ARCHITECTURE DES SYSTEMES PROGRAMMES

Baccalauréat STI2D-SIN

- ET 2.2.2 : Système architectural de la chaîne d'information
- ET 3.1.4 : Traitement programmé : Structure à base de microcontrôleurs et structures spécialisées
- SIN 2.1 : Traitement programmé et composants programmables
- SIN 3.1 : Implémentation d'un programme dans un composant programmable
- SIN 3.1 : Interfaçage de composants

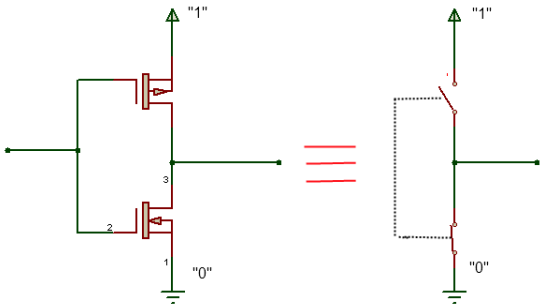
Objectifs

- identifier les éléments transformés et les flux
- décrire les liaisons entre les blocs fonctionnels
- identifier l'organisation structurelle

Transistors et logique binaire

Les structures de l'électronique numérique reposent toutes sur des composants appelés transistors.

Les transistors s'apparentent à des interrupteurs qui laissent passer ou non le courant électrique. L'utilisation de transistors complémentaires (quand l'un laisse passer le courant, l'autre le bloque) permet d'obtenir des cellules unitaires qui délivrent soit un niveau logique "0" ou un niveau logique "1". Ce fonctionnement binaire - on parle de logique binaire -explique le fait que l'on utilise la base 2 pour coder des programmes informatiques.



Le bit

De l'anglais ***binary digit***, le bit décrit une variable n'ayant que deux états possibles.

Par exemple : Un nombre peut être pair ou non.

Lorsque des combinaisons intermédiaires sont nécessaires, on associe plusieurs bits pour obtenir des variables de valeurs plus grandes.

L'octet

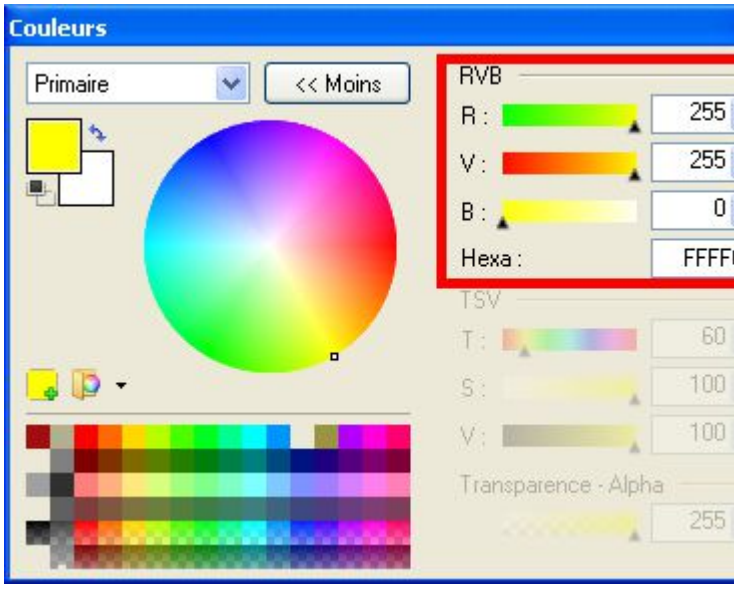
L'association de 8 bits forment un ***octet*** (***BYTE*** en anglais). Un octet peut représenter 256 valeurs différentes (2^8).

Parce que l'écriture binaire est fastidieuse, on écrit les octets en hexadécimal (base 16). Pour la base 16 voir le cours Systèmes de numération.

Par exemple la couleur d'un pixel à l'écran est caractérisée par 3 variables qui sont ROUGE, VERT et BLEU. Ainsi la couleur jaune est définie par ROUGE=255=$FF VERT=$FF=255 et BLEU = $00=0

Remarque : pour signaler un mot hexadécimal, on utilise la lettre $ ou h ou 0x.

Exemple : $FF=FFh=0xFF



Koctets - Kioctets

On avait pris pour habitude de qualifier 1024 octets comme un KOctet.

Cette donnée a changé il y a quelques temps pour être conforme au souhait des organismes de normalisation.

Ainsi 1 ***KOctet*** = 1000 octets

Une nouvelle unité est apparue qui est le ***KiOctet*** : le kilo informatique qui vaut 2^{10} octets soit 1024 octets.

Structures combinatoires

On appelle structure combinatoire, l'association d'éléments logiques pour lesquels un changement d'une variable d'entrée introduit immédiatement un changement possible de la sortie.

Exemple :

Soit un coffre fort disposant de quatre boutons. Si chaque bouton est positionné sur la bonne valeur, le coffre s'ouvre. L'ordre de manipulation n'a aucune importance. L'ouverture est simplement conditionnée par la bonne combinaison.

Il s'agit d'un fonctionnement combinatoire.

Les portes logiques sont des structures combinatoires.

Structures séquentielles

On appelle structure séquentielle, l'association d'éléments logiques faisant intervenir la notion de mémoire et d'états précédents pour provoquer le changement d'état de sortie de la structure.

Il s'agit d'un fonctionnement qui respecte une séquence. Il s'agit d'un fonctionnement séquentiel.

