I- PRINCIPE DE BASE D'UN ORDINATEUR

En général le mot ordinateur est donné à un élément ou à une machine qui exécute une séquence d'opérations sur des données et ceci d'une façon automatique. En électronique digitale, les ordinateurs manipulent des données qui sont représentées par des groupes de digits binaires. Un groupe de digits correspondant à une donnée s'appelle mot. En réalité ces combinaisons en langage informatique sont réparties comme suite :

Bit : un seul digit.

Nibble : une combinaison de quatre bits.

Octet : une combinaison de huit bits.

Mot : une combinaison de seize bits.

Long mot : une combinaison de trente deux bits.

Comme montré sur la figure-1, un système est constitué de trois parties fondamentales :

- Une unité centrale qui exécute les instructions et traite les données.
- Une mémoire qui stocke les instructions d'un programme et les données correspondantes.
- Une partie entrée/sortie qui permet à l'ordinateur de communiquer avec le monde extérieur.

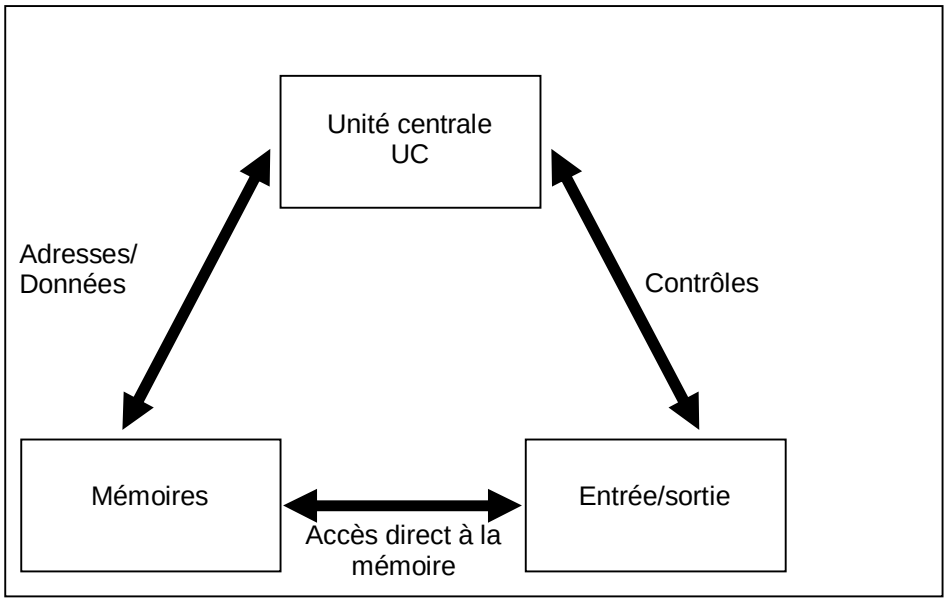


Fig-1- Dans cette structure de base d'un ordinateur, on constate que les différents constituants du système s'interconnectent par des moyens qui véhiculent : adresses, données et contrôles

II- SYSTEME A MICROPROCESSEUR

Dans un tel système l'unité centrale est représentée par le microprocesseur. Comme il est illustré par la figure-2, le microprocesseur communique avec les différentes parties de son système par l'intermédiaire de 3 bus :

- Bus d'adresses
- Bus de données
- Bus de contrôle

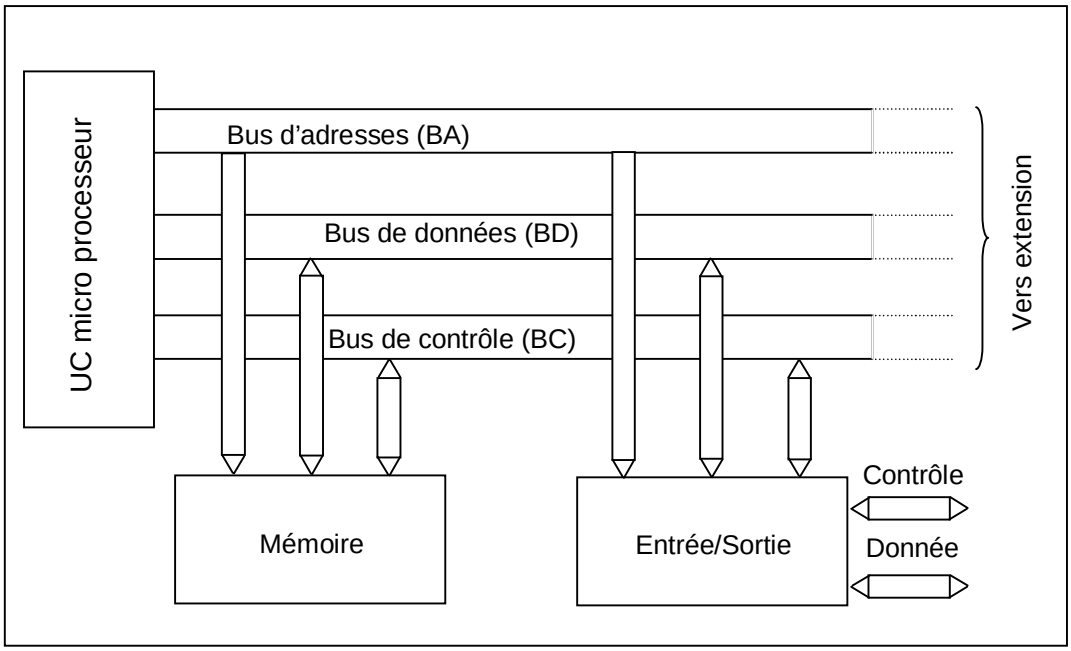


Fig-2- Dans un système à µP les différentes parties qui le constituent communiquent par l'intermédiaire de bus

La caractéristique principale d'un système est la longueur de la combinaison numérique qu'il peut manipuler. D'ailleurs c'est de cette caractéristique qu'il tire son nom :

Système à huit bits : il manipule des octets (bus de donnée de huit lignes).

Système à seize bits : il manipule des mots (bus de donnée de seize lignes).

II-1- MICROPROCESSEUR

Le microprocesseur appelé aussi CPU (Central Processing Unit) représente l'élément cerveau d'un système à microprocesseur. C'est à ce niveau que toutes les opérations sont effectuées, et que suite à un traitement, des décisions seront prises. C'est aussi l'élément qui gère tous les autres blocs du système. Le tableau 1 résume l'historique de quelques microprocesseurs de différence firme de fabrication ainsi que le nombre de transistors qu'ils intègrent.

Tableau-1: chronologies de quelques transistors

(https://fr.wikipedia.org/wiki/Chronologie_des_microprocesseurs)

Date	Nom	Manufacturier	Horloge	Nombre de transistors
1971	4004	Intel	740 kHz	2250
1972	8008	Intel	500 kHz	3500
1974	6800	Motorola	2 MHz	4100
1974	8080	Intel	2 MHz	6000
1974	1802	RCA	3.2 MHz	5000
1974	TMS-1000	Texas Inst.	400 kHz	8000
1975	6502	MOS Techno.	1 MHz	4000
1976	Z80	Zilog	2 MHz	8500
1976	8085	Intel	5 MHz	6500
1976	TMS-9900	Texas Inst.	3.3 MHz	8000
1977	6100	Intersil	4 MHz	20000
1978	8086	Intel	5 MHz	29000
1978	6801	Motorola	-	35000
1979	Z8000	Zilog	-	17500
1979	8088	Intel	5 MHz	29000
1979	6809	Motorola	1 MHz	40000
1979	68000	Motorola	8 MHz	68000

II-1-1 ARCHITECTURE INTERNE D'UN MICROPROCESSEUR

Bien que tous les microprocesseurs soient identiques dans leurs fonctions, ils diffèrent par leurs architectures internes. La figure-3 montre l'architecture interne d'un microprocesseur en générale.

II-1-1-1 UNITE ARITHMETIQUE ET LOGIQUE (U.A.L)

L'unité arithmétique et logique est la partie du processeur où toutes les opérations arithmétiques ou logiques sont effectuées. Le décodage de l'instruction contenue dans le registre RI détermine le type d'opération à effectuer

Nous citons quelques opérations types :

- Addition et Soustraction
- AND, OR, XOR
- Incrémentation (+1) – Décrémentation (-1)
- décalage à gauche ou à droite
- Rotation
- Complémentation

II-1-1-2- UNITE DE CONTROLE

La fonction de l'unité de contrôle est d'ordonner les opérations suivant le décodage de l'instruction présente dans RI. L'unité de contrôle fixe les différentes étapes nécessaires pour l'exécution de l'instruction. En utilisant l'entrée du signal d'horloge, l'unité de contrôle assure la synchronisation et fixe le temps nécessaire pour l'exécution de chaque type d'instruction.

II-1-1-3- Registres spéciaux

En générale un registre spécial est le registre principal dans un microprocesseur ; il est le plus utilisé lors de traitement des opérations. Fréquemment, il joue le rôle d'un port d'entrée/sortie du microprocesseur. Lors d'une opération donnée le registre spécial contient un opérande et une fois l'opération et terminée le résultat sera maintenu dans ce même registre spécial avant d'être transféré dans une autre partie du microprocesseur ou du système. La dimension du registre spécial correspond au mot manipulé par le système.

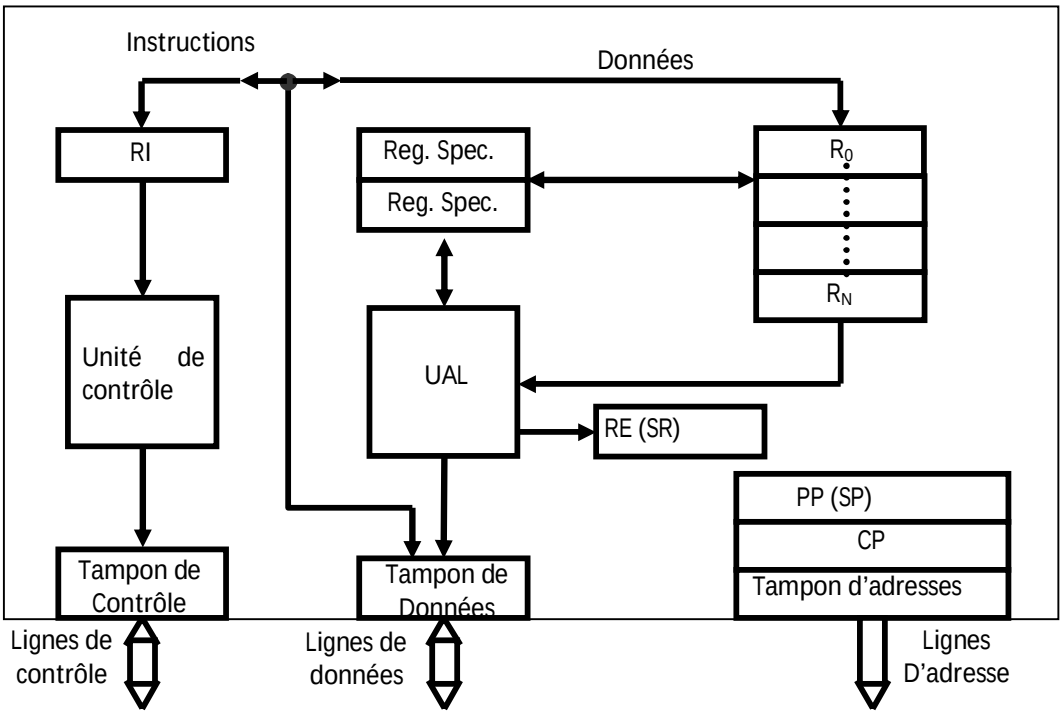


Fig-3- Ce schéma illustre d'une façon générale l'architecture interne d'un microprocesseur et la liaison entre ses différents blocs internes.

II-1-1-4- REGISTRES A USAGE GENERAL

En plus des registres spéciaux, certains microprocesseurs contiennent d'autres registres qui sont utilisés comme un moyen de stockage temporaire des données. Si un microprocesseur contient plusieurs registres, les programmes ne demandent pas à chaque fois un transfert des données à la mémoire. Ce facteur réduit considérablement le nombre d'accès à la mémoire externe ainsi que le temps d'exécution d'une instruction. Les microprocesseurs les plus développés, en plus de leur grand nombre de registres, contiennent une mémoire interne qui joue le rôle d'une mémoire cache.

II-1-1-5- COMPTEUR PROGRAMME

Un programme est une série ordonnée d'instructions qui sont stockées dans des emplacements successifs dans la mémoire. Chaque emplacement est défini par une combinaison binaire qui représente son adresse. Pour l'exécution d'une instruction l'unité centrale devra connaître l'emplacement de cette instruction; ceci est assuré par le compteur programme. Il contient toujours l'adresse de l'instruction qui devra être exécutée. A chaque fois qu'une instruction est exécutée, le compteur programme est incrémenté sauf dans le cas de certaines instructions comme BRANCH ou JUMP où il sera chargé par l'adresse de saut ou de branchement..

II-1-1-6- POINTEUR DE PILE

Vu que durant l'exécution d'un certain programme, l'ordinateur peut être appelé à interrompre le programme en cours pour passer à l'exécution d'un second programme plus important. Une fois l'exécution du second programme est terminée, le processeur devra retourner pour reprendre l'exécution du premier programme à partir de l'instruction ou l'interruption est arrivée. Ceci n'est possible que si l'adresse de cette dernière instruction soit sauvegardée juste après l'interruption dans une partie de la mémoire. La partie de la mémoire réservée à la sauvegarde de certains états du système durant l'interruption est appelée pile. Le pointeur de pile est un registre qui contient l'adresse de l'emplacement dans la pile qui est prête à la sauvegarde d'une donnée. La pile est du type LIFO : la première donnée sauvegardée sera la dernière à en sortir.

Exemple :

10	LOAD		Acc
11	03		
12	ADD	to	Acc
13	02		

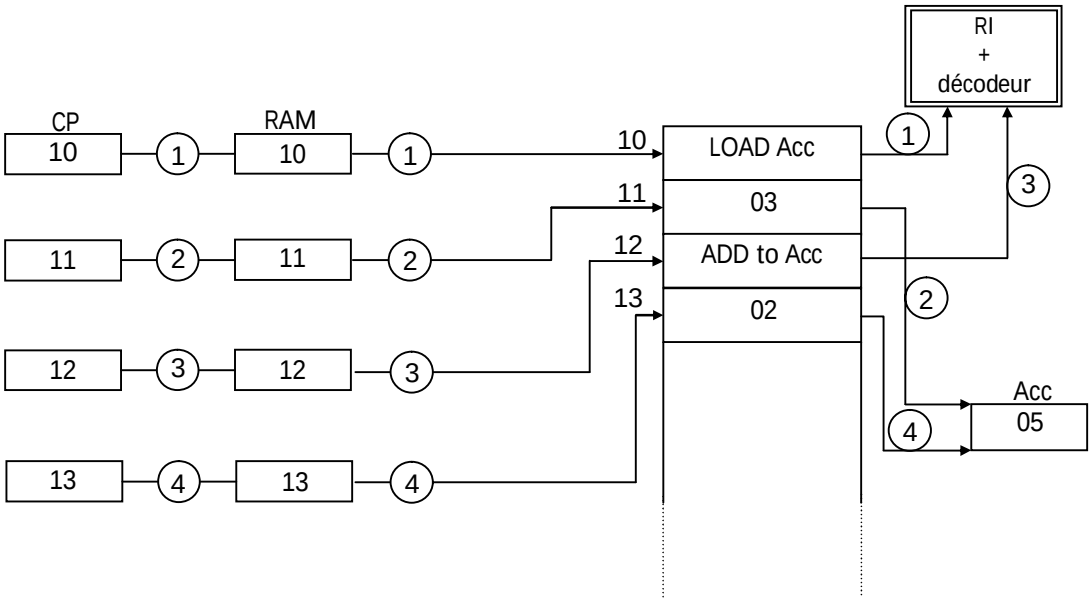


Fig-4 Sur ce schéma on présente les séquences introduites dans l'exécution de deux instructions (chargement de l'accumulateur par une donnée puis son addition à une deuxième donnée). Les séquences sont numérotées suivant leur ordre d'exécution.

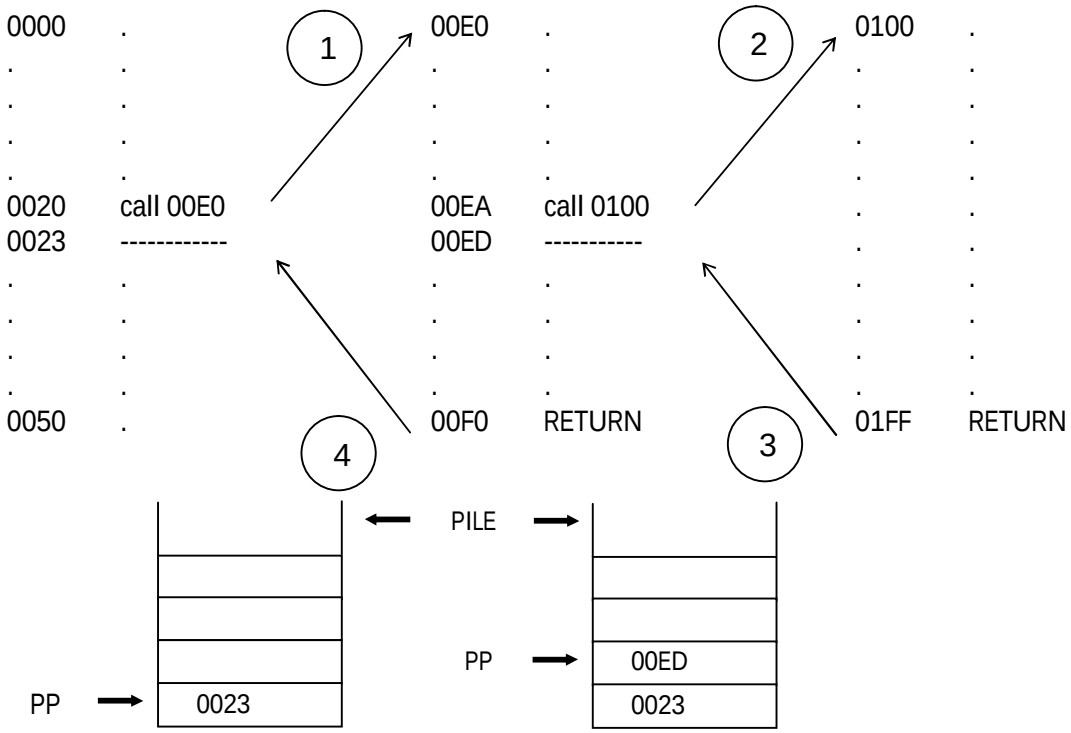


Fig-5 Cette figure montre l'utilisation de la pile lors de l'exécution d'instruction de branchement. Dans cet exemple seul le contenu du compteur programme est sauvegardé mais il peut y avoir d'autres registres tels que registre d'état, accumulateur…

La figure-5 montre le principe de travail d'une pile. On constate sur cette figure que juste avant l'exécution de l'instruction de branchement le microprocesseur sauvegarde l'adresse de retour. Comme il peut aussi être remarqué que le pointeur de pile PP pointe toujours à une case pleine de la pile, par conséquent l'empilement est toujours précédé d'une décrémentation alors que le dépilement est suivi d'une incrémentation.

La pile peut aussi être manipulée par l'utilisateur par deux instructions PUSH et POP, seulement il est conseillé de suivre une opération d'empilement par une opération de dépilement. Dans le cas contraire des données ou une partie du programme se trouvant dans la RAM risquent d'être effacées.

II-1-1-7- REGISTRE D'INSTRUCTION

C'est un registre qui contient le code de l'instruction en cours d'exécution. Une fois l'instruction est décodée; la partie contrôle génère les signaux appropriés pour accomplir l'exécution.

II-1-1-8- REGISTRE D'ETAT

A côté de l'UAL apparaît un registre particulier à N bits (N dépend du microprocesseur utilisé) dont les bits définissent l'état du processeur lors de l'exécution d'une instruction. Chacun de ces bits, réalisé physiquement par une simple bascule, est utilisé pour dénoter une condition particulière. Ces bits sont aussi appelés drapeaux et chacun d'eux est dédié à un état bien déterminé, nous citons :

A- Sign bit S (signe)

C'est un bit qui indique si un résultat est positif ou négatif. Du moment que le signe d'un nombre en binaire est toujours indiqué par son MSB, en pratique et pour un µP à 8 bits le bit S affiche l'état du huitième bit d'un résultat d'une opération donnée :

S=1 résultat négatif

S=0 résultat positif

B- Carry bit C(retenue)

L'état du bit C de la retenue indique qu'il y a un dépassement lors de l'exécution de certaines instructions telles que l'addition, la soustraction etc.... Pour des micro ordinateurs manipulant des octets (combinaison à 8 bits), le dépassement se traduit par une retenue du bit 7 comme montré sur la figure 6.

