# LE MICROPROCESSEUR : UNE REVOLUTION

L'apparition du microprocesseur en 1974 à été pour l'électronique une révolution. Ce mot a été beaucoup utilisé dans le passé en particulier à l'apparition des transistors puis des circuits intégrés mais il n'a jamais été aussi justifié.

La première révolution fut l'arrivée des **transistors** .

Ces derniers ont à partir des années 60 remplacé les tubes électroniques dans tous les domaines, ils consomment peu ,fonctionnent sous faible tension , sont fiables et en particulier à peu près insensibles aux chocs. Mais la structure des schémas n'a pas changé de façon fondamentale. Un circuit à transistors peut être réalisé avec des triodes ou pentodes , bien sûr la forte consommation des tubes et leur faible fiabilité interdit d'en assembler un trop grand nombre. Les premiers téléviseurs couleur à tubes en sont un bon exemple. C'était des monstres horriblement coûteux peu fiables et qui faisaient surtout office de radiateur .La TV couleur n'était commercialement possible qu'avec des transistors. Les ordinateurs constituent aussi un bon exemple, il est possible de fabriquer un ordinateur avec des tubes , ce fut d'ailleurs le cas des premières machines, mais avec plusieurs centaines de triodes elles étaient moins puissantes qu'une calculette actuelle vendue un euro et un technicien d'entretien devait être présent en permanence pour réparer les pannes fréquentes de l'engin !

Surtout, les transistors grâce à leur fiabilité et leur taille ont permis de mettre au point des structures de circuits qui auraient été irréalisables avec des tubes .

La seconde révolution est l'arrivée des **circuits intégrés** ,elle mérite encore moins ce qualificatif. Un circuit intégré n'est que l'association de transistors. Il est vrai que le gain en fiabilité a permis en associant des milliers de composants de réaliser des fonctions qu'il aurait été impossible de construire avec des transistors individuels .

Avec les **microprocesseurs** l'électronique à vécu une vraie révolution car c'est **la structure même des circuits** qui a été changée. L'électronique est passée de la structure câblée à une structure programmée.

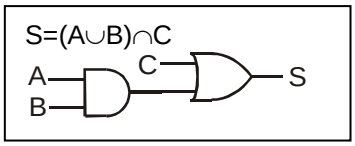
## Electronique câblée et électronique programmée

Les microprocesseurs sont essentiellement des circuits logiques

Soit à réaliser la fonction logique suivante mettant en œuvre 3 variables binaires A B et C

$S=(A \cap B) \cup C$

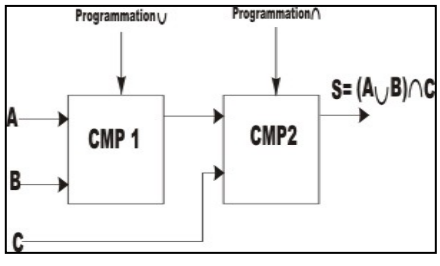
Nous utilisons ici la notation ensembliste , $\cup$ désignant le OU logique ( addition logique ) et $\cap$ le produit logique , pour éviter toute confusion avec les opérations arithmétique + et * .




Dans les années 60 le technicien utilisait des portes logiques ET et OU et proposait le circuit suivant mettant en œuvre deux boîtiers TTL 7402 et 7408

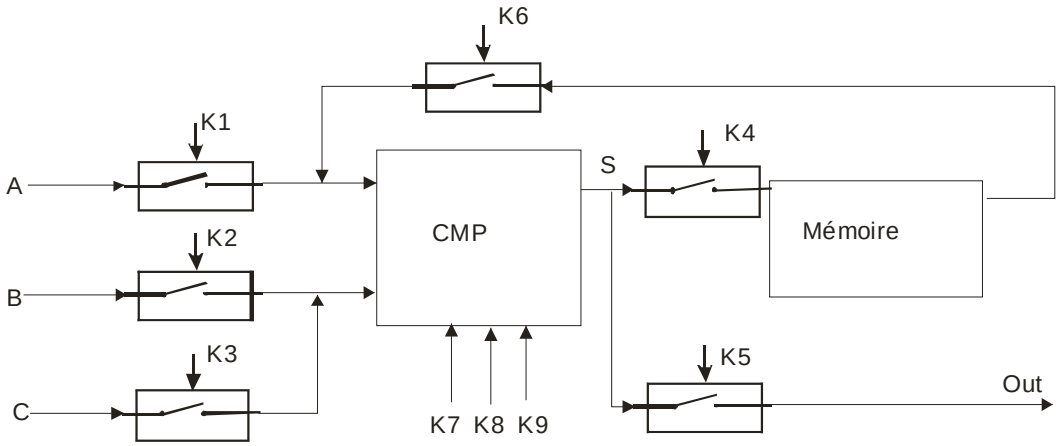
Au début des années 70 l'évolution des techniques d'intégration permet de fabriquer des circuits remplissant plusieurs fonctions arithmétiques (addition soustraction ) et logiques (et ou complément ) ,la fonction réalisée dépend de l'état de plusieurs pattes du circuit. Ce sont les **circuits multifonctions programmables** (CMP) Il est alors possible de réaliser la fonction précédente en n'utilisant qu'un seul type de circuit , chaque boîtier étant programmé séparément ? ( figure ci contre )

Le seul intérêt de cette solution est de faciliter les problèmes d'approvisionnement puisqu'il n'y a plus besoin que d'un seul,type de boîtier, mais chaque circuit individuellement très puissant est considérablement sous employé . La solution est bien sûr de faire effectuer a un seul CMP toutes les opérations nécessaires, mais il ne peut pas les effectuer en même temps , l'opération devient **séquentielle** et une horloge est nécessaire.



Pour le problème précédent la structure peut être la suivante.Le CMP est associé à une batterie d'interrupteurs ouverts ou fermés suivant les besoins.

L'opération s'effectue en deux étapes :




1° les interrupteurs K1 K2 K4 sont fermés K3 K5 et K6 ouverts , d'autre part les bits K7 K8 K9 programment le CMP comme une fonction OU .( par exemple 001) La sortie S contient la somme A∪B qui est introduite dans la mémoire via l'interrupteur K4

2° Les interrupteurs K1 K2 et K4 sont ouverts , K3 K5 K6 fermés , et les trois bits K7 K8 K9 programment le CMP en ET logique (Par exemple (010) Le CMP reçoit le contenu de la mémoire c'est-à-dire (A∪B) et le combine avec le signal C soit le résultat cherché qui est aiguillé vers la sortie out.

Il est bien sûr possible de réaliser un tel circuit avec des interrupteurs commandés à la main l'un après l'autre , mais il est beaucoup plus intéressant d'automatiser le processus. .Par exemple les bits de programmation des interrupteurs eu du CMP peuvent constituer des mots de N bits (ici 9 bits ) .Mot1 ; K9K8K7K6K5K4K3K2K1=001001011

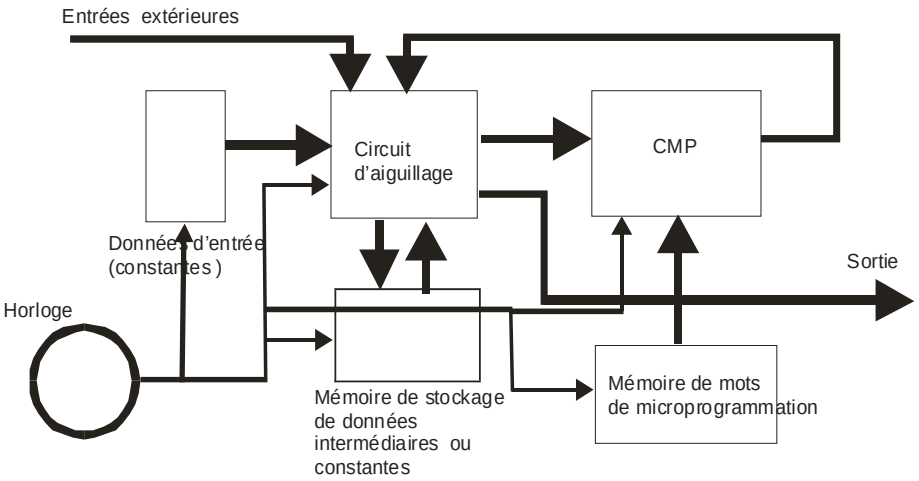
Mot2 010110100

Ces mots , nous les appellerons **mots de microprogrammation** seront stockés dans une mémoire , la **mémoire de microprogrammation** et seront lus l'un après l'autre par un signal commandé par l'horloge.

Pour effectuer des opérations plus compliquées , le circuit précédent doit être modifié, il prend la forme générale représentée sur la figure ci-dessous .

Les bits des signaux d'entrée sont stockés dans une mémoire d'entrée et les résultats envoyés dans une mémoire de stockage ou les résultats intermédiaires sont puisés.

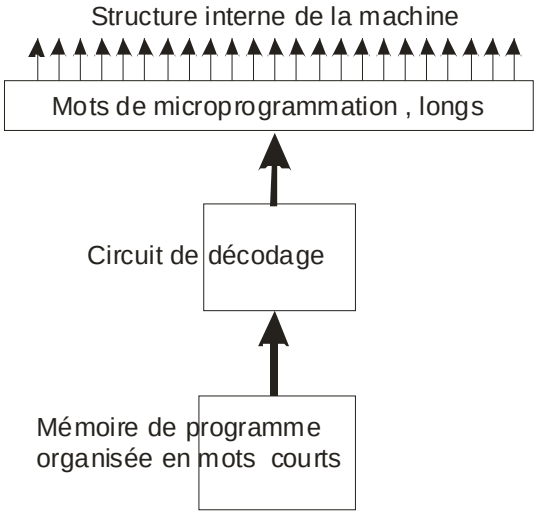
Le CMP est commandé par un circuit d'aiguillage complexe reliant entrée , sortie et mémoires .Les interrupteurs précédents sont alors beaucoup plus nombreux, plusieurs centaines peut être .



L'exécution de la tâche est alors pilotée par la suite des mots de microprogrammation qui commandent l'aiguillage des données entre le CMP et les circuits extérieurs. On voit apparaître la notion de programme .L'électronique est dite **programmée**.

