République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITE MUSTAPHA STAMBOULI DE MASCARA

FACULTÉ DES SCIENCES ET TECHNOLOGIE

Polycopié de Cours

Calculateurs et interfaçage

Ce cours est destiné aux étudiants des

Sciences et Technologie /L3 Télécommunications

2020

Avant Propos

Ce cours est dédié aux étudiants de la troisième année licence télécommunications inscris dans le domaine des sciences technologiques. Il permet aux étudiants de comprendre l'architecture des microprocesseurs (μP), leur fonctionnement, les périphériques qui peuvent être associés au (μP) et les différents interfaçages en relation. Ils leurs permet aussi de se familiariser avec plusieurs types de calculateurs. Une fois les connaissances de base assimilées, le (μP) 8086 d'Intel est pris comme exemple simple de calculateurs. Une étude et programmation détaillées sont données avec la manière dans laquelle il s'interface avec les modules externes tels que l'interface parallèle 8255 et l'interface série 8250.

A partir des notions de base présentées par ce cours, l'étudiant sera capable de réaliser des travaux pratiques permettant de programmer en langage machine (assembleur) une application d'acquisition des données.

Ce module est enseigné en temps que cours et travaux pratiques (TP) intégrés en semestre 5 pour une charge horaire totale de 37h30mn soit une séance de cours de 1h30mn et une séance de TP de 1 heure.

Table des matières

Chapitre VI Principes de l'implémentation d'un système logique synchrone par un circuit programmable

Liste des figures

Liste des tables

Chapitre I

Architecture d'un microprocesseur

Chapitre I. Architecture d'un microprocesseur

I-1-Introduction :

Le développement de l'électronique numérique a suscité l'apparition de plusieurs types de composants très puissants en particulier les systèmes micro-programmés.

Le développement de ces composants programmables remplace de plus en plus l'électronique classique vu que les circuits intégrés analogiques ou logiques ne peuvent plus résoudre des fonctions de plus en plus complexes.

Historiquement, les constructeurs (Fig. I.1) développèrent d'abord les systèmes micro-programmés intégrés dans les calculateurs de bureau ou de portable, avec des codes d'ordre orientés vers le calcul numérique. Puis, maitrisant cette technique ils offrirent des circuits d'usage généraux : ***Les microprocesseurs.*** Le 4004 d'Intel à 4 bits fut le premier microprocesseur à voir le jour en 1971 (Table I.1). La miniaturisation des transistors a permis d'augmenter considérablement la capacité d'intégration sur silicium.

Table I.1 : Évolution des microprocesseurs.

Date	Nom	Nombre de transistors	Finesse de gravure (nm)	Fréquence de l'horloge	Largeur des données	MIPS
1971	Intel 4004	2 300		108 kHz	4 bits/4 bits bus	0,06
1974	Intel 8008	6 000	6 000	2 MHz	8 bits/8 bits bus	0,64
1979	Intel 8088	29 000	3 000	5 MHz	16 bits/8 bits bus	0,33
1982	Intel 80286	134 000	1 500	6 à 16 MHz (20 MHz chez AMD)	16 bits/16 bits bus	1
1985	Intel 80386	275 000	1 500	16 à 40 MHz	32 bits/32 bits bus	5
1989	Intel 80486	1 200 000	1 000	16 à 100 MHz	32 bits/32 bits bus	20
1993	Pentium (Intel P5)	3 100 000	800 à 250	60 à 233 MHz	32 bits/64 bits bus	100
1997	Pentium II	7 500 000	350 à 250	233 à 450 MHz	32 bits/64 bits bus	300
1999	Pentium III	9 500 000	250 à 130	450 à 1 400 MHz	32 bits/64 bits bus	510
2000	Pentium 4	42 000 000	180 à 65	1,3 à 3,8 GHz	32 bits/64 bits bus	1 700
2004	Pentium 4 D (Prescott)	125 000 000	90 à 65	2.66 à 3,6 GHz	32 bits/64 bits bus	9 000
2006	Core 2 Duo (Conroe)	291 000 000	65	2,4 GHz (E6600)	64 bits/64 bits bus	22 000
2007	Core 2 Quad (Kentsfield)	2*291 000 000	65	3 GHz (Q6850)	64 bits/64 bits bus	2*22 000 (?)
2008	Core 2 Duo (Wolfdale)	410 000 000	45	3,33 GHz (E8600)	64 bits/64 bits bus	~24 200
2008	Core 2 Quad (Yorkfield)	2*410 000 000	45	3,2 GHz (QX9770)	64 bits/64 bits bus	~2*24 200
2008	Intel Core i7 (Bloomfield)	731 000 000	45	3,33 GHz (Core i7 975X)	64 bits/64 bits bus	?
2009	Intel Core i5/i7 (Lynnfield)	774 000 000	45	3 06 GHz (I7 880)	64 bits/64 bits bus	76383
2010	Intel Core i7 (Gulftown)	1 170 000 000	32	3,47 GHz (Core i7 990X)	64 bits/64 bits bus	147600
2011	(Sandy Bridge)		32			
2012	Intel Core i3/i5/i7 (Ivy Bridge)		22			

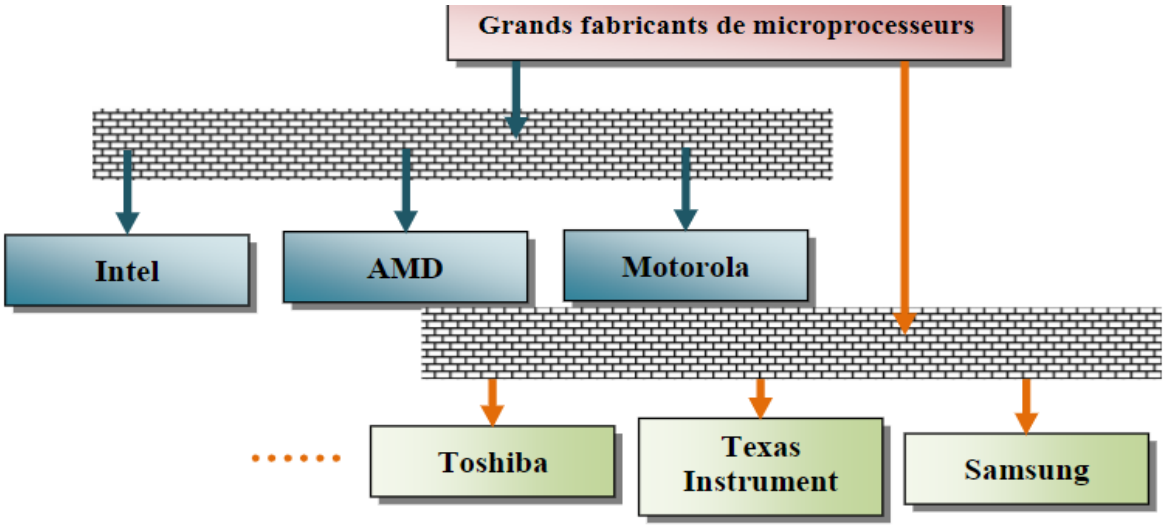


Figure I.1. Les grands fabricants de CPU

I-2-Introduction aux systèmes à base d'un microprocesseur :

I-2-1- Définition d'un microprocesseur :

Le microprocesseur (cerveau de l'ordinateur !), noté aussi **M.P.U. (Microprocessor unit)** ou encore **CPU (Central Processing Unit)** est un circuit intégré complexe à très grande échelle d'intégration **(VLSI: Very Large Scale Integration),** capable d'effectuer séquentiellement et automatiquement des suites d'opérations élémentaires.

Il remplit deux fonctions essentielles :

1. **Le traitement des données.** On parle d'unité de traitement. Cette fonction est dédiée à l'Unité Arithmétique et Logique (UAL ou ALU). Elle concerne la manipulation des données sous formes de procédures de transfert, d'opérations arithmétiques, d'opérations logiques,....
2. **Le contrôle du système.** Cette fonction se traduit par des opérations de décodage et d'exécution des ordres exprimés sous forme d'instructions.

I-2-2- Systèmes à base d'un microprocesseur :

Un système à base de microprocesseur est formé des trois éléments :

1. Une unité CPU (Central Processing Unit).
2. Une mémoire (ROM et RAM).
3. Des ports d'Entrées/Sorties.

Les trois modules sont interconnectés comme le montre la figure suivante (Fig. I.2) autour de trois bus :

1. **Bus de données :** C'est un ensemble de fils bidirectionnels (à double sens) qui va permettre le transfert de données entre les différents éléments du système.

2. **Bus d'adresses :** C'est un ensemble de fils unidirectionnel (dans un seul sens) qui permet d'adresser un élément par le microprocesseur.
3. **Bus de contrôles et de commandes :** C'est un bus qui permet de véhiculer les signaux de contrôles et de commandes tels que l'horloge, les signaux Rd/Wr (Read/Write : lecture/écriture),...

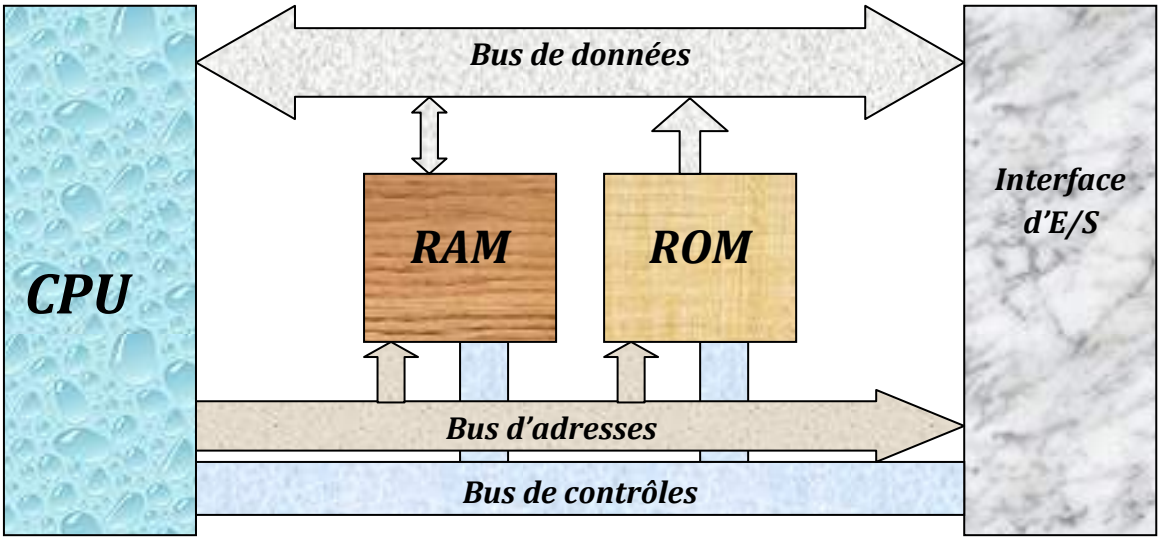
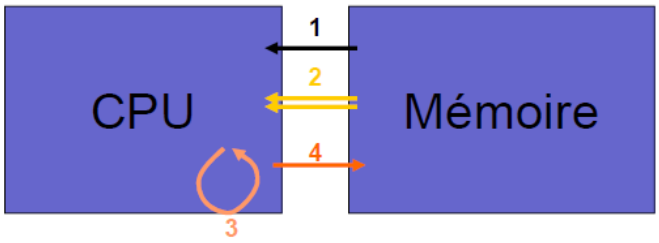


Figure I.2. Système à base de microprocesseur.

I-3- Traitement des instructions :

Le schéma ci-dessous montre un résumé de traitement des instructions d'un système à base du microprocesseur.



(1)- Charger une instruction depuis la mémoire.

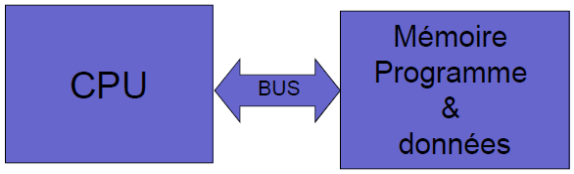
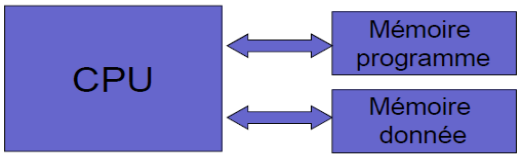
(2)- Charger les opérandes (données à traiter) depuis la mémoire.

(3)- Effectuer les calculs.

(4)- Stocker le résultat en mémoire.

Notant que ce schéma représente le modèle ou l'architecture de **Von Neumann** qui utilise une structure de stockage unique pour conserver à la fois les instructions et les données. Cette architecture s'oppose à celle de **Harvard** qui sépare physiquement la mémoire de données et la mémoire programme. Les différences majeures entre ces 2 technologies sont citées sur le tableau suivant :

Table I.2 : Différences entre les Architectures Von Neumann et Harvard

Modèle 1 : Von Neumann	**Modèle 2 : Harvard**
 ➢ Un seul chemin d'accès à la mémoire. • Un bus de données (programme et données). • Un bus d'adresse (programme et données).	 ➢ Séparation des mémoires programme et données • Un bus de données programme, • Un bus de données pour les données, • Un bus d'adresse programme, • Un bus d'adresse pour les données

I-4- Architecture externe d'un microprocesseur (8086) :

Le microprocesseur 8086 (Fig. I.3) qui est à la base des processeurs de la famille Intel 80×86 (8086, 80186, 80286, 80386, 80486, Pentium,…) est un circuit intégré de 40 pattes dans un boitier de forme DIL (Dual In Line : boitier de deux ligne parallèles) nommée aussi DIP (Dual In Line Package). Il est équipé d'un bus de données de 16 bits (AD0 : AD15) et un bus d'adresses de 20 bits (AD0 : AD15 & A16 : A19). Les pins ou broches AD sont associés à des bus d'adresses/données multiplexés. Selon plusieurs variantes, le µP 8086 fonctionne à des fréquences différentes (5, 8, 10,…) MHz.

Description des broches

- Vcc et GND : assurent l'alimentation électrique du microprocesseur.

- CLK: entrée destinée à recevoir le signal de l'horloge système, qui cadence le fonctionnement du microprocesseur.

-READY : permet la synchronisation des mémoires et périphériques lents avec le CPU 8086.

-RESET: un signal, de remise à l'état initial.

-$MN/\overline{MX}$: sélectionne entre l'un des deux modes de fonctionnement du 8086.

-Mode minimum: le CPU 8086 fonctionne d'une manière autonome et en monoprocesseur et il génère par lui même les signaux de bus de commande.

-Mode maximum : Ces signaux sont produit par un contrôleur de bus 8288, ce qui lui permet d'opérer dans un environnement multiprocesseur.

-$\overline{TEST}$: entrée de synchronisation.

-NMI : (Non Masquable Interrupt) : entrée de demande d'interruption prioritaire.

- INTR: entrée de demande d'interruption normale.

-$\overline{INTA}$: (INTerrupt Acknowledge) : indique que le microprocesseur a pris en compte l'interruption.

-HOLD: entrée de demande d'accès au bus.

-HLDA: indique que le microprocesseur a pris en compte la demande d'accès au bus.

-S0 à S7: signaux d'état indiquant le type d'opération sur le bus.

-A16/S3 à A19/S6: 4 bits de poids fort du bus d'adresses, multiplexés avec 4 bits d'état.

-AD0 à AD15: 16 bits de poids faible du bus d'adresses, multiplexés avec 16 bits de données, d'où la nécessité d'un multiplexage pour obtenir séparément les bits d'adresse et de données.

-$\overline{RD}$ (Read): signal de lecture d'une donnée.

-$\overline{WR}$ (Write): signal d'écriture d'une donnée.

-M/$\overline{IO}$: (Memory/Input/Output): le CPU 8086 adresse la mémoire ou les unités d'entrée/sortie.

-$\overline{DEN}$:(Data ENable): indique que la donnée est disponible sur le bus de données.

-ALE : (Adress Latch Enable) : indique que l'adresse est disponible sur le bus d'adresses.

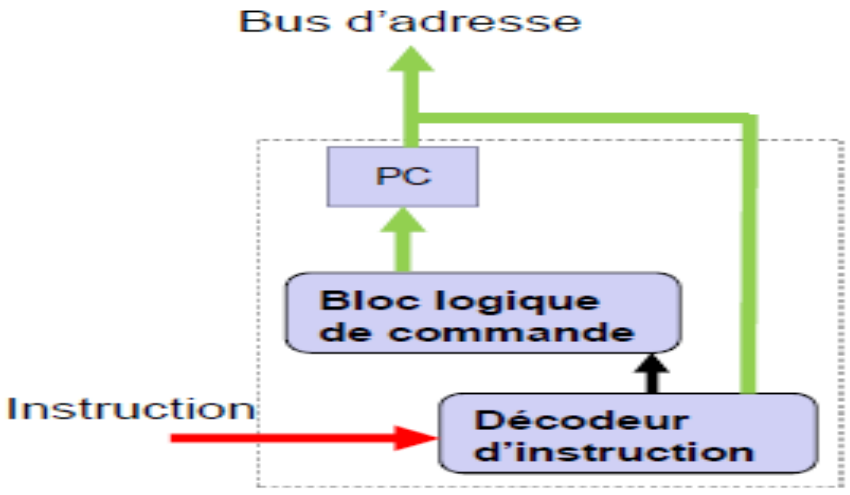
-DT/$\overline{R}$: (Data Transmit/ Receive): indique le sens de transfert des données.

-$\overline{BHE}$: (Bus High Enable): signal de validation de l'octet du poids fort du bus de données.

I-5- Unités principales du microprocesseur :

Un microprocesseur est construit autour de deux unités principales (Fig. I.4) : Une unité de commande et une unité de traitement.

I-5-1 Unité de commande: Elle permet de séquencer le déroulement des instructions en manipulant 3 parties :



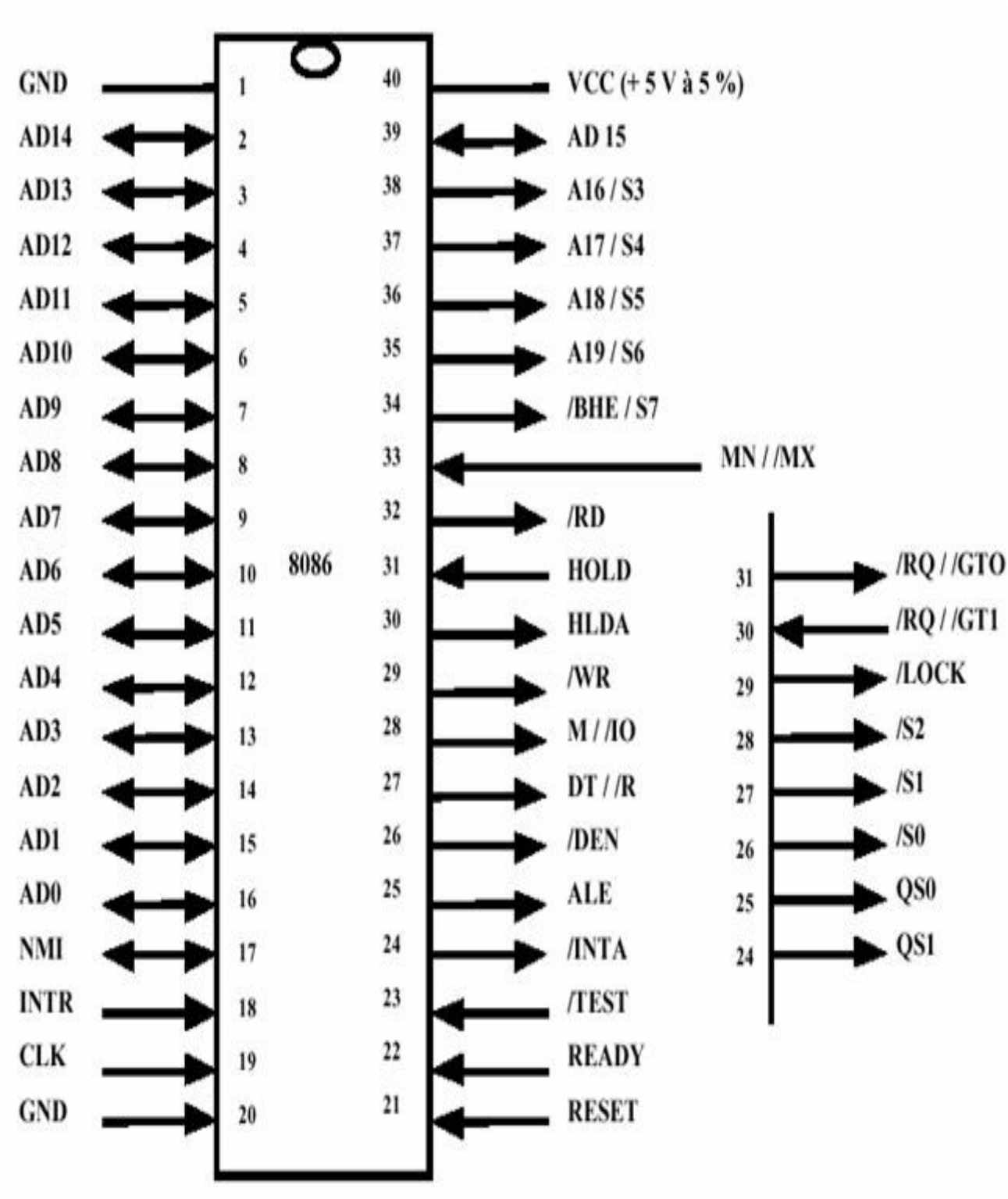


Figure I.3. Brochage du CPU 8086.