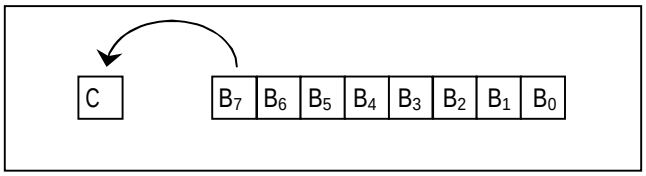


Fig-6 La retenue est obtenue par un débordement du bit B7 à la case C lors d'une opération qui affecte le registre d'état telle que l'addition, la soustraction, le décalage…

C- Half carry bit H (Retenue auxiliaire)

Cet indicateur est surtout utilisé quand des opérations sont effectuées sur des nombres codés en BCD.

Le bit H traduit le dépassement au niveau du bit 3 ou au niveau du bit 7.

D- Overflow bit V(bit de débordement)

Le rôle de ce bit est très important dans le cas où des opérations arithmétiques sont effectuées sur des opérandes signés (affectés de signe + ou -). Son état est exploité pour indiquer que le résultat d'une addition ou d'une soustraction risque d'être incorrect à cause d'un débordement du bit 6 vers le bit 7.

Pour bien comprendre et voir l'importance de ce bit, nous donnons des exemples qui illustrent différents cas :

Les nombres A et B dans les exemples ci-dessous sont donnés en hexadécimale.

Exemple-1:

A=+74⇒ 01110100

B=+45⇒ 01000101

⇒ A+B ⇒

01110100

\+

01000101

10111001

Cas d'un résultat faux : la somme de deux nombres positifs a donné un résultat négatif.

Pas de retenue du bit 7 vers C

Il y a retenue du bit 6 vers bit 7

Exemple-2:

A= -3 A=10000011

B= -4 B=10000100

En complément à 2.

$A_2^- = 11111101$

\+

$B_2^- = 11111100$

111111001

En passant une deuxième fois au complément à 2 du résultat obtenu on trouve 10000111= -7

Il y a retenue du bit 7 vers C

Il y a retenue du bit 6 vers bit 7

Cas d'un résultat juste: la somme de deux nombres négatifs a donné un résultat négatif.

Exemple-3:

A=+4A A=01001010

+

B=+1A B=00011010

01100100

le résultat de +64

pas de retenue du bit 7 vers C.

pas de retenue du bit 6 vers bit 7.

Cas d'un résultat juste : la somme de deux nombres positifs a donné un résultat positif.

Exemple-4:

A= -65 A=11100101

B= -56 B=11010110

En complément à 2

10011011

+

10101010

101000101

le résultat après complémentation est de 00111011

Il y a retenue du bit 7 vers C.

pas de retenue du bit 6 vers bit 7.

Cas d'un résultat faux : la somme de deux nombres négatifs a donné un résultat positif.

De ces quatre exemples on tire une règle telle que :

Si durant l'opération, le débordement du bit 6 vers bit 7 et la retenue qui résulte du bit 7 vers le drapeau C sont identique le résultat est juste, autrement le résultat est faux.

*-Pour l'addition

$$V = \overline{A_7}\,\overline{B_7}\,R_7 + A_7\,B_7\,\overline{R_7}$$

terme 1 : $\overline{A_7}\,\overline{B_7}\,R_7 = 1 \Rightarrow$ A> 0 B> 0 mais le résultat est négatif

terme 2 : $A_7\,B_7\,\overline{R_7} = 1 \Rightarrow$ A< 0 B< 0 mais le résultat est positif

**-Pour la soustraction

$$V = A_7\overline{B_7}\overline{R_7} + \overline{A_7}B_7R_7$$

terme 1 : $A_7\overline{B_7}\overline{R_7} = 1 \Rightarrow$ A<0 B>0 mais le résultat est positif

terme 2 : $\overline{A_7}B_7R_7 = 1 \Rightarrow$ A>0 B< 0 mais le résultat est négatif

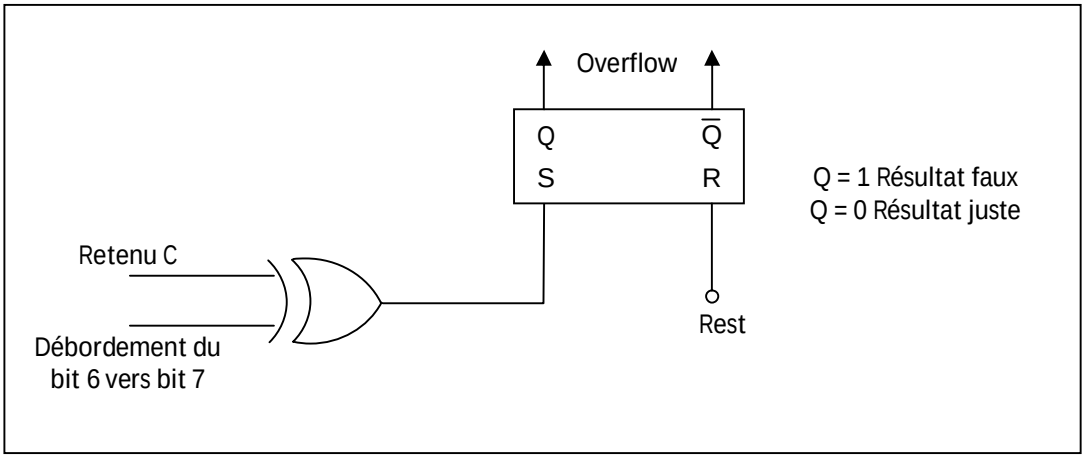


Fig-7 Circuit de principe, basé sur l'utilisation d'une bascule R-S, qui permet de générer le drapeau correspondant à l'overflow. Ce qui permet de déduire si lors d'une opération sur des nombres signés le résultat est juste ou faux.

E- Bit de parité (Parity bit) P

C'est un drapeau qui est mis à 1 à chaque fois que le nombre de bits égaux à 1 dans un résultat est pair. Attention, l'état de ce bit n'a aucune relation avec la parité d'un résultat, toutefois il indique la parité du nombre de digits égaux à 1 dans le résultat

Exemple :

0001011 Nbre de 1=4 paire

P=0 Nbre de 1 dans le résultat est impaire.

10011100 N^{bre} de 1=5 impaire

P=1 Nbre de1 dans le résultat est paire

F- Bit de Zéro (zero bit) Z

C'est un drapeau utilisé par le processeur pour tester si un résultat est nul ou non.

Z=1 résultat nul

Z=0 résultat non nul

II-1-2 Le microprocesseur 6800

II-1-2-1 Architecture interne du 6800

L'architecture du microprocesseur 6800 peut être scindée en deux blocs :

- Le boc de contrôle :

A ce niveau la tâche consiste à lire le code de l'instruction à être exécutée et le placer dans le registre d'instruction (reg. Inst. à 8bits). Le contenu du registre d'instruction sera transmis à l'unité de contrôle dont le rôle est le décodage de l'instruction pour la génération des signaux de contrôle qui en résultent.

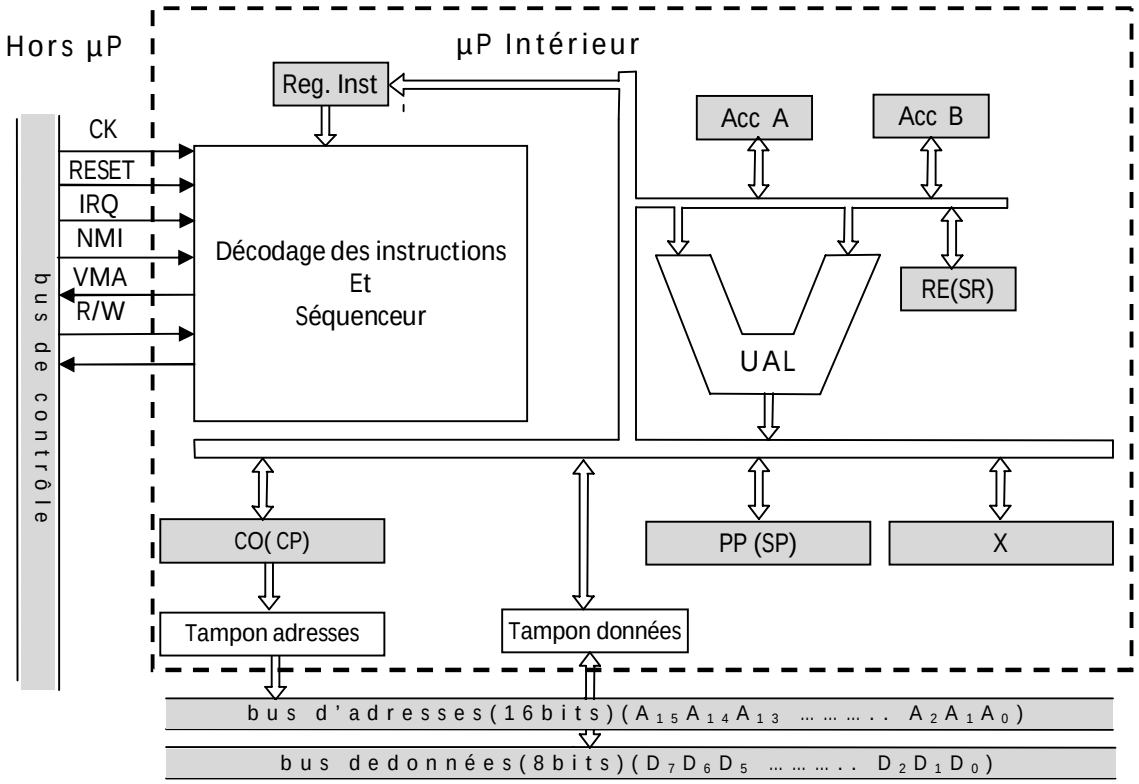


Fig.8 : Architecture interne du 6800

- Le boc de traitement :

Ce bloc est chargé de réaliser les opérations arithmétiques ou logiques, au niveau de l'UAL, suite au décodage de l'instruction en cours.

- Les registres principaux à 8 bits appelés accumulateurs (Acc et Acc B) sont utilisés pour contenir les opérandes de l'instruction en cours et du sauvegarde du résultat de l'opération réalisée. Le registre d'état (status register) RE(SE) travaille en étroite collaboration avec l'UAL, il permet d'afficher l'état l'opération tels que le résultat est : nul , négatif, déborde de 8 bits, ….
- On trouve aussi des registres de 16 bits dont la fonction st :

- Le compteur programme ou compteur ordinal CO (PC) contient toujours l'adresse suivante qui va être exécutée. Il permet d'exécuter d'un programme d'une façon séquentielle.
- Le pointeur de pile ou stack pointer est utilisé pour être initialisé avec le l'adresse du fond de la pile qui est choisie dans la mémoire RAM du système.
- Le registre indexe X est utilisé comme un registre qui parmi les fonctions que l'utilisateur peut choisir l'accès à des tables.

II-1-2-2 Architecture externe du 6800

Le microprocesseur MC6800 est présenté sous forme d'un circuit intégré de 40 broches tel qu'il est montré sur la figure qui peuvent être réparties en trois groupes qui définissent les trois bus du système conçu autour de ce microprocesseur : bus d'adresses, bus de données et bus de contrôle.

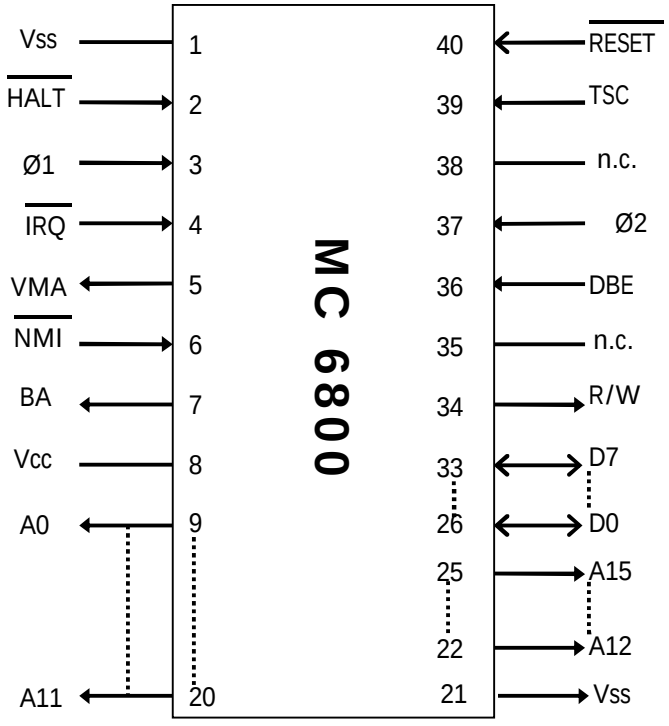


Fig.9 :Architecture externe du 6800

Bus d'adresses : C'est un bus de 16 lignes pouvant adresser un espace maximal de 2^{16} soit 64K réparti entre mémoire et entrées/sorties.

Bus de données : c'est un bus de 8lignes ; il véhicule l'information sous forme d'une combinaison numérique de 8bits entre les différents blocs qui constituent le système à microprocesseur.

Bus de contrôle : c'est un bus d'un ensemble de lignes dont chacune a une fonction bien déterminée :

R/W	:	c'est un signal issu du processeur pour indiquer le sens de transfert de l'information. Un état haut indique un cycle de lecture (donnée vers le processeur), un état bas indique un cycle d'écriture (donnée du processeur vers les autres éléments du système).
$\overline{\text{RESET}}$	:	réinitialisation du système
$\overline{\text{IRQ}}$	:	requête d'interruption. Un zéro sur cette ligne indique qu'il une demande d'un service. Si elle n'est pas inhibée par l'utilisateur dans son programme, une fois que l'instruction en cours est exécutée le processeur interrompe le programme pour se brancher à une routine de service.
$\overline{\text{NMI}}$	:	elle travaille de la même façon que NMI, la seule différence est que celle-là ne peut pas être inhibée par l'utilisateur. Elle est dite non masquable.
$\overline{\text{HALT}}$	:	c'est un signal d'entrée au processeur, quand il est activé le processeur s'arrête et libère ses bus en les plaçant en haute impédance.
Ø1 et Ø2 :		ce sont les signaux d'horloge, ils permettent de cadencer les séquences de travail du processeur.
VMA :		le processeur active ce signal en le mettant à un niveau haut pour informer les éléments du système que la combinaison numérique placée sur le bus d'adresses est une adresse valide.
TSC :		son activation place le bus d'adresses en haute impédance
BA :		son activation informe l'ensemble des éléments du système que les bus sont libérés.
DBE :		c'est un signal qui valide ou met en haute impédance le bus de données

II-3- MEMOIRES

II-3-1-INTRODUCTION

Pour un ordinateur, les facilités de stockage des données et des programmes sont indispensables. Sans ces facilités l'ordinateur est réduit à une simple calculatrice. L'information est gardée par les éléments de stockage sous forme binaire (1-0)

Les mémoires peuvent être caractérisées par les critères suivants :

a- Mémoire volatile

C'est une mémoire qui, une fois l'alimentation du système est coupée ou interrompue même pendant un temps très court, perd l'information qui s'y trouve.

b- Mémoire non volatile

C'est un type de mémoire où l'information stockée est gardée pour toujours, même après coupure de l'alimentation.

c- Mémoire à lecture destructive

Dans ce cas l'information est perdue à chaque fois qu'une lecture de cette dernière est effectuée. Pour remédier à ce problème l'opération lecture doit être suivie d'une opération d'écriture pour maintenir l'information stockée inchangée.

d- Mémoire à accès séquentiel

C'est une mémoire où tous les emplacements ou case, mémoire n'ont pas le même temps d'accès. Pour accéder à la case mémoire d'adresse N toutes les adresses de O à N-1 seront balayées. Finalement, on peut conclure que le temps d'accès à l'adresse N-1 est inférieur au temps d'accès à l'adresse N.

e- Mémoire à accès aléatoire(RAM)

Dans ce type de mémoire, le temps nécessaire pour la lecture ou l'écriture d'une information est indépendant de l'emplacement dans la mémoire où l'opération E\L doit être effectuée. C'est à dire toutes les adresses de O à N représentent le même temps d'accès.

f- Mémoire à E\L

Généralement sont des mémoires réservées à l'utilisateur du système. Car ce sont des mémoires où leurs contenus peuvent être écrits ou lus. Généralement on les appelle mémoires de données.

g- Mémoire à lecture :

Ce sont des mémoires, une fois programmées elles ne peuvent qu'être lues. Le plus souvent elles sont programmées par le concepteur de système. Elles sont appelées mémoires de programmes.

II-3-2- MEMOIRE A SEMI-CONDUCTEUR

La figure 8 montre la structure d'une mémoire à semi-conducteur. Comme il peut être constaté sur cette figure, la Mémoire est divisée en N mots, où N est une puissance de 2 et chaque mot est spécifié par une adresse. Toutes les cases mémoires ont le même nombre de bit (m) qui représente la longueur du mot de la machine.

Les mémoires à semi-conducteur sont les plus utilisées dans les micros ordinateurs. Elles sont du type non destructif et elles représentent un faible temps d'accès –«high speed memories»- de l'ordre de la nano seconde (ns). La partie mémoire d'un système est généralement constituée de deux parties indépendamment adressables RAM et ROM. les mémoires de la partie RAM sont classées dans deux catégories :

Mémoire statique où l'élément principal du stockage d'un digit est une bascule. Dans ce type de mémoire une fois l'information est écrite et tant que l'alimentation n'est pas coupée, elle sera inchangée.

Mémoire dynamique où l'élément principal de stockage d'un digit est une capacité. Vu les fuites dues à la décharge de la capacité en fonction de temps, il sera nécessaire de lire l'information et de la réécrire sur des intervalles de temps réguliers; cette procédure on la trouve dans littérature appelée rafraîchissement de la mémoire.

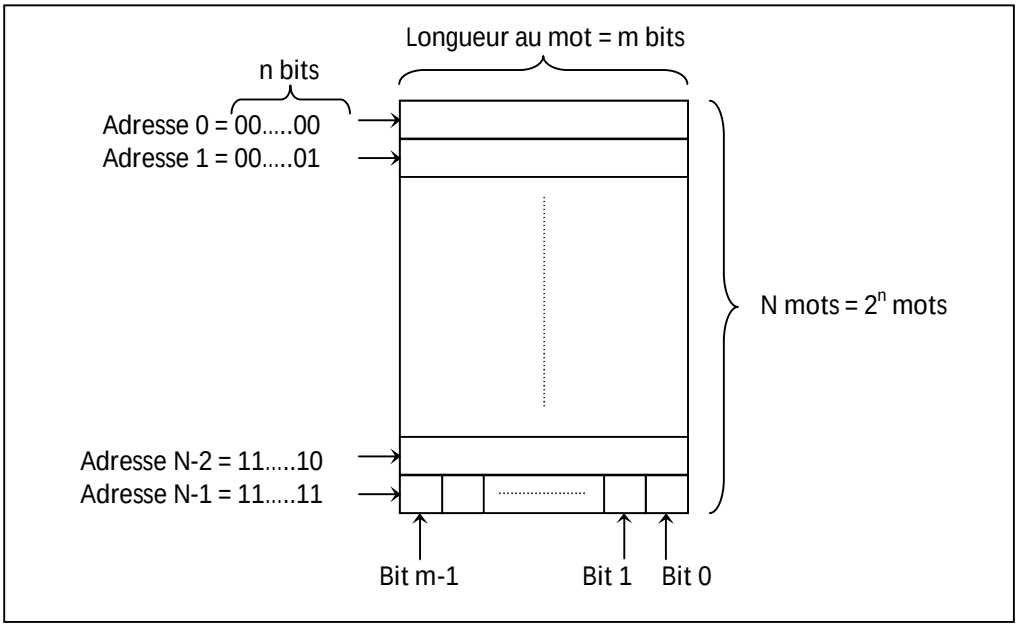


Fig-10La structure d'une mémoire à semi-conducteur de capacité $N=2^n$ mots de longueur m bits.

II-3-2-1- MEMOIRE A E\L (RAM) "RANDOM ACCESS MEMORY"

Bien que du point de vue littéraire le mot RAM veuille dire mémoire à accès aléatoire, techniquement il est donné aux mémoires à lecture et écriture.