**Mémoire programme et mémoire de données.**

Les données traitées par le circuit sont des mots de plusieurs bits ( 4 ou 8 dans les années 70 ) ils sont donc stockés dans des mémoires organisées en mots de cette longueur. Pour les mots de microprogrammation il en est autrement car ils sont constitués d'un grand nombre de bits .Or une mémoire organisée en mots de 100 bits par exemple n'est pas économique à construire. On peut remarquer cependant que si ces mots ont 100 bits leur nombre, c'est-à-dire le nombre de configurations viables de la machine , est considérablement inférieur à $2^{100}$ Il est alors possible d'utiliser un subterfuge. Supposons que le nombre de mots de programmation soit inférieur à 256 , chacun d'entre eux peut alors être désigné par un numéro codé sur 8 bits seulement. , ce sont ces octets , **octets de programme** qui seront stockés dans la mémoire de programme, un circuit de décodage sera chargé de reconstituer les mots de microprogrammation précédents nécessaires pour piloter les éléments intérieurs du circuit.




A ce niveau deux options sont possibles .

--La longueur des mots de programme est choisie égale à celle des mots de données; il est alors possible de les stocker dans **la même mémoire** .Nous verrons plus loin comment les identifier. Pour une machine traitant des mots de 8 bits il semble que cela limite le nombre de mots de microprogrammation à 256 , en fait il est possible de définir le mot de microprogrammation à partir de 2 octets ce qui fait passer le nombre maximum à 65536 , plus que largement suffisant dans tous les cas.

Cette solution a été retenue pour la quasi-totalité des microprocesseurs de première génération , c'est la structure de **Von Neumann**

- Il peut être plus commode de choisir des mots de programme plus longs 12 14 ou 16 bits , ils ne peuvent plus alors être stockés dans la même mémoire que les données , c'est la structure Harvard qui est en particulier celle des microprocesseurs de Microchip .

**L'apparition du microprocesseur en 1973**

Le premier circuit de ce type a été conçu par une petite société californienne alors inconnue INTEL à la suite d'une commande d'industriels japonais qui voulaient fabriquer une petite calculette électronique. Ce circuit fut le 4004 .Son apparition en septembre 1971 passa presque inaperçue, mais quelques ingénieurs travaillant à la construction des gros ordinateurs de l'époque remarquèrent que sa structure était très semblable à celle du cœur d'un ordinateur que l'on appelait le processeur . D'où le nom de **microprocesseur** qui sera donné au nouveau circuit. De nombreuses applications devenaient possibles et la société INTEL entama rapidement la construction d'un circuit plus puissant traitant des mots de 8 bits , ce fut le 8008 sorti en 1972 , d'emploi difficile ,qui fut rapidement amélioré pour devenir en 1973 le 8080 , le vrai premier microprocesseur de la nouvelle électronique.

Les concurrents ne restèrent pas les bras croisés mais ce n'est qu'en 1975 que Motorola commercialisa le 6800. Bien que les fonctionnalités soient sensiblement les mêmes ces deux circuits ont des structures un peu différentes, ces différences persisteront dans les composants suivants 8086 80186 … chez INTEL , 6809 68000 chez Motorola. Plus tard cette différence se retrouvera entre les microordinateurs type PC et MAC .Il y eu bien d'autres constructeurs , TEXAS ( Les TMS ) , NATIONAL SEMICONDICTORS (le SCAMP) PHILIPS ( 2650 ) ROCKWEL (6502 , le cœur des premiers Apple ) ZILOG ( avec le Z80 une très bonne amélioration du 8080 ) ,et bien sûr les japonais Tous ces microprocesseurs ont une structure de Von Neumann, les structures Harvard étaient encore rares jusqu'à l'apparition des microprocesseurs PIC de la société MICROCHIP dont le succès est impressionnant.

# LES CIRCUITS ASSOCIES AU MICROPROCESSEUR

## Les technologies

Les microprocesseurs sont des circuits logiques , ils sont construits avec les mêmes technologies que les autres circuits logiques .

**Technologie TTL** actuellement en voie de disparition La TTL consomme trop de courant pour permettre l'intégration sur une même puce de nombreux transistors , les microprocesseurs TTL furent rares (microprocesseurs en tranches des années 70 , disparus )

**Technologie MOS P ou N** ce fut le cas des premiers circuits Les MOS sont beaucoup plus faciles à intégrer que les bipolaires et consomment moins .

**Technologie CMOS** , elle est plus difficile à intégrer mais bénéficie d'une consommation très faible , nulle au repos ce qui permet de construire des ensembles très complexes .

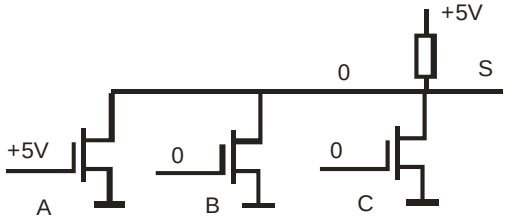
Les circuits modernes font appel le plus souvent à une structure CMOS mais des mélanges de technologies sont maintenant courantes .

## Les structures de sortie , le BUS

Pour relier ensemble un grand nombre de circuits , il est commode de les accrocher sur un seul conducteur , le BUS ,. Pour éviter alors les conflits entre sorties deux structures sont utilisées .

### La structure collecteur ouvert. (OP)

Le composant final d'un circuit est un transistor ou un MOS ne possédant pas de charge vers l'alimentation. Si plusieurs sorties sont reliées ensembles elles partagent une même charge R (figure ci contre ) , dès que l'un des transistor est conducteur il force le niveau de sortie à zéro . Le niveau bas est dominant. Le montage réalise un ET câblé.

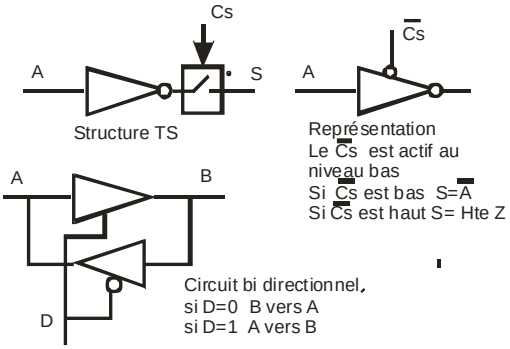



La MOS A au dont l'entrée est au niveau 1 force au niveau bas la sortie S quel que soit l'état des MOS B et C

### La sortie trois états (TS)

Un circuit trois états **(TS)** est l'association d'un circuit normal et d'un interrupteur série en sortie Cet interrupteur est commandé par un signal spécial de validation noté le plus souvent Cs ou /Cs suivant qu'il est actif au niveau 1 ou 0 . La sortie peut alors se présenter sous trois états O ou 1 si l'interrupteur est fermé, haute impédance si il est ouvert. Un circuit non sélectionné se comporte comme s'il n'était pas relié au BUS .

La structure TS permet en particulier de concevoir des circuits bi directionnels comme le montre la figure ci contre .

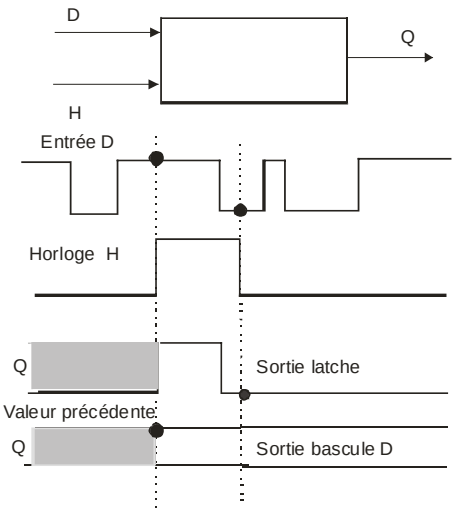



## Les décodeurs

Un décodeur est un circuit possédant N entrées et $2^N$ sorties, une seule d'entre elle étant active à la fois Il possède le plus souvent une sortie TS , c'est-à-dire une entrée de sélection comme décrit ci-dessus On rencontre souvent le circuit TTL 74138 qui possède 3 entrées , 3 entrées de sélection et 8 sorties Y7 à Y0 dont une seule est active au niveau bas .

### Les ' latches ' ( verrous )

Ce sont des dispositifs que l'on rencontre très souvent sur les cartes microprocesseur. Ils permettent de mémoriser au vol une information présente très brièvement sur un BUS
Ces circuits possèdent une entrée nommée souvent D , une sortie Q et une commande H (horloge)
En fait il existe deux catégories de circuits dont le comportement est différent : ( figure ci contre )




**Les latches** proprement dits :

Lorsque le signal d'horloge est au niveau haut la sortie Q recopie l'entrée D , la valeur ainsi recopiée reste bloquée lorsque H redescend au niveau bas .

| Latche | Q |
|---|---|
| H=1 | D |
| H=0 | Q précédent |

**La bascule D**

Q prend la valeur de D à l'instant du front de montée de l'horloge H

## Les Mémoires

Nous avons vu plus haut pourquoi elles sont indispensables en électronique programmée.

Une mémoire est **un assemblage d'éléments binaires**

Elles peuvent être :

à lecture seule , ce sont les mémoires mortes **ROM** (Read Only Memory )

à lecture et écriture possible , ce sont les **mémoires vives**

Les éléments binaires les constituant peuvent être individuels (mémoires en mots de 1 bit ) ou groupés pour former des mots de N bits (le plus souvent 8 ) Chaque élément bit ou mot est défini par son **adresse** .

On distingue encore

Les **mémoires séquentielles** dont les éléments sont placés en ligne l'un derrière l'autre .Pour en atteindre un il faut passer sur les précédents. C'est le cas des bandes magnétiques

Les **mémoires à accès aléatoire** , l'accès à chaque mot est directe .Penser à un dictionnaire l'adresse du mot est dans ce cas sa structure alphabétique elle-même.

