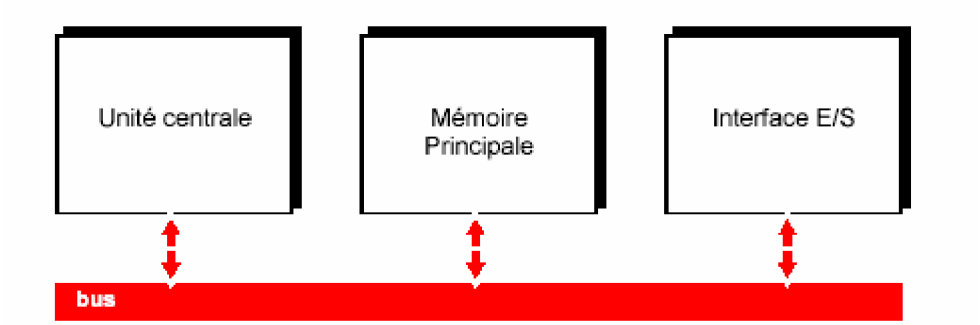
Architecture *de von Neumann*

Une unité centrale
. Une mémoire principale
. Des interfaces d'entrées/sorties
Les différents organes du système sont reliés par des voies de communication appelées **bus**.



I/- L'unité centrale

Elle est composée par le microprocesseur qui est chargé d'interpréter et d'exécuter les instructions d'un programme, de lire ou de sauvegarder les résultats dans la mémoire et de communiquer avec les unités d'échange. Toutes les activités du microprocesseur sont cadencées par On caractérise le microprocesseur par :

- Sa fréquence d'horloge : en MHz ou GHz
- Le nombre d'instructions par secondes qu'il est capable d'exécuter : en MIPS
- La taille des données qu'il est capable de traiter : en bits.

Le microprocesseur

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 exécution. Il doit aussi prendre en compte les informations extérieures au système et assurer leur traitement. C'est le cerveau du système.

A l'heure actuelle, un microprocesseur regroupe sur quelques millimètre carré des fonctionnalités toujours plus complexes. Leur puissance continue de s'accroître et leur encombrement diminue régulièrement respectant toujours, pour le moment, la fameuse *loi de Moore* (1).

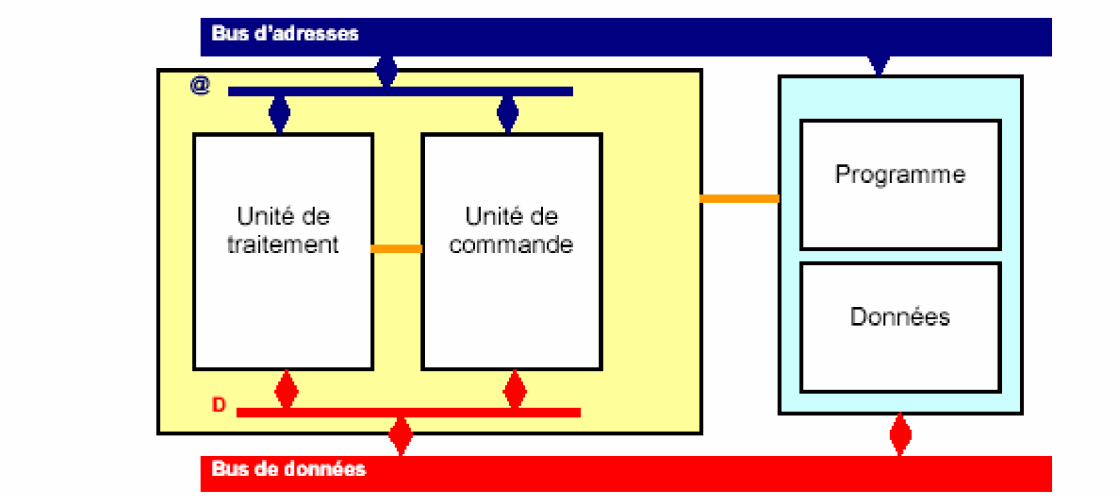
I.1 Architecture de base d'un microprocesseur

Un microprocesseur est construit autour de deux éléments principaux :

- ü Une unité de commande
- ü Une unité de traitement

associés à des registres chargées de stocker les différentes informations à traiter. Ces trois éléments sont reliés entre eux par des bus interne permettant les échanges d'informations.

(1) Moore (un des co-fondateurs de la société Intel) a émis l'hypothèse que les capacités technologiques permettraient de multiplier par 2 tous les 18 mois le nombre de transistors intégrés sur les circuits.



Remarques :
Il existe deux types de registres :

- Ø Les registres d'usage général permettent à l'unité de traitement de manipuler des données à vitesse élevée. Ils sont connectés au bus de données interne au microprocesseur.
- Ø Les registres d'adresses (pointeurs) connectés sur le bus adresses.

I.1.1 L'unité de commande

Elle permet de séquencer le déroulement des instructions. Elle effectue la recherche en mémoire de l'instruction. Comme chaque instruction est codée sous forme binaire, elle en assure le décodage pour enfin réaliser son exécution puis effectue la préparation de l'instruction suivante. Pour cela, elle est composée par :

Le compteur de programme constitué par un registre dont le contenu est initialisé avec l'adresse de la première instruction du programme. Il contient toujours l'adresse de l'instruction à exécuter.

Le registre d'instruction et le décodeur d'instruction : chacune des instructions à exécuter est rangée dans le registre instruction puis est décodée par le décodeur d'instruction.

