Cours et Travaux Dirigés sur :

Les microprocesseurs à 8 bits

Proposé par :

ABID Sabeur, Assistant à l'Ecole Supérieure des Sciences et Techniques de TUNIS (ESSTT).

Année Universitaire : 2002/2003

Chapitre 1 : Les systèmes à microprocesseur

Logique programmée

L'apparition des microprocesseurs sur le marché permet de concevoir tout autrement un système logique. Le microprocesseur est un circuit intégré complexe mais très évolué qui est capable de réaliser plusieurs fonctions logiques. Toutefois, puisque ce n'est plus un circuit destiné à réaliser une seule fonction logique, mais des dizaines de fonctions, il va donc falloir, à un instant donné, signaler au microprocesseur l'opération logique ou arithmétique à réaliser. Comment?

En envoyant au microprocesseur un mot binaire de 8 ou 16 bits par exemple, chacune des configurations de ce mot de plusieurs bits correspondant à *une* et *une seule* opération choisie. Ce mot binaire est une *instruction* puisqu'il est en effet chargé d'instruire le microprocesseur de la fonction à réaliser. Mais un système logique exige la réalisation d'une suite de nombreuses opérations. Aussi pour faire fonctionner notre système il faudra prévoir une suite ordonnée d'instructions et c'est ce qu'on appelle un *programme* ou pour plus de précision, dans notre cas, c'est le programme utilisateur. Ainsi, la partie correspondant au matériel (hardware) est conçue dans le boîtier du microprocesseur qui nécessite alors pour être commandé un programme, c'est-à-dire une suite d'instructions donc de mots binaires qui seront rangés dans une mémoire. Le microprocesseur ira chercher une instruction dans la mémoire, qui lui est extérieure, exécutera l'opération correspondante, puis ira chercher en mémoire la prochaine instruction du programme, l'exécutera, ira chercher une nouvelle instruction, l'exécutera et ainsi de suite. Une telle logique est dite *logique programmée* et elle correspond au *logiciel* (traduction de *software).* Le concepteur d'un système devra alors bien connaître le microprocesseur, son jeu d'instructions, c'est-à-dire l'ensemble des instructions différentes qu'il possède et fort de ces connaissances il pourra établir le programme correspondant à l'application envisagée.

Système à microprocesseur

Lorsqu'un système logique est conçu à partir d'un microprocesseur, il est appelé SYSTÈME A MICROPROCESSEUR. La conception et la réalisation d'un tel système comprennent deux parties bien distinctes :

• ***L'étude du « matériel »*** **:** quelques circuits intégrés LSI (Large Scale Integration = intégration à grande échelle soit 500 à 10 000 transistors dans un seul circuit LSI) constituent les circuits intégrés essentiels du système à microprocesseur. Ce sont

- le microprocesseur proprement dit.

- la mémoire RAM qui contiendra les données et les résultats définitifs ou provisoires.

- la mémoire ROM (à lecture seulement) qui contiendra la suite des instructions, c'est-à-dire le programme, qui permettront de faire fonctionner correctement le système à microprocesseur.

- l'interface parallèle programmable. Ce circuit LSI permet de connecter le microprocesseur aux circuits électroniques à commander et faisant partie du système logique, ces circuits électroniques acceptant des échanges de données du type parallèle (huit bits en même temps sur huit fils parallèles). Le système à microprocesseur peut comporter un ou. plusieurs exemplaires de cet interface parallèle programmable.

- l'interface série programmable. Ce circuit LSI permet un échange de données entre le microprocesseur et un a terminal à travers une ligne téléphonique donc à grande distance. Il permettra aussi lors de la mise au point ou de la maintenance du système d'assurer un dialogue entre l'utilisateur et le système par l'intermédiaire d'un organe de dialogue soit une télétype soit une console de visualisation à écran cathodique. Beaucoup de systèmes à microprocesseur ne comportant pas cet interface surtout destiné aux télécommunications.

• ***L'étude du « logiciel »***: par opposition au matériel le logiciel est l'ensemble des programmes nécessaires pour le bon fonctionnement du système à microprocesseur ainsi que tout ce qui concerne l'étude et la mise au point de ces programmes.

Le matériel (hardware) d'un système à microprocesseur : organisation d'un ordinateur

Les ordinateurs sont largement utilisés depuis les années cinquante. C'étaient, au début des machines volumineuses et chères réservées à l'administration et aux grandes entreprises. Au cours des dernières années, les tailles et les formes des ordinateurs se sont modifiées par suite de l'introduction d'un élément nouveau : le microprocesseur. Les ordinateurs fonctionnent avec un programme chargé; les Intercoordinateurs utilisent également le même concept. Un microordinateur comporte un microprocesseur et au moins un type quelconque de mémoire semi-conductrice.

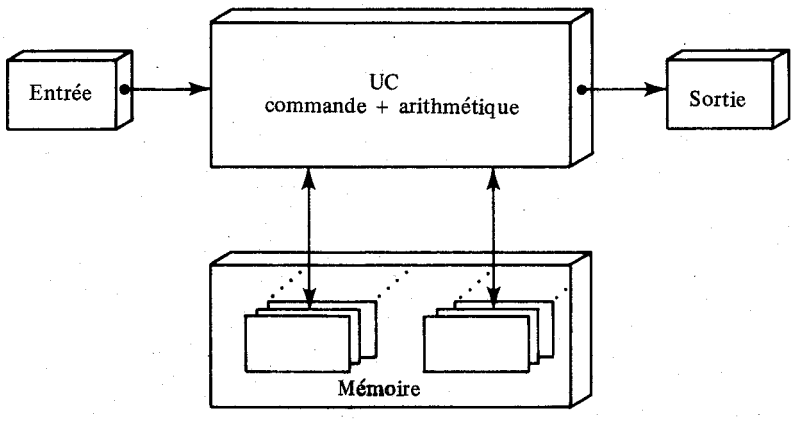


Fig.1.1 : Organisation générale d'un ordinateur

La figure 1.1. représente les parties traditionnelles d'un ordinateur. Cette organisation des éléments fonctionnels est bien souvent dite *architecture* de l'ordinateur. Le système le plus

élémentaire comprend cinq unités : l'unité d'entrée, les unités de contrôle et arithmétique (contenues dans l'UC ou unité centrale), l'unité de *mémoire* et l'unité de *sortie.*

Les unités physiques représentées sur la figure 1.1 constituent le matériel. Pour être utile, la mémoire *de programme* doit indiquer à l'UC ce qu'elle doit exécuter. La préparation d'une liste *d'instructions constitue* la programmation. La liste d'instructions ainsi définie est un programme qui est stocké temporairement ou de manière permanente dans la mémoire de programme. Ces programmes manipulent de l'information désignée sous le terme général de *données. Logiciel* est un terme général qui recouvre tous les programmes. Si le logiciel est stocké de manière permanente en mémoire de programme, il est parfois désigné comme micro-logiciel.

Un système de microordinateur est un ordinateur numérique. Il est classé dans les micro par suite de sa faible taille et de son bas prix. En général, le microprocesseur constitue la partie UC du système. La figure 1.2 représente l'architecture d'un microordinateur typique. Elle comprend les cinq éléments de base d'un ordinateur: (1)l'unité d'entrée, (2) l'unité de *contrôle* et (3) l'unité *arithmétique,* toutes deux comprises dans le microprocesseur, (4) l'unité de mémoire et, enfin, (6) l'unité de sortie.

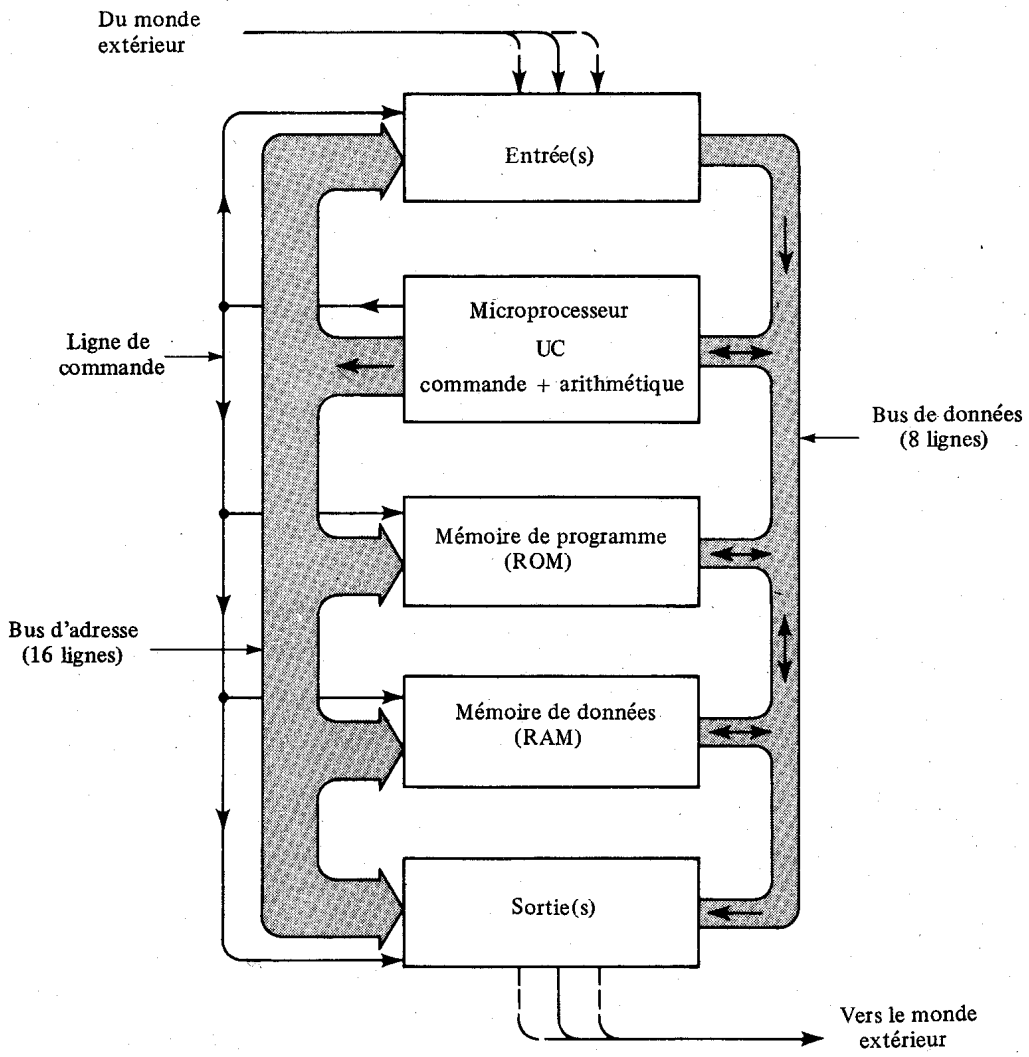


Fig.1.2 : Organisation typique de micro-ordinateur

Quand un programme est rangé de manière permanente, il est généralement placé dans une unité de mémoire dite *mémoire morte* (ROM pour *read-only memory).* La mémoire morte est, en général, une puce de CI programmée de manière permanente. La mémoire de données transitoires est, le plus souvent; constituée d'un circuit intégré (CI) appelé *mémoire vive* (RAM ou RWM pour *read/write memory,* le premier sigle étant le plus couramment utilisé). Les programmes de l'utilisateur qui sont de nature transitoire sont également rangés dans la partie RAM de la mémoire avec les données. Sur la figure 1.2, les RAM et ROM sont représentées distinctes parce qu'elles sont généralement constituées de CI différents.

Le logiciel (software) d'un système à microprocesseur

A. Le langage binaire (langage machine)

Pour donner des ordres au microprocesseur, il faut un langage binaire. Chaque opération logique ou arithmétique à effectuer est codée sur un mot binaire appelé *instruction* et l'ensemble des instructions nécessaires pour faire fonctionner tout ou partie du système comme le souhaite l'utilisateur est le programme. L'écriture de celui-ci en langage binaire est vite très fastidieuse, car elle conduit à de nombreuses suites de 0 et 1 et il est très facile de se tromper.

B. Le langage hexadécimal

Il serait plus correct de parler d'écriture hexadécimale des instructions binaires car c'est exactement le cas. Les instructions binaires, les adresses binaires sont exprimées dans le code hexadécimal ce qui est déjà beaucoup plus facile. Ainsi l'instruction 101111010011 s'exprimera en hexadécimal par BD3 expression qui prête beaucoup moins à erreur que son équivalent binaire. De même l'adresse la plus élevée dans le cas d'une mémoire de 26 octets s'écrira 1111111111111111 en binaire et FFFF en hexadécimal. Le code hexadécimal est très utilisé pour les microprocesseurs.

C. Le langage assembleur

L'hexadécimal est déjà beaucoup plus facile d'emploi que le binaire, mais les codes opérations des instructions ne sont pas significatifs. Ainsi, si l'addition binaire est représentée par l'expression 3F, celle-ci n'a rien qui peut faire songer à une addition comme le ferait par exemple l'expression mnémonique **ADD**, abréviation de addition. Aussi les informaticiens ont depuis longtemps représenté les différentes opérations logiques ou arithmétiques que peut effectuer un ordinateur par des expressions mnémoniques d'autant plus faciles à retenir qu'elles font penser aux opérations qu'elles représentent. Toutefois ces expressions mnémoniques sont presque toujours tirées du vocabulaire anglo-saxon. Ainsi une soustraction pourra être désignée par **SUB** (Substract), un chargement de registre sera désigné par **LD** (load), un décalage par **SHT** (shift), un saut par **JP** (jump), etc. Le programmeur utilisera donc l'ensemble de ces symboles mnémoniques pour constituer son programme. Nous dirons que le programme est en langage d'assemblage ou en assembleur. Mais comme le microprocesseur ne connaît que le binaire, il faudra traduire chaque expression mnémonique en langage binaire. Cette traduction sera faite par un programme spécial appelé **assembleur** et qui devra donc être mis en mémoire pour exécuter la traduction.

L'assembleur est un programme spécifique du microprocesseur. Il sera donc fourni par le constructeur.

D. Langages évolués

L'assembleur est fort utile mais il a l'inconvénient de n'être pas un langage universel puisque le programme assembleur pour le microprocesseur 6800 de Motorola sera différent du

programme assembleur spécifique du microprocesseur 8080A de Intel. Aussi les informaticiens ont créé des langages universels et qui, en même temps, sont plus puissants au point de vue des instructions : ainsi une instruction en langage évolué (c'est le nom réservé pour un langage universel) peut nécessiter plusieurs instructions binaires successives. Ces langages évolués sont donc des outils très puissants des ordinateurs, mais accessibles à l'utilisateur de microprocesseurs même si ce n'est pas toujours intéressant pour un système à microprocesseur. Comme langage évolué, il y a le FORTRAN spécifique des calculs et des algorithmes mathématiques, le COBOL bien adapté à l'informatique de gestion, l'ALGOL pour les mathématiques, le PROCOL langage nouveau spécifique des contrôles de processus, le PLI langage récemment créé par IBM et le PLM langage également très récent.

Là aussi un programme écrit en langage évolué devra être traduit en langage machine. C'est encore un programme conçu par le constructeur qui fera cette traduction. Il sera appelé compilateur. Ainsi le constructeur proposera un compilateur FORTRAN, un compilateur COBOL, un compilateur PLM, etc.

Les kits d'évaluation

Les constructeurs de microprocesseurs proposent également des cartes circuits imprimés avec le processeur, une RAM, une PROM et un petit programme moniteur pour permettre aux débutants de se familiariser avec le microprocesseur d'une part et pour lui faciliter la mise au point de ses programmes d'autre part.

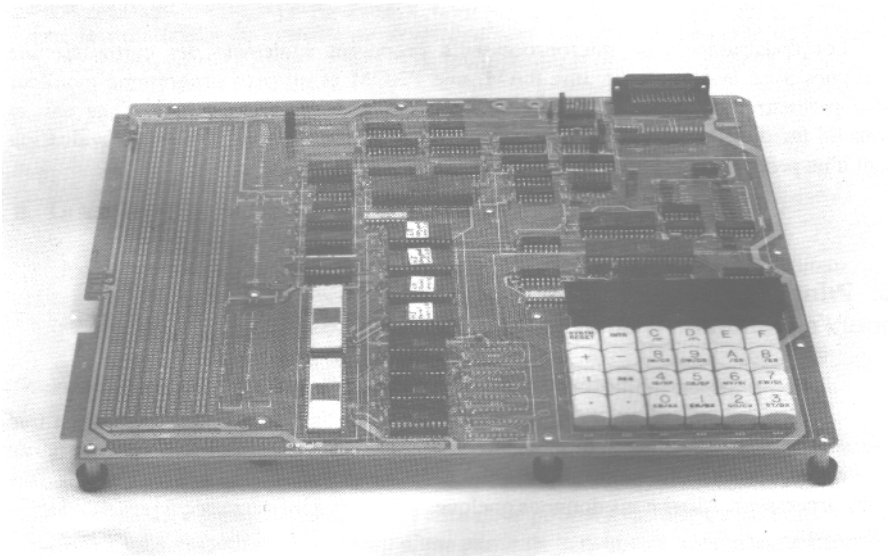


Fig.1.3 : Kit SDK 86 (Document INTEL)

Chapitre 2 : Nombres, codage et arithmétique informatique

1. Codes

A. Code binaire naturel

Lorsque nous écrivons 138 c'est l'écriture condensée de $1.10^2 + 3.10^1 + 8.10°$ où $10° = 10^1$ et 10^2 valent respectivement 1, 10 et 100 soit les puissances successives de la base 10 (unité, dizaine, centaine). Les coefficients qui multiplient les puissances de 10 peuvent prendre toutes les valeurs entières inférieures à la base donc 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 : c'est un *système de numération décimal* puisque la base vaut 10.

Le système binaire est un système de numération de base égale à 2. Les coefficients ne peuvent donc être que 0 ou 1. Les puissances successives de la base 2 sont $2°$, 2^1, 2^2, 2^3, 2^4, etc... soit 1, 2, 4, 8, 16, etc. Comme pour le système décimal nous pouvons utiliser une écriture condensée en ne retenant que les coefficients des puissances de 2. Ainsi le nombre binaire 1 0 0 0 1 0 1 0 vaut

$1 .2^7 + 0.2^6 + 0.2^5 + 0.2^4 + 1 .2^3 + 0.2^2 + 1 .2^1 + 0.2°$

$1.128 + 1 .8 + 1.2$

soit 138

Nous retrouvons notre nombre 138, mais il nous a fallu beaucoup plus de coefficients. Nous pouvons calculer le nombre binaire pour chacun des chiffres décimaux de 0 à 9. Pour cela quatre puissances de 2 nous sont nécessaires. Nous dirons qu'un chiffre décimal est exprimé par quatre *moments binaires ou* quatre bits. Ainsi le chiffre 4 sera représenté par 0100 soit $0.2^3 + 1.2^1 + 0.2^1 + 0.2°$ ce qui donne bien 4. Ce système de représentation des nombres est le *code binaire naturel.*

Avec n bits il est possible de représenter 2^n nombres, le plus grand de ces nombres valant $2^n - 1$. Le système binaire est celui utilisé dans les ordinateurs, les calculateurs ou les microprocesseurs.

B. Code BCD ou DCB

Pour exprimer notre nombre 138 il est également possible de coder en binaire sur 4 bits chacun des trois chiffres décimaux 1, 3, 8. Nous obtenons alors

138 = 0001 0011 1000

1 3 8

Ce système de codification est appelé en terme anglo-saxon BCD *(Binary Coded Decimal) ou,* en français, DCB *(Décimal Codé Binaire).*

Ce code est également utilisé dans les microprocesseurs.

C. Code ASCII

Il est nécessaire en informatique de pouvoir coder les nombres mais aussi les lettres de l'alphabet et certains symboles. Deux codes sont fréquemment utilisés : le code ASCII et le code ISO. Dans ces deux codes, les informations sont exprimées avec 8 bits. Ainsi lorsqu'on appuie sur l'une des touches du clavier d'une télétype, la sortie de la télétype donne la codification sur 8 bits en ASCII du caractère (nombre, lettre ou symbole) correspondant à la touche appuyée (7 bits + bit de parité).

Voir tableau ASCII ci-dessous.

CODE ASCII

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

La table ci-dessus donne l'expression hexa-décimale du code ASCII sur 7 bits (bit de parité = bit 7 = zéro) pour les chiffres, les lettres et les symbôles.

Exemple : 5 a pour expression héxadécimale de son code ASCII : 35 (colonne 3 ligne 5) ce qui veut dire que le code ASCII de 5 est 00110101.

Inversement le code ASCII 01010100 a pour expression hexadécimale 54 : c'est donc le code ASCII de la lettre T (colonne 5, ligne 4).

D. Code hexadécimal

Les codes binaire et BCD sont utilisés dans les ordinateurs pour faire des calculs. Les codes ASCII ne servent, eux, qu'à représenter les nombres, les lettres et les symboles en vue de les imprimer ou de les visualiser. Tous ces caractères codés en binaire, en BCD ou en ASCII sont rangés dans une mémoire qui n'est autre qu'une juxtaposition de cases mémoires, chacune

d'elles contenant un bit. L'ensemble de huit cases mémoires est une position mémoire car elle permet de loger un mot binaire, c'est à dire un caractère de 8 bits, tout au moins pour les microprocesseurs classiques à 8 bits. En cours d'utilisation il y aura dans la mémoire une quantité de caractères binaires. Lorsque l'on veut exprimer ces suites de bits, il devient fastidieux de les représenter comme ils sont réellement dans la mémoire c'est 1à dire en binaire. Aussi on utilise une représentation particulière plus facile, c'est le *code hexadécimal* qui fait correspondre à chaque groupe de 4 bits un nombre ou une lettre A, B, C, D, E ou F.

N	2^3	2^2	2^1	2^0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

. – *Code binaire naturel*

	Code hexa
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

2. – *Code hexadécimal*

Ainsi l'expression hexadécimale 2FA représente la suite binaire

0010 1111 1010 soit 001011111010

2 F A

Le code hexadécimal n'est qu'une commodité d'écriture, mais il est fondamental pour l'utilisation des microprocesseurs.

2. Fonctions logiques et opérateurs

Un interrupteur électrique ne peut être que dans l'un ou l'autre des deux états suivants : ouvert ou fermé. De même pour un relais électromagnétique. L'interrupteur, le relais peuvent constituer des *variables binaires,* c'est à dire ne possédant que deux états possibles. En informatique les variables binaires doivent pouvoir passer très rapidement d'un état à l'autre. Aussi on utilisera des transistors à jonction (bloqués ou saturés), des transistors MOS (conducteurs ou bloqués), des tores magnétiques (aimantés ou non), etc.

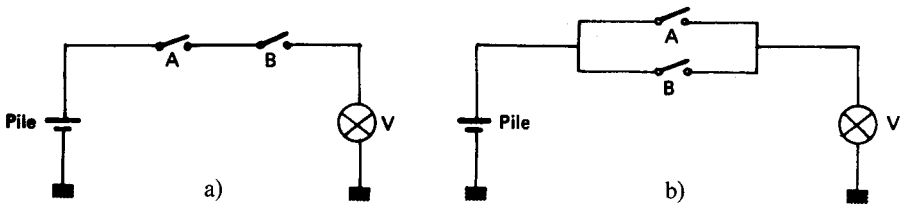


Fig. 2.1 : Fonctions logiques de base

Constituons un petit circuit d'éclairage avec une pile, deux interrupteurs A et B et un voyant V. Nous pouvons considérer A, B et V comme des variables binaires, le voyant étant allumé ou éteint, donner arbitrairement la valeur logique 1 aux interrupteurs quand ils sont fermés et 0 au voyant quand il est allumé. Pour la figure 2.1.a) nous pouvons dire :

le voyant V est allumé si A ET B sont fermés

ou

V a la valeur 1si A ET B ont la valeur 1.

ce que nous écrivons symboliquement V = A . B, le point symbolise la Fonction **ET**.