Cas particuliers : **Les registres** ce sont des mémoires ne contenant qu'un seul mot

Les **registres à décalage ,** un signal d'horloge peut décaler vers l'entrée ou la sortie le contenu du registre. On distingue les registres **LIFO** ( Dernier entré premier sorti , comme dans un ascenseur à un seul accès ) ou **FIFO** ( premier entré premier sorti comme dans un ascenseur à 2 portes opposées )

Il faut ajouter

Les mémoires **non volatiles** dont le contenu subsiste en cas de coupure de l'alimentation

Les mémoires **volatiles** qui peuvent être :

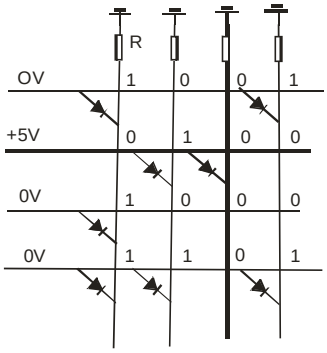
**Statiques** .Les données disparaissent si l'alimentation est coupée mais persistent tant que l'alimentation est active.

**Dynamiques** L'information est l'état d'un condensateur qui se décharge spontanément au bout d'une durée plus ou moins courte. Pour maintenir cette information il faut en permanence rafraîchir la mémoire en la lisant. Ces mémoires sont d'un emploi plus complexe elles ne sont pas utilisées avec les microprocesseur , au moins pour les applications simples. Elles constituent par contre le cœur de la mémoire des ordinateurs.

Par abus de langage des mémoires vives statiques à accès aléatoire sont appelées **RAM** (Randon Acces Memory )

## Structure d'une ROM

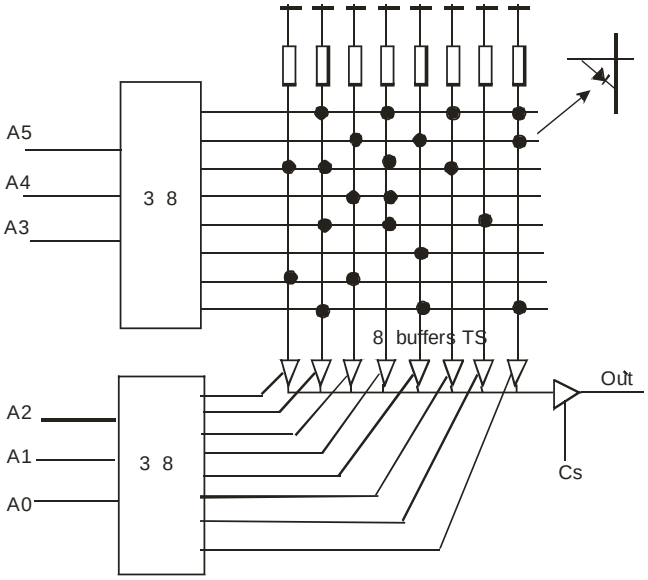
Les éléments binaires sont répartis aux nœuds d'un quadrillage de lignes et colonnes. Initialement l'élément binaire était une diode , la présence d'une diode représente un bit 1 son absence un bit 0 .Pour lire l'état d'un bit il suffit de porter au niveau 1 une ligne et une seule et de lire la tension sur la colonne ou se trouve l'intersection visée .( figure ci contre ) .




Cette structure n'est cependant pas acceptable dès que le nombre de bits à stocker dépasse quelques dizaines .Pour 1024 bits il faudrait 32 lignes et 32 colonnes soit 64 broches .

De plus le circuit de commande doit veiller à ce qu'une seule ligne soit portée au niveau haut.

Pour n'activer automatiquement qu'une seule ligne et réduire le nombre d'accès nécessaires, il suffit d'utiliser des décodeurs .La figure ci-dessous représente par exemple une mémoire morte de 64 bits .




Les lignes sont commandées par les sorties d'un décodeur 3 entrées 8 sorties dont une seule est active au 1 à la fois. La lecture des colonnes s'effectue de la même manière grâce à 8 portes trois états commandées par les sorties d'un second décodeur 3 8 . Ainsi 6 broches seulement assurent la lecture des 64 points mémoire . Les 6 bits A5A4A3A2A1A0 constituent un mot de 6 bits qui est l'**adresse** du point mémoire visé. Il faut noter qu'il était tout à fait possible de répartir les 64 points en 16 lignes et 4 colonnes , les deux décodeurs seraient alors un 4 16 pour les lignes et un 2 4 pour les colonnes .Le mot mémoire est le même. L'utilisateur n'a pas à connaître la répartition interne.

Une sortie TS est toujours présente . elle est essentielle pour permettre l'association des boîtiers comme nous le verrons plus loin.

Les ROM ainsi constituées ne peuvent être remplies qu'a la fabrication. Il n'est pas possible de disposer d'une ROM spécifique pour réaliser une maquette .

Le circuit précédent est organisé en mots de 1 bits, pour l'organiser en mots de N bits il suffit de placer en parallèle N plans identiques commandés par les mêmes fils d'adresse.

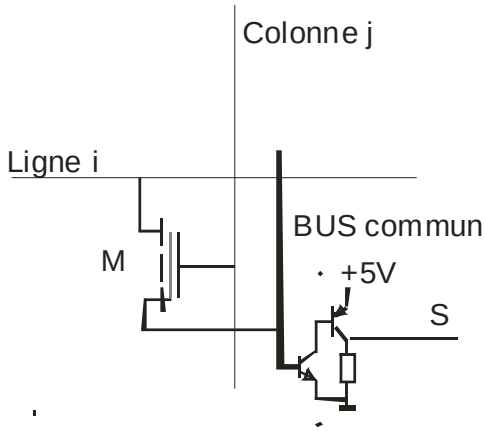
## Les PROM ,

Ce sont des PROM programmables par l'utilisateur .Leur structure est semblable à celle des ROM à diodes mais le point mémoire est un MOS , plus exactement un MOS à grille flottante. .

Leur fonctionnement est illustré par la figure suivante . Le point mémoire est un MOS M canal N dont le drain est relié au fil de ligne et la grille au fil de colonne. Si ces deux lignes sont portées au niveau haut (5V ) M est conducteur , son courant de source est injecté dans la base du transistor de sortie (commun à tous les points mémoire ) et le niveau de sortie est 1

Les premières mémoires MOS étaient construites en intégrant un MOS uniquement aux croisements mémorisant un 1 logique. Le 0 correspondait à une absence de MOS .

Pour autoriser la programmation par l'utilisateur un MOS est placé à la construction à toutes les intersections ligne-colonne mais une grille supplémentaire flottante.est intercalée entre le canal et la grille de commande normale Si cette grille annexe n'est pas chargée elle se comporte comme un écran et la grille de commande ne peut plus rendre le canal conducteur.Tout se passe comme si le MOS n'existait pas Le point mémoire stocke un niveau 0 .. En appliquant une impulsion de tension à un endroit convenable la grille flottante se charge ,cette charge est ensuite permanente ( plusieurs dizaines d'années,on dit parfois un siècle !) ,Le MOS devient alors sensible à une tension appliquée à la grille de commande , le point mémoire correspondant stocke un 1.. Pour écrire un bit il suffit d'appliquer une impulsion sur une entrée spéciale pendant que le point mémoire visé ou l'on veut entrer un 1 est sélectionné par son adresse.Le programmateur est peu coûteux et facile à construire , il permet d'entrer dans la mémoire les informations binaires désirées.




Avantage supplémentaire , il est **possible d'effacer** d'un bloc tout le champ mémoire en éclairant le substrat avec des rayons ultraviolets qui par effet photoélectrique rendent la silice qui isole les grilles conductrice .

Le premier circuit de ce type qui apparut sur le marché est le 2704 qui contenait 512 mots de 8 bits .D'emploi difficile il fut vite détrôné par la 2708 de taille double ( 1024 mots de 8 bits , 1K octets) .Ces circuits ont pratiquement disparu , la PROM la plus petite est actuellement la 2764 contenant 8k octets , et les plus grosses atteignent plusieurs méga octets . Tous ces boîtiers ont un brochage compatible .

## Les PROM Flash

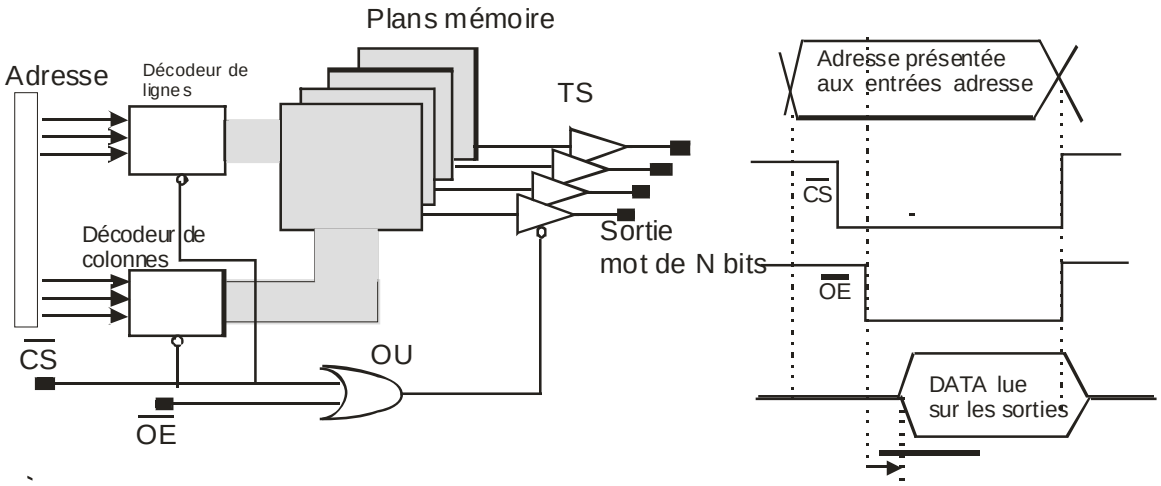
Sur un principe semblable on est maintenant capable de construire des boîtiers mémoire dont les MOS peuvent être individuellement ,par des impulsions électriques , rendus conducteur ( comme les PROM ) ou isolants ( comme pour l'effacement UV des PROM ) Ce ne sont pas tout à fait des RAM statiques non volatiles car le processus d'effacement est long (de l'ordre de la milliseconde ou plus ) . Ce sont les **mémoires FLASH** dont l'emploi se répand rapidement .