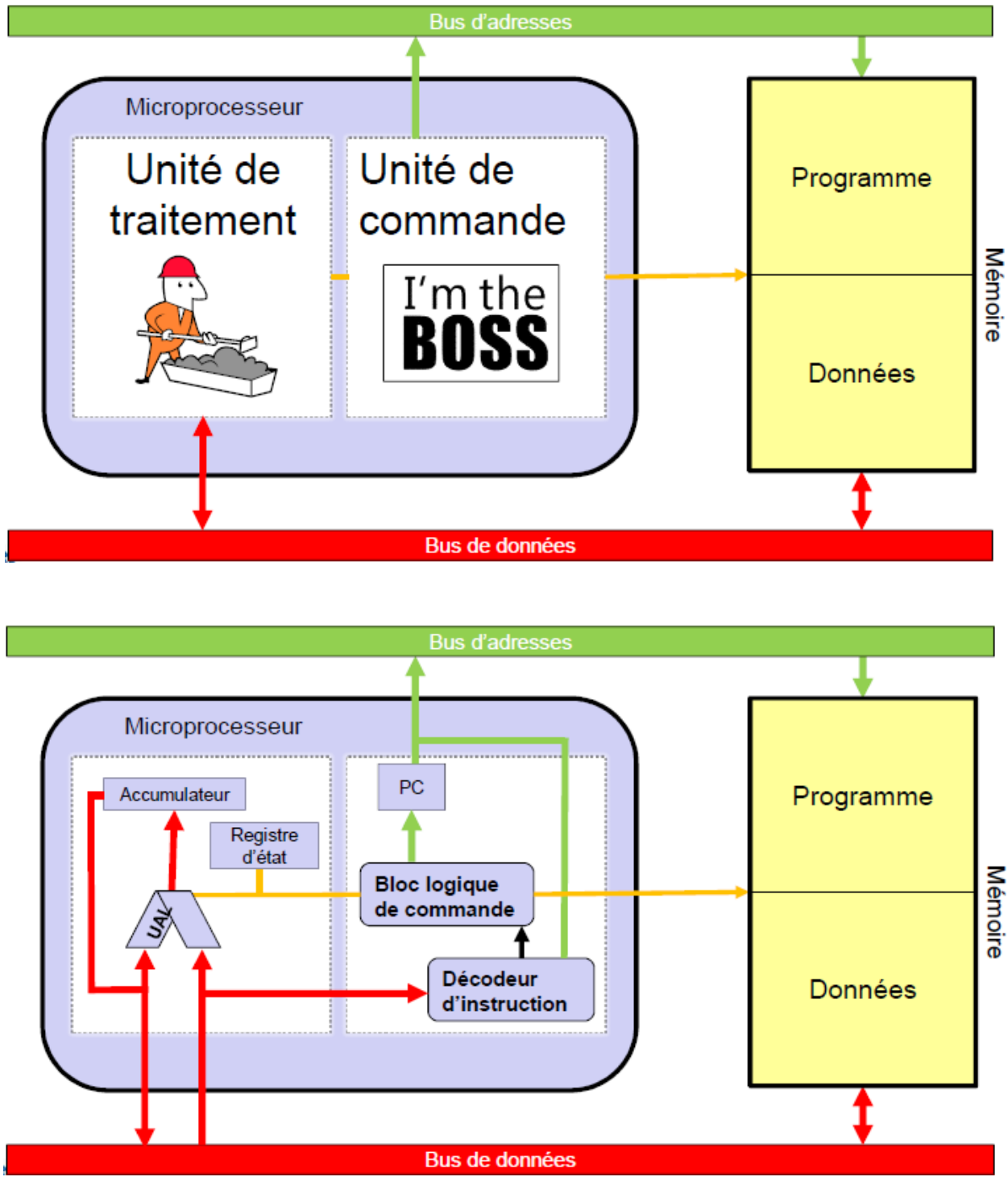


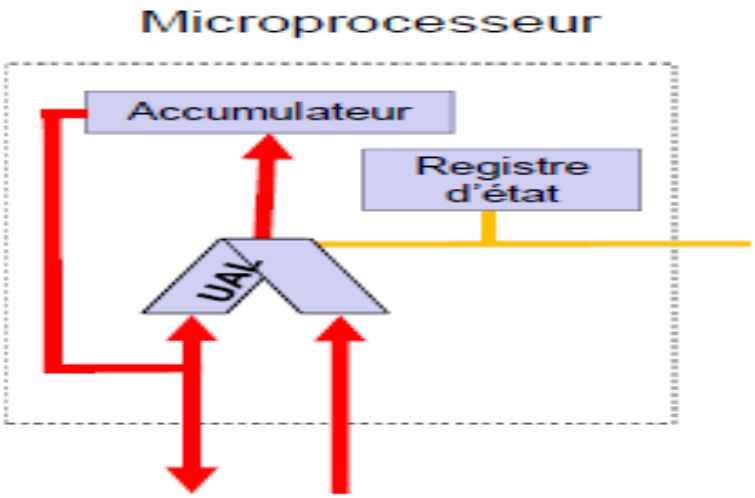
Figure I.4. Architecture interne d'un microprocesseur.

a- PC : (Programme Counter), appelé aussi Compteur Ordinal (Le compteur de programme). Il comporte un registre (défini dans la partie I.7) dont le contenu est initialisé avec l'adresse de la première instruction du programme, il contient toujours l'adresse de la prochaine instruction à exécuter.

b- Le décodeur d'instruction : Le mot binaire (instruction) doit être décodé pour savoir à quelle action correspond l'instruction. Chacune des instructions à exécuter est transférée depuis la mémoire dans le registre d'instruction puis est décodée par le registre décodeur qui génère les signaux logiques correspondants pour les communiquer au séquenceur.

c- Bloc logique de commande :(**Séquenceur**) Il organise l'exécution des instructions au rythme de l'horloge. Il élabore tous les signaux de synchronisation du microprocesseur en fonction de l'instruction qu'il va exécuter.

I-5-2 Unité de traitement: Elle regroupe les circuits qui assurent les traitements nécessaires à l'exécution des instructions :



a- Accumulateur : C'est un registre de travail qui sert à stocker le résultat des opérations réalisées par L'UAL.

b- Unité Arithmétique et Logique (UAL) : est un circuit complexe qui assure les fonctions logiques (ET, OU, comparaison, décalage, etc.) et arithmétique (addition, soustraction, etc.).

c- Registre d'état : Chacun des bits de ce registre dépend du résultat de la dernière opération effectuée par l'UAL. Exemple : Bit de retenue (carry : C), débordement (overflow : OV ou V), Zéro (Z) ...

Remarque : Il y a des registres d'adresse servi d'interface entre le bus des données internes et le bus d'adresse pour les deux unités.

I-6- Architecture interne du microprocesseur 8086:

On peut distinguer deux unités internes qui fonctionnent en parallèle:

1- **Unité d'Exécution –UE- (EU : Execution Unit) :** exécute les instructions qui lui sont transmises par l'UIB.

2- **Unité d'Interface de Bus –UIB- (BIU : Bus Interface Unit)** : Le rôle de cette unité est de récupérer et stocker les informations à traiter, et d'établir les transmissions avec les bus du système.

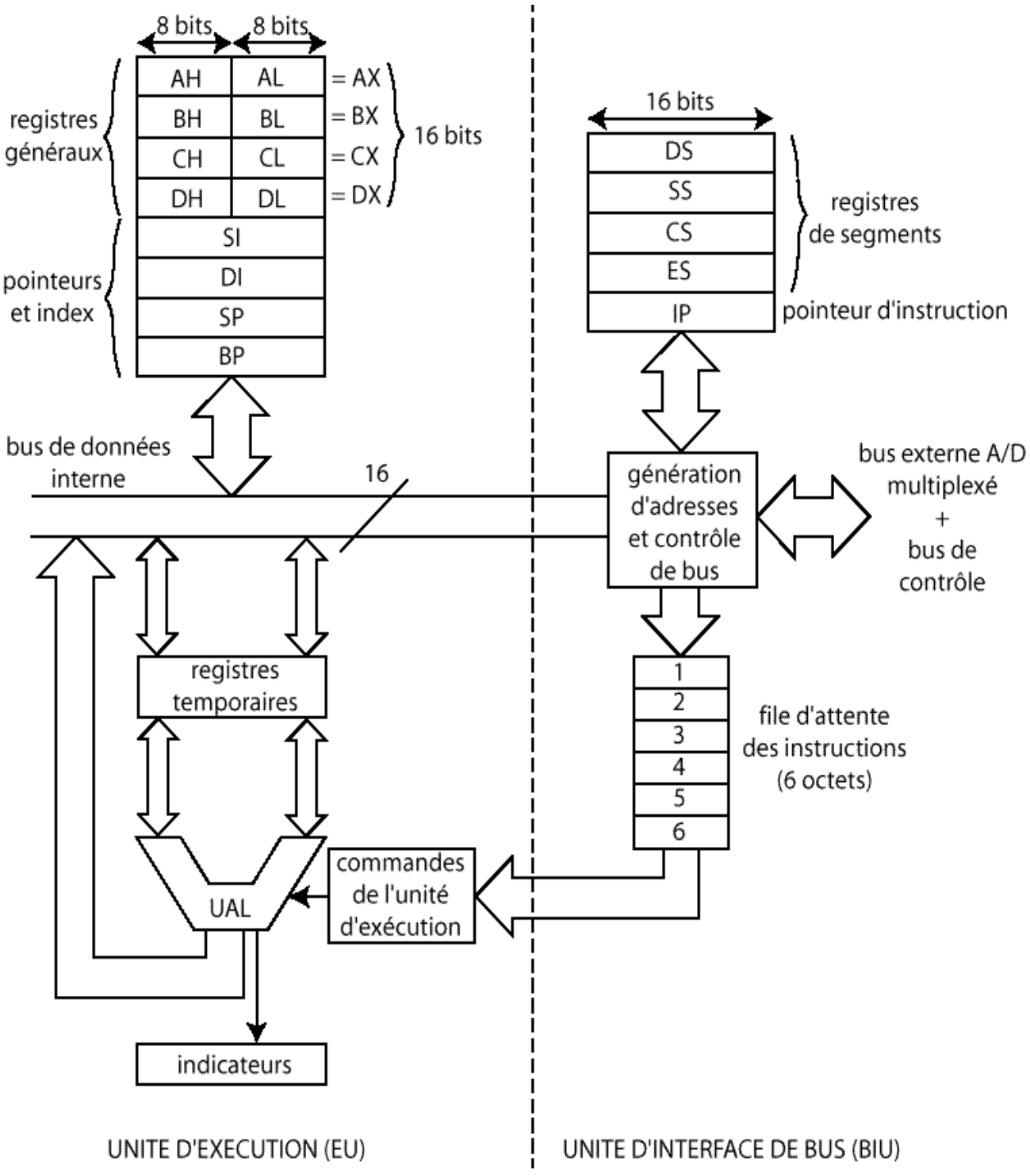


Figure I.5. Architecture interne du microprocesseur 8086

Remarque : Les deux unités fonctionnent simultanément, ainsi, pendant que l'unité d'exécution traite une instruction, la BIU recherche dans la mémoire les octets composant la prochaine instruction à exécuter. ***Ce mode de fonctionnement, dit pipeline****, accélère le processus d'exécution d'un programme, en réduisant le temps mort de l'exploitation du système de bus.*

I-7- Registres du microprocesseur 8086:

I-7-1-Définition :

Un registre est un ensemble de mémoires élémentaires (bascules- chacune permet de mémoriser un seul bit) qui servent à enregistrer ou à modifier des combinaisons binaires appelées mots ou mots binaires. On distingue deux catégories de registres: Les registres de mémoire ou enregistrements et les registres à décalage.

Exemple : un registre de 4 bits contient quatre bascules comme le montre la figure suivante :

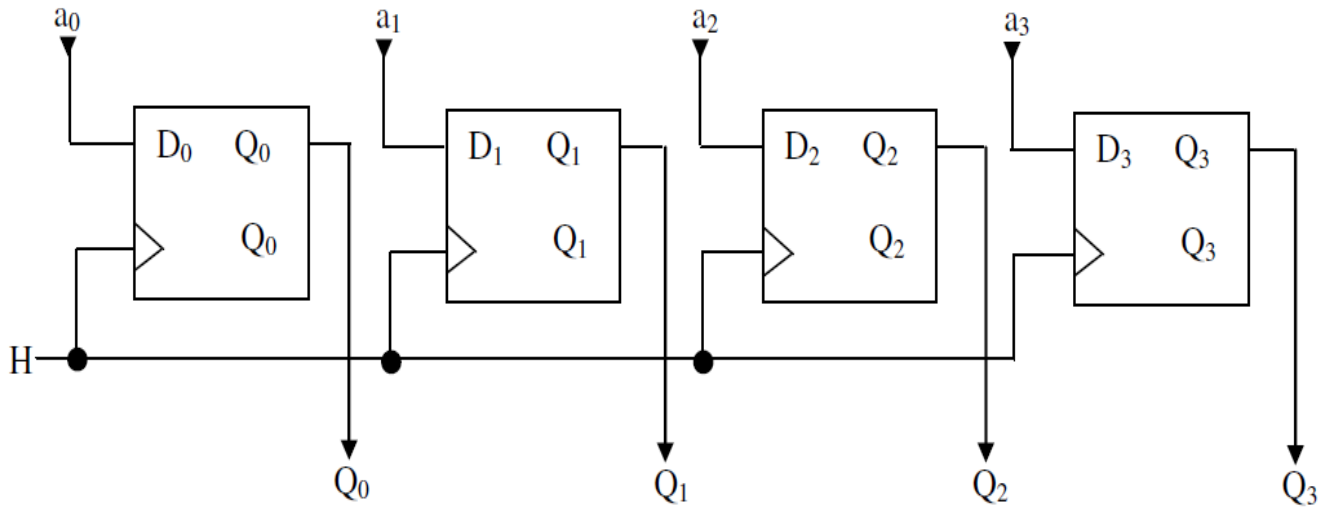


Figure I.6. Registre de 4 bits

I-7-2- Types des registres du CPU 8086:

Le microprocesseur (CPU) 8086 contient 14 registres répartis en 4 groupes :

1- **Registres généraux** : 4 registres sur 16 bits.

Ils peuvent être également subdivisés en deux registres de 8 bits supérieurs et inférieurs, et être référencés sous ***x*H** et ***x*L:** par exemple AH et AL.

AX =(**AH**,**AL**) : registre Accumulateur. (AH: contient l'octet du registre AX le plus fort : High)

BX =(**BH**,**BL**) : registre de Base. (BL : contient l'octet du registre BX le plus bas : Low).

CX =(**CH**,**CL**) : registre Compteur.

DX =(**DH**,**DL**) : registre des données.

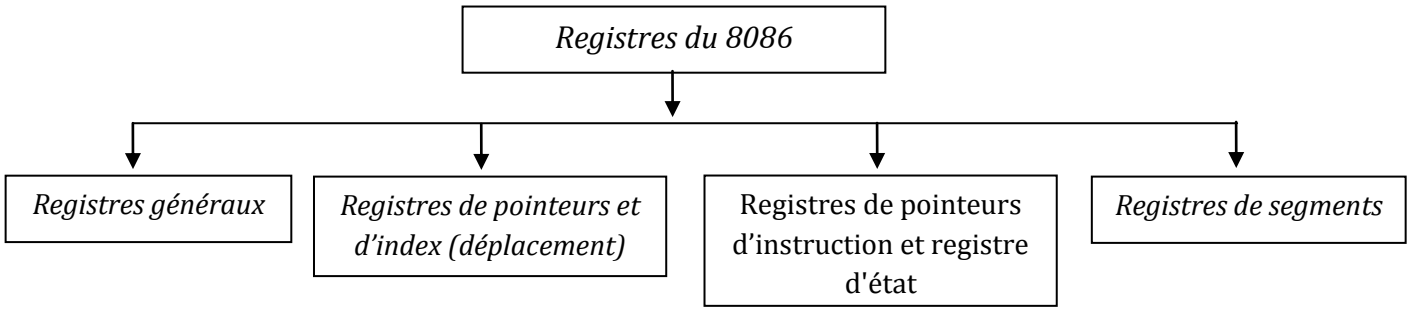


Figure I.7. Différents registres du CPU 8086

Ils servent à contenir temporairement des données. Ce sont des registres généraux mais ils peuvent être utilisés pour des opérations particulières.

AX comme accumulateur ;

BX comme registre de base pour l'adressage basé ;

CX comme compteur ;

DX Sert aussi pour quelques opérations mathématiques.

2- Registres de pointeurs et d'index (déplacement): 4 registres sur 16 bits.

a) Index :

- SI : Source Index : permet de pointer la mémoire et est souvent utilisé comme index de l'opérande source dans certaines instructions de déplacement.

- DI : Destination Index : permet aussi de pointer la mémoire et sur la destination pour les instructions de chaine de caratères.

b) Pointeurs :

- SP : (Stack Pointer), pointeur de pile, pointe sur le sommet de la pile (*La pile est une partie de la mémoire allouée par le système d'exploitation pour l'exécution d'une tâche*).

- BP : (Base Pointer), pointeur de base, pointe sur l'adresse de base de la pile.

Ils sont utilisés pour les transferts de chaînes d'octets entre deux zones mémoire.

Les registres pointeurs et d'index contiennent des adresses de cases mémoire et sont appelés aussi registre de déplacement.

3- Registre pointeur d'instruction et registre d'état : 2 Registre sur 16 bits.

a) Registre pointeur d'instruction (**IP** : Instruction Pointer): contient l'adresse de la prochaine instruction à exécuter.

b) Registre d'état (**PSW** : Program Status Word): la valeur représentée par ce nombre de 16 bits n'a aucune signification en tant qu'ensemble: ce registre est manipulé bit par bit. Il offre 16 bits dont seulement 9 sont utilisés.

R	R	R	R	O	D	I	T	S	Z	U	A	U	P	U	C
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

O,S,Z,A,P,C	*Indicateurs d'état*
D,I,T	*Indicateurs de contrôle*
R	*Bits réservé*
U	*Bits indéfini*

Figure I.8: Registre d'état (PSW)

On peut les regrouper en deux catégories:

✓ **Indicateurs d'état:**

CF : (Carry Flag), indicateur de retenue. Mis à 1 si un calcul produit une retenue ;

```
    10010110          11011001
+   01010100      +   01010010
CF=0 11101010     CF=1 00101011
```

PF : (Parity Flag), indicateur de parité. Mis à 1 si les 4 bits de poids faible du résultat contiennent un ;

AF : (Auxiliary Flag), indicateur de retenue auxiliaire. Mis à 1 si un calcul en BCD non compacté produit une retenue. ;

ZF : (Zero Flag), indicateur de zéro. Cet indicateur est positionné à 1 quand le résultat d'une opération est égala zéro.

SF : (Signe Flag), indicateur de signe. Prend la valeur du bit de poids fort de l'accumulateur après un calcul.

```
    10010110          11011001
+   01010100      +   01010010
SF=1 11101010     SF=0 00101011
```

OF : (Overflow Flag), indicateur de dépassement. Mis à 1 si une opération provoque un dépassement de capacité.

✓ **Indicateurs de contrôle:**

TF : (Trap Flag), indicateur d'exécution pas à pas. Mis à 1, il force le processeur à fonctionner pas à pas.

IF : (Interruption Flag), indicateur d'autorisation d'interruption. Mis à 1, il autorise les interruptions. S'il vaut 0, il les empêche.

DF : (Direction Flag), indicateur de direction. Fixe le sens (incrémentation et décrémentation) dans lequel seront effectuées les opérations de traitement de chaînes de caractères.

R : bit réservé ;

U : bit indéfini ;

4- Registres de segments : 4 registres sur 16 bits.

Les registres de segments, associés aux pointeurs et aux index, permettent au CPU 8086 d'adresser l'ensemble de la mémoire.

a) CS : (Code Segment) Segment de code: contient les instructions du programme et est associé au pointeur d'instruction IP. Il pointe sur le début du segment qui contient les codes des instructions du programme. Ainsi la prochaine instruction à exécuter se trouve à l'adresse logique CS:IP.

b) DS : (Data Segment) Segment de données : contient les données manipulées par le programme. Son adresse se trouve dans le registre de DS,

- Exemple : DS : SI (DS ↔SI).

c) ES : (Extra Segment) Segment supplémentaire : peut contenir des données. Les registres de segments DS et ES peuvent être associés (liés) à un registre d'index.

- Exemple : ES : DI (ES ↔ DI).

d) SS : (Stack Segment) Segment de pile : contient la pile de sauvegarde. Le registre SS peut être associé aux registres de pointeur de pile.

- Exemple : SS:SP (SS ↔ SP). SS:BP (SS ↔ BP).

I-8- Gestion de la mémoire par le CPU 8086 :

L'espace mémoire adressable par le 8086 est de 1 Mo. Le pointeur d'instruction fait 16 bits donc il y a possibilité d'adresser 2^{16} = 64 Ko (ce qui ne couvre pas la mémoire). La stratégie de gestion de la mémoire avec ce CPU est un peu particulière. Elle consiste à diviser l'espace mémoire adressable par le CPU 8086 en **segments**.

Définition du segment : est une zone mémoire de 64 Ko définie par son adresse de départ qui doit être un multiple de 16.

Pour désigner une case mémoire parmi les 2^{16} contenues dans un segment, il suffit d'une valeur sur 16 bits.

Ainsi, une case mémoire est repérée par le CPU 8086 au moyen de deux quantités sur 16 bits :

1) L'adresse d'un segment ;

2) Un déplacement ou **offset** (appelé aussi **adresse effective**) dans ce segment.

Cette méthode de gestion de la mémoire est appelée **segmentation de la mémoire**.

La donnée d'un couple (segment, offset) définit une **adresse logique**, notée sous la forme

(segment : offset)

L'adresse d'une case mémoire donnée sous la forme d'une quantité sur 20 bits (5 digits hexa) est appelée **adresse physique**, car elle correspond à la valeur envoyée réellement sur le bus d'adresses A0 - A19.

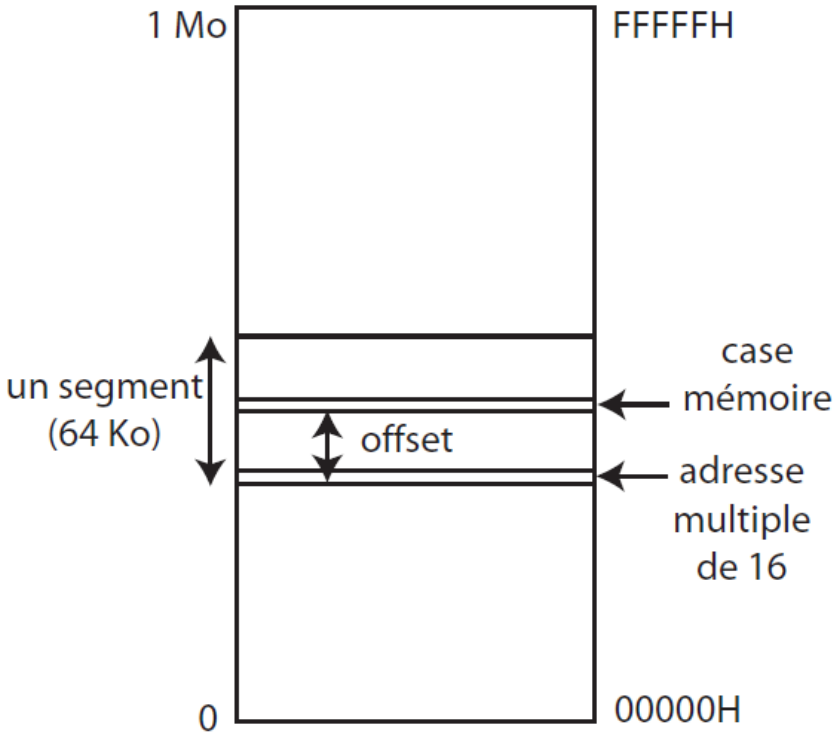
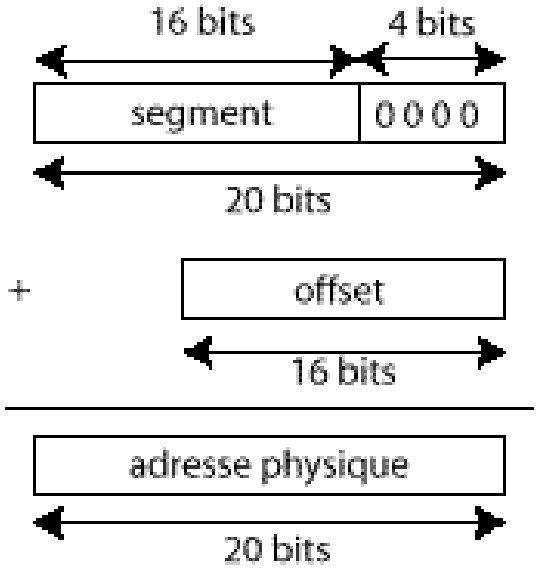


Figure I.9. Segmentation de la mémoire

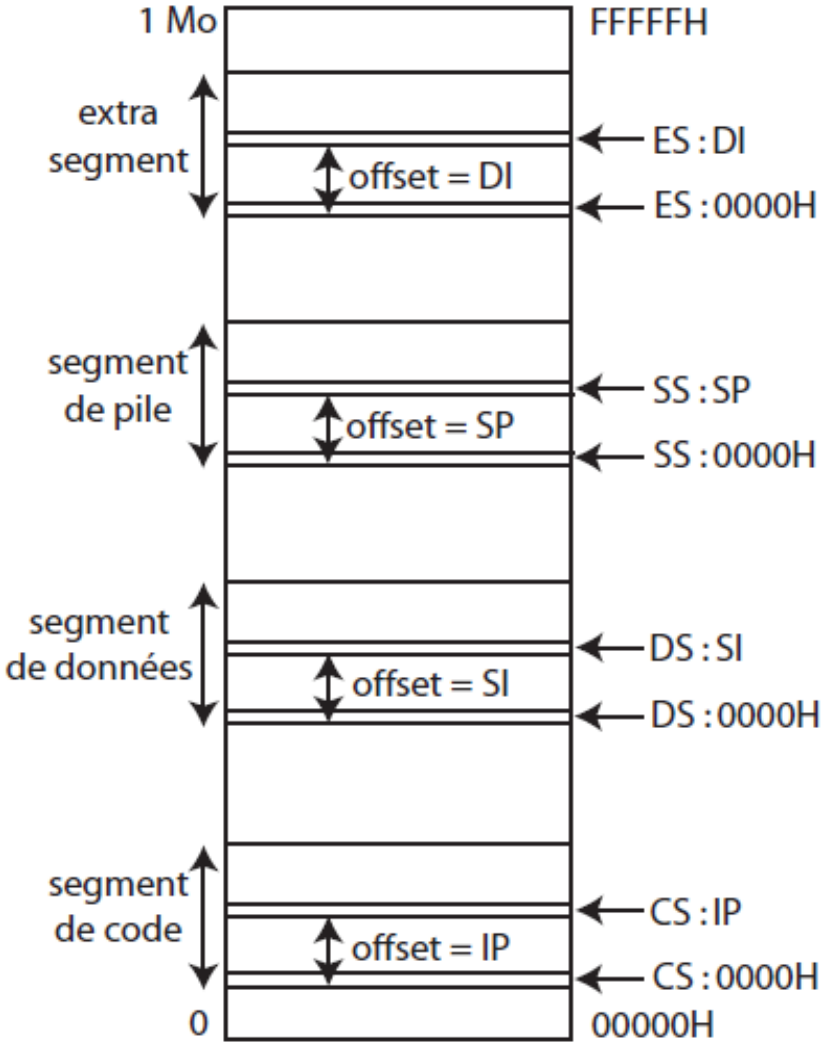
L'adresse physique se calcule par l'expression :

Adresse physique = 16 × segment + offset

On a la multiplication par 16 = 2^4 revient à effectuer un décalage de 4 positions vers la gauche donc le calcule s'effectue comme suit :



Le CPU 8086 peut accéder à 4 types de segments, dont leur adresse se trouve dans les registres de segment:



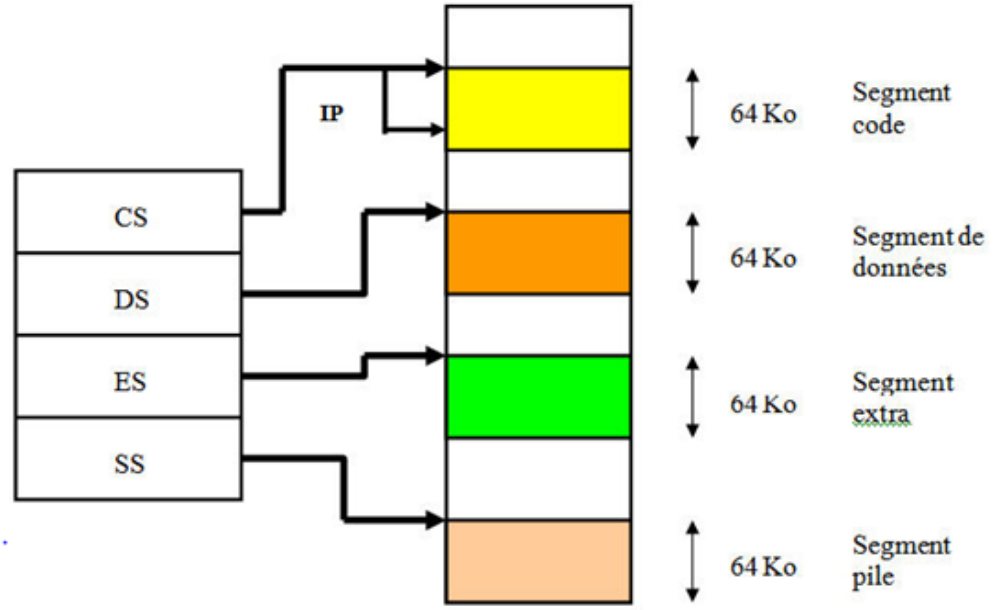


Figure I.10. Types des segments mémoire

Chapitre II

Les interfaces d’entrées-sorties

Chapitre II. Les interfaces d'entrées-sorties

II-1- Introduction :

On appelle entrées-sorties les échanges d'informations entre le processeur et les périphériques qui lui sont associés. De la sorte, le système peut réagir à des modifications de son environnement, voire le contrôler. Elles sont parfois désignées par l'acronyme I/O (l'anglais Input/Output) ou encore E/S pour (Entrées/Sorties).

