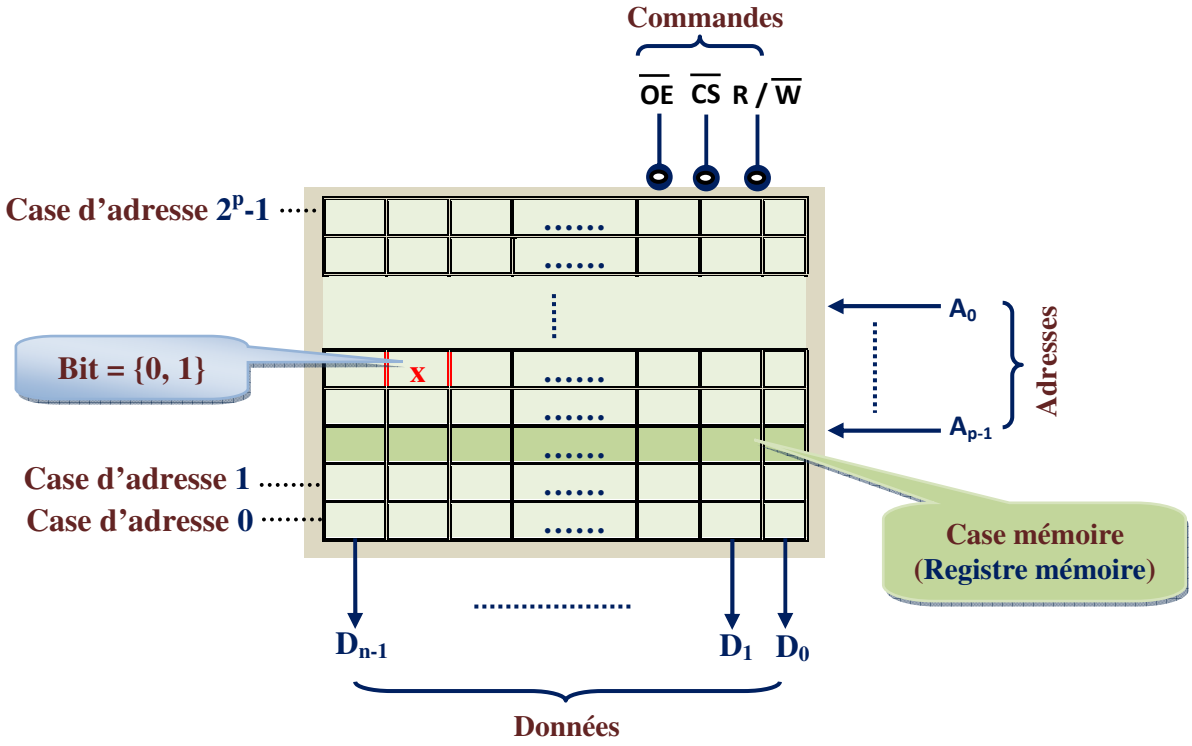


Figure 2.2 : *Tableau de mémoire de taille* $\mathbf{2^p}$ *cases.*

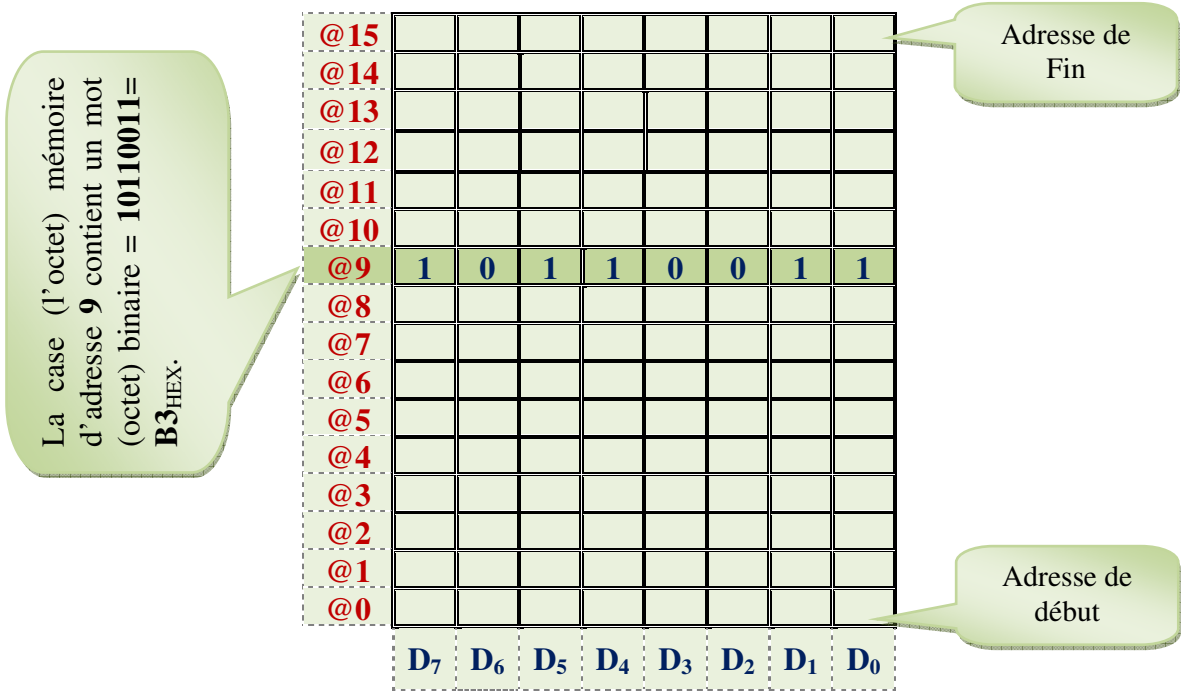


Figure 2.3 : *Tableau de mémoire de taille* 16 *octets.*

Table 1.1 : *Exemples des mémoires et leurs adressages.*

Nombre de lignes d'@	Taille mémoire	Adresses	
		Début	Fin
2	2^2 = 4 octets	0_{HEX}	3_{HEX}
4	2^4 = 16 octets	0_{HEX}	F_{HEX}
8	2^8 = 64 octets	00_{HEX}	FF_{HEX}
16	2^{16} = 64 Ko	0000_{HEX}	$FFFF_{HEX}$
20	2^{20} = 1 Mo	00000_{HEX}	$FFFFF_{HEX}$
32	2^{32} = ….	000000_{HEX}	$FFF0FF_{HEX}$
64	2^{64} = ….	---------	----------

On peut donc utiliser une mémoire soit en : **Lecture** soit en **Ecriture**, selon le principe de la figure ci-dessous..

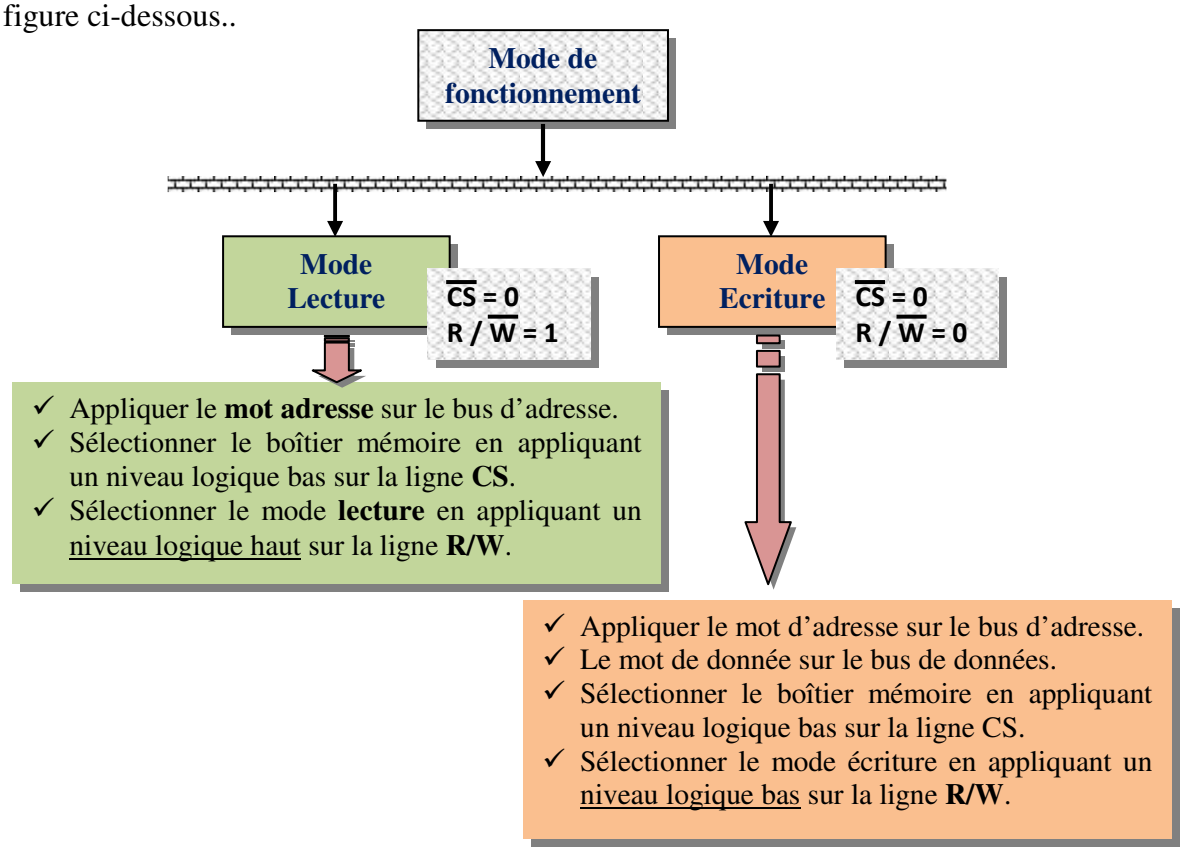


Figure 2.4 : *Mode de lecture/écriture des mémoires.*

2.2. Caractéristiques générales des mémoires

A partir des définitions précédentes, on peut résumer les caractéristiques des mémoires sur la figures ci-dessous.

- **La capacité :** C'est le nombre d'informations que peut contenir la mémoire. Elle s'exprime en **bits** ou en **mots** de n bits. Par exemple :
  - 1 Kilo Octet (**KO**) = 2^{10} = 1024 octets.
  - 1 Méga Octet (**MO**) = 1 KO x 1 KO = 2^{20} octets.
  - 1 Giga Octet (**GO**) = 1 MO x 1 KO = 2^{30} octets.
  - 1 Téra Octet (**TO**) = 1 GO x 1 KO = 2^{40} octets.

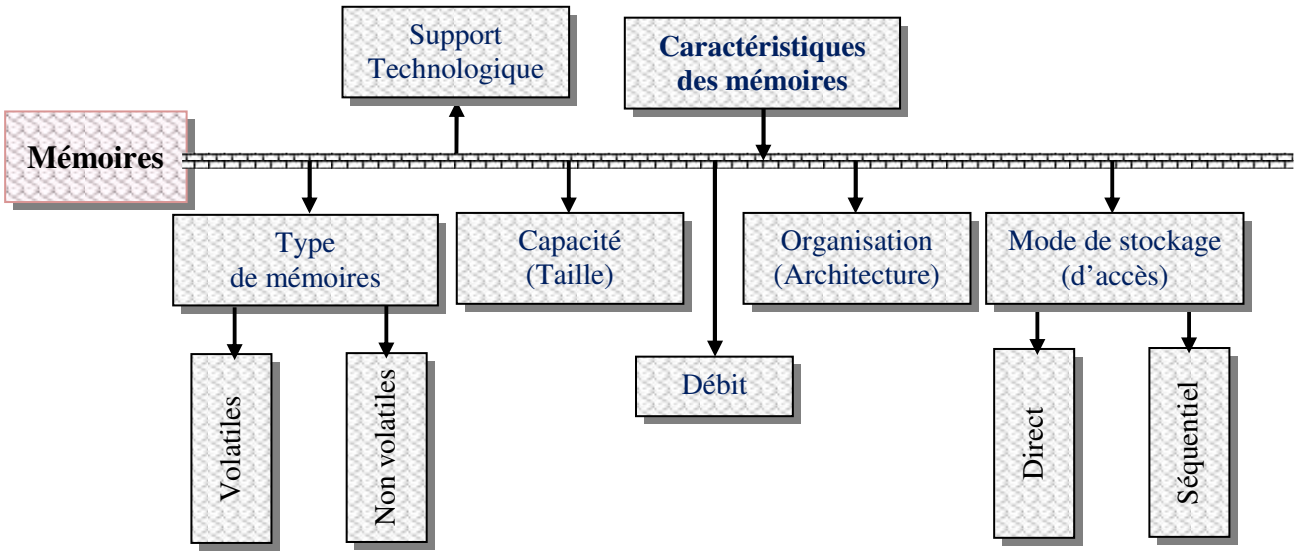


Figure 2.5 : *Caractéristiques des mémoires.*

- **L'organisation :** Elle définit le nombre de mots (cases) et la longueur de chaque mot. Par exemple :
  - ✓ Une mémoire de 64Kx 1 est constituée de 65536 mots de 1 bit. Sa capacité est donc de 64Kb (8Ko) ;
  - ✓ Une mémoire de 8Kx8 contient 8192 mots de 8 bits. Sa capacité est de 64Kb (8Ko) ;

Exemple: Soit une mémoire ayant une capacité de 8 KO.

1. Combien a-t-elle de lignes de données ?
2. Combien a-t-elle de lignes d'adresses ?
3. Quelle est sa capacité en octets ?

Réponse :

1. Cette mémoire possède 8 lignes de données.
2. Cette mémoire stocke 8KO =8x1024=8192 mots. Il y a donc 8192 cases mémoires. Puisque 8192 = 213, il faut donc 13 lignes d'adresse.
3. La capacité de cette mémoire est de 8192 octets.

- **Le temps d'accès:** C'est le temps qui s'écoule entre une demande d'information et le moment où elle est effectivement disponible sur le bus de données. En outre, c'est le temps s'écoulant entre le lancement d'une opération de lecture/écriture et son accomplissement.

- **Débit :** C'est le nombre d'informations lues ou écrites par seconde. Exemple : 300Mo/sec.

- **Volatilité :** Conservation ou disparition de l'information dans la mémoire hors alimentation électrique de la mémoire (RAM, ROM).

- **Méthodes d'accès :** Trois types d'accès, direct et séquentiel et semi-séquentiel.
  - ✓ **Accès séquentiel :**
    - Pour accéder à une information on doit parcourir toutes les informations précédentes.
    - Accès lent.
    - **Exemple :** Bandes magnétiques (K7 vidéo).
  - ✓ **Accès direct :**
    - Chaque information a une adresse propre.

- On peut accéder directement à chaque adresse.
- **Exemple** : mémoire centrale.

✓ **Accès semi-séquentiel :**

- Intermédiaire entre séquentiel et direct.
- **Exemple** : Disque dur, Accès direct au cylindre, Accès séquentiel au secteur sur un cylindre.

Support technologique : Trois grandes familles

- **Mémoires à semi-conducteur** : L'élément mémoire est construit autour des transistors (bipolaire ou MOS-FET) → Mémoires à accès plus rapide.
- **Mémoires à support magnétique :** Stockage d'un grand nombre d'information (Bandes magnétiques) → Accès lourd.
- **Mémoires à support magnétique :** Lecture avec LASER d'une piste gravée (CD, DVD) → Accès rapide.

2.3. Différents types de mémoires à semi-conducteurs

Deux grandes familles

- **Mémoires non volatiles** : **ROM** (Read Only Memory) dites mémoires **mortes**.
  - Leur contenu est fixe (ou presque ...) - Conservé en permanence.
- **Mémoires volatiles** : **RAM** (Random Access Memory) dites mémoires **vives**.
  - Leur contenu est modifiable - Perte des informations hors alimentation électrique- Accès sans contraintes.

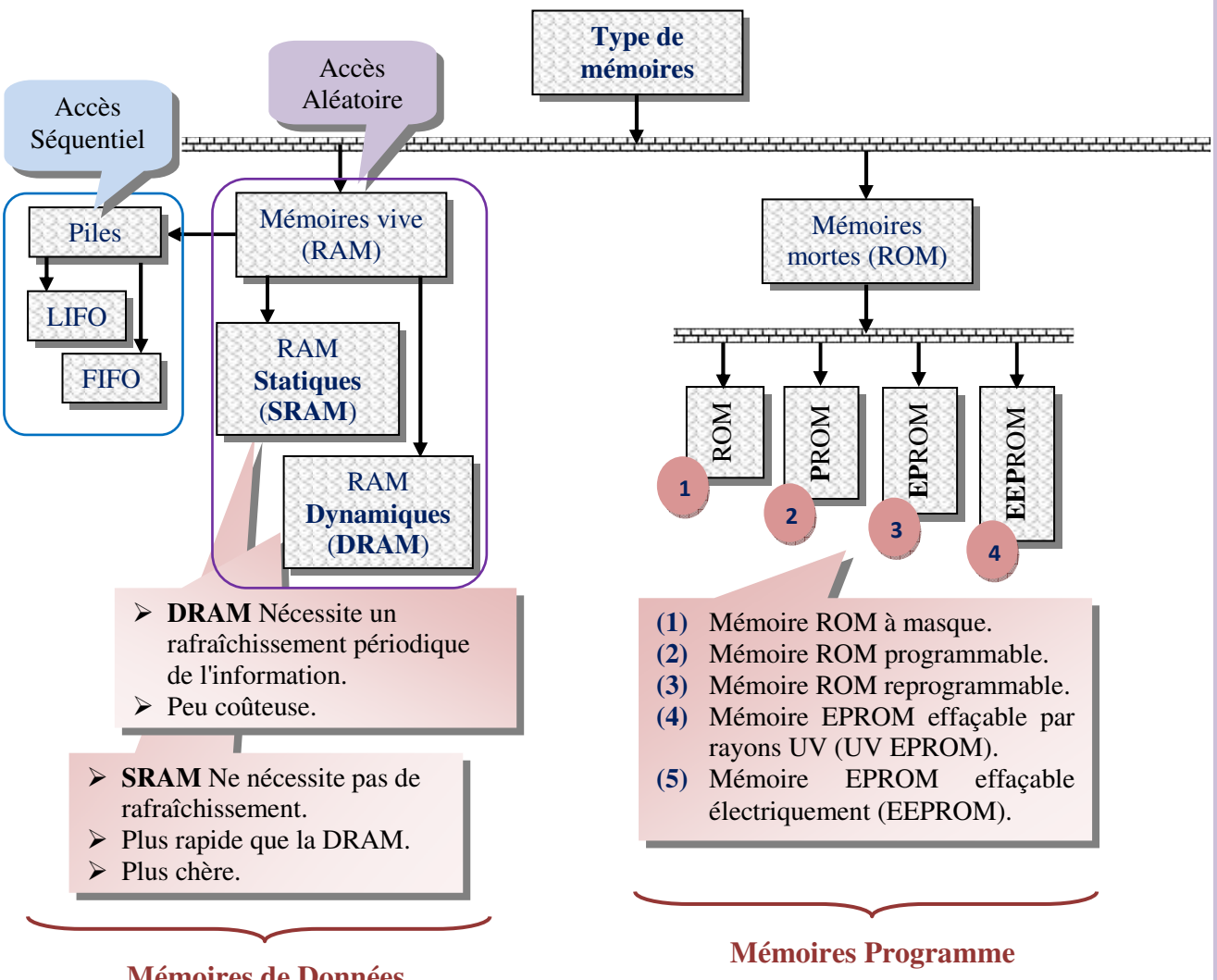
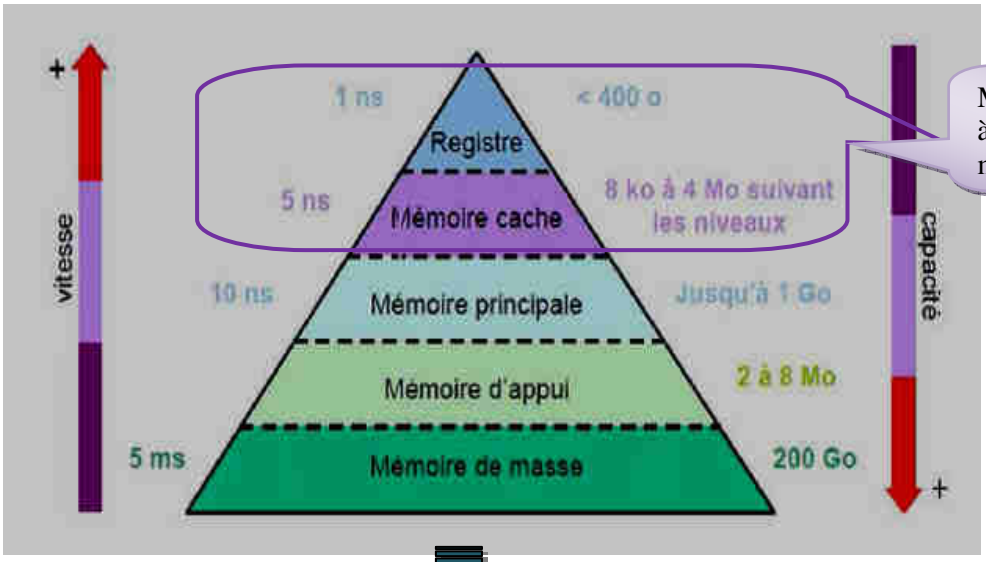


Figure 2.6 : *Familles des mémoires mortes et vives.*

Remarque : On remarque la relation oppositionnelle entre la vitesse et la capacité mémoire, comme illustre la figure ci-dessous. La taille augmente conduit à une application spéciale : Registre, mémoire cache, mémoire principale, …etc. De plus, la vitesse (le temps d'accès) augmente.



- Les **registres** sont les éléments de mémoire les plus rapides. Ils sont situés au niveau du processeur et servent au stockage des opérandes et résultats intermédiaires.
- La **mémoire cache** est une mémoire rapide de faible capacité destinée à accéléré l'accès à la mémoire centrale en stockant les données les plus utilisées.
- La **mémoire centrale** contient les programmes (code + données) et est plus lente que les deux mémoires précédentes.
- La **mémoire d'appui** est l'équivalent de la mémoire cache pour la mémoire de masse.
- La **mémoire de masse** est un support de stockage généralement de grande capacité et sert au stockage et à l'archivage des informations.

Figure 2.7 : *Relation oppositionnelle entre vitesse et capacité.*

On peut également donner une autre classification qui tient compte de l'éloignement par rapport au processeur (à l'intérieur ou à l'extérieur de microprocesseur) :

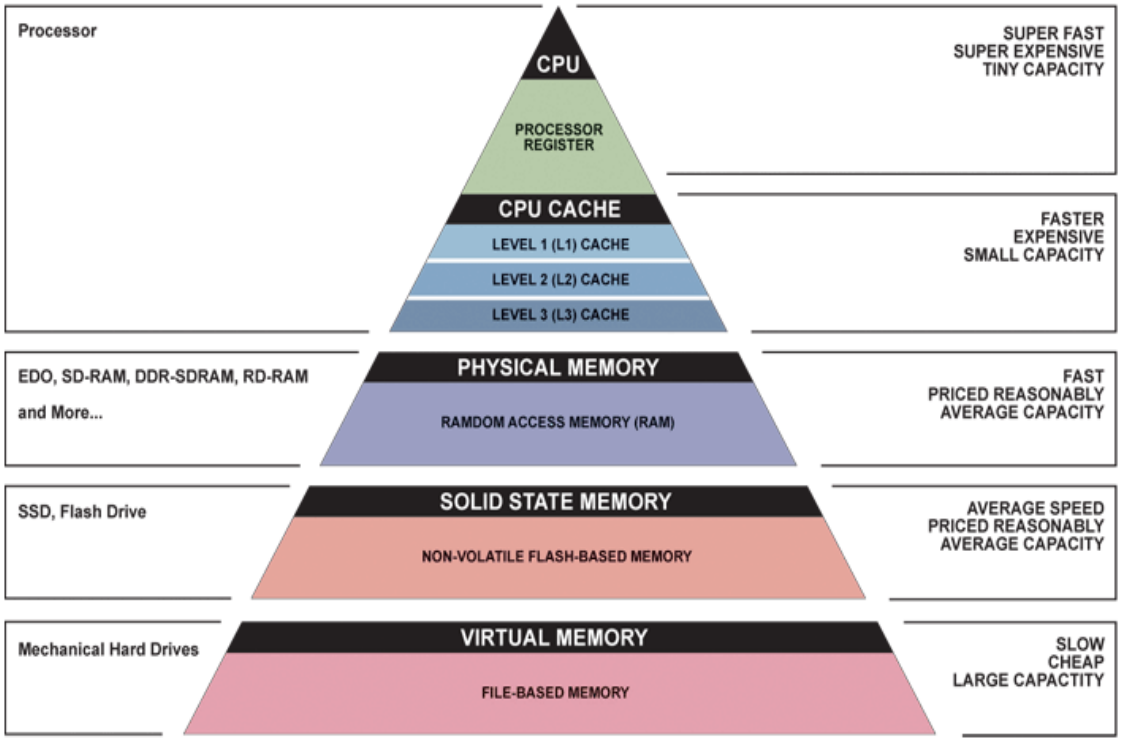


Figure 2.8 : *Classement des mémoires en fonction de l'éloignement par rapport au processeur.*

Classement logiciel des mémoires

On peut aussi classer les mémoires par leur utilisation ou leur contenu: **mémoire programme et mémoire données.**

Mémoire programme

- La mémoire programme contient la liste ordonnée des instructions à traiter par le processeur, c'est à-dire le **programme**. Elle sauvegarde les informations nécessaires au fonctionnement général du système dans un programme spécial appelé moniteur (gestion des différents périphériques et mémoires ainsi que les informations de transfert et de traitements des données).
- *Dans un système à microprocesseur la mémoire programme est en général de type ROM ou PROM.*

Mémoire données

- La mémoire **données** sauvegarde les entrées en provenance des périphériques d'entrées (capteurs, claviers, lecteur de disque, ...), les résultats intermédiaires de calcul du processeur, les données de sortie en partance vers les périphériques de sorties (afficheurs, écran vidéo, enregistreur de disque, ...).
- *La mémoire données est obligatoirement une mémoire de type RAM puisque la lecture et l'écriture y sont nécessaires en permanence.*

2.3.1. La mémoire vive : RAM

La mémoire vive ou RAM (***R**andom **A**ccess **M**emory* : *mémoire à accès aléatoire)*, est une mémoire **volatile**, cela signifie que si l'on coupe l'alimentation, les données qu'elle contient sont perdues. Ses principales caractéristiques sont:

- Elle sert à stocker les programmes exécutés par le microprocesseur ainsi que les données utilisées par ce microprocesseur.
- Elle est accessible en **lecture** et en **écriture**.
- Elle est organisée sous forme matricielle.

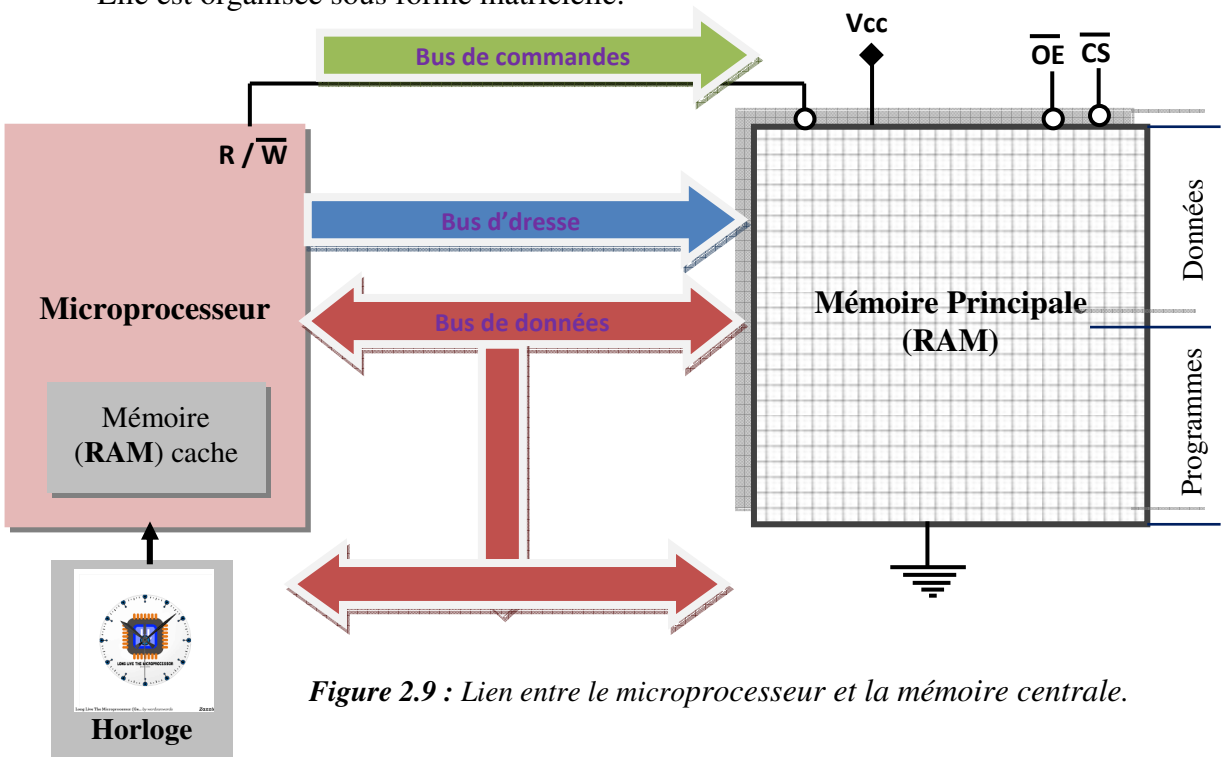


Figure 2.9 : *Lien entre le microprocesseur et la mémoire centrale.*

Algorithme 2.1 : *Principes d'accès à la mémoire RAM.*

1. Mettre sous l'alimentation le montage ;
2. Sélection du boitier $\overline{CS} = 0$;
3. Activer (connaitre) le type de l'opération :
   a. Lecture $R / \overline{W} = 1$;
   b. Ecriture $R / \overline{W} = 0$;
4. Connaitre l'adresse de la donnée à lire (**récupérer**) ou à écrire (**sauvegarder**) ;
5. Activation des sorties dans le cas de lecture.
6. Aller à l'opération suivante.

Selon la structure technologique de l'élément mémoire (Bit), il existe deux grandes familles de mémoires vives :

- RAM Statiques : **SRAM**.
- RAM Dynamiques : **DRAM**.

2.3.1.1. Organisation des mémoires RAM

Une mémoire est constituée de cellules mémoires (**bascules bistables**) qui sont regroupées pour former des cases mémoires.

- ✓ Chaque case mémoire est identifiée par une adresse unique obtenue par la combinaison binaire d'un ensemble de fils constituant le bus d'adresse.
- ✓ Cette adresse est généralement exprimée en Hexa.
- ✓ L'ensemble de fils permettant de lire ou d'écrire sur chaque cellule mémoire forme le **bus de données**.
- ✓ Lorsque le bus d'adresse est composé de n fils, la taille de la mémoire est $T=2^n$ cases ou mots.

Un composant mémoire est donc formé de :

- Une matrice composée de 2^p lignes et de **n** colonnes.
- A chaque intersection (ligne, colonne) on trouve des bascules (**SRAM**), ou des condensateurs-transistors (**DRAM**).

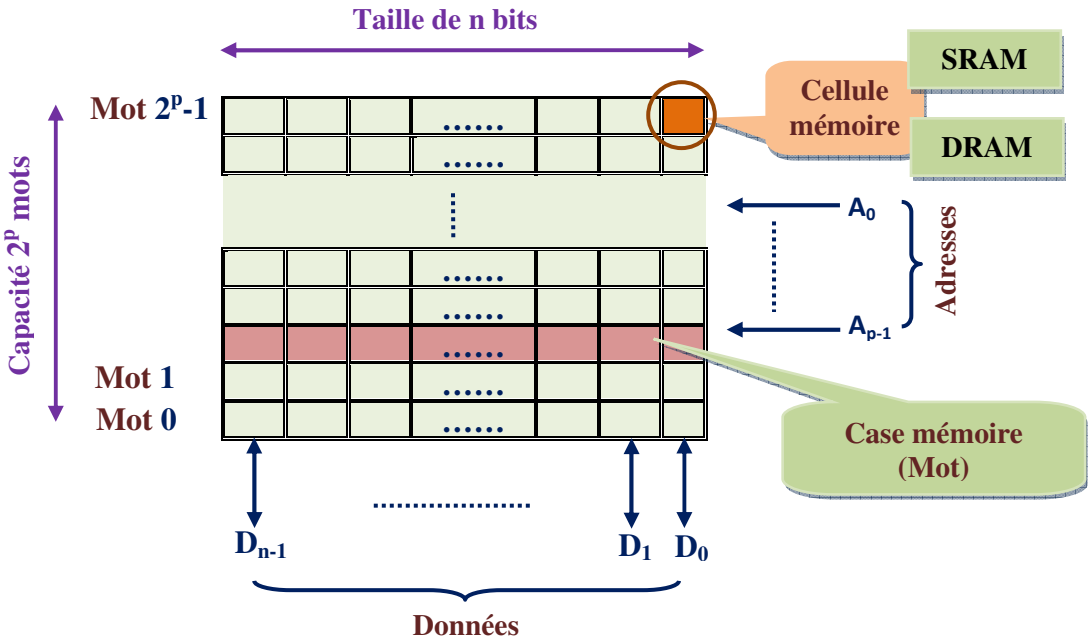


Figure 2.10 : *Organisation interne de base d'une mémoire RAM.*

a). Fonctionnement

Les boîtiers mémoires disposent d'un certain nombre de connections externes leur permettant d'être reliés ensemble sur un même bus de donnée et de pouvoir accéder à ce bus indépendamment des autres.

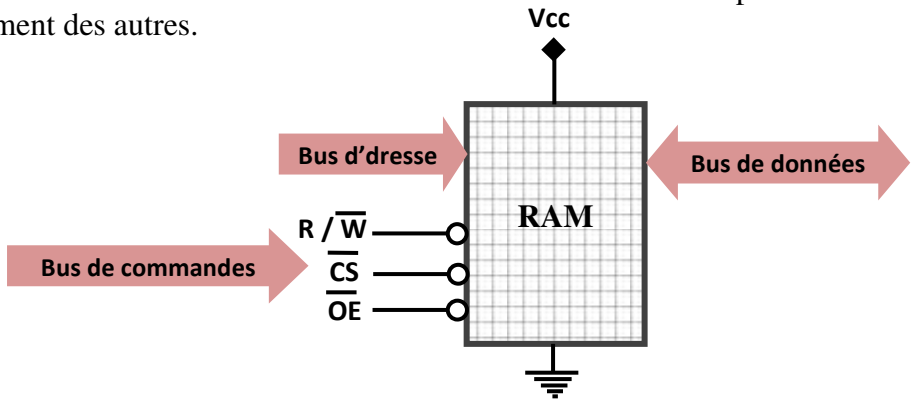


Figure 2.11 : *Schéma d'un circuit mémoire.*

✓ L'entrée notée **CE** (Chip Enable) ou **CS** (Chip Select), ***souvent complémentée***, permet de déconnecter électroniquement la mémoire du bus de données (bus de données à l'état haute impédance (**Hiz**) (circuit à 03 états)).

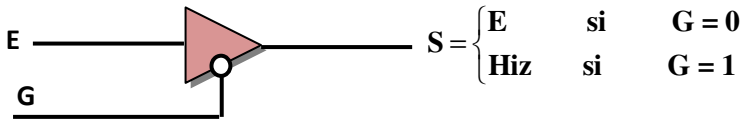


Figure 2.12 : *Circuit (buffer) à 03 états.*

- ✓ L'entrée R/W permet d'autoriser l'écriture ou la lecture de la mémoire.
  - Si $R/\overline{W} = 0$ une opération d'écriture est possible.
  - Si $R/\overline{W} = 1$ une opération de lecture est possible.
- ✓ Eventuellement les signaux de commande $\overline{RAS}$ et $\overline{CAS}$ pour les mémoires RAM dynamiques.

Lire une donnée mémoire consiste à

1. La sélectionner d'abord le boitier mémoire ;
2. Placer sur le bus d'adresse les coordonnées de la case mémoire;
3. Transférer son contenu sur le bus de données externe.

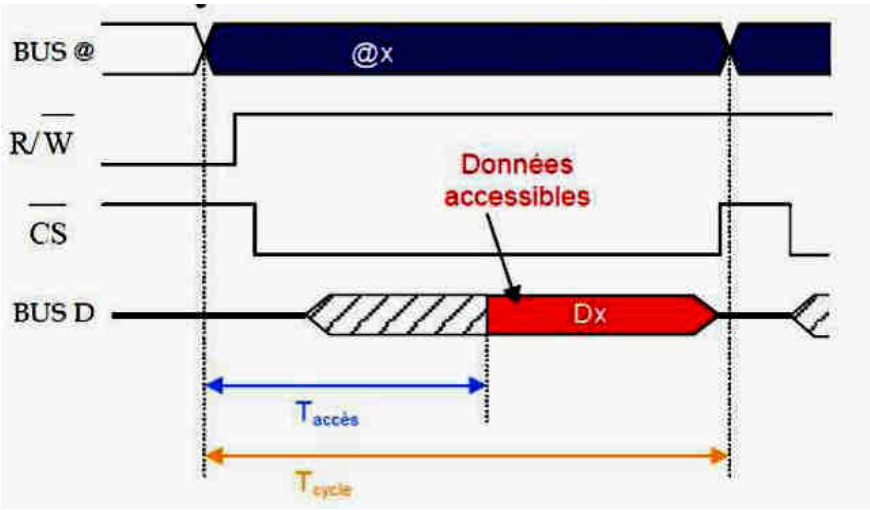


Figure 2.13 : *Chronogramme de lecture d'une mémoire RAM.*

Ecrie une donnée dans la mémoire consiste à

1. La sélectionner d'abord le boitier mémoire ;
2. Placer sur le bus d'adresse les coordonnées de la case mémoire à laquelle sera enregistrée l'information;
3. Transférer le contenu du bus de données externe à l'adresse sélectionnée.

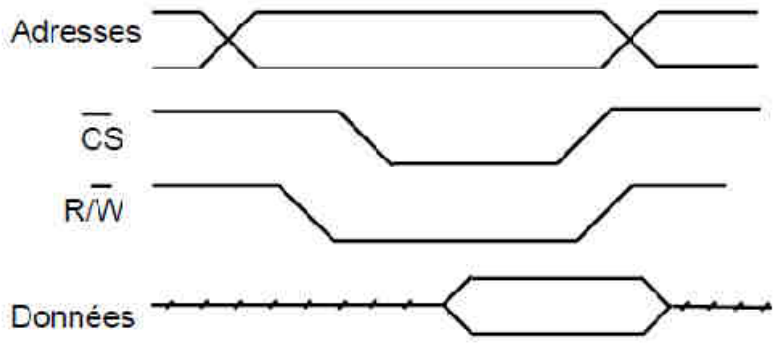


Figure 2.14 : *Chronogramme D'écriture dans la mémoire RAM.*

b). Structure physique de la mémoire

La mémoire est organisée sous la forme d'une grille (matrice) dont chaque nœud (**cellule**) correspond à un bit (bascule) matérialisé par des transistors et des condensateurs. Il y a deux technologies de fabrication des RAM : statiques (**SRAM**) et dynamiques (**DRAM**), elles ont chacune leur domaine d'application.

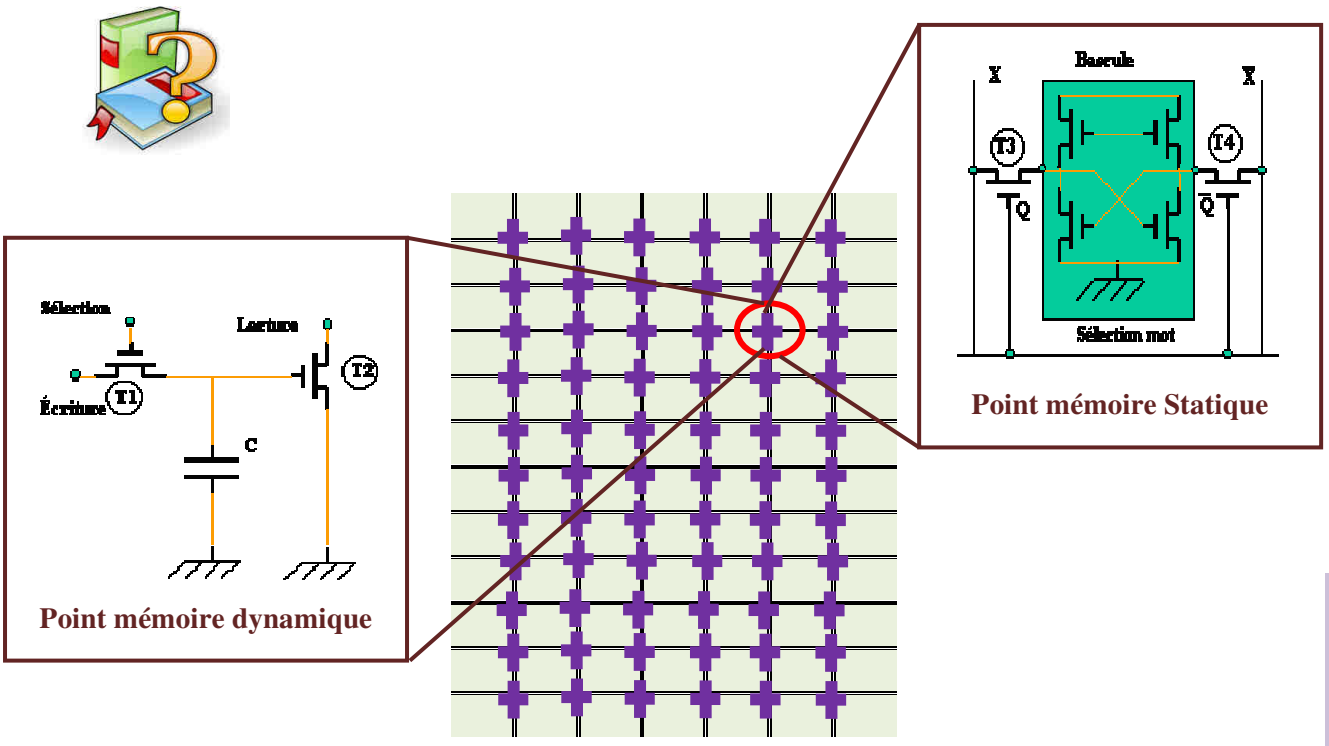


Figure 2.15 : *Configuration du tableau (grille ou matrice) de la mémoire.*

- ✓ Dans la **SRAM** (plus ancienne), les bits y sont mémorisés par des bascules électroniques dont la réalisation nécessite six transistors par bit à mémoriser.
  - Les informations y restent mémorisées tant que le composant est sous tension.
  - Les cartes mères utilisent une SRAM construite en technologie CMOS et munie d'une pile pour conserver de manière non volatile les données de configuration (*setup*) du BIOS.
  - Le circuit de cette RAM CMOS est associé au circuit d'horloge qui lui aussi a besoin de la pile pour fonctionner en permanence même quand l'ordinateur est éteint.
  - La SRAM est très rapide et est pour cette raison le type de mémoire qui sert aux mémoires cache.

- ✓ Dans la **DRAM** (réalisation beaucoup plus simple que la SRAM), où chaque bit d'une DRAM est mémorisé par une charge électrique stockée dans un petit condensateur.
  - Ce dispositif offre l'avantage d'être très peu encombrant mais a l'inconvénient de ne pas pouvoir garder l'information longtemps.
  - Le condensateur se décharge au bout de quelques millisecondes (ms).
  - Aussi pour ne pas perdre le bit d'information qu'il contient, il faut un dispositif qui lit la mémoire et qui la réécrit de suite pour recharger les condensateurs.
  - On appelle ces RAM des RAM dynamiques car cette opération de **rafraîchissement** doit être répétée régulièrement.

Quatre blocs principaux, intervient dans le fonctionnement adéquat d'une mémoire RAM, comme indiqué dans le tableau ci-dessous.

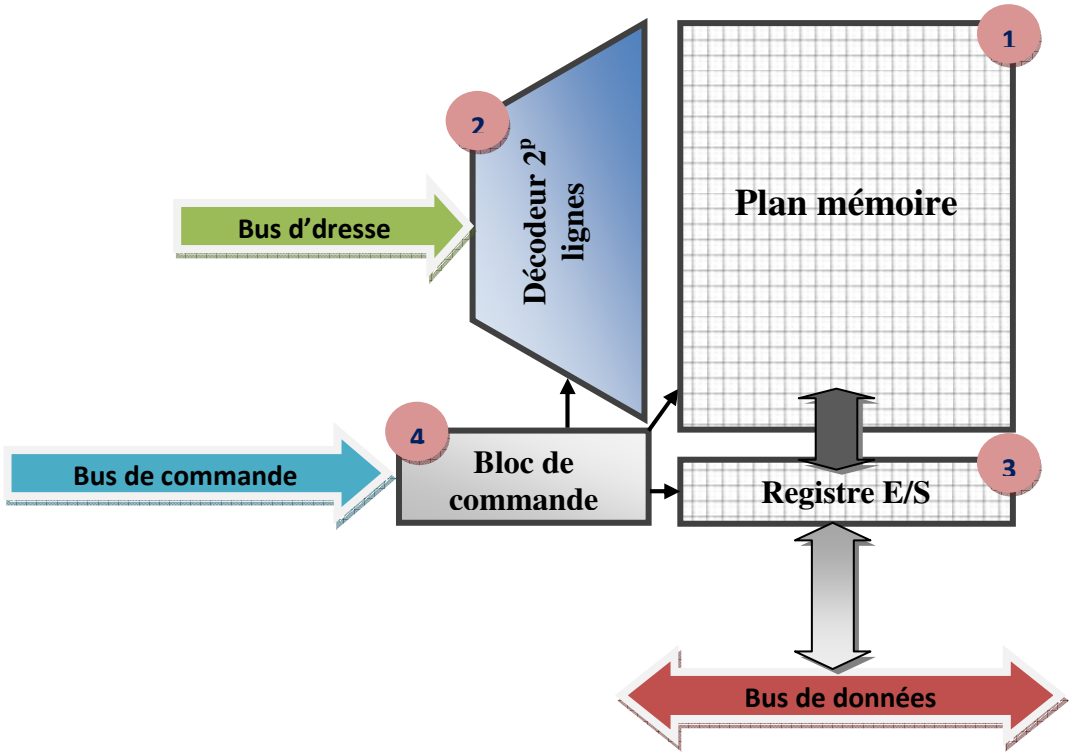


Figure 2.16 : *Configuration en bloc de la mémoire.*

Table 2.2 : *Description des blocs fonctionnels d'une RAM.*

Bloc	Nom du bloc	Fonction
1	Plan mémoire	Des points mémoires organisés en 2^p mots mémoires, dont leurs adresses vont de 0 à 2^p **-1** (p est le nombre des lignes d'adresse).
2	Décodeur	Gère la sélection de la ligne.
3	Registre d'entrée/sortie	Registre bidirectionnel avec fonctionnement à 3 états.
4	Logique de commande	Reçoit les instructions d'autorisation de sortie des données (autorisation de lecture ou d'écriture).

En plus des 4 blocs, il ya trois groupes de signaux pour faire fonctionner un mémoire RAM. Le rôle de caque bus est illustré dans le tableau ci-dessous.

Table 2.3 : *Description des Bus fonctionnels d'une RAM.*

Type de BUS	Sens	Fonction	
Bus de données	Bidirectionnel	Ss taille varie en fonction de la structure mémoire (Octet, double octet, …). Transporte les informations à lire à partie d'une adresse mémoire ou à écrire dans une adresse mémoire.	
Bus d'adresses	Unidirectionnel	Constitué de **n** fils, d'où **n** varie en fonction de la capacité mémoire (Octet, double octet, …). Ce bus **pointe** vers la case mémoire à lire ou à écrire.	
Bus de commande	/	$\overline{\mathbf{CS}}$	**C**hip **S**elect : Sélection (activation) de la puce.
		$\overline{\mathbf{OE}}$	**O**utput **E**nable : Activation des sorties, ce qui permet **le multiplexage de plusieurs boitiers** sur un seul bus.
		$\mathbf{R}/\overline{\mathbf{W}}$	Sélection du mode de lecture ou d'écriture de la mémoire.
		$\overline{\mathbf{RAS}}$ / $\overline{\mathbf{CAS}}$	Signaux de commande pour les RAM Dynamiques

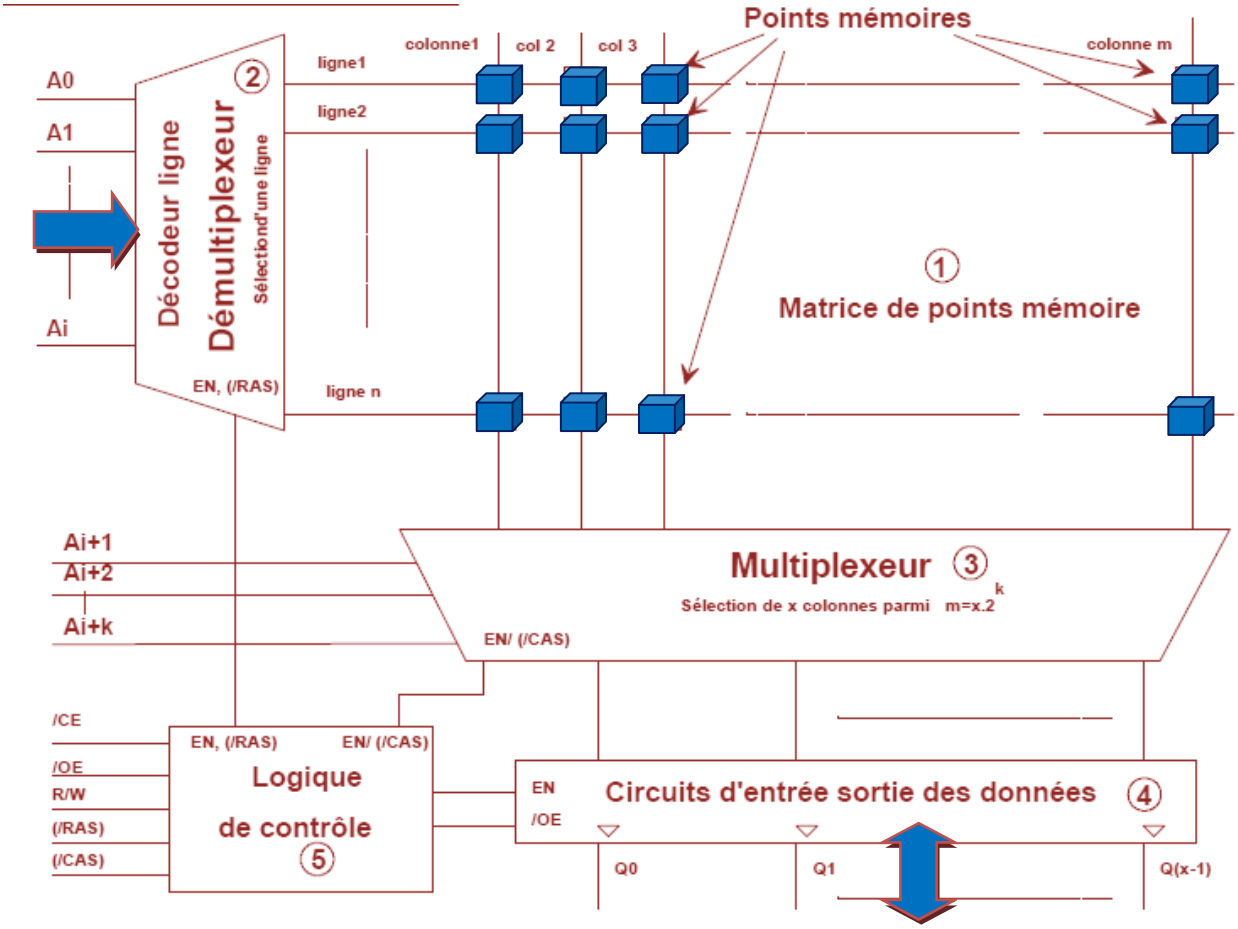


Figure 2.17 : *Organisation matricielle des mémoires à semi-conducteur (RAM).*

Exemple : Donner le type et la capacité de la mémoire schématisée ci-après:

Broche	Fonction
A_0-A_{12}	Adresses d'entrées
WE	Entrée d'écriture
CS1, CS2	Chip Select
OE	Output Enable
I/O1-I/O8	Données d'entrées/sorties
Vcc	Alimentation (+5V)

Solution :

- Le boîtier est une mémoire vive (RAM) car il possède des entrées d'écriture (WE) et de lecture (OE). Il dispose de:
  - - 13 bits d'adresse: A0-A12;
  - - 8 bits de données: I/O1-I/O8,

de sorte que sa capacité est $C=2^{>13}x8=2^{>3}x2^{>10}x8=8Ko$ ou 8 Kmots de 8 bits.

