## 1. Architecture d'un processeur

Un processeur est un composant intégré (en technologie C-MOS) ou une partie de composant susceptible d'exécuter des instructions selon un programme d'instructions préétabli sur des données numériques. Le programme, et parfois les données, sont enregistrés dans des mémoires numériques.

L'architecture est ce qui détermine d'emblée et de manière définitive les principales caractéristiques du processeur en particulier la rapidité d'exécution des instructions. Elles sont liées au nombre de bus internes, qui sont des liaisons parallèles à *N* bits, à la valeur même de *N,* aux opérations possibles sur les données transitant sur les bus et enfin à la puissance de calcul de l'unité centrale.

Les bus sont reliés à des registres temporaires pour stocker provisoirement les codes d'instructions, des adresses ou des données. D'autres types de registres existent:

- ✓ les registres de contrôle ou d'état, qui ne sont pas stockés en mémoire car ils sont adressés directement par une instruction, par mesure de sécurité. Ces registres sont essentiels car ils déterminent le déroulement précis d'une instruction. Certains d'entre eux sont en lecture seule, d'autres en écriture seule.
- ✓ les registres définissant les conditions de fonctionnement d'un port (série, DMA, hôte ...), des interfaces avec des mémoires de sortie ou du *Timer*, qui ont une adresse en mémoire de données.

Les instructions s'exécutant au cœur du processeur appelé *unité centrale,* comportant dans le cas le plus classique, une seule *unité arithmétique de traitement des données*. À l'entrée des unités de calcul constituant l'unité centrale,

les données sont chargées dans des registres temporaires qui, lors de la réalisation de l'instruction de calcul, vont être reliés à l'unité de calcul ; le résultat de l'opération sera disponible dans un registre (temporaire) de sortie. L'architecture est conçue pour que les instructions puissent :

- ✓ Charger les données dans un registre temporaire d'entrée en lisant le contenu d'une mémoire vive à une adresse donnée, les données transitant par un bus de données et les adresses pour la lecture par un autre bus ;
- ✓ Effectuer une opération logique ou arithmétique sur une donnée ou entre deux données dans l'unité centrale;
- ✓ Effectuer des tests sur le résultat obtenu (dépassement, signe...) ;
- ✓ Modifier certains registres ;
- ✓ Aller écrire un résultat à une adresse de la mémoire prévue.

Toutes ces instructions doivent en principe être exécutées en *un seul cycle d'horloge.*

L'architecture contient donc un noyau DSP si elle comporte une unité centrale laquelle est composée d'une ou plusieurs unités de traitement, chacune comportant des unités de calcul, dont l'ALU *(Arithmetic and Logic Unit)* le MAC *(Multiplier and ACcumulator),* le décaleur à barillet *(Barrel Shifter)* et enfin des multiplexeurs d'aiguillage des données. En plus de l'unité centrale, le noyau DSP comporte un *séquenceur,* pour envoyer les adresses des instructions enregistrées dans la mémoire programme, et un ou plusieurs *générateurs d'adresses* (parfois appelés pointeurs) agissant sur les bus d'adresses.

## 2. Architecture de Von Neumann et de Harvard

À chaque cycle d'horloge, le processeur sait par le compteur de programme l'instruction qu'il doit faire exécuter. Nous avons vu qu'il va chercher chaque instruction en mémoire, l'exécute avec les données correspondantes et retourne les données résultantes en mémoire.

Dans l'architecture de la machine de *Von Neumann,* le programme et les données sont enregistrés sur la même mémoire. Chaque instruction contient la commande de l'opération à effectuer et l'adresse de la donnée à utiliser, il faut donc souvent plusieurs cycles d'horloge pour exécuter une instruction. La *Figure 3.1* indique une architecture simple de Von Neumann, constituée d'un bus de données et de programme et d'un bus d'adresses.

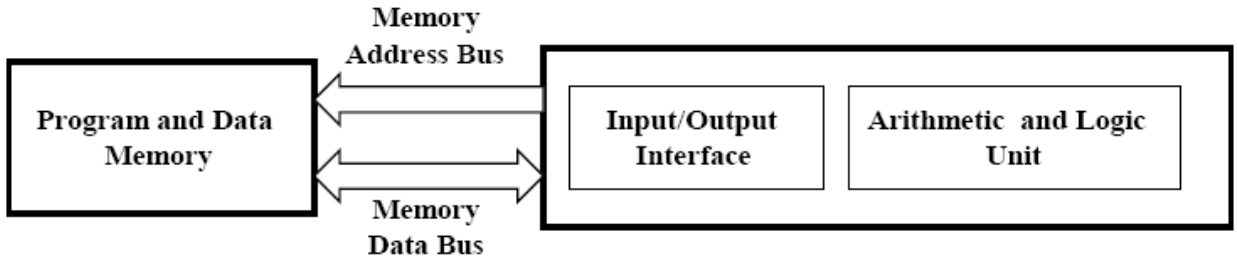


*Figure 3.1 : Architecture de Von Neumann*

On voit que les échanges s'effectuent de manière simple entre l'unité arithmétique et logique (ALU), c'est-à-dire l'unité centrale et la mémoire unique, par un bus transitant les codes de programme et les données. On a ainsi des données « collées » aux instructions. Les microprocesseurs et beaucoup de microcontrôleurs utilisent cette architecture car elle est très souple pour la programmation.

