GIF-3002 Introduction

Ce document rappelle certaine notions d'électronique et d'informatique nécessaires à la bonne compréhension du cours de Système Microprocesseur et Interfaces. La suite du cours assumera que les notions qui suivent sont assimilées, car déjà vue dans d'autres cours.

1 Historique des microprocesseurs

Les ordinateurs sont apparus vers la fin de la seconde guerre mondiale (1939, ABC --- 1945, Eniac). À l'époque, ils étaient immenses car chaque transistor était un tube cathodique. C'est aussi vers la fin de la seconde guerre mondiale qu'est apparue l'architecture Von Neumann, une architecture de base des ordinateurs modernes (voir plus loin).

L'apparition des transistors vers la fin des années 1960 (1954, Bipolar Junction Transistor, Texas Instrument --- 1960, Metal-Oxyde Semiconductor, Bell Labs) a permis de réduire considérablement la taille des circuits avec microprocesseurs.

Les transistors ont ensuite été intégrés dans de petits circuits électroniques simples (portes logiques ou amplificateur opérationnel par exemple). Puis, ils ont été intégré dans des circuits de plus en plus complexes (VLSI = Very Large Scale Integration).

Les premiers microprocesseurs avec transistors (de 1964 à 1971) étaient donc constitués de plusieurs circuits intégrés simples. Ils étaient conçus avec une logique câblée (le microprocesseur exécute chaque instruction avec du matériel unique) ou avec des micro instructions (le microprocesseur découpe chaque instruction en sous-instructions qui peuvent être communes à des instructions différentes)[1].

En 1971 est apparu le premier microprocesseur entièrement contenu sur un seul die : le 4004 d'Intel. Ce précurseur a rapidement été suivi d'autres microprocesseurs, de plus en plus puissants.

La Loi de Moore (1965, par Gordon Moore, co-fondateur d'Intel), décrit en partie l'augmentation de puissance des microprocesseurs : elle stipule que le nombre de transistors par pouce carré, pour les microprocesseurs, double à tous les deux ans (voir http://fr.wikipedia.org/wiki/Loi_de_Moore pour une illustration). Cette augmentation est permise grâce à l'amélioration technologique des techniques "d'écriture" des transistors à l'intérieur de silicone.

Les concepteurs de microprocesseurs ont utilisé les transistors, de plus en plus nombreux pour une surface donnée, de plusieurs façons. D'un côté, la puissance des microprocesseurs a été augmentée. D'un autre côté, le niveau d'intégration a augmenté :

[1] Dans les deux cas, du matériel exécute les instructions. De nos jours, les microprocesseurs découpent toutes les instructions en sous-instructions.

plusieurs composantes connexes aux microprocesseurs ont été intégrées dans le même circuit, dans la même matrice de silicone.

Parallèlement à l'augmentation de puissance des microprocesseurs causée par l'amélioration technologique, l'architecture des microprocesseurs a aussi été revue. Plusieurs concepts nouveaux ont vu le jour afin d'accélérer l'exécution des instructions, surtout dans les années 90: pipelines, caches, unité de gestion de la mémoire, prédiction des branchements, des systèmes de gestion de la consommation de puissance et plus…

Dans les années 2000-2010, les transistors, toujours plus nombreux par unité de surface, ont été essentiellement utilisés afin d'augmenter le nombre de cœurs à l'intérieur des microprocesseurs. Des avancées techniques, ont permis d'accélérer le traitement parallèle, au niveau matériel, de plusieurs processus différents.

1.1 Marché actuel du microprocesseur

De nos jours, il y a trois catégories importantes de systèmes à microprocesseurs vendus à large échelle : les systèmes embarqués, les serveurs et les ordinateurs personnels. Comme le cours met l'emphase sur les microcontrôleurs, vous êtes invités à lire :

(http://www.emittsolutions.com/images/microcontroller_market_analysis_2008.pdf).

En 2011, le marché des microcontrôleurs s'élèvera autour de 16 milliards ($US) selon plusieurs sources, avec une croissance autour de 8-10% par année, causée par la Chine et l'Asie. La plupart des microcontrôleurs servent dans des applications industrielles et pour l'automobile (moteur et coussin gonflable). Les plus gros vendeurs sont Renesas-NEC (fusionnés en 2009-2010, 25% du marché), Freescale (anciennement Motorola, 10 % du marché) et quelques ayant autour de 5-10% du marché (Microship (PIC), Atmel, ST Micro). Les architectures de cœur les plus utilisés sont Intel, Renesas, ARM, PIC et Atmel.

Par ailleurs, en 2011, le marché des microprocesseurs pour PC s'élèvera autour de 28 milliards selon plusieurs sources. Les plus gros vendeurs sont Intel (80% du marché) et AMD (19.8% du marché).

2 Fonctionnement d'un système microprocesseur et interfaces

2.1 Composantes du système

Les composantes suivantes se retrouvent dans tout SMI ou ordinateur:

