Cours Préparé par M. EL ALAMI HASSAN

PRESENTATION DU COURS :

Objectifs du cours
L'objectif de ce cours est de vous rendre capable de choisir, de programmer, d'utiliser un micro contrôleur et plus généralement de vous transmettre une culture des systèmes micro-programmés.

L'informatique industrielle

> *« L'informatique industrielle est une branche de l'informatique appliquée qui couvre l'ensemble des techniques de conception et de programmation, de systèmes informatisés à vocation industrielle, qui ne sont pas des ordinateurs. »*
> *(Source : Wikipédia)*

Domaines d'applications :

Alarme, automobile, aviation, instrumentation, médicale, téléphonie mobile, terminaux de paiement pour carte bancaire ...

A. LES CALCULATEURS NUMERIQUES

1° Généralités

Un calculateur numérique est un système ou un ensemble de blocs composés d'éléments et de circuits numériques qui se concertent pour exécuter un programme bien défini. Ce dernier est composé d'instructions codées et conservées dans une mémoire avec des données sur lesquelles ce programme travaille.

Quand le calculateur reçoit l'ordre d'exécution d'un programme quelconque, il suit et respecte un certain ordre jusqu'à ce que le programme prenne fin.

D'autre part, nous pouvons dire qu'un calculateur numérique n'est rien d'autre qu'une machine ultra rapide qui traite les données, résout des problèmes, prend des décisions, tout cela sous la gouverne d'un programme.

Un calculateur numérique peut être un Automate Programmable industriel, un micro ordinateur ou tout autre système à base d'un ou de plusieurs microprocesseurs.

2° Modèle d'architectures d'une machine

2.1.1 Architecture en couches

La figure 1.1 représente une hiérarchie qui caractérise l'architecture d'une machine standard. Au fur et à mesure que l'on monte dans la hiérarchie, l'abstraction est plus importante (à un niveau donné la machine utilise les services de niveaux inférieurs). Deux parties peuvent êtres distinguées :la partie système et la partie applications. La décomposition est la suivante :

_ *couche application*

Langages d'application : ces langages sont le plus souvent l'affaire de spécialistes. Ce sont des langages spécialisés de très haut niveau, très synthétiques et abstraits. Ils ont été conçus pour rendre l'utilisation des machines plus simple.

Langages évolués : ce sont des langages universels très utilisés par les scientifiques. D'innombrables langages ont été conçus mais les plus populaires sont Pascal, C, Fortran,Ada. Proches de l'algorithme, ils permettent de coder assez aisément et rapidement les problèmes scientifiques.

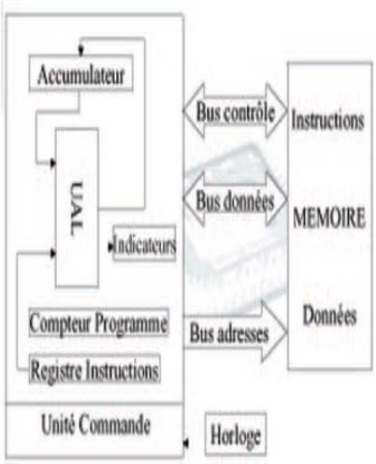
Langage d'assemblage : c'est l'expression symbolique du langage propre à la machine. Ce langage est constitué de mnémoniques qui décrivent le jeu d'instruction du processeur. Il est en effet très difficile (voire impossible) de programmer la machine en binaire directement, c'est pourquoi l'utilisateur qui veut travailler en utilisant les instructions du processeur, programme en langage d'assemblage. Un compilateur appelé assembleur permet de traduire ce langage symbolique en langage binaire (niveau macro machine).

_ *couche système*

Système d'exploitation : un ordinateur nu est inexploitable. Le rôle du système d'exploitation est d'offrir à l'utilisateur un ensemble de services permettant d'utiliser les ressources de la machine (mémoire, clavier ...). Le langage associé est composé de commandes que peut exécuter ce système (Unix, Linux, WindowsNT, WindowsXP ...).

Macro machine ou couche ISA : il s'agit essentiellement du processeur de la machine (Pentium,Athlon, PowerPC ...). Le langage associé est composé par le jeu d'instruction du processeur (langage machine du processeur ou langage binaire).

Micromachine ou micro architecture : l'exécution d'une instruction du langage machine est une opération qui nécessite également une opération de traduction en une séquence de micro opérations. Ce niveau est invisible par l'utilisateur mais constitue un niveau important pour le concepteur du processeur. Le langage associé est



composé de micro instructions que peut exécuter le processeur.

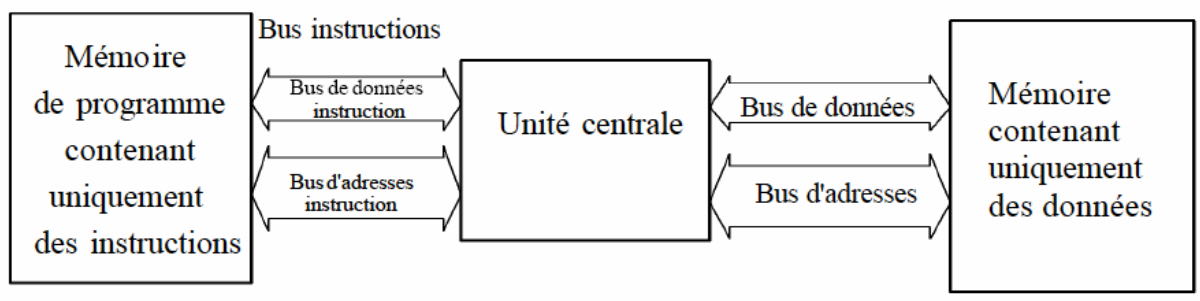
Machine physique ou couche logique : a ce niveau on décrit les composants électroniques qui composent l'ordinateur. Le langage associé est constitué des signaux logiques 0 et 1.

(voir figure)

2.1.2 Architecture de Von Neumann

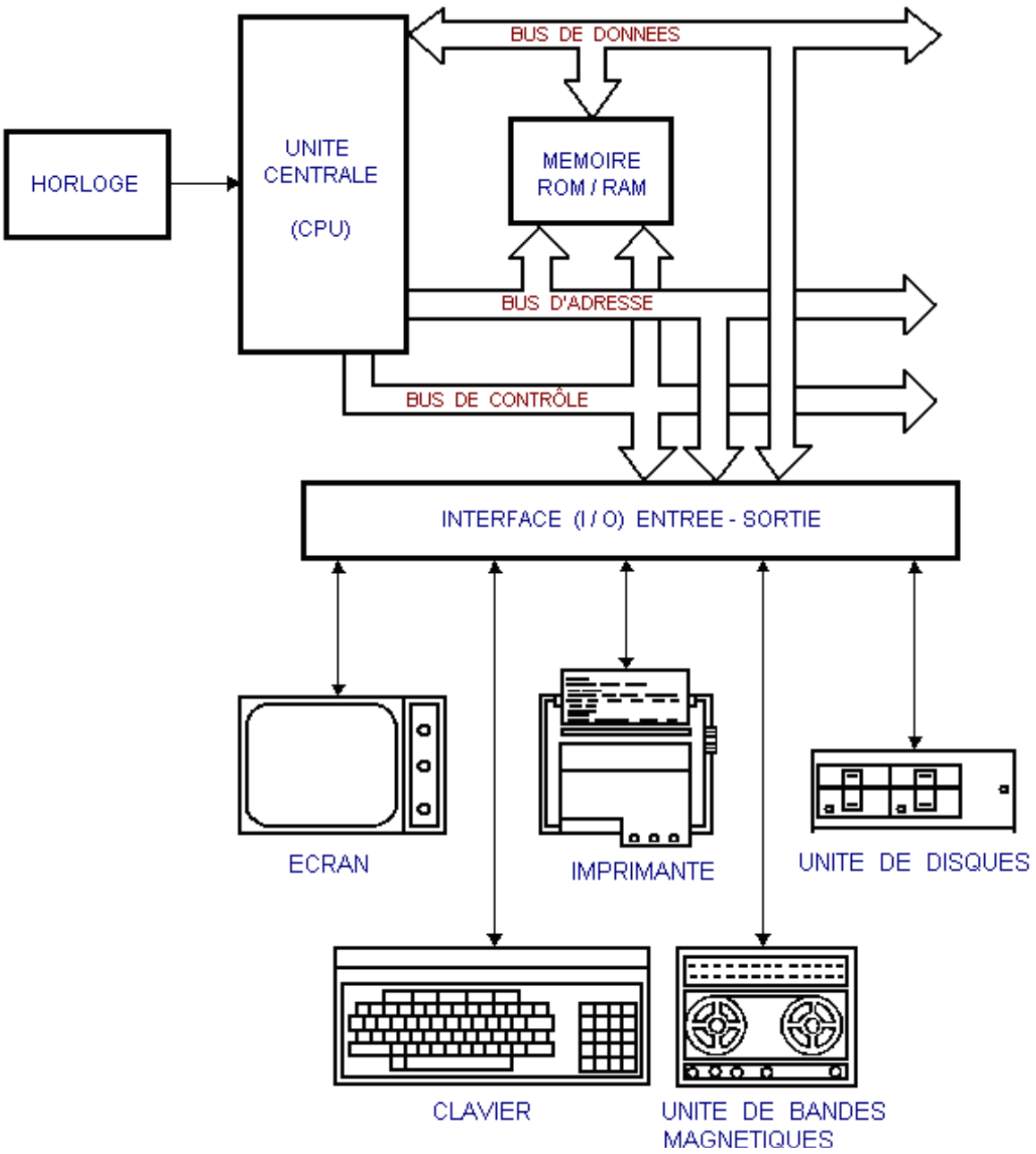
La proposition de Von Neumann est encore aujourd'hui à la base de l'architecture de presque tous les ordinateurs. La machine de Von Neumann est composée de cinq parties : la mémoire, l'unité arithmétique et logique, l'unité de contrôle, et les dispositifs d'entrées et de sorties. La figure 1.2 donne le schéma simplifié de cette machine.