Pour la figure 2.1.b) nous pouvons dire :

le voyant V est allumé si A OU B est fermé ou si les deux sont fermés

soit

V a la valeur 1 si A OU B a la valeur 1 ou si les deux ont la valeur 1

ce que nous écrivons symboliquement V = A + B ou encore A V B; le signe + ou le symbole V caractérise la FONCTION OU. Les fonctions logiques ET, OU sont les deux fonctions de base.

Nous donnons ci-dessous les dessins symboliques des circuits réalisant les différentes fonctions logiques, circuits appelés *opérateurs logiques* et qui utilisent des transistors en général.

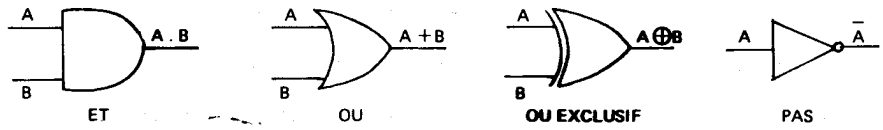


Fig. 2.2 : - Symboles des opérateurs logiques

3. Bascules, registres, compteurs, décodeurs

• *Une bascule logique* est un élément permettant de mémoriser la valeur d'une variable binaire (logique). Elle est constituée à l'aide de plusieurs transistors connectés entre eux de façon à conserver l'état qui a été fourni à l'entrée à un moment donné. La valeur mémorisée est disponible en permanence à la sortie de la bascule.

• *Un registre* est une juxtaposition de bascules permettant de décaler les valeurs mémorisées, c'est à dire de les faire passer sur commande d'une case à la suivante. Là encore les bits mémorisés sont disponibles en permanence aux sorties du registre. Plusieurs décalages successifs permettent de sortir tous les bits du registre pour les envoyer dans un autre circuit. Plusieurs types de registres existent selon que l'information à mémoriser est rentrée bit par bit

(entrée série) ou tous les bits en même temps (entrée parallèle); de même pour la sortie (figure 2.3).

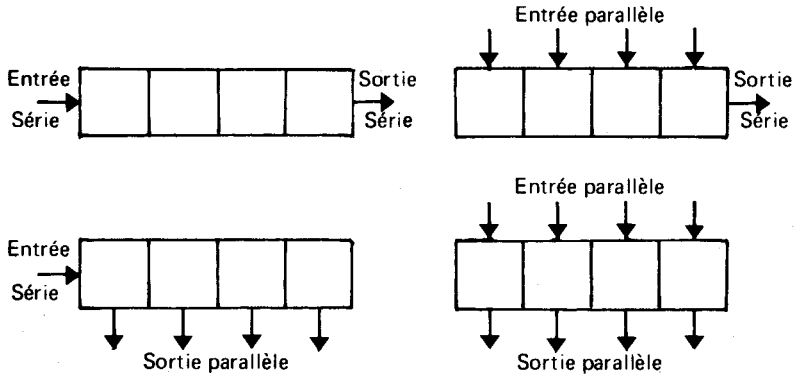


Fig. 2.3 : Registres de 4 bits

• *Un compteur* est un circuit comprenant *n* cases binaires, et comptant en binaire le nombre d'impulsions logiques qu'il reçoit sur son entrée. Ce nombre binaire mémorisé est disponible en permanence à la sortie parallèle du compteur. C'est l'équivalent d'un compteur kilométrique de voiture mais le nombre enregistré dans le compteur est codé en binaire et n'est pas visualisé.

• *Un décodeur* est un circuit qui établit une relation unique (ou plus exactement univoque) entre l'un des fils de sortie du décodeur et l'un des nombres binaires codés entre *n* fils d'entrée. Ainsi avec 4 fils d'entrée, un décodeur devra posséder 2^4 fils de sortie soit 16. Ainsi au nombre binaire 0101 correspondra la sortie S_5, au nombre binaire 1100 correspondra la sortie S_{12}, etc.

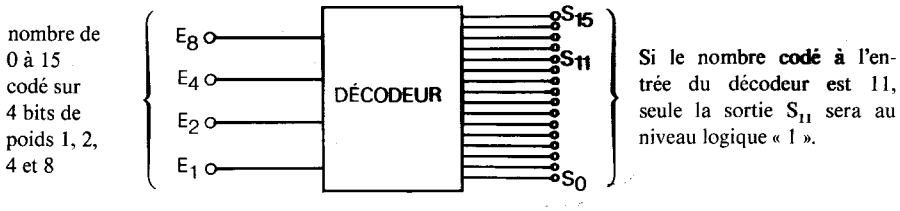


Fig. 2.4 : Décodeur

4. Logique trois états et bus

La logique que nous venons de voir a deux états 0 et 1. Une logique particulière dite à trois états possède un état de plus, qui est l'état *haute impédance,* d'où

• état 0 = niveau 0 volt à basse impédance;

• état 1 = niveau 5 volts à basse impédance;

• état haute impédance = aucun niveau de tension et haute impédance.

Cette logique est très intéressante pour la constitution d'un bus qui est un groupe de fils véhiculant de l'information d'un point à un autre.

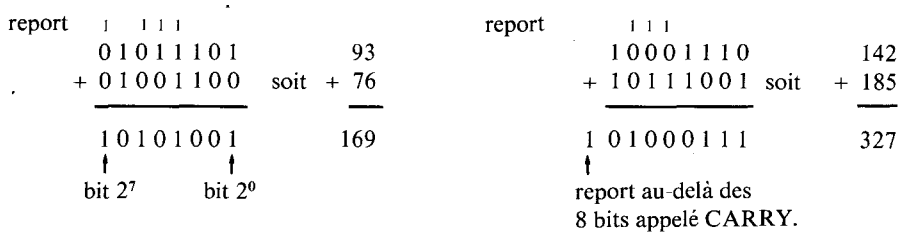
5. Arithmétique binaire

Tout microprocesseur permet d'effectuer une addition ou une soustraction de nombres binaires.

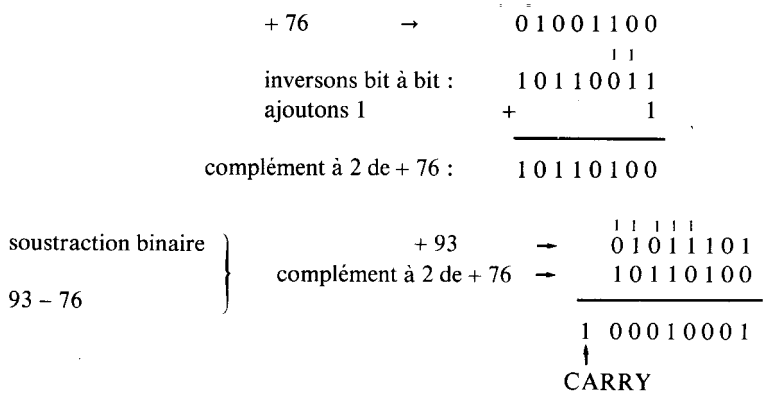
* *Addition binaire :* Le principe est le même que pour une addition dans le système décimal c'est à dire que l'addition de deux bits (ou de trois bits lorsqu'il y a un report précédent à prendre en compte) entraînera un report dès que le résultat dépassera 1. D'où :

0 + 0 = 0; 1 + 0 = 1; 0 + 1= 1. Aucun report pour ces trois cas.

1 + 1 = 0 et report = 1; 1 + 1 + 0 = 0 et report = 1. 1+1+1=l et report =1. En effet 1+1+1=3 soit11.



* *Soustraction binaire. soit* à effectuer la soustraction 93 - 76. Pour cela les nombres doivent être signés : le bit 2^7 prend la valeur 0 pour un nombre positif et la valeur 1 pour un nombre négatif. La soustraction s'effectue en fait en additionnant à 93 le « complément à 2 » de + 76. Ce complément à 2 est obtenu en inversant bit à bit l'expression binaire de + 76 et en y ajoutant 1 :



On ne tient pas compte du CARRY de sorte que le résultat est 00010001 soit 17 ce qui est bien la différence 93 - 76.

6. Mémoire

Une mémoire est une juxtaposition de cases-mémoires ou plus exactement de positions mémoires, c'est à dire d'ensembles de cases mémoires. Une mémoire contenant des mots de 8 bits (on les appelle des octets) sera une juxtaposition de positions mémoires de 8 bits. Pour sélectionner l'une de ces positions mémoires, il faudra connaître son implantation c'est à dire son *adresse.* Cette sélection se fera à partir d'un compteur et d'un décodeur; le compteur étant généralement externe à la mémoire. Le mot binaire à lire ou à écrire transitera par un registre d'entrée/sortie. Une mémoire ayant 16 bits d'adresse pourra contenir 2^{16} mots binaires soit 65536. En général on exprimera la capacité de la mémoire en kilomots, ou kilooctets pour des mots de 8 bits, ou plus simplement en K octets, le nombre K valant 1 024 (2^{10}). Donc avec 16 bits d'adresse la mémoire aura une capacité de 64 K octets. Une mémoire qui peut être lue ou

écrite est une mémoire lecture/écriture ou *mémoire vive ou RAM* (Random Access Memory). Une mémoire qui ne peut être que lue est une mémoire à lecture seulement, ou *mémoire morte ou ROM* (Read Only Memory) figure 2.5.

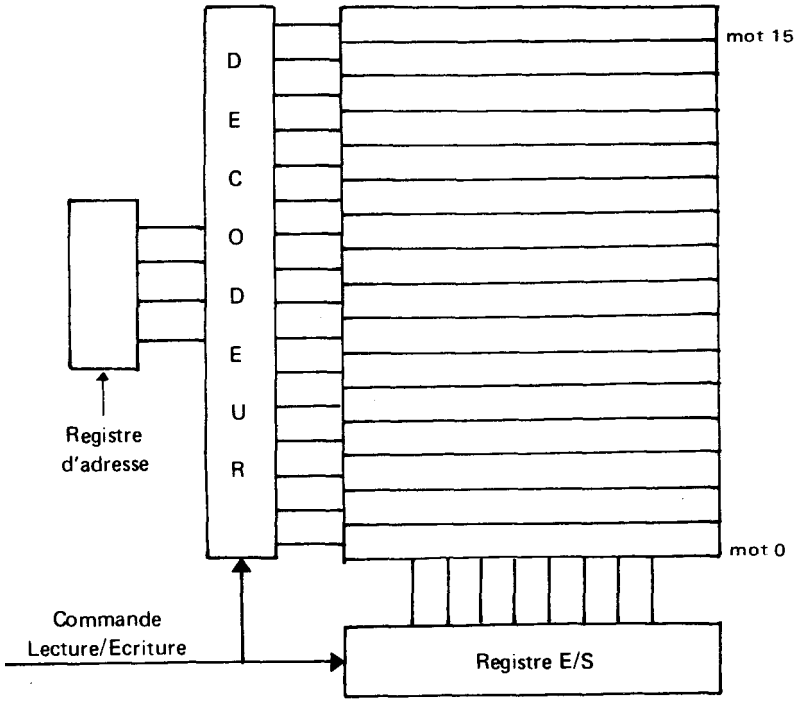


Fig. 2.5 : Mémoire de 16 mot

7. Applications 1

Ex.1 : Dans le cas de binaires, bit signifie ……..

Ex. 2 : Convertir en décimaux les binaires suivants

(a) 10000000 (c) 00110011 (e) 00011111

(b) 00010000 (d) 01100100 (f) 11111111

Ex.3 : Convertir en binaires les décimaux suivants

(a) 23 (b) 39 (c) 55 (d) 48

Ex.4 : 204_{10} =……..……$_2$

Ex.5 : 11101110_2 =……$_{10}$

Ex.6 : Ecrire les hexa suivants sous formé binaire

(a) C (c) F (e) 1A (g) A0 (i) 45

(b) 6 (d) E2 (f) 3D (h) 8B (j) D7

Ex.7 : Convertir les binaires suivants en hexa

(a) 1001 (c) 1101 (e) 10000000 (g) 00010101

(b) 1100 (d) 1111 (f) 01111110 (h) 11011011

Ex.8 : Convertir les hexas suivants en décimaux

(a) 7E (b) DB (c) 12A3 (d) 34CF

Ex.9 : 217_{10} =……$_{16}$

Ex.10 : $48\ 373_{10}$ =……$_{16}$

Ex.11 : Effectuer les additions binaires suivantes

(a) 1010	(b) 1101	(c) 01011011	(d) 00111111
+0101	+0101	+ 00001111	+00011111

Ex.12 : Effectuer les soustractions binaires suivantes

(a)1110	(b) 1010	(c) 01100110	(d) 01111000
- 1000	- 0101	- 00011010	- 00111111

Ex.13 : Indiquer si les nombres sous forme de complément à 2 qui suivent sont positifs ou négatifs

(a) 01110000 (b) 11001111 (c) 10001111 (d) 01010101

Ex.14 : Exprimer 1978 en binaire pur, en BCD et en ASCII.

Ex.15 : Sachant que N_1= 10010011 et N_2 =01111010 calculer les fonctions logiques N_1+ N_2, N_1.N_2, $\overline{N_2}$ et exprimer les résultats en hexadécimal.

Ex.16 : Effectuer l'addition 54 + 39 en binaire et exprimer le résultat en hexadécimal.

Ex.17 : Effectuer l'addition 54 + 39 en BCD.

Ex.18 :Effectuer la soustraction 54 - 39 en binaire.

Chapitre 3 : Architecture et fonctionnement d'un microprocesseur

8. Définitions

Le microprocesseur est une conséquence de la miniaturisation des circuits intégrés c'est un circuit LSI (Large Scale Integration), c'est à dire résultant d'une intégration à très grande échelle de plusieurs milliers de transistors sur un carré de quelques millimètres de côté et appelé « puce» (wafer en anglais). La technologie MOS, qui se prête à une forte densité d'intégration, a permis d'intégrer sur une seule puce, l'unité centrale d'un ordinateur, appelée aussi processeur d'où le nom donné à ce nouveau composant : MICROPROCESSEUR.

Comme l'unité centrale, le microprocesseur ne peut rien faire seul ; il a besoin de mémoires et d'interfaces d'entrée-sortie. Les mémoires vives à semi-conducteurs, résultant également de la technologie LSI et appelées RAM, sont volatiles contrairement aux mémoires à tores très utilisées autrefois avec les ordinateurs. Cela signifie que l'information mémorisée est perdue dès que l'alimentation de la mémoire est coupée. Aussi les programmes seront stockés non dans une RAM mais dans une ROM qui est une mémoire dans laquelle les instructions sont écrites une fois pour toutes, ineffaçables même si cette mémoire n'est pas alimentée. Ainsi il faut associer au microprocesseur, une mémoire ROM qui contiendra les programmes, une mémoire RAM qui contiendra les opérandes désignés d'une façon générale par « données », et des circuits d'interface permettant de connecter des organes électroniques ou électriques d'entrée-sortie au microprocesseur L'ensemble microprocesseur, RAM, ROM, circuits d'interface, constitue un ordinateur minuscule appelé MICROORDINATEUR. Ce dernier, associé aux organes d'entrée-sortie et éventuellement à d'autres circuits logiques nécessaires à l'application industrielle constitue un SYSTÈME A MICROPROCESSEUR. Le premier microprocesseur introduit sur le marché a été le 4004. Le 4004 est un microprocesseur 4 bits c'est à dire que ses opérations portent sur des opérandes de 4 bits, ce qui entraîne un bus de données de 4 bits, des mémoires RAM et ROM avec des mots de 4 bits. En 1972, la même société, Intel, proposa le premier microprocesseur, 8 bits, le 8008. Devant son succès, de nombreuses sociétés étudièrent un nouveau microprocesseur en se basant sur les avantages et les inconvénients du 8008. C'est ainsi qu'en 1974 furent proposés le 8080 par Intel, le 6800 par Motorola, le 2650 par Signétics, le PPS8 par Rockwell et d'autres encore. Chaque constructeur a ensuite amélioré son microprocesseur (ce qui a donné notamment le 8085 A

d'Intel, et le 6802 de Motorola) et conçu un micro-système, c'est à dire un microordinateur sur un seul circuit, par exemple le 8048 de Intel, le 6801 de Motorola, le 3870 de Fairchild.

Nous étudierons dans ce chapitre le matériel et le logiciel d'un microprocesseur ainsi que son fonctionnement. Bien qu'ils soient relatifs aux microprocesseurs 8 bits, les principes développés ici s'appliquent également aux microprocesseurs 16 bits.

9. Le « matériel » (architecture)

2.1.1 Architecture standard d'un microprocesseur

Un microprocesseur comprend essentiellement :

• *Une unité arithmétique et logique* (UAL en français, ALU en anglais) qui effectue les différents traitements : opérations arithmétiques et logiques, tests.

• *Une unité de commande* qui va chercher dans la mémoire chaque instruction à exécuter, la décode et génère en conséquence tous les signaux nécessaires pour l'exécution correcte de l'instruction. L'utilisateur d'un microprocesseur n'a pas à s'en soucier.

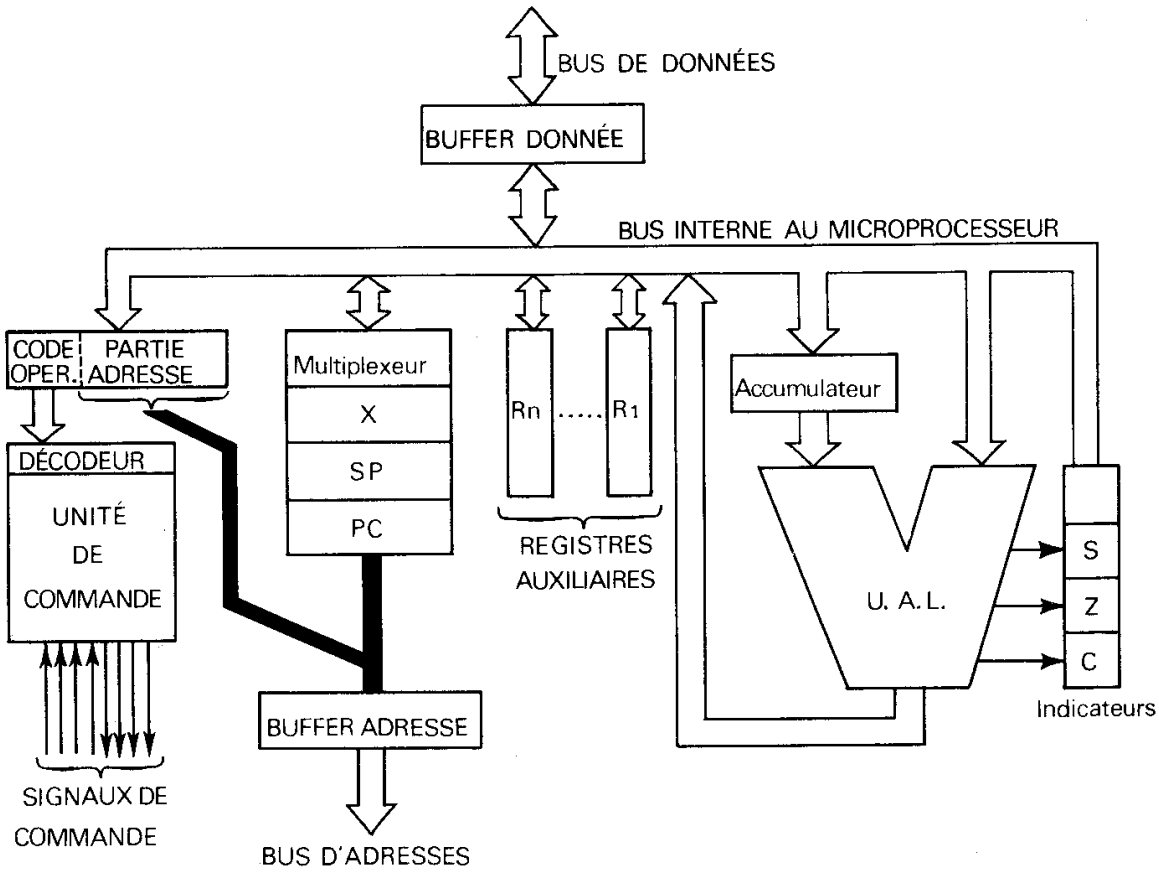


Fig. 3.1 : Architecture standard d'un microprocesseur

• *Des registres,* souvent utilisés pour des stockages provisoires d'informations. Certains ont une tâche bien précise, comme le compteur ordinal. Nous n'étudierons que les registres accessibles à l'utilisateur par la programmation.

2.1.1.1.1.1 A. L'accumulateur

Ce registre de 8 bits

• servira pour toutes les opérations arithmétiques et la plupart des opérations logiques. Il contiendra un opérande au début de l'opération puis le résultat à la fin de l'opération ;

• recevra les données en provenance de l'extérieur, ces données étant ensuite reprises de l'accumulateur pour être rangées en mémoire ;

• recevra en général les données en provenance de la mémoire et destinées à être dirigées vers un périphérique. Autrement dit, tout transfert E/S nécessitera un transfert Accumulateur/Port du périphérique concerné ;

• pourra servir de compteur ou de décompteur car il peut être incrémenté ou décrémenté par des instructions.

Certains microprocesseurs comme le 6800 et le 6502 ont deux accumulateurs.

2.1.1.1.1.2 B. Les registres auxiliaires

Ils sont plus ou moins nombreux selon le microprocesseur. Les microprocesseurs Intel 8080 A et 8085 A se caractérisent par de nombreux registres de 8 ou 16 bits.

2.1.1.1.1.3 C. Le compteur ordinal

Le compteur ordinal contient l'adresse de la prochaine instruction à exécuter. Son contenu est donc envoyé au registre adresse de la mémoire par l'intermédiaire du bus adresses. C'est un compteur car les instructions sont normalement rangées dans leur ordre d'exécution. L'exécution d'une instruction incrémente automatiquement le compteur ordinal désigné dans la littérature anglo-saxonne par *program counter* et en abréviation P.C. Toutefois, il est possible, dans certains cas, d'aller se brancher directement à la première adresse d'un sous-programme ou à une adresse bien définie (cas d'une boucle). Pour ce faire, la nouvelle adresse devra être chargée dans le compteur ordinal, soit par programme, soit automatiquement selon le type d'instruction et le type de microprocesseur.

2.1.1.1.1.4 D. Le registre instruction

Toutes les instructions du programme sont en mémoire RAM ou ROM. Le processeur ira donc chercher l'instruction en mémoire pour la stocker dans un registre à partir duquel les instructions seront décodées (c'est le cycle recherche nécessaire pour toute instruction). Ce registre est appelé *registre instruction.* Le décodage permettra de reconnaître une addition, une soustraction, un décalage, etc., et des signaux de commande seront activés en conséquence pour permettre l'exécution de l'instruction. Si besoin est, le microprocesseur retournera dans la mémoire pour chercher un opérande.

2.1.1.1.1.5 E. Le registre index

Ce registre permet l'adressage indexé. Il est destiné à recevoir une adresse aussi c'est un registre de 16 bits. Certains microprocesseurs peuvent avoir plusieurs index.

F. Le registre des indicateurs (appelé aussi registre de conditions)

Ce registre rassemble des bits d'état ou indicateurs, qui sont de précieuses informations résultant de l'exécution d'une instruction. L'utilisation de ces indicateurs relève du logiciel ; aussi nous les étudierons avec le logiciel d'un microprocesseur.

2.1.1.1.1.6 G. Le pointeur de pile

La pile est une petite partie de la mémoire RAM utilisée pour sauvegarder les contenus des registres lorsque cela est nécessaire, par exemple lors d'un appel de sous programme ou d'une interruption. L'adresse de la prochaine position mémoire à utiliser pour

la sauvegarde de données est à tout moment fournie par le contenu d'un registre appelé « pointeur de pile > (Stack Pointer en anglais). Nous étudierons en détail le fonctionnement de la pile quand nous aborderons le logiciel.