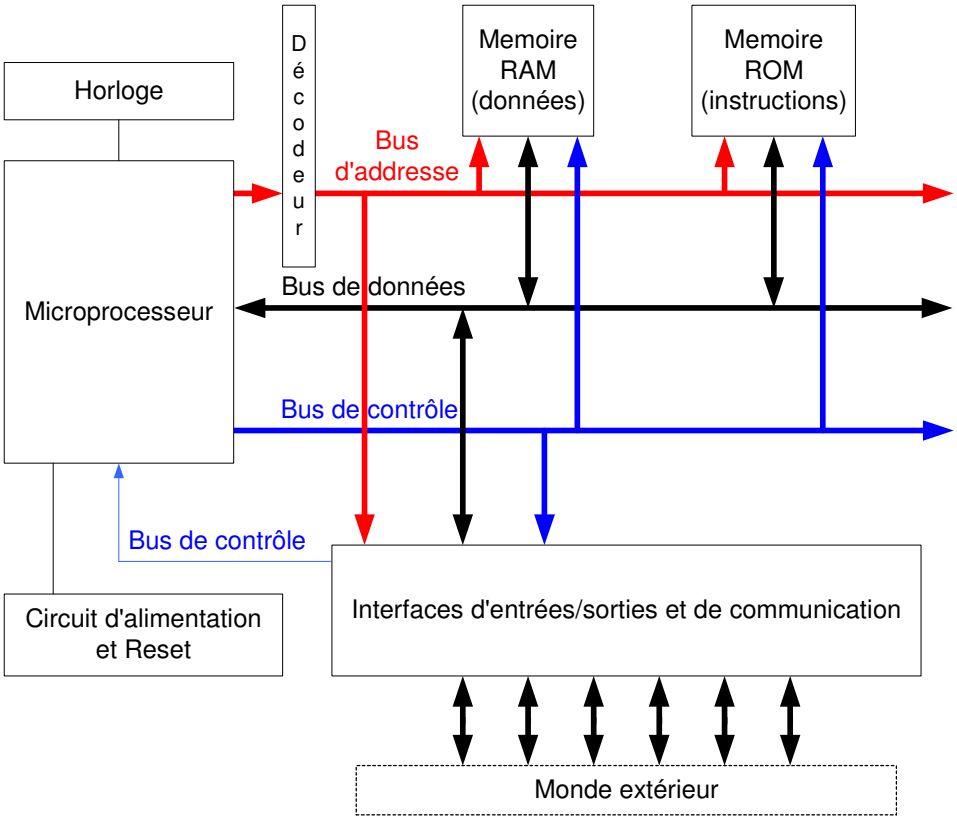


Figure 1 - Système microprocesseur avec ses interfaces

2.1.1 Le microprocesseur

- Le microprocesseur exécute des instructions.
- Le microprocesseur accomplit les tâches suivantes, en boucle perpétuelle :
  - Lire une instruction de la mémoire
  - Décoder l'instruction lue
  - Exécuter l'instruction
  - Lire une instruction de la mémoire
  - Décoder l'instruction lue
  - Exécuter l'instruction
  - …
- Le microprocesseur contient plusieurs éléments de base tel qu'illustré ci-dessous.
  - L'unité de contrôle (CCU) gère l'exécution des instructions.
  - Le MMU (Memory Management Unit) détermine l'adresse des instructions (ou des données) à traiter dans la mémoire (ou les caches).
  - L'unité de branchement gère les instructions de saut
  - L'unité d'accès aux données gère les instructions lisant ou écrivant une donnée en mémoire (Load/Store)
  - L'ALU (Arithmetical and Logical Unit) permet d'effectuer des opérations mathématiques ou booléennes
  - L'interface de bus gère les signaux des bus pour accéder à la mémoire

- Les registres sont des variables spéciales dans le microprocesseur (lire des bascules D) permettant d'entreposer temporairement des valeurs pour faire des calculs ou ayant un rôle particulier dans le microprocesseur.

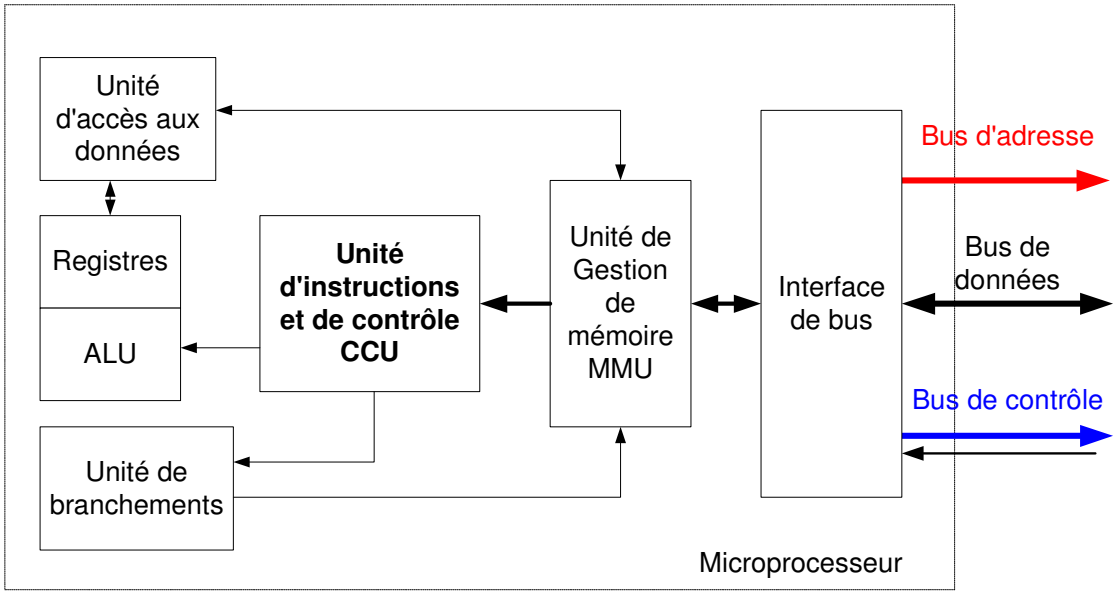


Figure 2 – Microprocesseur, architecture Von Neumann

2.1.2 La mémoire

- La mémoire contient les programmes.
- La mémoire contient les données.
- Il y a plusieurs types et catégories de mémoire. La mémoire RAM (SRAM, DRAM, DRR…) est une mémoire volatile qui contient habituellement les données traitées par vos programmes. La mémoire ROM (EEPROM, FLASH, ROM…) est une mémoire non-volatile qui contient habituellement les instructions de vos programmes.
- Pour lire la mémoire: une adresse doit être choisie à partir du bus d'adresse et l'activation de signaux sur le bus de contrôle doit déclencher l'opération. Les données de l'adresse choisie se retrouvent sur le bus de données.
- Pour écrire la mémoire: une adresse doit être choisie à partir du bus d'adresse et le bus de contrôle doit déclencher l'opération. Les données à l'adresse choisie sont remplacées par celles sur le bus de données.