- Les entrées sont les données envoyées par un périphérique (disque, réseau, clavier…) à destination de l'unité centrale.
- Les sorties sont les données émises par l'unité centrale à destination d'un périphérique (disque, réseau, écran…).

II-2- Définition:

Une ***interface d'entrées/sorties*** est un circuit intégré permettant au microprocesseur de communiquer avec l'environnement extérieur (périphériques) : clavier, écran, imprimante, modem, disques, processus industriel, ...

Les interfaces d'E/S sont connectées au microprocesseur à travers les bus d'adresses, de données et de commandes (Figs. II.1 et II.2).

Les points d'accès aux interfaces sont appelés ports.

Exemple :

interface	port	exemple de périphérique
interface parallèle	port parallèle	imprimante
interface série	port série	modem

II-3- Types d'interfaces :

II-3-1- L'interface parallèle (ex. 8255A):

(Interface périphérique programmable 8255A (Programmable Peripheral Interface : PPI)). Le rôle d'une interface parallèle 8255A est de transférer des données du microprocesseur vers des périphériques et inversement, tous les bits de données étant envoyés ou reçus simultanément.

Le 8255A est une interface parallèle programmable : peut être configurée en entrée et/ou en sortie par programme. C'est un circuit flexible et économique (cas de besoin de plusieurs ports). Le 8255A possède trois ports, qui sont utilisés pour le transfert bidirectionnel des données (Fig. II.3).

II-3-2- L'Interface série :

a. Principe d'une interface série :

Une interface série (Fig. II.4) permet d'échanger des données entre le microprocesseur et un périphérique bit par bit.

Exemples d'interfaces séries : • 8251 (Intel) ; • 8250 (National Semiconductor) ; • 6850 (Motorola).

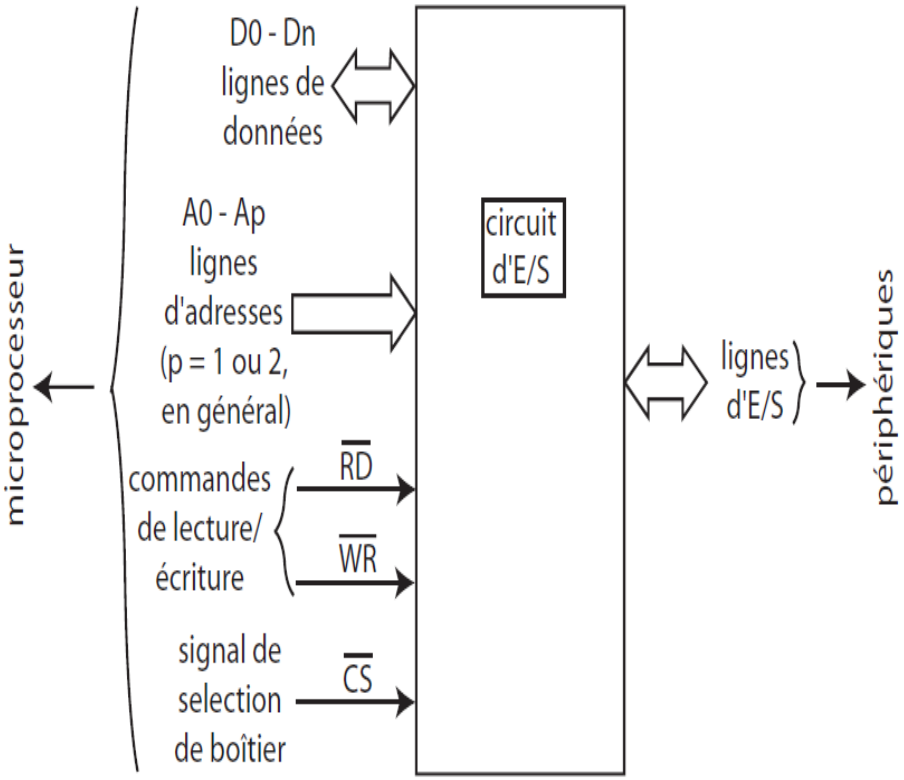


Figure II.1. Schéma synoptique d'un circuit d'E/S

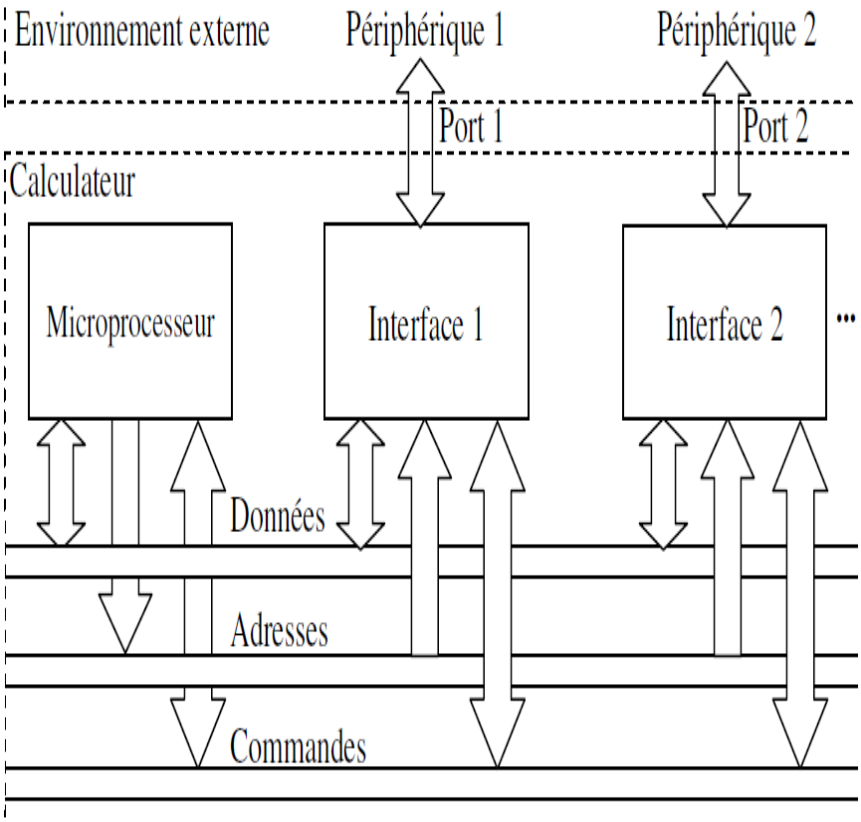
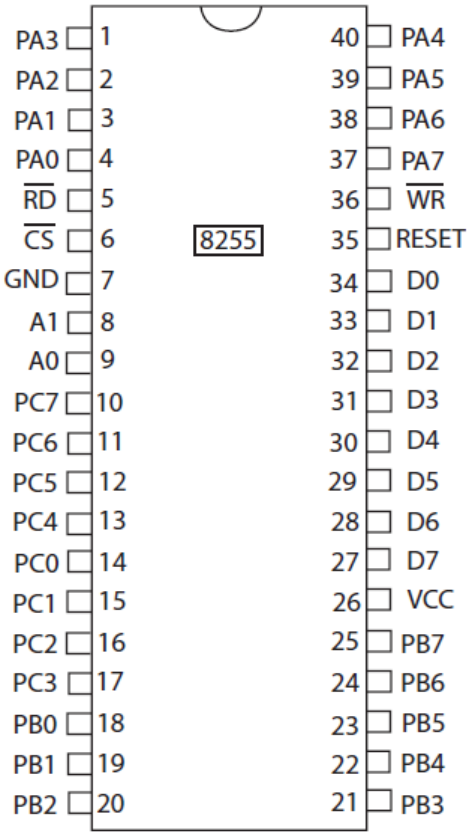
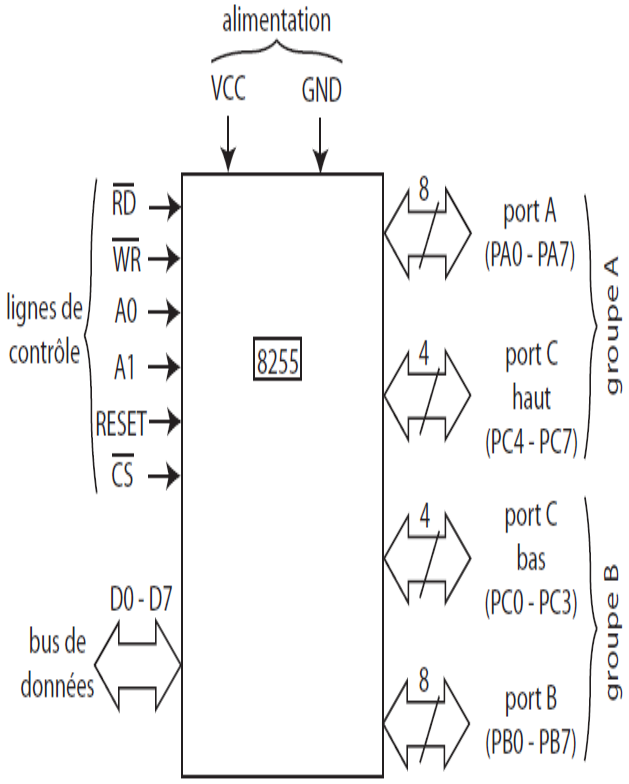


Figure II.2. Microprocesseur et circuits interfaces.



(a)



(b)

Figure II.3. (a) Schéma de brochage du 8255A.(b) Schéma fonctionnel du 8255A montrant les 3 ports A, B et C

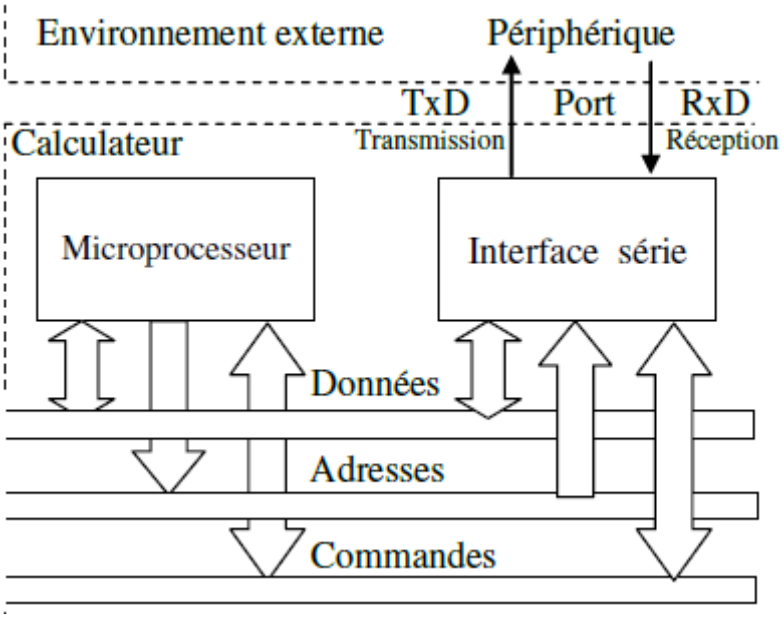


Figure II. 4 : Echange des données avec un périphérique bit à bit (série).

L'avantage de l'interface série par rapport à l'interface parallèle réside dans la diminution du nombre de connexions (1 fil pour l'émission, 1 fil pour la réception) tandis que l'inconvénient majeur s'identifie par une vitesse de transmission plus faible.
Le schéma de principe d'une interface série (Fig. II.5) montre les différents blocs de cette interface.
Les 2 signaux TxD (Transmit Data) et RxD (Receive Data) servent à transmettre les données. Les autres signaux sont pour contrôler l'échange de données.

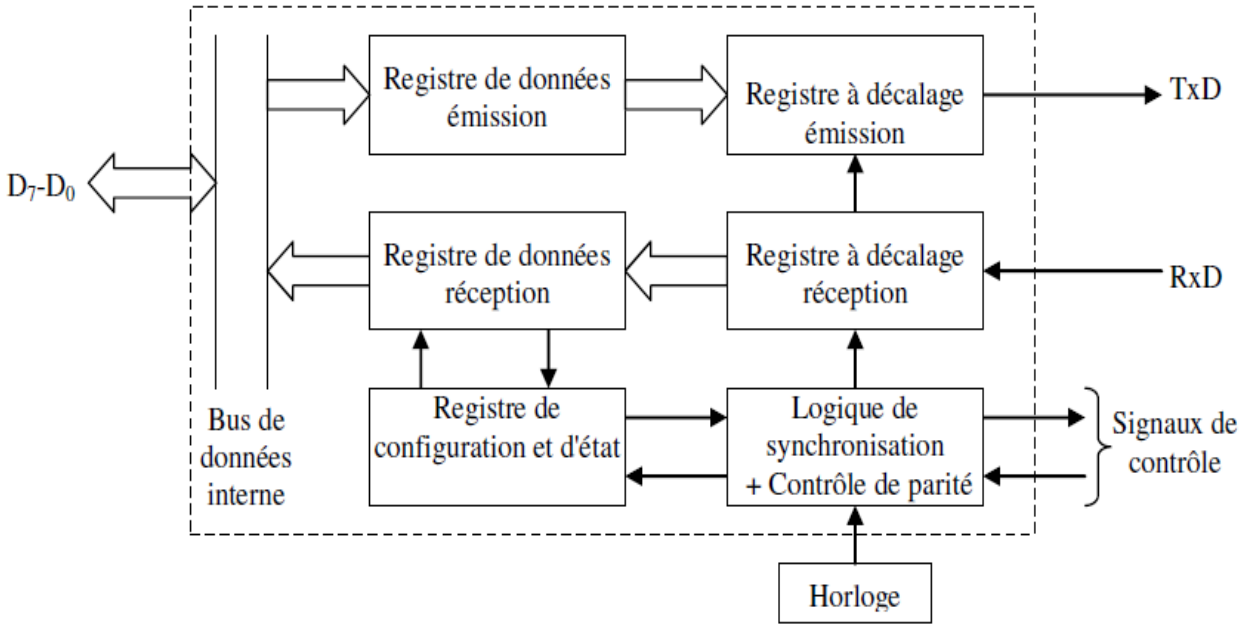


Figure II.5. Schéma de principe d'une interface série

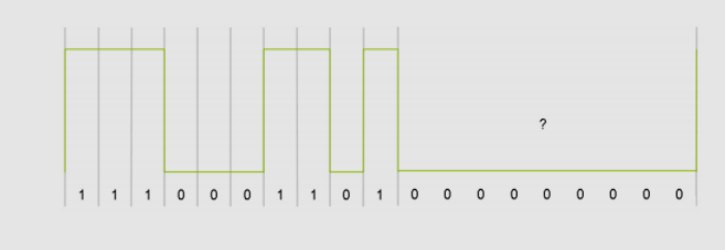
b. *Types de transmissions séries :*

Il existe deux types d'interfaces séries selon le mode de transmission:

• ***Asynchrone (UART: Universal Asynchronous Receiver Transmitter):*** chaque octet peut être émis ou reçu sans durée déterminée entre un octet et le suivant.

Dans cette transmission, seules les données sont transmises au récepteur. chaque équipement utilise son horloge pour traiter les bits transmis.

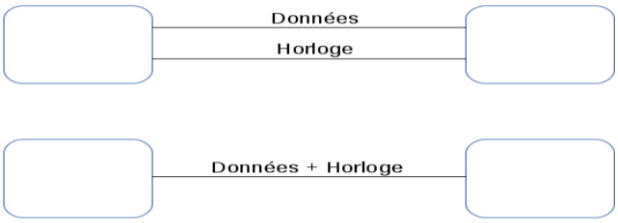
Exemple du code NRZ (Non Return to Zero) : Le bit 1 est représenté par un état significatif (par exemple, une tension clairement positive), et le bit 0 par un autre état significatif (par exemple, une tension clairement négative). Il n'existe pas d'état intermédiaire :



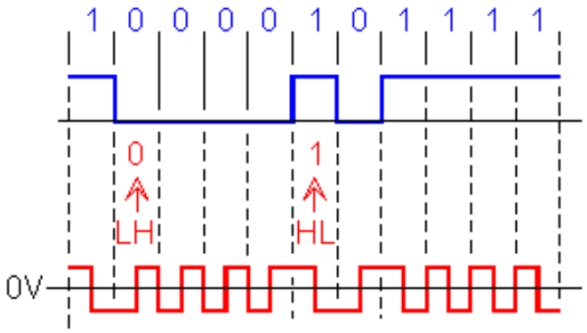
On remarque qu'il n'ya pas de transition générée lors d'une longue séquence de 0 (ou de 1), ce qui rend la synchronisation difficile voir impossible.

• ***Synchrone (USRT: Universal Synchronous Receiver Transmitter):*** les octets successifs sont transmis par blocs séparés par des octets de synchronisation.

Cette transmission doit assurer la transmission des données ainsi que l'horloge de synchronisation nécessaire à leur codage, ce qui permet de transmettre des séquences de bits plus longues (n octets) appelées trame:



Exemple du codage Manchester utilisé sur les réseaux Ethernet à 10Mbits/s :



Les vitesses de transmission pour ces deux modes sont normalisées par multiples et sous-multiples de 9600 bauds, l'unité baud correspondant à un bit par seconde.

La transmission asynchrone la plus utilisée est celle qui est définie par la norme RS232.

c. *Connexion de deux équipements par une liaison série RS232* :

Les équipements qui peuvent être connectés à travers une liaison série RS232 sont de deux types :

• les équipements terminaux de données (**DTE : Data Terminal Equipment**) qui génèrent les données à transmettre, ***Exemple : un ordinateur***.

• les équipements de communication de données (**DCE : Data Communication Equipment**) qui transmettent les données sur les lignes de communication, ***Exemple : un modem.***

Pour connecter ces équipements, on utilise des connecteurs normalisés DB9 ou DB25 :

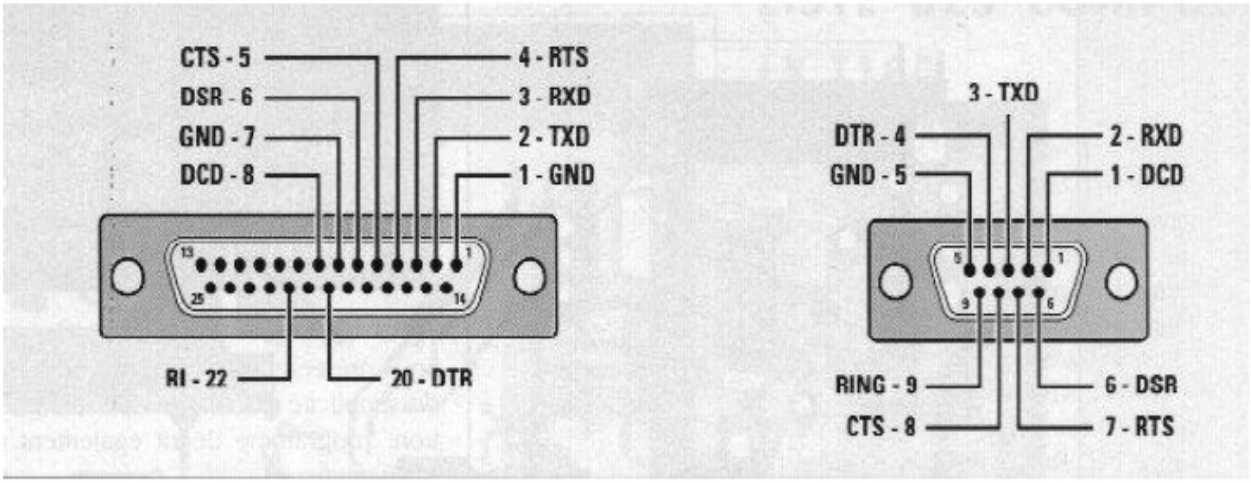


Table II.1 : Différents signaux transportés par les connecteurs DB9 et DB25

signal	n° broche DB9	n° broche DB25	description	sens	
				DTE	DCE
TxD	3	2	Transmit Data	sortie	entrée
RxD	2	3	Receive Data	entrée	sortie
RTS	7	4	Request To Send	sortie	entrée
CTS	8	5	Clear To Send	entrée	sortie
DTR	4	20	Data Terminal Ready	sortie	entrée
DSR	6	6	Data Set Ready	entrée	sortie
DCD	1	8	Data Carrier Detect	entrée	sortie
RI	9	22	Ring Indicator	entrée	sortie
GND	5	7	Ground	—	—

d. *L'interface série 8250 :*

Le 8250 est une interface série de type ***UART : (Universal Asynchronous Receiver Transmitter***). Elle possède 11 registres de 8 bits permettant de gérer la communication.

Comme il n'y a que 3 bits d'adresses (A0, A1 et A2), plusieurs registres doivent partager la même adresse. Pour remédier à ça, on utilise un bit d'un registre spécial : **DLAB** (Divisor Latch Access Bit = bit de poids fort du registre LCR : Registre du Contrôle de Ligne). En fonction de son état, on a accès soit au registre d'émission/réception, soit au diviseur d'horloge, soit au masque d'interruptions.