## Le boîtier de ROM typique

Une mémoire morte a de façon la plus générale la structure ci contre. Les deux décodeurs de ligne et colonne sont commandés par des signaux binaires constituant l'adresse de chaque point mémoire. , la (ou les sorties si le boîtier est organisé en mots) , est pilotée par un signal de sélection et très souvent par un autre signal appelé OE (Output Enable ) Le fil de sortie se trouve en état de haute impédance tant que CS et OE ne sont pas tous deux actifs .( au niveau bas pour le schéma proposé

Le graphique de droite est fondamental pour tout utilisateur qui construit un circuit utilisant des ROM . La ligne du haut représente schématiquement l'ensemble des signaux binaires constituant l'adresse , ces signaux doivent être stables pendant toute la durée de lecture . La ligne suivante représente l'application du signal de sélection /Cs puis /OE .Ce n'est que lorsque ces deux signaux sont au niveau actif bas que la donnée inscrite dans la mémoire apparaît sur les fils de sortie , avec un petit retard qui est le **temps d'accès** du boîtier .

) .




## Les RAM statiques (SRAM)

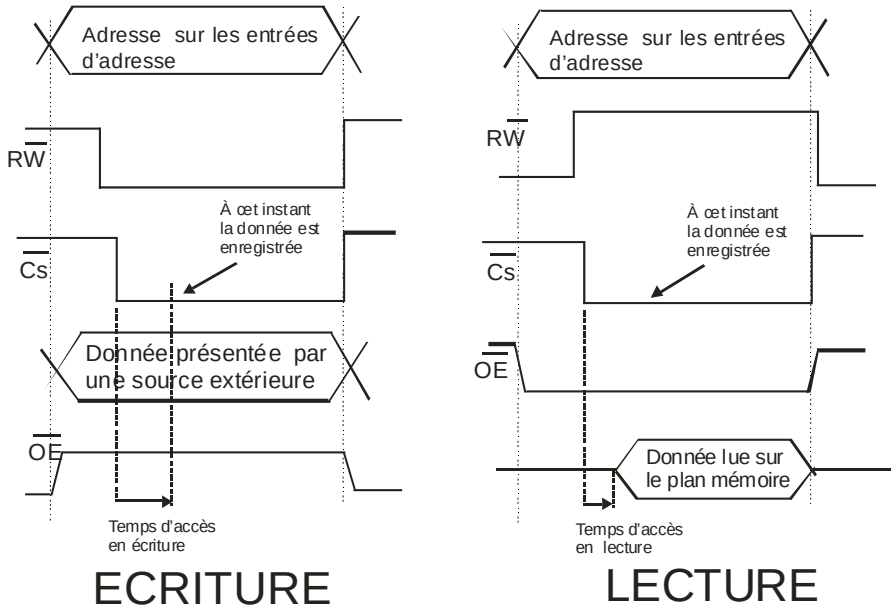
Le point mémoire est dans ce cas une bascule qui peut si elle est alimentée se trouver dans deux états .La structure ressemble à celle des ROM , décodeurs et plans mémoires mais il existe une Broche de plus permettant la lecture d'une case mémoire ou son écriture . .Ce signal supplémentaire est appelé /W ou R/W .Le tableau ci contre montre les différentes configurations possibles des signaux de commande du boîtier.

| /CS | /OE | R/W | Sortie |
|---|---|---|---|
| 1 | x | x | Ht Impédance |
| 0 | 0 | 1 | Lecture |
| 0 | 1 | 0 | Ecriture |

. Si R/W est au niveau haut le boîtier se comporte comme une ROM , le signal de sortie est le contenu du point mémoire adressé dès que /Cs est actif au niveau bas ? Il faut bien noter que dans ce cas la sortie doit être accessible ce qui impose /OE=0 .

Si au contraire R/W est au niveau bas , une donnée présentée à l'entrée est inscrite dans le plan mémoire dès que /Cs est actif. et le signal de validation de sortie au 1 . La présence du signal OE permet d'utiliser le même accès pour l'écriture (alors /OE=1) et la lecture (alors /OE=0). IL existe quelques rares boîtiers mémoire ayant des fils d'entrée et de sortie différents .

Ceci est résumé sur la figure ci-dessous qui représente les graphs des signaux d'écriture et lecture . On notera qu'en lecture la donnée est appliquée en sortie par le boîtier alors qu'en entrée elle doit être présentée sur les mêmes fils par un circuit extérieur.



## LA STRUCTURE DE L'UNITE CENTRALE DE MICROPROCESSEUR ( C P U )

La structure proposée au début de ce chapitre nous a permis d'introduire la notion de mots de microprogramme , puis de décodeurs , mémoire programme et structure de Von Neuman ou Harward .Elle ne correspond cependant pas à la réalité. Pour préciser la structure réelle de l'unité centrale d'un microprocesseur (Central Processing Unit ou **CPU** ) nous allons nous fixer une tâche et construire la structure capable de l'effectuer .

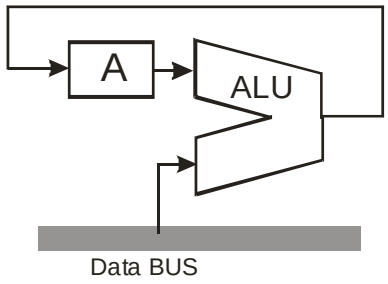
Nous supposons que le CPU est relié à une mémoire extérieure dans laquelle deux nombres P et Q ont été préalablement entrés dans deux cases MP et MQ ; nous cherchons alors à effectuer la somme A+B et la charger dans la case mémoire d'adresse MR .Soit symboliquement :

(MP) + (MQ) ⇒ (MR)

La parenthèse indique un contenu , (MP) contenu de la case d'adresse MP .

Le cœur d'un micropropcesseur est le circuit multifonctions programmable, on l'appelle dans ce cas une **Unité Arithmétique et Logique** (ALU) .Cette **ALU** effectue des opérations sur 2 opérandes ,l'un d'entre eux est placé dans un registre spécial nommé Accumulateur (A ) le second vient de l'extérieur, une mémoire par exemple via un groupe de conducteurs constituant le **BUS de DONNéES** (Data Bus DB ) .Le résultat de l'opération est rechargé dans l'accumulateur .Soit symboliquement :

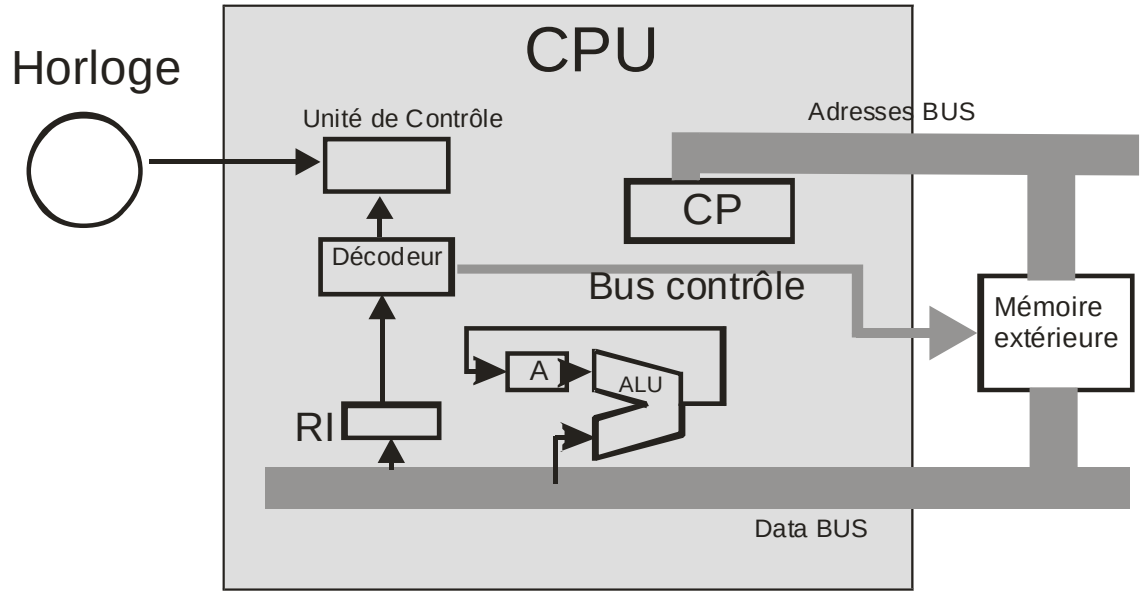
**(A)** □ **(DB)** ⇒ **(A)** le signe □ représentant une opération arithmétique ou logique




Le CPU pour dialoguer avec les mémoires doit être capable de leur fournir une adresse et des signaux de commande . Les adresses sont souvent des nombres binaires successifs , elles sont crées par un circuit qui ressemble à un compteur c'est le **compteur programme (CP )** ou Compteur Ordinal Les conducteurs amenant ces adresse constituent le **BUS d'ADRESSES** ;Les signaux d'écriture et lecture circulent sur un **BUS de CONTROLE**

Nous avons vu d'autre part que les mots de programme doivent être transformés en mots de microprogrammation , pour cela ils doivent être stockés provisoirement dans un registre spécial , le **REGISTRE D'INSTRUCTIONS** (RI) avant d'être appliquées au décodeur appelé **UNITE de CONTROLE**

A ce niveau la structure du système prend la forme suivante:

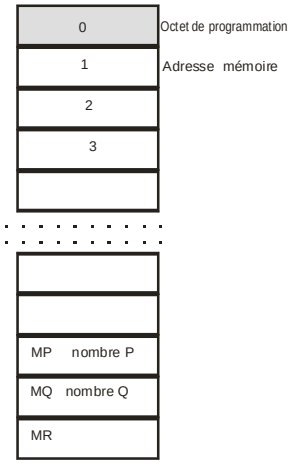



Avec cette structure la tache proposée plus haut nécessitera 3 étapes :

