

République Algérienne Démocratique et Populaire

Ministère de L'Enseignement Supérieur et de la Recherche Scientifique



FACULTE DE TECHNOLOGIE

DEPARTEMENT D'ELECTRONIQUE

SYSTEMES A MICROPROCESSEURS

Cours & Exercices

Support de cours destiné aux étudiants de la 3ème année Licence en Electronique

Préparé par :

Dr Nabil BOUKHENNOUFA

Préface

Un microprocesseur est un circuit logique programmable. Pour comprendre son fonctionnement, ce support de cours exige de l'étudiant de la troisième année Licence en Electronique d'avoir des connaissances préalables dans le domaine de l'électronique numérique et des notions de base acquises en algorithmiques. L'étude d'un microprocesseur demande des concepts à deux niveaux ; des concepts hardware et software : allant de la structure électronique jusqu'à la programmation. La structure hardware est la structure physique du microprocesseur, sa programmation le rend un composant vivant.

Cependant, ce support de cours présente l'approche hardware et software du microprocesseur 8 bits de la firme Intel connu sous le nom de 8085. Ce dernier qui est actuellement un outil pédagogique très simple et très utile pour les étudiants débutants dans le domaine des microprocesseurs.

Dans ce support de cours, j'ai essayé de construire un ensemble d'initiations qui comprennent une majeure partie sur le microprocesseur 8085 et ces interfaces et leurs programmations en assembleur. J'ai voulu mettre au clair ce qui me semble constituer le minimum de connaissances que doit posséder un étudiant en cycle universitaire "3ème année Licence en Electronique".

Tous les points évoqués sont loin d'englober la liste des connaissances à posséder dans ce domaine. Elles me paraissent cependant utiles pour constituer un solide bagage aux étudiants qui seraient amené à aborder des conceptions à base de microprocesseurs.

Ce support de cours donne dans la première partie un aperçu historique et des définitions de quelques concepts de base dans le domaine des systèmes à microprocesseurs.

Dans la deuxième partie, une étude a été abordée sur les types de mémoires, leurs technologies et leurs adressages. Aussi, une description détaillée a été donnée concernant le mode de fonctionnement du système à microprocesseur 8085. La gestion des interruptions et des interfaces d'entrées / sorties (Interfaces parallèle 8255A, interface série 8250, Timer 8254 et contrôleurs d'interruption PIC 8259) est une tâche plus ou moins complexe, elle nécessite plus de concentration surtout pendant la configuration des registres des interfaces.

Ce manuscrit contient des exemples permettant de bien comprendre l'application du microprocesseur 8085 dans les systèmes électroniques.

A la fin de ce support de cours, j'ai essayé de constituer une base d'un minimum d'exercices non résolus en laissant aux étudiants l'esprit d'initiative afin d'effectuer la conception et l'analyse des systèmes à base de microprocesseurs 8085.

Dr Nabil BOUKHENNOUFA

TABLE DES MATIERES

CHAPITRE I : LES REGISTRES

I. Généralités.....	1
II. Registre de mémorisation.....	1
II.1. Registre à écriture parallèle / lecture parallèle (Parallel In Parallel Out : PIPO).....	1
II.2. Registre à écriture série / lecture série (Serial In Serial Out : SISO).....	2
II.3. Registre à écriture série / lecture parallèle (Serial In Parallel Out : SIPO).....	2
II.4. Registre à écriture parallèle / lecture série (Parallel In Serial Out : PISO).....	2
III. Registres à décalage.....	3
III.1. Décalage à droite.....	3
III.2. Décalage à gauche.....	3
III.3. Registre à décalage en anneau ou registre à décalage circulaire (rotation).....	4
III.4. Registre à décalage universel bidirectionnel 4 bits "74194".....	4
IV. Compteurs.....	6
IV.1. Classification des compteurs.....	6
IV.2. Modes de comptage.....	6
IV.3. Exemple d'un compteur binaire synchrone à cycle complet à 3 bits.....	7

CHAPITRE II : LES MEMOIRES A SEMI CONDUCTEURS

I. Identification de la fonction.....	9
II. Architecture d'une mémoire.....	9
III. Types de mémoires intégrés.....	9
III.1. Mémoires mortes ROM (Read Only Memory).....	9
III.2. Mémoires vives (RAM : Random Access Memory).....	11
IV. Structure d'une mémoire à accès direct.....	12
V. Caractéristiques d'une mémoire.....	12
V.1. Capacité.....	12
V.2. Format.....	12
V.3. Vitesse.....	13
V.4. Consommation.....	13

V.5. Brochage et alimentation du boîtier.....	13
V.6. Cycle de fonctionnement.....	13
VI. Assemblage de boîtiers mémoire.....	14
VI.1. Augmentation de la taille des mots.....	15
VI.2. Augmentation du nombre de mots.....	15

CHAPITRE III : HISTORIQUE ET EVOLUTION DES ORDINATEURS

I. Historique des ordinateurs.....	16
II. Catégories d'ordinateurs.....	17
II.1. Superordinateurs.....	17
II.2. Ordinateurs centraux.....	17
II.3. Mini-ordinateurs.....	17
II.4. Micro-ordinateurs.....	18
III. Organisation d'un ordinateur.....	18
III.1. Unité centrale de traitement (Central Processing Unit : CPU).....	18
III.2. Mémoire.....	19
III.3. Unités d'entrées / sorties.....	19
III.4. Bus du système.....	19
III.4.1. Bus de données.....	20
III.4.2. Bus d'adresses.....	20
III.4.3. Bus de commandes et de contrôle.....	20
IV. Périphériques.....	20
IV.1. Périphériques d'entrée.....	20
IV.2. Périphériques de sortie.....	21
V. Microprocesseur.....	21
VI. Notion de programme.....	22
VII. Structures Von Neuman et Harvard.....	22
VII.1. Structure de Von Neuman.....	22
VII.2. Structure de Harvard.....	23

CHAPITRE IV : ARCHITECTURE ET FONCTIONNEMENT D'UN MICROPROCESSEUR

I. Historique.....	24
--------------------	----

II. Définition d'un microprocesseur.....	25
III. Fréquence de fonctionnement.....	25
IV. Architectures CISC et RISC.....	26
V. Architecture d'un microprocesseur.....	26
V.1. Registres.....	26
V.1.1. Accumulateur.....	26
V.1.2. Registre d'état (Flags).....	27
V.1.3. Compteur de programme.....	27
V.1.4. Registre d'instruction.....	27
V.1.5. Décodeur d'instruction.....	27
V.1.6. Registres d'adresses.....	28
V.1.7. Pointeur de pile.....	28
V.2. Unité arithmétique et logique (UAL).....	28
V.3. Unité de contrôle et commande.....	29
VI. Fonctionnement d'un système à base de microprocesseur.....	29
VI.1. Ecriture en mémoire (WRITE).....	29
VI.2. Lecture de la mémoire (READ).....	29
VI.3. Communication avec les entrées/sorties.....	30
VI.4. Interruptions.....	30
VI.5. Accès direct à la mémoire (DMA).....	31

CHAPITRE V : ETUDE D'UN MICROPROCESSEUR

I. Historique.....	32
II. Rôle du microprocesseur.....	32
III. Etude du microprocesseur 8085.....	33
IV. Architecture externe du 8085.....	33
IV.1. Bus d'adresses et bus de données.....	33
IV.2. Signaux de contrôle et d'état.....	35
IV.3. Alimentation et générateur de fréquence.....	35
IV.4. Signaux d'interruptions.....	35
IV.5. Ports d'E/S sériels.....	36
V. Démultiplexage des bus AD ₇ -AD ₀	36
VI. Architecture interne du 8085.....	37

VI.1. Registres de base du 8085.....	37
VI.2. Format d'une instruction.....	38
VI.3. Format de l'opcode.....	40
VI.4. Modes d'adressage.....	40
VI.4.1. Adressage par registre.....	41
VI.4.2. Adressage immédiat.....	41
VI.4.3. Adressage direct.....	42
VI.4.4. Adressage indirect.....	42
VII. Jeu d'instructions du 8085.....	43
VII.1. Instructions de transfert.....	43
VII.2. Instructions arithmétiques.....	45
VII.3. Instructions logiques.....	49
VII.4. Instructions logiques de rotation.....	51
VII.5. Instructions de comparaison.....	52
VII.6. Instructions de branchement.....	53
VII.6.1. Saut inconditionnel.....	53
VII.6.2. Sauts conditionnels.....	54
VII.6.3. Boucles en assembleur.....	55
VII.6.3.1. Boucles infinies.....	55
VII.6.3.2. Boucles conditionnelles.....	56
VII.7. Implémentation de la pile.....	56
VII.8. Sous programmes.....	58
VII.9. Appels conditionnels et instructions de retour.....	59
VIII. Cycle d'exécution d'une instruction.....	60
IX. Génération des délais.....	61
IX.1. Génération d'un délai utilisant un registre 8 bits.....	62
IX.2. Génération d'un délai utilisant un registre pair.....	62
IX.3. Génération d'un délai utilisant deux boucles.....	63
X. Exemple d'un programme assembleur d'un générateur du signal carré.....	63
XI. Lignes de transmission série du microprocesseur 8085 SOD et SID.....	64
XI.1. Ligne de transmission SOD.....	64
XI.2. Ligne de réception SID.....	65

CHAPITRE VI : LES INTERFACES D'ENTREES SORTIES

I. Généralités.....	67
II. Microprocesseur et circuits d'interfaces.....	67
III. Différents types d'interfaces.....	68
III.1. Interface périphérique programmable 8255A.....	68
III.1.1. Brochage du 8255A.....	68
III.1.2. Programmation du 8255A.....	71
III.1.3. Exemple d'application du 8255A.....	77
III.2. Interface série.....	80
III.2.1. Principe d'une interface série.....	81
III.2.2. Connexion de deux équipements par une liaison série RS232.....	81
III.2.3. Mise en œuvre d'une interface série, l'UART 8250.....	82
III.2.4. Ports USB (Universal Serial Bus).....	84
III.3. Timer (Temporisateur), Intel 8254.....	85

CHAPITRE VII : LES INTERRUPTIONS

I. Définition d'une interruption.....	89
II. Prise en charge d'une interruption par le microprocesseur.....	89
III. Interruptions du 8085.....	90
III.1. Interruptions logicielles (Instruction RST "Restart").....	91
III.2. Interruptions matérielles.....	91
III.2.1. Interruption non vectorisée INTR.....	91
III.2.2. Interruptions vectorisées.....	91
III.2.2.1. TRAP.....	91
III.2.2.2. RST 7.5, 6.5, et 5.5.....	91
IV. Contrôleur programmable d'interruptions, PIC 8259.....	93
IV.1. Initialisation du PIC 8259.....	97
IV.2. Registres de commande.....	99

SERIES D'EXERCICES

BIBLIOGRAPHIE

CHAPITRE I

LES REGISTRES

I. Généralités

Un registre est un ensemble de mémoire élémentaire (bascules) qui servent à enregistrer ou à modifier des combinaisons binaires appelées mots ou mots binaires. On distingue deux types de registres: Les registres de mémoire ou enregistrements et les registres à décalage.

II. Registre de mémorisation [1,2]

Les registres de mémorisation peuvent être classés selon la méthode d'écriture de données ou de lecture.

II.1. Registre à écriture parallèle / lecture parallèle (Parallel In Parallel Out : PIPO)

Tous les bits sont transmis et mémorisés en même temps dans ce type de registre (figure I.1).

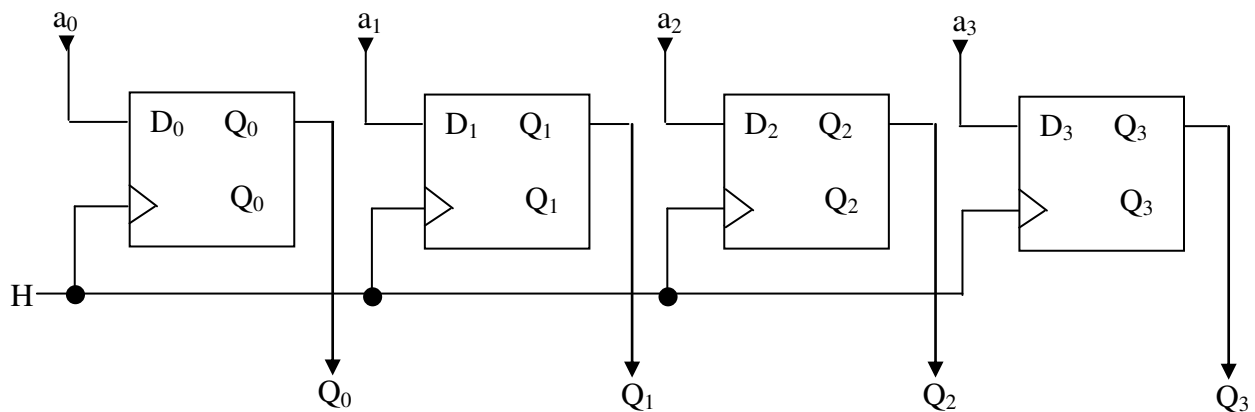


Figure I.1 : Registre PIPO

Exemple :

Avant le front :

$A_0=D_0$	$a_1=D_1$	$a_2=D_2$	$a_3=D_3$
1	0	0	1
Q_0	Q_1	Q_2	Q_3
0	0	0	0

Après le front :

Q_0	Q_1	Q_2	Q_3
1	0	0	1

II.2. Registre à écriture série / lecture série (Serial In Serial Out : SISO)

C'est un type de registre dans lequel les données arrivent en série et sont transmises en série (sur une seule sortie). C'est un circuit retardateur.

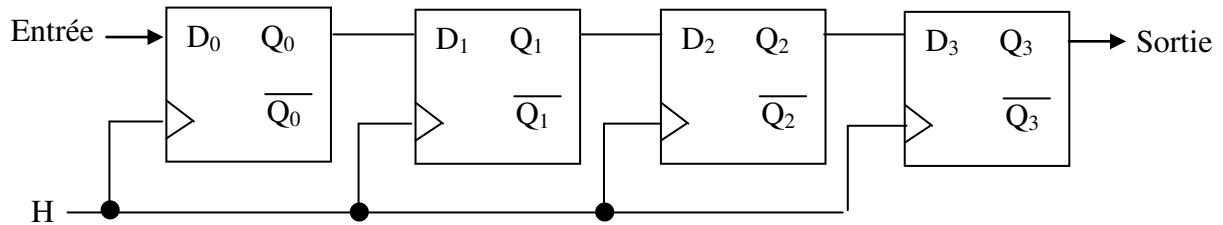


Figure I.2 : Registre SISO

II.3. Registre à écriture série / lecture parallèle (Serial In Parallel Out : SIPO)

C'est un type de registre dans lequel les données arrivent en série et ressortent en parallèle.

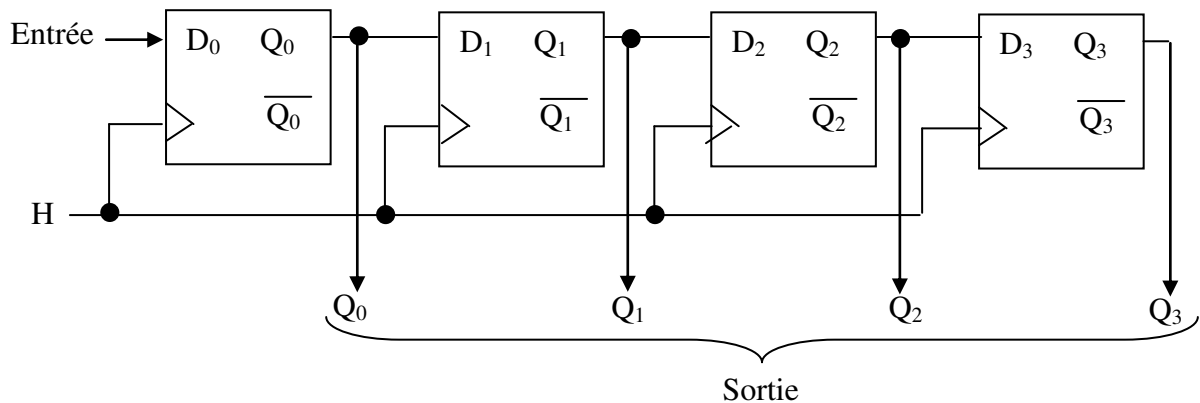


Figure I.3 : Registre SIPO

La transmission parallèle des informations d'un registre à un autre est la plus facile. La transmission série utilise peu d'éléments donc peu coûteuse.

II.4. Registre à écriture parallèle / lecture série (Parallel In Serial Out : PISO)

Le chargement dans ce type de registre s'effectue d'une manière parallèle et la récupération de l'information est sérielle. Dans ce type de registre on a besoin d'un signal de chargement et/ou de décalage.

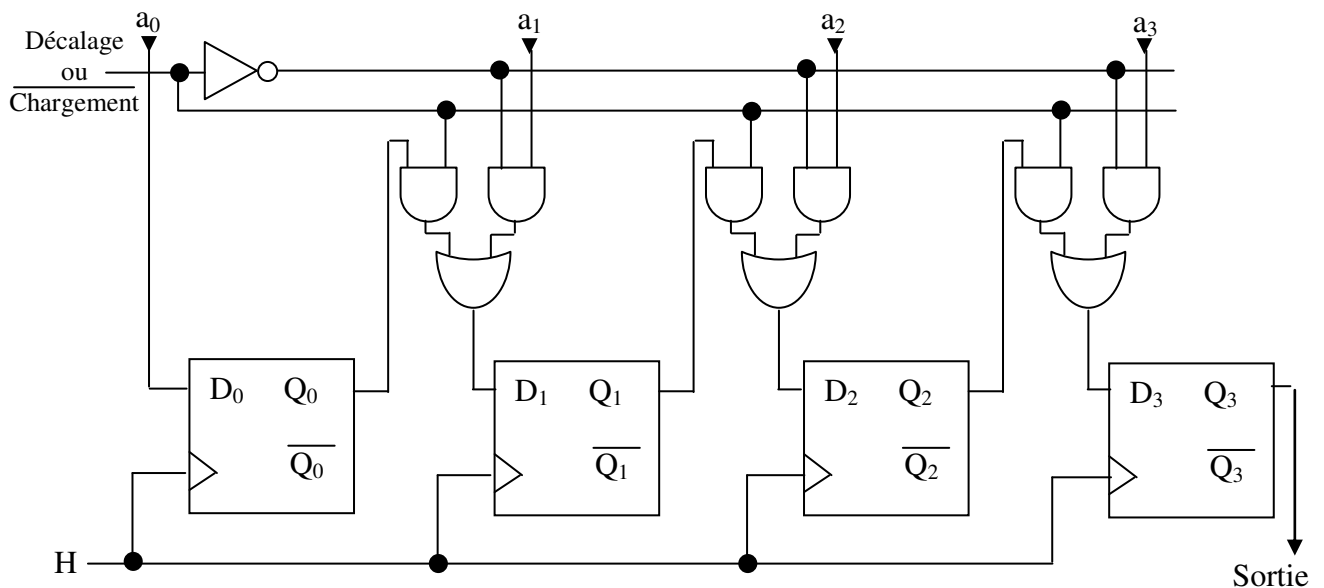


Figure I.4 : Registre PISO

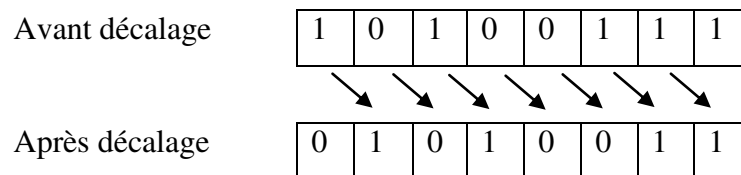
III. Registres à décalage [1,2]

Ce sont des circuits qui transfèrent les données d'une bascule à une autre, bit par bit (1 bit à la fois).

III.1. Décalage à droite

Il consiste à faire avancer l'information vers la droite.

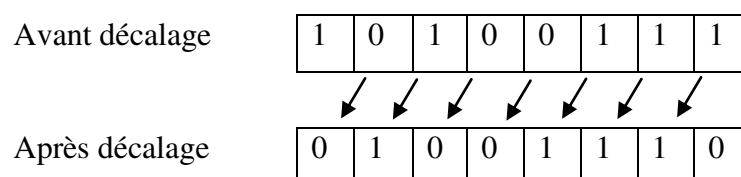
Exemple:



III.2. Décalage à gauche

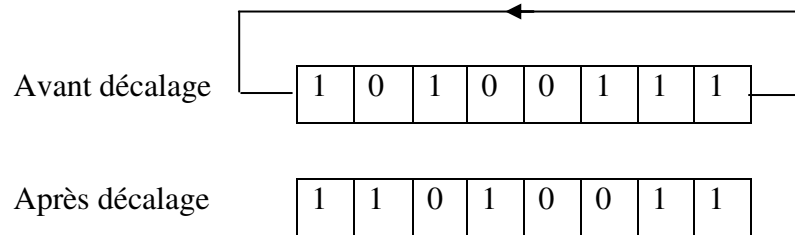
Il consiste à faire avancer l'information vers la gauche.

Exemple:

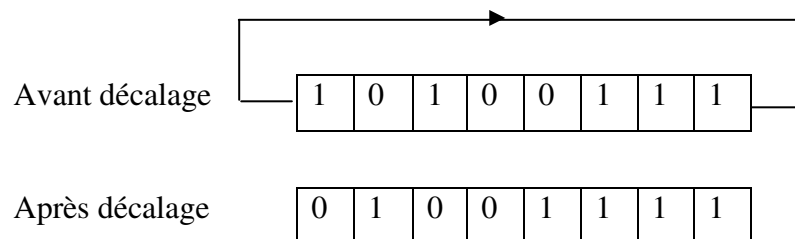


III.3. Registre à décalage en anneau ou registre à décalage circulaire (rotation)

* Rotation à droite:



* Rotation à gauche:



III.4. Registre à décalage universel bidirectionnel 4 bits "74194" [3]

Le brochage du circuit intégré 74194 est donné par la figure I.5.

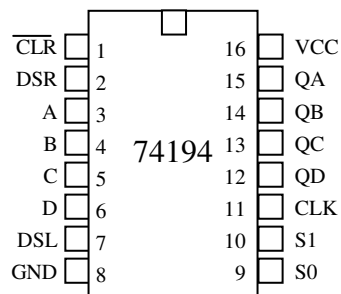


Figure I.5 : Brochage du 74194

* Spécificités du 74194

- Entrées d'horloge et de contrôle amplifiées
- Possibilité de décalage à gauche ou à droite
- Transfert synchrone des données en parallèle ou en série
- RAZ générale asynchrone
- Mode "maintien des sorties"

	Mode de fonctionnement	CLK	/CLR	S1	S0	DSR	DSL	QA	QB	QC	QD
1	Remise à zéro	X	0	X	X	X	X	0	0	0	0
2	Maintien des sorties	X	1	0	0	X	X	QA	QB	QC	QD
3	Décalage à gauche	↑	1	1	0	X	0	QB	QC	QD	0
4	Décalage à gauche	↑	1	1	0	X	1	QB	QC	QD	1
5	Décalage à droite	↑	1	0	1	0	X	0	QA	QB	QC
6	Décalage à droite	↑	1	0	1	1	X	1	QA	QB	QC
7	Chargement parallèle	↑	1	1	1	X	X	A	B	C	D

Tableau I.1 : Fonctionnement du 74194

1: On remarque que l'entrée de RAZ /CLR est active au niveau 0 et est prioritaire.

2: Lorsque S0=S1=0, l'état des sorties n'évolue pas.

3 et 4: En mode décalage à gauche, il faut placer S1 à 1 et S0 à 0, les états des sorties se décalent (par exemple, QA prend l'état présent sur QB avant le front montant (FM) sur (CLK). Il est important de constater que QD prend l'état présent sur l'entrée DSL.

5 et 6: Mode décalage à droite.

7: Mode chargement parallèle (S0=S1=1 et (FM) sur (CLK)). Ce mode permet de charger les états logiques des sorties qui seront décalées. Par exemple, la sortie QA prend l'état logique présent sur l'entrée (A).

* Autres registres à décalage

74164 : Registre à décalage 8 entrées

74165 : Registre à décalage 8 entrées, sortie inversée

74166 : Registre à décalage 8 entrées et entrée série

74299 : Registre à décalage bidirectionnel 8 bits, sortie 3 états

* Application des registres à décalage

La fonction registre à décalage est rencontrée principalement dans les applications de:

- Transmission de données numériques sur de longues distances,
- Génération de retard sur des signaux logiques ou numériques,
- Calcul numérique binaire
- Commande des moteurs pas à pas
- ...

IV. Compteurs [4]

Une bascule peut avoir deux états différents à sa sortie (0 et 1), et peut donc permettre de compter de 0 à 1. Avec deux bascules on peut avoir jusqu'à quatre états différents : 00, 01, 10 et 11, ce qui permet de compter de 0 à 3 en binaire naturel. Avec trois bascules on a huit états (de 000 à 111), et en général avec n bascules on a 2^n états : on peut donc compter de 0 à $2^n - 1$. Il reste à trouver comment doivent être connectées les n bascules entre elles pour réaliser un compteur, sachant qu'il existe plusieurs types de compteurs, et donc plusieurs techniques de réalisation.

IV.1. Classification des compteurs

En logique séquentielle, les compteurs peuvent être décrits en citant cinq caractéristiques :

1– Le sens de comptage. Il permet de différencier :

- * Les compteurs (évolution croissante de la valeur de sortie dans le temps)
- * Les décompteurs (évolution décroissante de la valeur de sortie dans le temps)

2– Le code dans lequel est exprimée la valeur sortie. Il permet de différencier :

- * Les compteurs en binaire naturel
- * Les compteurs BCD
- * Les compteurs en Code Gray
- * Etc...

3– Le type de basculement du compteur. Il permet de différencier :

- * Les compteurs asynchrones
- * Les compteurs synchrones

4– Le nombre de bits en sortie, ou l'intervalle de la valeur de sortie. Il permet de connaître l'ensemble des valeurs que peut prendre la valeur de sortie du compteur. Exemples : compteur 4 bits ; décompteur de 25 à 3 (sous entendu décompteur 5bits).

IV.2. Modes de comptage

Les compteurs sont différenciés par :

- * Les compteurs à cycle complet
- * Les compteurs à cycle incomplet

Exemples :

- Un compteur 4 bits qui compte de 0 à 15 en binaire naturel est un compteur à cycle complet, car sa valeur de sortie utilise toutes les combinaisons possibles de ses sorties.
- Un compteur 4 bits qui compte de 0 à 9 seulement (on l'appelle aussi compteur BCD) est un compteur à cycle incomplet, car les 16 combinaisons de ses 4 sorties ne sont pas toutes utilisées.
- Un décompteur 6 bits qui décompte de 53 à 12 est un décompteur à cycle incomplet.
- Si on parle d'un compteur binaire naturel 7 bits à cycle complet, on sait qu'il compte forcément de 0 à 127.

IV.3. Exemple d'un compteur binaire synchrone à cycle complet à 3 bits

Soit à synthétiser un compteur binaire à cycle complet à 3 bits

*** Table des transitions du compteur**

Etat présent			Etat futur			Entrées		
Q_2	Q_1	Q_0	Q_2^+	Q_1^+	Q_0^+	D_2	D_1	D_0
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	0
0	1	0	0	1	1	0	1	1
0	1	1	1	0	0	1	0	0
1	0	0	1	0	1	1	0	1
1	0	1	1	1	0	1	1	0
1	1	0	1	1	1	1	1	1
1	1	1	0	0	0	0	0	0

Tableau I.2 : Table des transitions d'un compteur complet à 3 bits

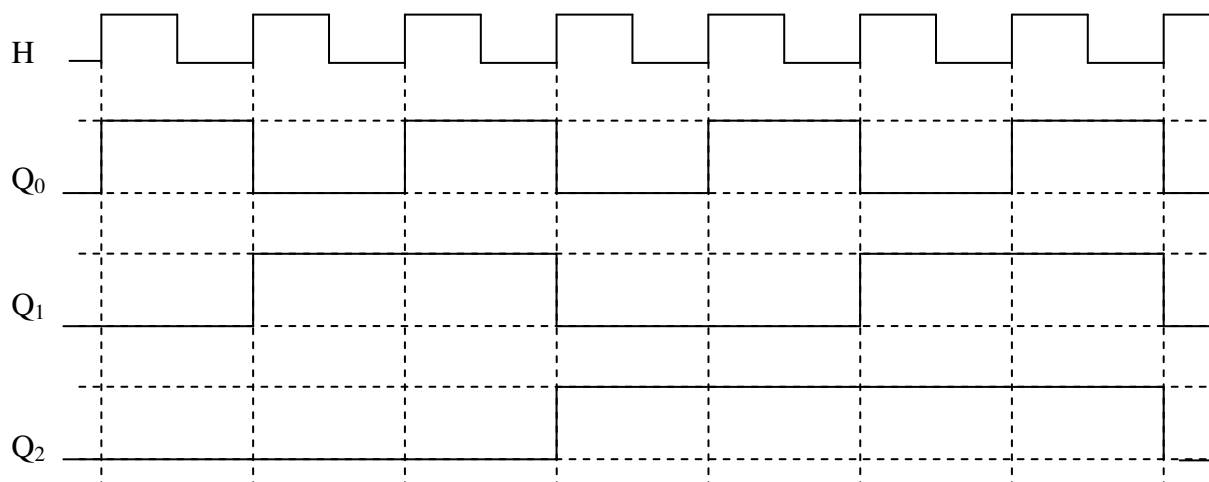
*** Chronogramme du compteur**

Figure I.6 : Chronogramme du compteur complet à 3 bits

*** Réalisation avec des bascules D**

$$D_2 = Q_2 \oplus (Q_1 \cdot Q_0)$$

$$D_1 = Q_1 \oplus Q_0$$

$$D_0 = /Q_0$$

*** Logigramme**

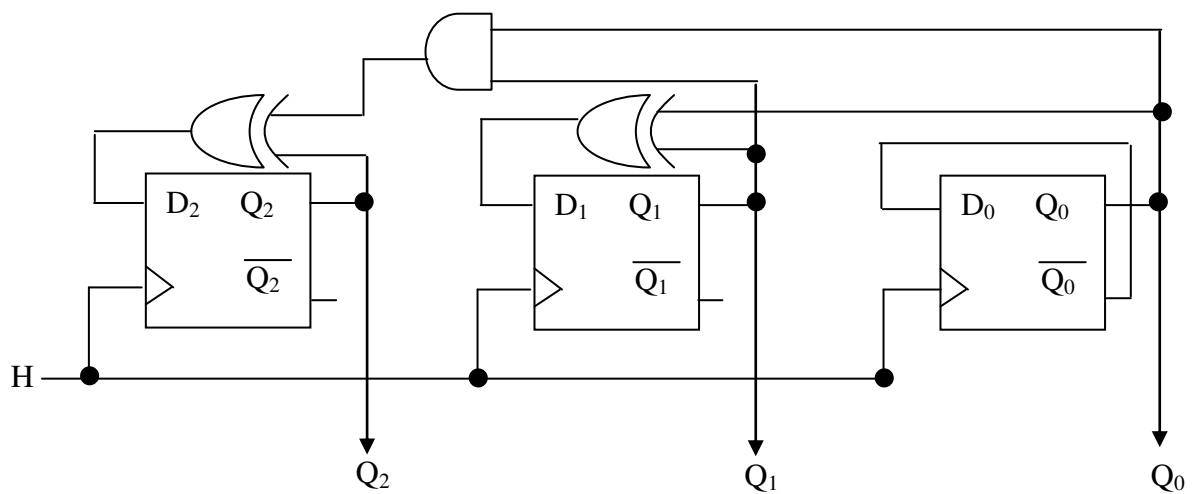


Figure I.7 : Logigramme du compteur complet à 3 bits

CHAPITRE II

LES MEMOIRES A SEMI CONDUCTEURS

I. Identification de la fonction

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.

II. Architecture d'une mémoire

L'adressage mémoire est la façon dont sont accédées des données stockées en mémoire. Une adresse mémoire est un nombre entier naturel qui désigne une zone particulière de la mémoire, ou juste le début d'une zone. Le plus souvent, une donnée peut être lue ou écrite (figure II.1).

Contrôle

- sélection de la mémoire
- sens de la donnée (écriture/lecture)
- ...

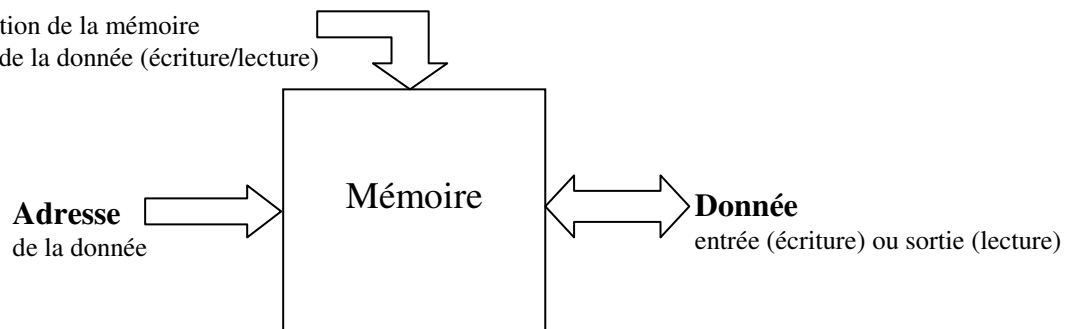


Figure II.1 : Architecture d'une mémoire

III. Types de mémoires [5-9]

III.1. Mémoires mortes ROM (Read Only Memory)

Les mémoires mortes sont à lecture seule : 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. Dans cette catégorie de mémoires on trouve :

- * **ROM Masquée** : elle est programmée une fois, par le constructeur, lors de la fabrication.
- * **PROM à fusibles ou FEPROM (Fuse Prom)** : une seule programmation est possible, elle est faite en « brûlant » des fusibles. Elle se programme à l'aide d'un équipement particulier appelé programmeur.
- * **OTP (One Time Prom)** : le contenu peut être modifié une fois par l'utilisateur à l'aide d'un programmeur.
- * **UVPROM** : elle est effaçable aux ultraviolets. Il faut 10 à 20 minutes pour l'effacer (effacement de toute la capacité de la mémoire). Elle est ensuite reprogrammable.
- * **EEPROM ou E2PROM** : Elle est effaçable électriquement (effacement adresse par adresse). Elle est ensuite reprogrammable.
- * **EPROM FLASH** : Elle est effaçable électriquement plus rapidement (effacement de toute la capacité de la mémoire). Son temps de programmation est plus rapide, et son coût de fabrication est plus faible que l'EEPROM.