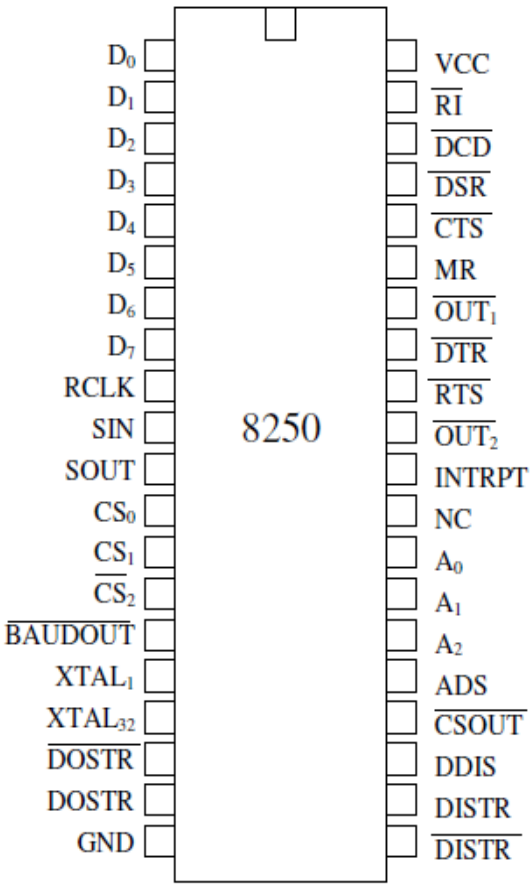


Figure II.6. Brochage du l'UART 8250.

e. Ports USB (Universal Serial Bus):

Ce bus permet la transmission de données en série. Cette nouvelle technique se doit d'être rapide, bidirectionnelle, synchrone, de faible coût et l'attachement d'un nouveau périphérique doit être dynamique. De plus, l'alimentation des équipements est possible. Le débit brut (émission plus réception) peut aller jusqu'à 12 Mbit/s au maximum.

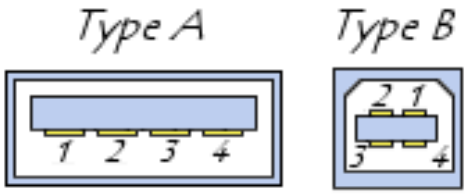
Le port USB possède 4 broches à savoir : une paire d'alimentation (VCC et GND) et une paire torsadée de données inversés (-DATA et +DATA). Au repos +DATA et -DATA sont à l'état haut et bas respectivement.

Table II.2. Sélection des registres du 8250

DLAB	A2	A1	A0	Registre
0	0	0	0	RBR : Receiver Buffer Register, registre de réception (accessible seulement en lecture)
0	0	0	0	THR : Transmitter Holding Register, registre d'émission (accessible seulement en écriture)
1	0	0	0	DLL : Divisor Latch LSB, octet de poids faible du diviseur d'horloge
1	0	0	1	DLM : Divisor Latch MSB, octet de poids fort du diviseur d'horloge
0	0	0	1	IER : Interrupt Enable Register, registre d'autorisation des interruptions
X	0	1	0	IIR : Interrupt Identification Register, registre d'identification des interruptions
X	0	1	1	LCR : Line Control Register, registre de contrôle de ligne
X	1	0	0	MCR : Modem Control Register, registre de contrôle modem
X	1	0	1	LSR : Line Status Register, registre d'état de la ligne
X	1	1	0	MSR : Modem Status Register, registre d'état du modem
X	1	1	1	SCR : Scratch Register, registre à usage général

Types de connecteurs USB :

Il existe deux types de connecteurs USB :



Les connecteurs dits de type A, dont la forme est rectangulaire et servant généralement pour les périphériques à faible bande passante (clavier, souris, webcam, etc.) ;

Les connecteurs dits de type B, dont la forme est carrée et utilisés principalement pour des périphériques à haut débit (disques durs externes, etc.).

f. IEEE 1394 ou FireWire (ligne de feu):

Standard récent, il sert pour les périphériques graphiques ou vidéo tel 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 périphériques alors que l'ordinateur est en cours de traitement.

L'interface série multiplexée FireWire suit à peu prés la même structure que le bus USB, si ce n'est qu'il utilise un câble composé de six fils (deux paires pour les données et pour l'horloge, et deux fils pour l'alimentation électrique) lui permettant d'obtenir un débit de 800 Mb/s. Ainsi, les deux fils dédiés à une horloge montrent la différence majeure qui existe entre le bus USB et le bus IEEE 1394, c'est-à-dire la possibilité de fonctionner selon les deux modes : asynchrone et synchrone.

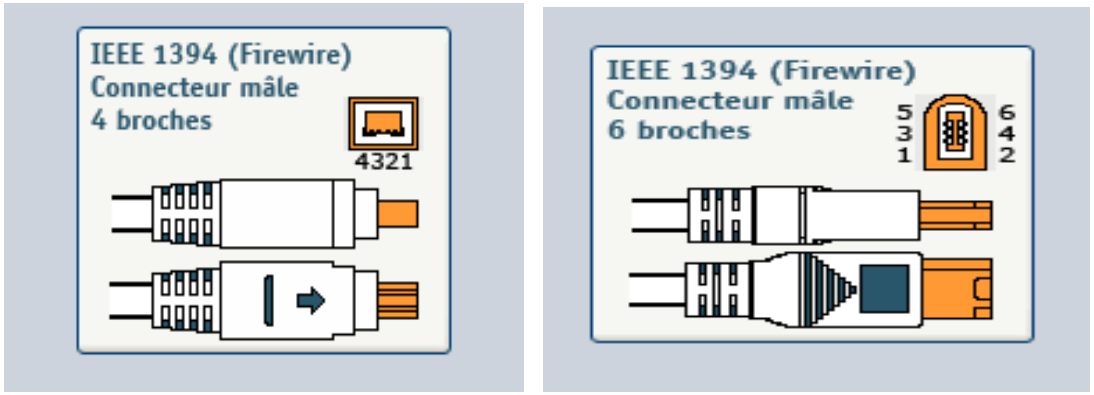


Figure II.7. Deux types de schéma pour le bus IEEE 1394 selon le nombre de broches

(a)

(b)

Figure II.8. Bus USB et bus FireWire (a),
Firewire female et male (b)

Chapitre III

Les échanges de données

Chapitre III. Les échanges de données

III-1- Introduction :

On a vu que chaque système à microprocesseurs est équipé d'une ou plusieurs interfaces (ou contrôleurs) d'entrées/sorties qui permettent d'assurer la communication entre le processeur et le monde extérieur. En plus des interfaces entrées/sorties programmables, il existe des liaisons E/S simples qui utilisent le minimum de ports. Les techniques d'échange de données à travers ces liaisons sont très importantes pour la performance du système. Dans le cadre d'échange entre plusieurs ordinateurs ou entre ordinateur et périphériques externes, il faut choisir le codage de l'information à transmettre. La transmission est caractérisée par le sens des échanges, le mode et la synchronisation entre émetteur et récepteur.

III-2- Modes d'échange d'informations:

Avant d'envoyer ou de recevoir des informations, le microprocesseur doit connaitre l'état du périphérique. Il doit savoir s'il est prêt à recevoir ou à transmettre une information pour que la transmission se fasse correctement. Il existe 2 protocoles d'échange de données entre le microprocesseur et les périphériques :

- **Mode programmé (Par scrutation ou interruption)** : Le microprocesseur sert d'intermédiaire entre la mémoire et le périphérique.
- **Mode DMA (accès directe à la mémoire)** : Le microprocesseur ne se charge pas de l'échange de données.

III-2-1 Mode programmé :

La façon la plus immédiate de gérer des échanges consiste à les prévoir dans le programme que l'ordinateur est entrain d'exécuter. La structure du programme peut être comme suit (en mode polling):

- **Attendre** la disponibilité de la donnée.
- **Consommer** la donnée par sa lecture dans le port des données par exemple et son rangement en mémoire.
- **Signaler** que la donnée est acquise.

a) Mode d'échange programmé par scrutation (*polling* ou attente active) :

Connue aussi sous le nom de « **technique de l'interrogation** », le microprocesseur interroge l'interface pour savoir si des transferts sont prés. Ce problème prend un aspect dynamique puisqu'on attend le survient d'un certain évènement mais nous ne savons pas exactement à quel

moment. La façon de détecter l'évènement « par interrogation » consiste à tester répétitivement le signal d'entrée jusqu'à ce qu'on obtienne la valeur logique voulue. Tant qu'il n'y a pas de réponse, le microprocesseur est en état d'attente ce qui est un inconvénient majeur. Il est monopolisé en permanence par l'interface d'entrée/sortie. Cette technique n'est utilisable que si l'attente est courte, elle convient donc pour les périphériques rapides qui transfèrent les informations par blocs (écran ou disque). Elle est à exclure dans le cas des périphériques lents (imprimante, clavier,…).

De plus, l'initiative de l'échange de données est dépendante du programme exécuté par le microprocesseur. Il peut donc arriver que des requêtes d'échange ne soient pas traitées immédiatement car le microprocesseur ne se trouve pas encore dans la boucle de scrutation.

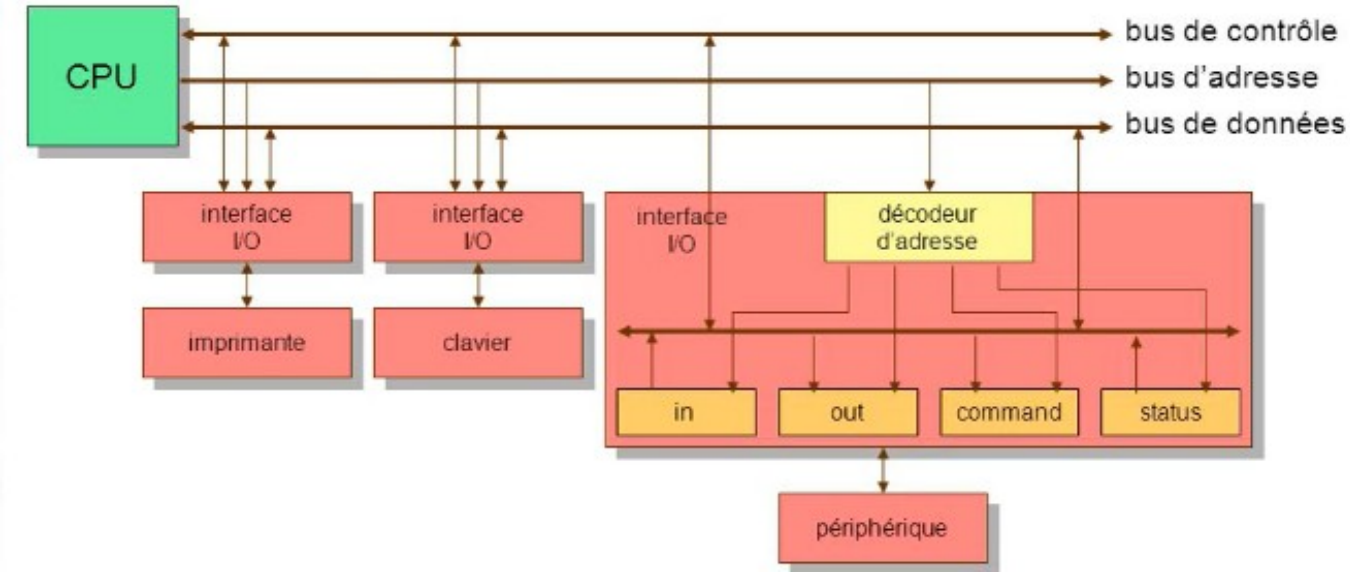


Figure III.1. Gestion des périphériques par interrogation

b) Mode d'échange de données par interruption (*interrupt*):

Une interruption est généralement un signal asynchrone au programme en cours, pouvant être émis par tout dispositif externe au microprocesseur. Ce dernier possède une ou plusieurs entrées réservées à cet effet. Le microprocesseur n'attend pas la disponibilité de la donnée, il peut mettre à profit cette période pour exécuter un autre programme.

L'interruption peut être logicielle (générée explicitement par un programme) ou matérielle (générée par des périphériques), ce qui est le cas dans cette partie du cours.

Dans un échange de données par interruption, le processeur exécute son programme principal. Dès que l'unité périphérique est prête pour un transfert, elle envoie un signal qui engendre une demande d'interruption en agissant sur la ligne d'interruption du processeur (IRQ : Interrupt ReQuest line). La tache principale est interrompue par un mécanisme de commutation, et le

contrôle passe à la procédure d'interruption qui redonne le contrôle après avoir terminé l'échange des données (Fig.III.2).

Cette technique convient au transfert des données avec des périphériques lents (imprimantes lentes,…). Elle est évidemment la seule utilisable dans le cas où le processeur doit réagir à des évènements imprévisibles, ce qui est le cas des transferts de données par caractères (clavier, souris,…).

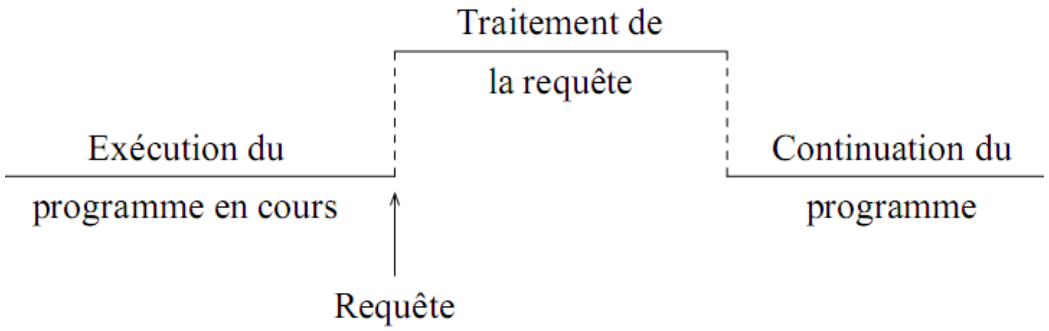


Figure III.2. Schéma synoptique d'un échange par interruption

Table III.1 : Listes des IRQs

Numéro	Usage typique	Numéro	Usage typique
IRQ 0	Horloge interne	IRQ 8	Horloge
IRQ 1	Clavier	IRQ 9	*Aucun*
IRQ 2	Réservé	IRQ 10	*Aucun*
IRQ 3	COM 2	IRQ 11	*Aucun*
IRQ 4	COM 1	IRQ 12	Souris PS/2
IRQ 5	Carte son	IRQ 13	FPU
IRQ 6	Lecteur disquettes	IRQ 14	IDE 1
IRQ 7	LPT 1	IRQ 15	IDE 2

c) Principe de fonctionnement d'une interruption :

S'il y'a une requête sur la ligne d'interruption du microprocesseur, ce dernier interrompt toutes ses activités et sauvegarde l'état présent pour la restituée après (registre, PC, accumulateurs, registre d'état). La sauvegarde se fait dans un registre particulier appelé **pile** (la première donnée sauvegardée sera la dernière à être restituée).

Remarques :

- ✓ *Certaine sources d'interruption sont **masquables (**peuvent être interdites ou autorisées). Elles possèdent leurs propres autorisations de fonctionnement sous la forme d'un bit à positionner (nommée le **masque d'interruption**).*

- ✓ *chaque source d'interruption possède un **vecteur d'interruption** où est sauvegardée l'adresse de départ du programme à exécuter.*
- ✓ *Les interruptions sont classées par ordre de **priorité.***

III-2-2 Mode d'échange des données par DMA (Direct Memory Access):

Ce mode permet le transfert de blocs de données entre la mémoire et un périphérique sans passer par le microprocesseur. Le circuit **contrôleur de DMA** prend en charge les différentes opérations. Le microprocesseur doit tout de même :

- ✓ Initialiser l'échange en donnant au DMA l'identification du périphérique concerné.
- ✓ Donner le sens du transfert.
- ✓ Fournir l'adresse du premier et du dernier mot concernés par le transfert.

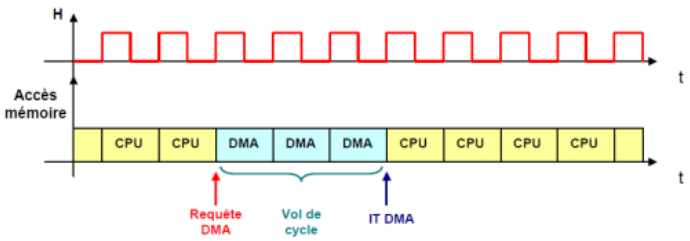
Le contrôleur DMA prend en charge :

- ✓ Les commandes pour le contrôleur de périphériques.
- ✓ Les commandes et adresses pour la mémoire.

Un contrôleur DMA est doté d'un registre d'adresses, registre de données, compteur et un dispositif de commande logique (câblée).

Pour chaque mot échangé, le DMA demande au processeur le contrôle du bus, effectue la lecture ou l'écriture mémoire à l'adresse contenue dans son registre et libère le bus. Il incrémente ensuite cette adresse et décrémente son compteur. Lorsque le compteur atteint zéro, le dispositif informe le processeur de la fin du transfert par une ligne d'interruption.

Le principal avantage est que pendant toute la durée du transfert, le processeur est libre d'effectuer un traitement quelconque. La seule contrainte est une limitation de ses propres accès mémoire pendant l'opération puisqu'il doit parfois retarder certains de ses accès pour permettre au dispositif d'accès direct à la mémoire d'effectuer les siens : il y'a apparition de vols de cycles :



Les dispositifs qui peuvent utilisés le contrôleur DMA sont :

-Lecteur de disquettes.

- disque dur.

- Lecteur et graveur CDROM.

- Contrôleur SCSI et PCI.
- Carte son.
- Carte graphique.

L'accès direct à la mémoire se déroule ainsi:

- **Le CPU charge IOAR et DC.**
- **Quand le contrôleur DMA est prêt, il active la ligne DMA_REQ. Le CPU donne le contrôle du bus au contrôleur DMA et active DMA_ACK.**
- **Le périphérique et la mémoire échangent les données. DC est décrémenté et IOAR incrémenté après chaque transfert.**
- **Si DC est égal à 0, le contrôleur DMA relâche la ligne DMA_REQ. Le CPU reprend le contrôle du bus et relâche DMA_ACK.**

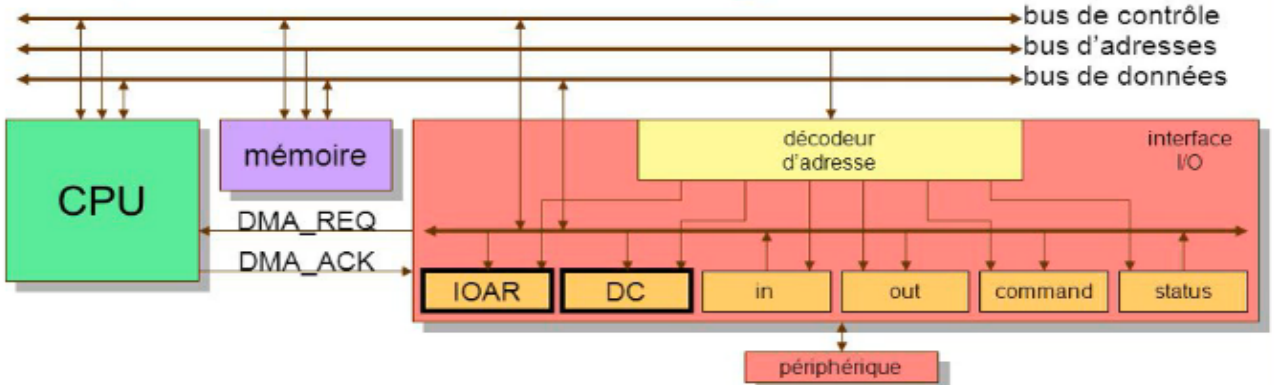


Figure III.3. Echange DMA des données entre périphérique et mémoire

Chapitre IV

Modes d'adressages et jeux d'instructions

Chapitre IV. Modes d'adressages et jeux d'instructions

IV-1- Introduction :

Le langage machine se compose d'instructions binaires telles qu'on les trouve en mémoire au moment de l'exécution d'un programme. En effet, les premiers programmes étaient écrit en binaire, c'était une tache difficile et exposée aux erreurs car il fallait aligner les séquences de bits dont la signification n'est pas évidente. Donc si le langage machine est parfaitement adaptés aux ordinateurs il ne convient pas aux programmeurs. C'est pour cela qu'il a été abandonné depuis longtemps. Pour faciliter la programmation, les programmes ont été écrits en donnant directement les noms abrégés des opérations, Ce sont des codes mnémoniques qu'on pouvait facilement mémoriser.

Contrairement aux langages évolués, tels que le C, Pascal… L'assembleur, ou « langage d'assemblage » est constitué d'instructions directement compréhensibles par le microprocesseur : c'est ce qu'on appelle un langage de bas niveau. Il est donc intimement lié au fonctionnement de la machine.

Dans cette partie du cours, nous allons étudier la programmation en langage assembleur d'un microprocesseur 8086.

IV-2- Architectures CISC et RISC :

Chaque microprocesseur reconnaît un ensemble d'instructions appelé **jeu d'instructions** (Instruction Set) fixé par le constructeur. Pour les microprocesseurs classiques, le nombre d'instructions reconnues varie entre 75 et 150 (microprocesseurs **CISC** : *Complex Instruction Set Computer*). Il existe aussi des microprocesseurs dont le nombre d'instructions est très réduit (microprocesseurs **RISC** : *Reduced Instruction Set Computer*) : entre 10 et 30 instructions, permettant d'améliorer le temps d'exécution des programmes.

Table IV.1 : différences entre les architectures RISC et CISC

Architecture CISC (Comprehensive Instruction Set Computer)	Architecture RISC (Reduced Instruction Set Computer)
instructions complexes prenant plusieurs cycles	instructions simples ne prenant qu'un seul cycle
instructions au format variable	instructions au format fixe
décodeur complexe (microcode)	décodeur simple (câblé)
peu de registres	beaucoup de registres
toutes les instructions sont susceptibles d'accéder à la mémoire	seules les instructions LOAD et STORE ont accès à la mémoire
beaucoup de modes d'adressage	peu de modes d'adressage
compilateur simple	compilateur complexe

<u>Remarque :</u> Un microprocesseur à architecture RISC est, en général, plus puissant: la durée moyenne d'exécution d'une instruction est plus courte.

IV-3- Définition de l'instruction :

Une instruction est définie par son code opératoire, valeur numérique binaire difficile à manipuler par l'être humain. On utilise donc une **notation symbolique** pour représenter les instructions : les **mnémoniques**. Un programme constitué de mnémoniques est appelé **programme en assembleur**.

Une instruction est composée de deux champs :

{Label :} Mnémonique {opérande} { ; commentaire}

- Le code **opération**, qui indique au processeur quelle instruction réaliser.
- Le champ **opérande** qui contient la donnée, ou la référence à une donnée en mémoire (son adresse).

L'opérande est un registre ou une case mémoire.

IV-4- Modes d'adressage :

Le microprocesseur 8086 possède 7 modes d'adressage:

- Mode d'adressage registre à registre.
- Mode d'adressage immédiat.
- Mode d'adressage direct.
- Mode d'adressage registre indirect.
- Mode d'adressage indexé.
- Mode d'adressage indirect indexé.
- Mode d'adressage relatif à une base.

IV-4-1 Mode d'adressage registre à registre :

Ce mode d'adressage concerne tout transfert ou toute opération, entre deux registres de même taille.

Dans ce mode l'opérande sera stocké dans un registre interne au microprocesseur.

Exemple (Fig.IV.1):

```
Mov  AX, BX   ; cela signifie que l'opérande stocker dans le registre BX
               sera transféré vers le registre AX.
```

MOV est l'abréviation du verbe « to move » : déplacer.