2.1.1.1.1.7 H. Les bus

Le bus de données est de 8 bits et le bus d'adresses de 16 bits. Les signaux de commande sont quelquefois rassemblés sous le nom de bus de commande. Le bus d'adresses reçoit le contenu du compteur ordinal qui est une adresse. Celle-ci est propagée par le bus à la ROM ou à la RAM ou à un organe d'entrée-sortie.

I. Les signaux d'entrée-sortie d'un microprocesseur

Nous les avons rassemblé sur la figure 3.2.

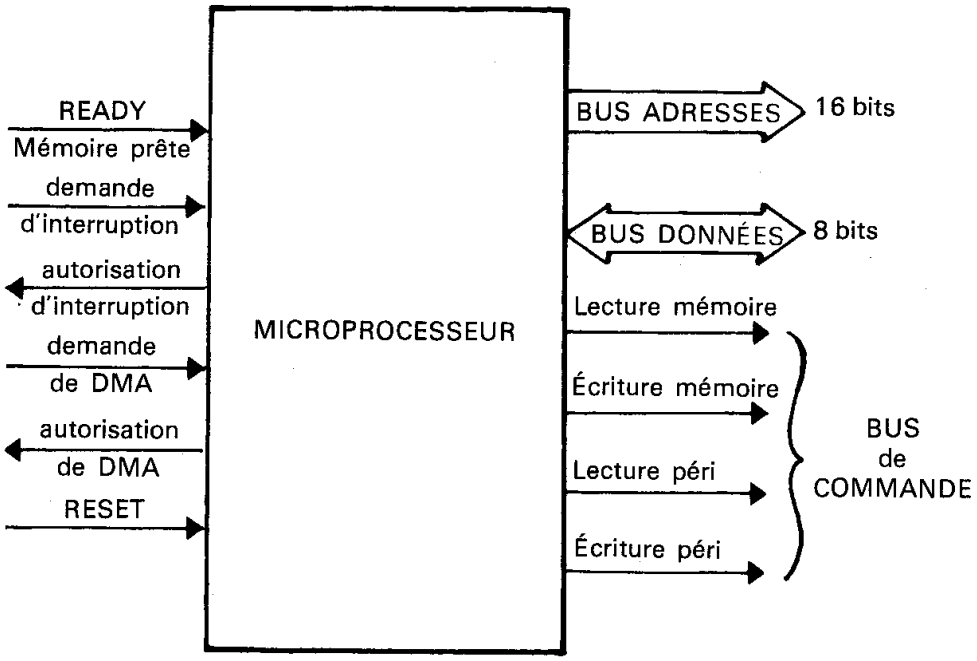


Fig. 3.2 : Les signaux usuels d'un microprocesseur

Bien entendu ces signaux porteront des appellations différentes d'un microprocesseur à un autre, mais ils se retrouveront presque toujours. Les quatre signaux de lecture et d'écriture seront souvent codés sur deux bits pour laisser disponible le maximum de broches. L'entrée RESET initialise le microprocesseur, c'est à dire remet généralement à zéro le contenu des registres ; le compteur ordinal, quant à lui, se trouve généralement chargé à une valeur fixée par le constructeur du microprocesseur, par exemple 0000 en hexadécimal pour le 8080 A et le 8085 A, de Intel.

2.1.2 Fonctionnement d'un microprocesseur

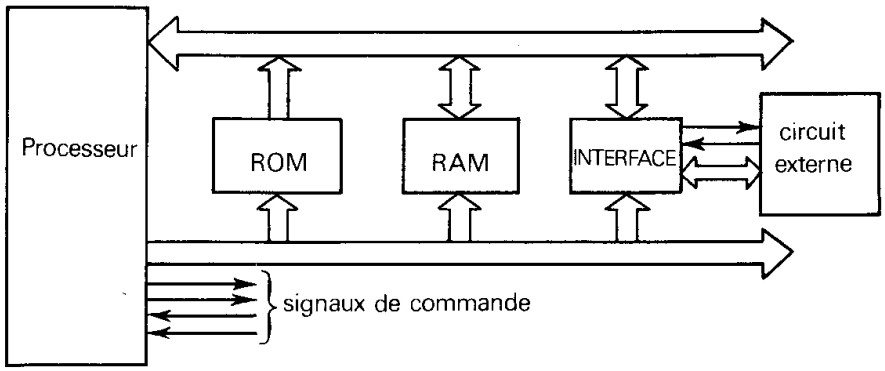
2.1.2.1.1.1 A. Phases recherche et exécution

Prenons l'instruction ADD R1. Elle signifie : ajouter le contenu du registre R1 à celui de l'accumulateur et transférer le résultat dans l'accumulateur. Cette instruction est codée sur un seul octet car elle ne comporte pas d'adresse. Supposons cette instruction rangée à l'adresse 1000 de la mémoire et supposons encore que le contenu du compteur ordinal soit 1000. Comment sera exécutée l'instruction? En deux phases.

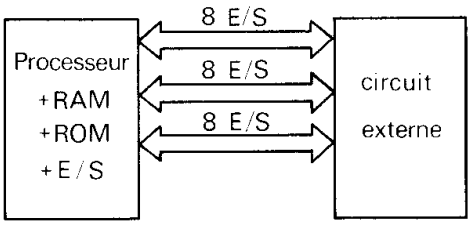
• *Phase recherche (ou* phase adressage). - Pendant cette phase le microprocesseur va chercher l'instruction en mémoire et l'amène dans le registre instruction où le code opération est décodé (ce code opération est le premier octet de l'instruction ainsi codée sur 8 bits).

• *Phase exécution. - Pendant* cette phase le microprocesseur exécute l'instruction.

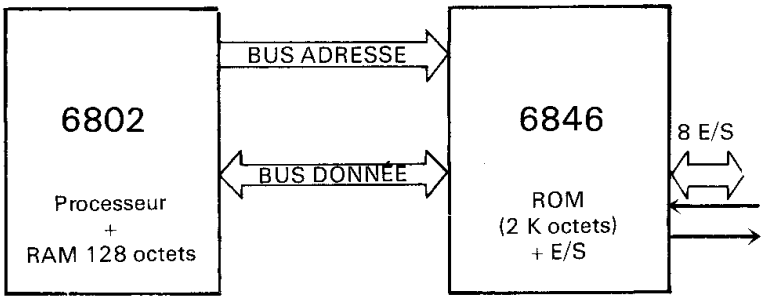
Les différentes architectures sont résumées dans la figure 3.3.



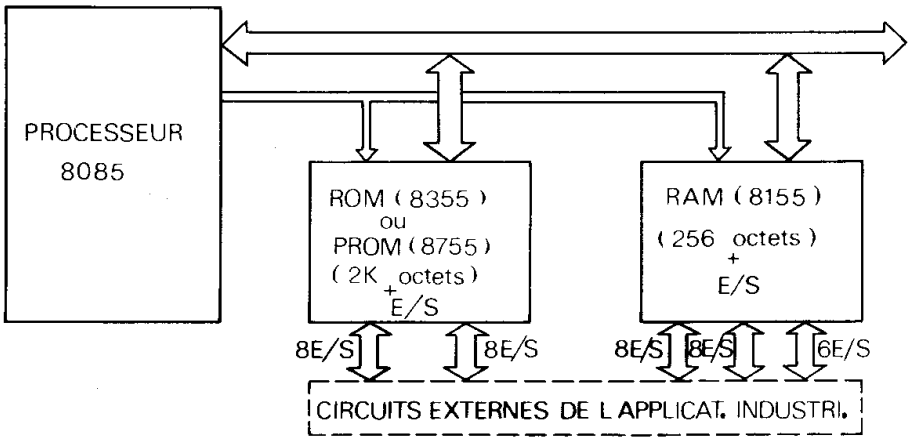
a) *Architecture standard d'un système à microprocesseur*



b) *Architecture du type « microsystème ».*



c) *Architecture à « E/S réparties » : 6802 et 6846*



d) *Architecture à « E/S réparties » : famille 8085 A*

2.1.2.2 Fig. 3.3 : Architectures des microprocesseurs

2.1.2.2.1.1 B. Fonctionnement de la pile

Un transfert E/S vers l'imprimante nécessite plusieurs instructions successives. Or un tel transfert est fréquent. Plutôt que de répéter à chaque transfert la suite des instructions nécessaires au transfert, nous appelons sous-programme cette succession d'instructions et nous lui donnons un nom arbitraire spécifique du transfert par exemple IMPRI. A chaque fois que nous souhaiterons faire un transfert vers l'imprimante, nous appellerons ce sous programme par une instruction spéciale désignée en langage mnémonique par CALL et suivie du nom du sous programme, exemple CALL IMPRI.

Pour cela le programme principal sera arrêté et le compteur ordinal sera chargé à l'adresse de la première instruction du sous-programme. Mais quand ce dernier sera exécuté le programme principal devra continuer, ce qui implique que l'adresse de la dernière instruction du programme principal ait été mémorisée quelque part. En fait c'est l'adresse de la prochaine instruction à exécuter du programme principal qui sera mémorisée dans une partie de la RAM appelée *pile.* Celle-ci fonctionne généralement de la façon suivante : la dernière information rentrée dans la pile est la première sortie (pile LIFO Last in first out). Mais les transferts ne sont pas les seuls à justifier un sous programme. Des calculs qui reviennent fréquemment devront être déclarés comme sous programme. Or un sous programme peut lui-même faire appel à un autre sous programme, d'où nécessité de stocker successivement plusieurs adresses de retour dans la pile. L'adresse de la première position mémoire libre dans la pile est en permanence stockée dans un registre appelé *pointeur de pile (Stack pointer* en anglais).

Cette utilisation de la pile pour la sauvegarde des adresses de retour de sous programme se fait automatiquement sans que l'utilisateur ait à s'en préoccuper. Par contre l'utilisateur peut utiliser consciemment la pile lorsqu'il désire sauvegarder momentanément le contenu d'un ou plusieurs registres. Pour ce faire le jeu d'instructions d'un microprocesseur comprend des instructions d'écriture dans la pile, PUSH pour le 8080 A et PSH pour le 6800, et des instructions de lecture dans la pile, POP pour le 8080 A et PUL pour le 6800.

C. Le jeu d'instructions

L'ensemble des instructions d'un microprocesseur peut se répartir en plusieurs catégories

- lecture
- écriture
- transferts de registre à registre - opérations arithmétiques
- incrémentation et décrémentation - fonctions logiques
- décalages et rotations - branchements
- instructions spéciales

Nous allons voir ces instructions en détailles dans le chapitre suivant.

Chapitre 4 : Les microprocesseurs 8 bits

Ce chapitre est consacré à l'étude de quelques microprocesseurs. Au lecteur qui doit s'initier à un microprocesseur bien défini, nous conseillons de n'étudier dans un premier temps que ce microprocesseur. De même nous lui conseillons de n'étudier que les interfaces et les exercices associés à ce microprocesseur.

10. Les microprocesseurs 8080 A et 8085 A

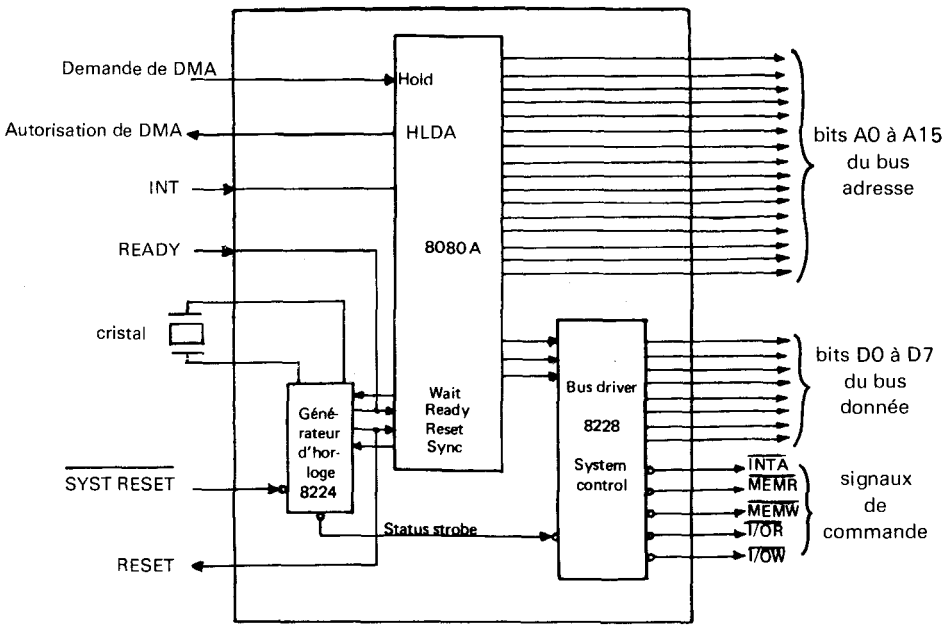
2.1.3 Le Processeur 8080 A de INTEL (« Matériel »)

Par souci de clarté nous allons supposer que le processeur est complet et idéal, c'est à dire qu'il posséde son horloge interne, ses buffers de sortie. Ce n'est pas le cas en général et pour cela il faut connecter un ou deux autres circuits au microprocesseur proprement dit. Nous allons étudier le microprocesseur et ses circuits annexes.

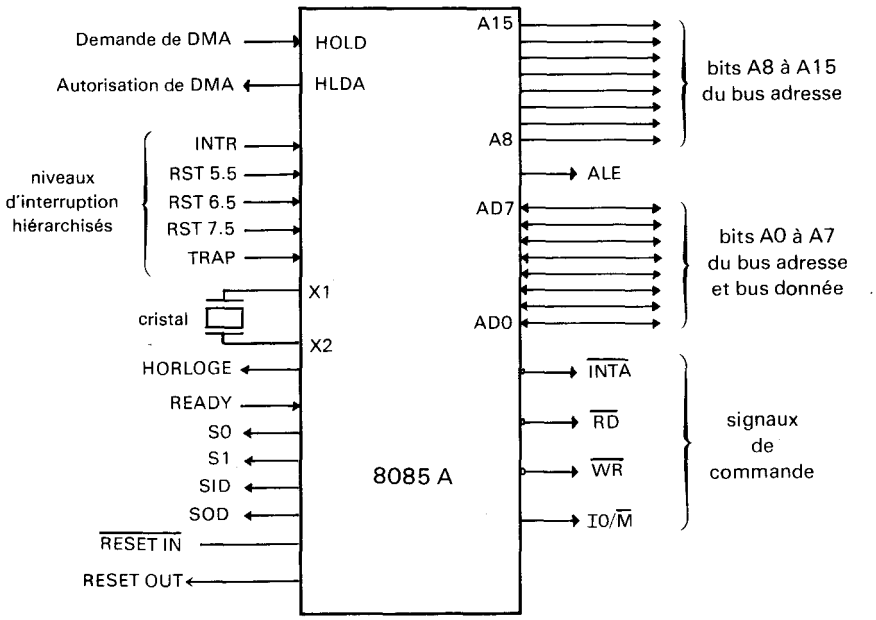
a. Organisation interne et externe du 8080 A

Le 8080 A possède, en plus de l'accumulateur A, du compteur ordinal PC et du pointeur de pile SP, six registres banalisés B, C, D, E, H et L qui peuvent être groupés par deux pour constituer des registres de 16 bits, ce sont les paires BC, DE, HL.

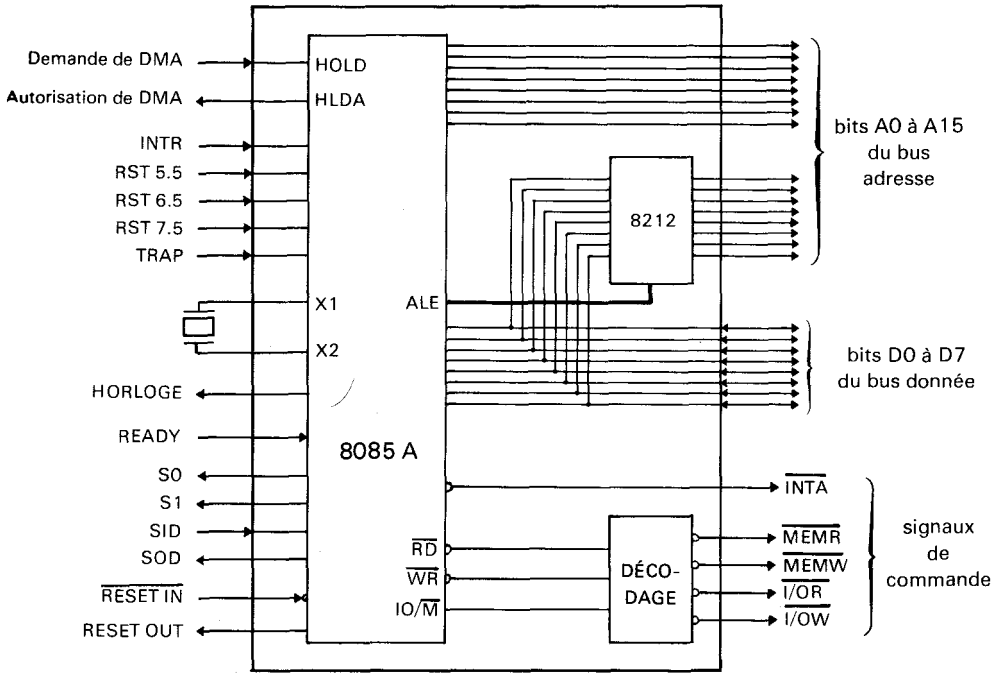
L'ensemble des trois circuits 8080 A + 8228 + 8224 constitue pratiquement le circuit Intel 8085 mais 8 bits d'adresse sont multiplexés avec les 8 bits de données. Alors que le 8080 A nécessite trois alimentations + 5, - 5, + 12 V, le 8085 A ne nécessite qu'une alimentation + 5 Volts.



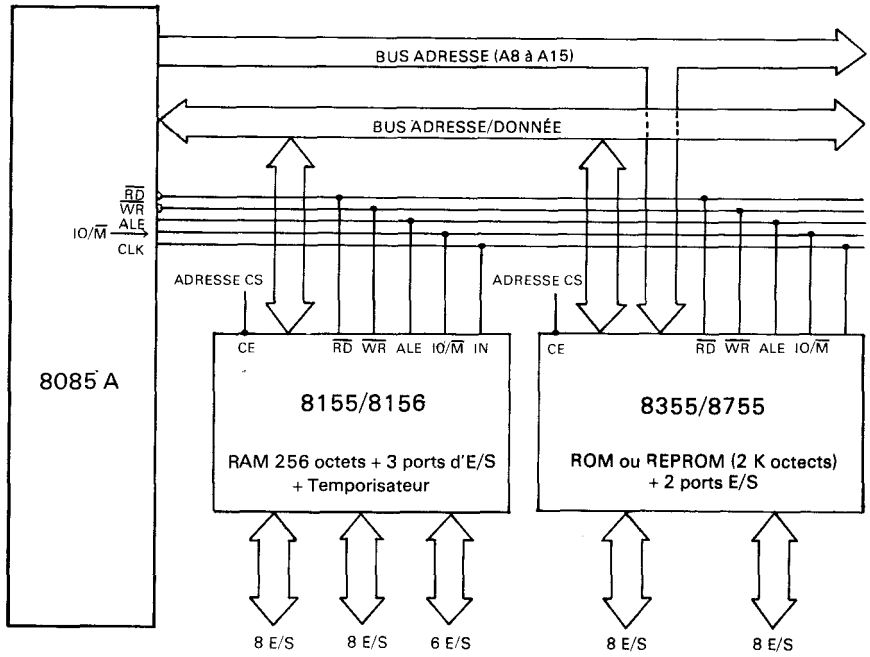
a) *Le processeur 8080 A*



b) *Le processeur 8085 A*



c) Le processeur 8085 A en mode démultiplexé (structure E/S par instruction E/S)



d) *Système minimal réalisé avec la famille 8085 A (mode multiplexé)*

– Les processeurs 8080 A et 8085 A

Fig. 4.1: Les processeurs 8080A et 8085A

- $\overline{\text{MEMR}}$ est le signal complémentaire de « lecture mémoire ». La barre sur le signal signifie que la commande sera active au niveau 0 et inactive au niveau logique 1.

- $\overline{\text{MEMW}}$ est le complément de « écriture mémoire ».

- $\overline{\text{I/OR}}$ et $\overline{\text{I/OW}}$ sont les équivalents de ces deux premiers signaux mais relatifs à la commande des périphériques. Leur présence signifie que le microprocesseur permet un adressage séparé des périphériques (ISOLATED I/O).

- $\overline{\text{INTA}}$ est le complément du signal « autorisation d'interruption ».

- INT est le signal « demande d'interruption ».

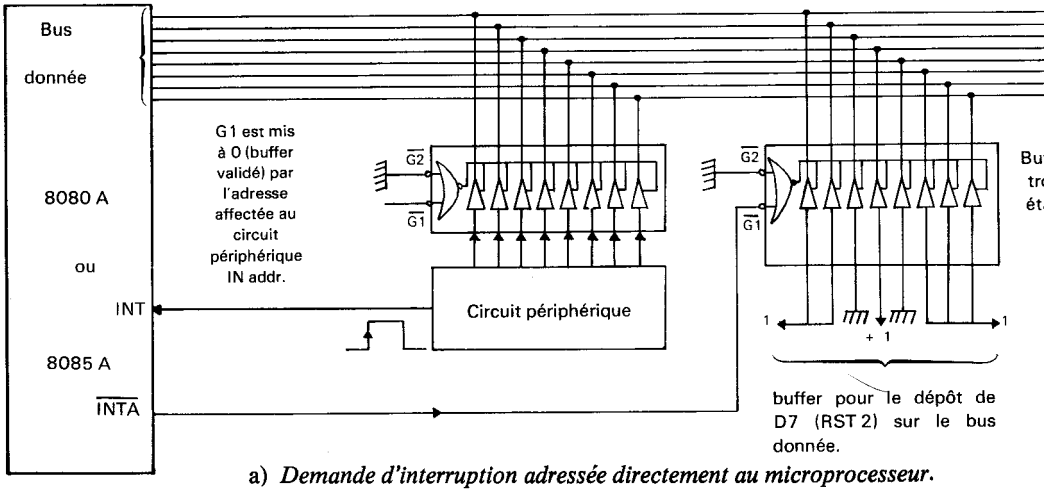
b. Les interruptions

Certains événements se produisant dans un système logique doivent pouvoir arrêter momentanément le déroulement du programme principal du système logique. Pour ce faire ces événements ont la possibilité de déclencher une demande d'interruption. Que sont ces événements? Ce sont : une demande de transfert d'entrée d'un circuit périphérique au microprocesseur, l'indication au microprocesseur de la fin du transfert de sortie d'un caractère, le signalement au microprocesseur d'une panne d'alimentation, d'une erreur de parité, d'une opération arithmétique impossible telle qu'une division par zéro, etc. Aussi tout microprocesseur offre la possibilité de gérer des interruptions. Le 8080 A possède une entrée « demande d'interruption » : c'est la broche INT. Dans ce cas lorsque le circuit périphérique a une donnée à transmettre au microprocesseur, il génère une demande d'interruption. Par exemple, si le circuit périphérique est un clavier, l'appui sur une touche quelconque, génère un « strobe » qui signifie une donnée est disponible en sortie du clavier, il faut que le microprocesseur vienne la lire. Ce « strobe » est alors utilisé comme demande d'interruption. Il peut être adressé directement au microprocesseur si le circuit périphérique est connecté au bus donnée du microprocesseur par l'intermédiaire d'un buffer trois états (fig. 4.2.a). Si le circuit périphérique (clavier par exemple) est connecté à un interface parallèle, le 8255 en l'occurrence, le « strobe » sera relié à une entrée spécifique de cet interface, l'entrée « donnée prête ». Si les interruptions sont autorisées dans cet interface, ce qui est indispensable, la demande d'interruption adressée sur l'entrée «donnée prête» est répercutée au microprocesseur par la sortie INTR (interrupt request) de l'interface parallèle 8255 ou 8255 A. Dans ce mode interruptible le microprocesseur est normalement occupé à l'exécution d'un programme et lorsque le circuit périphérique a une donnée à transmettre (pour un transfert d'entrée) il génère une demande d'interruption. *Le transfert d'entrée est donc effectué à l'initiative du circuit périphérique.*

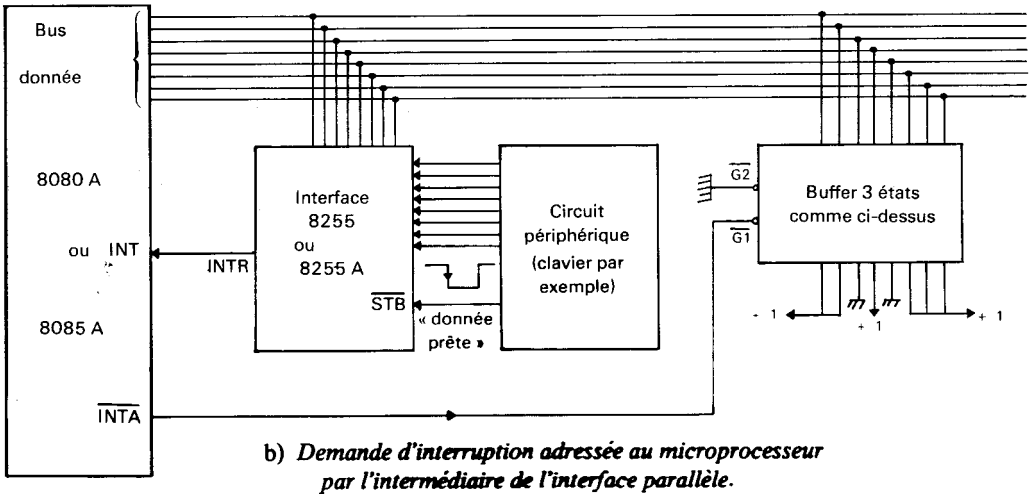
Pour qu'une demande d'interruption soit effectivement prise en compte, c'està-dire pour que le microprocesseur arrête l'exécution du programme principal pour exécuter le sous-programme d'interruption, il faut que les interruptions aient été autorisées :

- au niveau du microprocesseur, dans le cas de la figure 4.2.a, par l'instruction El insérée dans le programme principal;

- au niveau du microprocesseur et de l'interface parallèle dans le cas de la figure 4.2.b : au niveau du microprocesseur par l'instruction El insérée dans le programme principal ainsi qu'au niveau de l'interface par la programmation du mode interruptible de cet interface, et ceci par quelques instructions insérées dans le programme principal. S'il y avait eu un contrôleur d'interruption, il aurait fallu aussi autoriser les demandes d'interruption au niveau de ce circuit contrôleur d'interruption et qui est le 8259 de INTEL.