Exemple :

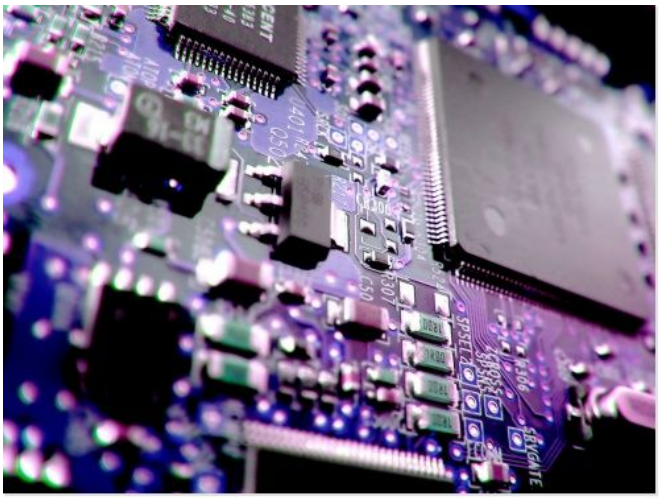
Une carte bancaire dispose d'un code secret de 4 chiffres. L'ordre de saisi à son importance. Pour que le code soit accepté, il faut que les 4 numéros soient exacts et saisis dans l'ordre.

Les registres, les bascules logiques, les compteurs sont des structures séquentielles.

Architecture minimale

Introduction

La plupart des dispositifs électroniques reposent aujourd'hui sur une structure minimale micro-programmée assurant le séquencement de tâches plus ou moins complexes selon un algorithme défini.



Selon la complexité ou l'importance des actions à réaliser cette structure peut être :

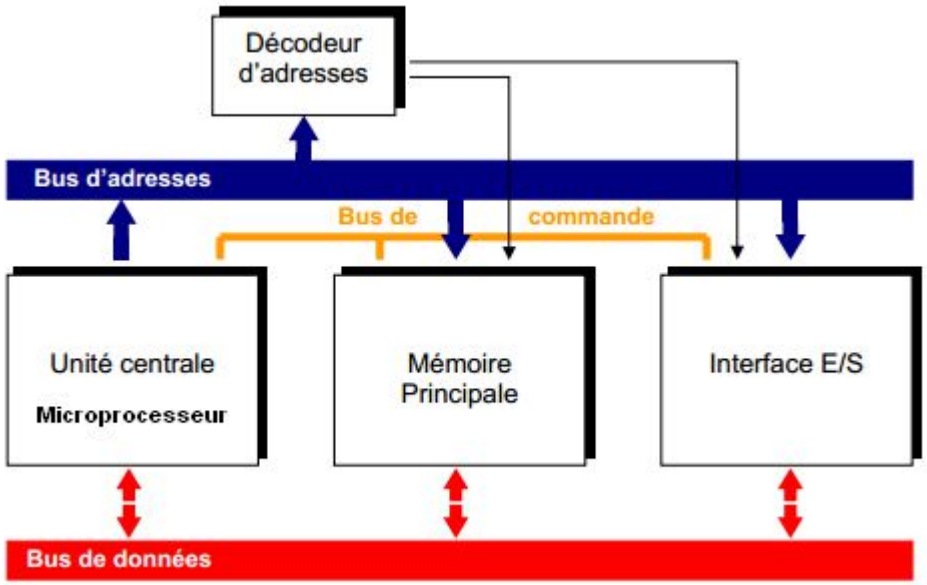
- un seul circuit intégré de forte intégration (*LSI : Large Scale Integration*) appelé ***micro-contrôleur***

- un ensemble de circuits qui dialoguent ensemble et qui constituent un ***système minimal*** pour lequel le microprocesseur serait le "cerveau".

L'avantage d'une structure micro programmée est qu'elle est ***adaptable*** à n'importe quel système travaillant dans des contraintes technologiques identiques (temps de réponse, encombrement, température, etc...) et qu'elle est capable de gérer un grand nombre de variables logiques ou numériques. Seuls ses liaisons avec l'extérieur et son programme de traitement seront changés. On utilise par exemple le même micro-contrôleur pour gérer un lave-linge ou l'ordinateur de bord d'une voiture. Enfin son ***coût de revient est faible***.

Etude du système de traitement

Schéma fonctionnel



Composition

John Von Neumann, s'appuyant partiellement sur les travaux d'Alan TURING, a élaboré en 1946 un modèle de traitement de l'information qui a court encore aujourd'hui.

Dans ce modèle, trois blocs fonctionnels sont indispensables :

- ***l'unité centrale*** ou ***CPU*** qui réalise les opérations
- la mémoire qui stocke les informations (programme + données)
- l'unité d'interfaçage qui permet de communiquer avec l'extérieur

Les données (codes de programme et variables) sont véhiculées par un ensemble de fils appelé ***BUS DE DONNEES***. Le bus de données est bidirectionnel et son nombre de fil dépend de la capacité de traitement du microprocesseur. 8 fils s'il s'agit d'un microprocesseur 8 bits.

Le bus d'adresse unidirectionnel véhicule le numéro de la case mémoire vers laquelle la donnée doit aller ou d'où elle doit venir.

Le bus de commande comporte des signaux utiles au fonctionnement de l'ensemble (signaux de cadencement, sélection de lecture ou d'écriture, etc....).

L'unité centrale

C'est 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 une horloge.