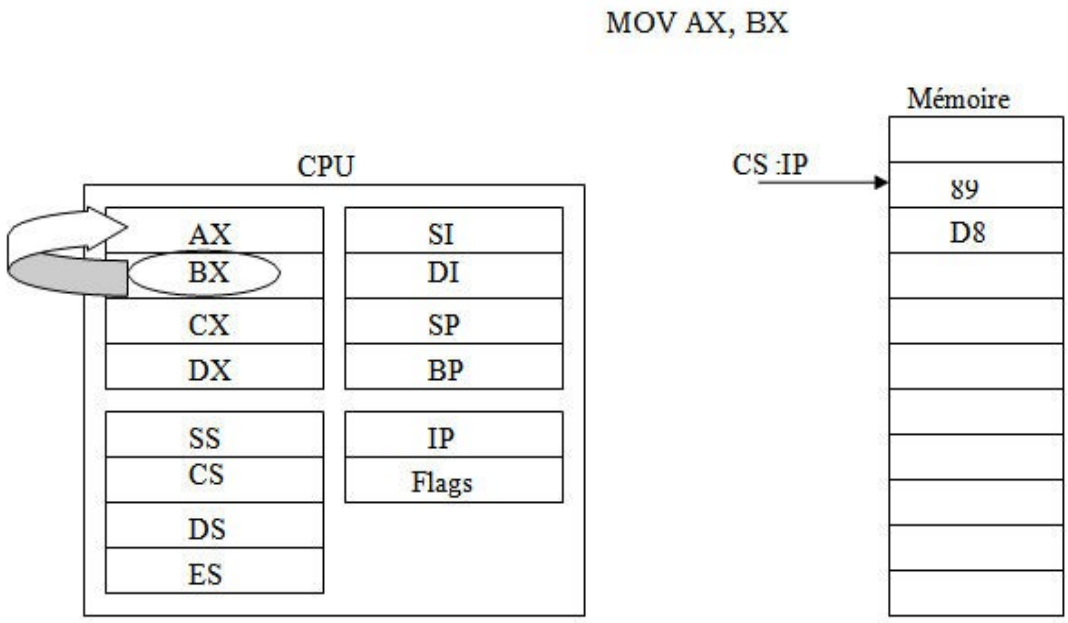


Figure IV.1. Adressage registre/registre

<u>Remarque :</u> Quand on utilise l'adressage registre, le microprocesseur effectue toutes les opérations d'une façon interne. Donc dans ce mode il n'y a pas d'échange avec la mémoire, ce qui augmente la vitesse de traitement de l'opérande.

IV-4-2 Mode d'adressage immédiat :

Dans ce mode d'adressage, l'opérande apparaît dans l'instruction elle-même.

<u>Exemple (Fig.IV.2):</u>

```
MOV   AX,500H ; cela signifie que la valeur 500H sera stockée
                immédiatement dans le registre AX
```

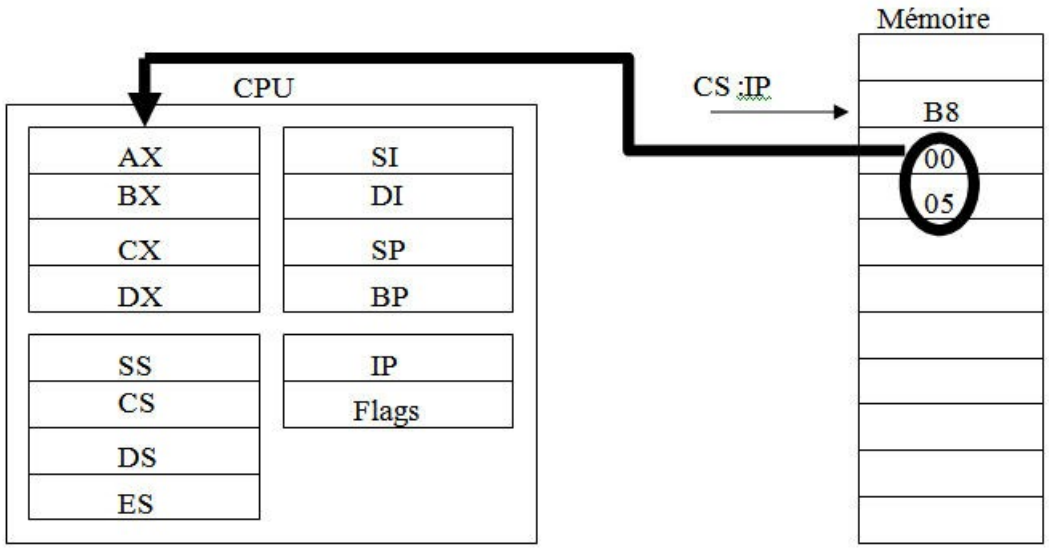


Figure IV.2. Adressage immédiat

Remarque : Pour les instructions telles que :

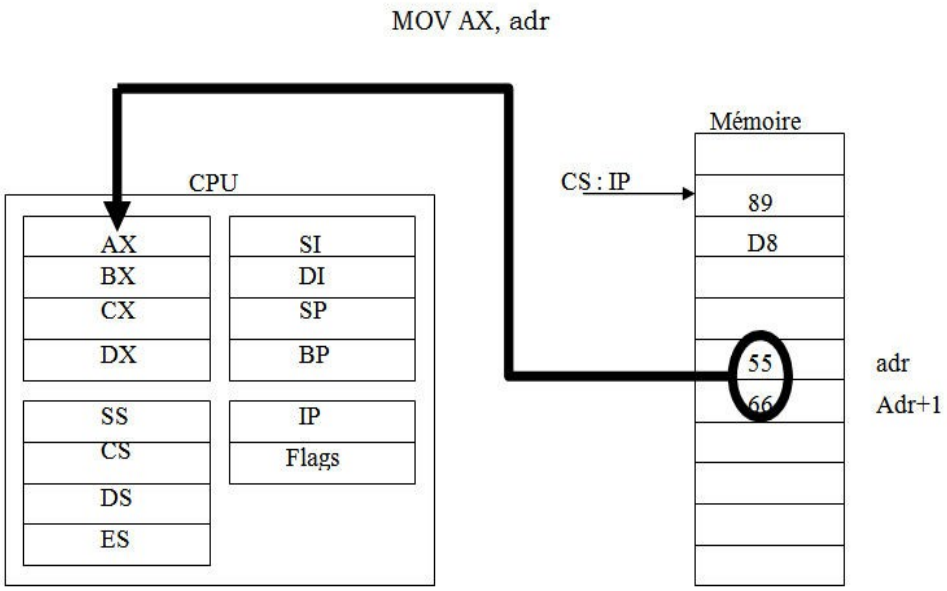
```
MOV    AX,-500H ; le signe(-) sera propager dans le registre jusqu'à
                  remplissage de ce dernier.
```

IV-4-3 Mode d'adressage direct (Fig.IV.3.a) :

Dans ce mode on spécifie directement l'adresse de l'opérande dans l'instruction.

```
MOV AX, [adr] ; La valeur adr (adresse de la case mémoire) est une
            constante (un déplacement) qui doit être ajouté au contenu
            du registre DS pour former l'adresse physique de 20 bits.
```

(a)



(b)

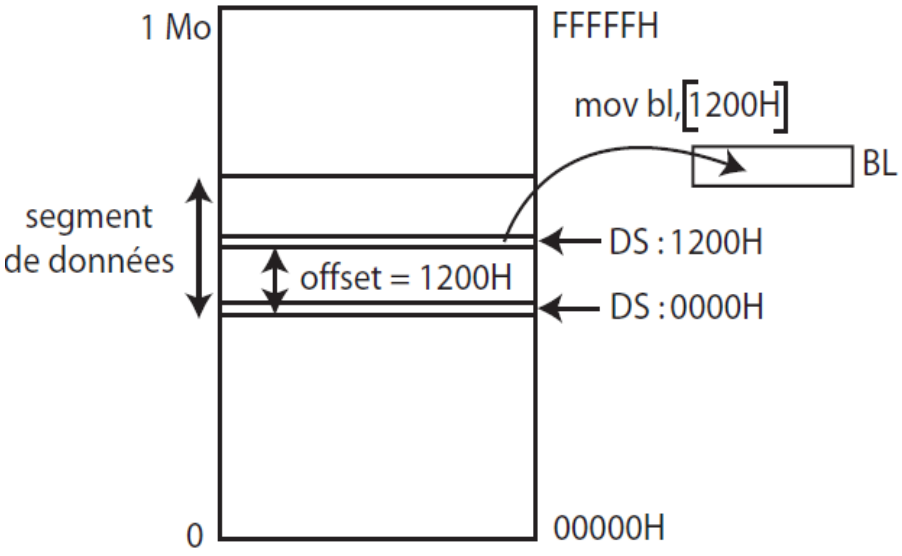


Figure IV.3. Adressage direct

Exemple (Fig.IV.3.b) :

```
MOV BL, [1200H] ; transfère le contenu de la case mémoire d'adresse
                  effective (offest) 1200H vers le registre BL
```

IV-4-4 Mode d'adressage indirect par registre ou basé (et basé avec déplacement) :

Dans ce mode d'adressage, l'adresse de l'opérande est stockée dans un registre qu'il faut bien évidemment le charger au préalable par la bonne adresse. L'adresse de l'opérande sera stockée dans un registre de base (*BX ou BP*) ou un indexe (*SI ou DI*).

Exemple (Fig.IV.4):

```
MOV  BX,offset adr
MOV  AX,[BX] ; transfère la donnée dont l'offset est contenu dans le
registre de base BX vers le registre AL. Le segment associé par défaut
au registre BX est le segment de données : on dit que l'adressage est
basé sur DS
```

Remarque: *Le symbole* [ ] *design l'adressage indirect.*

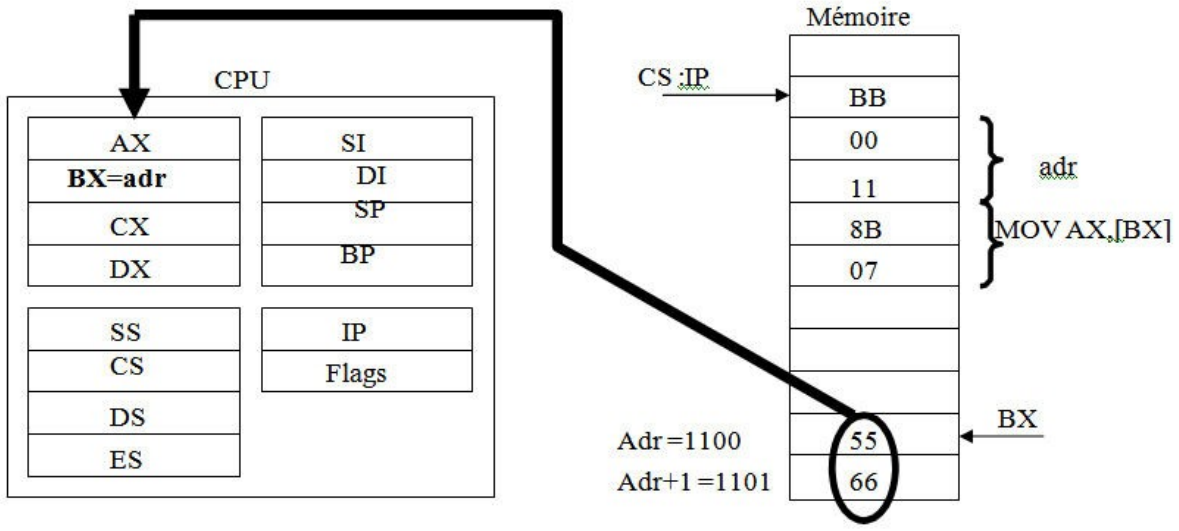


Figure IV.4. Adressage indirect basé

Exemple :

```
MOV AL, [BP] ; le segment par défaut associé au registre de base BP est
           le segment de pile. Dans ce cas, l'adressage est basé sur SS.
```

IV-4-5 Mode d'adressage indexé (et indexé avec déplacement) :

Semblable à l'adressage basé, sauf que l'offset est contenu dans un registre d'index SI ou DI, associés par défaut au segment de données. Les modes d'adressage basés ou indexés permettent la manipulation de tableaux rangés en mémoire.

Exemple:

```
MOV [DI], BX ; charge les cases mémoire d'offset DI et DI + 1 avec le
               contenu du registre BX.
```

Exemple:

```
MOV AL, [SI+200H] ; charge le registre AL avec le contenu de la case
mémoire dont l'offset est contenu dans SI plus un déplacement de 200H.
```

IV-4-6 Mode d'adressage indirect indexé (et indirect indexé avec déplacement) :

L'offset est obtenu en faisant la somme d'un registre de base et d'un registre d'index, et d'une valeur constante dans le cas avec déplacement. Ce mode d'adressage permet l'adressage de structures de données complexes : matrices, enregistrements…

Exemple :

```
MOV AH, [BX+SI+100H] ; Dans cet exemple, BX et SI peuvent être
       considérés comme indices de ligne et de colonne dans une matrice.
```

IV-4-7 Mode d'adressage relatif à une base :

Dans ce mode d'adressage, le déplacement est déterminé par : soit le contenu de BX, soit le contenu de BP, auquel est éventuellement ajouté un décalage sur 8 ou 16 bits signé. DS et SS sont pris par défaut.

Exemple (Fig.IV.5):

```
MOV AX,[BX]+2 ;Cela signifie que dans le registre AX on va mettre le
               contenu de la case mémoire pointée par BX+2.
```

Remarque : Les syntaxes suivantes sont identiques :

```
MOV AX,[BX+2]
MOV AX,[BX]+2
MOV AX,2[BX]
```

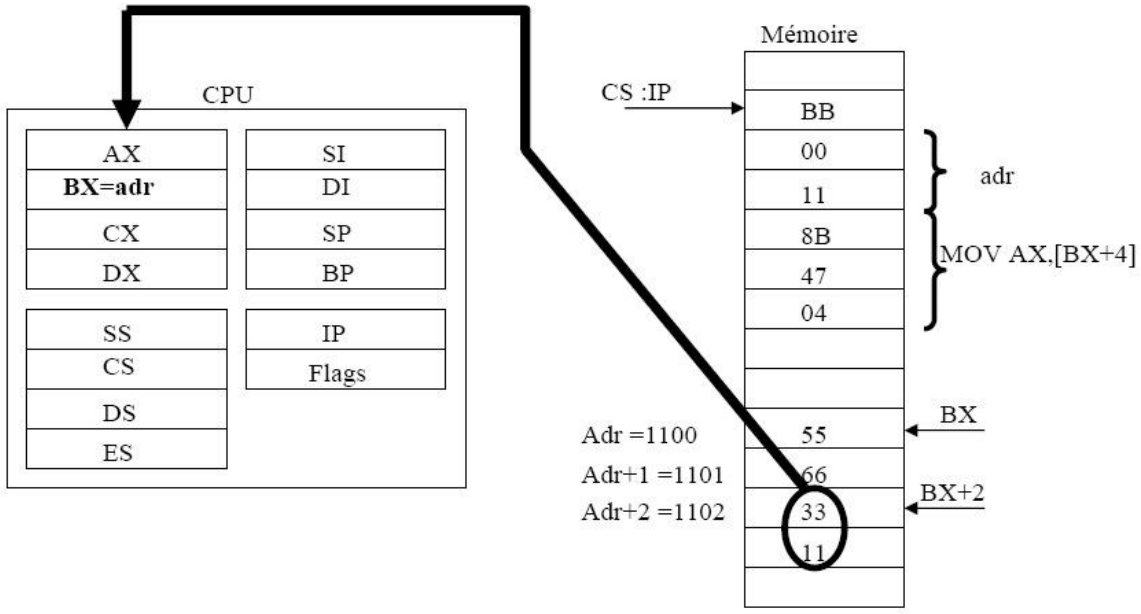


Fig. IV.5. Adressage relatif à une base

IV-5- Types des instructions :

Les instructions peuvent être classées en groupes :

- Instructions de transfert de données ;
- Instructions arithmétiques ;
- Instructions logiques ;
- Instructions de branchement ...

IV-5-1 Instructions de transfert de données :

Elles permettent de déplacer des données d'une **source** vers une **destination** :

```
MOV destination, source
```

• Registre vers mémoire;

• Registre vers registre;

• Mémoire vers registre ;

Remarque : le microprocesseur 8086 n'autorise pas les transferts de mémoire vers mémoire (Pour ce faire, il faut passer par un registre intermédiaire).

```
MOV  [Adresse 1], [Adresse 2] ; X Faute
```

Les modes d'adressage servent à spécifier l'adresse d'une case mémoire dans une instruction.

Exemples :

```
MOV AX,BX : charger le contenu du registre BX dans le registre AX.
                        (Mode d'adressage par registre).
```

```
MOV AL,12H ; charger le registre AL avec la valeur 12H.
                        (Mode d'adressage immédiat).
```

```
MOV BL,[1200H] ; transfère le contenu de la case mémoire d'adresse
                 effective (offset) 1200H vers le registre BL.
                        (Mode d'adressage direct).
```

*Remarque : dans le cas de l'adressage immédiat de la mémoire, il faut indiquer le **format** de la donnée : octet ou mot (2 octets) car le microprocesseur 8086 peut manipuler des données sur 8 bits ou 16 bits. Pour cela, on doit utiliser un **spécificateur de format** :*

Exemple (Fig.IV.6):

```
mov byte ptr [1100H],65H ; transfère la valeur 65H (sur 1 octet) dans la
                           case mémoire d'offset 1100H ;
mov word ptr [1100H],65H ; transfère la valeur 0065H (sur 2 octets) dans
                          les cases mémoire d'offset 1100H et 1101H.
```

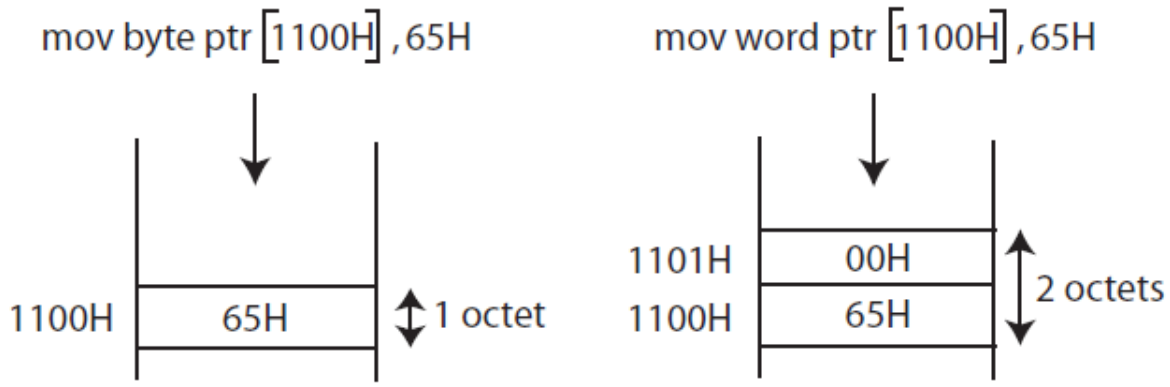


Figure IV.6. Adressage immédiat avec indicateur de format

IV-5-2 Instructions arithmétiques :

Les instructions arithmétiques de base sont l'**addition**, la **soustraction**, la **multiplication** et la **division** qui incluent diverses variantes. Plusieurs modes d'adressage sont possibles.

a. L'addition (ADD) :

Ajouter le contenu de l'opérande 1 au contenu de l'opérande 2, et placer le résultat de cette opération dans l'opérande 1.

```
ADD opérande 1, opérande 2
```

L'opération effectuée est : opérande1 ← opérande1 + opérande2.

Exemple: Ajoute le contenu de registre BX au contenu de l'accumulateur AL.

```
MOV AX , 4H
MOV BX, 2H
ADD BX, AX ;  BX← BX+AX le résultat de cette opération est placé dans le
            registre BX=6
ADD AX, BX ;  AX← AX+BX le résultat de cette opération est placé dans le
              registre AX=6
```

Exemple: Ajouter le contenu de la case mémoire d'offset 1100H au contenu de l'accumulateur AL.

```
MOV AL,4H
MOV [1100H],2H
ADD AL,[1100H] ; AL←AL+[1100H]
```

Incrémentation de 1 (INC):

```
INC opérande 1
```

L'opération effectuée est : opérande1 ← opérande 1 + 1.

Exemple :

```
MOV  AL, 3H
INC   AL   ; AL←AL+1 ;
```

b. La soustraction (SUB):

```
SUB opérande1, opérande 2
```

L'opération effectuée est : opérande1 ← opérande1 – opérande2.

Exemples :

```
MOV AX , 4H
MOV BX, 2H
SUB BX, AX ;  BX← BX-AX le résultat de cette opération est placé dans le
              registre BX=-2 (FFFE)
SUB AX, BX ;  AX← AX-BX le résultat de cette opération  est placé dans
              le registre AX=2
```

Décrémentation de 1 (DEC) :

```
DEC opérande1
```

L'opération effectuée est : opérande1 ← opérande1 – 1.

Exemple :

```
MOV  AL, 3H
DEC   AL   ; AL←AL-1 ;
```

c. La multiplication (MUL):

Cette instruction effectue la multiplication du contenu de AL par un opérande sur 1 octet ou du contenu de AX par un opérande sur 2 octets. Le résultat est placé dans AX si les données à multiplier sont sur 1 octet (résultat sur 16 bits), dans (DX,AX) si elles sont sur 2 octets (résultat sur 32 bits).

Exemples:

```
MOV AL, 6H
MOV BL, 2H
MUL BL ; AX←BL*AL
```

```
MOV AL,4H
MOV [111H],2H
MUL [111H]; AX←[111H](Contenu de la case mémoire (C'est-à-dire 2H)
           d'offset 111H)*AL ; et le résultat de cette opération est
           placé dans cette case.
```

```
MOV AX,4253
MOV BX,1689
MUL BX
```

```
MOV AL,43
MOV byte ptr [1200H],28
MUL byte ptr [1200H]
```

```
MOV AX,1234
MOV word ptr [1200H],5678
MUL word ptr [1200H]
```

d. La division (DIV):

Cette instruction effectue la division du contenu de AX par un opérande sur 1 octet ou le contenu de (DX,AX) par un opérande sur 2 octets. Résultat : si l'opérande est sur 1 octet, alors AL = quotient et AH = reste ; si l'opérande est sur 2 octets, alors AX = quotient et DX = reste.

Exemple :

```
MOV AX, 6H
MOV BL, 2H
DIV BL ; AX←AX/BL ; Diviser le contenu du registre AX par le contenu
         du registre BL, et placé le résultat dans le registre AX.
```

e. Autres instructions arithmétiques :

ADC : addition avec retenue ;

SBB : soustraction avec retenue ;

INC : incrémentation d'une unité ;

DEC : décrémentation d'une unité ;

IMUL : multiplication signée ;

IDIV : division signée.

IV-5-3 Instructions logiques :

Ce sont des instructions qui permettent de manipuler des données au niveau des bits. Les opérations logiques de base sont : ET, OU, Complément à 1….et Décalages et rotations.

Les différents modes d'adressage sont disponibles.

a. ET logique (And):

L'opération effectuée est : opérande1 ← opérande1 ET opérande2.

```
AND opérande1, opérande2
```

Exemple :

```
mov al,10010110B
mov bl,11001101B
and al, bl
```

→

$$\begin{array}{ll} AL = & 1\ 0\ 0\ 1\ 0\ 1\ 1\ 0 \\ BL = & 1\ 1\ 0\ 0\ 1\ 1\ 0\ 1 \\ \hline AL = & 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0 \end{array}$$

Application masquage de bits pour mettre à zéro certains bits dans un mot.

Exemple : masquage des bits 0, 1, 6 et 7 dans un octet :

7	6	5	4	3	2	1	0	
0	1	0	1	0	1	1	1	
0	0	1	1	1	1	0	0	← masque
0	0	0	1	0	1	0	0	

```
MOV AL, 01010111B
AND  AL, 00111100B
```

b. OU logique (Or):

L'opération effectuée est : opérande1 ← opérande1 OU opérande2.

```
OR opérande1, opérande 2
```

Application masquage: mise à 1 d'un ou plusieurs bits dans un mot.

Exemple : dans le mot 10110001B on veut mettre à 1 les bits 1 et 3 sans modifier les autres bits.

Les instructions correspondantes peuvent s'écrire :

```
MOV AH, 10110001B
Or AH,00001010B
```

7	6	5	4	3	2	1	0	
1	0	1	1	0	0	0	1	
0	0	0	0	1	0	1	0	← masque
1	0	1	1	1	0	1	1	

c. Complément à 1:

L'opération effectuée est : Opérande ← $\overline{\text{Opérande}}$.

```
NOT Opérande
```

Exemple :

```
mov al,10010001B
not al
```

$\rightarrow \quad AL = \overline{10010001}B = 0110111$

Table IV.2. Table de vérité du OU exclusif

a	b	s
0	0	0
0	1	1
1	0	1
1	1	0

d. OU exclusif (XOR):

L'opération effectuée est : opérande1 ← opérande1 ⊕ opérande2.

```
XOR opérande 1, opérande 2
```

Exemple :

Mise à zéro d'un registre :

```
MOV AL,25
XOR AL,AL
```

e. Instructions de décalages et de rotations:

Ces instructions déplacent d'un certain nombre de positions les bits d'un mot vers la gauche ou vers la droite. Dans les décalages, les bits qui sont déplacés sont remplacés par des zéros. Il y a les décalages logiques (opérations non signées) et les décalages arithmétiques (opérations signées).