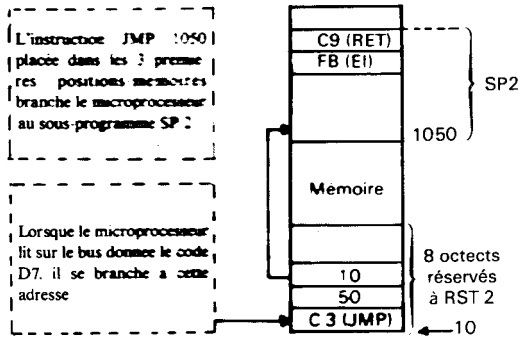


a) *Demande d'interruption adressée directement au microprocesseur.*



b) *Demande d'interruption adressée au microprocesseur par l'intermédiaire de l'interface parallèle.*

INSTRUCTIONS RST.N		
RST.N	Code opération	Adresse spécifique à RST.N
RST 0	C7	00 en hexa
RST 1	CF	08 »
RST 2	D7	10 »
RST 3	DF	18 »
RST 4	E7	20 »
RST 5	EF	28 »
RST 6	F7	30 »
RST 7	FF	38 »



c) *Branchement à l'adresse où est rangé SP 2.*

Fig. 28

Fig. 4.2. :Traitement d'une interruption sur l'entrée INT du 8080 A ou 8085 A

2.1.4 Le processeur 8085 A de Intel (« Matériel »)

Le circuit 8085 A est d'abord l'intégration sur une seule puce du 8080 A, de l'horloge 8224 et du système de contrôle 8228. Toutefois le bus adresse A 0 à A 15 n'est pas directement fourni sur les broches de sortie du 8085 A. Les huit bits d'adresse poids faibles A 0 à A 7 sont fournis sur le bus donnée pendant un temps relativement court mais suffisamment long pour permettre un verrouillage de ces huit bits d'adresse. Après quoi le bus donnée redevient vraiment le bus donnée. Pour que le verrouillage des huit bits d'adresse poids faibles soit possible il faut bien entendu que le 8085 A délivre une commande de verrouillage pour signaler à quel moment les bits A 0 à A 7 peuvent être prélevés : cette commande est le signal ALE (Address Latch Enable). Ce verrouillage se fait par un verrou trois états, le 8212 de Intel par exemple. Mais le 8085 A offre deux modes de fonctionnement et dans l'un de ces modes l'utilisateur n'a pas à effectuer ce verrouillage.

a. Organisation interne et externe du 8085 A

i. Mode démultiplexé

Dans ce mode le bus adresse est intégralement restitué, le signal ALE envoyant une commande de verrouillage au moment adéquat au verrou trois états qui est le 8212.

Comme le 8080 A permet des transferts E/S selon les 2 structures E/S possibles :

- structure E/S par instructions mémoire : les registres des interfaces et des organes d'E/S sont considérés comme des position-mémoires (MEMORY MAPPED I/0);

- structure E/S par instructions E/S : les registres des interfaces et des organes d'E/S sont adressés à partir d'instructions spéciales et par les signaux spéciaux que sont LECTURE PERI et ECRITURE PERI (ISOLATED I/0),

chacune de ces deux structures E/S conduit à une utilisation différente des signaux du 8085 A. Nous allons les étudier l'une après l'autre.

* Mode démultiplexé et structure E/S par instructions E/S (fig. 4.2. c)

C'est tout à fait le fonctionnement habituel du 8080 A. Pour retrouver alors les signaux du 8080 A que sont :

- $\overline{MEMR}$ lecture mémoire
- $\overline{MEMW}$ écriture mémoire
- $\overline{I/OR}$ lecture péri (lecture d'un circuit périphérique)
- $\overline{I/OW}$ écriture péri

Il nous faut effectuer un décodage très simple.

Le 8085 A délivre trois signaux :

- $\overline{RD}$ lecture
- $\overline{WR}$ écriture
- $IO/\overline{M}$ sélection de la destination de la commande de lecture ou d'écriture : mémoire si $IO/\overline{M} = 0$, circuit périphérique (organe d'E/S) si $IO/\overline{M} = 1$.

A partir de ces trois signaux il est facile de se constituer les quatre signaux du 8080 A.

- $\overline{MEMR} = \overline{RD} + IO/\overline{M}$
- $\overline{MEMW} = \overline{WR} + IO/\overline{M}$
- $\overline{I/OR} = \overline{RD} + IO/\overline{M}$
- $\overline{I/OW} = \overline{WR} + IO/\overline{M}$

* Mode démultiplexé et structure E/S par instructions mémoire.

C'est le fonctionnement du 8080 A en structure E/S par instructions mémoire. Toute adresse doit être univoque et par conséquent elle doit être celle d'une position-mémoires ou d'un registre mais pas les deux à la fois. Pour ce faire un procédé simple consiste à décoder deux bits d'adresse parmi les poids forts, généralement A 15 et A 14. Exemple :

A 15	A 14	
0	0	validation de la ROM ou REPROM
0	1	validation de la RAM
1	0	validation des circuits périphériques
1	1	inutilisé.

Dans ce mode

$\overline{RD}$ est bien entendu le signal de lecture et

$\overline{WR}$ le signal d'écriture

le signal $IO/\overline{M}$ m n'est pas utilisé.

ii. Mode multiplexé

Le 8085 A fait partie d'une famille de trois circuits LSI, le 8085 A, le 8155 et le 8355. Ces deux derniers circuits ont été conçus pour effectuer eux-mêmes le démultiplexage des bits A 0 à A 7 du bus adresse. Aussi ils reçoivent le signal ALE.

Le 8155 intègre sur une seule puce

- une RAM de 256 octets.
- deux ports d'E/S de huit bits (ports A et B)

- un port d'E/S de six bits (port C)
- un temporisateur programmable.

Le 8355 intègre sur une seule puce :

- une ROM de 2 K octects. Il existe un circuit identique au 8355 en version REPROM: le 8755.
- deux ports d'E/S de huit bits (ports A et B).

* Mode multiplexé et structure E/S par instructions E/S.

C'est le fonctionnement standard du 8085 A en mode multiplexé. Les circuits 8155 et 8355/8755 réalisent eux mêmes le décodage des signaux $\overline{RD}$, $\overline{WR}$ et I0/$\overline{M}$ pour obtenir les signaux $\overline{MEMR}$, $\overline{MEMW}$, I/0$\overline{R}$, T/0W. Aussi ils reçoivent le signal I0/M comme indiqué sur la fig. 4.1. d. Les ports d'E/S sont alors adressés par l'une des instructions IN ou OUT avec une adresse de 8 bits.

* Mode multiplexé et structure E/S par instructions mémoire.

Pour adresser les ports d'E/S comme des position mémoires il suffit de connecter les entrées I0/$\overline{M}$ du 8155 et du 8355/8755 à un poids d'adresse, A 15 par exemple. Dans ce cas toute adresse avec A 15 = 1 concerne les organes d'E/S et toute adresse avec A 15 = 0 concerne une mémoire (RAM ou ROM/REPROM). La sortie I0/$\overline{M}$ du 8085 A n'est pas utilisée.

b. Les signaux du 8085 A

• Le signal de sortie WAIT du 8080 A n'existe pas avec le 8085 A.

• L'entrée READY permet comme avec le 8080 A l'utilisation de mémoires lentes. Toutefois cette entrée est asynchrone pour le 8080 A et synchrone pour le 8085 A.

• HOLD et HLDA sont comme pour le 8085 A, respectivement la demande d'accès direct mémoire et l'autorisation d'accès direct mémoire.

• $\overline{\text{RESET IN}}$ est l'entrée Reset du 8085 A. Cette entrée attaque un trigger de Schmitt de sorte que le « reset » à la mise sous tension peut être effectué par un simple circuit RC. Le 8085 A fournit le signal $\overline{\text{RESET OUT}}$ lorsque l'entrée RESET IN a été activée.

• Les broches SID et SOD sont respectivement une entrée série et une sortie série. Toutefois elles sont loin d'être équivalentes à l'entrée série et la sortie série d'un interface série programmable comme le 8251.

• Les sorties S 0 et S 1 indiquent, sous forme codée, l'opération que le microprocesseur exécute. Ces deux bits S 0 et S 1 peuvent être prélevés et mémorisés à l'aide du signal ALE, agissant comme une commande de verrouillage.

• La fréquence de l'horloge est déterminée soit par un cristal connecté entre X 1 et X 2 soit par une horloge externe reliée à X 1.

• CLK est une sortie horloge dont la fréquence est la moitié de celle du quartz ou de l'horloge appliquée sur X 1.

• Les entrées « demande d'interruption » que nous allons étudier.

c. Les interruptions

Le 8085 A possède cinq niveaux hiérarchisés d'interruption qui sont, dans l'ordre de priorité décroissante: TRAP, RST 7.5, RST 6.5, RST 5.5 et INTR.

• Interruption non masquable TRAP

Tout front montant ou tout niveau 1 sur cette entrée déclenche le processus de prise en compte d'une interruption par le microprocesseur après qu'il ait terminé l'exécution de l'instruction en cours. C'est la seule demande d'interruption sans condition du 8085 A.

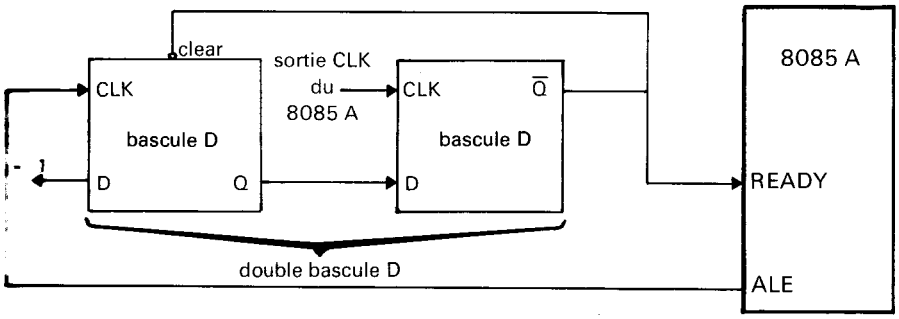
• Interruptions masquables RST 7.5, RST 6.5 et RST 5.5

Chacune de ces trois demandes d'interruption ne peut être prise en compte par le microprocesseur que si deux conditions sont réalisées :

- le bit d'état IE est positionné à 1. Il est mis à 1 par l'instruction El (autorisation des interruptions) et à 0 par l'instruction DI (inhibition des interruptions).
- la demande d'interruption n'est pas masquée.

• Interruption INTR

L'utilisation de INTR est celle de la broche INT du 8080 A ; ce qui signifie qu'une demande d'interruption sur cette broche entraîne le dépôt sur le bus donnée, par l'organe d'E/S qui a provoqué l'interruption, du code RST N affecté à cet organe d'E/S. Ainsi, si le niveau d'interruption RST 3 est affecté à un organe d'E/S, celui-ci déposera sur le bus donnée, suite à une demande d'interruption de sa part, le code de RST 3 soit DF.



a) Circuit de génération d'un état WAIT

Interruption	code à déposer sur le bus donnée	adresse à partir de laquelle est rangée l'instruction de saut au sous-programme
RST 0	C7	00
RST 1	CF	08
RST 2	D7	10
RST 3	DF	18
RST 4	E7	20
* TRAP		24
RST 5	EF	28
* RST 5.5		2C
RST 6	F7	30
* RST 6.5		34
RST 7	FF	38
* RST 7.5		3C

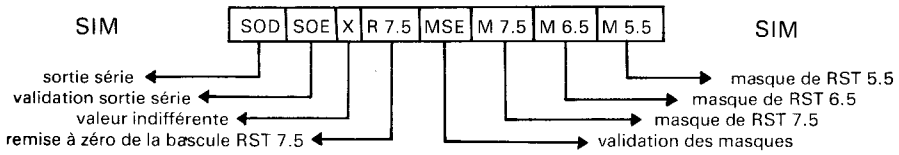
* = niveaux d'interruption spécifique au 8085 A

b) Instructions RESTAR T et leurs adresses

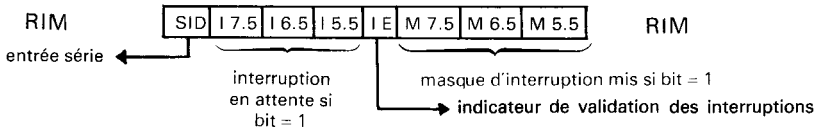
IO/$\overline{M}$	S1	SO	cycle machine (opération effectuée)
HI	0	0	état HALT
0	0	1	écriture mémoire
0	1	0	lecture mémoire
0	1	1	cycle FETCH
1	0	1	écriture péri
1	1	0	lecture péri
1	1	1	état $\overline{INTA}$

HI = haute impédance

c) Code des cycles machines



d) Synoptique de détermination du mot de commande de l'instruction SIM



e) Mot d'état envoyé dans l'accumulateurpar l'instruction RIM

Fig. 4.3 : Nouveautés du 8085 A par rapport au 8080 À

2.1.5 Le logiciel du 8080 A ou du 8085 A, de Intel

a. Les modes d'adressage.

• L'adressage registre

Il n'y a ni opérande, ni adresse.

Exemple : MOV A, B soit transfert du contenu de B dans A.

• L'adressage immédiat

Il est défini par la dernière lettre du code instruction ; soit I.

Exemple : MVI A, 08 qui signifie initialiser A à 08.

• L'adressage direct

L'adresse est exprimée sur 16 bits dans l'instruction.

Exemple : LDA 1000 H qui signifie lire la position-mémoire d'adresse 1000 en hexadécimal et transférer son contenu dans l'accumulateur.

• L'adressage indirect par registre

Les registres B, C, D, E, H et L sont organisés pour pouvoir être associés par deux, ce qui crée trois registres de 16 bits, appelés « paires de registres » : BC, DE et HL. Les deux premières paires BC et DE sont utilisées comme registres d'adressage indirect pour deux instructions :

- une instruction de lecture LDAX rp dans laquelle rp, qui signifie « register pair », désigne la paire de registres concernée, et qui est désignée par son premier registre, soit B ou D. Exemple : LDAX B qui signifie charger dans l'accumulateur le contenu de la position-mémoire dont l'adresse est le contenu de BC,

- une instruction d'écriture STAX rp.

Exemple : STAX D qui signifie écrire le contenu de l'accumulateur dans la position-mémoire dont l'adresse est le contenu de DE.

La paire HL est prévue pour remplacer le registre index qui n'existe pas. Elle joue le rôle d'un index avec un déplacement toujours nul. Toutes les instructions utilisant la paire HL pour registre d'adressage indirect sont reconnaissables par la lettre M dans le code instruction, cette lettre symbolisant une position-mémoire.

Exemple : MOV A, M qui signifie transférer dans l'accumulateur A, le contenu de la position-mémoire dont l'adresse est le contenu de HL, ce que nous représentons par (M) $\rightarrow$ A.

REMARQUE:

Pour les mots de 16 bits, l'octet poids faible est traité le premier ; il correspond au deuxième registre de la paire de registres.

Exemple : LHLD addr, qui signifie transférer dans la paire HL les contenus des position-mémoires d'adresses addr et addr + 1. Les transferts sont les suivants :

(addr) → L et (addr + 1) → H

b. Le jeu d'instructions

i. Instructions de lecture et d'écriture

Nous donnerons le code mnémonique de l'instruction et ce qu'elle fait. Une instruction est exprimée par 1, *2 ou 3* octets, le premier étant le code opération.

• *Lecture d'un port de périphérique.*

IN port (port = adresse du port sur 8 bits) (IN = input)

(Port) → A (A = accumulateur)

Cette dernière expression signifie : transférer le contenu (symbolisé par les parenthèses) du port concerné dans l'accumulateur, puisque c'est toujours ce dernier qui reçoit les données de l'extérieur. Une autre instruction permettra de transférer le contenu de l'accumulateur en mémoire.

• *Écriture dans un port de périphérique.*

OUT port (port= adresse du port sur 8 bits) (OUT = output)

(A) → Port

• *Lecture d'un octet en mémoire (adressage direct).*

LDA addr (addr = adresse sur 16 bits de la position mémoire)

(LDA = Load Accumulateur - charger J'accumulateur)

(M) → A

(M) symbolise le contenu de la position mémoire d'adresse addr. Cette instruction permet de lire un octet en mémoire.

• *Lecture de deux octets en mémoire (adressage direct).*

LHLD addr.

Deux positions mémoires seront concernées par cette instruction, donc deux adresses La partie adresse de l'instruction, représentée par addr, sera toujours l'adresse la plus faible et l'octet correspondant à cette adresse ira toujours dans le registre L abréviation de LOW (bas), H étant l'abréviation de HIGH (haut). Il en sera toujours ainsi pour les opérations sur la paire de registres H et L.

(M) → H L ce qui, décomposé, signifie

(addr) → L (addr + 1) → H

• *Lecture d'un octet en mémoire (adressage indirect explicite).*

LDAX rp X signifie que l'on va travailler sur une paire de registres, ce que confirme rp. Les registres auxiliaires du microprocesseur peuvent s'associer deux par **deux** et dans l'ordre alphabétique à partir de B puisque A est l'accumulateur : BC, DE et HL. Ici rp est l'une des paires BC ou DE.

(M) → A soit ((rp)) → A

Dans cette instruction, l'adressage est indirect puisque la partie adresse n'est plus l'adresse de la position mémoire mais l'indication du registre dont le contenu est l'adresse mémoire. Le registre contenant l'adresse mémoire est indiqué explicitement ce qui ne sera pas le cas de la prochaine instruction.

• *Lecture d'un octet en mémoire (adressage indirect implicite).*

Dans les instructions de lecture et d'écriture qui vont suivre l'adresse de la position mémoire concernée sera implicitement le contenu de la paire de registres H et L, lorsque dans une instruction nous verrons apparaître la lettre M symbolisant la mémoire.

MOV r, M (MOV = move = transférer)

(M) → r r désignant l'un ou l'autre des registres A, B, C, D, E, H ou L, qui sont les registres auxiliaires du microprocesseur.

L'instruction MOV C, M signifie:

(M) → C ou ((HL)) → C

((HL)) = contenu de la position-mémoire dont l'adresse est dans HL.

• *Lecture de deux octets dans la pile de la mémoire.*

POP rp rp = BC, DE, HL et PSW.

((SP)) → rp SP = pointeur de pile (16 bits)

PSW signifie Processor Status Word. C'est le mot d'état du processeur. Il est constitué du contenu de l'accumulateur et des indicateurs.

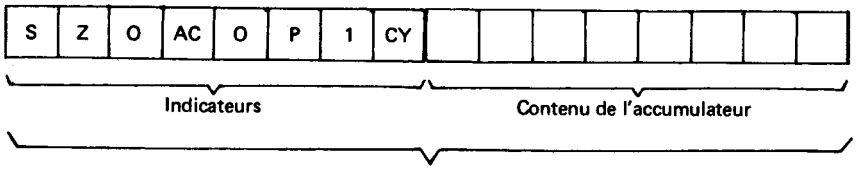


Fig. 4.4 : Mot d'état du processeur PSW

• *Écriture d'un octet en mémoire (adressage direct).*

STA addr **(STA = Store A = ranger en mémoire le contenu de l'accumulateur)**

(A) → M

• *Écriture de deux octets en mémoire (adressage direct).*

SHLD addr (SHLD = ranger en mémoire le contenu de HL)

(HL) → M

• *Écriture d'un octet en mémoire (adressage indirect explicite).*

STAX rp rp est l'une des paires de registres BC ou DE

(A) → M

Le contenu de l'accumulateur est transféré dans la position mémoire dont l'adresse est contenue dans BC ou DE.

• *Écriture d'un octet en mémoire (adressage indirect implicite).*

MOV M, r

(r) → M avec r = A, B, C, D, E, H ou L

Rappelons que l'adresse de la position mémoire concernée est dans HL, ce qui entraîne qu'il ne faudra pas oublier de mettre dans HL l'adresse souhaitée si elle n'y est pas.

• *Écriture d'un octet en mémoire (adressage immédiat).*

MVI M, data

data → M

La partie adresse de l'instruction est remplacée par l'opérande à écrire. Cette instruction est très pratique pour initialiser des positions mémoires. Ainsi MVI M, 7E transférera à la position mémoire dont l'adresse est dans HL la valeur 7E en hexadécimal soit 01111110 en binaire ou 126 en décimal.

• *Écriture de deux octets dans la pile de la mémoire.*

PUSH rp rp = BC, DE, HL et PSW

(rp) → (SP) SP = pointeur de pile (16 bits)

contenu de rp dans la position mémoire dont l'adresse est le contenu de SP.