**1** Chargement du nombre P dans l'accumulateur (MP) ⇒ (A)
**2** Addition du contenu de A et du nombre Q c'est-à-dire (A) + (MQ) ⇒ (A)
**3** Chargement du contenu de A dans la mémoire de résultats (A) ⇒ (MR)

C'est ce que l'on appelle **3 INSTRUCTIONS**

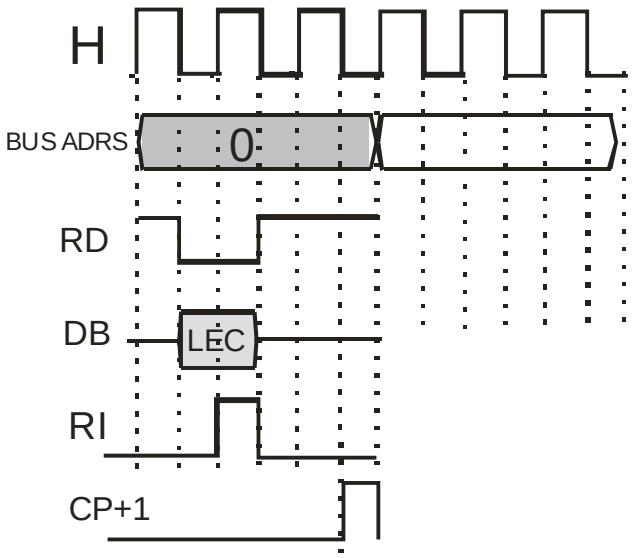
Chacune d'elles est une tache confiée au CPU qui est codée par un mot de programme que nous supposerons de 1 octet. A la mise sous tension du système le CPU doit aller chercher seul le premier des 3 octets de programme . Il est logique qu'il aille le chercher dans la première case de la mémoire .Cette première case d'adresse 0 contiendra donc un mot de programmation codant une lecture d'une case mémoire . Cet octet sera entré dans le registre d'instructions (RI) puis transformé en mot de microprogrammation Le CPU 'sait ' alors qu'il doit effectuer une lecture , encore faut il lui dire à quelle adresse. Cette adresse doit être indiquée quelque part, il est logique de la placer dans la seconde case de la mémoire . Le contenu de la mémoire associée au CPU est donc ( figure ci contre )




Essayons maintenant de préciser les signaux gérés par le CPU .

Le système est piloté par une horloge, tous les signaux sont synchrone de cette horloge. La figure ci-dessous représente un chronogramme possible.

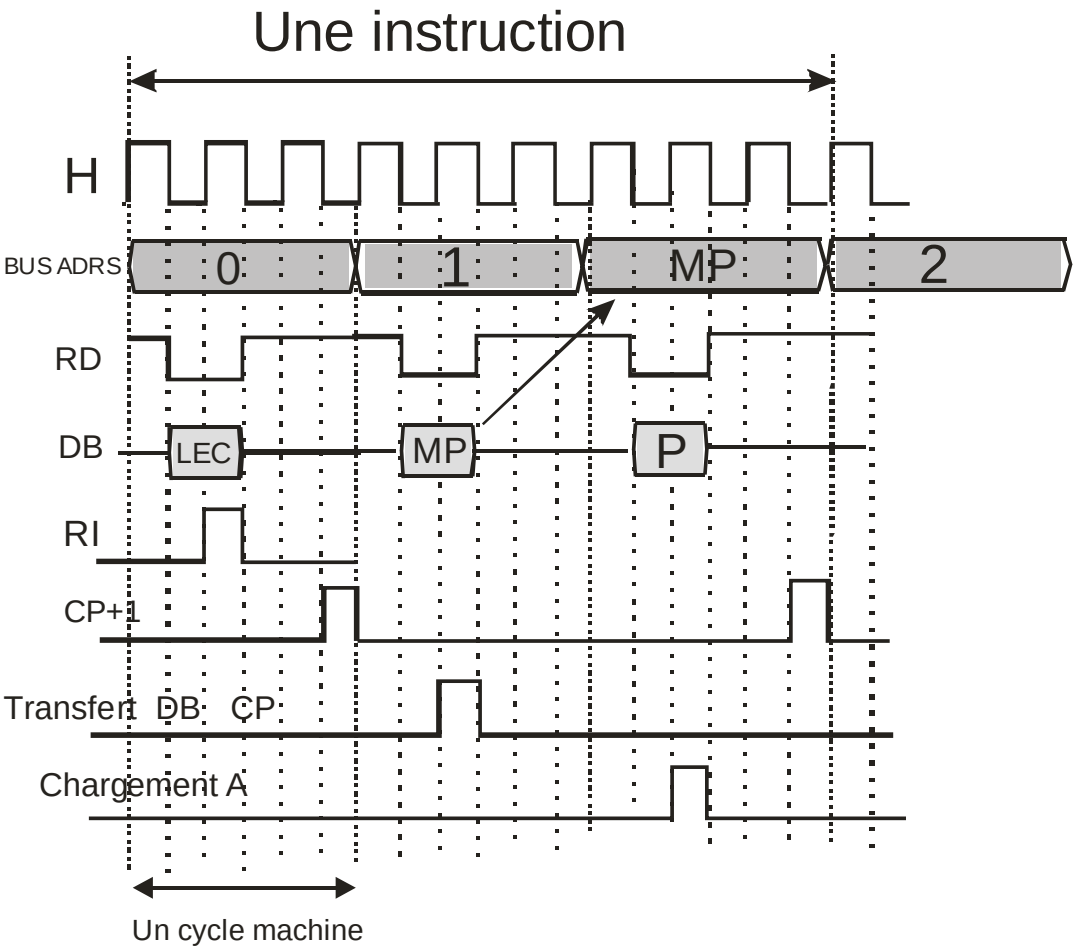
Le CPU présente sur le bus d'adresses , la première adresse soit 0 , il envoie alors sur le bus de contrôle un signal de lecture à la mémoire , cette dernière répond en plaçant sur le bus de données l'octet qu'elle contient c'est-à-dire l'octet de programmation codant une lecture , donnons lui un nom provisoire LEC .Le CPU génère alors une impulsion qui mémorise cet octet ( latche ) dans le registre d'instructions (RI) . Une période d'horloge est laissée au décodeur interne pour fabriquer le mot de microprogrammation . Sur le bus d'adresse la première adresse est restée stable pendant ces 3 périodes d'horloge . Cette durée est appelée un **cycle machine** .




L'instruction étant préparée par le CPU ce dernier 'sait ' qu'il doit aller lire la case mémoire suivante , il incrémente donc son compteur programme .

L'adresse 1 étant présentée sur le bus d'adresses , le CPU doit envoyer de nouveau un signal de lecture RD sur le bus de contrôle , lire l'octet qui est l'adresse MP du premier nombre , et le transférer sur le bus adresses pour lire ensuite la case MP .Ceci a lieu lors d'un troisième cycle machine ou le CPU génère encore un top de lecture , puis charge P dans l'accumulateur. Notons que cela suppose une liaison directe entre le compteur programme et le bus de données .

La première instruction est terminée, elle a duré 3 cycles machine .Le CPU doit chercher l'instruction suivante codée par un octet qu'il est logique de trouver à la case mémoire suivante soit 2 .D'ou un nouveau signal d'incrémentation de CP .




Ainsi une instruction peut être constituée de plusieurs cycles machine, ces derniers n'ayant pas nécessairement la même durée c'est-à-dire le même nombre de périodes d'horloge , contrairement à ce qui est proposé sur la figure qui n'est qu'un exemple possible .

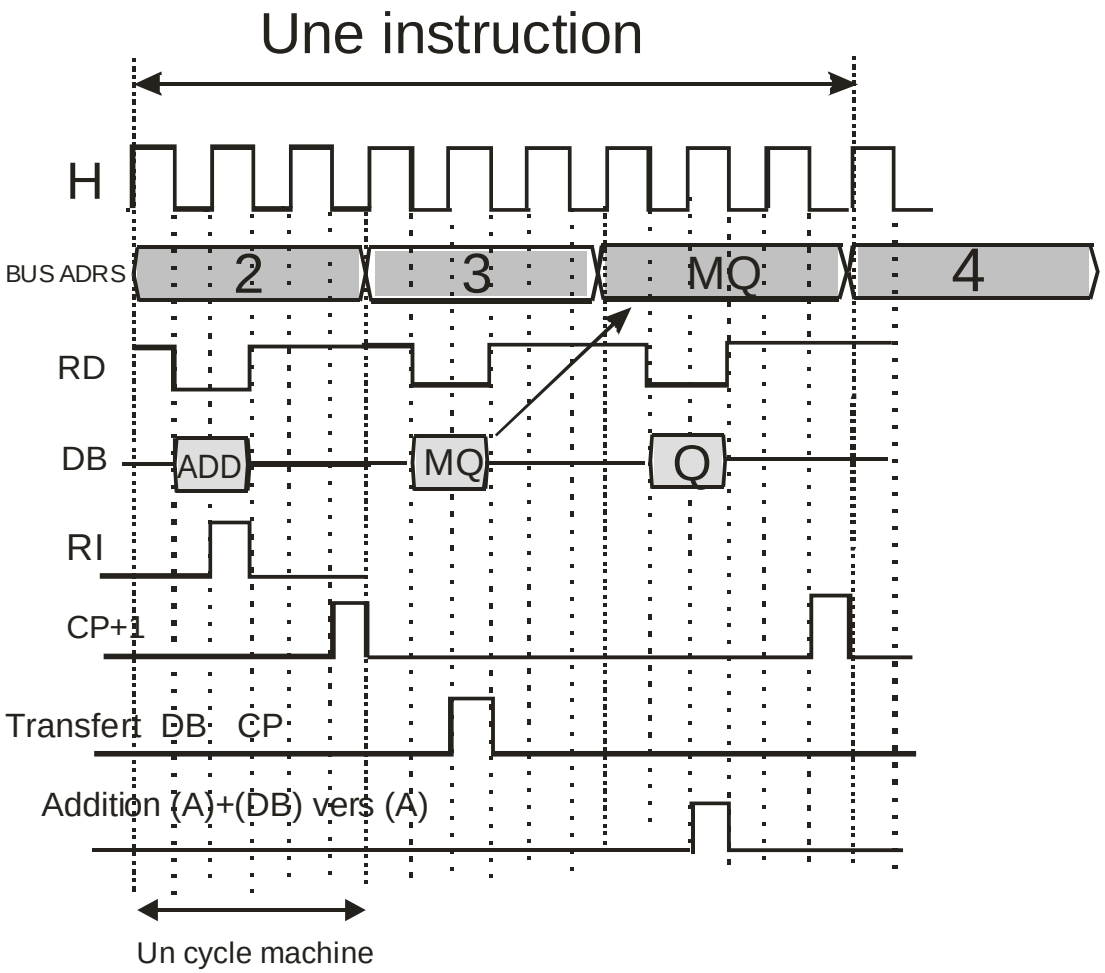
Les signaux sont très différents d'un microprocesseur à un autre .ainsi un 8080 utilise 9 périodes d'horloge par cycle machine alors que le 6800 n'en a besoin que d'une. Le 8080 piloté par un quartz à 18Mhz n'est pas plus rapide que le 6800 qui n'est piloté qu'a 1MHz .