Dans l'architecture dite *de Harvard* (car mise au point dans cette université américaine en 1930), on sépare systématiquement la mémoire de programme de la mémoire des données : l'adressage de ces mémoires est indépendant. La *Figure 3.2* indique une architecture simple de Harvard, constituée d'un bus de données, d'un bus de programme et de deux bus d'adresse.

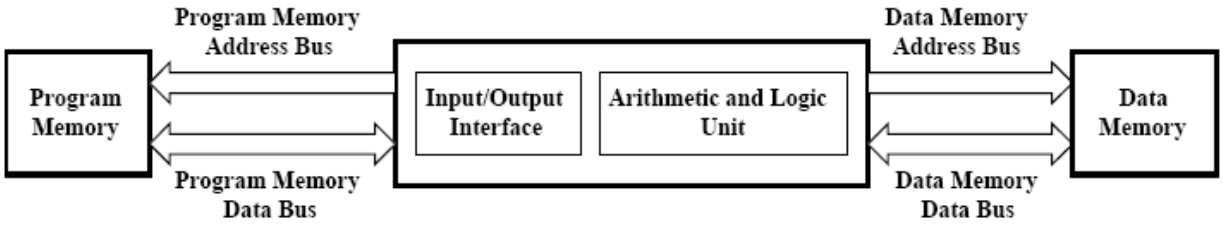


*Figure 3.2 : Architecture de Harvard*

On voit que les échanges s'effectuent de manière double entre l'unité centrale et les deux mémoires, ce qui permet une grande souplesse pour l'enregistrement et l'utilisation des données. D'ailleurs, la mémoire de programme est également utilisée en partie comme mémoire de données pour obtenir encore plus de possibilités de traitement avec des algorithmes complexes.

L'architecture généralement utilisée par les microprocesseurs est la structure Von Neuman (exemples : la famille Motorola 68XXX, la famille Intel 80X86). L'architecture Harvard est plutôt utilisée dans des microprocesseurs spécialisés pour des applications temps réels, comme les DSP.

Il existe cependant quelques rares DSP à structure Von Neuman. La raison de ceci est liée au coût supérieur de la structure de type Harvard. En effet, elle requiert deux fois plus de bus de données, d'adresses, et donc de broches sur la puce. Or un des éléments augmentant le coût de productions des puces est précisément le nombre de broches à implanter.

Pour réduire le coût de la structure Harvard, certains DSP utilisent l'architecture dite « Structure de Harvard modifiée ». À l'extérieur, le DSP ne propose qu'un bus de données et un bus d'adresse, comme la structure Von Neuman. Toutefois, à l'intérieur, la puce DSP dispose de deux bus distincts de données et de deux bus distincts d'adresses. Le transfert des données entre les bus externes et internes est effectué par multiplexage temporel.

## 3. Utilisation de pipelines

Pour améliorer les performances de l'unité de traitement, les DSP les plus récents utilisent la méthode du pipeline. Elle consiste à imposer un ordre et un rythme dans le déroulement des instructions de manière à optimiser en rapidité leur exécution. En un cycle processeur, les opérations élémentaires suivantes peuvent être exécutées en parallèle :

1. Aller chercher l'instruction en mémoire programme (*Fetch*) ;
2. Réaliser le décodage de l'instruction, et des adresses des opérandes (*Decode*) ;
3. Lire les opérandes en mémoire de données (*Read*) ;
4. Exécuter l'opération et écrire le résultat (*Execute*).

Le principe de pipeline consiste à découper le travail en tâches élémentaires de même durée pour permettre leur réalisation en parallèle. Il faut prévoir des registres tampon entre chaque opération élémentaire, ce qui montre le *Tableau 3 .1*. Il y a donc en permanence quatre instructions dans le pipeline, ce qui est le plus souvent invisible, sauf s'il y a « conflit » dans le pipeline, ce qui se passe quant l'une des quatre instructions exige plus d'un cycle d'horloge. Dans la plupart des cas, DSP « rattrape » le conflit en décalant alors les autres instructions d'un cycle.

| Instruction | t | t+1tclk | t+2tclk | t+3tclk | t+4tclk | t+5tclk | t+6tclk |
|---|---|---|---|---|---|---|---|
| **n** | Fetch | Decode | Read | ***Execute*** | … | … | … |
| **n+1** | | Fetch | Decode | Read | ***Execute*** | … | … |
| **n+2** | | | Fetch | Decode | Read | ***Execute*** | … |
| **n+3** | | | | Fetch | Decode | Read | ***Execute*** |

*Tableau 3.1 : Principe de pipeline*