Remarque : Il existe de nombreux types de mémoires vives (format). Celles-ci se présentent toutes sous la forme de barrettes de mémoire enfichables sur la carte-mère.

1. Les premières mémoires se présentaient sous la forme de puces appelées **DIP** (Dual Inline Package).

2. Désormais les mémoires se trouvent généralement sous la forme de barrettes, c'est-à-dire des cartes enfichables dans des connecteurs prévus à cet effet. On distingue habituellement trois types de barrettes de RAM (statiques et dynamiques : **SIMM** (Single Inline Memory Module)**, DIMM** (Dual Inline Memory Module) et **RIMM** (Rambus Inline Memory Module)**.**

c). Association de mémoires

En pratique les circuits mémoires ont des caractéristiques ne leur permettant pas de satisfaire directement les exigences d'espace mémoires dans les systèmes électroniques.

- ✓ On associe donc différents circuits mémoires afin de remplir ces exigences à savoir augmenter la zone adressable et ou la largeur des mots mémoires.
- ✓ On procède pour cela soit à un agrandissement du bus de données (**association parallèle**), soit à un agrandissement du bus d'adresse (**association série**), soit les deux la fois.

Par la suite, des exemples illustrent des notions très importantes que l'on doit connaître quand on interface plusieurs circuits mémoires.

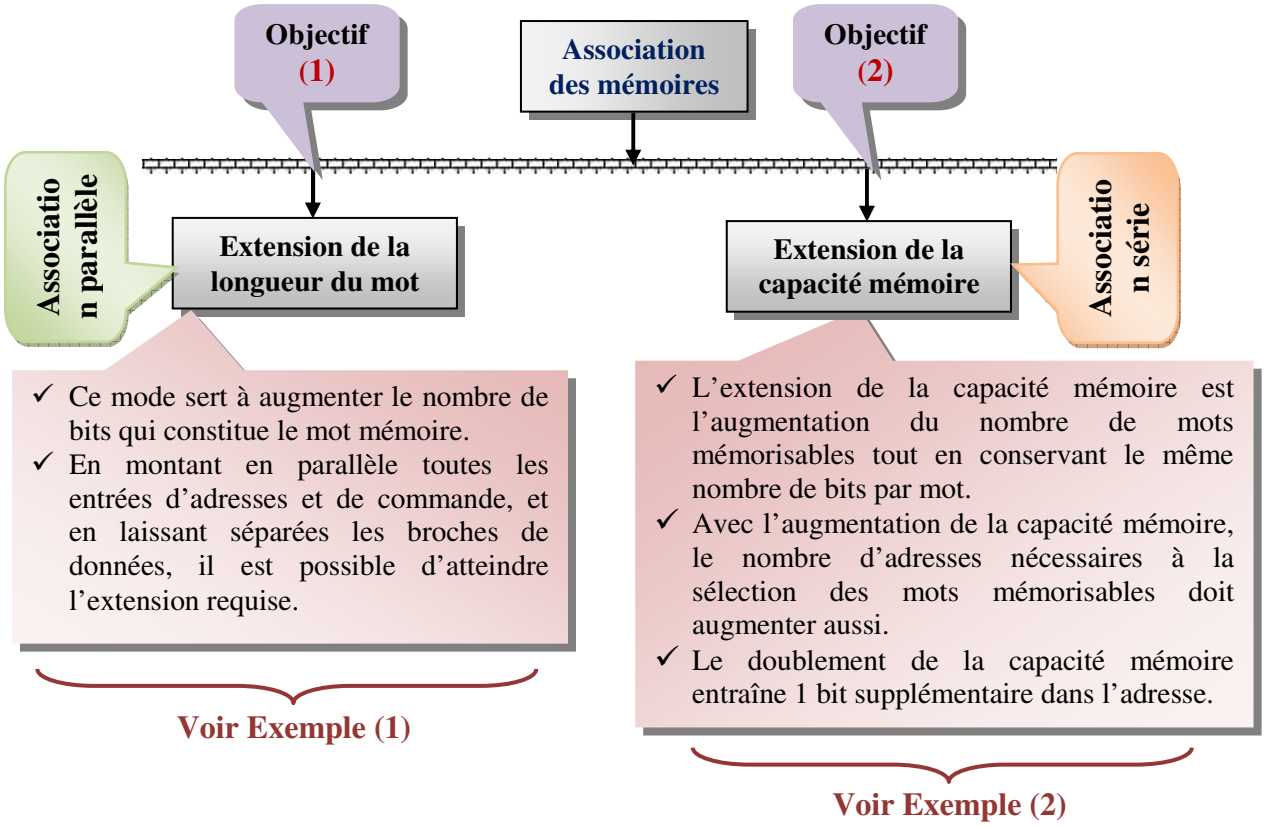


Figure 2.17 : *Association des mémoires.*

Exemple (1) : On souhaite construire une mémoire de 1K x 8 bits à partir de mémoires de 1K x 4 bits → Il est possible d'assembler deux de ces mémoires de 1K x 4 bits pour obtenir la mémoire recherchée. La figure ci-dessous montre comment on peut assembler les deux mémoires.

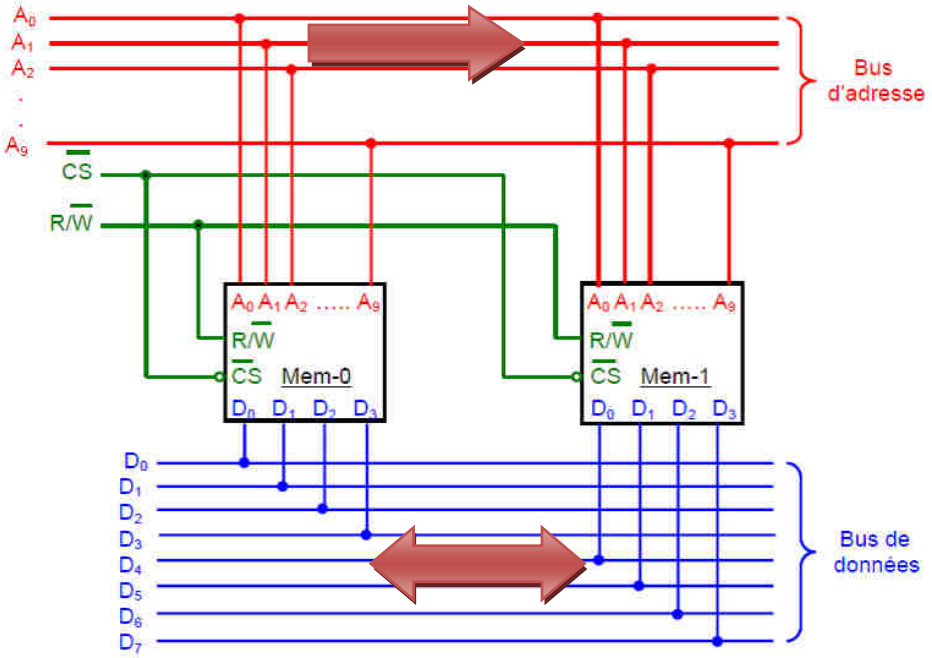


Figure 2.19 : *Extension de la longueur du mot.*

L'assemblage de deux mémoires de 1K x 4 bits est équivalent à une seule mémoire de capacité 1K x 8 bits.

Exemple (2) : Comme exemple, on veut réaliser une mémoire de 2K x 4 bits à partir de mémoires de 1K x 4 bits. → Il est possible en assemblant deux mémoires de 1K x 4 bits d'obtenir une mémoire de 2K x 4 bits, comme le montre la figure ci-dessous.

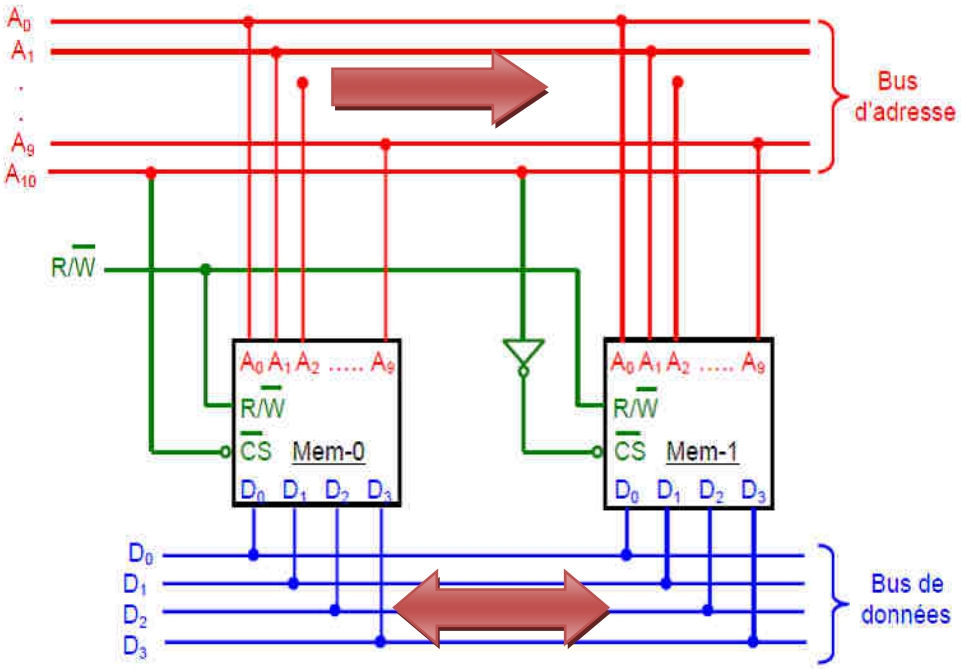


Figure 2.20 : *Extension de la capacité mémoire.*

2.3.2. La mémoire morte : ROM

Une **mémoire morte** (**ROM** ou **Read-Only Memory**) est une **mémoire non volatile**, c'est-à-dire une mémoire qui ne s'efface pas lorsque l'appareil qui la contient n'est plus alimenté en électricité.

Les mémoires mortes sont utilisées pour stocker les programmes de gestion des systèmes informatiques. A titre d'exemple,

- Les informations nécessaires au démarrage d'un ordinateur (**BIOS**, instructions de démarrage, microcode).
- Les équipements embarqués (relativement lents), pour contenir le programme de la partie numérique, en association avec une RAM dynamique pour traiter les données ;

Table 2.4 : *Description des mémoires mortes.*

Type	Abréviation	Description
ROM	Read-Only Memory	▪ Programmée chez le fabricant du circuit, pendant le processus de fabrication et selon les spécifications du client. ▪ Ce type de mémoire est très coûteux et ne convient qu'à des applications de très grande série.
PROM	Programmable ROM	▪ Programmable par l'utilisateur une seule fois et d'une façon définitive. ▪ Une fois programmée, la PROM ne diffère en rien d'une ROM. ▪ La programmation de ce type de mémoire consiste à effectuer la rupture de liaisons fusibles.
EPROM	Erasable PROM	▪ Programmable et effaçable, par l'utilisateur, autant de fois qu'il le désire. ▪ Cependant un effacement de la mémoire est nécessaire avant chaque nouvelle programmation, par exposition du composant pendant quelques minutes (20 mn à 30 mn en moyenne) à une source de rayons ultraviolets.
EEPROM	Electrically EPROM	▪ Est une version améliorée de l'EPROM. Elle garde les mêmes opérations d'écriture répétée et d'effacement, à la différence que l'opération d'effacement s'effectue par application d'une tension électrique.

2.3.2.1. Structure générale d'une mémoire morte

Une mémoire morte est constituée d'un ensemble de cellules mémoire dont chacune représente un **élément binaire** (**bit**).

Ces cellules sont organisées en lignes et colonnes selon une structure matricielle, donnée par la figure ci-dessous.

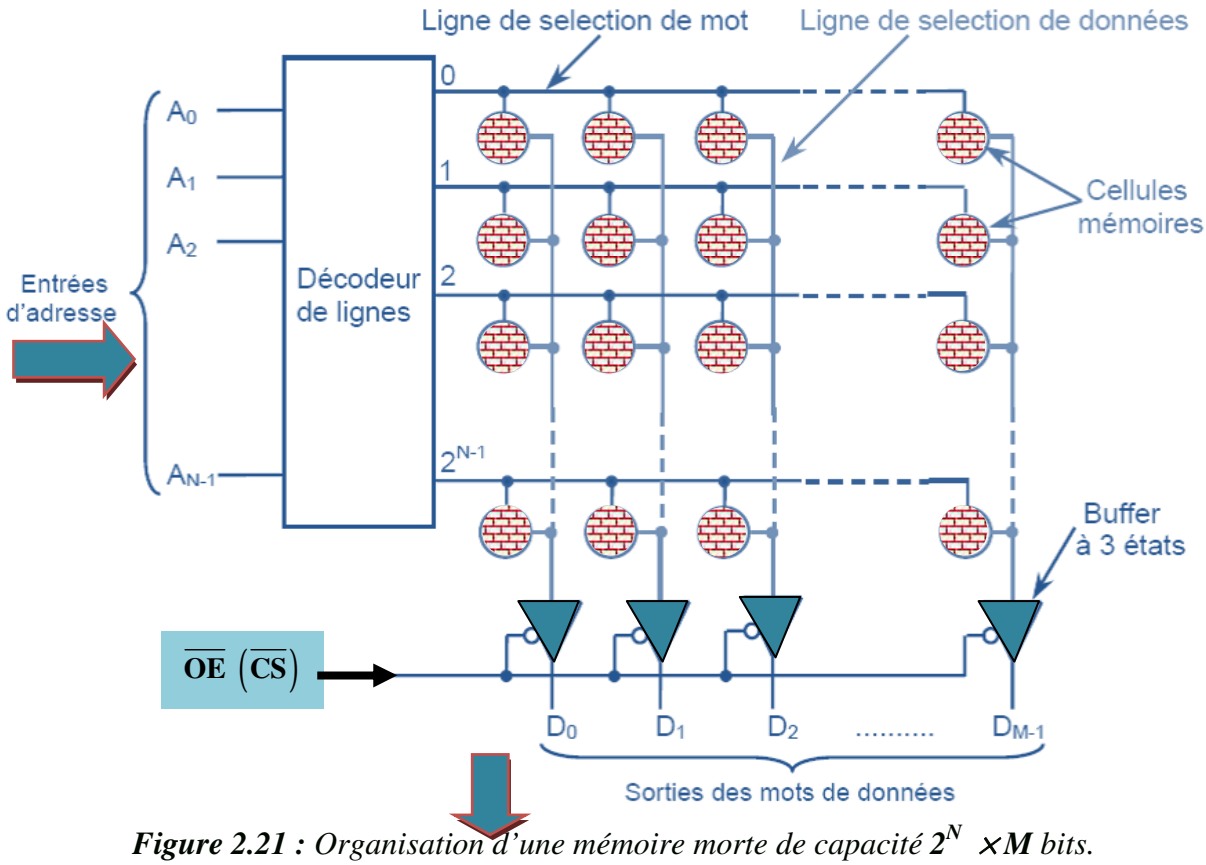


Figure 2.21 : *Organisation d'une mémoire morte de capacité* $2^N \times M$ *bits.*

2.3.2.2. Chronogramme de lecture d'une mémoire ROM

Le chronogramme de la figure ci-dessous illustre le processus de lecture d'une mémoire ROM.

1. La lecture commence par l'application d'une nouvelle adresse suivie d'une mise à zéro de l'entrée **CS** (**OE**).
2. Le retard entre l'instant de la présentation de l'adresse aux entrées de la ROM et le moment où la donnée passe de son état haute impédance pour apparaître à la sortie de la mémoire, s'appelle le temps d'accès $\mathbf{t_{ACC}}$.
3. Ce temps est une caractéristique de la mémoire ROM, il est donné par le fabricant du composant.
4. Un autre paramètre important de synchronisation est le temps $\mathbf{t_{OE}}$, qui est le retard entre l'instant de passage à l'état bas de **CS** (**OE**) et le moment où la donnée apparaît sur la sortie.

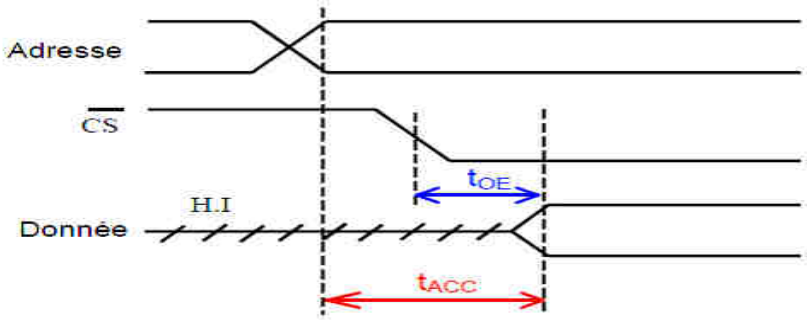
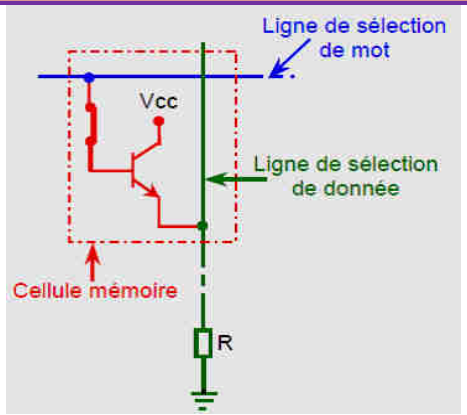
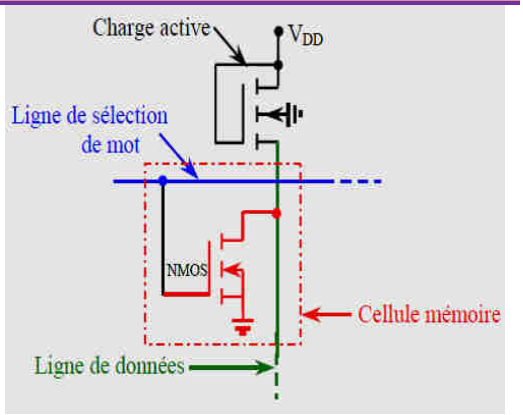
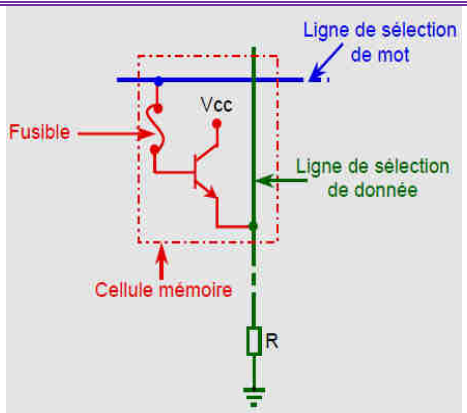
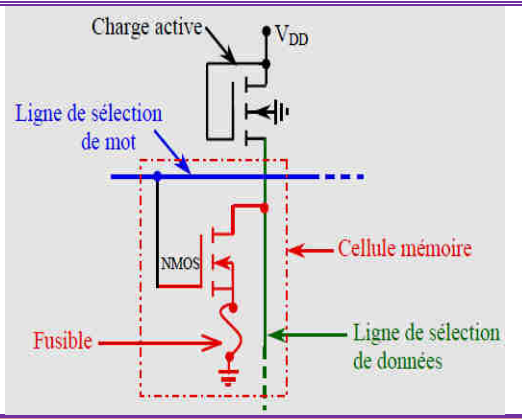
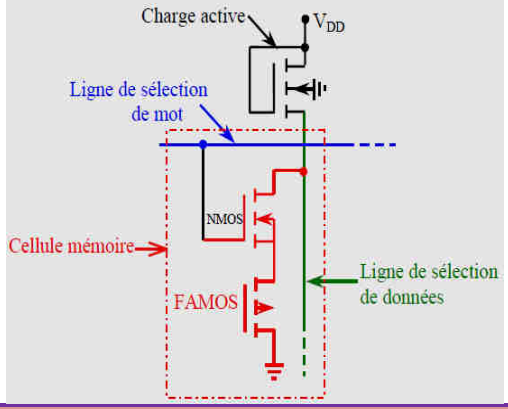
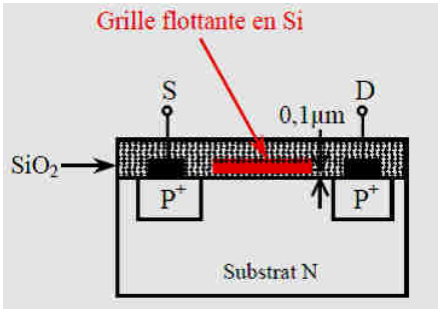
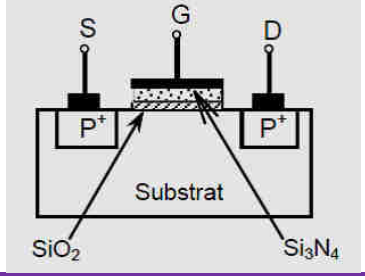


Figure 2.22 : *Chronogramme de lecture d'une mémoire ROM.*

La différence entre les types des mémoires mortes réside dans la structure technologique des cellules mémoires.

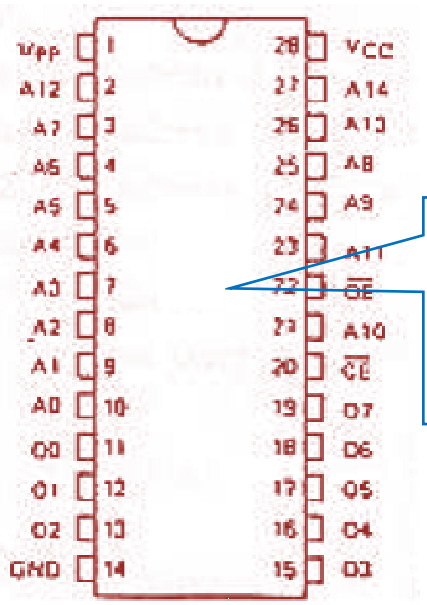
Table 2.5 : *Structure technologique des cellules mémoires mortes.*

Type	Structure technologique	
	Technologie Bipolaire	Technologie MOS
ROM		
PROM		
EPROM	Technologie FAMOS	
		
EEPROM	Technologie NMOS	
		

Remarques :

- ✓ La cellule mémoire d'une **EEPROM** a la même structure que l'**EPROM** sauf qu'il faut changer le transistor **FAMOS** par le **MNOS**.
- ✓ Quand le transistor **MNOS** est bloqué (programmé par une tension positive), la cellule inscrit le bit "**1**", par contre quand il est conducteur (programmé par une tension négative), la cellule inscrit un "**0**".
- ✓ Les temps d'accès à la mémoire morte est de l'ordre de grandeur de **150** nanosecondes comparativement à un temps d'accès d'environ **10** nanosecondes pour la mémoire vive.
- ✓ Pour accélérer le traitement des informations, les données stockées dans la mémoire morte sont généralement copiées dans une mémoire vive avant d'être traitées. On appelle cette opération le **shadowing**.

Exemple : Donner le type et la capacité de la mémoire schématisée ci-après:



Broche	Fonction
A0-A14	Adresses d'entrées
$\overline{CS}$	Chip Enable
$\overline{OE}$	Output Enable
O0-O7	Sorties

Solution :

- Le boîtier est une mémoire morte (ROM) car il ne possède pas d'entrée d'écriture.

Il dispose de :

- 15 bits d'adresse : A0-A14;
- 8 bits de données.

Sa capacité est donc: $C=2^{15} \times 8=2^5 \times 2^{10} \times 8=32$Ko ou encore 32 Kmots de 8 bits.

2.4. Décodage d'adresse – Interfaçage Microprocesseur/mémoire

Le décodage d'adresse permet de définir l'emplacement des différentes zones mémoires dans l'espace adressable. Pour cela sa fonction est de produire les différents signaux de sélection (CS) des circuits communiquant avec le µP : mémoires, CNA, CAN, circuit d'interface parallèle, interface série ...

- ✓ Si un circuit n'est pas sélectionné ($\overline{CS}=1$), ses entrées/sorties se trouvent en état **HAUTE IMPEDANCE** avec pour effet de **DECONNECTER** le circuit du bus de données.
- ✓ Ce décodage se fait à l'aide de fonctions logiques sous forme de circuits électroniques standards (portes TTL par exemple) mais peut être implanté dans le chipset de la carte mère.

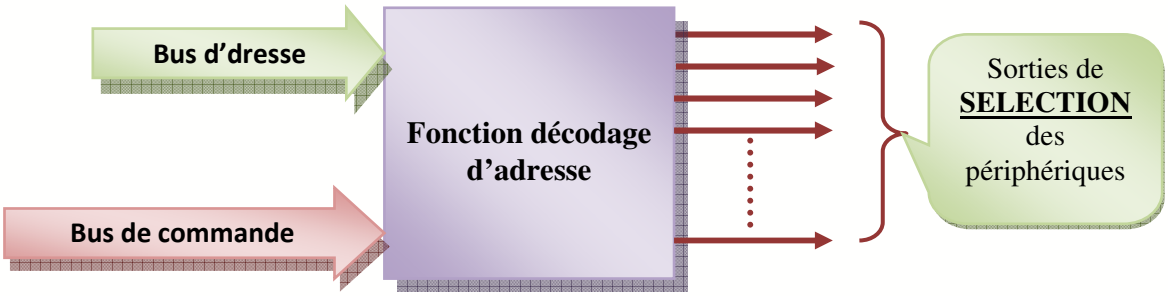


Figure 2.23 : *Décodage d'adresse.*

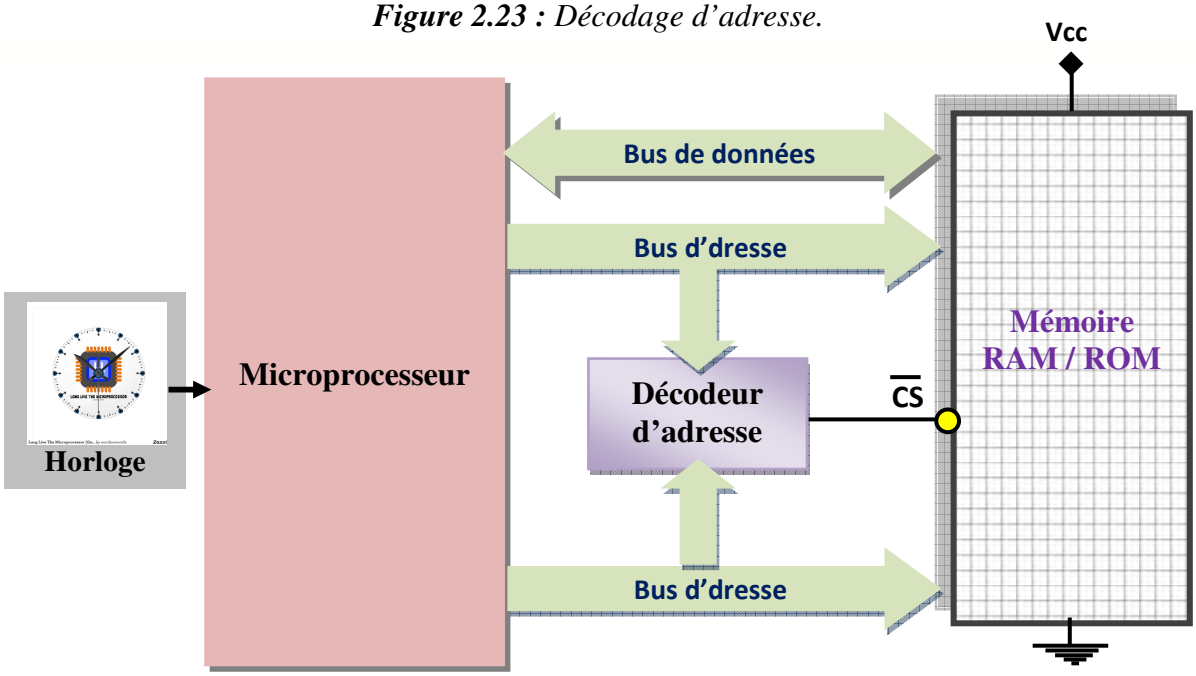
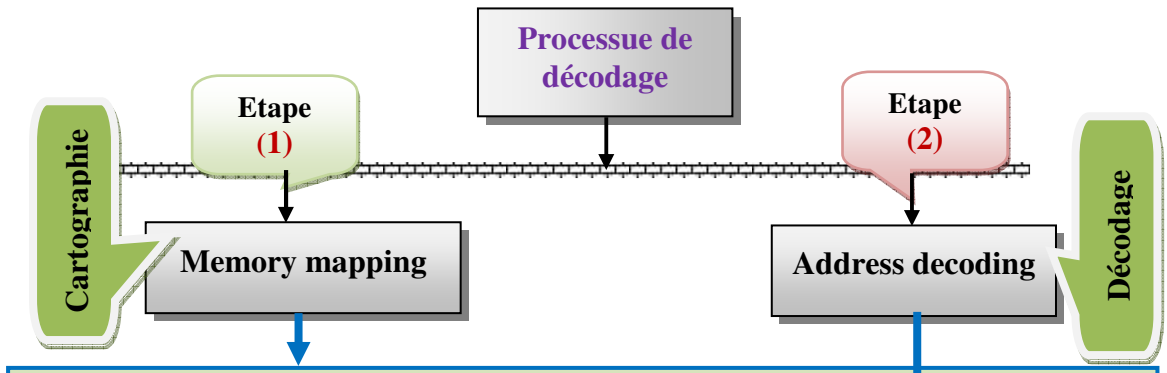
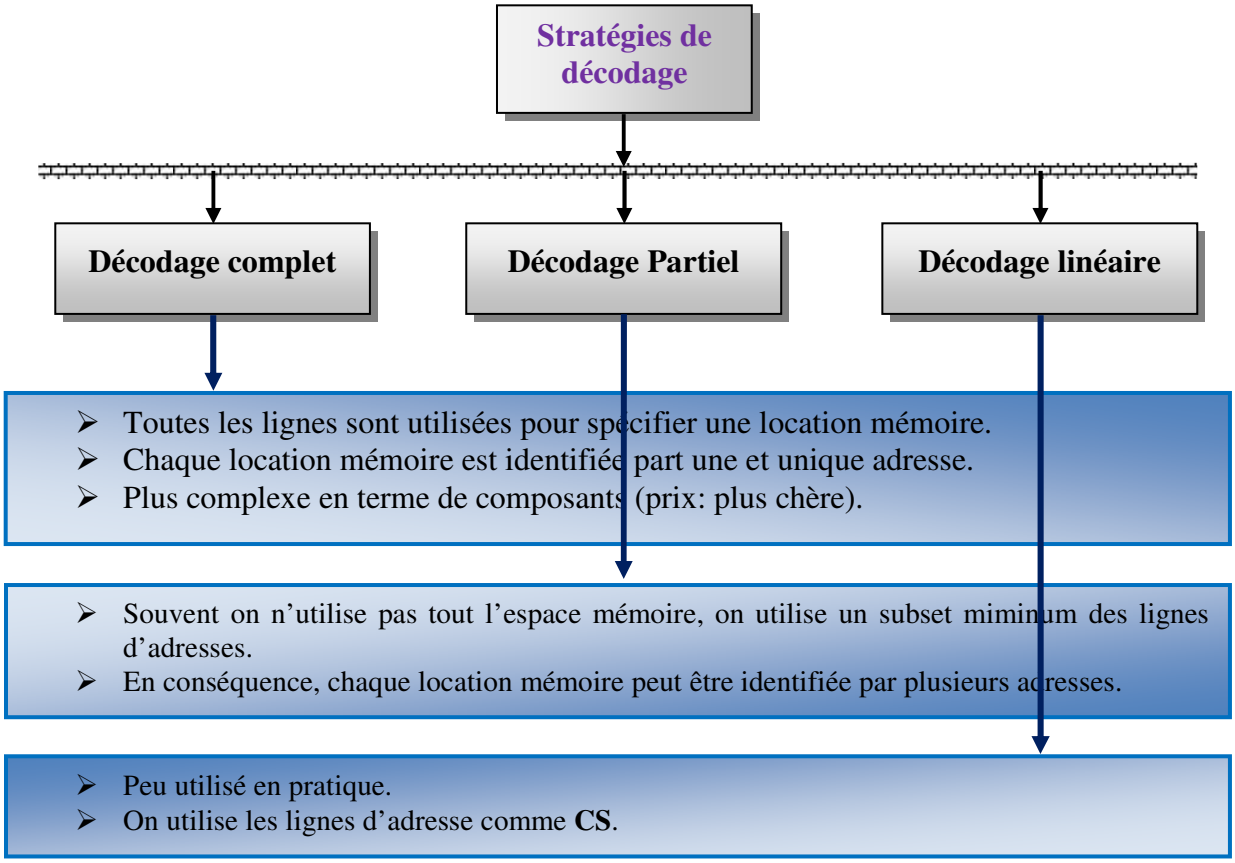


Figure 2.24 : *Interface microprocesseur-mémoires.*



- **Etape 1 : Mapping** est le processus de répartition de l'espace adressable entre les mémoires et les périphériques.

- **Etape 2 :** On divise le bus d'adresse en 2 parties:
  1. **N** bit plus significatif génère le CS.
  2. **M** bit moins significatif alimente le composant pour accéder aux différentes cellules mémoires ou registres internes.



Exemple de réalisation d'un décodeur d'adresses pour un bus 64 KO

Espace mémoire adressable = 64 KO → La taille de bus d'adresse = 16 lignes ($64 = 2^{16}$)

Circuit	Taille mémoire	Adresses occupées	
		Début	Fin
PROM1	4KO	$F000_{HEX}$	$FFFF_{HEX}$
PROM2	4KO	$E000_{HEX}$	$EFFF_{HEX}$
TIMER	/	$A000_{HEX}$	$A007_{HEX}$
ACIA	/	6000_{HEX}	6004_{HEX}
PIA	/	4000_{HEX}	4004_{HEX}
RAM	2KO	0000_{HEX}	$07FF_{HEX}$

* : N'est pas nécessaire au décodage.

Le décodage d'adresse peut s'effectuer de diverses manières :

- Par portes logiques,
- Décodeur intégré (type 74LS138) ou
- Par PAL (type 16L8),
  - → Il s'agit de toutes les façons de logique combinatoire.

$A_m \rightarrow \rightarrow \rightarrow$ Taille = 2^m

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	Adresse	Circuit
1	1	1	1	x	x	x	x	x	x	x	x	x	x	x	x	$FFFF_{HEX}$ $F000_{HEX}$	PROM1
1	1	1	0	x	x	x	x	x	x	x	x	x	x	x	x	$EFFF_{HEX}$ $E000_{HEX}$	PROM2
1	0	1	*	*	*	*	*	*	*	*	*	*	x	x	x	$A007_{HEX}$ $A000_{HEX}$	TIMER
0	1	1	*	*	*	*	*	*	*	*	*	*	*	x	x	6003_{HEX} 6000_{HEX}	ACIA
0	1	0	*	*	*	*	*	*	*	*	*	*	*	x	x	4003_{HEX} 4000_{HEX}	PIA
0	0	0	*	*	x	x	x	x	x	x	x	x	x	x	x	$07FF_{HEX}$ 0000_{HEX}	RAM

Décodage par portes logiques

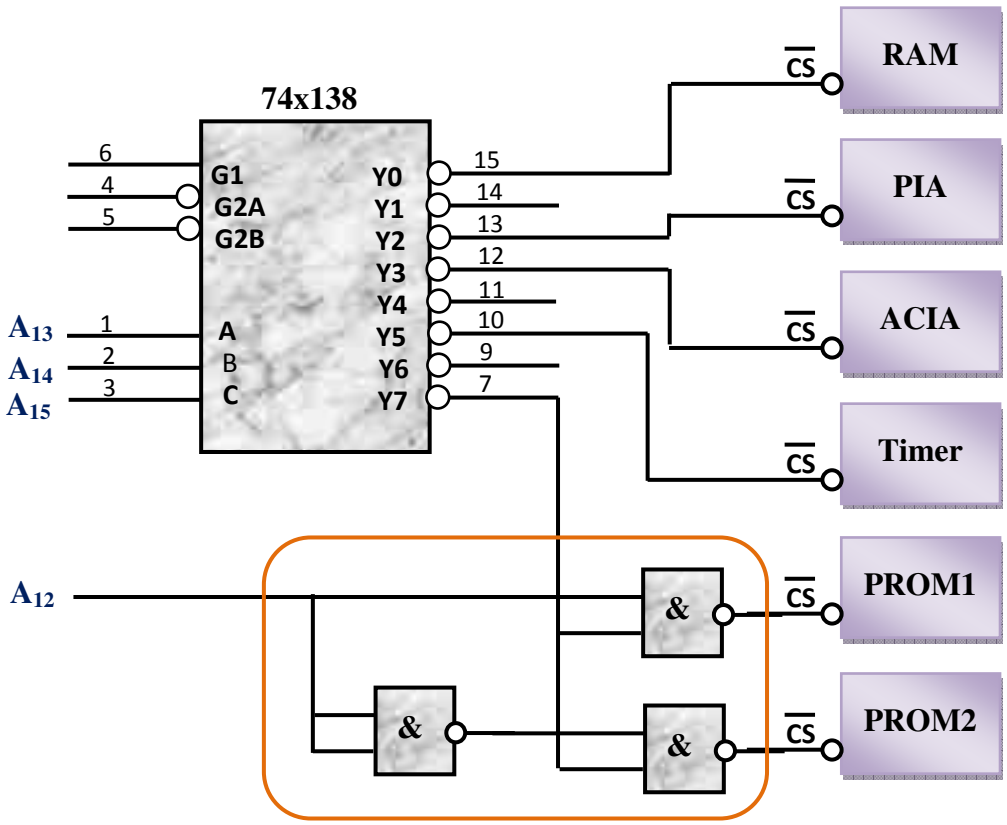
Les $\overline{CS}$ des circuits périphériques et mémoires sont généralement actifs à « **0** », on obtient donc les équations suivantes :

$$\begin{cases} \textbf{RAM} & \rightarrow \overline{CS} = A_{15} + A_{14} + A_{13} + A_{12} \\ \textbf{PIA} & \rightarrow \overline{CS} = A_{15} + \overline{A_{14}} + \overline{A_{13}} + A_{12} \\ \textbf{ACIA} & \rightarrow \overline{CS} = A_{15} + \overline{A_{14}} + A_{13} + A_{12} \\ \textbf{PROM1} & \rightarrow \overline{CS} = \overline{A_{15}} + \overline{A_{14}} + \overline{A_{13}} + A_{12} \\ \textbf{PROM1} & \rightarrow \overline{CS} = \overline{A_{15}} + \overline{A_{14}} + \overline{A_{13}} + \overline{A_{12}} \end{cases}$$

- ✓ On voit que la réalisation des ce décodeur nécessitera plusieurs circuits intégrés (des fonctions NAND, NOR, AND, OR), de plus certains périphériques nécessitent d'être synchronisés avec l'horloge E du microprocesseur, les signaux $\overline{CS}$ doivent alors être validés lors de l'état actif de E ce qui nécessite des portes supplémentaires.
- ✓ Cette solution est très rarement retenue.

Décodage par portes logiques

Le 74HC138 est un décodeur 3 vers 8 avec 3 entrées de validation et sorties actives à l'état bas:



2.5. Mémoires spéciaux : Piles

Ce sont des mémoires (petites taille) appelées **PILES (Stack),** où les mots ne sont pas adressables.

- ✓ On distingue deux catégories de mémoires vives à accès séquentiel :
  - ➢ Les files d'attente ou **FIFO** (First In First Out), et
  - ➢ Les piles ou **LIFO** (Last In First Out).

Table 2.6 : *Description des Piles.*

Type	Abréviation	Description
FIFO	First In - First Out	▪ Sont un type particulier de mémoires RAM, ou les mots ne sont pas adressables: les mots sont lus dans le même ordre dans lequel ils sont écrits (le premier écrit est le premier lu)
LIFO	Last In - First Out	▪ Sont un type particulier de mémoires RAM, ou les mots ne sont pas adressables: les mots sont lus dans l'ordre inverse dans lequel ils sont écrits (le dernier écrit est le premier lu)

- ✓ Deux opérations possibles sur les files d'attente :
  1. **PUSH** (**EMPILER**): Mettre une information au sommet de la pile, en poussant vers le bas les informations déjà présentes dans la pile.
  2. **POP** (**DEPILER**): Prend l'information qui se trouve au sommet de la pile, en poussant tout le contenu de la pile vers le sommet.

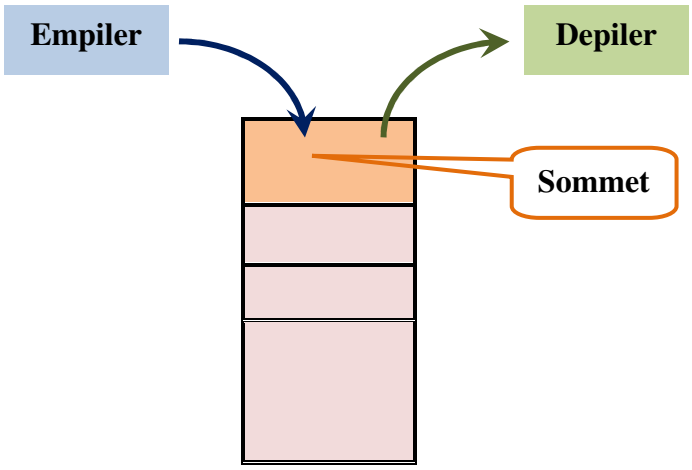


Figure 2.25 : *Représentation graphique de la Pile.*

Les piles sont utilisées dans la solution de plusieurs types de problèmes :

- Pour sauvegarder le contexte des sous-programmes lors de l'appelle des autres sous-programmes.
- Dans la compilation des programmes.
- Des problèmes du monde réel dont les données respect le protocole LIFO

La figure x illustre l'architecture d'une pile FIFO. Il s'agit en fait d'une RAM double-port dont un bus d'adresses est dédié à la lecture et l'autre à l'écriture, et dont l'adressage est géré par des compteurs.

- ✓ A chaque écriture ou lecture, le pointeur correspondant est incrémenté par comptage.
- ✓ Ce dispositif sert le plus souvent d'interface entre deux flots de données de fréquences différentes.
- ✓ La capacité de la FIFO, ainsi que les mécanismes prévenant les dépassements doivent être dimensionnés en fonction de l'application à traiter.
- ✓ Les LIFO ne comportent qu'un pointeur associé à un compteur / décompteur qui s'incrémente à chaque écriture et se décrémente à chaque lecture.

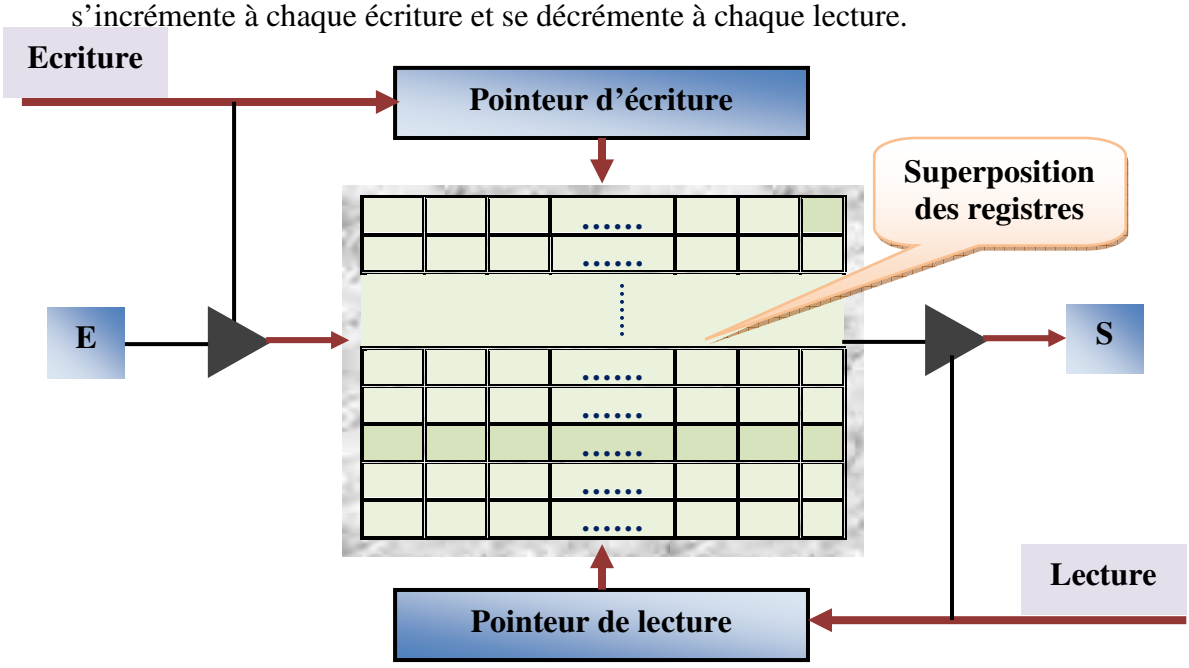


Figure 2.26 : *Architecture d'une pile FIFO.*

- **Remarques :** L'utilisation des piles sera détaillée dans le chapitres Microprocesseurs.

2.6. Exercices

EXERCICE 01 :

- Matérialiser **technologiquement** un **Bit** :
  - Cas des mémoires RAM statiques et dynamiques.
  - Cas des mémoires PROM.
- Quelles sont les principales différences entre la DRAM et la SRAM ? Où utilise-t-on de la DRAM ? De la SRAM ?
- Quelles sont les principales différences entre la RAM et la ROM ? Où utilise-t-on de la ROM Classez les mémoires suivantes par taille, par rapidité : RAM, registres, disques durs, cache L1, cache L2, CD-ROM.
- Pourquoi utilise-t-on des mémoires caches?

EXERCICE 02 :

- Combien de bits d'information peuvent être transmis *simultanément* sur une ligne électrique ?
- Comment peut-on transmettre simultanément 4 bits ?
- Soit un bus de données de 8 bits :
  - Quel est le plus petit nombre binaire que l'on peut y représenter ?
  - Et le plus grand ?
  - Donnez ces nombres en base 2, 10 et 16.
- Soit un bus d'adresse de 2 bits :
  - Combien d'adresses différentes peut-on y représenter ?
  - Même question pour 20 bits.
- Quel est l'espace adressable par un processeur 16 bits à 32 bits d'adresse ?
- Combien de pattes " adresse " y-a-t-il sur un module de mémoire de 1 Mo (mots de 8 bits) ?

EXERCICE 03 :

On considère une mémoire EPROM type 27C256 de 32 Ko.

1. Déterminer le nombre de bits du bus d'adresse et du bus de données de cette mémoire.
2. Donner son schéma fonctionnel.
3. Calculer le nombre de zones et l'adresse de début et de fin des zones accessibles dans les cas suivants :

   - $A_{14} = 0$.
   - $A_{13} = 1$.
   - $A_{13} = 0$; $A_{11} = 0$.

EXERCICE 04 :

On souhaite constituer un bloc mémoire de 10Ko à partir de blocs élémentaires de 4Ko et 2Ko. Donner les équations de sélection des différentes circuits mémoires dans les cas suivants :

1. Le bloc global commence en 0000H.
2. Le bloc global commence en 0800H.

EXERCICE 05 :

On souhaite insérer, dans un système à microprocesseur ayant un bus d'adresse $A_{15} - A_0$ et un bus de données $D_{15} - D_0$ et les signaux $\overline{WR}$ pour l'écriture et $\overline{RD}$ pour la lecture, les circuits mémoires suivants :

- Une mémoire morte de 16Ko implantée à partir de 2000H.
- Une mémoire morte de 8Ko implantée à partir de 8000H.
- Une mémoire morte de 4Ko implantée juste après celle de 8Ko.

Donner les équations de sélection des ces mémoires.

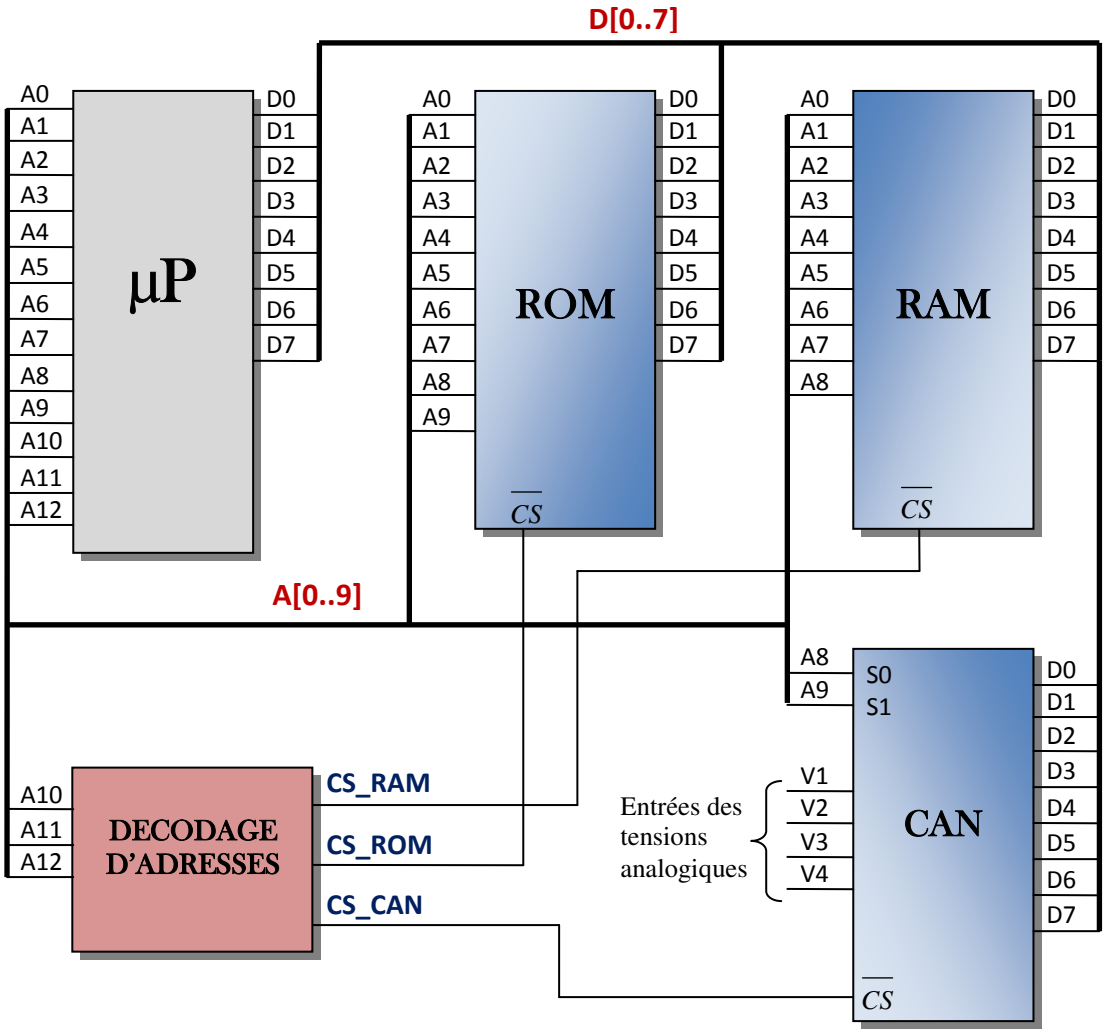
EXERCICE 06 :

On considère un système à base d'un microprocesseur comprenant les divers éléments suivants :

- Un microprocesseur 8 bits ;
- Une mémoire ROM ;
- Une mémoire RAM ;
- Un Convertisseur Analogique – Numérique (CAN) permettant de convertir plusieurs tensions analogiques.