La seconde instruction s'exécute comme la première :
2 sur le bus adresse pour lecture du mot code définissant une addition ADD
- Top lecture mémoire RD
- Transfert dans RI pour décodage
- Incrémentation CP

3 sur le bus adresse pour accéder à la mémoire ou est stocké Q
- Top de lecture RD
- Lecture de l'adresse MQ
- Transfert de MQ sur le bus adresses

MQ sur le bus adresses
- Top de lecture RD
- Lecture du nombre Q
- Addition du contenu de A soit P avec le nombre lu sur DB transfert automatique du résultat dans A

Les signaux correspondants sur les BUS peuvent être les suivants .




Enfin pour la derniére:

4 sur le bus d'adresses
top de lecture
transfert dans RI
incrémentation CP
5 sur le bus d'adresses
top lecture
lecture adresse de la case de résultat MR
transfert de cette adresse sur le bus adresses
signal d'écriture pour transférer le contenu de A dans la mémoire
incrémentation ce CP

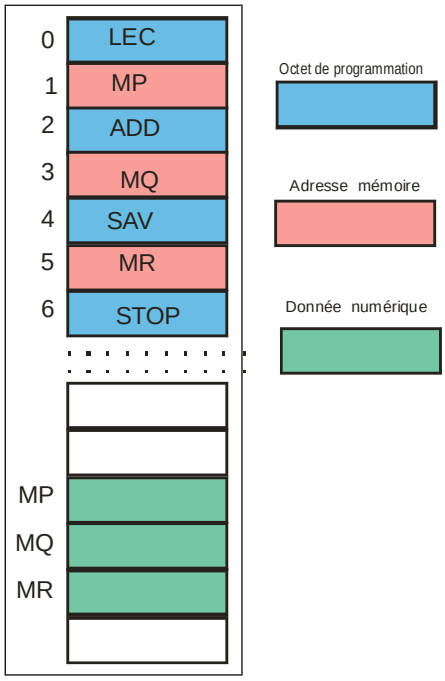
Notons:

Le premier cycle machine de chaque instruction envoie un octet de programme dans le registre d'instructions, c'est au cours de ce cycle que le CPU ' apprend ' ce qu'il doit faire ensuite. Ce cycle initial est appelé **Fetch cycle** dans la littérature anglo saxonne.

Le dernier cycle machine d'une instruction se termine par une incrémentation du compteur programme .

Les chronogrammes précédents ne sont qu'une possibilité pour chaque type de circuit il faut se plonger dans la documentation du constructeur. Le plus souvent l'utilisateur n'a pas besoin de connaître ces détails, ils ne sont utiles que lorsque rien ne fonctionne on veut ausculter le circuit avec une sonde d'oscilloscope .

Examinons maintenant le contenu de la mémoire, 0 2 4 6 contiennent des octets de programmation, les autres des adresses ou des données. Rien ne distingue mot de programme adresse ou donnée si ce n'est leur position relative qui est connue par le CPU.




**Remarque** : *Nous avons supposé que l'adresse était constituée d'un seul octet , il n'y a alors que 256 cases possibles , en réalité les microprocesseurs 8 bits disposent d'adresses sur 16 bits qui exigent 2 octets mais le principe reste le même .*

## Les constituants secondaires du CPU

La structure décrite est minimale , il est commode d'ajouter au CPU des composants supplémentaires qui faciliteront grandement son utilisation .

Des **registres d'état** pour stocker par exemple la retenu lors d'une opération

Des registres pour stocker les adresses de retour lors de l'appel à un sous programme .C'est le pointeur de pile (**Stack Pointer** ) qui fera l'objet d'un paragraphe .

Des compteurs supplémentaires ou **TIMERS** pour faciliter la réalisation de retards

**Des mémoires vives** permettant de stocker plus rapidement que dans un boîtier extérieur des résultats de calcul ou des constantes .

### CPU à ROM interne

Les possibilités d'intégration actuelles permettent de placer dans le CPU de très nombreux registres ou mémoires annexes. De nombreux boîtiers possèdent ainsi une mémoire morte intégrée de plusieurs kilo octets pour y charger le programme .Si cette mémoire de programme est une ROM , le boîtier ne peut être programmé qu'a la fabrication. Ce sont le plus souvent des mémoires du type PROM à MOS comme décrit plus haut. Le programme ayant été mis au point il faut ' bruler ' la PROM interne comme on le fait pour une 2764. Chaque type de circuit possède sa procédure de programmation propre .Plus souples d'emploi sont les boîtiers dont la PROM interne de programme est une **FLASH** , c'est le cas du PIC16F84.La programmation et l'effacement se fait alors très rapidement .

Pour d'autres familles ayant une ROM interne plus difficilement programmable il existe des boîtiers sans mémoire interne ( **ROMLESS** ) , pour la mise au point, le programme est alors introduit dans une mémoire extérieure .

### Les Timers

Ce sont des compteurs.Il est possible de les charger à une valeur initiale par une instruction , ensuite ils sont décrémentés par l'horloge du système,lorsque le compteur est vide un signal ( ou une interruption ) est envoyé permettant à l'utilisateur de déclencher une action . Il est aussi possible de réaliser des retards par programme mais dans ce cas le CPU travaille pendant ce retard et on ne peut pas lui confier une autre tache.

### Le chien de garde ( Watch Dog )

C'est un système permettant d'éviter les arrêts intempestifs du CPU .Si une impulsion parasite fait dérailler le système d'acquisition des octets en mémoire , le CPU peut confondre donnée adresse et mot code , il se met alors à faire n'importe quoi.

Le principe est le suivant: le programmeur prévoit que le programme s'il se déroule normalement génère régulièrement une impulsion qui décharge un condensateur. Si le programme 'se plante ' ces tops n'existent plus , le condensateur se charge et lorsque le niveau de tension à ses bornes atteint un certain seuil , un RESET du CPU est déclenché ..

## Association CPU mémoire

Pour permettre l'accès à une mémoire extérieure le CPU doit :
Présenter une adresse sur le bus d'adresse
Envoyer les signaux convenables pour effectuer une lecture ou écriture

La plupart des microprocesseurs 8 bits ont un bus d'adresse de 16 fils , ils peuvent donc piloter une mémoire de $2^{16}$=65536 mots , mais il est rare qu'une capacité aussi grande soit utilisée.
Les signaux de commande fournis peuvent être différents d'un circuit à un autre .Le 8080 , Z80 et de façon générale les microprocesseurs de la famille INTEL délivrent deux signaux distinct d'écriture et lecture actifs au niveau bas /RD et /WR

Nous avons vu qu'à la mise sous tension un CPU (de la famille INTEL au moins ) commence à lire le programme à partir de la case 0 .Cette case doit donc se trouver dans une ROM de programme qui conserve son contenu en absence d'alimentation. Dans cette ROM on ne peut pas stocker de données , il faut donc utiliser une RAM . Ces deux composants ROM et RAM doivent se partager les 64k disponibles .
Certains composants , comme les horloges , convertisseurs analogique digital,etc.. se comportent aussi comme des mémoires (de faible capacité ) ils devront eux aussi se positionner dans les 64k ,que l'on appelle le **champ mémoire** du CPU

Supposons que nous voulions utiliser seulement 2k de mémoire , 1k de ROM programme et 1k de RAM pour charger des données ou des résultats.

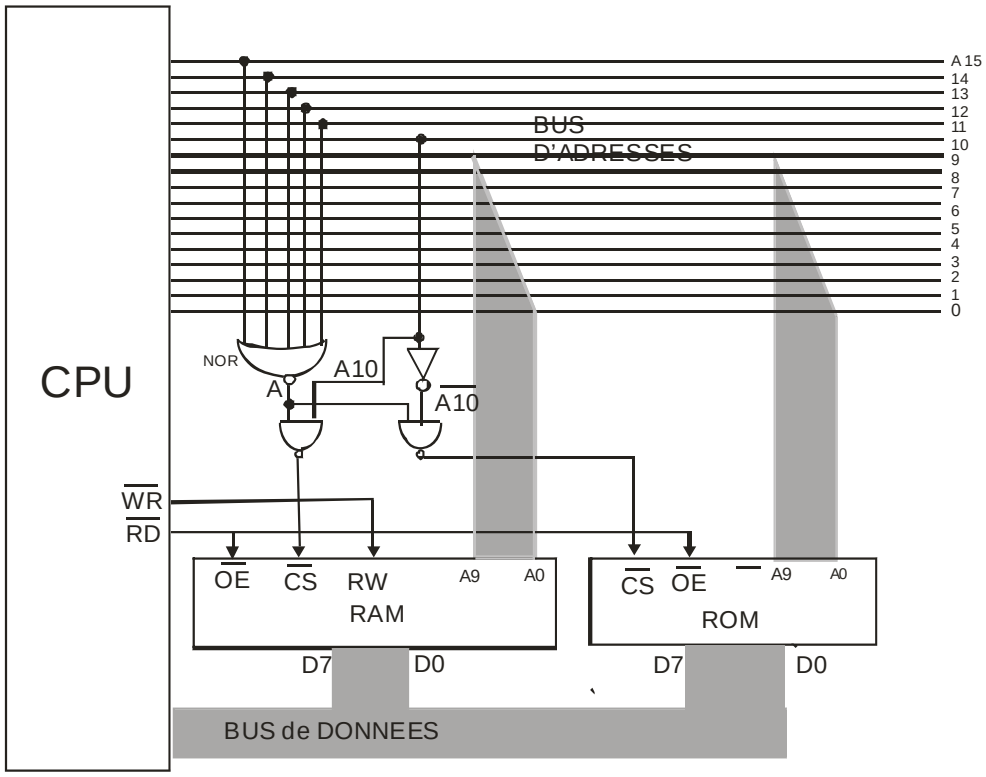
Un boîtier de 1k comprend 10 fils d'adresse , et, si c'est une ROM , 2 fils de commande /CS et ./OE
Un boîtier de RAM de même capacité possède également 10 fils d'adresse mais 3 entrées de commande /CS /OE et R/W .
Les cases mémoire définies par le CPU ont des adresses sur 16 bits qui vont de 0000.0000.0000.0000 ( 0000 en notation hexadécimale ) à 1111.1111.1111.1111 ( FFFF) .