2.1.3 Structure de Harvard



La différence se situe au niveau de la séparation ou non des mémoires programmes et données. La structure de Harvard permet de transférer données et instruction simultanément, ce qui permet un gain de performances.

3° Eléments de base d'un calculateur numérique

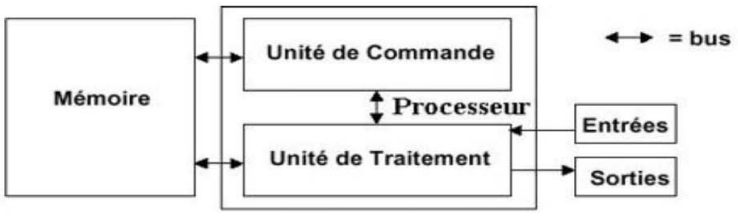


Un ordinateur est constitué de processeurs, de mémoires et de dispositifs d'entrée/sortie judicieusement interconnectés. Du point de vue utilisateur, un ordinateur est généralement composé des éléments suivants :

_ d'une unité centrale (le boîtier),
_ d'un moniteur (l'écran),
_ de périphériques externes (souris, imprimantes, scanner, clavier),
_ d'interfaces d'entrée-sortie (cartes USB, SCSI, cartes d'extension diverses),
_ de périphériques internes (cartes sons, vidéo, *: : :*),
_ d'un disque dur, d'un lecteur de disquettes, d'un lecteur de CD-ROM, DVD …

La carte mère (voir figure 1.3) est l'élément principal de du boitier de l'ordinateur car c'est sur elle que sont connectés les éléments constitutifs de l'ordinateur. Celle-ci est composée principalement :

_ d'un processeur (cerveau de l'ordinateur),
_ d'un ou de plusieurs chipset(s) : à coté du processeur se trouve un jeu de composant soudé sur la carte mère, appelé chipset. Son rôle est capital, il prend en charge le contrôle de la circulation des informations : l'horloge du système, la mémoire vive, l'accès direct mémoire (Direct Memory Access), le clavier, la souris et les connecteurs d'extension. Il régit tous les échanges au sein du PC en aiguillant les données sur les différents bus de la carte mère : le bus mémoire, PCI, AGP etc.
_ de la mémoire : BIOS ROM, RAM et mémoire cache,
_ d'un contrôleur de disques permettant le contrôle du(des) disque(s) dur(s), du lecteur CDROM et du lecteur de disquettes,
_ des périphériques internes connectés sur les bus ISA, PCI ou AGP *: : :*
_ des circuits gérant les Entrées/Sorties,
_ d'une horloge et d'une pile pour le CMOS.



3.1 Le microprocesseur(CPU)

Un microprocesseur est un circuit intégré complexe caractérisé par une très grande intégration et doté des facultés d'interprétation et d'exécution des instructions d'un programme. Il est chargé d'organiser les tâches précisées par le programme et d'assurer leur traitement. C'est le 'cerveau' du système. Il est parfois appelé CPU (Central Processing Unit) et est caractérisé par sa fréquence d'horloge (en MHz), sa largeur des bus de données et d'adresse, sa mémoire adressable, le nombre de transistors et la taille de la gravure (en microns). Ses performances sont caractérisées par le CPI (Cycle Par Instruction) et le MIPS (Millions d'Instructions Par Secondes) qui dépendent directement de la fréquence d'Horloge et son jeu d'instructions.

Un microprocesseur est construit autour de deux éléments principaux :

1. une unité de commande
2. une unité de traitement

Ces éléments sont associés à des registres chargés de stocker les différentes informations à traiter. L'ensemble sont reliés entre eux par des bus internes (de données et d'adresses) permettant l'échange d'informations.

Remarque (jeu d'instructions et langage). La première étape de conception d'un microprocesseur consiste à définir son jeu d'instructions élémentaires, notamment le type d'instructions, leur codage, les modes d'adressage et le temps d'exécution relatif aux diverses instructions. Le seul langage compris par le microprocesseur est le langage machine. Pour faciliter la tâche du programmeur, on utilisera le langage assembleur (propre à chaque microprocesseur) composé de mnémoniques (opérations de transfert, arithmétiques et logiques). Pour éviter la difficulté de mise en œuvre de l'assembleur, on utilise un langage de haut niveau (C, Pascal, Java, etc.) plus adapté aux applications. Pour que ce dernier soit compris par le microprocesseur, il faut le compiler (compilateur : traduction en assembleur) puis l'assembler (assembleur : conversion en code machine).

3.1.1. Unité de commande

C'est la partie commande du microprocesseur. Elle permet de séquencer le déroulement des instructions. Elle effectue la recherche en mémoire de l'instruction. Comme chaque instruction est codée en binaire, l'unité de commande assure le décodage pour réaliser l'exécution puis préparer l'instruction suivante. Elle est composé par :

-le compteur de programme (PC, Program Counter) : registre qui contient l'adresse de l'instruction à exécuter;

-le registre et le décodeur d'instruction (IR, Instruction Register) : stockage et décodage des instructions à exécuter;

-le séquenceur : bloc logique de commande qui organise l'exécution des instructions au rythme d'une horloge. Il élabore tous les signaux de synchronisation internes ou externes (bus de commande).

3.1.2 Unité de traitement

C'est la partie opérative du microprocesseur. Il s'agit du cœur du microprocesseur. Cette unité regroupe les circuits qui assurent les traitements nécessaires à l'exécution des instructions. Elle est composée de :

-Une Unité Arithmétique et Logique (ALU : Arithmetic and Logical Unit) : ce circuit assure les fonctions logiques (comparaison, décalage, ET, OU, etc.) et arithmétiques (addition,soustraction, etc.) .

-Un registre d'état constitué (généralement) de 8 bits dont chacun représente l'état de la dernière opération effectuée par l'ALU : Les principaux indicateurs d'état (ou drapeaux flag) sont la retenue (Carry), la retenue auxiliaire (AC), le signe (S), le débordement (overfow), le zéro (Z) et la parité (P) .

-Des accumulateurs qui sont des registres servant à stocker une opérande au début et le résultat à la fin d'une opération.

3.1.4 Cycle d'exécution d'une instruction

Un microprocesseur ne comprend qu'un nombre limité d'instructions (codées en binaire). Le traitement d'une instruction peut alors être décomposé en 3 phases principales :

Phase 1 : recherche de l'instruction à traiter

1. L'unité de commande émet un ordre de lecture et place la valeur contenue dans le compteur de programme (PC) (adresse de l'instruction) sur le bus d'adresses.
2. Après un certain temps (d'accès à la mémoire), le contenu de la case mémoire sélectionnée est disponible sur le bus de données.
3. L'instruction est stockée dans le registre IR.

Phase 2 : décodage de l'instruction

1. L'unité de commande transforme l'instruction (codée sous forme de mots binaires) en une suite de commandes élémentaires.
2. L'unité de commande récupère (si besoin) sur le bus de données la valeur d'une donnée en provenance de la mémoire.
3. L'opérande est stocké dans un registre de base.

Phase 3 : exécution de l'instruction

1. Le micro programme réalisant l'instruction est exécuté.
2. Les drapeaux sont positionnés.
3. L'unité de commande positionne le compteur de programme (PC) pour l'instruction suivante.

3.1.5 Familles d'architecture des microprocesseurs

Il existe actuellement 2 grandes familles :

-Architecture **CISC** (Complex Instruction Set Computer) : elle est historiquement la plus ancienne. Elle consiste à privilégier des instructions plus complexes (elle nécessite un décodeur complexe, le micro code) du fait de la lenteur de la mémoire (comparativement au microprocesseur).

C'est donc une architecture qui comporte un grand nombre d'instructions. Pour une tâche donnée, on exécute un petit nombre d'instructions qui nécessitent chacune un grand nombre de cycles.

-Architecture **RISC** (Reduced Instruction Set Computer) : elle découle d'une étude statistique qui a prouvé que 80 % des traitements des langages de haut niveau faisaient appel à seulement 20 % des instructions microprocesseur, d'où l'idée de réduire ce jeu d'instruction. Étant donné cette réduction, la possibilité d'une réalisation à base de séquenceur câblé libère de la surface et permet ainsi d'augmenter le nombre de registres ou d'unités de traitement (chaque instruction s'exécute en un cycle d'horloge). Toutefois, cette architecture impose l'emploi d'un compilateur très évolué. Le choix entre ces 2 types d'architectures se fait en fonction des applications visées et en comparant le nombre d'instructions et de cycles nécessaires à ces applications.

3.2 Mémoire Principale

3.2.1 généralités

Définition (mémoire). Une mémoire est un circuit à semi-conducteur permettant d'enregistrer, de conserver et de restituer des informations (appelées données).
C'est cette capacité de mémorisation qui explique la polyvalence des systèmes numériques et leur adaptabilité à de nombreuses situations. Les données peuvent être lues ou écrites. Il y a écriture lorsqu'on enregistre des informations en mémoire et lecture lorsqu'on récupère des informations précédemment enregistrées.

3.2.2 Organisation et caractéristiques

Une mémoire peut être décrite comme une armoire de rangement dont chaque tiroir constitue une case mémoire qui peut contenir un seul élément (généralement un octet ou bien un mot de longueur une puissance de 2, définit par la largeur du bus de données). Le nombre de case est alors élevé et il est nécessaire de pouvoir les repérer par un numéro appelé adresse.
Avec une adresse codée sur 'n' bits, il est possible de référencer 2^*n* cases mémoire (définit par la largeur du bus d'adresses). En plus de ces 2 bus, un boîtier mémoire comporte une entrée de commande (pour sélectionner la lecture ou l'écriture) et une entrée de sélection (pour la mise en haute impédance des entrées/sorties du boîtier).
Une opération de lecture (resp. écriture) suit le cycle d'instructions suivant :
1. sélection de l'adresse,
2. choix de l'opération à effectuer ($R=W$),
3. sélection de la mémoire ($CS = 0$),
4. lecture (resp. écriture) de la donnée.