Dans les rotations, les bits déplacés dans un sens sont réinjectés de l'autre coté du mot.

- **Décalage logique vers la droite** (SHift Right) :

Cette instruction décale l'opérande de *n* positions vers la droite.

```
SHR opérande, n
```

Exemple :

```
MOV AL,11001011B
SHR AL,1
```

→ entrée d'un 0 à la place du bit de poids fort ; le bit sortant passe à travers l'indicateur de retenue CF.

Remarque : si le nombre de bits à décaler est supérieur à 1, ce nombre doit être placé dans le registre CL ou CX.

Exemple :

Décalage de AL de trois positions vers la droite :

```
MOV CL, 3
SHR AL, CL
```

- **Décalage logique vers la gauche** (SHift Left) :

Cette instruction décale l'opérande de *n* positions vers la gauche.

```
SHL opérande, n
```

Exemple :

```
MOV   AL, 11001011B
SHL   AL, 1
```

→ entrée d'un 0 à la place du bit de poids faible ; le bit sortant passe à travers l'indicateur de retenue CF.

Même remarque que précédemment si le nombre de positions à décaler est supérieur à 1.

Chapitre V

Les mémoires

Chapitre V. Les mémoires

V-1 Introduction :

Le fonctionnement des systèmes automatisés et des systèmes de traitement de l'information nécessite le stockage d'informations qui peuvent être:

- Des instructions relatives à des programmes.
- Des valeurs numériques relatives à des données.

Ces systèmes doivent être dotés d'une fonction « ***mémoire*** » qui peut recevoir, stocker et restituer une information.

V-2 Définition :

Une mémoire est un dispositif (circuit intégré, support magnétique, etc.) qui emmagasine les informations et les restitue à la demande. Les informations peuvent être des instructions d'un programme, des données associées ou des résultats intermédiaires. Chaque mot est identifié par une adresse.

On ce limite dans ce cours par l'étude des mémoires à semi-conducteurs.

V.3 Structure générale d'une mémoire :

Un bloc mémoire peut être représenté comme sur la figure V.1. Pour pouvoir identifier individuellement chaque mot (donnée) on utilise *n* lignes d'adresse. Une ligne de commande (R/W) indique si la mémoire est accédée en écriture (l'information doit être mémorisée) ou en lecture (l'information doit être restituée). La ligne de validation ou de sélection du bloc (CS) permet de valider le bloc mémoire (l'information mémorisée se retrouve sur la ligne de sortie et permet d'inhiber ce bloc (les lignes des données sont isolées). Les lignes de données véhiculent aussi bien les données à stocker que les données lus.

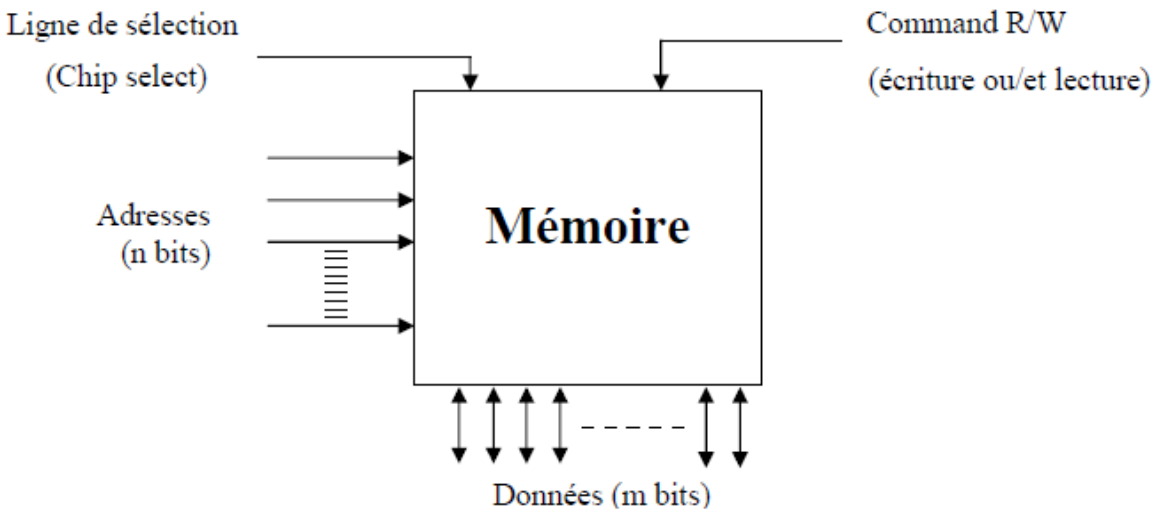


Figure V.1: Structure générale d'une mémoire

V.4 Schéma fonctionnel d'une mémoire :

Le nombre de lignes d'adresses dépend de la capacité de la mémoire : n lignes d'adresses permettent d'adresser 2^n cases mémoire :

- 8 bits d'adresses permettent d'adresser 256 octets.
- 16 bits d'adresses permettent d'adresser 65536 octets (= 64 Ko).

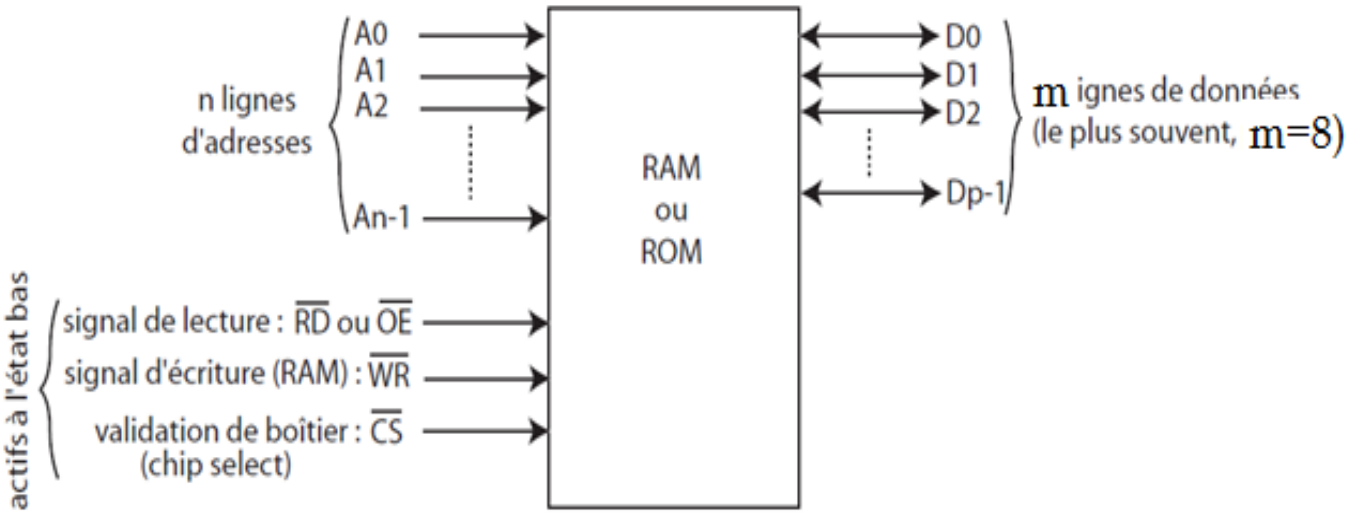


Figure V.2. Structure fonctionnelle d'une mémoire

V.5 Interfaçage microprocesseur/mémoire :

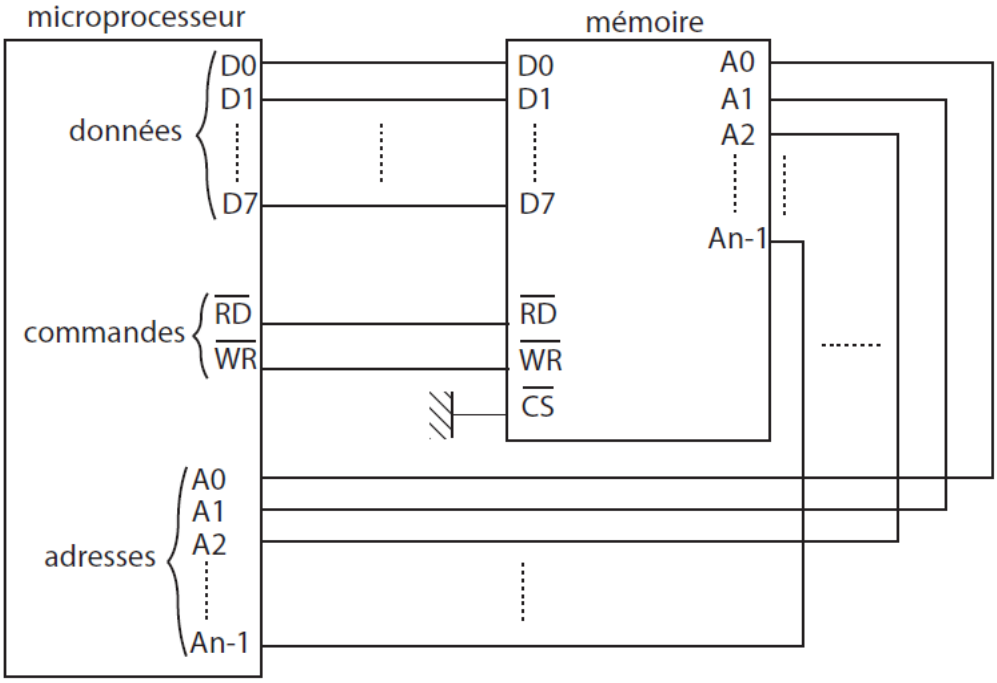


Figure V.3. Montage d'interfaçage microprocesseur/mémoire

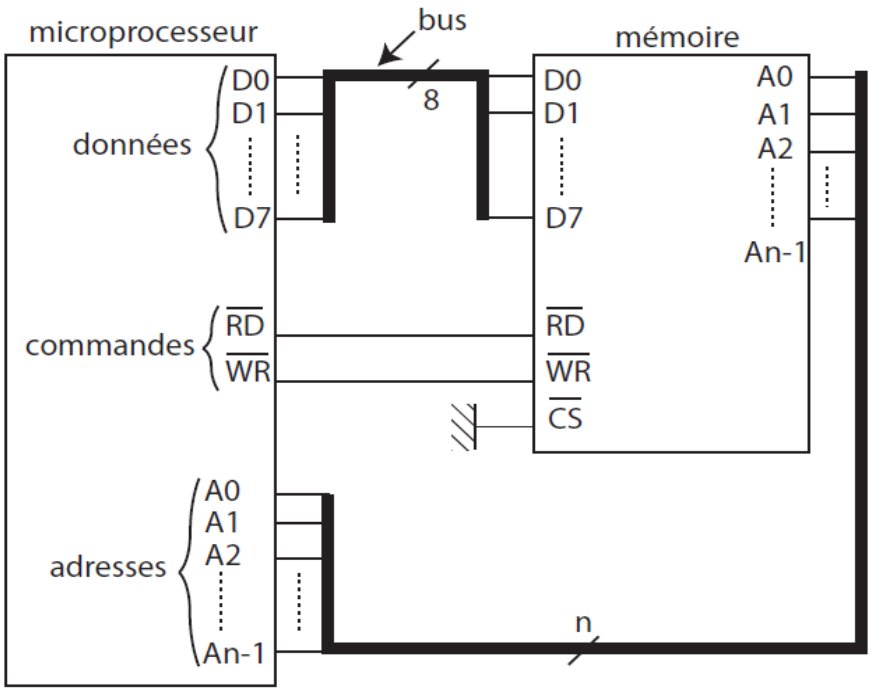


Figure V.4. Représentation condensée (plus pratique) de l'interfaçage mémoire/microprocesseur

V.6 Organisation interne de la mémoire:

Une mémoire est constituée d'un ensemble de cellules organisées sous une forme matricielle (Fig.V.5). Chacune de ces cellules peut stockée un seul bit et chaque ligne de la matrice correspond à un mot (une donnée) de m bits.

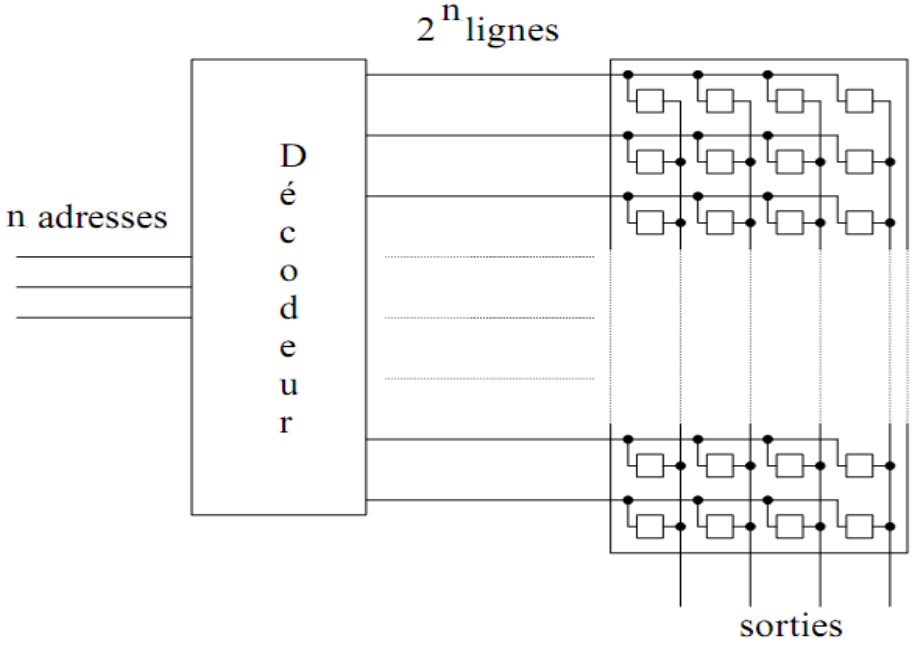


Figure V.5. L'organisation des cellules à l'intérieur d'une mémoire

V.7 Caractéristiques :

1- Format : représente le nombre de bits de la donnée (m).

2- Nombre de cases mémoires : Nombre de données pouvant être stockées (N).

$N=2^n$ (où *n* est le nombre de bits d'adresse)

3- Temps d'accès : temps d'obtention d'une information contenue dans la mémoire.

4- Le débit : c'est le nombre maximum d'informations lues ou écrites par seconde.

5- Capacité : représente le nombre total de bits peuvent être stockés :

$C=N \times m$

Pour des raisons de simplification, on exprime la capacité par les unités décrites sur le tableau suivant:

Table V.1. Unités informatiques

1 Octet = 8 bits [Octet (Français) = Byte (Anglais)] [***Attention : bit ≠ byte (=8bits)***]	
En kilo-octets (Ko)	1 Ko = 2^{10} octets = 1024 Octets
En Méga-octets (Mo)	1 Mo = 2^{20} octets = 1048576 Octets
En Giga-octets (Go)	1 Go = 2^{30} octets = 1073741824 Octets
En Tera-octets (To)	1 To = 2^{40} octets = 1099511627776 Octets

Exemple : Calcul de la capacité d'une mémoire de 8 bits de données et de 16 bits d'adresse.

Solution :

Format : m=8; *Nombre de cases mémoires :* $N = 2^{16} = 65536$

Capacité : $C = m \times N = 8 \times 65536 = 524288$ bits

Pour l'exemple précédent C = 524288 bits

C = 524288/8 = 65536Octets = 65536/1024 = 64 Ko

V.8 Différents Types de Mémoires :

Jusqu'à la fin des années 1970, on utilisait des mémoires à tores magnétiques, lentes et de faibles capacités. Actuellement, on n'utilise que des mémoires à semiconducteurs. On distingue deux grandes catégories de mémoires: les mémoires volatiles et les mémoires non-volatiles.

V-8-1 Mémoires vives (Volatiles)(RAM : Random Access Memory):

(Mémoire à Accès Aléatoire). Les mémoires vives sont à lecture et à écriture : la donnée est disponible en lecture (sortie) ou en écriture (entrée). Une valeur écrite peut être lue à n'importe quel moment par la suite. Le contenu est perdu si on coupe l'alimentation, on dit que ce type de mémoire est volatile. Dans cette catégorie de mémoires, le processus de mémorisation est effectué:

- Soit à l'aide d'une bascule de type D (**SRAM**).
- Soit à l'aide d'un micro-condensateur (condensateur grille-substrat d'un transistor MOS) (**DRAM**).

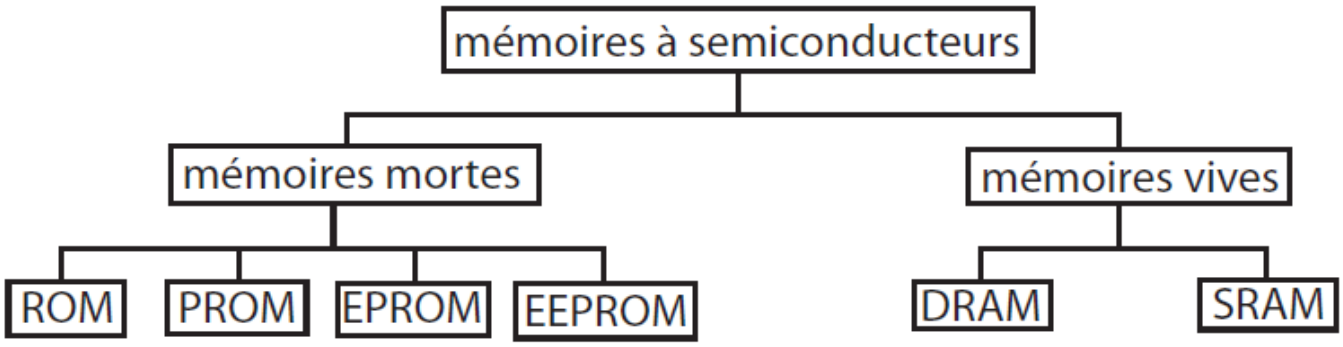


Figure V.6. Types de mémoires à semiconducteurs.

a. La mémoire RAM statique (SRAM) :

(**S**tatic **R**andom **A**ccess **M**emory -- Mémoire Statique à Accès Aléatoire). Dans laquelle les informations sont mémorisées par une ***bascule*** et conservées tant que l'alimentation est présente, elle est réalisée en technologie MOS (Metal Oxide Semiconductor). La figure V.7 montre deux structures de base différentes pour une cellule mémoire de type SRAM.

b. La mémoire RAM dynamique (DRAM) :

(**D**ynamic **R**andom **A**ccess **M**emory -- Mémoire Dynamique à Accès Aléatoire). Qui utilise un ***condensateur*** comme cellule mémoire (un bit mémorisé) de l'information. C'est-à-dire basée sur la charge de condensateurs : condensateur chargé = 1, condensateur déchargé = 0. Cette information tend à se dégrader à cause des courants de fuites, ce qui nécessite un rafraîchissement périodique (de l'ordre de la milli seconde).

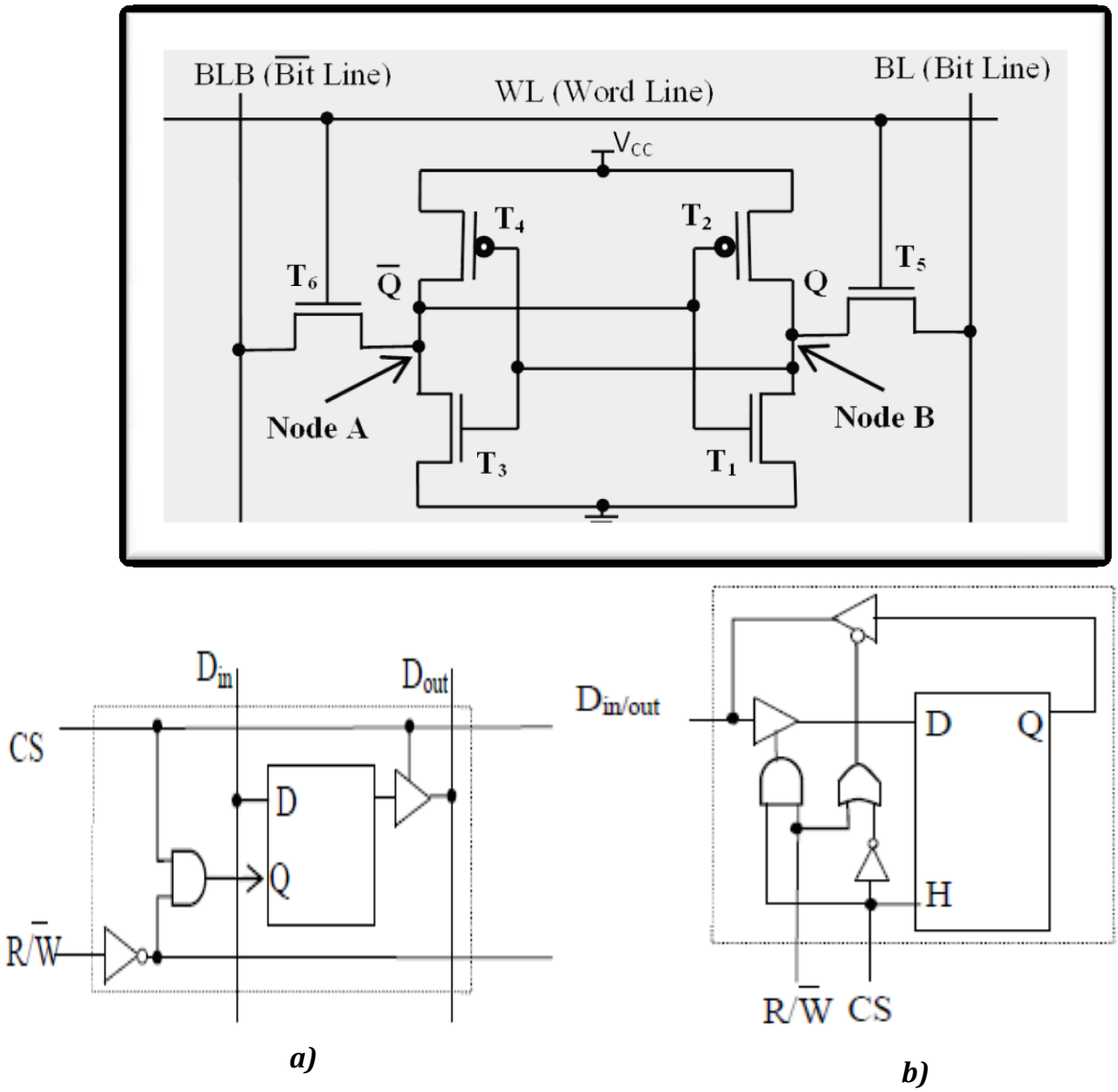


Figure V.7. Structure de base d'une cellule mémoire SRAM avec: a) entrées/sorties séparées, b) entrées/sorties non séparées.

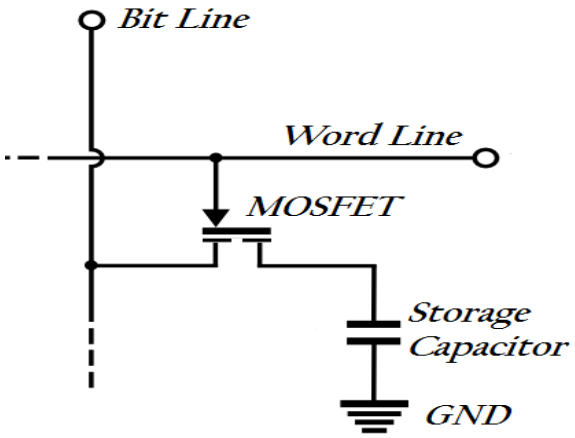


Figure V.8. Structure de la cellule DRAM.

