

LECON 3 : LES MICROPROCESSEURS

I. OBJECTIFS

À la fin de cette leçon, vous serez capable de :

- Décrire la structure et la composition d'un microprocesseur;
- Décrire les différentes architectures de microprocesseur;
- Décrire les caractéristiques d'un microprocesseur ;
- Décrire le fonctionnement du microprocesseur;

II. INTRODUCTION AUX MICROPROCESSEURS

1. Définition d'un microprocesseur

Un microprocesseur ou CPU (Central Processing Unit) est l'organe de l'ordinateur qui permet d'exécuter les instructions des programmes informatiques en mémoire. C'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 représente le cerveau ou moteur de l'ordinateur.

Un microprocesseur est un processeur dont la taille a été miniaturisée pour que les composants soient réunis en un seul circuit intégré. Le terme **microprocesseur** et **processeur** sont donc équivalents.

A l'heure actuelle, un microprocesseur regroupe sur quelques millimètres carrés 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*.

2. Fonctionnement d'un micro-processeur

Le **processeur** (noté **CPU**, pour *Central Processing Unit*) est un circuit électronique cadencé au rythme d'une horloge interne, grâce à un cristal de quartz qui, soumis à un courant électrique, envoie des impulsions, appelées « **top** ». La **fréquence d'horloge** (appelée également **cycle**, correspondant au nombre d'impulsions par seconde, s'exprime en Hertz (Hz). Ainsi, un ordinateur à 1 GHz possède une horloge envoyant 1 000 000 000 de battements par seconde. Une instruction qui requiert 6 cycles d'horloge sera donc exécutée en un temps de:

$$6 \times (1/1\,000\,000\,000) = 6 \text{ ns } (6 \times 10^{-9})$$

La fréquence d'horloge est généralement un multiple de la fréquence du système (*FSB, Front-Side Bus*), c'est-à-dire un multiple de la fréquence de la carte mère.

A chaque top d'horloge le processeur exécute une action, correspondant à une instruction ou une partie d'instruction. L'indicateur appelé **CPI** (*Cycles Par Instruction*) permet de représenter le nombre moyen de cycles d'horloge nécessaire à l'exécution d'une instruction sur un microprocesseur. La puissance du processeur peut ainsi être caractérisée par le nombre d'instructions qu'il est capable de traiter par seconde. L'unité utilisée est le **MIPS** (Millions d'Instructions Par Seconde) correspondant à la fréquence du processeur que divise le *CPI*.

3. Caractéristiques

Deux paramètres principaux permettent de caractériser un microprocesseur :

- son architecture
- sa fréquence
- taille des registres généraux : 32 ou 64 bits
- niveaux de cache : L1, L2, L3
- nombre de cœurs
- technologies de l'HyperThreading, de Virtualisation, turbo boots, pipeline, etc.
- jeu d'instructions SSE, AVX
- La puce graphique : la plupart des processeurs de nos jours embarquent une puce HD Graphics 2000 ou 3000. Ce GPU intégré suffit à toutes les tâches bureautiques et même à des jeux.
- condition de limiter la résolution.
- ...

Ces 2 facteurs principaux (architecture, fréquence) conditionnent les performances de la machine qui sont mesurées :

- soit en MIPS (million of instructions per second) : pour les instructions sur les entiers
- soit MFLOPS (million of floating point operations per second) : pour les calculs sur les réels
- soit en secondes par l'intermédiaire d'applications spécifiques (bureautique, jeux, multimédia, ...)

Processeurs 32 bits et 64 bits

Un processeur est dit 32 bits ou 64 bits si la largeur de ses registres généraux a une taille de 32 ou 64 bits. Un saut technologique avait déjà été franchi lors de l'apparition du Intel 80386 qui permettait le passage du 16 bits au 32 bits.

Qu'en est-il des performances de l'architecture 64 bits ?

- si on exécute un programme 32 bits dans une architecture 64 bits, on obtient normalement les *mêmes performances* qu'en 32 bits, ce qui semble normal.
- par contre, si on recompile un programme en 64 bits, *il s'exécute plus rapidement en mode 64 bits que sa version 32 bits*. En effet :
 - d'une part le code est moins long (i.e. comprend moins d'instructions, même si généralement la taille du programme est plus longue)
 - et d'autre part, il est naturellement optimisé par le fait que l'on peut utiliser plus de registres et donc faire moins souvent appel à la mémoire en étant contraint de sauvegarder temporairement des données ou de les recharger (cf. produit de matrices en assembleur).
- Les processeurs 32 bits ne peuvent normalement pas adresser plus de 4 Gio (2^{32} octets) de mémoire centrale, tandis que les processeurs 64 bits peuvent en adresser 16 Eio (2^{64} octets). C'est pourquoi dès qu'il y a plus de 4 Gio de RAM sur une machine, la mémoire au-delà de ce seuil ne sera directement adressable qu'en mode 64 bits.

AMD fut le premier à introduire la technologie 64 bits en 2003 avec ses processeurs Athlon 64 (architecture K8). Les Athlon 64 ont pour caractéristique de pouvoir exécuter des programmes aussi bien en 32 bits qu'en 64 bits. On parle également d'architecture x86-64.

Remarque :

Des tests ont montré que pour certains programmes, le passage en 64 bits permet de diminuer de 2/3 le temps d'exécution par rapport au 32 bits avec les Core 2 Duo.

4. Evolution des micro-processeurs

En 1969, le microprocesseur a été inventé par un ingénieur et un physicien d'Intel : Marcian Hoff (surnommé Ted Hoff) et Federico Faggin. Marcian Hoff a formulé l'architecture du microprocesseur (une architecture de bloc et un jeu d'instructions). Le premier microprocesseur commercialisé, le 15 novembre 1971, est l'Intel 4004 4 bits, suivi par l'Intel 8008 à 8 bits et qui servi initialement à fabriquer des contrôleurs graphiques en mode texte. Jugé trop lent par le client qui en avait demandé la conception, il devint un processeur d'usage général.

Ces processeurs sont les précurseurs des Intel 8080, Zilog Z80, et de la future famille des Intel x86. Federico Faggin est l'auteur d'une méthodologie de conception nouvelle pour la puce et la logique,

fondée pour la première fois sur la technologie *silicon gate* développé par lui en 1968 chez Fairchild. Il a aussi dirigé la conception du premier microprocesseur jusqu'à son introduction sur le marché en 1971.

Le leader incontesté sur le marché des microprocesseurs pour le standard PC malgré est Intel, concurrencé par AMD (Advanced Micro Devices), compagnie américaine a été fondée en 1969.

Voici ci-dessous l'évolution des processeurs INTEL :

Date	Nom	Nombre de transistors	Finesse de gravure (nm)	Fréquence de l'horloge	Largeur des données	MIPS
1971	Intel 4004	2 300	10 000	108 kHz	4 bits/4 bits bus	0,06
1974	Intel 8008	6 000	6 000	2 MHz	8 bits/8 bits bus	0,64
1979	Intel 8088	29 000	3 000	5 MHz	16 bits/8 bits bus	0,33
1982	Intel 80286	134 000	1 500	6 à 16 MHz (20 MHz chez AMD)	16 bits/16 bits bus	1
1985	Intel 80386	275 000	1 500	16 à 40 MHz	32 bits/32 bits bus	5
1989	Intel 80486	1 200 000 (800nm)	1 000 à 800	16 à 100 MHz	32 bits/32 bits bus	20
1993	Pentium (Intel P5)	3 100 000	800 à 250	60 à 233 MHz	32 bits/64 bits bus	100
1997	Pentium II	7 500 000	350 à 250	233 à 450 MHz	32 bits/64 bits bus	300
1999	Pentium III	9 500 000	250 à 130	450 à 1 400 MHz	32 bits/64 bits bus	510
2000	Pentium 4	42 000 000	180 à 65	1,3 à 3,8 GHz	32 bits/64 bits bus	1 700
2004	Pentium 4 D (Prescott)	125 000 000	90 à 65	2.66 à 3,6 GHz	32 bits/64 bits bus	9 000
2006	Core 2 Duo (Conroe)	291 000 000	65	2,4 GHz (E6600)	64 bits/64 bits bus	22 000
2007	Core 2 Quad (Kentsfield)	2*291 000 000	65	3 GHz (Q6850)	64 bits/64 bits bus	2*22 000 (?)
2008	Core 2 Duo (Wolfdale)	410 000 000	45	3,33 GHz (E8600)	64 bits/64 bits bus	~24 200
2008	Core 2 Quad (Yorkfield)	2*410 000 000	45	3,2 GHz (QX9770)	64 bits/64 bits bus	~2*24 200
2008	Intel Core i7 (Bloomfield)	731 000 000	45	3,33 GHz (Core i7 975X)	64 bits/64 bits bus	?
2009	Intel Core i5/i7 (Lynnfield)	774 000 000	45	3,06 GHz (i7 880)	64 bits/64 bits bus	76 383
2010	Intel Core i7 (Gulftown)	1 170 000 000	32	3,47 GHz (Core i7 990X)	64 bits/64 bits bus	147 600
2011	Intel Core i3/i5/i7 (Sandy Bridge)	1 160 000 000	32	3,5 GHz (Core i7 2700K)	64 bits/64 bits bus	
2011	Intel Core i7/Xeon (Sandy Bridge-E)	2 270 000 000	32	3,5 GHz (Core i7 3970X)	64 bits/64 bits bus	1 ou 2
2012	Intel Core i3/i5/i7 (Ivy Bridge)	1 400 000 000	22	3,5 GHz (Core i7 3770K)	64 bits/64 bits bus	
2013	Intel Core i3/i5/i7 (Haswell)	1 400 000 000	22	3,8 GHz (Core i7 4770K)	64 bits/64 bits bus	
2014	Intel Core i3/i5/i7 (Broadwell)	1 400 000 000	14	3,8 GHz (Core i7 5775R)	64 bits/64 bits bus	
2015	Intel Core i3/i5/i7 (Skylake)	1 750 000 000	14	4 GHz (Core i7 6700K)	64 bits/64 bits bus	
2016	Intel Core i3/i5/i7 (Kabylake)	?	14	4.2 GHz (Core i7 7700K)	64 bits/64 bits bus	
2017	Intel Core i3/i5/i7 (Cannonlake)	?	10		64 bits/64 bits bus	

Figure 1 : Evolution des processeurs Intel

Remarque

Selon la **loi de Moore**, édictée en 1965 par Gordon E. Moore, cofondateur de la société Intel, prévoyait que les performances des processeurs (par extension le nombre de transistors intégrés sur silicium) doubleraient tous les 12 mois. Cette loi a été révisée en 1975, portant le nombre de mois à 18. La loi de Moore se vérifie encore aujourd'hui.