2.1.3 Les bus et le décodeur d'adresse

- Un bus est un groupe de lignes. C'est une façon de nommer un ensemble de fils électriques.
- Le bus d'adresse indique quel périphérique du microprocesseur est sélectionné. De plus, le bus d'adresse indique au périphérique sélectionné quel emplacement de mémoire ou quel registre[2] il doit utiliser. La taille du bus d'adresse détermine la quantité maximum de mémoire ou d'I/O que peut utiliser le microprocesseur.
- Le décodeur d'adresse détermine quelle mémoire ou interface sera lue ou écrite. Les périphériques non sélectionnés sont en haute impédance sur le bus de donnée (tri state).
- Le bus de données est un ensemble de lignes électriques permettant le transfert des données.
    - Un processeur 32 bits a un bus de donnée de 32 bits? Pas nécessairement : son bus interne est de 32 bits.
- Le bus de contrôle contient le reste ☺. Par exemple, il s'agit de toutes les lignes du microprocesseur lui permettant de gérer la direction et le temps des données sur le bus des données (lecture, écriture). Il s'agit également des lignes d'interruption qui permettent aux interfaces de signaler un évènement exceptionnel au microprocesseur.

2.1.4 Les Interfaces

- Les interfaces sont un ensemble de modules électroniques qui se branchent au microprocesseur et permettent au monde extérieur d'interagir avec le système microprocesseur.
- Les interfaces sont gérées de la même façon que la mémoire par rapport à la lecture et à l'écriture.

[2] Les périphériques aussi peuvent avoir des registres! Il s'agit toujours d'une variable spéciale dans une mémoire locale et très rapide ayant un rôle à jouer à l'intérieur du circuit intégré. Voir plus loin pour plus de détails…

- Des lignes de contrôle existent pour gérer certaines interfaces spécifiques.
- Les interfaces peuvent générer des interruptions (ex. touche de clavier)

2.2 Exécution de programme

Un système à microprocesseur fonctionne très simplement :

- Le microprocesseur lit une instruction de la mémoire
- Le microprocesseur exécute l'instruction
    - Certaines instructions demanderont au microprocesseur de lire ou d'écrire des données dans la mémoire. Habituellement, une donnée lue de la mémoire est mise dans un registre et la valeur écrite dans une donnée provient d'un registre.
    - Certaines instructions demanderont au microprocesseur d'effectuer une opération arithmétique ou booléenne. Habituellement, l'opération s'effectuera sur un ou plusieurs registres.
    - Certaines instructions demanderont au microprocesseur de changer la séquence d'exécution des instructions
    - Certaines instructions demanderont au microprocesseur d'accéder aux interfaces en spécifiant l'adresse de l'interface et la direction de l'échange.
- Le microprocesseur lit la prochaine instruction et il l'exécute…

2.2.1 Instructions et Jeux d'instructions

Chaque instruction exécutée par un microprocesseur a un opcode (code op en français!) et des paramètres. L'opcode décrit la nature de l'instruction, c'est-à-dire la tâche que doit réaliser le microprocesseur. Il dit si l'instruction est une addition ou soustraction, un branchement, un load, et ainsi de suite. Les paramètres, quant à eux, disent comment ou sur quelle donnée/registre doit s'effectuer la tâche. Les paramètres dépendent de l'opcode. Par exemple, une instruction d'addition pourrait avoir trois paramètres : les deux opérandes à additionner et le numéro du registre qui contiendra le résultat de l'addition.

Toutes les instructions sont des séquences de bit. Dans les architectures RISC (Reduced Instruction Set Computer), les instructions ont toute la même longueur, c'est-à-dire le même nombre de bits. Dans les architectures CISC (Complex Instruction Set Computer), les instructions ont des longueurs variables, qui dépendent de l'opcode. Enfin, dans les architectures hybrides, le processeur supporte des instructions ayant uniquement 2 longueurs différentes. De nos jours, presque tous les nouveaux microprocesseurs ont des architectures RISC ou hybrides, même si certains microprocesseurs (ceux d'Intel par exemple) semblent avoir des instructions de longueurs diverses : les instructions complexes (CISC) sont découpées en instructions de longueur fixe par le microprocesseur…

Si on suppose un microprocesseur ayant des instructions sur 32 bits, 128 instructions possibles ($128 = 2^7$) et 16 registres ($16 = 2^4$), alors, on pourrait découper certaines instructions ainsi :

Instruction	Opcode	Paramètres			
AddReg Dest, Source1, Source2	Add (7 bits)	Dest (4 bits)	Source1 (4 bits)	Source2 (4 bits)	Extra (13 bits)
LOAD Reg, Memoire	Load R/M (7 bits)	Reg (4 bits)	Adresse de mémoire (21 bits)		
JMP Adresse	JMP (7 bits)	Adresse de la prochaine instruction relative au compteur de programme (PC) (25 bits)			

2.3 Bus et accès à la mémoire/périphérique

Pour lire une instruction ou une donnée de la mémoire, le microprocesseur effectue habituellement les tâches suivantes :

1) Mettre l'adresse de l'instruction ou de la donnée sur le bus d'adresse.
2) Activer le signal de lecture de la mémoire. Ce signal est un signal du bus de contrôle.
3) Attendre que la mémoire mette l'instruction ou la donnée sur le bus de donnée.
4) Lire l'instruction ou la donnée sur le bus de données.

Pour écrire une donnée de la mémoire, le microprocesseur effectue habituellement les tâches suivantes :

1) Mettre l'adresse de la donnée sur le bus d'adresse.
2) Mettre la valeur à écrire sur le bus de données
3) Activer le signal d'écriture de la mémoire. Ce signal est un signal du bus de contrôle.
4) Attendre que la mémoire ait sauvegardé la donnée provenant du bus de données.

La lecture ou l'écriture de périphérique se fait habituellement de la même façon que la lecture ou l'écriture de données en mémoire.

Une plage d'adresses est attribuée à chaque mémoire et à chaque périphérique. Ces adresses servent à désigner une cellule de mémoire ou les registres d'un périphérique donné.