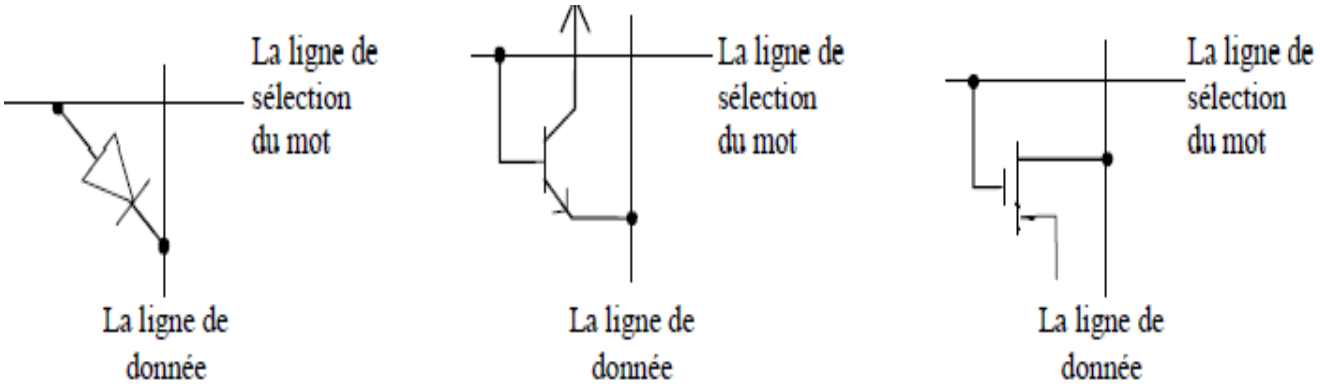
Les mémoires DRAMs, qui offrent une plus grande densité d'information et un coût plus faible, sont utilisées pour la mémoire centrale, alors que les mémoires SRAMs, plus rapides, sont utilisées lorsque le facteur vitesse est critique, notamment pour des mémoires de petite taille comme les caches et les registres.

TableV.2. Différences majeures entre SRAM et DRAM

Mémoires	Avantages	Inconvénients
SRAM	- Très rapide. - Utilisation Simple.	- Compliqué à réaliser.
DRAM	- Intégration élevée. - Faible coût. - Faible consommation.	- Nécessite un rafraîchissement périodique à cause du courant de fuite des condensateurs.

V-8-2 Mémoires mortes (Non-Volatiles) (ROM : Read Only Memory):

(Mémoire à Lecture Seule). Mémoire à lecture seule, sans écriture. Son contenu est programmé une fois pour toutes par le constructeur. C'est-à-dire la donnée n'est disponible qu'en lecture (sortie). L'inscription de données en mémoire est possible, cela s'appelle la programmation. Le contenu est permanent et ne dépend pas de l'alimentation, on dit que ce type de mémoire est non volatile.



a) à base de diode *b) à base de transistor bipolaire* *c) à base de transistor MOS*

Figure V.9. Cellule mémoire ROM

Dans cette catégorie de mémoires on trouve :

ROM Masquée (Mask ROM) :

Elle est programmée une fois, par le constructeur, lors de la fabrication. C'est-à-dire, Mask ROM est une mémoire dont le contenu, programmé lors de sa fabrication, ne peut être ni modifié, ni effacé par l'utilisateur. *Masque de fabrication à*:

- Diodes disposées sur un réseau de lignes et de colonnes.
- Ou à transistors dont sont effectuées des coupures à leurs bases.

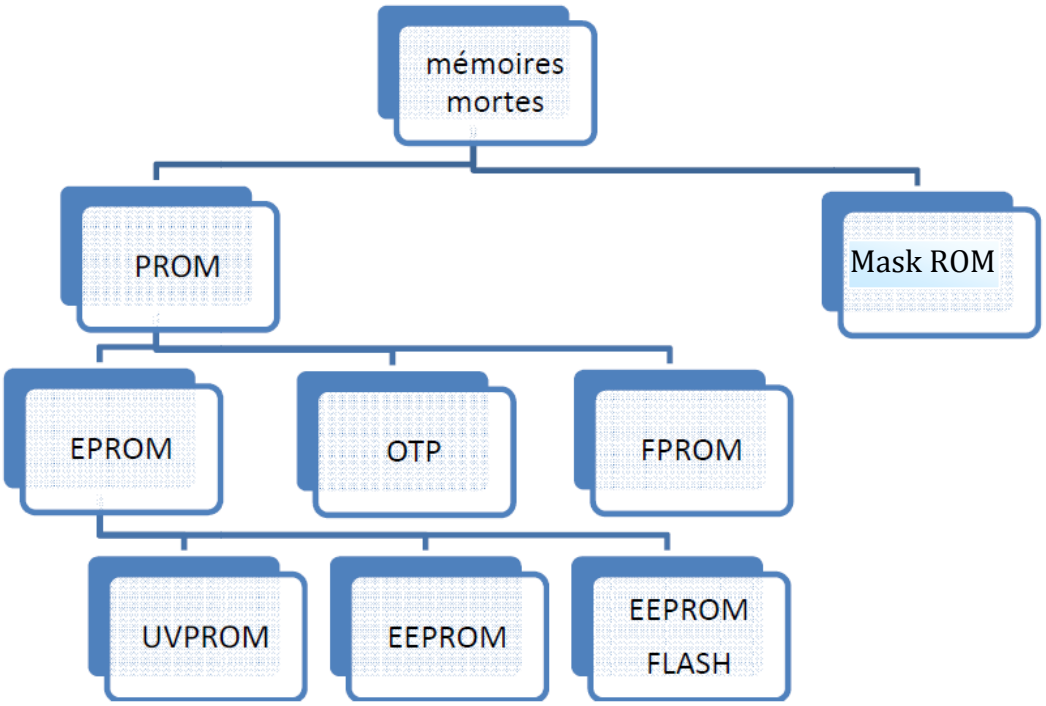


Figure V.10. Types de mémoire ROM.

Chapitre VI

Principes de l'implémentation d'un système logique synchrone par un circuit programmable

Chapitre VI. Principes de l'implémentation d'un système logique synchrone par un circuit programmable

VI-1 Introduction :

La logique programmable est la plus utilisée actuellement par les objets techniques tels que les microcontrôleurs (µc), les microprocesseurs (µp) et les mémoires, et ce, dans presque tous les domaines de notre vie (industriel, spatial, informatique, médical,…). Pour réaliser l'interfaçage entre ces objets techniques, on utilise généralement des fonctions à base de compteurs, registres, fonctions logiques élémentaires,…. Le nombre de circuits nécessaires pour effectuer ces fonctions peut devenir très vite important.

Pour diminuer les coûts de fabrication, de développement et de maintenance, les fabricants de circuits intégrés ont donné naissance aux Circuits Logiques Programmables PLD (Programmable Logic Device). Ces circuits sont capables de réaliser plusieurs fonctions logiques dans un seul circuit et de faire évoluer ces fonctions par reprogrammation, ce qui n'est pas le cas si les fonctions sont réalisées à base de circuits logiques classiques.

Les premiers brevets pour de tels composants datent des années 1980 à 1990 mais c'est au début des années 1990 qu'ils se sont généralisés pour conduire à l'apparition de composants plus performants comme les FPGA (Field Programmable Gate Array ou réseau de cellules logiques programmables) qui se sont développés plus récemment. De telles technologies sont dédiées pour développer les systèmes de traitement numérique qui requièrent des opérations en temps réel tels que les machines qui nécessitent un système de régulation ou de contrôle pour fonctionner.

VI-2 Classification des circuits numériques:

Les circuits numériques se répartissent sur trois grandes familles des circuits bien distinctes, comme le montre la figure VI.1:

VI-2-1 Les circuits standards :

Les fabricants proposent des composants standards ayant des fonctions plus ou moins complexes. L'association de ces composants sur un circuit imprimé permet de réaliser un système numérique.

Ces circuits standards se présentent sous forme de 3 critères à savoir :

a) Les fonctions simples :

- ✓ Certains circuits combinatoires de moyenne complexité MSI (Medium Scale Integration) sont considérés comme des circuits standards de base.

- ✓ On peut trouver aussi les circuits SSI (Small Scale Integration), qui réalisent des fonctions combinatoires ou séquentielles élémentaires.

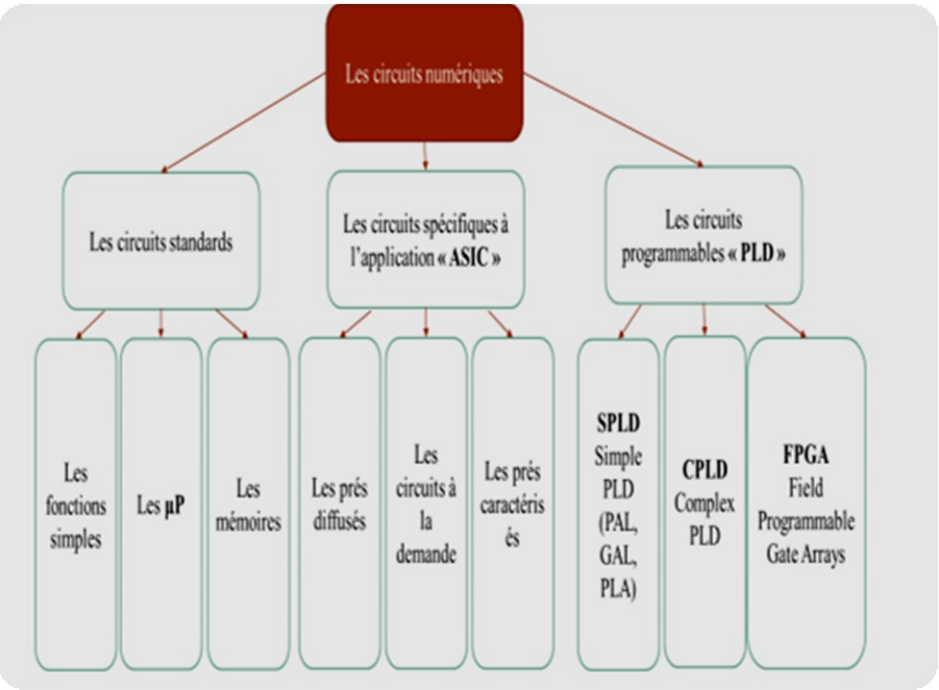


Figure VI.1. Les différentes catégories des circuits numériques

b) Les microprocesseurs :

- ✓ Processeur à usages général : Pour minimiser l'impact du coût de conception et de fabrication des circuits intégrés les plus complexes, les fabricants ont crée des circuits de traitement numérique dont l'usage final n'est pas connu à la fabrication.
- ✓ DSP (Digital Signal Processor) : Utilisés généralement dans les applications du traitement numérique du signal en temps réel. On les retrouve dans les Modems (RTC, ADSL), les téléphones mobiles, les appareils multimédia, les récepteurs GPS,…

c) Les mémoires :

Ce sont des dispositifs qui permettent de stocker puis de restituer les informations. Les mémoires à semi-conducteurs se distinguent en deux classes :

- ✓ Les mémoires vives qui sont volatiles en l'absence de l'alimentation électrique.
- ✓ Les mémoires mortes qui conservent l'information ce qui permet de les considérer comme des circuits logiques programmables.

VI-2-2 Les circuits spécifiques à l'application ASIC (Application Specific Integration Circuit) :

L'ASIC est un circuit intégré qui permet un câblage direct des applications spécifiques sur le silicium. Il constitue la troisième génération des circuits intégrés, apparu au début des années 80. L'ASIC présente une personnalisation de son fonctionnement, selon l'utilisation :

- ✓ Une réduction du temps de développement.
- ✓ Une augmentation de la densité d'intégration et de la vitesse de fonctionnement.

On distingue :

a) Les prés diffusés (Gate Array) :

Les éléments logiques existent déjà physiquement sur le circuit, seules les connexions peuvent être définies.

b) Les circuits à la demande (Full-Custom) :

Leur intérêt est de créer des circuits standards, des circuits programmables, des bibliothèques de cellules standards ou d'IP, pour utiliser les technologies les plus récentes.

Tous les éléments utilisés sont développés par le concepteur ; le fabricant réalise tous les niveaux de masque. Tout est modifiable : transistors (type, caractéristiques), connexions,…

c) Les prés caractérisés (Standard Cell) :

Le circuit est un assemblage de cellules placées/routées et les éléments logiques sont choisis dans une bibliothèque de portes, les connexions sont libres.

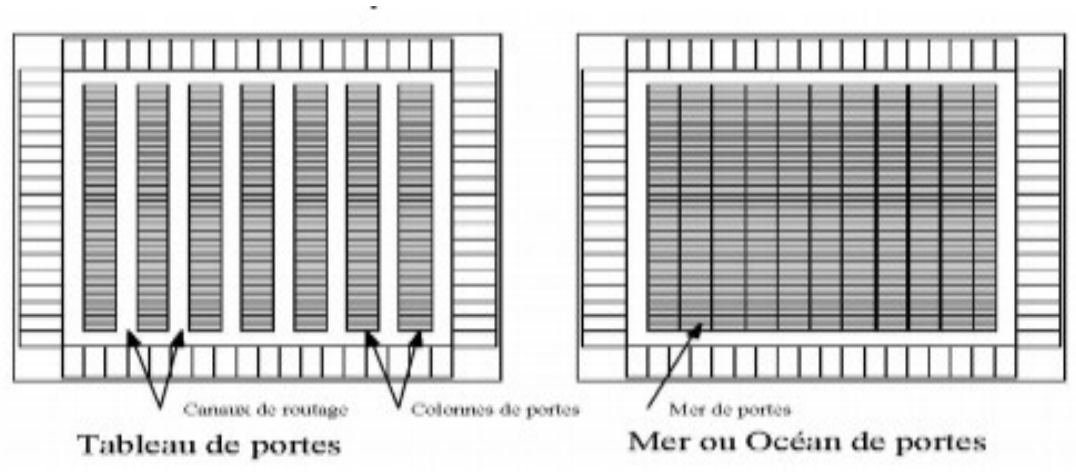


Figure VI.2. ASIC prés diffusés

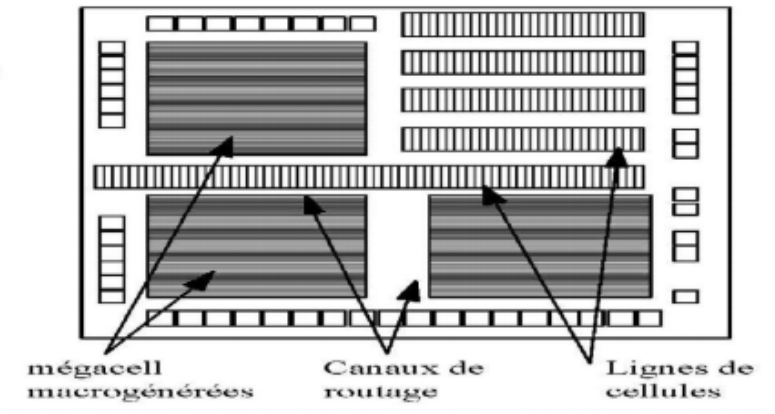


Figure VI.3. ASIC prés caractérisés

VI.2.3 Les circuits logiques programmables (PLD) :

Un PLD est un nom générique donné à l'ensemble des circuits monolithiques formés de cellules logiques (comportant des fusibles, des anti-fusibles ou de la mémoire) qui peuvent être programmés et reprogrammés par l'utilisateur. Le nom anti-fusible vient de la programmation des connexions qui s'effectue par fermeture de circuits, comparé aux fusibles où l'on ouvre les circuits.

VI.3 La logique programmable ; concepts et classification:

VI.3.1 Concepts :

Un circuit logique programmable se définit comme un composant discret contenant des modules de logique combinatoire et séquentielle dont les interconnexions sont désignées par programmation.

Notant que le terme « programmation » est utilisé fréquemment pour personnaliser les réseaux logiques reconfigurables et modifiables, donc le terme « reconfiguration » décrit mieux cette procédure au lieu du terme « reprogrammation ». Il n'y a pas de programmation en logiciel (contrairement à un microprocesseur) car les connexions ou les composants de ces circuits sont à base de portes logiques et des bascules.

VI.3.2 Structure de base d'un PLD :

Représentée sur la figure VI.4, elle se compose de :

- ✓ Un bloc d'entrée qui permet de fournir au bloc combinatoire l'état de chaque entrée et de son complément.
- ✓ Un ensemble d'opérateurs « ET » sur lesquels viennent se connecter les variables d'entrée et leurs compléments.

- ✓ Un ensemble d'opérateurs « OU » sur lesquels les sorties des opérateurs « ET » sont connectées.
- ✓ Un bloc de sortie.
- ✓ Un bloc d'entrée-sortie qui comporte une porte 3 état et une broche d'entrée/sortie.

Le bloc combinatoire programmable est formé de matrices « ET » et de matrices « OU » car toute fonction logique combinatoire peut être écrite comme somme de produit, d'interconnexions de ces matrices qui doivent être programmables par un logiciel, utilisant des fusibles qui seront grillés lors de la programmation. La figure VI.5 montre deux exemples de matrices « OU » et « ET » avant programmation et après programmation.

Une fois brulé, le fusible ne peut plus être utilisé. Pour rendre ces dispositifs réutilisables, l'effacement électrique ou par UV est utilisé aujourd'hui comme solutions technologiques pour aboutir à la reprogrammation successive.

Le bloc de sortie est souvent appelé macro-cellule que l'on nomme OLMC (Output Logic Macro Cell signifiant macro-cellule logique de sortie). Cette macro-cellule comporte :

- Une porte OU exclusif, une bascule D.
- Des multiplexeurs qui permettent de définir différentes configurations et un dispositif de bouclage sur la matrice ET.
- Des fusibles de configuration.

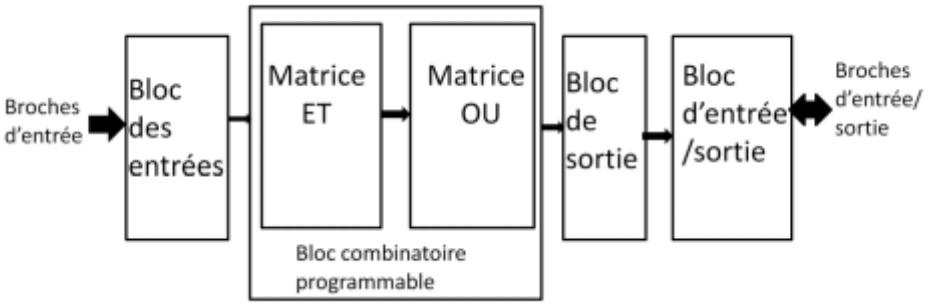


Figure VI.4. Structure de base d'un PLD

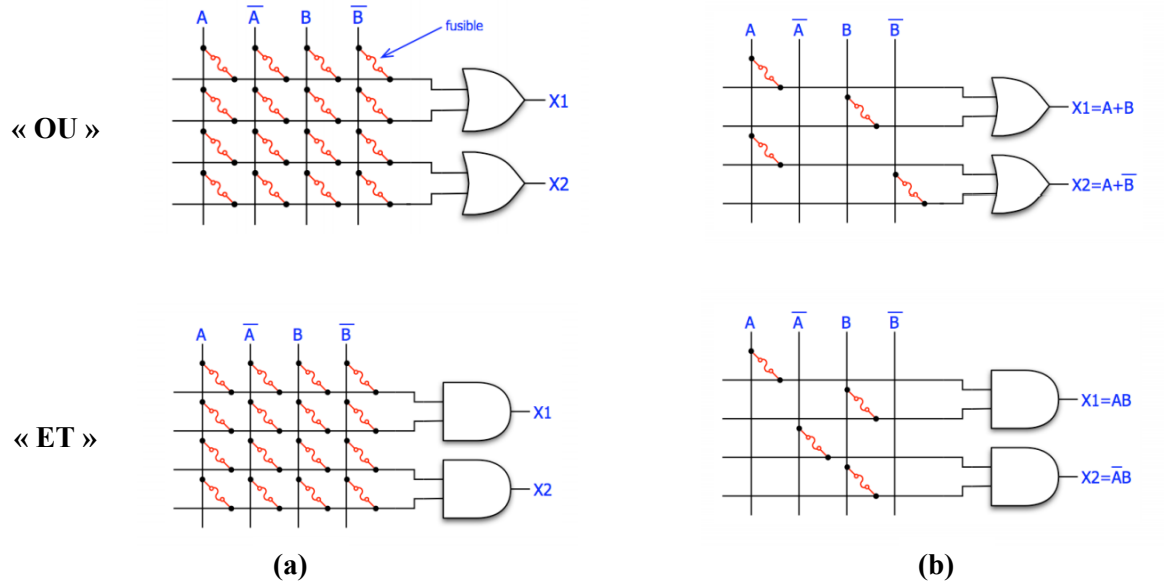


Figure VI.5. Exemples de matrices « OU » , « ET »
non programmées (a) et programmées (b)

VI.3.3 Les différentes familles des PLD :

La classification des PLD peut se révéler délicate et difficile selon les critères abordés.

On peut les classer suivant leurs structures internes à savoir : le nombre d'entrées, de sorties, de connexions programmables et le niveau d'intégration (Table VI.1). La classification peut suivre aussi le caractère programmable des matrices « ET » et « OU », ce qui permet de citer trois types de PLD :

- ✓ **PROM (Programmable Read-Only Memory)** : Matrice « ET » fixe et matrice « OU » programmable. C'est une mémoire où la matrice « ET » sert de décodeur d'adresse ; pour chaque valeur d'adresse la PROM produit une valeur qui lui a été programmée.
- ✓ **PAL (Programmable Array Logic)** : Matrice « ET » programmable suivie d'une matrice « OU » fixe.
- ✓ **PLA (Programmable Logic Array)** : Les deux matrices sont programmables.

Néanmoins, les circuits logiques programmables se classent en trois principaux circuits programmables (cités dans la figure VI.1):

- **SPLD**
- **CPLD**
- **FPGA**

Table VI.1. Différentes familles des PLD

Type	Nombre de Porte intégré	Matrice ET	Matrice OU	Effaçable
PAL	10 à 100	Programmable	Fixe	Non
GAL	10 à 100	Programmable	Fixe	Electriquement
EPLD	100 à 3000	Programmable	Fixe	U-V
FPLA	2000 à 3000	Programmable	programmable	Eclectiquement
FPGA	Plus de 50.000	Programmable	programmable	Électriquement

VI.4 Les SPLD (Simple PLD) :

Les SPLD sont des circuits logiques programmables élémentaires constituées : d'un ensemble de portes « ET » sur lesquelles viennent se connecter les variables d'entrée et leurs compléments, ainsi qu'un ensemble de portes « OU » sur lesquelles les sorties des opérateurs « ET » sont connectées (figure VI.6).

Il existe trois types des SPLD :

VI.4.1 Les PAL :

Les PAL (Programmable Array Logic ou réseau logique programmable) sont développés au début des années 70 par MMI (ex-AMD) à base des matrices « ET » programmables et des matrices « OU » fixes. Leur programmation se fait par destruction de fusibles, ce qui fait qu'aucun fusible n'est grillé à l'achat de la PAL.

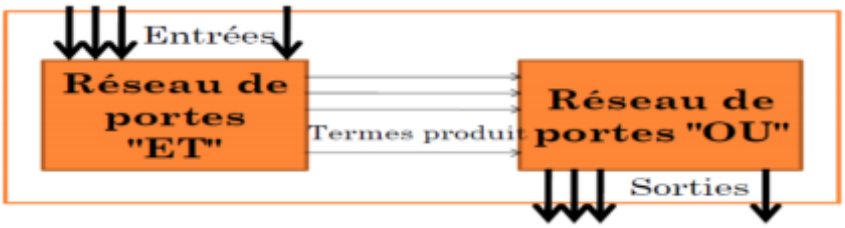


Figure VI.6. Architecture globale d'un SPLD

On distingue parmi les PAL plusieurs appellations :

- ✓ **Les PAL combinatoires (simples)** : Constitués de fonctions de logique combinatoire.
- ✓ **Les PAL séquentiels** qui se classent en :
  - **PAL à registres** constitués de logique combinatoire et séquentielle (registre).
  - **PAL asynchrone registre** où le signal de l'horloge n'est pas le même pour toutes les bascules. Ces circuits sont munis d'un multiplexeur qui permet de shunter le registre.
  - **PAL versatiles** plus évolués.