L'espace mémoire RAM est réservé aux données variables; qui peuvent être chargées aux cours de l'exécution d'un programme.

II-3-2-2- MEMOIRE A LECTURE (ROM) "READ ONLY MEMORY"

Comme son nom l'indique, ROM est une mémoire à lecture uniquement. C'est une mémoire réservée aux programmes, données invariantes et à des tables.

Une fois l'utilisateur a préparé les informations qui doivent être écrites dans la ROM, sous forme d'un listing d'un programme ou d'une table de vérité, le fabriquant réalise un masque interprétant les informations données par l'utilisateur afin de programmer d'une façon permanente la ROM.

II-3-2-3- PROM "PROGRAMMABLE ROM"

C'est un type de ROM qui peut être programmé par l'utilisateur une seule fois, et ceci grâce à un équipement spécial de programmation de PROM. Le fabricant délivre les composants avec tous les bits mémoire initialisés généralement à des 1. les PROM sont plus chers que les ROM et plus flexibles.

II-3-2-4- EPROM "ERASABLE PROM"

Ce sont des PROM que l'utilisateur peut les effacer et par conséquent pouvoir changer la configuration binaire qui s'y trouvait avant. Pour effacer le contenu précédent de la mémoire PROM, le composant sera enlevé du circuit pour être exposé à l'UV pour une période de 15 à 20 mn comme spécifier par le constructeur. L'opération d'effacement n'est pas sélective, elle est totale pour tout le composant. Les EPROM sont plus chers que PROM.

II-3-2-5- EAROM "Electrically Altrable ROM"

Ces mémoires n'ont pas besoin d'UV pour être effacées, mais elles peuvent être programmées directement sur le circuit sans qu'elles soient enlevées. L'opération d'écriture peut être de l'ordre de la milliseconde.

II-4- INTERFACAGE DES MEMOIRES A SEMICONDUCTEUR

II-4-1 INTRODUCTION

En générale une mémoire à SC se présente sous forme d'un circuit intégré avec :

- Des broches pour son adressage dont le nombre définit la capacité de la mémoire.
- Des broches pour la donnée dont le nombre définit la longueur du mot.
- Des broches pour le contrôle de la lecture et de l'écriture.
- Une broche ou deux pour la sélection du composant.

Nous présentons dans la figure 11, d'une façon générale la configuration externe d'un circuit intégré type mémoire. Du moment que tous les composants d'un système à microprocesseur et parmi eux les mémoires partagent les mêmes lignes d'adresse et les mêmes lignes de donnée, chacun de ces composants est doté d'une ligne de sélection qui sera activé que la communication le concerne. C'est ainsi qu'il n'y aura aucun conflit entre les composants.

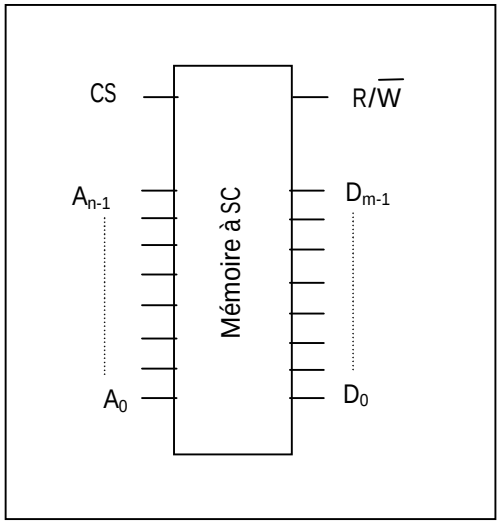


Fig-11 le brochage d'une mémoire d'une capacité de 2^n (broches d'adresse de n lignes A_0...A_{n-1}) mots de longueur m bits (broches de donnée de m lignes D_0...D_{m-1}).

II-4-2 DECODAGE PARTIEL D'ADRESSE

Dans cette forme de décodage d'adresse, bien que l'espace même que peut adresser le µp est de 2^n, seule une partie de cet espace est utilisée. C'est la forme de décodage la plus simple est par conséquent la moins chère.

Exemple : Donnons le schéma de connexion de deux blocks de 2 Koctets chacun, dans un système à µp à 8 bits avec un bus d'adresse de 16 lignes (A_0.....A_{15}). En utilisant le décodage partiel d'adresse.

L'un des circuits correspondant à la solution de l'exemple ci-dessus est donné sur la figure 12. Comme il peut être remarqué sur cette figure, les lignes d'adresse de A10 à A14

n'interviennent pas dans l'adressage des deux composants CI1 et CI2. Seule la ligne A15 est utilisée pour la sélection de l'un des deux circuits CI1 et CI2. Par conséquent, un espace de 32 Koctets est attribué à chacune des deux mémoires.

Le décodage partiel d'adresse est une forme d'interfaçage qui est utilisée dans les petits systèmes conçus pour une tache bien déterminée, dans les systèmes à coût réduit et surtout là où une éventuelle extension de la mémoire n'est pas prévue.

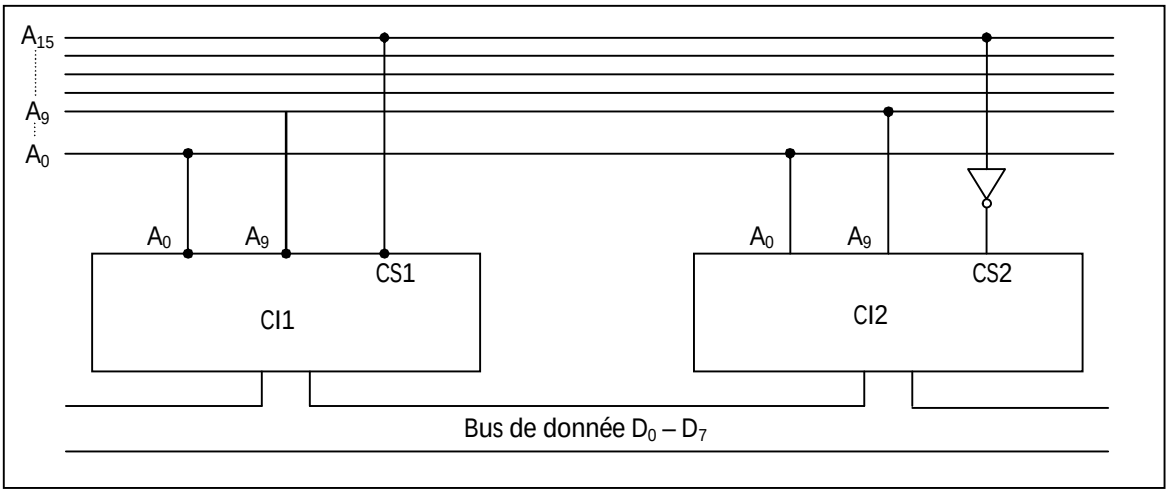


Fig-12 Exemple d'adressage à décodage partiel ou deux de 2Koctets chacune, se partage tout l'espace mémoire avec 32Koctets pour chacune.

II-4-3 DECODAGE COMPLET D'ADRESSE

Un système à microprocesseur est dit à décodage complet d'adresse si chaque emplacement dans la mémoire répond à une et une seule adresse, ainsi toutes les lignes d'adresses sont utilisées pour accéder à une case mémoire.

Exemple : connexion de 2 circuits mémoires de 16 Koctets chacun dans un système à microprocesseur à 8 bits, avec un bus d'adresse de 16 lignes.

Pour répondre à cet exemple il serait intéressant de partager tout l'espace mémoire en quatre blocs de 16Koctets chacun. Après on attribue à chacun des deux circuits un bloc et les deux blocs restant seront prévus pour une éventuelle extension. Pour cela il suffit d'utiliser un décodeur, (1 parmi 4), exemple du 74LS139, qui suivant la valeur de A14 et A15 sélectionne un bloc.

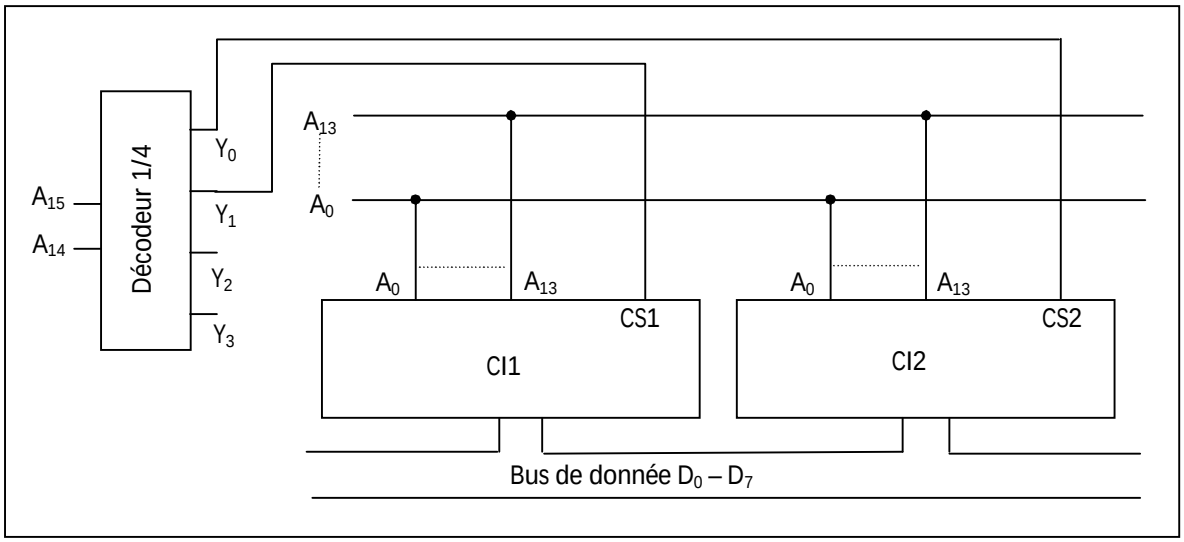


Fig-12 Le circuit de cette figure utilise un décodeur 1 parmi 4 permettant de diviser l'espace mémoire total de 64 Koctets en 4 blocs de 16 Koctets . Le premier est attribué au circuit CI1 et le deuxième est attribué au circuit CI2. Par contre les deux blocs 3 et 4 peuvent être à leur tour divisés en sous bloc ou utilisés tels qu'ils sont pour une éventuelle extension.

II-5 Programmation du 6800

Un système à microprocesseur est un système électronique programmable ; quelque soit la tâche à réaliser elle doit être spécifiée par un ensemble de séquences bien ordonnées écrites dans un langage de programmation que l'on nomme programme. C'est dans cette optique que plusieurs langages de programmation sont développés. On trouve les langages qui utilisent notre langage courant et qui permettent de traduire aisément en programme l'algorithme donné par opérateur pour la solutionner un problème donné ; on trouve ici les langages de programmation tels que C, C++, Fortran, Pascal…… qui sont dits langages évolués ou haut niveau ; ils sont plutôt proches de l'opérateur que de la machine. Travailler avec ces langages ne demande pas une connaissance approfondie sur le microprocesseur de la machine utilisée.

La machine travaille en binaire le 0 et 1, un langage de programmation qui est très proche de la machine mais reste facilement manipulable par un opérateur a été développé appelé assembleur.

II-5-1 Modes d'adressage du 6800

Le mode d'adressage définit la manière avec laquelle l'opérande est représenté dans une instruction. Le mode d'adressage définit la souplesse de programmation d'un système à microprocesseur. Dans le cas du 6800 de Motorola on rencontre 6 modes d'adressage.

Implicite ou inhérent : dans ce mode l'opérande est déduit directement du code de l'instruction. Une fois l'instruction est décodée l'opérande à traiter est directement déduit.

Immédiat : dans ce mode d'adressage l'opérande représente la donnée à manipuler. Après décodage de l'instruction le microprocesseur sort à la mémoire pour chercher la donnée qui vient juste après l'instruction qui vient d'être exécutée.

Direct : l'opérande dans ce cas représente l'adresse de page 0 où se trouve la donnée à traiter. L'adresse est donnée par un seul octet qui représente l'octet faible de l'adresse, l'octet fort est pris comme 0.

Absolu ou étendu : Il est identique au mode d'adressage direct à la seule différence que dans ce cas l'opérande est une adresse de 2 octets.

Indexé : c'est un mode où représente la valeur qu'il faut ajouter au registre index IX pour avoir l'adresse de la donnée à traiter par l'instruction en cours.

Relatif : l'opérande est un nombre signé de 8 bits (-128 à 127) qu'il faut ajouter au contenu du compteur programme PC afin d'effectuer un saut.

II-5-2 l'assembleur

L'assembleur est dit de bas niveau parce qu'il est très proche de la machine. Par conséquent, programmer en assembleur nécessite des connaissances au préalable sur le microprocesseur utilisé. Une instruction en assembleur peut être décomposée en quatre zones telles qu'il est montré par le tableau suivant.

Etiquette	mnémonique	opérande	commentaire
Définit la référence de l'instruction à laquelle on veut effectuer un saut ou un branchement	Définit l'abréviation qui indique la nature de l'opération allant être effectuée par le microprocesseur.	Les données qui vont être traitées par l'opération spécifiée par le mnémonique	Cette partie est facultative, elle est utile pour commenter la tâche réalisée par un ensemble d'instructions.

II-5-3 Jeu d'instructions du 6800

a- Groupes du Jeu d'instructions du 6800

Le 6800 dispose d'un jeu d'instructions réparties en groupe suivant les opérations à effectuer tel qu'il est résumé ci-dessous.

Instructions de transfert pour les accumulateurs A et B	
LDAA : Chargement de l'Acc A par l'opérande suivant le mode d'adressage spécifié	LDAB : Chargement de l'Acc B par l'opérande suivant le mode d'adressage spécifié
PULA : Chargement de l'Acc A pour le contenu de la case mémoire pointée par SP	PULB : Chargement de l'Acc B pour le contenu de la case mémoire pointée par SP
STAA : Sauvegarde du contenu de l'Acc A dans la mémoire suivant le mode d'adressage spécifié	STAB : Sauvegarde du contenu de l'Acc B dans la mémoire suivant le mode d'adressage spécifié
PSHA : sauvegarde du contenu de l'Acc A dans la case mémoire pointée par SP	PSHB : sauvegarde du contenu de l'Acc B dans la case mémoire pointée par SP
TBA : transfert du contenu de l'Acc B vers l'Acc A	TAB : transfert du contenu de l'Acc A vers l'Acc B
TPA : transfert du contenu du registre d'état SR vers l'Acc A	
TAP : transfert du contenu l'Acc A vers le registre d'état SR	

Instructions de transfert pour les registres à 16 bits indexe IX et pointeur de pile SP	
LDX : Chargement du registre IX par l'opérande suivant le mode d'adressage spécifié	LDS : chargement du pointeur de pile SP par l'opérande suivant le mode d'adressage spécifié
STX : Sauvegarde du contenu du registre indexe IX dans la mémoire (nécessite 2 cases mémoires suivant le mode d'adressage spécifié)	STS : Sauvegarde du contenu du pointeur de pile SP dans la mémoire (nécessite 2 cases mémoires) suivant le mode d'adressage spécifié
TSX : transfert du contenu du pointeur de pile SP vers le registre indexe IX	TXS : transfert du contenu du registre indexe IX vers le pointeur de pile SP

Instructions pour les opérations arithmétiques	
ABA : additionner les contenus de l'Acc A et l'Acc B, le résultat sera gardé dans l'Acc A	SBA : soustraire le contenu de l'Acc B de celui de l'Acc A, le résultat sera gardé dans l'Acc A
ADC A: additionner au contenu de l'Acc A l'opérande spécifié par le mode d'adressage ainsi que l'état du bit de la retenue, le résultat sera gardé dans l'accumulateur A	ADC B:additionner au contenu de l'Acc B l'opérande spécifié par le mode d'adressage ainsi que l'état du bit de la retenue, le résultat sera gardé dans l'accumulateur B
ADD A:additionner au contenu de l'Acc A l'opérande spécifié par le mode d'adressage, le résultat sera gardé dans l'accumulateur A	ADD B: additionner au contenu de l'Acc B l'opérande spécifié par le mode d'adressage, le résultat sera gardé dans l'accumulateur B
SBC A : soustraire au contenu de l'Acc A l'opérande spécifié par le mode	SBC B : soustraire au contenu de l'Acc B l'opérande spécifié par le mode

d'adressage ainsi que l'état du bit de la retenue, le résultat sera gardé dans l'accumulateur A	d'adressage ainsi que l'état du bit de la retenue, le résultat sera gardé dans l'accumulateur B
SUB A : soustraire au contenu de l'Acc A l'opérande spécifié par le mode d'adressage, le résultat sera gardé dans l'accumulateur A	SUB B : soustraire au contenu de l'Acc B l'opérande spécifié par le mode d'adressage, le résultat sera gardé dans l'accumulateur B
SEI : mettre à 1 le bit d'interruption.	SEC : mettre à 1 la retenue C SEV : mettre à 1 l'overflow V
DAA : ajustement décimal de l'Acc A de le cas où manipule le code BCD par l'ajout du nombre 6 au nibble à corriger.	

Autres instructions pour les opérations arithmétiques	
INC : incrémentation par l'addition d'un 1 ; c'est instruction dont l'opérande indique l'accumulateur ou la mémoire à incrémenter	DEC : décrémentation par la soustraction d'un 1 ; c'est instruction dont l'opérande indique l'accumulateur ou la mémoire à incrémenter
INX : incrémentation du registre index IX	DEX : décrémentation du registre index IX
INS : incrémentation du pointeur de pile SP	DES : décrémentation du pointeur de pile SP
CLR : Mettre à zéro l'undes accumulateurs ou l'emplacement mémoire spécifié par son adresse	NEG : rendre négatif (en complément à 2) de l'un des deux accumulateurs oul'emplacement mémoire spécifié par son adresse
CMP : la donnée spécifiée par l'opérande est comparée au contenu de l'accumulateur indiqué par le mnémonique	CPX :la donnée de 16 bits spécifiée par l'opérande est comparée au contenu duregistre indexe IX

instructions pour des opérations logiques	
AND : le ET logique entre l'accumulateur et la donnée spécifiée par l'opérande	ORA : le OU logique entre l'accumulateur et la donnée spécifiée par l'opérande
EOR : le XOR logique entre l'accumulateur et la donnée spécifiée par l'opérande	COM : le complément à 1 de l'accumulateur ou d'un emplacement mémoire
LSR : décalage logique à droite de l'accumulateur ou l'emplacement mémoire. On pousse le MSB par un 0 et le bit 0 sort au bit retenue C.	ASL : décalage arithmétique à gauche de l'accumulateur ou l'emplacement mémoire. On pousse le LSB par un 0 et le bit 7 sort au bit retenue C.
ASR : décalage arithmétique à droite de l'accumulateur ou l'emplacement mémoire. On pousse le MSB par l'état du bit 7 et le bit 0 sort au bit retenue C.	ROR : on fait une rotation à droite de l'accumulateur ou l'emplacement mémoire. On pousse le MSB par la retenue C et le LSB vient à la retenue C.
ROL : on fait une rotation à gauche de l'accumulateur ou l'emplacement mémoire. On pousse le LSB par la retenue C et le MSB vient à la retenue C.	BIT : On effectue un AND logique entre l'accumulateur et l'opérande, pour voir il y a égalité ou non. Les contenu de l'accumulateur et de l'opérande restent inchangés.