La mémoire morte étant immuable une fois qu'elle a été construite, il n'est pas question d'aller y écrire par la suite.

La constitution d'une mémoire morte est un jeu d'enfant : il suffit, dans le circuit de la mémoire vive, que chaque fil d'adresse soit connecté (par une diode) ou non à chaque fil de donnée (selon la valeur que le constructeur a voulu inscrire dans ce bit à cette adresse), ce dernier étant raccordé à la masse via une résistance. En pratique nous aurons la figure II.2.

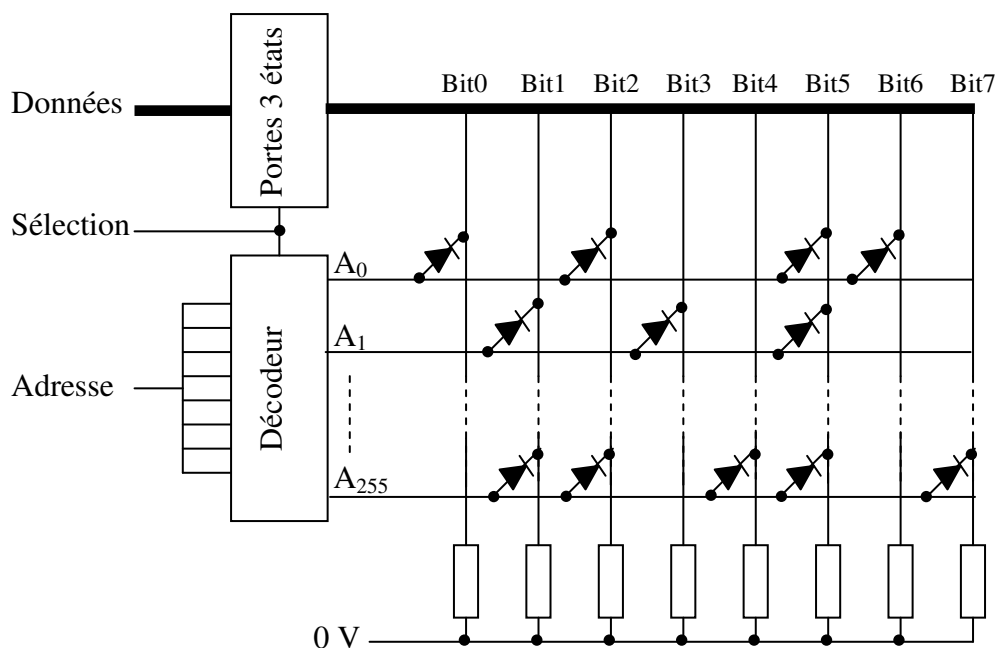


Figure II.2 : Structure interne d'une mémoire ROM

III.2. Mémoires vives (RAM : Random Access Memory)

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 on trouve :

* **La mémoire RAM statique (SRAM)** dans laquelle les informations sont mémorisées par une bascule et conservées tant que l'alimentation est présente, elle est réalisée en technologie MOS (4 ou 6 transistors pour une bascule).

* **La mémoire RAM dynamique (DRAM)** qui utilise un condensateur comme cellule mémoire (un bit mémorisé) de l'information. 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).

Avantage : plus grande intégration et moindre coût.

Inconvénient : utilisation d'une logique supplémentaire de rafraîchissement de la mémoire (intégrée pour certaines au boîtier, donc transparent pour l'utilisateur). Avec la SDRAM (utilisée notamment dans les PC) les temps d'accès sont plus faibles (pratiquement équivalent à ceux de la SRAM).

Le montage de la figure II.3 constitue une mémoire à 256 octets. Chaque bit est désigné par un carré (bascule).

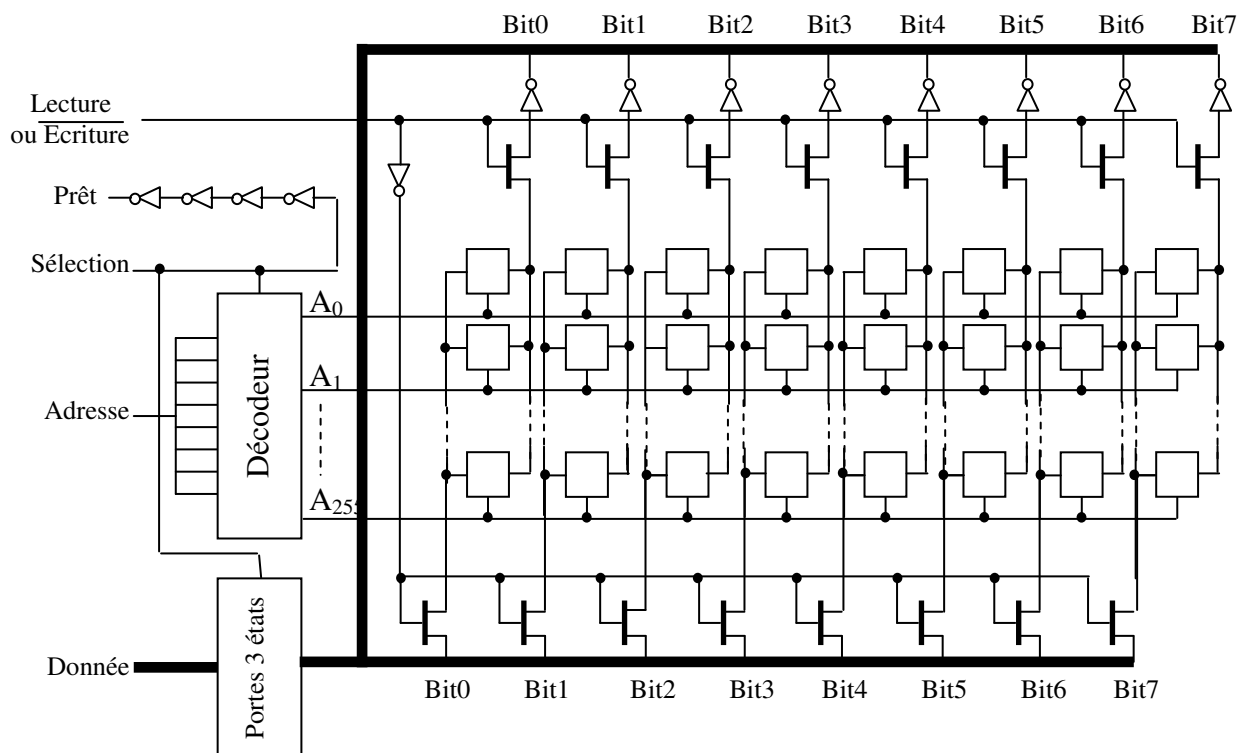


Figure II.3 : Structure interne d'une mémoire RAM

IV. Structure d'une mémoire à accès direct

La mémoire (figure II.4) peut être vue comme un large vecteur (tableau) de mots ou octets.

- Un mot mémoire stocke une information sur D bits.
- Un mot mémoire contient plusieurs cellules mémoire.
- Une cellule mémoire mémorise un seul bit.
- Chaque mot possède sa propre adresse.
- Une adresse est un numéro unique qui permet d'accéder à un mot mémoire.
- Les adresses sont séquentielles (consécutives).
- La taille de l'adresse (le nombre de bits) dépend de la capacité de la mémoire.

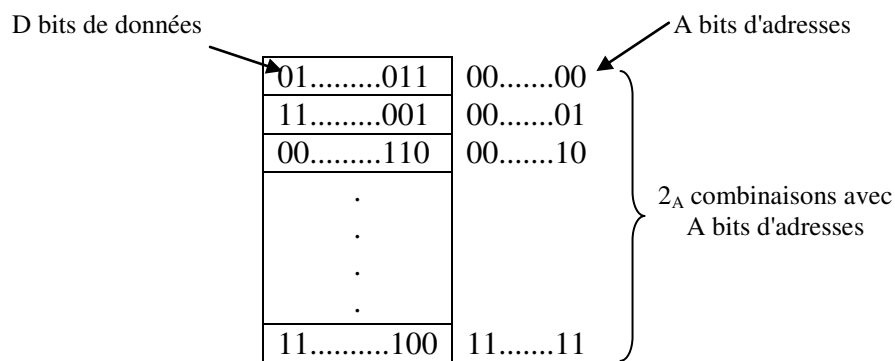


Figure II.4 : Structure d'une mémoire à accès direct

V. Caractéristiques d'une mémoire

V.1. Capacité

La capacité d'une mémoire représente le nombre de bits que l'on peut adresser, elle est exprimée en bits.

$$\text{Capacité d'adressage en bits} = 2^A \times D \text{ bits}$$

Exemple : EPROM _ M 27C512 : mémoire de 512 Kbits soit :

$$512 \times 1024 = 524288 \text{ bits}$$

V.2. Format

C'est l'arrangement des bits dans la mémoire, il correspond au format du bus de données.

Par exemple pour l' EPROM M 27C512 le format est de 8 bits (1 Octet).

Rappels : 1ko = 1 Kilo Octet = 2^{10} Octets = 1024 Octets,

1Mo = 1 Méga Octet = 2^{10} KO = 2^{20} Octets,

1Go = 1 Giga Octet = 2^{10} MO = 2^{30} Octets.

V.3. Vitesse

Deux paramètres principaux caractérisent la vitesse d'une mémoire :

- * le temps d'accès : c'est le temps qui sépare le moment où les adresses sont positionnées et stables et le moment où les données sont disponibles sur le bus de sortie (100ns pour l'EPROM M 27C512) ;
- * le temps de cycle : c'est l'intervalle de temps minimum qui doit séparer deux demandes d'écriture ou de lecture.

V.4. Consommation

Elle varie suivant l'état de la mémoire.

- * Etat actif : au moment où une opération de lecture ou d'écriture est demandée, la consommation est plus importante (30mA pour l'EPROM M 27C512).
- * Etat passif : au moment où la mémoire n'est pas sollicitée, la consommation est plus faible (100µA pour l'EPROM M 27C512).
- * Etat de veille : pour diminuer la consommation sans perdre d'informations, on alimente les mémoires (RAM) sous tension réduite (3V au lieu de 5V), mais le système ne peut fonctionner.

V.5. Brochage et alimentation du boîtier

En général les mémoires sont alimentées sous +5V.

Broches caractéristiques :

A : lignes d'adresse (Address Inputs).

I/O : données d'entrées / sorties (data Inputs / Outputs).

VDD et GND : alimentation (Power et GrouND).

R /W : contrôle des entrées lecture / écriture pour les RAM (Read / Write control input).

CE : validation du circuit (Chip Enable). Si le circuit n'est pas validé, les sorties sont en haute impédance.

OE : validation des sorties (Output Enable). Si les sorties ne sont pas validées elles sont en haute impédance.

V.6. Cycle de fonctionnement

Les échanges d'une mémoire avec l'extérieur se font suivant une procédure bien définie donnée par l'état des signaux de gestion de la mémoire.

Exemple : Cycle de lecture d'une SRAM

Pour lire le contenu d'une mémoire, il faut :

- présenter une adresse sur le bus d'adresse,
- sélectionner le boîtier par l'intermédiaire du « Chip Enable (CE) » ou du « Chip Select (CS) »,
- valider la mémoire en position lecture (R /W en position lecture _ read),
- au bout d'un temps d'accès, les données sont disponibles en sortie sur le bus de données. Les broches de sortie passent de l'état haute impédance à l'état actif.

Chronogrammes :

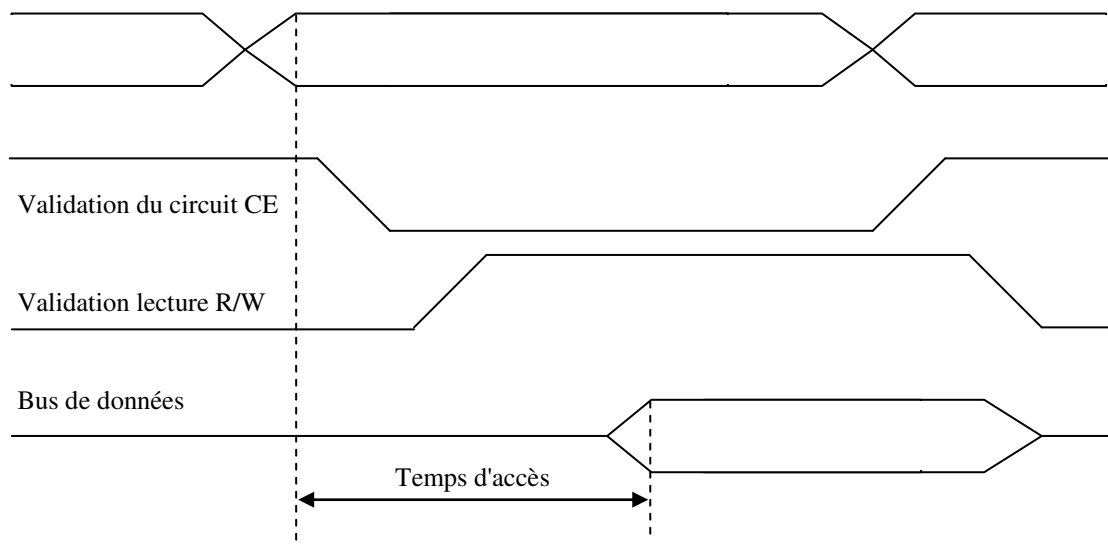


Figure II.5 : Chronogrammes d'accès mémoire

VI. Assemblage des boîtiers mémoire [9]

Les techniques d'intégration ne permettent pas d'obtenir des boîtiers ayant des capacités ou des formats suffisants pour toutes les applications. Il est alors nécessaire d'associer plusieurs boîtiers pour augmenter la longueur des mots ou le nombre de mots. D'autre part, l'association de plusieurs blocs peut permettre d'améliorer les performances temporelles de la mémoire en faisant fonctionner plusieurs blocs en parallèle.

Pour obtenir des mémoires de grandes tailles, on associe plusieurs boîtiers mémoires. Ces blocs sont assemblés :

- pour augmenter la taille des mots de la mémoire
- pour augmenter le nombre de mots dans la mémoire

VI.1. Augmentation de la taille des mots

La figure II.6 montre qu'il est aisé d'associer deux boîtiers de 2^k mots de n bits pour obtenir un bloc de 2^k mots de $2n$ bits. L'adressage doit être appliqué simultanément aux deux circuits, l'un fournissant les n bits de bas poids et l'autre les n bits de haut poids.

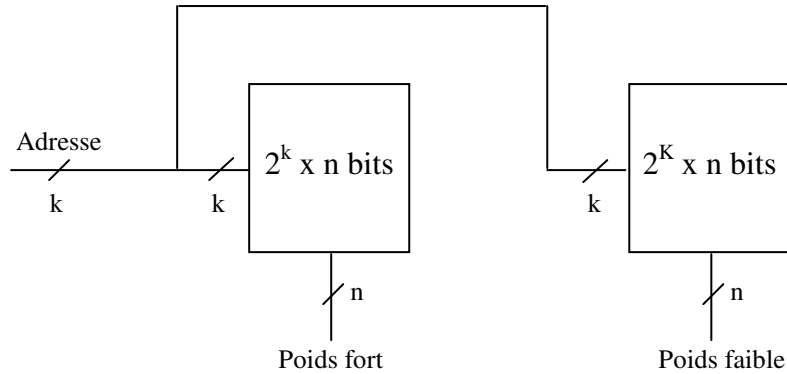


Figure II.6 : Augmentation de la taille mémoire

VI.2. Augmentation du nombre de mots

De même la figure II.7 montre la réalisation d'un bloc de 4×2^k mots de n bits à l'aide de 4 boîtiers de $2^k \times n$ bits. Il nous faut $k+2$ lignes d'adresse. Les k bits de bas poids de l'adresse sont appliqués simultanément sur les 4 boîtiers. Les deux bits de haut poids attaquent un décodeur à quatre sorties. Chacune de ces quatre lignes permet de sélectionner un boîtier (entrée de validation du boîtier : CS). Un seul boîtier est alors connecté aux lignes de sortie.

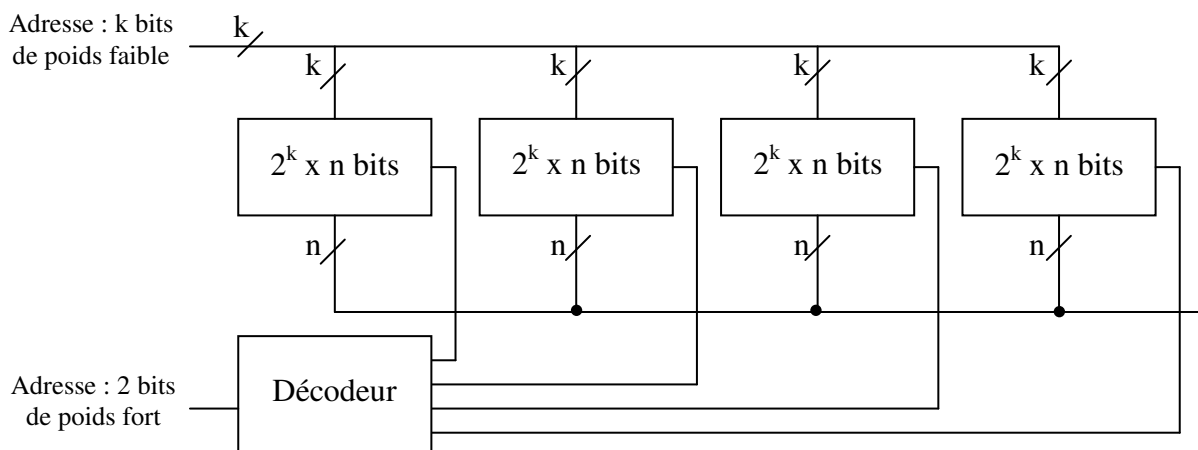


Figure II.7 : Augmentation du nombre de mots

CHAPITRE III

HISTORIQUE ET EVOLUTION DES ORDINATEURS

I. Historique des ordinateurs [10]

L'ordinateur, un nom donné en 1955 par J. Perret, est une machine à traiter les données (l'information). Il contient principalement un processeur, une mémoire et des mécanismes. Il est composé d'un écran, d'une unité centrale, d'un clavier, d'une souris et d'une carte vidéo. On peut lui ajouter plusieurs périphériques comme l'imprimante, le scanner, etc.

L'histoire du développement de l'ordinateur se présente de la manière suivante :

- En 1937, Howard Aiked crée le Mark 1 qui est un ordinateur programmable de 17 m de long sur 2,5 m de haut; son temps de calcul est de 5 fois plus rapide que celui de l'homme;
- En 1938, Konrad Zuse a créé le Z3, le premier ordinateur qui a utilisé le binaire au lieu du décimal et qui fonctionne grâce à des relais électromécaniques;
- En 1947, il y a eu l'amélioration du Mark 1 (qui a été appelé Mark 2). L'unique différence entre les deux était le remplacement des engrenages par des composants électriques;
- 1942, L'ABC (Atanasoff Berry Computer) est créée. Il représente le premier ordinateur à tube (lampe allumée = 1, lampe éteinte = 0);
- En 1946, c'est la disparition des pièces mécaniques dans l'ENIAC (Electronic Numerial Integrator And Computer) ; il occupe 1500 m² et l'apparition des premiers programmes stockés en mémoire. Cet ordinateur de 30 tonnes consommait 140 kilowatts et effectuait 330 multiplications par seconde;
- En 1971, c'est la série des micros ordinateurs intégrant les transistors avec la naissance du Kenback 1 qui disposait d'une mémoire de 256 octets;
- En 1976, il y a eu la création du premier ordinateur Apple doté d'un microprocesseur et d'un clavier qui avaient consisté à améliorer davantage non seulement la capacité de stockage, mais aussi le temps de traitement des données;
- En 1981, c'est le grand tournant dans l'histoire de l'ordinateur avec l'apparition du premier ordinateur personnel (Personal Computer). Cette époque constitue le point de départ de tout ce que nous connaissons aujourd'hui comme ordinateur.

II. Catégories d'ordinateurs [11,6]

II.1. Superordinateurs

Cette sorte d'ordinateur est plus grosse qu'un être humain de taille adulte. Cette taille est due au nombre impressionnant de processeurs qu'elle contient. À leurs débuts, les superordinateurs contenaient de 4 à 16 processeurs. Un seul de ces processeurs peut suffire à faire fonctionner un ordinateur central (qui était utilisé pour les compagnies). Maintenant, les superordinateurs utilisent des dizaines de milliers de micro-processeurs afin de subvenir aux besoins des meilleurs ordinateurs les plus utilisés.

Les superordinateurs sont souvent utilisés pour accomplir certaines tâches telles que les calculs pour la prévision météorologique, pour les calculs complexes (en partie pour les scientifiques), pour analyser le génome humain ainsi que plusieurs autres fonctions. Ces ordinateurs valent plusieurs dizaines de millions de dollars.

II.2. Ordinateurs centraux

Ces engins électroniques ne sont pas aussi immenses que les superordinateurs, mais si nous comparons les deux sortes d'ordinateurs, de la même époque, l'ordinateur central est moins puissant que le superordinateur. Ces machines sont très fiables, car ils peuvent continuer de fonctionner même s'il manque d'électricité ou qu'il y a des mises à jour à faire ou s'il y a des réparations à faire à l'intérieur et ils peuvent fonctionner durant des années sans arrêt. Ils sont aussi fiables, car ils peuvent partager des ressources au travers les réseaux. C'est pourquoi ce sont les banques, les ministères, les chaînes de magasins, etc. qui s'en servent. Ce type d'ordinateurs coûte dans les centaines de milliers jusqu'à quelques millions de dollars.

II.3. Mini-ordinateurs

Cette catégorie n'est plus tellement significative aujourd'hui; mise à part le fait qu'ils étaient presque toujours des systèmes multiutilisateurs, leur architecture est comparable aux bons micro-ordinateurs courants. Au début, il s'agissait simplement d'appareils moins puissants que les ordinateurs centraux et disponibles à un prix plus abordable. Aujourd'hui, les systèmes qui ressemblent le plus à un vrai « mini » servent de serveurs (web ou réseau) et sont souvent très proches d'un micro-ordinateur très bien équipé. Par contre, leur grande capacité de communication et de redondance (pour éviter les problèmes lors des pannes) sont rarement vues dans des « ordinateurs personnels » ordinaires. Ce type d'ordinateur coûte quelques dizaines à quelques centaines de milliers de dollars.

II.4. Micro-ordinateurs

Leur puissance ne cesse d'augmenter, au point où c'est souvent la capacité des équipements qu'on peut y ajouter qui les différencient des mini-ordinateurs. Par ailleurs, ils sont rarement utilisés en tant que système multiutilisateurs. On peut distinguer deux catégories : les ordinateurs de table (desktop) et les portables/blocs-notes (incluant les ordinateurs de poche et les assistants personnels).

Les principaux modèles d'aujourd'hui sont soit de type PC (avec Windows ou Linux) ou soit des Macintosh. Voici l'exemple classique : IBM PC. Ces ordinateurs coûtent quelques centaines à plusieurs milliers de dollars.

III. Organisation d'un ordinateur [7,8,12-14]

Un ordinateur joue un rôle principal dans les sociétés industrielles. Un ordinateur peut être classé comme un circuit programmable qui permet de commander les processus industriels.

Un ordinateur permet de lire les instructions depuis la mémoire, acquérir les données binaires comme entrées en vue de les traiter et offrir les résultats comme sorties.

Une machine programmable peut être représentée par quatre composants : Unité centrale, mémoire, unités d'entrée entrées et unités de sorties comme le montre la figure III.1. Un tel système est nommé partie hardware. Un ensemble d'une suite d'instructions est appelé programme et un ensemble de programmes est nommé partie software.

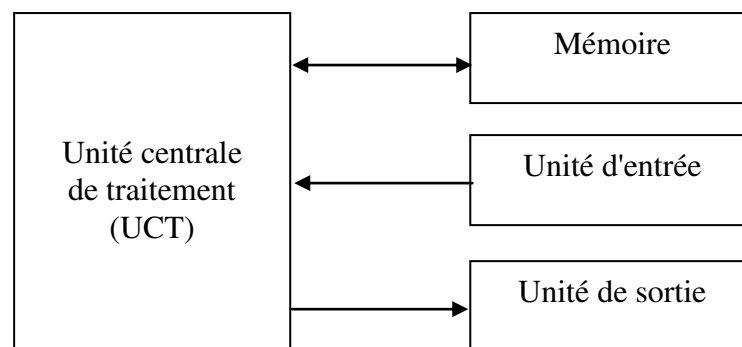


Figure III.1 : Organisation d'un ordinateur

III.1. Unité centrale de traitement (Central Processing Unit : CPU)

C'est la partie la plus importante de l'ordinateur. Elle s'occupe du traitement et du stockage de l'information. Une CPU est formée par les trois éléments fonctionnels interconnectés suivants :

- Registres.

- UAL : Unité arithmétique et logique.
- Circuit de contrôle.

III.2. Mémoire

Il s'agit de l'ensemble des ressources de stockage que l'ordinateur peut employer pour sauvegarder des données afin de les réutiliser ultérieurement.

La mémoire sert au rangement de deux types d'informations :

- * Des données : les informations traitées par le microprocesseur.
- * Des instructions : ensemble d'informations codées qui gère l'activité de l'ordinateur.

L'utilisation de mémoires est donc obligatoire dans tous systèmes utilisant un microprocesseur, dont les ordinateurs.

Remarque

La mémoire morte (ROM : Read Only Memory) range en général le programme d'initialisation du système (exemple dans le PC elle range le BIOS : Basic Input Output system).

La mémoire vive (RAM : Random Access Memory) sert au rangement des programmes utilisateurs c'est une mémoire volatile.

III.3. Unités d'entrées / sorties

Les unités d'entrées sorties vont permettre à l'ordinateur de communiquer avec le monde extérieur, Nous trouvons des ports utilisés exclusivement pour l'entrée, et d'autres ports exclusivement pour la sortie. Il existe aussi des ports bidirectionnels. Donc le microprocesseur peut lire des données à partir d'une interface d'entrée (exemple souris, clavier disque dur, etc.). De même il peut restituer le résultat de son traitement au monde extérieur en adressant des interfaces de sortie (tel que les imprimantes le clavier, etc...) donc les interfaces d'entrées / sorties vont soulager le microprocesseur pour la communication avec le monde extérieur .

III.4. Bus du système [15]

Les trois modules sont interconnectés comme le montre la figure précédente autour de trois bus : bus de données, bus d'adresses et bus de contrôles et commandes.

Bus : Il s'agit de plusieurs pistes électroniques qui sont reliées au microprocesseur. Ces bus assurent la communication interne et externe du microprocesseur.

III.4.1. Bus de données

C'est un ensemble de fils bidirectionnels qui va permettre le transfert de données entre les différents éléments du système. C'est par ce bus que sont transmises les données qui doivent être traitées par le microprocesseur. A l'inverse, c'est également par ce bus que transitent les résultats en sortie du microprocesseur. Autrement dit, toutes les données entrantes et sortantes du microprocesseur sont véhiculées par le bus de données qui fixe la longueur du mot échangé avec la mémoire.

III.4.2. Bus d'adresses

Ce bus permet d'adresser un élément par le microprocesseur. Il est unidirectionnel, il détermine la capacité maximale d'adressage du système, c'est à dire le nombre maximum de mots de la mémoire associée (ex : 16 bits "adressent" 64 Kmots).

III.4.3. Bus de commandes et de contrôle

C'est un bus qui permet de véhiculer les signaux de contrôles et de commandes tels que l'horloge les signaux Rd/Wr etc ... Ce bus sert à coordonner tous les échanges d'informations décrits précédemment. Il véhicule des données qui valident la mémoire et les ports d'entrées / sorties. Il introduit des délais d'attente lorsque des informations sont envoyées à un périphérique qui présente une vitesse de traitement réduite. Le bus de commandes évite les conflits de bus lorsque deux éléments cherchent à communiquer en même temps.

Remarque :

Dans certains cas, le bus de données et le bus d'adresses sont multiplexés sur un seul bus. Une logique externe doit alors effectuer le démultiplexage.

IV. Périphériques [15]

On peut classer généralement les périphériques en deux types : les périphériques d'entrée et les périphériques de sortie :

IV.1. Périphériques d'entrée

Servent à fournir des informations (ou données) au système informatique tel que clavier (frappe de texte), souris (pointage), scanner (numérisation de documents papier), micro, webcam, etc.

IV.2. Périphériques de sortie

Servent à faire sortir des informations du système informatique tel que : écran, imprimante, haut-parleur, etc.

V. Microprocesseur [5-8,15,16]

Le microprocesseur, élément le plus important de tout l'ordinateur, tire son nom de sa fonction. Il est l'une des seules entités de la machine (en tout cas la plus importante), capable d'exécuter des instructions. C'est lui qui lit vos programmes, et les exécute, en travaillant avec la mémoire, ...

Le premier microprocesseur, lui a été commercialisé, le 15 novembre 1971, c'est l'Intel 4004 4-bits . Il fut suivi par l'Intel 8008. Ce microprocesseur a servi initialement à fabriquer des contrôleurs graphiques en mode texte, mais jugé trop lent par le client qui en avait demandé la conception, il devint un processeur d'usage général. Ces processeurs sont les précurseurs des Intel 8080, Zilog Z80, et de la future famille des Intel x86.

Un microprocesseur se caractérise aujourd'hui par différentes fonctions. La première est le jeu d'instructions qu'il est capable d'exécuter, pouvant aller de dizaines à des milliers d'instructions différentes. La deuxième est la complexité de son architecture qui se mesure par le nombre de transistors présents : plus ce nombre est élevé, plus la complexité des tâches à traiter peuvent augmenter. La troisième est la vitesse de son horloge qui dicte le rythme de travail. Enfin, le microprocesseur se caractérise par le nombre de bits qu'il peut traiter (4 à ses débuts, 128 en 2011). A sa création, il était capable d'effectuer un peu moins d'un million d'instructions par seconde. Aujourd'hui, il en traite plus de 10 milliards.

Les composants d'un microprocesseur sont principalement composés de 3 éléments qui sont une unité de commande (UC), une unité arithmétique et logique (UAL) et des registres. Ces trois éléments sont reliés entre eux par un bus interne, celui-ci permettant les échanges de données entre les différentes parties du microprocesseur.

Le microprocesseur manipule les chiffres "0" et "1" appelés **bits**. Chaque microprocesseur possède son propre jeu d'instructions sous la forme d'un langage machine. Cependant, il est difficile de manipuler les programmes avec ce dernier. Les instructions possèdent des appellations abrégées qui forment le langage assembleur.

Dans les systèmes à microprocesseur, un **mot** est une unité de base manipulée par un microprocesseur. On parle aussi de mot machine ou de **word**. La taille d'un mot s'exprime en bits. Pour les architectures matérielles « grand public », elle est donnée en **octets**, et est souvent utilisée pour classer les microprocesseurs (32 bits, 64 bits, etc.). Par ailleurs, un microprocesseur est d'autant plus rapide que ses mots sont longs, car les données qu'il traite à

chaque cycle sont plus importantes. Sur les microprocesseurs qui peuvent manipuler différentes tailles de données, la taille des mots est choisie arbitrairement, dans le but d'avoir une convention de nommage (en particulier, les instructions dont la mnémonique ne contient pas d'indication explicite de taille s'appliquent à des mots). On prend généralement la taille des principaux registres de données, ou la taille du bus de données.

La nomenclature la plus couramment utilisée par les éditeurs de langages de développement logicielle est normalisée¹ comme suit :

* Donnée de 8 bits : « octet », parfois « byte »

* Donnée de 16 bits : « mot » ou « word »

VI. Notion de programme [13]

Un programme informatique est un ensemble d'instructions destinées à être exécutées par un ordinateur.

Un programme est une suite d'instructions qui spécifie étape par étape, de manière non ambiguë, des représentations de données et des calculs. Les instructions sont destinées à manipuler les **données** lors de l'exécution du programme.

Un programme **source** est un code écrit dans un langage de programmation. Il peut être compilé vers une forme binaire, ou directement interprété.

Un programme **binaire** décrit les instructions à exécuter par un microprocesseur sous forme numérique. Ces instructions définissent un langage machine.

VII. Structures Von Neuman et Harvard [13]

Ce type de microprocesseur incorpore principalement deux unités logiques de base :

* L'unité Arithmétique et Logique (ou ALU en anglais), chargé de réaliser les opérations centrales (de type multiplications, additions, soustractions, rotation, etc.),

* L'unité en charge des Entrées/Sorties, qui commande le flux de données entre le cœur du microprocesseur et les mémoires ou les ports.

Il existe deux types fondamentaux de structures, dites « Von Neuman » et « Harvard ».

VII.1. Structure de Von Neuman

Un microprocesseur basé sur une structure Von Neuman stocke les programmes et les données dans la même zone mémoire (figure III.2). Une instruction contient le code opératoire et l'adresse de l'opérande.