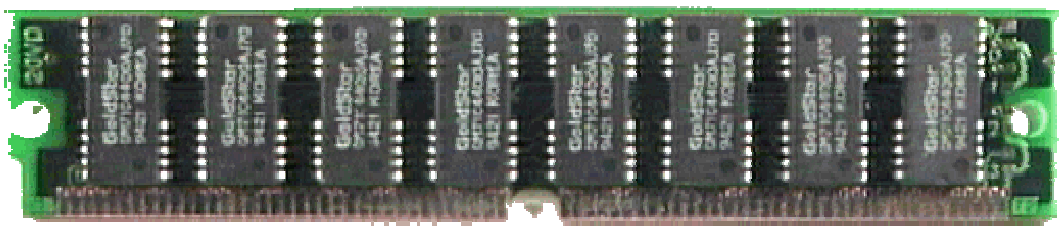
Les caractéristiques principales du microprocesseur sont :

- 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

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.



Remarque : concernant les disques durs ou autres CD on les appelle le plus souvent des mémoires de masse en raison de leur grande capacité de stockage

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…).

Elle est associée à des composants appelés COUPLEURS.

Adresses et décodage d'adresses

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

En effet, le microprocesseur peut communiquer avec les différentes mémoires et les différents boîtier 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'adresses » est donc nécessaire afin de fournir les signaux de

sélection de chacun des composants. Ces signaux se nomment généralement ***Chip Select*** (CS).

Etude du microprocesseur

Cette étude se limitera à l'examen simplifié du modèle de Von Neumann.

Les microprocesseurs ne cessent d'évoluer tant en terme de

- vitesse d'exécution,
- de consommation
- et d'intégration.

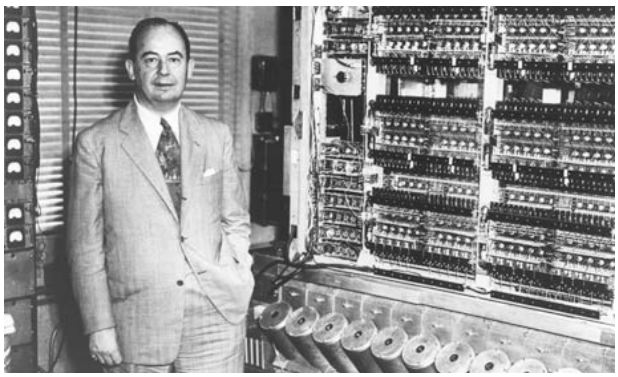
Le tableau ci-dessous présente l'évolution de 1971 à 2010 :

Date	Nom	Nombre de transistors	Finesse de gravure (µm)	Fréquence de l'horloge	Largeur des données	MIPS
1971	4004	2 300		108 kHz	4 bits/4 bits bus	
1974	8080	6 000	6	2 MHz	8 bits/8 bits bus	0,64
1979	8088	29 000	3	5 MHz	16 bits/8 bits bus	0,33
1982	80286	134 000	1,5	6 à 16 MHz (20 MHz chez AMD)	16 bits/16 bits bus	1
1985	80386	275 000	1,5	16 à 40 MHz	32 bits/32 bits bus	5
1989	80486	1 200 000	1	16 à 100 MHz	32 bits/32 bits bus	20
1993	Pentium	3 100 000	0,8 à 0,28	60 à 233 MHz	32 bits/64 bits bus	100
1997	Pentium II	7 500 000	0,35 à 0,25	233 à 450 MHz	32 bits/64 bits bus	300
1999	Pentium III	9 500 000	0,25 à 0,13	450 à 1 400 MHz	32 bits/64 bits bus	510
2000	Pentium 4	42 000 000	0,18 à 0,065	1,3 à 3,8 GHz	32 bits/64 bits bus	1 700
2004	Pentium 4D « Prescott »	125 000 000	0,09 à 0,065	2.66 à 3,6 GHz	32 bits/64 bits bus	9 000
2006	Core 2™ Duo	291 000 000	0,065	2,4 GHz (E6600)	64 bits/64 bits bus	22 000
2007	Core 2™ Quad	2*291 000 000	0,065	3 GHz (Q6850)	64 bits/64 bits bus	2*22 000 (?)
2008	Core 2™ Duo (Penryn)	410 000 000	0,045	3,33 GHz (E8600)	64 bits/64 bits bus	~24 200
2008	Core 2™ Quad (Penryn)	2*410 000 000	0,045	3,2 GHz (QX9770)	64 bits/64 bits bus	~2*24 200
2008	Intel Core i7 (Nehalem)	731 000 000	0,045 (2008) 0,032 (2009)	2,66 GHz (Core i7 920) 3,33 GHz (Core i7 Ext. Ed. 975)	64 bits/64 bits bus	?
2009	Intel Core i5/i7 (Lynnfield)	774 000 000	0,045 (2009)	2,66 GHz (Core i5 750) 2,93 GHz (Core i7 870)	64 bits/64 bits bus	?
2010	Intel Core i7 (Gulftown)	1 170 000 000	0,032	3,33 GHz (Core i7 980X)	64 bits/64 bits bus	?

Les premiers microprocesseurs se basaient sur une architecture Von Neumann tandis que des processeurs nécessitant plus de puissance de calcul reposaient sur une architecture Harward.