Imposons nous d'utiliser pour loger la ROM les 1024 premiers emplacements ,c'est-à-dire les case d'adresses (0000.0000.0000.0000 à 0000.0011.1111.1111 ( 0000 à 03FF) et pour la RAM les 1024 cases suivantes ( 0000.0100.0000.0000 à 0000.0111.1111.1111 soit en hexa 0400 à 07FF °

Examinons l'état des 16 fils du bus d'adresse lorsque le CPU accède à ces cases .

| Fils du bus d'adresses | A 15 | A 14 | A 13 | A 12 | 1 11 | A 10 | A 9 | A 8 | A 7 | A 6 | A 5 | A 4 | A 3 | A 2 | A 1 | A 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Première case ROM | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Dernière case ROM | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Première case RAM | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Dernière case RAM | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Nous remarquons que toutes les combinaisons possibles des états 1 et 0 des lignes A9 à A0 se rencontrent aussi bien dans les adresse ROM que RAM , les fils A15 à A11 sont tous à 0 mais A10 à pour état 0 dans la ROM et 1 dans la RAM . C'est cette différence que nous allons exploiter pour diriger les signaux de commande vers l'un ou l'autre des boîtiers.
Le câblage peut alors être le suivant :




Les sorties ou entrées de données des deux boîtiers sont en parallèle , aucun conflit n'est à craindre car un seul des deux boîtiers sera sélectionné .

Les bornes d'accès adresse A9 A0 (10 fils par boîtier ) sont également câblés en parallèle.

Si le CPU veut lire dans la ROM il applique une adresse comprise entre 0000 et 03FF , les 5 premiers fils A15-A11 sont au 0 , le NOR à 5 entrée fournit donc en A un niveau 1 , A10 est au 0 et /A10 au 1 , seul le signal de sélection de la ROM est au niveau bas, la RAN non sélectionnée ne charge pas le bus de données. Lors de la lecture par le CPU le signal /RD actif au niveau bas appliqué à l'entrée /OE du boîtier ouvre la ROM en lecture.

Si le CPU veut effectuer une opération dans la RAM il met A10 à 1 , cette fois c'est le /CS de la RAM qui est actif. Pour une lecture /RD est au niveau bas , donc /OE alors que /WR est au niveau de repos haut. La RAM avec /CS=/OE=0 R/W=1 est bien en lecture.
Pour une écriture /OE est au 1 car /RD est au repos et R/W au 0 , c'est bien une configuration d'écriture.

En utilisant une autre porte et à 5 entrées sur les 5 premiers fils mais en inversant préalablement un ou plusieurs d'entres eux il est possible de rajouter des boîtiers mémoire aux adresses restées libres de 07FF à FFFF .

Si aucune autre mémoire n'est envisagée on peut supprimer le NOR .Les 5 premiers bits d'adresse sont ignorés , dans ce cas toute case mémoire des 2 boîtiers précédents possède dans le champ mémoire du CPU 32 adresses différentes équivalentes correspondant aux 32 combinaisons possibles des 5 premiers bits d'adresse. , on dit que ce sont les **adresses fantômes** .

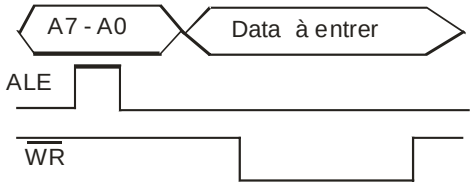
Très souvent on choisit de placer la ROM à partir de 0 comme précédemment mais la RAM à partir de l'adresse 8000 , dans ce cas le seul fil A15 suffit à discriminer RAM ou ROM .

On trouve souvent dans les cartes microprocesseur le décodeur 3 $\Rightarrow$ 8 74138 , recevant les 3 premiers fils du bus d'adresse A51 14 13 il découpe le champ mémoire en 8 zones de 8K octets .

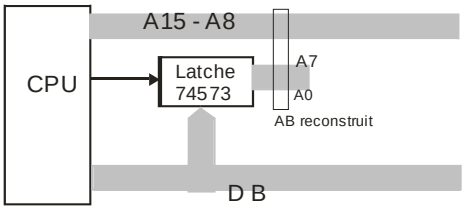
## Les BUS multiplexés

Pour disposer de 16 fils d'adresse il faut bloquer 16 broches du circuit. Pour économiser ces accès il est fait souvent appel à la technique du **multiplexage**. C'est le cas du 8031 , 8 fils d'adresse A15 – A8 sont disponibles directement sur le boîtier. Au début de chaque cycle machine les 8 bits de plus faible poids A7-A0 sont sortis sur le bus de données en même temps qu'un signal de

validation qui permet à un latche de les prélever au vol. Ce n'est qu'ensuite que sont envoyés les signaux classiques d'écriture ou lecture . Les 16 fils d'adresse sont ainsi reconstitués au début de chaque cycle machine .




## Le STACK POINTER et les appels sous programme

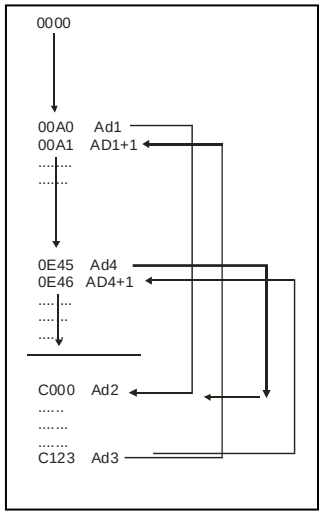



Le pointeur de pile (STACK POINTER) est un constituant majeur du CPU . Très souvent il est nécessaire d'utiliser des sous programmes pour simplifier la structure d'un programme . Examinons l'exemple suivant .

Le CPU exécute un programme à partir de l'adresse 0 jusqu'à l'adresse AD1 ( par exemple 00A0 ) , à cet endroit il est fait appel à un sous programme placé entre AD2( C000) et AD3 ( C123) , ce SP terminé le CPU reprend son exécution à partir de AD1+1( 00A1) , en AD4 ( OE45) il rencontre de nouveau un appel au même SP , et enfin termine en repartant de AD4+1 .( OE46)

Ce n'est pas un saut classique car l'adresse de retour change à chaque fois. AD1+1 puis AD4+1 . Avant chaque saut cette adresse de retour doit être sauvée , c'est le rôle du STACK POINTER .

Dans certains circuits peu performants (ou des calculatrices de poche ) les adresses de retour sont stockées dans une mémoire réservée à cet effet . Dans ce cas la taille de cette mémoire limite le nombre se sous programmes qu'il est possible d'imbriquer . La solution retenue ici est de stocker ces adresses de retour dans la même mémoire que les autres données , mémoire interne ou externe au CPU .Le Stack est alors un pointeur (au sens des informaticiens ) qui indique à quel endroit s'effectue de stockage temporaire .




Pour l'exemple nous allons supposer qu'au départ du programme le registre **SP** (Stack Pointer ) a été chargé avec l'adresse 8000 ( il faut bien sûr que le CPU possède de la RAM à cette adresse . )

Examinons alors ce qui se passe dans la mémoire au voisinage de cette adresse lors d'un saut vers un sous programme .

Initialement l'adresse 8000 contient n'importe quoi , pas forcément zéro. Lors du premier saut les deux octets définissant l'adresse de retour sont stockés en 8000 et 8001 alors que le contenu du SP augmente de 2 et passe à 8002 .

Adresse de retour dans (SP) et (SP+1) puis (SP)+2 $\Rightarrow$(SP)

Lorsque dans le sous programme le CPU rencontre l'instruction de retour , il vient lire le contenu des adresse SP-1 et SP-2 et bascule leur contenu dans son compteur programme ce qui permet de continuer l'exécution au point ou elle avait été interrompue., puis il restitue à SP son contenu initial :

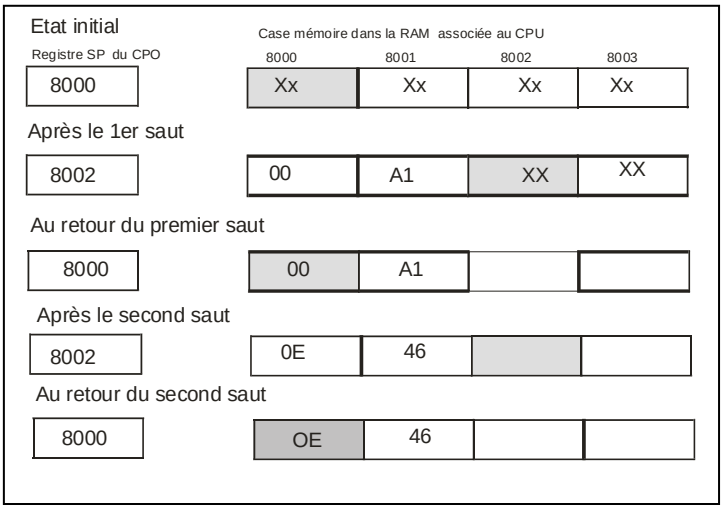
Nouvelle adresse CP (SP-1) et (SP-2) puis (SP)-2 $\Rightarrow$ (SP)

Même mécanisme pour le second saut .La figure ci contre illustre ce mécanisme .( sur la figure les cases en gris sont celles qui sont pointées par le Stack Pointer )

Remarquons que les adresses de retour ne sont pas effacées dans la mémoire , elles sont seulement modifiées au saut suivant .