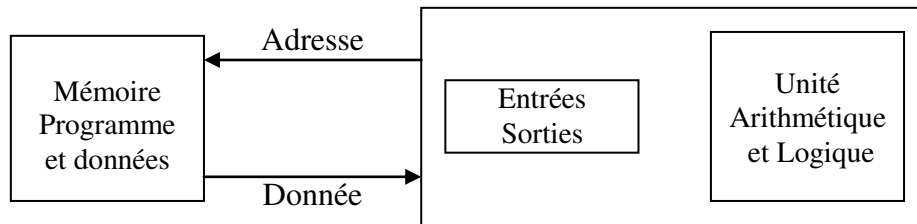


Figure III.2 : Structure de Von Neuman

VII.2. Structure de Harvard

Cette structure se distingue de l'architecture Von Neuman uniquement par le fait que les mémoires programmes et données sont séparées. L'accès à chacune des deux mémoires se fait via un chemin distinct (figure III.3). Cette organisation permet de transférer une instruction et des données simultanément, ce qui améliore les performances.

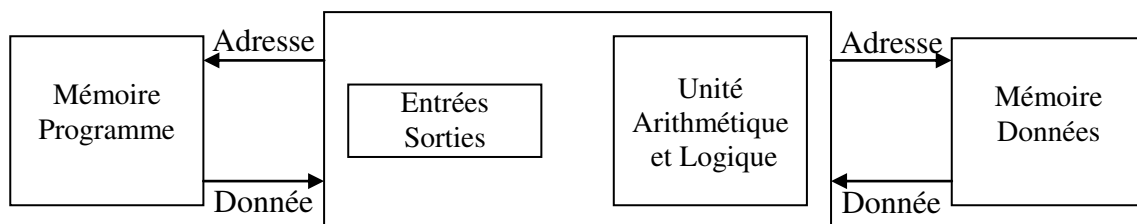


Figure III.3 : Structure de Harvard

CHAPITRE IV

ARCHITECTURE ET FONCTIONNEMENT D'UN MICROPROCESSEUR

I. Historique [16]

Le premier microprocesseur, l'Intel 4004 de la société informatique Intel, est réalisé en 1971. Il est composé de 2300 transistors et exécute 60000 instructions par seconde. En comparaison, un microprocesseur moderne comme l'Intel Pentium 4 comprend plusieurs dizaines de millions de transistors et exécute plusieurs milliards d'instructions par seconde. En fait, l'histoire du microprocesseur est intimement liée à celle du micro-ordinateur et de la micro-informatique en général. L'Intel 8080, créé en 1974, est notamment l'un des premiers microprocesseurs (8bits) adaptés à un ordinateur individuel. Il a fortement influencé l'architecture du Z80 de la société Zilog et, dans une moindre mesure, celle de la gamme des microprocesseurs 80x86 (80186, 80286, 80386, 80486) et les différentes évolutions du Pentium, équipant hier ou aujourd'hui la grande majorité des PC. Par ailleurs, les micro-ordinateurs Macintosh sont équipés de microprocesseurs Motorola (famille du 68000, intégré au premier Macintosh commercialisé en 1984) et, depuis 1998, de microprocesseurs RISC (Reduced Instruction Set Computing) de la série Power PC, développés en collaboration par les firmes américaines Apple, IBM et Motorola. Pionnier de la micro-informatique et créateur du premier microprocesseur, Intel domine le marché des microprocesseurs avec environ 80% de parts de marché. La société AMD suit de loin avec pratiquement 20% et toutes les autres sociétés représentent moins de 1% du marché. La situation est un peu différente pour les microprocesseurs spécialisés où d'autres entreprises sont encore présentes (notamment Texas Instruments aux États-Unis et SMT Electronic en Europe).

Il faut mettre en tête que dans un ordinateur, il n'y a pas qu'un seul processeur mais il existe de nombreux processeurs qui gèrent indépendamment : la carte son, la carte graphique, la carte SCSI, etc...Mais c'est le microprocesseur ou CPU (Central Processing Unit) qui est chargé de traiter toutes les informations de ces composants.

Les processeurs successifs se sont en effet construits petit à petit en ajoutant à chaque processeur des instructions et des fonctionnalités supplémentaires, mais en conservant à chaque fois les spécificités du processeur précédent.

L'évolution des microprocesseurs d'Intel au cours du temps sont résumés dans le tableau IV.1. Cette évolution dépend essentiellement de la densité d'intégration, la fréquence, la taille du bus d'adresses et la taille des données à traiter.

Microprocesseur	Année d'apparition	Nombre de transistors	Vitesse initiale de l'horloge	Bus d'adresses	Bus de données	Mémoire adressable
4004	1971	2.300	108 KHz	10 bits	4 bits	640 octets
8008	1972	3.500	200 KHz	14 Bits	8 bits	16 K
8080	1974	6.000	2 MHz	16 Bits	8 bits	64 K
8085	1976	6.500	5 MHz	16 Bits	8 bits	64 K
8086	1978	29.000	5 MHz	20 Bits	16 bits	1 M
8088	1979	29.000	5 MHz	20 Bits	8 bits*	1 M
80286	1982	134.000	8 MHz	24 Bits	16 bits	16 M
80386	1985	275.000	16 MHz	32 bits	32 bits	4 G
80486	1989	1.2 M	25 MHz	32 bits	32 bits	4 G
Pentium	1993	3.1 M	60 MHz	32 bits	32/64 bits	4 G
Pentium Pro	1995	5.5 M	150 MHz	36 bits	32/64 bits	64 G
Pentium II	1997	8.8 M	233 MHz	36 bits	64 bits	64 G
Pentium III	1999	9.5 M	650 MHz	36 bits	64 bits	64 G
Pentium IV	2000	42 M	1.4 GHz	36 bits	64 bits	64 G

* 8 bits externes et 16 bits internes

Tableau IV.1 : Evolution des microprocesseurs

II. Définition d'un microprocesseur

Le microprocesseur est le cerveau de l'ordinateur. Il permet de manipuler, de circuler les informations et d'exécuter les instructions stockées en mémoire. C'est le composant essentiel d'un ordinateur, où sont effectués les principaux calculs. Toute l'activité de l'ordinateur est cadencée par une horloge unique.

III. Fréquence de fonctionnement

Les microprocesseurs sont cadencés par un signal d'horloge (signal oscillant régulier imposant un rythme au transfert entre circuit). Au milieu des années 1980, ce signal avait une fréquence de 4 à 8 MHz. Dans les années 2000, cette fréquence atteint 3 GHz. Plus cette fréquence est élevée, plus le microprocesseur peut exécuter à un rythme élevé les instructions de base des programmes.

L'augmentation de la fréquence présente des inconvénients :

- la dissipation thermique d'un circuit donné est proportionnelle au carré de sa fréquence de fonctionnement, cela implique d'avoir une solution de refroidissement du processeur adaptée ;

- la fréquence est notamment limitée par les temps de commutation des portes logiques : il est nécessaire qu'entre deux « coups d'horloge », les signaux numériques aient eu le temps de parcourir tout le trajet nécessaire à l'exécution de l'instruction attendue ; pour accélérer le traitement, il faut agir sur de nombreux paramètres (taille d'un transistor, interactions électromagnétiques entre les circuits, etc.) qu'il devient de plus en plus difficile d'améliorer (tout en s'assurant de la fiabilité des opérations).

IV. Architectures CISC et RISC

Les processeurs généraux actuels se répartissent en deux grandes catégories appelées CISC pour Complex Instruction Set Computer et RISC pour Reduced Instruction Set Computer. Les processeurs de ces deux catégories se distinguent par la conception de leurs jeux d'instructions. Les processeurs CISC possèdent un jeu étendu d'instructions complexes. Chacune de ces instructions peut effectuer plusieurs opérations élémentaires comme charger une valeur en mémoire, faire une opération arithmétique et ranger le résultat en mémoire. Au contraire, les processeurs RISC possèdent un jeu d'instructions réduit où chaque instruction effectue une seule opération élémentaire. Le jeu d'instructions d'un processeur RISC est plus uniforme. Toutes les instructions sont codées sur la même taille et toutes s'exécute dans le même temps (un cycle d'horloge en général).

V. Architecture d'un microprocesseur [5-8,15,16]

Un microprocesseur est formé par les trois éléments fonctionnels interconnectés suivants :

- ▶ Registres.
- ▶ UAL : Unité arithmétique et logique.
- ▶ Circuit de contrôle.

V.1. Registres

Dans un processeur, un banc de registres est une mémoire interne au processeur, dans laquelle sont rassemblés les registres du processeur. Dans les microprocesseurs, les bancs de registres sont généralement réalisés à l'aide de RAM statique (bascules).

V.1.1. Accumulateur

Il s'agit d'un registre d'usage général recevant des opérandes, des résultats intermédiaires ou des résultats provenant de l'unité arithmétique et logique. Ils évitent des appels fréquents à la mémoire, réduisant ainsi les temps de calcul. Donc la plupart des opérations arithmétiques et logiques se font dans l'accumulateur.

V.1.2. Registre d'état (Flags)

Le registre d'état sert à contenir l'état de certaines opérations effectuées par le processeur. Par exemple, quand le résultat d'une opération est nul, Ceci provoque un bit spécifique (bit Zéro) "Z" du registre d'état à avoir la valeur "1". Le registre d'état contient aussi d'autres bits : signe du résultat, retenu de l'addition ou report de soustraction, ...

V.1.3. Compteur de programme

Le compteur de programme contient l'adresse de l'instruction suivante en mémoire qui doit être exécutée. Autrement dit, il doit indiquer au processeur la prochaine instruction à exécuter. Le registre compteur de programme est constamment modifié après l'exécution de chaque instruction afin qu'il pointe sur l'instruction suivante. Les microprocesseurs de la famille Intel dépendent entièrement du registre compteur de programme pour connaître l'instruction suivante.

V.1.4. Registre d'instruction

Chaque opération que le microprocesseur doit effectuer est codée (c'est-à-dire pour chaque instruction on assigne un code qui ne peut pas être modifié ni changé par un autre code) appelé «instruction code» ou «opération code». Pour exécuter une instruction le microprocesseur transmet l'adresse se trouvant dans le registre compteur de programme à la mémoire, la mémoire retourne au microprocesseur l'octet adressé par ce dernier (le code de l'instruction). Celui-ci sera stocker dans un registre appelé registre d'instructions. Donc Le registre d'instructions contient la prochaine instruction à être exécutée par le processeur. Cette instruction sera acheminée (par un bus de données) au décodeur d'instructions qui sera chargé de l'interpréter.

V.1.5. Décodeur d'instruction

C'est lui qui va interpréter l'instruction contenue dans le registre d'instruction. C'est-à-dire quelle est l'opération à effectuer (transfert, addition, soustraction, branchement,...) et comment aller chercher les opérandes requis pour cette opération (par exemple, les nombres à additionner). Le décodeur d'instructions communique alors avec l'unité de commandes et de contrôles qui pourra déclencher les événements en conséquence. Par exemple, si le décodeur du microprocesseur 8085 reçoit l'octet 00000110 (le MVI B, donnée 8 bits), il sait que le processeur doit aller chercher un autre octet en mémoire (donnée) pour compléter l'instruction.

V.1.6. Registres d'adresses

Ces registres servent à gérer l'adressage de la mémoire. En effet le processeur peut utiliser un registre ou une paire de registres pour accéder à un emplacement mémoire, et puisque les registres peuvent être incrémenté ou décrémenter donc on peut accéder facilement à des données qui se trouvent en mémoire d'une manière adjacente (tel que les tableaux).

V.1.7. Pointeur de pile

Le pointeur de pile pointe vers le dessus de la pile (TOS : Top Of Stack) en mémoire. Une pile est un ensemble de données placées en mémoire de manière à ce que seulement la donnée du "dessus" soit disponible à un instant donné. Pour aller chercher la donnée sous celle du dessus par exemple, on doit d'abord enlever celle du dessus. Le rôle du pointeur de pile (et de la pile vers laquelle il pointe) est le suivant: Quand un processeur exécute une instruction, il est possible qu'il soit interrompu par une "Interruption" (c'est-à-dire un appel au processeur qui est prioritaire aux instructions du programme qu'il traite). Il doit alors arrêter de s'occuper de l'instruction qu'il traite présentement pour s'occuper de l'interruption. Quand l'interruption sera traitée, il retournera à l'instruction qu'il traitait quand il a été interrompu. Mais pour cela, il doit se rappeler de cette instruction ainsi que de l'état de certains registres au moment où il traitait l'instruction. Donc pour ne pas les perdre, il les placera temporairement dans une pile (à l'intérieur de la mémoire RAM par exemple) et pourra les récupérer une fois l'interruption traitée. Le pointeur de pile (SP) donne donc l'adresse en mémoire de cette pile temporaire.

Les piles offrent un nouveau moyen d'accéder à des données en mémoire principale, qui est très utilisé pour stocker temporairement des valeurs.

V.2. Unité arithmétique et logique (UAL)

Comme son nom l'indique, cette unité peut exécuter deux types d'opérations.

- **Opérations arithmétiques** : Elles incluent l'addition et la soustraction qui sont des opérations de base (une soustraction est une addition avec le complément à deux). Les données traitées sont considérées dans des représentations entières.
- **Opérations logiques** : Ces opérations sont effectuées bit à bit sur les bits de même poids de deux mots, tel que And, Or, Not, Ou exclusif, de même les opérations de rotation et de décalage (arithmétique et logique).

Elle reçoit ses opérandes (les octets qu'elle manipule) du bus de données. Celles-ci peuvent provenir de l'accumulateur A et des registres ou de la mémoire. A la fin d'une opération, l'UAL peut aller modifier certains bits du registre d'état.

V.3. Unité de contrôle et commande

Synchronisée par le signal de l'horloge, c'est elle qui déclenche les événements dans le processeur (on peut remarquer à ce sujet qu'elle est connectée à toutes les autres composantes du processeur). Par exemple, quand une information passe dans un bus, cette information est destinée à un seul endroit (par exemple, un registre). C'est donc l'unité de commande et de contrôle qui va "déverrouiller" l'entrée de cette destination pour que l'information qui circule sur le bus puisse y entrer (et ne pas entrer ailleurs en même temps).

Il s'agit donc essentiellement d'un automate exécutant les différentes séquences propres à chaque instruction. Cet automate peut être réalisé de plusieurs façons (câbler ou micro-programmer et dans les deux cas le jeu d'instructions est fixe). La plupart des unités de traitement sont micro-programmées et donc à jeux d'instructions fixes.

VI. Fonctionnement d'un système à base de microprocesseur [5-8,13,15,16]

VI.1. Ecriture en mémoire (WRITE)

Pour écrire une donnée dans la mémoire (figure IV.1.a), le microprocesseur doit placer l'adresse de la donnée sur le bus d'adresses (son emplacement dans la mémoire) puis il place la donnée sur le bus de données et enfin génère le signal WRITE (ordre d'écriture dans la mémoire).

VI.2. Lecture de la mémoire (READ)

Pour lire une donnée de la mémoire (figure IV.1.b), le microprocesseur doit connaître son emplacement, en effet il dépose son adresse sur le bus d'adresses puis génère le signal READ (il demande une opération de lecture de la mémoire) alors la donnée sera acheminée vers le microprocesseur à travers le bus de données. La donnée sera stockée dans un registre dans le microprocesseur.

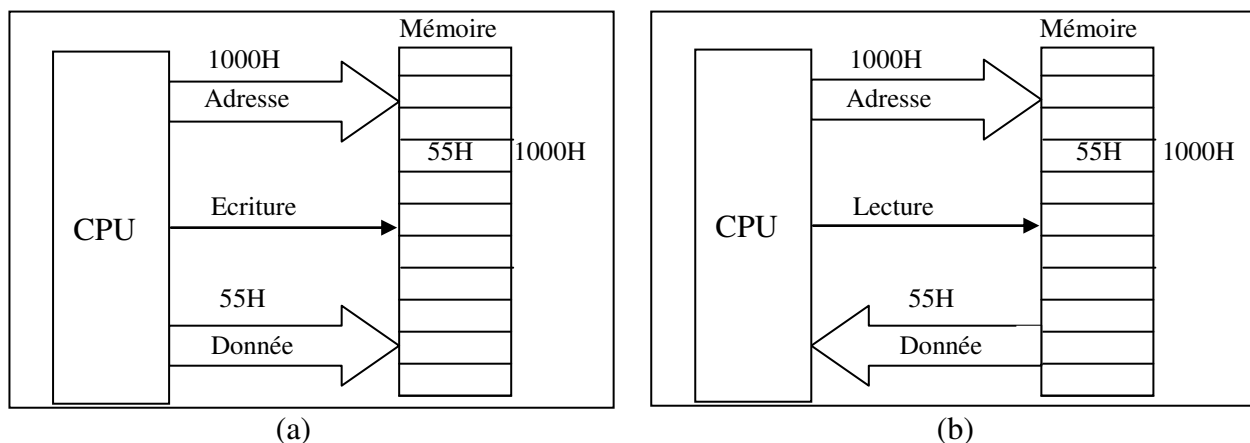


Figure IV.1 : Ecriture et lecture mémoire

VI.3. Communication avec les entrées/sorties

Il peut s'agir d'un flux d'informations de l'extérieur vers l'ordinateur (acquisition via le clavier, une connexion réseau, un disque dur, etc...), ou d'un flux de l'ordinateur vers l'extérieur (écran, réseau, disque, etc...).

Les données échangées entre un périphérique et le processeur transitent par l'interface (ou contrôleur) associé à ce périphérique. L'interface possède de la mémoire tampon pour stocker les données échangées (suivant le type d'interface, cette mémoire tampon fait de 1 seul octet à quelques Méga-octets). L'interface stocke aussi des informations pour gérer la communication avec le périphérique :

- des *informations de commande*, pour définir le mode de fonctionnement de l'interface: sens de transfert (entrée ou sortie), mode de transfert des données (par scrutation ou par interruption), etc. Ces informations de commandes sont communiquées à l'interface lors de la phase d'initialisation de celle-ci, avant le début du transfert.
- des *informations d'état*, qui mémorisent la manière dont le transfert est effectué (erreur de transmission, réception d'informations, etc). Ces informations sont destinées au processeur.

Lors de l'exécution des instructions d'entrées/sorties, le processeur met à 1 sa borne IO/\overline{M} et présente l'adresse E/S sur le bus d'adresse. Le signal IO/\overline{M} indique aux circuits de décodage d'adresses qu'il ne s'agit pas d'une adresse en mémoire principale, mais de l'adresse d'une interface d'entrées/sorties.

Remarque :

La communication entre le microprocesseur et les interfaces d'entrées/sorties peut être série (sur un seul fil bit par bit) ou parallèle (sur plusieurs fils).

VI.4. Interruptions

Les interruptions permettent au matériel (périphérique) de communiquer avec le processeur. Dans certains cas, on désire que le processeur réagisse rapidement à un évènement extérieur : arrivée d'un paquet de données sur une connexion réseau, frappe d'un caractère au clavier, modification de l'heure. Les interruptions sont surtout utilisées pour la gestion des périphériques de l'ordinateur, en effet les systèmes à base de microprocesseurs peuvent comporter plusieurs éléments matériels tels que l'écran, les lecteurs de CD, lecteurs de DVD, les ADC (Analog to digital converter) et DAC (digital to analog converter) etc... Mais la majorité de ces périphériques n'ont besoin du microprocesseur qu'à certains moments. Si un périphérique nécessite une intervention, il génère lui-même une demande d'interruption.

Une interruption est signalée au processeur par un signal électrique sur une borne spéciale. Lors de la réception de ce signal, le processeur “traite” l’interruption dès la fin de l’instruction qu’il était en train d’exécuter. Le traitement de l’interruption consiste soit :

- à l’ignorer et passer normalement à l’instruction suivante : c’est possible uniquement pour certaines interruptions, nommées ***interruptions masquables***. Il est en effet parfois nécessaire de pouvoir ignorer les interruptions pendant un certain temps, pour effectuer des traitements très urgents par exemple. Lorsque le traitement est terminé, le processeur ***démasque*** les interruptions et les prend alors en compte.
- à exécuter un ***traitant d’interruption*** (interrupt handler). Un traitant d’interruption est un programme qui est appelé automatiquement lorsqu’une interruption survient. L’adresse de début du traitant est donnée par la table des ***vecteurs d’interruptions*** (voir chapitre interruptions).

Remarque :

Parfois le microprocesseur est sollicité par plusieurs interruptions en même temps, pour répondre à ces appels un ordre de priorité est souvent pris en compte pour leurs traitements.

Les interruptions augmentent considérablement l’efficacité du processeur.

Les interruptions sont de deux types :

- Interruptions matérielles.
- Interruptions logicielles.

VI.5. Accès direct à la mémoire (DMA)

Lorsqu’un transfert en mémoire est nécessaire de la mémoire RAM à un port d’E/S, le microprocesseur lit le premier octet en mémoire et le charge dans l’un des registres du microprocesseur. Le microprocesseur écrit ensuite l’octet rangé précédemment sur le port d’E/S approprié.

Il en résulte que le microprocesseur effectue des opérations de lecture et d’écriture répétées. Ainsi, un certain temps est perdu entre le traitement de chaque octet. Pour remédier à ce problème, une procédure est mise au point pour l’accès direct à la mémoire (Direct Memory Access), qui permet de transférer des données de la mémoire RAM au port d’E/S sans passer par le microprocesseur. Pour cela, un contrôleur DMA, qui reprend le rôle du microprocesseur, c’est à dire qu’il gère les transferts de la RAM aux ports d’E/S.

CHAPITRE V

ETUDE D'UN MICROPROCESSEUR

I. Historique [17]

Le premier microprocesseur a été fabriqué par INTEL en 1971. C'était un 4 bits baptisé 4004 destiné à équiper des calculatrices de bureau. En 1972 INTEL produit le premier microprocesseur 8 bits baptisé 8008 par référence au précédent. Ce microprocesseur était destiné à répondre à un contrat de fabrication d'un terminal. En réalité le 8008 s'est avéré trop lent pour satisfaire au cahier des charge du terminal et INTEL a décidé de tenter le lancement de ce produit sur le marché grand public. L'énorme succès de cette initiative fut à l'origine de la fabrication massive des microprocesseurs. A la suite du succès du 8008, INTEL produisit dès 1974 le 8080 qui constituera le premier élément de la future famille de microprocesseurs de ce fabriquant. En 1974, MOTOROLA, autre fondeur de silicium, décide de lancer le 6800 qui constituera lui aussi le début d'une grande famille. Les années 70 voient alors apparaître de petites entreprises de fabrication de microprocesseurs souvent constituées par des transfuges des deux grandes compagnies. On peut notamment citer MOSTEK avec son 6502 très inspiré du 6800 et ZILOG avec son Z80 qui constitue une amélioration technique du 8080 (augmentation du nombre de registres, simplification de l'alimentation...).

Très vite les 8 bits commencent à atteindre la limite de leurs performances et les constructeurs se penchent sur de nouvelles solutions. Les principaux reproches faits aux 8 bits sont : - peu d'espace mémoire accessible (64K octets) - peu de types d'informations manipulés (1 ou 2 octets) - pas adaptés aux architectures multiprocesseurs - peu de modes d'adressage Pour toutes ces raisons les constructeurs commencent à étudier une nouvelle génération de microprocesseurs. De sorte que l'on verra apparaître des microprocesseurs 8 bits plus puissants permettant de manipuler plus facilement des informations sur 16 bits et offrant de nouvelles possibilités d'adressage comme le 6809 de MOTOROLA ou le 8085 d'INTEL.

II. Rôle du microprocesseur

Le microprocesseur est chargé d'effectuer les fonctions suivantes :

- Fournir les signaux de synchronisation et de commande à tous les éléments du système,
- Prendre en charge les instructions et les données dans la mémoire,
- Transfert des données entre la mémoire et les dispositifs d'E/S et vice versa,

- Décoder les instructions,
- Effectuer les opérations arithmétique et logiques commandées par les instructions,
- Réagir aux signaux de commande produits par les entrées/sorties comme le signal "Reset" et les interruptions.

III. Etude du microprocesseur 8085 [16]

Le microprocesseur Intel 8085 est apparu en 1976. Il est équipé d'un bus de données de 8 bits et un bus d'adresses de 16 lignes d'adresses.

Il était compatible au niveau du code binaire avec le plus célèbre Intel 8080, mais demandait moins de matériel environnant, ce qui permit la création de micro-ordinateurs plus simples et moins chers à construire.

Le « 5 » dans le numéro du modèle provient du fait que les 8085 exigeaient seulement une alimentation de +5V plutôt que les +5V, -5V et +12V exigés par les 8080. Cependant, il était plus lent que le 8080. Sa vitesse était de 1.5 million d'instructions à la seconde avec une horloge à 6.144 MHz et 4 cycles par instruction.

Le 8085 était vendu dans un pack à 40 broches, ce qui lui permettait d'utiliser un bus d'adresses sur 16 bits et un bus de données sur 8 bits, donnant un accès facile à 64 KO de mémoire. Il a sept registres de 8 bits (dont six peuvent être combinés dans trois registres de 16 bits), un pointeur de pile sur 16 bits et un compteur de programme sur 16 bits. Le 8085 possède 256 ports d'entrées/sorties accessibles par des instructions dédiées.

IV. Architecture externe du 8085 [16]

Le microprocesseur 8085 se présente sous la forme d'un boîtier DIP (Dual In-line Package) à 40 broches. La figure V.1 montre le schéma de brochage du 8085. Tout les signaux peuvent être classés en six groupes : (1) Bus d'adresses, (2) Bus de données, (3) Bus de contrôle et signaux d'état, (4) Alimentation et générateur de fréquence, (5) Signaux d'interruptions (6) Ports d'E/S sériels.

IV.1. Bus d'adresses et bus de données

Le nombre de lignes d'adresses du 8085 est 16 lignes $A_{15}-A_0$ dont le poids faible est multiplexé temporellement avec le bus de données AD_7-AD_0 . D'où la nécessité d'un démultiplexage pour obtenir séparément les bus d'adresses et de données :

- ✓ 8 bits de données (microprocesseur 8 bits) ;
- ✓ 16 bits d'adresses, d'où $2^{16} = 64$ Ko d'espace mémoire adressable par le 8085.

Le démultiplexage des signaux AD_7 à AD_0 se fait en mémorisant l'adresse lorsque celle-ci est présente sur le bus Adresses/Données, à l'aide d'un verrou (latch), ensemble de bascules D. La commande de mémorisation de l'adresse est générée par le microprocesseur : c'est le signal **ALE** (Address Latch Enable).

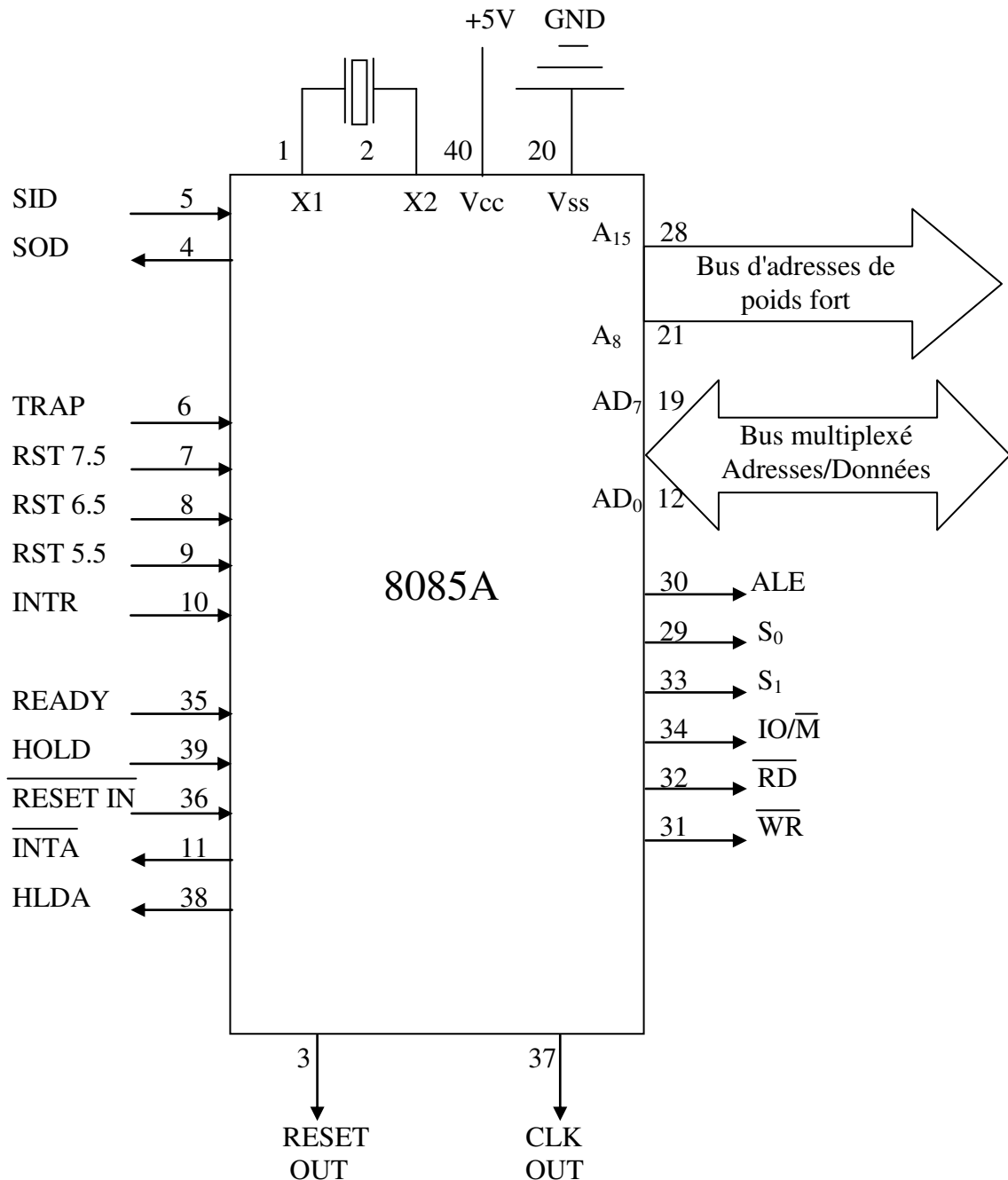


Figure V.1 : Architecture externe du 8085

IV.2. Signaux de contrôle et d'état

ALE : A l'état haut, implique que les bits présents sur le bus A/D sont les lignes d'adresses. Un circuit 74373 mémorise ces bits à sa sortie.

RD : Read, signal de lecture d'une donnée, actif niveau bas.

WR : Write, signal d'écriture d'une donnée, actif niveau bas.

$\text{IO}/\overline{\text{M}}$: Input-Output / Memory, indique si le 8085 adresse la mémoire ($\text{IO}/\overline{\text{M}} = 0$) ou les entrées/sorties ($\text{IO}/\overline{\text{M}} = 1$).

S_1 et S_0 : Signaux d'état indiquant le type d'opération en cours sur le bus. Rarement utilisés dans les petits systèmes.

IV.3. Alimentation et générateur de fréquence

Vcc : Alimentation +5V

GND : Masse de référence

X_1 et X_2 : Un Quartz est relié aux deux broches pour générer un signal carré périodique. Pour générer une fréquence de 3MHz, le quartz doit avoir une fréquence de 6MHz.

CLK (OUT) : Cette broche peut être utilisée comme signal d'horloge pour d'autres circuits.

IV.4. Signaux d'interruptions

Le 8085 possède cinq signaux d'interruptions.

INTR (Interrupt Request) : C'est le signal envoyé par une interface indiquant une demande d'interruption.

$\overline{\text{INTA}}$: Le 8085 répond à INTR en envoyant 0 sur la ligne $\overline{\text{INTA}}$ (Interrupt Acknowledge).

RST7.5 RST6.5 et RST5.5 : Interruptions de redémarrage. Ce sont des interruptions vectorisées qui transfèrent le contrôle à une position mémoire spécifique. Ces lignes sont prioritaire que INTR. Les priorités sont classées selon l'ordre décroissant de 7.5, 6.5 à 5.5.

TRAP : (Non Maskable Interrupt) : interruption prioritaire par rapport à INTR.

HOLD et HLDA : signaux de demande d'accord d'accès direct à la mémoire (DMA).

READY : entrée de synchronisation avec la mémoire.

RESET IN : Quand ce signal est à 0, le compteur de programme est remis à 0, les bus sont en haute impédance et le microprocesseur redémarre.

RESET OUT : Ce signal indique que le microprocesseur vient d'être redémarrer, ce signal peut être utilisé par d'autres circuits.

IV.5. Ports d'E/S sériels

Le 8085 possède deux signaux pour l'implémentation de la transmission série : SID (Serial Input Data) et SOD (Serial Output Data). Dans la transmission série, les bits de données sont transmises sur une seule ligne l'un après l'autre comme dans le cas de la ligne téléphonique.

V. Démultiplexage des bus AD₇-AD₀

La figure IV.2 montre l'utilité des latches pour le démultiplexage de bus. Le bus AD₇-AD₀ est connecté aux entrées des latches 74LS373. ALE est connectée à la broche de validation (G) des latches. Quand ALE est à l'état haut, les latches sont transparents; ça veut dire que les sorties des latches changent en fonction des entrées. Quand le microprocesseur envoie un 0 sur ALE, l'adresse est mémorisée jusqu'au prochain ALE et la sortie des latches représente le bus d'adresses de poids faible A₇-A₀. Le schéma de la figure V.2 indique une lecture mémoire de la valeur 4FH à partir de l'adresse 2005H.