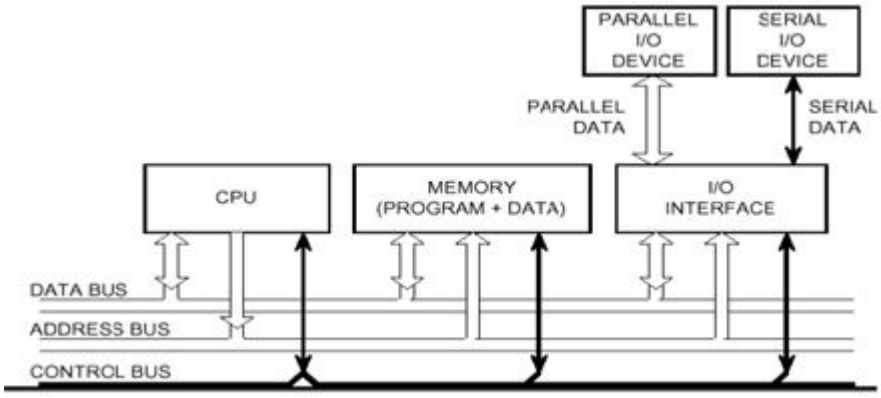
Depuis quelques années, on constate que les microprocesseurs utilisent les atouts des deux architectures.

Le modèle Von Neumann

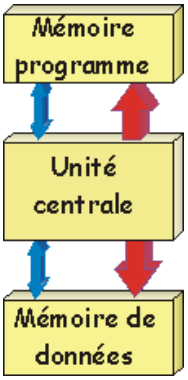


Le principe de base de l'architecture de Von Neumann est que les données et les instructions sont véhiculées sur un même bus qui est le bus de données.

Pour accéder à ces données situées en mémoire, ou dans des cases mémoires des circuits périphériques, le bus d'adresse permet de véhiculer un mot binaire qui défini où se trouvent les données.



Remarque : Dans une architecture Harvard, les données et les instructions circulent sur deux bus différents.



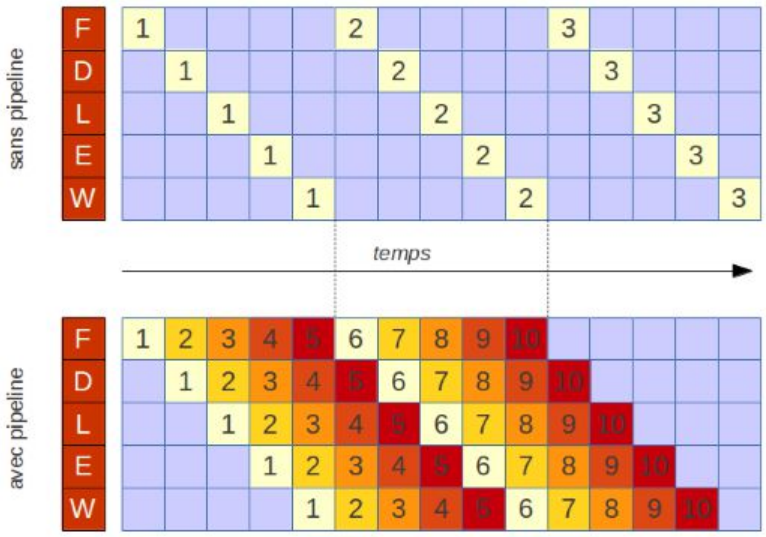
CISC RISC

On appelle un processeur ***CISC***, *Complex Instruction Set Computer*, un processeur disposant d'un nombre important d'instructions (plus de 300 pour un 68HC11 de Motorola)

Or statistiquement, 80% des traitements des langages de haut niveau font appel à 20% d'instructions, d'où l'idée de développer des processeurs ayant moins d'instructions mais dont le traitement est optimisé.

Ces processeurs RISC, Reduced Instruction Set Computer, ont été développé par Microchip avec les microcontrôleurs PIC ou par IBM avec les PowerPC, SPARC ou Alpha.

L'optimisation est obtenue par l'usage de nombreux registres internes et des méthodes de *pipelining*.



F:fetch; D: Decode; L: load; E:Execute; W:write

(sources Jean-Michel Richer - Université d'Anger)

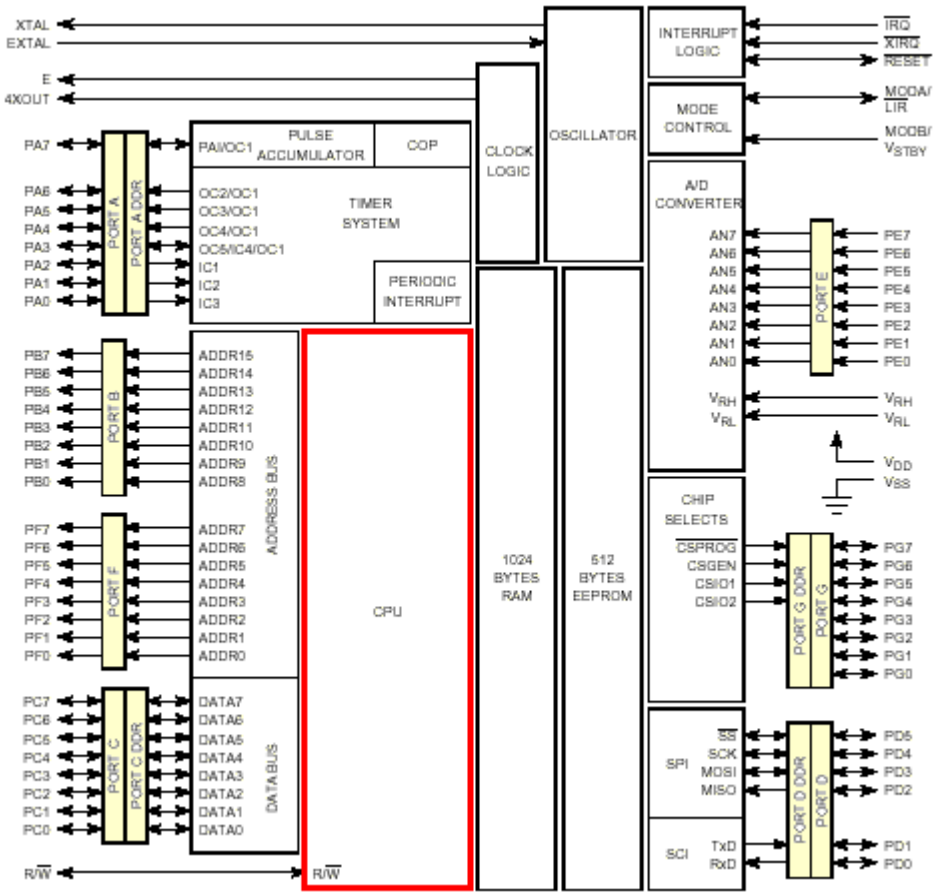
Actuellement, la plupart des processeurs CISC utilisent des principes d'optimisation du RISC.