Si plusieurs sous programmes sont imbriqués les adresses de retour successives sont écrites plus loin 8002 9003 etc …

Le programmeur n'a pas besoin de toucher à la pile , sa gestion est automatique , mais en la modifiant il peut faire des choses subtiles , mais dangereuses.



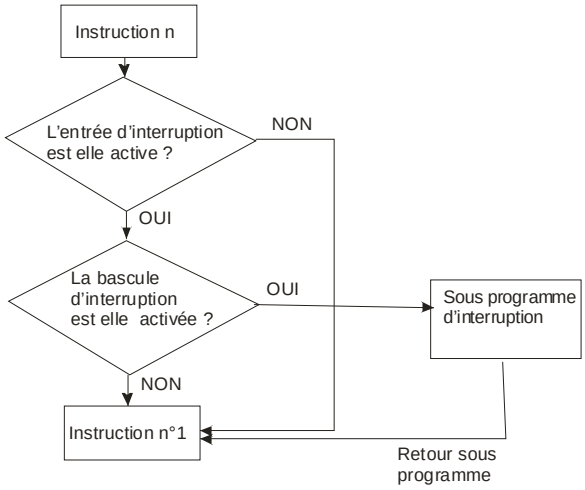

Pour les différents types de microprocesseur le mécanisme réel peut être légèrement différent de celui qui a été décrit mais le principe demeure .

## Les interruptions

Ce sont des sauts à sous programme provoqués par des évènements extérieurs. Un CPU possède en général une broche (ou plusieurs ) spéciale l'entrée d'interruption , une bascule interne , la bascule B d'interruption . Le fonctionnement de base est décrit par la figure ci contre .

Le processus d'aiguillage est naturellement câblé dans le circuit CPU et ne ralentit en aucune façon le déroulement du programme .

Lorsque plusieurs entrées d'interruption sont prévues il est possible d'en autoriser certaines et d'en interdire d'autres grâce à un masque d'interruptions.




### Les microcontroleurs

Pour piloter un système complet le CPU doit être associé à des mémoires vives pour stocker les résultats intermédiaires, morte pour le programme mais il faut aussi des circuits d'interface avec l'extérieur ou capables d'effectuer certaines taches ; horloges temps réel ou convertisseurs analogiques numériques etc…

Cet ensemble est réuni sur une carte de circuit imprimé qui peut avoir une taille importante.

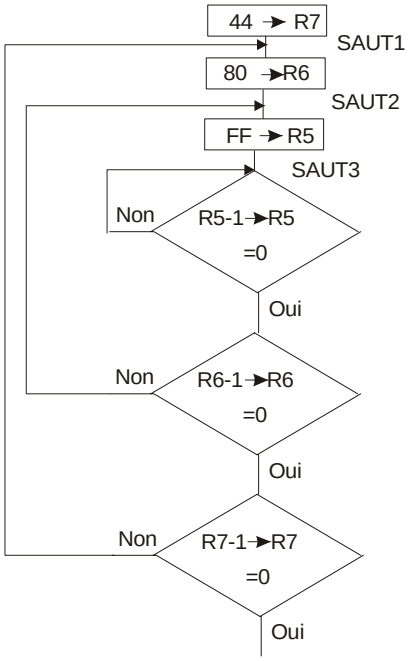
Les possibilités d'intégration actuelles permettent de loger tout cela sur une seule puce ,c'est ce que l'on appelle un **microcontroleur** .

## L'ASPECT LOGICIEL : L'ASSEMBLEUR

Le microprocesseur fonctionne grâce à une suite de mots de programme stockés en ROM Le travail du programmeur est donc de construire cette suite de mots,( d'octets pour un microprocesseur 8 bits de structure Von Neumann ).Pour cela il doit connaître les différentes instructions exécutables par le CPU dont il dispose et des codes machine correspondants.

A l'époque préhistorique c'est à dire avant 1980 on ne possédait rien de plus et la construction du programme était souvent une tache laborieuse.

Soit par exemple à construire une boucle de retard pour un microprocesseur 8051 , le premier mot code se trouvant dans la ROM à l'adresse 1000 (hexadécimal ) .Le programme peut être le suivant : ( les valeurs 44 80 FF ont été choisies pour obtenir une durée précise )




1 Charger 44H dans le registre interne R7
2 Charger 80H dans le registre interne R6
3 Charger FFH dans le registre interne R5
4 Décrémenter R5 , si non nul recommencer ,sinon ligne suivante
5 Décrémenter R6 , si non nul sauter à la ligne 3 sinon ligne suivante
6 Décrémenter R7, si non nul sauter à la ligne 2 , sinon ligne suivante
7 Sortie

Parmi les instructions du 8051 nous utiliserons :

- Chargement d'un nombre dans un registre interne Rx .Cette instruction se code sur 2 octets 7F suivi du nombre à charger ici 7F et 44 pour la première ligne

7F et 80 pour la seconde

7F et FF pour la ligne 3

Décrémentation et saut si résultat nul , sinon exécution de l'instruction suivante

Le code prend 2 octets , le second représente la longueur du saut ( en complément à 2 , maximum ±127 , sur un octet ), le premier octet est DF pour R7 DE pour R6 DD pour R4

A ce niveau nous pouvons commencer à écrire le contenu de la mémoire (à partir de l'adresse 1000 valeur imposée )

| Adresse | Octet | Commentaire |
|---|---|---|
| 1000 | 7FH | Ligne 1 |
| 1001 | 44H | |
| 1002 | 7FH | Ligne 2 |
| 1003 | 80H | |
| 1004 | 7FH | Ligne 3 |
| 1005 | FFH | |
| 1006 | DFH | Ligne 4 |
| 1007 | xx | xx définit le saut. Le CPU doit sauter de 1008 à 1006 Soit un saut de –2 , en complément à 2 FEH |
| 1008 | DE | Ligne 5 |
| 1009 | Yy | saut de 1009 à 1004 soit –5 codé en hexa FBH |
| 1010 | DD | Ligne 6 |
| 1011 | Zz | Saut vers 1002 soit -9 codé F7H |

Il faut à chaque fois connaître le code machine de l'instruction et la position du code dans la mémoire pour calculer les sauts. Si au cours du travail vous avez oublié une instruction , c'est

la catastrophe , pour l'intercaler il faut recalculer les valeurs de tous les sauts. Lorsque le programme dépasse quelques lignes il est presque impossible d'éviter les erreurs.

Pour faciliter le travail chaque constructeur de CPU fourni un outil logiciel adapté , **l'assembleur** .Ce logiciel :

Evite à l'utilisateur de manipuler directement les codes machine. Chaque instruction porte un nom son **Mnémonique** , 3 à 5 lettres

Evite le calcul fastidieux des sauts , une position mémoire peut être désignée par un nom

Autorise l'ajout de commentaires très utiles pour expliquer le rôle de chaque instruction lors d'une relecture ultérieure .

Cet outil travaille à partir d'un **fichier texte** écrit par l'utilisateur sur un ordinateur quelconque muni d'un éditeur de texte .C'est le **fichier source**

Pour l'exemple précédent ce fichier source est : :

```
ORG 1000        ;définit la première case en mémoire .C'est un commentaire précédé d'un point virgule ; Cette
                commande n'est pas à proprement parler une instruction mais une directive d'assemblage .
MOV R7,#44H     ;MOV est le mnémonique désignant l'instruction , il est suivi de R7 indiquant quel est
                le registre interne visé .Le # indique que ce qui suit est un nombre , ici en hexadécimal ( le H)
SAUT1 :MOV R6,#80H  ; le mot SAUT1 suivi de : indique une position mémoire vers laquelle devra
                se faire un saut
SAUT2 : MOV R5,#255 ;en absence du H le nombre est accepté en décimal , l'assembleur peut
                même l'accepter en binaire 11111111B
SAUT3 :DJNZ R7,SAUT3 ;DJNZ est le mnémonique de l'instruction ,
DJNZ R6,SAUT2
DJNZ R5,SAUT1
```

Sur chaque ligne ce qui suit le point virgule est un commentaire destiné uniquement au programmeur

Ce fichier texte est ensuite introduit dans le programme assembleur qui construit plusieurs fichiers contenant les octets codes .

Par exemple le fichier .LST ressemble à ceci :

*ORG 1000*

| | | | |
|---|---|---|---|
| *1000* | | *MOV R7,#44H* | *7F 44* |
| *1002* | *SAUT1 :* | *MOV R6,#80H* | *7F 80* |
| *1004* | *SAUT2 :* | *MOV R5,#FFH* | *7F FF* |
| *1006* | *SAUT3 :* | *DJNZ R5,SAUT3* | *DD FE* |
| *1008* | | *DJNZ R6,SAUT2* | *DE FB* |
| *100A* | | *DJNZ R7,SAUT1* | *DF F7* |

Notez que chaque instruction est notée sur une ligne, la colonne de gauche représente les adresses ( en hexadécimal) , la colonne suivante les noms des positions mémoires , ensuite le mnémonique de l'instruction et le nombre associé éventuel et enfin les octets codes.

L'assembleur fournit aussi un fichier ne contenant que les octets codes sans commentaire

*7F 44 7F 80 7F FF DD FE DE FB DF F7*

C'est ce fichier qui sera exploité par le programmateur chargé de placer les codes dans la ROM de programme .

Les assembleurs peuvent faire aussi des choses bien plus compliquées , chacun d'entre eux possède un format particulier que le programmeur doit connaître mais c'est un outil pratiquement indispensable .

Il existe aussi des outils encore plus puissants , permettant d'écrire des programmes en utilisant les langages évolués PASCAL BASIC C etc … ce sont les **compilateurs** dont nous ne parlerons pas ici .