Aucune de ces instructions ne modifie les indicateurs sauf POP PSW.

ii. Instructions de transfert de données entre registres

Ces instructions seront utiles pour sauvegarder des données se trouvant dans l'accumulateur A et qui seraient perdues à la suite d'opérations sur l'accumulateur, pour incrémenter ou décrémenter une adresse, ce qu'il n'est pas possible de faire directement à partir de la mémoire, mais qui le devient en transférant l'adresse dans un registre, pour sauvegarder le contenu de HL quand cette paire de registres doit recevoir une adresse pour un adressage indirect, pour translater un programme en mémoire, etc...

• MOV r1, r2, r1, r2 = A, B, C, D, E, H, L

(r2) → r1

Signalons que si r1 = r2 l'instruction ne veut plus rien dire et il ne se passera rien.

Par contre l'instruction MOV M, M est interdite car elle serait interprétée comme un ordre d'arrêt du microprocesseur.

• MVI r, $data_8$, (data représente une donnée d'un octet)

data → r = A, B, C, D, E, H, L.

Ces deux dernières instructions sont à rapprocher des instructions MOV M, r; MOV r, M ; MVI M, data, qui, elles, sont référencées mémoire.

• LXI rp, $data_{16}$ (data représente une donnée de deux octets)

X rappelle que cette instruction porte sur des paires de registres, en l'occurrence bu paires BC, DE, HL et même le pointeur de pile SP qui, lui, contient 16 bits.

Data → rp.

Comme toujours pour les paires de registres, le premier octet de la donnée ira da ns le registre désigné par la lettre la plus faible dans l'ordre alphabétique, B pour la paire OC. D pour DE, H pour HL. Pour SP il n'y a pas de confusion puisque SP est un registre de 16 bits.

• PCHL

(HL) → PC PC = compteur ordinal

• SPHL

(HL) → SP SP = pointeur de pile.

• XCHG échange des contenus des paires HL et DE.

Soit (H) ↔ (D) et (L) ↔ (E)

• XTHL Le contenu de L est échangé avec celui de la position-mémoire dont l'adresse est le contenu du pointeur de pile, SP. Le contenu de H est échangé avec celui de la position-mémoire dont l'adresse est le contenu du pointeur de pile plus un soit (SP) + 1.

Aucune de ces instructions ne modifie les indicateurs.

c. Insiructions arithmétiques

Ces instructions vont souvent modifier les indicateurs associés à l' U.A.L. Nous avons jusqu'ici vu deux indicateurs, le CARRY et le ZÉRO,respectivement désignés en abrégé par CY et Z.

• L'indicateur SIGNE désigné par S indique le signe du nombre dans l'accumulateur. C'est le bit 2^7 de ce dernier.

S = 0 nombre positif S = 1 nombre négatif

• L'indicateur PARITE désigné par P. - Lorsque le nombre de 1 contenus dans l'accumulateur est pair l'indicateur de parité prend la valeur 1. Lorsque ce nombre est impair l'indicateur a la valeur 0.

• ADD r avec r = A, B, C, D, E, H, L et aussi M.

Pour la mémoire M l'adresse est le contenu de HL.

Pour la plupart des opérations arithmétiques, la mémoire s'ajoutera à la liste des registres auxiliaires :

$$(A) + (r) \rightarrow A$$

• ADC r avec r = A, B, C, D, E, H, L et M

$$(A) + (CY) + (r) \rightarrow A$$

• ADI data

$$(A) + data \rightarrow A$$

• ACI data

$$(A) + (CY) + data \rightarrow A$$

• DAA *(decimal adjust accumulator).* - *Cette* instruction est une correction nécessaire pour faire l'addition de nombres codés en BCD.

• SUB r r = A, B, C, D, E, H, L et M

$$(A) - (r) \rightarrow A$$

• SBB r r = A, B, C, D, E, H, L et M

$$(A) - (CY) - (r) \rightarrow A$$

• SUI data

$$(A) - data \rightarrow A$$

• SBI data

$$(A) - (CY) - data \rightarrow A$$

Pour chacune des instructions arithmétiques que nous venons de voir tous les indicateurs sont concernés: Z, S, P, CY, AC.

Seule l'instruction arithmétique suivante ne peut modifier qu'un seul indicateur : CY.

• DAD rp

rp désigne l'une des paires de registres BC, DE, HL, SP

$$(HL) + (rp) \rightarrow HL$$

L'indicateur CARRY est donc affecté par toutes les opérations arithmétiques.

d. Instructions d'incrémentation ou de décrémentation

• INR r r = A, B, C, D, E, H, L et M

(r) + 1 → r incrémentation

• DCR r r A, B, C, D, E, H, L et M

(r) - 1 → r décrémentation

Les deux instructions ci-dessus affectent tous les indicateurs sauf le carry.

• INX rp rp = BC, DE, HL, SP

(rp) + 1 → rp

• DCX rp rp = BC, DE, HL, SP

(rp) – 1 → rp

Les deux instructions ci-dessus n'affectent aucun indicateur. L'indicateur CARRY n'est donc jamais affecté par une incrémentation et une décrémentation.

e- Instructions de branchement

Elles sont de trois types : des sauts à une adresse du programme principal (JUMP), des appels d'un sous-programme (CALL) et des sauts pour retourner au programme principal à la fin d'un sous programme (RETURN). Chacune de ces trois instructions peut être inconditionnelle ou conditionnelle, c'est à dire dépendant de la valeur d'un des indicateurs CARRY, ZERO, SIGNE et PARITÉ. D'où la liste de toutes les instructions possibles :

inconditionnel	JMP addr	CALL addr	RET
saut si CY= 1	JC addr	CC addr	RC addr
saut si CY= 0	JNC addr	CNC addr	RNC addr
saut si Z= 1	JZ addr	CZ addr	RZ addr
saut si Z= 0	JNZ addr	CNZ addr	RNZ addr
saut si S= 0	JP addr	CP addr	RP addr
saut si S= 1	JM addr	CM addr	RM addr

saut si P= 1	JPE addr	CPE addr	RPE addr
saut si P= 0	JP0 addr	CP0 addr	RP0 addr

pour ces deux dernières conditions PE = parity even (P = 1), P0 = parity odd (P = 0), addr est l'adresse de 16 bits où se fera le saut. Pour cela cette adresse sera transférée dans le compteur ordinal. L'instruction RST n (restart) permet de se brancher à un sous-programme d'interruption dont l'adresse est 0000000000NNN000, NNN est le nombre n de trois bits (vecteur).

Aucun indicateur n'est affecté par l'une quelconque des instructions de branchement.

f. Opérations logiques

• ANA r r = A, B, C, D, E, H, L et M

(A) Λ (r) → A le symbole Λ est le ET logique

• ORA r r = A, B, C, D, E, H, L et M

(A) V (r) → A le symbole V est le OU logique

• XRA r r = A, B, C, D, E, H, L et M

(A) $\veebar$ (r) → A le symbole $\veebar$ est le OU exclusif

• ANI data (A) Λ data → A

• ORI data (A) V data → A

• XRI data (A) $\veebar$ data → A

Ces six dernières instructions affectent tous les indicateurs. Le carry est mis à zéro.

• CMP r r - A, B, C, D, E, H, L et M.

C'est une comparaison faite en faisant la soustraction (A) - r mais le résultat ne va nulle part. Seuls les indicateurs sont modifiés

si (A) = (r)	Z=1
si (A) < (r)	CY=1
si (A) > (r)	CY = 0

• CPI data

C'est encore une comparaison faite par la soustraction (A) - data, le résultat n'allant nulle part, les indicateurs, eux, étant modifiés.

si (A) = data Z = 1
si (A) < data CY = 1
si (A) > data CY = 0

• CMA C'est la complémentation à 1 du contenu de A

$$\overline{(A)} \rightarrow A$$

Aucun indicateur n'est affecté.

• CMC C'est la complémentation du « carry »

$$\overline{(CY)} \rightarrow C\ Y$$

• STC C'est la mise à « 1 » du « carry » :

$$1 \rightarrow CY$$

Pour ces deux dernières instructions seul CY est affecté.

g. Instructions de décalage

RLC décalage gauche d'un bit dans le « carry »

RAL décalage gauche d'un bit avec le « carry » dans la boucle

RRC décalage droite d'un bit dans le « carry »

RAR décalage droite d'un bit avec le « carry » dans la boucle

Pour ces quatre décalages seul le « carry » est affecté

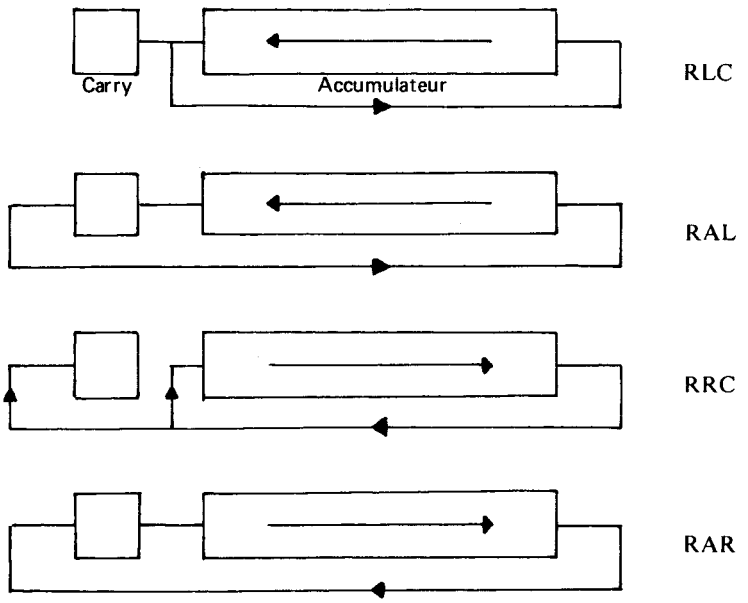


Fig. 4.5. : structions de décalage

h. instructions de commande du microprocesseur

• EI enable interrupt. - Cette instruction autorise le microprocesseur à prendre en compte d'éventuelles demandes d'interruptions.

• DI disable interrupt. - Cette instruction interdit au microprocesseur de prendre en compte d'éventuelles demandes d'interruption.

• HALT. - Cette instruction arrête le microprocesseur jusqu'à ce que se produise une interruption, ce qui veut dire que si on utilise cette instruction alors qu'on a inhibé les interruptions, le processeur sera arrêté, pour toujours ou tout au moins jusqu'à ce que l'on fasse un « reset ».

• NOP. - Cette instruction incrémente le compteur ordinal mais il ne se passe rien. Elle peut être utile pour rajouter des instructions dans un programme si on a mis des NOP.

i. Particularités du jeu d'instructions du 8080 A ou du 8085 A.

• Les instructions LDA et IN ne positionnent aucun indicateur. Pour positionner les adicateurs lorsque cela est nécessaire, il faut utiliser une instruction arithmétique ou logique qui ne modifie pas le contenu de l'accumulateur.

Exemple :

```
IN      addr
ORI     00
```

• Le 8080 A ou 8085 A ne possède pas d'instruction véritable de lecture indexée mais à possède trois paires de registre permettant l'adressage indirect par registre : HL BC, DE. La paire HL est privilégiée : l'adresse est implicitement contenue dans HL lorsque l'instruction porte la référence mémoire M comme ADD M - MOV M,r - MVI M,data etc. Les instructions de lecture ou d'écriture en adressage indirect par HL sont des instructions MOVE.

Aux paires BC et DE sont associées deux et uniquement deux instructions : LDAX rp et STAX rp.

• Les instructions LHLD et SHLD permettent respectivement une lecture et une écriture d'une donnée de 16 bits.

• L'instruction DCX rp ne positionne aucun indicateur. Pour positionner l'indicateur Z qui s'associe bien à cette instruction, il faut faire le OU logique des deux registres constituant rp.

Exemple :

```
DCX     D
MOV     A, D
ORA     E
```

• Les instructions d'addition et de soustraction ne peuvent se faire en adressage direct, ce qui est dommage. Ainsi l'instruction ADD addr n'existe pas. Ces instructions se font entre l'accumulateur et un autre registre ou en adressage indirect par registre avec la paire HL.

Exemples :

```
LDA   addr 1          LXI H,  addr 2
MOV   B, A            LDA     addr 1
LDA   addr 2          ADD  M
ADD   B
```

• Une position-mémoire peut être chargée en adressage immédiat par l'instruction MVI M, data.

• Pour les opérations relatives aux registres de 16 bits, l'octet poids faible est traité le premier et par conséquent correspond à addr.

Exemple :

LHLD addr : (addr) → L et (addr + 1) → H

• Dans la traduction en hexadécimal d'une instruction assembleur, les adresses ou les données de 16 bits sont exprimées avec l'octet poids faible en premier.

Exemple :

LXI H, 1050 donne 215010

LDA 2070 donne 3A7020

• Les instructions de décalage et de rotation ne peuvent se faire que sur l'accumulateur A.

• Le 8080 A ou le 8085 A possède des instructions d'appel conditionnel d'un sous programme et de retour conditionnel d'un sous programme.

• Le pointeur de pile SP ne peut pas être chargé à partir de la mémoire. Il est chargé en adressage immédiat par LXI SP.

• Le 8080 A ou 8085 A ne possède pas d'instruction de transfert dans l'accumulateur des états des indicateurs.

j. Instructions spécifiques au 8085 A

Ce sont les instructions SIM et RIM.

11. Les Microprocesseurs 6800, 6802 et 6809

2.1.6 Le processeur 6800 de Motorola (« Matériel») : *Organisation interne et externe.*

Le 6800 est un microprocesseur 8 bits dans un boîtier de quarante broches réalisé en technologie MOS canal N. Il possède un bus de données de 8 bits et un bus d'adresses de 16 bits, ce qui lui permet d'adresser 2^{16} position mémoires soit 64 K octets. Il ne nécessite qu'une seule tension d'alimentation + 5 volts. Alors que les microprocesseurs Intel 8080 A et 8085 A sont conçus pour travailler beaucoup avec les registres, le 6800 a été étudié pour une utilisation abondante de la mémoire, puisqu'il ne possède qu'un seul registre de travail autre que l'accumulateur A. Ce registre de travail est d'ailleurs un deuxième accumulateur, ne permettant toutefois pas les additions décimales. La figure 4.6 montre les registres du 6800. Le registre indicateur est appelé registre de conditions et désigné par CC (registre code condition).

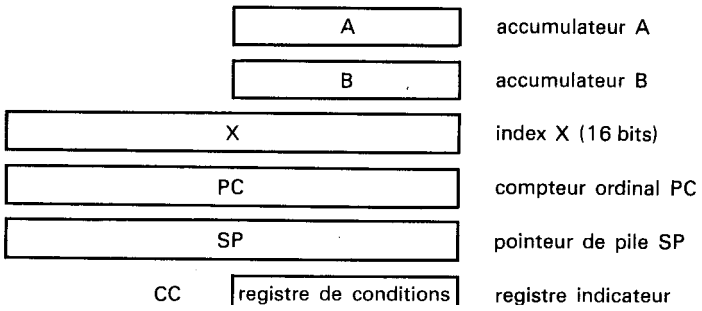


Fig. 4.6 : Les registres du 6800 et du 6802

Le 6800 communique avec les interfaces selon la structure E/S par instructions mémoire ; c'est-à-dire qu'il adresse un registre d'un interface avec les mêmes instructions qu'une position-mémoire. Deux signaux sont utilisés pour cet adressage : R/W (lecture/écriture) et VMA. ∅2. Ce dernier est la fonction ET de VMA (signal prenant l'état « 1 » lorsqu'une adresse est valide sur le bus d'adresses) et de l'horloge ∅2.

Le 6800 possède deux entrées interruption :

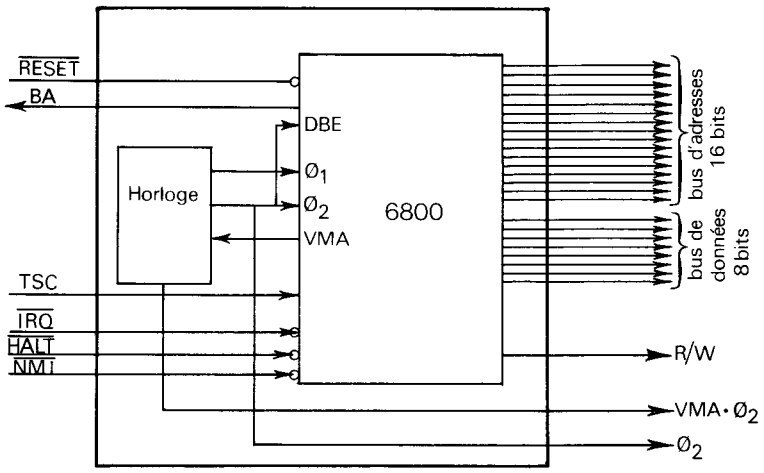
- $\overline{\text{IRQ}}$ demande d'interruption masquable par programme.

- $\overline{\text{NMI}}$ demande d'interruption non masquable.

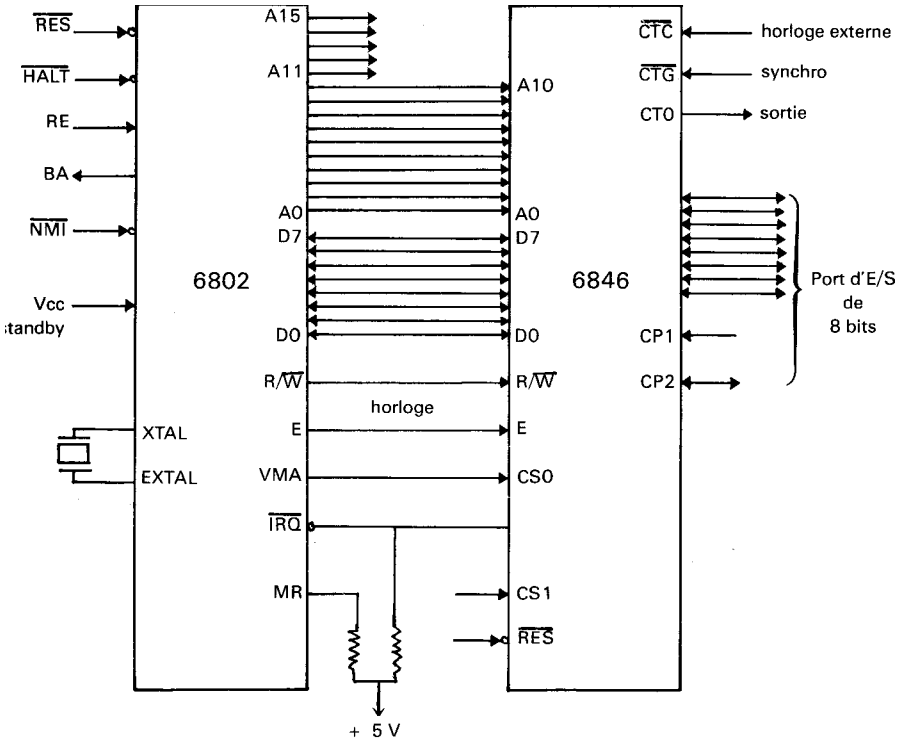
- $\overline{\text{HALT}}$: le microprocesseur s'arrête et met à haute impédance le bus d'adresses, le bus de données et le signal R/W.

- TSC (Three State Control) : le bus de données ne passe pas à l'état haute impédance. Ce procédé un peu particulier ne permet de transmettre que quelques caractères à la fois ; il est appelé ACCÈS DIRECT MÉMOIRE PAR VOL DE CYCLE. Pour l'une ou l'autre de ces deux demandes d'interruption, la sortie VMA est à zéro alors qu'elle est

nécessaire pour l'adressage des mémoires. Aussi un signal extérieur VMA devra être créé et mis en ou-câblé avec le VMA du processeur.



a) Le processeur 6800



b) Système minimal avec le processeur 6802

Fig. 4.7: Les processeurs 6800 et 6802

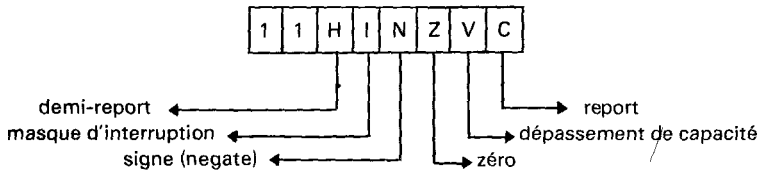


Fig. 3!

Fig. 4.8 : Le registre de conditions du 6800 et du 6802

La réponse à la demande de DMA par le signal HALT est :

BA (bus available = bus disponible)

L'initialisation du microprocesseur se fait par le niveau logique « 0 » du signal RESET qui doit être créé par un trigger et non un circuit RC.

Le 6800 possède deux accumulateurs A et B mais aucun registre auxiliaire si ce n'est B qui peut bien entendu remplir cette fonction, Il possède également un registre index X de 16 bits qui permet l'adressage indexé. Toute instruction faisant appel à l'accumulateur devra préciser s'il s'agit de A ou B; exemple: LDA A 0100 ; LDA B 0100. Pour l'adressage indexé il suffira de mettre la lettre X après l'opérande; exemple : LDAA 5,X

2.1.7 Le processeur 6802 de MOTOROLA (« Matériel »)

Le 6802 est une intégration dans un même boîtier du microprocesseur 6800, d'un oscillateur d'horloge, et d'une mémoire RAM de 128 octets situés entre les adresses hexadécimales 0000 et 007F. Les entrées TSC (Three state control) et DBE (Data Bus Enable) du 6800 n'existent plus avec le 6802. Les autres broches du 6800 subsistent. De nouvelles broches sont apparues, ce sont :

• MEMORY READY (MR). Cette entrée «Mémoire prête », compatible TTL se trouvait sur le circuit horloge du 6800. Elle permet la connexion au 6802 de mémoire ou d'organes d'E/S lents.

• Vcc standby. Cette broche est l'alimentation de 32 octets de la RAM intégrée et des circuits de commande de cette RAM. Il est alors possible, en cas de détection de la chute de la tension d'alimentation Vcc, de sauvegarder des informations dans ces 32 octets. La consommation est de 8mA pour Vcc standby=5,25volts.

• RAM ENABLE (RE). Cette entrée compatible TTL valide, par le niveau logique 1 la RAM de 128 octets intégrée dans le 6802. Un niveau 0 sur cette entrée met la RAM hors circuit.

• XTAL et EXTAL. L'oscillateur interne au 6802 peut être piloté par un quartz fonctionnant selon le mode «fondamental-résonance série ». Un diviseur par 4 a été intégré dans le 6802 de sorte que pour une horloge à 1MHZ il est possible d'utiliser un quartz à 4MHZ.

Remarque : Le 6802 peut remplacer le 6800 dans toutes les configurations où ce dernier a été utilisé, avec l'avantage de l'horloge intégrée et des 128 octets de RAM qui pallie l'absence de registres auxiliaires du 6800. Mais le 6802 peut également être associé à un autre circuit LSI et constituer un système minimal à 2 boîtiers. Ce circuit est le 6846 qui peut d'ailleurs être utilisé avec le 6800. C'est une mémoire de 2K octets associée à un port d'E/S de 8 bits et à un temporisateur programmable.

Tous les circuits de la famille du 6800, excepté le circuit horloge qui n'est plus nécessaire, sont utilisables avec le 6802 dont le jeu d'instructions est rigoureusement celui du 6800.

2.1.8 Le microprocesseur 6809 de Motorola (« Matériel »)

Le 6809 de Motorola est une version considérablement améliorée du 6800 et qui bien entendu remédie aux points faibles de ce dernier. De plus, le circuit horloge est intégré dans le 6809. Sa principale caractéristique est un jeu d'instructions très puissant, avec un nombre non négligeable d'instructions portant sur des données de 16 bits. Ce microprocesseur accepte tous les circuits de la famille 6800 ; toutefois la compatibilité logicielle n'est pas totale : le 6809 possède toutes les instructions du 6800 mais un programme objet du 6800 ne peut pas directement fonctionner sur une carte 6809 car les codes binaires des instructions sont différents. Pour qu'il fonctionne, le programme source correspondant, écrit en assembleur 6800, doit être réassemblé avec l'assembleur 6809.