Structure du processeur

Microprocesseur - Microcontôleur

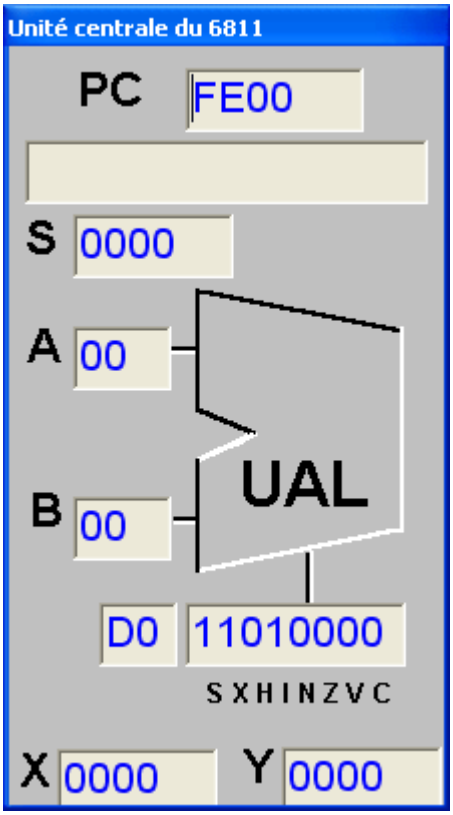
Ce qui distingue un microcontôleur d'un microprocesseur est que le microcontôleur intègre, en plus de l'unité centrale (***CPU : Central Processing Unit***), également des structures périphériques :

- mémoires
- structures d'interfaçage parallèle
- structure d'interfaçage série
- structures de conversions analogique et numériques
- circuits temporisateurs
- etc...



Structure interne d'une CPU

Reprenons l'exemple du 68HC11F1.



- Son ***unité arithmétique et logique*** (***UAL***) permet de réaliser des opérations sur 8 bits et certaines opérations sur 16 bits. On dit qu'il s'agit d'un processeur 8/16 bits.
- Les ***accumulateurs*** A et B sont des registres de 8 bits qui contiennent les opérandes d'une opération. Certaines instructions associent ces deux accumulateurs pour réaliser des opérations sur 16 bits : A (8bits) +B (8bits) = D (16bits).
- Le ***Program Counter*** (***PC***) ou compteur ordinale est un registre de 16bits qui contient en permanence l'adresse de la prochaine instruction à exécuter
- Le ***Stack Pointer*** (S) est un registre de 16 bits qui contient l'adresse de la pile mémoire. Cette dernière est nécessaire pour faciliter la sauvegarde des données temporaires.
- Lié à l'UAL le ***Code Condition Register*** (CCR) ou drapeau donne des indications sur un résultat : valeur nulle, valeur négative, résultat avec retenu, etc... Voir ci-dessous.
- X et Y sont des registres de 16 bits qui permettent de manipuler les adresses. Cette CPU est donc capable d'adresser 2^{16} adresses différentes.

L'unité de contrôle qui n'est pas représentée sur la figure ci-contre permet de cadencer et d'organiser l'exécution des instructions.

Le registre de code condition

Selon le résultat d'une opération les bits du registre CCR sont positionnés à "1" ou "1". Ces bits sont utilisés par le microprocesseur pour effectuer par la suite des tests conditionnels de type SI... ALORS.. SINON... FINSI

Les bits les plus courants sont :

- C : Carry : positionné à "1" si le résultat d'une opération possède une retenue.
- Z : Zero : Positionné à "1" si le résultat d'une opération est nulle.
- N : Negative : Positionné à "1" si le résultat d'une opération est négatif. Plus précisement si le bit de poids fort est mis à "1".

Exécution d'une instruction

Type d'instruction

Il existe différents type d'instructions :

- opérations arithmétiques ou logiques : elles sont en lien avec l'UAL : Addition, complémentation...
- opérations d'échange ou de transfert : sauver l'accumulateur, charger l'accumulateur avec une donnée en mémoire...
- opérations liées au fonctionnement du processeur : arrêt logiciel, masquage d'interruption...

Dans certains cas les opérations ne nécessitent aucun opérande, et l'instruction tient sur un octet : exemple : Incrémenter A (INCA). On parle ***d'adressage implicite***.

D'autres instructions nécessitent un opérande de 1 octet spécifié immédiatement à la suite de l'instruction: Exemple : Ajouter $20 à la valeur de l'accumulateur (LDAA #$20). Il s'agit d'un ***adressage immédiat***.