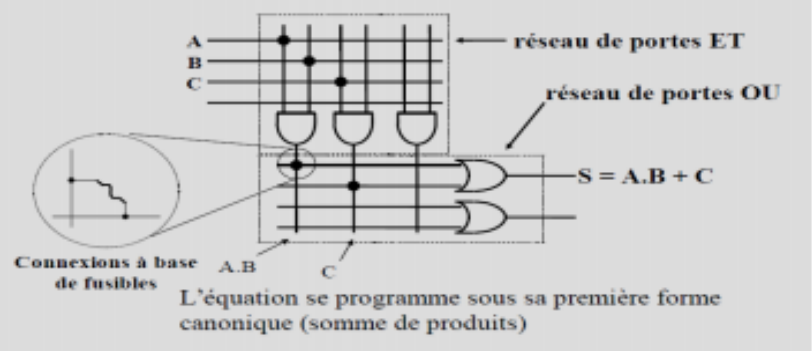
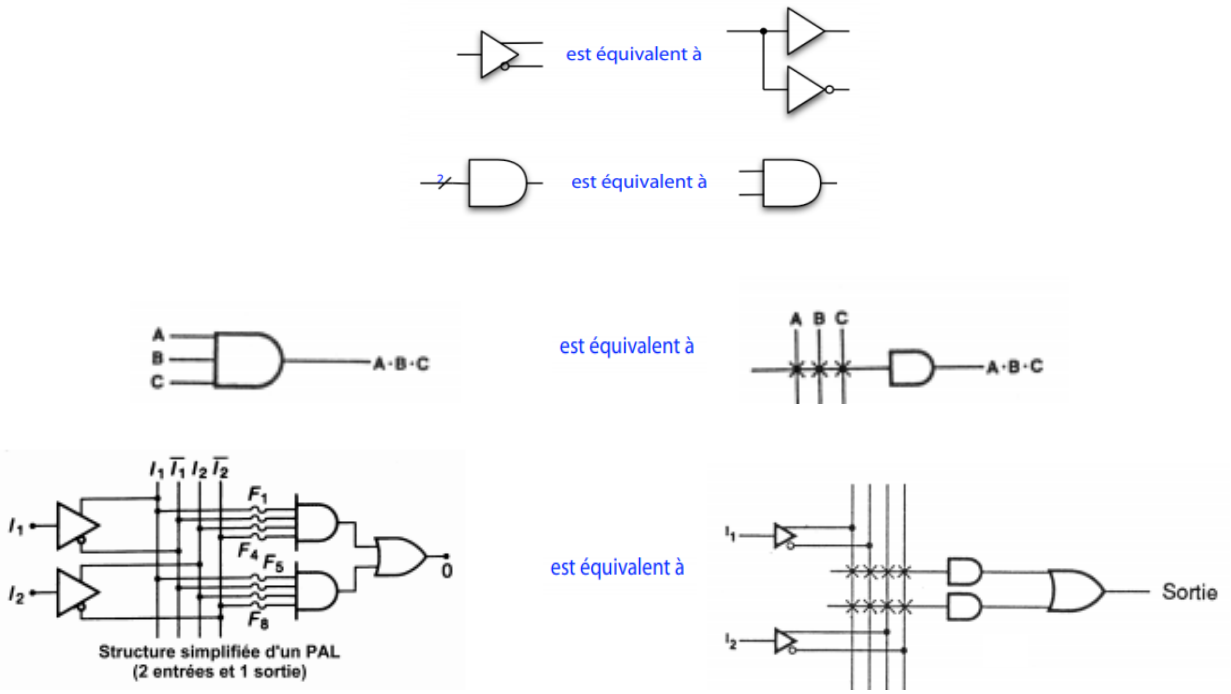


Figure VI.7. Architecture d'un PAL

La représentation schématique des PAL demande beaucoup d'espace pour représenter un PAL en entier. Les industriels ont adopté une autre représentation en simplifiant les connexions par les conventions suivantes :



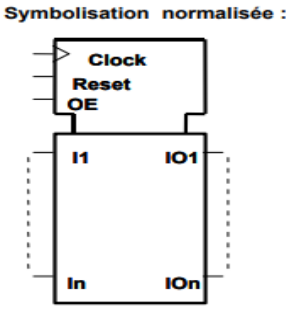


Figure VI.8. Symbolisation et représentation des PAL

VI.4.2 Les GAL :

Les GAL (Generic Array Logic ou réseau logique générique), nommés aussi PAL CMOS sont des PAL effaçables par UV ou électriquement. Pour remédier à l'inconvénient majeur des PAL (une seule programmation), les fusibles irréversibles sont remplacés par des transistors CMOS qui peuvent être régénérés. La consommation des GAL est beaucoup plus faible que les PAL à fusibles bipolaires.

VI.4.3 Les PLA :

Les PLA (Programmable Logic Array ou tableau de logique programmable) diffèrent des dispositifs ultérieures dans la mesure où les fonctions de porte « ET » et « OU » sont programmables.
La table VI.2 récapitule les trois technologies des circuits SPLD.

VI.5 Les CPLD (Complex PLD) :

Ces circuits logiques programmables peuvent être vus comme une intégration de plusieurs PLD simples reliés entre eux par une matrice d'interconnexion (Programmable Interconnect PI). Ils ont une capacité en nombre de portes et en possibilités de configuration très supérieure à celle des PALS (SPLD), puisqu'on la possibilité de configurer les interconnexions entre les blocs en plus de la configuration des SPLD.

Table VI.2. Comparaison entre les SPLD

Type	Plan ET	Plan OU	Technologie	Utilisation classique
PAL	Programmable	Fixe	Bipolaire	Décodage, machine à état
GAL	Reprogrammable	Fixe	CMOS	Décodage, machine à état
PLA	Programmable	Programmable	CMOS	Fonctions logiques complexes

VI.6 Les FPGA (Field Programmable Gate Array):

A la différence des CPLD, les FPGA (réseau de cellules logiques programmables) sont assimilables à des ASIC programmables par l'utilisateur. En 1984 Xilinx a présenté le FPGA, un circuit prés diffusé programmable dont le concept est basé sur l'utilisation d'une unité appelée LUT (Look Up Table) comme élément combinatoire de la cellule de base.

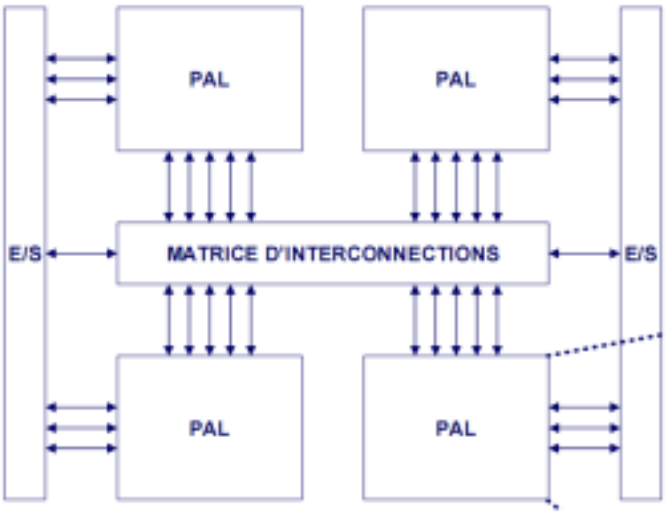


Figure VI.9. Schéma d'un CPLD

Les FPGA sont constitués d'une matrice de blocs logiques programmables (**CLB** : Configurable Logic Block) ayant plusieurs milliers de portes et entourés de blocs d'entrée-sortie programmable (**CIOB** : Configurable Input Output Block). L'ensemble est relié par un réseau d'interconnexions programmable (Figure VI.10).

Les connexions programmables sur ce réseau sont réalisées par des transistors MOS dont l'état est contrôlé par des cellules mémoires SRAM (Static Random Access Memory). Ainsi, toute la configuration d'un FPGA est contenue dans ces cellules.

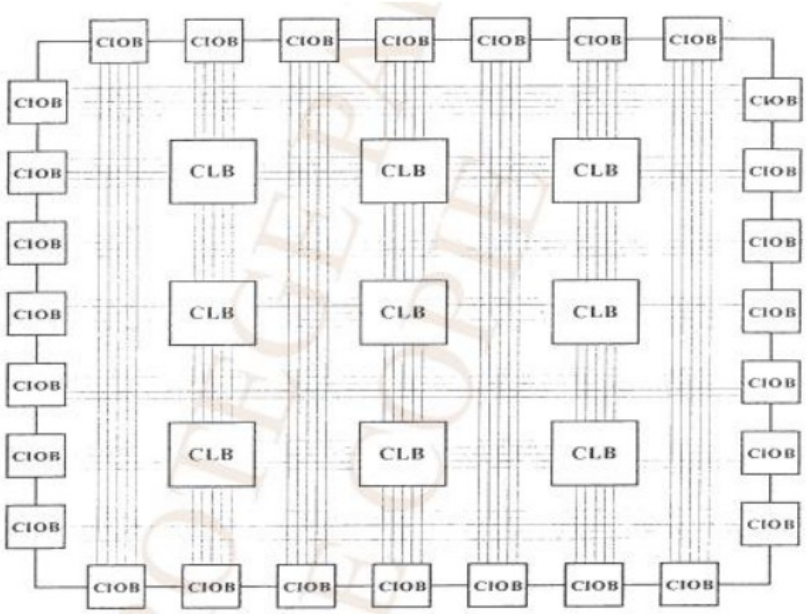


Figure VI.10. Architecture interne des FPGA

VI.6.1 Structure des CLB :

Les blocs logiques configurables (nommés aussi macro-cellules ou cellules logiques ou éléments logiques) sont constitués de parties combinatoires et séquentielles (LUT, bascule et multiplexeur, figure VI.11). La structure de LUT est caractérisée par une table de vérité ainsi qu'une mémoire pour créer n'importe quelle fonction logique combinatoire et router les différentes interconnexions entre les entrées et sorties des circuits FPGA. On l'appelle aussi générateur de fonctions.

Une LUT de 16 bits permet de réaliser n'importe quelle fonction combinatoire 4 vers 1. A chaque LUT peut être adjoint un registre piloté par une horloge et un multiplexeur.

Chaque fabricant de FPGA (Xiling, Altera, Actel, Texas Instrument,…) a réalisé des regroupements de LUT et d'ajout de capacité supplémentaire.

La figure VI.12 représente une structure détaillée d'un CLB SPARTAN.

VI.6.2 Structure des CIOB :

Chaque bloc IOB contrôle une broche du composant et peut être défini en entrée, en sortie, en entrée/sortie ou être inutilisé. Le rôle principal des interfaces d'entrée/sortie est de transmettre et de recevoir des données. Néanmoins, l'interface peut être dotée d'options telles que des registres, impédances et buffers.

Les IOB sont constitués de registres, de diviseurs de tensions, de résistances de rappel pull up et autres ressources spécifiques (figure VI.13).

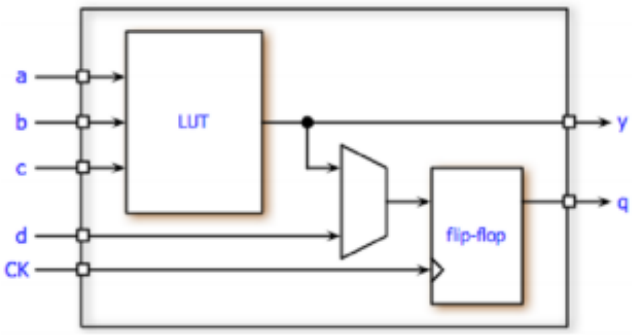


Figure VI.11. Un élément logique de l'FPGA

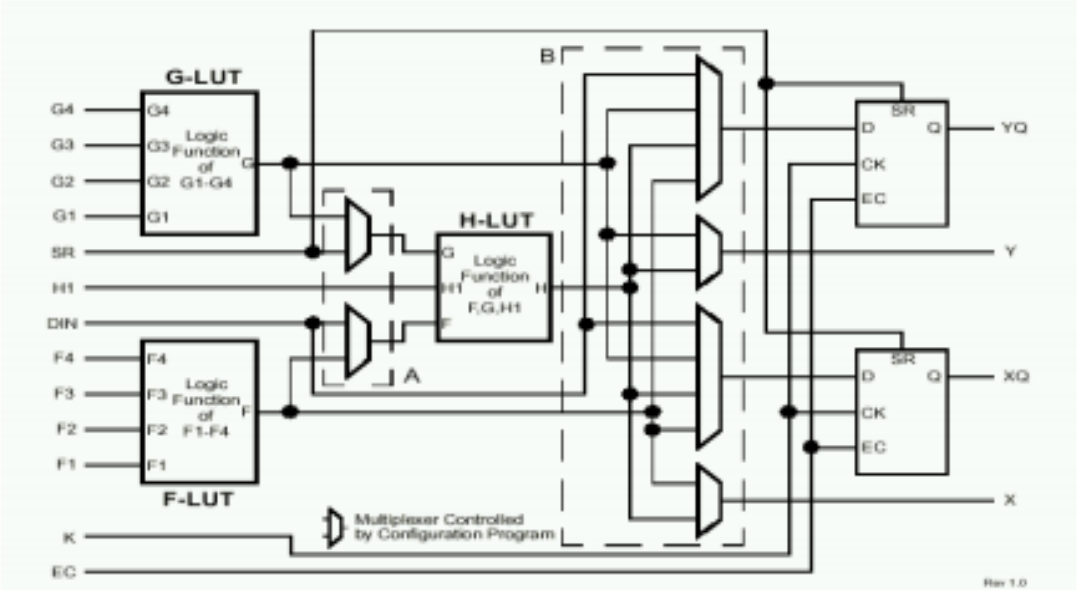


Figure VI.12. Structure d'un CLB SPARTAN

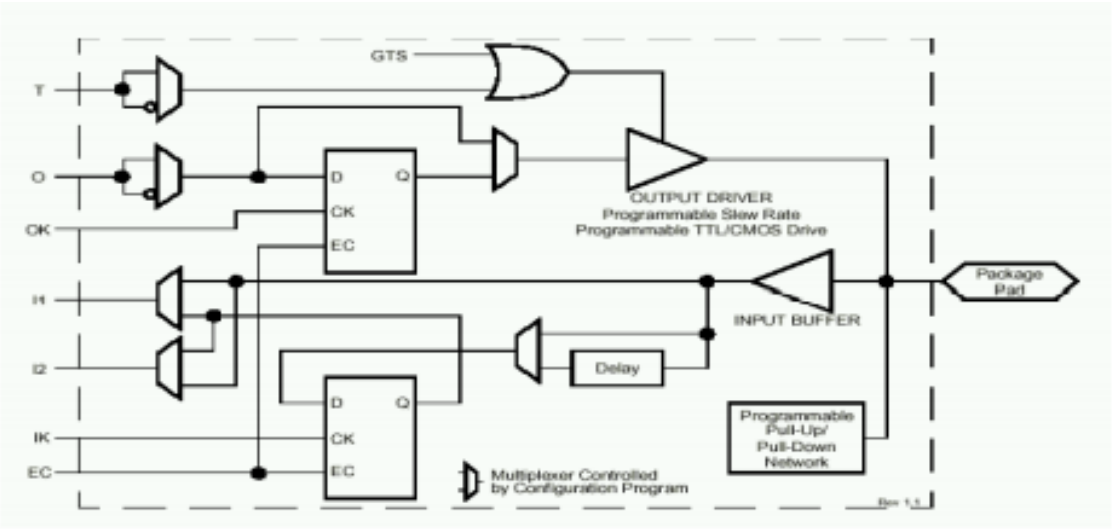


Figure VI.13. Structure d'un IOB SPARTAN

VI.7 Caractérisation des Circuits Logiques Programmables (PLD) :

Les PLD sont caractérisés par :

- ✓ Le nombre d'entrées
- ✓ Le nombre de sorties
- ✓ Le nombre de termes produits par sortie
- ✓ Le retard de propagation
- ✓ La consommation de puissance
- ✓ La technologie

La plupart de ces paramètres apparaissent dans le nom du circuit, par exemple :

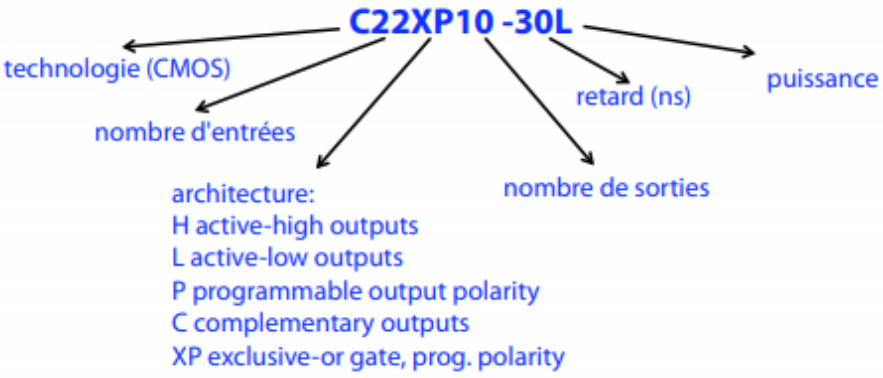


Figure VI.14. Caractéristiques d'un circuit PLD

VI.8 Programmation ou configuration des circuits logiques (FPGA):

Le processus de développement d'une application utilisant un circuit programmable comme le FPGA permet le câblage par programmation de la structure interne du circuit. Ce processus qui définit l'implémentation du circuit numérique passe par un certain nombre d'étapes allant de la description du système à la programmation du composant (figure VI.15) :

VI.8.1 Description de l'application (sous fichier source) :

Cette étape permet de réaliser un fichier source qui renferme les données descriptives ou de configuration du circuit à réaliser. Cela se fait par un langage de description matérielle HDL (Hardware Description Language) qui permet de matérialiser les structures électroniques de différents circuits. Les plus anciens HDL (dits de bas niveau) sont ABEL, PALASM, ORCAD/PLD. Les langages dits de haut niveau, Verilog et VHDL (Very High speed HDL) sont crées pour le développement des circuits intégrés logiques complexes tels que les FPGA. Le but de ces derniers HDL est double :

- ✓ **La simulation** : où la représentation du circuit devient exécutable.
- ✓ **La synthèse ou compilation** : qui transforme la description du circuit en une netlist de portes logiques.

VI.8.2 Compilation (traduction et optimisation) :

Une description VHDL subit toute une suite de transformations avant que le circuit soit effectivement configuré. C'est le rôle du synthétiseur qui consiste à traduire la description du circuit à réaliser en bloc disponibles dans la technologie utilisée. Il applique aussi diverses optimisations logiques au circuit et produit enfin une netlist.

VI.8.3 Placement-routage :

Le netlist est ensuite adapté aux primitives logiques du circuit PLD cible comme le FPGA. Cette étape de ciblage technologique permet de passer d'une description structurelle à une description spécialisée. La tache de placement/routage consiste à disposer et connecter les ressources logiques nécessaires à l'implémentation sur la surface du circuit.

Des contraintes spatiales ou temporelles peuvent être données au placeur/routeur pour fixer par exemple la position des ports d'entrée/sortie ou bien une période d'horloge maximale, ce qui est indispensable pour les systèmes d'exploitation en temps réel RTOS (Real Time Operating System).

VI.8.4 Simulation :

Il faut s'assurer que le circuit fonctionne correctement avant de le configurer sur le système.

VI.8.5 Configuration (programmation) :

La dernière étape avant la programmation consiste à générer un bitstream : c'est un fichier contenant tous les bits de configuration du circuit. La configuration ce fait par le téléchargement de ce fichier sur le circuit par une interface dédiée.

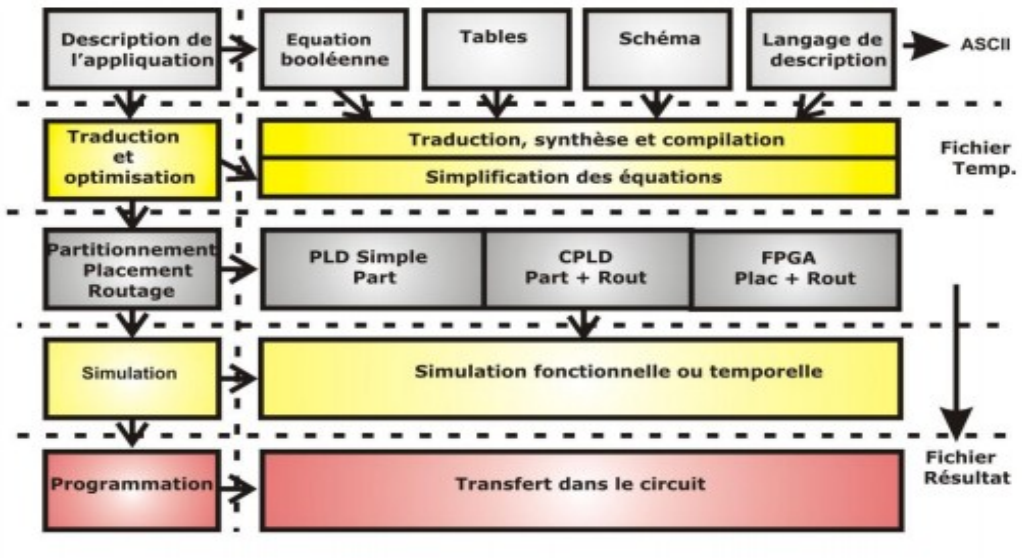


Figure VI.15. Etapes du développement pour un circuit logique

VI.9 Environnement de programmation :

L'outil de développement des circuits programmables est un logiciel de programmation défini comme étant un environnement intégré de développement des systèmes numériques. Son objectif est l'implantation matérielle sur les PLD complexes tels que les FPGA.

Exemples: Xillinx ISE, Altera Quartus, Lattice ISP Level, Altium Designer, …

Le logiciel intègre différent outils permettant de passer à travers tout le flot de conception d'un système numérique. En effet, il dispose de:

- ✓ Editeur de textes, de schémas et de diagramme d'états.
- ✓ Compilateur.
- ✓ simulateur.
- ✓ Outil pour la synthèse et la vérification.
- ✓ bibliothèque IEEE.
- ✓ Outil pour la gestion des contraintes temporelles.
- ✓ Outil pour l'implantation sur FPGA.

Annexe

Programme d'enseignement de la matière

Calculateurs et Interfaçage

Semestre : 5

Unité d'enseignement : UEM 3.1

Matière :Calculateurs et interfaçage

VHS : 37h30 (cours : 1h30, TP : 1h00)

Crédits : 3

Coefficient : 2

Objectifs de l'enseignement :

Le traitement numérique du signal exige, aujourd'hui, une implémentation matérielle en temps réel. Les circuits programmables sont à portée de main. Mais leurs utilisations nécessitent une maîtrise parfaite par le spécialiste. L'étudiant doit donc commencer par maîtriser les fondements de base des systèmes à microprocesseurs suivie par une étude détaillée sur l'exploitation des cartes à microprocesseurs 16 bits.

Connaissances préalables recommandées :

L'électronique combinatoire et séquentielle.

Contenu de la matière :

Chapitre 1. Approche des circuits programmables (1 semaine)
Architecture de base, Modèle de Von Neumann, l'unité centrale, la mémoire principale, les interfaces d'entrées/sorties, les bus, décodage d'adresses

Chapitre 2. Architecture d'un microprocesseur 16 bits (5 semaines)
Architecture interne, Brochage, Registres spéciaux, Modes d'adressages, Jeux d'instructions, Différentes architectures : RISC, CISC, Harvard

Chapitre 3. Etude générale des interfaces d'entrées-sorties (3 semaines)
Descriptions générales des circuits PIO, USART, Timer (brochage, architecture interne, modes de fonctionnement simplifié).

Chapitre 4. Les échanges de données (2 semaines)
Généralités, Protocoles d'échanges de données (par test du bit d'état du périphérique (polling), par interruption, par accès direct en mémoire).

Chapitre 5. Les mémoires (2 semaines)
Organisation d'une mémoire, caractéristiques d'une mémoire, différents types de mémoire RAM et ROM, critères de choix d'une mémoire, notion de hiérarchie mémoire, les mémoires caches.

Chapitre 6. Principes de l'implémentation d'un système logique synchrone par un circuit programmable (2 semaines)
Configuration d'un circuit programmable, Description, RTOS : system temps réel pour des applications industrielles

TP Calculateurs et interfaçage

TP1 : Initiation au Kit du microprocesseur et programmation,
TP2 : Opérations arithmétiques et logiques,
TP3 : Boucles et structures de contrôle,
TP4 : Les sous-programmes,