La figure 4.9 donne tous les registres du 6809 en représentant en hachuré les registres nouveaux par rapport à ceux du 6800. Nous retrouvons les accumulateurs A et B de 8 bits du 6800 mais ils peuvent être associés pour donner l'accumulateur D de 16 bits. Il existe deux registres d'index de 16 bits X et Y et un registre d'adressage indirect également de 16 bits et désigné par U. Ce registre U et le pointeur de pile S peuvent également être utilisés comme index mais leur manipulation est plus délicate que pour X et Y.

Le compteur ordinal PC est inchangé. Le registre de condition CC contient les bits d'états du 6800, à savoir CARRY désigné par C, OVERFLOW désigné par V, ZÉRO désigné par Z, NEGATE désigné par N, HALF-CARRY désigné par H.

Il contient également le bit d'état ENTIRE, désigné par E et spécifique du 6809, ainsi que deux bits de masque : le bit I relatif à l'interruption IRQ et le bit F relatif à l'interruption FIRQ.

Le 6809 possède en effet un niveau supplémentaire d'interruption matérielle $\overline{IRQ}$ et deux interruptions logicielles supplémentaires SW12 et SW13.

• Bit de masque I

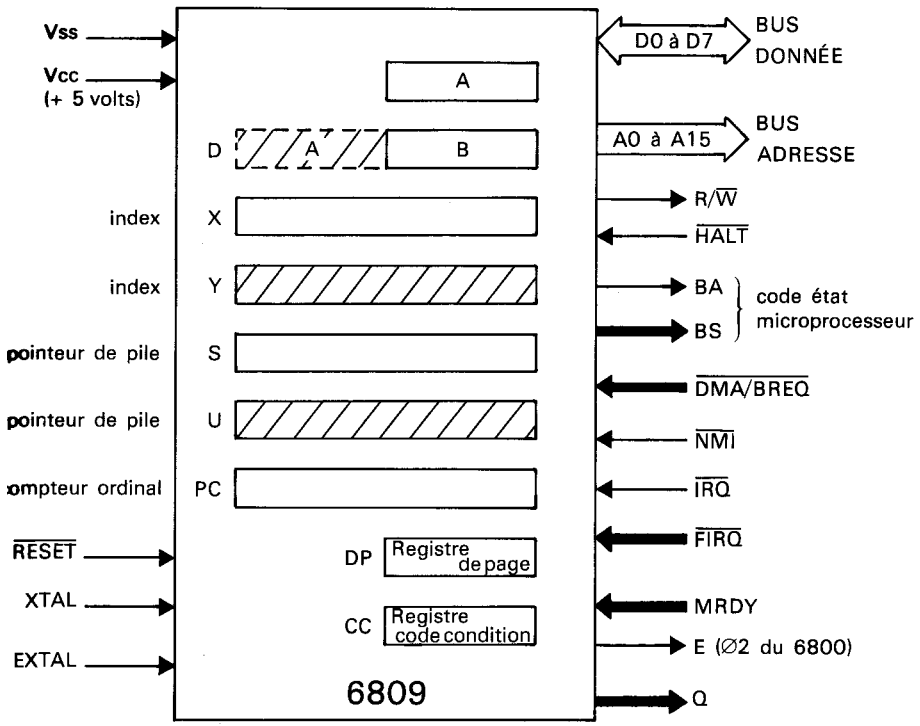
Une demande d'interruption sur l'entrée $\overline{IRQ}$ ne peut être prise en compte par le microprocesseur si I est au niveau 1. Or ce bit I est positionné à 1 :

- soit par la prise en compte d'une demande d'interruption, cette prise en compte résultant de l'activation par le passage au niveau 0 de l'une des entrées interruption suivantes : $\overline{NMI}$, $\overline{FIRQ}$, $\overline{IRQ}$ et $\overline{RESET}$,
- soit par l'interruption logicielle SWI.

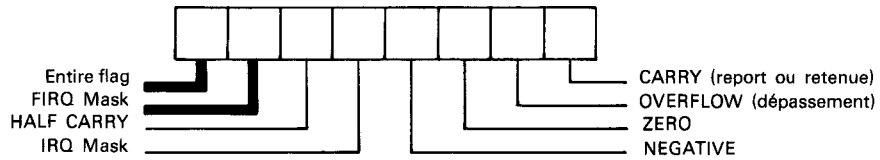
• Bit de masque F

Une demande d'interruption sur l'entrée $\overline{FIRQ}$ ne peut être prise en compte par le microprocesseur si F est au niveau 1. Or ce bit F est positionné à 1 :

- soit par la prise en compte d'une demande d'interruption, cette prise en compte résultant de l'activation par le passage au niveau 0 de l'une des entrées interruption suivantes : $\overline{NMI}$, $\overline{FIRQ}$ et $\overline{RESET}$,
- soit par l'interruption logicielle SWI.



a) *Registres et signaux du 6809*



b) *Registre de conditions du 6809*

NIVEAU D'INTERRUPTION	INTERRUPTION ACTIVÉE PAR	ADRESSES
RESET	niveau bas	FFFE et FFFF
NMI	front descendant	FFFC et FFFD
SWI	l'instruction SWI	FFFA et FFFB
IRQ	niveau bas	FFF8 et FFF9
FIRQ	niveau bas	FFF6 et FFF7
SWI2	l'instruction SWI2	FFF4 et FFF5
SWI3	l'instruction SWI3	FFF2 et FFF3

c) *Les interruptions du 6809*

Fig. 4.9 : Organisation interne et externe du 6809.

• Bit d'état E (ENTIRE)

Ce bit est mis à 1 lorsque tout le contexte machine vient d'être sauvegardé dans la pile, soit les contenus des registres A, B, X, Y, U, S, CC et DP. Il est donc un moyen de distinction entre $\overline{IRQ}$ et $\overline{FIRQ}$ puisque cette dernière n'entraîne la sauvegarde que du compteur ordinal et du registre de conditions.

Le 6809 possède encore un autre registre ; le registre de page DP (direct page register). Ce registre permet d'étendre l'adressage direct sur 8 bits à tout le champ mémoire la partie adresse de l'instruction contient l'octet poids faible de l'adresse, tandis que l'octet poids fort est le contenu du registre de page. Ce mode d'adressage revient à diviser les 64 K octets du champ mémoire en 256 pages de 256 octets chacune. Le registre DP indique alors le numéro de la page en hexadécimal, soit de 00 à FF.

Étudions maintenant les signaux du 6809. Ce sont :

- les 16 fils du bus adresse A0 à Al 5.
- les 8 fils du bus donnée D0 à D7.
- READ/WRITE : R/$\overline{W}$ lecture/écriture.
- $\overline{RESET}$: C'est une entrée de type trigger de Schmitt ; aussi un simple circuit RC peut être utilisé pour cette fonction de remise à l'état initial.
- $\overline{HALT}$: Un niveau bas sur cette entrée arrête le microprocesseur à la fin de l'instruction en cours, sans perdre les données. Les bus adresse et donnée passent à l'état haute impédance, ce qui est indiqué par l'état haut du signal BA. Le signal BS passe également à l'état 1. Dans cet état $\overline{HALT}$ le microprocesseur :

- ne répond pas aux demandes d'interruption $\overline{FIRQ}$ et $\overline{IRQ}$,
- mémorise l'activation des signaux $\overline{NMI}$ et $\overline{RESET}$ pour une prise en compte dès que l'état $\overline{HALT}$ cesse,
- accepte toute demande d'accès direct mémoire ou de cycle de rafraîchissement pour les mémoires dynamiques, sur l'entrée $\overline{DMA/BREQ}$,

• $\overline{DMA/BREQ}$ entrée demande d'accès direct mémoire ou de cycle de rafraîchissement dans le cas d'une mémoire dynamique. Le microprocesseur répond en mettant à 1 les signaux BA et BS.

- BUS AVAILABLE (BA) et BUS STATUS (BS)

Ces deux signaux sont utilisés pour coder sur deux bits, l'état du microprocesseur comme le montre le tableau ci-dessous.

• XTAL et EXTAL. Ces entrées connectent l'oscillateur du 6809, intégré dans ce composant, à un quartz externe fonctionnant dans le mode résonance parallèle. Toutefois il est possible d'utiliser une horloge externe TTL au lieu de l'oscillateur du 6809. Dans ce cas, l'entrée XTAL est mise à la masse et l'horloge externe est appliquée à l'entrée EXTAL, sa fréquence devant être quatre fois supérieure à celle de fonctionnement du 6809.

Décodage de BA et BS

BA	BS	ÉTAT DU MICROPROCESSEUR
0	0	**Fonctionnement normal du 6809, acquisition de l'adresse de**
0	1	branchement relative à une interruption $\overline{\text{RESET}}$, $\overline{\text{NMI}}$, $\overline{\text{FIRQ}}$, $\overline{\text{IRQ}}$, SWI, SW12 ou SW13.
1	0	Suite à l'exécution de l'instruction SYNC, le microprocesseur attend une interruption.
1	1	État HALT ou bus adresse et donnée à l'état haute impédance.

• E et Q sont deux horloges. E est semblable au signal ∅2 du 6800 et Q est une nouvelle horloge déphasée de 270° par rapport à E. Les données sont verrouillées sur le front descendant de E et les adresses sont validées sur le front montant de Q.

• MEMORY READY (MRDY). Cette entrée permet l'utilisation de mémoires plus lentes en allongeant la durée à l'état haut de l'horloge E et donc la durée à l'état bas de Q. L'allongement des horloges ne peut dépasser 10 microsecondes et doit être un multiple de T/4 si T est la période de l'horloge de fonctionnement du 6809 ; de plus, il ne se fait pas si le signal VMA est au niveau zéro.

• Les lignes d'interruption $\overline{\text{NMI}}$, $\overline{\text{FIRQ}}$, $\overline{\text{IRQ}}$.

2.1.9 Le logiciel du 6800, 6802 et 6809 de Motorola

Les microprocesseurs 6800 et 6802 ont exactement les mêmes modes d'adressage et le même jeu d'instructions. Le fait qu'ils possèdent deux accumulateurs A et B double un nombre important d'instructions. Pour éviter deux instructions identiques, l'une portant sur A, l'autre sur B, nous représentons l'accumulateur par la lettre Y, exemple LDA Y qui sous-entend les deux instructions LDA A et LDA B. Pour le 6809, ces deux instructions deviennent LDA et LDB. Bien que le suffixe H soit admis par l'assembleur 6800 pour la notation hexadécimale, c'est toujours le suffixe $ (dollar) qui est utilisé dans la documentation Motorola.

Les appellations de certains modes d'adressage par Motorola prêtent à confusion. Ainsi l'adressage direct classique, universellement utilisé, portent deux appellations chez Motorola :

lorsque ce mode d'adressage a une adresse de 8 bits, ce qui est appelé adressage page-zéro, il est appelé *adressage direct* par Motorola

lorsque ce mode d'adressage a une adresse de 16 bits, il est appelé *adressage étendu* par Motorola.

Le 6809 a des modes d'adressage plus nombreux que le 6800. Ainsi l'adressage direct sur 8 bits du 6800 est obligatoirement restreint à la page zéro, c'est-à-dire au champ mémoire compris entre les adresses $ 0000 et $ 00FF. Pour le 6809 cet adressage est étendu à n'importe quelle page grâce au registre de page DP qui contient le numéro de la page, c'est-à-dire l'octet poids fort de l'adresse.

Pour plus d'information sur le logiciel de ces microprocesseurs, voir la régférence [1] ;

Chapitre 5 : Les mémoires

12. Constitution d'une mémoire statique

2.1.10 Principes généraux d'une RAM

Dans une mémoire statique, l'information stockée y reste indéfiniment tant que la mémoire est reliée aux alimentations et tant qu'on n'écrit pas autre chose dans la position-mémoire concernée. Les mémoires à semi-conducteurs sont adressables par position - mémoire c'est-à-dire par groupe de 8 bits ou 16 bits. Nous considérerons ici les mémoires orientées vers les microprocesseurs à 8 bits et donc adressables par 8 bits. Nous pouvons considérer ces 8 bits sur une ligne (ligne de mots). Il faut donc un fil d'adresse par ligne. Le registre d'adresse sera suivi d'un décodeur de lignes. Sur la figure 54 nous avons représenté une mémoire de 4 mots de 4 bits. Sur cette figure chaque carré est une case-mémoire contenant un bit. Les 4 cases d'une même ligne forment un mot. Comme il y a 4 mots il suffit de deux fils d'adresses A_0 et A_1. Les sorties des 4 cases d'une même colonne sont réunies en OU-CABLE, ce qui est permis par le circuit de sortie de chaque case qui est du type «collecteur ouvert» ou du type « logique trois états ». Ces deux types permettent de réunir les quatre sorties *puisque une seule de ces quatre sorties sera validée par l'adressage.*

Les entrées sont réunies également ce qui ne pose pas de problème, mais là encore une seule case par colonne est validée par la ligne adresse. Sur la figure 5.4, nous n'avons pas représenté le circuit de commande lecture/écriture par souci de clarté. Notre très modeste mémoire peut se représenter sous la forme d'une « boîte noire » ou d'un module de 4 mots de 4 bits comme sur la figure 5.2.

Si nous voulons constituer une mémoire de 4 mots de 8 bits à partir de notre module il nous suffira de prendre deux modules, de réunir leurs lignes adresses, puisque le nombre de mots reste 4 mais chaque ligne valide 8 bits. Le signal L/E est unique pour les deux modules. Par contre les sorties et les entrées sont doublées (fig. 5.3).

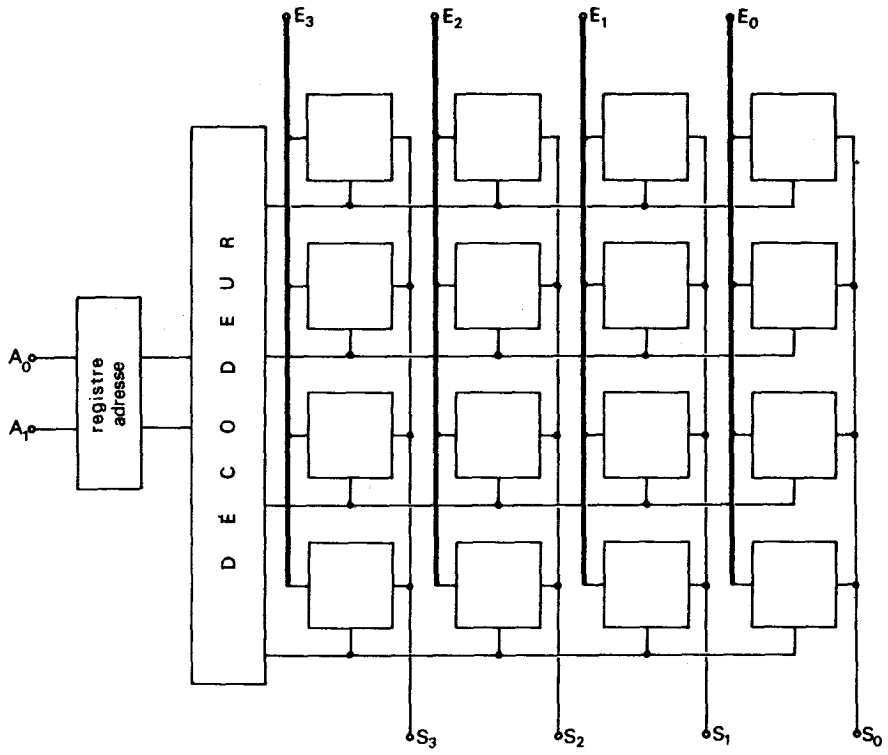


Fig. 5.1. ***Mémoire de 4 mots de 4 bits***

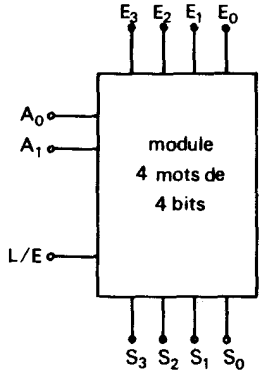


Fig. 5.2. ***Représentation de la mémoire de la figure 5.1***

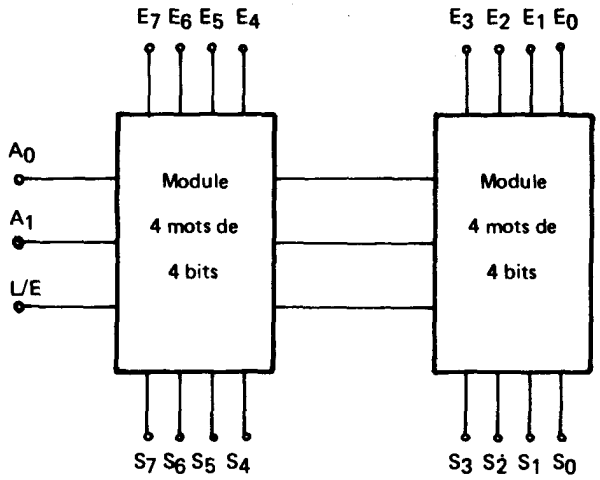
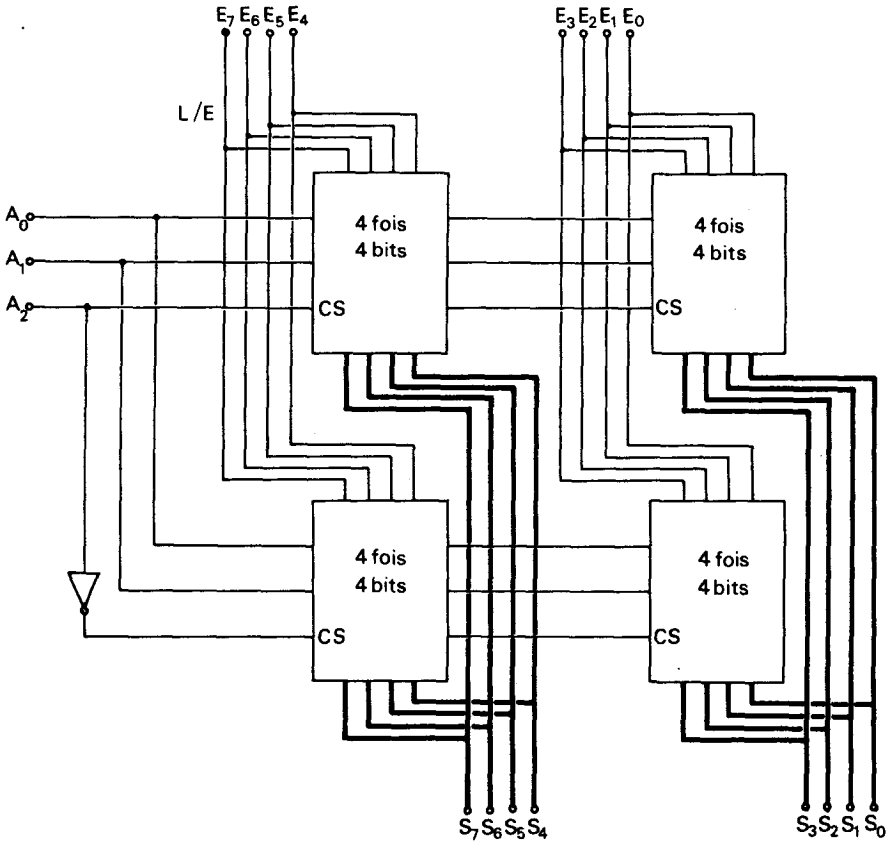


Fig. 5.3. - ***Constitution d'une mémoire de 4 mots de 8 bits***



Fig, 5.4. - ***Constitution d'une mémoire de 8 mots de 8 bits***

De la même façon, nous pourrions obtenir avec trois modules une mémoire de 4 mots de 12 bits. Comment constituer maintenant une mémoire de 8 mots de 8 bits? Cette fois il nous faut deux fois notre mémoire de 4 mots de 4 bits. Mais les 2 bits d'adresse ne suffisent plus. Il faut en effet un troisième bit A_2 qui ira valider un mot, soit dans les deux modules supérieurs (mémoire de 4 mots de 8 bits), soit dans les deux modules inférieurs. Aussi chaque module doit posséder une entrée supplémentaire qui permettra de valider le module ou de l'inhiber, ce qui revient à sélectionner les modules, d'où la désignation de cette entrée supplémentaire CHIP SELECT soit CS (sélection de module). Cette sélection se fera par A_2. ou A_2. Elle est aussi désignée par CE (CHIP ENABLE).

Notre mémoire de 8 mots de 8 bits (fig. 5.4) nous a nécessité un décodage très simple $\overline{A_2}$ et A_2. Si nous avions voulu faire une mémoire de 16 mots de 8 bits avec nos modules il nous aurait fallu faire le décodage pour A_2, A_3 ce qui aurait permis de valider 1 rangée parmi 4. Ce décodage est inutile si nous utilisons un module de 16 mots de 1 bit, donc de même capacité que notre module initial. En effet dans ce cas nous disposons des quatre lignes d'adresses qu'il suffit de réunir à tous les modules. Chacun de ces modules délivre un bit en sortie (fig. 5.5).

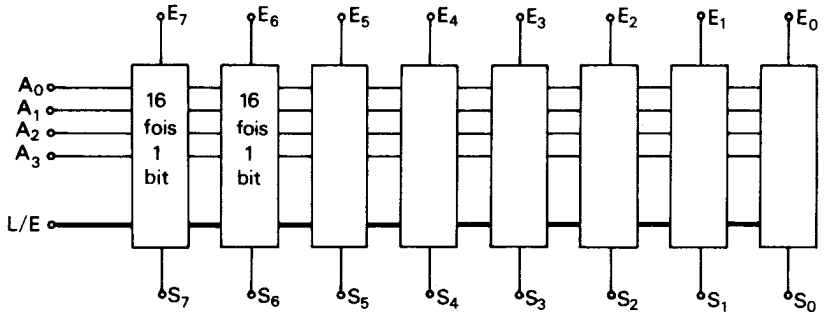


Fig. 5.5. - Constitution *d'une mémoire de 16 mots de 8 bits*

Comment transformer notre module 4 mots de 4 bits en un module 16 mots de 1 bit? Il suffit de sélectionner un des 4 mots de sortie à partir des deux bits d'adresse A_2, et A_3, conformément à la figure 5.6.

Cette structure a également l'avantage de ne nécessiter qu'un seul amplificateur de lecture, ce qui réduit la consommation de la mémoire. Toutefois l'utilisateur n'a pas à faire la distinction entre les lignes d'adresse horizontales et les lignes d'adresse verticales.

Certaines mémoires ont un décodage interne pour l'adjonction de bits d'adresse comme nous en avons vu la nécessité. Ces bits d'adresse devront agir sur les CHIP SELECT, les appellations seront CS_1, CS_2, CS_3, par exemple ou le complément de ces signaux. Ainsi la réalisation d'une mémoire de 16 mots de 8 bits avec deux signaux de sélection de chips CS_1, et CS_2, serait conforme à la figure 5.7. Dans ce cas chaque module ne sera validé que si un niveau 0 est disponible sur chacun des signaux CS_1, et CS_2. Compte tenu du câblage que nous avons réalisé cela sera le cas si $A_2 = 0$ et $A_3 = 0$ pour la rangée supérieure, ce qui correspond aux adresses binaires allant de 0000 à 0011 soit de 0 à 3. Pour la seconde rangée cela sera le cas si $A_2 = 1$, compte tenu de l'inverseur inséré entre A_2 et CS_1, et $A_3 = 0$, ce qui correspond aux adresses binaires allant de 0100 à 0111 soit de 4 à 7. Même raisonnement pour les autres rangées. Nous voyons que seuls des inverseurs sont nécessaires avec ce type de décodage.