Instruction de saut et branchement	
JMP : saut inconditionnel à une adresse de 16 bits spécifiée par l'opérande. Ou l'opérande donne le déplacement de 8 bits qu'il faut à IX pour avoir l'adresse à laquelle il faut aller.	BRA : c'est un saut conditionnel à une adresse en mode relatif.
BEQ : branchement s'il y a égalité	BNE : branchement s'il n'y a pas égalité.
BHI : branchement si supérieur pour les nombres non signés	BGT : branchement si supérieur pour les nombres signés (complément à 2)
BLE : branchement si inférieur ou égal pour les nombres signés (complément à 2)	BGE : branchement si supérieur ou égal pour les nombres signés (complément à 2)
BLS : branchement si inférieur ou égal pour les nombres non signés	BLT : branchement si inférieur pour les nombres signés (complément à 2)
BMI : branchement si le résultat est négatif	BPL : branchement si le résultat est positif
BSR : Appel d'un sous programme en mode relatif.	JSR : Appel d'un sous programme, l'opérande est l'adresse à laquelle il faut aller.
BCC : branchement si la retenue C est 0	BCS : branchement si la retenue C est 1
BVC : branchement si l'overflow V est 0	BVC : branchement si l'overflow V est 1

b- codes d'instructions du 6800

Opérations de transfert							
Instructions	Code de l'instruction						Nature de l'opération
	Imp.	Rel.	Imm.	Dir.	Ind.	Abs.	
LDA A			86	96	A6	B6	Charger l'Acc A par…
STA A				97	A7	B7	sauvegarder l'Acc A dans..
LDA B			C6	D6	E6	F6	Charger l'Acc B par..
STA B				D7	E7	F7	sauvegarder l'Acc B dans..
LDX			CE	DE	EE	FE	Charger registre index IX par..
STX				DF	EF	FF	sauvegarder registre index IX dans..
LDS			8E	9E	AE	BE	Charger pointeur de pile SP par..
STS				9F	AF	BF	sauvegarder pointeur de pile SP dans..
TAB	16						Transférer contenu de A vers B
TBA	17						Transférer contenu de B vers A
TAP	06						Transférer contenu de A vers registre RE
TPA	07						Transférer contenu du registre RE vers A
TXS	35						Transférer contenu de IX vers SP
TSX	30						Transférer contenu de SP vers IX
PSH A	36						Transférer contenu de A vers la pile
PSH B	37						Transférer contenu de B vers la pile
PUL A	32						Récupérer A de la pile
PUL B	33						Récupérer B de la pile

Opérations arithmétiques							
Instructions	Code de l'instruction						Nature de l'opération
	Imp.	Rel.	Imm.	Dir.	Ind.	Abs.	
ADD A			8B	9B	AB	BB	Additionner avec l'Acc A …
ADD B			CB	DB	EB	FB	Additionner avec l'Acc B ….
ADC A			89	99	A9	B9	Additionner donnée et retenue à l'Acc A…
ADC B			C9	D9	E9	F9	Additionner donnée et retenue à l'Acc B

SUB A			80	90	A0	B0	Soustraire de l'Acc A
SUB B			C0	D0	E0	F0	Soustraire de l'Acc B
SBC A			82	92	A2	B2	Soustraire donnée et retenue de l'Acc A
SBC B			C2	D2	E2	F2	Soustraire donnée et retenue de l'Acc B
DAA	19						Ajustement décimal de l'Acc A
INC A	4C						Incrémenter l'Acc A
INC B	5C						Incrémenter l'Acc B
INX	08						Incrémenter le registre indexé IX
INS	31						Incrémenter le pointeur de pile SP
INC					7C	5C	Incrémenter le contenu d'une mémoire
DEC A	4A						Décrémenter l'Acc A
DEC B	5A						Décrémenter l'Acc B
DEX	09						Décrémenter le registre indexé IX
DES	34						Décrémenter le pointeur de pile SP
DEC					6A	7A	Décrémenter le contenu d'une mémoire
NEG A	40						Complément à 2 de l'Acc A
NEG B	50						Complément à 2 de l'Acc B
NEG					60	70	Complément à 2 d'une case mémoire

Opérations logiques, décalage et rotation							
Instructions	Code de l'instruction						Nature de l'opération
	Imp.	Rel.	Imm.	Dir.	Ind.	Abs.	
AND A			84	94	A4	B4	ET logique avec l'Acc A....
AND B			C4	D4	E4	F4	ET logique avec l'Acc B...
ORA A			8A	9A	AA	BA	OU logique avec l'Acc A
ORA B			CA	DA	EA	FA	OU logique avec l'Acc B...
EOR A			88	98	A8	B8	OU Exclusif logique avec l'Acc A
EOR B			C8	D8	E8	F8	OU Exclusiflogique avec l'Acc B
COM A	43						Complémenter l'Acc A
COM B	53						Complémenter l'Acc B
COM					63	73	Complémenter la donnée spécifiée par son adresse
ASL A	48						décalage arithmétique à gauche de l'Acc A. On pousse le LSB par un 0 et le bit 7 sort au bit retenue C.
ASL B	58						décalage arithmétique à gauche de l'Acc B.
ASL					68	78	décalage arithmétique à gauche de la donnée spécifiée par son adresse.
ASR A	47						décalage arithmétique à droite de l'Acc A. On pousse le MSB par l'état du bit 7 et le bit 0 sort au bit retenue C.
ASR B	57						décalage arithmétique à droite de l'Acc B.
ASR					67	77	décalage arithmétique à droite de l'emplacement mémoire.
LSR A	44						décalage logique à droite de l'Acc A. On pousse le MSB par un 0 et le bit 0 sort au bit retenue C.
LSR B	54						décalage logique à droite de l'Acc B.
LSR					64	74	décalage logique à droite de l'emplacement mémoire.

ROL A	49						on fait une rotation à gauche de l'Acc A. On pousse le LSB par la retenue C et le MSB vient à la retenue C.
ROL B	59						on fait une rotation à gauche de l'Acc B.
ROL					69	79	on fait une rotation à gauche de l'emplacement mémoire.
ROR A	46						on fait une rotation à droite de l'Acc A. On pousse le MSB par la retenue C et le LSB vient à la retenue C.
ROR B	56						on fait une rotation à droite de l'Acc B.
ROR					66	76	on fait une rotation à droite de l'emplacement mémoire.

Opérations set, reset et comparaison							
Instructions	Code de l'instruction						Nature de l'opération
	Imp.	Rel.	Imm.	Dir.	Ind.	Abs.	
CLR A	4F						Mettre les bits de l'Acc A à zéro
CLR B	5F						Mettre les bits de l'Acc B à zéro
CLR					6F	7F	Mettre les bits de la mémoire à zéro
CLC	0C						Mettre la retenue C à zéro du registre RE
CLI	0^E						Mettre le bit I à zéro du registre RE
CLV	0A						Mettre le bit V à zéro du registre RE
SEC	0D						Mettre la retenue C à 1 du registre RE
SEI	0F						Mettre le bit I à 1 du registre RE
SEV	0B						Mettre le bit V à 1 du registre RE
CMP A			81	91	A1	B1	Comparer l'Acc A avec une donnée
CMP B			C1	D1	E1	F1	Comparer l'Acc B avec une donnée
CPX			8C	9C	AC	BC	Comparer le registre IX avec une donnée
CBA	11						Comparer l'Acc A avec l'Acc B

Opérations Branchement et de saut							
Instructions	Code de l'instruction						Nature de l'opération
	Imp.	Rel.	Imm.	Dir.	Ind.	Abs.	
BCC		24					Branchement si retenue égale à zéro(C=0)
BCS		25					Branchement si retenue égale à 1(C=1)
BEQ		27					Branchement s'il ya égalité
BGE		2A					Branchement si ≥ en complément à 2
BGT		2E					Branchement si > en complément à 2
BHI		22					Branchement si > pour sans signe
BLE		2F					Branchement si ≤ en complément à 2
BLS		23					Branchement si ≤ pour sans signe
BLT		2D					Branchement si < en complément à 2
BMI		2B					Branchement si < 0
BNE		26					Branchement si ≠
BPL		2A					Branchement si > 0
BRA		20					Branchement inconditionnel
BSR		2D					Branchement à un sous programme
BVC		28					Branchement si débordement égale à zéro(V=0)
BVS		29					Branchement si débordement égale à 1(V=1)
JMP					6E	7E	Saut inconditionnel à une adresse
JSR					AD	BD	Saut à un sous programme

II-6- ENTREES-SORTIES

II-6-1- INTRODUCTION

Un ordinateur communique avec son environnement (utilisateur, processus à contrôler, autres ordinateurs…etc) par l'intermédiaire de ses moyens d'Entrées / Sorties. Il reçoit des programmes et données, et il délivre les résultats après traitement. La communication peut être locale ou à grande distance. Il y a une grande variété de périphériques, ces derniers diffèrent les uns des autres par la vitesse de travail et le format de données.

Quelques types de périphériques sont :

- CRT displays
- Clavier
- Floppy disk
- Monitor
- Imprimante
- Cassette magnétique
- D/A, A/D convertisseur

II-6-2- INTERRUPTION

L'inconvénient principal réside dans le fait que le périphérique doit avoir une vitesse compatible avec la vitesse de CPU. Rares sont les périphériques qui répondent à cette exigence. Par conséquent, si le périphérique n'est pas prêt à recevoir au moment où le CPU a envoyé l'information, cette dernière sera perdue. Pour remédier à ce problème il y a deux possibilités :

a- Avant de transférer une donnée, le CPU doit vérifier ou tester l'état du périphérique concerné par la communication. Si le périphérique n'est pas prêt, le CPU reste dans une boucle d'attente comme montré sur l'organigramme de la figure 13. Comme il peut être remarqué dans ce cas, que le CPU perd beaucoup de temps à attendre afin que les périphériques soient prêts.

b- Dans cette deuxième possibilité c'est le périphérique qui informe le CPU qu'il est prêt à recevoir la donnée qui lui a été envoyée. Ceci est réalisé par le principe d'interruption. L'interruption peut arriver au moment où le CPU est en train d'exécuter un programme, suivant la nature et la configuration du système l'interruption peut être servie ou ignorée. Cependant, l'interruption peut provenir d'un périphérique parmi tant d'autres, le processeur dans ce cas doit reconnaître l'élément responsable de la requête d'interruption pour sélection la routine de service adéquate. Sur la figure 14 on donne un petit exemple de circuit de principe d'une génération d'interruption ainsi que la réponse du processeur pour l'acceptation ou non de cette dernière.

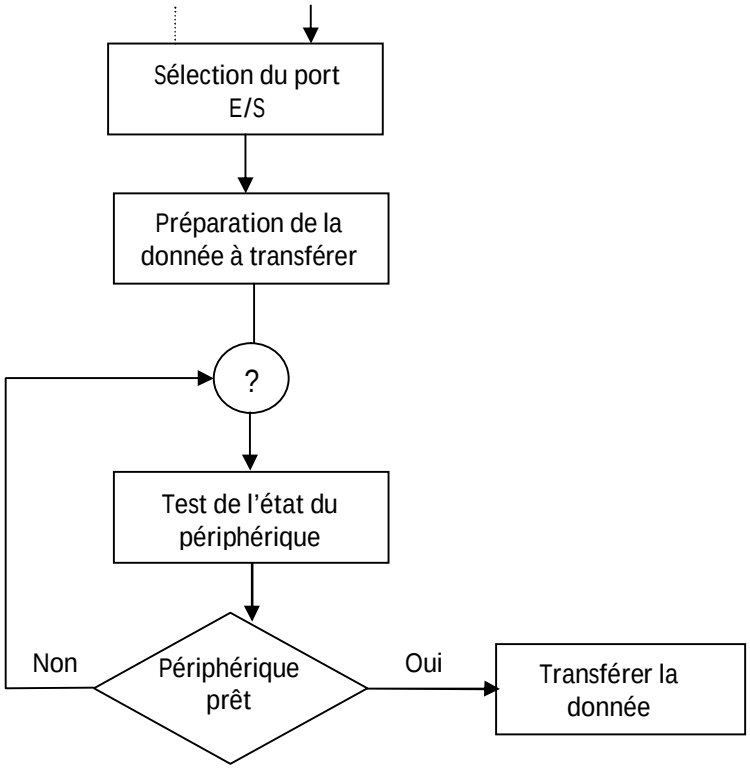


Fig-13 Cet organigramme montre les étapes de test d'un microprocesseur pour déterminer si un périphérique a besoin de service.

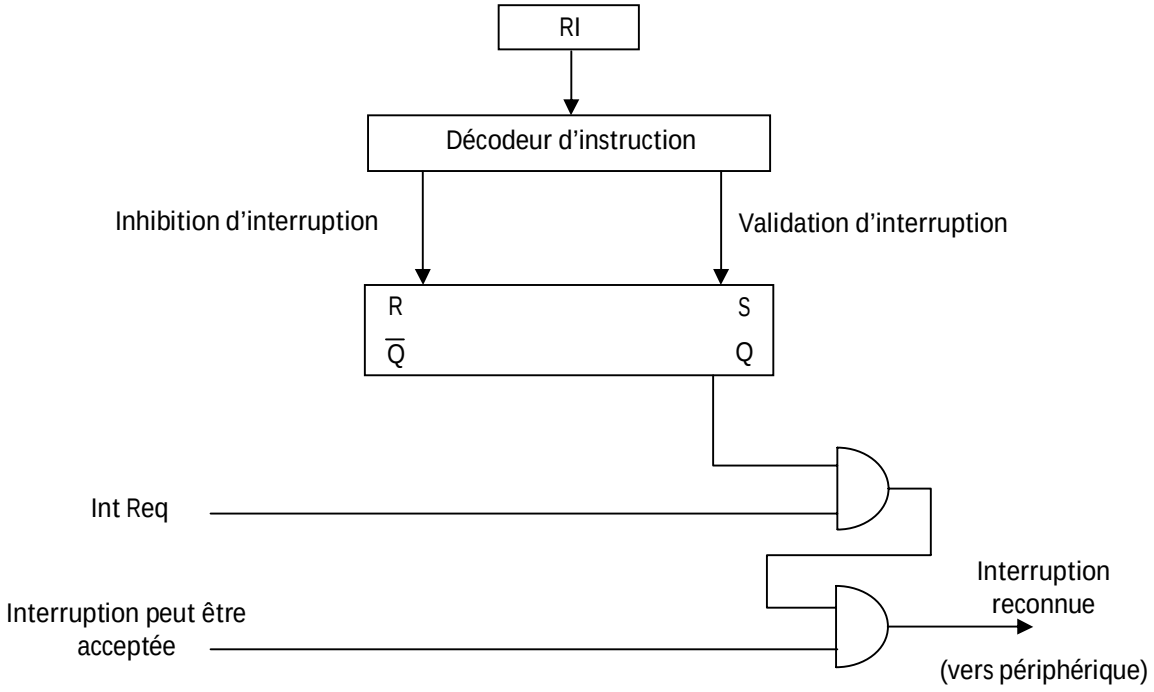


Fig-14 Ce circuit montre la logique de génération d'interruption ainsi que le circuit de validation ou d'inhibition, qui peut être actionné par l'utilisateur. Ainsi que la ligne de réponse du processeur.

II-6-2-1- SEQUENCES D'UNE INTERRUPTION

Bien que l'utilisation des systèmes à microprocesseur nécessite une documentation minimale sur le processeur noyau de ce système pour sa bonne exploitation. Il est cependant utile, de donner d'une façon générale les différentes étapes lors d'une requête d'interruption :

a- Demande d'interruption : (INT-IRQ-INTREQ)

Elle est faite par le changement d'état d'une ligne du microprocesseur réservée à cette fonction. Le processeur détecte l'interruption soit sur un front (montant ou descendant) ou par un état (haut ou bas).

b- Interruption reconnue : (INTACK-ACK-IACK)

Le microprocesseur termine l'instruction en cours et dans certains cas de microprocesseurs un signal sera généré pour informer le périphérique que sa requête est acceptée. En réponse le périphérique place un vecteur sur le bus de donnée qui permet au processeur de se brancher à la bonne routine de service. En l'absence de ce signal le périphérique maintient sa demande d'interruption jusqu'à ce qu'elle soit acceptée.

c- Sauvegarde d'état du CPU

Avant de servir le périphérique, il est nécessaire que le CPU s'assure qu'après le sous programme de service, il peut retourner au programme qui était en cours d'exécution avant que l'interruption arrive.

La sauvegarde des contenus des registres du CPU se fait dans la pile.

d- Identification du périphérique

Du moment que la demande d'interruption arrive au microprocesseur sur une seule ligne, et sachant que plusieurs périphériques peuvent être liés au système à microprocesseur alors il sera indispensable d'identifier le périphérique demandant l'interruption pour choisir le sous programme de service correspondant.

e- Servir le périphérique

Une fois le périphérique est identifié, le CPU se branche à la routine de service correspondante.

f- Récupération des états de CPU

Après avoir terminé le sous programme d'interruption, tous les registres du CPU seront rechargés par les données qu'ils contenaient juste avant l'arrivée de l'interruption.

g- Retour à l'exécution du programme qui était en cours au moment où l'interruption est arrivée.

II-6-3-2- CAS DE PLUSIEURS PERIPHERIQUES

Dans un système où plusieurs périphériques peuvent interrompre le CPU; l'élément demandant l'interruption doit être identifié pour que le CPU puisse se brancher au sous programme correspondant. Du moment qu'une seule ligne est réservée à la demande de l'interruption, deux méthodes sont utilisées pour la sélection des périphériques d'interruption :

- Sélection par software
- –Sélection par hardware.

a- Sélection par software

En réponse à une interruption, la routine de sélection ou d'identification commence par tester l'état d'interruption (bit) pour chaque périphérique afin de déterminer le quel a demandé une interruption. Le test commence par le périphérique de plus haute priorité. L'organigramme de la figure 15 montre la procédure de recherche de l'élément ayant demandé une interruption.

b- Sélection par hardware

Pour des systèmes qui nécessitent une grande vitesse de travail un contrôleur de priorité d'interruption est disponible (PIC). Il génère un vecteur adresse, correspondant au périphérique qui a demandé l'interruption, qui constitue la partie inférieure (lower byte) de l'adresse du commencement du sous programme de service de l'interruption. La sélection par hardware est plus rapide que celle par software. La figure 16 montre une logique simple de génération d'interruption et de priorité pour 4 périphériques.

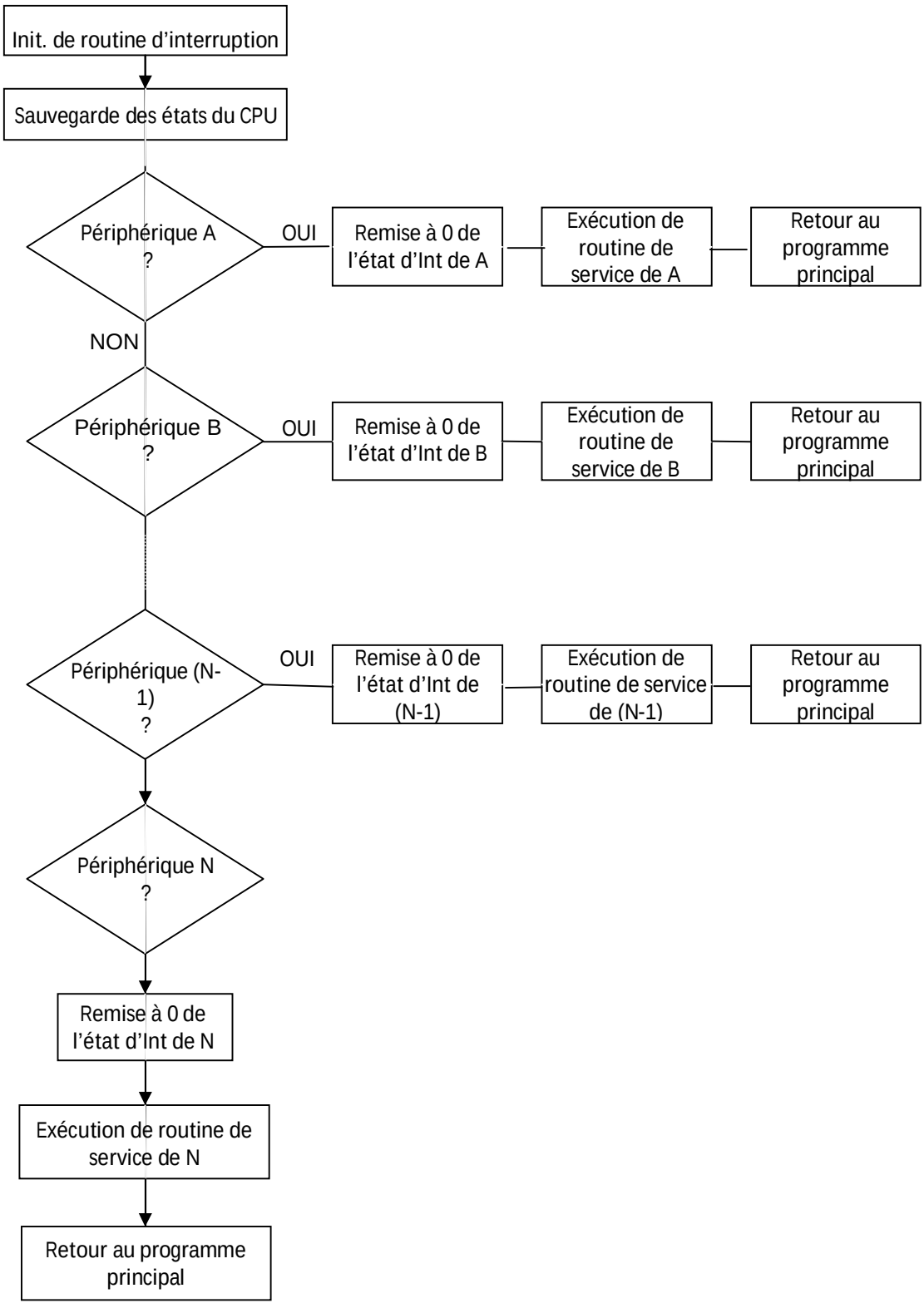


Fig-15 Organigramme de test de détermination de l'élément ayant demandé une interruption et son service. A chaque requête d'interruption le processeur par les mêmes étapes de test.