Questions :

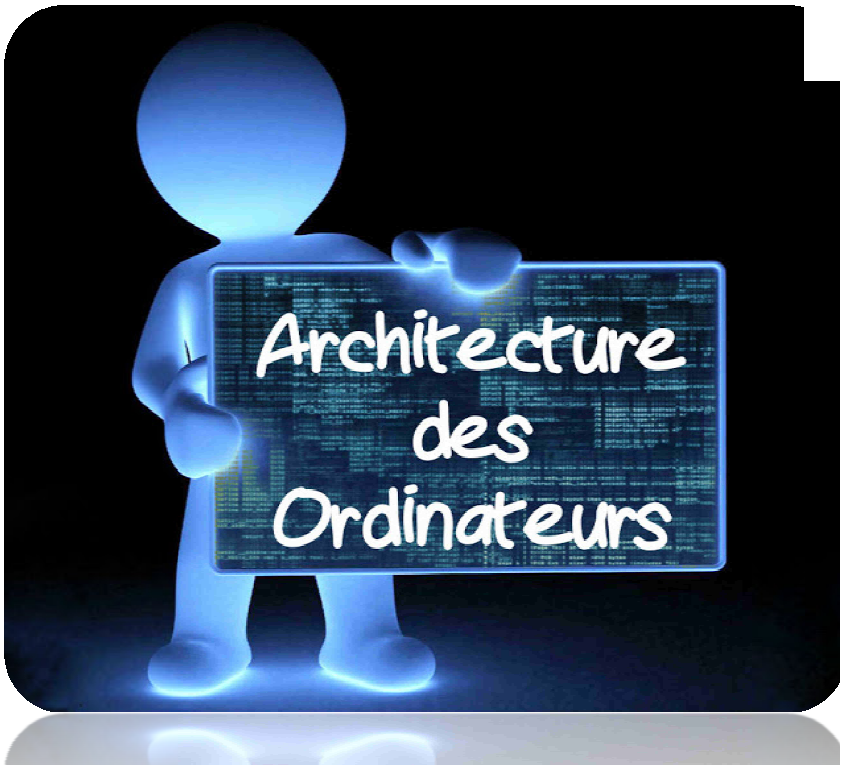
- Calculer la capacité de la mémoire RAM en octets et en kbits.
- Calculer la capacité de la mémoire ROM en ko et en kbits.
- Comment se nome le bus D[0..7] ?
- Comment se nome le bus A[0..7] ?
- Que signifie le terme $\overline{CS}$? Sur quel état logique cette entrée est-elle valide ?
- Cocher la bonne réponse : Lorsque le signal $\overline{CS}$ de la RAM et de la ROM est à l'état logique « 1 »….
  - Les sorties de la RAM et de la ROM sont à « 1 » ou à « 0 » suivant les valeurs contenues dans les mémoires.
  - Les sorties du circuit intégré sont en haute impédance « Hiz ».
- Cocher la bonne réponse : Dans le cas où les signaux $\overline{CS}$ de la RAM et de la ROM sont à l'état bas en même temps…
  - Il n'y a pas de conflit sur le bus de données.
  - Il y a un conflit car les signaux D0 à D7 de ces deux composants sont présents sur le bus de données.
- Si on veut lire les données de la RAM, quels circuits doit-on désactiver afin d'éviter un conflit sur le bus de données ?
- Si on veut lire les données de la ROM, quels circuits doit-on désactiver afin d'éviter un conflit sur le bus de données ?
- Si on veut lire les données du CAN, quels circuits doit-on désactiver afin d'éviter un conflit sur le bus de données ?
- Expliquer succinctement le rôle du décodage d'adresses dans un tel système. Donner un exemple. Discuter…


D[0..7]
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
µP
D0
D1
D2
D3
D4
D5
D6
D7
ROM
RAM
CS
A[0..9]
A8
A9
S0
S1
CAN
V1
V2
V3
V4
Entrées des tensions analogiques
DECODAGE D'ADRESSES
CS_RAM
CS_ROM
CS_CAN


© Can Stock Photo - csp26373312

CHAPITRE **3**

Architecture des ordinateurs



CHAPITRE 3

ARCHITECTURE DES ORDINATEURS

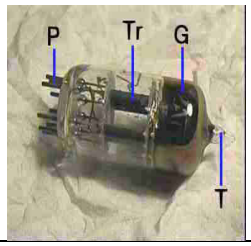
3.1. Introduction, historique et évolution

L'ordinateur a, depuis le jour de sa création, évolué d'une manière plus rapide. L'objectif principal est de créer des machines capables de remplacer l'homme dans ses tâches routinières. Son évolution continue de s'accélérer et n'est pas prête de s'arrêter.

- ✓ D'un autre côté, la population mondiale ne cesse de s'accroître.
- ✓ Comment alors gérer cette masse de personnes ? L'ordinateur pourrait bien apporter une réponse à cette question.
- ✓ Une société informatisée comblerait davantage les besoins de tous et chacun.

Un bref historique sur l'évolution des ordinateurs est donné par le tableau 3.1. Les avantages de chaque génération seront aussi illustrés.

Table 3.1 : *Générations des ordinateurs.*

Génération	Elément de base	Epoque	Période / Caractéristiques	
/		Mécanique	XVII siècle	Préhistoire
1	Tubes à vide 	Electromé-canique	1946-1947	▪ Ordinateur à cartes perforées et à bandes magnétiques. ▪ Programmation physique en langage machine. ▪ Calcul numérique (trigonométrie). ▪ Appareils immenses, lourds, énergie élevée. ▪ Prix élevé / capacité et performance.
2	Transistors 	Electronique	1956-1971	▪ Transistor => augmentation de la fiabilité. ▪ Utilisation de mémoires de masse pour le stockage périphériques. ▪ Temps d'accès moyen (de l'ordre de la micro-seconde). ▪ Fonctionnement séquentiel des systèmes de programmation (langages évolués):FORTRAN. ▪ Mainframes.

3	Circuits intégrés 		1971-1980	▪ Miniaturisation des composants (circuits intégrés). ▪ Apparition des systèmes d'exploitation. ▪ Concepts de temps partagés. ▪ Machines polyvalentes et de capacité variée ▪ Appareils modulaires et extensibles ▪ Multitraitement (plusieurs programmes à la fois). ▪ Télétraitement (accès par téléphone). ▪ UNIX. ▪ Mini ordinateurs.
4	Microprocesseurs 		1982-2000	▪ Miniaturisation extrêmes des composants. ▪ Apparition des microprocesseurs. ▪ Diversification des champs d'application. ▪ Apparition de la micro-informatique. ▪ L'aspect logiciel prend le pas sur l'aspect matériel.
≥ 5	Intelligence artificielle (Systèmes intelligents)		2000-…	▪ Miniaturisation des composants poussée à l'extrême. ▪ Vitesse proche de celle de la lumière. ▪ Processeurs en parallèle. ▪ Nouvelles structures et représentations des données.

- Les machines les plus marquantes durant l'époque mécanique sont celles de **Schkard** (**1663**) et Pascal (**1642**). En **1673**, le mathématicien **Leipnitz** met au point une machine capable de faire automatiquement les opérations d'adition, de soustraction, de multiplication et de division.
- Le concept de programme enregistré est inventé par Babbage. Babbage à conçu en **1833** une machine capable d'effectuer une séquence d'opérations sans l'intervention de l'utilisateur → Le premier calculateur programmable. Son principe de fonctionnement est :
  - Lire une carte perforée.
  - Exécuter l'opération correspondante.
  - Lire la carte suivante.

❖ Les innovations se succèdent alors à une vitesse de plus en plus grande pour aboutir aux calculateurs que nous connaissant aujourd'hui et qui, pour la majorité, répondent encore au schéma décrit par **Von Neuman** dans son rapport de **1945**.

❖ De génération en génération, le développement des ordinateurs n'aura pas de limites, et les chercheurs se préoccupent aujourd'hui sur la possibilité de réaliser une **machine qui imitent** le **comportement intelligent** de l'être humain, c-à-d, des **microordinateurs intelligents**.

❖ Donc, la multiplicité croissante de la technologie des ordinateurs (**HARD**) et la simplicité de traitement des connaissances (**SOFT**), rendre la généralisation de l'utilisation des ordinateurs dans tous les domaines de la vie quotidienne.

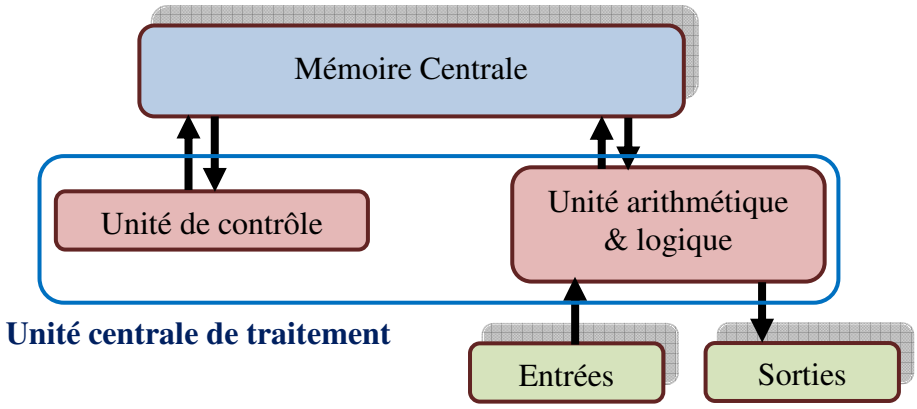


Figure 3.1 : *Architecture de Von Neuman.*

3.2. Concept de base

Les paragraphes ci-dessous illustrent les définitions des termes de base interviennent dans le domaine de l'ordinateur.

- Un **ordinateur** est une machine de traitement de l'information. Il est capable
  - D'acquérir de l'information,
  - De la stocker,
  - De la transformer en effectuant des traitements quelconques,
  - Puis de la restituer sous une autre forme.

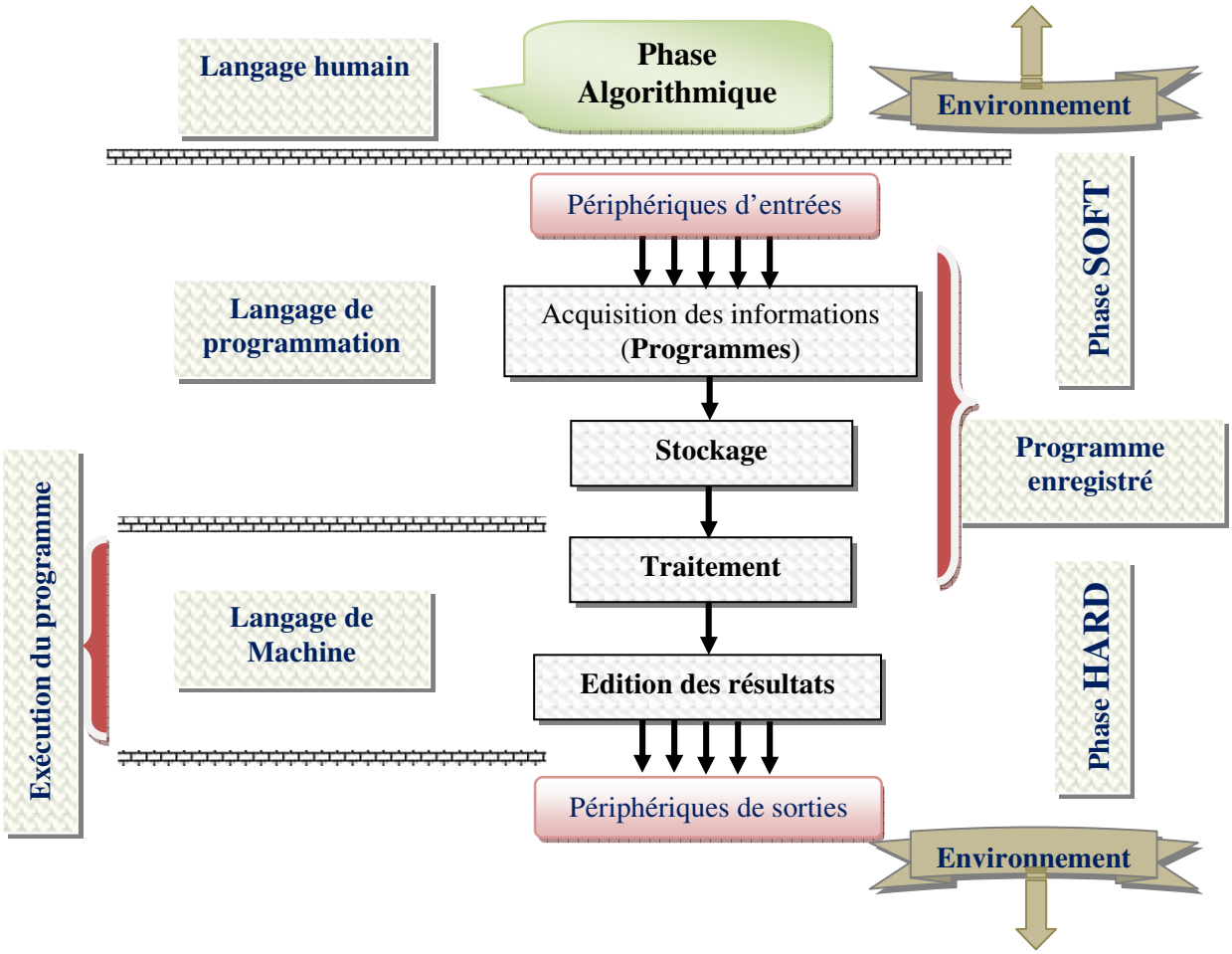


Figure 3.2 : *Phases de traitement des informations par ordinateurs.*

- Le mot **informatique** vient de la contraction des mots information et automatique.
- Nous appelons information tout ensemble de données.
- On distingue généralement différents types d'informations : textes, nombres, sons, images, etc., mais aussi les instructions composant un programme.
- Toute information est manipulée sous **forme binaire** (ou **numérique**) par l'ordinateur.
- Un **programme** est une suite **d'instructions** élémentaires, qui vont être exécutées dans un ordre bien déterminé par le processeur afin d'effectuer une tâche.
- Ces **instructions** correspondent à des actions très simples, comme additionner deux nombres, lire ou écrire une case mémoire, etc. Chaque instruction est **codifiée** en mémoire sur quelques octets.
- Le **processeur** ou **microprocesseur,** ou tout simplement **l'unité centrale (UC)** (le cœur de l'ordinateur) est capable d'exécuter des programmes en **langage machine**, c'est à dire composés d'instructions très élémentaires suivant un codage précis.
- Chaque type de processeur est capable d'exécuter un certain ensemble d'instructions, son **jeu d'instructions.**
- Pour écrire un programme en **langage machine**, il faut donc connaître les détails du fonctionnement du processeur qui va être utilisé.
- L'ordinateur **n'exécute** pas les instructions au fur et à mesure qu'on lui fournis par les **périphériques** (d'entrées/sorties), ils les **enregistrent** et les **exécutent** lorsqu'il sera active → Notion de programmes enregistrés.
- **Périphériques d'entrées** : Permettent d'envoyer des informations à l'Unité Centrale (clavier, …).
- **Périphériques de sorties** Permettent d 'envoyer les résultats à l'extérieur de l'Unité Centrale (Écrans, Imprimantes, …)
- **L'enregistrement des programmes** (en **mémoires**), permet
  - La réalisation des répétitions, des branchements et l'exécution conditionnelle.
  - Les instructions du programme peuvent être manipulées par le programme lui même ou par d'autre programme → partage des ressources, …
  - La vitesse d'exécution des programmes est indépendante de la vitesse d'introduction des instructions qui constituent ce programme.
- Le **transfert** (l'échange) des données entre les composants de l'ordinateur s'effectue via les **BUS** :
  - Bus de données.
  - Bus d'adresse.
  - Bus de commande (contrôle).

  On appelle **bus**, un ensemble de liaisons physiques (câbles, pistes de circuits imprimés, etc.) pouvant être exploitées en commun par plusieurs dispositifs afin de communiquer (**lien de communication**).
- Un **Logiciel** est ensemble de programmes relatif à des traitements d'informations (ex. Windows, Word...).
- **Système d'Exploitation** est une interface (**Soft**) entre l'utilisateur et la machine (**Hard**). Plus formellement, c'est un ensemble de programmes dont la fonction est de :
  - Gérer les ressources Physiques (processeur, mémoire, disques, etc.) et Logiques (fichiers et bases de données, etc.) .
  - Contrôler les entrées-sorties.
  - Ordonnancer les travaux.
  - Gérer les erreurs.

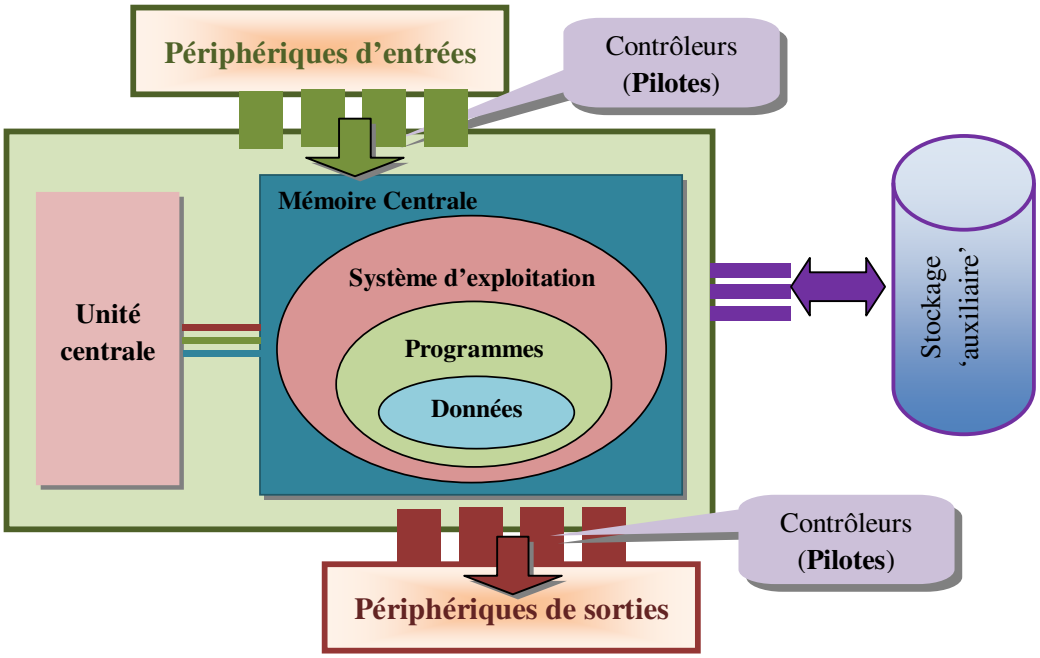


Figure 3.3 : *Structure de l'ensemble SOFT-HARD.*

3.3. Organisation d'un ordinateur en blocs fonctionnels

Tous les ordinateurs contiennent des éléments constitutifs de base :

- Une unité centrale (UC).
- La mémoire centrale (MC).
- Les unités d'échange ou de transfert (UE).

Ces blocs constitutifs sont reliés entre eux par trois **BUS**, comme le montre le schéma de principe de la figure 3.x. Ces trois bus ont pour nom :

- Le bus de données,
- Le bus d'adresse et
- Le bus de commande (ou de contrôle).

Des dispositifs d'entrée et de sortie sont branchés sur les ports d'entrée et de sortie. Un **port** désigne une interface matérielle dans un ordinateur à travers laquelle des données sont échangées avec des périphériques.

3.3.1. Mémoire Centrale (MC)

La mémoire c'est un organe (Dispositif électronique de nos jours), capable de contenir (d'accueillir), de conserver et de restituer à la demande sans les modifier de grandes quantités d'information.

Différents types de mémoires sont bien discutées dans le chapitres des mémoires. Les Deux grandes classes sont :

1. **La mémoire vive RAM (Random Access Memory)** (SRAM-DRAM)
   - Mémoire dans laquelle on peut lire et écrire.
   - Mémoire *volatile* (perd son contenu dès la coupure du courant).
2. **La mémoire morte ROM (Read Only Memory)** (ROM-PROM, EPROM, EEPROM)
   - Mémoire dans laquelle on ne peut que lire.
   - Mémoire permanente (conserve indéfiniment son contenu).

On peut aussi classer les mémoires par leur utilisation ou leur contenu: **mémoire programme et mémoire données.**

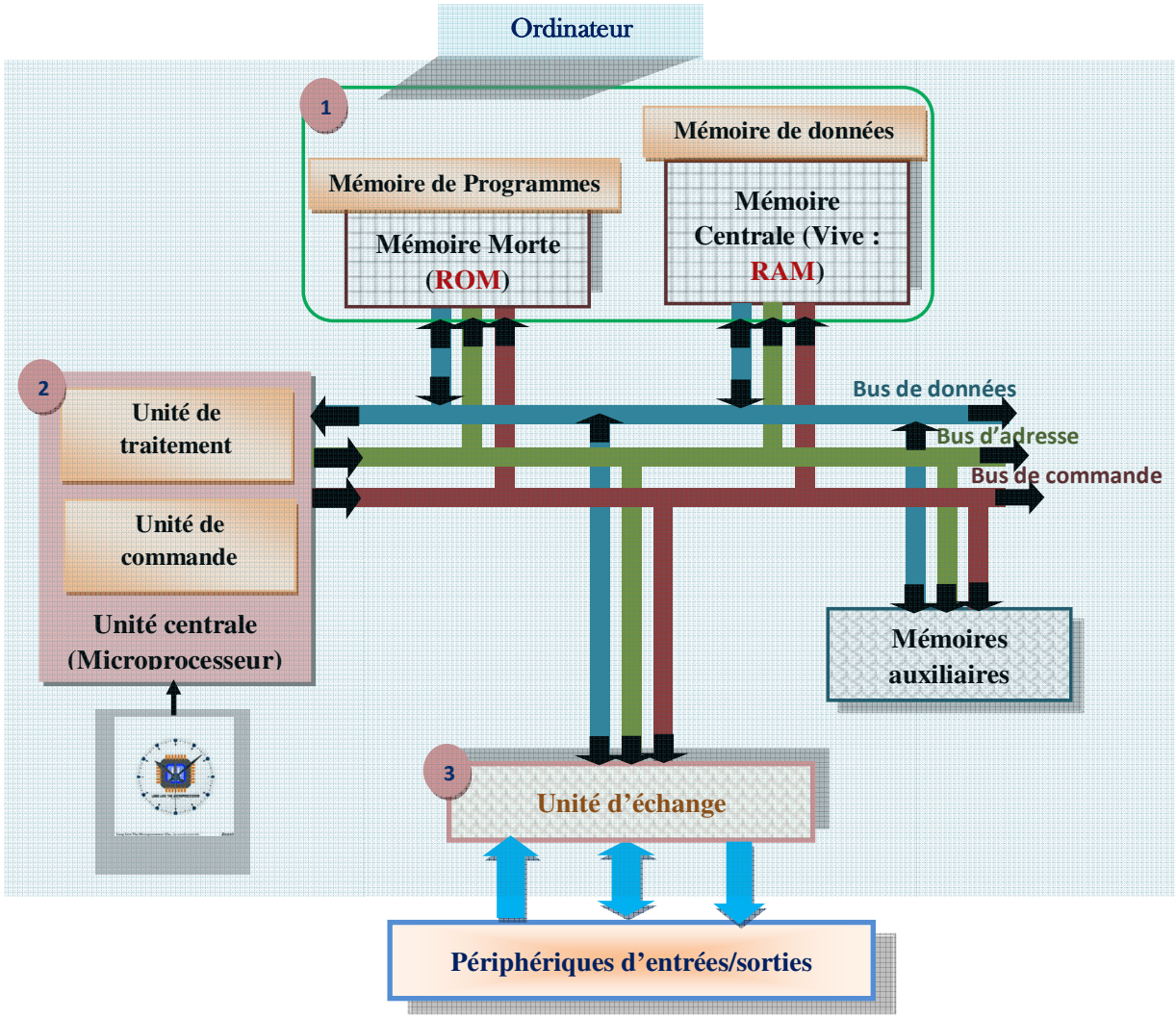


Figure 3.4 : *Architecture schématique d'un ordinateur.*

Mémoire programme

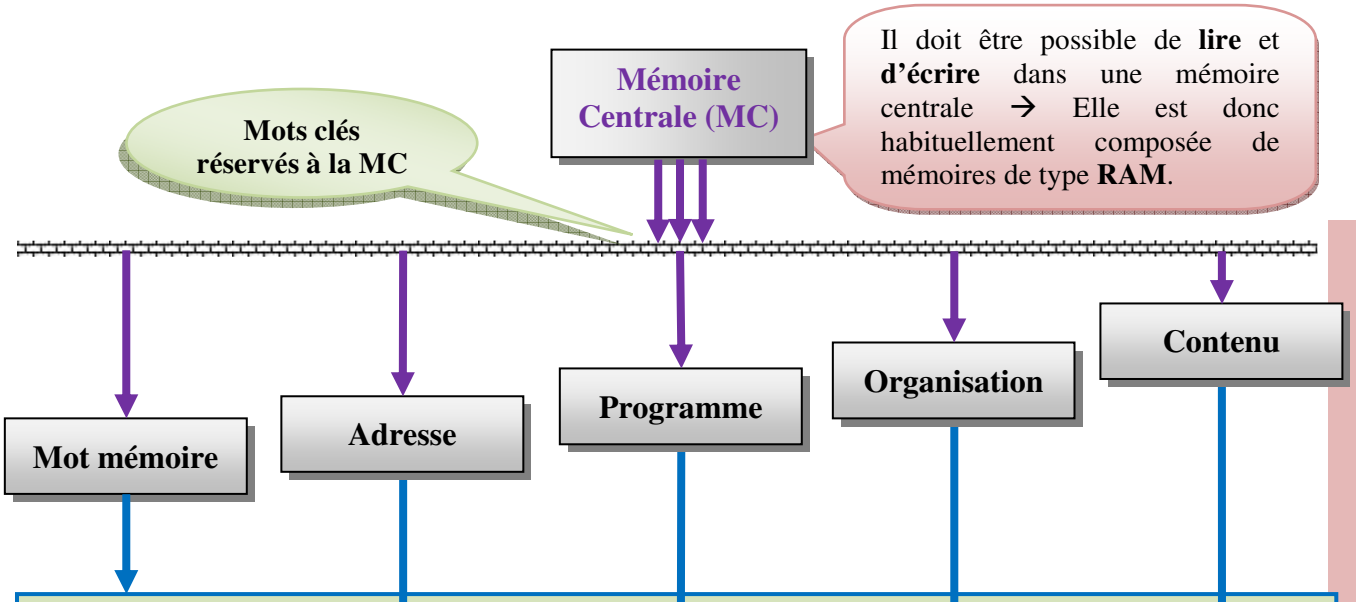
- La mémoire programme contient la liste ordonnée des instructions à traiter par le processeur, c'est à-dire le **programme**. Elle sauvegarde les informations nécessaires au fonctionnement général du système dans un programme spécial appelé moniteur (gestion des différents périphériques et mémoires ainsi que les informations de transfert et de traitements des données).
- *Dans un système à microprocesseur la mémoire programme est en général de type ROM ou PROM.*

Mémoire données

- La mémoire **données** sauvegarde les entrées en provenance des périphériques d'entrées (capteurs, claviers, lecteur de disque, ...), les résultats intermédiaires de calcul du processeur, les données de sortie en partance vers les périphériques de sorties (afficheurs, écran vidéo, enregistreur de disque, ...).
- *La mémoire données est obligatoirement une mémoire de type RAM puisque la lecture et l'écriture y sont nécessaires en permanence.*

Les unités de mesure de stockage de l'information sont :

Le bit (une bascule)
L'octet = 2^3 bits = 8 bits. (noté 1 o)
Le Kilo-octet = 2^{10} octets =1024 o (noté 1 Ko)
Le Méga-octet = 2^{20} octets =$(1024)^2$ o (noté 1 Mo)
Le Giga-octet = 2^{30} octets =$(1024)^3$ o (noté 1 Go)
Le Téra-octet = 2^{40} octets =$(1024)^4$ o (noté 1 To)…



- Le **Mot mémoire (case mémoire)** c'est un regroupement de **n** bits (**n bascules**) constituant une case mémoire dans la mémoire centrale. Ils sont tous numérotés.
- **L'Adresse mémoire** : c'est le numéro d'un mot-mémoire (case mémoire) dans la mémoire centrale.
- **Un Programme** c'est un ensemble d'instructions préalablement codées (en binaire) et enregistrées dans la mémoire centrale sous la forme d'une liste séquentielle d'instructions.
  - Cette liste représente une suite d'actions élémentaires que l'ordinateur doit accomplir sur des données en entrées, afin d'atteindre le résultat recherché.
- La mémoire centrale est **organisée** en bits et en mots. Chaque mot-mémoire est repéré bijectivement par son adresse en mémoire centrale.
- **Contenu** : La mémoire centrale contient des informations sous forme binaire. Deux sortes d'informations :
  - Des programmes (des instructions),
  - Des données.

Une case mémoire doit contenir soit des instructions soit des données. Donc, on peut donc utiliser une mémoire soit en : **Lecture** soit en **Ecriture**, selon le principe de la figure ci-dessous.

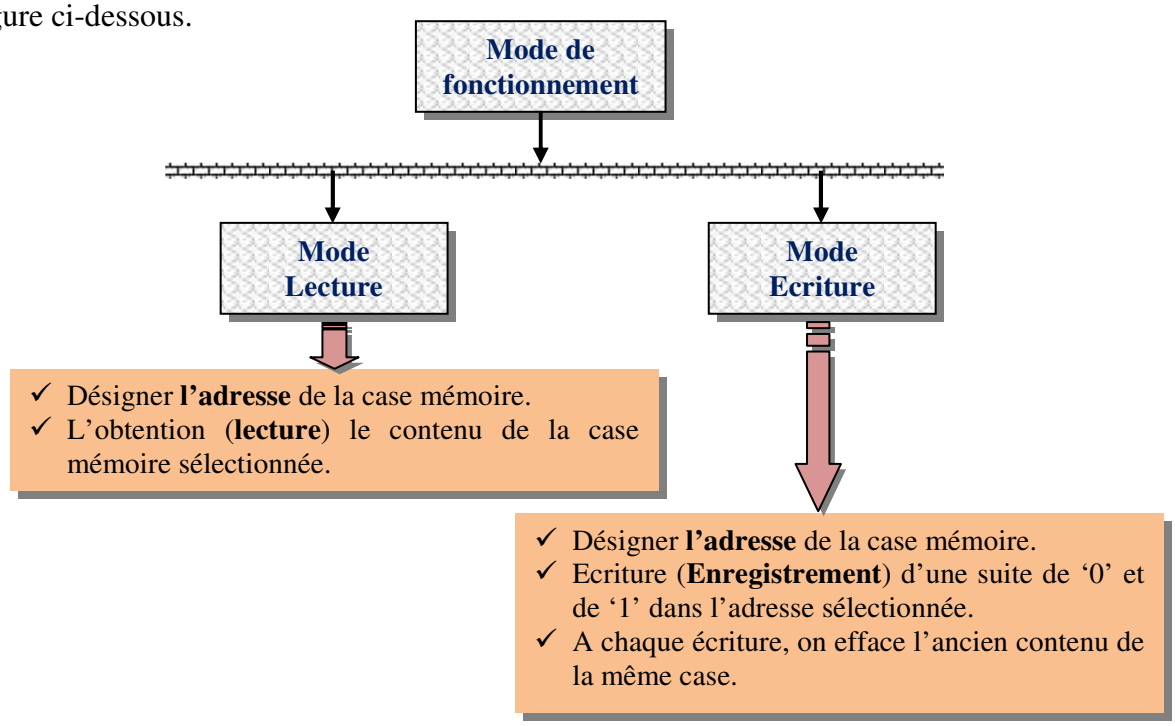


Figure 3.5 : *Mode de lecture/écriture de la mémoire centrale.*

L'acquisition et la restitution d'une information sont effectuées grâce aux deux opérations de lecture et d'écriture. Deux registres sont associés à la mémoire pour permettre ces deux opérations :

- **Registre d'adresse** (**RA**) : Contient l'adresse du mot (case) mémoire à lire ou à écrire.
- **Registre d'information** ou de **données → Registre Mot** (**RM**) : Reçoit l'information lue à partir de la mémoire, où destinée à y être écrite.

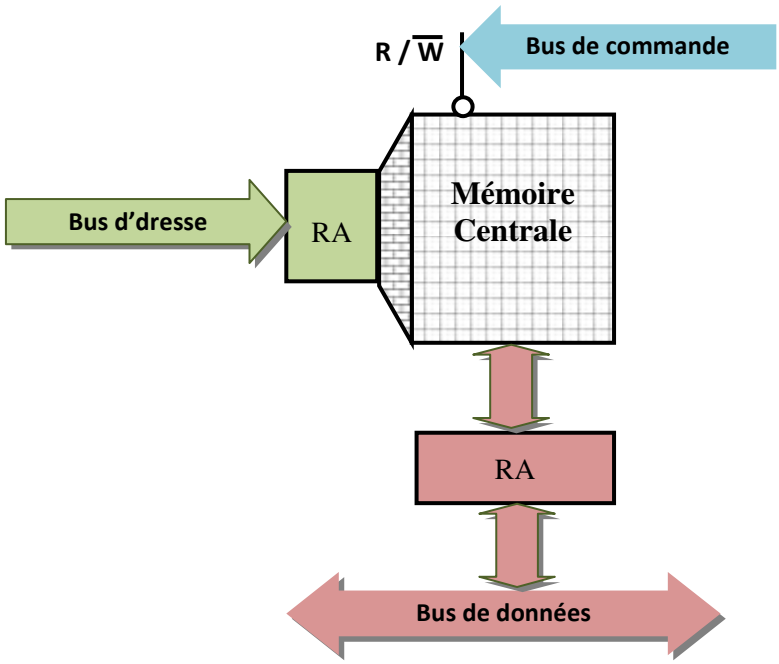


Figure 3.6 : *Structure de la mémoire centrale.*

3.3.1. Unité centrale (UC)

L'unité centrale est le **Cerveau** de l'ordinateur. L'UC se charge de l'exécution des programmes et de la coordination entre les différents organes de l'ordinateur. Elle est composée de deux unités fondamentales :

- Unité de traitement ou Unité Arithmétique et Logique (**UAL**).
- Unité de commande.
- Des registres généraux (**RG**).

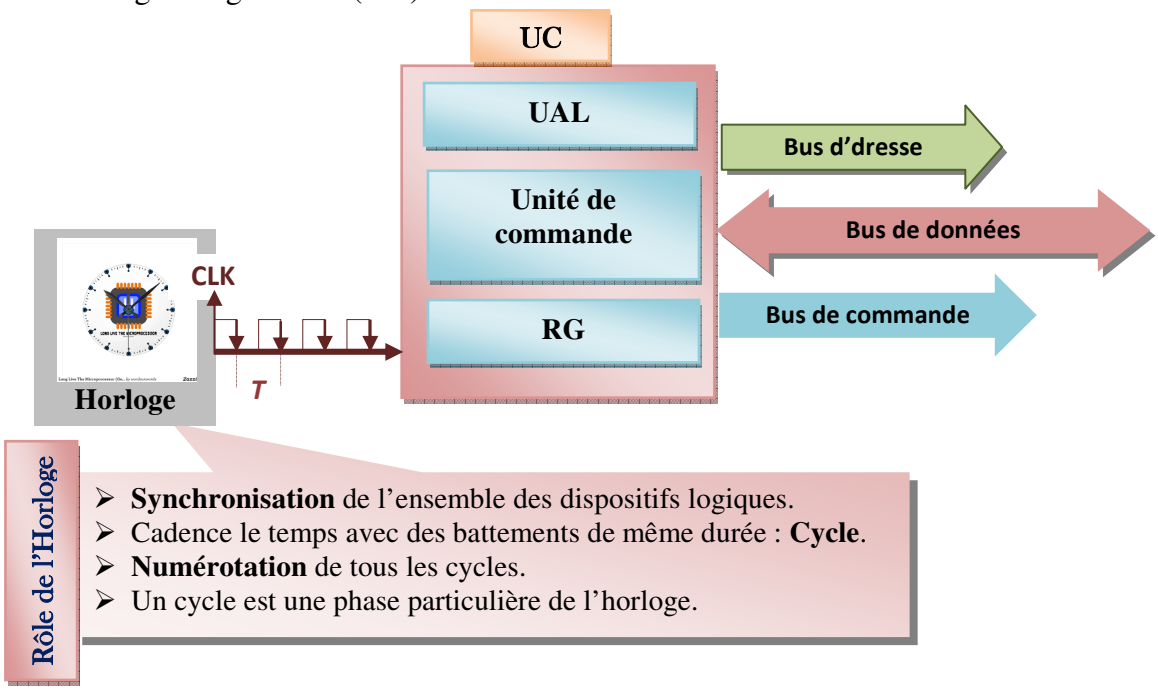


Figure 3.7 : *Structure de l'unité centrale.*

Les principales caractéristiques d'un microprocesseur sont :

- **Le jeu d'instructions** qu'il peut exécuter. Un processeur peut exécuter plusieurs douzaines d'instructions différentes.
- La **complexité de son architecture**. Cette complexité se mesure par le nombre de **transistors** contenus dans le microprocesseur. Plus le microprocesseur contiendra de transistors, plus il pourra plus rapide.
- Le nombre de **bits** que le processeur peut traiter en une instruction (**taille des registres internes du microprocesseur ou la taille de bus de données**). Les premiers microprocesseurs ne pouvaient additionner des nombres de plus de **4** bits en une seule instruction. Ils devaient donc exécuter plusieurs instructions pour additionner des nombres de **32** ou **64** bits. Les microprocesseurs actuels (en 2007) peuvent traiter des nombres sur **64** bits en une seule instruction.
- **La vitesse maximale de l'horloge** qu'il peut supporter. Le rôle de l'horloge est de cadencer le rythme du travail du microprocesseur. Plus la vitesse de l'horloge augmente, plus le microprocesseur exécute d'instructions en une seconde.

Le tableau suivant décrit les principales caractéristiques des **microprocesseurs** (unités centrales) fabriqués par Intel et montre la fulgurante évolution des microprocesseurs autant en augmentation du nombre de transistors, en miniaturisation des circuits et en augmentation de puissance.

Table 3.2 : *Évolution des microprocesseurs.*

Date	Nom	Nombre de Transistors	Finesse de gravure (µm)	Fréquence de l'horloge	Largeur des données	MIPS
1971	4004	2 300			4 bits/4 bits bus	
1974	8080	6 000	6	2 MHz	8 bits/8 bits bus	0,64
1979	8088/8086	29 000	3	5 MHz	16 bits/8 bits bus	0,33
1982	80286	134 000	1,5	6 MHz	16 bits/16 bits bus	1
1985	80386	275 000	1,5	16 MHz	32 bits/32 bits bus	5
1989	80486	1 200 000	1	25 MHz	32 bits/32 bits bus	20
1993	Pentium	3 100 000	0,8	60 MHz	32 bits/64 bits bus	100
1997	Pentium II	7 500 000	0,35	233 MHz	32 bits/64 bits bus	300
1999	Pentium III « !!! »	9 500 000	0,25	450 MHz	32 bits/64 bits bus	510
2000	Pentium 4C	42 000 000	0,18	1,5 GHz	32 bits/64 bits bus	1 700
2004	Pentium 4D « Prescott »	125 000 000	0,09	3,6 GHz	32 bits/64 bits bus	9 000
2006	Core 2™ Duo	291 000 000	0,065	2,4 GHz (E6600)	64 bits/64 bits bus	22 000
2007	Core 2™ Quad	2*291 000 000	0,065	3 GHz (Q6850)	64 bits/64 bits bus	2*22 000 (?)
2008	Core 2™ Duo (Penryn)	410 000 000	0,045	3,16 GHz (E8500)	64 bits/64 bits bus	~24 200
2008	Core 2™ Quad (Penryn)	2*410 000 000	0,045	3,2 GHz (QX9770)	64 bits/64 bits bus	~2*24 200

3.3.1.1. Unité de traitement

C'est l'unité de calcul ou tout simplement **Unité Arithmétique et Logique (UAL).**

- ✓ Elle contient tous les circuits électroniques qui réalisent effectivement les opérations désirées.
- ✓ Ces opérations sont principalement les opérations arithmétiques (l'addition, la soustraction, la multiplication, la division) et les opérations logiques (Inversion des bits, ET, OU, OU exclusif).
- ✓ Des registres et des indicateurs sont associés à l'UAL :
  - ⌖ Les registres servent à contenir les opérandes et les résultats intermédiaires.
  - ⌖ Les indicateurs (**Registre d'état**) pour indiquer l'état du résultat : Résultat nul, >0, <0, débordement, …etc.
- ✓ Ces registres sont accessibles aux programmeurs (l'utilisateur).

En général, pour réaliser une opération, il faut connaitre trois adresses :

1. Adresse du premier opérande.
2. Adresse du deuxième opérande.
3. L'adresse à laquelle on doit ranger le résultat (dans les registres ou dans la MC).

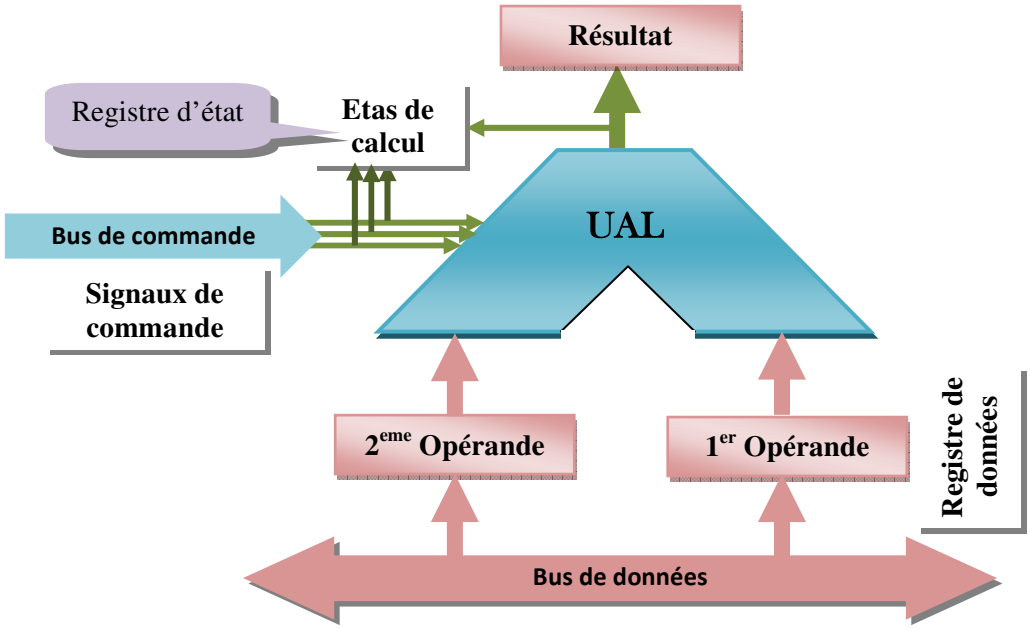


Figure 3.8 : *Structure de l'unité arithmétique et logique.*

Le résultat de l'opération est mis dans un registre spécial (l'**accumulateur** (**ACC**)) dans le cas d'une machine à une adresse, dans des registres internes dans le cas de plusieurs adresses, ou dans la mémoire centrale dans certaines situations → **voir partie exécution d'une instruction**.

Table 3.3 : *Modes d'accès aux opérandes et aux résultats.*

Opérande 1	Opérande 2	Résultat	Mode d'dressage	
Registre G	Registre G	Registre G Accumulateur	Adressage par registre	**Pas d'accès à la MC**
Accumulateur	Registre G	Registre G Accumulateur	Adressage par registre	
Registre G	Accumulateur	Registre G Accumulateur	Adressage par registre	
Registre G / ACC	Mémoire	Registre G Accumulateur	Adressage Mémoire	**Accès à la MC (Lecture – Ecriture)**
Mémoire	Registre G / ACC	Registre G Accumulateur	Adressage Mémoire	
Mémoire	Registre G / ACC	Mémoire	Adressage Mémoire	
Registre G / ACC	Mémoire	Mémoire	Adressage Mémoire	

3.3.1.2. L'unité de commande

L'unité de commande est chargée de commander et de gérer tous les différents constituants de l'ordinateur (contrôler les échanges, gérer l'enchaînement des différentes instructions, etc...). en d'autres termes, son rôle est de coordonner et de synchroniser toutes les actions (micro opérations) durant l'exécution des tâches. Elle est composée au minimum de :

- D'un registre instruction (**RI**),
- D'un compteur ordinal (**CO**),
- D'un séquenceur ou générateur de séquences (**GS**) (décodeur de fonctions).

- D'une horloge.

- Le **registre d'instruction** qui contient l'instruction en cours d'exécution (une instruction comporte plusieurs champs : un champ code-opération et entre 0 et 3 champs-opérande).
- Le **compteur ordinal** (CO) [**PC :** Program Counter ou **IP** : Instruction Pointer] qui contient l'adresse de la prochaine instruction à exécuter (qu'il faut aller chercher en mémoire (partie des programmes)).
- Le **générateur de séquences** un dispositif de décodage des instructions (décodeur) et un séquenceur de commandes qui active les circuits nécessaires à l'exécution de l'instruction en cours. Cette unité a besoin des signaux d'une **horloge** pour enchaîner les commandes.
- **L'horloge** est généralement externe à l'unité.

Les registres de l'unité de commande ne sont pas directement accessibles aux programmeurs.

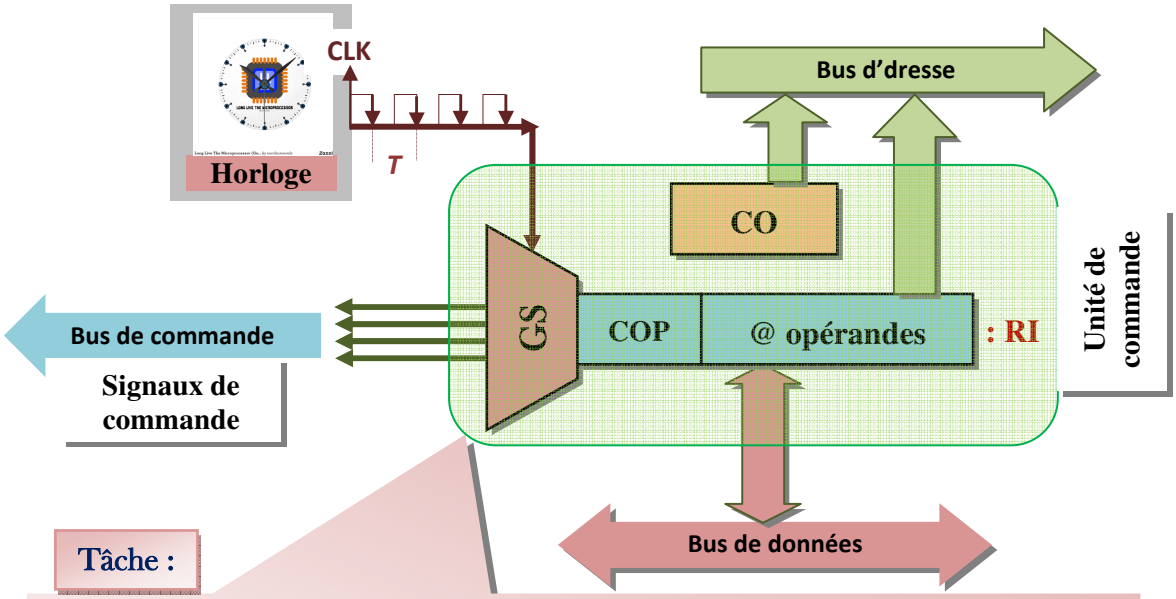


Figure 3.9 : *Structure de l'unité de commande.*

3.3.3. Unité d'échange (UE)

L'UE gère en entrée comme en sortie le transfert d'un ensemble d'informations entre la l'unité / mémoire centrale et les unités périphériques. Les unités périphériques se répartissent en deux classes :

- ✓ Les unités qui permettent à l'ordinateur d'échanger des données avec l'extérieur (écran, clavier, imprimante, modem, …).
- ✓ Les **mémoires auxiliaires** (disques, bandes, cartouches magnétiques, disques SSD, …) qui permettent de stocker de façon permanente beaucoup d'informations à un moindre coût (mémoire de masse). Elles sont utiles car la mémoire centrale est volatile et les informations s'effacent quand on éteint la machine, tandis que ces supports auxiliaires sont des mémoires permanentes.

À chaque catégorie d'unités périphériques est associé un **contrôleur de périphériques** qui s'occupe de la **gestion** de ces unités et de l'interface avec les unités d'E/S.

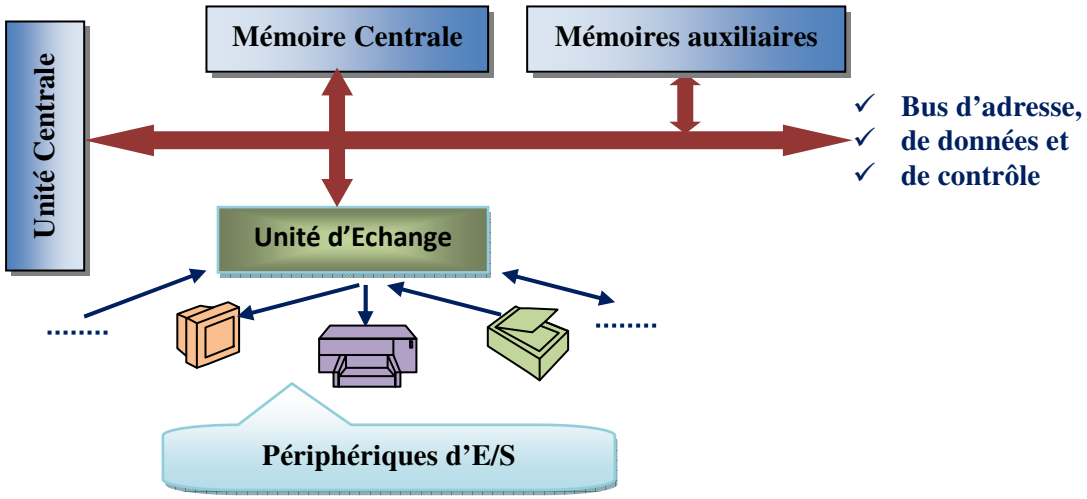


Figure 3.10 : *Structure de l'unité d'échange.*

Pour piloter les périphériques, l'unité d'entrées-sorties disposera

- ✓ D'un registre mémorisant l'adresse du périphérique (**X**),
- ✓ Le **registre de sélection du périphérique** (**Sel_Perph**), et
- ✓ D'un registre permettant l'échange d'informations entre unité centrale et les périphériques, le **registre d'échange** (**RE**) à la manière du registre mot de l'unité de mémoire.