III. STRUCTURE D'UN MICRO-PROCESSEUR

1. Composition d'un microprocesseur

Un processeur n'est pas une unité de calcul. Cette dernière est incluse dans le processeur, mais il est composé :

- d'une **unité de traitement** pour le traitement des données (Les calculs arithmétiques et logiques).
- d'une **unité de commande et de contrôle** (UC ou UCC) qui contrôle le mouvement des données et des instructions, ainsi que les opérations de l'UAL.
- des **registres** spécifiques pour le stockage temporaire des données et instructions
- Une **unité de gestion des bus** (ou *unité d'entrées-sorties*), qui gère les flux d'informations entrant et sortant, en interface avec la mémoire vive du système ;
- d'une **horloge** qui fournit un signal régulier pour synchroniser tout le fonctionnement du processeur. Elle est présente dans les processeurs synchrones mais absente des processeurs asynchrones et des processeurs autosynchrones.

2. Structure de base d'un processeur

Un microprocesseur est construit autour de deux éléments principaux :

- Une unité de commande
- Une unité de traitement associée 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.

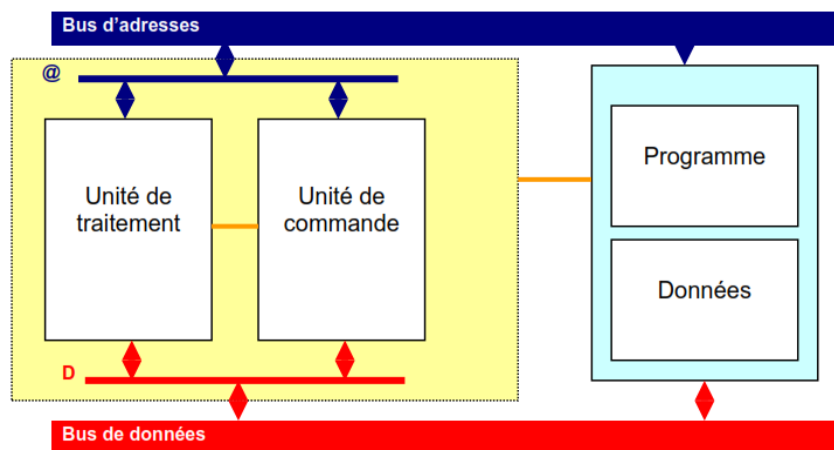


Figure 2 : Structure de base d'un microprocesseur

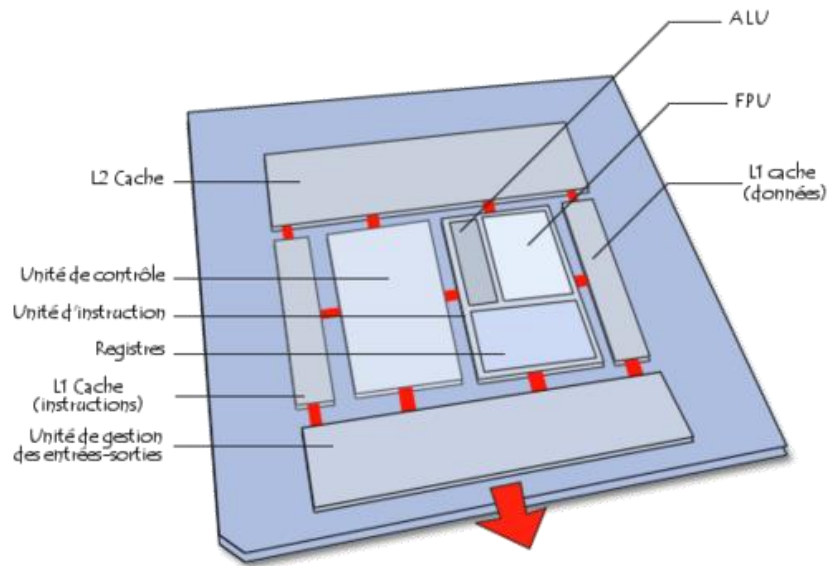


Figure 3 : Représentation physique simplifiée d'un microprocesseur

Un processeur possède aussi trois types de bus :

- bus de données, qui définit la taille des données pour les entrées–sorties, dont les accès à la mémoire (indépendamment de la taille des registres internes) ;
- bus d'adresse, qui permet, lors d'une lecture ou d'une écriture, d'envoyer l'adresse où elle s'effectue, et donc définit le nombre de cases mémoire accessibles ;
- bus de contrôle, qui permet la gestion du matériel, via les interruptions.

3. Schéma fonctionnel

Le schéma fonctionnel d'un processeur se présente comme suit :

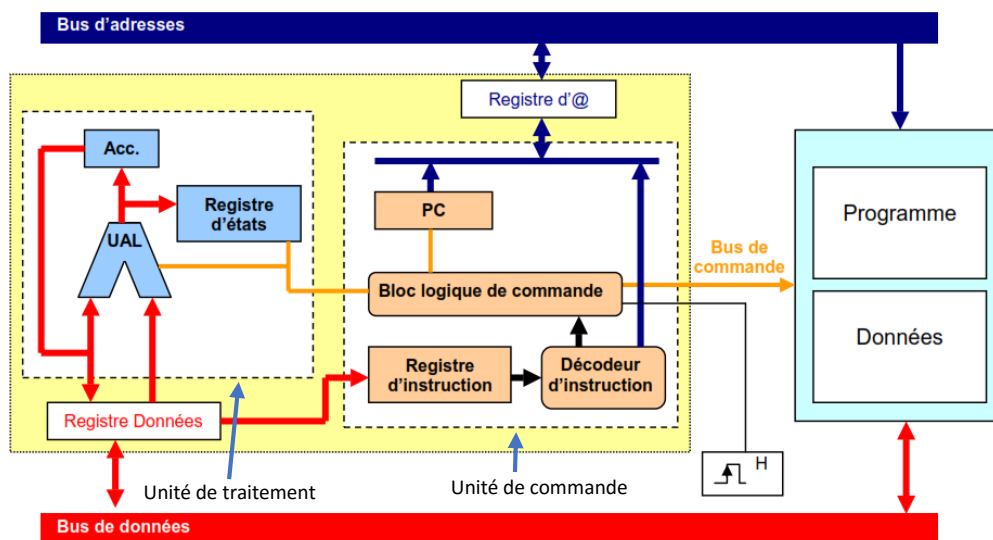


Figure 4 : Schéma fonctionnel d'un processeur

IV. L'UNITE DE TRAITEMENT


1. Définition

C'est le cœur du microprocesseur, il permet d'effectuer les calculs arithmétiques et logiques.

2. Composition de l'unité de traitement

L'unité de traitements regroupe les circuits qui assurent les traitements nécessaires à l'exécution des instructions :

- ✚ **L'Unité Arithmétique et Logique (UAL)** : c'est un circuit complexe qui assure les fonctions logiques (AND, OR, XOR), de Comparaison ($<$, $<=$, $>$, $>=$), Décalage, et d'arithmétique (Addition, soustraction).
- ✚ **Le registre d'état** : il 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**)

-  **Le registre d'état** : il 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**)

- retenue (**carry : C**)
- retenue intermédiaire (**Auxiliary-Carry : AC**)
- signe (**Sign : S**)
- débordement (**overflow : OV ou V**)
- zéro (**Z**)
- parité (**Parity : P**)

- **Les accumulateurs** : ce 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.
- **L'unité de virgule flottante** (notée **FPU**, pour *Floating Point Unit*), qui accomplit les calculs complexes non entiers que ne peut réaliser l'unité arithmétique et logique.

V. L'UNITE DE COMMANDE ET DE CONTROLE

1. Définition

Elle permet de séquencer le déroulement des instructions ; elle commande et contrôle le fonctionnement de l'UAL, de la mémoire et des E/S. 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 par l'UAL, puis effectue la préparation de l'instruction suivante.

2. Composition de l'UC

L'UC est composée par :

- **le compteur de programme (PC)** ou compteur ordinal 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 (IR) 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)** : Le séquenceur, ou **unité de contrôle**, se charge de gérer le processeur. Il peut décoder les instructions, choisir les registres à utiliser, gérer les interruptions ou initialiser les registres au démarrage. Il fait appel à l'unité d'entrée-sortie pour communiquer avec la mémoire ou les périphériques. 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, et met à jour le Compteur ordinal (CO). Il s'agit d'un automate réalisé soit de façon câblée (obsolète), soit de façon micro-programmée, on parle alors de microprocesseur.
- **Horloge système** : sert à synchroniser le travail des différentes unités participant à l'exécution d'une instruction

VI. LES REGISTRES

1. Définition

Les **registres** sont des petites mémoires internes au processeur et très rapides, pouvant être accédées facilement. Ce sont les éléments de mémoire les plus rapides et servent au stockage des opérandes et des résultats intermédiaires.

Un plus grand nombre de registres permettra au processeur d'être plus indépendant de la mémoire. La taille des registres dépend de l'architecture, mais est généralement de quelques octets et correspond au nombre de bit de l'architecture (un processeur 8 bits aura des registres d'un octet).