Les circuits de mémoire ne sont pas tous identiques. La différence se fait par leurs caractéristiques dont les principales sont :
La capacité : c'est le nombre total de bits que contient la mémoire (exprimé en Ko ou Mo),
Le format des données : largeur (en bits ou octets) du mot mémorisable,
Le temps d'accès : temps qui s'écoule entre l'instant où l'opération ($R=W$) est lancée et l'instant où la donnée est effectivement disponible sur le bus de données. On parle alors d'accès direct et de temps d'accès constant (indépendant de l'adresse).
Le temps de cycle : intervalle minimal qui sépare 2 demandes successives de lecture (ou écriture),
Le débit : nombre maximum d'informations lues (ou écrites) par secondes,
La volatilité : elle caractérise la permanence des informations dans la mémoire (donnée volatile si elle disparaît lors d'absence d'alimentation, non volatile dans le cas contraire).
Les critères de choix sont (principalement) la capacité, la vitesse, la consommation et le coût.

2.2.3 Mémoire vive : RAM

Définition 1.2 (RAM). Une mémoire vive (RAM : Random Access Memory) sert au stockage temporaire des données. Elle doit avoir un temps de cycle très court pour ne pas ralentir le microprocesseur.
Ces mémoires sont en général volatiles et sont divisées en 2 grandes familles : statiques et dynamiques. La mémoire statique (SRAM) est un composant dont chaque bit est constitué d'une bascule (contenant entre 4 et 6 transistors). Dans la mémoire dynamique (DRAM), l'information est mémorisée sous forme d'une charge électrique stockée dans un condensateur (transistor MOS), augmentant ainsi la capacité d'intégration et réduisant la consommation. Toutefois, l'inconvénient majeur réside dans l'obligation, pour les DRAM , d'un rafraîchissement périodique (pour compenser les courants de fuite du condensateur), ce qui impose une gestion plus complexe et un temps d'accès aux informations plus long.
En général les DRAM sont utilisées pour la mémoire centrale alors que les SRAM sont plus adaptées aux mémoires cache et aux registres.

1.2.2.4 Mémoire morte : ROM

Définition 1.3 (ROM). Pour certaines applications, il est nécessaire de pouvoir conserver des informations de façon permanente même lorsque l'alimentation est interrompue. On utilise alors des mémoires non volatiles appelées mémoires mortes (ROM : Read Only Memory). La seule opération possible est une opération de lecture.
L'opération d'écriture (qui est possible) est appelée programmation. La méthode peut varier selon le type de ROM :
ROM : elle est programmée par le constructeur, son contenu est non modifiable. Sa densité d'intégration est élevée et elle est très rapide.
PROM : c'est une ROM programmable une seule fois par l'utilisateur et rapidement dont le coût est relativement faible.
EPROM : c'est une ROM reprogrammable (mais entièrement à chaque reprogrammation) et souvent (un millier de fois). L'écriture est toutefois plus lente (environ 1000 *X*) que sur une RAM.
EEPROM : c'est une EPROM améliorée car elle est effaçable électriquement mot par mot. L'utilisation en RAM est très lente et son coût de réalisation est relativement élevé.
FLASH EPROM : communément appelée mémoire flash (clé USB, lecteur MP3, PDA, appareil photo numérique), c'est une EEPROM particulière basée sur 2 technologies (NOR : cellules en parallèle, bus d'adresse et de données dédiés ou NAND : cellules en série, interface d'E/S indirecte, implantation plus dense).

Remarque 1.2 (Mémoire cache ou antémémoire). Malgré des progrès technologiques identiques pour les mémoires et les microprocesseurs, le décodage des adresses et la lecture/écriture d'une donnée restent des phénomènes incompressibles. Pour éviter les temps de latence qui en découle, on dispose d'une mémoire (appelée mémoire cache ou antémémoire) très rapide entre le microprocesseur et la mémoire. Le principe repose sur le fait que la mémoire cache conserve les mots mémoires les plus fréquemment employés (plutôt que de les stocker dans la mémoire principale).
Cette mémoire cache est séparée en plusieurs niveaux (cache interne : L1 et cache externe : L2 et L3) et le cache interne (L1) est séparé en 2 (l'un pour les instructions, l'autre pour les données). Il faut noter que la somme des espaces mémoires nécessaires dépasse presque toujours la taille physique de la mémoire. Pour pallier cet inconvénient, on utilise de la mémoire virtuelle avec les notions associées de pagination, segmentation et TLB (tampon de traduction).

3.4 Mémoires Secondaires

Les besoins en capacité des mémoires augmentent considérablement (avec la taille des fichiers par exemple). On utilise alors des mémoires secondaires souvent appelées mémoire de masse en rapport avec la taille élevée de leur capacité. il s'agit principalement des disques.

3.4.1 Disques magnétiques

Un disque magnétique est constitué de plusieurs plateaux circulaires empilés et de plusieurs têtes de lecture/écriture. On distingue 2 catégories de disques magnétiques :
disque dur : plateau métallique, tête de lecture électromagnétique flottante sur coussin d'air (disque Winchester),
disque souple, **disquette** : plateau souple, tête en contact avec la surface.
Il existe deux différents standards de disque dur : IDE et SCSI qui utilisent les bus et contrôleurs d'interface du même nom. On peut aussi rajouter les disques RAID dont la technologie diffère et fait appel au parallélisme.

D'après le site web : http://fr.wikipedia.org/wiki/Microprocesseur, le premier micro processeur a été inventé, en 1971, par l'ingénieur d'Intel Mr Marcian Hoff (surnommé Ted Hoff). Et le premier microprocesseur qui a été commercialisé, le 15 novembre 1971, est l'Intel 4004 (à 4-bits). Il fut suivi par l'Intel 8008. Ce microprocesseur a servi initialement à fabriquer des contrôleurs graphiques en mode texte, mais jugé trop lent par le client qui en avait demandé la conception, il devint un processeur d'usage général. Ces processeurs sont les précurseurs des Intel 8080 Zilog Z80 et de la future famille des Intel x86. Le tableau suivant décrit les principales caractéristiques des microprocesseurs fabriqués par Intel et

montre la fulgurante évolution des microprocesseurs autant en augmentation du nombre de transistors, en miniaturisation des circuits et en augmentation de puissance.

Date	Nom	Transistors	Finesse de gravure (micro m)	Fréquence	Largeur des données	MIPS
1971	4004	2 300			4 bits / 4 bits bus	
1974	8080	6 000	6,0	2 MHz	8 bits / 8 bits bus	0,64
1979	8088	29 000	3,0	5 MHz	16 bits / 8 bits bus	0,33
1982	80286	134 000	1,5	6 MHz	16 bits / 16 bits bus	1,00
1985	80386	275 000	1,5	16 MHz	32 bits / 32 bits bus	5,00
1989	80486	1 200 000	1,0	25 MHz	32 bits / 32 bits bus	20,0
1993	Pentium	3100 000	0,8	60 MHz	32 bits / 64 bits bus	100
1997	Pentium II	7 500 000	0,35	233 MHz	32 bits / 64 bits bus	300
1999	Pentium III	9 500 000	0,25	450 MHz	32 bits / 64 bits bus	510
2000	Pentium 4C	42 000 000	0,18	1,5 GHz	32 bits / 64 bits bus	1 700
2004	Pentium 4D	125 000 000	0,09	3,6 GHz	32 bits / 64 bits bus	9 000
2006	Core 2TM Duo	291 000 000	0,065	2,4 GHz	64 bits / 64 bits bus	22 000
2007	Core 2TM Quad	2 x 291 000 000	0,065	3 GHz	64 bits / 64 bits bus	2 x 22 000
2008	Core 2TM Duo (Penryn)	410 000 000	0,045	3,16GHz	64 bits / 64 bits bus	Env. 24 200

Date *: l'année de commercialisation du microprocesseur.*
Nom *: le nom du microprocesseur.*
Transistors *: le nombre de transistors contenus dans le microprocesseur.*
Finesse de gravure *: le diamètre (en micromètres) du plus petit fil reliant deux composantes du microprocesseur. En*
comparaison, l'épaisseur d'un cheveu humain est de 100 microns !
Fréquence de l'horloge *: la fréquence de l'horloge de la carte mère qui cadence le microprocesseur. MHz = millions*
de cycles par seconde. GHz = milliards de cycles par seconde.
Largeur des données *: le premier nombre indique le nombre de bits sur lequel une opération est faite. Le second*
nombre indique le nombre de bits transférés à la fois entre la mémoire et le microprocesseur.
MIPS *: le nombre de millions d'instructions complétées par le microprocesseur en une seconde.*

B)Etude du microprocesseur 6809

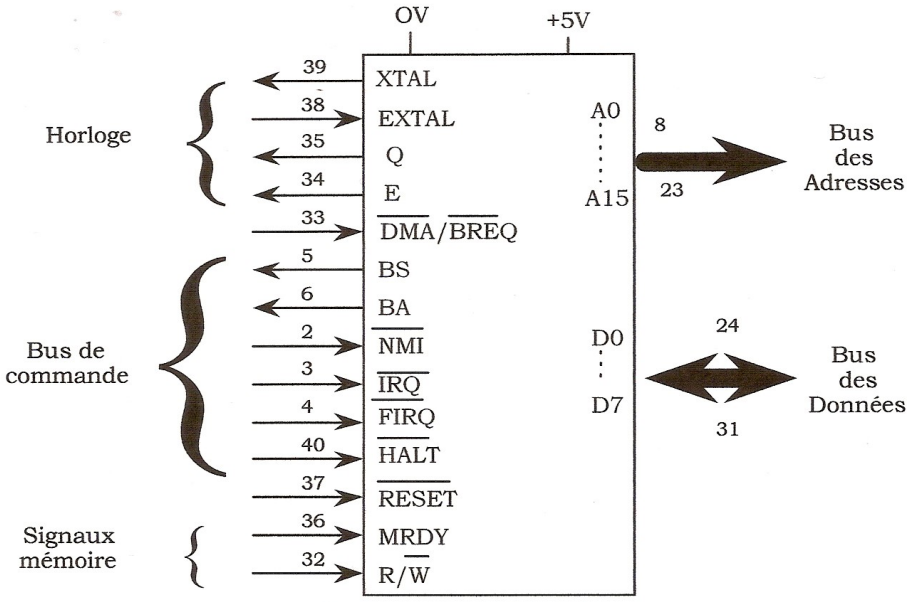
Le microprocesseur 6809 est un processeur 8 bits dont l'organisation interne est orientée 16 bits. Il est fabriqué en technologie MOS canal N et se présente sous la forme d'un boîtier DIL 40 broches. Il est mono tension(5V).