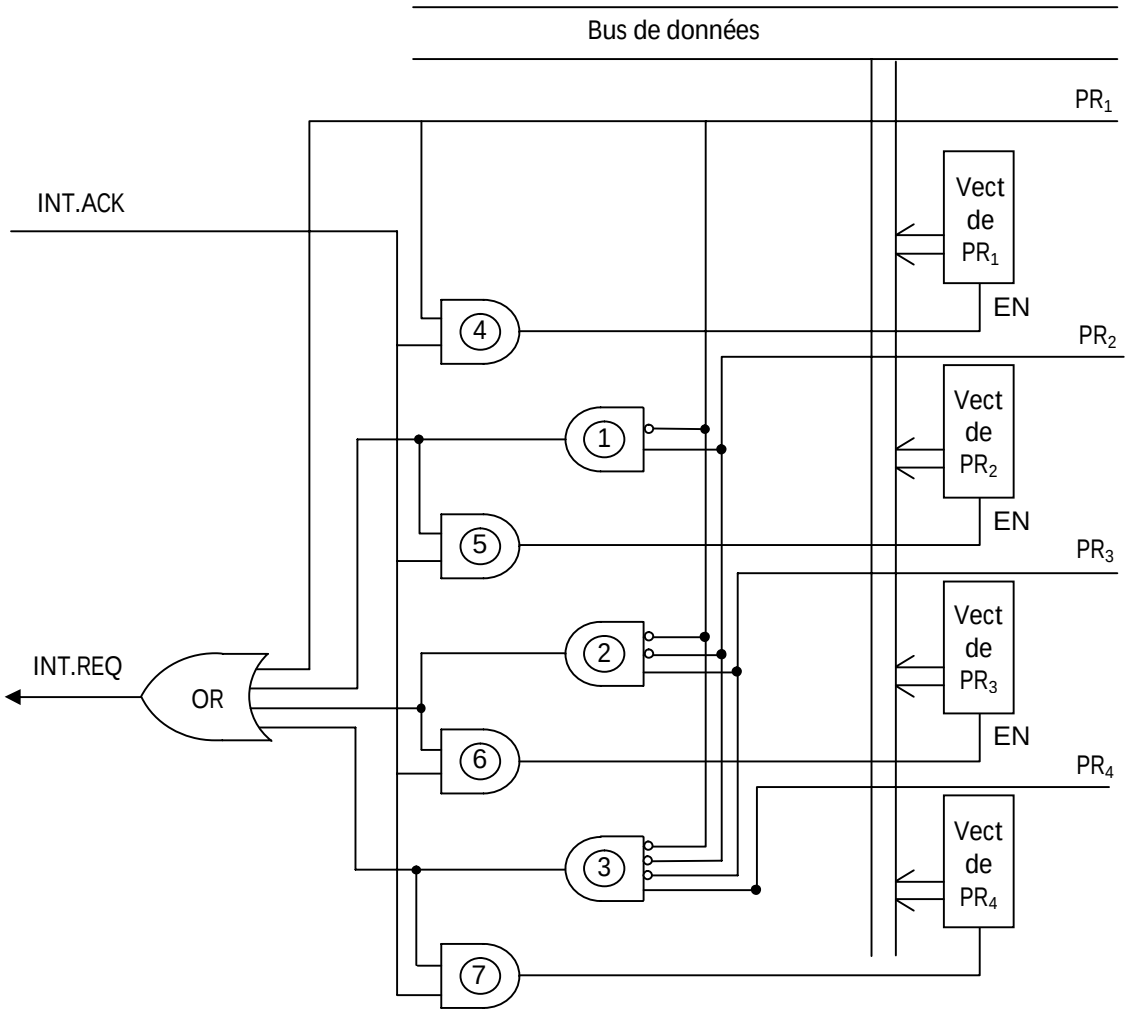


Fig-16 Le circuit de cette figure est un schéma électrique simple qui montre la logique hardware permettant de gérer une priorité de service pour quatre périphériques PR_1...PR_4.

III- ENTREES/SORTIES SERIE

III-1- TRANSFERT SERIE

Bien que les systèmes à microprocesseur traitent des données sous forme d'une combinaison de n bits à la fois (n= longueur du mot du système), plusieurs périphériques manipulent l'information sous forme série (un bit à la fois). Vu que le système et son périphérique manipulent l'information sous deux formats différents (parallèle pour le système et série pour le périphérique), un circuit intermédiaire entre les deux parties est indispensable pour assurer la conversion série/parallèle (entrée du processeur) ou parallèle/série (sortie du processeur).

III-1-1- CONVERSION PARALLELE/SERIE OU SERIE/PARALLELE

Dans le cas du transfert de données du système vers le périphérique une conversion parallèle/série est demandée dans le cas contraire c'est la conversion série/parallèle qui est demandée, ceci est obtenu par simple emploi de deux types de registres à décalage comme il est illustré par la figure 17.

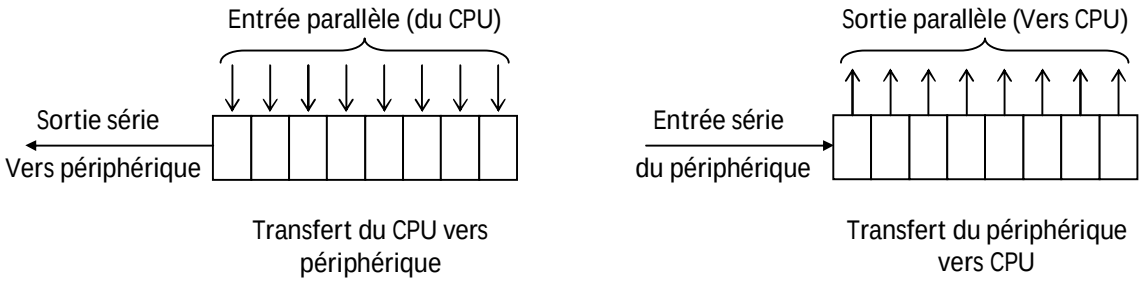


Fig-17 Registre à décalage parallèle série pour la sortie du processeur et série parallèle pour l'entrée du processeur

III-1-2- TRANSMISSION A DISTANCE

Quand la transmission doit être effectuée sur des grandes distances comme dans le cas d'un ordinateur principal et ses terminaux, l'information série est convertie en tonalité audio par des modems (modulateur démodulateur) et elle est envoyée sur une ligne téléphonique dont le principe est représenté par la figure 18.

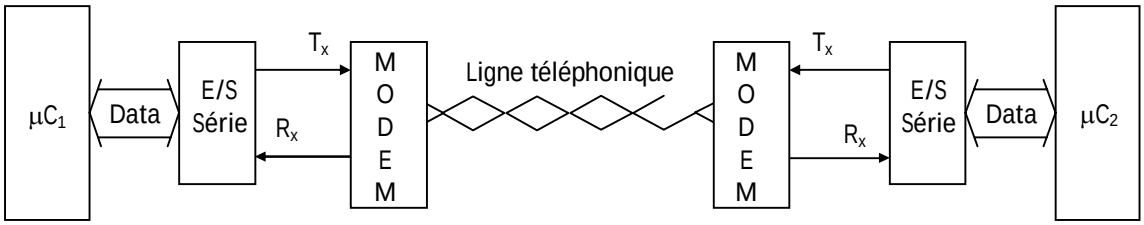


Fig-18. Transmission série à distance entre deux entités. La ligne de transmission se termine de chaque partie par un modem.

III-1-3- TRANSMISSION SERIE ASYNCHRONE

En générale la transmission série est faite sous forme asynchrone (pas d'horloge synchronisation entre émetteur et récepteur). Les caractères sont transmis sous une forme standard ou code c'est ainsi que des équipements de différentes firmes peuvent être utilisés pour un même système sans aucun problème. Le code ASCII (American standard code for information and interchange) est le code le plus utilisé. C'est un code où le caractère est envoyé sur 11 bits dont 1 bit de commencement, 7 bits d'information, 1 bit de parité et 2 bits de fin de caractère comme il est illustré par la figure 19.

L'unité utilisée pour la vitesse dans la transmission série est le *BAUD*.

Baud = Nombre de bits transmis par seconde.

Les vitesses standards sont : 110,300,1200,2400, 9600 et 19200 baud.

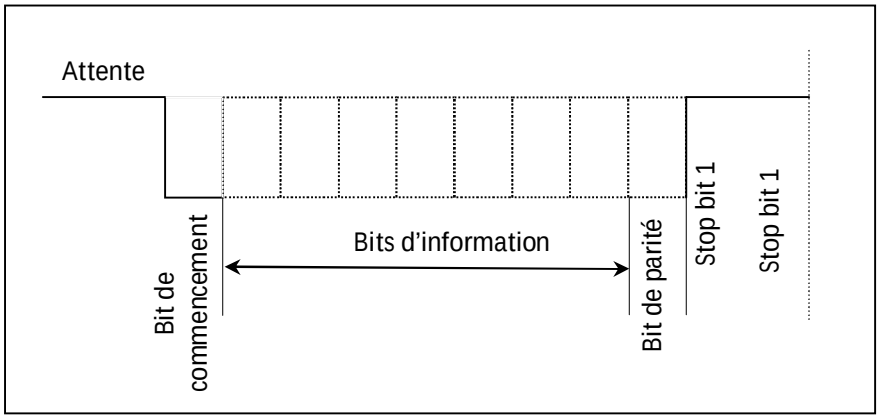


Fig-19 Etat d'une ligne de transmission série lors d'une transmission d'un caractère de 8 bits avec un start bit, un bit de parité et deux bits de stop.

Si Nc = Nombre de bits dans un caractère.

B = baud rate = Nombre de bits transmis par seconde.

t_b = temps nécessaire pour la transmission d'un bit et qui est donné par :

$$t_b = \frac{1}{B}$$

Afin d'éviter le problème causé par des parasites, le temps correspondant à la transmission d'un bit est fixé par une horloge de l'UART de la façon suivante :

$$t_b = K\, t_{cl}$$

avec

K = constante égale à 16 ou 64 le cas où K = 16 est le plus utilisé

t_{cl} = cycle d'horloge

Exemple :

Dix caractères sont transmis par seconde. Chaque caractère est constitué d'un bit de commencement (start bit), 8 bits d'information, un bit de parité et deux bits de fin de caractère. Sachant que 16 périodes d'horloge par temps d'un bit sont utilisées, on demande de calculer :

Vitesse de transmission: 1/ B (baud rate)

Le temps nécessaire à la transmission d'un bit (t_b)

La fréquence de l'horloge avec laquelle travaille l'UART

III-2- CIRCUIT D'E/S SERIE (UART)

Pour la transmission asynchrone les systèmes nécessitent un circuit d'interface qui peut générer et tester le bit de parité, reconnaître et ajouter les bits de commencement et de fin de caractère. Il peut aussi identifier les bits qui véhiculent l'information et fixer la vitesse de transmission.

Pour répondre à ces exigences, un composant électronique programmable UART (Universal Asynchronous Receiver Transmitter) est mis à la disposition de l'utilisateur.

Les fonctions assurées par L'UART sont :

- Conversion Série / Parallèle des données vers le CPU
- Conversion Parallèle / Série des données vers le périphérique.
- Addition du bit de parité, bit de commencement et bit(s) de fin caractère pour la transmission.
- Identification du bit parité, bit de commencement et bit(s) de fin de caractère pour les éliminer à la réception.
- Détection de l'erreur s'il y en à une.

  F_e = frame error

  P_e = parity error

  O_e = overrun error
- Assure le protocole d'échange d'information.
- Assure la synchronisation avec le système durant la réception.

Pour donner une idée claire sur l'UART nous avons ajouté à son schéma de principe de la figure 20, deux autres schémas des figures 21 et 22 qui présentent sous forme séparée le coté réception et le coté émission.

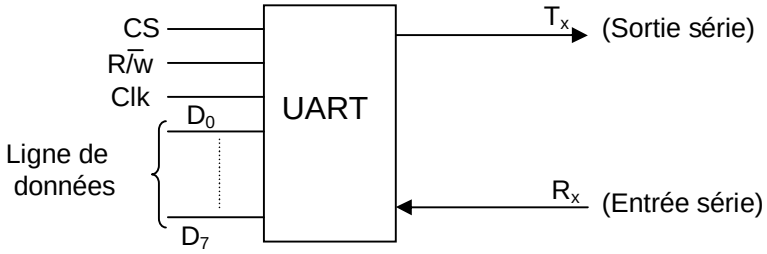


Fig-20 Configuration externe minimale d'un UART. Avec un chip select CS, Read/Write R/W, l'horloge Clk, la donnée D_0...D_7, T_X transmission série et R_X réception

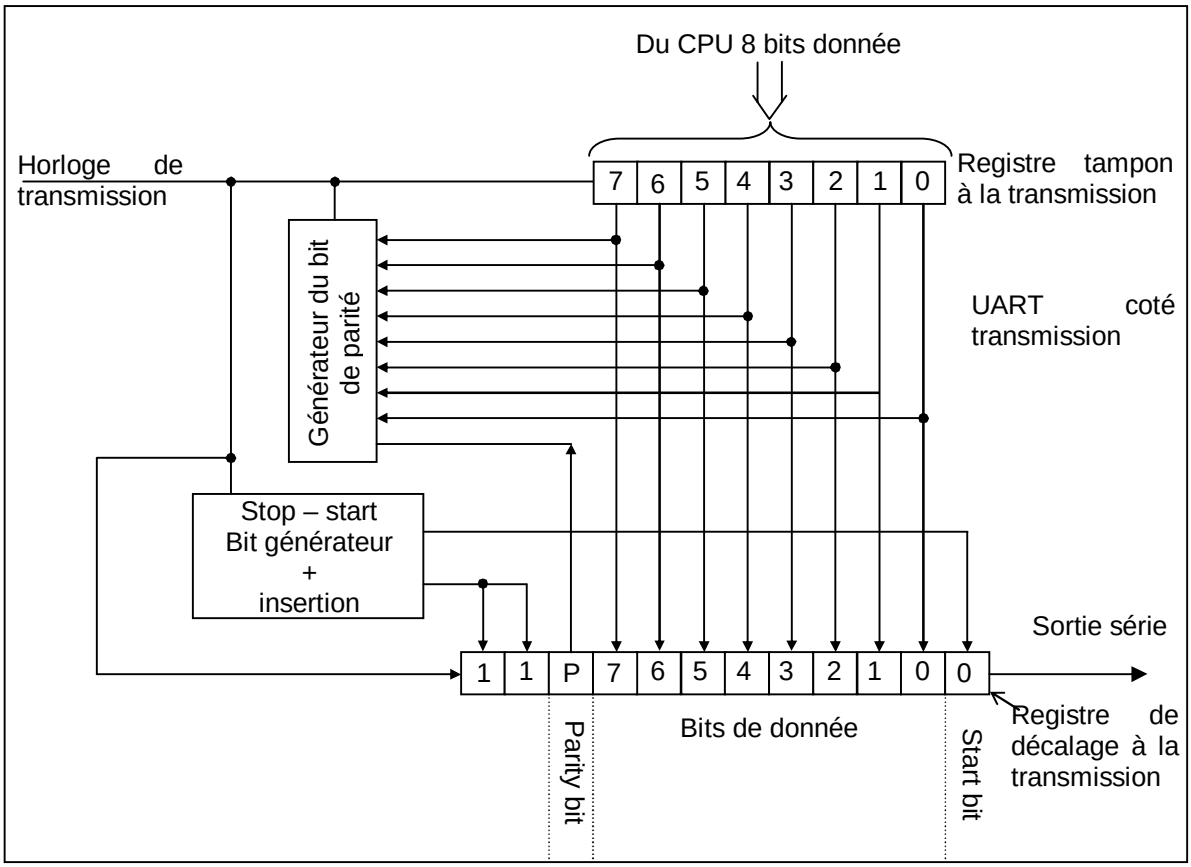


Fig-21 Schéma de principe de la structure interne d'un UART coté émission. On constate à ce niveau la génération du start bit, du bit de parité et des deux bits de stop.

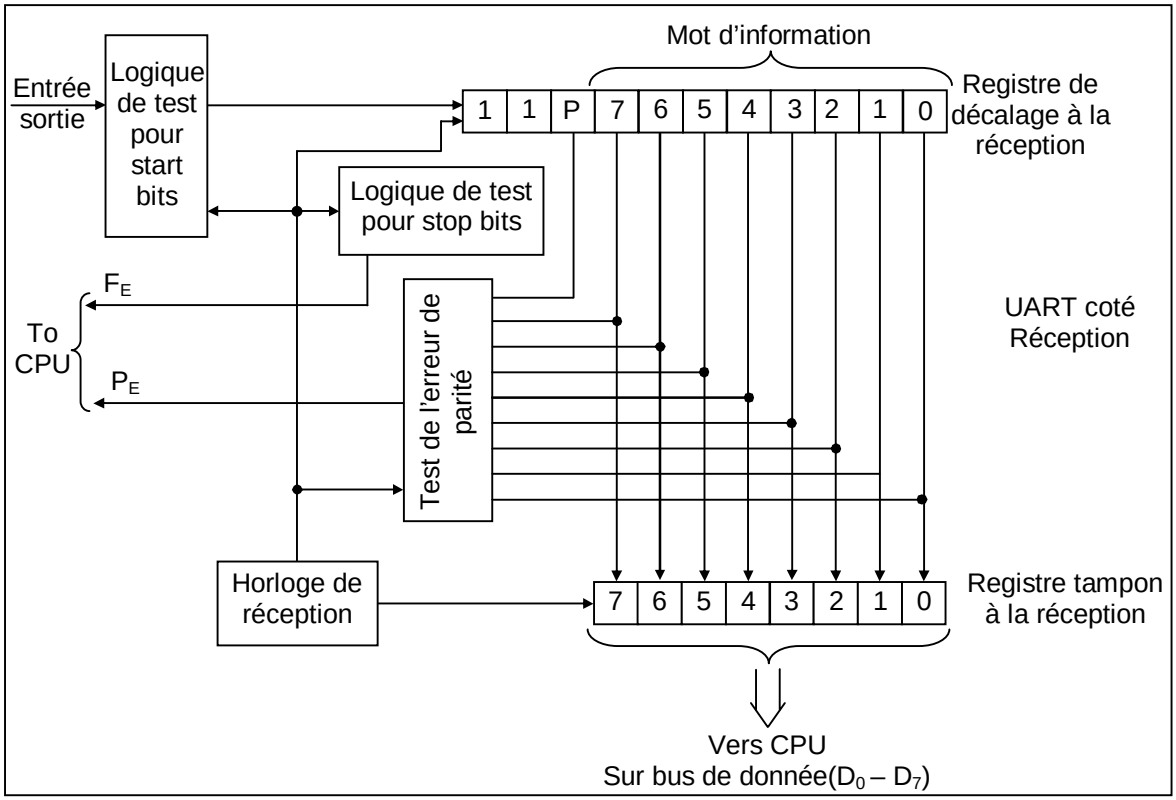


Fig-22 Schéma de principe de la structure interne d'un UART coté réception. On constate à ce niveau que les bits générés du côté émission sont séparés, il ne passe que les bits du caractère émis.

IV- ENTREE/SORTIE PARALLELE

IV-1- INTRODUCTION

Quelques périphériques manipulent les données dans un format parallèle comme dans le cas de convertisseur N/A et A/N. Vu que les périphériques diffèrent par la longueur du mot utilisé, par la vitesse de transfert de données et par les exigences du protocole de transfert, il est difficile de concevoir des interfaces parallèles universelles. Cependant, des circuits d'interfaces parallèles programmables (PIO ou PPI) sont disponibles.

Sous le contrôle d'un programme ces circuits peuvent répondre aux exigences d'une large variété de périphériques.