Enfin des instructions peuvent spécifier l'emplacement de la donnée par une adresse sur un ou deux octets : exemple : mettre le contenu de la case mémoire d'adresse $1000 dans l'accumulateur (LDAA $1000). Il s'agit d'un ***adressage direct*** ou ***étendu***.

Quelque soit le type d'instruction, le code binaire qui la caractérise décrit la manière dont elle doit être exécutée.

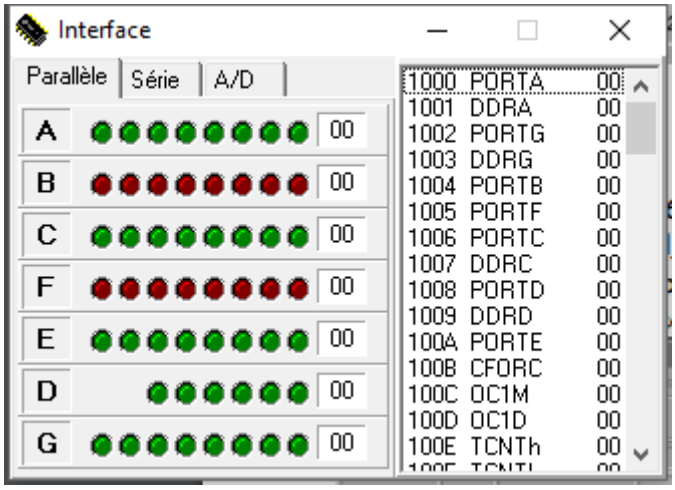
C'est ***l'Unité de Contrôle*** composée du registre d'instruction, du décodeur et du séquenceur qui assure l'exécution de l'instruction.

Les coupleurs d'entrées/sorties

On appelle ***coupleurs*** des circuits destinés à permettre la communication entre le microprocesseur et son environnement.

On distingue généralement :

- le coupleur parallèle qui est capable de transmettre plusieurs bits simultanément. Selon le type, il peut recevoir des signaux d'entrées (capteurs logiques par exemple) ou commander des préactionneurs en sortie (transistors de puissance, relais, etc...)
- le coupleur série est destiné à gérer les liaisons séries synchrones ou asynchrones (selon le type).



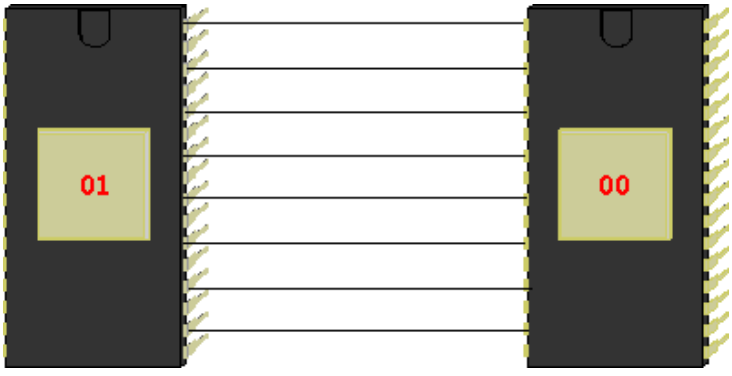
Les coupleurs sont toujours associés à des registres :

- de configuration
- d'entrée/sortie

La lecture du document technique du microcontrôleur est un préalable avant de vouloir brancher ou programmer un coupleur.

Transmission parallèle/Coupleur parallèle

On réalise une communication parallèle lorsque plusieurs bits sont transférés sur des fils différents d'un émetteur vers un récepteur.



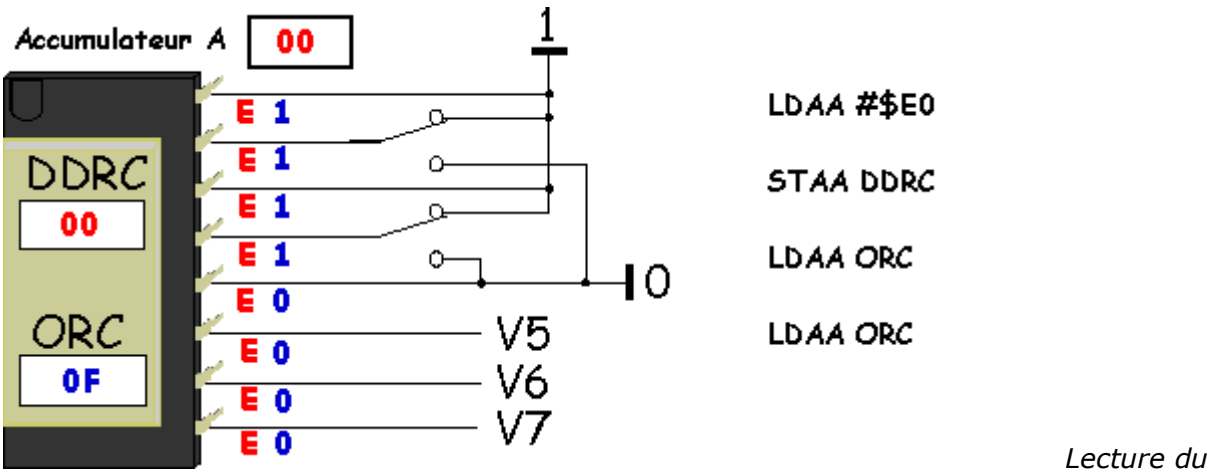
Le rôle du coupleur parallèle est d'assurer cette liaison entre "l'extérieur" et le microprocesseur. Pour ce faire, il dispose généralement d'un registre de configuration, d'un registre d'entrée sortie, et est en liaison avec le microprocesseur par l'intermédiaire du bus de donnée et du bus d'adresse.