Il existe deux versions différenciées par l'horloge.

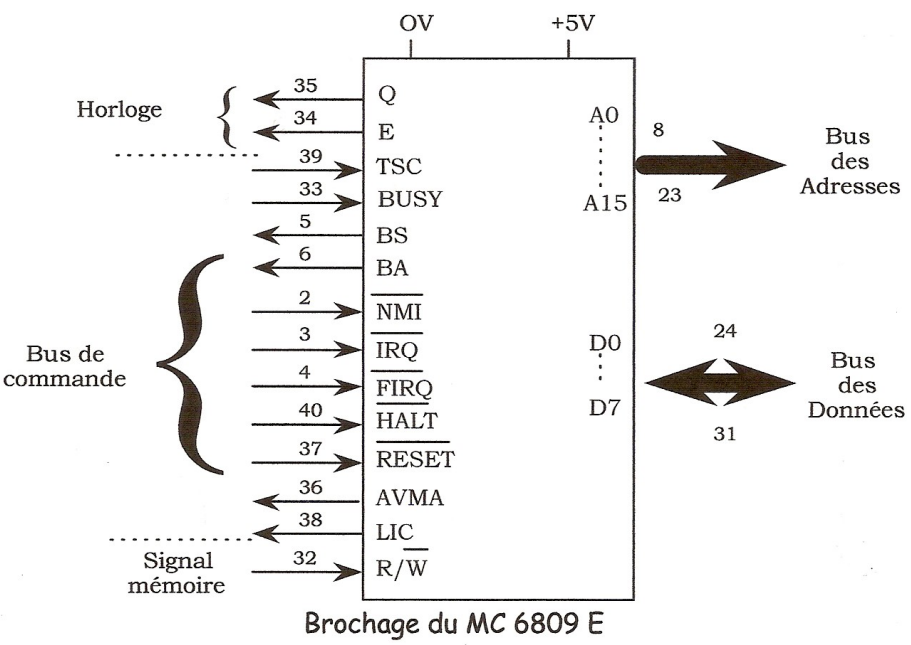
le 6809 est rythmé par une horloge interne (f=1 MHZ, 1.5 MHZ et 2 MHZ).

le 6809E est rythmé par une horloge externe.

Ce dernier est adapté aux applications multiprocesseur. Il présente la particularité de pouvoir être synchronisé par une horloge extérieure. Compatibilité complète entre les 2 versions.



Brochage du MC 6809



Brochage du MC 6809 E

Présentation du brochage

l'alimentation (Vss - Vcc)

Le bus des données 8 bits (Do à D7)

Ces huit broches sont bidirectionnelles. Elles permettent la communication avec le bus des données interne du microprocesseur. Chaque broche peut "piloter" 1 charge TTL et 8 entrées de circuits appartenant à la famille 6800. Bus en logique 3 états.

Le bus des adresses 16 bits (A0 à A15)

Ces broches unidirectionnelles transfèrent l'adresse 16 bits fournie par le microprocesseur au bus d'adresse du système. Mêmes caractéristiques électriques que pour le bus des données. Bus en logique 3 états.

⚠les adresses sont validées sur le front montant de Q.

Le bus de contrôle

La broche **Read/**Write

Cette broche indique le sens de transfert des données sur le bus des données. Ligne à logique 3 états

R/ W = 1 lecture en cours (D0 - D7 sont des entrées)

R/ W = 0 écriture en cours (D0 - D7 sont des sorties)

⚠cette ligne est validée sur le front montant de Q.

Les lignes d'état du bus

BA (Bus available) et **BS** (Bus state) Information qui permet de connaître l'état du microprocesseur à tout moment.

BA BS Etat

0 0 normal

0 1 reconnaissance d'interruption

1 0 reconnaissance de synchronisation externe

1 1 arrêt bus disponible

1er **cas** : Le microprocesseur est en fonctionnement normal, il gère les bus d'adresses et de données.

2ème **cas** : le microprocesseur est en phase de reconnaissance d'interruption pendant deux cycles. Cet état correspond à la recherche des vecteurs d'interruption : Reset, NMI, IRQ, SW1,2 et 3. (vecteur : adresse mémoire du programme)

3ème **cas** : Ce signal apparaît lorsque le microprocesseur rencontre l'instruction de synchronisation externe (niveau bas sur SYNC). Il attend alors cette synchronisation sur une des lignes d'interruption. Les bus sont en haute impédance pendant ce temps.

Dernier cas : Correspond à l'arrêt du microprocesseur (niveau bas sur HALT). Le microprocesseur laisse la gestion des bus des données et des adresses à un circuit annexe (contrôleur de DMA). Les bus sont en haute impédance. La ligne BA au niveau haut indique que les bus sont en haute impédance.

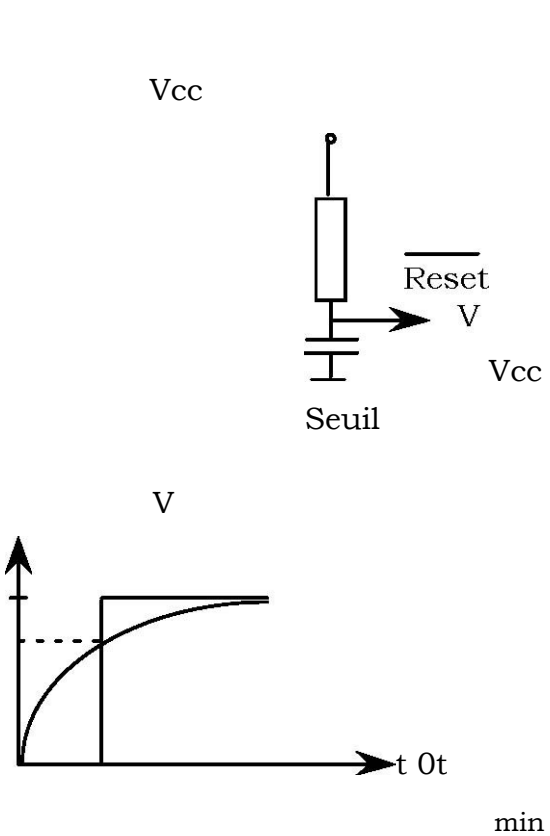
Broche d'initialisation **RESET**

Un niveau bas sur cette broche entraîne une réinitialisation complète du circuit. Conséquences :

l'instruction en cours est arrêté le registre de pagination (DP) est mis à zéro les interruptions IRQ et FIRQ sont masquées l'interrruption non masquable NMI est désarmée

Pour être active cette ligne doit être maintenue à un niveau bas durant un temps suffisamment long (plusieurs cycles d'horloge).

schéma adopté généralement



Le P.C. est initialisé avec le contenu des vecteurs d'initialisation qui se trouvent aux adresses $FFFE et $FFFF.

Ce contenu représente l'adresse du début du programme qui sera exécuté par le microprocesseur.

la broche : **HALT** (Arrêt du microprocesseur).

Un niveau bas sur cette broche provoque l'arrêt du microprocesseur (mais à la fin de l'exécution de l'instruction en cours). Il n'y a pas perte des données. (BA = BS = 1)

Dans ce cas : les demandes d'interruption IRQ et FIRQ sont inhibées les demandes d'accès direct (DMA) à la mémoire sont autorisée. les demandes d'interruptions RESET et NMI sont prises en compte
mais leur traitement est différé.

les broches d'interruption **NMI** (No Masquable Interrupt) **IRQ** (Interrupt Request) **FIRQ** (Fast Interrupt Request)

Entrées (actives sur un niveau bas) qui peuvent interrompre le fonctionnement normal du microprocesseur sur front descendant de Q.

Entrées d'horloge **XTAL et EXTAL** (Extension crystal)

La fréquence du quartz (horloge) est quatre fois la fréquence du microprocesseur.

Eout représente le signal d'horloge commun au système. Il permet la synchronisation du microprocesseur avec la périphérie.

Qout représente le signal d'horloge en quadrature avec Eout.

Les données sont lues ou écrites sur le front descendant de Eout.
Les adresses sont correctes à partir du front montant de Qout.

Broches complémentaires du bus de contrôle :

MRDY (Memory ready) :

Cette broche de commande permet d'allonger la durée de Eout pour utiliser des mémoires à temps d'accès long. Active sur un niveau bas. l'allongement est un multiple de un quart de cycle et sa valeur maximale est de 10 cycles.

DMA/BREQ (Direct Memory Acces/Bus Request).

Cette broche permet de suspendre l'utilisation des bus du microprocesseur, pour faire de l'accès direct ou du rafraîchissement mémoire.

fonctionnement : Pendant que Q est au niveau haut (si DMA/BREQ bas) cela entraîne l'arrêt du microprocesseur à la fin du cycle en cours ... et non de l'instruction.

(BA = BS = 1 ce qui veut dire que tous les bus sont en haute impédance). le circuit ayant généré cette commande dispose de 15 cycles machines avant que le microprocesseur ne reprenne le contrôle des bus.

Broches spécifiques au 6809 E

Entrées d'horloge : **EIN** et **QIN**

Ce sont deux broches dans lesquelles on applique des signaux identiques à Eout et Qout du 6809. Ces signaux doivent aussi être fournis à l'ensemble du système (signaux de synchronisation).

TSC (Tree States Control).

Cette broche a le même rôle que l'entrée DMA/BREQ précédente. Possibilité de faire du DMA afin de réaliser des opérations de :

rafraichissement partage de bus avec un autre microprocesseur

LIC (Last Instruction Cycle).

Cette broche de sortie est à l'état haut pendant le dernier cycle de chacune des instructions exécutées par le microprocesseur. Le cycle qui suit ce signal est donc toujours un cycle de recherche de code opératoire d'une instruction.