Bloc logique de commande (ou séquenceur) : Il organise l'exécution des instructions au rythme d'une horloge. Il élabore tous les signaux de synchronisation internes ou externes (bus de commande) du microprocesseur en fonction des divers signaux de commande provenant du décodeur d'instruction ou du registre d'état par exemple. Il s'agit d'un automate réalisé soit de façon câblée (obsolète), soit de façon micro programmée, on parle alors de micro microprocesseur.

I.1.2 L'unité de traitement

C'est le cœur du microprocesseur. Elle regroupe les circuits qui assurent les traitements nécessaires à l'exécution des instructions :

L'Unité Arithmétique et Logique (UAL) est un circuit complexe qui assure les fonctions logiques (ET, OU, Comparaison, Décalage, etc…) ou arithmétique (Addition, soustraction).

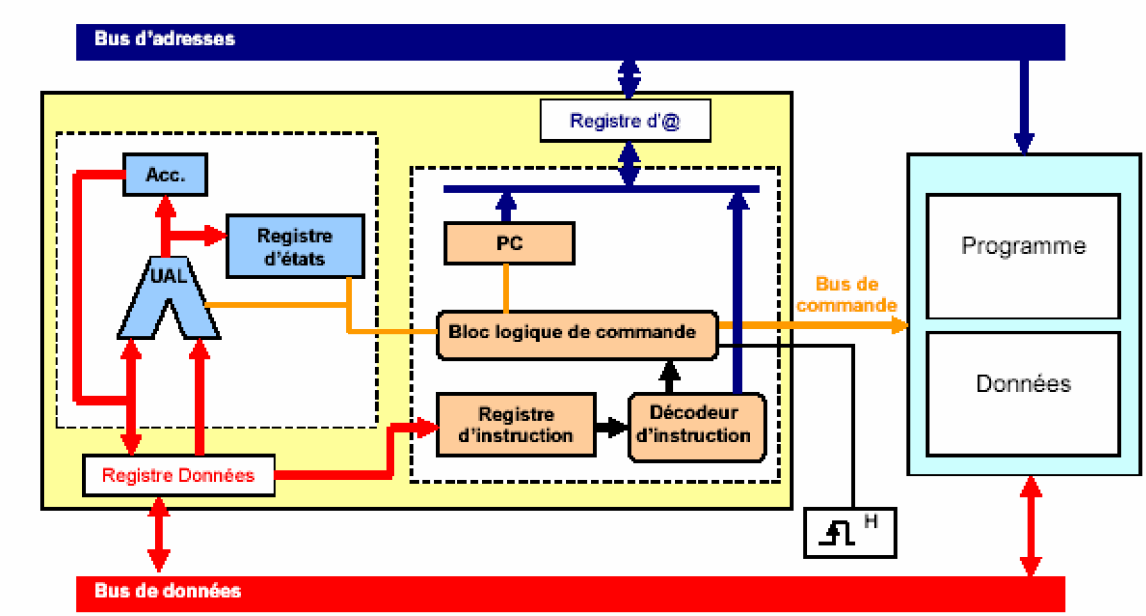
Le registre d'état est généralement composé de 8 bits à considérer individuellement. Chacun de ces bits est un indicateur dont l'état dépend du résultat de la dernière opération effectuée par l'UAL. On les appelle *indicateur d'état* ou *flag* ou *drapeaux*. Dans un

programme le résultat du test de leur état conditionne souvent le déroulement de la suite du programme. On peut citer par exemple les indicateurs de :

- Retenue **(carry : C)**
- Retenue intermédiaire **(Auxiliary-Carry : AC)**
- Signe **(Sign : S)**
- Débordement **(overflow : OV ou V)**
- Zéro **(Z)**
- Parité **(Parity : P)**
-

Les accumulateurs sont des registres de travail qui servent à stocker un opérande au début d'une opération arithmétique et le résultat à la fin de l'opération.

I.1.3 Schéma fonctionnel

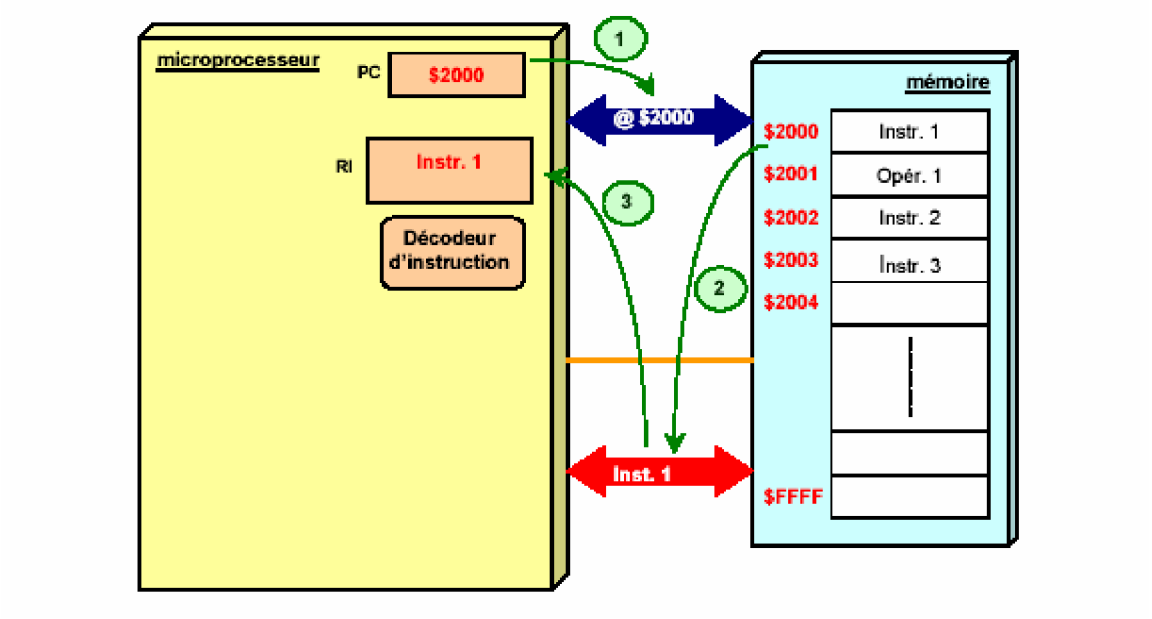


I.2 Cycle d'exécution d'une instruction

Le microprocesseur ne comprend qu'un certain nombre d'instructions qui sont codées en binaire. Le traitement d'une instruction peut être décomposé en trois phases.