Dans la plupart des nouveaux microprocesseurs, chaque adresse est unique. Par exemple, les adresses 0 à 1023 sont attribuées à la ROM, les adresses 8192 à 16383 sont réservées à la RAM et les adresses 65536 à 65663 sont attribuées aux périphériques.

Pour certains systèmes microprocesseurs, les adresses sont répétées. Il peut y avoir plusieurs adresses 0 : il peut y avoir l'adresse 0 pour la mémoire d'instruction et l'adresse 0 pour la mémoire de données. Il peut également y avoir l'adresse 0 pour la mémoire interne et l'adresse 0 pour la mémoire externe. Il peut aussi y avoir l'adresse 0 pour la mémoire et l'adresse 0 pour les périphériques. Il peut également y avoir les adresses 0

pour les registres et pour le reste. Enfin, il peut y avoir des adresses 0 pour toutes les combinaisons possibles de ces éléments.

Lorsqu'un microprocesseur a plusieurs plages d'adresses différentes, le microprocesseur supporte habituellement des instructions différentes qui permettent d'accéder à chacune de ces plages d'adresses. Par exemple, s'il existe des adresses de 0 à 0xFFFFF pour la mémoire et de 0 à 0xFF pour les périphériques une instruction permettra d'accéder à la mémoire (MOV par exemple) et une autre instruction permettra d'accéder aux périphériques (IN par exemple).

Les adresses réservées aux périphériques sont communément appelées "ports" dans la littérature.

2.4 Interruptions

Pour signaler un évènement, les périphériques utilisent souvent des interruptions : une tension prédéterminée est appliquée sur une broche du microprocesseur et cette tension, une valeur digitale, signale une interruption.

Un contrôleur d'interruption est souvent ajouté au système microprocesseur afin de gérer les interruptions de plusieurs périphériques. Le contrôleur d'interruption reçoit les signaux de plusieurs périphériques et active la broche d'interruption du microprocesseur au besoin. Le contrôleur peut déterminer la priorité des interruptions ou masquer celles-ci.

Une interruption est dite "masquée" lorsqu'elle est désactivée. Le mot "masquer » est employé parce que le signal d'interruption est conservé de telle sorte que l'interruption se produira immédiatement après la réactivation de l'interruption, si l'interruption a eu lieu pendant qu'elle était désactivée.

Une interruption non-masquable (NMI) est une interruption qui ne peut être désactivée.

Les microprocesseurs ont habituellement plusieurs broches d'interruption.

Lors d'une interruption, le microprocesseur exécute habituellement les actions suivantes :

- Termine l'instruction (ou les instructions !) en cours
- Détermine ou obtient du contrôleur d'interruption le numéro et la priorité de l'interruption.
- Vérifie si l'interruption peut être traitée : l'interruption peut être exécutée si elle n'est pas masquée et si une interruption de priorité supérieure ou égale n'est pas déjà traitée.
- Le microprocesseur détermine l'adresse de la routine/fonction qui sert à traiter l'interruption (ISR = Interrupt SubRoutine = routine exécutée lorsque l'interruption se produit).

- Le microprocesseur sauvegarde l'adresse de l'instruction en cours d'exécution pour savoir où reprendre l'exécution quand l'ISR sera terminée.
- Le microprocesseur sauvegarde certains registres et drapeaux automatiquement (optionnel, la sauvegarde se fait sur la pile).
- Le microprocesseur effectue un saut à l'adresse de l'ISR.

Les interruptions ont des priorités (voir la table ci-dessous) et des numéros. Une interruption peut interrompre une autre interruption si elle est de priorité plus élevée.

Il est possible de gérer les interruptions par interrogations successives (polling) plutôt qu'avec une ligne dédiée: à intervalle régulier le système demande à tous les périphériques s'ils ont un évènement à signaler.

2.5 Microcontrôleurs

Pour limiter le nombre de circuits intégrés nécessaires dans un système microprocesseur, il est courant d'ajouter, sur le même die que le microprocesseur, de la mémoire et certains périphériques. Le circuit intégré résultant est un microcontrôleur (microcontrôleur = microprocesseur + mémoires + interfaces/périphériques).