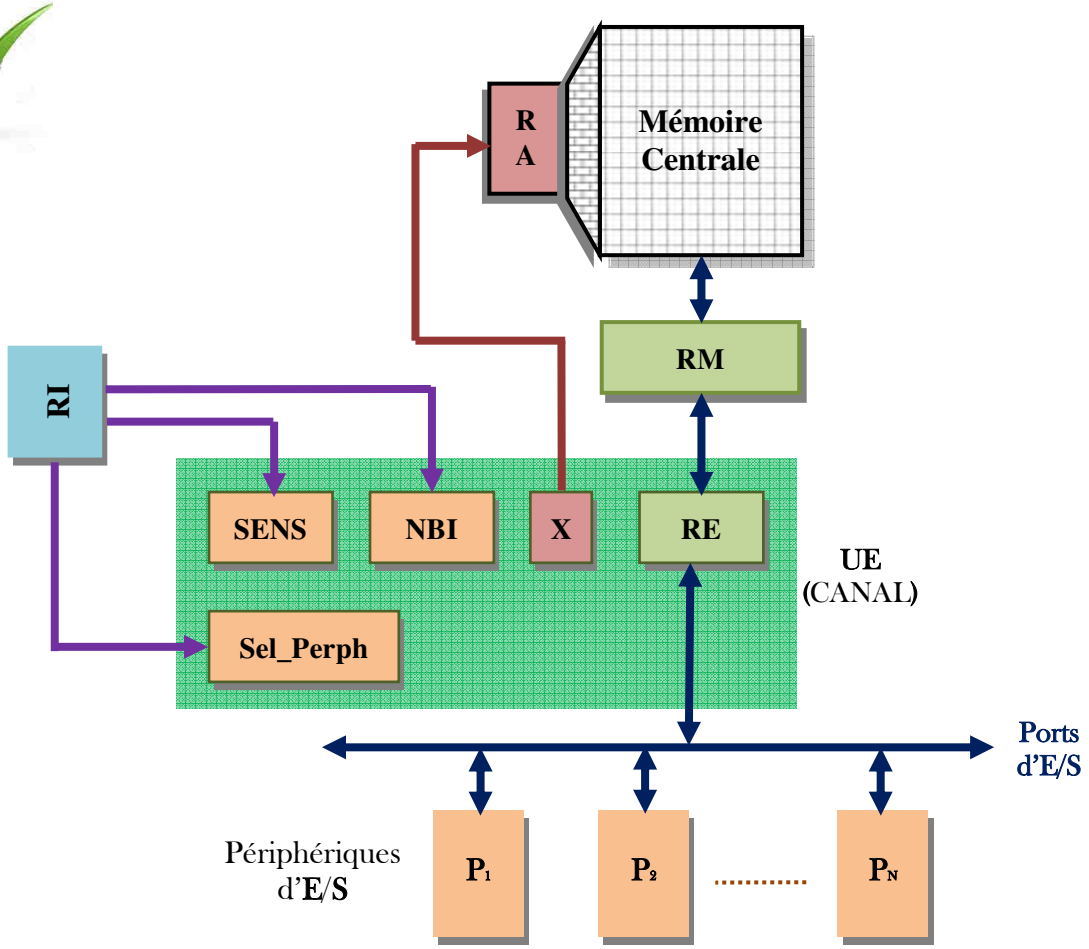


Figure 3.11 : *Fonctionnement de l'unité d'échange.*

Table 3.4 : *Eléments de l'UE.*

Elément	Description	Rôle
SENS	Sens de transfert des données	✓ Pour réaliser le transfert des informations, l'UE (CANAL) doit connaitre : ▪ Le **SENS** du transfert et ▪ L'adresse du l'unité périphérique concernant d'une part l'adresse de rangement (**X**) de la première information et le nombre d'informations (**NBI**) à transférée d'autre part. ✓ Ces informations sont fournis par une instruction et sont transférées dans l'UE qui peut alors sélectionnées sans l'intervention de l'unité de commande.
NBI	Nombre d'informations à transférer	
X	Adresse de rangement dans la MC	
RE	Registre d'échange	
Sel_Perph	**Sélection du périphérique**	
$P_1, P_2, \ldots, P_N$	Périphériques d'**E/S**	
RA	Registre d'adresse	
RM	Registre mot	
RI	Registre d'instruction	

- L'UE est un organe de calcul qui incrémenté de 1l'adresse du registre et décrémente de 1 le nombre d'informations à chaque transfert.
- Elle s'arrête quand toutes les informations ont été transférées, c-à-d, quand le nombre d'information **NBI = 0** (Echange RM et RE).

3.4. Fonctionnement de l'unité de centrale

Le fonctionnement de l'unité centrale est schématiquement toujours le même, même si aujourd'hui il est de plus en plus complexe. Il suit les étapes suivantes :

1. Aller chercher l'instruction à exécuter dans la mémoire morte (qui est la **mémoire de programme**).
2. Aller chercher la (ou les) donnée(s) (si besoin car parfois ce n'est pas nécessaire) sur laquelle l'instruction doit opérer en mémoire vive ou en registres.
3. **Exécuter** l'instruction.
4. **Ranger** le résultat dans la mémoire vive ou dans un registre.
5. Recommencer.

Cette suite d'opérations se répète jusqu'à la fin du programme. Donc, pour exécuter une instruction, 2 cycles se succèdent :

i. Cycle de **recherche de l'instruction** (cycle FETCH).
ii. Cycle **d'exécution** de l'instruction.

3.4.1. Format d'une instruction

✓ Chaque processeur (UC) possède un certain **nombre limité d'instructions** qu'il peut exécuter. Ces instructions s'appellent **Jeu d'instructions**. Le Jeu d'instructions décrit l'ensemble des opérations élémentaires que l'UC peu exécuter.

✓ Les instructions peuvent être classifiées en 5 catégories :
- Instructions arithmétiques et logiques.

- Instructions d'affectations.
- Instructions de branchement.
- Instructions de transfert.
- Instructions d'entrées/sorties.

✓ Une instruction est composée de deux champs :

- Un champ **Code opération** (**COP**) représentant l'action que le processeur doit effectuer.
- Un champ des **opérandes** ou les **adresses des opérandes** (**@ opérandes**) définissant les paramètres de l'action (emplacement des opérandes). Un opérande peut s'agir d'une donnée ou bien d'une adresse mémoire.

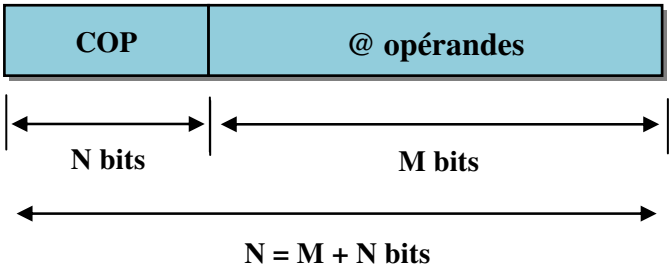


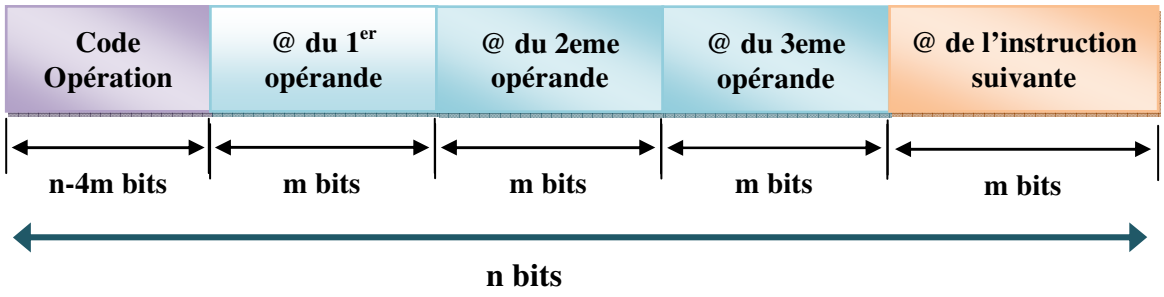
Figure 3.12 : *Format d'une instruction.*

- La taille de l'instruction dépend du type de l'instruction et du type des opérandes.
- Les instructions sont stockées dans la mémoire centrale (Partie des programmes) et les données sont stockées soit dans la mémoire centrale (Partie des données) soit dans des registres.

✓ Le format d'une instruction varie d'une machine à une autre. Une machine peut avoir des instructions avec des formats differents. Il est perfois fait référence à une machine par le nombre des champs adresses contenus dans ses instructions :

- Machine à **0** adresse.
- Machine à **1** adresse.
- Machine à **2** adresses.
- Machine à **3** adresses.
- Machine à **4** adresses.

3.4.1.1. Format à 4 adresses



Exemple : **ADD X, Y, Z, W;**

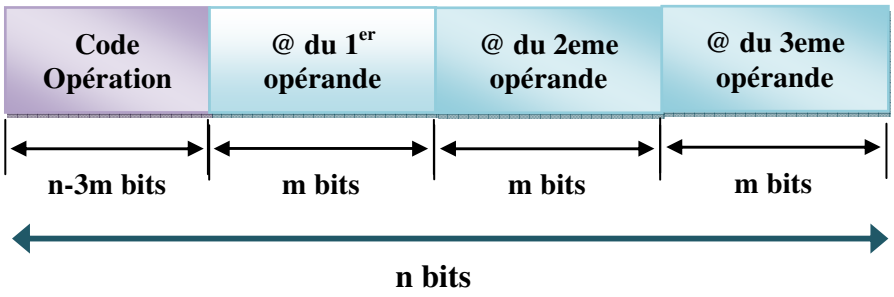
L'instruction signifie: (**Z**) ← (**X**) + (**Y**) , où:

✓ Additionner les contenus des cellules de mémoire d'adresse **X** et **Y**,

- ✓ Placer le résultat de l'addition dans la cellule d'adresse **Z**,
- ✓ Aller chercher la prochaine instruction à la cellule d'adresse **W**.

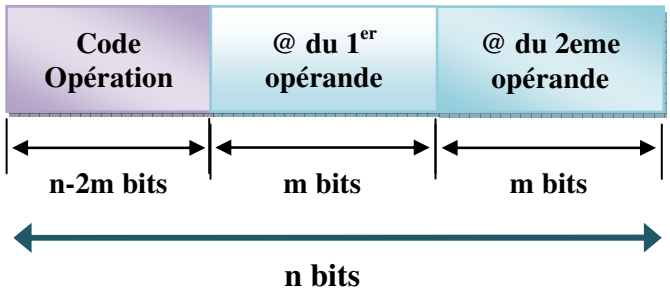
Les instructions à 4 adresses sont très rares, car elles sont très longues. Les constructeurs préfèrent prévoir un compteur ordinal dans l'unité de contrôle qu'un opérande servant à indiquer l'adresse de la prochaine instruction.

3.4.1.2. Format à 3 adresses



- ✓ Ces instructions ont le même format que celles utilisées dans l'exemple de programme vus précédemment.
- ✓ Un processeur fonctionnant avec de telles instructions localise l'instruction suivante à l'aide d'un registre **compteur ordinal**.
- ✓ Le **compteur ordinal** doit comporter m bits de position, car il est destiné à recevoir des adresses.

3.4.1.3. Format à 2 adresses

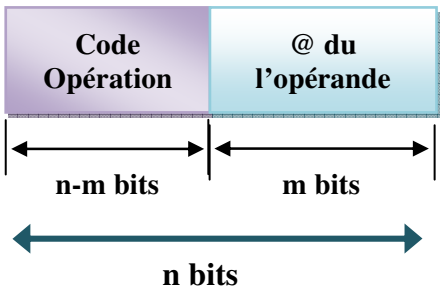


- ✓ Les instructions à 2 adresses sont les plus courantes.
- ✓ L'adresse du résultat est implicitement choisie: celui-ci est rangé à l'adresse du 2ième opérande ou à celle du premier opérande selon le choix fait par le constructeur.

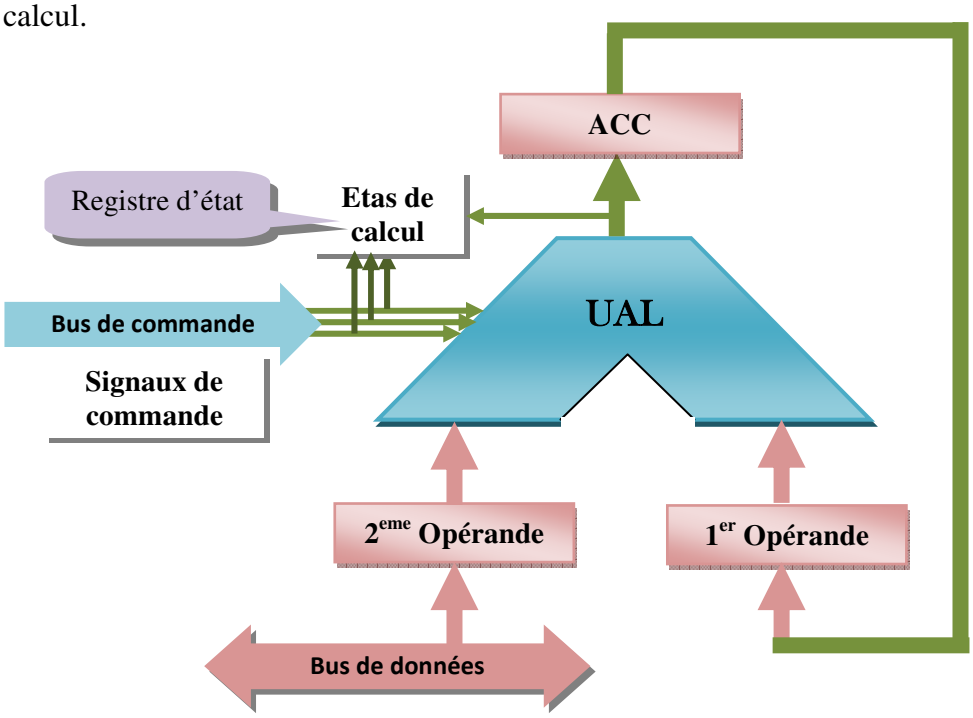
Exemple: L'expression **(X) = (A + B * C) / D** peut être programmée de la façon suivante:

MUL B, C;	(C) ← (B) * (C)
ADD A, C ;	(C) ← (A) * (C)
DIV C, D ;	(D) ← (C) / (D)
MOV D, X ;	(X) ← (D)

3.4.1.4. Format à 1 adresse



- ✓ Une instruction à une adresse nécessite non seulement un compteur ordinal, mais aussi un registre supplémentaire pour stocker temporairement les valeurs intermédiaires pendant le calcul.
- ✓ Ce registre, appelé **accumulateur** (**ACC**), peut contenir la donnée source ou le résultat du calcul.



Exemple: Le calcul de l'expression **(X) = (A + B * C) / D** peut être programmé de la façon suivante:

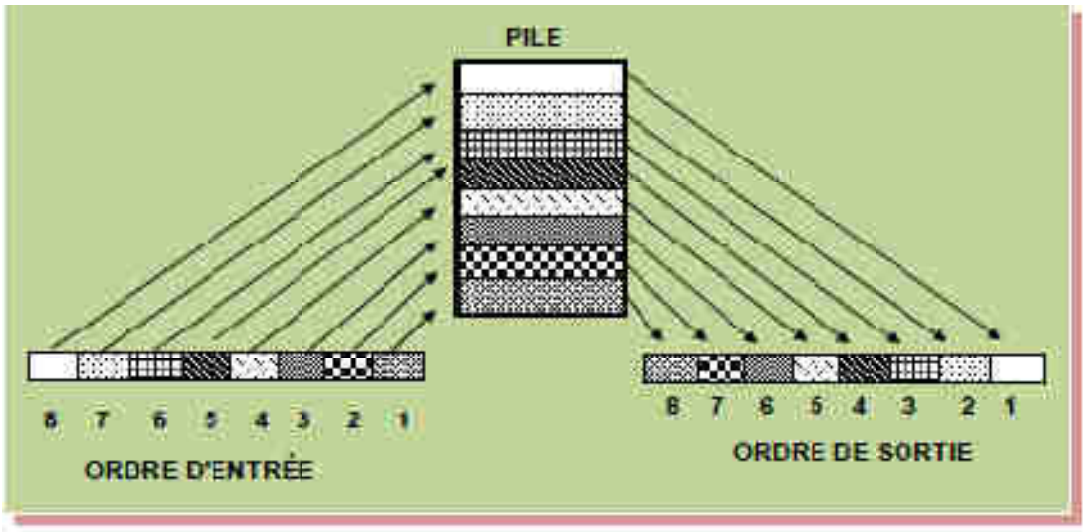
LOAD B ;	ACC ← (B)
MUL C;	ACC ← Contenu de l'ACC. + (C)
ADD A;	ACC ← Contenu de l'ACC. + (A)
DIV D;	ACC ← Contenu de l'ACC. / (D)
STORE X;	(X) ← Contenu de l'ACC.

Remarque: Les instructions de rupture inconditionnelle sont un exemple d'instruction à 1 opérande.

3.4.1.5. Format à 0 adresse

- ✓ Le processeur à **0** adresse possède un accumulateur particulier qu'on nomme pile.
- ✓ Une pile est un rangement par ordre chronologique.
- ✓ Contrairement à un accumulateur ordinaire, placer une donnée dans une pile ne détruit pas son contenu précédent: celui-ci sera plutôt repoussé au fond de la pile.
- ✓ Généralement, l'accès aux données se fait dans l'ordre inverse où celles-ci ont été rangées dans la pile; on désigne souvent ce type d'accès par le sigle anglais LIFO (Last In, First Out).

- ✓ Cela implique qu'on a accès qu'à la donnée qui est au sommet de la pile (soit la dernière entrée) ou aux données se trouvant dans des emplacements consécutifs à partir du sommet.



- ✓ L'instruction à 0 adresse permet de modifier la pile en effectuant une opération sur les deux données à partir du sommet, et en plaçant le résultat de l'opération au sommet de la pile.

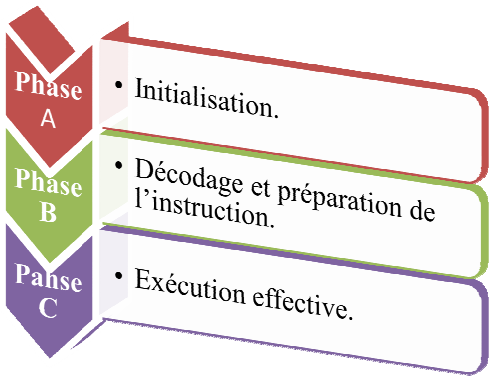
Exemple: Pour ranger des données dans la pile ou pour les en retirer, on utilise deux instructions :

PUSH A: Place le contenu de A dans la pile (et pousse les contenus précédents vers le fond de la pile).

POP A: retire la valeur située au sommet de la pile et la range à l'adresse A (et fait "remonter" les contenus précédents d'une position vers le sommet)

3.4.2. Etapes d'exécution d'une instruction

L'exécution d'une instruction nécessite trois phases :



- ✓ Chaque phase comporte un certain nombre **d'opérations élémentaires** dans un ordre précis par l'unité de commande.
- ✓ La communication entre les différents blocs et registres est assurée par les **BUS**.

3.4.2.1. Phase A : Initialisation

L'initialisation où tout simplement la lecture de l'instruction à partir de la mémoire centrale. Le déroulement de cette phase est illustré par le tableau ci-dessous.

Table 3.5 : *Phase d'initialisation d'une instruction.*

Phase élémentaire (A)	Micro opération	Description
(A1)	**RA ← (CO)**	✓ L'adresse de l'instruction en cours d'exécution sera transférée du CO vers le registre d'adresse (RA).
(A2)	**CO ← (CO) + 1**	✓ Le CO s'incrémente et pointe vers la prochaine instruction à exécuter.
(A3)	**RM ← (RA)**	✓ Lecture de l'instruction à partir de l'adresse spécifiée par CO. Avant de l'accès, il faut passer par le registre RA. L'instruction récupérée sera transférée vers le registre de données RM associé à la MC.
(A4)	**RI ← (RM)**	✓ L'instruction récupérée sera transférée du registre de données RM vers le registre RI dans l'unité de commande.

- La phase d'initialisation est **obligatoire** quelque soit le type de l'instruction.
- Une **Micro opération** ((**A1**), (**A2**), …) est une opération élémentaire effectuée pendant une impulsion (Top) d'horloge (ou cycle machine) sur l'information stockée dans un emplacement mémoire (registre, case mémoire)

3.4.2.2. Phase B : Décodage et préparation de l'instruction

Cette phase consiste à l'analyse et préparation de l'exécution de l'instruction en cous d'exécution. Le déroulement de cette phase est illustré par le tableau ci-dessous.

Table 3.6 : *Phase de décodage et préparation de l'instruction.*

Phase élémentaire (B)	Micro opération	Description
(B1)	**GS ← COP**	✓ Le code opération (COP) de l'instruction est transmis vers le générateur de séquences dans l'unité de commande qui s'occupe du décodage et de génération des commandes adaptées à cette instruction.
(B2)	**RA ← @ OP**	✓ La partie adresses des opérandes (@ **OP**) de l'instruction qui donne l'adresse des opérandes est transférée vers le registre RA associé à la MC.
(B3)	**RM ← (RA)**	✓ Lecture de l'opérande à partir de la partie des données de la MC et le transfert vers l'UAL pour l'exécution de l'instruction.

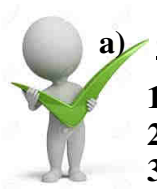
3.4.2.3. Phase C : Exécution effective de l'instruction

Phase d'exécution effective de l'instruction en cours et préparation de l'exécution de la prochaine instruction. Le déroulement de cette phase est illustré par le tableau ci-dessous.

Table 3.7 : *Phase d'exécution effective de l'instruction.*

Phase élémentaire (C)	Micro opération	Description
(C)	ACC ← (ACC) COP (RM)	✓ Le code opération (COP) de l'instruction est transmis vers le générateur de séquences dans l'unité de commande qui s'occupe du décodage et de génération des commandes adaptées à cette instruction.

Exemple dans le cas d'une opération utilisant une seule opérande en mémoire centrale



a) <u>Cycle de recherche (fetch) :</u> Initialisation et préparation de l'instruction

1. Le **CO** contient l'adresse de l'instruction à exécuter : On la place dans **RA**.
2. Transfert de la donnée stockée en mémoire à l'adresse indiquée dans **RA** dans **RM**.
3. La donnée stockée dans **RM** est placée dans **RI**.
   - ✓ Cette donnée est le code de l'opération à exécuter + adresse de l'opérande.
4. Le décodeur lit cette donnée et détermine
   - ✓ L'adresse en mémoire de l'opérande.
   - ✓ Le type d'opération qui doit être exécutée.
5. Le décodeur place dans **RA** l'adresse où est stocké l'opérande en mémoire.
6. Le décodeur informe le séquenceur du type d'opération à exécuter.
7. **CO** est incrémenté pour pointer vers la prochaine instruction à rechercher

b) <u>Cycle d'exécution</u>

8. L'opérande est lue en mémoire et placée dans **RM**.
9. Le contenu de **RM** est placé dans un accumulateur (**ACC**) ou un registre qui stocke les données en paramètres des opérations.
10. Le séquenceur demande à l'UAL d'exécuter l'opération.

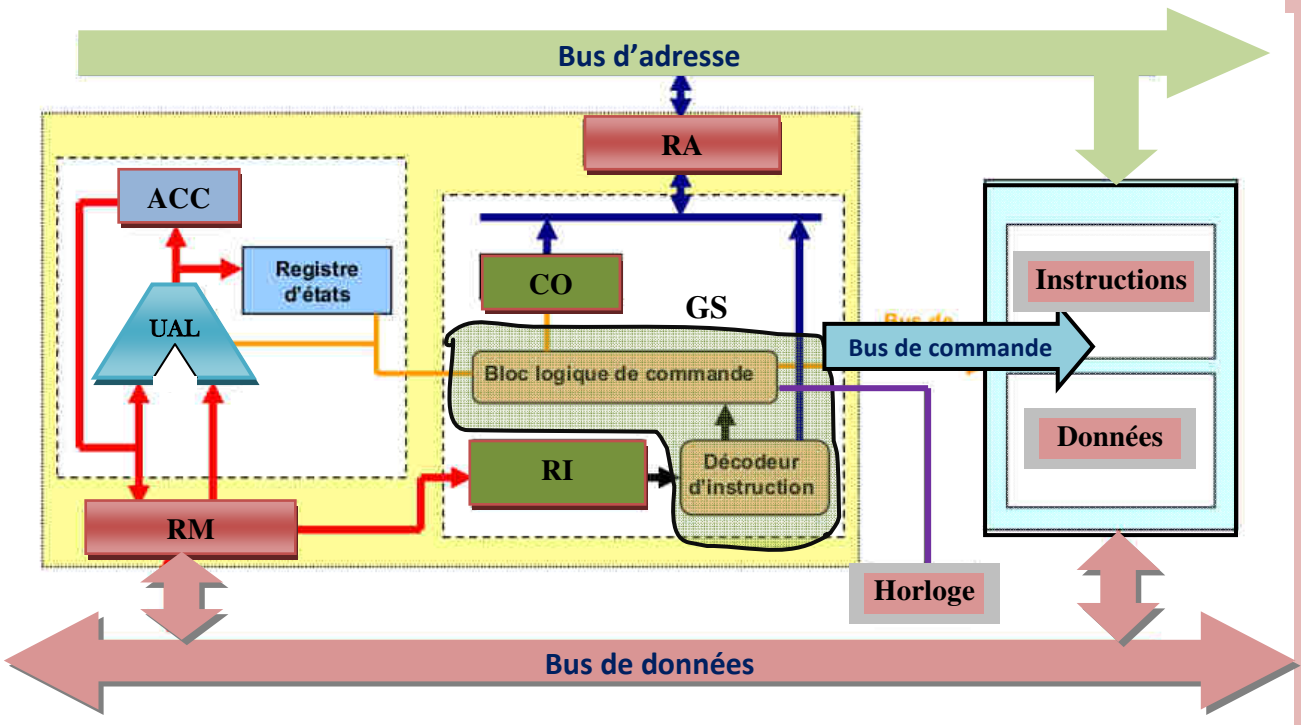


Figure 3.13 : *Cheminement d'exécution des instructions.*

Remarques : Cycle d'exécution, variantes selon les instructions :

- ✓ Si l'opération de branchement (saut dans la séquence d'instructions) :
  - ✓ On place le contenu de **RM** dans **CO**.
  - ✓ Pas de calcul à réaliser.
- ✓ Si l'opération d'écriture en mémoire du résultat d'une opération précédente :
  - ✓ On place le contenu d'un registre de l'accumulateur dans RM pour l'écrire ensuite en mémoire.
  - ✓ Pas de calcul à réaliser.

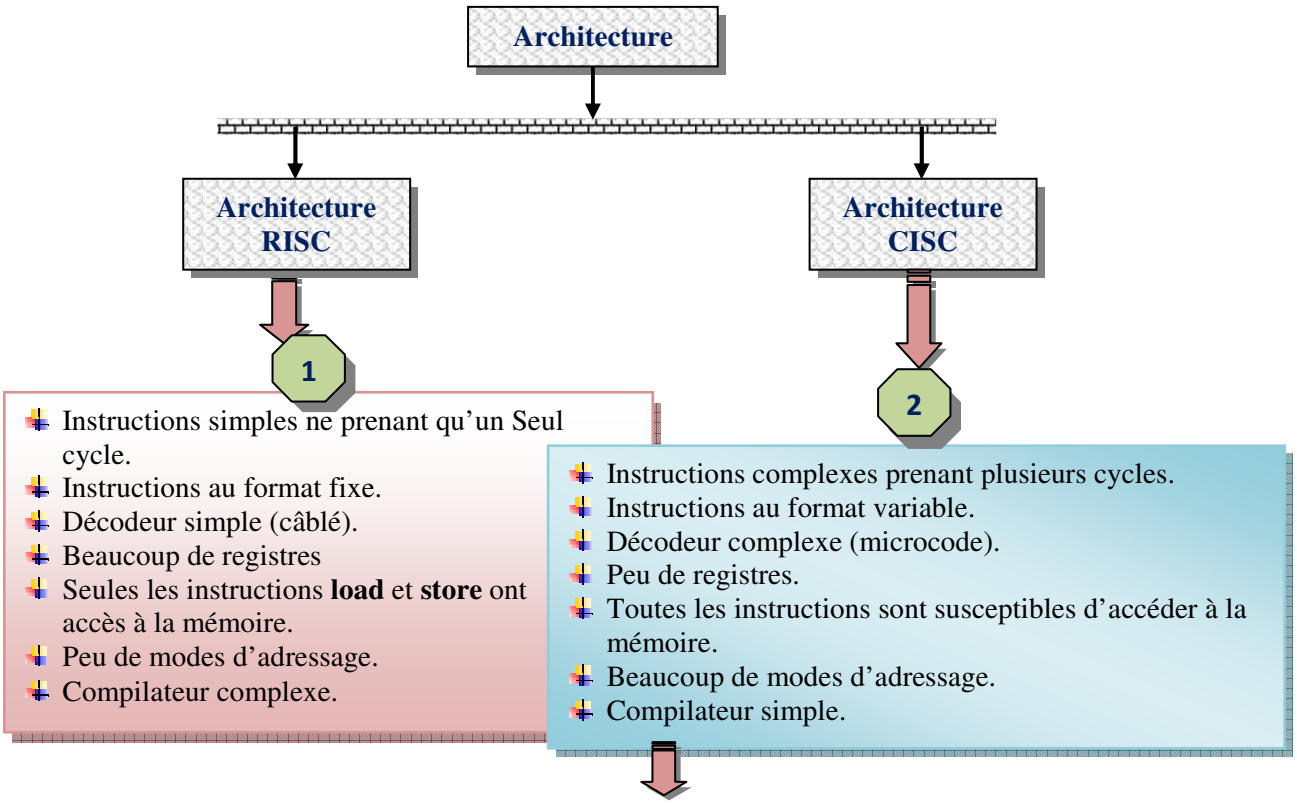
3.5. Notions d'architecture RISC et CISC

Actuellement l'architecture des microprocesseurs se composent de deux grandes familles :

- ✓ L'architecture **CISC** (Complex Instruction Set Computer).
- ✓ L'architecture **RISC** (Reduced Instruction Set Computer)

Le choix dépendra des applications visées. En effet,

- ➢ Si on diminue le nombre d'instructions, on crée des instructions complexes (**CISC**) qui nécessitent **plus de cycles** pour être décodées et
- ➢ Si on diminue le nombre de cycles par instruction, on crée des instructions simples (RISC) mais on augmente alors le nombre d'instructions nécessaires pour réaliser le même traitement.



Améliorations de l'architecture de base

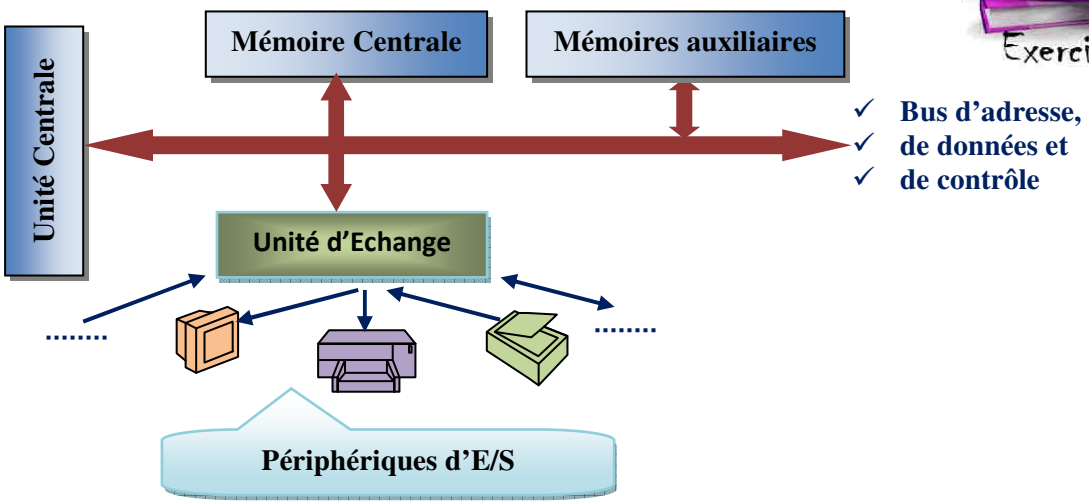
L'ensemble des améliorations des microprocesseurs visent à diminuer le temps d'exécution du programme.

- La première idée qui vient à l'esprit est d'augmenter tout simplement la fréquence de l'horloge du microprocesseur.
- Mais l'accélération des fréquences provoque un surcroît de consommation ce qui entraîne une élévation de température.
- On est alors amené à équiper les processeurs de systèmes de refroidissement ou à diminuer la tension d'alimentation.
- Une autre possibilité d'augmenter la puissance de traitement d'un microprocesseur est de diminuer le nombre moyen de cycles d'horloge nécessaire à l'exécution d'une instruction.
- Dans le cas d'une programmation en langage de haut niveau, cette amélioration peut se faire en optimisant le compilateur.
- Il faut qu'il soit capable de sélectionner les séquences d'instructions minimisant le nombre moyen de cycles par instructions.
- Une autre solution est d'utiliser une architecture de microprocesseur qui réduise le nombre de cycles par instruction

3.7. Exercices

EXERCICE 01 :

Soit le schéma de la figure suivante :

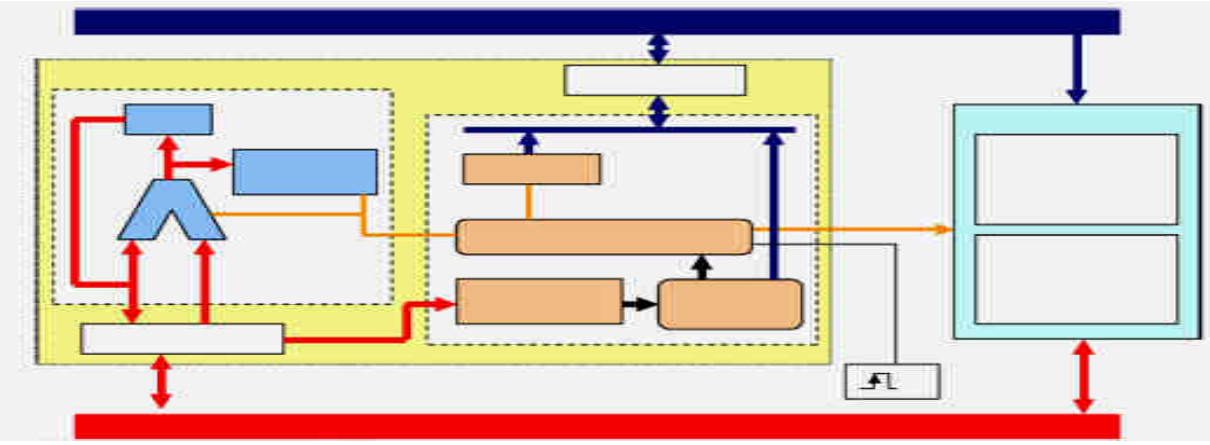


- Expliquer le rôle de chaque bloc.
- Matérialiser électriquement les bus (d'adresse, de données et de commande).
- Quel est le rôle de l'Unité d'Echange (UE) d'un ordinateur.
- Expliquer le processus d'échange (les micro-opérations) :

  Ordinateur ←-→ Environnement extérieur (Périphériques).

EXERCICE 02 :

Complétez le schéma fonctionnel ci-dessous. Quel est le rôle fondamental de ce schéma bloc. Discuter…



Utiliser le format d'instruction suivant :

COP	@ opérandes

COP : Code Opération
@ opérandes: Adresse de l'opérande

pour expliquer le cheminement des informations pour l'exécution des instructions suivantes :

- **ADD ACC, B** ; /* **B** est un registre et **ACC** est l'Accumulateur */
- **ADD ACC, (Adr)** ; /* **Adr** est une adresse dans la mémoire centrale et **ACC** est l'Accumulateur. **(Adr)** est le contenu de **Adr** */
- **ADD (Adr), Acc ;**
- **MOV Acc, B ;**
- **MOV (Adr), Acc ;**
- **NOP ;** /* Aucune opération */

Donner le nombre des micro-opérations nécessaires à l'exécution des instructions précédentes.

CHAPITRE **4**

Etude d'un microprocesseur

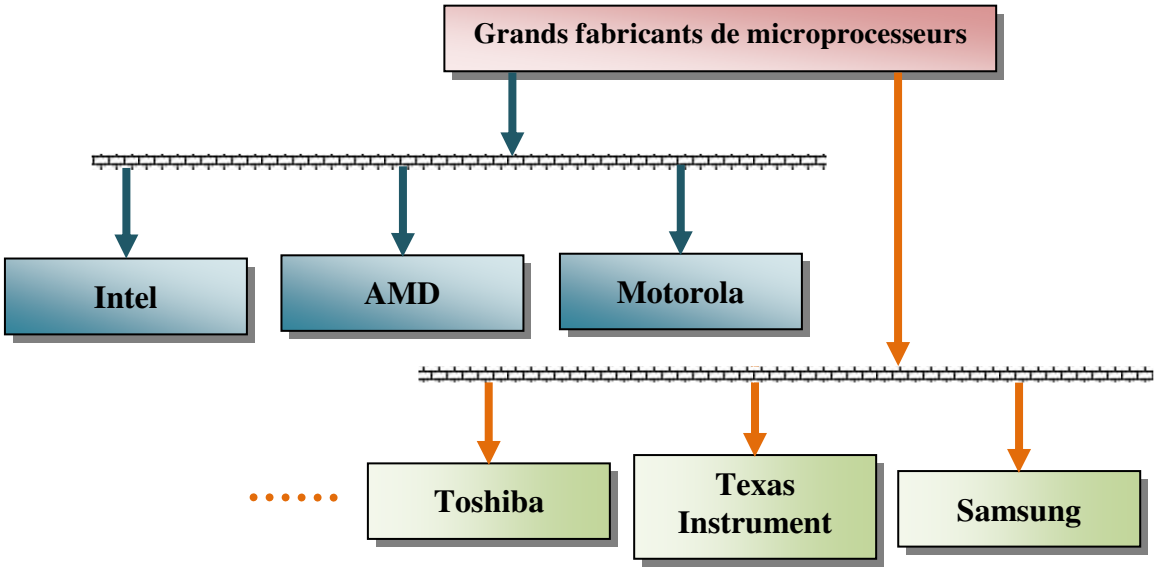
CHAPITRE 4

ETUDE D'UN MICROPROCESSEUR

4.1. Généralités

Le microprocesseur, (ou **CPU**) est le composant essentiel d'un ordinateur qui interprète les instructions et traite les données d'un programme.

- ✓ C'est un des composants nécessaires au fonctionnement de tous les types d'ordinateurs.
- ✓ C'est l'unité **intelligente** de traitement des informations.
- ✓ Son travail consiste à lire des programmes (des suites d'instructions), à les décoder et à les exécuter.
- ✓ Les années 80 voyaient l'émergence de ces circuits avec les Zylog Z80, 6800 de Motorola, le 8085 d'Intel qui est souvent utilisé en tant que microcontrôleur.
- ✓ Avec l'arrivée des PC-XT d'IBM et l'utilisation du 8088, Intel devenait maître du marché fin des années 80.
- ✓ Il existe des processeurs basés sur l'architecture CISC et d'autres basés sur l'architecture RISC.
- ✓ Certains processeurs sont difficilement classifiables comme le CPU i486 également appelé 80486.



L'histoire des microprocesseurs est intimement liée à celle de la technologie des semi-conducteurs. Le tableau suivant décrit les principales caractéristiques des microprocesseurs (unités centrales) fabriqués par Intel et montre la fulgurante évolution des microprocesseurs

autant en augmentation du nombre de transistors, en miniaturisation des circuits et en augmentation de puissance.

Table 4.1 : *Évolution des microprocesseurs.*

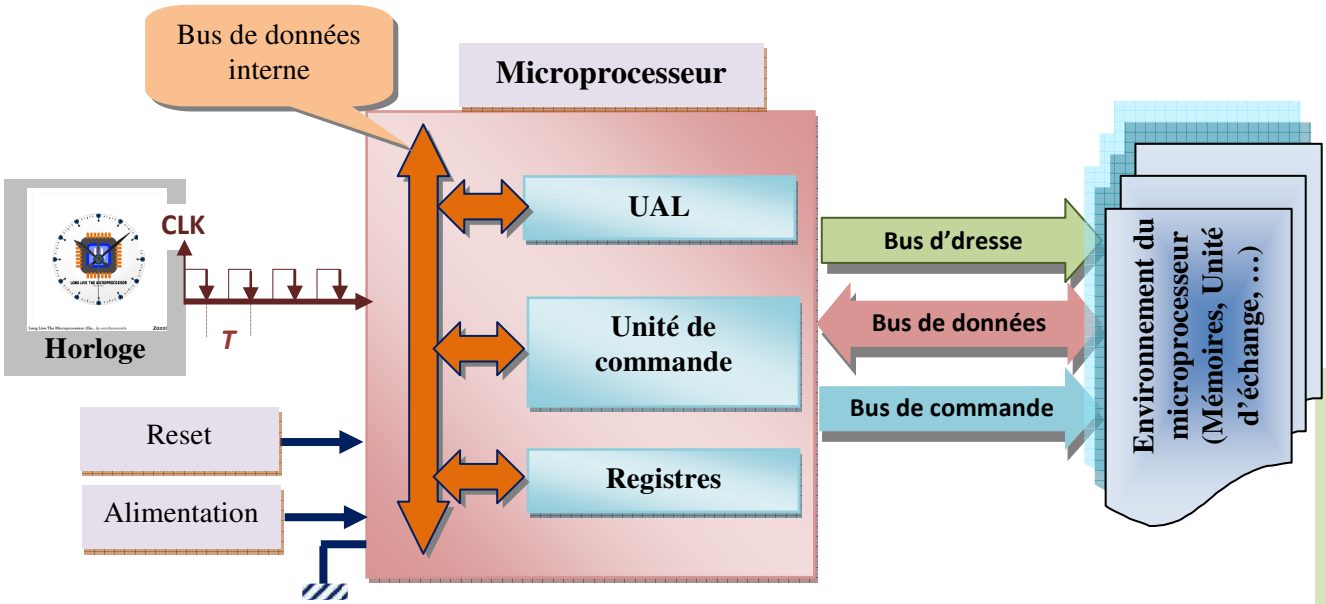
Date	Nom	Nombre de Transistors	Finesse de gravure (µm)	Fréquence de l'horloge	Largeur des données	MIPS
1971	4004	2 300			4 bits/4 bits bus	
1974	**8080 / 8085**	**6 000**	**6**	**2 MHz**	**8 bits/8 bits bus**	**0,64**
1979	8088/8086	29 000	3	5 MHz	16 bits/8 bits bus	0,33
1982	80286	134 000	1,5	6 MHz	16 bits/16 bits bus	1
1985	80386	275 000	1,5	16 MHz	32 bits/32 bits bus	5
1989	80486	1 200 000	1	25 MHz	32 bits/32 bits bus	20
1993	Pentium	3 100 000	0,8	60 MHz	32 bits/64 bits bus	100
1997	Pentium II	7 500 000	0,35	233 MHz	32 bits/64 bits bus	300
1999	Pentium III « !!! »	9 500 000	0,25	450 MHz	32 bits/64 bits bus	510
2000	Pentium 4C	42 000 000	0,18	1,5 GHz	32 bits/64 bits bus	1 700
2004	Pentium 4D « Prescott »	125 000 000	0,09	3,6 GHz	32 bits/64 bits bus	9 000
2006	Core 2™ Duo	291 000 000	0,065	2,4 GHz (E6600)	64 bits/64 bits bus	22 000
2007	Core 2™ Quad	2*291 000 000	0,065	3 GHz (Q6850)	64 bits/64 bits bus	2*22 000 (?)
2008	Core 2™ Duo (Penryn)	410 000 000	0,045	3,16 GHz (E8500)	64 bits/64 bits bus	~24 200
2008	Core 2™ Quad (Penryn)	2*410 000 000	0,045	3,2 GHz (QX9770)	64 bits/64 bits bus	~2*24 200

- **Largeur des données** : Le premier nombre indique **la taille de bus de données**. Le second nombre indique **la taille de bus d'adresse**.
- **MIPS** : Le nombre de millions d'instructions complétées par le microprocesseur en une seconde.

Un microprocesseur est constitué de:

- ✓ Une unité de **commande** qui lit les instructions et les décode;
- ✓ une unité de **traitement** (**UAL** - unité arithmétique et logique) qui exécute les instructions;
- ✓ D'un ensemble de mémoire appelés **registres**;
- ✓ D'un **bus de données** externe;
- ✓ D'un **bus d'adresse** externe;
- ✓ D'un **bus de commande** externe;
- ✓ D'un bus de données interne reliant l'unité de commande, l'UAL et les registres.

Lorsque tous ces éléments sont regroupés sur une même puce, on parle alors de **Microprocesseur**. La figure 4.1 donne une idée sur l'architecture interne d'un microprocesseur. Sur cette figure nous pouvons voir les 3 bus qui permettent au microprocesseur de communiquer avec l'extérieur.



Les principales caractéristiques d'un microprocesseur sont :

- Le fabriquant du circuit (Intel, AMD, …).
- Le type de boitier DIP, PLCC, …).
- Technologies de fabrication : NMOS, PMOS, CMOS.
- La **complexité de son architecture → Echelle d'intégration**. Cette complexité se mesure par le nombre de **transistors** contenus dans le microprocesseur.
- Le nombre de **bits** que le processeur peut traiter en une instruction (**taille des registres internes du microprocesseur ou la taille de bus de données**) (**4, 8, 16, 32, ...**).
- La taille de bus d'adresse, qui permet de définir l'espace mémoire accessible par le microprocesseur.
- **Le jeu d'instructions** qu'il peut exécuter. Un processeur peut exécuter plusieurs douzaines d'instructions différentes.
- **La vitesse maximale de l'horloge** qu'il peut supporter → **Vitesse d'exécution des instructions**.
- **Consommation d'énergie.**
- …
- …
-

Figure 4.1 : *Architecture schématique d'un microprocesseur.*

4.2. Les différentes familles des microprocesseurs 8 bits

Les fabricants des microprocesseurs 8 bits les plus connus sont :

- ✓ Intel,
- ✓ Zilog,
- ✓ Motorola et
- ✓ National Semiconductor,

comme schématise la figure ci-dessous.

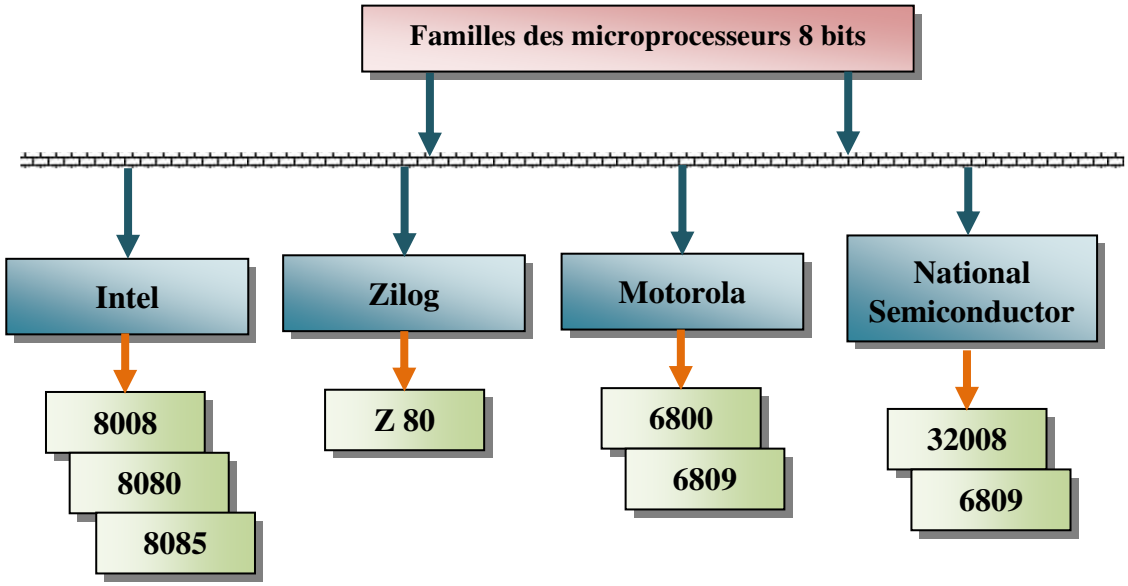


Figure 4.2 : *Les différentes familles des microprocesseurs 8 bits.*

4.3. Etude d'un microprocesseur 8 bits : 8088/8085

L'**Intel 8085** est un microprocesseur **8 bits** fabriqué par Intel au milieu des années 1970.

- ✓ Il était compatible au niveau du code binaire avec le plus célèbre Intel 8080, mais demandait moins de matériel environnant, ce qui permit la création de micro-ordinateurs plus simples et moins chers à construire.
- ✓ Disponible en version à 40 broches
- ✓ Le « **5** » dans le numéro du modèle provient du fait que les 8085 exigeaient seulement une alimentation de +5V plutôt que les +5V, -5V et +12V exigés par les 8080.
- ✓ Il existe en plusieurs versions 8085A, 8085AH, 8085AH-1 et 8085AH-2.
- ✓ Cependant, il était **plus lent** que le 8080.
- ✓ Fonctionnement à 3 MHz, 5 MHz et 6 MHz ;
- ✓ 1,3 µs par cycle d'instruction pour le 8085AH, 0,8 µs pour 8085AH-2 et 0,67 µs pour le 8085AH-1 ;
- ✓ Générateur d'horloge interne (avec quartz externe ou réseau LC ou RC) ;
- ✓ Possède 4 vecteurs d'interruption dont 1 non masquable
- ✓ Le **8088/8085** ont parfois été utilisés dans des ordinateurs basés sur le système d'exploitation CP/M.
- ✓ Ils furent par la suite supplantés par le **Zilog Z80**, compatible et plus efficace, qui remporta la majeure partie du marché des ordinateurs CP/M et des ordinateurs personnels du milieu et de la fin des années 1980.
- ✓ Le **8085** fut utilisé ultérieurement comme microcontrôleur (surtout grâce au coût réduit des composants).
- ✓ Ainsi, il équipait le dispositif à bandes DECtape et le terminal vidéo VT100.
- ✓ Il continua donc à être produit pendant toute la durée de vie de ces produits.
- ✓ De même, il fut embarqué sur le robot de la mission **Mars Pathfinder**.
- ✓ Il est actuellement encore utilisé dans l'enseignement.

4.3.1. Brochage du up8085

Le microprocesseur 8085 (8085A : version légèrement révisée du 8085) est fabriqué sous forme de boitier DIP à 40 broches comme illustrée par la figure 4.3.