Phase 1: *Recherche de l'instruction à traiter*

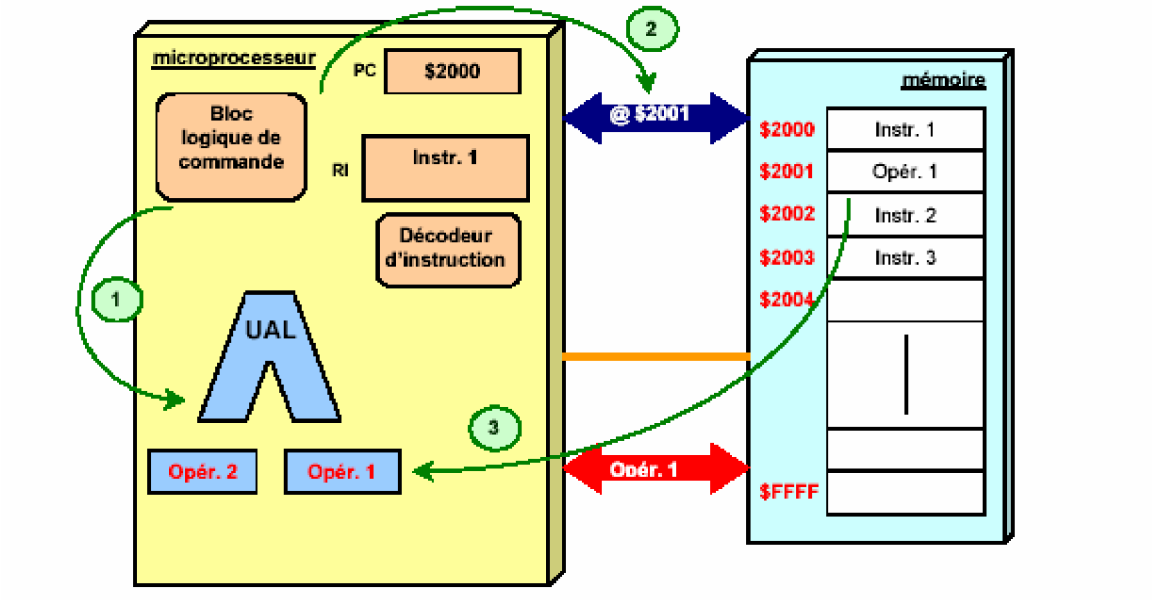
1. Le PC contient l'adresse de l'instruction suivante du programme. Cette valeur est placée sur le bus d'adresses par l'unité de commande qui émet un ordre de lecture.
2. Au bout d'un certain temps (temps d'accès à la mémoire), le contenu de la case mémoire sélectionnée est disponible sur le bus des données.
3. L'instruction est stockée dans le registre instruction du processeur.



Phase 2 : *Décodage de l'instruction et recherche de l'opérande*

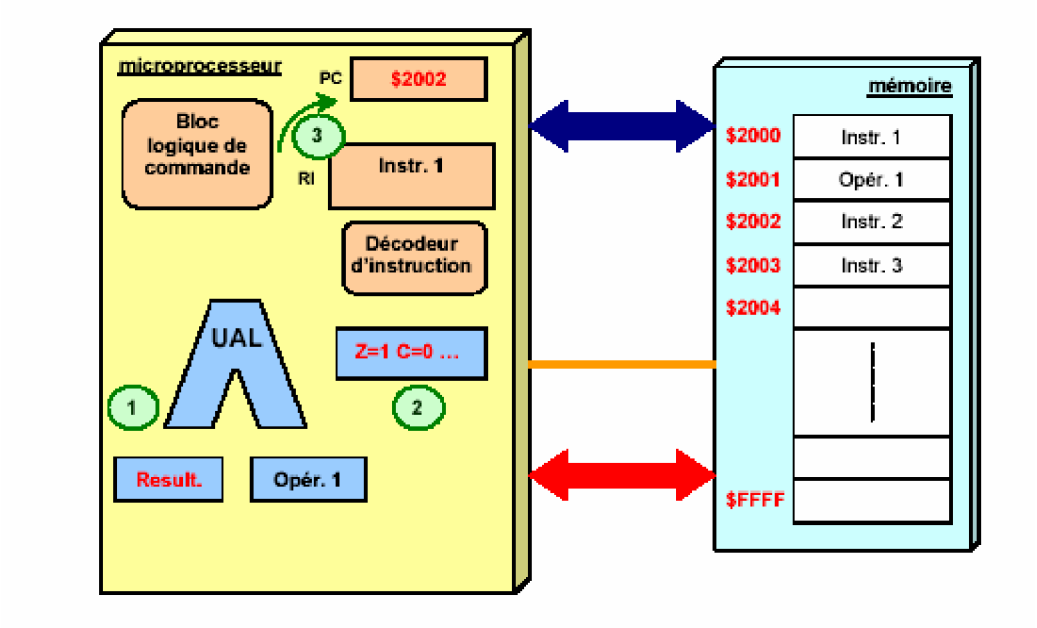
Le registre d'instruction contient maintenant le premier mot de l'instruction qui peut être codée sur plusieurs mots. Ce premier mot contient le code opératoire qui définit la nature de l'opération à effectuer (addition, rotation,...) et le nombre de mots de l'instruction.