IV-2- CARACTERISTIQUES D'UN CIRCUIT D'E/S PARALLELE

Le circuit d'entrée/sortie programmable PIO dont on présente sa structure générale sur la figure 23, dispose des caractéristiques suivantes :

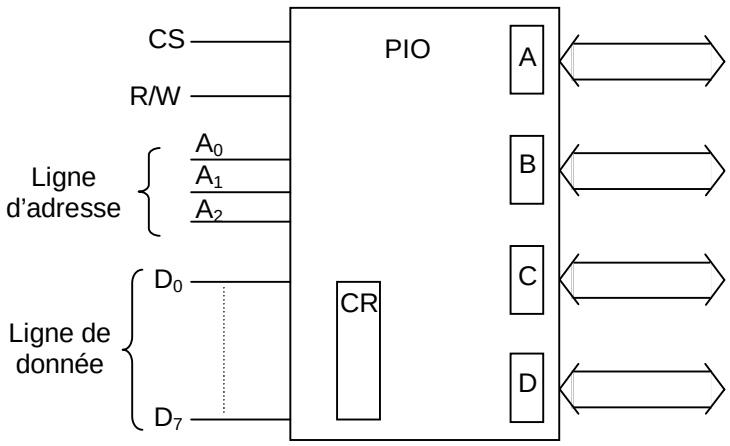


Fig-23 dans cette figure on montre une configuration générale d'un circuit type PIO. Il dispose de broches de communication avec le périphérique à travers ses ports A, B, C, et D et des broches d'interface avec son maître.

- 2 à 4 ports de communication bidirectionnels qui peuvent être programmés en entrée ou en sortie ou les deux en même temps.
- Tampons et verrous de données pour la sortie et l'entrée.
- Signaux d'états et de contrôle pour le protocole de communication.
- L'interfaçage direct avec le CPU; Bus d'adresse, bus de données, bus de contrôle.
- Un registre de contrôle qui permet au CPU de programmer le PIO pour assurer différentes fonctions.
- Un temporisateur piloté par l'horloge du CPU.

- Le temporisateur peut être utilisé pour générer un retard ou mesurer ce retard pour une seule impulsion – dans ce cas le temporisateur est utilisé en mode monostable.
- Il peut être aussi utilisé en mode oscillateur, pour la génération d'un train d'impulsions.

IV-2-1- SEQUENCES D'UNE OPERATION

En utilisant le signal d'horloge, le CPU assure proprement les séquences ou étapes nécessaires pour l'exécution d'une instruction. Pour chaque type d'opération le processeur réserve un temps bien déterminé, ce temps est appelé cycle machine. Il est égale à un nombre entier du cycle d'horloge. La figure 24 illustre 3 cycles machine M1, M2 et M3 qui correspondent respectivement au cycle de recherche du code d'instruction (fetch operation), à la lecture d'une donnée et de l'écriture d'une donnée.

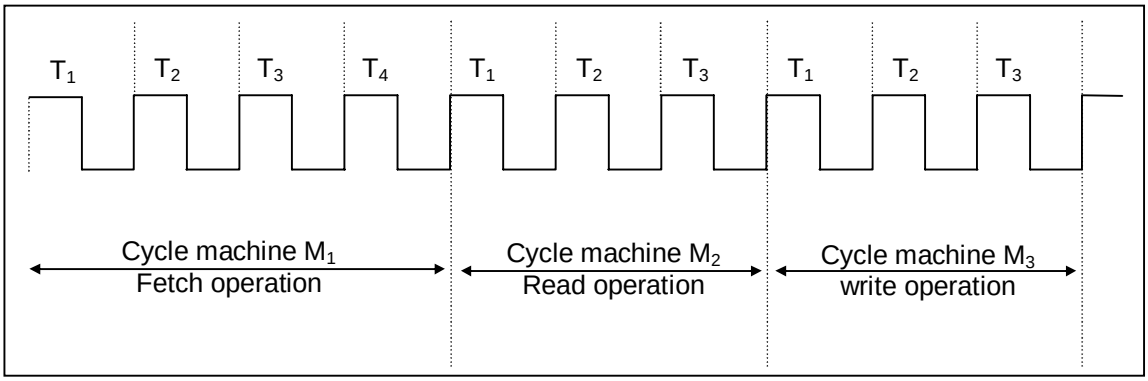


Fig-24 M1, M2 et M3 représentent les cycles machines nécessaires pour un microprocesseur pour sortir chercher une donnée se trouvant dans la mémoire du système.

M_i = cycle de machine

T = période de l'horloge

IV-2-2- CYCLE DE RECHERCHE D'INSTRUCTION (FETCH CYCLE) M_1

La routine de recherche d'instruction s'étale sur une période multiple de la période d'horloge tel que M1=nT (dans le cas dela figure 24 n=4) qui correspond au cycle de la machine M1. Durant ce cycle le MPU effectue une opération lecture de la case mémoire adressée par le contenu du PC. Ainsi l'information lue est placée dans le registre d'instruction RI afin d'être décodée. La période d'horloge par laquelle ce cycle machine dépasse les deux autres est nécessaire pour le temps de décodage de l'instruction.

IV-2-3- CYCLE DE LECTURE

Le cycle de lecture est très similaire au cycle de recherche d'instruction. Le diagramme de la figure 24 montre qu'il y a une différence dans le nombre de périodes d'horloge (4 dans M1 et 3 dans M2).Les différences majeurs sont :

- Le cycle pour lecture qui s'étale sur 3 périodes d'horloges quant au cycle de recherche, il s'étale sur 4. La 4ème période de l'horloge dans M_1 est due à l'opération de décodage supplémentaire.
- Dans le cycle M_1, la lecture est effectuée de la mémoire dont l'adresse est dans PC; Alors que dans le cas du cycle M_2 la lecture peut être de la mémoire dont l'adresse est spécifiée par l'instruction en cours d'exécution ou d'un port d'E/S.

IV-2-4- Cycle d'écriture (Write cycle) M_3

Ce cycle d'écriture est identique au cycle M2 la seule différence est dans la nature d'opération, écriture au lieu de lecture.

IV-3- CIRCUIT D'E/S PARALLELE LE 8155

Comme montré sur le schéma de la figure 25, le 8155 contient :

- Une mémoire statique (RAM) de 256 bytes.
- Deux ports d'E/S A,B à 8 bits et un port d'E/S C à 6 bits.
- Un compteur temporisateur à 14 bits.

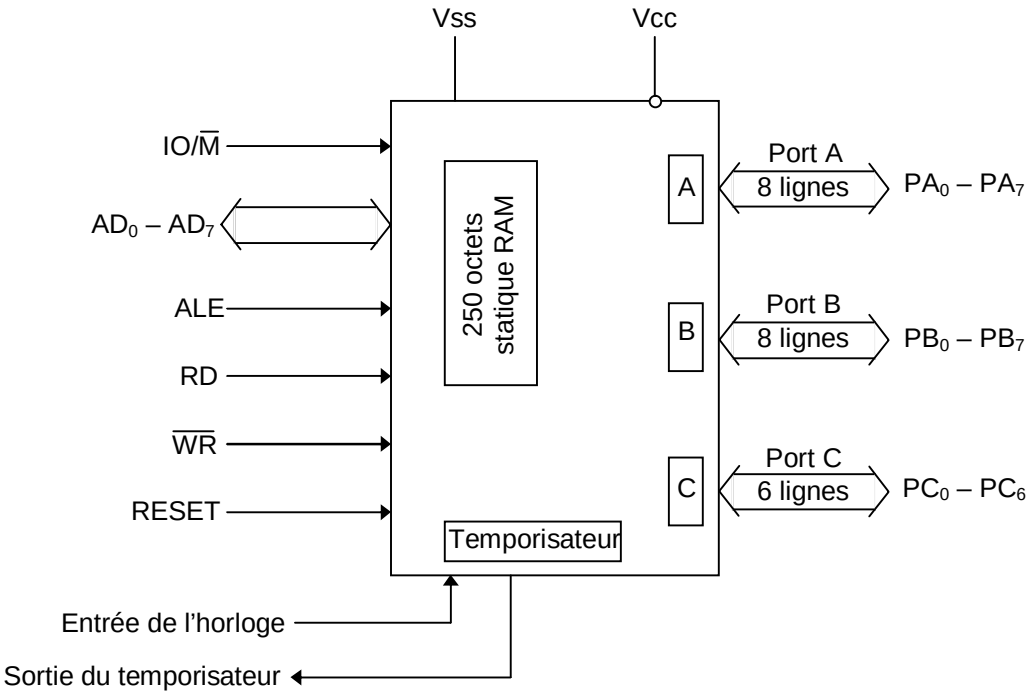


Fig-25 Configuration externe et interne du circuit d' E/S programmable 8155.

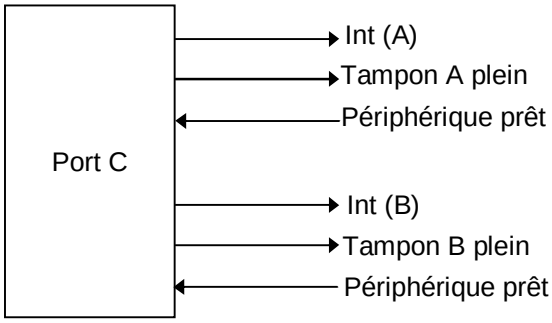


Fig-26 Configuration du port C entant que port de contrôle pour les deux ports A et B

IV-3-1- SECTION ENTREE/SORTIE

La section d'E/S est constituée de cinq (05) registres :

A\ Registre de commande et d'états (C/S)

En réalité, il est composé de deux registres qui sont sélectionnés par la même adresse xxxxx000(20)mais :

- le registre de commande est sélectionné pour l'écriture d'une commande, et il ne peut pas être lu.
- Le registre d'états est sélectionné pour la lecture d'une information sur les états des ports et du temporisateur, l'opération écriture ne peut pas être effectuée sur ce registre.

B\ Registre du port A

C'est un port à usage général qui peut être programmé en entrée ou en sortie par la commande écrite dans le registre vc/s. Les lignes de communication avec l'extérieur de ce port sont au nombre de 8 (PA_0.....PA_7). L'adresse de sélection du registre est xxxxx001(pour le cas du 8155sur le SDK85) dans le laboratoire d'automatisme l'adresse est 2F.

C/ Registre du port B

Ce registre fonctionne d'une façon tout à fait identique à celle du port A. La ligne de communication avec l'extérieur sont au nombre de 8 (PB_0.....PB_7) et l'adresse du sélection du registre est xxxxx010.

D/ Registre du port C

C'est un registre à 6 bits c'est à dire il communique avec l'extérieur par l'intermédiaire de 6 lignes (PC_0-PC_5) l'adresse de sélection de ce port est xxxxx011. Le port à deux modes de fonctionnements :

- Mode où C travaille d'une façon analogue à celle de port A et B.
- Mode où les lignes du port C sont utilisés comme lignes de contrôle pour port A; port B ou les deux, suivant la programmation.

IV-3-2- SECTION TEMPORISATEUR

La section temporisateur est assurée par compteur à décrémentation. A chaque impulsion d'horloge le compteur est décrémenté par 1 jusqu'à ce que le compteur atteigne la valeur finale (TC).

Le nombre à décompter est représenté par une combinaison binaire de 14 bits chargés dans deux registres :

- L'octet du poids faible est chargé dans le registre d'adresse xxxxx100.
- Les six bits de l'octet du poids fort sont chargés dans le registre d'adresse xxxxx101. Les deux plus significatifs bits de ce registre (bit6 et bit7) sont utilisés pour spécifier le mode de fonctionnement du temporisateur.

Fig-27 Le mot à charger dans le timer est un mot de 14 bits contenu dans deux registres. Avec les deux bits de l'octet du poids fort pour le mode de fonctionnement du timer.

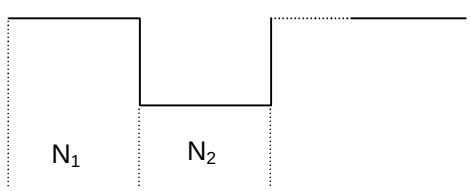
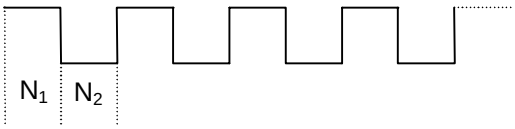
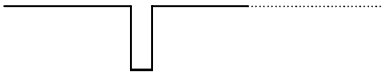
M_1	M_2	Sortie du temporisateur	
0	0		Le temporisateur fait sortir une période d'un signal carré. Si le nombre chargé dans les deux registres : - Est paire $N_1 = N_2$ - Est impaire $N_1 = N_2+1$ (cycle d'horloge)
0	1		Le temporisateur fait sortir un signal carré (continu). Avec même remarque pour N_1 et N_2.
1	0		Une impulsion est générée (une seule) à la fin du décomptage.
1	1		Un train d'impulsion est généré d'une (façon continue).

Fig-28 Tableau résumant le mode de fonctionnement du timer.

IV-3-2-1- PROGRAMMATION DU R/C

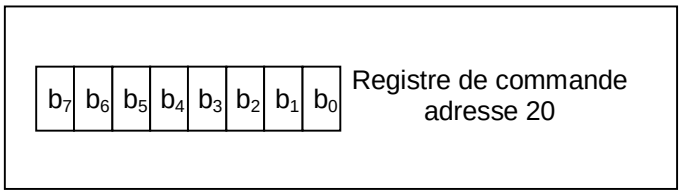


Fig-29 Le registre de contrôle et sa programmation

- Bit b_0 : réservé pour port A

  $b_0 = 0 \Rightarrow$ le port A est programmé en ENTREE.

  $b_0 = 1 \Rightarrow$ le port A est programmé en SORTIE.

- Bit b_1 : réservé pour port B

  $b1 = 0 \Rightarrow$ le port B est programmé en ENTREE.

  $b1 = 1 \Rightarrow$ le port B est programmé en SORTIE.

- Bit b_2 et b_3: réservés pour port C

  $b2 = 0, b3 = 0 \Rightarrow$ le port C est programmé en ENTREE.(ALT1)

  $b2 = 1, b3 = 1 \Rightarrow$ le port C est programmé en SORTIE.(ALT2)

b2= 1 , b3= 0 ⇒ PC0, PC1, PC2 utilisés comme lignes de contrôle pour le port A. avec PC_0 ligne d'interruption, PC_1 ligne d'information (registre tampon du port est plein) , PC_2 ligne utilisé par le périphérique pour dire qu'il est prêt. Les lignes PC_3, PC_4, et PC_5 sont des sorties.

b2= 0 , b3= 1 ⇒ PC0, PC1, PC2 utilisés comme lignes de contrôle pour le port A.

PC_3, PC_4, PC_5 utilisés comme lignes de contrôle pour le port B.

- Bit b_4 : réservé pour port A

  b_4= 1 ⇒ port A est utilisé en mode avec interruption.

  b_4= 0 ⇒ port A est utilisé en mode sans interruption.

- Bit b_5 : réservé pour port B

  b_5= 1 ⇒ port B est utilisé en mode avec interruption.

  b_5= 0 ⇒ port B est utilisé en mode sans interruption.

- Bit b_6 et b_7: réservés au temporisateur

  b_6= 0 , b_7= 0 ⇒ n'effectue aucune opération, est sur le temporisateur.

  b_6= 1 , b_7= 0 ⇒ ARRET- si le compteur est entrain de compter il sera arrêté; sinon aucune opération n'est effectuée.

  b_6= 0 , b_7= 1 ⇒ ARRET après que le compte soit terminé. Si le compte n'a pas encore commencé aucune opération n'est effectuée.

  b_6= 1 , b_7= 1 ⇒ COMMENCE le compte.

  - Si le compte n'a pas encore commencé : charge les deux registres par la longueur du compte, et le mode de sortie du temporisateur et il déclenche le compteur.
  - Si le compteur est en cours, alors après la fin du compte, les deux registres seront chargés ainsi que le mode de sortie du temporisateur puis le compteur sera déclenché.

- Registre d'état :

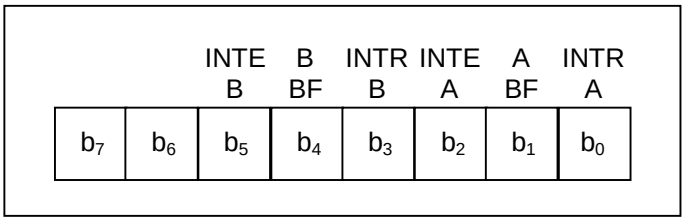


Fig-30 Le registre d'état est un registre de 8 bits avec chaque bit est réservé à donner une idée sur l'état du processeur.

b_0 : Demande d'interruption port A.

b_1 : Registre tampon du port A est plein ou vide (ENTREE/SORTIE).

b_2 : Port A interruption validée.

b_3 : Demande d'interruption port B.

b_4 : Registre tampon du port B est plein ou vide (ENTREE/SORTIE).

b_5 : Port B interruption validée.

b_{50} : C'est un bit est mis à un à chaque fois le compte final est atteint (temporisateur). Il sera remis à zéro par simple lecture du registre d'état ou par le RESET. C'est un bit d'interruption.

IV-4- CIRCUIT D'E/S PARALLELE LE PIA 6820 ou PIA 6821

Le PIA est un circuit intégré de 40 broches, il a été conçu pour le contrôle du transfert parallèle des données entre le système à microprocesseur et des éléments externes au système. L'architecture interne et la configuration externe du PIA sont données par la figure 31.

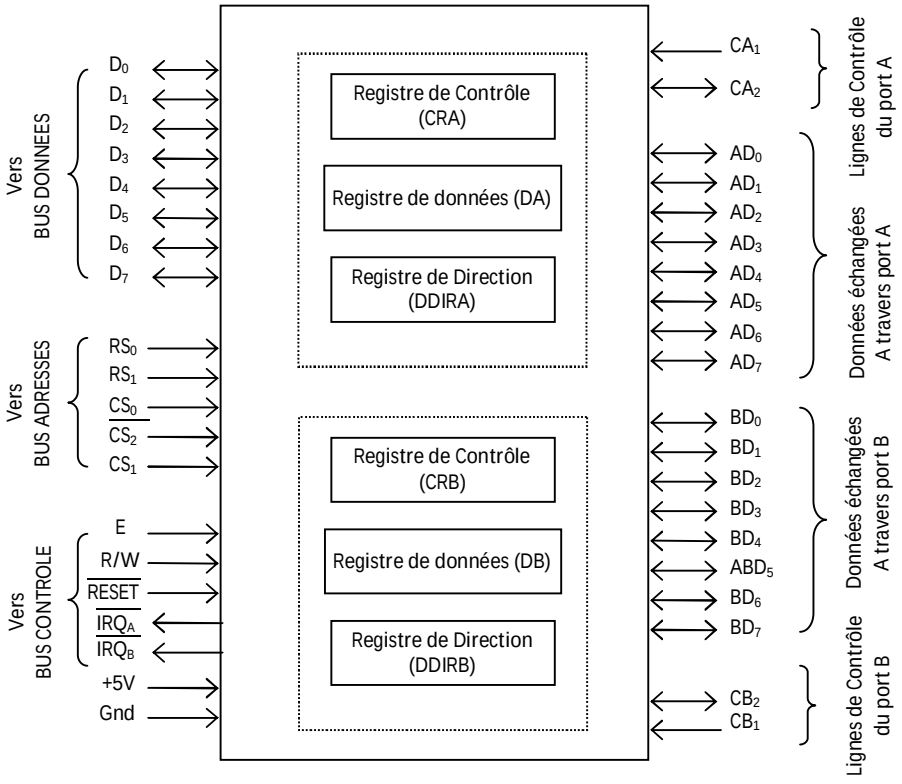


Fig-31 Configurations interne et externe d'un PIA

Comme on peut le constater sur la figure 31, le PIA peut être scindé en deux parties identiques en fonctionnement mais qui représentent en réalité une petite différence au niveau des courants de sortie. Chacun des deux ports (port A et port B) permet à un système à microprocesseur de communiquer avec son extérieur par un échange d'information sur 8 bits. Les deux lignes de contrôle (CA1 et CA_2 pour port A, CB_1 et CB_2 pour port B) assurent le handshake dans le cas où l'élément externe n'est pas compatible en vitesse avec le microprocesseur.

IV-4-1 Les registres du PIA

Les deux ports A et B d'un PIA peuvent être exploités d'une façon indépendante. Chacun des deux ports contient 3 registres ; un registre de contrôle (CRA ou CRB), un registre de données (DA ou DB) et un registre de direction (DDIRA ou DDIRB).

La communication du microprocesseur avec le PIA se fait de la même façon que l'accès à une zone mémoire de 4 cases mémoires grâce à l'adresse véhiculée par les lignes d'adresses A_0 et A_1 respectivement connectées à RS_0 et RS_1.

Avant toute communication entre le microprocesseur et le PIA , les registres de contrôle CRA ou CRB et DDRA et DDRA (suivant le port à utiliser) doivent être configurés suivant le mode et la direction du transfert de la donnée qui va avoir lieu.