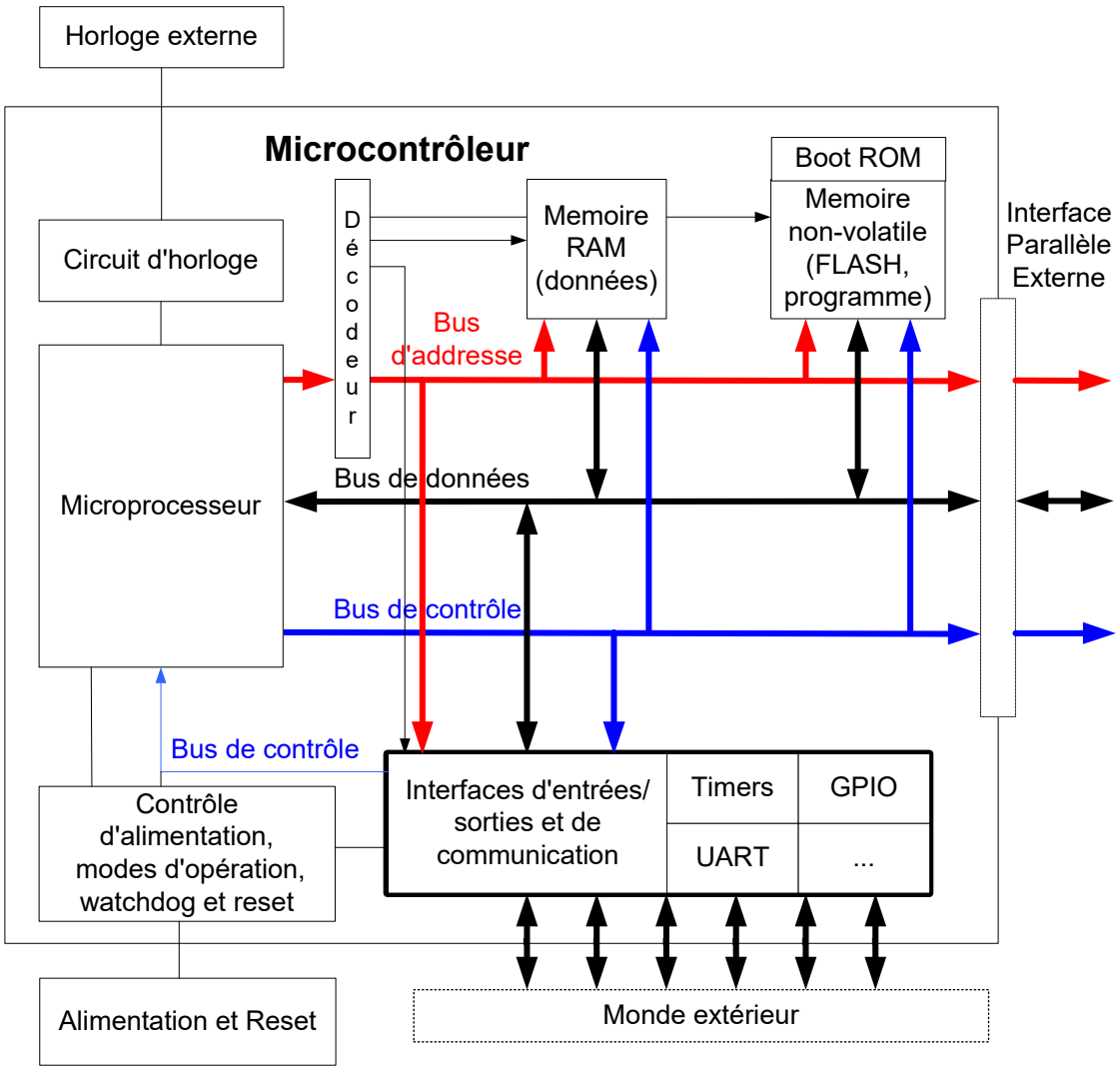


Figure 3 - Microcontrôleur

Un microcontrôleur contient habituellement de la mémoire RAM (parfois en très petite quantité -i.e. 32 bytes- mais souvent en quantité beaucoup plus importante), presque toujours de la mémoire ROM (parfois aussi en très petite quantité –i.e. 2 kilobytes-, mais souvent en quantité beaucoup plus importante).

Au démarrage, le microcontrôleur exécute presque toujours du code en ROM. Souvent, la mémoire ROM est divisée en deux partie : une partie non-programmable (ROM, PROM) et une partie reprogrammable (FLASH). La partie de code en mémoire non-reprogrammable (Bootrom ou Bootloader) est un programme exécuté au démarrage du microcontrôleur qui permet habituellement la reprogrammation du microcontrôleur, c'est-à-dire l'écriture de la partie reprogrammable de la mémoire non-volatile.

Le Bootrom a un rôle similaire au rôle du BIOS dans les ordinateurs de bureau.

Habituellement, le Bootrom cède automatiquement le contrôle au programme de l'usager dans la FLASH (saut inconditionnel à une adresse prédéterminée) à moins de détecter un niveau de tension donné (signal digital) sur une broche spécifique du microprocesseur.

> *Pour plusieurs microcontrôleurs, les tensions retrouvées sur certaines broches spécifiques du microcontrôleur déterminent la mémoire contenant les premières instructions exécutées. Ne pas démarrer systématiquement avec le bootrom permet un démarrage plus rapide.*

Le programme de l'usager peut alors accéder à plusieurs interfaces ou périphériques propres au microcontrôleur. À partir d'instructions précises exécutées par le microprocesseur, il peut, par exemple, envoyer un octet de donnée sur le port série (UART) du microcontrôleur ou lire une tension digitale sur une broche du microcontrôleur.

Habituellement, du point de vue du programmeur, les périphériques du microcontrôleur sont accessibles sous la forme de registres. Chaque périphérique dispose d'un ensemble de registres qui dirigent son opération. Par exemple, un registre du périphérique UART déterminera la vitesse à laquelle communiquera le microcontrôleur sur le port série.

> *Les registres sont des variables spéciales ayant un rôle spécial à jouer à l'intérieur d'un circuit intégré. Le microprocesseur a des registres, la mémoire peut contenir des registres, les périphériques ont des registres… Il s'agit habituellement d'un ensemble de bascules D qui conservent leurs valeurs jusqu'à ce qu'un coup d'horloge de la bascule (une écriture de registre) vienne les changer.*

Plusieurs microcontrôleurs possèdent une interface permettant de communiquer avec une mémoire parallèle externe. Cependant, cette interface est parfois absente parce qu'elle requiert beaucoup de broches.

2.5.1 Microcontrôleur vs microprocesseur

Microprocesseur :

- Avantages: Puissance, versatilité, flexibilité
- Inconvénients: glue logique requise, besoin d'interfacer manuellement, puis de programmer l'accès à ces interfaces pour le reste du code (pilotes de périphériques), fiabilité diminuée, conception matérielle et logicielle nettement plus difficile.
- Utilisé dans tous vos ordinateurs, avec des systèmes d'exploitation!

Microcontrôleur :

- Avantages: tout en un, fiabilité, faible coût et faible espace, simplicité générale, développement rapide, beaucoup de choix.
- Inconvénients: prix payé pour des fonctions que l'on n'utilise pas dans une application souvent dédiée à une tâche unique. Idem avec l'énergie utilisée.
- Utilisé dans toutes les applications embarquées!

De nos jours, les microprocesseurs pour ordinateur de bureau tendent à intégrer des périphériques (carte graphique par exemple) alors que les microcontrôleurs sont de plus en plus puissants.