2. Les différents types de registre

Parmi les registres, on peut citer :

- **le registre accumulateur (ACC)**, stockant les résultats des opérations arithmétiques et logiques de l'UAL;
- **le registre instruction (RI)**, contenant l'instruction en cours de traitement ; Ce registre est chargé au début du cycle d'exécution par l'instruction dont l'adresse est donnée par le compteur de programme PC
- **le compteur ordinal (CO ou PC pour *Program Counter*)**, contenant l'adresse de la prochaine instruction à traiter. À chaque début de cycle d'exécution, l'instruction à exécuter est chargée dans le registre IR à partir de l'adresse contenue dans le registre PC. Ensuite, le registre PC est incrémenté pour pointer sur l'instruction suivante.
- **le registre tampon**, stockant temporairement une donnée provenant de la mémoire.
- **Registres spécialisés** : destinés pour certaines opérations comme les registres de décalages, registres des opérations arithmétiques à virgule flottante, ...etc.
- les **registres généraux**, qui servent à stocker les données allant être utilisées (ce qui permet d'économiser des allers-retours avec la mémoire).
- **le registre d'état (PSW, *Processor Status Word*)**, permettant de stocker des indicateurs sur l'état du système. Les indicateurs les plus fréquents sont :
 - signe : le bit de signe du résultat de la mémoire de la dernière opération arithmétique
 - zéro : à 1 lorsque le résultat est 0
 - retenue : à 1 lorsqu'une opération a générée une retenue
 - égal : à 1 si le résultat d'une comparaison est une égalité
 - débordement : à 1 lorsqu'une opération a provoqué un débordement
 - interruption : indique si le fonctionnement normal peut être interrompu
 - superviseur : indique un mode privilégié
- **le pointeur de pile** : il sert à stocker l'adresse du sommet des piles, qui sont en fait des structures de données généralement utilisées pour gérer des appels de sous-programmes,

VII. LES INSTRUCTIONS

1. Définition

Une **instruction** est l'opération élémentaire que le processeur peut accomplir. Les instructions sont stockées dans la mémoire principale, en vue d'être traitée par le processeur.

Une instruction est composée de deux champs :

- le **code opération**, représentant l'action que le processeur doit accomplir ;
- le **code opérande**, définissant les paramètres de l'action. Le code opérande dépend de l'opération. Il peut s'agir d'une donnée ou bien d'une adresse mémoire.

Code opération	Code opérande
----------------	---------------

Le nombre d'octets d'une instruction est variable selon le type de donnée (l'ordre de grandeur est de 1 à 4 octets).

2. Jeu d'instructions

On appelle **jeu d'instructions** l'ensemble des opérations élémentaires qu'un processeur peut accomplir. Le jeu d'instruction d'un processeur détermine ainsi son architecture, sachant qu'une même architecture peut aboutir à des implémentations différentes selon les constructeurs.

Les différents types d'instructions ou jeu d'instructions réalisées par l'UAL sont :

- **Transfert de données** : chargement, déplacement, stockage, transfert de données entre registre-registre ou mémoire-registre, ...etc.
- **Opérations arithmétiques** : opérations telles que les additions, soustractions, divisions ou multiplication.
- **Opérations logiques** : opérations ET, OU, NON, NON exclusif, etc.
- **Contrôle de séquence**: branchement impératif ou conditionnel, boucle, appel de procédure, ...etc.
- **Entrées/sorties**: Lecture, Ecriture, Affichage, ...etc.
- **Manipulations diverses**: décalages de bits, conversions de format, incrémentation ou décrémentation du registre, ...etc.

Chaque type de processeur possède son propre jeu d'instruction. On distingue ainsi les familles de processeurs suivants, possédant chacun un jeu d'instruction qui leur est propre :

- 80x86 : le « x » représente la famille. On parle ainsi de 386, 486, 586, 686, etc.

- ARM
- IA-64
- MIPS
- Motorola 6800
- PowerPC
- SPARC
- ...

Cela explique qu'un programme réalisé pour un type de processeur ne puisse fonctionner directement sur un système possédant un autre type de processeur, à moins d'une traduction des instructions, appelée **émulation**. Le terme « **émulateur** » est utilisé pour désigner le programme réalisant cette traduction.

3. Le cycle d'exécution d'une instruction

Ce travail est effectué par la CPU en des étapes comme suit :

- Recherche de l'instruction (fetch) : l'instruction est lue depuis la mémoire
- Interprétation de l'instruction (decode) : l'instruction est décodée pour déterminer à quelle action elle correspond
- Exécution (execute) :
 - Recherche de données (fetch data) : l'exécution de l'instruction peut demander la lecture de données dans la mémoire ou depuis un module d'E/S
 - Traitement des données (process data) : l'exécution de l'instruction peut demander des opérations arithmétiques ou logiques sur les données
 - Ecriture des données (write data) : l'exécution de l'instruction peut demander l'écriture du résultat dans la mémoire ou depuis un module d'E/S

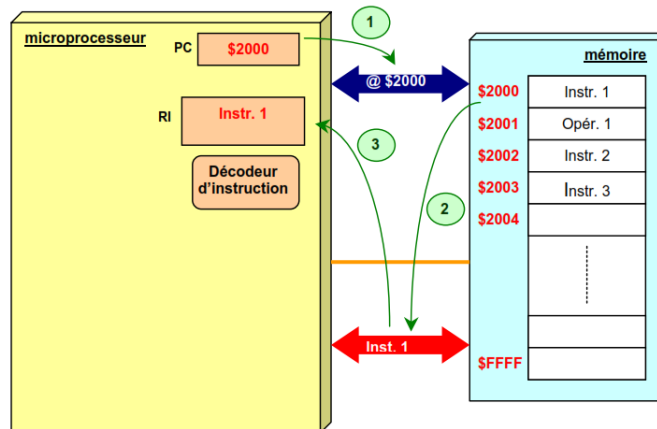
Toutes les machines ont un schéma d'exécution similaire, qui consiste donc à exécuter la boucle suivante :

- ❖ Répéter
 - Fetch
 - Decode
 - execute

Nous allons dans la partie qui suit expliquer de façon plus détaillée ce cycle d'exécution avec des illustrations à l'appui.

Phase 1: Recherche de l'instruction à traiter (fetch)

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.
4. Le compteur ordinal ou Program Counter (registre PC) est incrémenté de 1



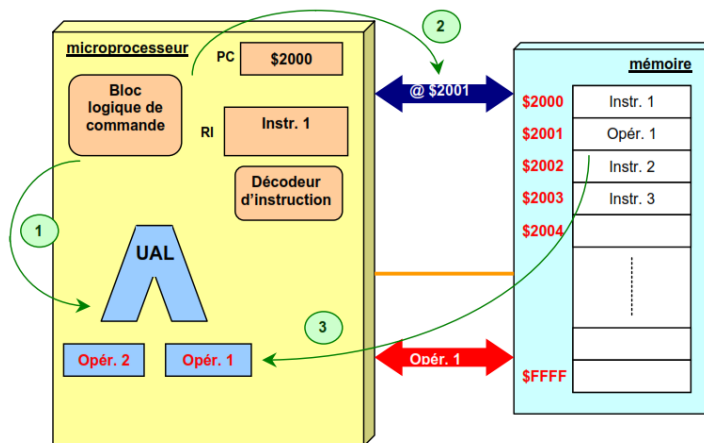
- *Fetch :*

- $MAR \leftarrow PC$
- L'unité de contrôle demande une lecture de la mémoire principale
- Le résultat de la lecture est placé dans le MBR
- $IR \leftarrow MBR$
- $PC \leftarrow PC + 1$

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

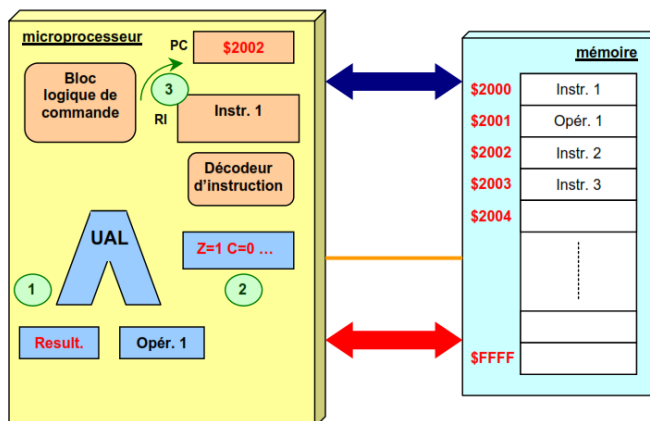
Le registre d'instruction contient maintenant le premier mot de l'instruction qui peut être codée sur plusieurs mots. Ce premier mot contient le code opératoire qui définit la nature de l'opération à effectuer (addition, rotation,...) et le nombre de mots de l'instruction.

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 stockée dans un registre.



Phase 3 : Exécution de l'instruction

1. Le microprogramme 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.



VIII. LES MEMOIRES CACHES

1. Définition

La **mémoire cache** (également appelée *antémémoire* ou *mémoire tampon*) est une mémoire rapide permettant de réduire les délais d'attente des informations stockées en mémoire vive.

En effet, la mémoire centrale de l'ordinateur possède une vitesse bien moins importante que le processeur. Il existe néanmoins des mémoires beaucoup plus rapides, mais dont le coût est très élevé. La solution consiste donc à inclure ce type de mémoire rapide à proximité du processeur et d'y stocker temporairement les principales données devant être traitées par le processeur.

2. Les différents types de mémoires caches

Les ordinateurs récents possèdent plusieurs niveaux de mémoire cache :

- La **mémoire cache de premier niveau** (appelée **L1 Cache**, pour **Level 1 Cache**) est directement intégrée dans le processeur. Elle se subdivise en 2 parties :
 - La première est le cache d'instructions, qui contient les instructions issues de la mémoire vive décodées lors de passage dans les pipelines.
 - La seconde est le cache de données, qui contient des données issues de la mémoire vive et les données récemment utilisées lors des opérations du processeur.

Les caches du premier niveau sont très rapides d'accès. Leur délai d'accès tend à s'approcher de celui des registres internes aux processeurs.

- La **mémoire cache de second niveau** (appelée **L2 Cache**, pour **Level 2 Cache**) est située au niveau du boîtier contenant le processeur (dans la puce). Le cache de second niveau vient s'intercaler entre le processeur avec son cache interne et la mémoire vive. Il est plus rapide d'accès que cette dernière mais moins rapide que le cache de premier niveau.
- La **mémoire cache de troisième niveau** (appelée **L3 Cache**, pour **Level 3 Cache**) autrefois située au niveau de la carte mère (utilisation de la mémoire centrale), elle est aujourd'hui intégrée directement dans le CPU.