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	Port A			Port B		
X				X				X				X	X	0	1	CRA					
X				X				X				X	X	0	0		DDIRA	DA			
X				X				X				X	X	1	1				CRB		
X				X				X				X	X	1	0					DDIRB	DB

a- Le registre de contrôle CR

A l'alimentation du système à microprocesseur, sa remise à zéro ou à la remise à zéro du PIA tous ses registres sont mis à zéro. Par conséquent, les lignes AD0----AD7 et BD0----BD7, CA2 et CB2 sont des entrées et toutes les interruptions sont inhibées. Comme le bit 2 du registre de contrôle est mis zéro :

- ✓ l'adresse A0=0 et A1=0 est attribuée au registre de direction DDRA
- ✓ l'adresse A1=1 et A0=0 est attribué au registre de direction DDRB

les tableaux ci-dessous montrent les registres de contrôle CRA et CRB et l'interprétation de leurs bits.

Fonctions des bits du registre de contrôle port A CRA							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IRQ1	IRQ2	Commande CA2			DDRA Selection	commande CA1	

Configuration des bits 5 4 et 3 du CRA			CA2 demande d'interruption	Drapeau demande d'interruption IRQ2	Demande d'interruption coté port A vers µP
Bit 5	Bit 4	Bit3		Bit 6	
0	0	0	Activée par front descendant (↓)	Mis à 1 sur front (↓)de CA2	(IRQ_A) inhibée elle reste dans niveau haut
0	0	1	Activée par front descendant (↓)	Mis à 1 sur front (↓)de CA2	(IRQ_A) passe à un niveau bas une fois le bit CRA bit 6 passe à 1
0	1	0	Activée par front montant (↑)	Mis à 1 sur front (↑)de CA2	(IRQ_A) inhibée elle reste dans niveau haut
0	1	1	Activée par front montant (↑)	Mis à 1 sur front ↑de CA2	(IRQ_A) passe à un niveau bas une fois le bit CRA bit 6 passe à 1

Configuration des bits 5 4 et 3 du CRA			Mode	Description de l'effet de la configuration des bit 5, 4 et 3 dur CRA
Bit 5	Bit 4	Bit3		
1	0	0	Poignée de main en lecture	✓ La transition sur CA1 (requête d'interruption CA2 à 1. ✓ L'instruction de lecture du port A met CA2 à 0
1	0	1	Sortie d'impulsion	La lecture du port A met CA2 à 0 pendant un cycle.
1	1	0	Par programme	CA2 mis à 0 tant que Bit 3 de CRA est 0
1	1	1	Par gramme	CA2 mis à 1 tant que Bit 3 de CRA est 1

Configuration des bits 0 et 1 CRA		Front actif pour CA1	drapeau demande d'interruption IRQ1	Requête d'interruption par port A vers µP
Bit 1	Bit 0		Bit 7	
0	0	Activée par front descendant (↓)	Mis à 1 sur front (↓)de CA1	(IRQ_A) inhibée elle reste dans niveau haut
0	1	Activée par front descendant (↓)	Mis à 1 sur front (↓)de CA1	(IRQ_A) passe à un niveau bas une fois le bit CRA bit 7 passe à 1
1	0	Activée par front montant (↑)	Mis à 1 sur front (↑)de CA1	(IRQ_A) inhibée elle reste dans niveau haut
1	1	Activée par front montant (↑)	Mis à 1 sur front ↑de CA1	(IRQ_A) passe à un niveau bas une fois le bit CRA bit 7 passe à 1

Fonctions des bits du registre de contrôle port B CRB							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IRQ1	IRQ2	Commande CB2			DDRB Selection	commande CB1	

Configuration des bits 5 4 et 3 du CRB			CB2 demande d'interruption	Drapeau demande d'interruption IRQ2	Demande d'interruption coté port B vers µP
Bit 5	Bit 4	Bit3		Bit 6	
0	0	0	Activée par front descendant (↓)	Mis à 1 sur front (↓)de CB2	(IRQ_B) inhibée elle reste dans niveau haut
0	0	1	Activée par front descendant (↓)	Mis à 1 sur front (↓)de CB2	(IRQ_B) passe à un niveau bas une fois le bit CRB bit 6 passe à 1
0	1	0	Activée par front montant (↑)	Mis à 1 sur front (↑)de CB2	(IRQ_B) inhibée elle reste dans niveau haut
0	1	1	Activée par front montant (↑)	Mis à 1 sur front ↑de CB2	(IRQ_B) passe à un niveau bas une fois le bit CRB bit 6 passe à 1

Configuration des bits 5 4 et 3 du CRB			Mode	Description de l'effet de la configuration des bit 5, 4 et 3 dur CRB
Bit 5	Bit 4	Bit3		
1	0	0	Poignée de main en lecture	✓ La transition sur CB1 (requête d'interruption CB2 à 1. ✓ L'instruction de lecture du port B met CB2 à 0
1	0	1	Sortie d'impulsion	La lecture du port B met CB2 à 0 pendant un cycle.
1	1	0	Par programme	CB2 mis à 0 tant que Bit 3 de CRB est 0
1	1	1	Par gramme	CB2 mis à 1 tant que Bit 3 de CRB est 1

Configuration des bits 0 et 1 CRB		Front actif pour CB1	drapeau demande d'interruption IRQ1	Requête d'interruption par port B vers µP
Bit 1	Bit 0		Bit 7	
0	0	Activée par front descendant (↓)	Mis à 1 sur front (↓)de CB1	(IRQ_B) inhibée elle reste dans niveau haut
0	1	Activée par front descendant (↓)	Mis à 1 sur front (↓)de CB1	(IRQ_B) passe à un niveau bas une fois le bit CRB bit 7 passe à 1
1	0	Activée par front montant (↑)	Mis à 1 sur front (↑)de CB1	(IRQ_B) inhibée elle reste dans niveau haut
1	1	Activée par front montant (↑)	Mis à 1 sur front ↑de CB1	(IRQ_B) passe à un niveau bas une fois le bit CRB bit 7 passe à 1

b- Le registre de direction DDIR

Chacun des deux ports A et B contient un registre de direction, respectivement DDRA et DDRB, dont le nombre (8) de bits correspond au nombre des lignes de communication avec l'extérieur du port concerné. Chaque bit du registre sert à programmer ligne correspondante comme entrée

ou sortie. Un 1 écrit dans un bit du DDRA programme la ligne correspondante comme sortie, et un 0 la programme en entrée.

b- Le registre de données D

Le registre de données DA ou DB sert comme une case mémoire dans laquelle on écrit la donnée à faire sortir vers l'extérieur à travers le port correspondant quand ce dernier est programmé en sortie.

Si le port est programmé en entrée la donnée qui arrive de l'extérieur est placée dans le registre par sa lecture la donnée est récupérée.

V-MICROPROCESSEUR 68000 DE MOTOROLA

V-1 INTRODUCTION

Incontestablement, parmi les percées technologiques les plus significatives, a été l'émergence des circuits intégrés à haut niveau d'intégration.

L'extension des possibilités de traitement des circuits intégrés tend à faire croître le nombre de connexions externes nécessaires, conduisant à des boîtiers plus grands. On rencontre souvent des microprocesseurs 8bits dans des boîtiers de 40 broches. Toutefois, il est difficile de faire incorporer des microprocesseurs de 16bits dans des boîtiers de 40 broches sans faire appel à un multiplexage temporel.

Cela dit, avec un tel multiplexage, un ensemble de broches est utilisé pour différentes fonctions mais pas à la fois. Un tel partage temporel est une solution valable, mais elle se traduit inévitablement par une diminution de vitesse de traitement, et des interfaçages externes plus complexes.

Les microprocesseurs 8bits à l'exemple du 6800, 8085 et le Z80 sont très bien adaptés quant à une série d'applications. Toutefois la limitation liée à la longueur 8bits du mot est suffisante pour les rendre inadaptés pour des applications assez complexes. Les microprocesseurs 16bits ont fait leurs apparitions, ce sont des processeurs puissants et sophistiqués avec des espaces d'adressage de l'ordre du Méga Octets remédiant à quelques handicapes des processeurs 8bits.

Dans son temps le MC 68000 était l'une des conceptions les plus complexes jamais tentées sur un seul circuit intégré. C'est un microprocesseur 16bits qui est né du projet MACSS (MOTOROLA ADVANCED COMPUTER SYSTEM on SILICON) qui a été lancé dans le courant de l'année 1976. Avec comme but à atteindre est surtout la conception d'un microprocesseur monolithique dont les performances reposeraient sur deux points essentiels, il s'agit bien entendu de la simplicité et de l'orthogonalité. Cette dernière caractéristique consiste en la banalisation des registres internes vis à vis des modes d'adressage et des instructions en général.

Côté fabrication, pour le MC 68000 la technique utilisée pour cet effet (en l'occurrence HMOS) devait permettre l'amélioration de performances telles que de réduire la surface d'une cellule de mémoire ainsi que le facteur de qualité définie par le produit: consommation x vitesse.

Livré au cours de l'année 1979, le MC 68000 a vu la plupart des objectifs atteints.

V-2 - CONFIGURATION EXTERNE

La structure des bus du MC 68000 est très classique. Le microprocesseur 68000 se présente sous forme d'un boîtier de 64 broches, figure 31, qui regroupent un bus d'adresses, un

bus de données et un bus de contrôle avec 2 broches pour Vcc, 2 broches pour Vss et 1 broche pour l'horloge.

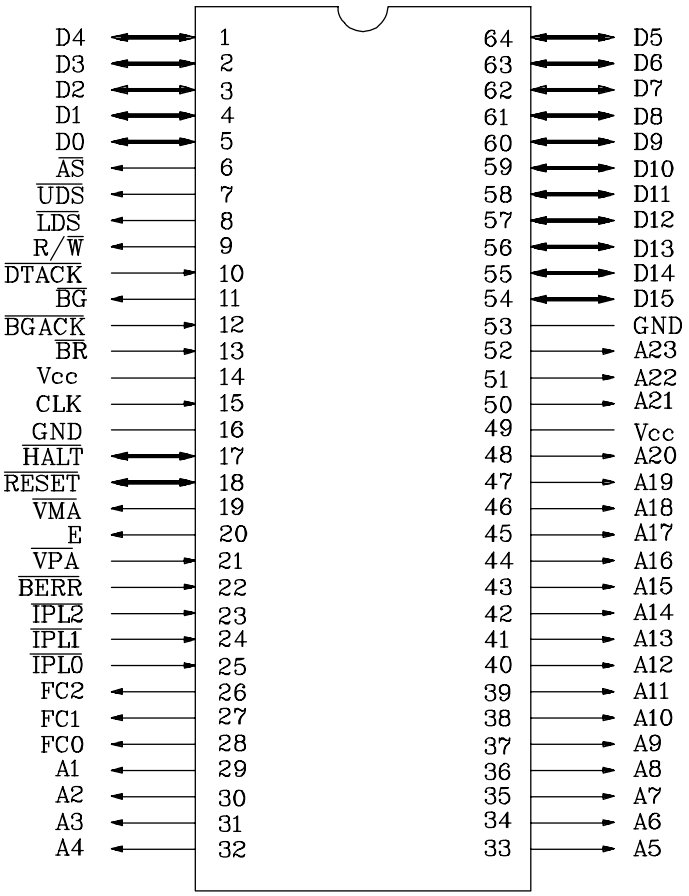


Fig-31- Brochage du 68000

V-2-1- BUS D'ADRESSES

Le bus d'adresses A1...A23 est non multipléxé et unidirectionnel. A0 est générée intérieurement et grâce à une logique de décodage de signaux sont générés en sortie respectivement $\overline{LDS}$ (lower adresse strobe) et $\overline{UDS}$ (upper data strobe), et qui ont comme fonction de définir le type de l'information concernée par le cycle en cours comme montré sur le tableau tab-1. Donc avec ces 23 lignes d'adresses avec $\overline{LDS}$ et $\overline{UDS}$ le MC 68000 est capable d'adresser un espace mémoire de 16 Moctets.

En cas d'interruption les lignes d'adresses A1, A2 et A3 font sortir le niveau de l'interruption demandée. Le reste du bus passe à l'état haut.

R / $\overline{W}$	$\overline{LDS}$	$\overline{UDS}$	OCTET D0--D7	OCTET D8--D15	OPERATION
0	0	1	validées	non validées	écriture octet
1	0	1	validées	non validées	lecture octet
0	1	0	non validées	validées	écriture octet
1	1	0	non validées	validées	lecture octet
0	0	0	validées	validées	écriture mot
1	0	0	validées	validées	lecture mot

Tab-1- Sélection du type de données pour écriture ou lecture

V-2-2- BUS DE DONNEES

Ce bus est constitué de 16 lignes D0...D15 bidirectionnelles permettant ainsi de véhiculer l'information de et vers le microprocesseur sous forme:

- Octet sur D0...D7 Octet bas
- Octet sur D8...D15 Octet haut
- Mot sur D0...D15 Octet bas et Octet haut en même temps.
- D0...D7 permettent aussi de véhiculer le numéro de vecteur d'interruption en cas de requête d'interruption.

V-2-3- BUS DE CONTROLE

L'ensemble des lignes de contrôle peuvent être réparties en six groupes:

- Contrôle de bus asynchrone: $\overline{AS}$, R / $\overline{W}$, $\overline{UDS}$, $\overline{LDS}$ et $\overline{DTACK}$.
- Contrôle d'arbitrage du bus: $\overline{BR}$, $\overline{BG}$ et $\overline{BGACK}$.
- Contrôle du système: $\overline{BERR}$, $\overline{RESET}$ et $\overline{HALT}$.
- Contrôle du bus synchrone: E, $\overline{VPA}$ et $\overline{VMA}$.
- Contrôle d'états du processeur: FC0, FC1 et FC2.
- Niveau de la requête d'interruption: $\overline{IPL0}$, $\overline{IPL1}$ et $\overline{IPL2}$.

1- CONTROLE DU BUS ASYNCHRONE

L'ensemble des lignes de ce groupe intervient suivant un ordre bien déterminé lors d'un transfert de données tel qu'il est illustré par les deux exemples donnés sur les figures 32 et 33.

- $R / \overline{W}$ est une sortie à trois états. Elle fixe le sens de transfert de l'information placée sur le bus de données. Il s'agira d'une écriture si $R / \overline{W}$=0 ou d'une lecture si $R / \overline{W}$=1.

- $\overline{AS}$ (Adresse Strobe) est une sortie à trois états. Elle permet de verrouiller une adresse valide sur le bus d'adresses. Une fois l'adresse est placée sur le bus correspondant le microprocesseur insert $\overline{AS}$ ($\overline{AS}$=0) pour indiquer au périphérique adressé que l'adresse placée sur le bus est validée. Il est généralement exploité par la logique de décodage d'adresse permettant de sélectionner l'élément adressé.

- $\overline{LDS}$ / $\overline{UDS}$ (Lower Data Strobe et Upper Data Stobe respectivement) sont toutes deux des sorties à trois états. L'activation de l'une ou des deux permettent de définir le type de donnée lors d'un transfert comme montré sur le tableau Tab-1.

- $\overline{DTACK}$ (Data Transfert Acknowledge) est une entrée. A l'état actif cette ligne informe le microprocesseur qu'un transfert de données est réalisé quelque soit son sens de transfert.

En cas de lecture, elle permet de verrouiller les données et le cycle se termine.

En cas d'écriture, son activation est reconnue par le processeur comme cycle terminé.

Les figures 32 et 33 montrent les chronogrammes d'écriture et de lecture d'une donnée vers et de la mémoire.

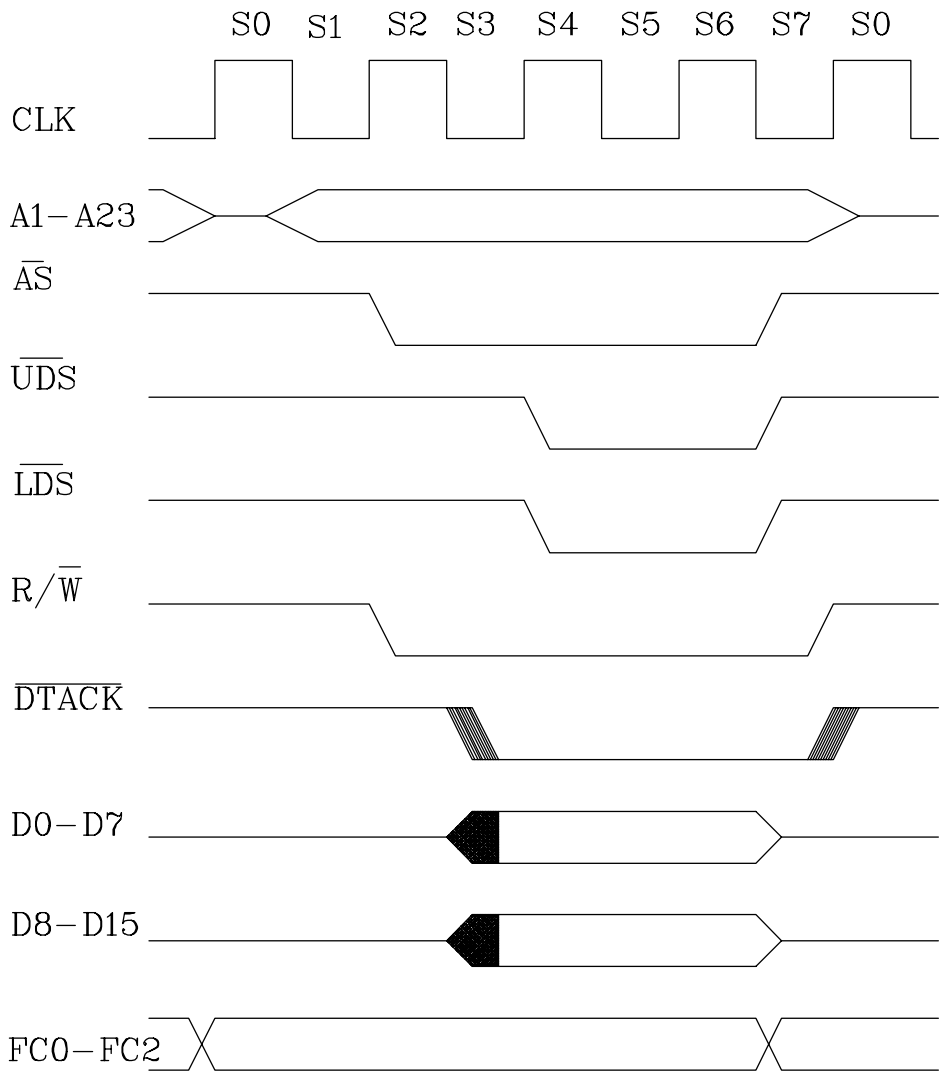


Fig-32- Cycle écriture

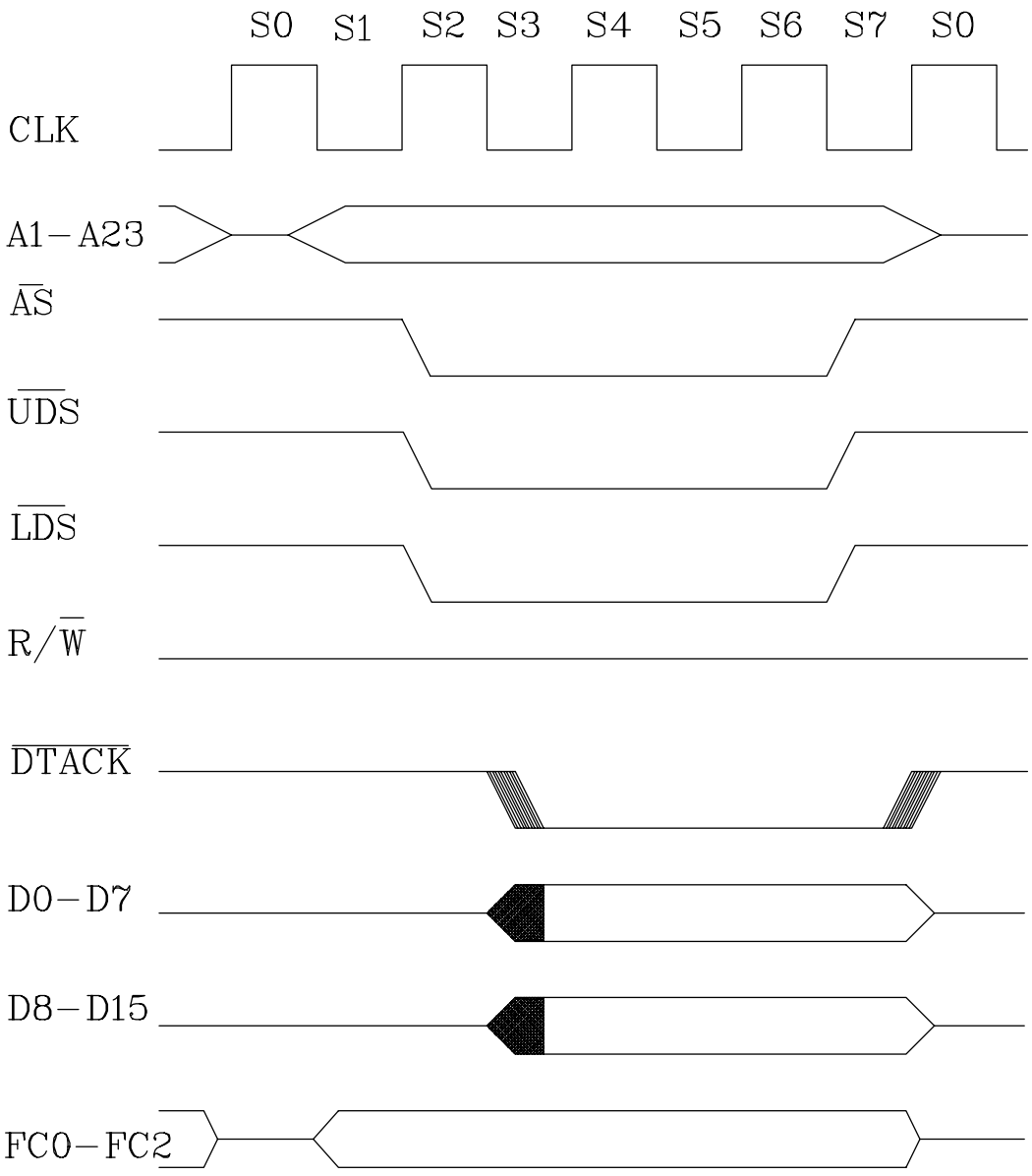


Fig-33- Cycle lecture

2- CONTROLE D'ARBITRAGE DU BUS

Ce sont des lignes qui permettent de transférer le contrôle des bus du système à un élément du circuit autre que le processeur maître.