AVMA (Advanced Valid Memory Access) (Contrôle des ressources communes en multiprocesseur)

C'est une broche de sortie qui signale un prochain accès au bus par le microprocesseur. Cette sortie au niveau haut signifie que le microprocesseur utilisera les bus au cours du cycle suivant. La nature prédictive de ce signal permet d'améliorer le fonctionnement d'un système multiprocesseurs à bus partagé. Elle permet un contrôle efficace des ressources communes d'un dispositif multiprocesseur.

BUSY : Occupation des bus :

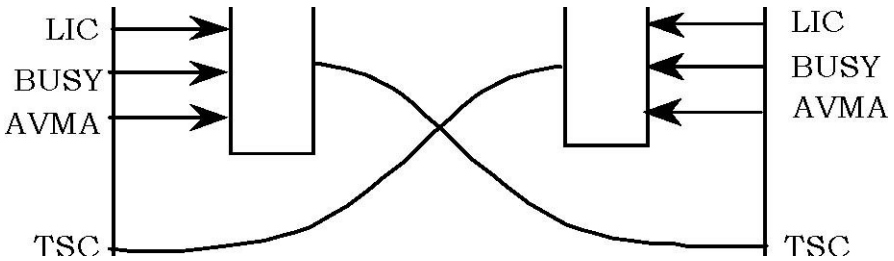
Sortie mise au niveau haut pendant les instructions du type : lecture, écriture et exécution du premier octet d'un opérande constitué de deux octets (une adresse par exemple).

Dans un système multiprocesseur, ce signal indique le besoin pour un microprocesseur de disposer des bus au cours du prochain cycle pour assurer l'intégrité de l'opération en cours.

Cela évite l'adressage simultané d'une même zone mémoire par 2 microprocesseurs.

On ne doit pas ⚠ activer TSC quand BUSY est actif.

Exemple d'application :



LIC : fin d'exécution de l'instruction BUSY : à besoin du bus au prochein cycle AVMA : va utiliser les bus au prochain cycle

D'autre part, le microprocesseur 6809 est composé de neuf registres internes que nous allons décrire ci dessous

-**Les accumulateurs A et B** sont deux registres de taille 8 bits jouant des rôles identiques, à part pour les deux instructions ABX et DAA.

-**Le double accumulateur D** est en fait la concaténation des deux registres A et B, l'accumulateur A représentant les poids forts et B les poids faibles. L'accumulateur D dispose d'instructions spécifiques : addition, soustraction, comparaison, etc. permettant de travailler directement sur 16 bits.

- Le registre DP (Direct Page) : est un registre 8 bits. Il forme la partie haute de l'adresse à pointer dans le cas d'un adressage direct. Il est automatiquement remis à zéro par un Reset.

Les registres d'index X et Y : Ils sont de taille 16 bits. D'autre part, ils permettent d'adresser tout l'espace mémoire avec en plus la capacité d'être pré-décrémenté ou post-incrémenté (ce qui facilite le traitement de variables en tables).

-**Le compteur programme (Program Counter - PC)** : est un registre de 16 bits. Il peut pointer sur toutes les cases mémoires. Il contient l'adresse de la prochaine instruction à exécuter. Pour vos applications, il faut l'initialiser à l'adresse de départ du programme à exécuter.

-Les pointeurs de pile U et S : Ce sont deux registres 16 bits qui fonctionnent tous deux identiquement. Ils opèrent en mode « dernier entré ® premier sorti ». Ces deux registres peuvent à l'occasion servir de registre d'index avec la totalité des possibilités de X et Y. Le registre S (System) est utilisé par le 6809 pour toutes les opérations de sauvegarde en cas d'interruption ou de saut à un sous-programme (adresse de retour). Le registre U (User) est entièrement réservé à l'utilisateur.

- Le registre d'état (Code Condition Register - CCR) : Ce registre définit à tout instant l'état du microprocesseur résultant d'une instruction. Il est composé de 8 bits jouant chacun un rôle important pour les instructions de sauts ou de branchements conditionnels, … :

> **« C » Carry** : Ce bit prend la valeur 1 chaque fois que le résultat d'une instruction arithmétique ou logique dépasse 8 bits, c'est à dire nous avons une retenue. On peut considérer cette retenue comme le 9ème bit pour les accumulateurs A et B.
> **N.B.**« C » reste à zéro dans les cas contraires !
>
> **« V » Overflow** : C'est un bit de débordement. Il est mis à 1 si le résultat d'un complément à 2 déborde :c'est à dire dépasse l'octet. Sinon, il reste à 0.
>
> **« Z » Zéro** C'est un bit qui indique simplement si le résultat d'une instruction ou d'une opération est nul. Dans ce cas, il est mis à 1. Pour tout autre cas, il est mis à zéro .
> **« N » Negative** : C'est un indicateur de signe. Il est positionné à 1 si le résultat d'une instruction ou d'une opération est négatif. Sinon, il reste à 0.

« I » Interrupt : C'est un indicateur d'interruption. Il est positionné, en général, par le programmeur sauf quelque fois sur initiative du microprocesseur. Il s'agit du masque pour la prise en compte des demandes d'interruption masquables : IRQ. Il sera automatiquement positionné à 1 si une demande d'interruption IRQ ou NMI.

« H » Half Carry : C'est un indicateur de demi-retenue. Lors d'une opération, le mP traite les octets bit par bit. Si en traitant le 4ème bit de l'octet, il doit faire une retenue vers le bit 4. Il a l'obligeance de nous avertir en mettant H à 1.
N.B. Cet indicateur est exploité par l'instruction « DAA » (Decimal Adjustment of Accumulator A) pour formater en code BCD des additions de nombres BCD.

« F » (FIRQ mask) Ce bit conditionne le traitement de la ligne d'interruption FIRQ. Si F = 1, les interruptions seront masquées. Il est à zéro après un Reset et dans ce cas, FIRQ est dévalidée. Si l'utilisateur le force à 1, FIRQ est traitée.
N.B. Cette indicateur est positionné également par : NMI et SWI !

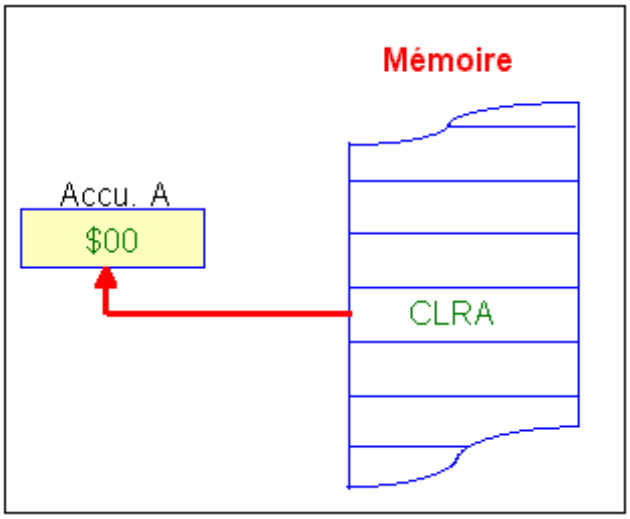
« E » (Entire flag) Ce bit nous renseigne sur le nombre de registres rangés dans la pile. Il est utilisé par l'instruction RTI pour déterminer le nombre d'octets que la pile doit restituer.

Différents modes d'adressage

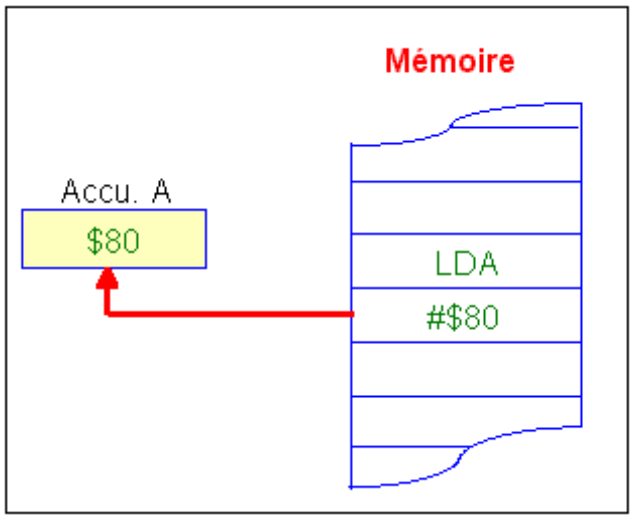
Le µP 6809 possède 59 instructions 'de base'. Cela peut paraître faible, mais combinées aux différents modes d'adressage, elles offrent 1464 possibilités. Signalons au passage qu'une instruction comporte de un à quatre octets. Généralement, **le premier** octet indique l'action à effectuer, les suivants précisent les opérandes ou sur quelques registres cette action agira. Dans ce qui suit, nous allons présenter, succinctement, quelques modes d'adressage correspondant à ce microprocesseur.

→ **Adressage inhérent ou implicite** : Le code opération contient toute l'information nécessaire à l'exécution de l'instruction. Les instructions correspondant sont codées :
- soit sur un seul octet, exemple : ABX, ASLA, RORA, NEGA, COMA, DEÇA, ...
- soit sur deux octets, exemple : SWI2, TFR, PSHS, PULU, CWAY, ...

Exemples : **CLRA**, initialisation de l'accumulateur **A** par $00
INCB, incrémentation de **1** de l'accumulateur **B.**
AAX, addition de l'accumulateur **A** au registre **X**.