Tous ces niveaux de cache permettent de réduire les temps de latence des différentes mémoires lors du traitement et du transfert des informations. Pendant que le processeur travaille, le contrôleur de cache de premier niveau peut s'interfacer avec celui de second niveau pour faire des transferts d'informations sans bloquer le processeur. De même, le cache de second niveau est interfacé avec celui de la mémoire vive (en l'absence de cache de troisième niveau intégré), pour permettre des transferts sans bloquer le fonctionnement normal du processeur.

IX. LES ARCHITECTURES DE BASE DE PROCESSEUR

Actuellement l'architecture des microprocesseurs se composent de deux grandes familles :

- L'architecture **CISC** (Complex Instruction Set Computer)
- L'architecture **RISC** (Reduced Instruction Set Computer)

Toutefois, des améliorations ont été apportées à ces deux architectures pour aboutir à de nouvelles formes d'architectures, dites architectures améliorées.

1. L'architecture CISC

a. Définition

L'architecture **CISC** (*Complex Instruction Set Computer*, soit « *ordinateur à jeu d'instruction complexe* ») consiste à câbler dans le processeur des instructions complexes, difficiles à créer à partir des instructions de base.

L'architecture **CISC** est utilisée en particulier par les processeurs de type 80x86. Ce type d'architecture possède un coût élevé dû aux fonctions évoluées imprimées sur le silicium.

D'autre part, les instructions sont de longueurs variables et peuvent parfois nécessiter plus d'un cycle d'horloge. Or, un processeur basé sur l'architecture CISC ne peut traiter qu'une instruction à la fois, d'où un temps d'exécution conséquent.

C'est donc une architecture avec un grand nombre d'instructions (jeu d'instructions important) où le microprocesseur doit exécuter des tâches complexes par instruction unique. Pour une tâche donnée, une machine CISC exécute ainsi un petit nombre d'instructions mais chacune nécessite un plus grand nombre de cycles d'horloge. Le code machine de ces instructions varie d'une instruction à l'autre et nécessite donc un décodeur complexe.

b. Pourquoi cette architecture ?

Par le passé la conception de machines CISC était la seule envisageable. En effet, vu que la mémoire travaillait très lentement par rapport au processeur, on pensait qu'il était plus intéressant de soumettre au microprocesseur des instructions complexes. Ainsi, plutôt que de coder une opération complexe par plusieurs instructions plus petites (qui demanderaient autant d'accès mémoire très lent), il semblait préférable d'ajouter au jeu d'instructions du microprocesseur une instruction complexe qui se chargerait de réaliser cette opération. De plus, le développement des langages de haut niveau posa de nombreux problèmes quant à la conception de compilateurs. On a donc eu tendance à incorporer au niveau processeur des instructions plus proches de la structure de ces langages.

2. L'architecture RISC

a. Définition

Un processeur utilisant la technologie **RISC** (*Reduced Instruction Set Computer*, soit « ordinateur à jeu d'instructions réduit ») n'a pas de fonctions évoluées câblées.

Les programmes doivent ainsi être traduits en instructions simples, ce qui entraîne un développement plus difficile et/ou un compilateur plus puissant. Une telle architecture possède un coût de fabrication réduit par rapport aux processeurs CISC. De plus, les instructions, simples par nature, sont exécutées en un seul cycle d'horloge, ce qui rend l'exécution des programmes plus rapide qu'avec des processeurs basés sur une architecture CISC. Enfin, de tels processeurs sont capables de traiter plusieurs instructions simultanément en les traitant en parallèle.

C'est donc une architecture dans laquelle les instructions (jeu d'instructions) sont en nombre réduit (chargement, branchement, appel sous-programme). Chacune de ces instructions s'exécute ainsi en un cycle d'horloge. Bien souvent, ces instructions ne disposent que d'un seul mode d'adressage. Les accès à la mémoire s'effectuent seulement à partir de deux instructions (Load et Store). Par contre, les instructions complexes doivent être réalisées à partir de séquences basées sur les instructions élémentaires, ce qui nécessite un compilateur très évolué dans le cas de programmation en langage de haut niveau.

b. Comparaison des deux architectures

Le choix dépendra des applications visées. En effet, si on diminue le nombre d'instructions, on crée des instructions complexes (CISC) qui nécessitent plus de cycles pour être décodées et si on diminue le nombre de cycles par instruction, on crée des instructions simples (RISC) mais on augmente alors le nombre d'instructions nécessaires pour réaliser le même traitement.

Architecture RISC	ARCHITECTURE CISC
<ul style="list-style-type: none">- instructions simples ne prenant qu'un seul cycle- instructions au format fixe- décodeur simple (câblé)- beaucoup de registres- seules les instructions LOAD et STORE ont accès à la mémoire- peu de modes d'adressage- compilateur complexe	<ul style="list-style-type: none">- instructions complexes prenant plusieurs cycles- instructions au format variable- décodeur complexe (microcode)- peu de registres- toutes les instructions sont susceptibles d'accéder à la mémoire- beaucoup de modes d'adressage- compilateur simple

X. LES AMELIORATIONS TECHNOLOGIQUES DE L'ARCHITECTURE DE BASE DE MICROPROCESSEURS

L'ensemble des améliorations des microprocesseurs visent à diminuer le temps d'exécution du programme.

- La première idée qui vient à l'esprit est d'augmenter tout simplement la fréquence de l'horloge du microprocesseur. Mais l'accélération des fréquences provoque un surcroît de consommation ce qui entraîne une élévation de température. On est alors amené à équiper les processeurs de systèmes de refroidissement ou à diminuer la tension d'alimentation.
- Une autre possibilité d'augmenter la puissance de traitement d'un microprocesseur est de diminuer le nombre moyen de cycles d'horloge nécessaire à l'exécution d'une instruction. Dans le cas d'une programmation en langage de haut niveau, cette amélioration peut se faire en optimisant le compilateur. Il faut qu'il soit capable de sélectionner les séquences d'instructions minimisant le nombre moyen de cycles par instructions. Une autre solution est d'utiliser une architecture de microprocesseur qui réduise le nombre de cycles par instruction.

Au cours des années, les constructeurs de microprocesseurs (appelés *fondeurs*), ont mis au point un certain nombre d'améliorations permettant d'optimiser le fonctionnement du processeur. Nous allons voir ci-dessous les principales améliorations.

1. Le pipeline

a. Définition

Le **pipeline** (ou *pipelining*) est une technologie visant à permettre une plus grande vitesse d'exécution des instructions en parallélisant des étapes.

b. Principe

L'exécution d'une instruction est décomposée en une succession d'étapes et chaque étape correspond à l'utilisation d'une des fonctions du microprocesseur. Lorsqu'une instruction se trouve dans l'une des étapes, les composants associés aux autres étapes ne sont pas utilisés. Le fonctionnement d'un microprocesseur simple n'est donc pas efficace.

L'architecture pipeline permet d'améliorer l'efficacité du microprocesseur. En effet, lorsque la première étape de l'exécution d'une instruction est achevée, l'instruction entre dans la seconde étape de son exécution et la première phase de l'exécution de l'instruction suivante débute. Il peut donc y avoir une instruction en cours d'exécution dans chacune des étapes et chacun des composants du microprocesseur peut être utilisé à chaque cycle d'horloge. L'efficacité est maximale. Le temps d'exécution d'une instruction n'est pas réduit mais le débit d'exécution des instructions est considérablement augmenté. Une machine pipeline se caractérise par le nombre d'étapes utilisées pour l'exécution d'une instruction, on appelle aussi ce nombre d'étapes le nombre d'étages du pipeline.

Pour mieux comprendre le mécanisme du pipeline, il est nécessaire au préalable de comprendre les phases d'exécution d'une instruction. Les phases d'exécution d'une instruction pour un processeur contenant un pipeline « classique » à 5 étages sont les suivantes :

- **LI** : *Lecture de l'Instruction* (en anglais *FETCH instruction*) depuis le cache ;
- **DI** : *Décodage de l'Instruction* (*DECODE instruction*) et recherche des opérandes (Registre ou valeurs immédiate);
- **EX** : *Exécution de l'Instruction* (*EXECute instruction*) (si ADD, on fait la somme, si SUB, on fait la soustraction, etc.);
- **MEM** : *Accès mémoire* (*MEMory access*), écriture dans la mémoire si nécessaire ou chargement depuis la mémoire ;
- **ER** : *Ecriture* (*Write instruction*) de la valeur calculée dans les registres.

Les instructions sont organisées en file d'attente dans la mémoire, et sont chargées les unes après les autres.

Grâce au pipeline, le traitement des instructions nécessite au maximum les cinq étapes précédentes. Dans la mesure où l'ordre de ces étapes est invariable (LI, DI, EX, MEM et ER), il est possible de créer dans le processeur un certain nombre de circuits spécialisés pour chacune de ces phases.

L'objectif du pipeline est d'être capable de réaliser chaque étape en parallèle avec les étapes amont et aval, c'est-à-dire de pouvoir lire une instruction (LI) lorsque la précédente est en cours de décodage (DI), que celle d'avant est en cours d'exécution (EX), que celle située encore précédemment accède à la mémoire (MEM) et enfin que la première de la série est déjà en cours d'écriture dans les registres (ER).

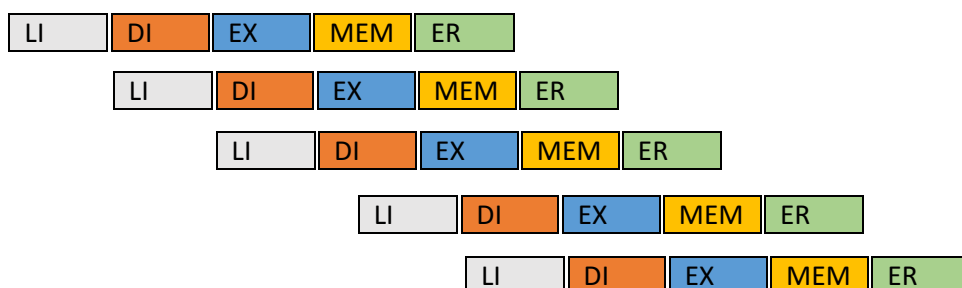


Figure 5 : Représentation d'un pipeline

Il faut compter en général 1 à 2 cycles d'horloge (rarement plus) pour chaque phase du pipeline, soit 10 cycles d'horloge maximum par instruction. Pour deux instructions, 12 cycles d'horloge maximum seront nécessaires ($10+2=12$ au lieu de $10*2=20$), car la précédente instruction était déjà dans le pipeline. Les deux instructions sont donc en traitement dans le processeur, avec un décalage d'un ou deux cycles d'horloge). Pour 3 instructions, 14 cycles d'horloge seront ainsi nécessaires, etc.