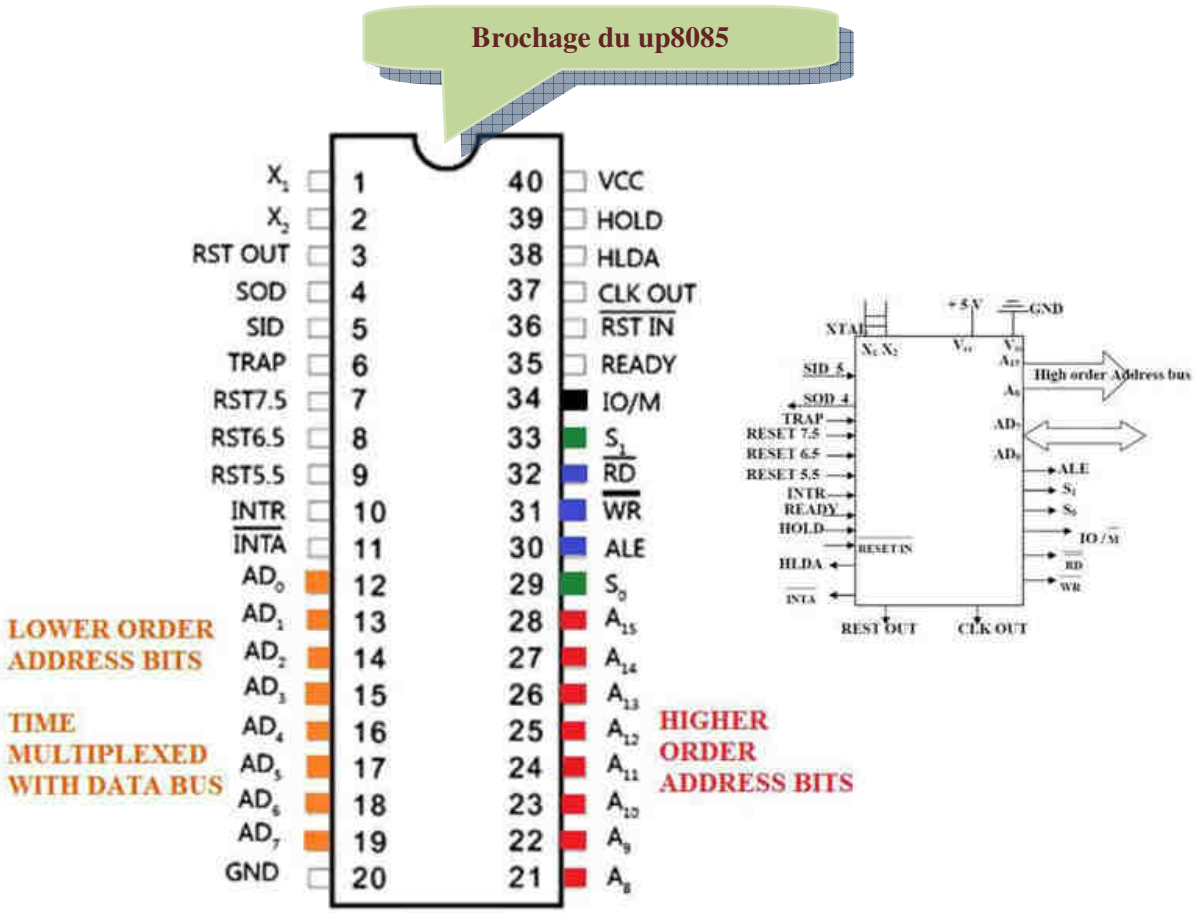


Figure 4.3 : *Brochage du microprocesseur 8085.*

- 16 broches pour le bus d'adresse → Espace mémoire adressable $\mathbf{2^{16} = 64\ Ko}$.
- 08 broches pour le bus de donnée.
- La fonction de chaque broche du up8085 est donnée par le tableau 4.2.

Table 4.2 : *Fonctions des broches du up8085.*

Broche	Type (E/S)	Description
AD0-AD7	Bidirectionnel, 3-états	✓ Bus d'adresse / données.
A8-A15	Sortie, 3-états	✓ Bus d'adresse.
ALE	Sortie à 3 états	✓ Validation de verrou d'adresse.
$\overline{RD}$	Sortie, 3-états	✓ Commande de lecture.
$\overline{WR}$	Sortie, 3-états	✓ Commande d'écriture.
IO/$\overline{M}$	Sortie, 3-états	✓ Indicateur E/S ou mémoire.
S0, S1	Sortie	✓ Indicateurs d'état de bus.
READY	Entrée	✓ Requête de mode attente.
SID	Entrée	✓ Entrée de données sérielles.
SOD	Sortie	✓ Sortie de données sérielles.
HOLD	Entrée	✓ Requête d'attente.

HOLDA	Sortie	✓ Accusé de réception d'attente.
INTR	Entrée	✓ Requête d'interruption.
TRAP	Entrée	✓ Requête d'interruption non masquable.
RST 5.5 RST 6.5 RST 7.5	Entrée	✓ Requêtes d'interruption matérielles vectorisées.
$\overline{\text{INTA}}$	Sortie	✓ Accusé de réception d'interruption.
$\overline{\text{TESET IN}}$	Entrée	✓ Réinitialisation système.
RESET OUT	Sortie	✓ Réinitialisation périphériques.
X1, X2	Entrée	✓ Connexions cristal ou RC.
CLK	Entrée	✓ Connexions cristal ou RC.
VCC, GND	/	✓ Alimentation, Masse.

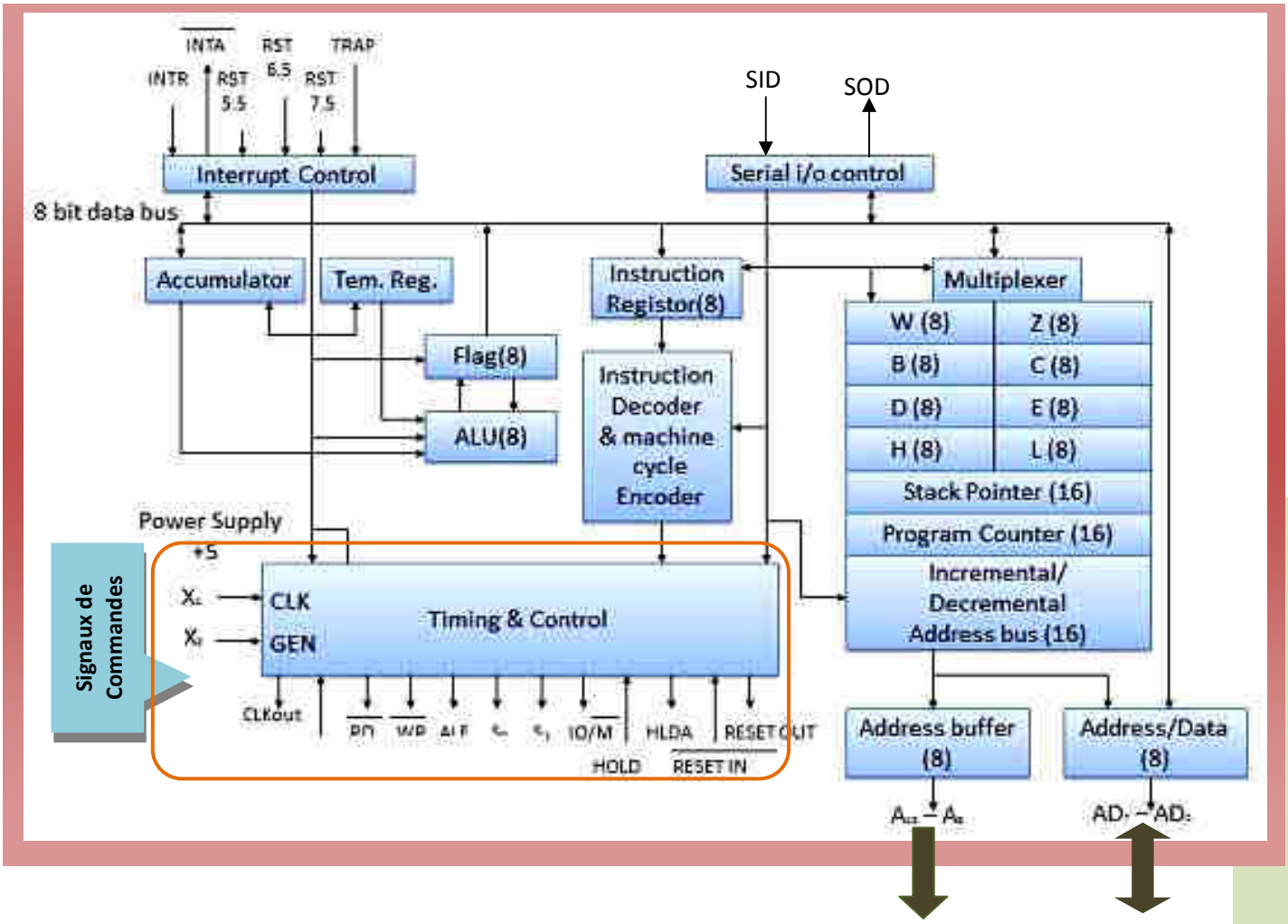
Les sorties **IO/$\overline{\text{M}}$**, **S0** et **S1** sont des signaux de commande qui informent les périphériques du type de cycle machine que le up8085 est en train d'exécuter. Le tableau ci-dessous illustre les combinaisons correspondantes de signaux de sorties des broches **IO/$\overline{\text{M}}$**, **S0** et **S1**.

Table 4.3 : *Combinaisons correspondantes de signaux de sorties des broches* **IO/$\overline{\text{M}}$**, ***S0*** *et* ***S1*** *du up8085.*

Broches			Etat du cycle machine
IO/$\overline{\text{M}}$	**S1**	**S0**	
0	**0**	**0**	✓ Ecriture mémoire.
0	**1**	**0**	✓ Lecture mémoire.
1	**0**	**1**	✓ Ecriture E/S.
1	**1**	**0**	✓ Lecture E/S.
0	**1**	**1**	✓ Extraction code opération (COP).
1	**1**	**1**	✓ Accusé réception d'interruption.
Hi-z	**0**	**0**	✓ Halte.
Hi-z	**x**	**x**	✓ Attente.
Hi-z	**x**	**x**	✓ Réinitialisation.
Hi-z : Etat de haute impédance. **X :** Etat non spécifié.			

4.3.2. Architecture interne du up8085

La figure ci-dessous illustre l'architecture interne du up8085.



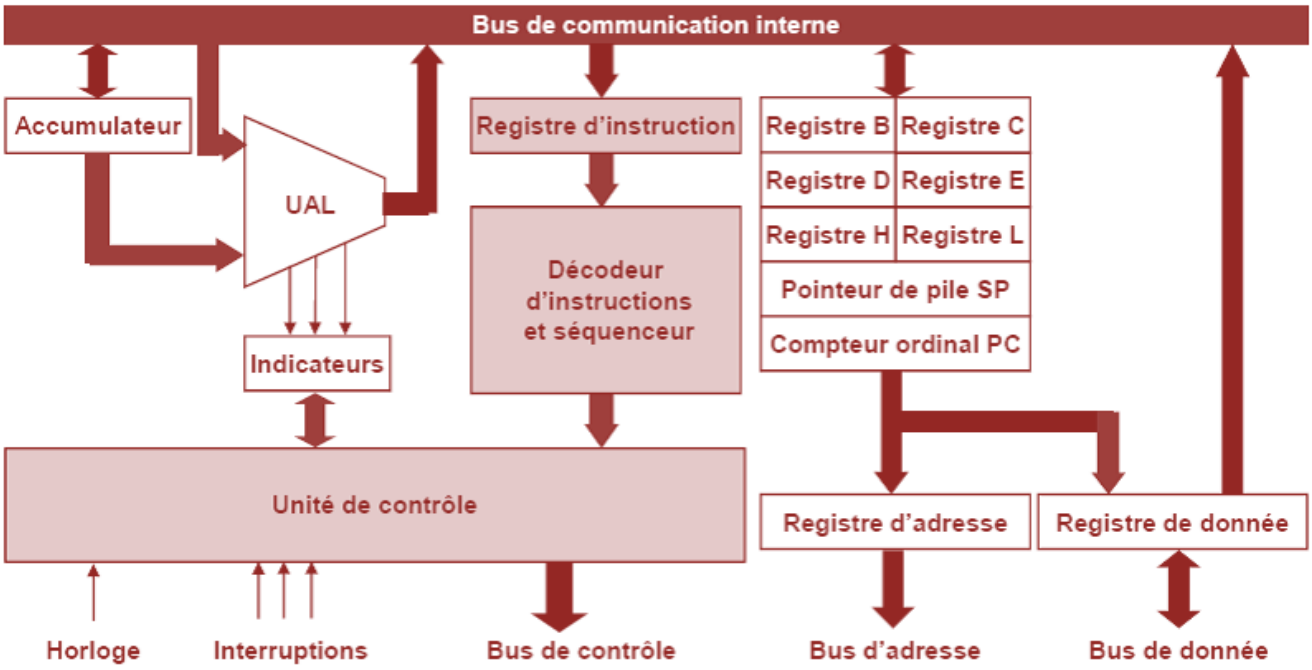
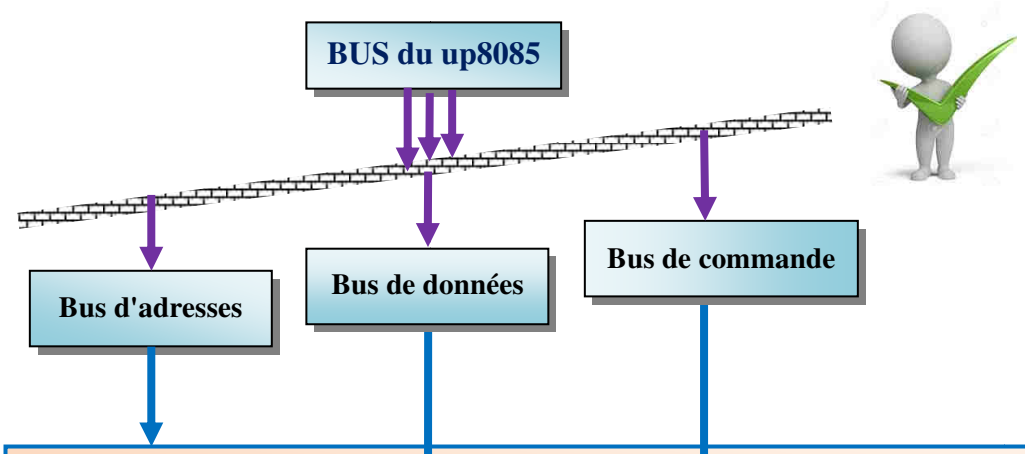


Figure 4.4 : *Architecture interne du microprocesseur 8085.*



- ✓ Se compose de **16 lignes d'adresse**: **A0… A15.**
- ✓ Fonctionne en mode continu (unidirectionnel): Les bits d'adresse sont toujours envoyés du up8085 aux périphériques, et pas l'inverse.
- ✓ 16 lignes d'adresse sont capables d'adresser un espace mémoire total de 216 = 65.536 (64 Ko) emplacements dans la mémoire
- ✓ Adresses mémoire: De 0000_{HEX} à $FFFF_{HEX}$.

- ✓ Se compose de **8 lignes de données**: **D0… D7.**
- ✓ Fonctionne en mode bidirectionnel: Les bits d'informations sont envoyés du up8085 aux périphériques, aussi bien que des périphériques au MPU.
- ✓ Plage des données: $00_{HEX} - FF_{HEX}$.

- ✓ Le bus de commande est utilise aux transferts de signaux de commande.
- ✓ Il se compose de diverses lignes portant les signaux de commande tels que l'activation de droit de lecture/écriture, les bits de drapeau.
- ✓ Les signaux de commande typiques sont:
  - ➢ Lecture de la mémoire.
  - ➢ L'écriture en mémoire.
  - ➢ La lecture E/S.
  - ➢ L'écriture E/S.

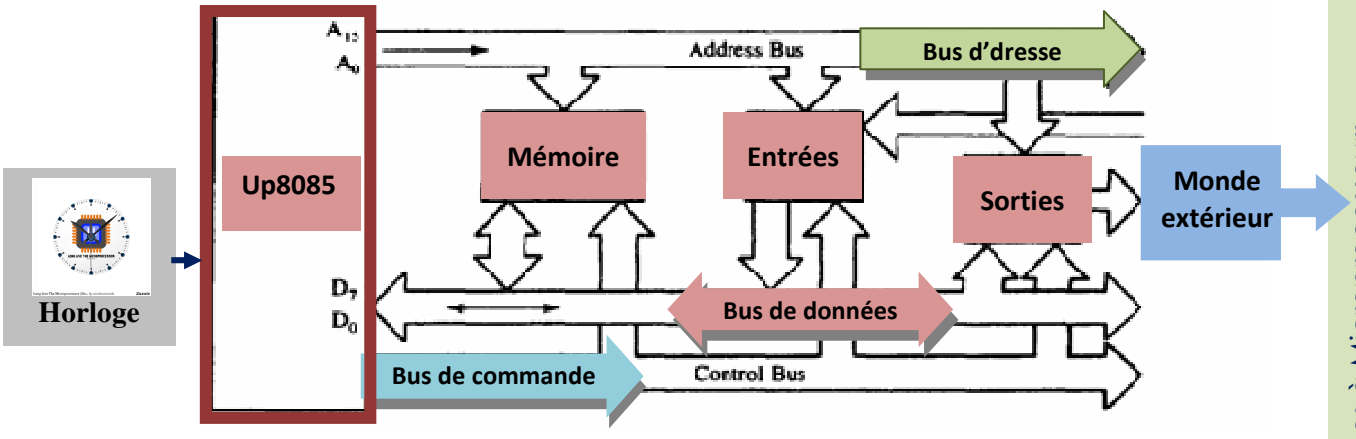


Figure 4.5 : *Bus du microprocesseur 8085.*

L'unité centrale (CPU - Central Processing Unit) regroupe 5 blocs fonctionnels :

- Unité de contrôle.
- Unité arithmétique et logique.
- Registres généraux.
- Registres spécialisés.
- Interfaçage avec le monde extérieur.

Unité de contrôle

- Elle contrôle la totalité du fonctionnement de l'unité centrale :
  - Lecture, décodage, et exécution des instructions.
  - Lecture et écriture des données en mémoire centrale.
  - Lecture et écriture des registres.
  - Contrôle de l'unité arithmétique et logique.
  - Contrôle de l'interface avec l'extérieur :
    - Bus d'adresse et de données.
    - Fonctions d'accès à la mémoire centrale.
    - Interruptions, ...

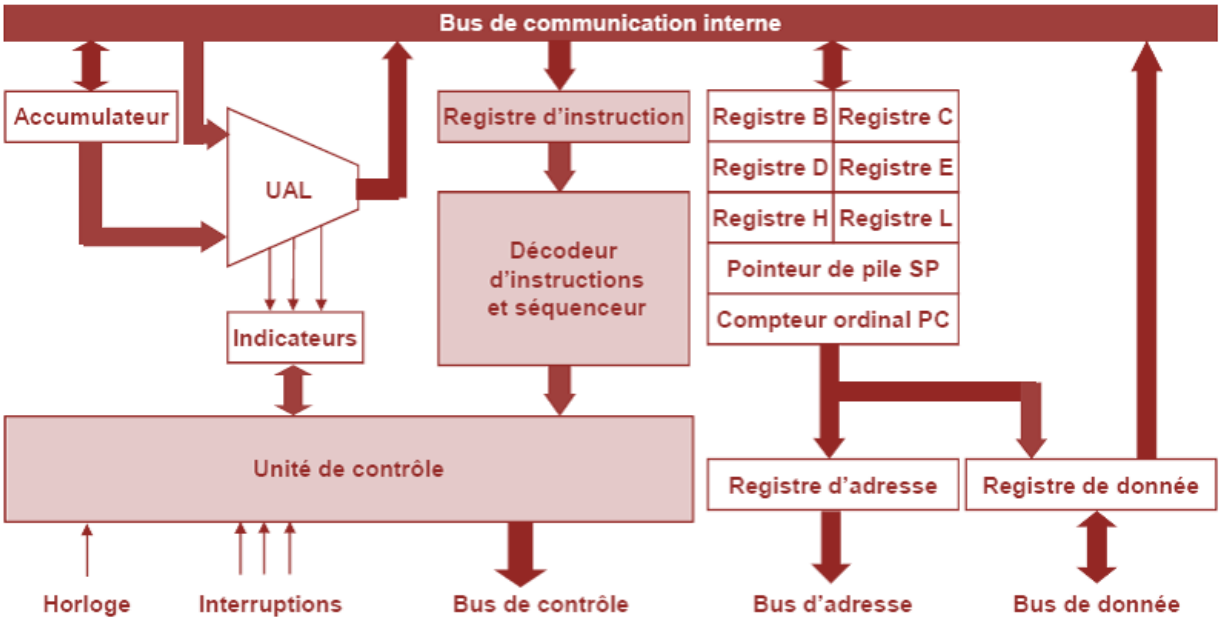


Figure 4.6 : *Unité de contrôle du microprocesseur 8085.*

Unité arithmétique et logique (UAL)

- **Rôle** : Calcul d'opérations élémentaires :
  - Opérations arithmétiques :
    - Addition, soustraction, multiplication, division.
    - Changement de signe.
  - Opérations logiques :
    - ET, OU, OU exclusif, Négation.
    - Décalage, rotation, …
- Traite des mots de taille fixe (1, 2, 4 octets)
- Génère les indicateurs

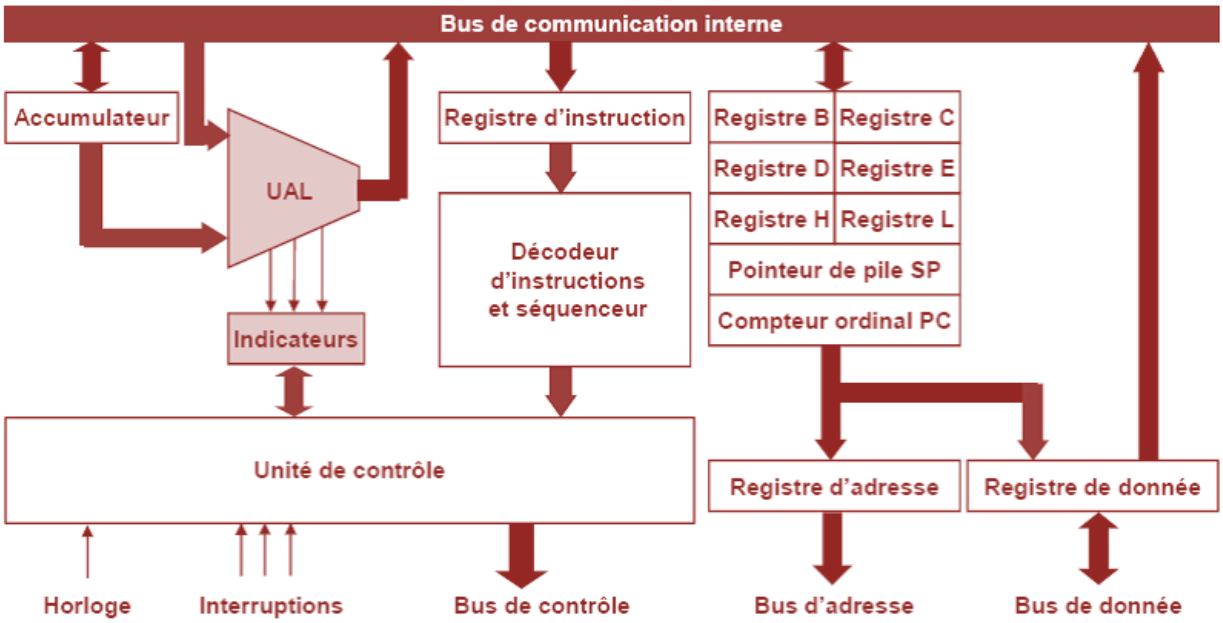


Figure 4.7 : *UAL du microprocesseur 8085.*

Registres :

- **Registres généraux**
  - Opérandes pour l'unité arithmétique et logique.
  - Résultats des calculs (accumulateur).
  - En nombre variable (2 à plusieurs dizaines).
  - Taille = taille des mots traités par l'UAL.

- **Registres spécialisés**
  - Compteur ordinal (Program Counter - PC).
  - Registre d'état (Status Register - SR) → PSW.
  - Pointeur de pile (Stack Pointer - SP).

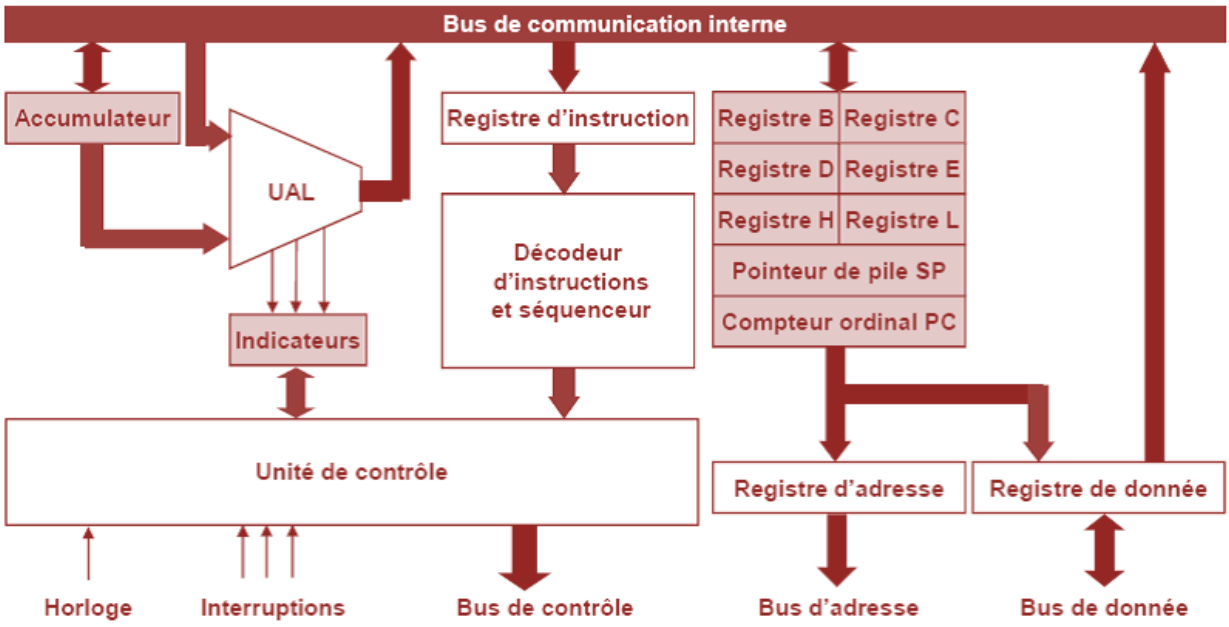
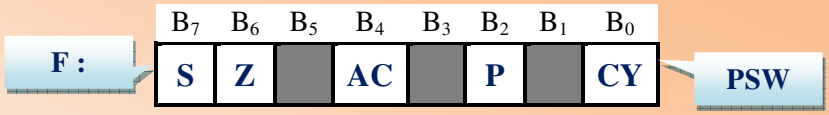


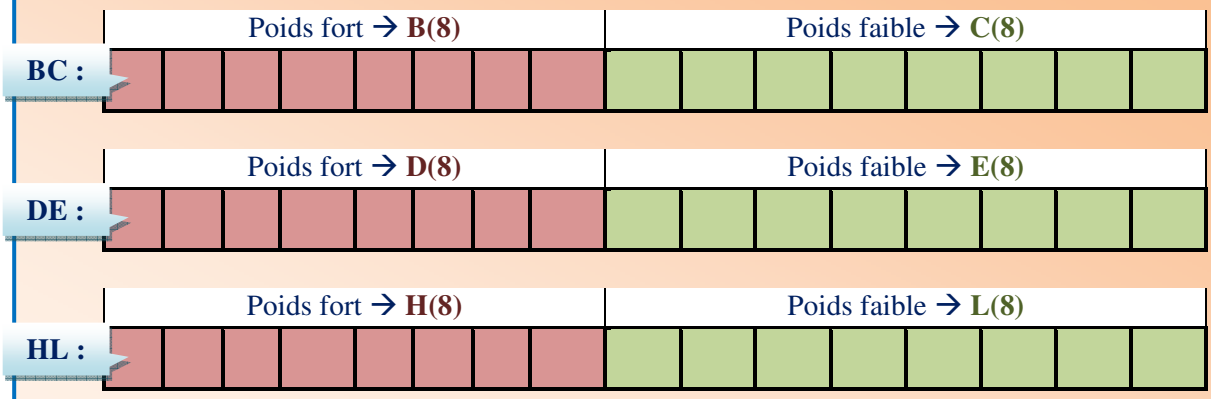
Figure 4.8 : *Registres du microprocesseur 8085.*

- ✓ A (registre 8 bits) est le nœud de toutes les opérations du up 8085
- ✓ Registre drapeau (**F**): comporte 5 drapeaux (sollicite seulement pour les opérations mathématiques) qui sont:



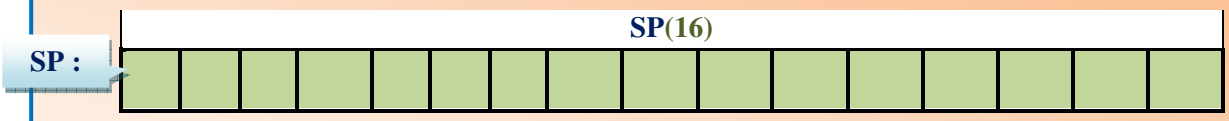
- **S**- Sign bit ou signe bit (le tout premier bit): il est soit 0 ou 1.
- **Z**- zero bit (prend la valeur 1 quand des 8 bits, aucun n'est 0).
- **AC**- Auxillary Carry bit (prend la valeur 1 si s'ajoute 1 au 5e bit du milieu allant de la droite vers la gauche dans une addition mathématique).
- **P**- parity bit ou bit de parité (prend la valeur 1 le nombre total des bits en 1 est paire)**.**
- **CY**- carry bit (prend la valeur 1 si l'addition ne nous amène pas a 9 bits au lieu de 8).

✓ **Les registres généraux :**



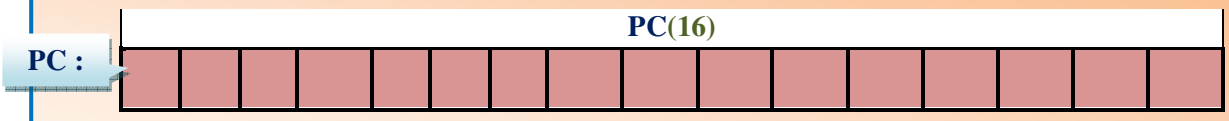
- Quelques instructions utilisent les paires BC et DE comme pointeurs d'adresse.
- La paire HL (dite pointeurs de données par Intel) peut être utilisée pour pointer sur les adresses.

✓ **Pointeur de PILE ou stack pointer (SP):**

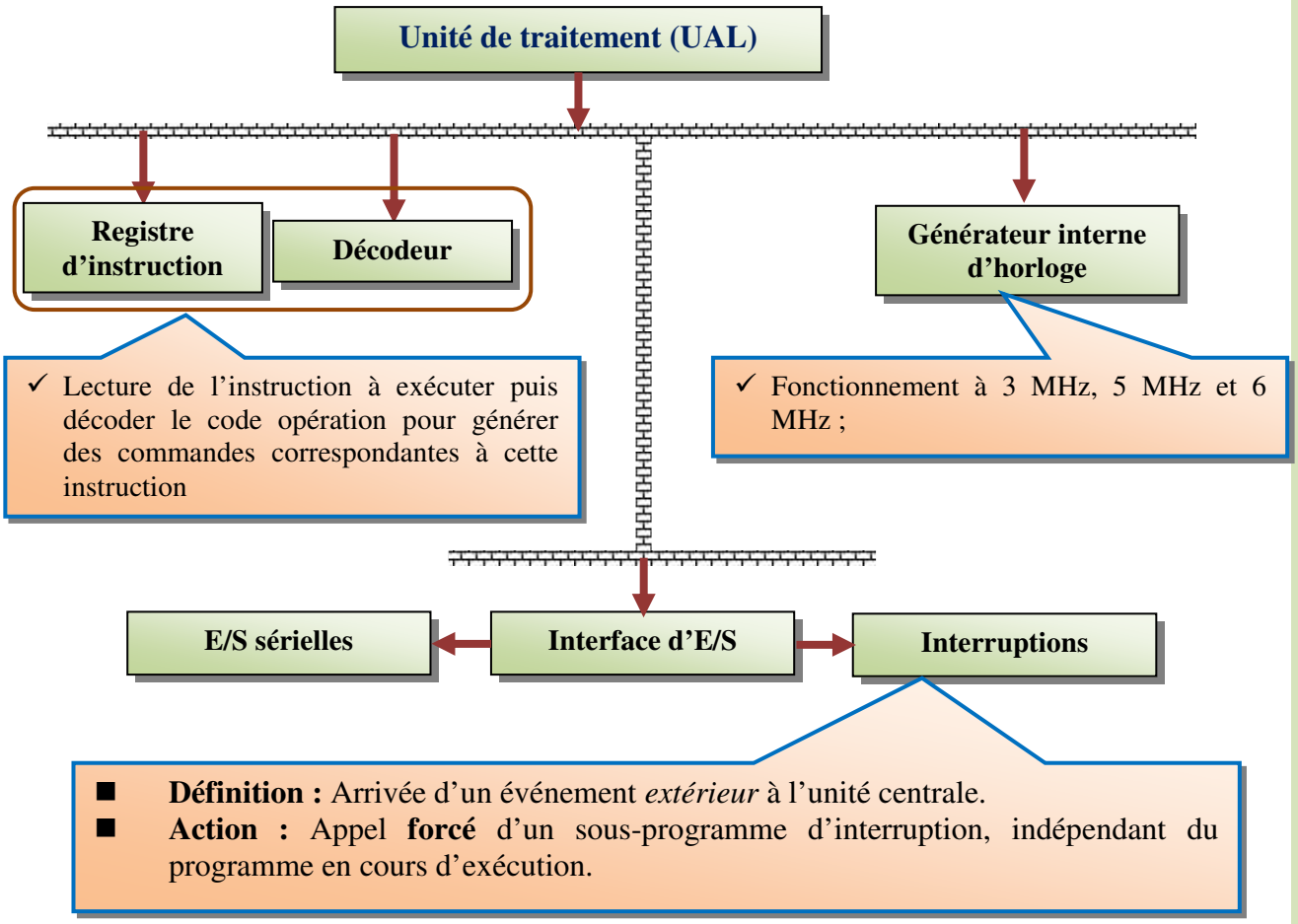


- SP ou pointeur d'adresse (ou de données) qui pointe toujours sur le sommet de la pile dans la RAM.

✓ **Compteur d'instruction (Program Counter) (PC):**



- PC pointe toujours sur la case mémoire de la prochaine l'instruction à exécuter.



La mémoire centrale (RAM) peut être représentée par le schéma ci-dessous. La taille adressable par le up8085 est de 2^{16} = 64 Ko.

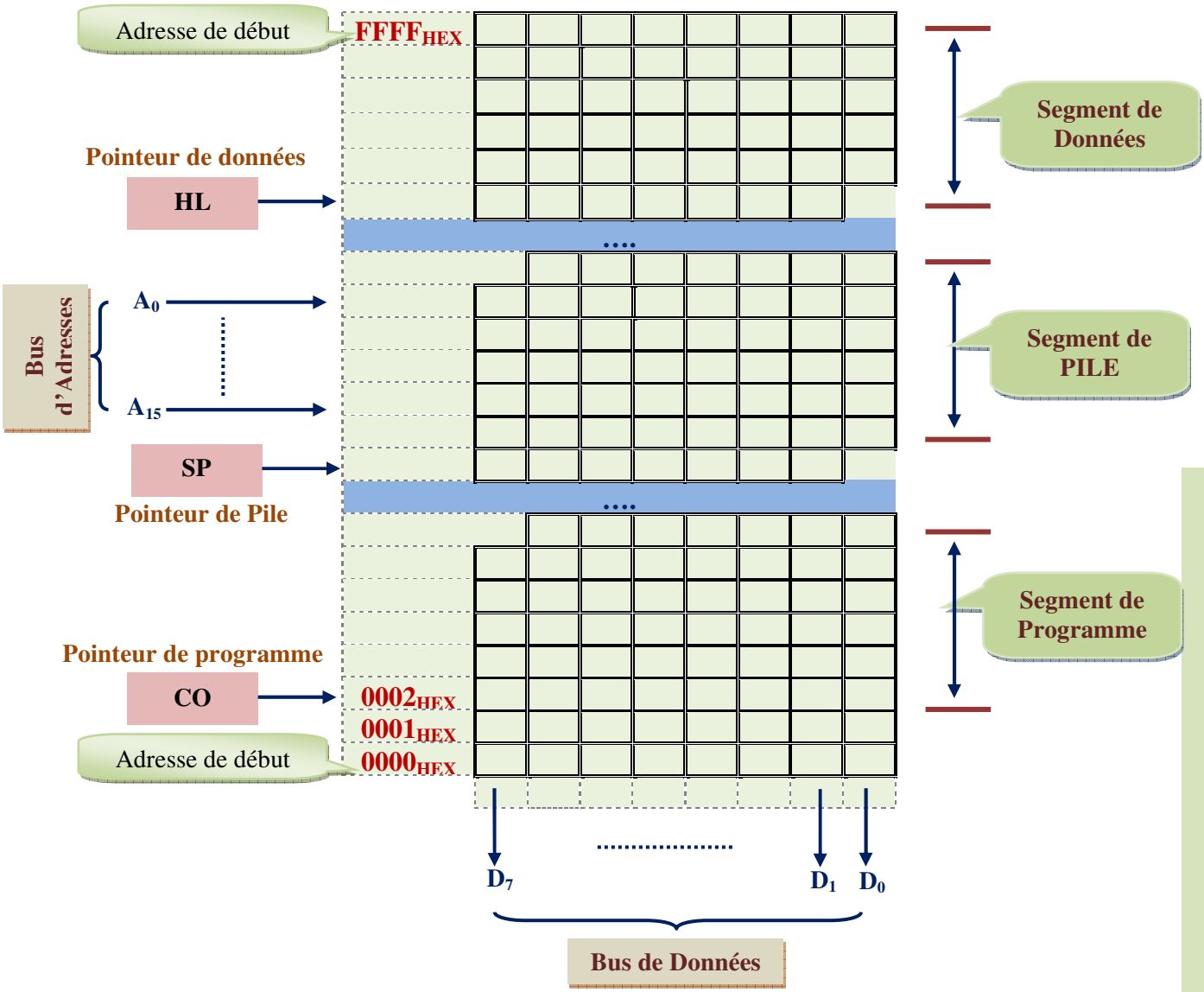


Figure 4.9 : *Segmentation de la mémoire du up8085.*

- **AD0- AD7: 8** bits de poids faible du bus d'adresses, **multiplexés** avec 8 bits de données.
- Le bus AD est multiplexé (multiplexage temporel) d'où la nécessité d'un **démultiplexage** pour obtenir séparément les bus d'adresses et de données:
  - ✓ 8 bits de données (microprocesseur 8 bits).
  - ✓ 16 bits d'adresse d'où 2^{16} = 64 Ko d'espace mémoire adressable par le up8085.
- Le démultiplexage des signaux **AD0- AD7** se fait en mémorisant l'adresse lorsque celle-ci est présente sur le bus A/D, à l'aide d'un **VERROU** (**LATCH**) → Ensemble des bascules **D**.
- La commande de mémorisation de l'adresse est générée par le up8085 est le signal **ALE** (Adresse Latch Enable).
  - ✓ ALE = 1, le circuit latch est transparent (Q = D).
  - ✓ ALE = 0, mémorisation de la dernière valeur D sur les sorties Q.

Les signaux de lecture (RD) ou d'écriture (WR) ne sont générés par le up8085 que lorsque les données sont présentes sur le bus A/D.

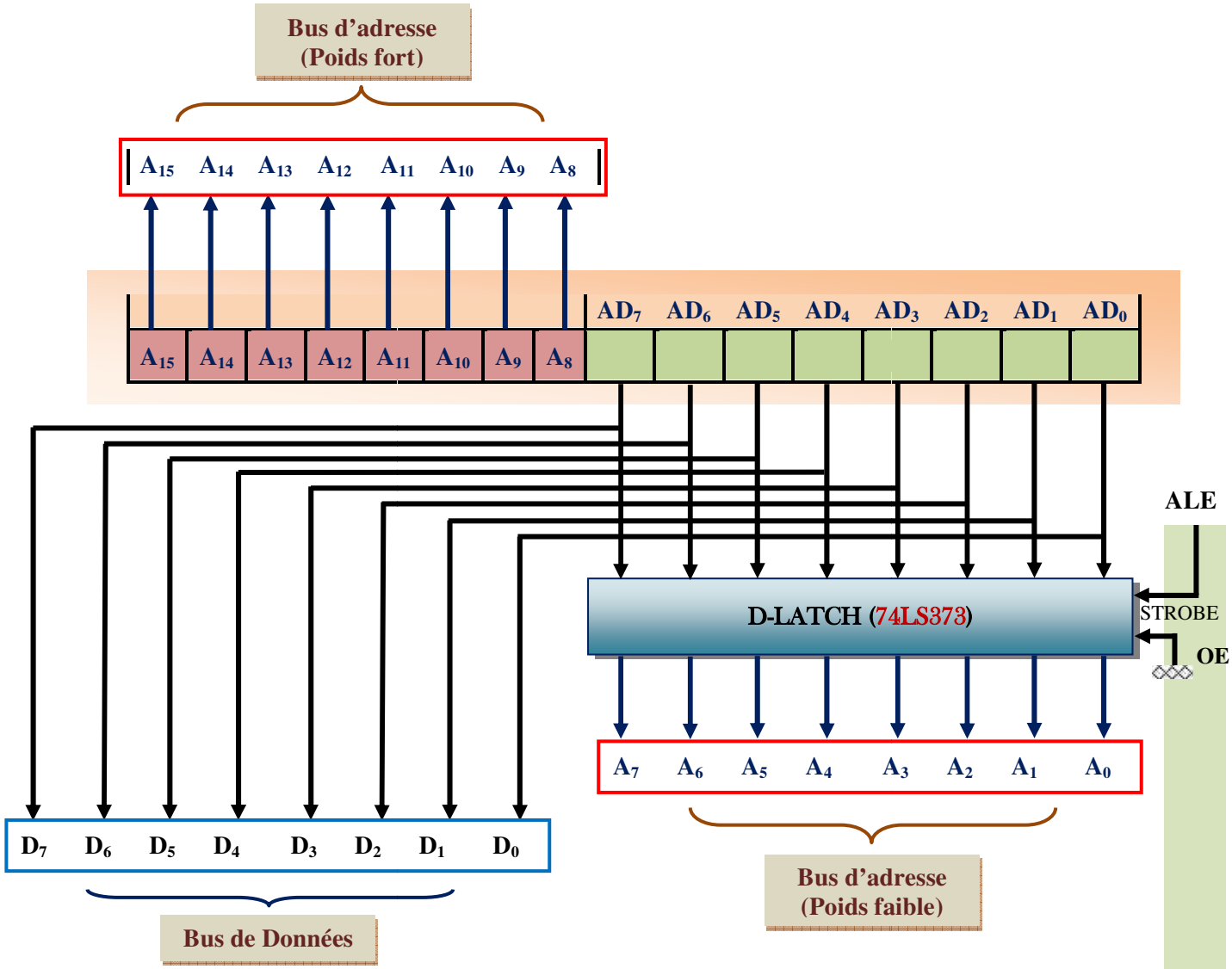


Figure. 4.10 : *Démultiplexage des lignes d'adresses/données du up8085.*

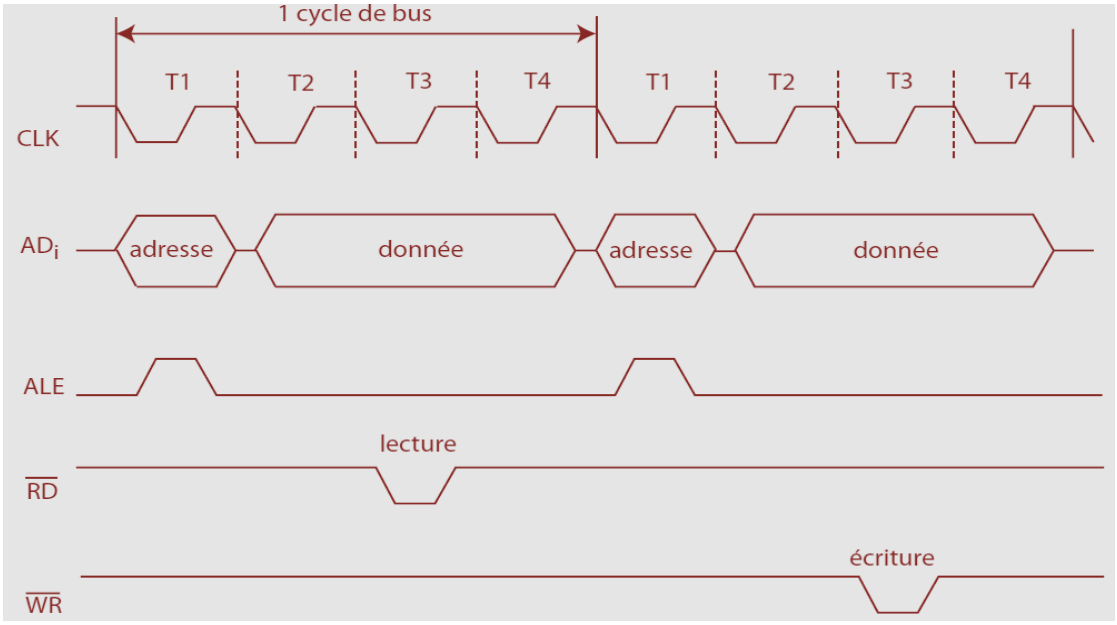


Figure 4.11 : *Chronogramme de bus d'adresse/données.*

Exemple :

- Conception d'un système à base du up8085 avec :
  - 32 Ko de mémoire RAM, en utilisant des mémoires RAM 2 x 16Ko.
  - 32 Ko de mémoire ROM, en utilisant des mémoires ROM 2 x 16Ko.

➔ 04 circuits ayant 14 lignes d'adresses pour chacun.

Le tableau ci-dessous illustre l'espace mémoire réservé pour chaque circuit mémoire. Les signaux de commande des boitiers (CS) sont générés par le décodeur d'adresse 2x4 (ou 3x8), comme suit :

$$\begin{cases} \textbf{ROM 1} & \rightarrow \overline{CS} = \overline{A_{14}} \cdot \overline{A_{15}} \\ \textbf{ROM 2} & \rightarrow \overline{CS} = A_{14} \cdot \overline{A_{15}} \\ \textbf{RAM 1} & \rightarrow \overline{CS} = \overline{A_{14}} \cdot A_{15} \\ \textbf{RAM 2} & \rightarrow \overline{CS} = A_{14} \cdot A_{15} \end{cases}$$

Décodage		Adresses																
A15	**A14**	**A13**	**A12**	**A11**	**A10**	**A9**	**A8**	**A7**	**A6**	**A5**	**A4**	**A3**	**A2**	**A1**	**A0**	**Espace mémoire**	**Adresse**	**Circuit**
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000H	Début	ROM 1
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-		-		
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3FFFH	Fin	
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000H	Début	ROM 2
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-		-		
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	7FFFH	Fin	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8000H	Début	RAM 3
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-		-		
1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	BFFFH	Fin	
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	C000H	Début	RAM 4
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-		-		
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	FFFFH	Fin	

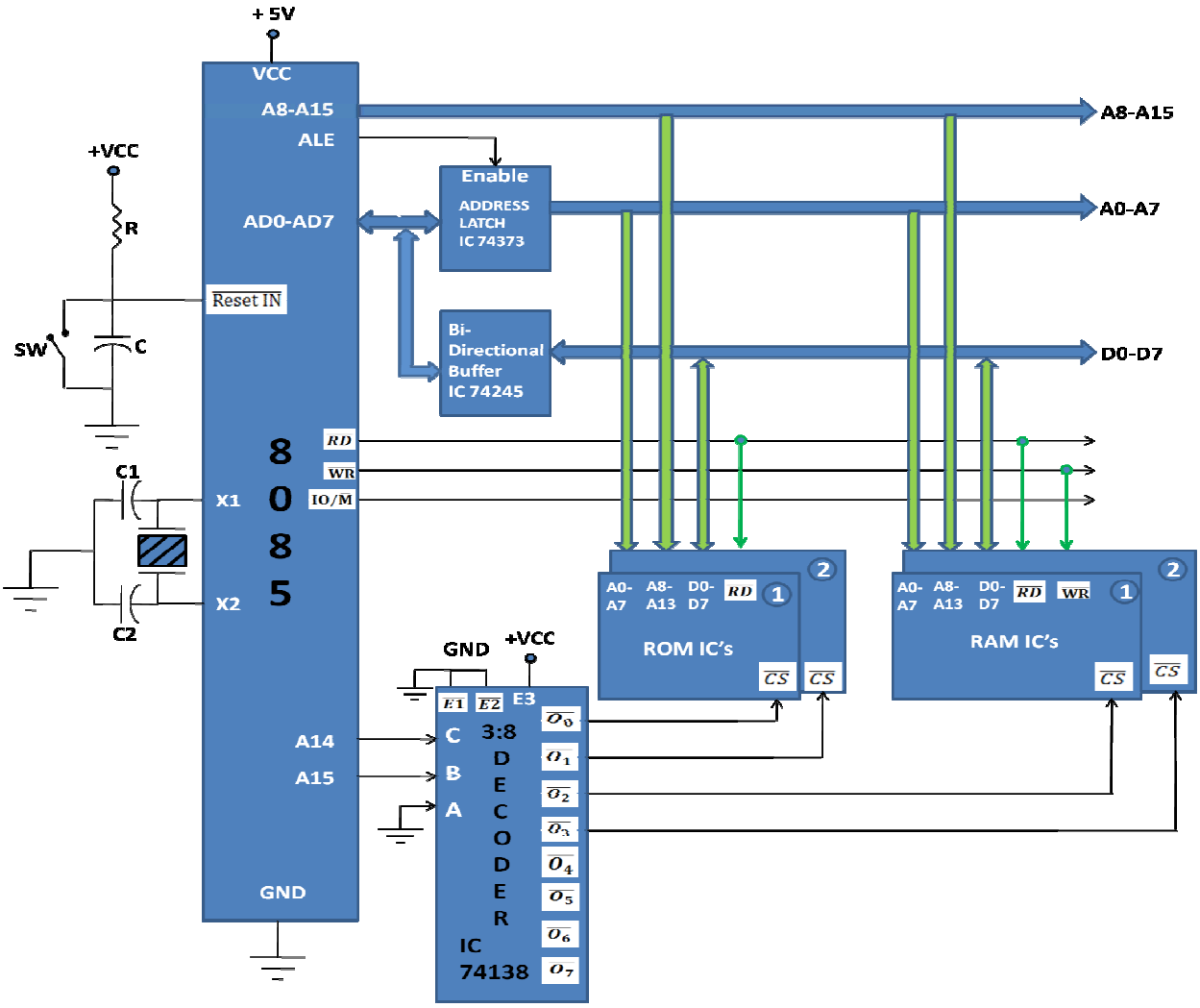


Figure 4.12 : *Interfaçage up8085-mémoires.*

Le up8085 possède cinq broches d'interruptions :

- ✓ L'interruption **INTR** correspond `a l'interruption du 8080 avec laquelle elle est compatible.
- ✓ **RST5.5** est une interruption masquable.
- ✓ **RST6.5** est une interruption masquable.
- ✓ **RST7.5** est une interruption masquable.
- ✓ **Trap** est une interruption non masquable.

⌖ Les instructions masquables peuvent être activées ou désactivées toutes ensembles en utilisant les instructions **EI** et **DI**.