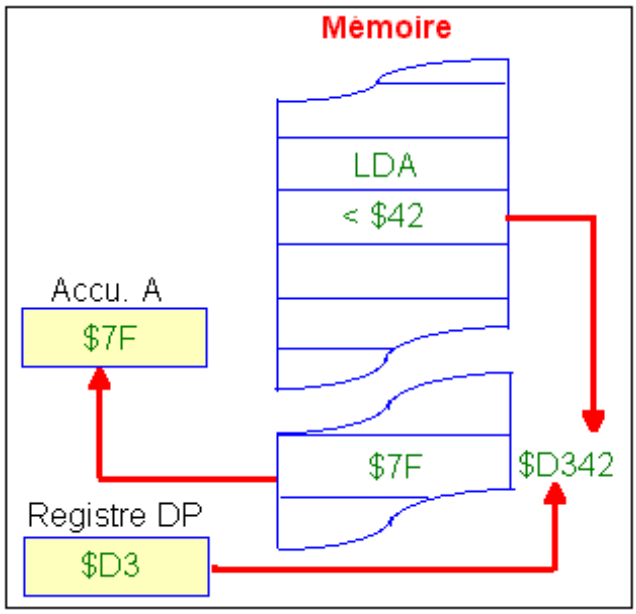
→ **Adressage immédiat** : Dans ce mode d'adressage, le code opératoire est suivi d'une valeur qui est l'opérande de l'instruction (sur un ou deux octets). Ceci permet de charger les registres internes du µP avec la valeur de l'opérande. Il existe trois types d'instructions dans ce mode d'adressage.
*Exemple : **LDA #$80,** Chargement de **A** avec la valeur hexadécimal **80***



Le symbole « **$** » précise que la donnée est en hexadécimal. Pour les instructions **LDr** (**r** : registre quelconque), la valeur à charger dans le registre doit être du même type que l'accumulateur (8 bits pour les accumulateurs **A, B** et 16 bits pour les registres **X, Y, U, S et D**). Ce type d'adressage permet d'initialiser les registres internes du microprocesseur.

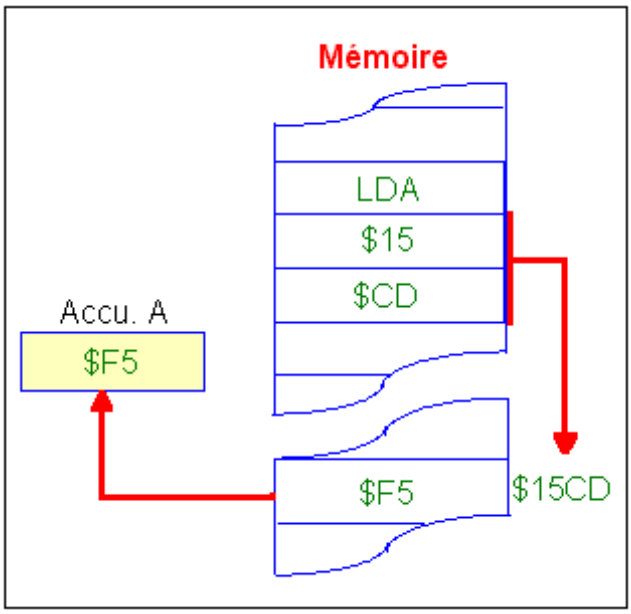
N.B : *Ces instructions peuvent atteindre quatre octets comme LDS (Chargement du pointeur de Pile par avec le contenu mémoire) ou CMPU (Comparaison mémoire avec le pointeur de Pile).*

→ **Adressage direct** : Ce mode d'adressage présente l'avantage de ne nécessiter que 2 octets pour avoir accès à des données situées sur l'ensemble de l'espace mémoire du µP. Le premier octet définit le code opératoire, le second représente les 8 bits de poids faibles de l'adresse effective dont les 8 bits de poids fort se trouvent dans le registre de page du µP (DPR). Il suffit donc d'initialiser le registre de page (DPR) pour pouvoir travailler en adressage direct sur 256 octets de la page choisie ; au delà, il faut de nouveau accéder au DPR. A la mise sous tension, le registre de page est mis à zéro. On aura donc accès aux 256 octets de la page 0. Il existe deux types d'instructions dans ce mode d'adressage. On notera que l'adressage direct sera spécifié par le signe « < » placé devant l'opérande, dans la syntaxe assembleur.



*Exemple : Si le registre de page vaut **$D3** : l'instruction **LDA <$42** chargera l'accumulateur « **A** » par le contenu de la case mémoire **$ D342** qui est $**7F**.*
NB : Ce mode présente l'avantage d'être exécuté rapidement

→ **Adressage étendu** : Dans ce cas, le champ adresse qui suit l'instruction contient l'adresse effective sur deux octets, ça permet d'atteindre toute la mémoire. Il existe deux types d'instructions dans ce mode d'adressage.
Exemple : Charger l'accumulateur A avec la valeur hexadécimal $F5 qui est le contenu de l'adresse $15CD (LDA $15CD ou LDA > $15CD).

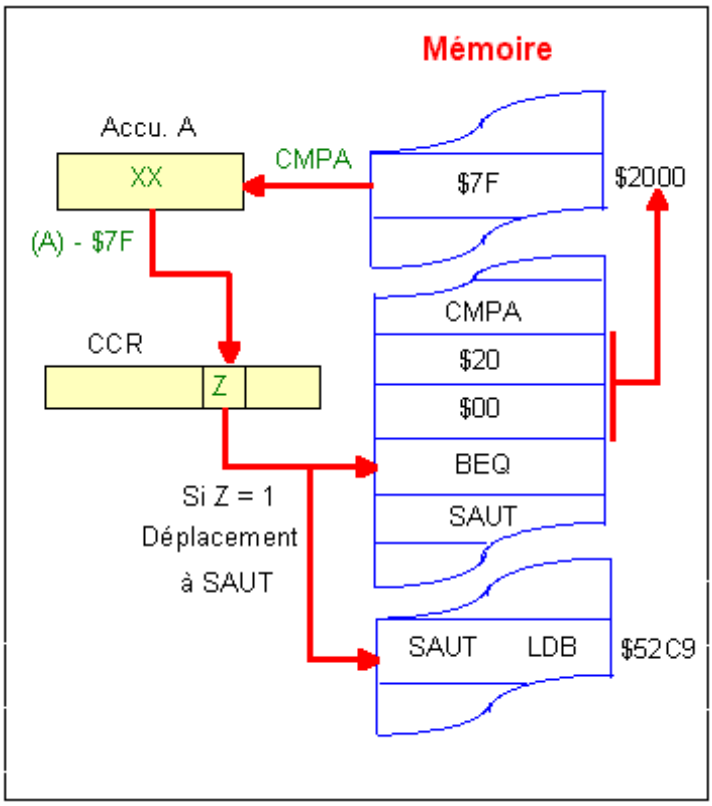


Autres exemples :
LDA $ 1000 → charge A avec le contenu de l'adresse $ 1000
SBCB $ 29D0 → soustrait à B le contenu de l'adresse $ 29D0 et le contenu du bit C de CCR
CMPX $ 4800 → compare X au contenu de $ 4800-$ 4801

→ **Adressage relatif court** : Ce mode d'adressage est réservé pour les instructions de branchement qui sont d'une longueur de 2 octets. Le premier octet détermine le code opératoire qui spécifie le type de branchement en même temps que le test correspondant. Le deuxième octet correspond au déplacement qui est soit positif ou négatif : il est stocké sous forme binaire d'un nombre en complément à 2 de 8 bits dont la valeur décimale se situe entre -128 et +127. Une instruction de branchement est toujours précédée par une opération qui peut s'assimiler à un test. Le résultat de cette opération affecte 0 ou 1 sur un ou plusieurs bits du registre CCR. Le branchement se fait, ou se fait pas, suivant l'état 0 ou 1 d'un ou plusieurs bits du registre codes conditions (CCR).

→ **Adressage relatif long** : Ce mode d'adressage est identique au précédent. Les instructions sont codées sur 4 octets, les 2 premiers déterminent le code opération. Les 3ème et 4ème octets correspondent à la valeur signée du déplacement (le déplacement est codé sur 16 bits :la valeur décimale est comprise entre -32 768 et +32 767).
Exemple : Comparer le contenu de l'accumulateur A avec le contenu de l'emplacement mémoire $2000. S'il y a égalité entre les deux contenus, exécuter un branchement à l'adresse $52C9 (étiquette SAUT), sinon continuer le programme. En assembleur il faut :

Donner une étiquette : SAUT EQU $52C9. Puis, écrire que le branchement doit se faire s'il y a égalité de deux contenus, après comparaison : CMPA $2000 BEQ SAUT



→ **Adressage indexé** : Dans ce mode d'adressage, l'adresse effective/absolue de l'opérande s'obtient en faisant la somme entre un déplacement (offset) associé au code opération et une base qui est contenue dans l'un des registres suivants : *Index (X ou Y), Pointeur de pile (U et S) ou Compteur ordinal (PC).*

Adresse Effective = Base + Déplacement

Enfin ce mode d'adressage permet aussi la **pré-décrémentation** et la **post-incrémentation** simple ou double. Ca permet de travailler sur des emplacements mémoires adjacents, possibilités intéressantes pour le traitement des tables de données.

→ **Adressage indexé (Déplacement Nul)** : C'est le fonctionnement le plus simple de ce mode d'adressage. Le registre d'index contient l'adresse effective de l'octet à manipuler, Exemple :

LDA 0, X → Charge A avec le contenu de l'adresse pointée par la valeur de X.

LDA , X → Charge A avec le contenu de l'adresse pointée par la valeur de X.

→ **Adressage Indexé (avec auto-incrémentation)** : Dans ce mode, l'incrémentation du Registre d'Index aura lieu après l'exécution de l'opération, Exemple :

LDA , X+ Charge l'Accumulateur A avec le contenu de l'adresse $ 1000, mais IX contiendra $ 1001 après l'instruction terminée.

LDD , X++ Charge le Registre D avec les contenus des deux cases d'adresses adjacentes : $1000 & $ 1001 et incrémente deux fois IX après, donc IX = $ 1002.