1. L'unité de commande transforme l'instruction en une suite de commandes élémentaires nécessaires au traitement de l'instruction.
2. Si l'instruction nécessite une donnée en provenance de la mémoire, l'unité de commande récupère sa valeur sur le bus de données.
3. L'opérande est stocké dans un registre.



Phase 3 : *Exécution de l'instruction*

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



I.3 Jeu d'instructions

I.3.1 Définition

La première étape de la conception d'un microprocesseur est la définition de son jeu d'instructions. Le jeu d'instructions décrit l'ensemble des opérations élémentaires que le microprocesseur pourra exécuter. Il va donc en partie déterminer l'architecture du microprocesseur à réaliser et notamment celle du séquenceur. A un même jeu d'instructions peut correspondre un grand nombre d'implémentations différentes du microprocesseur.

I.3.2 Type d'instructions

Les instructions que l'on retrouve dans chaque microprocesseur peuvent être classées en 4 groupes :

Transfert de données pour charger ou sauver en mémoire, effectuer des transferts de registre à registre, etc…

Opérations arithmétiques : addition, soustraction, division, multiplication

Opérations logiques : ET, OU, NON, NAND, comparaison, test, etc…

Contrôle de séquence : branchement, test, etc…

I.3.3 Codage

Les instructions et leurs opérandes (paramètres) sont stockés en mémoire principale. La taille totale d'une instruction (nombre de bits nécessaires pour la représenter en mémoire) dépend du type d'instruction et aussi du type d'opérande. Chaque instruction est toujours codée sur un nombre entier d'octets afin de faciliter son décodage par le processeur. Une instruction est composée de deux champs :

Le code instruction, qui indique au processeur quelle instruction réaliser.

Le champ opérande qui contient la donnée, ou la référence à une donnée en mémoire (son adresse).

Exemple :

Code instruction	Code opérande
1001 0011	0011 1110

Le nombre d'instructions du jeu d'instructions est directement lié au format du code instruction.

Ainsi un octet permet de distinguer au maximum 256 instructions différentes.

I.3.4 Mode d'adressage

Un mode d'adressage définit la manière dont le microprocesseur va accéder à l'opérande. Les différents modes d'adressage dépendent des microprocesseurs mais on retrouve en général :

L'adressage de registre où l'on traite la donnée contenue dans un registre
L'adressage immédiat où l'on définit immédiatement la valeur de la donnée
L'adressage direct où l'on traite une donnée en mémoire

Selon le mode d'adressage de la donnée, une instruction sera codée par 1 ou plusieurs octets.

I.3.5 Temps d'exécution

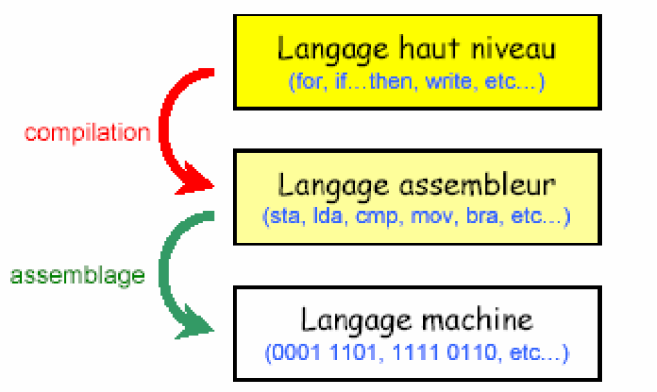
Chaque instruction nécessite un certain nombre de cycles d'horloges pour s'effectuer. Le nombre de cycles dépend de la complexité de l'instruction et aussi du mode d'adressage. Il est plus long d'accéder à la mémoire principale qu'à un registre du processeur. La durée d'un cycle dépend de la fréquence d'horloge du séquenceur.

I.4 Langage de programmation

Le langage **machine** est le langage compris par le microprocesseur. Ce langage est difficile à maîtriser puisque chaque instruction est codée par une séquence propre de bits. Afin de faciliter la tâche du programmeur, on a créé différents langages plus ou moins évolués.

Le langage **assembleur** est le langage le plus « proche » du langage machine. Il est composé par des instructions en général assez rudimentaires que l'on appelle des **mnémoniques**. Ce sont essentiellement des opérations de transfert de données entre les registres et l'extérieur du microprocesseur (mémoire ou périphérique), ou des opérations arithmétiques ou logiques. Chaque instruction représente un code machine différent. Chaque microprocesseur peut posséder un assembleur différent.

La difficulté de mise en œuvre de ce type de langage, et leur forte dépendance avec la machine a nécessité la conception de langages de **haut niveau**, plus adaptés à l'homme, et aux applications qu'il cherchait à développer. Faisant abstraction de toute architecture de machine, ces langages permettent l'expression d'algorithmes sous une forme plus facile à apprendre, et à dominer (C, Pascal,Java, etc…). Chaque instruction en langage de haut niveau correspondra à une succession d'instructions en langage assembleur. Une fois développé, le programme en langage de haut niveau n'est donc pas compréhensible par le microprocesseur. Il faut le **compiler** pour le traduire en assembleur puis l'**assembler** pour le convertir en code machine compréhensible par le microprocesseur. Ces opérations sont réalisées à partir de logiciels spécialisés appelés *compilateur* et *assembleur*.