Les interruptions **RST5.5**, **RST6.5** et **RST7.5** peuvent être activées ou désactivées individuellement en utilisant l'instruction **SIM**.

a) Modes d'adressage

- Permettent de **localiser précisément** les opérandes d'une instruction → Cheminement des informations (données).
- Certaines architectures offrent les instructions disponibles avec différents modes d'adressage, d'autres possèdent des modes d'adressage spécifique à chaque instruction.
- Le up8085 utilise 5 modes d'adressage, à savoir :
  1. Adressage implicite.

2. Adressage immédiat.
3. Adressage par registre.
4. Adressage Direct.
5. Adressage indirect.

Par la suite, on utilise la nomenclature de la table ci-dessous.

Table 4.4 : *Nomenclature utilisée.*

Nomenclature	Description
OS	✓ Opérande source.
OD	✓ Opérande destination.
INST	✓ Instruction (COP).
R	✓ Registre.
M	✓ Case mémoire.
adr	✓ Adresse mémoire.
[adr]	✓ Contenu d'une adresse mémoire.
val	✓ Valeur numérique.
OP	✓ Opérande.

OD ← (OD) INST (OS) ;

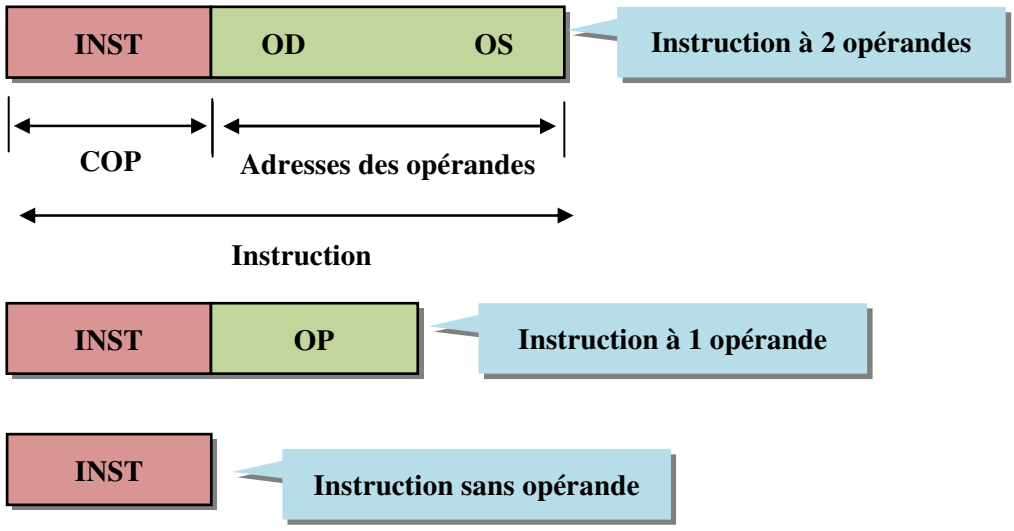


Figure 4.13 : *Différents types des instructions du up8085 .*

Table 4.5 : *Modes d'adressage du up8085 .*

Mode	Instruction	Exemple	Description
Adressage implicite	**INST R ;**	**INC A ;**	✓ La valeur ou la localisation de(s) opérandes est spécifiée directement dans l'instruction. ✓ Il s'agit d'instructions qui travaillent

			sur des registres spécifiques.
Adressage immédiat	**INST R/M, val**	**ADD A, #3C ;**	✓ Les données à traiter sont disponibles dans l'instruction elle-même.
Adressage par registre	**INST R, R ;**	**ADD A, BC ;**	✓ Dans ce type d'adressage, les données d'un registre sont passées a un autre registre
Adressage direct	**INST R, [adr] ;** **INST [adr] ,R ;**	**ADD A, 6EH ;**	✓ L'adresse des données est spécifiée dans l'instruction elle-même.
Adressage indirect	**INST R, [R] ;** **INST [R], R;**	**ADD A, @R0 ;**	✓ L'adresse de la donnée est disponible dans le registre spécifié.
Adressage indexé	**/**	**JMP @A +DPTR**	✓ Ce mode est similaire au mode indirect sauf que cette fois, l'adresse finale est obtenue par l'addition d'un index à une adresse de base.

4.4. Jeu d'instruction du up8085

Intel groupe les instructions du 8080/8085 sous les rubriques mentionnées ci-dessous.

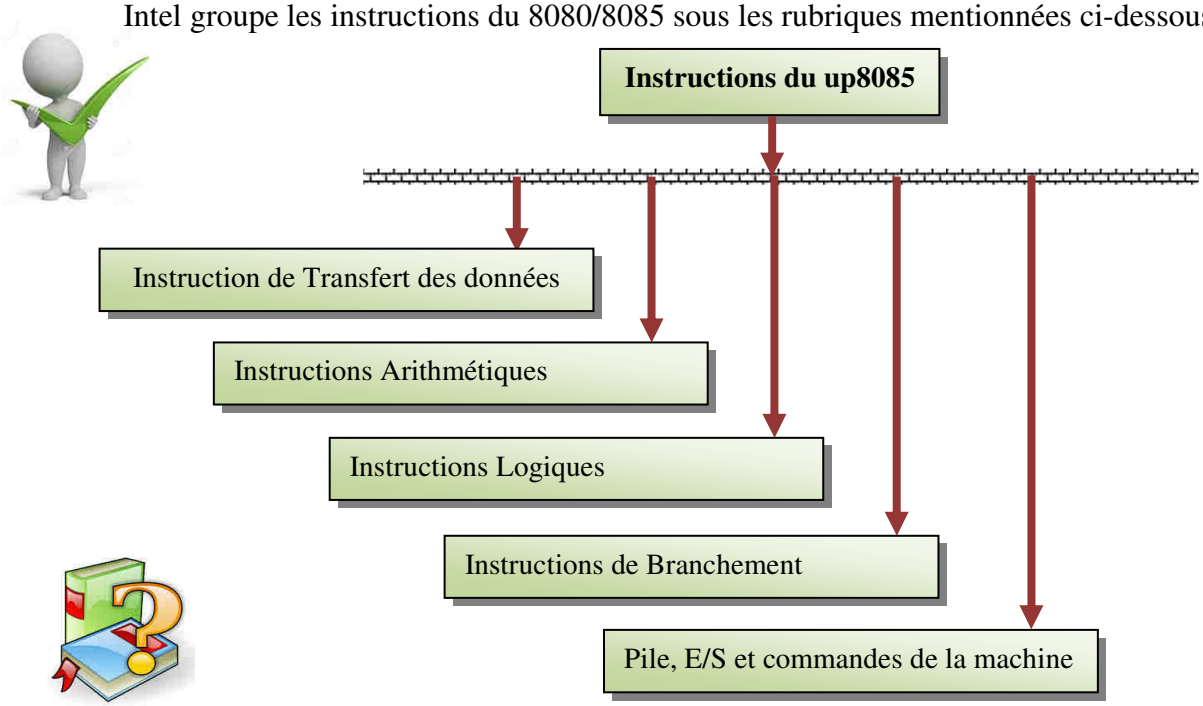


Figure 4.14 : *Groupes d'instructions du up8085.*

Table 4.6 : *Jeu d'instruction du up8085.*

	COP	Code mnémonique	Description
Instructions de Transfert des données	**MOV**	MOV R1, R2 ;	✓ Déplacer le contenu d'un registre vers un autre. **[R1] ← [R2].**
		MOV R, M ;	✓ Déplacer le contenu d'une case mémoire vers un registre. **[R] ← [M].**
		MOV M, R ;	✓ Déplacer le contenu d'un registre vers une case mémoire. **[M] ← [R].**
	MVI	MVI R, data ;	✓ Déplacer une donnée **immédiate** vers un registre. **[R] ← data**.

		MVI M, data ;	✓ Déplacer une donnée **immédiate** vers une case mémoire. **M ← data.**
	LXI	LXI R, data 16 ;	✓ Load register pair immediate. **[R] ← data 16 bits, [R] ← 8 LSBs of data.**
	LDA	LDA addr ;	✓ Load Accumulator direct. **[A] ← [addr].**
	STA	STA addr ;	✓ Store accumulator direct. **[addr] ← [A].**
	LHLD	LHLD addr ;	✓ Load H-L pair direct. **[L] ← [addr], [H] ← [addr+1].**
	SHLD	SHLD addr ;	✓ Store H-L pair direct. **[addr] ← [L], [addr+1] ← [H].**
	LDAX	LDAX R ;	✓ LOAD accumulator indirect. **[A] ← [[R]].**
	STAX	STAX R ;	✓ Store accumulator indirect. **[[R]] ← [A].**
	XCHG	XCHG ;	✓ Exchange the contents of H-L with D-E pair. **[H-L] ←→ [D-E].**
Instructions arithmétiques	**ADD**	ADD R ;	✓ Add register to accumulator. **[A] ← [A] + [R].**
		ADD M ;	✓ Add memory to accumulator. **[A] ← [A] + [M].**
	ADC	ADC R ;	✓ Add register with carry to accumulator. **[A] ← [A] + [R] + [CS].**
		ADC M ;	✓ Add memory with carry to accumulator. **[A] ← [A] + [M] + [CS].**
	ADI	ADI data ;	✓ Add immediate data to accumulator. **[A] ← [A] + data.**
	ACI	ACI data ;	✓ Add with carry immediate data to accumulator. **[A] ← [A] + data + [CS].**
	DAD	DAD R ;	✓ Add register pair to H-L pair. **[H-L] ← [H-L] + [R].**
	SUB	SUB R ;	✓ Subtract register from accumulator. **[A] ← [A] – [R].**
		SUB M ;	✓ Subtract memory from accumulator. **[A] ← [A] – [M].**
	SBB	SBB R ;	✓ Subtract register from accumulator with borrow. **[A] ← [A] – [R] – [CS].**
		SBB M ;	✓ Subtract memory from accumulator with borrow. **[A] ← [A] – [M] – [CS].**
	SUI	SUI data	✓ Subtract immediate data from accumulator. **[A] ← [A] – data.**
	SBI	SBI data ;	✓ Subtract immediate data from accumulator with borrow. **[A] ← [A] – data – [CS].**
	INR	INR R ;	✓ Increment register content. **[R] ← [R] + 1.**
		INR M ;	✓ Increment memory content. **[M] ← [M] + 1.**
	DCR	DCR R ;	✓ Decrement register content. **[R] ← [R] - 1.**
		DCR M ;	✓ Decrement memory content.

			[M] ← [M] - 1.
	INX	INX R ;	✓ Increment register pair. **[R] ← [R] + 1.**
	DCX	DCX R ;	✓ Decrement register pair. **[R] ← [R] - 1.**
	DAA	DAA ;	✓ Decimal adjust accumulator.
Instructions logiques	**ANA**	ANA R;	✓ AND register with accumulator. **[A] ← [A] AND [R].**
		ANA M ;	✓ AND memory to accumulator. **[A] ← [A] AND [M].**
	ANI	ANI data ;	✓ AND immediate data with accumulator. **[A] ← [A] AND data.**
	ORA	ORA R ;	✓ OR register with accumulator. **[A] ← [A] AND [R].**
		ORA M ;	✓ OR memory with accumulator. **[A] ← [A] OR [M].**
	ORI	ORI data ;	✓ OR immediate data with accumulator. **[A] ← [A] OR data.**
	XRA	XRA R ;	✓ EXCLUSIVE –OR register with accumulator **[A] ← [A] XOR [R].**
		XRA M ;	✓ EXCLUSIVE –OR memory with accumulator **[A] ← [A] XOR [M].**
	XRI	XRI data ;	✓ EXCLUSIVE-OR immediate data with accumulator **[A] ← [A] XOR data.**
	CMA	CMA ;	✓ Complement the accumulator **[A] ← [A].**
	CMC	CMC ;	✓ Complement the carry status. **[CS] ← [CS].**
	CMP	CMP R ;	✓ Compare register with accumulator. **[A] - [R].**
		CMP M ;	✓ Compare memory with accumulator. **[A] - [M].**
	CPI	CPI data ;	✓ Compare immediate data with accumulator. **[A] –data.**
	RLC	RLC ;	✓ Rotate accumulator left. **[An+1] ← [An], [A0] ← [A7], [CS] ← [A7].**
	RRC	RRC ;	✓ Rotate accumulator right. **[A7] ← [A0], [CS] ← [A0], [An] ← [An+1].**
	RAL	RAL ;	✓ Rotate accumulator left through carry. **[An+1] ← [An], [CS] ← [A7], [A0] ← [CS].**
	RAR	RAR ;	✓ Rotate accumulator right through carry. **[An] ← [An+1], [CS] ← [A0], [A7] ← [CS]**
Instructions de branchement	**JMP**	JMP addr(label)	✓ Branchement (Saut) **non conditionné** vers une instruction spécifiée par une adresse : addr(label).
	J*	Branchement conditionné	
		JZ addr(label) ;	✓ Jump if the result is zero.
		JNZ addr(label) ;	✓ Jump if the result is not zero.
		JC addr(label) ;	✓ Jump if there is a carry.
		JNC addr(label)	✓ Jump if there is no carry.
		JP addr(label) ;	✓ Jump if the result is plus.

		JM addr(label) ;	✓ Jump if the result is minus.
		JPE addr(label) ;	✓ Jump if even parity.
		JPO addr(label) ;	✓ Jump if odd parity.
	CALL	CALL addr(label) ;	✓ **Unconditional CALL**: call the subroutine identified by the operand.
	RET	RET;	✓ Return from subroutine.
	RST	RST n (Restart);	✓ Restart is a one-word CALL instruction. ✓ The content of the program counter is saved in the stack. ✓ The program jumps to the instruction starting at restart location.
Pile, E/S et contrôle de la machine	**IN**	IN port-address ;	✓ Input to accumulator from I/O port. **[A] ← [Port].**
	OUT	OUT port-address ;	✓ Output from accumulator to I/O port. **[Port] ← [A].**
	PUSH	PUSH R;	✓ Push the content of register pair to stack.
		PUSH PSW;	✓ PUSH PSW (PUSH Processor StatusWord .
	POP	POP R ;	✓ Pop the content of registerpair, whichwassaved, fromthe stack.
		POP PSW ;	✓ Pop Processor StatusWord.
	HLT	HLT ;	✓ Halt.
	XTHL	XTHL	✓ Exchange stack-top with H-L.
	EI	EI ;	✓ Enable Interrupts.
	DI	DI ;	✓ Disable Interrupts.
	SPHL	SPHL ;	✓ Move the contents of H-L pair to stack pointer.
	SIM	SIM ;	✓ Set Interrupt Masks.
	RIM	RIM ;	✓ Read Interrupt Masks.
	NOP	NOP ;	✓ No Operation.

a) Piles (Stacks)

- ✓ La pile est un secteur de mémoire identifié par le programmeur pour la mémoire temporaire d'information.
- ✓ La pile est une structure de LIFO.
- ✓ La pile se développe normalement dans l'arrière-plan de la mémoire.
- ✓ En d'autres termes, le programmeur définit le fond de la pile dans l'idée de réduire la plage d'adresses.

Dans le up8085, la pile est définie en plaçant le registre de SP (Stack Pointer ou Indicateur de Pile).

LXI SP, FFFFH ;

Ceci place l'indicateur de pile a la cellule $FFFF_{HEX}$ (fonds de mémoire pour les 8085).

- ✓ Deux opérations possibles sur les files d'attente :
  3. **PUSH** (**EMPILER**): Mettre une information au sommet de la pile, en poussant vers le bas les informations déjà présentes dans la pile.
  4. **POP** (**DEPILER**): Prend l'information qui se trouve au sommet de la pile, en poussant tout le contenu de la pile vers le sommet.

PUSH R ; /* **R = (RH-RL)** : Registre sur 16 bits*/

- **((SP)-1) ← (RH)** → Le contenu du registre supérieur de la pair R est transféré à la mémoire dont l'adresse est inferieur de 1 au contenu du registre SP.

- **((SP)-2) ← (RL)** → Le contenu du registre inferieur de la paire R est transféré à la mémoire dont l'adresse est inférieure de 2 au contenu du registre SP.
- **(SP) ← (SP)-2** → Le contenu du registre SP est décrémenté de 2.

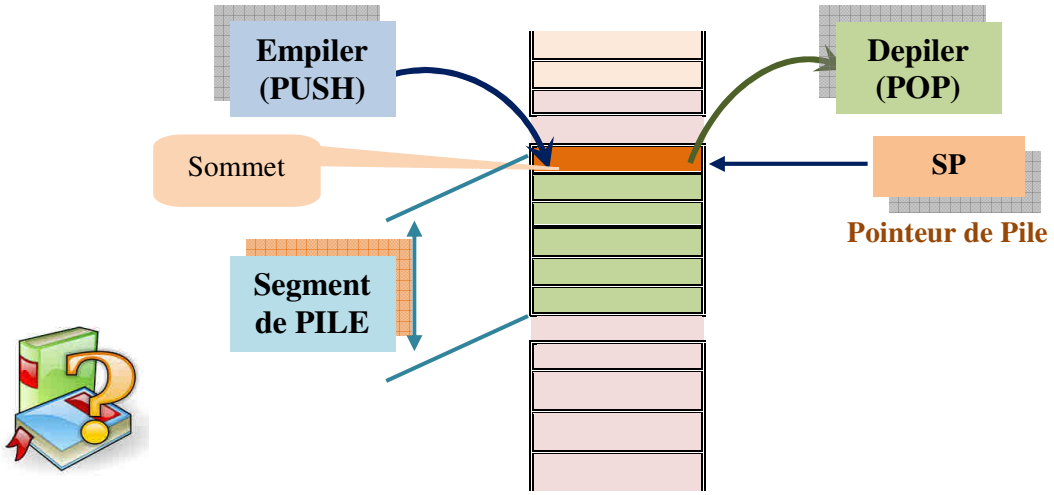


Figure 2.15 : *Représentation graphique de la Pile.*

PUSH PSW ; /* Push Processsor Status Word : Empiler le registre d'état du up8085.

- **((SP)-1) ← (A)** → Le contenu de l'accumulateur est transféré à la mémoire dont l'adresse est inferieur de 1 au contenu du registre SP.
- **$((SP)-2)_0 \leftarrow (CY)$**
- **$((SP)-2)_1 \leftarrow X$**
- **$((SP)-2)_2 \leftarrow (P)$**
- **$((SP)-2)_3 \leftarrow X$**
- **$((SP)-2)_4 \leftarrow (AC)$**
- **$((SP)-2)_5 \leftarrow X$**
- **$((SP)-2)_6 \leftarrow (Z)$**
- **$((SP)-2)_7 \leftarrow (S)$**

Le contenu des indicateurs est assemblé en un PSW, et ce mot est transféré à la mémoire dont l'adresse est inférieure de 2 au contenu du registre SP.

- **(SP) ← (SP)-2** → Le contenu du registre SP est décrémenté de 2.

POP R ; /* **R = (RH-RL)** : Registre sur 16 bits*/

- **(RL) ← ((SP))** → Le contenu de la mémoire dont l'adresse est spécifié par le registre SP est transféré au registre inférieur de la pair R.
- **RH) ← (((SP)+1)** → Le contenu de la mémoire dont l'adresse est supérieur de 1 au contenu du registre supérieur de la pair R.
- **(SP) ← (SP)+2** → Le contenu du registre SP est incrémenté de 2.

POP PSW ; /* Push Processsor Status Word : Dépiler le registre d'état du up8085.

- **((SP)-1) ← (A)** → Le contenu de l'accumulateur est transféré à la mémoire dont l'adresse est inferieur de 1 au contenu du registre SP.
- **$(CY) \leftarrow ((SP))_0$**
- **$(P) \leftarrow ((SP))_2$**
- **$(AC) \leftarrow ((SP))_4$**
- **$(Z) \leftarrow ((SP))_6$**
- **$(S) \leftarrow ((SP))_7$**
- **(A) ← ((SP)+1)**

Le contenu de la mémoire dont l'adresse est spécifié par le contenu du registre SP est utilisé à la récupération des indicateurs de condition.

Le contenu de la mémoire dont l'adresse est supérieure de 1 au contenu de SP est transféré au registre A

- **(SP) ← (SP)+2** → Le contenu du registre SP est incrémenté de 2.

3.5. Programmation du microprocesseur 8080/8085

Les programmes sources seront écrits en langage assembleur 8085. Le format utilisé par Intel divise chaque ligne en langage assembleur selon les champs suivant

Etiquette	Code Opération (COP)	Opérande (s)	Commentaires

Exemple 1 :

	LXI	H, 2020H ;	Initialiser le registre HL à 2020H
	MVI	B, 01H ;	Initialiser le registre B à 01H
	MOV	A, H ;	Transférer le contenu de registre H vers l'accumulateur
	CMA		Complémenter l'accumulateur
	ADD	B ;	A ← (A) + (B)

Exemple :

	LDA	2000H ;	// Load multiplicant to accumulator
	MOV	B, A ;	// Move multiplicant from A(acc) to B register
	LDA	2001H ;	// Load multiplier to accumulator
	MOV	C, A ;	// Move multiplier from A to C
ETIQ :	**ADD**	B ;	// Add B(multiplier) with A
	DCR	C ;	// Decrement C, it act as a counter
	JNZ	**ETIQ** ;	// Jump to L if C reaches 0
	STA	2010H ;	// Store result in to memory
	HLT		// End

3.5.1. Les soubroutines

Une soubroutine est un groupe d'instructions qui seront employées a plusieurs reprises dans différents emplacements du programme.

- ✓ Plutôt que répéter les mêmes instructions plusieurs fois, elles peuvent être groupes dans une soubroutine qu'on appellerait a partir de différents emplacements.
- ✓ Dans le langage d'assemblage, une soubroutine peut exister n' importe où dans le code.
- ✓ Cependant, il est usuel de placer des soubroutines séparément du programme principal.
- ✓ Le modèle 8085 a deux instructions pour traiter avec les soubroutines.
- ✓ L'instruction **CALL** est utilisée comme moyen de réorienter l'exécution du programme a la soubroutine.
- ✓ L'instruction **RET** est employée pour renvoyer l'exécution au programme d'appel.

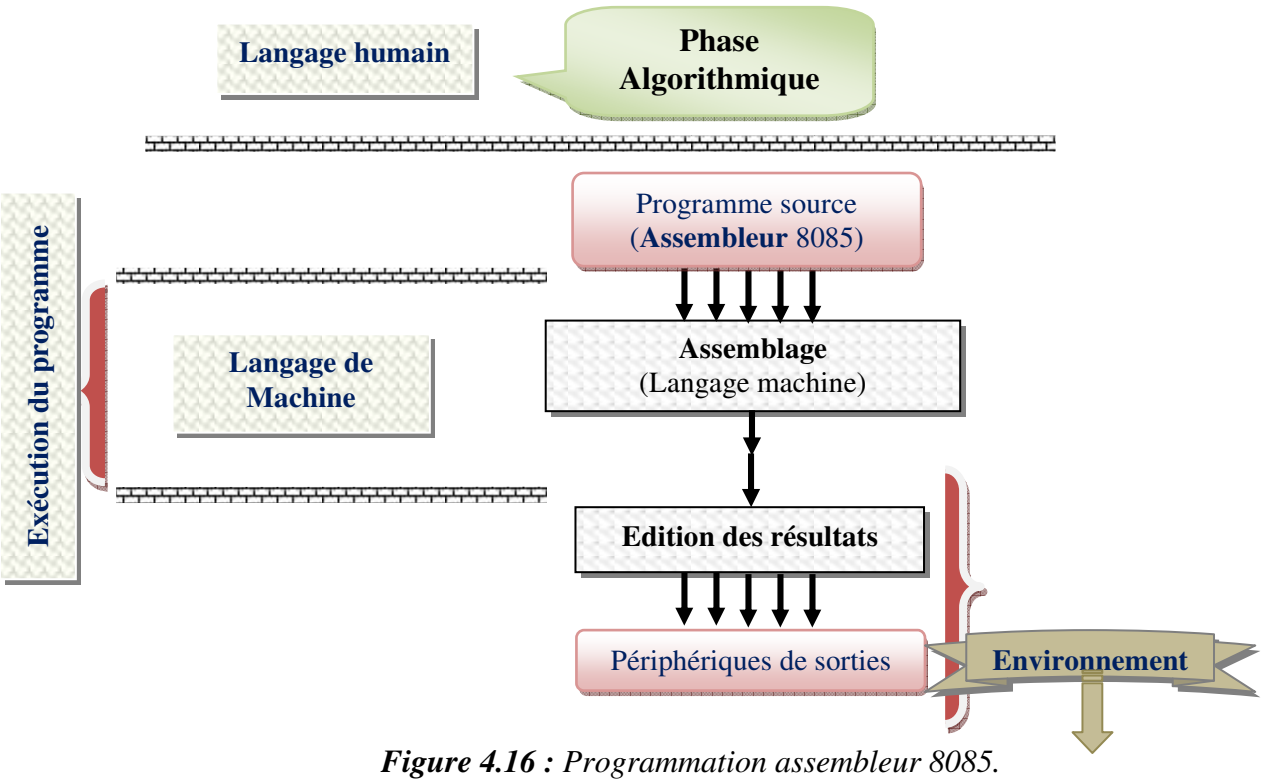


Figure 4.16 : *Programmation assembleur 8085.*

4.6. Exercices

EXERCICE 01 :

Remplir la table suivante par des réponses convenables.

Question	Réponse
Le principal circuit de travail dans un microprocesseur	----------
La majorité du traitement et des transferts de données dans les microprocesseurs 8080/8085 s'effectue dans :	----------
Le circuit qui traite les données se nomme :	----------
Le terme identifiant l'ensemble des registres de l'UCT est :	----------
Les registres de traitement dans l'UCT de microprocesseurs 8080/8085 sont désignés par :	----------
Le registre qui stocke l'octet instruction est appelé le :	----------
Le registre qui renferme l'adresse de la prochaine instruction à exécuter se nomme :	----------
Après l'exécution de chaque instruction, quel compteur doit être augmenté :	----------
La sortie du compteur d'adresse d'instruction est envoyé au :	----------
Qu'est-ce qui retient l'adresse du mot instruction ou de la donnée qui doit être lue dans la mémoire vive ou morte ?	----------
Comment s'appelle le premier octet d'une	----------

instruction, qui indique à l'UCT la tâche à effectuer ?	
Dans une instruction de deux octets, le second octet renferme une :	----------
Dans une instruction à trois octets, les deuxième et troisième octets contiennent l' _____ d'une donnée	----------
Après l'exécution d'une instruction à un seul octet, le compteur d'adresse d'instruction est augmenté de combien de fois ?	----------
Vrai ou Faux : Les instructions dans un programme sont stockés dans des emplacements consécutifs de la mémoire.	----------
L'adresse d'une instruction est fournie au TA par :	----------
L'adresse d'une donnée est fournie par l'instruction ou :	----------
La plupart des instructions, quand elles dont exécutées, provoquent un :	----------
Les lignes parallèles par lesquelles passent les mots binaires sont appelées :	----------
Quel genre de transfert est effectué lorsque tous les bits d'un mot binaire sont transférés simultanément d'un endroit à un autre.	----------
Tous les transferts de données à l'intérieur de l'UCT sont exécutés par quel genre de bus de données ?	----------
Les microprocesseurs ont habituellement deux bus externes, ce sont le bus de _____ et le bus d'_____.	----------
Si une donnée peut circuler dans les deux directions on dit que le bus est :	----------
Vrai ou Faux : La majorité des microprocesseurs à 8 bits ont un bus d'adresses de 16 bits lequel permet d'adresser jusqu'à 64k emplacements en mémoire	----------
L'UCT vient de lire une instruction LDA. Le code d'opération est alors stocké dans le registre d'instruction tandis que les octes d'adresses sont dirigés vers :	----------
L'instruction LDA est exécutée. La donnée désignée par le tampon d'adresse est alors chargée dans :	----------

EXERCICE 02 :

Soit l'instruction suivante :

ADD B ;

Avec (A) = C00FH, (B) = 0040H.

1. Quel est le mode d'adressage utilisé ?

2. Donner le contenu des registres et des cases mémoires concernées, avant et après l'exécution de l'instruction.
3. Trouver la valeur des drapeaux du PSW.

EXERCICE 03 :

Ecrire un programme qui permute le contenu de 2 cases mémoires 8000 et 8001.

EXERCICE 04 :

Nous voulons calculer la somme des valeurs des cases mémoires 8000 et 800F et mettre le résultat dans la case 8020.

EXERCICE 05 :

Ecrire programme qui calcule la somme des valeurs positives et le nombre de valeurs négatives d'un tableau constitué des cases mémoires 8020 à 8040. On enregistre les résultats dans les cases mémoires 8050 et 8051 respectivement.

EXERCICE 06 :

Faire une opération d'addition de 2 nombres de 2 octets chacun qui sont dans cases 8000, 8001 et 8002, 8003 ; on met le résultat dans les cases mémoires 8010 et 8011.

EXERCICE 07 :

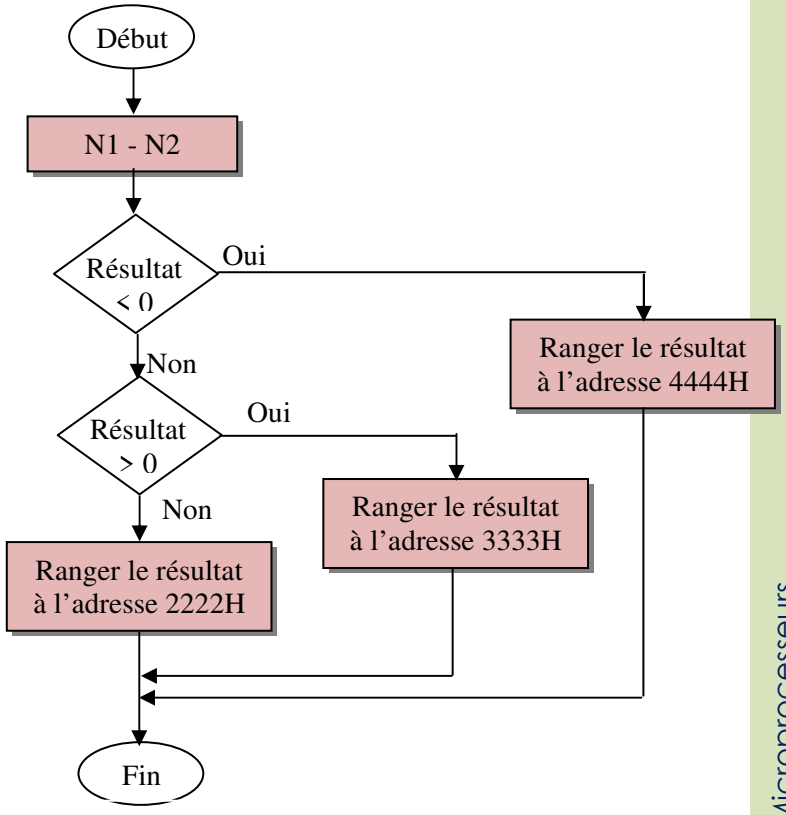
Transférer un block mémoire de longueur 10 octets de l'adresse 8000 à l'adresse 8080.

EXERCICE 08 :

Si l'instruction 'NOP' prend 1µs du temps d'exécution. Ecrire un programme de temporisation de 1 ms.

EXERCICE 9 :

Soit l'organigramme suivant :

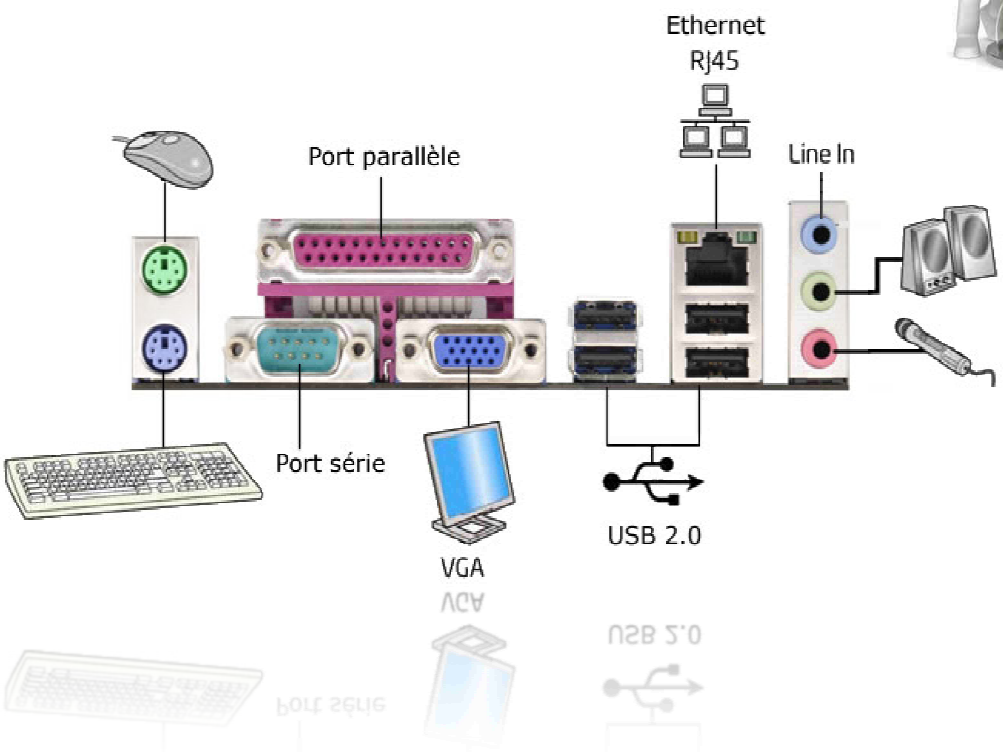


N1 et N2 sont des nombres se trouvant à l'adresse 1100H et 1101H, respectivement.

Ecrire en langage assembleur 8086 le programme correspondant à l'organigramme précédent.

CHAPITRE 5

Les interfaces d'Entrées / Sorties



CHAPITRE 5

LES INTERFACES D'ENTREES/SORTIES

5.1. Généralités

Un système de traitement à microprocesseur communique avec les périphériques extérieurs par écriture ou lecture de valeurs numériques binaires (données codées).

L'interface, appelée aussi **unité d'échange d'entrée-sortie** ou **PORT** d'entrée-sortie est un sous ensemble matérielle, logiciels et des spécifications permettant à l'UC d'échanger des informations avec le monde extérieur. Autrement dit, l'**interface** d'entrée/sortie est la fonction qui permet de transférer les données entre le système de traitement et un périphérique (imprimante, écran vidéo, clavier, capteurs, souris, ...).

Les **fonctions principales** (rôle) de l'interface d'entrée-sortie sont :

- ✓ La communication avec le **CPU** et les périphériques.
- ✓ La mémorisation temporaire des données.
- ✓ La détection et la correction des erreurs.
- ✓ Adaptation physique et logique.
- ✓ Accès par adressage.
- ✓ Synchronisation des échanges.

5.2. Les différents types d'interfaces

On distingue en général deux sortes d'interfaces :

- ✓ L'interface bus interne.
- ✓ L'interface externe.

La structure d'une carte interface est représentée par le schéma de la figure 5.1. L'interface physique constitue :

- ✓ Les signaux de donnée, d'adresse et de contrôle avec leurs caractéristiques électriques et temporelles.
- ✓ Les connecteurs, les câbles et l'assignation des bronches aux signaux

Son principe de fonctionnement est le suivant :

- L'unité centrale (microprocesseur) **LIT** ou **ECRIT** le contenu du registre de données dans l'interface qui se charge de la communication avec le périphérique.
- Les bits des mots de données (8 bits) inscrits dans le registre de données peuvent alors être transférés par une liaison
  - o Parallèle,
  - o Série ou sous d'autres formes : USB, ...etc.

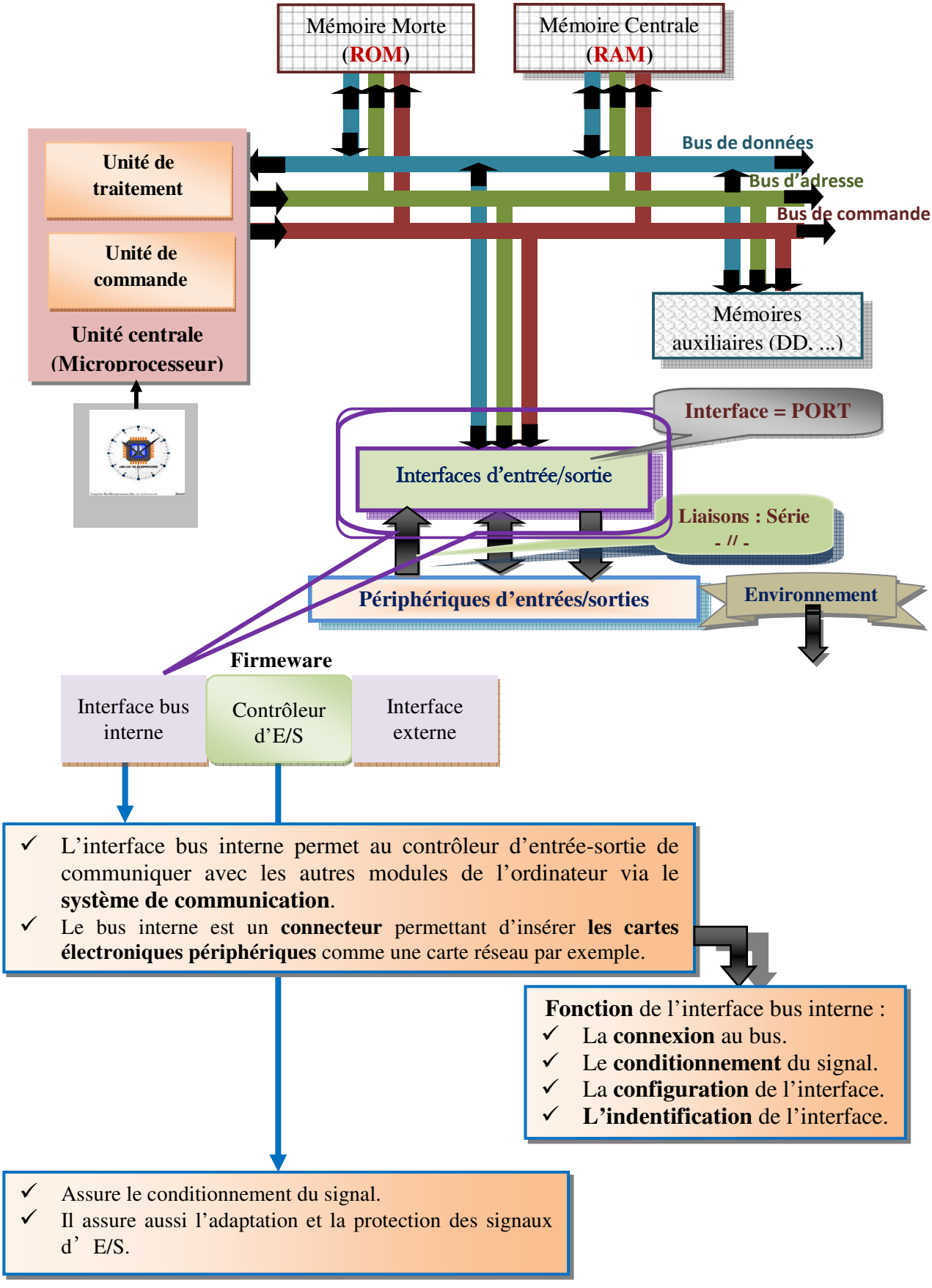


Figure 5.1 : *Système de traitement de l'information.*

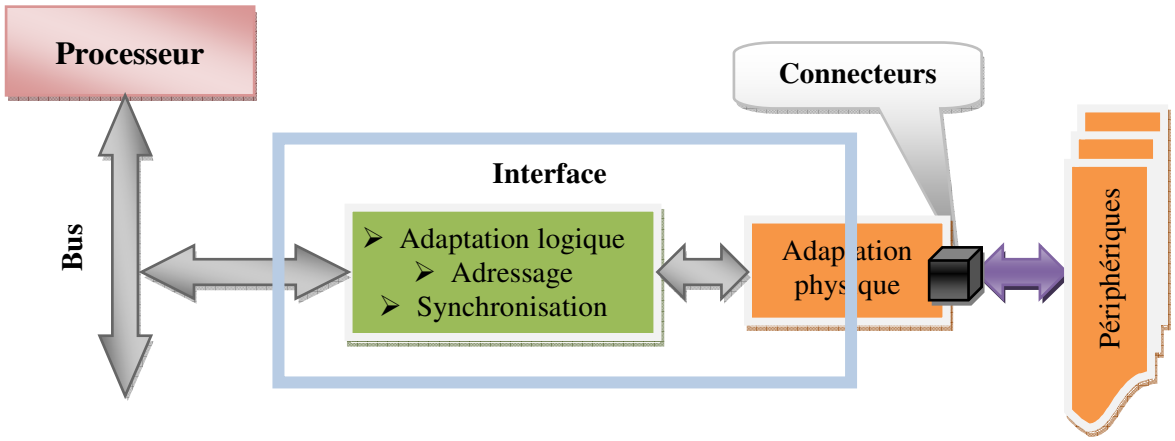
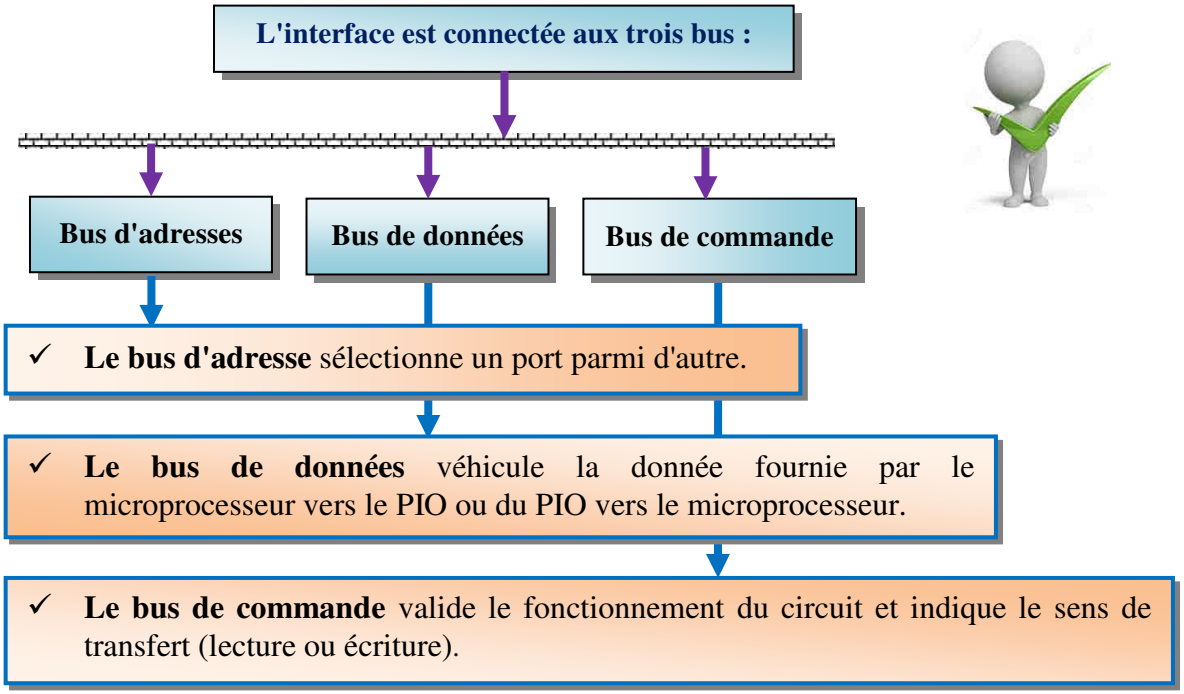


Figure 5.2 : *Emplacement et rôle de l'interface.*

Avant d'envoyer ou de recevoir des informations, le microprocesseur doit connaître l'état du périphérique. En effet, le microprocesseur doit savoir si un périphérique est prêt à recevoir ou à transmettre une information pour que la transmission se fasse correctement.

Il existe 2 modes d'échange d'information :

- ✓ Le mode programmé par scrutation ou interruption où le microprocesseur sert d'intermédiaire entre la mémoire et le périphérique
- ✓ Le mode en accès direct à la mémoire (**DMA**) où le microprocesseur ne se charge pas de l'échange de données



Pour piloter les périphériques, l'interface d'entrées-sorties disposera

- ✓ D'un registre mémorisant l'adresse du périphérique (**X**),
- ✓ Le **registre de sélection du périphérique** (**Sel_Perph**), et
- ✓ D'un registre permettant l'échange d'informations entre unité centrale et les périphériques, le **registre d'échange** (**RE**) à la manière du registre mot de l'unité de mémoire.

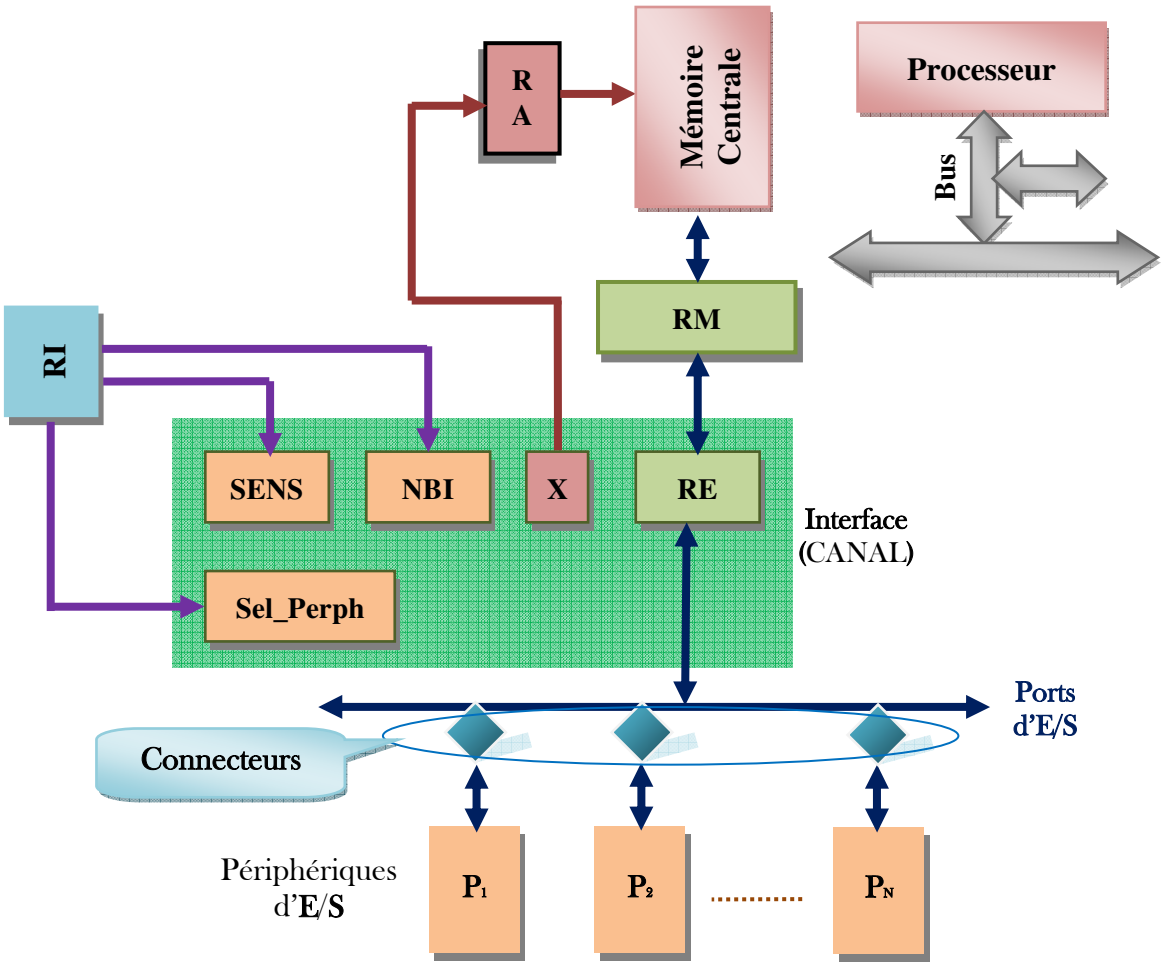


Figure 5.3 : *Fonctionnement de l'interface d'E/S.*

Table 5.1 : *Eléments de l'interface d'E/S.*

Elément	Description	Rôle
SENS	Sens de transfert des données	✓ Pour réaliser le transfert des informations, l'UE ou l'interface d'E/S (CANAL) doit connaitre : ▪ Le **SENS** du transfert et ▪ L'adresse du l'unité périphérique concernant d'une part l'adresse de rangement (**X**) de la première information et le nombre d'informations (**NBI**) à transférée d'autre part. ✓ Ces informations sont fournis par une instruction et sont transférées dans l'UE qui peut alors sélectionnées sans l'intervention de l'unité de commande.
NBI	Nombre d'informations à transférer	
X	Adresse de rangement dans la MC	
RE	Registre d'échange	
Sel_Perph	Sélection du périphérique	
P_1, P_2, …, P_N	Périphériques d'**E/S**	
RA	Registre d'adresse	
RM	Registre mot	
RI	Registre d'instruction	

Parmi **les connecteurs** (**adaptateurs**) classiques des interfaces d'E/S (voir figure 5.3), on peut citer par exemple:

- ✓ Les connecteurs mini-DIN 6 bronches femelles de l'interface souris PS/2.
- ✓ Les connecteurs sub-D 25 bronches femelles de l'interface parallèle.
- ✓ Le connecteur jack femelle de l'entrée audio.
- ✓ Le connecteur jack femelle de sortie audio.
- ✓ Connecteur jack femelle du microphone.
- ✓ Connecteurs USB.
- ✓ Les connecteurs sub-D 9 bronches femelles de l'interface série.
- ✓ Les connecteurs mini-DIN 6 bronches femelles de l'interface clavier PS/2.
- ✓ etc.

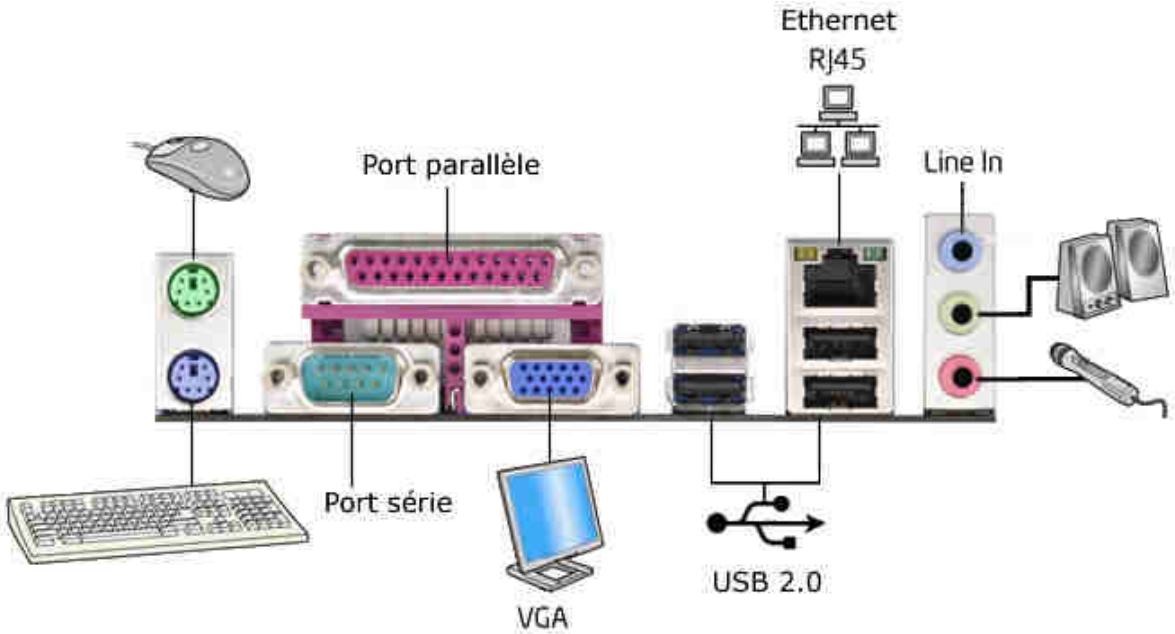


Figure 5.4 : *Connecteurs classiques des interfaces d'E/S.*

Selon le mode d'échange des données E/S, on peut classer les interfaces d'Entrées/Sorties selon la figure ci-dessous.