Motorola

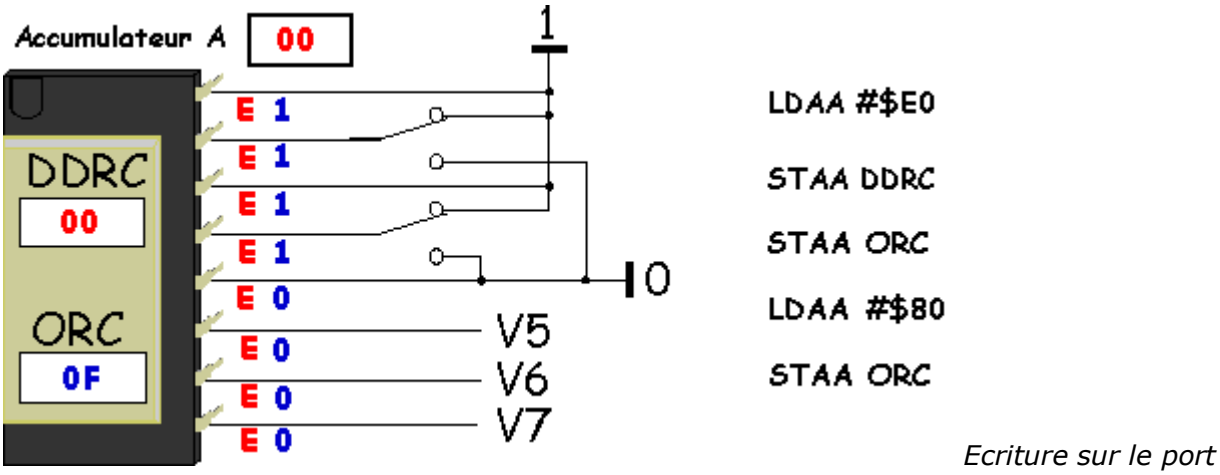
Chez le constructeur MOTOROLA, le registre DDR spécifie le sens de circulation es informations.

- Un bit de DDR à 0 spécifie une configuration en entrée.

- Un bit de DDR à 1 spécifie une configuration en sortie.



Lecture du Port C



Ecriture sur le port C

Microchip

Le fabriquant des micro-contrôleur PIC utilise le même principe que Motorola.

Le registre qui détermine le sens des informations s'appelle TRIS.

Un bit à "0" caractérise une sortie et un bit à "1" caractérise une entrée

- Pour le port A : TrisA
- Pour le port B : TrisB
- etc....

Exemple de programmation du TRIS en langage C

```
TRISA = 0x3F;//0x3F = %00111111 les 6 bits de poids faible sont en entrée

TRISD = 0x00;//0x00 = %00000000 le port est en sortie
```

Les ports d'entrée sortie sont appelés PortA, PortB, etc....

Exemple de programmation du Port en langage C

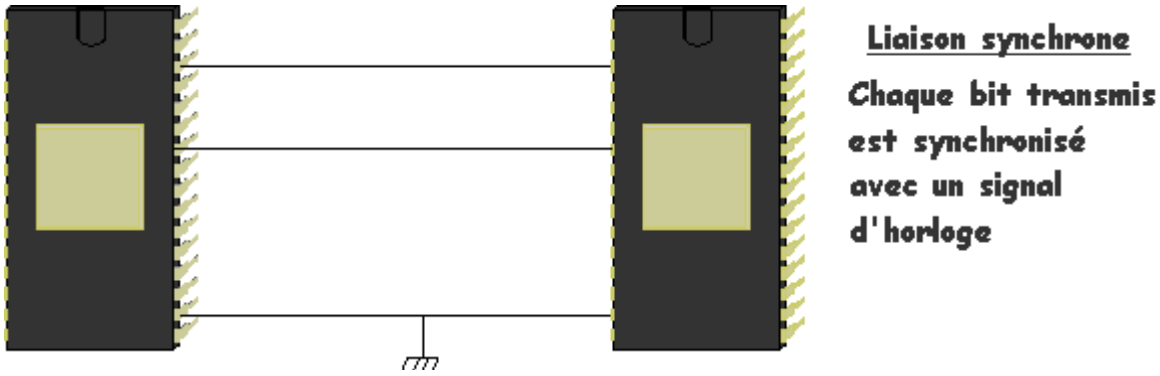
```
config=PortA; //affecte à la variable config les états du port A
PortD.F0=1; //fixe le bit de poids le plus fort de PortD à 1
```

Transmission série/Coupleur série

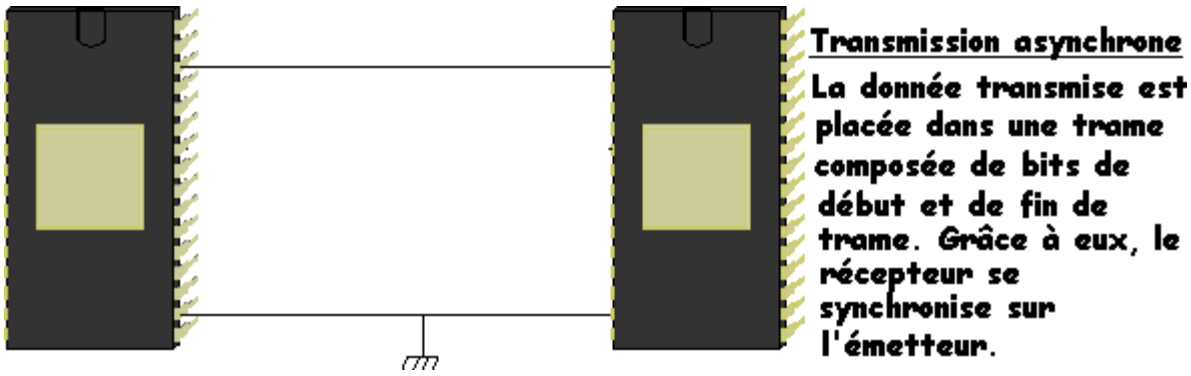
Dans une transmission série, les bits sont transmis un à un entre l'émetteur et le récepteur.