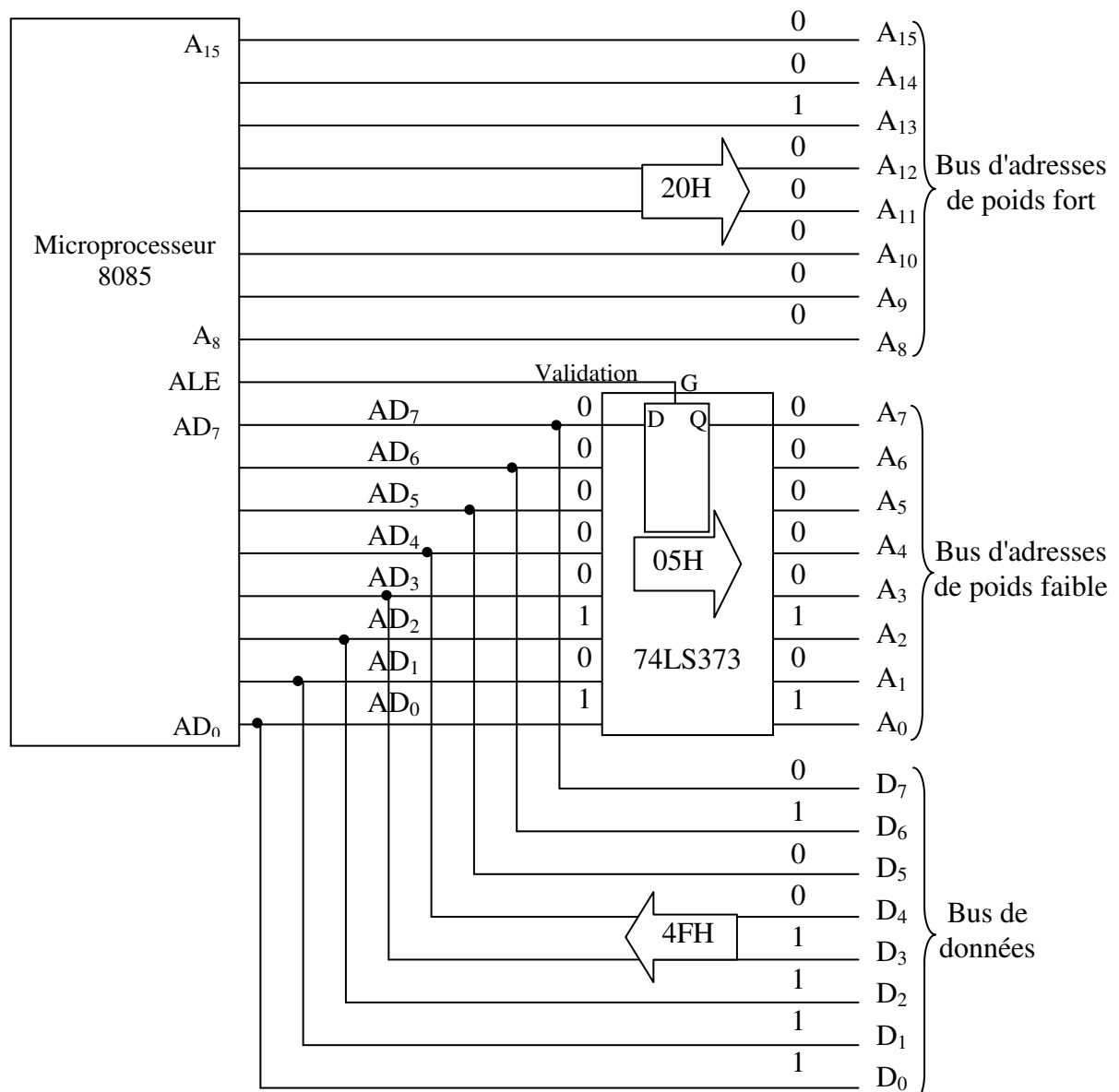


Figure V.2 : Démultiplexage des bus d'adresses et de données AD₇-AD₀

VI. Architecture interne du 8085 [16]

La structure interne du microprocesseur est d'une très grande complexité. Pour essayer de la comprendre on peut cependant la diviser en quatre grandes sections : Accumulateur, registres généraux, UAL, Registre d'instructions, Registres d'adresses, décodeur d'instructions, contrôleur d'interruptions, contrôleur des entrées / sortie série et séquenceur.

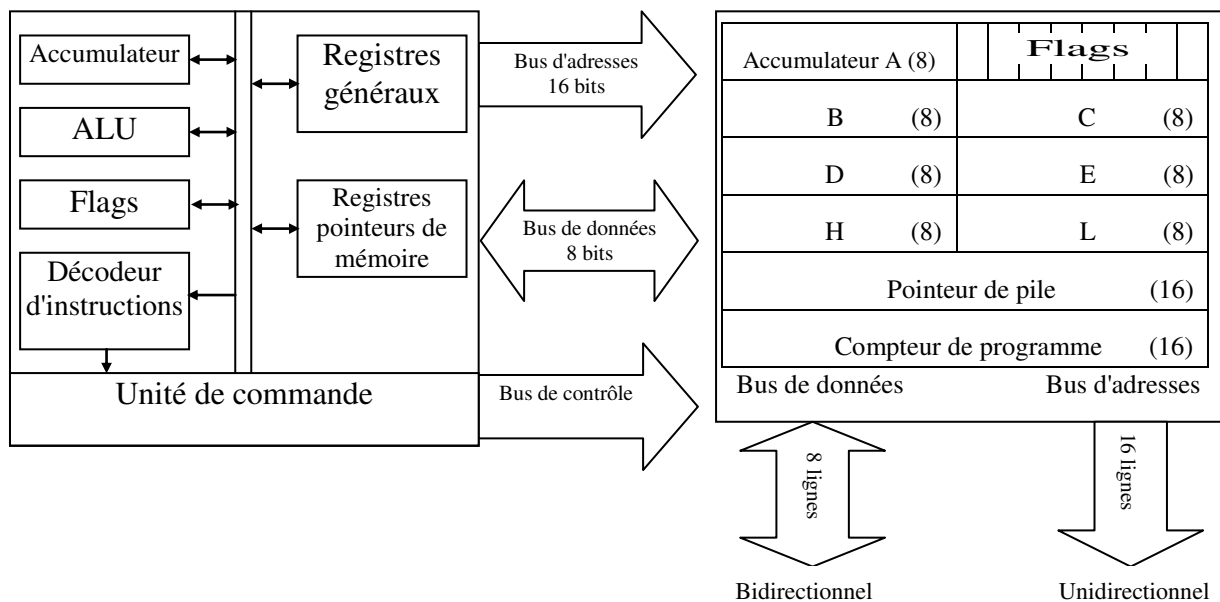


Figure V.3 : Architecture interne de base du microprocesseur 8085

Tous les blocs sont connectés avec des connexions internes appelés bus interne. Les opérations arithmétiques et logiques sont réalisées par l'unité arithmétique et logique. Les résultats sont stockés dans l'accumulateur et les drapeaux (flags). Le 8085 utilise le bus d'adresses pour envoyer les adresses vers les mémoires et périphériques, les huit lignes de données sont les lignes de transfert vers/des mémoires/interfaces d'E/S.

VI.1. Registres de base du 8085

Accumulateur : C'est le registre A de 8 bits. Il fait partie de l'unité arithmétique et logique. Il est utilisé pour le stockage des données et des résultats des opérations arithmétiques et logiques

- B, C, D, E, H et L : Registres généraux de 8 bits multi usages. Ils ont une utilité proche de la RAM, mais étant interne au microprocesseur, ils ont des fonctionnalités additionnelles. Ces registres peuvent être combinés comme des registres pairs BC, DE et HL pour réaliser des opérations sur 16 bits.

- PC (Program Counter): Compteur de Programme, c'est un registre de 16 bits qui supporte l'adresse du prochain octet mémoire à chercher. Quand un octet est chargé, le PC s'incrémente automatiquement par 1 (+1), pour qu'il pointe sur le prochain octet à charger.
- SP (Stack Pointer) : Pointeur de pile de taille 16 bits, il a pour rôle d'indiquer au microprocesseur l'adresse de la prochaine case disponible dans la pile.

Registre d'état: C'est l'ensemble des drapeaux (flags) qui forment le registre d'état de 8 bits. Le registre d'état associé à l'accumulateur forme le contexte ou PSW (Program Status Word) de 16 bits.

On mentionne ici les indicateurs (flags) affectés par une instruction arithmétique ou logique, on utilise les notations suivantes :

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
S	Z		AC		P		CY

Figure V.4 : Bits du registre d'état

Les flags sont des drapeaux ou des indicateurs d'état (registre d'état). Ils indiquent l'état d'une opération effectuée par l'unité arithmétique et logique. Ce registre contient 5 bits S, Z, AC, P et CY.

S: Sign-flag (Bit de signe). C'est le bit D₇ du résultat de l'opération.

Z: Zero-flag. Si le résultat de l'opération égale à 0 ce bit est mis à 1. Autrement il est mis à 0.

AC : Auxilliary Carry-flag. Dans une opération arithmétique, quand une retenue est générée par le bit D₃ à D₄ le AC est mis à 1 sinon il est mis à 0. Il est utilisé dans les codes BCD et il n'est pas utilisé par le programmeur.

P : Parity-flag. Après une opération arithmétique ou logique, si le résultat contient un nombre pair de 1, le drapeau P est affecté par 1. Si le résultat contient un nombre impair de 1, le drapeau P est affecté par 0.

CY : Carry-flag. Dans une opération arithmétique, quand une retenue lors d'une addition ou un report lors d'une soustraction est générée par les huit bits le CY est mis à 1 sinon il est mis à 0. Ce bit est accessible par le programmeur.

VI.2. Format d'une instruction

Le programme est une suite d'instruction. Une instruction est un mot binaire décrivant l'action que doit exécuter le microprocesseur, par exemple copier une case mémoire dans une autre, ajouter le contenu de deux cases mémoires etc.

Ces instructions sont des mots binaires qu'on écrit généralement en hexa, mais pour une plus grande facilité on les désignera par leurs « mnémoniques ». La mnémonique est une abréviation en caractère latin.

Une instruction est formée par un ou plusieurs octets. Le premier champ représente l'opération à effectuer et qui est appelée l'**opcode**. Dans le 8085 l'opcode a toujours une taille de 1 octet. L'instruction est formée soit par un l'opcode ou par l'opcode et un opérande. Ce dernier peut être spécifié par différentes formes. Il peut être une donnée de 8 bits (ou 16 bits), un registre interne ou une adresse d'une position mémoire.

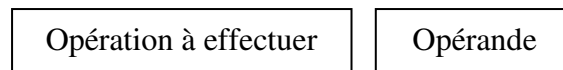


Figure V.5 : Format d'une instruction

L'ensemble des instructions du 8085 est divisé en trois groupes :

1. Instructions de tailles 1 octet (pas d'opérande)
2. Instructions de tailles 2 octets (opérande de 8 bits)
3. Instructions de tailles 3 octets (opérande de 16 bits)

Exemple 1 :

CMA : Complémenter le registre A.

L'opcode correspondant est 2FH. CMA est la mnémonique. Cette instruction n'a pas d'opérande. Sa taille est donc de 1 octet (une position mémoire).

Exemple 2 :

MVI A, 28H : Ecrire la valeur 28H dans le registre A.

L'opcode de MVI A est 3EH. L'opérande 8 bits est 34h. La taille de l'instruction est 2 octets.

Exemple 3 :

LDA 1234H : Charger le contenu de l'adresse 1234H dans l'accumulateur A.

L'opcode correspondant est 3AH. LDA est la mnémonique. Cette instruction possède l'opérande 16 bits 1234H. Sa taille est donc de 3 octets.

Donc si L'adresse de CMA est 1000H, on obtiendra ce qui suit :

1000H	2FH	--> CMA	1003H	3AH	-->LDA 1234H
1001H	3EH	--> MVI A,28H	1004H	34H	
1002H	28H		1005H	12H	

VI.3. Format de l'opcode

Toutes les opérations, les registres les flags sont identifiés par un code spécifique. Par exemple les registres internes sont comme suit:

Codes	Registres	Codes	Registres pairs
000	B	00	BC
001	C		
010	D	01	DE
011	E		
100	H	10	HL
101	L		
111	A	11	AF ou SP
110	Réservé pour les opérations mémoire		

Tableau V.1 : Codage des registres

Quelques codes d'instructions sont identifiés dans le tableau V.2 comme suit:

Fonction	Code de l'opération
1. Rotation à gauche de chaque bit de l'accumulateur par une position	00000111=07H
2. Additionner les contenus du registre et de l'accumulateur	10000 SSS
3. Copier (MOV) le contenu du registre Rs (source) au registre Rd (destination)	01 DDD SSS Rd Rs

Tableau V.2 : Codage des instructions

Cette instruction est complétée par ajouter le code du registre. Par exemple :

Le code de ADD B est : 10000 000=80H.

Cette instruction est complétée par ajouter le code des deux registres. Par exemple :

Le code de MOV C,A est : 01 001 111=4FH

VI.4. Modes d'adressage

Les façons de désigner les opérandes constituent les "modes d'adressage". Le microprocesseur 8085 possède quatre modes d'adressage :

- Mode d'adressage par registre.
- Mode d'adressage immédiat.
- Mode d'adressage direct.
- Mode d'adressage indirect.

VI.4.1. Adressage par registre

Ce mode d'adressage concerne tout transfert ou toute opération, entre deux registres de 8 bits. Dans ce mode l'opérande sera stocké dans un registre interne au microprocesseur.

Exemple :

MOV A, B ; cela signifie que l'opérande stocké dans le registre B sera transféré vers le registre A. Quand on utilise l'adressage par 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.

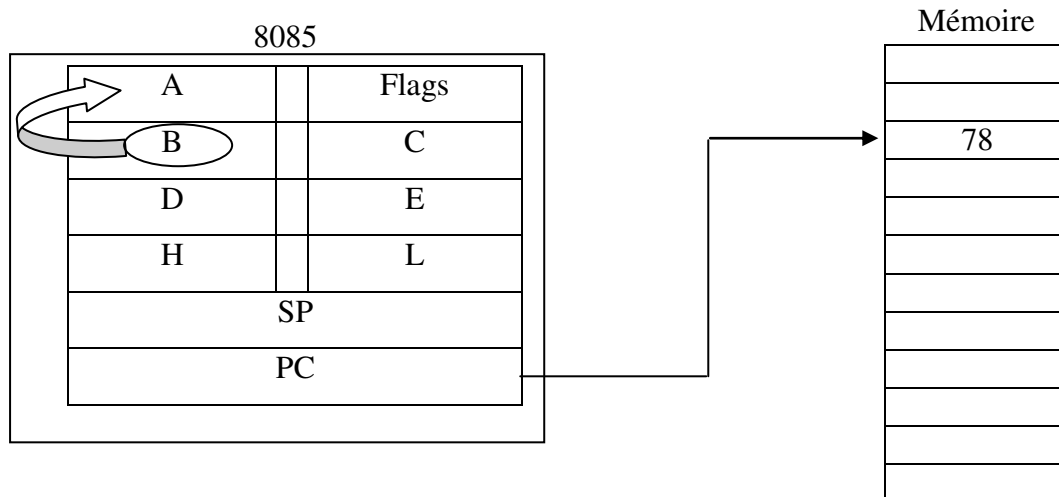


Figure V.6 : Adressage par registre

VI.4.2. Adressage immédiat

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

MVI A,50H ; cela signifie que la valeur 50H sera stockée immédiatement dans le registre A.

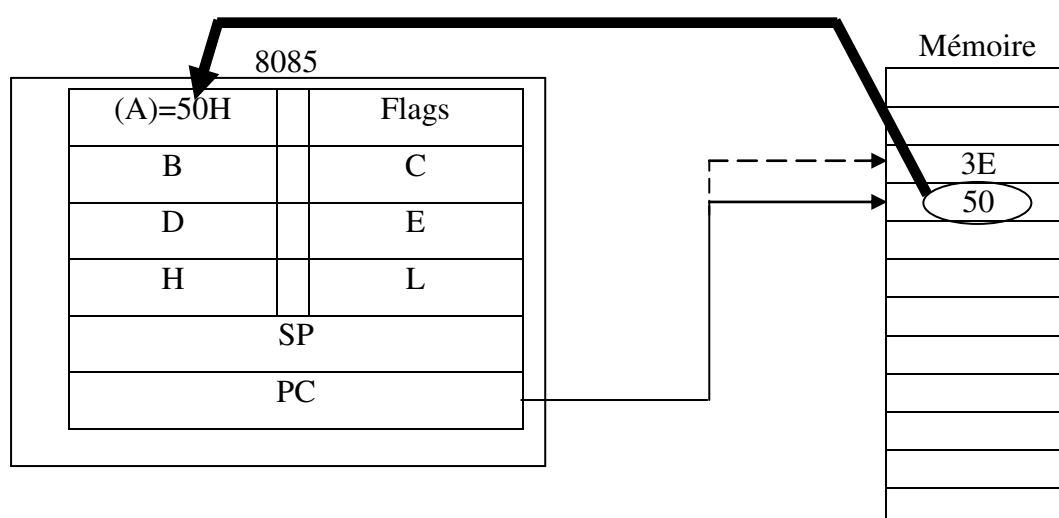


Figure V.7 : Adressage immédiat

Remarque :

Pour charger A avec la valeur 01H, on doit utiliser MVI A,01H. Pour charger A avec la valeur -01H, on doit utiliser MVI A,FFH. Tel que FFH est le complément à 2 de 01H sur 8 bits.

VI.4.3. Adressage direct

Dans ce mode on spécifie directement l'adresse de l'opérande dans l'instruction. Par exemple pour charger l'accumulateur A par le contenu de l'adresse 2000H, on doit écrire : LDA 2000H

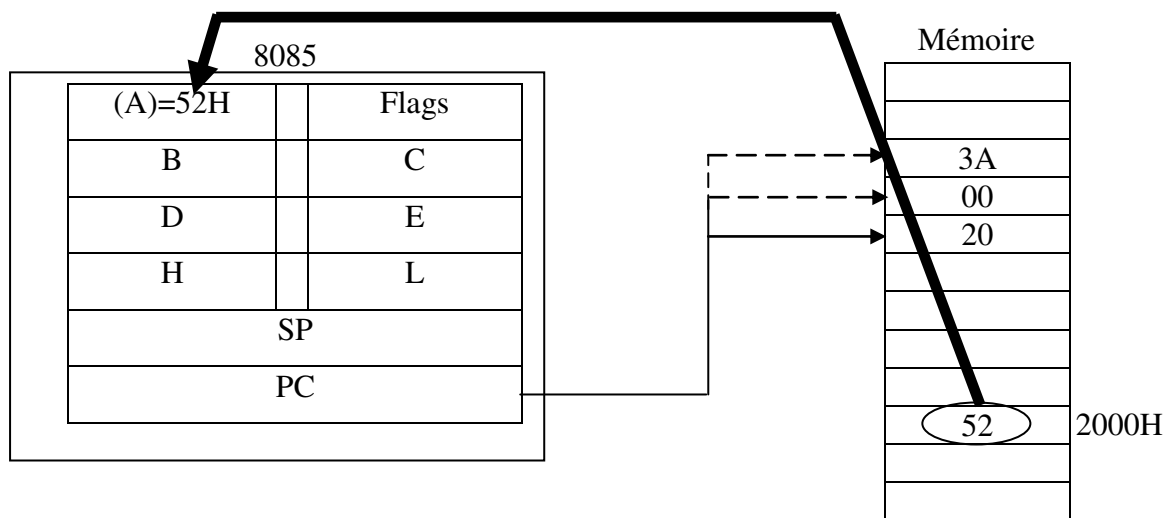


Figure V.8 : Adressage direct

VI.4.4. Adressage indirect

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 pair B, D, H ou SP.

Exemple :

LXI H,1234H ; adressage immédiat codé par 21H 34H 12H

MOV A,M ; adressage indirect codé par 7E

Pour la deuxième instruction (MOV A,M), le contenu de la case mémoire dont l'adresse se trouve dans le registre pair H ((HL)=1234H) est mis dans l'accumulateur A.

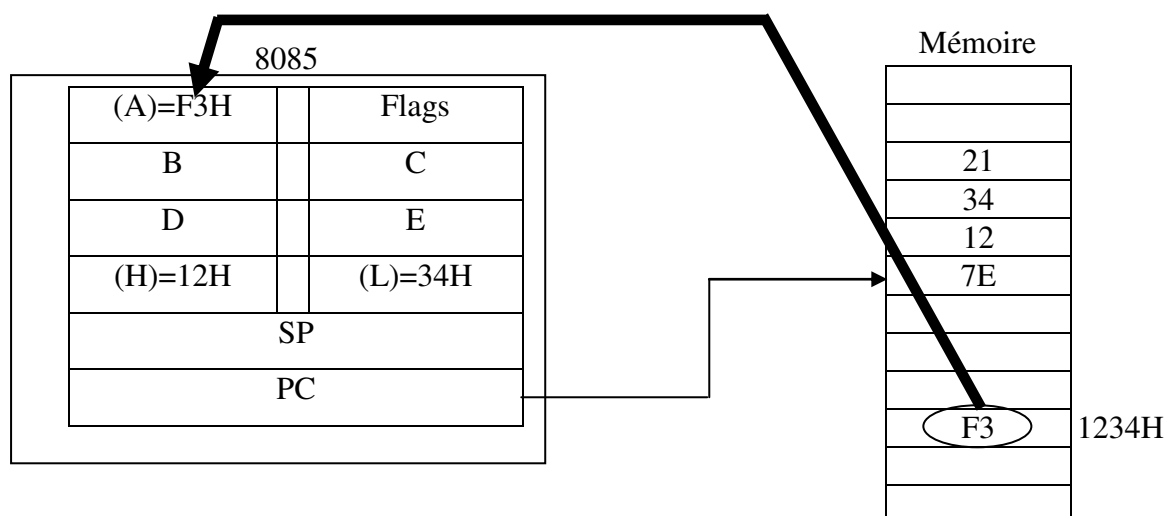


Figure V.9 : Adressage indirect

VII. Jeu d'instructions du 8085 [16]

Pour la déclaration des donnée on doit insérer après la fin du programme (HLT) les mnémoniques # ORG et # DB. Par exemple on veut initialiser la zone mémoire commençant à partir de l'adresse mémoire C050H avec les données 85H,54H,30H,5EH,3AH,BFH, on doit écrire les lignes suivantes :

```
# ORG C050
```

```
# DB 85H,54H,30H,5EH,3AH,BFH
```

Les instructions du 8085 sont divisées en trois groupes :

- Instructions de transfert
- Instruction arithmétiques et logiques
- Instructions de branchement

VII.1. Instructions de transfert

Ce premier groupe d'instructions permet de transférer (copier) des données d'un endroit un autre (d'un registre ou mémoire ou interface d'E/S) appelé source vers un autre registre (ou E/S ou mémoire) appelée destination. En effet, le contenu de la source n'est pas déplacé, mais copié dans la destination sans modifier le contenu de la source. Les instructions de transfert sont résumées comme suit:

1. MOV : Move : Copier un octet de données depuis un registre ou adresse mémoire

Syntaxe : MOV Rd,Rs ou MOV M,Rs ou MOV Rd,M

Tel que Rd : Registre destination Rs: Registre source et M adresse spécifiée par HL

Exemple :

```
MOV C,D ou MOV M,B
```

2. MVI : Move Immediate : Charger un octet de données

Syntaxe : MVI Rd, donnée 8 bits

Tel que Rd : Registre destination

Exemple :

```
MVI A,14H
```

3. OUT : Output to port : Envoyer un octet de données depuis A vers une interface de sortie

Syntaxe : OUT adresse du port de sortie

Exemple :

```
OUT 08H
```

4. IN : Input from Port : Recevoir un octet de données d'une interface d'entrée dans A

Syntaxe : IN adresse du port d'entrée

Exemple :

IN 09H

5. LXI : Load register pair Immediate : Charger un mot (16bits) de données

Syntaxe : LXI Rp, donnée 16 bits

Tel que Rp : Registre pair

Exemple :

LXI B,1400H ; (B)=14H et (C)=00H

6. LDA : Load Accumulator : Charger l'accumulateur par le contenu d'une adresse mémoire

Syntaxe : LDA Adresse 16 bits

Exemple :

LDA 2000H

7. LDAX : Load Accumulator : Charger l'accumulateur par le contenu d'une adresse mémoire spécifié dans un registre B ou D

Syntaxe : LDAX Rp

Tel que Rp : Registre pair B ou D

Exemple :

LDAX B

8. STA : Store Accumulator : Stocker l'accumulateur dans une case mémoire

Syntaxe : STA Adresse 16 bits

Exemple :

STA 3000H

9. STAX : Store Accumulator : Stocker l'accumulateur dans une case mémoire dont l'adresse est spécifié dans un registre B ou D

Syntaxe : STAX Rp

Tel que Rp : Registre pair B ou D

Exemple :

STAX D

10. LHLD : Load HL registers direct : Charger le registre pair H par le contenu de la case mémoire.

Syntaxe : LHLD Adresse 16 bits

Exemple :

LHLD 2000H

11. SHLD : Store HL registers direct : Charger le contenu de la case mémoire par le contenu du registre pair H.

Syntaxe : SHLD Adresse 16 bits

Exemple :

SHLD 3000H

12. XCHG : Exchange the contents of HL with DE : Echange les contenus de HL et DE.

Syntaxe : XCHG

13. XTHL : Exchange the top of the stack with HL : Echange le top de la pile avec HL.

Syntaxe : XTHL

14. SPHL : Copy HL registers into the stack pointer : Copie les contenus des registres HL dans le registre SP.

Syntaxe : SPHL

15. PCHL : Copy HL registers into the program counter : Copie les contenus des registres HL dans le registre CP.

Syntaxe : PCHL

VII.2. Instructions arithmétiques

Le microprocesseur 8085 effectue plusieurs opérations arithmétiques à savoir : l'addition (ADD ou ADI), la soustraction (SUB ou SUI), l'incrémentation 8 bits (INR), l'incrémentation 16 bits (INX), la décrémentation 8 bits (DCR) et la décrémentation 16 bits (DCX).

Pour les instructions ADD, ADI, SUB et SUI, le microprocesseur assume que l'accumulateur A est par défaut l'un des deux opérandes et le résultat sera stocker dans l'accumulateur A. Les flags seront affectés.

Les instructions d'incrémentation et de décrémentation affectent le contenu du registre spécifié et affectent tout les flags à l'exception du CY.

1. ADD : Addition : Additionner le contenu de l'accumulateur A avec le contenu du registre ou case mémoire.

Syntaxe : ADD R ou ADD M

Tel que R : Registre 8 bits et M : adresse d'une case mémoire

Exemple :

MVI A,04H

MVI B,03H

ADD B

Ce qui donne $(A)=(A)+(B)=04H+03H=07H$ S=0 Z=0 CY=0

2. ADI : Addition Immédiate : Additionner le contenu de l'accumulateur A avec une valeur

Syntaxe : ADI donnée 8 bits

Tel que R : Registre 8 bits

Exemple :

MVI A,FFH

ADI 01H

Ce qui donne $(A)=(A)+01H=FFH+01H=00H$ S=0 Z=1 CY=1

3. SUB : Soustraction : Soustraire le contenu du registre du contenu de l'accumulateur A ou case mémoire.

Syntaxe : SUB R ou SUB M

Tel que R : Registre 8 bits et M : adresse d'une case mémoire

Exemple :

MVI A,FFH

MVI B,03H

SUB B

Ce qui donne $(A)=(A)-(B)=FFH-03H=FCH$ S=1 Z=0 CY=0

4. SUI : Soustraction Immédiate : Soustraire une valeur du contenu de l'accumulateur A

Syntaxe : SUI donnée 8 bits

Tel que R : Registre 8 bits

Exemple :

MVI A,05H

SUI 02H

Ce qui donne $(A)=(A)-02H=05H-02H=03H$ S=0 Z=0 CY=0

5. INR : Incrémentation : Incrémenter le contenu d'un registre ou case mémoire par 1 (+ 1)

Syntaxe : INR R ou INR M

Tel que R : Registre 8 bits et M : adresse d'une case mémoire

Exemple :

MVI A,05H

INR A

Ce qui donne $(A)=(A)+1=05H-1=06H$ S=0 Z=0

6. DCR : Décrémentation : Décrémenter le contenu d'un registre ou case mémoire par 1 (- 1)

Syntaxe : DCR R ou DCR M

Tel que R : Registre 8 bits et M : adresse d'une case mémoire

Exemple :

MVI A,01H

DCR A

Ce qui donne $(A)=(A)-1=01H-1=00H$ S=0 Z=1

7. INX : Incrémentation 16bits : Incrémenter un registre pair ou SP (16bits)

Syntaxe : INX Rp

Tel que Rp : Registre pair (B, D ou H) ou SP

Exemple :

LXI SP,2000H

INX SP

Ce qui donne $(SP)=(SP)+1=2000H+1=2001H$ S=0 Z=1

8. DCX : Décrémentation 16bits : Décrémenter un registre pair ou SP (16bits)

Syntaxe : DCX Rp

Tel que Rp : Registre pair (B, D ou H) ou SP

Exemple :

LXI B,3000H

DCX B

Ce qui donne $(B)=(B)-1=3000H-1=2FFFH$ S=0 Z=0

9. ADC : Add register contents with carry : Additionner registre 8 bits ou case mémoire avec

A et CY

Syntaxe : ADC R ou ADC M

Tel que R : Registre 8 bits et M: adresse d'une case mémoire

Exemple :

MVI A,30H

MVI B,F0H

ADD B ; (A)=30H+F0H=20H et CY=1

ADC B ; (A)=20H+F0H+1=11H

10. ACI : Add immediate 8 bits data with carry : Additionner une donnée 8 bits avec A et CY**Syntaxe :** ACI donnée 8 bits**Exemple :**

MVI A,30H

MVI B,F0H

ADD B ; (A)=30H+F0H=20H et CY=1

ACI 30H ; (A)=20H+30H+1=51H

11. SBB : Subtract register contents with borrow : Soustraire registre 8 bits ou case mémoire et report CY de A.**Syntaxe :** SBB R ou SBB M

Tel que R : Registre 8 bits et M : adresse d'une case mémoire

Exemple :

MVI A,30H

MVI B,20H

ADI FFH ; (A)=30H+FFH=2FH et CY=1

SBB B ; (A)=2FH-20-1=0EH

12. SBI : Subtract immediate 8 bits data with borrow : Soustraire donnée 8 bits et report CY de A.**Syntaxe :** SBI donnée 8 bits**Exemple :**

MVI A,30H

ADI FFH ; (A)=30H+FFH=2FH et CY=1

SBI 20H ; (A)=2FH-20-1=0EH

13. DAA : Decimal Adjust Accumulator : Ajustement décimal de l'accumulateur**Syntaxe :** DAA

Exemple :

MVI A,38H

MVI B,43H

ADD B ; A=7BH

DAA ; A= 81H

VII.3. Instructions logiques

Le microprocesseur est un circuit logique programmable. L'ensemble des opérateurs logiques est composé de : AND, OR, XOR et NOT. Le résultat de l'opération sera stocker dans l'accumulateur A. Ces opérateurs engendrent une mise à zéro du flag CY. Ils modifient l'état de S, Z et P suivant le résultat à l'exception de CMA. L'opérateur AND sert au masquage à 0 et l'opérateur OR sert au masquage à 1. Les opcodes de ces opérateurs sont :

1. ANA : And : And logique entre l'accumulateur A et le contenu du registre ou case mémoire.

Syntaxe : ANA R ou ANA M

Tel que R : Registre 8 bits et M : adresse d'une case mémoire

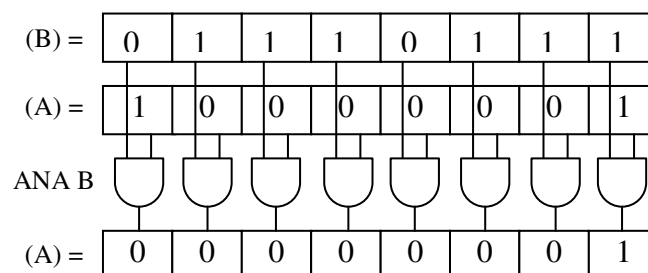
Exemple :

MVI A,81H

MVI B,77H

ANA B

Ce qui donne (A)=01H. S=0 Z=0 P=0



2. ANI : And Immédiate : And logique entre le contenu de l'accumulateur A et une valeur 8 bits.

Syntaxe : ANI donnée 8 bits**Exemple :**

MVI A,55H

ANI 01H

Ce qui donne (A)=54H. S=0 Z=0 P=0

3. ORA : Or : Or logique entre l'accumulateur A et le contenu du registre ou case mémoire.

Syntaxe : ORA R ou ORA M

Tel que R : Registre 8 bits et M : adresse d'une case mémoire

Exemple :

MVI A,81H

MVI B,7EH

ORA B

Ce qui donne (A)=FFH. S=1 Z=0 P=1

4. ORI : Or Immédiate : Or logique entre le contenu de l'accumulateur A et une valeur 8 bits.

Syntaxe : ORI donnée 8 bits

Exemple :

MVI A,55H

ORI 02H

Ce qui donne (A)=57H. S=0 Z=0 P=0

5. XRA : Xor : Ou exclusif entre l'accumulateur A et le contenu du registre ou une case mémoire.

Syntaxe : XRA R ou XRA M

Tel que R : Registre 8 bits et M : adresse d'une case mémoire

Exemple :

MVI A,80H

MVI B,7EH

XRA B

Ce qui donne (A)=FEH. S=1 Z=0 P=0

6. XRI : Xor Immédiate : Ou exclusif entre le contenu de l'accumulateur A et une valeur 8bits.

Syntaxe : XRI donnée 8 bits

Exemple :

MVI A,55H

XRI 02H

Ce qui donne (A)=57H. S=0 Z=0 P=0

7. CMA : Complémenter A : Complémenter l'accumulateur A (inverser tous les bits de A).

Syntaxe : CMA

Exemple :

MVI A,55H

CMA

(A)=0101 0101b, après l'exécution de CMA, on aura (A)=1010 1010b=AAH.

L'instruction CMA n'affecte aucun flag. Le mode d'adressage est implicite.