2.6 Digital Signal Processor (DSP) et Architecture Harvard

Une vaste gamme de microcontrôleurs ou microprocesseurs existent pour faire du traitement de signal (audio/vidéo principalement). Ces micros ont une architecture particulière et possèdent des instructions spéciales pour accélérer les calculs (MAC = Multiply-ADD pour accélérer les FFTs!). En fait, les DSPs ont une architecture Harvard qui permet de lire les instructions et les données simultanément :

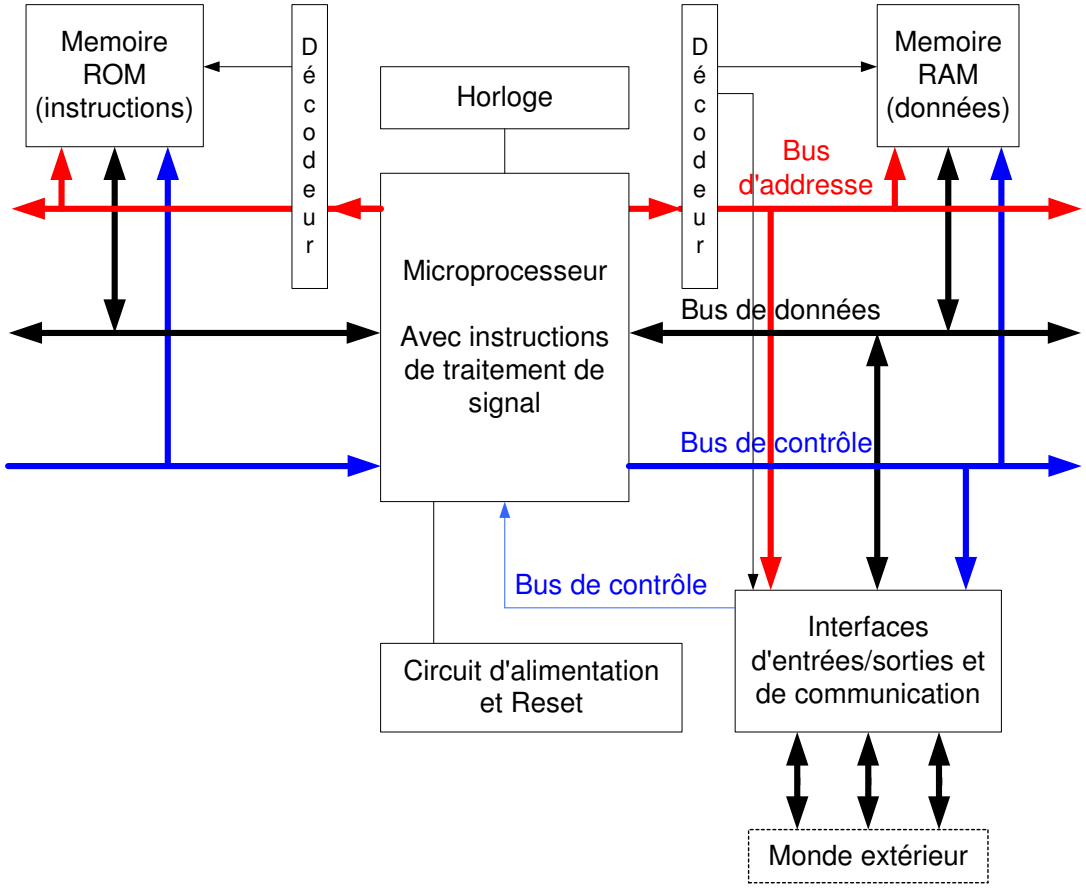
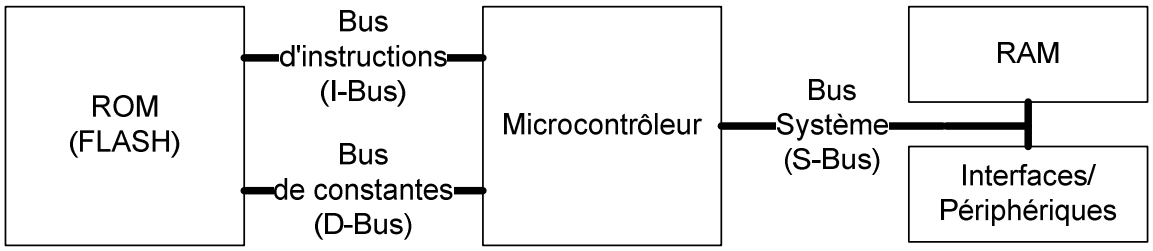


Figure 4 – DSP, Architecture Harvard

L'architecture Harvard se retrouve dans plusieurs microprocesseurs et microcontrôleurs : son application ne se limite pas aux DSPs ! En effet, utiliser deux bus différents pour accéder aux instructions et aux données a plusieurs avantages : les deux accès peuvent se faire simultanément (voir pipeline d'instructions), le DMA peut se faire pendant que le microprocesseur lit ses instructions (voir prochaine section), le temps de latence nécessaire pour répondre à une interruption est plus court (voir les prochains cours), etc.

Beaucoup de microprocesseur ayant une architecture Harvard ont trois bus plutôt que deux, surtout lorsqu'ils ont un pipeline d'instruction (ce qui est le cas de presque tous les

micros modernes !). Il y a un bus additionnel pour accéder aux constantes dans la mémoire ROM (FLASH) :



Le bus de constantes permet d'éviter des stalls automatiques de pipeline lorsque le microprocesseur lit des données en mémoire FLASH (une table de constantes par exemple). Sans ce bus, une instruction comme *LOAD Rx, [Mémoire FLASH]* (lire une adresse de la mémoire FLASH et mettre la valeur lue dans le registre Rx du microcontrôleur) utiliserait deux fois le bus d'instructions (I-Bus) et ne pourrait s'exécuter en même temps que la lecture d'une autre instruction.