Le schéma ci-dessous fait la comparaison d'un modèle sans pipeline et un modèle avec pipeline :

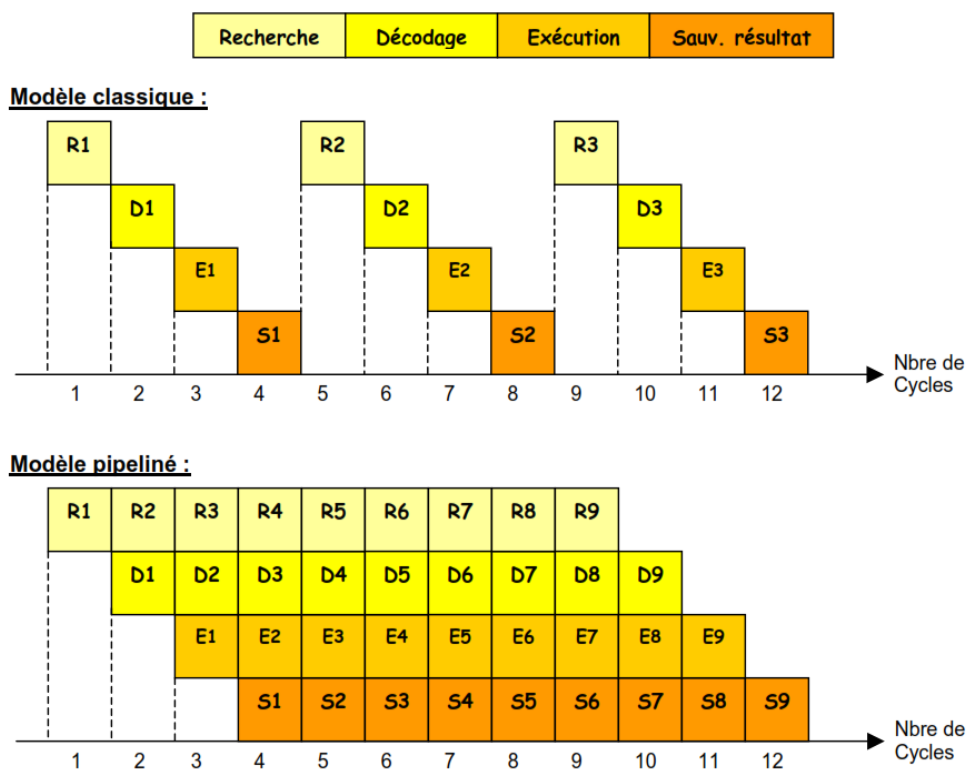


Figure 6 : Comparaison d'un modèle classique avec un modèle pipeline

Le principe du pipeline est ainsi comparable avec une chaîne de production de voitures. La voiture passe d'un poste de travail à un autre en suivant la chaîne de montage et sort complètement assemblée à la sortie du bâtiment. Pour bien comprendre le principe, il est nécessaire de regarder la chaîne dans son ensemble, et non pas véhicule par véhicule. Il faut ainsi 3 heures pour faire une voiture, mais pourtant une voiture est produite toute les minutes !

Il faut noter toutefois qu'il existe différents types de pipelines, de 2 à 40 étages, mais le principe reste le même.

c. Gain de performance

Dans cette structure, la machine débute l'exécution d'une instruction à chaque cycle et le pipeline est pleinement occupé à partir du quatrième cycle. Le gain obtenu dépend donc du nombre d'étages du pipeline. En effet, pour exécuter n instructions, en supposant que chaque instruction s'exécute en k cycles d'horloge, il faut :

- $n.k$ cycles d'horloge pour une exécution séquentielle.
- k cycles d'horloge pour exécuter la première instruction puis $n-1$ cycles pour les $n-1$ instructions suivantes si on utilise un pipeline de k étages

Le gain obtenu est donc de :

$$G = \frac{n.k}{n+k-1}$$

Donc lorsque le nombre n d'instructions à exécuter est grand par rapport à k , on peut admettre qu'on divise le temps d'exécution par k .

Remarque :

Le temps de traitement dans chaque unité doit être à peu près égal sinon les unités rapides doivent attendre les unités lentes.

Exemples :

L'Athlon d'AMD comprend un pipeline de 11 étages.

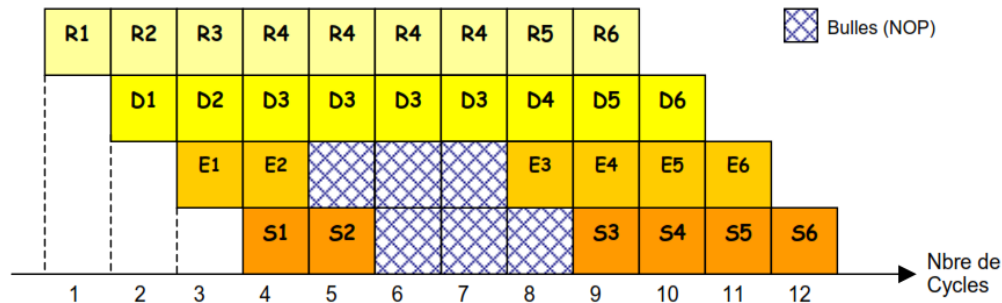
Les Pentium 2, 3 et 4 d'Intel comprennent respectivement un pipeline de 12, 10 et 20 étages.

d. Problèmes

La mise en place d'un pipeline pose plusieurs problèmes. En fait, plus le pipeline est long, plus le nombre de cas où il n'est pas possible d'atteindre la performance maximale est élevé. Il existe 3 principaux cas où la performance d'un processeur pipeliné peut être dégradé ; ces cas de dégradations de performances sont appelés des **aléas** :

- **aléa structurel** qui correspond au cas où deux instructions ont besoin d'utiliser la même ressource du processeur (conflit de dépendance),
- **aléa de données** qui intervient lorsqu'une instruction produit un résultat et que l'instruction suivante utilise ce résultat avant qu'il n'ait pu être écrit dans un registre,
- **aléa de contrôle** qui se produit chaque fois qu'une instruction de branchement est exécutée. Lorsqu'une instruction de branchement est chargée, il faut normalement attendre de connaître l'adresse de destination du branchement pour pouvoir charger l'instruction suivante. Les instructions qui suivent le saut et qui sont en train d'être traitées dans les étages inférieurs le sont en général pour rien, il faudra alors vider le pipeline. Pour atténuer l'effet

des branchements, on peut spécifier après le branchement des instructions qui seront toujours exécutées. On fait aussi appel à la **prédiction de branchement** qui a pour but de recenser lors de branchements le comportement le plus probable. Les mécanismes de prédiction de branchement permettent d'atteindre une fiabilité de prédiction de l'ordre de 90 à 95 %.



Lorsqu'un aléa se produit, cela signifie qu'une instruction ne peut continuer à progresser dans le pipeline. Pendant un ou plusieurs cycles, l'instruction va rester bloquée dans un étage du pipeline, mais les instructions situées plus en avant pourront continuer à s'exécuter jusqu'à ce que l'aléa ait disparu. Plus le pipeline possède d'étages, plus la pénalité est grande. Les compilateurs s'efforcent d'engendrer des séquences d'instructions permettant de maximiser le remplissage du pipeline. Les étages vacants du pipeline sont appelés des « bulles » de pipeline, en pratique une bulle correspond en fait à une instruction NOP (No Operation) émise à la place de l'instruction bloquée.

2. Notion de cache mémoire

a. Définition

La mémoire cache, qui permet d'accélérer les traitements en diminuant les accès à la mémoire vive. Le cache d'instructions reçoit les prochaines instructions à exécuter, le cache de données manipule les données. Parfois un cache unifié est utilisé pour les instructions et les données. Plusieurs niveaux (*levels*) de caches peuvent coexister, on les désigne souvent sous les noms de L1, L2, L3 ou L4.

b. Les raisons

L'écart de performance entre le microprocesseur et la mémoire ne cesse de s'accroître. En effet, les composants mémoire bénéficient des mêmes progrès technologiques que les microprocesseurs mais le décodage des adresses et la lecture/écriture d'une donnée sont des étapes difficiles à accélérer. Ainsi, le temps de cycle processeur décroît plus vite que le temps d'accès mémoire entraînant un goulot d'étranglement. La mémoire n'est plus en mesure de délivrer des informations aussi rapidement que le processeur est capable de les traiter. Il existe donc une latence d'accès entre ces deux organes.

c. Principe

Depuis le début des années 80, une des solutions utilisées pour masquer cette latence est de disposer une mémoire très rapide entre le microprocesseur et la mémoire. Elle est appelée **cache mémoire**. On compense ainsi la faible vitesse relative de la mémoire en permettant au microprocesseur d'acquérir les données à sa vitesse propre. On la réalise à partir de cellule SRAM de taille réduite (à cause du coût). Sa capacité mémoire est donc très inférieure à celle de la mémoire principale et sa fonction est de stocker les informations les plus récentes ou les plus souvent utilisées par le microprocesseur. Au départ cette mémoire était intégrée en dehors du microprocesseur mais elle fait maintenant partie intégrante du microprocesseur et se décline même sur plusieurs niveaux. Le principe de cache est très simple : le microprocesseur n'a pas conscience de sa présence et lui envoie toutes ses requêtes comme s'il agissait de la mémoire principale :

- Soit la donnée ou l'instruction requise est présente dans le cache et elle est alors envoyée directement au microprocesseur. On parle de **succès** de cache. (a)
- soit la donnée ou l'instruction n'est pas dans le cache, et le contrôleur de cache envoie alors une requête à la mémoire principale. Une fois l'information récupérée, il la renvoie au microprocesseur tout en la stockant dans le cache. On parle de **défaul** de cache. (b)

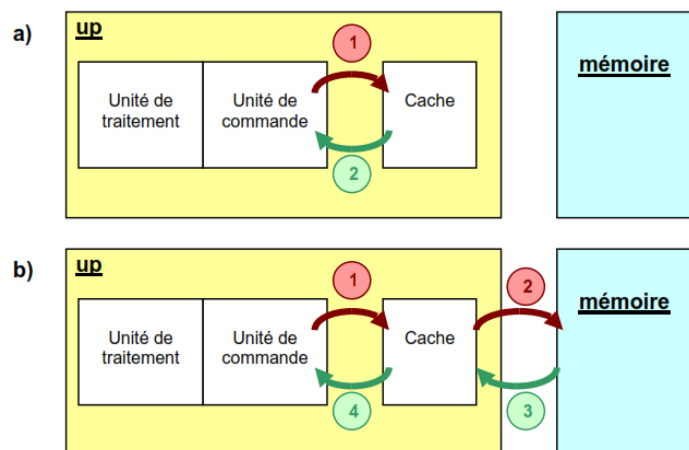


Figure 7 : Fonctionnement d'une cache mémoire

Bien entendu, le cache mémoire n'apporte un gain de performance que dans le premier cas. Sa performance est donc entièrement liée à son taux de succès. Il est courant de rencontrer des taux de succès moyen de l'ordre de 80 à 90%.