Le dernier point à étudier est l'adaptation de la mémoire au bus de données bi-directionnel. En effet une mémoire a ses entrées et sorties distinctes. Il faudra pouvoir soit connecter le bus données aux entrées de la mémoire, les sorties étant déconnectées (haute impédance), soit connecter les sorties de la mémoire sur le même bus données, les entrées étant déconnectées. Pour cela nous utiliserons des circuits spéciaux prévus par chaque constructeur de microprocesseur.

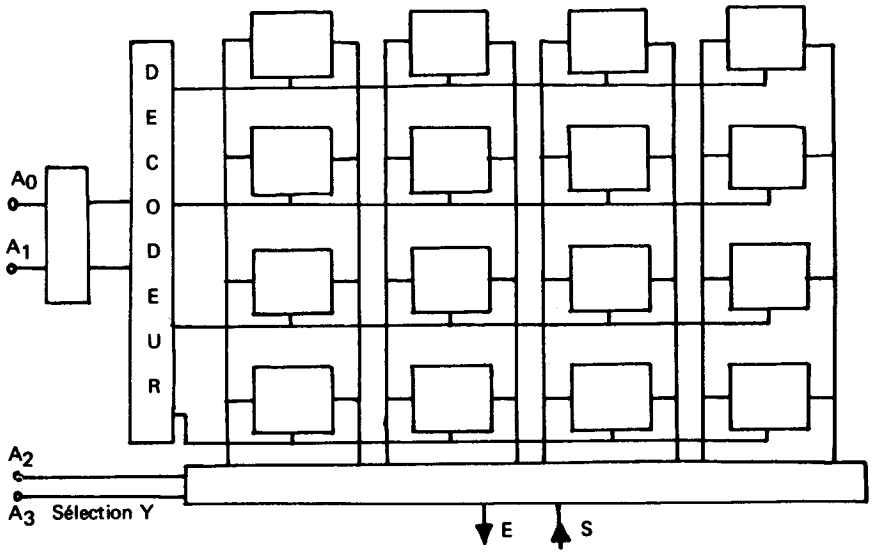


Fig. 5.6. - Sélections *horizontale et verticale du bit*

2.1.11 Principes généraux d'une ROM

La constitution d'une mémoire ROM ou PROM ou REPROM relève des mêmes principes que pour une RAM. Toutefois cette mémoire ne peut être que lue. Donc, seules les sorties sont connectées au bus de données.

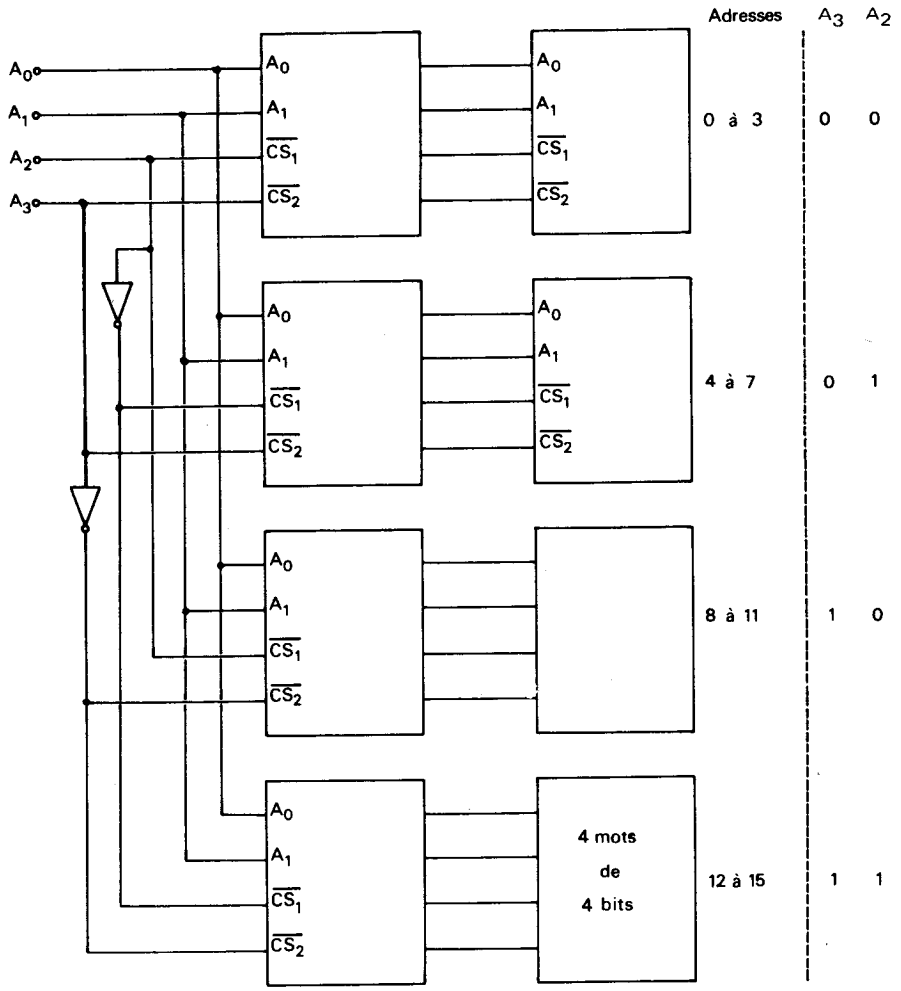


Fig. 5.7. - Utilisation de plusieurs « chip select »

13. Constitution d'une mémoire dynamique

L'élément de base d'une cellule mémoire dynamique (fig. 5.8) est la capacité grille-source d'un MOS T_1 aux bornes de laquelle est emmagasinée une tension logique « 1 » ou « 0 », à l'aide d'un commutateur T_2 conducteur pendant le temps de charge, bloqué ensuite. Si la tension V_{GS} est de polarité et d'amplitude convenables le MOS T1, sera saturé, sinon il sera bloqué.

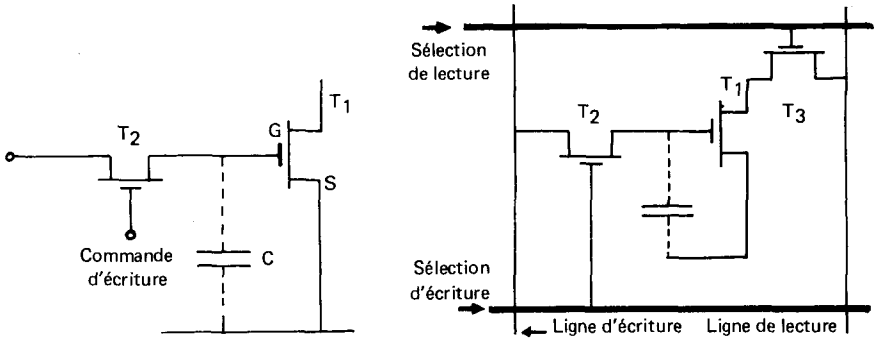


Fig. 5.9. : Cellule mémoire dynamique

Pour lire l'état bloqué ou saturé du MOS T_1 on aura recours à un MOS T_3, qui sera conducteur lors d'une lecture. Nous arrivons ainsi à trois MOS. Pour une cellule il y aura :

• une ligne de sélection de lecture,

• une ligne de sélection d'écriture,

• une ligne d'entrée des données (pour l'écriture),

• une ligne de sortie des données (pour la lecture).

D'autres structures de cellules MOS existent mais toutes sont basées sur le stockage d'une tension aux bornes d'une capacité. Or la tension aux bornes de la capacité ne peut pas se maintenir indéfiniment pour deux raisons :

• la capacité a une résistance de fuite non infinie,

• la grille du MOS se comporte comme un générateur de courant constant, celui-ci traversant la capacité C et donc modifiant la valeur stockée à ses bornes.

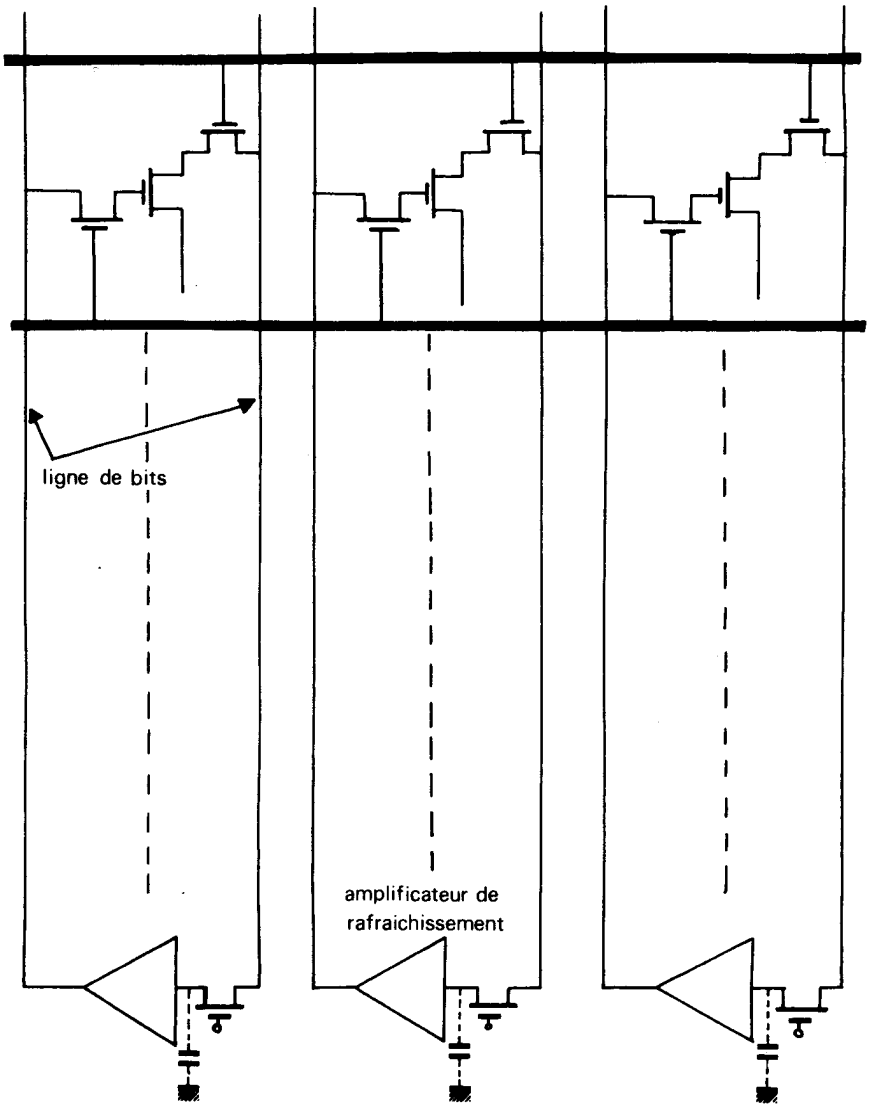


Fig.5.9. Rafraîchissement d'une mémoire dynamique

Pour rafraîchir toute la mémoire il faudra adresser et rafraîchir successivement chacune des lignes. Il en résulte que les entrées adresse de la mémoire recevront :

• soit les fils adresse de sélection de lignes en provenance du bus adresse;

• soit les fils adresse de sélection de lignes en provenance d'un circuit spécial de rafraîchissement ayant sa propre horloge ;

D'où la nécessité d'un multiplexage des fils d'adresse, le multiplexeur étant commandé par le rafraîchissement ou l'absence de rafraîchissement. Mais il y a rarement autant de lignes à rafraîchir qu'il y a d'adresses décodées comme le montre la figure ci-dessus (fig. 5.9.) : quatre lignes à rafraîchir, seize adresses décodées.

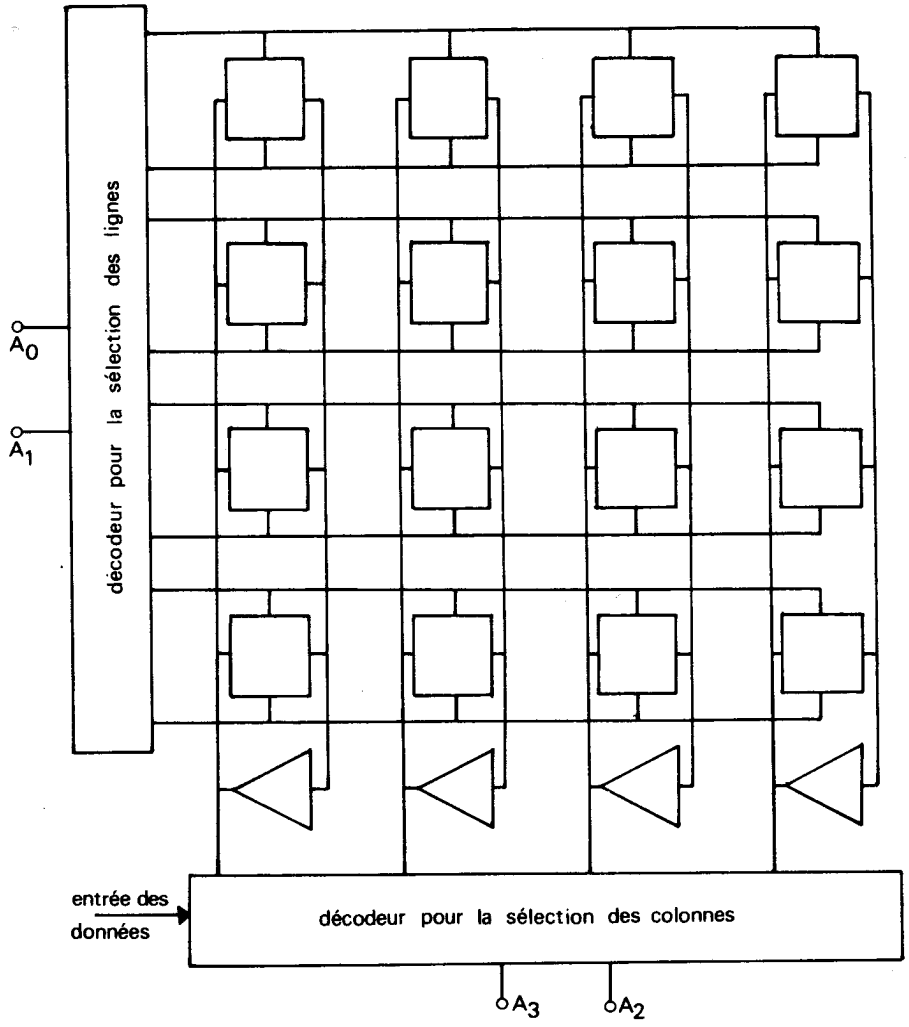


Fig. 5.10. ***Mémoire de 16 mots de 1 bit***

La figure 5.10. est une mémoire de 16 mots de 1 bit mais dont l'organisation interne est une mémoire de 4 mots de 4 bits. Ce qui veut dire qu'il y a 2 fils d'adresse pour la sélection des lignes (sélection X) et 2 fils d'adresse pour la sélection des colonnes (sélection Y). Seules les quatre lignes devront être rafraîchies, ce qui correspond aux fils d'adresse de sélection X.

Notre mémoire de 16 mots constitue un module. Pour faire un mot de 8 bits il faudra 8 modules juxtaposés. Mais s'il y a quatre rangées de modules il faudra deux fils d'adresse supplémentaires et ces deux fils devront être pris en compte pour le rafraîchissement. Ainsi à titre d'exemple nous donnons l'organisation d'une mémoire dynamique de 16 K mots de 8 bits à partir de modules mémoires de 4 K mots de 1 bit mais organisés en 64 mots de 64 bits (fig. 5.11).

Pour les 64 mots il faut 6 bits d'adresse A_0 à A_5 par exemple. Pour les 16 K mots il faut 4 rangées de modules donc deux bits d'adresse A_{12} et A_{13}, les bits A_6 à A_{11} étant réservés à la sélection Y de chaque module. Pour conclure et pour simplifier nous dirons que pour effectuer un rafraîchissement il faut faire une lecture en inhibant les sorties de la mémoire, et il faut sélectionner :

• *une ligne,* c'est-à-dire un mot, à l'intérieur de chaque module, donc envoyer une adresse sur les fils de sélection X;

• *et une rangée de modules* à l'intérieur de la mémoire totale, donc activer une et une seule des sorties du décodeur de rangées de modules ce qui revient à envoyer une adresse sur les bits correspondants.

En général, un rafraîchissement doit être fait toutes les 2 milli-secondes. Les constructeurs de mémoire, ou de microprocesseurs proposent souvent un circuit intégré réalisant la fonction de rafraîchissement. Dans l'exemple que nous avons pris, nous avons utilisé deux « buffers » unidirectionnels pour le raccordement du bus bidirectionnel aux entrées de la mémoire et aux sorties, ces dernières se faisant sur des fils différents de ceux des entrées.

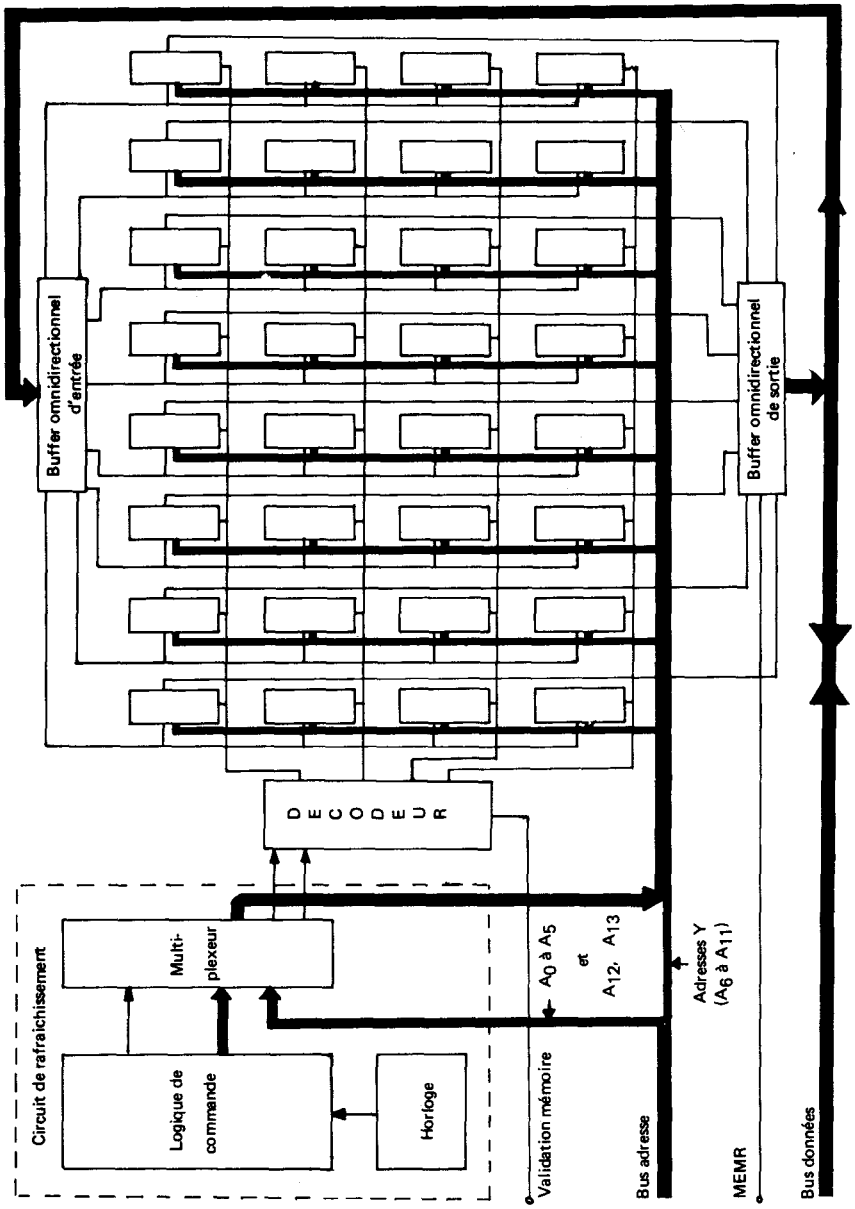


Fig.5.11 : Mémoire dynamique RAM de 16 K mots de 8 bits

Chapitre 6 : L'interface parallèle programmable

14. Modes de transfert des données : Mode programmé

Les transferts sont décidés par le microprocesseur et correspondent à l'exécution d'une instruction d'entrée (lecture) ou de sortie (écriture) placée dans le programme. Pour faciliter l'interfaçage les constructeurs de microprocesseurs proposent dans leur catalogue un circuit d'interface pour la transmission parallèle des informations et un autre circuit d'interface pour la transmission série. Le circuit d'interface pour les transmissions parallèles comporte souvent un registre tampon par périphérique (8 bits), de sorte que les transferts E/S se ramènent tout simplement à une lecture ou une écriture de ce registre appelé « Port ». Pour cela une adresse est affectée à chaque port, et envoyée par le bus adresse.

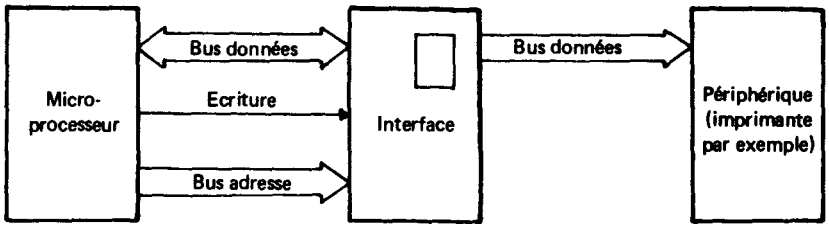


Fig. 6.1. Écriture de données en mode programmé

• *Écriture. - Le* microprocesseur envoie l'ordre d'écriture et les données par l'intermédiaire du bus données. Celles-ci sont mémorisées dans le registre tampon, et deviennent disponibles à la sortie du registre tampon même si elles ne sont plus présentes sur le bus. Le périphérique les enregistre alors (fig. 6.1).

• *Lecture. - Le* périphérique doit avoir des informations stables et permanentes qu'il envoie au registre tampon. En effet celui-ci ne mémorise pas les données provenant du périphérique. Il se contente de les mettre à la disposition du microprocesseur. Lors de l'exécution d'une lecture de ce périphérique, le microprocesseur vient lire les données et les envoie en mémoire (fig. 6.2).

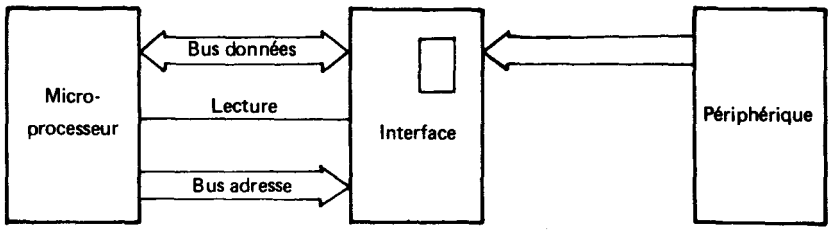


Fig. 6.2 : Lecture de données en mode programmé

Pour s'adapter au plus grand nombre possible d'applications le circuit interface délivre 16 à 24 fils de sortie selon les constructeurs. Et ces fils peuvent servir soit d'entrée, soit de sortie et

ceci selon un certain nombre de choix possibles : par groupe de 12, de 8 ou de 4, selon n'importe quelle configuration pour un groupe donné ou même pour chacun des fils de sortie comme c'est le cas pour l'interface Motorola du 6800. Ce choix sera fait par le programmeur lors de la rédaction de son programme et sera spécifique de l'application. Comment alors informer le microprocesseur de ce choix? Nous le ferons en associant au registre données de l'interface un registre de commande qui mémorisera le code binaire relatif à la configuration choisie et qui lui sera envoyé par le microprocesseur lors de l'exécution des premières instructions du sous-programme affecté au périphérique. Ce code binaire est généralement un mot de 8 bits *(Control Word)*. Rappelons nous en effet que tout transfert E/S nécessite l'exécution d'un sous-programme spécifique du périphérique, ce sous-programme étant appelé lors de l'exécution du programme principal.