→ **Adressage Indexé (avec auto-decrémentation)** : Dans ce mode, le contenu du Registre d'Index est décrémenté avant de pointer l'adresse effective, Exemple :

Si le Registre d'Index IX contient $ 1000, LDB,-X Commence par décrémenter le

Registre IX. Donc IX = $ 0FFF et charge l'Accumulateur B avec le contenu de $ 0FFF

N.B *Le travail en auto-incrémentation/décrémentation simple facilite le traitement de tables de données sur 8 bits et le double de tables sur 16 bits.*

Jeu d'instruction

ABX	ajoute l"accumulateur B à X (non signé)
ADCA, ADCB	addition mémoire-accumulateur avec retenue
ADDA, ADDB	addition mémoire-accumulateur sans retenue
ADDD	addition mémoire avec accumulateur D (16 bits)
ANDA, ANDB	ET logique mémoire-accumulateur
ANDCC	ET logique du CCR avec la mémoire
ASL, ASLA, ASLB	décalage à gauche
ASR, ASRA, ASRB	décalage à droit
BCC, LBCC	branchement si pas de retenue (bit C)
BCS, LBCS	branchement si retenu (bit C)
BEQ, LBEQ	branchement su égal à zéro (bit Z)
BGE, LBGE	branchement si supérieur ou égale à zéro
BGT, LBGT	branchement si supérieur à zéro
BHI, LBHI	branchement si plus grand que
BHS, LBHS	branchement si plus grand ou égale à
BITA, BITB	test d'un bit mémoire-accumulateur
BLE, LBLE	branchement si inférieur ou égale à zéro
BLO, LBLO	branchement si plus petit que
BLS, LBLS	branchement si plus petit ou égale à
BLT, LBLT	branchement si inférieur à zéro

BMI, LBMI	branchement su négatif (bit N)
BNE, LBNE	branchement su différent de zéro (bit Z)
BPL, LBPL	branchement si positif (bit N)
BRA, LBRA	branchement inconditionnel
BRN, LBRN	branchement n'ayaint jamais lieu ??????
BSR, LBSR	branchement à un sous-programme
BVC, LBVC	branchement si pas de dépassement (bit V)
BVS, LBVS	branchement si dépassement (bit V)
CLR, CLRA, CLRB	mise à zéro mémoire ou accumulateur
CMPA, CMPB	comparaison mémoire-accumulateur
CMPD	comparaison mémoire-accumulateur D (16 bits)
CMPS, CMPU	comparaison pointeur de pile-mémoire
CMPX, CMPY	comparaison index-mémoire
COM, COMA, COMB	complémentation mémoire ou accumulateur
CWAI	ET logique di CCR et attente d'interruption
DAA	ajustement décimale de A
DEC, DECA, DECB	décrémentation de 1 mémoire ou accumulateur
EORA, EORB	OU exclusif mémoire-accumulateur
EXG D,R	echange de D et de R
EXG R1,R2	échange de R1 et R2 (R1, R2 = A, B, CC, DP)
INC, INCA, INCB	incrémentation de 1 mémoire ou accumulateur
JMP	saut inconditionnel
JSR	saut à un sous-programme
LDA, LDB	chargement d'un accumulateur à partie de la mémoire
LDS, LDU	chargement de la pile à partie de

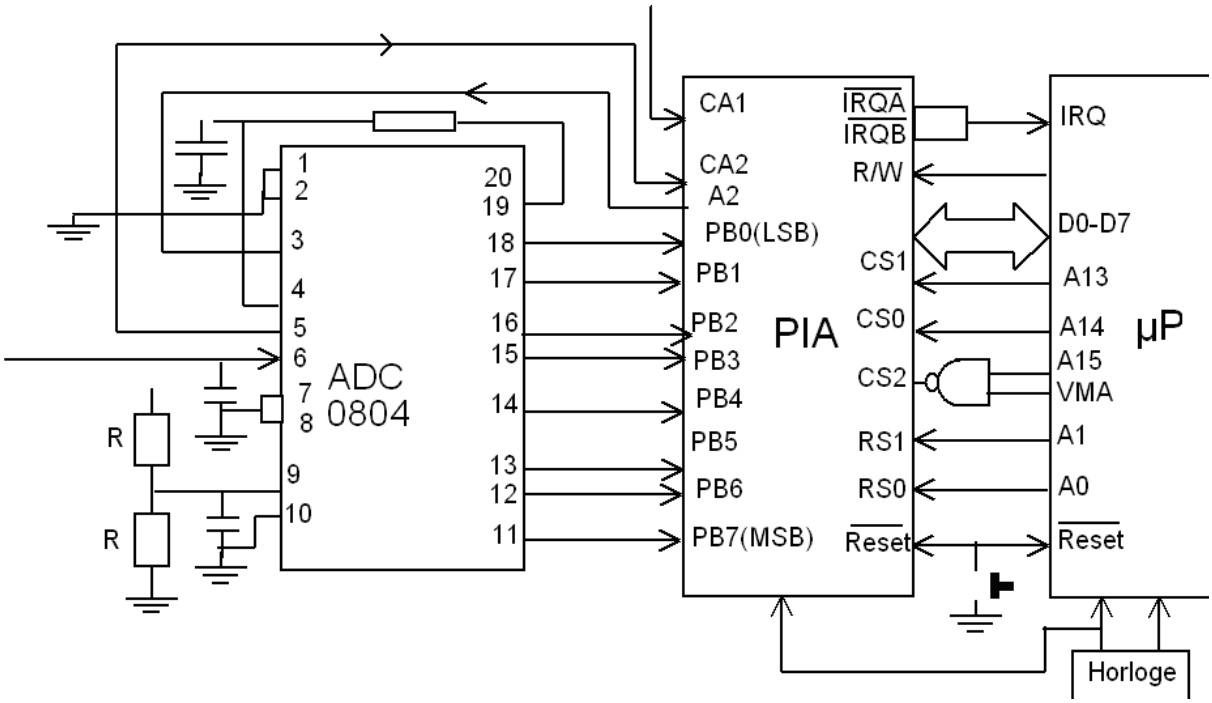
	la mémoire
LDX, LDU	chargement de l'index à partie de la mémoire
LEAS, LEAU	chargement de l'adresse effective dans le pointeur de pile
LEAX, LEAY	chargement de l'adresse effective dans l'index
LSL, LSLA, LSLB	décalage logique à gauche mémoire ou accumulateur
LSR, LSRA, LSRB	décalage logique à droite mémoire ou accumulateur
MUL	multiplication non signée (AxB=D)
NEG, NEGA, NEGB	négation mémoire ou accumulateur
NOP	pas d'opération réalisée
ORA, ORB	OU logique mémoire-accumulateur
ORCC	OU logique du CCR avec la mémoire
PSHS	
PSHU	
PULS	
PULU	
ROL, ROLA, ROLB	rotation à gauche mémoire ou accumulateur
ROR, RORA, RORB	rotation à droite mémoire ou accumulateur
RTI	retour d'interruption
RTS	retour de sous-programme
SBCA, SBCB	soustraction accumulateur-mémoire avec retenue
SEX	extension du signe B au travers de l'accumilateur A
STA, STB	stockage accumulateur en mémoire
STD	stockage de D en mémoire (16 bits)
STS, STU	stockage de pointeru de pile en mémoire
STX, STY	stockage de l'index en mémoire
SUBA,	soustraction accumulateur

SUBB	mémoire sans retenue
SUBD	soustraction D-mémoire (16 bits)
SW11, SW12, SW13	interruption par logiciel
SYNC	synchronisation avec une interruption
TFR D,R	transfert de D dans R
TFR R,D	transfert de R dans D
TFR R1,R2	transfert de R1 dans R2
TST, TSTA, TSTB	test d'une mémoire ou accumulateur

Le Coupleur d'E/S parallèle de la famille Motorola

Le coupleur d'entrée/sortie parallèle est un circuit d'interface programmable qui porte aussi le nom d'adaptateur d'interface périphérique (ou PIA : Peripheral Interface Adapter). Le circuit est prévu pour être connecté à un bus de type 6800, 6802, 6809, ... et, de ce fait, ne nécessite aucun circuit d'adaptation autre qu'un éventuel décodage d'adresse pour placer le PIA dans l'espace mémoire adressable par le mP. Le MC 6820 fût le premier coupleur d'entrée sortie

commercialisé par la firme Motorola. Il n'est plus fabriqué ces derniers temps car le MC 6821 le remplace en corrigeant quelques défauts. Le MC 6822 est identique au MC 6821 sauf au niveau des sorties qui peuvent débiter un courant plus important. Le PIA communique avec le mP à l'aide des signaux (compatibles TTL) représentés sur la figure ci dessous.



Les différents signaux échangés entre le PIA et le mP sont :

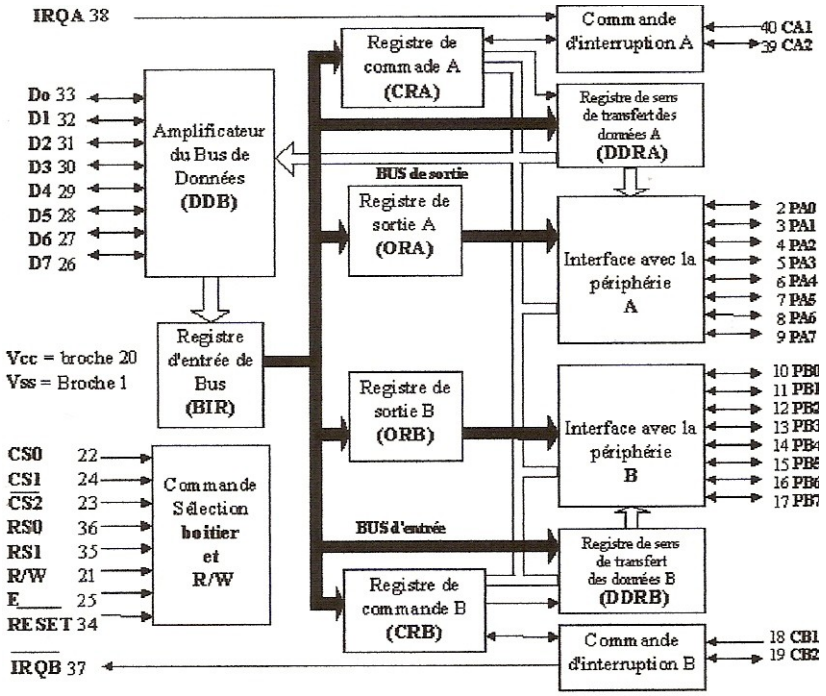
→ D0 à D7 du PIA sont reliées à D0 à D7 du mP et servent à véhiculer les données.
→ E (Enable) du PIA est connectée à l'entrée d'horloge du mP. Elle joue le rôle d'entrée d'horloge pour le PIA.
→ R/W du PIA est reliée à la ligne R/W du mP. Elle sert pour gérer les opérations de lecture ou écriture entre le PIA et le mP.
→ Reset du PIA est connectée à la fois à l'entrée Reset du mP et au poussoir de Reset (ou à la circuiterie de remise à zéro automatique à la mise sous tension). Cette entrée permet la remise à zéro du PIA.
→ CS0, CS1 et CS2 sont des lignes de sélection du boîtier PIA qui ne sera activé que lorsque CS0= CS1 = 1 et CS2 = 0. Ces lignes sont reliées à la circuiterie de décodage d'adresse du système afin de fixer l'adresse de base du PIA.
→ RS0 et RS1 permettent de sélectionner les registres internes du PIA selon le tableau ci-dessous. Généralement, RS0 est reliée à A0 et RS1 est reliée à A1, ce qui place les quatre registres les uns à la suite des autres.
→ IRQA et/ou IRQB seront ou non reliées à la ligne IRQ du mP 6802 ou à IRQ, FIRQ ou NMI du
mP 6809 selon que l'on souhaitera ou non que le PIA puisse générer des interruptions avec son côté A, ou B, ou les deux.