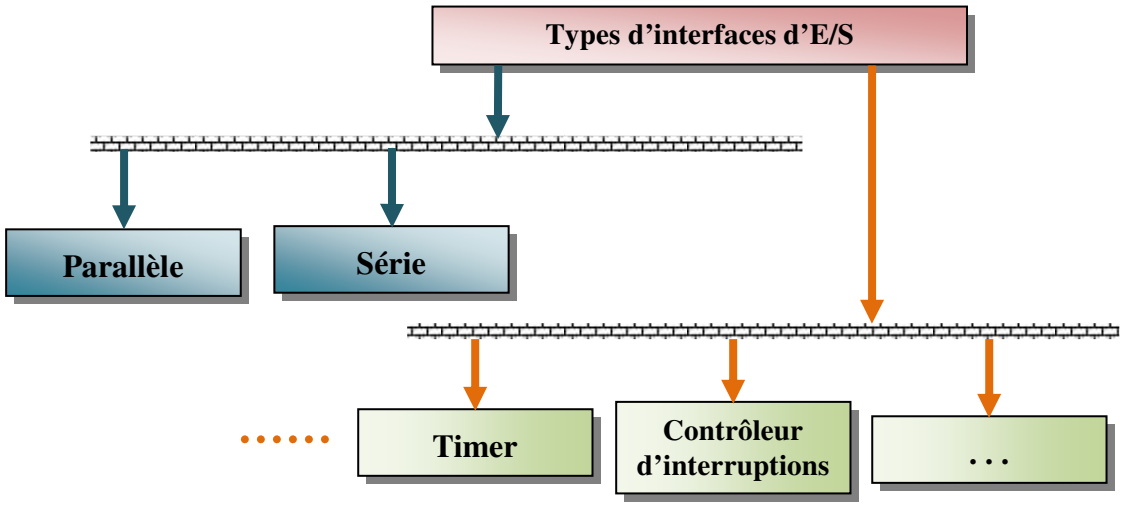
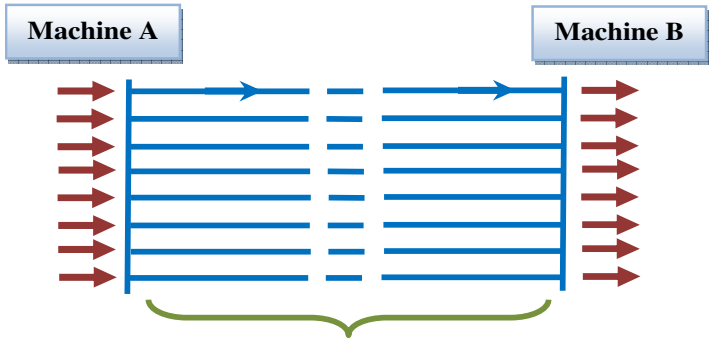


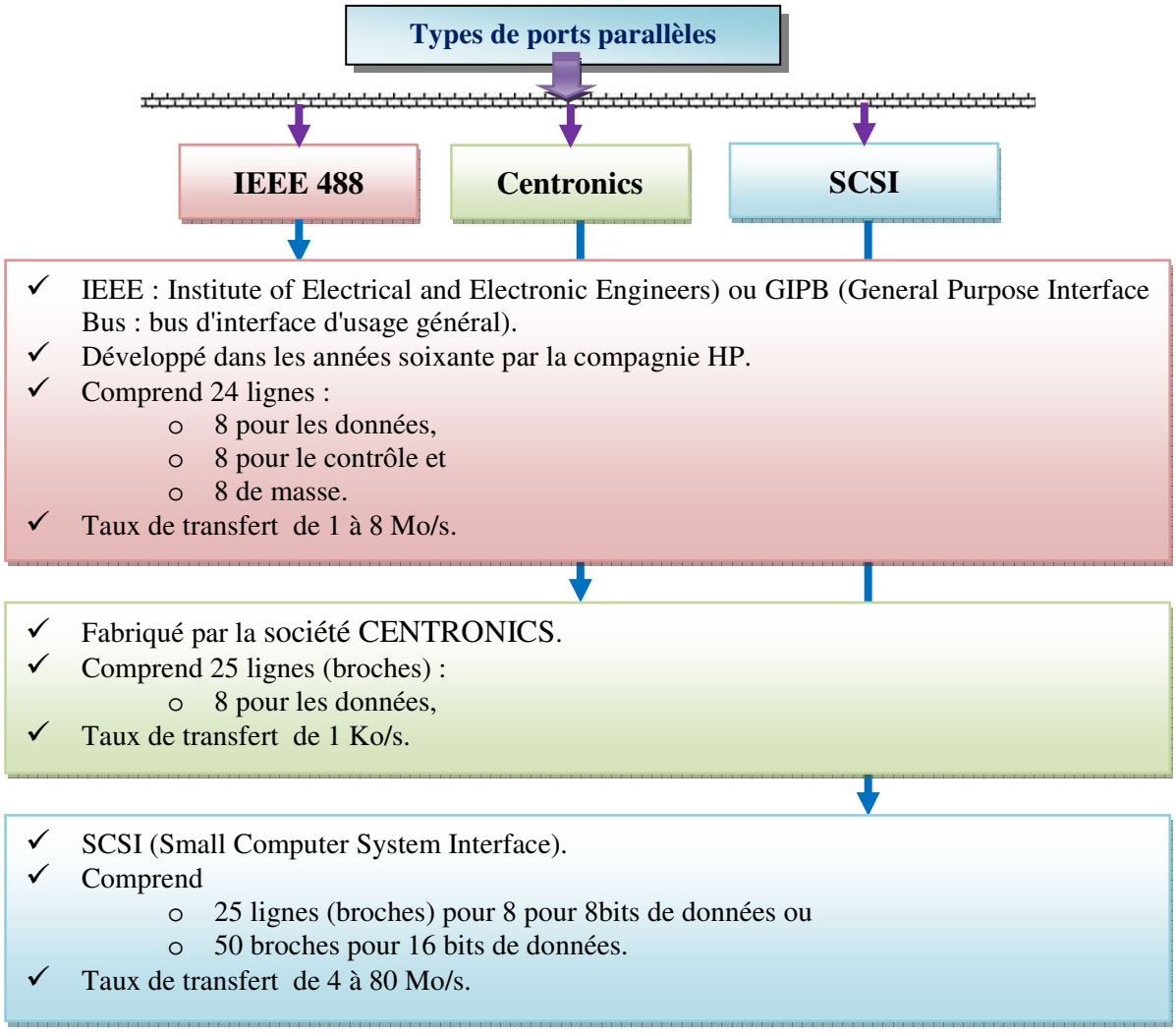
Figure 5.5 : *Différentes types d'interfaces.*

5.2.1. Interface parallèle

La transmission de données en parallèle consiste à envoyer des données simultanément sur plusieurs canaux (fils). Les ports parallèles présents sur les ordinateurs personnels permettent d'envoyer simultanément 8 bits (un octet) par l'intermédiaire de 8 fils.



- ✓ Les "**n = 8**" bits de la donnée à transmettre entre le système et le périphérique sont envoyés simultanément.
- ✓ Le câble de transmission nécessite un nombre important de conducteurs (**n = 8** bits d'information + la masse + des lignes de contrôle).
- ✓ Le temps de transmission d'un mot est très rapide (court).
- ✓ La transmission parallèle ne permet pas de couvrir des distances très importantes (prix du câble + pertes d'informations, …etc).



La structure générale de l'interface parallèle (8 bits) est schématisée par la figure ci-dessous.

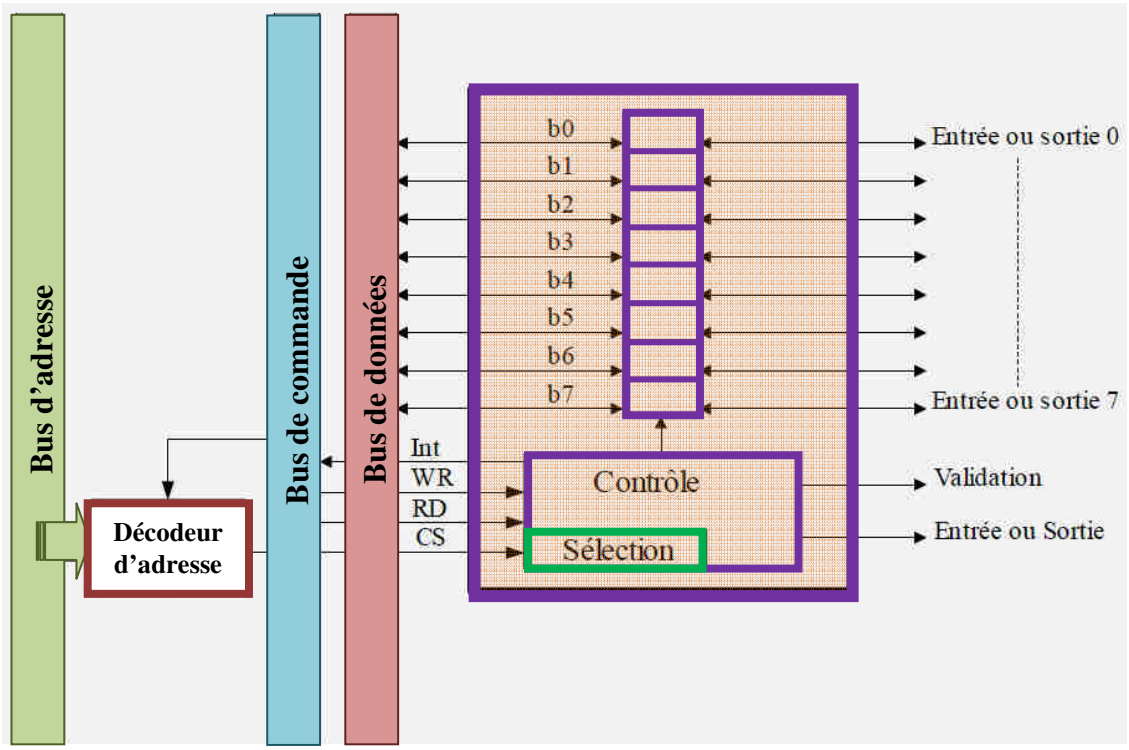


Figure 5.6 : *Structure de l'interface parallèle.*

- ✓ Les premiers ports parallèles bidirectionnels permettaient d'atteindre des débits de l'ordre de 2.4Mb/s. Toutefois des ports **parallèles améliorés** ont été mis au point afin d'obtenir des débits plus élevés:
  - ➢ **Le port EPP** (*Enhanced Parralel Port*) a permis d'atteindre des débits de l'ordre de 8 à **16 Mbps**.
  - ➢ **Le port ECP** (*Enhanced Capabilities Port*), mis au point par la société *HP* et *Microsoft*. Il reprend les caractéristiques du port EPP en lui ajoutant un support *Plug and Play*, c'est-à-dire la possibilité pour l'ordinateur de reconnaître les périphériques branchés.

Les ports parallèles sont, comme les ports séries, intégrés à la carte-mère. Les connecteurs DB25 permettent de connecter un élément extérieur (une imprimante par exemple).

5.2.1.1. Interface (parallèle) périphérique programmable PPI 8255

C'est l'un des **circuits programmables** (peut être programmé en entrée ou en sortie par programme) les plus courants est le circuit Interface périphérique programmable PPI Intel 8255. Il dispose de 4 groupes (de 4 ou 8 lignes) d'entrées/sorties.

- Son rôle est de transférer des données du microprocesseur vers des périphériques et inversement.
- Tout les bits de données sont envoyés ou reçus simultanément.

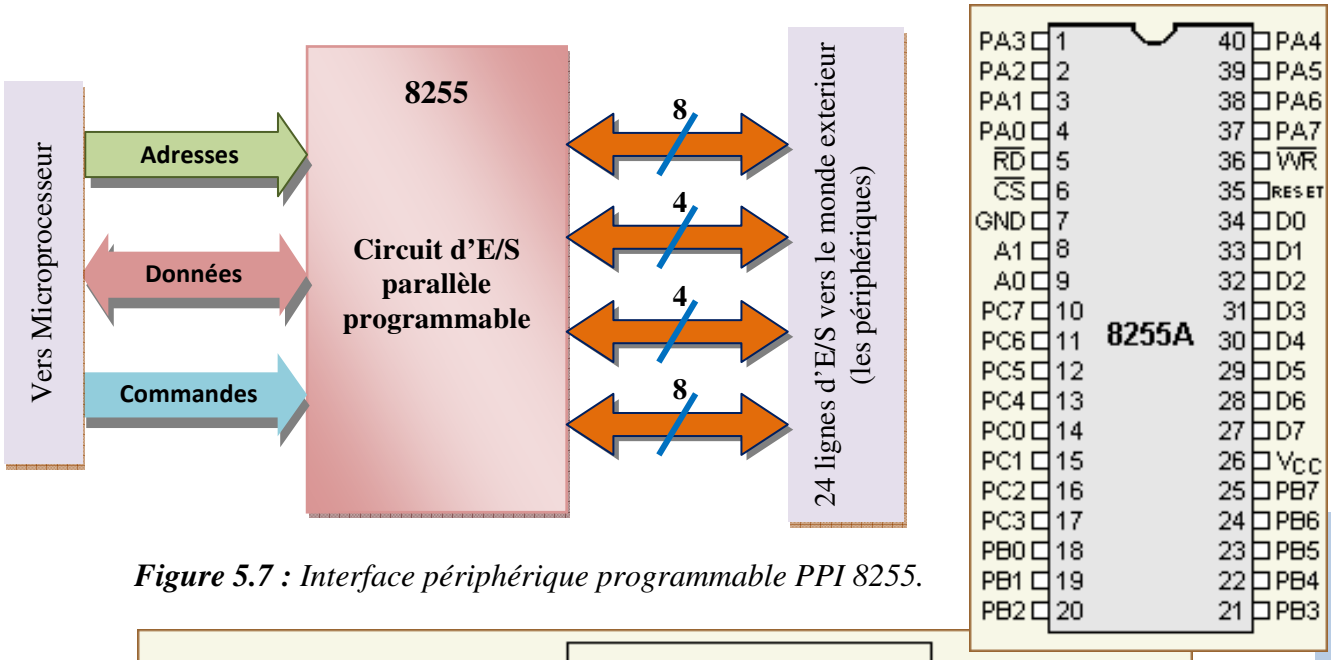


Figure 5.7 : *Interface périphérique programmable PPI 8255.*

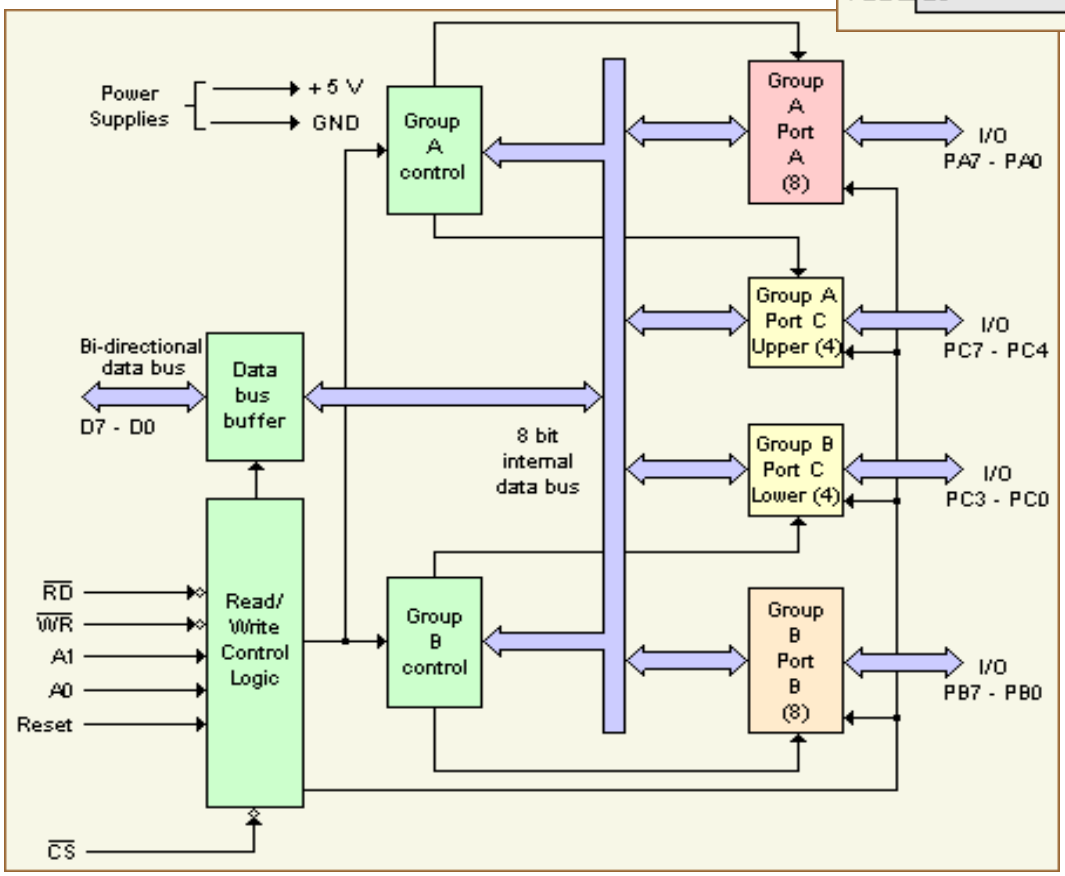


Figure 5.8 : *Architecture interne du circuit PPI 8255.*

Le 8255 contient 4 registres :

- Trois registres contenant les données présentent sur les ports A, B et C.
- Un registre de commande pour la configuration des ports A, B, et C en entrées et/ou en sorties.
- Les lignes d'adresse A0 et A1 définissent les adresses des registres du 8255.

A1	A0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	Opération
0	0	0	1	0	✓ Lecture du port A
0	1	0	1	0	✓ Lecture du port B
1	0	0	1	0	✓ Lecture du port C
0	0	1	0	0	✓ Ecriture du port A
0	1	1	0	0	✓ Ecriture du port B
1	0	1	0	0	✓ Ecriture du port C
1	1	1	0	0	✓ Ecriture du registre de commande
x	x	x	x	1	✓ Pas de transaction
1	1	0	1	0	✓ Illégal
x	x	1	1	0	✓ Pas de transaction

Le 8255 permet 3 modes de fonctionnement différents (modes 0, 1 et 2)

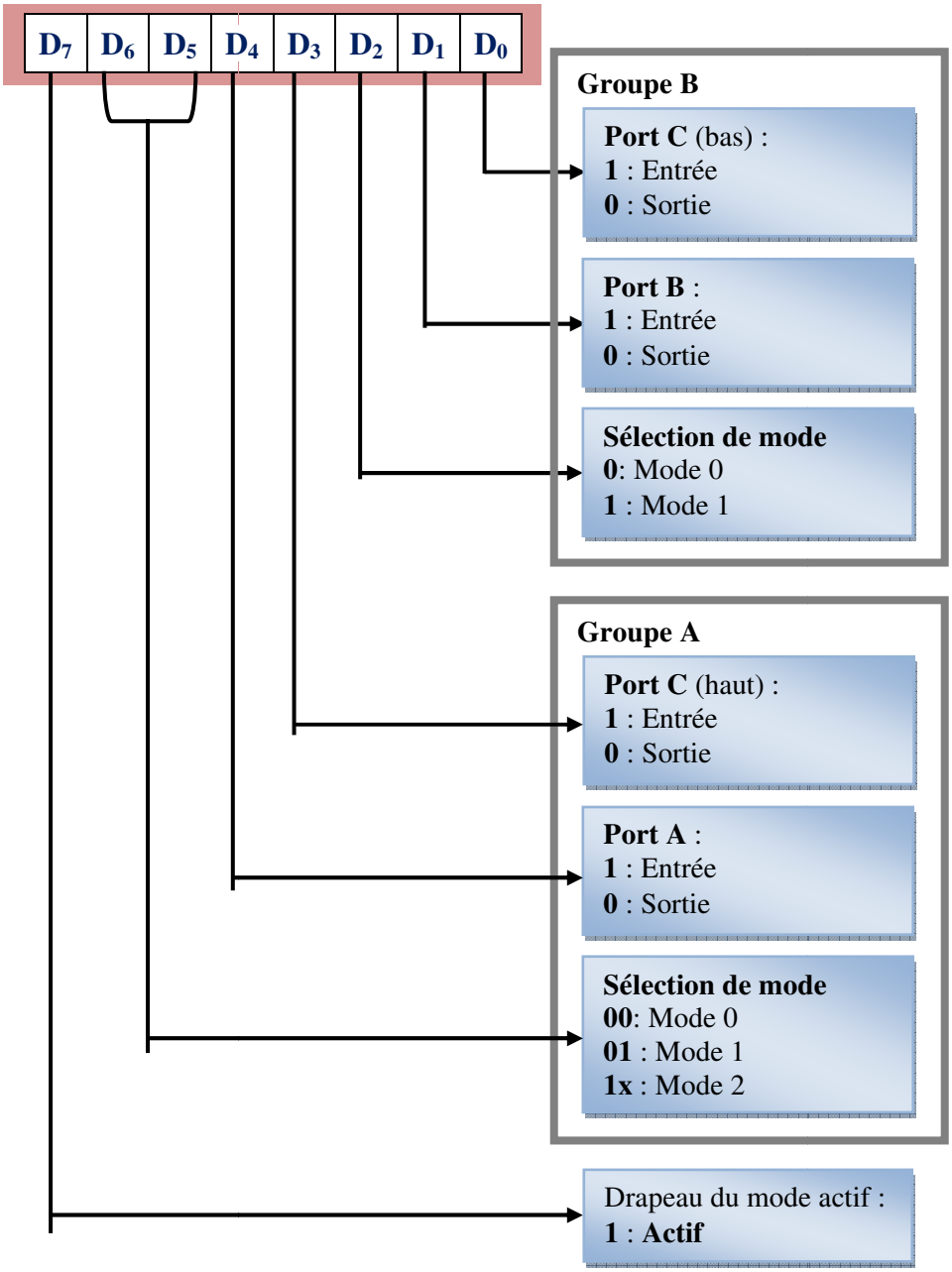
- **Mode 0**: Ports A et B sont configurés en entrée ou en sortie et le port C est divisé en deux groupes de 4-bits qui sont configurés en entrée ou en sortie.
- **Mode 1**: Même chose que mode 0, sauf que le port C est utilisé pour le *handshaking* et le contrôle.
- **Mode 2**: Port A est bidirectionnel (entrée et sortie) et Port C est utilisé pour le *handshaking*. Port B n'est pas utilisé. Ce qui donne 24 lignes commandables.

- Les configurations se font par l'écriture d'un mot de commande dans un registre de contrôle à l'adresse : adresse de base +3.
- Les ports A, B et C sont, respectivement, aux adresses : adresse de base, adresse de base+1, adresse de base +2.

Le 8255 comporte 4 adresses :

Adresse	A1	A0	Description
base + 0 (300h)	0	0	Port A
base + 1 (301h)	0	1	Port B
base + 2 (302h)	1	0	Port C
base + 3 (303h)	1	1	Contrôle

La configuration des différents ports du 8255 est effectuée par l'écriture d'un mot de configuration à son adresse de contrôle. Voici la structure de ce mot de contrôle:



Le tableau ainsi que montage ci-dessous illustrent le mode de d'dressage et d'interfaçage du 8255 avec le microprocesseur 8085.

Chip select address lines	Address lines to select port	HEX address	Selected I/O						
A7	A6	A5	A4	A3	A2	A1	A0		
1	0	0	0	0	0	0	0	80H	PORT A
1	0	0	0	0	0	0	1	81H	PORT B
1	0	0	0	0	0	1	0	82H	PORT C
1	0	0	0	0	0	1	1	83H	Chip select register

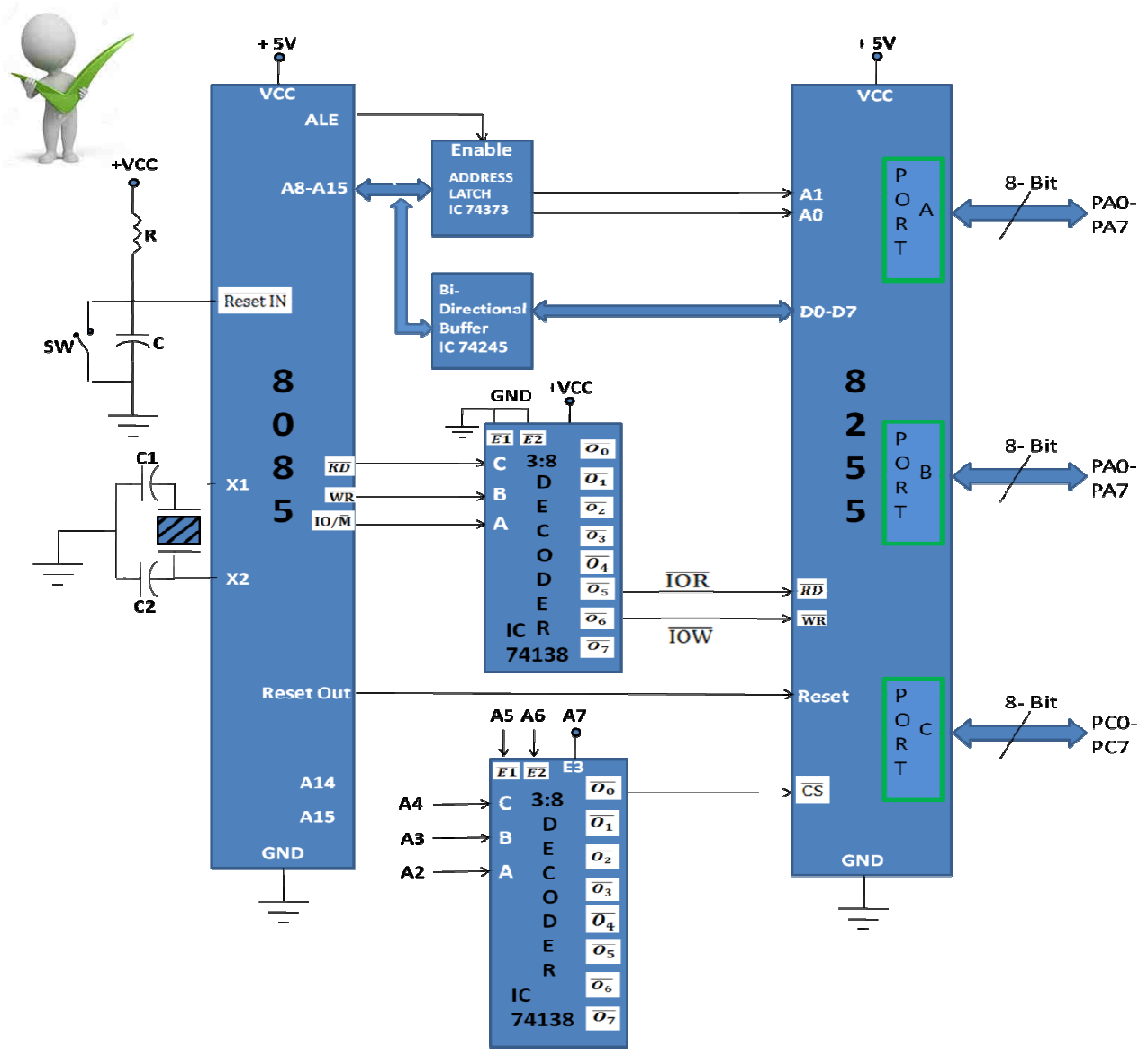
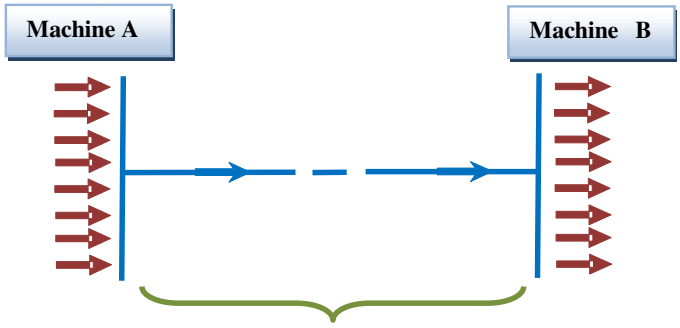


Figure 5.9 : *Interfaçage du 8255 avec le microprocesseur 8085.*

.

5.2.2. Interface Série

- ✓ Les ports (interfaces) série (Norme **RS-232**) représentent les premières interfaces ayant permis aux ordinateurs d'échanger des informations avec le "*monde extérieur*".
- ✓ Le terme *série* désigne un envoi de données *via* un fil unique: les bits sont envoyés les uns à la suite des autres.
- ✓ Les "**n**" bits de la donnée à transmettre entre le système et le périphérique sont envoyés les uns après les autres (en série).
- ✓ Le câble de transmission nécessite un nombre réduit de conducteurs:
  - Tx transmission,
  - Rx réception,
  - La masse + des lignes de contrôle (3 fils minimum).
- ✓ Le temps de transmission d'un mot est plus **important** (**10 fois plus**) qu'une liaison parallèle, mais permet de couvrir des distances importantes.
- ✓ C'est un composant spécialisé appelé SIO (de l'anglais: Serial Input-Output) qui réalise matériellement l'interface série.
- ✓ Ce port permet de connecter une table traçante, un Minitel, un modem, etc.



Liaison série

- Le nombre de bits envoyés pendant un temps d'une seconde définit la vitesse de liaison en baud (**1 baud = 1 bit par seconde**, **bit de données et de contrôle**).

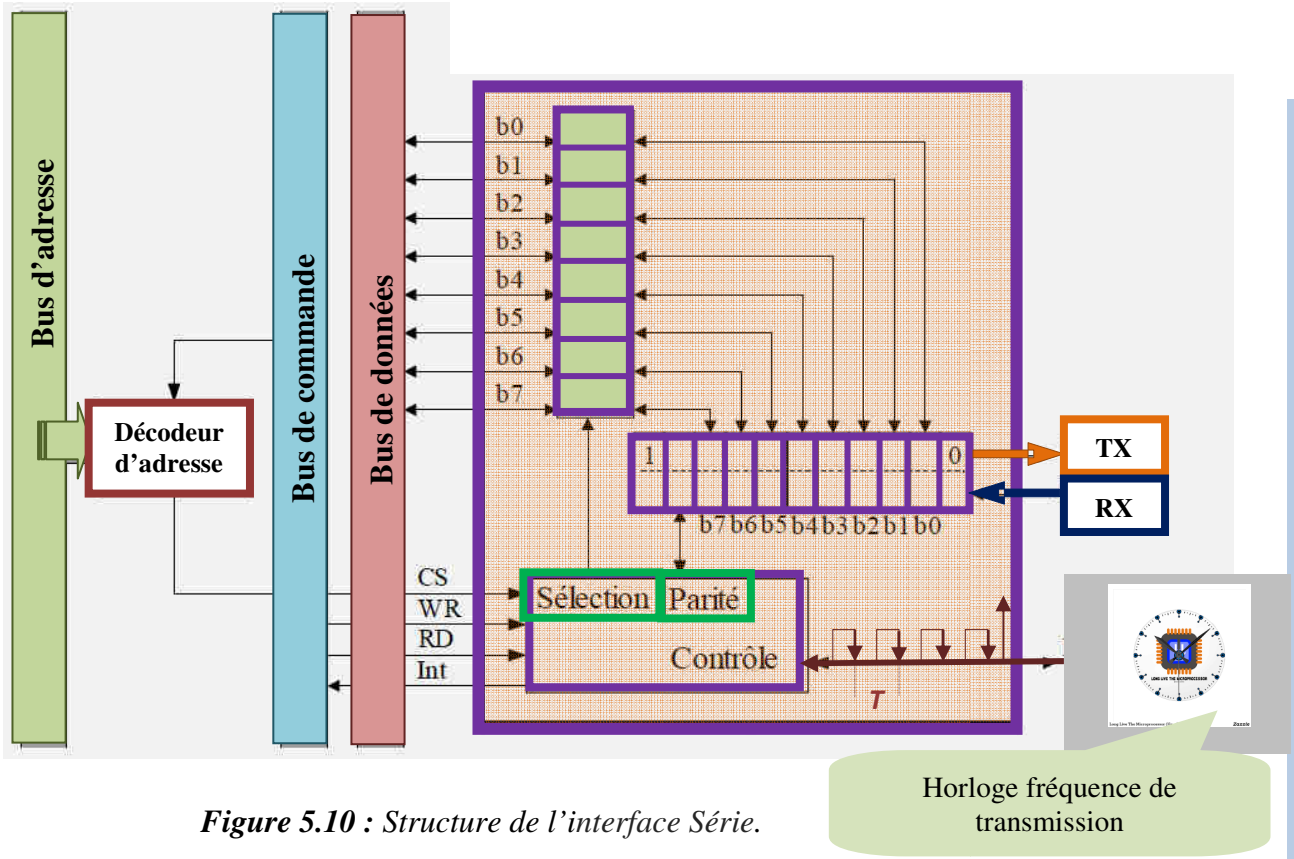
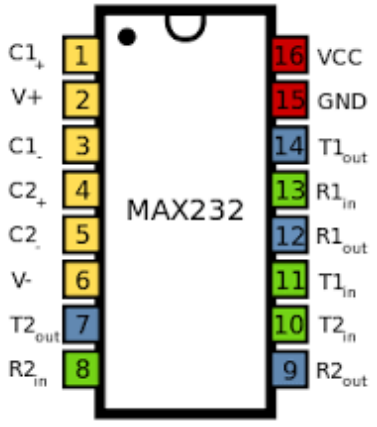


Figure 5.10 : *Structure de l'interface Série.*

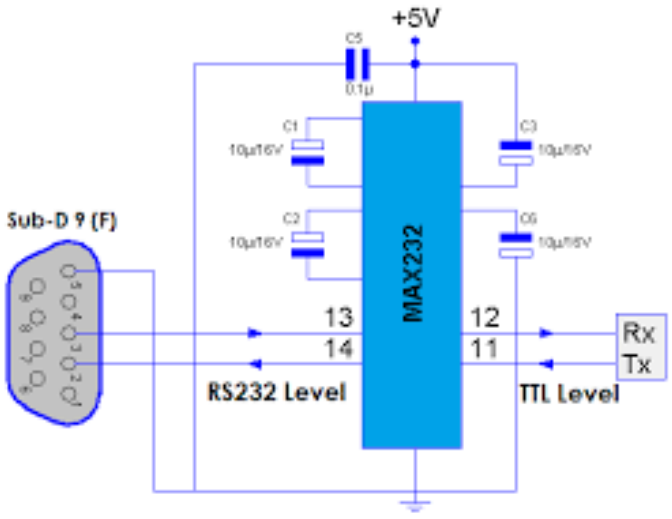
- **Exemple de circuit :**

L'adaptation des données se fait à l'aide d'un circuit adaptateur de ligne (ex : MAX232), qui transforme les niveaux logiques issus du système numérique en niveaux logiques compatibles avec les normes RS-232C et vice versa.

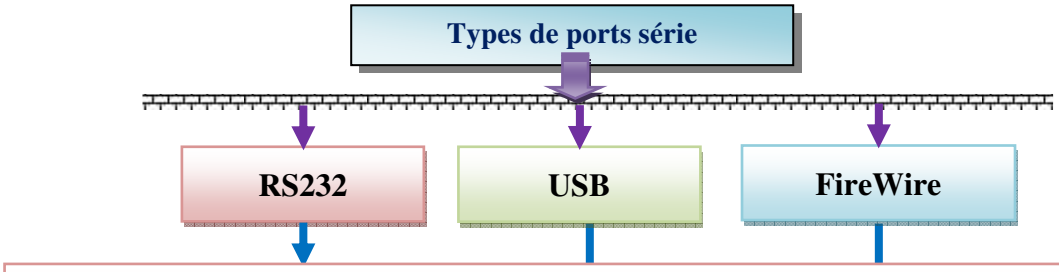
Circuit MAX232(A) en Boitier DIP



Nbr	Name	Purpose	Signal Voltage	Capacitor Value MAX232	Capacitor Value MAX232A
1	C1+	+ connector for capacitor C1	capacitor should stand at least 16V	1µF	100nF
2	V+	output of voltage pump	+10V, capacitor should stand at least 16V	1µF to VCC	100nF to VCC
3	C1-	- connector for capacitor C1	capacitor should stand at least 16V	1µF	100nF
4	C2+	+ connector for capacitor C2	capacitor should stand at least 16V	1µF	100nF
5	C2-	- connector for capacitor C2	capacitor should stand at least 16V	1µF	100nF
6	V-	output of voltage pump / inverter	-10V, capacitor should stand at least 16V	1µF to GND	100nF to GND
7	T2out	Driver 2 output	RS-232		
8	R2in	Receiver 2 input	RS-232		
9	R2out	Receiver 2 output	TTL		
10	T2in	Driver 2 input	TTL		
11	T1in	Driver 1 input	TTL		
12	R1out	Receiver 1 output	TTL		
13	R1in	Receiver 1 input	RS-232		
14	T1out	Driver 1 output	RS-232		
15	GND	Ground	0V	1µF to VCC	100nF to VCC
16	VCC	Power supply	+5V	see above	see above



PC Serial PORT communication using MAX232



- ✓ Taux de transfert maximum de 1 Mbaud.
- ✓ C'est l'un des protocoles les plus vieux et les plus répandus.
- ✓ La plupart des ordinateurs possèdent une interface RS232, sur laquelle un modem, un clavier ou une souris peuvent venir se connecter.

- ✓ USB : *Universal Serial Bus*.
- ✓ Permet de connecter à l'unité centrale jusqu'à **127 périphériques** et de connecter et déconnecter l'un de ces périphériques alors que l'ordinateur est en cours de traitement.
- ✓ Le standard USB supporte la communication "Plug and Play".
- ✓ l'indique, basé sur une architecture de type série.
- ✓ Il s'agit toutefois d'une interface entrée-sortie beaucoup plus rapide que les ports série standards.
- ✓ L'architecture USB est une amélioration techniques de l'interface série.
- ✓ L'architecture USB a pour caractéristique de fournir l'alimentation électrique aux périphériques qu'elle relie.
- ✓ Elle utilise pour cela un câble composé de quatre fils
  - La masse **GND**,
  - L'alimentation **VBUS** et
  - Deux fils de données appelés **D-** et **D+**.

- ✓ IEEE 1394 ou FireWire (ligne de feu).
- ✓ Taux de transfert maximum de 400 Mbaud.
- ✓ Standard récent, il sert pour les périphériques graphiques ou vidéo tels que la caméra numérique.
- ✓ Il permet en outre de connecter jusqu'à **63** périphériques sur une même unité centrale et de connecter et déconnecter l'un de ces périphérique alors que l'ordinateur est en cours de traitement.

Remarques :

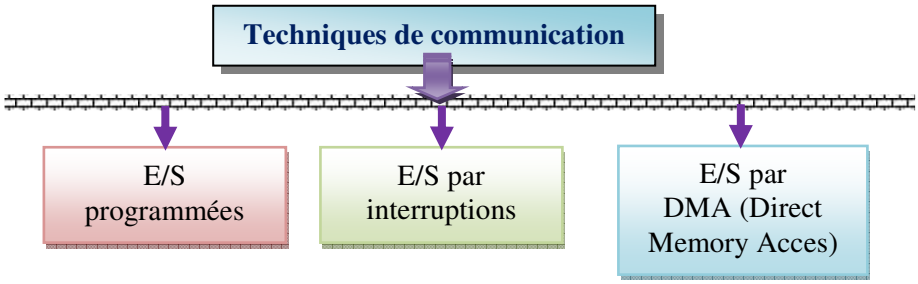
- La norme USB permet le chaînage des périphériques, en **utilisant une topologie en bus ou en étoile**.
- Les périphériques peuvent alors être soit connectés les uns à la suite des autres, soit **ramifiés**.
- La **ramification** se fait à l'aide de boîtiers appelés hubs (ou concentrateurs), comportant une seule entrée et plusieurs sorties.
- Certains sont actifs (fournissant de l'énergie électrique), d'autres passifs.

5.3. Programmation (gestion) des interfaces d'E/S

A chaque type d'interface est associé un contrôleur (circuit) d'E/S. Les contrôleurs d'E/S ont plusieurs fonctions:

- ✓ Lire ou écrire des données du périphérique.
- ✓ Lire ou écrire des données du processeur.
- ✓ Contrôler le périphérique et lui faire exécuter des séquences de tâches.
- ✓ Tester le périphérique et détecter des erreurs.
- ✓ Mettre certaines données du périphérique ou du processeur en mémoire tampon afin d'ajuster les vitesses de communication.

Il existe plusieurs techniques pour communiquer à partir du processeur vers un périphériques à travers un contrôleur E/S.



L'adressage des Entrées/Sorties

Chaque périphérique est pourvu d'une adresse spécifique. Deux manières d'adressage:

1- **Adressage direct** : Les périphériques sont adressés distinctement des positions mémoires.
2- Adressage des périphériques comme des positions mémoires.

Mode Accès direct mémoire (DMA)

- ✓ Relier directement le périphérique à la mémoire sans intervention de la CPU.
- ✓ Très grande vitesse de transfert de données.
- ✓ Trois techniques :
  - DMA par arrêt du microprocesseur
  - DMA par vol de cycle
  - DMA multiplexé

5.6. Exercices

Justifier le choix de votre réponse aux questionnaires ci-dessous.

1. **La souris est un périphérique :**
   - D'entrées.
   - De sorties.
   - D'entrées et de sorties.

2. **Les périphériques de sorties se connectent sur :**
   - Le port PS/2.
   - Les ports séries et parallèles.
   - L'alimentation.

3. **Les ports séries et parallèles sont déclarés dans :**
   - Le config.sys.
   - L'autoexec.bat.
   - Le Bios.

4. **Le mode de transmission utilisé par un port séries est :**

- Unidirectionnel.
- Bidirectionnel.

5. Les ports parallèles sont nommés :

- COM1, COM2, …
- LPT1, LPT2, …
- POR1, POR2, …

6. Citez au moins quatre types de cartes d'extension.

-
-
-
-

7. Le bus PCI travaille avec quel type de carte ?

- Des cartes ISA ou PCI.
- Des cartes PCI.
- Des cartes VLB ou PCI.

8. Quels sont les trois paramètres que l'on a besoin de configurer pour qu'une carte d'extension fonctionne correctement ?

-
-
-

9. L'adresse d'entrées/sorties est utilisée pour :

- Éviter les conflits d'IRQ.
- Définir une plage de l'espace adressable du processeur pour communiquer avec la carte.
- Déclarer au processeur de quel type de carte il s'agit.

10. Le canal DMA permet au processeur de se décharger d'une partie de son travail en assurant la communication directe entre la carte et la mémoire.

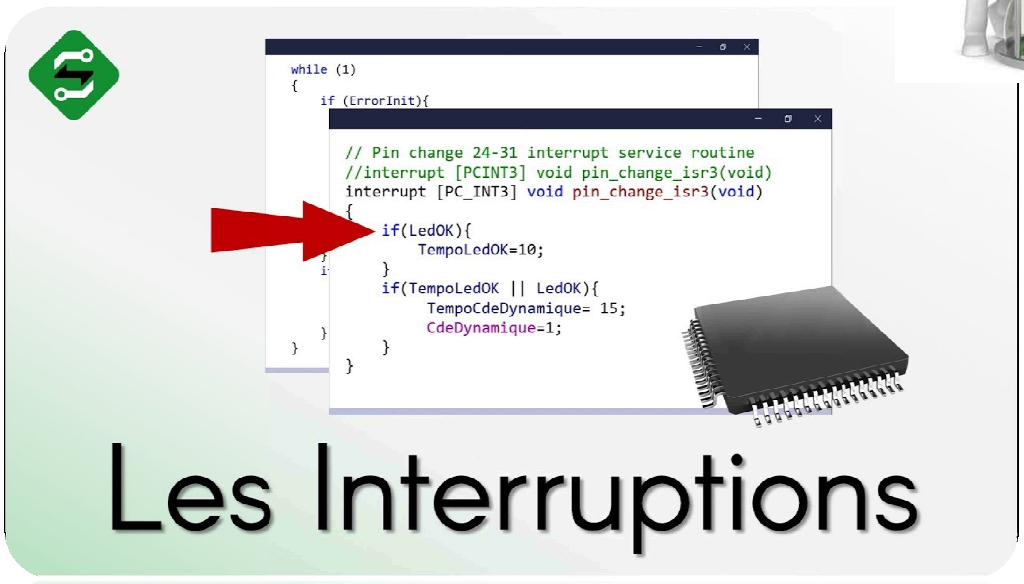
- Vrai.
- Faux.

11. Quelle est la résolution maximale obtenue par une carte VGA ?

- 640 x 480 x 16 couleurs.
- 800 x 600 x 16 couleurs.
- 640 x 480 x 256 couleurs.
- 800 x 600 x 256 couleurs.

CHAPITRE 6

Les interruptions



CHAPITRE 6

LES INTERRUPTIONS

6.1. Définition d'une interruption

Afin d'améliorer les performances du microprocesseur (unité centrale), c'est-à-dire le libérer de certaines tâches, la gestion des périphériques d'entrées-sorties est exécutée par des contrôleurs des périphériques. Comme indique la figure ci-dessous.

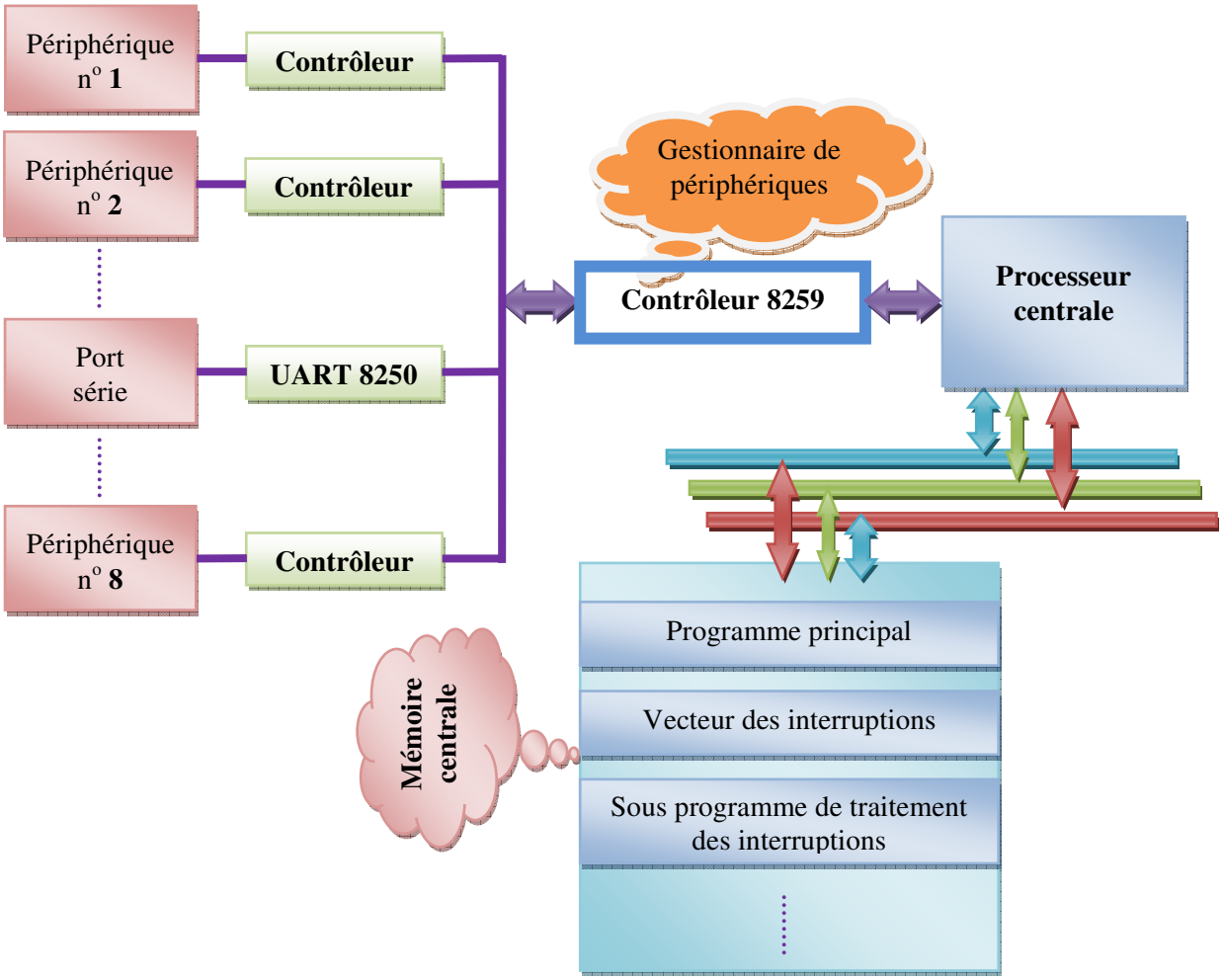


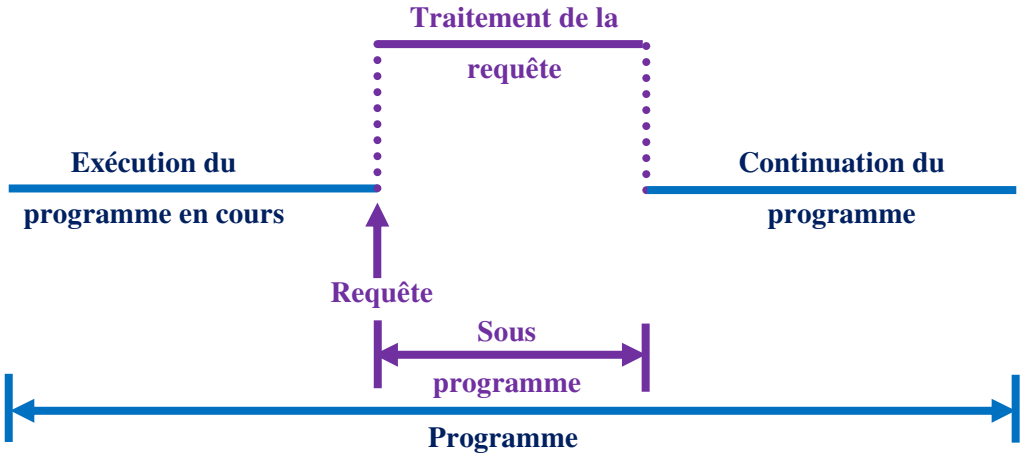
Figure 6.1 : *Synoptique de la gestion des périphériques.*

Le microprocesseur est en permanence susceptible d'exécuter un programme (une tâche quelconque).