- $\overline{BR}$ (Bus Request) est une entrée qui à l'état bas traduit une demande d'accès aux bus système. Cette requête peut provenir, dans un environnement multiprocesseur, d'un microprocesseur qui désire prendre le contrôle des bus du système.

- $\overline{BG}$ (Bus Grant) est une sortie active bas. Un état bas sur cette ligne informe le périphérique qui a insérer $\overline{BR}$ que sa requête est prise en considération.

- $\overline{\text{BGACK}}$ (Bus Grant Acknowledge) est une entrée active bas. Activée par le demandeur de contrôle des bus pour dire au microprocesseur maître que les bus sont sous son contrôle.

3- CONTROLE DU SYSTEME

- $\overline{\text{RESET}}$ est une ligne bidirectionnelle active bas.

En entrée par son état active elle permet d'initialiser le système y compris le microprocesseur.

En sortie elle est générée par le microprocesseur suite à l'exécution d'une instruction de type RST. Le MC 68000 initialise tous les éléments de son environnement tout en restant indifférent.

- $\overline{\text{HALT}}$ bidirectionnelle active bas.

Activée en entrée, le microprocesseur termine le cycle en cours puis place toutes les lignes à trois en haute impédance et le reste à l'état inactif.

Activée en sortie, le microprocesseur informe son environnement qu'il est à l'arrêt.

- $\overline{\text{BERR}}$ (Bus Error) est une entrée active bas. L'état active sur cette ligne indique qu'une anomalie est survenue durant le cycle machine en cours. Cette anomalie peut être:

- L'élément adressé ne répond pas.
- Echec dans l'acquisition du numéro du vecteur d'interruption.
- Accès à la mémoire illégale.

4- CONTROLE DU BUS SYNCHRONE

Les lignes de ce groupe permettent de maintenir une complète compatibilité entre le microprocesseur asynchrone MC68000 est la famille synchrone MC6800.

- E (Enable) c'est un signal qui permet de synchroniser le transfert entre le MC68000 et les périphériques de la famille MC6800. Il est généré à partir d'une horloge flottante interne au MC68000 et dont la période est 10 fois la période de l'horloge injectée au MC68000.

- $\overline{VPA}$ (Valid Peripheral Adresse) est une entrée activée par un élément de la famille MC6800 pour avertir le microprocesseur maître que le transfert doit être synchronisé par E.

- $\overline{VMA}$ (Valid Mémory Adresse) En réponse à le MC68000 se synchronise puis active la sortie afin de confirmer l'adresse véhiculée par le bus d'adresse. Cette ligne est généralement utilisée pour la sélection des circuits de la famille MC6800.

5- CONTROLE D'ETATS DU PROCESSEUR

Les lignes FC0, FC1 et FC2 (Function Code) sont des sorties à trois états. Elles sont positionnées au début de chaque cycle machine et la logique affichée sur ces lignes permet de définir l'état du processeur comme illustré par le tableau Tab-2.

FC 2	FC 1	FC 0	ETAT	MODE
0	0	0	RESERVE	UTILISATEUR
0	0	1	DONNEES	UTILISATEUR
0	1	0	PROGRAMME	UTILISATEUR
0	1	1	RESERVE	UTILISATEUR
1	0	0	RESERVE	SUPERVISEUR
1	0	1	DONNEES	SUPERVISEUR
1	1	0	PROGRAMME	SUPERVISEUR
1	1	1	INTERRUPTION	SUPERVISEUR

Tab-2- Etats du processeur suivant FC0, FC1 et FC2

6- DEMANDE D'UNE INTERRUPTION

L'état logique des lignes $\overline{IPL0}$, $\overline{IPL1}$ et $\overline{IPL2}$ permet au microprocesseur de déterminer le niveau du demandeur d'interruption. Si ce niveau n'est pas strictement supérieur au masque fixé au niveau du registre d'état, la requête d'interruption sera rejetée.

V-3 Modes d'adressage du 68000

Le MC68000 dispose de 56 instructions, en comparaison avec le MC6800 est un nombre très réduit. Ce qui rend ce jeu d'instructions très performant c'est les modes d'adressage qui lui sont associés.

Un mode d'adressage définit le moyen utilisé par une instruction à chercher une donnée. Le MC68000 dispose de 14 modes d'adressage regroupés comme suit:

Adressage registre direct.

Adressage registre Indirect.

Adressage absolu.

Adressage immédiat.

Adressage relatif au compteur programme.

Adressage implicite.

V-3-1-Adressage registre direcT

Dans ce mode d'adressage on trouve deux cas: adressage direct registre de donnée et adressage direct registre d'adresse.

a- Adressage direct registre de donnée

Dans ce mode d'adressage l'opérande est spécifié par le contenu d'un registre de donnée. Dans ce mode l'instruction, la taille de l'opérande peut être octet, mot ou long mot.

Dans le cas de transfert de donnée de type d'octet ou mot les bits non concernés du registre restent inchangés.

Exemple

MOVE.W D1, D4 → Transférer un mot du registre D1 vers le registre D4.

0 1 2 5 1 A B 4	→D1		0 1 2 5 1 A B 4	→D1	
0 0 0 0 0 0 0 0	→D4		0 0 0 0 1 A B 4	→D4	
Avant exécution			Après exécution		

ADD.B D0, D3 → L'addition d'un octet du registre D0 au contenu du registre D3.

0 1 2 5 1 A B 4	→D0		0 1 2 5 1 A B 4	→D0	

00000000	→D3	000000B4	→D3
Avant exécution		Après exécution	

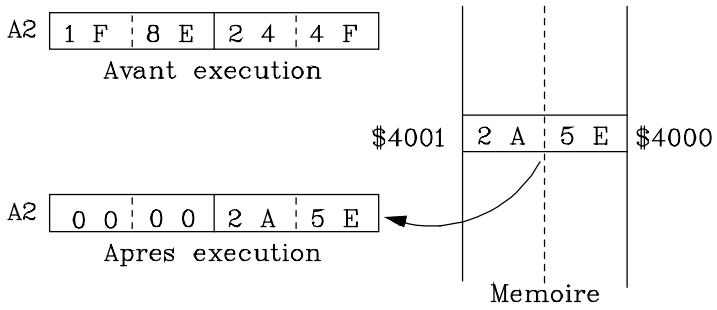
b- Adressage direct registre d'adresse

Ici l'opérande est défini par le contenu d'un registre d'adresse. L'instruction travaille exclusivement sur mot ou long mot.

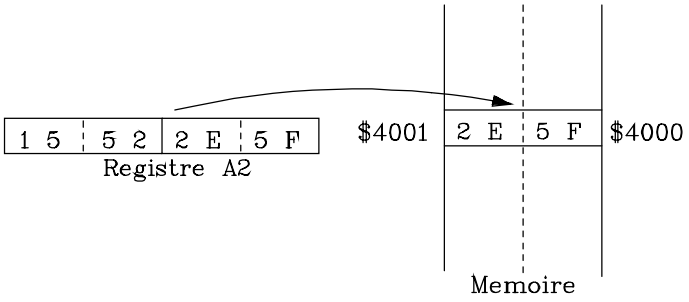
Dans le cas de transfert de mot vers un registre d'adresse il y aura extension de signe sur les 32 bits du registre.

Exemple:

MOVE.W $4000, A2 Transfert d'un mot de la mémoire au registre A2. On remarque que le mot le plus significatif est une extension de signe



MOVE.W A2, $4000 Transfert d'un mot du registre A2 à la mémoire



V-3-2-Adressage registre indirect

Dans ce mode d'adressage l'opérande est spécifié par son adresse qui est donnée par le contenu d'un registre d'adresse.

Ce mode d'adressage regroupe les cas suivants:

- Adressage registre indirect.
- Adressage registre indirect avec postincrémentation.

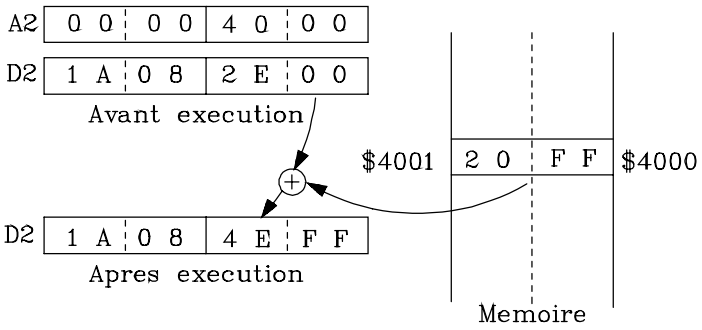
- Adressage registre indirect avec prédécrémentation.
- Adressage registre indirect avec déplacement.
- Adressage registre indirect avec déplacement et index court.
- Adressage registre indirect avec déplacement et index long.

a- Adressage registre indirect

L'adresse effective de l'instruction est donnée par le contenu de la case mémoire pointée par le contenu du registre d'adresse spécifié par l'instruction en cours. La taille de donnée manipulée peut être octet, mot ou long mot.Dans le cas du mot ou du long mot le contenu du registre d'adresse doit obligatoirement être paire.

Exemple:

ADD.L (A0), D2 Addition du long mot adressé par le contenu de A0 au registre D2.



b- Adressage registre indirect avec déplacement

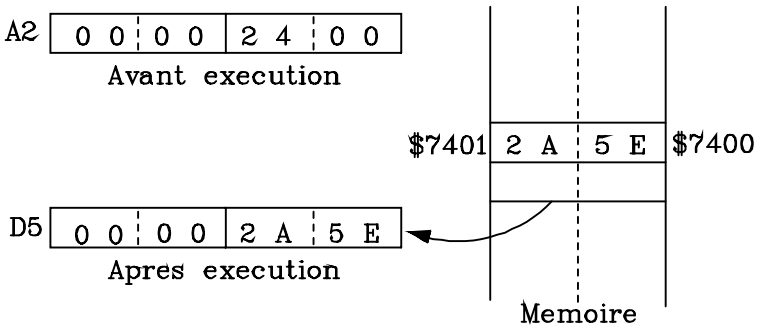
L'adresse de l'opérande est obtenue à partir de l'addition de:

Contenu d'un registre d'adresse Ai.

Déplacement donné par 16 avec extension de signe.

Exemple:

MOVE.W $5000 (A2), D5 Transfert du mot pointée (A2)+$5000 vers le registre D1.



c- Adressage registre indirect avec postincrémentation

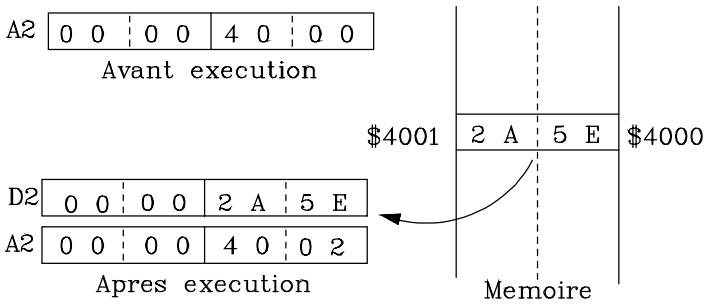
Dans ce mode d'adressage le principe est toujours le même que celui de l'adressage registre indirect. Cependant, dans ce deuxième cas le registre qui contient l'adresse et qui est spécifié par l'opérande de l'instruction doit être incrémenté après exécution de l'instruction par N.

N=1 dans le cas manipulation d'octets.

N=2 dans le cas de manipulation de mots.

N=4 dans le cas de manipulation de longs mots.

MOVE.W (A2) +, D2 Transfert du mot pointée par le contenu de A2 vers le registre D2 puis A2=A2+2.



d- Adressage registre indirect avec prédécrémentation

Avant de chercher l'opérande pointé par le contenu d'un registre d'adresse, le contenu du registre d'adresse utilisé doit être décrémenté par un nombre N.

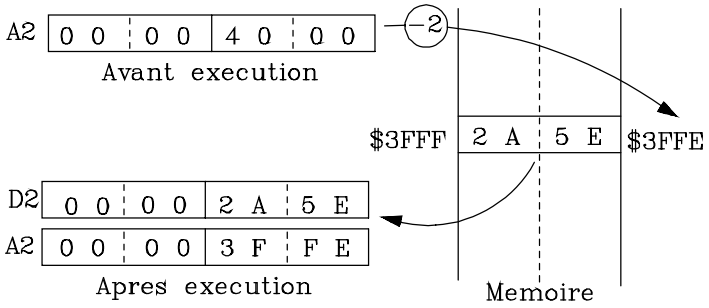
N=1 dans le cas manipulation d'octets.

N=2 dans le cas de manipulation de mots.

N=4 dans le cas de manipulation de longs mots.

Exemple:

MOVE.W - (A2), D2 Transfert du mot pointée par [(A2)-2)] vers le registre D2.



e- Adressage registre indirect avec déplacement et index court

L'instruction dans ce mode d'adressage spécifie comme dans les cas précédents l'adresse de l'opérande, mais cette adresse est calculée à partir du :

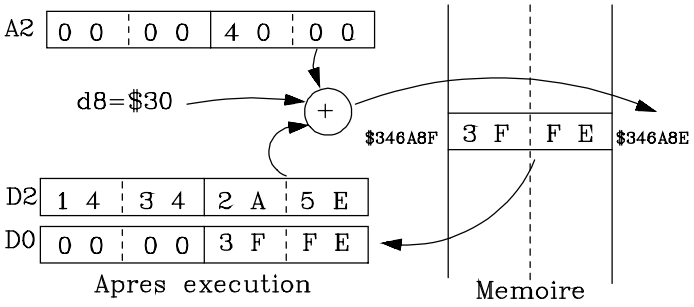
Contenu d'un registre d'adresse Ai.

Mot contenu du registre index Xi qui peut être registre de donnée ou d'adresse.

Déplacement donné par un nombre signé de 8 bits.

Exemple:

MOVE.W $30(A2,D2.W), D0 Transfert du mot adressé par (A2) +D2+$30 vers D0.



f- Adressage registre indirect avec déplacement et index long

Cette adresse est calculée à partir du :

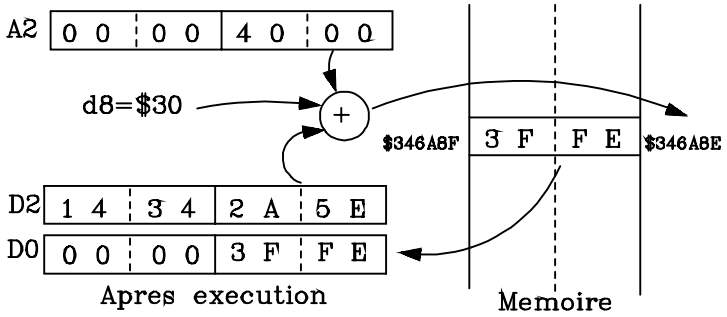
Contenu d'un registre d'adresse Ai.

Long mot contenu du registre index Xi

Déplacement donné par un nombre signé de 8 bits.

Exemple:

MOVE.W $30(A2,D2.L), D0 Le long mot contenu dans le registre index D2 est additionné avec le déplacement signés $30 le avec le contenu de A2 pour obtenir l'adresse du mot qu'il faut transférer vers D.



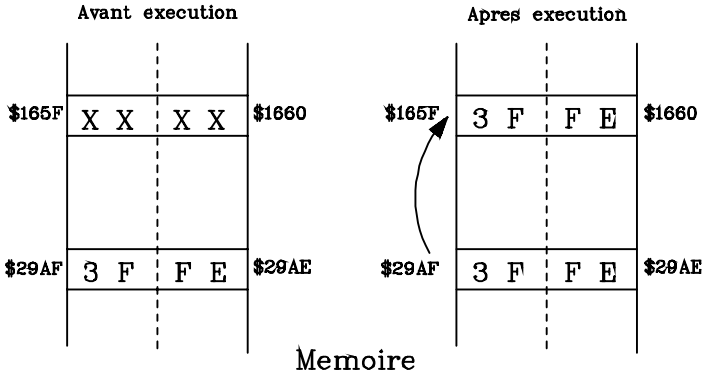
V-3-3- Adressage absolu

Dans ce mode d'adressage l'opérande spécifie l'adresse de la donnée. On rencontre dans ce mode d'adressage deux types:

- Adressage absolu court où l'adresse est donnée par un mot

Exemple:

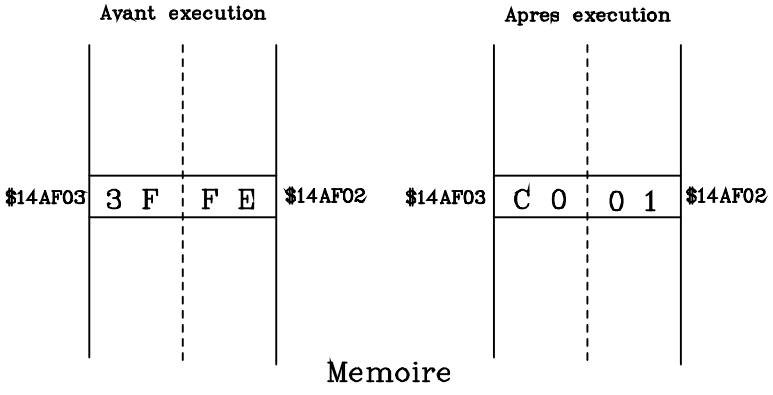
MOVE.W $28AE, $1660 transfert de donnée entre deux cases mémoire.



- Adressage absolu long où l'adresse est donnée sur 24 bits.

Exemple:

NOT.W $14AF02 Négation du contenu de la mémoire spécifie par l'adresse $14AF02



L'importance de ce mode d'adressage c'est qu'il permet de faire un transfert de mémoire en mémoire sans passer par des registres intermédiaires.

V-3-4- Adressage immédiat

Dans ce mode d'adressage l'opérande spécifie la donnée elle-même qui peut être de taille octet, mot ou long mot.

Exemple:

ADD.B #$20, D0 ajouter au contenu de D0 la donnée de taille octet $20

0 1 2 5 1 A B 4	→D0	0 1 2 5 1 A D 4	→D0
Avant exécution		après exécution	

V-3-5-Adressage relatif au compteur programme

Le principe utilisé dans ce mode d'adressage est basé sur le fait que l'adresse effective est calculée à partir du contenu du compteur programme PC. Suivant le calcul effectué on peut regroupé ce mode en deux types:

Compteur de programme avec déplacement où AE= (PC) + d16.

Compteur programme avec index où AE= (PC) + (Xi) + d8.

V-3-6-Adressage implicite

Dans ce mode d'adressage l'instruction fait implicitement référence à certains registres tels que PC, SSP, USP et SR.