VII.4. Instructions logiques de rotation

Les instructions logiques concernées dans ce paragraphe consistent à des rotations des bits de l'accumulateur A. Ces rotations s'effectuent soit vers la gauche ou vers la droite avec ou sans Carry. Les opérations de rotation sont RLC, RAL, RRC et RAR. Le résultat de l'opération sera toujours stocker dans l'accumulateur A.

1. RLC : Rotate Accumulator Left : Rotation à gauche de A.

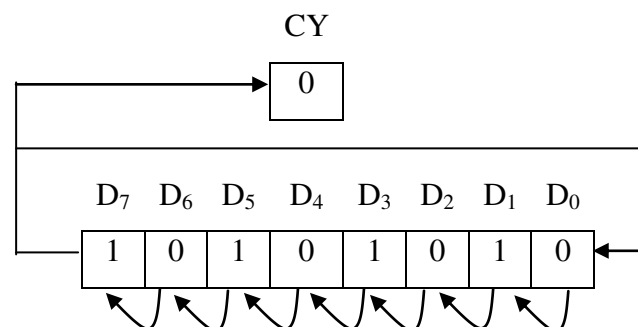
Chaque bit est décalé vers la gauche d'une seule position. Le bit D₇ deviendra D₀.

CY est modifié selon la valeur de D₇.

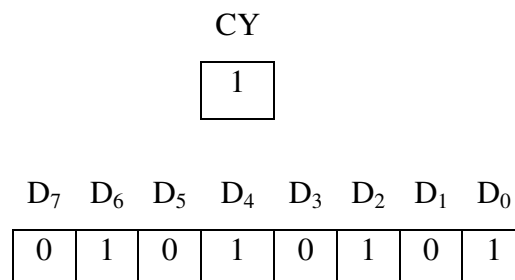
La rotation à gauche de A engendre la multiplication de la valeur de A par 2 à condition que le bit D₇ soit à 0.

Exemple :

1. Avant RLC (A)=AAH et CY=0



1. Après RLC (A)=55H et Y=1

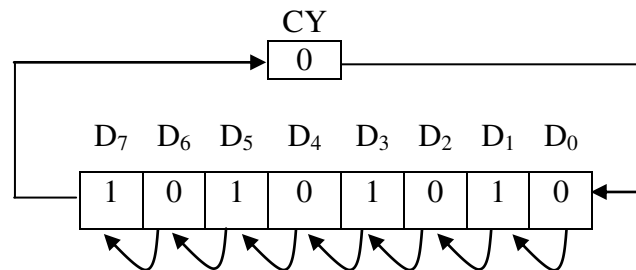


2. RAL : Rotate Accumulator Left Through Carry : Rotation à gauche de A par CY.

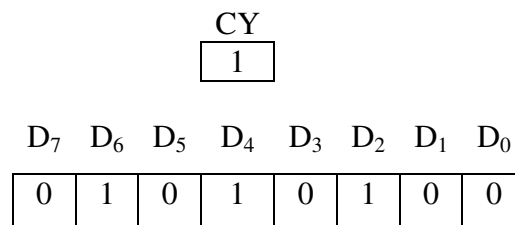
Chaque bit est décalé vers la gauche d'une seule position. Le bit D₇ deviendra CY et le CY est décalé vers D₀.

Exemple :

1. Avant RAL (A)=AAH et CY=0



2. Après RAL (A)=54H et Y=1

**3. RRC : Rotate Accumulator Right : Rotation à droite de A.**

Chaque bit est décalé vers la droite d'une seule position. Le bit D₀ deviendra D₇.

CY est modifié selon la valeur de D₀.

La rotation à droite de A engendre la division de la valeur de A sur 2 à condition que le bit D₀ soit à 0.

4. RAR : Rotate Accumulator Right Through Carry : Rotation à droite de A par CY.

Chaque bit est décalé vers la droite d'une seule position. Le bit D₀ deviendra CY et le CY est décalé vers D₇.

VII.5. Instructions de comparaison

Le 8085 possède deux types d'opérations de comparaison à savoir : CMP et CPI.

CMP : Compare with Accumulator : Comparaison avec l'accumulateur A.

CPI : Compare Immediate with Accumulator : Comparaison en immédiat avec l'accumulateur A.

Le microprocesseur compare un octet de donnée (ou contenus registre / mémoire) avec le contenu de l'accumulateur A par soustraction de la donnée de A et indique si la donnée est supérieur ou égale ou inférieur ou égale au contenu de l'accumulateur. Cependant les deux opérandes ne seront pas modifiés.

1. CMP R ou CMP M

R : registre 8 bits et M : Adresse d'une case mémoire spécifiée par HL

- ✓ C'est une instruction à un seul octet.
- ✓ Compare le contenu de registre/case mémoire avec le contenu de A.
- ✓ Si $(A) < (R/M)$, CY est mis à 1 et Z est mis à 0.
- ✓ Si $(A) = (R/M)$, Z est mis à 1 et CY est mis à 0.
- ✓ Si $(A) > (R/M)$, CY et Z sont mis à 0.
- ✓ Aucun contenu n'est modifié par l'instruction CMP, seulement les flags sont affectés par l'opération de soustraction.

2. CPI donnée 8 bits

- ✓ C'est une instruction à deux octets.
- ✓ Compare le deuxième octet (donnée de 8bits) avec le contenu de A.
- ✓ Si $(A) < \text{donnée 8 bits}$, CY est mis à 1 et Z est mis à 0.
- ✓ Si $(A) = \text{donnée 8bits}$, Z est mis à 1 et CY est mis à 0.
- ✓ Si $(A) > \text{donnée 8 bits}$, CY et Z sont mis à 0.
- ✓ Aucun contenu n'est modifié par l'instruction CPI, seulement les flags sont affectés par l'opération de soustraction.

VII.6. Instructions de branchement

Les instructions de branchement permettent au microprocesseur de changer l'exécution séquentielle du programme sous certaines conditions ou sans condition. Les instructions de branchement sont classées en trois catégories:

Les instructions de saut,

Les instructions CALL et RETURN,

Les instructions de redémarrage.

VII.6.1. Saut inconditionnel


Le jeu d'instructions du 8085 inclue une seule instruction de saut inconditionnel. Ce dernier permet au programmeur de réaliser des boucles.

Syntaxe : JMP Adresse 16 bits

C'est une instruction à trois octets

Exemple de boucle:

Adresse mémoire	Code machine	Etiquette	Mnémonique
2000	DB	START:	IN 00H
2001	00		
2002	D3		OUT 01H
2003	01		JMP START
2004	C3		
2005	00		
2006	20		


VII.6.2. Sauts conditionnels

Les instructions de saut conditionnel permettent au microprocesseur de faire une décision sous certaines conditions indiquées par les flags. Après des opérations logiques et arithmétiques les flags sont mis à 0 ou à 1. Les instructions de branchement inconditionnel vérifient les flags pour décider d'interrompre ou ne pas interrompre l'exécution séquentielle du programme.

Les quatre flags du registre d'état utilisés par les instructions de branchement sont : CF, Z, S et P. Deux instructions de branchement sont associées avec chaque flag. La séquence du programme peut être changée ou pas suivant la condition de test. Par exemple avec (JC) la retenue de l'addition existe et avec (JNC) la retenue est absente. Les instructions de branchement appartiennent au mode d'adressage immédiat.

Toutes les instructions de branchement possèdent trois octets ; le second octet spécifie l'octet faible de l'adresse et le troisième octet représente le poids fort de l'adresse mémoire de saut.

On résume les instructions de branchement en ce qui suit :

Opcode	Opérande	Description
JC	16 bits	Saut si retenue (si le résultat génère une retenue : CY=1)
JNC	16 bits	Saut si pas de retenue (Y=0)
JZ	16 bits	Saut si nul (si le résultat est nul : Z=1)
JNZ	16 bits	Saut si non nul (si le résultat est différent de 0 : Z=0)
JP	16 bits	Saut si plus grand (si D ₇ =0 et S=0)
JM	16 bits	Saut si plus petit (si D ₇ =1 et S=1)
JPE	16 bits	Saut si pair (P=1)
JPO	16 bits	Saut si impair (P=0)

Remarque

Il existe deux instructions de manipulation du flag CY à savoir :

1. STC : mise à 1 de CY

2. CMC : Complémenter CY

Exemple :

Le programme suivant affiche la valeur de l'accumulateur A qui est le résultat de la somme entre D et E si CY=0 sinon il affiche 01H (CY=0). Le déroulement du programme est le suivant :

Adresse mémoire	Code machine	Etiquette	Mnémonique
2000	16		MVI D,9BH
2001	9B		
2002	1E		MVI E,A7H
2003	A7		
2004	7A		MOV A,D
2005	83		ADD E
2006	D2		JNC AFFICHER
2007	0B		
2008	20		
2009	3E		MVI A,01H
200A	01		
200B	D3	→ AFFICHER:	OUT 01H
200C	00		
200D	76		HLT

VII.6.3. Boucles en assembleur

Les programmes illustrés auparavant peuvent être résolus manuellement. Cependant, des difficultés de traitement auront lieu dans le cas des tâches répétitives, comme par exemple l'addition de 100 nombres ou transfert de milliers d'octets. La programmation assembleur offre au programmeur la possibilité de répéter des séquences : cela est nommé bouclage. Une boucle permet au microprocesseur de changer la séquence d'exécution et répéter les tâches une nouvelle fois. Cela est accompli par l'utilisation des instructions de branchement. En plus peut être des compteurs sont ajoutés pour définir le nombre de répétition. Les boucles sont classées selon deux groupes : boucles infinies et boucles conditionnelles.

VII.6.3.1. Boucles infinies

Ce type de boucles est utilisé pour la répétition infinie. Son principe est illustré par la figure 10.a. Un programme avec une boucle infinie ne s'arrête que lorsque le système reçoit un RESET (Voir section architecture externe du 8085).

VII.6.3.2. Boucles conditionnelles

Une boucle conditionnelle est : utilisée pour la répétition avec certaines conditions. Elle est réalisée à base des instructions de branchement. Ces instructions vérifient l'état des flags (Zero, Carry,...) et répètent les tâches concernées si les conditions sont satisfaites. Ces boucles utilisent généralement la notion de comptage (figure 10.b).

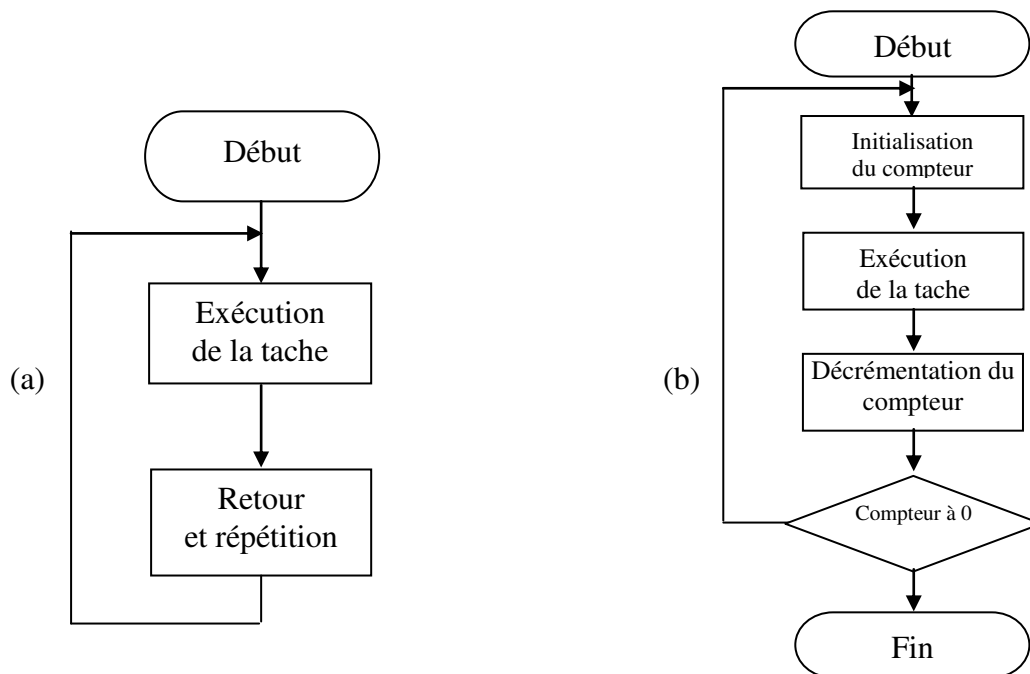


Figure V.10 : Principe des boucles

VII.7. Implémentation de la pile

Comme il a été mentionné au chapitre précédent, le pointeur de pile (SP) pointe vers le dessus de la pile (zone mémoire).

Une pile dans un système à microprocesseur peut être décrite comme un ensemble de cases mémoire en lecture et écriture. Cette zone mémoire est utilisée pour stocker les octets temporairement durant l'exécution d'un programme.

Le début de la pile est défini dans le programme par l'utilisation de l'instruction LXI SP, qui charge l'adresse mémoire de la pile dans le registre pointeur de pile (SP). En général SP pointe vers le top de la pile. Par exemple si SP est chargé par 2099H (LXI SP,20099H), le stockage des données commence à partir de 2098H et continue selon l'ordre 2097H, 2096H,... La taille de la pile dépend de la taille de la mémoire disponible.

Les contenus des registres pairs peuvent être stockés dans la pile (deux octets en même temps) par l'utilisation de l'instruction PUSH. Pour transférer les données depuis la pile vers les registres POP est utilisée. Deux octets sont transférés de/vers la pile, ce qui implique que lors d'un PUSH SP se décrémente de 2 et lors d'un POP SP s'incrémente de 2.

Instructions PUSH et POP**Syntaxe :** PUSH Rp et POP Rp

Rp : registre pair B, D ou H ou registre PSW (Program Status Word) qui est l'ensemble Accumulateur (A) et registre d'état (F).

Exemple 1:

LXI B,1234H

LXI SP,2000H

PUSH B

POP B

Exemple 2:

Les instructions PUSH et POP peuvent être utilisées pour échanger les contenus des registres pairs B et D. Le programme est le suivant :

PUSH B

PUSH D

POP B

POP D

La figure V.11 montre comment une valeur est stockée dans la pile et la figure V.12 montre comment elle est récupérée :

Pour le registre pair B=1234H et SP=2000H

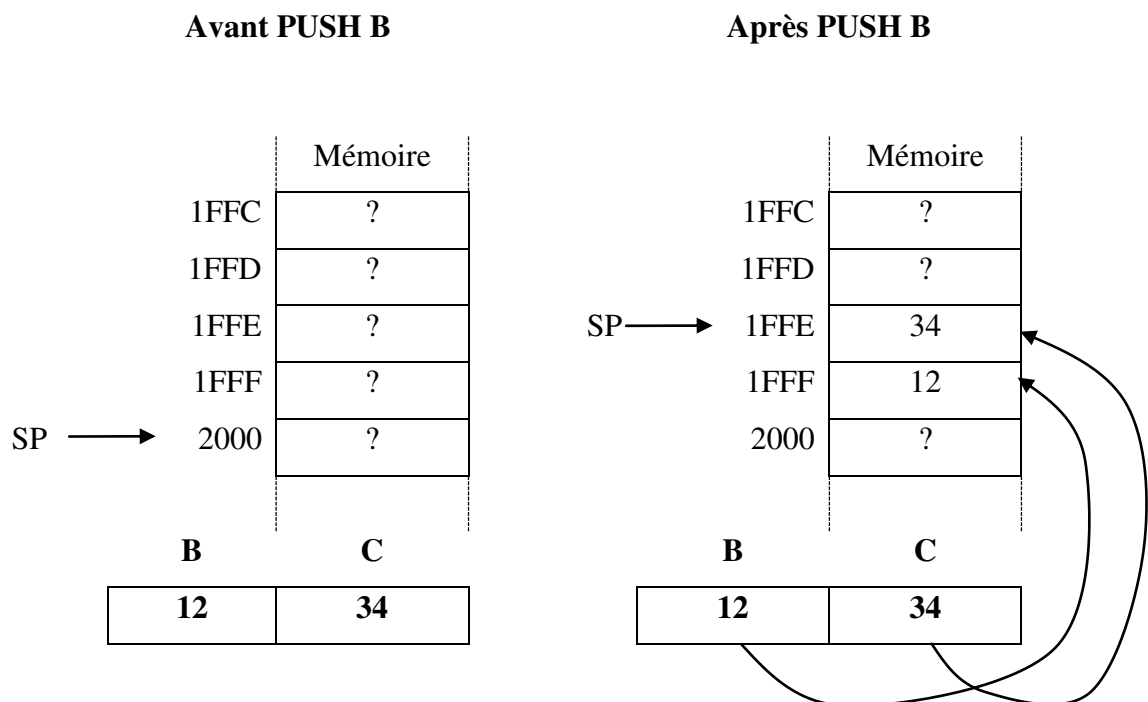


Figure V.11 : Exécution de PUSH B

Pour SP=2000H

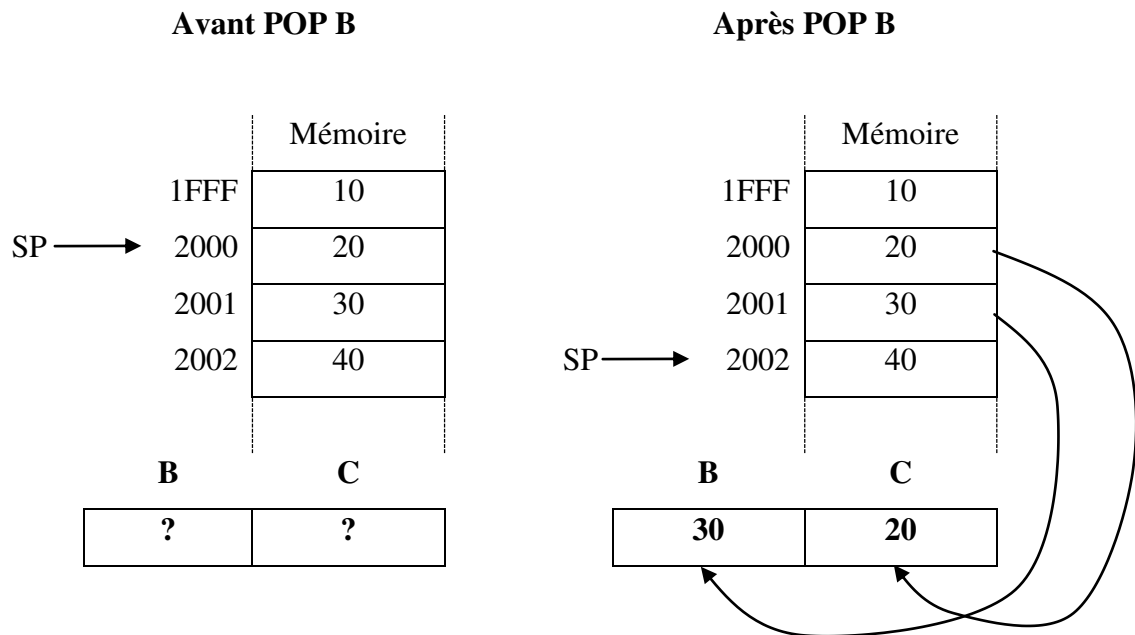


Figure V.12 : Exécution de POP B

VII.8. Sous programmes

Un sous programme est un groupe d'instructions écrites séparément du programme principal pour réaliser une tâche qui apparaît d'une manière répétitive dans le programme principal. Pour éviter la répétition des mêmes blocs d'instructions, un sous programme est écrit et appelé quand le programme principal a besoin.

L'appel du sous programme est réalisé par l'instruction **CALL**, le retour au programme principal est assuré par l'instruction **RET** (retour). CALL est utilisée dans le programme tandis que RET est utilisé à la fin du sous programme.

Quand un sous programme est appelé, le contenu du CP est mémorisé dans la pile. Le contenu du CP représente dans ce cas l'adresse de l'instruction qui va suivre CALL. Dès l'exécution de RET le 8085 récupère l'adresse de retour au programme de la pile.

1. Instruction CALL

Syntaxe : CALL Adresse 16 bits du sous programme

- ✓ C'est une instruction à 3 octets
- ✓ Sauvegarde le contenu du PC dans la pile
- ✓ Décrémente SP de 2
- ✓ Effectue un saut inconditionnel vers l'adresse du sous programme

2. Instruction RET

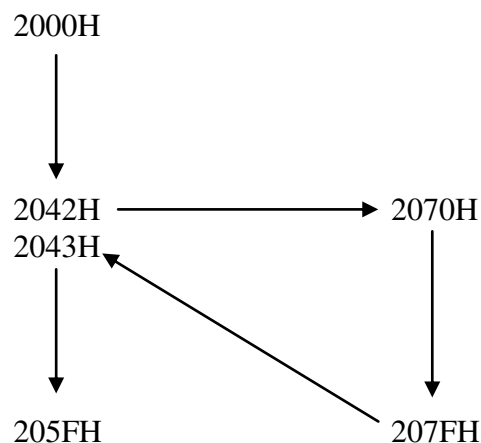
Syntaxe : RET

- ✓ C'est une instruction à un seul octet.
- ✓ Récupère la valeur de CP initial de la pile (adresse de retour au programme principal).
- ✓ Incrémente SP de 2.
- ✓ Retour incondtionnel au programme principal.

Exemple :

Adresse mémoire	Code machine	Commentaire
2000H ↓	LXI SP,2400H ↓	; Initialise SP avec 2400H
2040H 2041H 2042H 2043H ↓	CALL 2070H ↓ Instruction suivante ↓	; Appel le sous programme logé dans 2070H
205FH 2070H ↓	HLT Première instruction du sous programme ↓	; Fin du programme principal ; Début du sous programme
207FH	RET	; Fin du sous programme

Le déroulement de ce programme s'effectue de la manière suivante :



VII.9. Appels conditionnels et instructions de retour

Les appels conditionnels et les instructions de retour sont basés sur quatre conditions (flags) : CY, Z, S et P. Ces conditions sont testés par la vérification des flags. Dans le cas d'un appel conditionnel, l'exécution est transférée vers le sous programme si la condition est vérifiée, autrement le programme continue à s'exécuter. Dans le cas d'une instruction de retour conditionnel, la séquence retourne au programme principal si la condition est vérifiée, autrement la séquence du sous programme continue. L'appel conditionnel et le retour conditionnel sont résumés comme suit :

1. Appels conditionnels

CC	Appel sous programme si CY=1
CNC	Appel sous programme si CY=0
CZ	Appel sous programme si Z=1
CNZ	Appel sous programme si Z=0
CM	Appel sous programme si S=1
CP	Appel sous programme si S=0
CPE	Appel sous programme si P=1
CPO	Appel sous programme si P=0

2. Retours conditionnels

RC	Appel sous programme si CY=1
RNC	Appel sous programme si CY=0
RZ	Appel sous programme si Z=1
RNZ	Appel sous programme si Z=0
RM	Appel sous programme si S=1
RP	Appel sous programme si S=0
RPE	Appel sous programme si P=1
RPO	Appel sous programme si P=0

VIII. Cycle d'exécution d'une instruction [16]

Pour illustrer le cycle d'exécution d'une instruction à deux ou trois octets, on considère par exemple l'instruction MVI A,32H. L'opcode de cette instruction est 3EH et l'opérande est 32H. C'est une instruction de chargement de l'accumulateur A par la valeur 32H. On peut calculer le temps nécessaire pour l'exécution de cette instruction à partir de la récupération de l'opcode et l'opérande et le cycle complet de lecture mémoire si la fréquence du système est 2MHz.

Cette instruction est formée de deux octets ; le premier est l'opcode et le deuxième est l'opérande. Le 8085 a besoin de deux cycles machine pour la lecture mémoire. Cette instruction demande sept cycles d'horloge pour ces deux accès mémoire.

Pendant T_1 , le microprocesseur envoie 011 pour les signaux IO/\overline{M} et les signaux d'état S_1 et S_0 . Il transmet l'adresse mémoire 2000H du registre CP sur le bus d'adresses, 20H sur $A_{15}-A_8$ et 00H sur AD_7-AD_0 et incrémente CP à 2001H pour pointer au prochain octet. Le signal ALE est à l'état haut pendant T_1 . Ce dernier mémorise le poids faible du bus AD_7-AD_0 à la sortie des latches. Pendant T_2 , le 8085 active le signal \overline{RD} . Le cycle de recherche de l'opcode est achevé après T_3 . Durant T_4 , le 8085 décode l'opcode et identifie le type de l'instruction, dans

ce cas il a besoin de lire le prochain octet. Après T_3 le contenu du bus $A_{15}-A_8$ est inconnu et le bus de données AD_7-AD_0 est mis en haute impédance.

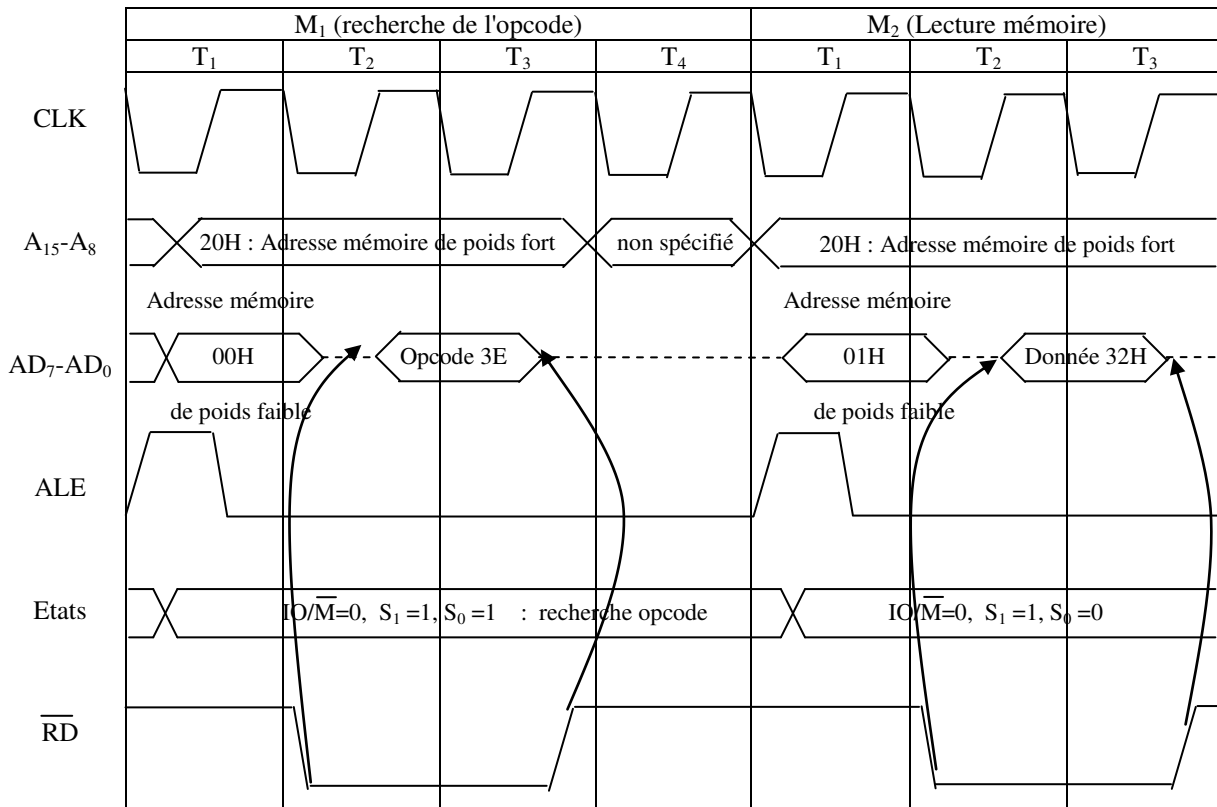


Figure V.13 : Cycle d'exécution d'une instruction

Le 8085 envoie l'adresse 2001H sur le bus d'adresses et incrémente le registre PC pour pointer vers la prochaine adresse 2002H. Le deuxième cycle machine M_2 est identifié comme une lecture mémoire ($IO/\overline{M}=0$, $S_1=1$ et $S_0=0$) et ALE est mis à 1. Le signal \overline{RD} devient actif.

Au front montant de T_2 , la mémoire met la donnée 32H sur le bus de données et le 8085 lit et stocke l'octet dans l'accumulateur durant T_3 .

Le temps du cycle machine pour cet exemple est calculé comme suit :

- Fréquence d'horloge : $f=2$ MHz
- Un cycle = période d'horloge ($1/f$)=0.5 μ s
- Temps de recherche de l'opcode : $(4.T) \times 0.5 = 2$ μ s
- Temps de lecture mémoire : $(3.T) \times 0.5 = 1.5$ μ s
- Temps d'exécution de l'instruction : $(7.T) \times 0.5 = 3.5$ μ s

IX. Génération des délais [16]

Un délai est réalisé par un chargement d'un registre avec une valeur et la décrémentation de ce registre par utilisation d'une boucle jusqu'à la valeur 0 du registre. Le délai est déterminé par la période de l'horloge du système et le temps nécessaire pour l'exécution de chaque

instruction dans la boucle. Les délais sont utilisés dans diverses applications à savoir : gestion de la circulation routière, transferts séries,...

IX.1. Génération d'un délai utilisant un registre 8 bits

Un registre 8 bits est utilisé pour supporter le nombre de répétitions de la boucle. Donc son maximum est (FFH). Ce registre sera décrémenter par l'instruction DCR. Cette dernière affecte le flag Z. Un exemple de programme délai est donné comme suit :

Etiquette	Opcode	Opérande	Commentaires	Nombre de cycles
	MVI	C,FFH	Charger C avec un comptage 8 bits	7
BOUCLE:	DCR	C	Décrémenter (C) par 1	4
	JNZ	BOUCLE	Si résultat $\neq 0$, saut arrière à BOUCLE	10/7

Si TL désigne le délai de la boucle, T est la période de l'horloge du système, N10 est l'équivalent du nombre en décimal du registre de comptage, on aura :

$$TL = T \times \text{Nombre de cycle de la boucle} \times N10$$

Si la fréquence de l'horloge est 2MHz, alors $T = 0.5\mu s$

$$TL = 0.5 \times 10^{-6} \times 14 \times 254 + 0.5 \times 10^{-6} \times 1$$

$$TL = 1783\mu s \approx 1.8ms$$

IX.2. Génération d'un délai utilisant un registre pair

Un registre pair est utilisé pour définir le nombre de répétitions de la boucle du délai (maximum FFFFH). Le registre est décrémenté par l'utilisation de l'instruction DCX. Cependant l'instruction DCX n'affecte pas le flag Z. Des instructions supplémentaires doivent être ajoutées. Le programme suivant traite ce cas :

Etiquette	Opcode	Opérande	Commentaires	Nombre de cycles
	LXI	B,FFFFH	Charger BC avec un comptage 16 bits	10
BOUCLE:	DCX	B	Décrémenter (BC) par 1	6
	MOV	A,C	Placer le contenu de C dans A	4
	ORA	B	OR (B) avec (C) pour affecter le flag Z	4
	JNZ	BOUCLE	Si résultat $\neq 0$, saut arrière à BOUCLE	10/7

Le temps du délai est calculé comme suit : DCX, MOV, ORA et JNZ nécessitent 24 cycles pour l'exécution. La boucle est répétée FFFFH fois, c'est à dire 65535 fois en décimal.

Si le système fonctionne avec une fréquence de 2MHz, la période d'un cycle d'horloge est 0.5us, donc le délai TL de la boucle vaut :

$$TL = 0.5 \times 10^{-6} \times 24 \times 65534 + 0.5 \times 10^{-6} \times 21 \approx 786ms$$

Le délai total TD=TL si on néglige le temps d'exécution de l'instruction LXI.

IX.3. Génération d'un délai utilisant deux boucles

Deux boucles peuvent être utilisées (une à l'intérieur de l'autre). Par exemple le registre C est utilisé dans la boucle interne (BOUCLE1) et le registre B est utilisé dans la boucle externe (BOUCLE2). Les instructions suivantes sont donc utilisées :

	MVI B,FFH	7T
BOUCLE2:	MVI C,FFH	7T
BOUCLE1:	DCR C	4T
	JNZ BOUCLE1	10/7T
	DCR B	4T
	JNZ BOUCLE2	10/7T

$$TL1 = 1783us$$

$$TL2 = 255 \times (TL1 + 21 \text{ cycles} \times 0.5us) - (10 - 7) \times 0.5us$$

$$TL2 = 457ms$$

X. Exemple d'un programme assembleur d'un générateur du signal carré

On veut écrire un programme assembleur qui permet de générer un signal carré de période 500us. On suppose que la période de l'horloge du système est 325ns. Le signal de sortie est récupéré du bit D₀ de l'accumulateur A.

L'impulsion doit apparaître pendant 250us (1 logique) et disparaître pendant 250us (0 logique). Donc on doit charger l'accumulateur par la valeur AAH (1010 1010 en binaire) et on doit effectuer la rotation à chaque itération de boucle. Le délai de 250us peut être obtenu facilement avec un registre 8 bits. L'instruction ANI est utilisée pour le masquage des bits à 0.

*Calcul du délai COMPTAGE [16]

Le délai total doit inclure le délai de la boucle DELAI (TL) et le temps d'exécution des instructions hors la boucle DELAI (T0).