Remarque :

- Un cache utilisera une carte pour savoir quels sont les mots de la mémoire principale dont il possède une copie. Cette carte devra avoir une structure simple.
- Il existe dans le système deux copies de la même information : l'originale dans la mémoire principale et la copie dans le cache. Si le microprocesseur modifie la donnée présente dans le cache, il faudra prévoir une mise à jour de la mémoire principale.

- Lorsque le cache doit stocker une donnée, il est amené à en effacer une autre. Il existe donc un contrôleur permettant de savoir quand les données ont été utilisées pour la dernière fois. La plus ancienne non utilisée est alors remplacée par la nouvelle.
- A noter que l'on peut reprendre le même principe pour les disques durs et CD/DVD.

3. Le parallélisme

a. Définition

Le **parallélisme** consiste à exécuter simultanément, sur des processeurs différents, des instructions relatives à un même programme. Cela se traduit par le découpage d'un programme en plusieurs processus traités en parallèle afin de gagner en temps d'exécution.

Ce type de technologie nécessite toutefois une synchronisation et une communication entre les différents processus, à la manière du découpage des tâches dans une entreprise : le travail est divisé en petits processus distincts, traités par des services différents. Le fonctionnement d'une telle entreprise peut être très perturbé lorsque la communication entre les services ne fonctionne pas correctement.

b. Principe et fonctionnement

Nous allons maintenant aborder le parallélisme de données, qui consiste à traiter des données différentes en parallèle. De nombreuses situations s'y prêtent relativement bien : traitement d'image, manipulation de sons, vidéo, rendu 3d, etc. Mais pour exploiter ce parallélisme, il a fallu concevoir des processeurs adaptés. Une solution simple est d'utiliser plusieurs processeurs qui exécutent la même instruction, chacun sur des données différentes. Cette solution a autrefois été utilisée sur certains supercalculateurs, comme les Thinking machines CM-1 et CM-2. Ces ordinateurs possédaient environ 64000 processeurs minimalistes, qui exécutaient tous la même instruction au même cycle d'horloge. Mais ce genre de solution est vraiment quelque chose d'assez lourd, qui ne peut être efficace et rentable que sur des grosses données, et sur des ordinateurs chers et destinés à des calculs relativement importants. Ces architectures ont depuis été remplacées par des architectures qui exploitent le parallélisme de données au niveau de l'unité de calcul, celle-ci pouvant exécuter des calculs en parallèle.

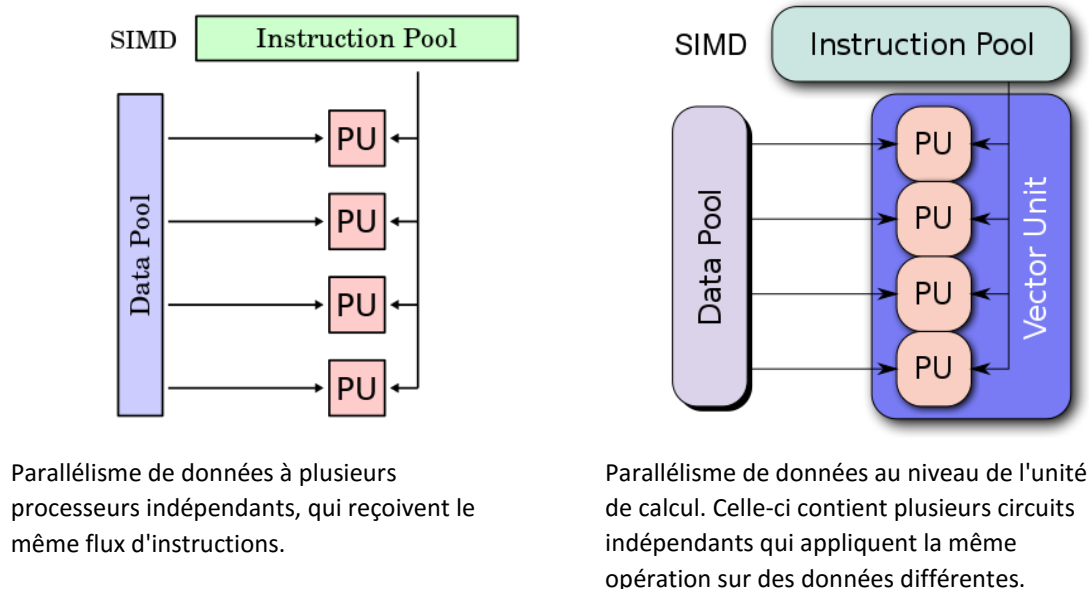


Figure 8 : Fonctionnement du parallélisme

Instructions SIMD

Certains processeurs fournissent des instructions capables de traiter plusieurs éléments en parallèle. Ces instructions sont appelées des **instructions vectorielles**. Ces instructions vectorielles travaillent sur un plusieurs nombres entiers ou flottants placés les uns à côté des autres, qui forment ce qu'on appelle un vecteur. Quand on exécute une instruction sur un vecteur, celle-ci traite ces entiers ou flottants en parallèle, simultanément.

En tentant d'obtenir des performances scalaires et au-delà, on a abouti à diverses méthodes qui conduisent le processeur à un comportement moins linéaire et plus parallèle. Lorsqu'on parle de parallélisme de processeur, deux techniques de conception sont utilisées :

- parallélisme au niveau instruction (en anglais : *instruction-level parallelism*, ILP) ;
- parallélisme au niveau thread (en anglais : *thread-level parallelism*, TLP).

L'ILP vise à augmenter la vitesse à laquelle les instructions sont exécutées par un processeur (c'est-à-dire augmenter l'utilisation des ressources d'exécution présentes dans le circuit intégré). Le TLP vise à augmenter le nombre de threads que le processeur pourra exécuter simultanément. Chaque méthode diffère de l'autre d'une part, par la façon avec laquelle elle est implémentée et d'autre part, du fait de leur efficacité relative à augmenter les performances des processeurs pour une application.

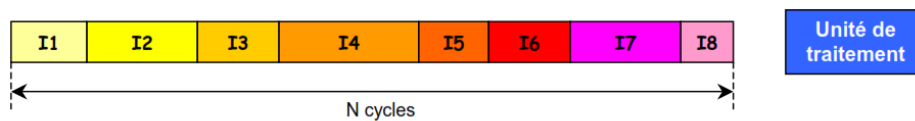
4. La technologie superscalaire

Une autre façon de gagner en performance est d'exécuter plusieurs instructions en même temps. La technologie **superscalaire** (en anglais *superscaling*) consiste à disposer plusieurs unités de traitement

(UAL) en parallèle afin de pouvoir traiter plusieurs instructions par cycle. Les instructions sont alors réparties entre les différentes unités d'exécution. Il faut donc pouvoir soutenir un flot important d'instructions et pour cela disposer d'un cache performant.

Dans un processeur superscalaire, plusieurs instructions sont lues et transmises à un répartisseur qui décide si les instructions seront exécutées en parallèle (simultanément) ou non. Le cas échéant, les instructions sont réparties sur les unités d'exécution disponibles. En général, plus un processeur superscalaire est capable d'exécuter d'instructions en parallèle et plus le nombre d'instructions exécutées dans un cycle sera élevé.

Architecture scalaire :



Architecture superscalaire :

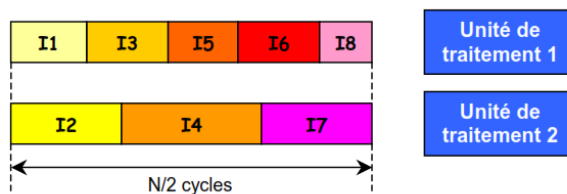


Figure 9 : Comparaison Architecture scalaire et architecture superscalaire

Remarque :

C'est le type d'architecture mise en œuvre dans les premiers Pentium d'Intel apparus en 1993.

La plupart des difficultés rencontrées dans la conception des architectures de processeurs superscalaires résident dans la mise au point du répartisseur. Le répartisseur doit être disponible rapidement et être capable de déterminer sans erreur si les instructions peuvent être exécutées en parallèle, il doit alors les distribuer de façon à charger les unités d'exécution autant qu'il est possible. Pour cela, le pipeline d'instructions doit être rempli aussi souvent que possible, créant le besoin d'une quantité importante de mémoire cache. Les techniques de traitement aléatoire comme la prédiction de branchement, l'exécution spéculative et la résolution des dépendances aux données deviennent cruciales pour maintenir un haut niveau de performance. En tentant de prédire quel branchement (ou chemin) une instruction conditionnelle prendra, le processeur peut minimiser le temps que tout le pipeline doit attendre jusqu'à la fin d'exécution de l'instruction conditionnelle. L'exécution spéculative améliore les performances modestes en exécutant des portions de code qui seront, ou ne seront pas, nécessaires à la suite d'une instruction conditionnelle. La résolution de la dépendance aux données est obtenue en réorganisant l'ordre dans lequel les instructions sont exécutées en optimisant la disponibilité des données.

5. Architecture pipeline et superscalaire

Le principe est de d'exécuter les instructions de façon pipelinée dans chacune des unités de traitement travaillant en parallèle.