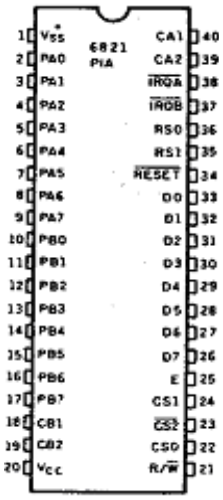
Description générale
Comme le schématise la figure suivante, le PIA est un composant électronique à 40 broches réalisé en technologie N-MOS. Ce circuit dispose côté « extérieur » de 16 lignes

d'entrées/sorties programmables individuellement et indépendamment les unes des autres en entrées ou en sorties.
Cette programmation se fait par logiciel et non par des mises à la masse ou au +5V de broches (ce qui implique qu'elle peut être changée de manière dynamique dans un programme).
D'autre part,une ligne d'entrée/sortie peut jouer alternativement le rôle d'entrée et de sortie, comme nous le verrons lors des exemples d'utilisation par la suite.
Il dispose, de plus, de quatre lignes de dialogue pouvant générer des interruptions dont deux sont des entrées (CA1 et CB1), les deux autres étant programmables en entrée ou en sortie (CA2 et CB2).
Comme tous les circuits périphériques de la famille Motorola, le PIA est vu par le mP, comme quatre positions mémoires : le dialogue avec le circuit et avec ce qui est relié aux lignes d'entrées/sorties du PIA se fait donc par des lectures et des écritures mémoires aux quatre adresses correspondantes.

Structure interne

Le PIA est un système pratiquement symétrique comportant deux ports de communication appelés port « A » et port « B » (voir la figure suivante). Chaque port comprend 8 lignes programmables en entrée/sortie, et ceci une à une. Ces lignes s'appellent PA0 à PA7 pour le port « A » et PB0 à PB7 pour le port « B ».





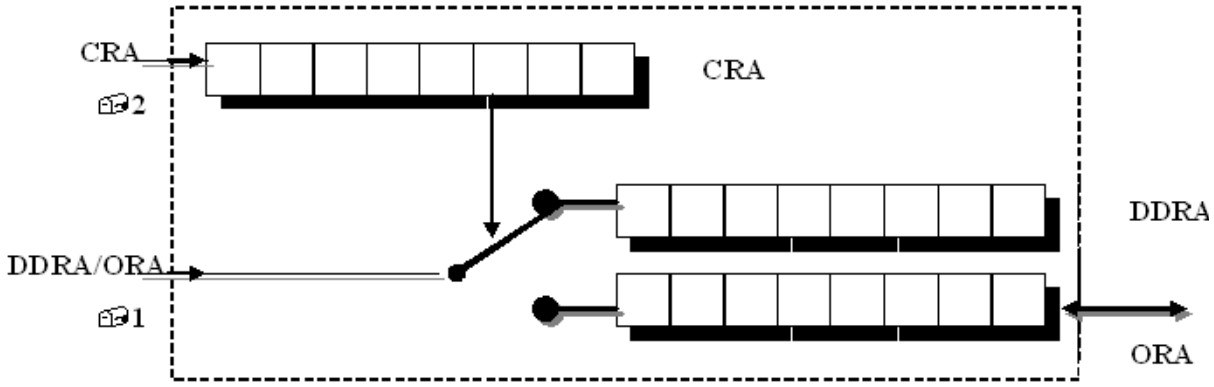
Le PIA possède six registres internes (voir la figure ci-dessus), soit trois registres par port :

→ DDRA Data Direction Register A : C'est un registre de direction de données pour le port « A ». Il contient le mot fixant le sens de transfert (en entrée ou en sortie) pour chacune des lignes de données. Un état « 1 » définit une broche en sortie et un état « 0 » définit une broche en entrée
→ ORA (Output Register A) ou registre de sortie pour le port « A » : Ce nom est assez mal choisi ; en effet, ce registre est celui dans lequel le mP viendra placer les données à faire sortir du PIA (là le nom est correct) mais c'est dans ce même registre que l'on viendra lire les données présentes sur celles des lignes PA0 à PA7 qui auront été programmées en entrée. En résumé, ORA est en fait un registre « image » des lignes PA0 à PA7.
→ CRA Control Register A : C'est un registre de contrôle permettant, pour le port « A », de définir le mode de fonctionnement des lignes d'interruption et de dialogue CA1 et CA2 ainsi que les possibilités de génération d'interruption via la ligne IRQA.
→ DDRB idem, mais pour le port « B ».
→ ORB idem, mais pour le port « B ».
→ CRB idem, mais pour le port « B ».
N.B. *Quand on effectue un Reset, le contenu de tous les registres du PIA est mis à 0, y compris le registre DDRA ou DDRB. Il en résulte que toutes les lignes se trouvent configurées en entrée.*

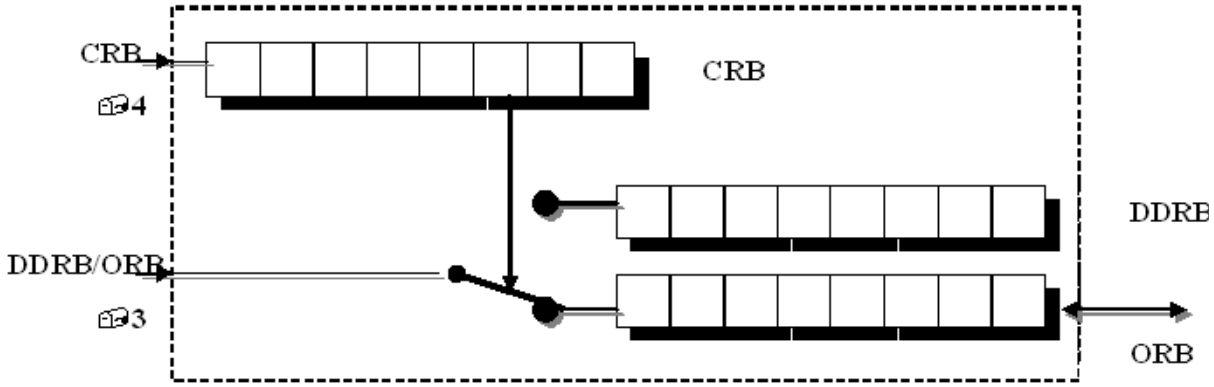
Le PIA est composé également :
→Des Amplificateurs de bus bidirectionnels à trois états.
→ D'une ligne de contrôle interprétant les divers signaux de commande issus du bus du mP.
→ Des Amplificateurs d'interface pour les lignes PA0 à PA7 pour le port « A » et PB0 à PB7 pour le port « B ».
→D'une ligne de contrôle qui à partir de CA1 et CA2 pour le port « A » et CB1 et CB2 pour le port « B » génère une interruption IRQA ou IRQB qui sera envoyée vers le mP.

<u>Principe de fonctionnement</u>
Toute la « science » du PIA repose sur la manipulation du contenu de ses six registres internes Voyons maintenant comment on peut accéder à ORA ou à DDRA puisque ces deux registres occupent la même adresse. Pour se faire, reportons nous à la figure suivante, qui précise le rôle de chaque bit du registre de contrôle CRA ou CRB.

Exemple : Accéder au DDRA



Exemple : Accéder au ORB



Le bit c2 permet de sélectionner l'un des registres ORA ou DDRA : si c2 = 0, on accède au registre DDRA ; si ce bit est à 1, on accède au registre ORA.

N.B. *i) Le même raisonnement étant valable pour le port (B) du PIA.*
ii) Cette façon de faire permet d'une part de réduire l'espace mémoire occupé par le PIA (quatre emplacements au lieu de six) et d'autre part, gagner une broche au niveau de son boîtier (puisqu'il suffit de deux lignes RS0 et RS1 pour sélectionner les six registres, alors qu'il en faudrait trois sans cette astuce !).

Procédure d'initialisation du PIA

La procédure d'initialisation du PIA est généralement la suivante :
1°) On accède au registre de contrôle (CRA ou CRB) du port concerné et l'on y place $00 = (00000000), ce qui autorisera l'accès au registre (DDRA ou DDRB) puisque le bit « c2 » est à 0.
2°) On accède ensuite au registre de direction de données (DDRA Ou DDRB) où l'on écrit la configuration désirée.
3°) On accède de nouveau au registre de contrôle (CRA ou CRB) dans lequel on écrit le mot de contrôle correspondant à la fonction désirée et où le bit « c2 » est à 1 : cela permet ensuite d'accéder aux lignes d'entrées/sorties via le registre de sortie (ORA ou ORB)

Accéder au CRA
Accéder au DDRA
Port A en entrée
Accéder au CRA
Accéder au ORA

Accéder au CRB
Accéder au DDRB
Port B en sortie
Accéder au CRB
Accéder au ORB