1. Le nombre des instructions hors la boucle DELAI sont sept ; six instructions à partir de l'étiquette ROTATION et l'instruction JMP ROTATION.

$$T0 = 46 \text{ cycles} \times 325 \text{ ns} = 14.95us$$

2. Le délai de la boucle DELAI comprend deux instructions (DCR et JNZ) avec 14 cycles à l'exception du dernier cycle qui possède 11 cycles.

$$TL = 14 \text{ cycles} \times 325\text{ns} \times (\text{COMPTAGE}-1) + 11 \text{ cycles} \times 325\text{ns}$$

$$TL = 4.5\mu\text{s} \times (\text{COMPTAGE}-1) + 3.575\mu\text{s}$$

3. Le délai total demandé TD est 250 μs . Cependant COMPTAGE peut être calculé comme suit : $TD = T_0 + TL$

$$250\mu\text{s} = 14.95\mu\text{s} + 4.5\mu\text{s} (\text{COMPTAGE}-1) + 3.575\mu\text{s}$$

$$\text{COMPTAGE} = 52.4_{10} = 34\text{H}$$

Adresse mémoire	Code HEX	Etiquette	Mnémonique	Commentaires
XX00	16		MVI D,AAH (7T)	;Charger la forme AAH
01	AA			
02	7A	ROTATION:	MOV A,D (4T)	;Charger la forme dans A
03	07		RLC (4T)	;Changer la forme AAH à 55H et ;vice versa
04	57		MOV D,A (4T)	;Enregistrer (A)
05	E6		ANI 01H (7T)	;Masquer les bits D ₇ à D ₁
06	01			
07	D3		OUT PORT1 (10T)	;Sortie à 1 ou à 0
08	PORT1			
09	06		MVI B,COMPTAGE (7T)	;Charger le délai pour 250 μs
0A	COMPTAGE			
0B	05	DELA:	DCR B (4T)	;Prochain comptage
0C	C2		JNZ DELAI (10/7T)	;Répéter jusqu'à avoir (B)=0
0D	0B			
0E	XX			
0F	C3		JMP ROTATION (10T)	;Retourner pour changer le niveau logique
10	02			
11	XX			

XI. Lignes de transmission série du microprocesseur 8085 SOD et SID [16]

Le 8085 possède deux broches pour la transmission série programmables. : SOD (Serial Output Data) et SID (Serial Input Data). Le transfert est commandé par les instructions SIM et RIM. Ces instructions sont utilisées dans le processus d'interruption ou le transfert sériel.

XI.1. Ligne de transmission SOD

Cette ligne est interprétée pour la transmission série comme suit :

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
SOD	SDE	X	Pour les interruptions				

SDE=0 : SOD désactivé

SDE=1 : SOD activé

Instructions :

MVI A,80H ; Mise à 1 de D₇ dans l'accumulateur

RAR ; D₆=1 et mettre CY dans D₇

SIM ; Sortir D₇ par la ligne SOD

Dans cette suite d'instructions, La transmission série est active et l'instruction SIM envoie le bit CY par la position D₇.

Exemple :

Dans cet exemple un sous programme SODATA doit transmettre un caractère, stocké dans le registre B, en série sur la ligne SOD. Un compteur est réglé sur 11 bits : Bit Start + Donnée sur 8 bits + 2 bits Stop. La durée d'un bit transmis est environ 9.1ms.

```

SODATA:  MVI C,0BH      ; Le contenu de C est 11
          XRA A          ; Mise à 0 de CY
NEXTBIT: MVI A,80H      ; D7 de l'accumulateur à 1
          RAR            ; Placer CY dans D7 et mise à 1 de D6
          SIM            ; Sortir D7
          CALL BITTIME   ; Attente de 9.1 ms
          STC            ; CY=1
          MOV A,B        ; Placer le code du caractère dans A
          RAR            ; Placer D0 du caractère dans CY, décalage de 1 dans D7
                          ; et continuer le décalage pour chaque itération de boucle
          MOV B,A        ; stocker A dans B
          DCR C          ; Un bit transmis, décrémenter le compteur
          JNZ NEXTBIT    ; Si tous les bits ne sont pas transmis, retour
          RET

```

XI.2. Ligne de réception SID

L'instruction RIM est utilisée pour la réception des données en série. Cette instruction est interprétée par ce qui suit :

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
SID Pour les interruptions							

Remarque :

L'instruction RIM est similaire à l'instruction IN, à l'exception que RIM lit un seul bit et le place dans l'accumulateur à D₇. Aussi SIM est équivalente à OUT, mais SIM envoie un seul bit. Les lignes SID et SOD du 8085 éliminent le besoin d'un port d'entrée/sortie.

Pour recevoir un caractère, une procédure consiste à sauvegarder les bits provenant sur la ligne SID.

Après la lecture du premier bit D₀, ce bit est stocké dans CY avec l'instruction RAL. Le bit est stocké et décalé à droite par l'utilisation de RAR. Cette procédure est répétée huit fois. A la neuvième itération, le sous programme retourne au programme principal en ignorant les deux bits Stop.

CHAPITRE VI

LES INTERFACES D'ENTREES SORTIES

I. Généralités [5,6]

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**, issu de 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...).

Exemple :

- taper sur les touches du clavier envoie une série de codes vers le processeur ; ces codes sont considérés comme des données d'entrée ;
- le microprocesseur affiche les résultats de traitement des données sur un écran ; ce sont des données de sortie. Habituellement, l'écran est géré par un programme de gestion d'affichage.

Les objectifs de ce chapitre sont : Enumération des éléments du 8255A "Programmable Peripheral Interface" (PPI) et explication de ces modes de fonctionnement. Aussi, ce chapitre donne une description détaillée de l'interface série (8250) et du Timer (8254) et leurs différentes configurations.

II. Microprocesseur et circuits d'interfaces [5,6]

Les interfaces d'entrée sortie sont connectées au microprocesseur à travers les bus d'adresses, de données et de commandes comme le montre la figure VI.1. Les points d'accès aux interfaces sont appelés **ports**.

Le 8085 dispose d'un espace mémoire de 64 Ko (adresse d'une case mémoire sur 16 bits) et d'un espace d'E/S de 256 octets.

Le signal permettant de différencier l'adressage de la mémoire de l'adressage des ports d'E/S est la ligne $\overline{IO/\overline{M}}$:

- Pour un accès à la mémoire, $\overline{IO/\overline{M}} = 0$;
- Pour un accès aux ports d'E/S, $\overline{IO/\overline{M}} = 1$.

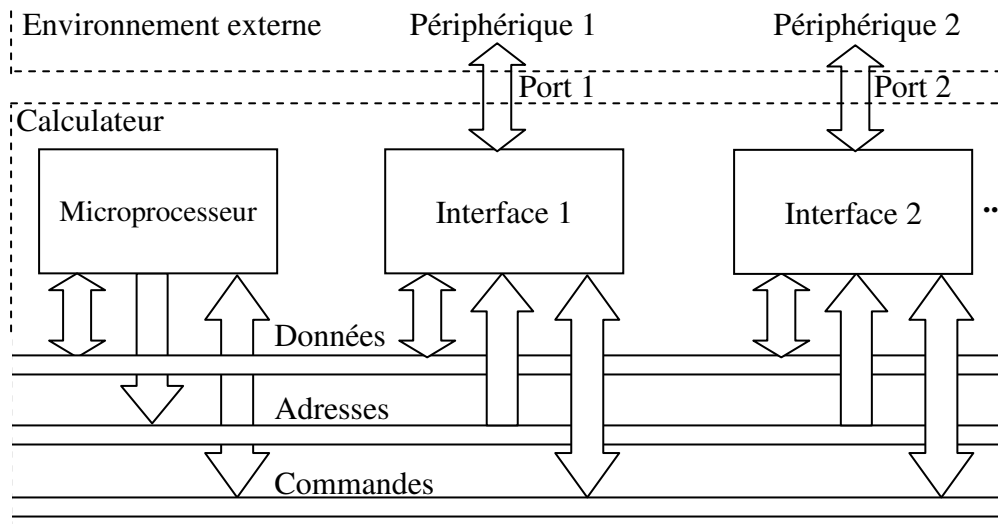


Figure VI.1 : Microprocesseur et circuits interfaces

III. Différents types d'interfaces [16]

Il existe plusieurs types d'interfaces d'entrée sortie de la famille INTEL à savoir: l'interface périphérique programmable 8255A, l'interface série 8250, le contrôleur d'interruptions 8259A, le Timer 8254 et le DMA 8237.

III.1. Interface périphérique programmable 8255A (Programmable Peripheral Interface : PPI)

Le rôle d'une interface 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.

III.1.1. Brochage du 8255A

Le 8255A contient 24 broches d'entrée / sorties groupés dans trois ports de 8 bits A, B et C. Les huit bits du port C peuvent être utilisés individuellement ou groupés en deux sous groupes de 4 bits : C_{Haut} et C_{Bas} . Le fonctionnement de ces registres est identifié par les bits du registre de contrôle.

Un registre de données assure la liaison entre le bus de données extérieur et le registre de contrôle ainsi que les ports d'entrées/sorties.

La logique de commande est composée de six lignes :

- \overline{RD} : Ce signal permet d'effectuer une opération de lecture. Quand ce signal est à 0, le microprocesseur lit la donnée du port sélectionné du 8255A.
- \overline{WR} : Ce signal permet d'effectuer une opération d'écriture. Quand ce signal est à 0, le microprocesseur écrit la donnée dans le port sélectionné du 8255A.
- **Reset** : : C'est un signal actif à l'état haut, il permet à la fois d'effacer le registre de contrôle et mettre les ports en mode entrée.
- \overline{CS} , A_1 , A_0 : Ce sont les signaux de sélection. \overline{CS} à l'état bas sélectionne le circuit 8255A et A_1 , A_0 spécifient le port sélectionné ou le registre de contrôle comme le montre le tableau suivant :

\overline{CS}	A_1	A_0	Registre
0	0	0	Port A
0	0	1	Port B
0	1	0	Port C
0	1	1	Registre de contrôle
1	X	X	8255A non sélectionné

Tableau VI.1 : Adressage des ports et registre de contrôle

Exemple :

Par exemple, les adresses des ports dans la figure sont calculées à partir des lignes \overline{CS} , A_1 et A_0 . La ligne \overline{CS} est à l'état haut quand $A_7=1$ et A_6 jusqu'à A_2 sont à l'état bas. Quand ces signaux seront combinés avec A_1 et A_0 , les adresses des ports seront entre 80H et 83H.

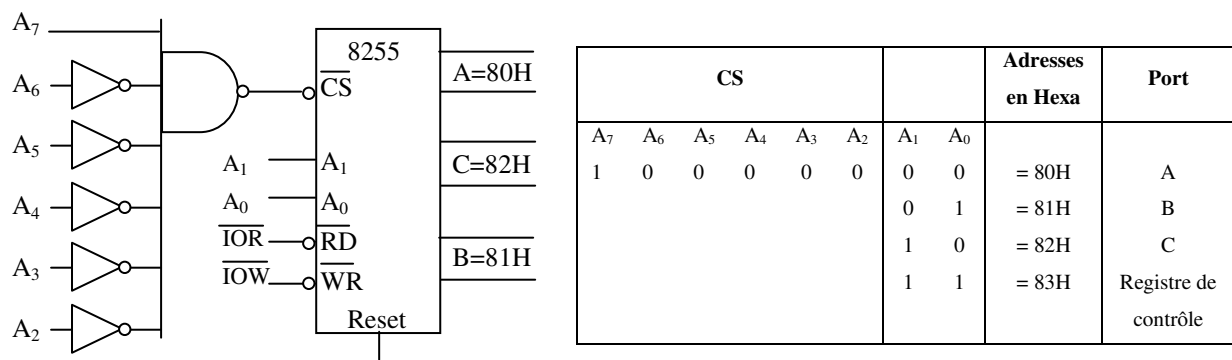


Figure VI.2 : Exemple d'adressage du 8255A

La figure précédente montre un registre appelé registre de contrôle. Les contenus de ce registre est appelé mot de contrôle qui spécifie la fonction d'entrée sortie de chaque port. Ce registre n'est pas accessible à une opération de lecture.

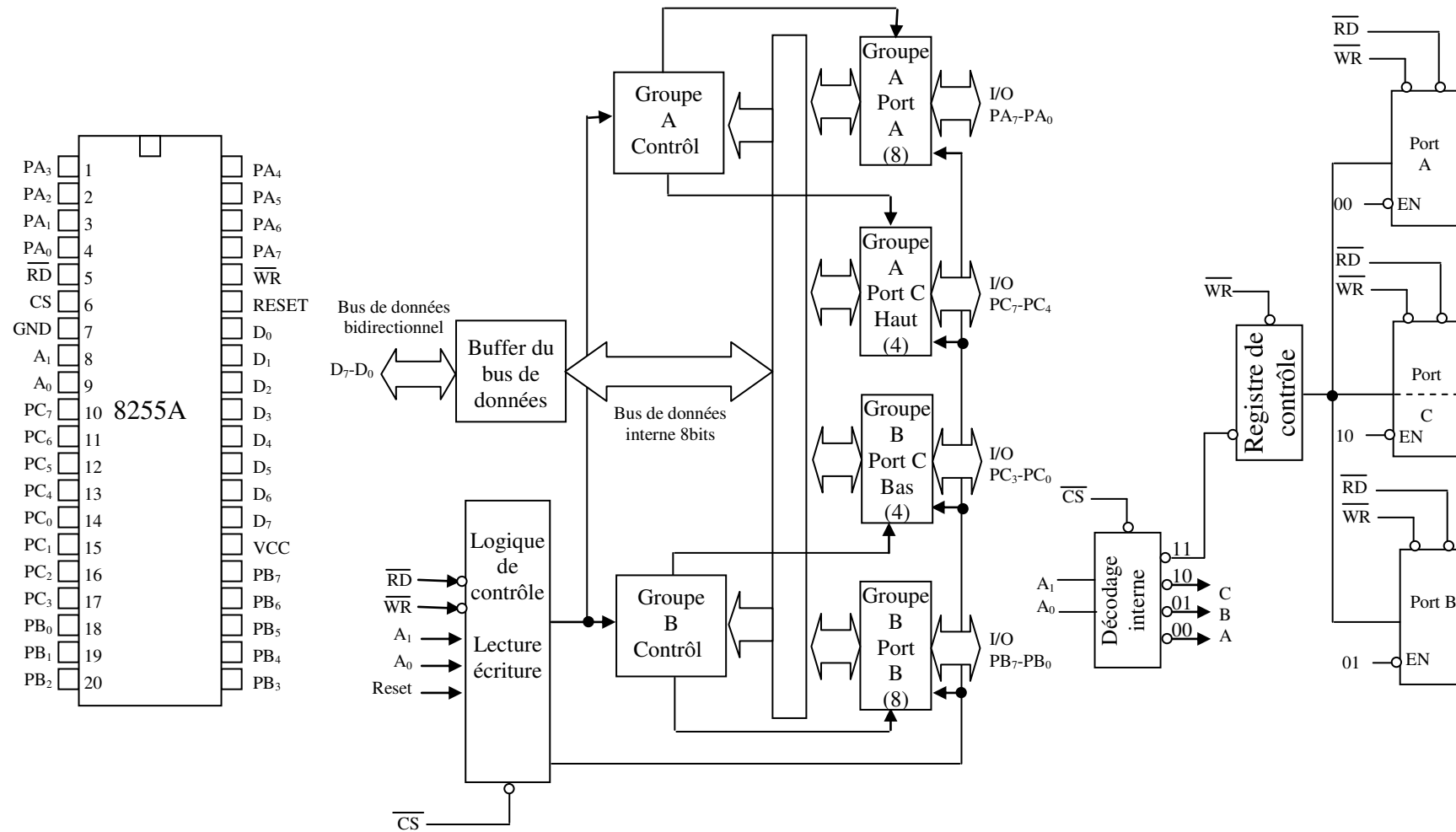


Figure VI.3 : Structure interne et externe du 8255A

III.1.2. Programmation du 8255A

On peut programmer le 8255A selon trois modes :

Mode 0 : Entrée / sortie de base.

Mode 1 : Entrée / sortie échantillonnée.

Mode 2 : Bus bidirectionnel.

Le format ainsi que le choix des modes se fait à partir du mot de contrôle suivant :

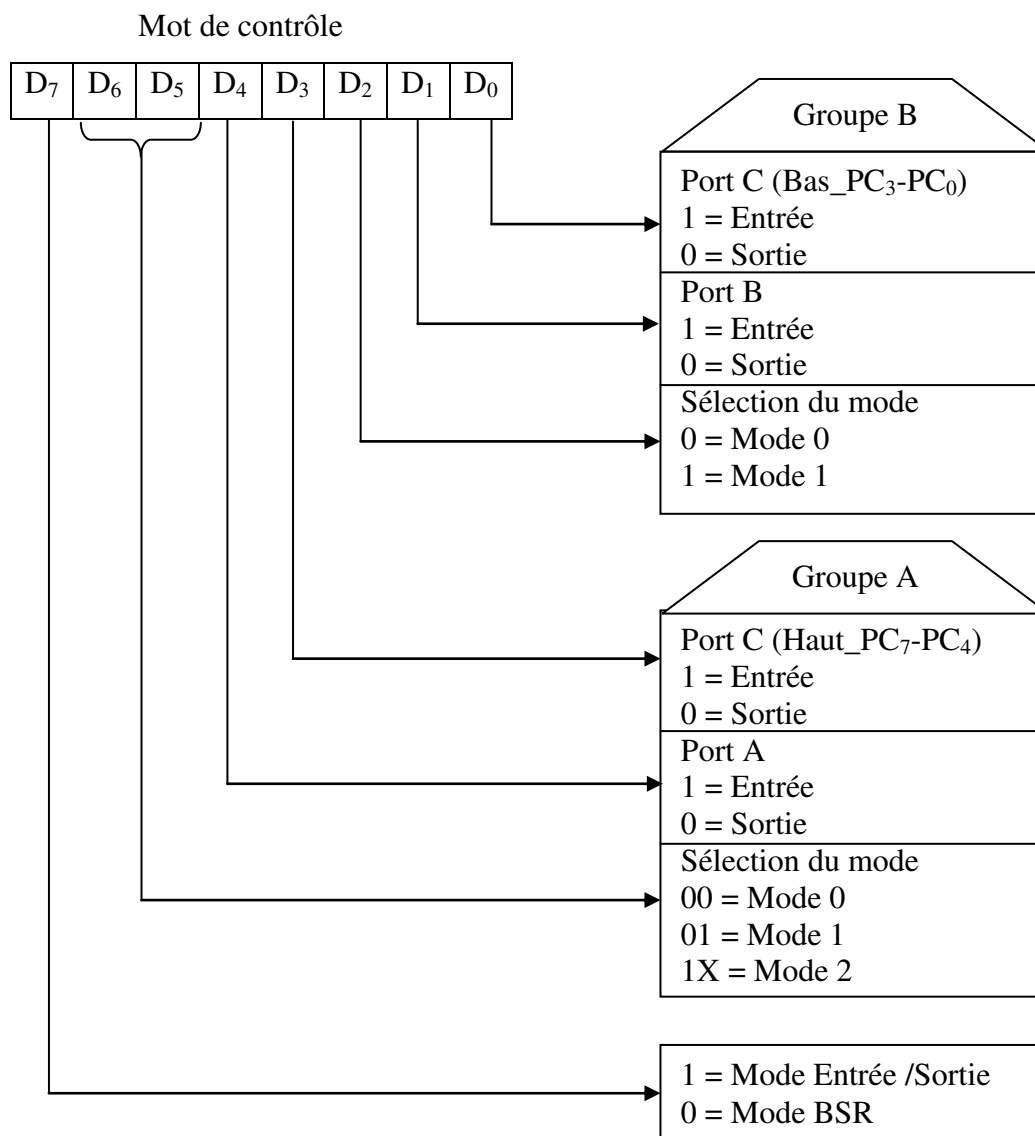


Figure VI.4 : Format du mot de contrôle du 8255A pour le mode d'entrée/sortie

Le bit D₇ du registre de contrôle spécifie à la fois la fonction d'E/S ou Bit Set/Reset. Si le bit D₇=1, les bits D₆-D₀ déterminent les fonctions d'E/S avec différents modes. Si le bit D₇=0, le registre C fonctionne en mode Bit Set/Reset.

Pour communiquer avec les périphériques à travers le 8255A, trois étapes sont à considérer :

1. Détermination des adresses des ports A, B, C et le contenu du registre de contrôle en tenant compte du circuit de sélection et des lignes d'adresses.
2. Ecrire le mot de contrôle dans le registre de contrôle
3. Ecrire les instructions d'entrée / sortie pour communiquer avec les périphériques à travers le port A, B et C.

* Mode BSR

Ce mode concerne seulement les huit bits du port C, qui peuvent être mis à 1 ou à 0 par configuration du registre de contrôle. Un mot de contrôle avec le bit $D_7=0$ décrit le mode BSR. Les opérations d'E/S des ports A et B ne sont pas affectés par le mode BSR. Pour la mise à 1 ou à 0 dans le port C, un mot de contrôle est écrit dans le registre de contrôle et non pas dans le port C. Le mot de contrôle est spécifié par ce qui suit :

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	
0	X	X	X	Sélection des bits			S/R	
↓ Mode BSR	↓ Non utilisés						↘	Set=1 Reset=0
				0	0	0		= Bit 0
				0	0	1		= Bit 1
				0	1	0		= Bit 2
				0	1	1		= Bit 3
				1	0	0		= Bit 4
				1	0	1		= Bit 5
				1	1	0		= Bit 6
				1	1	1		= Bit 7

Tableau VI.2 : Registre de contrôle

Exemple :

Ecrire un sous programme du mode BSR pour la mise à 1 des bits PC₇ et PC₃ et ensuite leurs remise à 0 après 10ms. On suppose que le délai 10ms est disponible.

Pour la mise à 1 du bit PC₇, le mot de contrôle = 0 0 0 0 1 1 1 1 = 0FH

Pour la mise à 0 du bit PC₇, le mot de contrôle = 0 0 0 0 1 1 1 0 = 0EH

Pour la mise à 1 du bit PC₃, le mot de contrôle = 0 0 0 0 0 1 1 1 = 07H

Pour la mise à 0 du bit PC₃, le mot de contrôle = 0 0 0 0 0 1 1 0 = 06H

L'adresse du registre de contrôle = 83 H (figure VI.2)

Sous programme :

```
BSR:  MVI A,0FH          ; Charger l'octet 0FH dans l'accumulateur
      OUT 83H            ; PC7=1
      MVI A,07H          ; Charger l'octet 07H dans l'accumulateur
      OUT 83H            ; PC3=1
      CALL DELAY         ; Délai de 10ms
```

```
MVI A,06H      ; Charger l'octet 06H dans l'accumulateur
OUT 83H        ; PC3=0
MVI A,0EH      ; Charger l'octet 0EH dans l'accumulateur
OUT 83H        ; PC7=0
RET
```

Mode 0 : Entrée ou sortie simple

Dans ce mode, les ports A et B sont utilisés comme deux ports d'entrée/sortie simples de 8 bits et le port C en tant que deux ports de quatre bits. Chaque port (ou demi port dans le cas de C) peut être programmé en entrée ou en sortie. Les caractéristiques des entrées sorties sont comme suit :

1. Les sorties sont mémorisés,
2. Les entrées ne sont pas mémorisées,
3. Les ports ne tiennent pas compte des handshake (poignés de main) et des interruptions.

Exemple :

1. Identifier les adresses du port de la figure suivante :

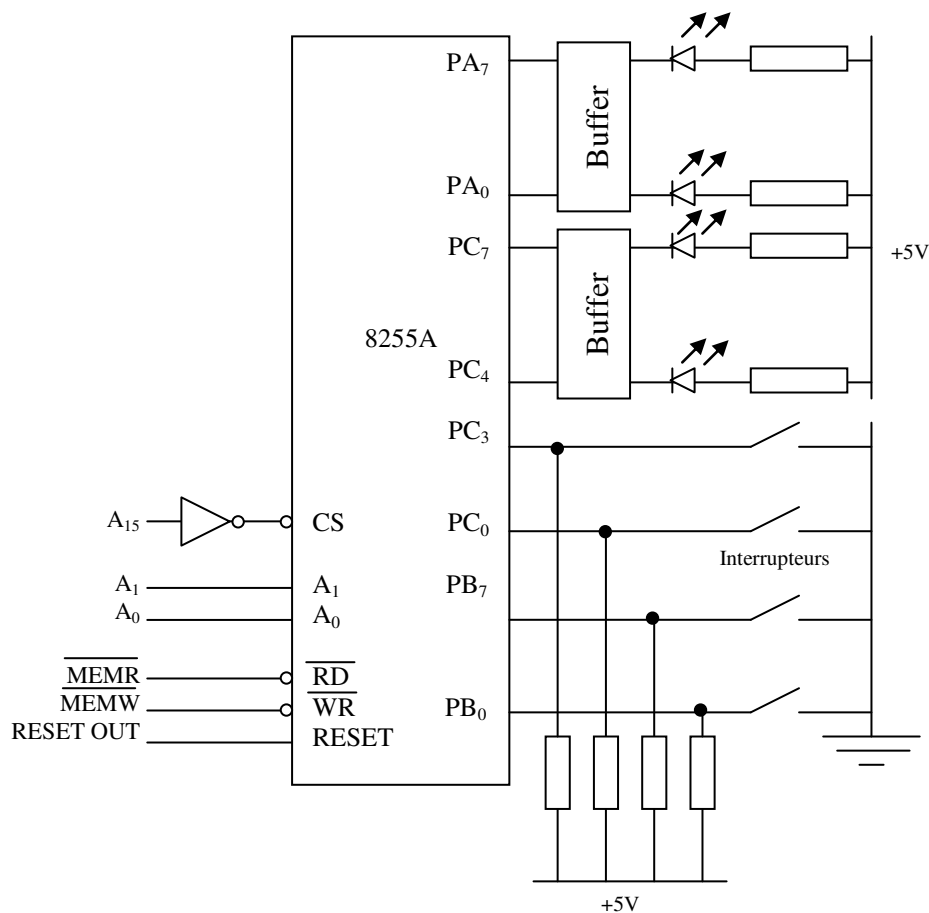


Figure VI.5 : Exemple d'application du 8255A en mode 0 en entrée et en sortie

2. Identifier le mot de contrôle du mode 0 pour configurer le port A et le port C_{Haut} en tant que ports de sortie et le port B et le port C_{Bas} en tant que ports d'entrée.
3. Ecrire le programme pour lire les états des interrupteurs et afficher la lecture du port B au port A et la lecture du port C_{Haut} au port C_{Bas}.

Solution :

1. Adresses des ports et registre de contrôle :

Port A : 8000H (A₁=0 et A₀=0)

Port B : 8001H (A₁=0 et A₀=1)

Port C : 8002H (A₁=1 et A₀=0)

Registre de contrôle : 8003H (A₁=1 et A₀=1)

2. Mot de contrôle :

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	
1	0	0	0	0	0	1	1	=83H

3. Programme :

MVI A,83H ; Charger l'accumulateur avec le mot de contrôle

STA 8003H ; Ecrire le mot dans le registre de contrôle pour initialiser les ports

LDA 8001H ; Lire les interrupteurs du port B

STA 8000H ; Afficher la lecture au port A

LDA 8002H ; Lire les interrupteurs du port C

ANI 0FH ; Masquer les quatre bits de poids fort du port C ; Ces bits ne sont pas des entrées

RLC ; Faire la rotation et placer la donnée au poids fort de l'accumulateur

RLC

RLC

RLC

STA 8002H ; Afficher la lecture au port C de poids fort

HLT

Mode 1 : Entrée ou sortie avec Handshake

Les caractéristiques de ce mode sont :

1. Deux ports A et B peuvent être configurés comme entrées ou sorties.
2. Chaque port utilise trois lignes du port C comme signaux de Handshake. Les deux lignes restantes du port C peuvent être utilisées comme des entrées ou sorties simples.
3. Les données d'entrée ou sortie sont mémorisées.
4. Une logique d'interruption peut être utilisée.

Mode 1 : en entrée

En entrée les ports A et B sont configurés en entrée alors que PC_2 , PC_1 et PC_0 assurent le handshake pour B et PC_3 , PC_4 et PC_5 pour A (PC_6 et PC_7 peuvent être programmés en entrée ou en sortie).

\overline{STB} : (Strobe input) : Généré par un périphérique (actif niveau bas), indique qu'il a transmis un octet de donnée. Le 8255A répond au \overline{STB} et génère IBF et INTR. \overline{STB}_A et connectée à la broche PC_4 et \overline{STB}_B et connectée à la broche PC_2 .

IBF : (Input Buffer Full) : Ce signal est une réponse de la part du 8255A, il indique que le latch d'entrée a reçu la donnée. Ce signal sera remis à zéro quand le microprocesseur lit la donnée.

INTR (Interrupt Request) : C'est un signal de sortie qui peut être utilisé pour interrompre le microprocesseur. Ce signal est généré si \overline{STB} , IBF et INTE (Internal flip-flop) sont tous à 1. Ces signaux seront remis à 1 au front montant du signal \overline{RD} émis par le microprocesseur.

INTE (Interrupt Enable) : C'est une bascule interne utilisée pour activer ou désactiver la génération de INTR. $INTE_A$ et $INTE_B$ sont commandés par le mot de contrôle du port C (PC_4 pour le Port A et PC_2 pour le Port B). Le mot d'état peut être lu dans l'accumulateur si le port C est lu par le microprocesseur.

Le handshake en entrée est résumé par la figure suivante :

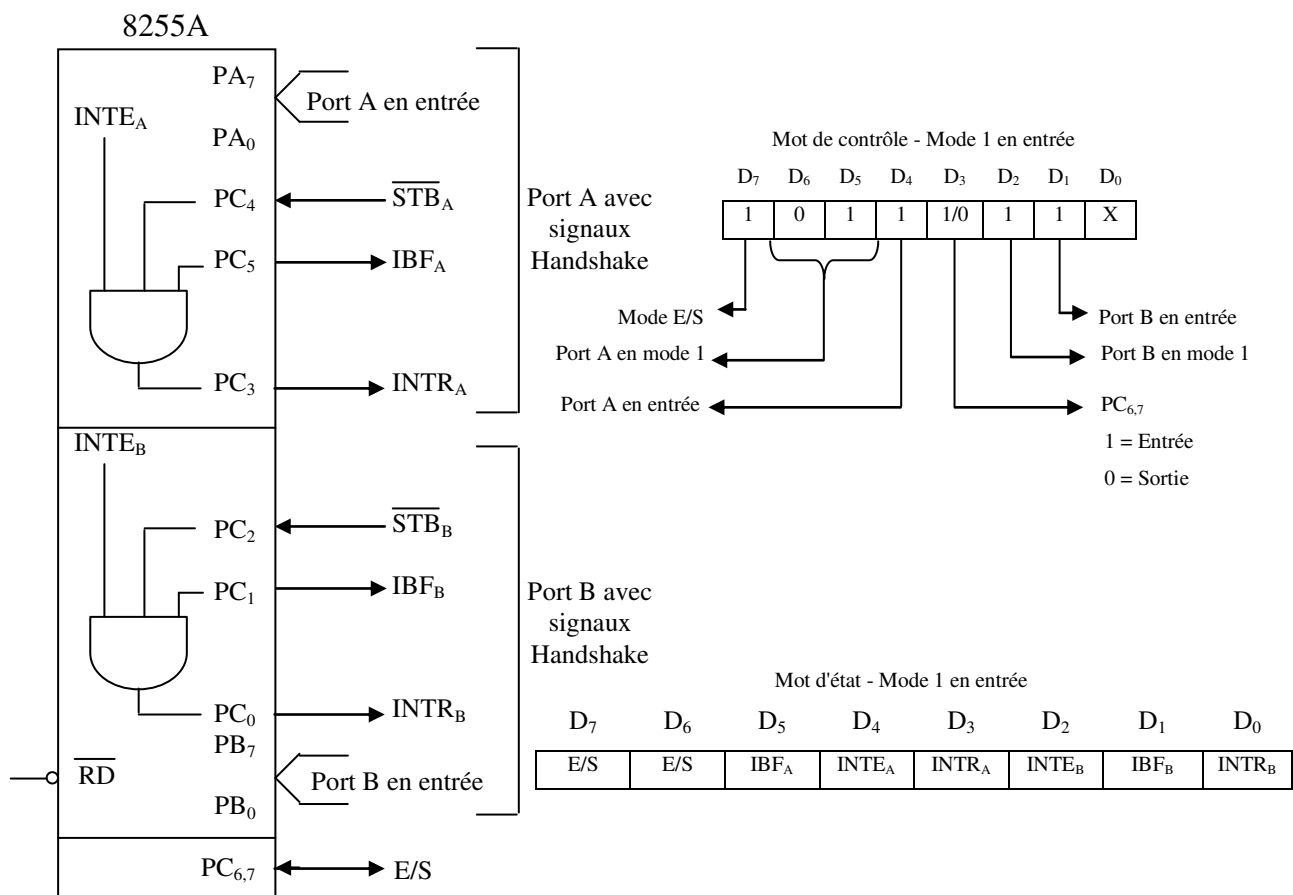


Figure VI.6 : 8255A en mode 1 en entrée

Mode 1 : en sortie

La figure VI.7 montre les signaux de contrôle quand les ports A et B sont configurés comme ports de sortie. Ces signaux sont définis comme suit :

- **OBF (Output Buffer Full) :** C'est un signal de sortie. Mis à l'état bas quand le microprocesseur écrit la donnée dans le latch de sortie du 8255A. Ce signal indique à un périphérique de sortie qu'une nouvelle donnée est prête pour la lecture. Il sera remis à l'état haut après la réception de $\overline{\text{ACK}}$ par le 8255A de la part d'un périphérique.
- $\overline{\text{ACK}}$ (Acknowledge) : C'est un signal d'entrée qui provient d'un périphérique qui doit envoyer un 0 quand il reçoit une donnée des ports du 8255A.
- **INTR (Interrupt Request) :** C'est un signal de sortie. Il est mis à 1 par un front montant du signal $\overline{\text{ACK}}$. Ce signal peut être utilisé pour interrompre le microprocesseur pour demander la prochaine donnée. Le signal INTR est mis à 1 quand $\overline{\text{OBF}}$, $\overline{\text{ACK}}$ et INTE sont tous à 1. INTR est remis à 0 au front descendant de $\overline{\text{WR}}$.
- **INTE (Interrupt Enable) :** C'est une bascule interne du port, elle a besoin d'être à 1 pour générer le signal INTR. INTE_A et INTE_B sont commandés par les bits PC_6 et PC_2 respectivement à travers le mode BSR.
- **$\text{PC}_{4,5}$:** Ces deux lignes peuvent être configurées en entrée ou sortie.