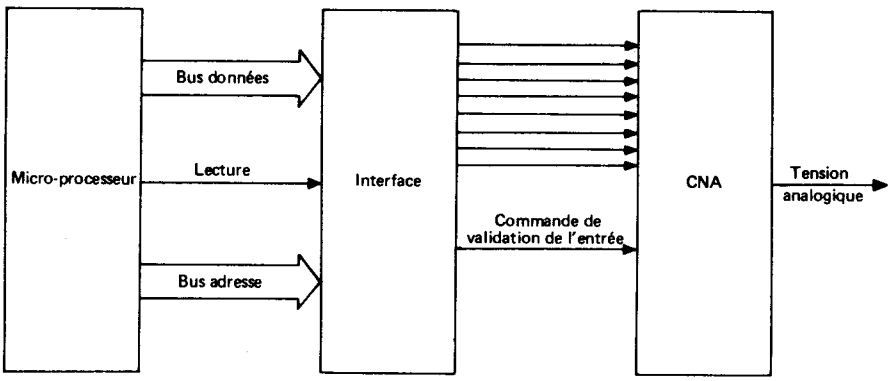


Fig. 6.3 : Commande d'un convertisseur numérique/analogique en mode programmé

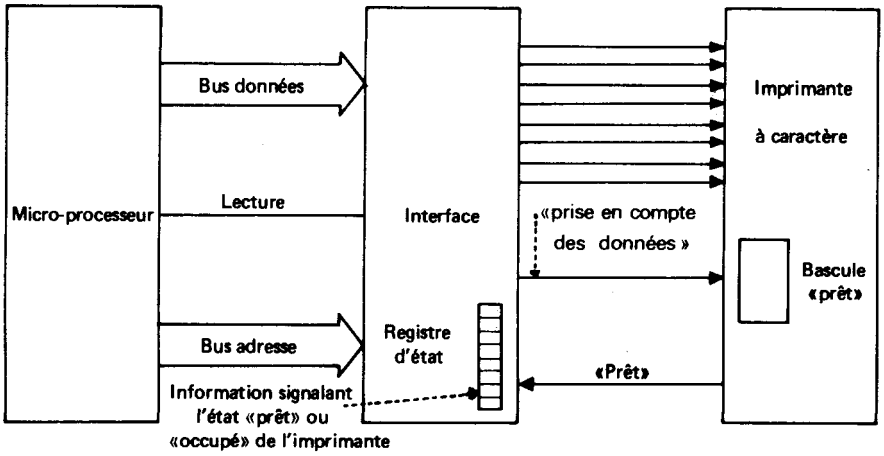


Fig. 6.4 : Commande d'une imprimante en mode programmé

15. Architecture d'un interface

2.1.12 Signaux circulant entre l'interface et le périphérique d'un interface

Quatre types de signaux peuvent circuler entre l'interface programmable et un périphérique, la présence simultanée de ces quatre types de signaux étant d'ailleurs possible mais peu fréquente :

• des *données* transmises généralement par 8 fils parallèles puisque les périphériques associés à un microprocesseur de 8 bits fonctionnent le plus souvent par caractère de 8 bits;

• des *signaux de commande : ils* seront déterminés par l'utilisateur pour s'adapter, tant au périphérique qu'à son mode d'exploitation;

• des *signaux d'état* transmettant des informations comme « périphérique prêt », «.périphérique ou circuit externe ayant des données à transmettre », ceci dans le cas des modes non interruptibles;

• des *signaux de dialogue* prévus dans le circuit interface sur des broches précises pour que d'une part les données soient transmises à un organe récepteur prêt à les recevoir, et que d'autre part l'organe émetteur soit informé que les données sont bien parvenues à destination.

Les trois premiers types de signaux seront choisis par l'utilisateur parmi les broches E/S disponibles sur l'interface, la désignation E/S n'étant pas limitée au sens strict de données E/S.

2.1.13 Signaux circulant entre l'interface et le processeur

Ces signaux peuvent être

• des *données* par le bus données qui transportera les données proprement dites et les signaux d'état.

• une *adresse* par le bus adresse

- deux ou trois fils pour la sélection d'un port et à l'intérieur de ce port d'un registre (données ou commande);

- un fil pour la sélection de l'interface qui contient à cet effet une broche « sélection de boîtier » (chip select).

• des *demandes d'interruption.*

• des *commandes particulières*

- remise à zéro des registres de l'interface;
- inhibition des interruptions (sur certains microprocesseurs).

2.1.14 Tableau récapitulatif

Le tableau suivant donne les principaux signaux d'un interface (fig. 6.5).

TYPE DE TRANSFERT	OPERATIONS ET SIGNAUX D'UN TRANSFERT E/S
Transfert D'entrée	1. Le périphérique envoie le signal « données prêtes ». 2. L'interface répond en envoyant le signal « donnés reçues».
Transfert de sortie	1 , Le microprocesseur teste le bit d'état « périphérique prêt ». 2. L'interface dépose les données sur le bus. 3. L'interface envoie le signal « données prêtes » ou « prise en compte des données ». 4. Le périphérique répond en envoyant à l'interface le signal « données reçues ».

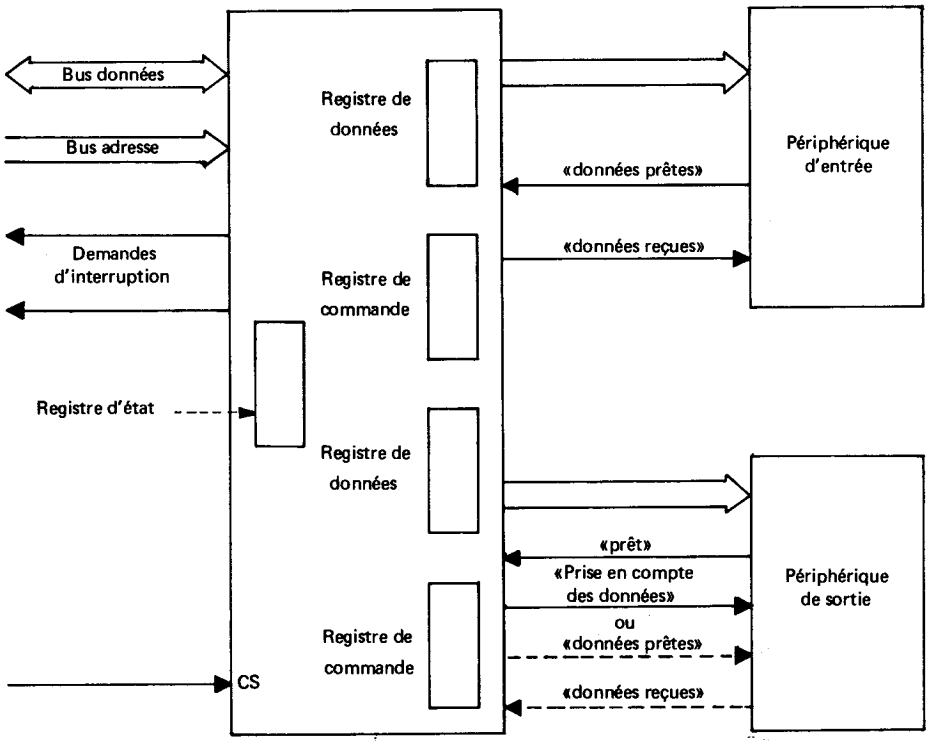


Fig.6.5 . - Signaux usuels d'un interface

Les signaux nécessaires pour réaliser un transfert entrée/sortie dépendent du périphérique et du type de transfert entrée ou sortie.

2.1.15 Programmation d'un interface.

Un interface programmable s'adaptera à un périphérique donné par une programmation de l'interface. Cette programmation comprend généralement trois fonctions distinctes :

- Remise à zéro des registres de l'interface.

- Affectation d'un sens de transfert aux broches d'E/S de l'interface en fonction du péri phérique et de son fonctionnement. Rappelons que ces broches E/S seront utilisées

  • soit comme ligne de transmission de données;

  • soit comme signaux d'état informant l'interface de l'état « prêt n ou « occupé » di

  périphérique;

  • soit comme signaux de commande du périphérique;

  • soit comme signaux de dialogue.

- Initialisation à « 0 » ou à « 1 » des broches E/S définies par l'utilisateur comme signaux d'état ou signaux de commande.

La remise à zéro se fait généralement par l'envoi d'une impulsion sur une entré spéciale souvent désignée par RESET. L'affectation des sens du transfert, de même que l'initialisation

à zéro se fait par l'écriture d'un mot binaire dans un registre de l'interface Là encore l'architecture de ces deux mots dépend du type de microprocesseur; il est alors indispensable de consulter le manuel du constructeur.

16. Étude de l'interface parallèle programmable du 8080 A : (circuit 8255 ou 8255 A)

2.1.16 Architecture générale

Cet interface comprend trois registres de 8 bits, A, B et C (appelés « PORTS » et un registre de commande. La sélection d'un registre parmi ces quatre se fait à l'aide des bits d'adresse Ao et A_1.

A,	Ao	Partie adressée
0	0	port A
0	1	port B
1	0	port C
1	1	Registre de commande

—

Supposons l'entrée CS du 8255 reliée au bit A_3 du bus d'adresse. Dans ce cas nous avons:

adresse du port A :	11110100	F4
adresse du port B :	11110101	F5
adresse du port C :	11110110	F6
adresse du registre de commande :	11110111	F7

Ces adresses sont celles de l'un des 8255 du KIT 8080 A. Une impulsion RESET remet à zéro le contenu de tous les registres de l'interface.

2.1.17 Fonctionnement du 8255 ou du 8255 A en mode 0

Dans ce mode, appelé mode 0 du 8255, ce circuit possède 24 E/S réparties en quatre groupes. Le sens de transfert des données est impérativement te meure a l'intérieur de chacun des quatre groupes qui sont

- le port A 8 E/S ,
- le port B 8 E/S
- la partie inférieure du port C 4 E/S (port C lower)
- la partie supérieure

du port C 4 E/S (port C upper).

Selon l'application souhaitée chaque groupe devra être déclaré en entrée ou en sortie : c'est la programmation des E/S du 8255.

Pour chacun des quatre groupes une E/S peut être utilisée comme donnée proprement dite ou comme signal d'état. En plus de cette possibilité les deux groupes du port C permettent d'utiliser chaque E/S en signal de commande pour générer une impulsion positive ou négative.

Dans ce mode les données de sortie sont mémorisées dans le 8255. Elles resteront donc à leurs valeurs dans le périphérique tant qu'elles ne seront pas changées dans le 8255. II n'en est pas de même des entrées : celles-ci ne sont pas mémorisées. Il importe donc de les lire lorsqu'elles sont valides.

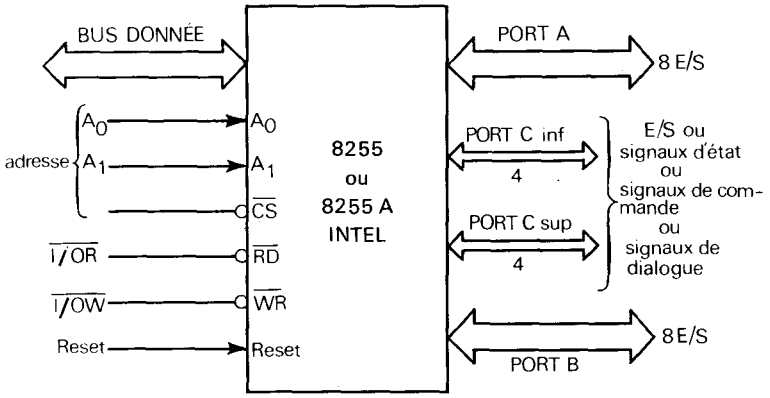
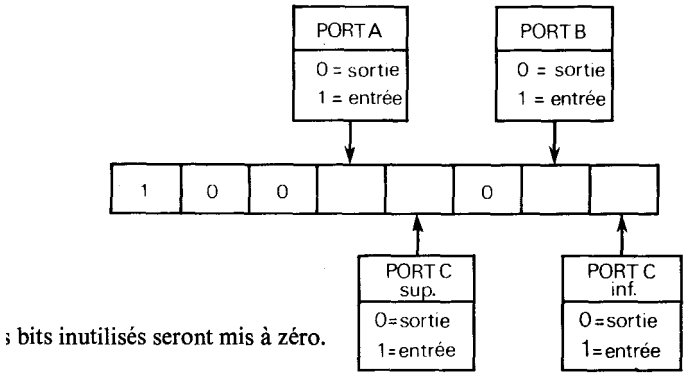


Fig. 6.6 : L'interface parallèle programmable du 8080 A de INTEL

2.1.18 Mot de commande

Celui-ci est déterminé à partir du synoptique ci-dessous



; bits inutilisés seront mis à zéro.

Fig.6.7: Détermination de mot de commande en mode 0

2.1.19 Programmation des E/S du 8255 ou 8255 A.

a) déterminer le mot de commande en fonction de l'application souhaitée et artir du tableau de la figure6.7.

b) mettre le mot de commande ainsi déterminé dans l'accumulateur à l'aide de truction MVI A. Cette instruction permet de transférer directement une donnée dans l'accumulateur. Ainsi si le mot de commande est 92 l'instruction sera : MVI A, 92.

c) transférer le mot de commande de l'accumulateur dans le registre de com~de du 8255 soit donc à l'adresse F7 comme nous l'avons vu. C'est l'instruction d'écriture périphérique (instruction OUT) qui est chargée de cette opération. D'où l'instruction : OUT F7.

d) Exemple

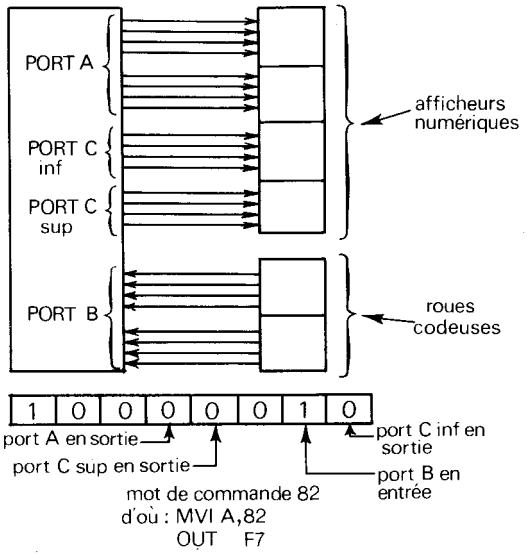


Fig. 6.8 : Commande d'afficheur numérique

Différentes instructions du logiciel du 8080 A ou du 8085 A de INTEL

Type d'instruction	Instruction	Opération accomplie	rég. concern. par r ou rp	Indicateurs affectés
incrémentation et décrémentation	INR M INR r DCR M DCR r INX rp DCX rp	(M) + 1 → M (r) + 1 → r (M) − 1 → M (r) − 1 → r (rp) + 1 → rp (rp) − 1 → rp	A, B, C, D, E, H, L A, B, C, D, E, H, L BC, DE, HL, SP BC, DE, HL, SP	Z, S, P, AC Z, S, P, AC Z, S, P, AC Z, S, P, AC aucun aucun
branchement	JMP addr CALL addr RET JC, CC, RC addr JNC, CNC, RNC addr JZ, CZ, RZ addr JNZ, CNZ, RNZ addr JP, CP, RP addr JM, CM, RM addr JPE, CPE, RPE addr JPO, CPO, RPO addr	saut inconditionnel à l'adresse addr appel sous-programme d'adresse addr retour de sous-programme saut, appel SP, retour SP si CY = 1 saut, appel SP, retour SP si CY = 0 saut, appel SP, retour SP si Z = 1 saut, appel SP, retour SP si Z = 0 saut, appel SP, retour SP si signe = 0 saut, appel SP, retour SP si signe = 1 saut, appel SP, retour SP si pair saut, appel SP, retour SP si impair		AUCUN
instructions de décalage	RLC RAL RRC RAR	déc. gauche, CY hors boucle déc. gauche, CY dans boucle déc. droit, CY hors boucle déc. droit, CY dans boucle	VOIR BAS DE LA PAGE	CY
instructions spéciales	EI DI NOP HLT	Interruption autorisée Interruption non autorisée Rien Arrêt microprocesseur		AUCUN

LOGICIEL DU 8080 A OU DU 8085 A DE INTEL

Type d'instruction	Instruction	Opération accomplie	rég. concernés par r ou rp	Indicateurs affectés
lecture périphérique écriture périphérique	IN addr OUT addr	(port) → A (A) → port		AUCUN
lecture mémoire	LDA addr LHLD addr LDAX rp MOV r,M POP rp	(M) → A (M) → HL ((rp)) → A (M) → r ((SP)) → rp	BC, DE A, B, C, D, E, H, L PSW, BC, DE, HL	AUCUN sauf pour POP PSW
écriture mémoire	STA addr SHLD addr STAX rp MOV M, r MVI M data PUSH rp	(A) → M (HL) → M (A) → (rp) (r) → M data → M (rp) → (SP)	BC, DE A, B, C, D, E, H, L PSW, BC, DE, HL	AUCUN
transferts de données	MOV r1, r2 MVI r data LXI rp data PCHL SPHL XCHG XTHL	(r2) → r1 data → r data → rp (HL) → PC (HL) → SP (HL) ↔ (DE) (HL) ↔ ((SP))	A, B, C, D, E, H, L A, B, C, D, E, H, L BC, DE, HL, SP	AUCUN
opérations arithmétiques	ADD M ADD r ADC M ADC r ADI data ACI data DAA SUB M SUB r SBB M SBB r SUI data SBI data DAD rp	(A) + (M) → A (A) + (r) → A (A) + (CY) + (M) → A (A) + (CY) + (r) → A (A) + data → A (A) + (CY) + data → A correction décim. (A) − (M) → A (A) − (r) → A (A) − (CY) − (M) → A (A) − (CY) − (r) → A (A) − data → A (A) − (CY) − data → A (HL) + (rp) → HL	A, B, C, D, E, H, L A, B, C, D, E, H, L A, B, C, D, E, H, L A, B, C, D, E, H, L BC, DE, HL, SP	TOUS CY
fonctions logiques	ANA M ANA r ORA M ORA r XRA M XRA r ANI data ORI data XRI data CMP M CMP r CPI data CMA CMC STC	(A) . (M) → A (A) . (r) → A (A) + (M) → A (A) + (r) → A (A) ⊕ (M) → A (A) ⊕ (r) → A (A) . data → A (A) + data → A A ⊕ data → A (A) − (M) (A) − (r) (A) − data ($\overline{A}$) → A ($\overline{CY}$) → CY 1 → CY	A, B, C, D, E, H, L A, B, C, D, E, H, L A, B, C, D, E, H, L A, B, C, D, E, H, L	Indicateurs remis à zéro CY TOUS CY, AC CY, AC CY, AC CY, AC aucun CY CY

- Pour les opérandes ou les adresses de 16 bits, l'octet poids faible est traité le premier.
- La lettre M dans le code instruction symbolise l'adressage indirect par HL.

INST.	HEXA.	INST.	HEXA.	INST.	HEXA.	INST.	HEXA.
ACI	CE	DCR L	2D	MOV C,M	4E	POP D	D1
ADC A	8F	DCR M	35	MOV D,A	57	POP H	E1
ADC B	88	DCX B	0B	MOV D,B	50	POP PSW	F1
ADC C	89	DCX D	1B	MOV D,C	51	PUSH B	C5
ADC D	8A	DCX H	2B	MOV D,D	52	PUSH D	D5
ADC E	8B	DCX SP	3B	MOV D,E	53	PUSH H	E5
ADC H	8C	DI	F3	MOV D,H	54	PUSH PSW	F5
ADC L	8D	EI	FB	MOV D,M	56	RAL	17
ADC M	8E	HLT	76	MOV E,A	5F	RAR	1F
ADD A	87	IN	DB	MOV E,B	58	RC	D8
ADD B	80	INR A	3C	MOV E,C	59	RET	C9
ADD C	81	INR B	04	MOV E,D	5A	RLC	07
ADD D	82	INR C	0C	MOV E,E	5B	RM	F8
ADD E	83	INR D	14	MOV E,H	5C	RNC	D0
ADD H	84	INR E	1C	MOV E,L	5D	RNZ	C0
ADD L	85	INR H	24	MOV E,M	5E	RP	F0
ADD M	86	INR L	2C	MOV H,A	67	RPE	E8
ADI	C6	INR M	34	MOV H,B	60	RPO	E0
ANA A	A7	INX B	03	MOV H,C	61	RRC	0F
ANA B	A0	INX D	13	MOV H,D	62	RST 0	C7
ANA C	A1	INX H	23	MOV H,E	63	RST 1	CF
ANA D	A2	INX SP	33	MOV H,H	64	RST 2	D7
ANA E	A3	JC	DA	MOV H,L	65	RST 3	DF
ANA H	A4	JM	FA	MOV H,M	66	RST 4	E7
ANA L	A5	JMP	C3	MOV L,A	6F	RST 5	EF
ANA M	A6	JNC	D2	MOV L,B	68	RST 6	F7
ANI	E6	JNZ	C2	MOV L,C	69	RST 7	FF
CALL	CD	JP	F2	MOV L,D	6A	RZ	C8
CC	DC	JPE	EA	MOV L,E	6B	SBB A	9F
CM	FC	JPO	E2	MOV L,H	6C	SBB B	98
CMA	2F	JZ	CA	MOV L,L	6D	SBB C	99
CMC	3F	LDA	3A	MOV L,M	6E	SBB D	9A
CMP A	BF	LDAX B	0A	MOV M,A	77	SBB E	9B
CMP B	B8	LDAX D	1A	MOV M,B	70	SBB H	9C
CMP C	B9	LHLD	2A	MOV M,C	71	SBB L	9D
CMP D	BA	LXI B	01	MOV M,D	72	SBB M	9E
CMP E	BB	LXI D	11	MOV M,E	73	SBI	DE
CMP H	BC	LXI H	21	MOV M,H	74	SHLD	22
CMP L	BD	LXI SP	31	MOV M,L	75	SPHL	F9
CMP M	BE	MOV A,A	7F	MVI A	3E	STA	32
CNC	D4	MOV A,B	78	MVI B	06	STAX B	02
CNZ	C4	MOV A,C	79	MVI C	0E	STAX D	12
CP	F4	MOV A,D	7A	MVI D	16	STC	37
CPE	EC	MOV A,E	7B	MVI E	1E	SUB A	97
CPI	FE	MOV A,H	7C	MVI H	26	SUB B	90
CPO	E4	MOV A,L	7D	MVI L	2E	SUB C	91
CZ	CC	MOV A,M	7E	MVI M	36	SUB D	92
DAA	27	MOV B,A	47	NOP	00	SUB E	93
DAD B	09	MOV B,B	40	ORA A	B7	SUB H	94
DAD D	19	MOV B,C	41	ORA B	B0	SUB L	95
DAD H	29	MOV B,D	42	ORA C	B1	SUB M	96
DAD SP	39	MOV B,E	43	ORA D	B2	SUI	D6
DCR A	3D	MOV B,H	44	ORA E	B3	XCHG	EB
DCR B	05	MOV B,L	45	ORA H	B4	XRA A	AF
DCR C	0D	MOV B,M	46	ORA L	B5	XRA B	A8
DCR D	15	MOV C,A	4F	ORA M	B6	XRA C	A9
DCR E	1D	MOV C,B	48	ORI	F6	XRA D	AA
DCR H	25	MOV C,C	49	OUT	D3	XRA E	AB
		MOV C,D	4A	PCHL	E9	XRA H	AC
		MOV C,E	4B	POP B	C1	XRA L	AD
		MOV C,H	4C			XRA M	AE
		MOV C,L	4D			XRI	EE
						XTHL	E3