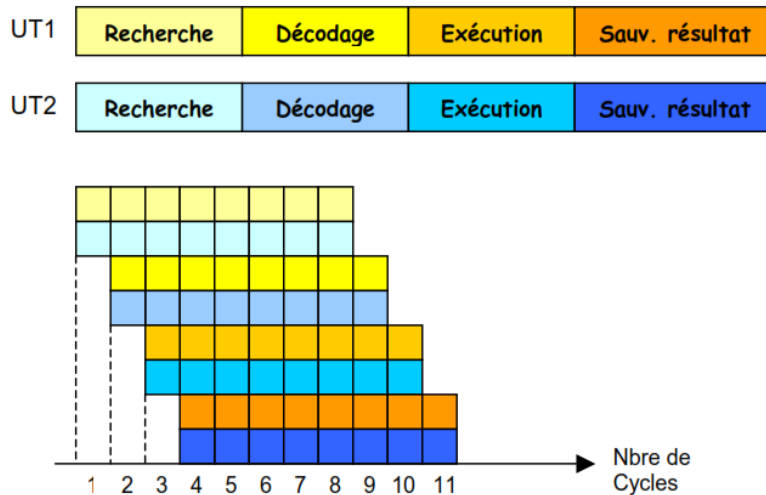


Figure 10 : Fonctionnement d'une architecture pipeline et superscalaire

Les architectures à pipeline et superscalaires augmentent le parallélisme (ILP) des processeurs en permettant à un processeur unique d'exécuter des instructions à un rythme de plus d'une instruction par cycle. La plupart des processeurs d'aujourd'hui ont au moins une partie superscalaire.

6. HyperThreading

Un **thread** désigne un « **fil d'exécution** » dans le programme ; c'est-à-dire une suite linéaire et continue d'instructions qui sont exécutées séquentiellement les unes après les autres.

Un **thread** ou **fil d'exécution** (**processus léger, tâche**) est similaire à un processus car tous deux représentent l'exécution d'un ensemble d'instructions d'un processeur. Du point de vue de l'utilisateur, ces exécutions semblent se dérouler en parallèle. Toutefois, les threads partagent le même espace mémoire protégé, les mêmes ressources et le même espace mémoire.

La technologie **HyperThreading** (ou *Hyper-Threading*, noté *HT*, traduisez *HyperFlots* ou *HyperFlux*) consiste à définir deux processeurs logiques au sein d'un processeur physique. Ainsi, le système reconnaît deux processeurs physiques et se comporte en système multitâche en envoyant deux threads simultanés, on parle alors de **SMT** (*Simultaneous Multi Threading*). Cette « supercherie » permet d'utiliser au mieux les ressources du processeur en garantissant que des données lui sont envoyées en masse.

Le SMT est le partage d'un cœur de processeur superscalaire (les pipelines, les unités de calcul et les caches) entre plusieurs threads. Les processeurs non SMT passent alternativement d'un thread à

l'autre pour l'exécution des instructions, alors que des processeurs SMT peuvent allouer des unités de calcul à des threads différents simultanément. Le but est d'améliorer l'utilisation des ressources.

Un processeur est dit **multithread** s'il est capable d'exécuter efficacement plusieurs threads simultanément. Contrairement aux systèmes multiprocesseurs (tels les systèmes multi-cœur¹), les threads doivent partager les ressources d'un unique cœur¹ : les unités de traitement, le cache processeur et le translation lookaside buffer ; certaines parties sont néanmoins dupliquées : chaque *thread* dispose de ses propres registres et de son propre pointeur d'instruction. Là où les systèmes multiprocesseurs incluent plusieurs unités de traitement complètes, le *multithreading* a pour but d'augmenter l'utilisation d'un seul cœur en tirant profit des propriétés des *threads* et du parallélisme au niveau des instructions. Comme les deux techniques sont complémentaires, elles sont parfois combinées dans des systèmes comprenant de multiples processeurs *multithreads* ou des processeurs avec de multiples cœurs *multithreads*.

L'hyperthreading (où Simultaneous Multithreading = SMT) représente la première tentative chez Intel de **simulation** d'un processeur double cœur. Cette technologie a été mise en place sur le Pentium 4 fonctionnant à 3.06 Ghz sorti en Novembre 2002.

L'hyperthreading permet d'utiliser au mieux les ressources du processeur lorsqu'il est amené à gérer plusieurs threads simultanément. Lorsqu'un programme nécessite énormément de ressources, l'hyperthreading permet de garder la main sur le système.

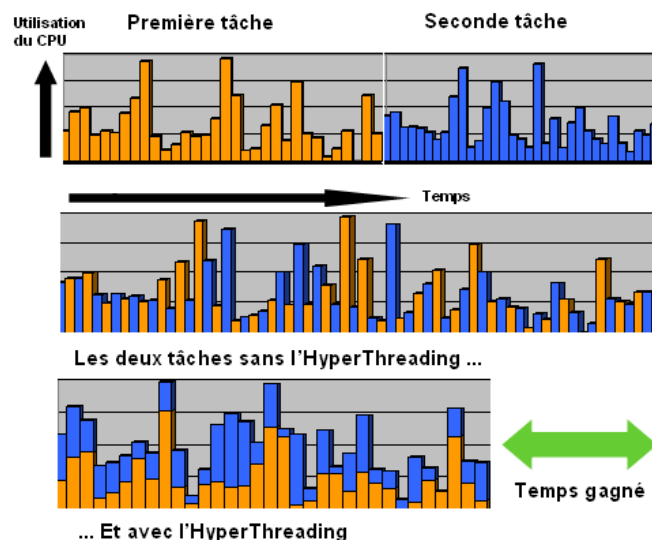


Figure 11 : HyperThreading vs sans HyperThreading

7. Le Multi-cœurs

a. Définition

Un **cœur** (physique) est un ensemble de circuits capables d'exécuter des programmes de façon autonome. Toutes les fonctionnalités nécessaires à l'exécution d'un programme sont présentes dans ces cœurs.

Un **processeur standard** possède un cœur (on dit qu'il est *single-core*). Plusieurs instructions peuvent être traitées par le cœur d'un processeur mais ce sera toujours une par une, en série.

Un **processeur multi-cœurs** est tout simplement un processeur composé de plusieurs cœurs. Ainsi, Un **processeur dual-core** ou bi-cœurs contient deux cœurs, un **processeur quad-core** quatre cœurs. Un processeur dual-core, à fréquence d'horloge égale, possède d'une puissance de calcul deux fois plus importante.

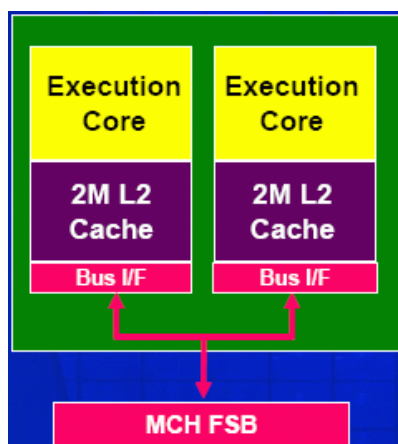
Sur les systèmes mono-cœurs, le système d'exploitation change régulièrement le programme en cours d'exécution pour passer d'un programme à un autre. Ces changements sont très rapides, et ne sont donc pas visibles par l'utilisateur, qui a l'impression que plusieurs programmes s'exécutent en même temps. Sur les systèmes multi-cœurs, si plusieurs applications sont exécutées simultanément sur l'ordinateur, celui-ci peut dès lors répartir ce travail entre les cœurs de processeurs, plutôt que d'effectuer les opérations en alternance sur un seul processeur. La notion d'*affinité processeur* (*processor affinity*) permet de lier un cœur donné à une application ou à un périphérique pour obtenir une réactivité meilleure.

b. Composition d'un cœur

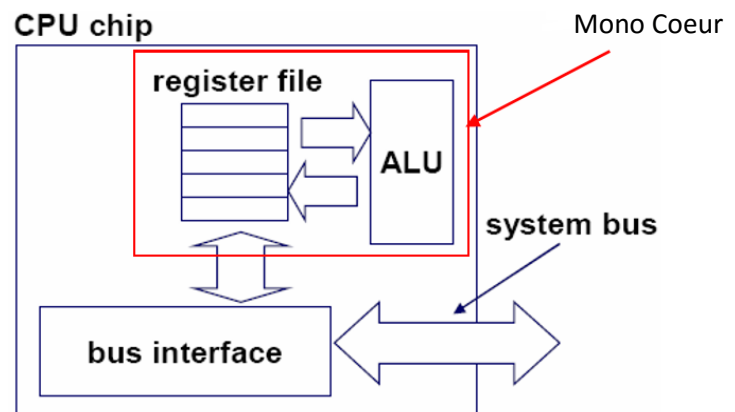
Le cœur comprend les éléments suivants :

- les caches L1
- les circuits de décodage des instructions
- les circuits de prédiction de branchement
- les unités d'exécution
- les registres

Une mémoire cache peut être dédiée à un cœur ou peut être partagée entre plusieurs cœurs.



Dual Core



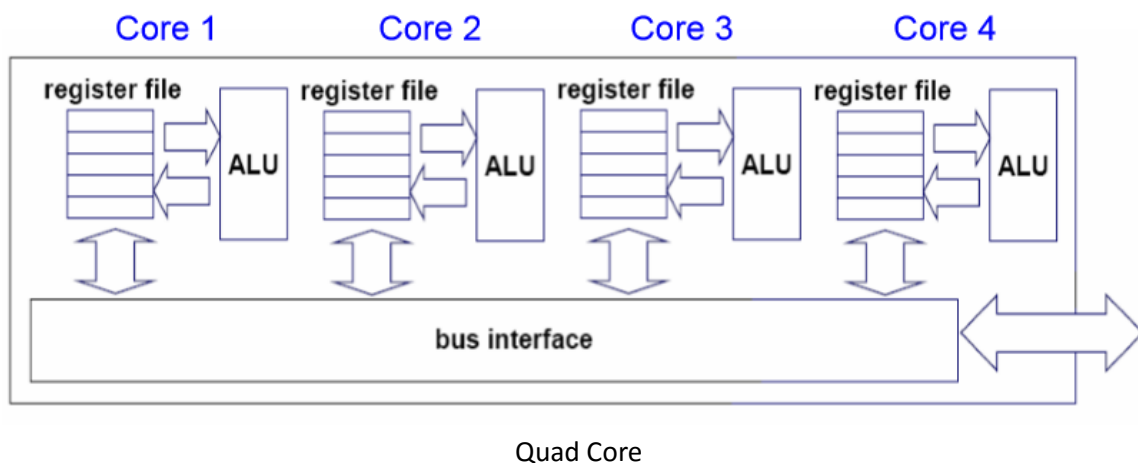


Figure 12 : Processeur multi-cœurs

c. Pourquoi le multi-cœur ?