Exemple de programme :

Code machine (68HC11)	Assembleur (68HC11)	Langage C
@00 C6 64 @01 B6 00 @03 1B @04 5A @05 26 03	LDAB #100 LDAA #0 ret ABA DECB BNE ret	A=0 ; for (i=1 ; i<101 ; i++) A=A+i ;

I.5 Performances d'un microprocesseur

On peut caractériser la puissance d'un microprocesseur par le nombre d'instructions qu'il est capable de traiter par seconde. Pour cela, on définit :

Le **CPI** (Cycle Par Instruction) qui représente le nombre moyen de cycles d'horloge nécessaire pour l'exécution d'une instruction pour un microprocesseur donné.

Le **MIPS** (Millions d'Instructions Par Seconde) qui représente la puissance de traitement du microprocesseur.

$$MIPS = \frac{F_H}{CPI}$$ avec F_H en MHz

II/- La mémoire principale

Elle contient les instructions du ou des programmes en cours d'exécution et les données associées à ce programme. Physiquement, elle se décompose souvent en :

– une mémoire morte (**ROM** = Read Only Memory) chargée de stocker le programme. C'est une mémoire à lecture seule.

– une mémoire vive (**RAM** = Random Access Memory) chargée de stocker les données intermédiaires ou les résultats de calculs. On peut lire ou écrire des données dedans, ces données sont perdues à la mise hors tension.

Les disques durs, disquettes, CDROM, etc… sont des périphériques de stockage et sont considérés comme des mémoires secondaires.

II.1 Organisation d'une mémoire

Une mémoire peut être représentée comme une armoire de rangement constituée de différents tiroirs. Chaque tiroir représente alors une case mémoire qui peut contenir un seul

élément : des **données**. Le nombre de cases mémoires pouvant être très élevé, il est alors nécessaire de pouvoir les identifier par un numéro. Ce numéro est appelé **adresse**. Chaque donnée devient alors accessible grâce à son adresse

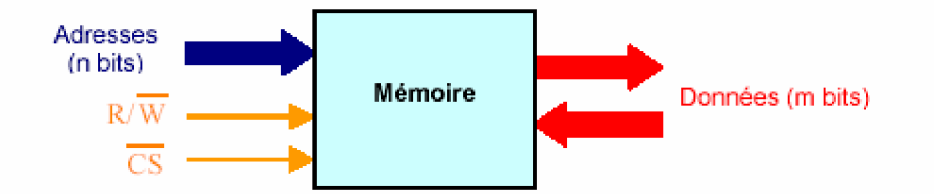
Adresse	Case mémoire
7 = 111	
6 = 110	
5 = 101	
4 = 100	
3 = 011	
2 = 010	
1 = 001	
0 = 000	0001 1010

Avec une adresse de *n* bits il est possible de référencer au plus 2^n cases mémoire.

Chaque case est remplie par un mot de données (sa longueur *m* est toujours une puissance de 2). Le nombre de fils d'adresses d'un boîtier mémoire définit donc le nombre de cases mémoire que comprend le boîtier. Le nombre de fils de données définit la taille des données que l'on peut sauvegarder dans chaque case mémoire.

En plus du bus d'adresses et du bus de données, un boîtier mémoire comprend une entrée de commande qui permet de définir le type d'action que l'on effectue avec la mémoire (lecture/écriture) et une entrée de sélection qui permet de mettre les entrées/sorties du boîtier en haute impédance.

On peut donc schématiser un circuit mémoire par la figure suivante où l'on peut distinguer :



Les entrées d'adresses
Les entrées de données
Les sorties de données
Les entrées de commandes :

- une entrée de sélection de lecture ou d'écriture. (R/W)
- une entrée de sélection du circuit. (CS)

Une opération de lecture ou d'écriture de la mémoire suit toujours le même cycle :

1. Sélection de l'adresse
2. Choix de l'opération à effectuer (R/$\overline{W}$)
3. Sélection de la mémoire ($\overline{CS}$ = 0)
4. Lecture ou écriture la donnée

Remarque :

Les entrées et sorties de données sont très souvent regroupées sur des bornes bidirectionnelles.

II.2- Caractéristiques d'une mémoire

La capacité : c'est le nombre total de bits que contient la mémoire. Elle s'exprime aussi souvent en octet.

Le format des données : c'est le nombre de bits que l'on peut mémoriser par case mémoire. On dit aussi que c'est la largeur du mot mémorisable.

Le temps d'accès : c'est le temps qui s'écoule entre l'instant où a été lancée une opération de lecture/écriture en mémoire et l'instant où la première information est disponible sur le bus de données.

Le temps de cycle : il représente l'intervalle minimum qui doit séparer deux demandes successives de lecture ou d'écriture.

Le débit : c'est le nombre maximum d'informations lues ou écrites par seconde.

Volatilité : elle caractérise la permanence des informations dans la mémoire. L'information stockée est volatile si elle risque d'être altérée par un défaut d'alimentation électrique et non volatile dans le cas contraire.

Remarque :

Les mémoires utilisées pour réaliser la mémoire principale d'un système à microprocesseur sont des mémoires à semi-conducteur. On a vu que dans ce type de mémoire, on accède directement à n'importe quelle information dont on connaît l'adresse et que le temps mis pour obtenir cette information ne dépend pas de l'adresse. On dira que l'accès à une telle mémoire est aléatoire ou direct.