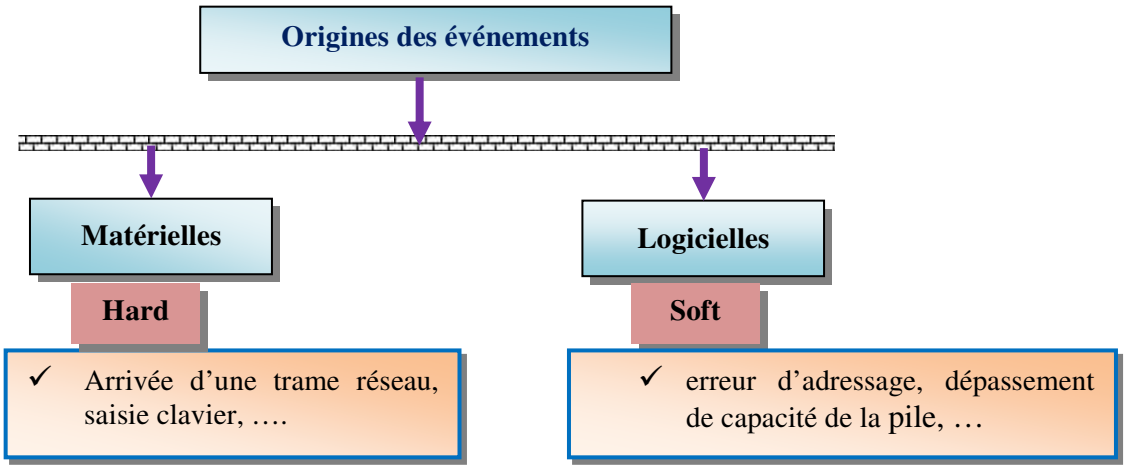
- A quel moment le microprocesseur va prendre en compte les **événements** extérieurs à la séquence d'instructions qu'il exécute **?**

- ✓ **Requête d'un périphérique.**
- ✓ **Appui sur un bouton poussoir.**
- ✓ **Passage d'un objet devant un capteur.**
- ✓ **…etc.**

- Un événement correspond à une **tâche** à exécuter par le microprocesseur.
- Une tâche est codée par une **procédure** (sous programme).
- L'exécution de cette tâche nécessite un délai (rupture).



- Une interruption est un **signal déclenché** par un **événement interne** à la machine ou **externe** (envoyé au microprocesseur par un périphérique ou un programme) pour l'avertir d'un évènement à traiter.



- Le microprocesseur suspend alors l'exécution en cours à la fin de l'opération courante pour réagir à cet évènement et le traiter (le plus souvent, exécution d'une routine).

- Une fois la tâche effectuée, il reprend le cours normal de ses opérations à l' endroit où il avait été interrompu.
- Ces événements peuvent être destinés au processus lui-même.
- Ces événements peuvent être destinés à un autre processus.

Il ya deux méthodes possibles pour recevoir les données provenant des périphériques :

1. **Scrutation périodique (Polling)** : où le programme principal contient des instructions qui lisent cycliquement l'état des ports d'E/S → **Interrogation** en permanence les ports d'E/S.
   - ✓ **Avantages** : Programmation simple.
   - ✓ **Inconvénients** : Pertes de temps et de données sont possibles.
2. **Interruption :** Lorsqu'une donnée apparait sur un périphérique, le circuit d'E/S le signale au microprocesseur pour que celui-ci effectue la lecture de la donnée → C'est une **demande d'interruption** (Interrupt Request : IRQ).
3. Une méthode par Accès Direct à la Mémoire (**DMA**) permet de gérer le transfert de façon autonome.

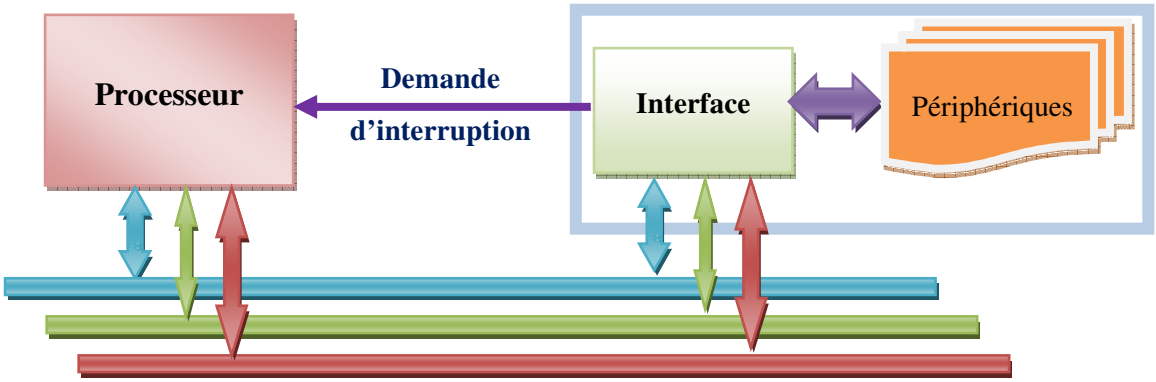


Figure 6.2 : *Demande d'interruption.*

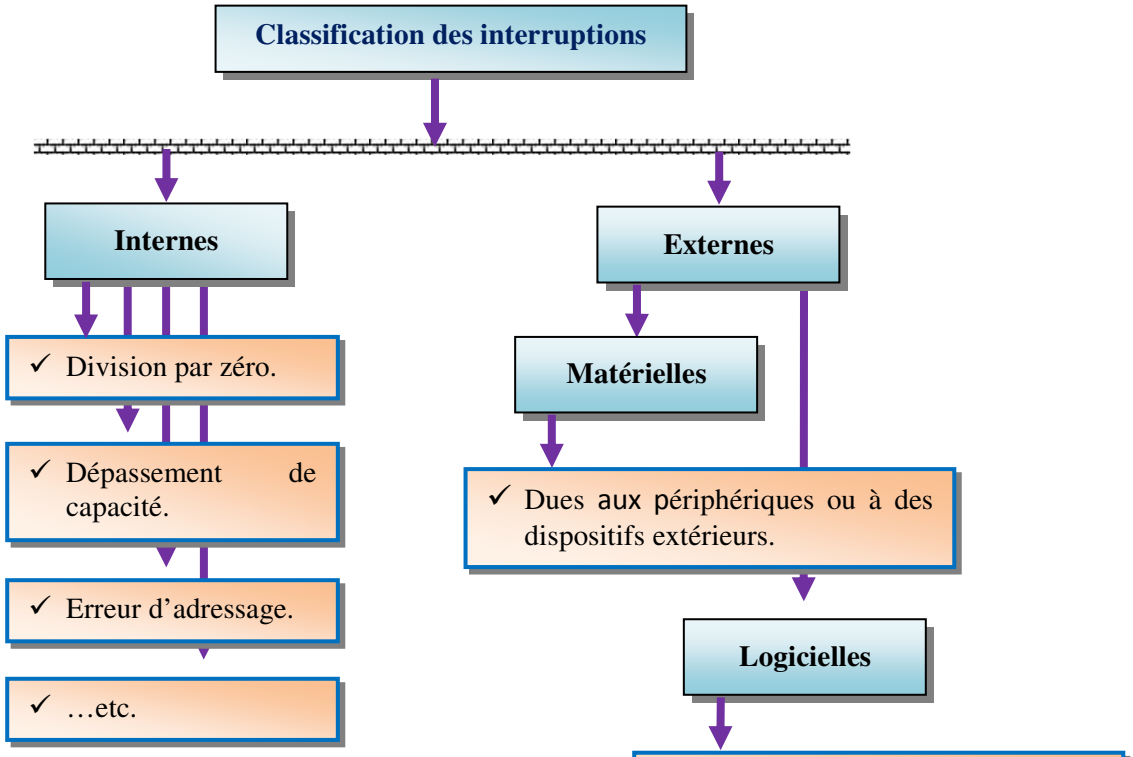


Figure 6.3 : *Classification des interruptions.*

- Le processeur effectue une lecture des ports d'E/S seulement lorsqu'une donnée est disponible, suite à une demande IRQ par le périphérique lui même. Ce qui permet d'éviter le test en permanence des ports d'E/S ainsi que la sécurité des données.

6.2. Processus de traitement d'une interruption par le microprocesseur

- Comment signaler au microprocesseur un événement asynchrone?

Système d'interruptions

Une **interruption** est

➔ Un arrêt temporaire de l'exécution normale d'un programme informatique par le microprocesseur

➔ Afin d'exécuter un autre programme (**routine d'interruption**).

Le mode de dialogue entre le processeur est le périphérique demandeur de requête est schématisé par la figure ci-dessus.

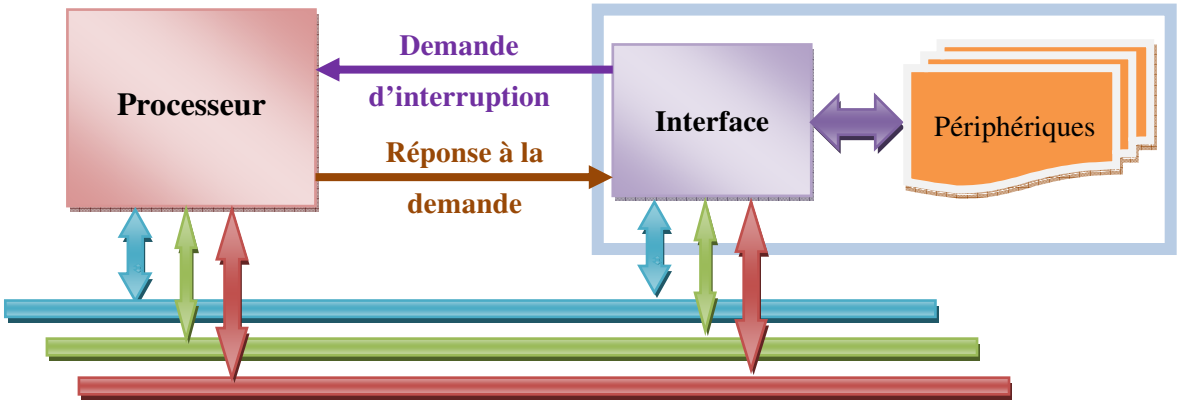


Figure 6.4 : *Mode de dialogue Processeur - Périphérique.*

A la suite d'une **demande d'interruption** par un périphérique d'E/S, le processus de l'exécution d'une interruption sera résumé par les étapes suivantes :

1. Le processeur termine l'exécution de l'instruction en cours d'exécution.
2. Le processeur range le contenu des principaux registres dans un espace mémoire de sauvegarde spéciale (la **pile**) : Drapeaux (flags), pointeurs d'instruction (adresse de retour), …etc. ➔ *Sauvegarder l'état présent du microprocesseur.*
3. Le processeur émet un accusé de réception de demande d'interruption (Interrupt Achnowledge) (**Réponse à la demande**) indiquant au périphérique d'E/S que la demande d'interruption est acceptée ou non (Interruptions **masquées** ou non).

4. Le processeur abandonne l'exécution du programme en cours et va exécuter un **sous-programme** de service de l'interruption (**ISR** : Interrupt Service Routine).
5. Après l'exécution de sous-programme ISR, l'état précédent du processeur sera restauré à partir de la pile → Restitution de l'état du microprocesseur.
6. Le processeur reprend l'exécution du programme déjà abandonné.

Le **masquage** d'une interruption se fait généralement en positionnant un flag dans le registre d'état.
Il existe aussi des interruptions **non masquables** qui sont toujours prises en compte par le processeur (Interruptions **prioritaires**)→ (ex : **Reset**).

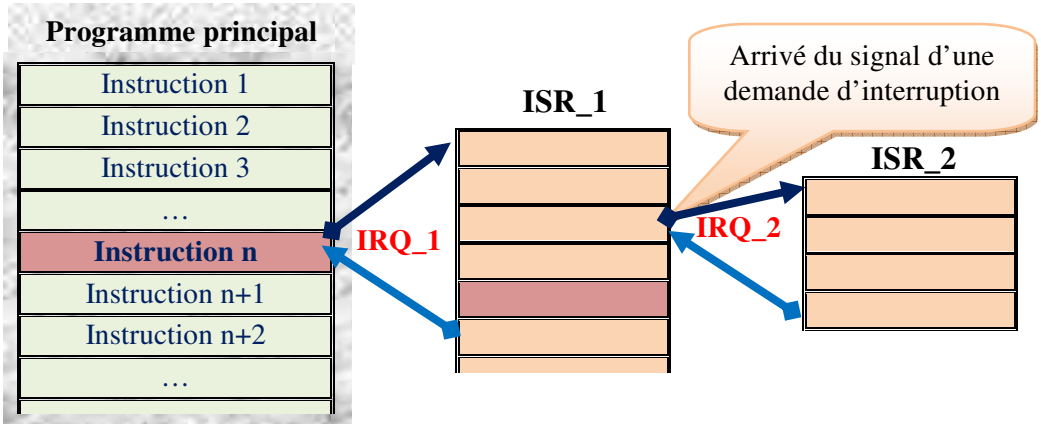


Figure 6.5 : *Principe de l'exécution d'une interruption.*

Le sous-programme de service de l'interruption doit être terminé par l'instruction **IRET** (retour d'interruption).
Un système peut accepter plusieurs sources d'interruption. Chacune est **configurable** par registre (registre d'interruption).

1. Sélectionner les interruptions qui nous intéressent
2. Valider les interruptions de façon globale
3. Ecrire le/les sous programme d'interruption
4. Définir les priorités entres interruptions

En résumé, l'algorithme suivant décrire les étapes de déroulement de l'exécution d'une interruption.

Algorithme 6.1 : *Etapes de déroulement d'une interruption.*

1. Terminer l'instruction en cours
2. Sauvegarder l'état du processeur.
3. Interdire les interruptions de niveau moins élevé.
4. Accéder au **ISR**.
5. Exécuter le **ISR**.
6. Restituer l'état du processeur.
7. Reprendre l'exécution du programme abandonné.

6.2.1. Adresses des sous-programmes d'interruptions

Lorsqu'une interruption survient, le microprocesseur a besoin de connaitre l'adresse de **ISR**.

- ✓ Pour cela, la source d'interruption place sur le bus de données un **code numérique** indiquant la nature de l'interruption.
- ✓ Le microprocesseur utilise ce code pour rechercher dans une table en mémoire centrale l'adresse du sous programme d'interruption à exécuter.
- ✓ Chaque élément de cette table s'appelle un **vecteur d'interruption → Interruptions vectorisées.**

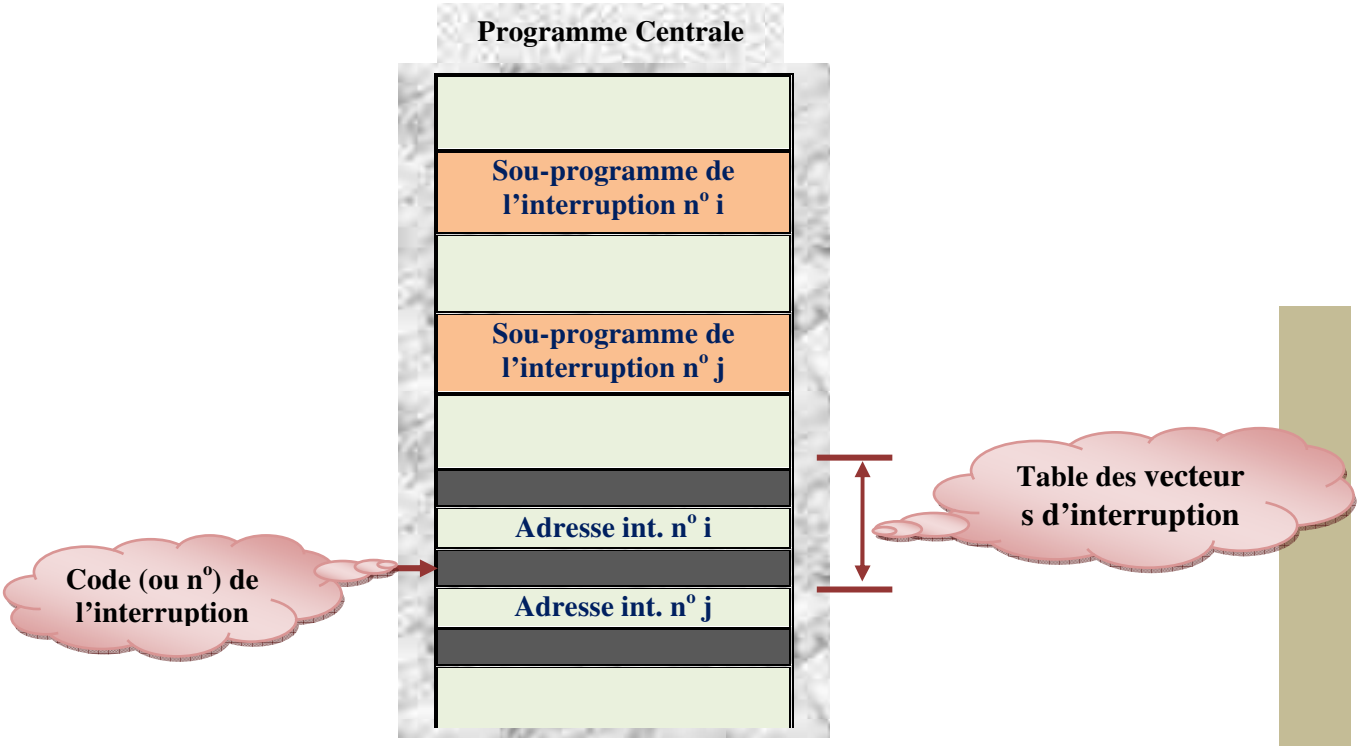
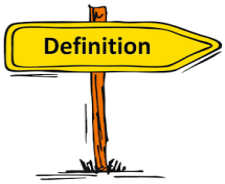


Figure 6.6 : *Table des vecteurs d'interruptions.*

- ✓ La table des vecteurs d'interruptions est chargée par le système d'exploitation au démarrage du système.
- ✓ La table des vecteurs d'interruptions peut être modifiée en cours de fonctionnement.

6.3. Processus de traitement d'une interruption

6.3.1. Interruptions matérielles



Une interruption matérielle est un arrêt de l'exécution d'un programme suite à un **événement matériel →**

- Les demandes d'interruptions matérielles sont effectuées par les périphériques d'E/S.
- Les périphériques d'E/S utilisent les signaux d'interruption du microprocesseur pour signaler une requête d'interruption.
- La gestion des interruptions matérielles s'effectue par un circuit contrôleur : PIC (**Programmable Interrupt Contrôler**).

- Le PIC résout les **priorités des interruptions** et les transmet ensuite au microprocesseur

- le signal d'interruption (la requête) et
- Le numéro d'interruption.

- Un PIC peut traiter un nombre limité de demandes d'interruption simultanément.

- Plusieurs PIC peuvent être utilisés pour augmenter le nombre de demandes.

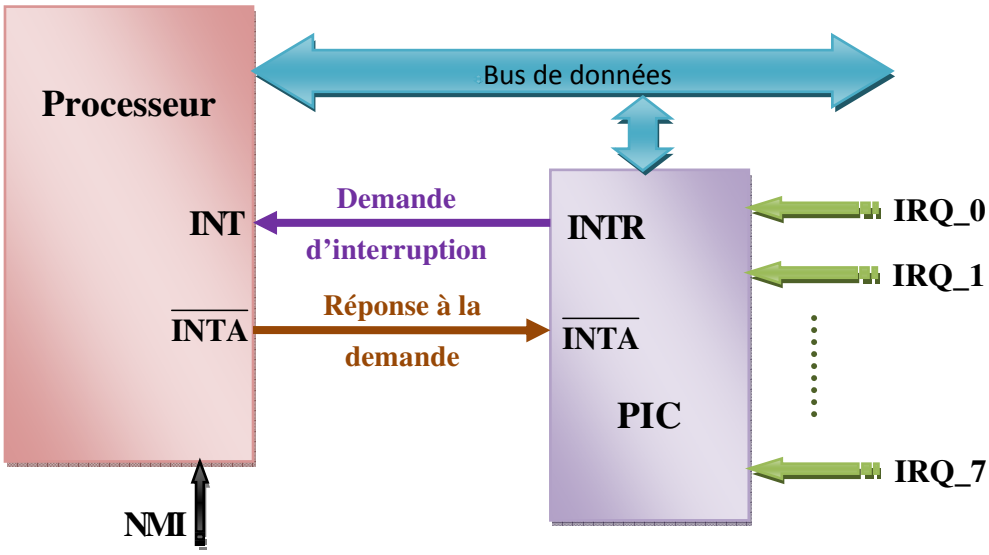


Figure 6.7 : *Schéma de connexion d'un contrôleur d'interruption.*

Contrôleur d'interruption 8259 d'Intel

- ✓ Pour pouvoir connecter plusieurs périphériques utilisant les interruptions, le PIC 8259 est largement utilisé.
- ✓ Le circuit 8259 est un circuit programmable. Il dispose de plusieurs registres d'E/S. Ce composant intercepte toutes les demandes d'interruption des périphériques et les communique au microprocesseur selon leurs priorités.
- ✓ PIC 8259 peut gérer jusqu'à 8 demandes d'interruptions matérielles.
- ✓ Son rôle est de :
  - Recevoir les demandes d'interruptions des périphériques d'E/S.
  - Définir l'ordre de priorité des interruptions provenant de différents périphériques.
  - Générer le signal INTR pour le microprocesseur.
  - Emettre le numéro de l'interruption sur le bus de données.

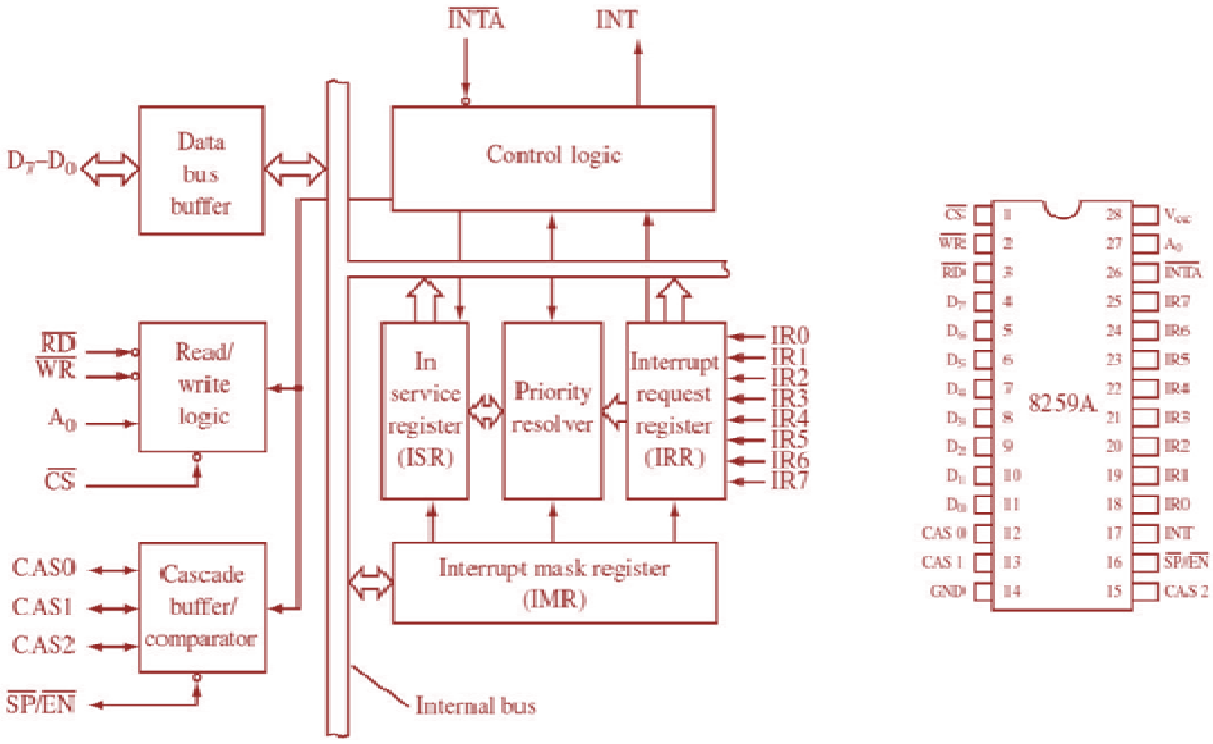


Figure 6.7 : *Brochage et architecture interne du PIC 8259.*

Table 6.1 : *Fonctions des broches du PIC 8259.*

Broche	Description
IR0 - IR7	✓ Niveaux d'interruptions.
D0-D7	✓ Bus de données.
INT	✓ Requête (demande) d'interruption.
$\overline{\text{INTA}}$	✓ Accusé de réception d'interruption.
$\overline{\text{RD}}$	✓ Commande de lecture.
$\overline{\text{WR}}$	✓ Commande d'écriture.
$\overline{\text{CS}}$	✓ Sélection du boitier.
A0	✓ Connexion au bus d'adresse.
CAS 0 **CAS 1** **CAS 2**	✓ Multiplication des interruptions.
SP-EN	✓ PIC en mode maître ou esclave.
VCC, GND	✓ Alimentation, Masse.

6.3.1.1. Traitement d'une interruption

Le processus de traitement des interruptions par le PIC 8259 s'effectue selon le schéma de la figure ci-dessous.

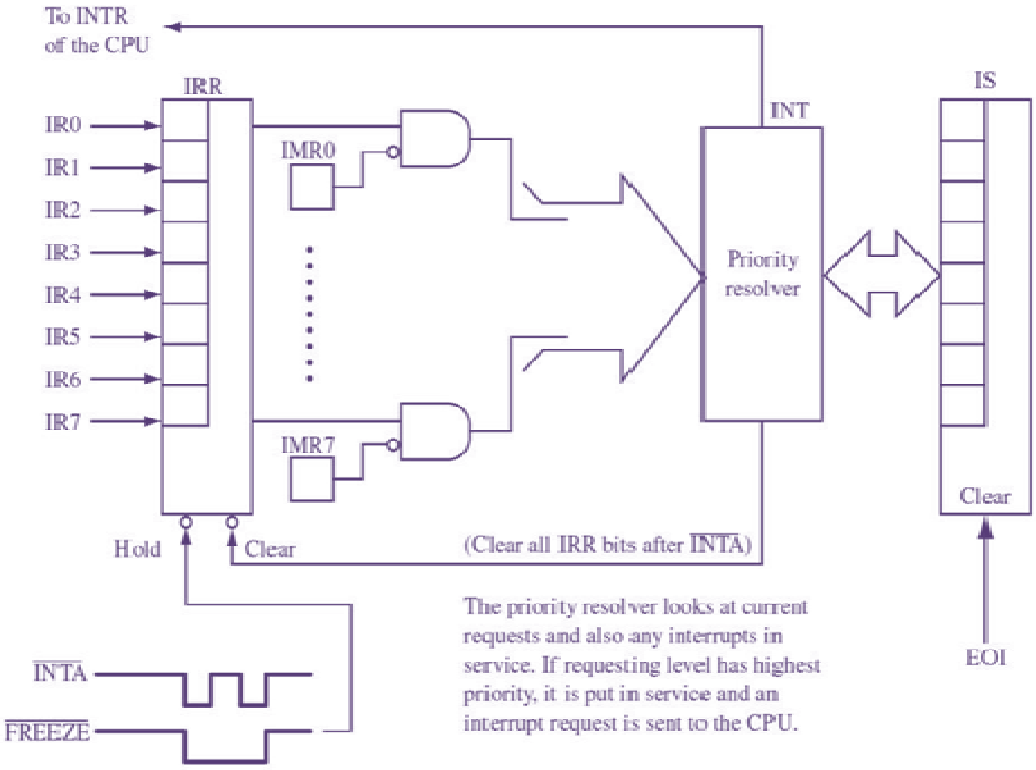


Figure 6.8 : *Traitement des interruptions par le PIC 8259.*

L'algorithme ci-dessous résume les étapes de traitement d'une interruption par le PIC 8259.

Algorithme 6.2 : *Etapes de traitement d'une interruption par le PIC 8259.*

1. Demande du périphérique IRQ0-7.
2. Réception par le PIC + positionnement IRR.
3. Evaluation de la demande (Priorité).
4. PIC informe le microprocesseur => INT.
5. Le microprocesseur prend connaissance du flag IF du registre d'état + contexte et envoi le signal d'accusé de réception => INTA.
6. Réception de INTA par le PIC.
7. Positionnement de ISR et IRR.
8. PIC envoi sur le bus de donnée le type de l'interruption.
9. Le microprocesseur déduit son traitement => Table des vecteurs à l'indice 4*N° de l'interruption.
10. Branchement du **sous-programme** de service de l'interruption (**ISR**).
11. Reprise de la tâche interrompue à la fin di ISR.

6.3.1.2. Interfaçage du up 8085 avec le PIC 8295

Le montage ci-dessous schématise l'interfaçage du up 8085 avec le PIC 8295.

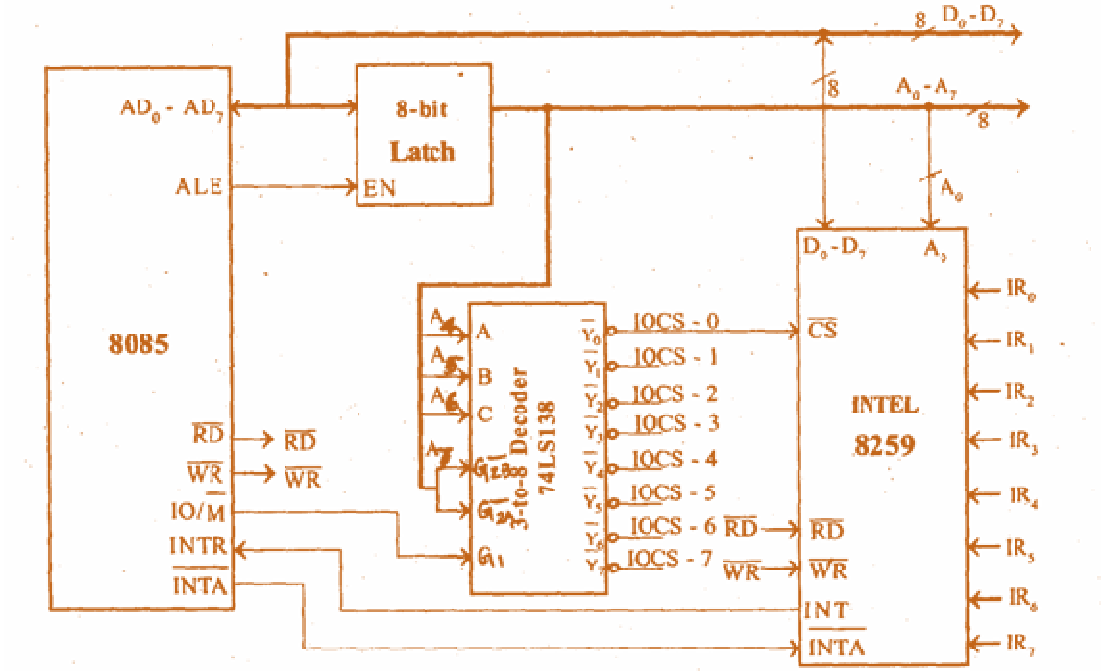


Figure 6.9 : *Interfaçage du up8085 avec le PIC 8295.*

- ✓ Il nécessite deux adresses internes et qui sont A =0 ou A = 1.
- ✓ Il peut être soit la mémoire assignée ou un E/S assignée dans le système. L'interfaçage de 8259 avec le up 8085 représenté sur la figure précédente du type E/S dans le système → lien avec des périphériques extérieurs.
- ✓ Le bus de données D0-D7 sont connectés aux broches D0-D7 du PIC 8259.
- ✓ La ligne d'adresse A0 du up8085 est connectée à la broche A0 du PIC 8259 pour fournir une adresse interne.
- ✓ Le 8259 exige un signal de sélection (chip select signal). Utilisant un décodeur 3x8 pour générer le signal de sélection (CS).
- ✓ Les lignes d'adresse A4, A5 et A6 sont utilisées comme entrées du décodeur 74LS138.
- ✓ Le signal de contrôle IO/M (niveau bas) est utilisé comme signal d'activation haut du décodeur et la ligne A7 comme signal d'activation bas du décodeur.
- ✓ Les adresses d'E/S du PIC 8259 sont données par la table ci-dessous.

Table 6.2 : *Table des adresses du PIC 8259.*

	Binary Address								Hexa address
	Decoder input/ enable			Input to address pin of 8259					
	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
For A_0 of 8259 to be zero	0	0	0	0	x	x	x	0	00
For A_0 of 8259 to be one	0	0	0	0	x	x	x	1	01

Note : *Don't care "x" is considered as zero.*

L'interaction entre le microprocesseur 8085 et le PIC 8259 est donnée par la procédure décrite ci-dessous.

Etape 1 : Le 8259 devrait être programmé en envoyant le mot d'Initialisation de commande (ICW) et mot opérationnel de commande (OCW). Ces mots de commande informeront 8259 à propos de ce qui suit,

1. Type du signal de l'interruption (Level triggered / Edge triggered).
2. Type du microprocesseur (8085/8086).
3. L'adresse d'appel et son intervalle (4 ou 8).
4. Masquage des interruptions.
6. L'ordre de priorité des interruptions.
7. Type de fin d'interruptions.

Etape 2 : Une fois que le PIC 8259 est programmé, il est prêt à accepter le signal d'interruption. Lorsqu'il reçoit une interruption par l'une des lignes d'interruption IR0-IR7 il vérifie la présence de sa priorité et vérifie également si elle est masquée ou non.

Etape 3 : Si l'interruption précédente est terminée et si la demande actuelle a la plus haute priorité et démasqué, elle est entretenu.

Etape 4 : Pour l'entretien de cet interrompre le 8259 va envoyer le signal INT à la broche INTR de 8085.

Etape 5 : En réponse, il attend un accusé de réception INTA (niveau bas) du processeur 8085.

Etape 6 : Lorsque le microprocesseur accepte l'interruption, il envoie trois réponse INTA (niveau bas) une par une.

Etape 7 : En réponse à la première, deuxième et troisième INTA (bas), le 8259 fournira le code opération CALL, l'octet de poids faible de l'adresse d'appel et l'octet haut d'adresse d'appel, respectivement. Une fois que le processeur reçoit le code opération d'appel et son adresse, il enregistre le contenu du compteur de programme (PC) dans la pile et charger l'adresse CALL dans le PC et commencer l'exécution de la routine de service d'interruption stocké dans cette adresse d'appel.

IMPORTANT

Si le nombre de demandes d'interruptions est supérieur à 8, on peut placer plusieurs 8259 en **cascade**, comme indique la figure ci-dessous.

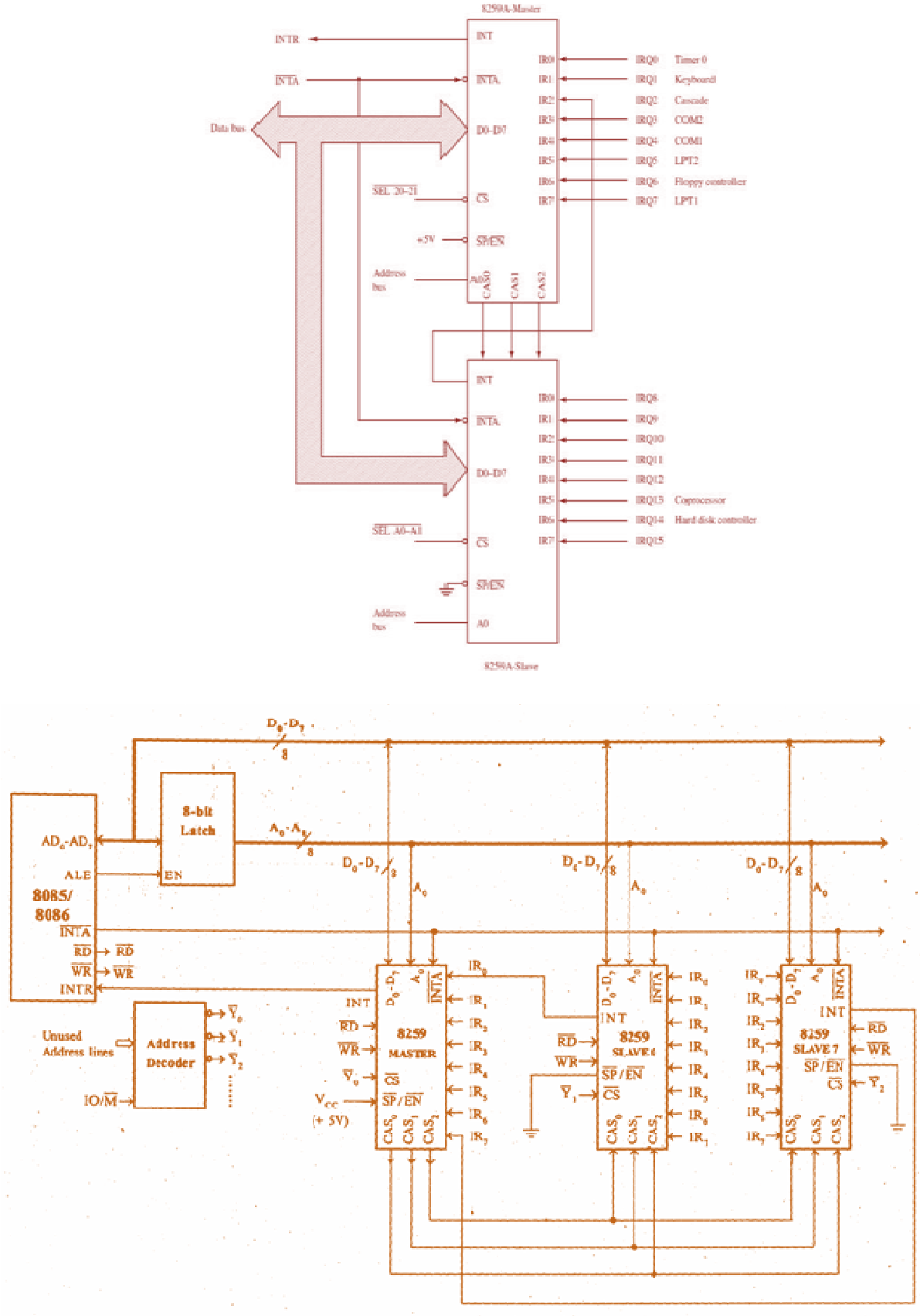


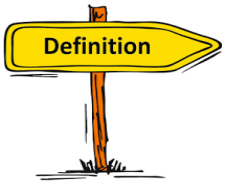
Figure 6.10 : *Mise en cascade des PIC 8295.*

Les instructions utilisées par le microprocesseur 8085 pour le traitement des interruptions sont données par le tableau ci-dessous.

Table 6.3 : instructions des interruptions du up 8085.

EI	EI ;	✓ Enable Interrupts.
DI	DI ;	✓ Disable Interrupts.
SIM	SIM ;	✓ Set Interrupt Masks.
RIM	RIM ;	✓ Read Interrupt Masks.

6.3.2. Interruptions logicielles



Les interruptions logicielles sont générées par un programme d'application ou par le système d'exploitation. Une interruption logicielle est un arrêt de l'exécution d'un programme pour exécuter une routine d'interruption du **DOS** ou **BIOS.**

- Pour les microprocesseurs Intel, les interruptions logicielles (**externes**) sont provoquées par l'instruction **INT** suivie du numéro d'interruption
  - ✓ **INT 21h** : Pour les interruptions du DOS.
  - ✓ **INT 14h** : Pour les interruptions du BIOS.

Exemples de type d'interruption du 8085/8086 :

N° -	Adresse -	Fonction
10 -	040-043 -	BIOS: Fonction vidéo.
11 -	044-047 -	BIOS: Déterminer configuration.
12 -	048-04B -	BIOS: Déterminer la taille mémoire de la RAM.
13 -	04C-04F -	BIOS: Fonctions disquettes/disque dur.
14 -	050-053 -	BIOS: Accès à l'interface sérielle.
15 -	054-057 -	BIOS: Fonctions cassettes ou étendues.
16 -	058-05B -	BIOS: Test du clavier.
17 -	05C-05F -	BIOS: Accès à l'imprimante parallèle.
18 -	060-063 -	Appel du BASIC en ROM.
19 -	064-067 -	BIOS: Lancer système (ALT CTRL DEL).
1A -	068-06B -	BIOS: Lire date et heure.

- Pour les interruptions logicielles (**internes**) sont provoquées selon l'état du registre d'état du microprocesseur, exemple
  - ✓ Si flag **OF =1** indique un overflow → interruption de type 4 est générée par l'instruction spéciale **INT0**.
  - ✓ Le résultat d'une division est de taille supérieur à s destination => interruption de type 0 est déclenchée.
  - ✓ Le flag **TF** a été mis à 1 => le CPU génère une IT de type 1 après chaque instruction ce qui permet de faire du pas à pas.

6.4. Exercices

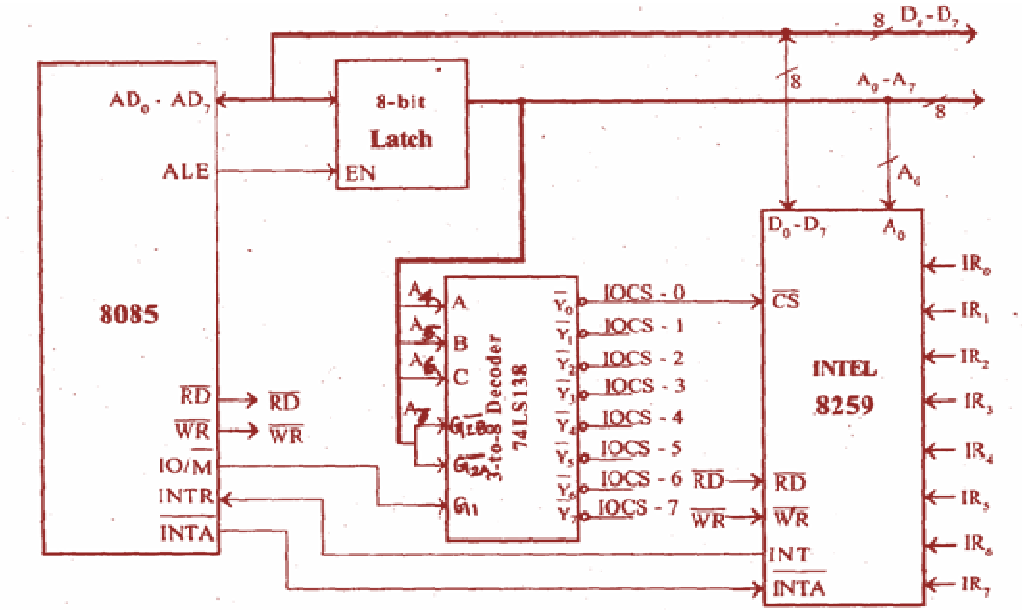
EXERCICE 01 :

- Définir une interruption ?
- D'un point de vue logiciel, que fait généralement un microprocesseur lorsqu'il détecte un signal.
- Pourquoi un ordinateur se sert-il de la pile lorsqu'une interruption se produit? d'interruption provenant d'un périphérique?

- Qu'est-ce qu'une interruption masquée?
- A quoi sert le contrôleur d'interruption programmable de votre ordinateur?
- Combien de sources différentes peuvent interrompre le microprocesseur ?
- Qu'appelle t-on le masquage d'interruption ? Quel registre est concerné ?
- Donnez la procédure pour autoriser l'ensemble des interruptions suivantes : le timer0, la liaison série et le convertisseur A/D.
- Etudier les niveaux de priorités des interruptions ?
- Combien de niveaux de priorités peut-on choisir pour les interruptions ? Comment le niveau est- il sélectionné ?
- Si deux interruptions de niveau de priorité différent arrivent en même temps, c'est l'interruption qui a la plus forte priorité qui s'exécutera en premier.
- Comment fait-on pour départager deux priorités de même niveau qui arrivent en même temps ?
- Qu'est ce qu'un vecteur d'interruption ? Qu'est ce qu'une table des vecteurs d'interruption ?
- Combien de place mémoire possède t on pour coder l'ensemble de ce sous programme d'interruption ? Que convient-il de faire si nous devons utiliser plus de place pour coder ce sous programme ?
- Quelles sont les autres actions à réaliser au début du sous programme, et à la fin du sous programme d'interruption ?
- Pourquoi utilise-t-on des interruptions logicielles plutôt que des appels de fonction pour appeler des routines du système d'exploitation afin d'accéder aux périphériques?
- Quel est l'effet d'une interruption sur le registre PC?
- Y a-t-il une différence entre l'instruction permettant de revenir d'une interruption et celle permettant de revenir d'une fonction? Y a-t-il une différence de comportement du microprocesseur dans ces deux cas?

EXERCICE 02 :

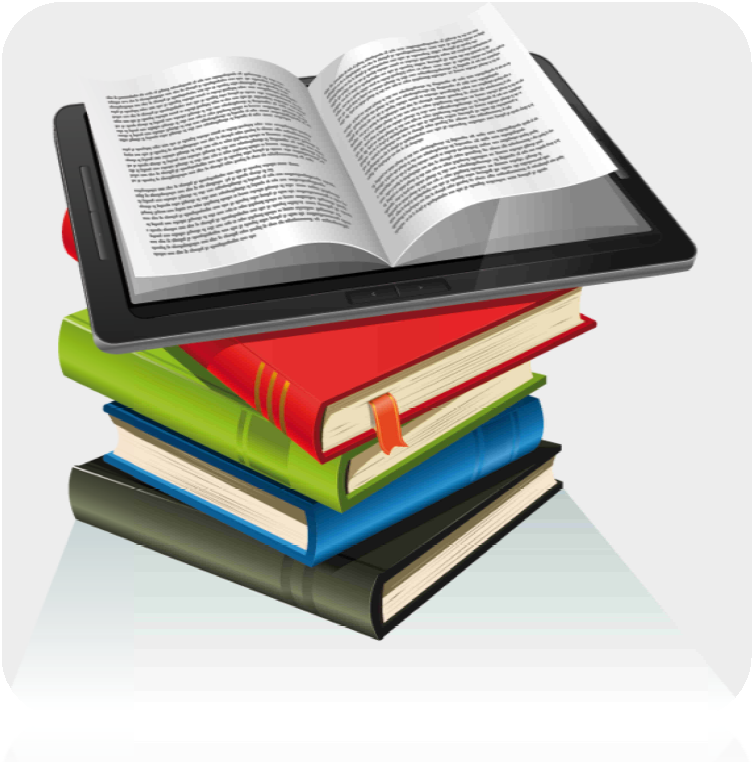
- Etudier le processus de traitement des interruptions par le microprocesseur 8085.
- Soit le montage suivant :



- ✓ Expliquer le rôle de chaque bloc du montage.
- ✓ Etudier en détail l'architecture interne du PIC 8259. Spécifier l'organisation de ces registres internes. Discuter…
- ✓ Mettre sous forme d'organigramme l'interaction entre le up8085 et le PIC 8259.

- ✓ Expliquer le processus de traitement des interruptions effectué par le montage précédent.
- ✓ Etudier le cas de la mise en cascade des PIC :
  - 02 PIC 8259.
  - 03 PIC 8259.
  - 04 PIC 8259.

Les Références Biblio-Webo graphiques



REFERENCES BIBLIO-WEB GRAPHIQUES

Références Bibliographiques

[1]. Letocha ; Introduction aux circuits logiques ; Mc-Graw Hill.
[2]. J.M. Bernard, J. Hugon ; De la logique câblée aux microprocesseurs, Tomes 1 à 4 ; Eyrolles.
[3]. R. Delsol ; Electronique numérique, Tomes 1 et 2 ; Edition Berti.
[4]. P. Cabanis ; Electronique digitale ; Edition Dunod.
[5]. M. Gindre ; Logique séquentielle ; Edition Ediscience.
[6]. J. P. Vabre et J. C. Lafont ; Cours et problèmes d'électronique numérique ; Ellipses, 1998.
[7]. R. Katz ; Contemporary Logic Design ; 2nd ed. ; Prentice Hall, 2005.
[8]. M. Aumiaux ; L'emploi des microprocesseurs ; Masson, Paris, 1982.
[9]. M. Aumiaux ; Les systèmes à microprocesseurs ; Masson, Paris, 1982.
[10]. R.L. Tokheim ; Les microprocesseurs, Tomes 1 et 2 ; série Schaum, McGraw Hill.
[11]. H. Chemmali, Notes de cours, TEC 480, Département d'Electronique, Université de Sétif, 1993-1994.
[12]. J.C. Buisson ; Concevoir son microprocesseur, structure des systèmes logiques ; Ellipses, 2006.
[13]. A. Tanenbaum ; Architecture de l'ordinateur ; Dunod.
[14]. P. Zanella, Y. Ligier, E. Lazard ; Architecture et technologie des ordinateurs ; Dunod.
[15]. J.M. Trio ; Microprocesseurs 8086-8088 : Architecture et programmation, Coprocesseur de calcul 8087, Eyrolles.
[16]. H. Lilen ; Cours fondamental des microprocesseurs ; Dunod, 1993.
[17]. Thomas L. Floyd ; Systèmes numériques : Concepts et applications ; Les éditions Renaold Goulet Inc, 2000.
[18]. Jean-Claude Lafond, Jean Paul Vabre ; Cours et problèmes d'électronique numérique : 124 exercices avec solutions ; Edition Marketing, 1986.
[19]. J.C. Buisson ; Concevoir son microprocesseur : Structure des systèmes logiques ; Ellipses, 2006.

Références Webgraphiques

[20]. http://www.info.univ-angers.fr/~richer/ensl3i_crs4.php
[21]. http://ambroise.brou1.free.fr/en_007.htm .
[22]. http://www.courstechinfo.be/Hard/Memoire.html
[23]. http://sti.ac-orleans-tours.fr/spip2/IMG/pdf/Memoires_complet.pdf
[24]. http://www.les-electroniciens.com/sites/default/files/cours/sam1a_coursv11.pdf
[25]. http://pf-mh.uvt.rnu.tn/320/1/Microprocesseurs_et_Microcontr%C3%B4leurs.pdf
[26]. http://infomaroon.blogspot.com/2008/04/composition-dun-processeur.html
[27]. http://www.zseries.in/embedded%20lab/8085%20microprocessor/memory%20mapping.php#.VzBsa6zmodU
[28]. http://deptinfo.unice.fr/twiki/pub/Minfo05/ApprofondissementSysteme/11_GestionEntreeSorties.pdf
[29]. http://www.eyrolles.com/Chapitres/9782212116717/chap5_Lange.pdf
[30]. http://infotroniquedz.ble.fr/Files/6_sm5_archi_ch5_interruptions.pdf
[31]. https://www.fichier-pdf.fr/2012/10/11/cours-micro/cours-micro.pdf
[32]. http://www.8085projects.info/Interfacing-of-PIC-8259-with-8085.html

Feuille des remarques – Cours -

ENSEIGNANT : ______________________ **SALLE :** ________

Remarques	Page
………………………………………………………………………	………
………………………………………………………………………	………
………………………………………………………………………	………
………………………………………………………………………	………
………………………………………………………………………	………
………………………………………………………………………	………
………………………………………………………………………	………
………………………………………………………………………	………
………………………………………………………………………	………
………………………………………………………………………	………
………………………………………………………………………	………
………………………………………………………………………	………
………………………………………………………………………	………
………………………………………………………………………	………
………………………………………………………………………	………
………………………………………………………………………	………
………………………………………………………………………	………
	………
………………………………………………………………………	………
………………………………………………………………………	………
………………………………………………………………………	………
………………………………………………………………………	………
………………………………………………………………………	………
………………………………………………………………………	………
………………………………………………………………………	………
………………………………………………………………………	………
………………………………………………………………………	………
………………………………………………………………………	………
………………………………………………………………………	………
………………………………………………………………………	………
………………………………………………………………………	………
………………………………………………………………………	………
………………………………………………………………………	………
………………………………………………………………………	………

Feuille des remarques – TD -

ENSEIGNANT : ______________________ **SALLE :** ________

Remarques	Page
....	....
....	....
....	....
....	....
....	....
....	....
....	....
....	....
....	....
....	....
....	....
....	....
....	....
....	....
....	....
....	....
....	....
	....
....	....
....	....
....	....
....	....
....	....
....	....
....	....
....	....
....	....
....	....
....	....
....	....
....	....
....	....
....	....