2.7 DMA

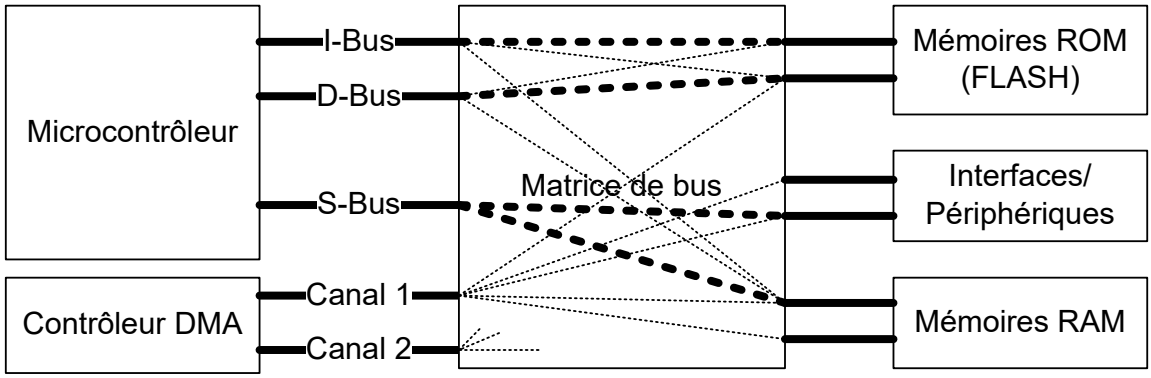
Dans certains systèmes microprocesseurs, un ou des coprocesseurs exécutent des tâches spécialisées à la demande des programmes. Ces coprocesseurs, contrôlés par le microprocesseur (qui est lui-même contrôlé par les instructions de vos programmes), permettent de libérer le processeur principal pour qu'il puisse réaliser d'autres tâches.

La tâche la plus commune pouvant être exécutée par un coprocesseur est le transfert de données d'un périphérique vers la mémoire ou vice-versa. On retrouve donc, dans presque tous les systèmes microcontrôleurs (sauf pour les moins dispendieux), un contrôleur de DMA (Direct Memory Access). Le contrôleur de DMA, dont les registres sont programmés et configurés par l'usager, transfert des données d'un périphérique à la mémoire (ou vice-versa), sans intervention du microprocesseur.

Habituellement, le transfert par DMA s'effectue sur une plage contigüe de mémoire. Le contrôleur de DMA communique les octets de données un par un (ou deux par deux, voire même plus!) jusqu'à ce que toute la plage de mémoire soit lue ou écrite.

2.8 Matrice de Bus

Dans la plupart des systèmes microprocesseurs avec DMA, il existe des bus additionnels entre le contrôleur de DMA, la mémoire et les périphériques pour que le microprocesseur puisse accéder à la mémoire de donnée pendant le transfert par DMA.



Il existe une multitude de connections possibles entre les diverses composantes des systèmes microcontrôleurs modernes. La matrice de bus permet :

- D'éviter de nombreux stalls de pipeline lorsque le microcontrôleur accède à des régions de mémoire différentes.
- De permettre l'exécution d'instructions à partir de plusieurs régions de la mémoire (si un programme est copié en mémoire RAM par exemple).
- De permettre le transfert par DMA sans nuire à l'exécution d'instructions par le microcontrôleur.
- D'avoir plusieurs bus pour les périphériques, dont certains réservés aux périphériques rapides (USB, Ethernet par exemple) et d'autres réservés aux périphériques plus lents.

3 Composantes analogiques pour circuits numériques

Voir Composantes analogiques de circuits numériques.ppt.

4 Format des données

Dans un ordinateur, toutes les données (nombre entiers, fraction, caractère d'imprimerie) sont représentées en binaire (0 ou 1). Des conventions permettent de traduire des séquences de bits (010100011…) en symboles numériques (123, 4.5) ou alphabétiques ('A','-').

Habituellement les séquences de bits ont une taille finie : celle d'un mot. Un ***mot***, en informatique, définit le nombre de bits qu'il faut pour représenter une donnée ou une instruction.

En informatique, un ***octet*** (byte) est une séquence de 8 bits. Un ***short***, un ***int***, ou un ***long*** sont des séquences de bits plus longues que 8bits (de plus en plus longues) et multiples de 8 bits. Par exemple, un short sera 16 bits, un int sera 32 bits et un long sera 64 bits.

4.1 Représentation d'entiers positifs et hexadécimal

Lorsqu'un entier positif est représenté par une séquence de N bits, chaque bit vaut une puissance de 2 de l'entier positif. Par exemple, 10010011_b (le $_b$ indique qu'il s'agit de binaire) représentera $1*2^7 + 0*2^6 + 0*2^5 + 1*2^4 + 0*2^3 + 0*2^2 + 1*2^1 + 1*2^0$, soit 147_d (le $_d$ indique qu'il s'agit de décimal), de la même façon que 234_d représente $2*10^2 + 3*10^1 + 4*10^0$.

En informatique, les nombres sont souvent écrits en hexadécimal (avec une base 16, avec des nombres allant de 0 à F où F vaut 15_d). Cela permet de remplacer 4 bits par un seul symbole hexadécimal ($2^4 = 16$). Par exemple, la valeur 1101_b sera remplacée par D_h ou *0x*D (le $_h$ or le *0x* indique qu'il s'agit d'hexadécimal).

Pour passer de l'hexadécimal au décimal, il suffit de multiplier les digits hexadécimaux par leur puissance de 16 correspondante. Pour passer du décimal à l'hexadécimal, il faut diviser les digits décimaux par 16 plusieurs fois jusqu'à ce qu'il n'y ait plus de restes et générer le nombre hexadécimal à partir des restes.

4.2 Représentation d'entiers signés

Pour qu'une séquence de bits représente un entier signé, il faut établir une convention. Il faut décider comment les bits indiqueront que le nombre est négatif.

Deux conventions principales existent : 1) le bit de signe et 2) la notation complément 2.

Lorsqu'un bit de signe est utilisé, le bit le plus significatif du mot indique si le nombre est positif ou négatif (habituellement, un '1' indique un nombre négatif). Par exemple 10010011_b, indiquera -19 (le 1 le plus à droite –le plus significatif- indique un nombre négatif, 0010011_b indique 19).

En notation complément 2, le bit le plus significatif du mot a une valeur négative. Si le mot a N bits, ce bit vaudra $-2^{(N-1)}$. Par exemple, 10010011_b indiquera -128 + 19, soit -109.