A l'inverse, pour accéder à une information sur bande magnétique, il faut dérouler la bande en repérant tous les enregistrements jusqu'à ce que l'on trouve celui que l'on désire. On dit alors que l'accès à l'information est séquentiel. Le temps d'accès est variable selon la position de l'information recherchée. L'accès peut encore être semi-séquentiel : combinaison des accès direct et séquentiel.

Pour un disque magnétique par exemple l'accès à la piste est direct, puis l'accès au secteur est Séquentiel.

III/- Les interfaces d'entrées/sorties

Elles permettent d'assurer la communication entre le microprocesseur et les périphériques. (Capteur, clavier, moniteur ou afficheur, imprimante, modem, etc…).

III.1- L'interface d'entrée/sortie

III.1.1 Rôle

Chaque périphérique sera relié au système par l'intermédiaire d'une interface (ou contrôleur) dont le rôle est de :

Connecter le périphérique au bus de données.

Gérer les échanges entre le microprocesseur et le périphérique.

III.1.2 Constitution

Pour cela, l'interface est constituée par :

Un **registre de commande** dans lequel le processeur décrit le travail à effectuer (sens de transfert, mode de transfert).

Un ou plusieurs **registres de données** qui contiennent les mots à échanger entre le périphérique et la mémoire

Un **registre d'état** qui indique si l'unité d'échange est prête, si l'échange s'est bien déroulé, etc…

On accède aux données de l'interface par le biais d'un espace d'adresses d'entrées/sorties.

III.2 Techniques d'échange de données

Avant d'envoyer ou de recevoir des informations, le microprocesseur doit connaître l'état du périphérique. En effet, le microprocesseur doit savoir si un périphérique est prêt à recevoir ou à transmettre une information pour que la transmission se fasse correctement. Il existe 2 modes d'échange d'information :

Le mode programmé par **scrutation** ou **interruption** où le microprocesseur sert d'intermédiaire entre la mémoire et le périphérique. Le mode en accès direct à la mémoire (**DMA**) où le microprocesseur ne se charge pas de l'échange de données.

III.2.1 Echange programmé

III.2.1.1 Scrutation

Dans la version la plus rudimentaire, le microprocesseur interroge l'interface pour savoir si des transferts sont prêts. Tant que des transferts ne sont pas prêts, le microprocesseur attend.

L'inconvénient majeur est que le microprocesseur se retrouve souvent en phase d'attente. Il est complètement occupé par l'interface d'entrée/sortie. De plus, l'initiative de l'échange de données est dépendante du programme exécuté par le microprocesseur. Il peut donc arriver que des requêtes d'échange ne soient pas traitées immédiatement car le microprocesseur ne se trouve pas encore dans la boucle de scrutation.

Ce type d'échange est très lent.

III.2.1.2 Interruption

Une interruption est un signal, généralement asynchrone au programme en cours, pouvant être émis par tout dispositif externe au microprocesseur. Le microprocesseur possède une ou plusieurs entrées réservées à cet effet. Sous réserve de certaines conditions, elle peut interrompre le travail courant du microprocesseur pour forcer l'exécution d'un programme traitant la cause de l'interruption.

Dans un échange de données par interruption, le microprocesseur exécute donc son programme principal jusqu'à ce qu'il reçoive un signal sur sa ligne de requête d'interruption. Il se charge alors d'effectuer le transfert de données entre l'interface et la mémoire.

Principe de fonctionnement d'une interruption :

Avant chaque exécution d'instructions, le microprocesseur examine si il y a eu une **requête** sur sa ligne d'interruption. Si c'est le cas, il interrompt toutes ces activités et sauvegarde l'état présent (registres, PC, accumulateurs, registre d'état) dans un registre particulier appelé **pile**. Les données y sont ''entassées'' comme on empile des livres (la

première donnée sauvegardée sera donc la dernière à être restituée). Ensuite, il exécute le programme d'interruption puis restitue l'état sauvegardé avant de reprendre le programme principale.

Remarques :

Certaine source d'interruption possède leur propre autorisation de fonctionnement sous la forme d'un bit à positionner, on l'appelle le masque d'interruption.

On peut donc interdire ou autoriser certaines sources d'interruptions, on les appelle les interruptions masquables.

Chaque source d'interruption possède un vecteur d'interruption où est sauvegardé l'adresse de départ du programme à exécuter.

Les interruptions sont classées par ordre de priorité. Dans le cas où plusieurs interruptions se présentent en même temps, le microprocesseur traite d'abord celle avec la priorité la plus élevée.

III.2.2 Echange direct avec la mémoire

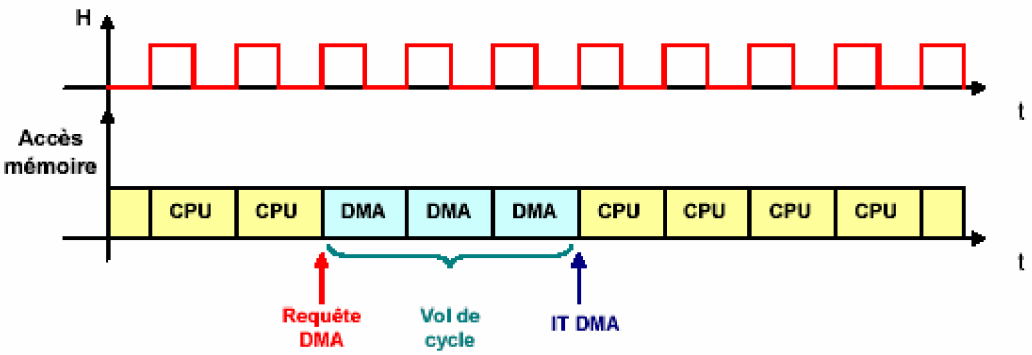
Ce mode permet le transfert de blocs de données entre la mémoire et un périphérique sans passer par le microprocesseur. Pour cela, un circuit appelé **contrôleur de DMA** (Direct Memory Access) prend en charge les différentes opérations.