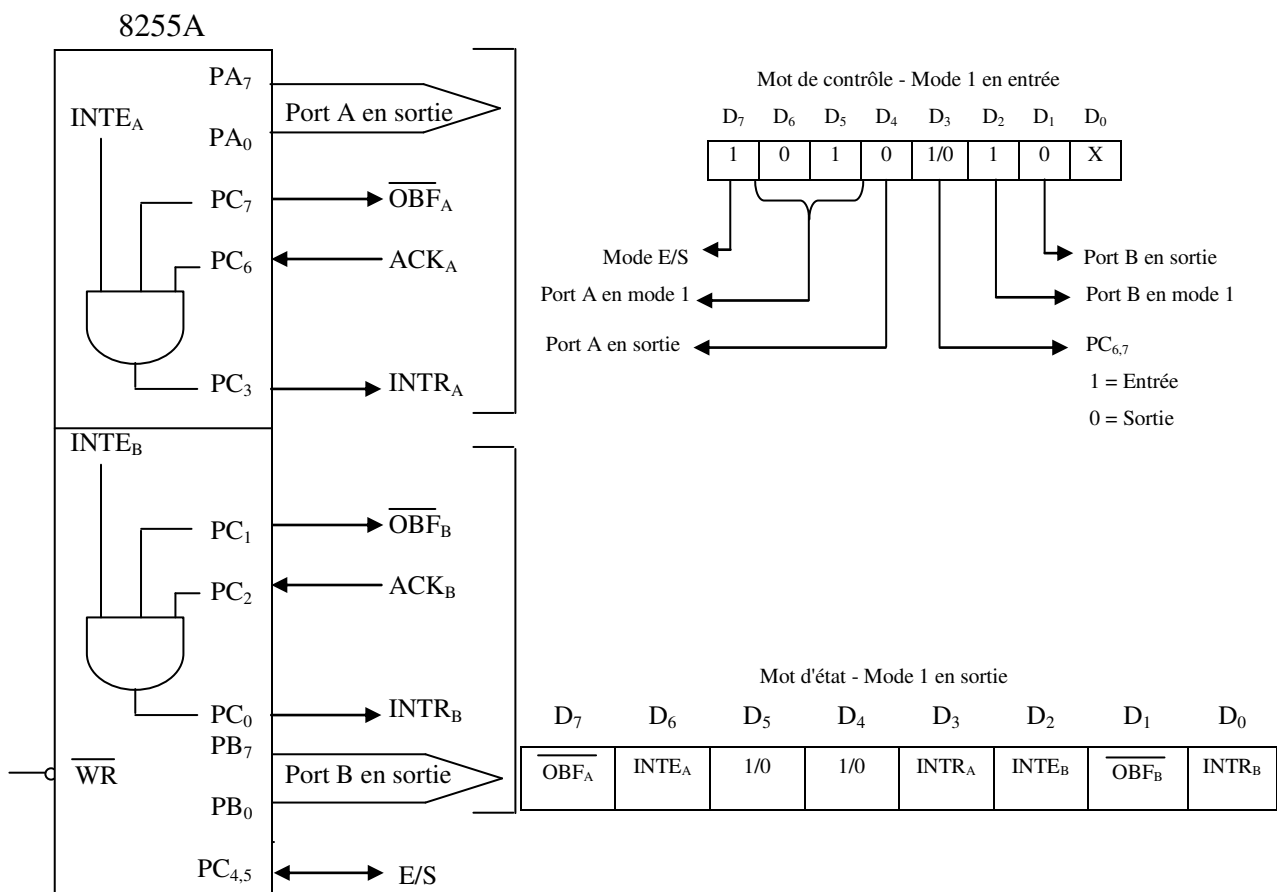


Figure VI.7 : 8255A en mode 1 en sortie

III.1.3. Exemple d'application du 8255A

La figure VI.8 montre un circuit d'interfaçage utilisant un 8255A en mode 1. Le port A est désigné en entrée pour un clavier avec interruption et le port B est désigné comme sortie pour être utilisé par l'imprimante.

1. Trouver les adresses par analyse du circuit logique.
2. Déterminer le mot de contrôle pour configurer le port A en entrée et le port B en sortie en mode 1.
3. Déterminer le mot BSR pour activer INTE_A (port A).
4. Déterminer l'octet de masquage pour vérifier la ligne $\overline{\text{OBF}}_B$ en entrée ou sortie (port B).

Ecrire les instructions d'initialisation et le sous programme de l'imprimante pour envoyer les caractères stockés en mémoire.

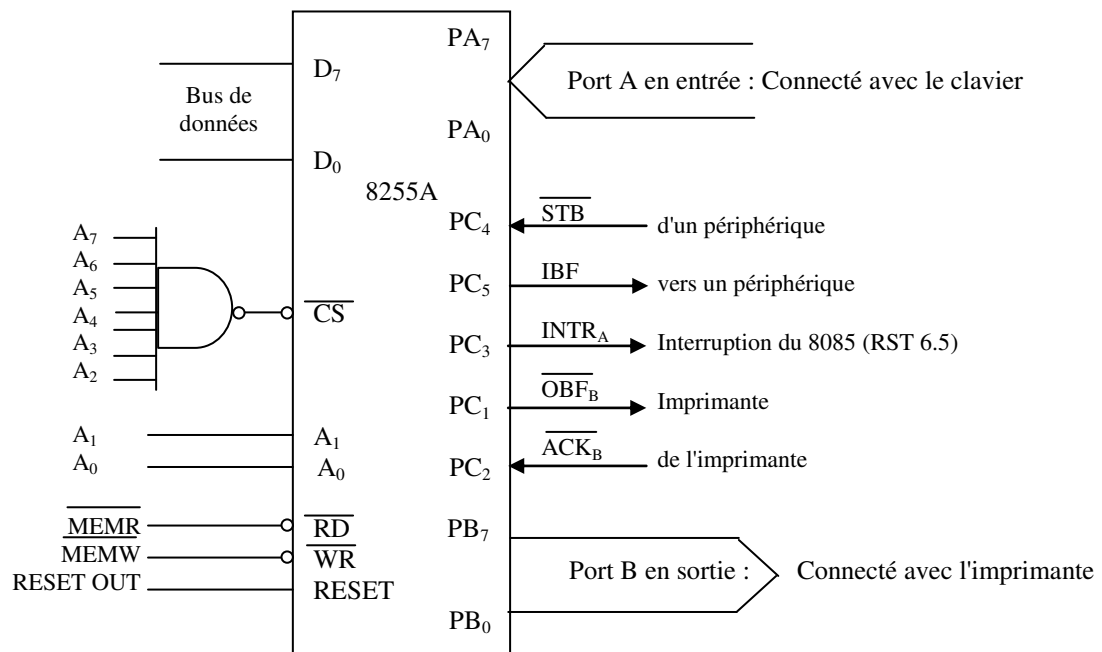


Figure VI.8 : Exemple d'application du 8255A en mode 1 en entrée et en sortie

Solution :

1. Adresses des ports et registre de contrôle :

Port A : FCH ($A_1=0$ et $A_0=0$)

Port B : FDH ($A_1=0$ et $A_0=1$)

Port C : FEH ($A_1=1$ et $A_0=0$)

Registre de contrôle : FFH ($A_1=1$ et $A_0=1$)

2. Mot de contrôle configuré pour port A en entrée et port B en sortie en mode 1

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	
1	0	1	1	0	1	0	0	=B4H
Fonctionnement en E/S	Port A en mode 1		Port A en entrée	PC _{4,5} Peu importe	Port B en mode 1	Port B en sortie	Bit D ₀ Peu importe	

Tableau VI.3 : Registre de contrôle

Pour générer le signal d'interruption $INTR_A$, la bascule $INTE_A$ doit être mise à 1, ce qui peut être effectué par l'utilisation du mode BSR pour mettre à 1 PC_4 .

La sortie de l'imprimante (port B) a un état contrôlé. Cependant l'état de la ligne \overline{OBF}_B peut être vérifié par la lecture du bit D₁ du port C_{Bas}.

3. Mot BSR pour mettre à 1 $INTE_A$. Pour la mise à 1 pour l'activation de l'interruption du port A ($INTE_A$), le bit PC_4 doit être mis à 1.

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	
0	0	0	0	1	0	0	1	=09H
Mode BSR		Peu importe			Bit $PC_{4,5}$		Bit D ₀ mis à 1	

4. Mot d'état pour vérifier \overline{OBF}_B

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
X	X	X	X	X	X	\overline{OBF}_B	X

Il faut masquer l'octet par 02H.

5. Initialisation du programme

```

MVI A,B4H      ; Port A en entrée et port B en sortie en mode 1
OUT FFH
MVI A,09H      ; Mise à 1 de  $INTE_A$  ( $PC_4$ )
OUT FFH        ; par utilisation du mode BSR
EI             ; Accepter les interruptions
CALL PRINT     ; passer à une autre tâche

```

Sous programme d'impression

```

PRINT:  LXI H,MEM ; Pointer vers le début de la zone des caractères
        MVI B,COUNT ; Nombre de caractères à imprimer
        NEXT: MOV A,M ; Lire le caractère
        MOV C,A ; Enregistrer caractère
STATUS: IN FEH ; Lire le port C pour l'état  $\overline{OBF}$ 
        ANI 02H ; Masquer tous les bits à l'exception de D1

```

```

JZ STATUS      ; Si état bas, l'imprimante n'est pas prête : attente par une
MOV A,C        ; boucle
OUT FDH        ; Envoyer un caractère au port B
INX H          ; Pointer vers le prochain caractère
DCR B
JNZ NEXT
RET

```

Mode 2 : Transfert bidirectionnel des données

Ce mode est utilisé en premier lieu dans des applications comme le transfert entre deux microordinateurs ou contrôleur d'interface des disques amovibles. Dans ce mode, le port A peut être configuré en mode bidirectionnel et le port B en mode 0 ou en mode 1. Le port A utilise cinq signaux du port C comme signaux handshake pour le transfert des données. Les trois signaux restants du port C peuvent être utilisés comme des E/S simple ou comme un handshake pour le port B. La figure VI.9 illustre ce mode.

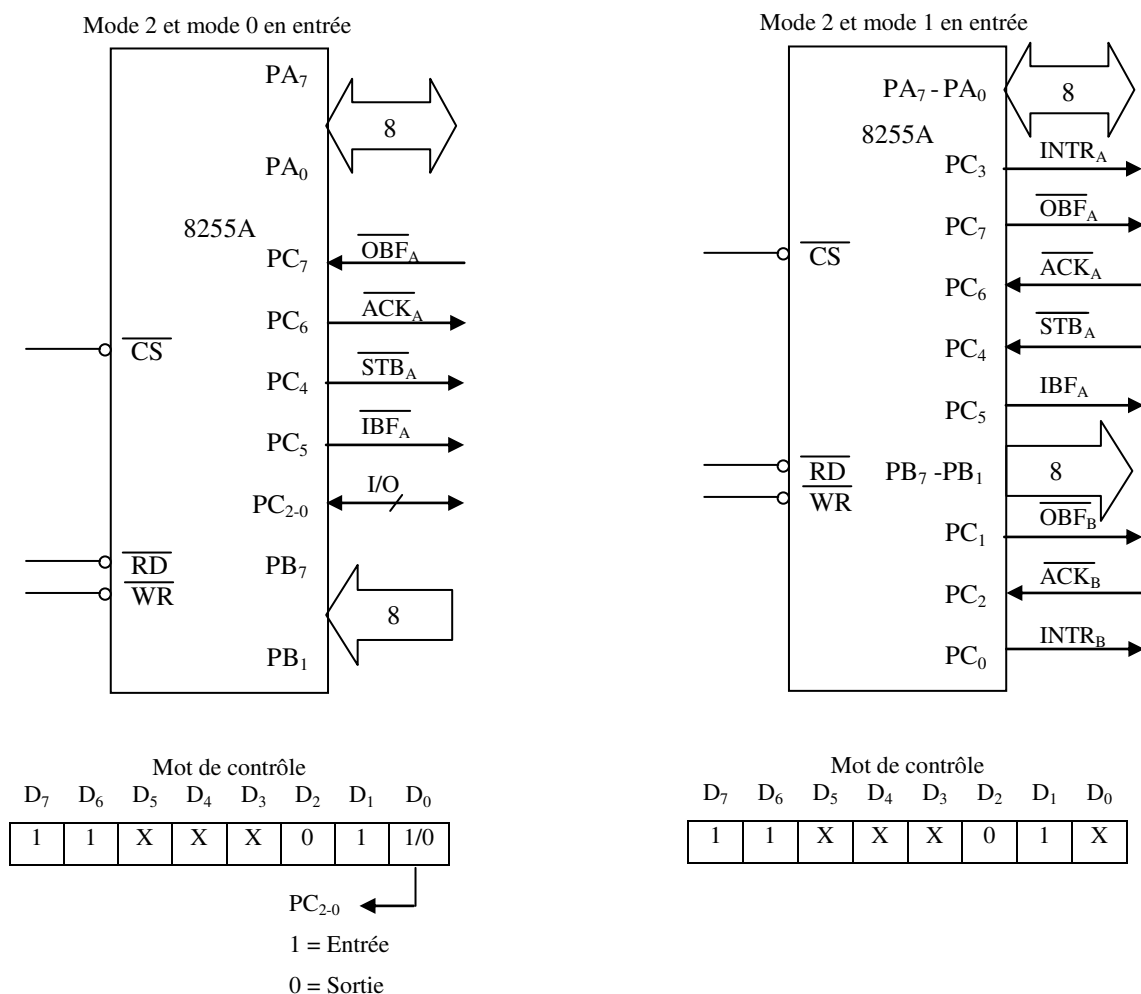


Figure VI.9 : 8255A en mode 2

III.2. Interface série [18]

Une interface série (Figure VI.10) permet d'échanger des données entre le microprocesseur et un périphérique bit par bit.

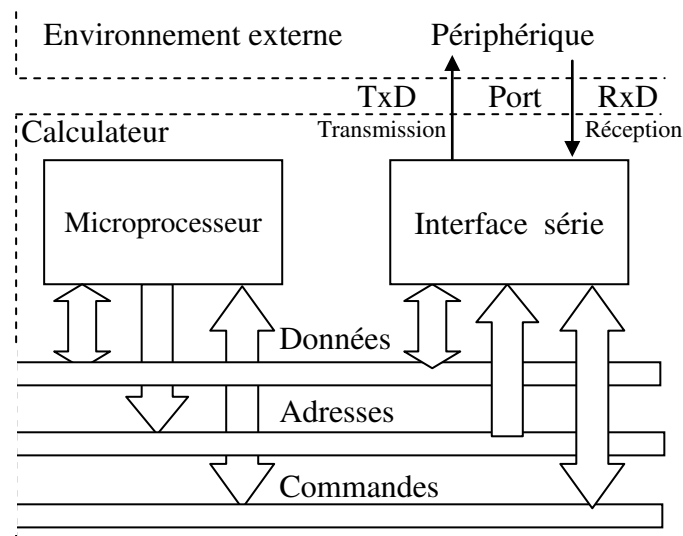


Figure VI.10 : Echange des données avec un périphérique bit à bit (série)

Avantage : diminution du nombre de connexions (1 fil pour l'émission, 1 fil pour la réception).

Inconvénient : vitesse de transmission plus faible que pour une interface parallèle.

Il existe deux types de transmissions séries :

- Asynchrone : chaque octet peut être émis ou reçu sans durée déterminée entre un octet et le suivant ;
- Synchrone : les octets successifs sont transmis par blocs séparés par des octets de synchronisation.

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

Exemple

Transmission du caractère 'E' (code ASCII 45H = 01000101B) sous forme série selon la norme RS232 :

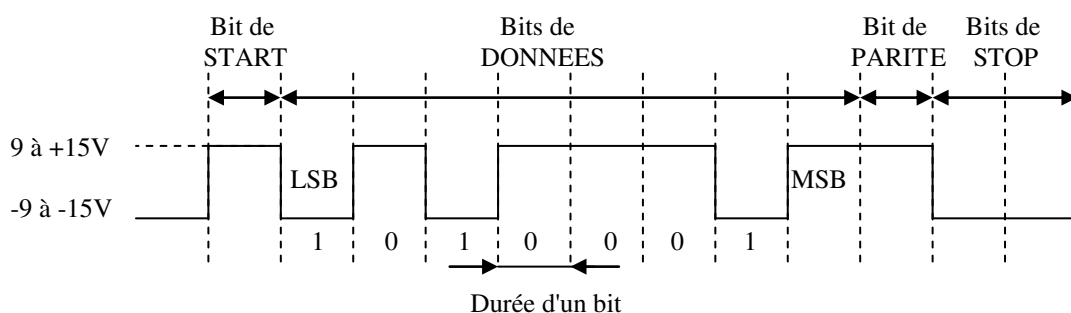


Figure VI.11 : Format d'un caractère pour une transmission série asynchrone

- L'état 1 correspond à une tension négative comprise entre -9 et -15 V, l'état 0 à une tension positive comprise entre $+9$ et $+15$ V. Au repos, la ligne est à l'état 1 (tension négative) ;
- Le bit de **Start** marque le début de la transmission du caractère ;
- Les bits de données sont transmis l'un après l'autre en commençant par le bit de poids faible. Ils peuvent être au nombre de 5, 6, 7 ou 8. Chaque bit est maintenu sur la ligne pendant une durée déterminée T . L'inverse de cette durée définit la fréquence de bit = nombre de bits par secondes = vitesse de transmission. Les vitesses normalisées sont : 50, 75, 110, 134.5, 150, 300, 600, 1200, 2400, 4800, 9600 bits/s ;
- Le bit de **Parité** (facultatif) est un bit supplémentaire dont la valeur dépend du nombre de bits de données égaux à 1. Il est utilisé pour la détection d'erreurs de transmission ;
- Les bits de **Stop** (1, 1.5 ou 2) marquent la fin de la transmission du caractère.

III.2.1. Principe d'une interface série

Un circuit intégré d'interface série asynchrone s'appelle un UART : Universal Asynchronous Receiver Transmitter) ; une interface série synchrone/asynchrone est un USART.

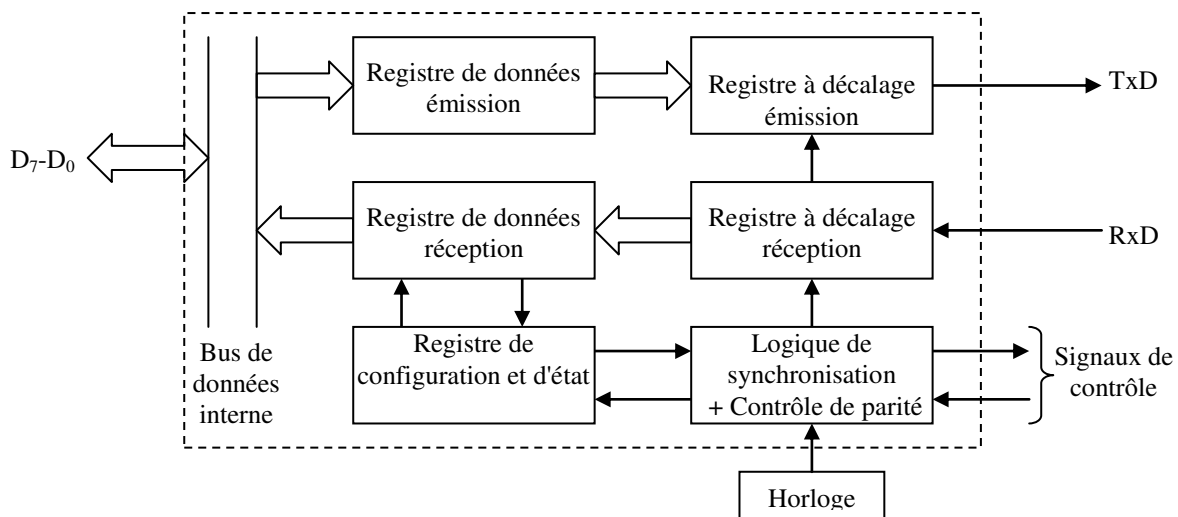


Figure VI.12 : Principe d'une interface série

III.2.2. 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.

Différents signaux sont transportés par les connecteurs du port RS232. Le tableau donne une description de ces signaux.

Signal	DB9 N° broche	DB25 N° broche	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	-	-

Tableau VI.4 : Lignes du port série RS232

Seuls les deux signaux TxD et RxD servent à transmettre les données. Les autres signaux sont des signaux de contrôle de l'échange de données.

- Quand l'équipement DTE veut transmettre des données, il active le signal DTR. Si le DCE est prêt à recevoir les données, il active le signal DSR puis le signal DCD : la communication peut débuter,
- Lorsque l'équipement DTE a une donnée à émettre, il active le signal RTS. Si le DCE peut recevoir la donnée, il active CTS : le DTE envoie la donnée sur la ligne TxD,
- Si l'équipement DCE veut demander une pause dans la transmission, il désactive CTS : le DTE arrête la transmission jusqu'à ce que CTS soit réactivé. C'est un contrôle matériel du flux de données ;
- Lorsque la transmission est terminée, les signaux RTS, CTS, DTR, DCD et DSR sont successivement désactivés.

III.2.3. Mise en œuvre d'une interface série, l'UART 8250

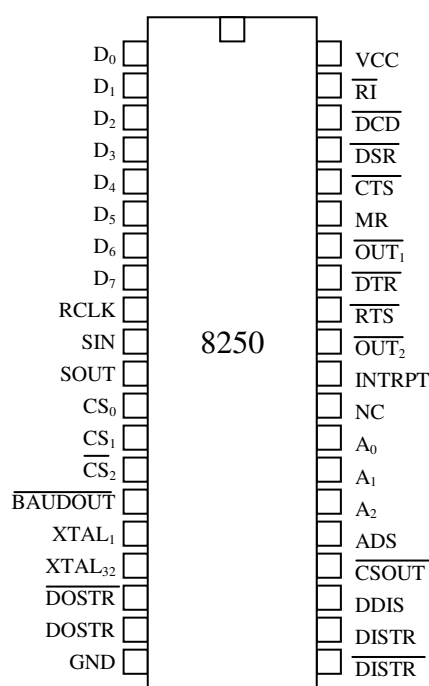


Figure VI.13 : Brochage du l'UART 8250

Le 8250 possède 11 registres. Comme il n'y a que 3 bits d'adresses (A_0 , A_1 et A_2), plusieurs registres doivent se partager la même adresse :

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

Tableau VI.5 : Sélection des registres du 8250

En fonction de l'état de DLAB (Divisor Latch Access Bit = bit de poids fort du registre LCR), on a accès soit au registre d'émission/réception, soit au diviseur d'horloge, soit au masque d'interruptions.

Line Control Register (LCR)

Bits 0 et 1 : longueur du mot transmis,	bit 1	bit 0	
	0	0	→ 5 bits
	0	1	→ 6 bits
	1	0	→ 7 bits
	1	1	→ 8 bits
Bit 2 : nombre de bits de stop,	0 → 1 bit de stop,		
	1 → 1.5 bits de stop si 5 bits, 2 bits de stop sinon;		
bit 3 : autorisation de parité,	0 → pas de parité,		
	1 → parité générée et vérifiée ;		
bit 4 : sélection de parité,	0 → parité impaire,		
	1 → parité paire;		
bit 5 : /			
bit 6 : /			
bit 7 : DLAB (Divisor Latch Access bit),	0 → accès aux registres d'émission, de réception et IER,		
	1 → accès au diviseur d'horloge.		

Line Status Register (LSR)

bit 0 : 1 → donnée reçue ;

bit 1 : 1 → erreur d'écrasement de caractère ;

bit 2 : 1 → erreur de parité ;

bit 3 : 1 → erreur de cadrage (bit de stop non valide) ;

bit 4 : 1 → détection d'un état logique 0 sur RxD pendant une durée supérieure à la durée d'un mot ;

bit 5 : 1 → registre de transmission vide ;

bit 6 : 1 → registre à décalage vide ;

bit 7 : non utilisé, toujours à 0.

Diviseur d'horloge (DLM,DLL)

La vitesse de transmission est fixée par la valeur du diviseur d'horloge :

$$vitesse(bits/s) = \frac{\text{fréquence d'horloge}}{16 \times (DLM, DLL)}$$

Exemple de calcul :

Vitesse de transmission désirée = 1200 bit/s, fréquence d'horloge = 1.8432 MHz.

Détermination de la valeur du diviseur d'horloge :

$$\text{diviseur} = \frac{\text{fréquence d'horloge}}{16 \times \text{vitesse}} = \frac{1.8432 \times 10^6}{16 \times 1200} = 96 \Rightarrow DLM = 0 \quad \text{et} \quad DLL = 96$$

III.2.4. Ports USB (Universal Serial Bus)

Le nouveau Bus USB de connexion série, promu par les principaux constructeurs informatiques et éditeurs de logiciels (Compaq, DEC, IBM, Intel, Microsoft, NEC, Nortel) et dont les premières spécifications doivent permettre le raccordement sur une prise unique de plusieurs équipements divers (imprimante, téléphone, modem, fax, clavier, souris, scanners, écrans...). On désire ainsi éviter la multiplication actuelle des connecteurs sur les PC. 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 débit brut doit être partagé entre tous les appareils connectés au bus.

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.

III.3. Timer (Temporisateur), Intel 8254 [16]

Le Timer 8254 est un contrôleur d'entrées/sortie. Il fait le couplage avec une horloge pour mettre à disposition de la machine un mécanisme de mesure de temps.

Le schéma de la figure IV.14 donne une représentation minimale de connexion en se limitant au bus de données et aux signaux permettant de sélectionner les registres du contrôleur.

Les bits A_0 et A_1 permettent d'accéder à l'un des quatre registres internes : un registre par compteur (Compteur 0 : $A_1=0$ et $A_0=0$, Compteur 1 : $A_1=0$ et $A_0=1$, Compteur 2 : $A_1=1$ et $A_0=0$) et registre de commande ($A_1=1$ et $A_0=1$). Le CS est le résultat du décodage d'adresses des autres bits du bus d'adresses.

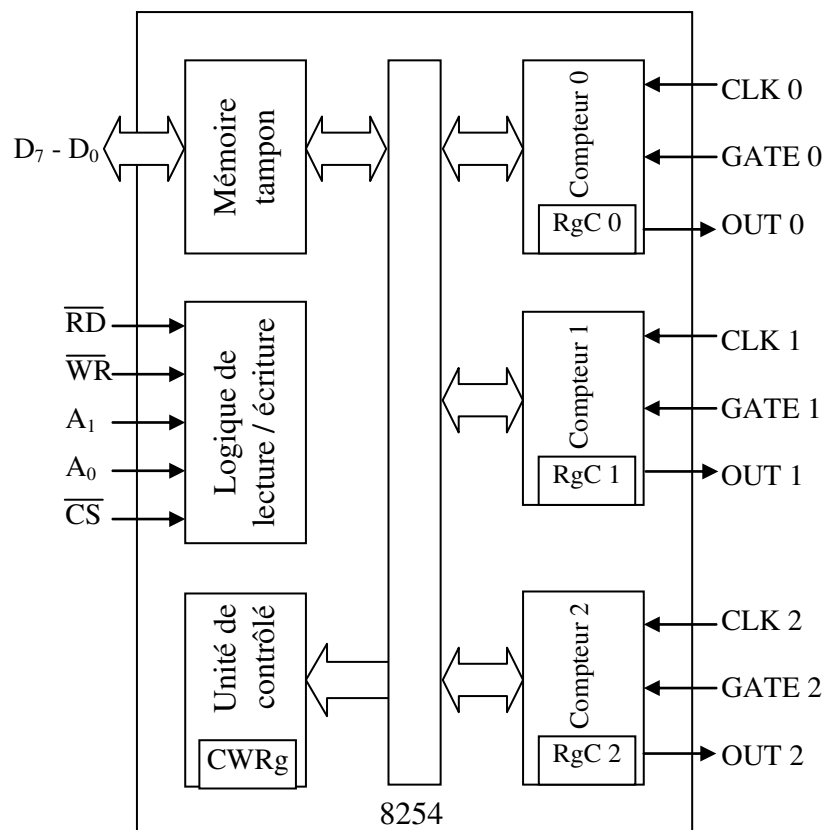


Figure VI.14 : Schéma bloc interne du 8254

Le circuit comporte trois unités fonctionnelles identiques, appelés compteurs, axées sur le comptage du temps.

Un compteur comporte deux entrées matérielles (CLK), Gate et une entrée logicielle, le compteur RgC. l'entrée CLK reçoit des impulsions qui peuvent être périodiques ou non.

GATE est une entrée matérielle de validation du comptage. Le registre 16 bits RgC est accessible par programmation et peut être pré positionné à une certaine valeur. Chaque impulsion sur l'entrée CLK décrémente le compteur. La sortie OUT est initialement à 0 passe à 1 lorsque RgC passe à la valeur 0. A partir de l'initialisation du registre compteur, le 8254

propose 6 modes de fonctionnement. C'est la programmation du registre CWRg (Control Word Register) qui permet de configurer le mode de fonctionnement d'une unité fonctionnelle avec les valeurs des bits M2 à M0. SC1 et SC0 servent à définir dans un mot de commande l'unité de comptage (Sélection du Compteur 0 : SC1=0 et SC0=0, Compteur 1 : SC1=0 et SC0=1, Compteur 2 : SC1=1 et SC0=0) faisant l'objet du paramétrage par le mot de commande. Les bits RW0 et RW1 définissent le mot de transfert des données. Le bit BCD permet de réaliser un comptage BCD.

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	
SC1	SC0	RW1	RW0	M2	M1	M0	BCD	
							0	: Comptage binaire
							1	: Comptage BCD
				0	0	0		: Mode 0
				0	0	1		: Mode 1
				X	1	0		: Mode 2
				X	1	1		: Mode 3
				1	0	0		: Mode 4
				1	0	1		: Mode 5
		0	0					: Commande du latch Compteur
		0	1					: R/W LSB 8 bits
		1	0					: R/W MSB 8 bits
		1	1					: R/W LSB ensuite MSB
0	0							: Sélection du Compteur 0
0	1							: Sélection du Compteur 1
1	0							: Sélection du Compteur 2
1	1							: Commande de lecture

Tableau IV.6 : Mot de contrôle CWRg du 8254

Exemple du mode 0

Ce mode est adapté au comptage d'évènements. Après avoir programmé l'unité de comptage par le biais du registre de commande et initialisé le compteur. Lorsque GATE passe à 1. Le compteur est alors décrémenté à chaque évènement signalé par une impulsion sur l'entrée CLK. La sortie OUT passe de sa valeur initiale 0 à la valeur 1 lorsque le comptage arrive à 0. OUT reste alors au niveau haut jusqu'au chargement d'une nouvelle valeur dans le registre compteur. La sortie OUT peut servir de signal d'interruption.

Exemple :

1. Identifier les adresses des ports et du registre de contrôle et le compteur 2 de la figure IV.15
2. Ecrire un sous programme pour initialiser le mode 0 pour un comptage de 50.000.
3. Ecrire le programme principal pour afficher les secondes par appel du sous programme plusieurs fois.

Solution :

1. Adresses des ports : CS est actif si $A_7=1$ et le registre de contrôle est sélectionné par $A_1=1$ et $A_0=1$. Le compteur 2 est sélectionné quand $A_1=1$ et $A_0=0$. Supposons que les lignes d'adresses non utilisées A_6 à A_2 sont à 0. On aura ce qui suit :

Registre de contrôle = 83H

Compteur 2 = 82H

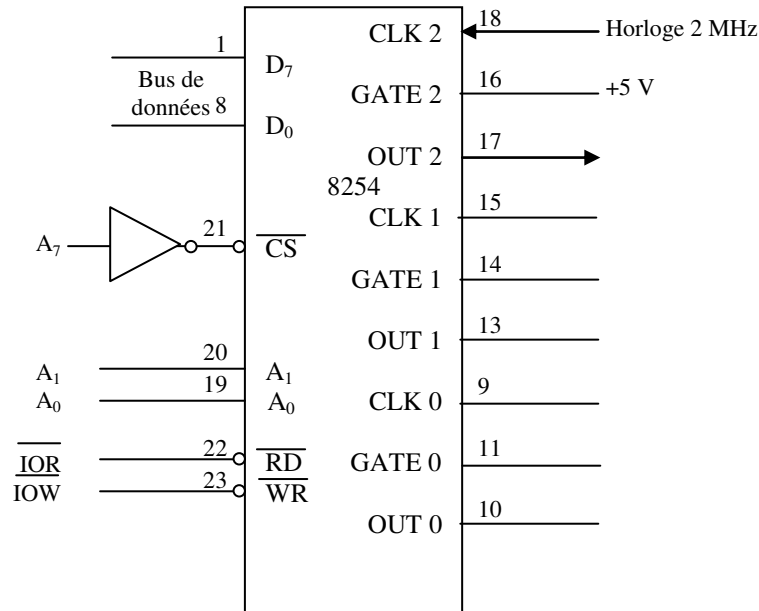


Figure VI.15 : Exemple de sélection du 8254

2. Sous programme : pour initialiser le 8254, une opération de chargement avec la valeur B0H est donnée comme suit :

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
1	0	1	1	0	0	0	0

Mot de contrôle pour mémoriser le comptage : les bits D₅ et D₄ doivent être à 0. Dans ce cas le contenu du mot de contrôle est 80H.

```

COMPTEUR: MVI A,B0H      ; Initialiser compteur 2
           OUT 83H        ; Ecrire dans le registre de contrôle
           MVI A,OCTETB   ; l'octet de poids faible de comptage 50000
           OUT 82H        ; Charger le compteur 2 avec l'octet de poids faible
           MVI A, OCTETH   ; l'octet de poids fort de comptage 50000
           OUT 82H        ; Charger le compteur 2 avec l'octet de poids fort
LECTURE:   MVI A,80H      ; Mot de contrôle pour mémoriser le comptage
           OUT 83H        ; Ecrire dans le registre de contrôle
           IN 82H         ; Lecture de l'octet de poids faible

```

MOV D,A	; Stocker l'octet de poids faible dans D
IN 82H	; Lecture de l'octet de poids fort
ORA D	; OR les octets de poids faible et fort pour mettre Z à 1
JNZ LECTURE	; Si comptage $\neq 0$ aller au prochain comptage
RET	

3. Programme principal : Le sous programme COMPTEUR offre un délai de 25 ms (50000 x 0.5 us Horloge) ; Si cette routine est appelé 40 fois, le délai total sera une seconde.