La notation complément 2 est très largement répandue parce que les additions et les soustractions se font toujours de la même manière (la manière traditionnelle!), peu importe le signe des nombres additionnés ou soustraits. Par exemple, si on soustrait 0x24 de 0xC3 (0xC3 – 0x24), on obtient 0x9F : le résultat (-61 - 36 = -97) est bon même si le premier opérande est négatif, tout comme la réponse.

4.3 Représentation de fractions

Pour qu'une séquence de bits représente une fraction, il faut aussi établir une convention.

Encore une fois, deux conventions principales existent : la virgule flottante et la norme IEEE754. Il existe cependant une multitude de conventions moins universelles.

4.3.1 Virgule flottante

Dans la virgule flottante, une certaine quantité de bits du mot représentent une valeur entière et le restant des bits représentent la partie fractionnaire de la fraction. Par exemple, 10010.011_b pourrait indiquer 18.375 si on décide que les bits valent, de gauche à droite 2^4, 2^3, 2^2, 2^1, 2^0, 2^{-1}, 2^{-2}, 2^{-3}. D'autres conventions sont possibles!

4.3.2 Norme IEEE754

Selon la norme IEEE754, les fractions sont représentées sur 32 bits (float) ou 64 bits (double). Sur 32 bits, la plupart des fractions sont représentés ainsi:

- Le bit le plus significatif est un bit de signe (S)
- Les 8 bits qui suivent, sont des bits d'exposant (E, un entier positif sur 8 bits)
- Les 23 bits restants forment la mantisse (M). Le bit le plus significatif de la mantisse vaut 2^{-1} et le moins significatif vaut 2^{-23}.
- S EEEEEEEE MMMMMMMMMMMMMMMMMMMMMMM
- Fraction = $(-1)^S * 2^{E-127} * (1+\text{mantisse})$

La norme IEEE754 permet de représenter certaines valeurs spéciales lorsque E vaut 0 ou 255.

```
S EEEEEEEE FFFFFFFFFFFFFFFFFFFFFFF
0 1      8 9                     31
```

The value V represented by the word may be determined as follows:

- If E=255 and F is nonzero, then V=NaN ("Not a number")
- If E=255 and F is zero and S is 1, then V=-Infinity
- If E=255 and F is zero and S is 0, then V=Infinity
- If 0<E<255 then V=(-1)**S * 2 ** (E-127) * (1.F) where "1.F" is intended to represent the binary number created by prefixing F with an implicit leading 1 and a binary point.
- If E=0 and F is nonzero, then V=(-1)**S * 2 ** (-126) * (0.F) These are "unnormalized" values.
- If E=0 and F is zero and S is 1, then V=-0
- If E=0 and F is zero and S is 0, then V=0

Exemples:

```
0 10000000 00000000000000000000000 = +1 * 2**(128-127) * 1.0 = 2
0 10000001 10100000000000000000000 = +1 * 2**(129-127) * 1.101 = 6.5
1 10000001 10100000000000000000000 = -1 * 2**(129-127) * 1.101 = -6.5
```

Norme IEEE754, 32 bits, Tiré de
http://www.psc.edu/general/software/packages/ieee/ieee.html

4.3.3 Autres représentations

Il existe autant de façons de représenter des fractions que vous pouvez en imaginer. Lorsqu'on représente une fraction selon un format "maison", il faut tenir compte de plusieurs facteurs : le nombre de bits disponibles pour représenter la fraction, les valeurs min et max, la précision requise, le temps de calcul avec les fractions, l'affichage pour l'humain, etc…

Voici deux exemples pour illustrer les propos ci-dessus :

Exemple 1 : Le protocole NEMA des GPS (NMEA 0183) encode toutes les valeurs en ASCII pour que ces messages soient lisibles d'un terminal quelconque. Ainsi, le nombre 372.1 sera encodé avec 5 caractères : '3', '7', '2', '.' et '1'. Vous retrouverez, pour ce nombre, les octets 0x33, 0x37, 0x32, 0x2E et 0x31 en mémoire…

Exemple 2 : Vous avez une sonde de température qui mesure des températures allant de -30ºC à 70 ºC et la sonde a une précision de 0.1 ºC. Malheureusement, toutes les lectures de la sonde de température doivent être entreposées sur 10 bits seulement (de 0 à 1023). Solution : vous décidez que 0 vaut -30 ºC, que 1 vaut -29.9ºC… et que 70ºC est représenté par la valeur 1000!

4.4 Représentation de caractères

Pour représenter des caractères alphabétiques avec des bits, on utilise une table de correspondance. Par exemple les bits 0110 0101 pourrait signifier 'A'.

Il existe deux tables de correspondances très répandues : l'ASCII et l'Unicode. L'ASCII est une table où chaque caractère est encodé sur 8 bits. La taille d'un fichier texte est petite, mais seulement 256 caractères sont représentés. La table de l'Unicode, quant à elle, est plus grosse et plus universelle: chaque caractère est représenté sur 16 bits ou plus.

4.5 Données en mémoire ou transfert de données

Lorsque l'on sauvegarde une donnée sur plusieurs octets dans une mémoire ayant des mots de 8 bits, il faut souvent choisir l'ordre des octets. Les octets les plus significatifs (ceux qui valent le plus) peuvent être placés en dernier ou en premier dans la mémoire.

Les termes Little Endian (petit boutiste en français) et Big Endian (gros boutiste) indiquent l'ordre des données sur plusieurs octets en mémoire. Une donnée sauvegardée en Little Endian aura ses octets les plus significations aux adresses de mémoire les plus hautes. Une donnée sauvegardée en Big Endian aura ses octets les plus significations aux adresses de mémoire les plus basses.

> *Les occidentaux que nous sommes visualisent les nombres en Big Endian conventionnellement. Ainsi, la valeur 0x1234 aura l'octet 0x12 à l'adresse 0 et l'octet 0x34 à l'adresse 1.*