Le DMA se charge entièrement du transfert d'un bloc de données. Le microprocesseur doit tout de même :

- Ø Initialiser l'échange en donnant au DMA l'identification du périphérique concerné.
- Ø Donner le sens du transfert.
- Ø Fournir l'adresse du premier et du dernier mot concernés par le transfert.

Un contrôleur de DMA est doté d'un registre d'adresse, d'un registre de donnée, d'un compteur et d'un dispositif de commande (logique câblée). Pour chaque mot échangée, le DMA demande au microprocesseur le contrôle du bus, effectue la lecture ou l'écriture mémoire à l'adresse contenue dans son registre et libère le bus. Il incrémente ensuite cette adresse et décrémente son compteur. Lorsque le compteur atteint zéro, le dispositif informe le processeur de la fin du transfert par une ligne d'interruption.

Le principal avantage est que pendant toute la durée du transfert, le processeur est libre d'effectuer un traitement quelconque. La seule contrainte est une limitation de ses propres accès mémoire pendant toute la durée de l'opération, puisqu'il doit parfois retarder certains de ses accès pour permettre au dispositif d'accès direct à la mémoire d'effectuer les siens : il y a apparition de vols de cycle.



III.3 Types de liaisons

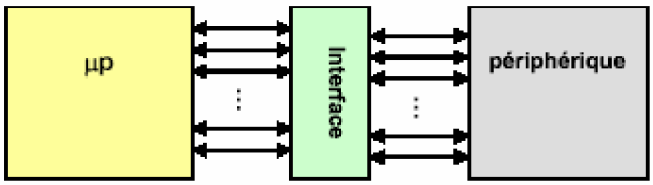
Les systèmes à microprocesseur utilisent deux types de liaison différentes pour se connecter à des périphériques :

Liaison parallèle
Liaison série

On caractérise un type de liaison par sa **vitesse de transmission** ou **débit** (en bit/s).

III.3.1 Liaison parallèle

Dans ce type de liaison, tous les bits d'un mot sont transmis simultanément. Ce type de transmission permet des transferts rapides mais reste limitée à de faibles distances de transmission à cause du nombre important de lignes nécessaires (coût et encombrement) et des problèmes d'interférence électromagnétique entre chaque ligne (fiabilité). La transmission est cadencée par une horloge



Exemple :
Bus PCI, AGP dans un PC.

III.3.2 Liaison série

Dans ce type de liaison, les bits constitutifs d'un mot sont transmis les uns après les autres sur un seul fil. Les distances de transmission peuvent donc être plus beaucoup plus importantes mais la vitesse de transmission est plus faible. Sur des distance supérieures à quelques dizaines de mètres, on utilisera des modems aux extrémités de la liaison.
La transmission de données en série peut se concevoir de deux façons différentes :

- En mode synchrone, l'émetteur et le récepteur possède une horloge synchronisée qui cadence la transmission. Le flot de données peut être ininterrompu.
- En mode asynchrone, la transmission s'effectue au rythme de la présence des données. Les caractères envoyés sont encadrés par un signal *start* et un signal *stop*.
-

Principe de base d'une liaison série asynchrone :

Afin que les éléments communicants puissent se comprendre, il est nécessaire d'établir un protocole de transmission. Ce protocole devra être le même pour chaque élément.

Paramètres rentrant en jeu :

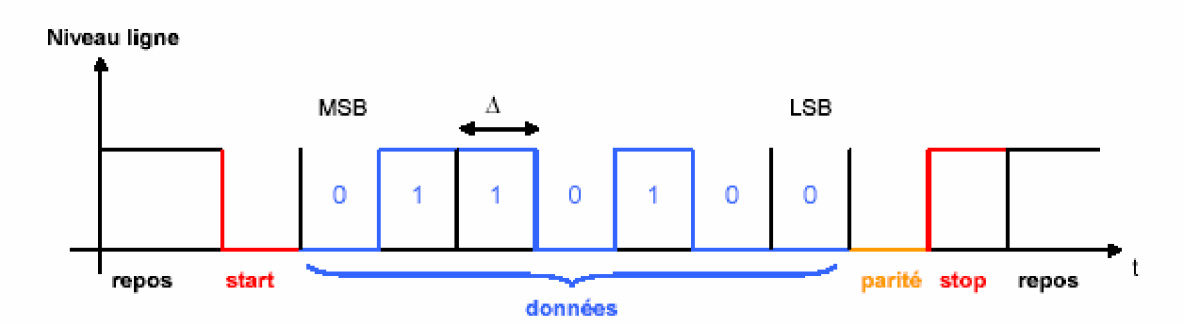
- **Longueur des mots transmis** : 7 bits (code ASCII) ou 8 bits.
- **Vitesse de transmission** : les vitesses varient de 110 bit/s à 128000 bit/s et détermine les fréquences d'horloge de l'émetteur et du récepteur.
- **parité** : le mot transmis peut être suivis ou non d'un bit de parité qui sert à détecter les erreurs éventuelles de transmission. Il existe deux types de parité : paire ou impaire. Si on fixe une parité paire, le nombre total de bits à 1 transmis (bit de parité inclus) doit être paire. C'est l'inverse pour une parité impaire.
- **bit de start** : la ligne au repos est à l'état 1 (permet de tester une coupure de la ligne). Le passage à l'état bas de la ligne va indiquer qu'un transfert va commencer. Cela permet de synchroniser l'horloge de réception.
- **bit de stop** : après la transmission, la ligne est positionnée à un niveau 1 pendant un certains nombre de bit afin de spécifier la fin du transfert. En principe, on transmet un, un et demi ou 2 bits de stop.