	LXI SP,STACK	; Initialise SP
	MVI B,00H	; Efface B pour charger le nombre en seconde
SECOND:	MVI C,28H	; C est chargé par 28H=40 ₁₀
WAIT:	CALL COMPTEUR	: Attente pour 25 ms
	DCR C	
	JNZ WAIT	; Est-ce une seconde? Sinon retour et attente
	MOV A,B	
	ADI 01H	; Ajouter une seconde
	DAA	
	OUT PORT1	
	MOV B,A	; Mémoriser les secondes
	JMP SECOND	; Retour et recommencer pour la prochaine seconde

CHAPITRE VII

LES INTERRUPTIONS

I. Définition d'une interruption [9]

Soit un microprocesseur qui doit échanger des informations avec un périphérique : Il y a deux méthodes possibles pour recevoir les données provenant des périphériques :

1- **Scrutation** périodique (ou **polling**) : le programme principal contient des instructions qui lisent cycliquement l'état des ports d'E/S.

Avantage : facilité de programmation.

Inconvénients : – Perte de temps s'il y a de nombreux périphériques à interroger ;

– De nouvelles données ne sont pas toujours présentes ;

– Des données peuvent être perdues si elles changent rapidement.

2- **Interruption** : lorsqu'une donnée apparaît sur un périphérique, le circuit d'E/S le signale au microprocesseur pour que celui-ci effectue la lecture de la donnée : c'est une demande d'interruption (IRQ : Interrupt Request) :

Avantage : le microprocesseur effectue une lecture des ports d'E/S seulement lorsqu'une donnée est disponible, ce qui permet de gagner du temps et d'éviter de perdre des données.

Exemples de périphériques utilisant les interruptions :

- Clavier : demande d'interruption lorsqu'une touche est enfoncée ;
- Port série : demande d'interruption lors de l'arrivée d'un caractère sur la ligne de transmission.

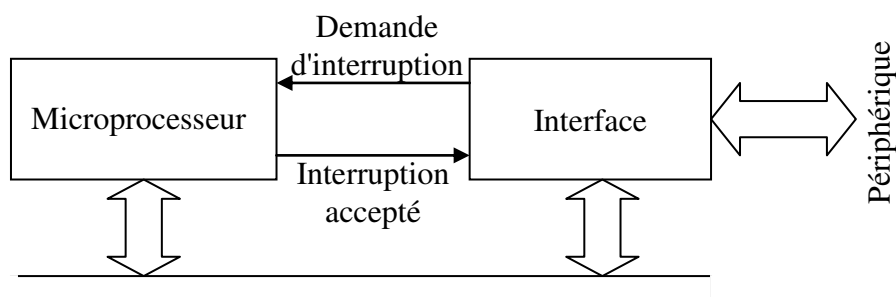


Figure VII.1 : Signaux d'interruption entre microprocesseur /interface

II. Prise en charge d'une interruption par le microprocesseur [9]

A la suite d'une demande d'interruption par un périphérique et si la demande d'interruption est acceptée :

- Le microprocesseur termine l'exécution de l'instruction en cours et désactive les interruptions,
- Il range le contenu du compteur de programme dans la pile, adresse de la prochaine instruction à exécuter, pour un éventuel retour,
- Il émet un accusé de réception (Interrupt Acknowledge : \overline{INTA}) indiquant au circuit d'E/S que la demande d'interruption est acceptée :
- Il abandonne l'exécution du programme en cours et va exécuter un sous-programme de service de l'interruption (ISR : Interrupt Service Routine),
- Après l'exécution de l'ISR, les registres sont restaurés à partir de la pile et le microprocesseur reprend l'exécution du programme qu'il avait abandonné.

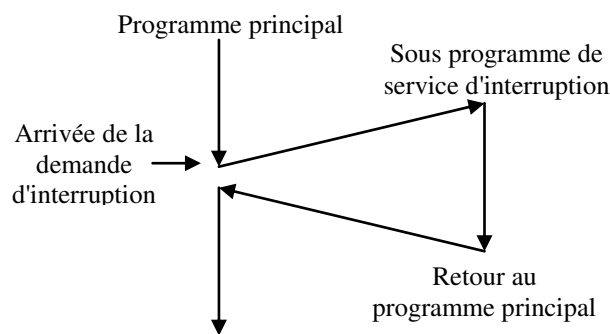


Figure VII.2 : Etapes d'exécution de l'ISR

Remarques

1. Le microprocesseur peut refuser la demande d'interruption : celle-ci est alors masquée. Le masquage d'une interruption se fait généralement en positionnant une bascule "EI" (Enable Interrupt).
2. Il existe une interruption non masquable (TRAP) qui est toujours prise en compte par le microprocesseur
3. La dernière instruction d'un sous-programme de service d'interruption doit être l'instruction "RET".

III. Interruptions du 8085 [16]

Un service d'interruption masquable peut être activé par l'instruction "EI". Dans ce cas une bascule "EI" est mise à 1 sinon l'interruption est désactivée par l'instruction "DI" (IE=0).

Deux types d'interruptions sont reconnus par le 8085 :

- Interruptions logicielles produites par l'instruction RST.
- Interruptions matérielles produites par l'activation d'au moins de l'une des cinq lignes RST7.5, RST6.5, RST5.5, INTR et TRAP du microprocesseur.

III.1. Interruptions logicielles (Instruction RST "Restart")

Le 8085 comprend huit instructions RST (Tableau VII.1). Ce sont des instructions à un seul octet. Ces instructions sont des appels qui permettent de transférer l'exécution des programmes à des positions mémoire bien spécifiées. Un retour au programme principal sera assuré par la récupération du CP de la pile dès l'exécution de l'instruction RET.

RST 0	CALL 0000H	RST 4	CALL 0020H
RST 1	CALL 0008H	RST 5	CALL 0028H
RST 2	CALL 0010H	RST 6	CALL 0030H
RST 3	CALL 0018H	RST 7	CALL 0038H

Tableau VII.1 : Adresses d'appels des interruptions logicielles

III.2. Interruptions matérielles

Il existe deux types d'interruptions pour le 8085 : non vectorisées et vectorisées.

III.2.1. Interruption non vectorisée INTR

Dès la réception d'un signal INTR de la part d'une interface d'entrée, le microprocesseur envoie un accusé de réception sur la ligne INTA pour indiquer à l'interface que l'interruption a été acceptée (si EI=1). La position mémoire du sous programme d'interruption (ISR) est inconnu. Donc c'est à l'interface d'envoyer au microprocesseur l'adresse de l'ISR.

III.2.2. Interruptions vectorisées

Le 8085 possède quatre interruptions vectorisés par ordre de priorité : TRAP (Interruption non masquable), RST 7.5, RST 6.5, RST 5.5. Les positions des sous programmes d'interruption sont comme suit :

TRAP	0024H
RST 7.5	003CH
RST 6.5	0034H
RST 5.5	002CH

Tableau VII.2 : Adresses d'appels des lignes d'interruptions matérielles

III.2.2.1. TRAP

TRAP est reconnu par NMI (Non Maskable Interrupt). Cette interruption ne peut être inhibée. C'est une interruption utilisée dans les situations critiques comme problème d'alimentation,...

III.2.2.2. RST 7.5, 6.5, et 5.5

Ce sont des interruptions masquables activés par EI et SIM (Set Interrupt Mask) dont la description est donnée en ce qui suit.

* Instruction SIM

C'est une instruction à un seul octet. Cette instruction lit le contenu de l'accumulateur et active ou désactive les instructions masquables RST 7.5, RST 6.5 et RST 5.5 suivant le contenu de A.

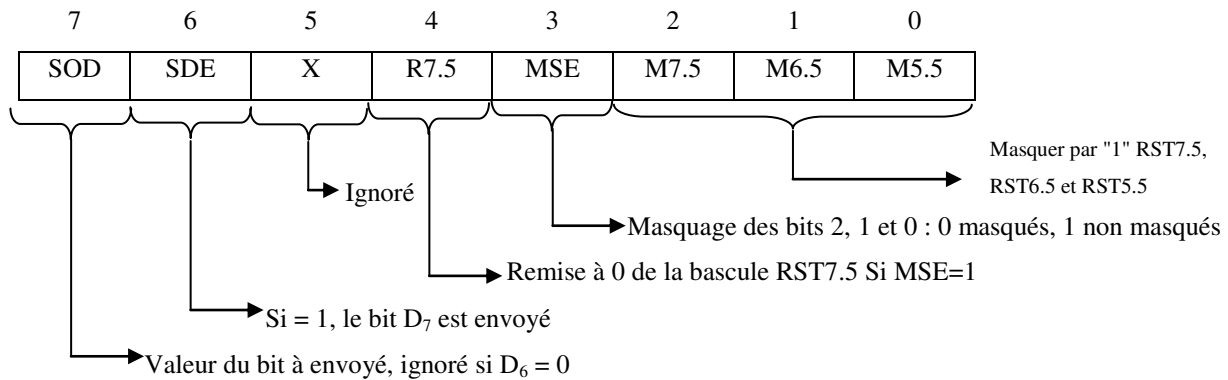


Figure VII.3 : Configuration des interruptions matérielles RST

1. Le bit D₃ est un bit de contrôle, si ce bit est à 1 les bits D₂, D₁ et D₀ sont effectives. Le niveau logique 0 de ces trois bits active l'interruption et 1 refuse les interruptions. Sinon pour D₃=0, ces trois bits sont ignorés.
2. Le bit D₄ est utilisé pour la commande de RST 7.5. Si ce bit = 1, RST 7.5 est mise à zéro. Ceci est utilisé pour ignorer RST 7.5.
3. Les bits D₇ et D₆ sont utilisés dans la transmission série.

Exemple :

1. Pour accepter les interruptions dans un système à base 8085.

EI

MVI A,08H

SIM

2. Pour la mise à 0 de l'interruption 7.5

MVI A, 18H

SIM

* Interruptions en attente

Le fait qu'il existe plusieurs lignes d'interruption, quand une interruption est servie, l'autre interruption doivent attendre. Le 8085 possède une deuxième instruction appelée RIM (Read Interrupt Mask) pour savoir les interruptions en attente.

* Instruction RIM

C'est une instruction à un seul octet. Cette instruction est définie comme suit :

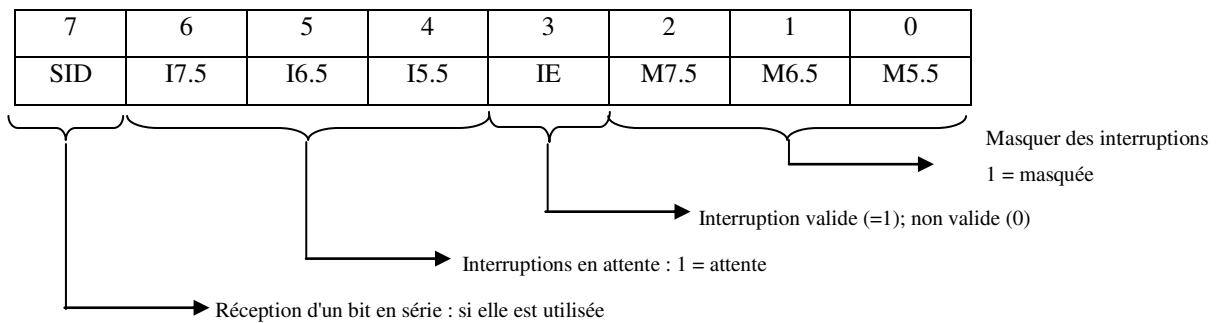


Figure VII.4 : Lecture des états des interruptions matérielles

1. Pour la lecture de l'état des interruptions masquées. Cette instruction charge l'accumulateur A avec 8 bits indiquant l'état des masques.
2. Pour l'identification des interruptions en attente. Les bits D₆, D₅ et D₄ identifient les interruptions en attente.
3. Pour la réception sérielle des données, le bit D₇ est utilisé.

Exemple :

On suppose que le 8085 est entrain d'exécuter une demande d'interruption RST 7.5. Une vérification de l'attente de RST 6.5 par exemple. S'il y a une attente, une activation de RST 6.5 doit être faite sans l'affectation des autres interruptions, sinon retour au programme principal.

```

RIM          ; Lecture du masque d'interruptions
MOV B,A      ; Sauvegarder le masque dans B
ANI 20H      ; Vérifier s'il y a une attente de RST 6.5
JNZ SUIVANT
EI
RET          ; RST 6.5 non pas en attente, retour au programme principal
SUIVANT:    MOV A,B      ; RST est en attente
ANI 0DH      ; Valider RST 6.5 par mise à 0 de D1
ORI 08H      ; Valider SIM par mise à 1 de D3
SIM
JMP SERV     ; Aller exécuter le service de la routine RST 6.5
  
```

IV. Contrôleur programmable d'interruptions, PIC 8259 [16]

Pour pouvoir connecter plusieurs périphériques utilisant des interruptions, on peut utiliser le contrôleur programmable d'interruptions 8259 dont le rôle est de :

- recevoir des demandes d'interruptions des périphériques,
- résoudre les priorités des interruptions,

- générer les signaux d'interruptions,
- émettre le numéro de l'interruption sur le bus de données.

Un 8259 peut gérer jusqu'à 8 demandes d'interruptions matérielles. Le brochage d'un tel circuit est donné par la figure VII.5.

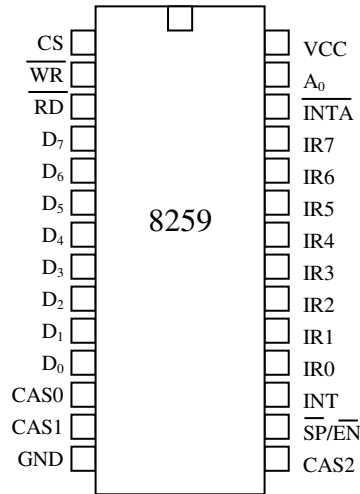


Figure VII.5 : Brochage du 8259

CAS0, 1, 2: Utilisées dans la mise des PIC8259s en cascade pour étendre le nombre d'interruptions du microprocesseur jusqu'à 64. En mode Esclave, les CAS0,1,2 sont ignorés.

IR0..7 : sont des entrées de requête (Interrupt Request) ; suivant l'initialisation, on peut choisir des entrées sensibles au niveau haut ou au front montant.

SP/EN (Slave Programming/Enable): (SP/EN = '1' → Maître, SP/EN = '0' → Esclave).

Remarque : si le nombre de demandes d'interruptions est supérieur à 8, on peut placer plusieurs 8259 en cascade :

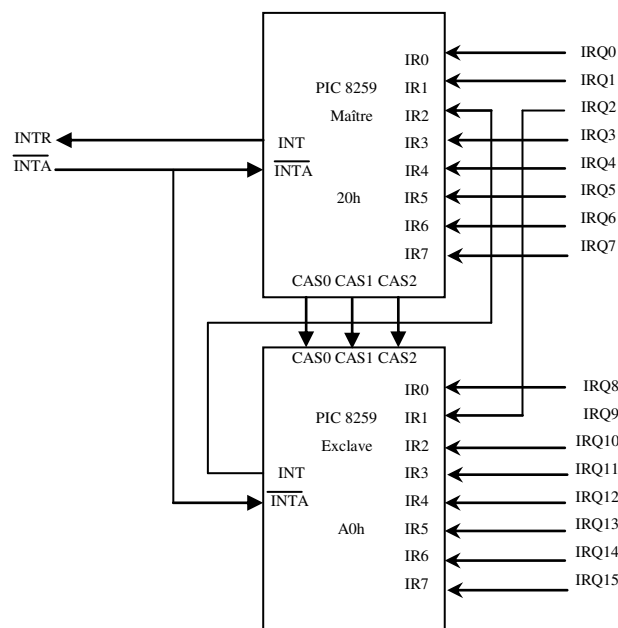


Figure VII.6 : Mise en cascade de deux 8259

Le composant prévoit deux modes de gestion des priorités:

- La rotation automatique: dans ce mode, lorsque un périphérique a été servi, il reçoit la priorité la plus basse de façon à ce que dans le pire des cas, un périphérique demandeur n'ait qu'à attendre que les sept autres périphériques aient été servis au plus une seule fois.
- La rotation spécifique: dans ce mode, le programmeur peut changer l'entrée affectée à la priorité la plus basse, les autres priorités étant fixées en fonction de cette dernière.

Dans les PC, c'est la deuxième solution qui a été retenue. Ce mode est programmé lors de l'initialisation de votre carte mère par le BIOS. C'est l'entrée IRQ7 qui reçoit la priorité la plus basse et IRQ0 la priorité la plus élevée. Le fait que le second contrôleur soit mis en cascade sur l'entrée IRQ2 du premier contrôleur, nous aurons dans l'ordre décroissant de priorité les entrées suivantes:

IRQ0, IRQ1, IRQ8, IRQ9, IRQ10, IRQ11, IRQ12, IRQ13, IRQ14, IRQ15, IRQ3, IRQ4, IRQ5, IRQ6, IRQ7.

Le PIC dispose de registres internes permettant la gestion des interruptions. On peut citer:

- IRR: (Interrupt request Register)
- IMR: (Interrupt Mask Register)
- ISR: (In Service Register)

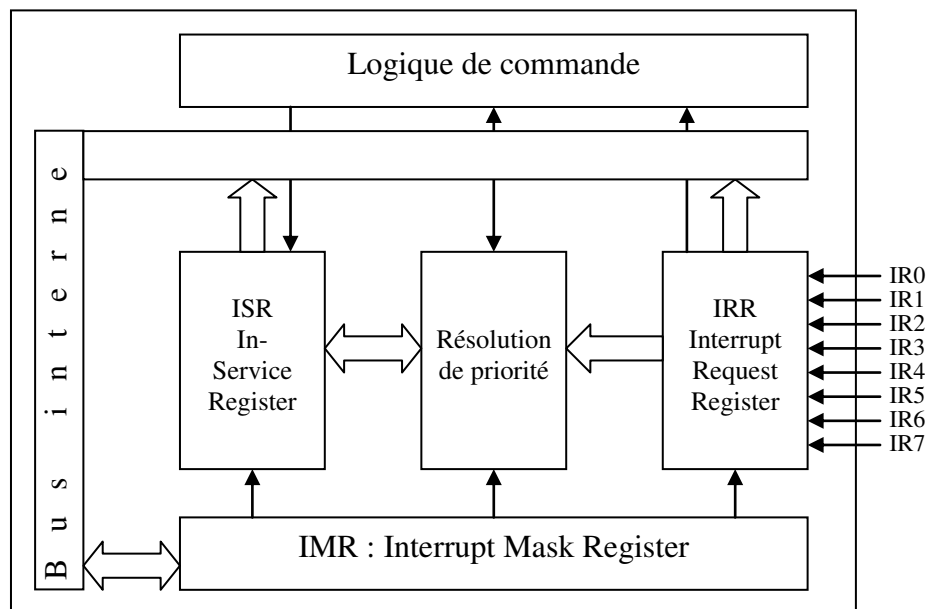


Figure VII.7 : Diagramme Interne du PIC 8259

Les huit lignes d'entrée passent à travers le registre de masquage qui va déterminer si une entrée est masquée ou pas. Si elle est masquée, la demande arrivant sur cette ligne ne sera pas prise en compte. Si elle n'est pas masquée, la demande sera enregistrée dans le registre de

demande d'interruption qui la maintiendra jusqu'au moment où elle sera servie. En fonction des priorités programmées, le PIC pourra déterminer de quelle interruption il doit s'occuper. Il envoie donc une demande INT vers le microprocesseur en activant la borne INT de ce dernier. Le microprocesseur termine son instruction en cours et va renseigner au PIC qu'il accepte la demande en activant sa borne $\overline{\text{INTA}}$ (Interrupt Acknowledge).

Dès la réception de ce signal, le PIC va positionner le bit correspondant à l'interruption servie dans le registre ISR à '1' et le bit correspondant dans le registre IRR à '0'.

Le microprocesseur va alors envoyer un deuxième signal $\overline{\text{INTA}}$ indiquant au PIC qu'il doit déposer sur le bus de données le numéro de l'interruption logicielle qui doit être servie. Ce numéro est construit de la façon suivante par le PIC: les trois bits de poids faible correspondent au numéro de l'interruption matérielle tandis que les cinq bits de poids fort correspondent à une valeur préprogrammée lors de l'initialisation du PIC. Pour les PC, le PIC maître reçoit la valeur 00001 et le PIC esclave la valeur 01110. Si l'on prend l'IRQ matérielle 3 (PIC maître IRQ 3), cela correspondra donc à 00001011b (0BH) tandis que si l'on prend l'IRQ matérielle 10 (PIC esclave IRQ 2), cela correspondra à 01110010 (72).

Analysons maintenant la gestion des priorités:

Partons d'une condition initiale où l'interruption matérielle 4 est servie. Nous retrouverons alors les données suivantes dans les différents registres:

IR7	IR6	IR5	IR4	IR3	IR2	IR1	IR0
0	0	0	0	0	0	0	0
IS7	IS6	IS5	IS4	IS3	IS2	IS1	IS0
0	0	0	0	0	0	0	0

- Supposons que deux demandes d'interruptions se présentent sur les entrées IR6 et IR2 du PIC. L'IRQ6 étant moins prioritaire que l'interruption 4 en cours de traitement, la demande restera en attente.

L'IRQ2 étant plus prioritaire que l'interruption 4 en cours de traitement, elle sera prise en compte par le PIC. Celui-ci enverra donc une demande INT vers le microprocesseur qui répondra par la succession de deux signaux INTA. Les registres contiendront alors les données suivantes:

IR7	IR6	IR5	IR4	IR3	IR2	IR1	IR0
0	1	0	0	0	0	0	0
IS7	IS6	IS5	IS4	IS3	IS2	IS1	IS0
0	0	0	1	0	1	0	0

Il est important de signaler que la routine de l'IRQ4 en cours d'exécution a été interrompue au bénéfice de la routine de l'IRQ2. Lorsque cette routine se termine, elle envoie un EOI (End Of Interrupt) vers le PIC. Celui ci, en analysant son registre ISS sait qu'il s'agit d'un signal de fin d'interruption pour l'IRQ2.

Voici le contenu des registres:

IR7	IR6	IR5	IR4	IR3	IR2	IR1	IR0
0	1	0	0	0	0	0	0

IS7	IS6	IS5	IS4	IS3	IS2	IS1	IS0
0	0	0	1	0	0	0	0

Le processeur ayant terminé l'exécution de la routine de l'IRQ2, il revient à l'exécution du code avant interruption: il s'agissait de la routine de l'IRQ4. Lorsque cette routine se termine, elle envoie un EOI (End Of Interrupt) vers le PIC. Celui ci, en analysant son registre ISS sait qu'il s'agit d'un signal de fin d'interruption pour l'IRQ4. Voici le contenu des registres:

IR7	IR6	IR5	IR4	IR3	IR2	IR1	IR0
0	1	0	0	0	0	0	0

IS7	IS6	IS5	IS4	IS3	IS2	IS1	IS0
0	0	0	0	0	0	0	0

Le PIC peut alors prendre en compte la demande de l'IRQ6 puisqu'il n'y a plus aucune autre demande plus prioritaire en cours de traitement.

Redirection IRQ2/IRQ9: malgré le fait que cette interruption ait été réservée dès les premiers PC XT pour une mise en cascade d'un deuxième contrôleur, certains constructeurs ont utilisé cette interruption pour une toute autre fonction en l'utilisant pour certains périphériques. Lorsque les PC AT sont apparus sur le marché, il a fallu jouer d'artifice pour maintenir une compatibilité avec de tels périphériques. C'est l'IRQ9 qui fut alors utilisée pour remplacer l'IRQ2 mais le problème était que l'IRQ9 était présente sur le contrôleur esclave alors que l'IRQ2 se trouvait initialement sur le contrôleur maître. Le problème fut réglé de la façon suivante: au niveau de la routine associée à l'IRQ9, une EOI est envoyé vers le PIC esclave et ensuite un ISR est appelé pour l'IRQ2.

IV.1. Initialisation du PIC 8259

1- Configuration de l'ICW1 (20h pour le 1er 8259 et A0h pour le 2ème) :

- Bits 7,6 et 5 à 0
- Bit 4 à 1 pour marquer le début de la séquence d'initialisation

- Bit 3 à 1 pour déclencher les interruptions par niveau ou à 0 si par front
- Bit 2 à 0
- Bit 1 à 1 si le 8259 est seul ou à 0 si cascadié
- Bit 0 à 0 de ICW4

2- Configuration de l'ICW2 (21h pour le 1er 8259 et A1h pour le 2ème) :

L'ICW2 est là pour gérer le décalage du numéro de l'IRQ et de l'interruption associée dans la table des vecteurs d'interruption. Donc :

- pour le premier 8259 : ICW2 = 08h. De 08h à 0Fh.
- pour le deuxième 8259 : ICW2 = 70h. De 70h à 77h.

3- Configuration de l'ICW3 (Seulement si cascadiés) (21h ou A1h)

Utilisé uniquement quand plusieurs PICs sont cascadiés. Un PIC à 8 PICs → 64 Interruptions Hardware.

- ICW3 pour le maître : Bit correspondant à entrée à 1 pour esclave ; Supposant que bit 7 et bit 4 sont à '1' → Deux circuits esclaves utilisés connectés à IR7 et IR4
- ICW3 pour l'esclave : Bits 0,1,2 pour le numéro d'esclave ; Numéro de IR du PIC Maître qui est connecté vers ce PIC Esclave.

Exemple :

Initialisation du PIC8259 avec les données suivantes:

CPU8085, adresse 20h, Seul (pas de Cascade), IRx en Trigger Niveau, INT 08H sur IR0. Le PIC8259 étant en mode EOI normale.

ICW1 → 000 1 1 0 1 1 → 1BH (Besoin du ICW4, PIC Seul, IRx Trigger Niveau)

ICW2 → 0000 1000 → 08H (000 → IR0 → INT 08H)

ICW3 → Pas Besoin, un seul PIC est utilisé.

ICW4 → 0000 0000 → 00H

Voici le programme d'Initialisation du PIC8259

```

MVI A,1BH    ; ICW1
OUT 20H      ; Vers Port 20H
MVI A,08H    ; ICW2
OUT 21H      ; Vers Port 20H
MVI A,01H    ; ICW4
OUT 21H      ; Vers Port 21H

```

IV.2. Registres de commande

Après initialisation du 8259, peut accéder aux mots de commande OCW1-2-3

- OCW1 : IMR (Interrupt Mask Register), adresse : 21h/A1h. Chaque bit représente une interruption, 1 = masquée.
- OCW2 : Registre de commande de EOI, adresse : 20h/A0h. Pour y accéder il faut les bits 3 et 4 à 0. C'est ici qu'il faut signaler une EOI en écrivant 20h, ce qui aura pour effet d'effacer le bit correspondant dans l'ISR et donc de permettre toute interruption inférieure ou égale. Si les bits 2, 1 et 0 sont à 011 respectivement les niveaux de priorité sont : IR3, IR4, IR5, IR6, IR7, IR0, IR1, IR2.
- OCW3 : Permet de lire l'IRR (Interrupt Request Register) ou l'ISR (Interrupt Service Register) en positionnant les bits correspondants puis en relisant OCW3. Pour y accéder il faut que le bit 3 soit à 1 et le 4 à 0, toujours à l'adresse 20h/A0h

Exemple :

La figure montre le schéma d'un système utilisant le 8259. Quatre sources sont connectés aux lignes IR du 8259 ; Signal d'urgence, Clavier, Convertisseur A/N et Imprimante. Le signal d'urgence possède la plus haute priorité et l'imprimante la plus faible priorité.

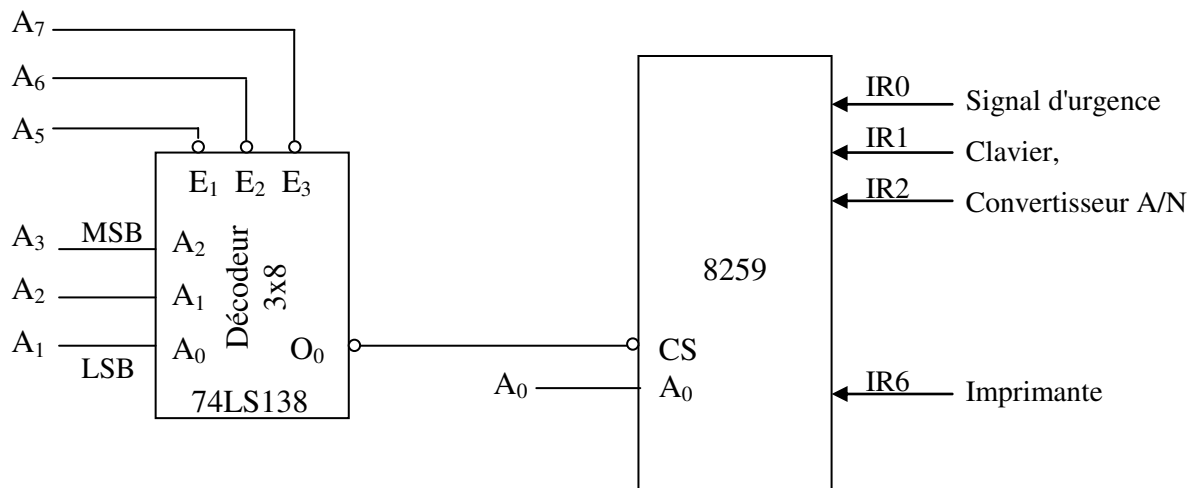


Figure VII.8 : Exemple d'application du 8259

Instructions d'initialisation

DI

MVI A,76H ; ICW1

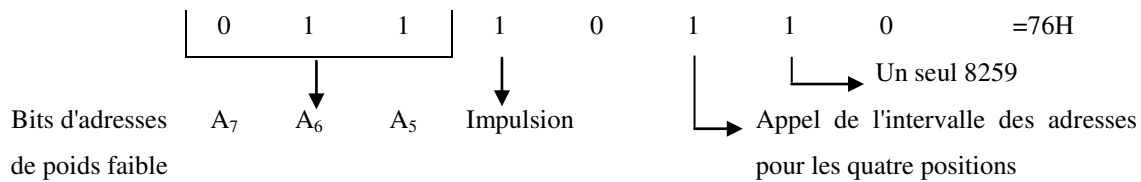
OUT 80H ; Initialisation du 8259

MVI A,20H ; ICW2

OUT 81H ; Initialisation du 8259

1. DI désactive les interruptions, ce qui implique que le processus d'initialisation ne sera pas interrompu.

2. Le mot de commande 76H spécifie les paramètres suivants :



L'octet de poids faible de l'adresse d'appel IR0 est :

A ₇	A ₆	A ₅	A ₄	A ₃	A ₂	A ₁	A ₀	
0	1	1	0	0	0	0	0	=60H

Les bits d'adresses A₄-A₀ sont fournies par le 8259. Les adresses résultantes sont quatre adresses.

3. L'adresse du port du 8259 pour ICW est 80H; A₀ doit être mise au niveau logique 0 et les autres bits sont déterminés par le décodeur.

4. Le mot de commande ICW2 est 20H, qui spécifie l'octet fort de l'adresse d'appel.

5. L'adresse du port du ICW2 est 81H ; A₀ doit être mise à 1.

BIBLIOGRAPHIE

- [1] P. Cabanis, "Electronique digitale", Dunod, Paris, 1985.
- [2] J. Letocha , "Introduction aux circuits logiques", Mc-Graw Hill, Canada, 1997.
- [3] 74LS194 Datasheet,
<http://www.alldatasheet.com/datasheet-pdf/pdf/51050/fairchild/74LS194.html>
- [4] M. Gindre, "Electronique numérique, Logique séquentielle, cours et exercices", Ediscience, 1994.
- [5] R.L. Tokheim, "Les microprocesseurs", Mc-Graw Hill, Canada, 1989.
- [6] M. Aumiaux, "L'emploi des microprocesseurs", Masson, Paris, 1982.
- [7] M. Aumiaux, "Les systèmes à microprocesseurs", Masson, Paris, 1982.
- [8] J.C. Buisson, "Concevoir son microprocesseur, structure des systèmes logiques", Ellipses, Paris, 2006.
- [9] H. Lilen, "Cours fondamental des microprocesseurs", Dunod, Paris, 1993.
- [10] J. Mbumba, "L'ordinateur et son évolution", L'informatique au service des archives, 2008.
- [11] H. Lilen, "Cours fondamental des microprocesseurs", Dunod, Paris, 1993.
- [12] A. Tanenbaum, "Architecture de l'ordinateur, cours et exercices", Dunod, Paris, 2011.
- [13] P. Zanella, Y. Ligier et E. Lazard, "Architecture et technologie des ordinateurs", Dunod, Paris, 2013.
- [14] J.C. Buisson, "Concevoir son microprocesseur : Structure des systèmes logiques", Ellipses, Paris, 2006.
- [15] J.M. Bernard et J. Hugon, "De la logique câblée aux microprocesseurs", Eyrolles, Paris, 1978.
- [16] R. S. Gaonkar, "Microprocessor Architecture, programming and applications with the 8085", Prentice Hall, New York, 1996.
- [17] M. Dalmau, "Cours microprocesseur", Université de Pau, France.
- [18] P. André, "La liaison RS232 : Description technique et mise œuvre", Dunod, Paris, 2002.