Liaison synchrone - Liaison asynchrone

Dans une liaison ***synchrone***, une horloge cadence la transmission ce qui permet au récepteur de réceptionner l'état logique du signal au bon moment.



Cette liaison présente l'inconvénient de nécessiter trois fils entre l'émetteur et le récepteur. Une liaison ***asynchrone***, évite cela mais impose la présente de bits de synchronisation (***start*** et ***stop***).



Transmission asynchrone RS232

La liaison série la plus classique est la liaison RS232. Les ports associés s'appellent souvent ***UART*** : *Universal Asynchronous Receiver Transmitter*

Ici, on synchronise les données grâce à des bits supplémentaires rajoutées aux données et qui permettent au récepteur de se resynchroniser sur l'émetteur.

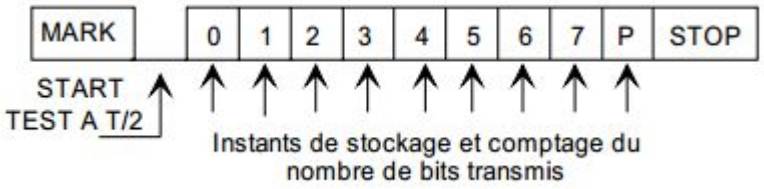
Ces bits sont :

•le bits de start : il est à 0. C'est lui qui signale le début d'un nouvel octet

•les bits de STOP : ils sont à 1 ou 2. Ils signale la fin de l'octet transmis

•le bit de parité : il n'est pas forcément présent. Il s'agit d'un bit placé avant le bit de stop et qui permet de vérifier si la donnée à été correctement transmise. Si le nombre de bits à "1" était pair le bit de parité est à un. Le bit de parité peut être en logique positive (EVEN) ou négative (ODD).

•MARK : il s'agit de l'état de repos en l'absence de transmission.



Dans une transmission série la distance dépend de la vitesse de transmission :

- 533m à 1200bauds (bits/seconde)
- 76m à 9600bds...

Parité

Le bit de parité permet de détecter certains cas de mauvaise transmission.

Si la parité est paire, le nombre de bits à "1" des données et de la parité doit être pair.

Exemples

- pour la donnée 01100001 le bit de parité sera à 1
- pour 11100001 le bit de parité sera à 0

Si la parité est impaire, le nombre de bits à "1" des données et de la parité doit être impair.

Exemples

- pour la donnée 01100001 le bit de parité sera à 0
- pour 11100001 le bit de parité sera à 1

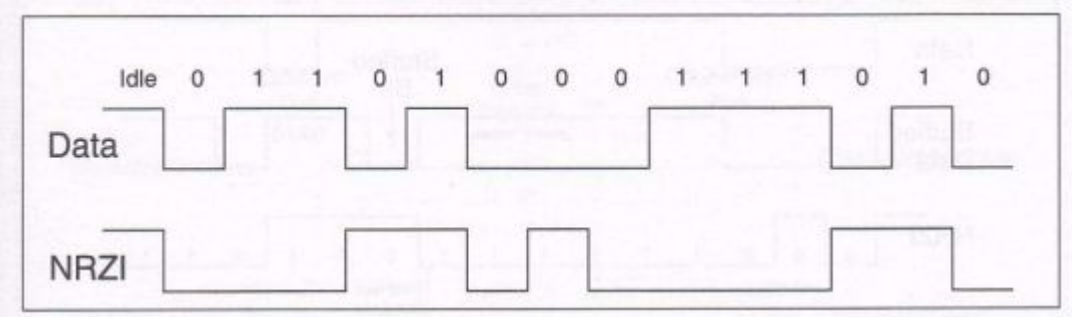
USB

La liaison USB (***Universal Serial Bus***) est un bus série qui permet des connexions à chaud des périphériques sur les ordinateurs. Le connecteur

USB dispose de quatres fils

- une alimentation de +5V
- La référence de tension GND
- de deux signaux différentiels (RS485)

Les chronogrammes de transmissions sont différents du protocole RS232 dans le sens où les signaux respectent le codage NRZI (Non Retour à Zero Inversé) : un niveau logique 1 (NL1) est représenté par un non changement d'état du signal alors qu'un NL0 entraîne un changement d'état.



La synchronisation des mots transmis se fait un mot de start de 8 bits et deux bits de stop. Le protocole de transmission est bien plus complexe que le RS232 et dépasse le cadre de ce cours.

Programmation du port série en C sur un 16F877

initialisation de l'UART à 3125bits/seconde, 8bits de données, un bit de stop et pas de parité

```
UART1_init(31250);
```

Lecture du message reçu en série :

```
while (UART1_Data_Ready()==0); // on attend la donnée
  code=UART1_Read();//code prend la valeur reçue
```

Ecriture d'un caractère sur le port de sortie série :

```
UART1_write(caractere);
```