Déroulement d'une transmission :

Les paramètres du protocole de transmission doivent toujours être fixés avant la transmission.
En l'absence de transmission, la liaison est au repos au niveau haut pour détecter une éventuelle coupure sur le support de transmission. Une transmission s'effectue de la manière suivante :

1. L'émetteur positionne la ligne à l'état bas : c'est le bit de **start**.
2. Les bits sont transmis les un après les autres, en commençant par le bit de poids fort.
3. Le bit de parité est éventuellement transmis.
4. L'émetteur positionne la ligne à l'état haut : c'est le bit de **stop**.

Exemple : transmission d'un mot de 7 bits $(0110100)_2$ – Parité impaire – 1 bit de Stop



Horloge = $F=\frac{1}{\Delta}$ Hz

Vitesse de transmission = $v=\frac{1}{\Delta}$ bits/s

Contrôle de flux :

Le contrôle de flux permet d'envoyer des informations seulement si le récepteur est prêt (modem ayant pris la ligne, tampon d'une imprimante vide, etc…). Il peut être réalisé de manière logiciel ou matériel.

Pour contrôler le flux de données matériellement, il faudra utiliser des lignes de contrôle supplémentaire permettant à l'émetteur et au récepteur de s'informer mutuellement de leur état respectif (prêt ou non).

Dans un contrôle de type logiciel, l'émetteur envoie des données et lorsque le récepteur ne peut plus les recevoir (registre plein), il envoie une information à l'émetteur pour le prévenir, via la liaison série. L'émetteur doit donc toujours être à l'écoute du récepteur avant d'envoyer une donnée sur la ligne.

IV/- Les bus

Un bus est un ensemble de fils qui assure la transmission du même type d'information. On retrouve trois types de bus véhiculant des informations en parallèle dans un système de traitement programmé de l'information :

- **un bus de données** : bidirectionnel qui assure le transfert des informations entre le microprocesseur et son environnement, et inversement. Son nombre de lignes est égal à la capacité de traitement du microprocesseur.

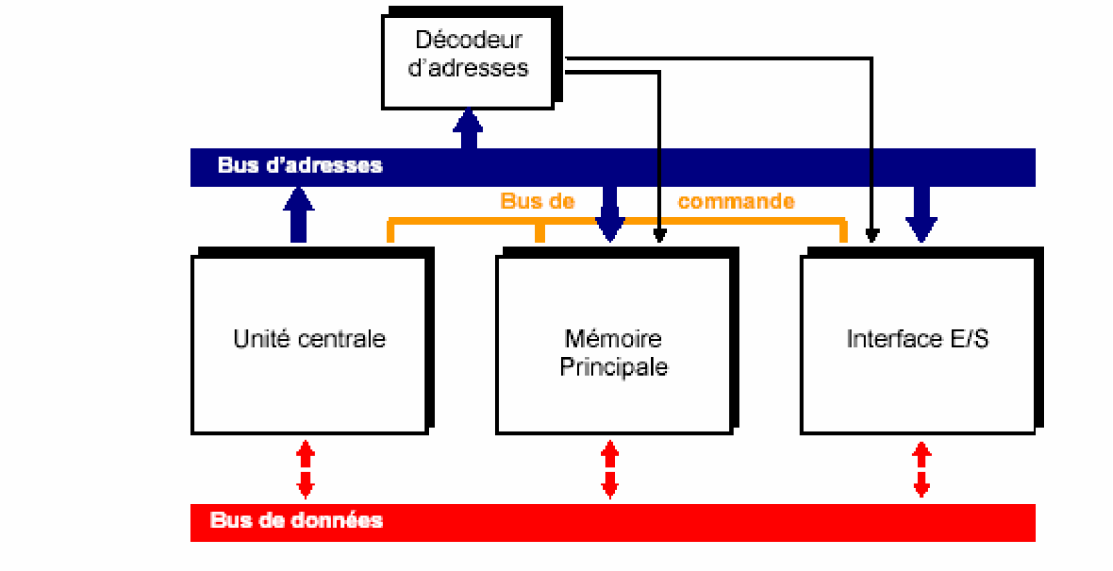
- **un bus d'adresses**: unidirectionnel qui permet la sélection des informations à traiter dans un *espace mémoire* (ou *espace adressable*) qui peut avoir 2n emplacements, avec n = nombre de conducteurs du bus d'adresses.
- **un bus de commande**: constitué par quelques conducteurs qui assurent la synchronisation des flux d'informations sur les bus des données et des adresses.

IV.6 Décodage d'adresses

La multiplication des périphériques autour du microprocesseur oblige la présence d'un **décodeur d'adresse** chargé d'aiguiller les données présentes sur le bus de données.

En effet, le microprocesseur peut communiquer avec les différentes mémoires et les différents boîtiers d'interface. Ceux-ci sont tous reliés sur le même bus de données et afin d'éviter des conflits, un seul composant doit être sélectionné à la fois.

Lorsqu'on réalise un système micro programmé, on attribue donc à chaque périphérique une zone d'adresse et une fonction « décodage d'adresse » est donc nécessaire afin de fournir les signaux de sélection de chacun des composants.



Remarque : lorsqu'un composant n'est pas sélectionné, ses sorties sont mises à l'état « haute impédance » afin de ne pas perturber les données circulant sur le bus. (elle présente une impédance de sortie très élevée = circuit ouvert).