✚ La limitation de fréquence.

- D'un point de vue technologique, les constructeurs semblent être arrivés devant un mur de fréquence. Avant l'apparition des processeurs multi-cœurs, les constructeurs de processeurs augmentaient la puissance de leurs produits en élevant la fréquence de calcul de leurs processeurs mono-cœurs. Mais cette méthode a fini par atteindre ses limites. En effet, l'augmentation de fréquence d'un processeur cause rapidement des problèmes de surchauffe, le refroidissement à air (ventilateur) n'étant plus suffisant.
- Les Pentium 4 arrivent en fin de vie en 2006. Intel prévoyait de les faire évoluer pour atteindre des fréquences proches de 7 à 10 Ghz mais s'est heurté à un problème majeur : la consommation de ses processeurs n'a cessé d'augmenter, l'architecture Netburst (introduite avec le Pentium 4) n'étant pas adaptée à la montée en fréquence au-delà de 4 Ghz sans utiliser de système de refroidissement conséquent. Cependant Intel dispose d'une architecture performante et qui consomme peu d'énergie.

Pour lutter contre cette surchauffe, il faut passer à un refroidissement par eau délicat et onéreux, ou bien passer à des ventilations de plus grande taille et vitesse, créatrices de nuisances sonores. D'autre part le besoin croissant en énergie des microprocesseurs était problématique notamment pour les ordinateurs portables.

Les processeurs mono cœurs les plus puissants utilisés dans les ordinateurs de grande distribution ont des fréquences ne dépassant pas en général, les 3 à 4 GHz, car au-delà la température devient trop importante.

C'est pour contourner cette limite que les constructeurs se sont tournés vers la fragmentation des puces. Il existait déjà des ordinateurs fonctionnant avec plusieurs processeurs distincts (par exemple, les supercalculateurs). L'idée ici est de reproduire ce parallélisme au sein d'un unique processeur : en bref, introduire plusieurs unités de calcul dans un même processeur. Le principe est simple : plutôt que d'avoir un processeur « simple » à fréquence élevée, on utilise par exemple deux cœurs, de fréquence moitié moindre. On obtient alors un processeur théoriquement de même puissance, mais de fréquence d'horloge plus basse, et de consommation électrique réduite. Le processeur ne rencontre pas les problèmes d'alimentation et de surchauffe de son homologue mono cœur : la puissance dissipée double quand on double les cœurs, alors qu'elle serait quadruplée si on doublait la fréquence d'horloge.

Le bruit thermique

La miniaturisation des puces a atteint un seuil tel que les fuites de courant électrique produisent un important bruit thermique. Aussi, ces fuites de courant électrique doivent être compensées afin d'acheminer toute l'énergie nécessaire au bon fonctionnement du processeur, entraînant une surconsommation globale du processeur. En outre, l'augmentation des températures peut d'une part causer des dommages physiques permanents à la puce et peut d'autre part dilater les matériaux et donc augmenter les délais de communication au sein du circuit. En général, l'augmentation des températures sur la puce est préjudiciable aux performances du processeur, à sa puissance et à sa fiabilité. Ces températures élevées pourraient être dissipées à condition d'améliorer les systèmes de refroidissement. Cependant, des améliorations dans ces techniques de dissipation thermique sont complexes et encombrantes, les rendant économiquement et ergonomiquement inintéressantes.

Ainsi, de nouvelles solutions pour outrepasser ces limites thermiques sont nécessaires. Même si les transistors plus petits ont le potentiel d'être utilisés à des fréquences élevées, les contraintes thermiques continuent de restreindre leurs fréquences de fonctionnement. Les concepteurs de puces ont atténué les complications thermiques par l'utilisation des **architectures multi-cœurs**, qui continuent d'augmenter le débit du système.

d. Les avantages du multi-cœur

Argumentation de la puissance de calcul d'un processeur.

Avoir **plusieurs cœurs** permet d'exécuter plusieurs instructions de façon simultanée, ainsi, le processeur peut traiter plusieurs programmes à la fois en optimisant leurs temps d'exécution. Toutes fois, il faut que les logiciels et les systèmes d'exploitation sachent gérer correctement ces processeurs afin qu'un gain significatif soit perceptible. Ainsi, sous Windows, seules les versions à partir de Vista peuvent exploiter correctement ces processeurs. Dans ce cas, la version 64 bits est conseillée. Un logiciel qui est **compatible avec le multi-cœur** fonctionne lui beaucoup plus rapidement puisqu'il peut exécuter plusieurs instructions en même temps. C'est le cas notamment des logiciels de retouche photo ou les jeux vidéo. Si vous n'utilisez votre ordinateur que pour consulter vos mails ou regarder des vidéos, vous n'avez pas vraiment besoin d'un **processeur** avec

beaucoup de cœurs. Cependant, à l'heure actuelle, ces logiciels n'arrivent pas à exploiter 100 % les possibilités.

Cœurs comme indice de performance d'un processeur

Le **nombre de cœurs** est le premier indicateur de la **puissance d'un processeur** mais il ne faut pas se focaliser uniquement sur cette donnée, d'autres caractéristiques telles que la fréquence, la mémoire cache, les technologies utilisées...

e. Les inconvénients de l'augmentation du nombre de cœurs

Un espace restreint et goulot d'étranglement

Au fur et à mesure que les microprocesseurs se sont miniaturisés, les réseaux de connexions entre les différents éléments constitutifs de ces puces se sont densifiés. Pour cette raison, l'augmentation du nombre de cœurs sur les puces afin d'améliorer les performances amène inévitablement à la situation où les interconnexions deviennent le goulot d'étranglement de cette quête de performances.

Ainsi, l'utilisation de circuits intégrés en 3D est une solution permettant de contourner le manque d'espace sur les puces.

Problème de chaleur

Bien vrai qu'un processeur multi-cœurs ne rencontre pas les problèmes d'alimentation et de surchauffe de son homologue mono cœur, mais l'augmentation de la puissance de calcul et la miniaturisation des puces conduit à des températures très élevées sur ces dernières.

Ces dernières années, la gestion dynamique thermique (DTM) est apparue dans les états de l'art comme une méthode pour faire face à des contraintes thermiques liées aux processeurs. Les méthodes DTM permettent de consulter l'état thermique actuel d'un processeur. Pour ce faire, ces informations sont récoltées soit directement au moyen de capteurs de température, soit indirectement à l'aide des analyseurs de performance. Ensuite, des modifications sont apportées aux paramètres de configuration du processeur de façon à régler sa température de manière appropriée. La plupart des techniques DTM entraînent une certaine forme de perte de performances. Par exemple, en diminuant la largeur du pipeline, la quantité d'énergie dissipée sera diminuée également, mais, en contrepartie, les cycles par instruction seront augmentés. La recherche sur les techniques de DTM tente de maîtriser les contraintes thermiques, tout en minimisant la dégradation des performances. L'une des techniques importantes en DTM est la capacité de contrôler

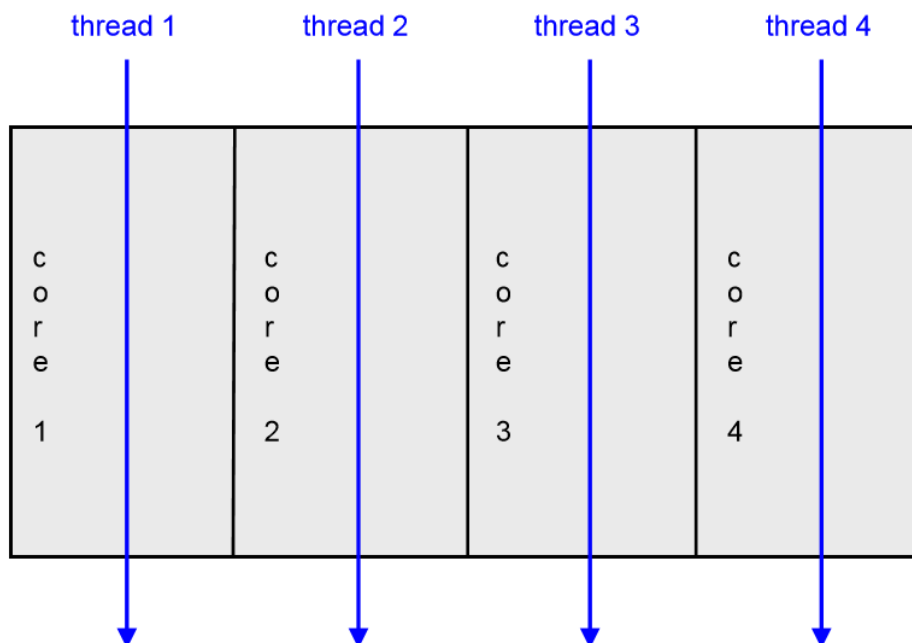
dynamiquement l'exploitation physique des paramètres, à savoir, la fréquence et la tension de chaque cœur.

Gestion des ressources

L'augmentation du nombre de cœurs dans les microprocesseurs est une conséquence de l'évolution du matériel. On note cependant de nombreuses répercussions sur les systèmes d'exploitation et plus précisément sur la gestion des ressources par ces derniers. Dans le cas des processeurs mono-cœur, le modèle de Von-Neumann a su s'imposer comme un modèle de référence tant qu'au niveau des logiciels qu'un niveau des systèmes d'exploitations grâce à sa simplicité.

8. Multithreading simultané et architecture multi-cœurs

Cette technique est une combinaison de l'architecture multi-cœur et de l'architecture multithreading. Elle consiste à exécuter plusieurs threads sur un même cœur. Elle est illustrée par le schéma ci-dessous.



Multi cœurs sans hyperthreading

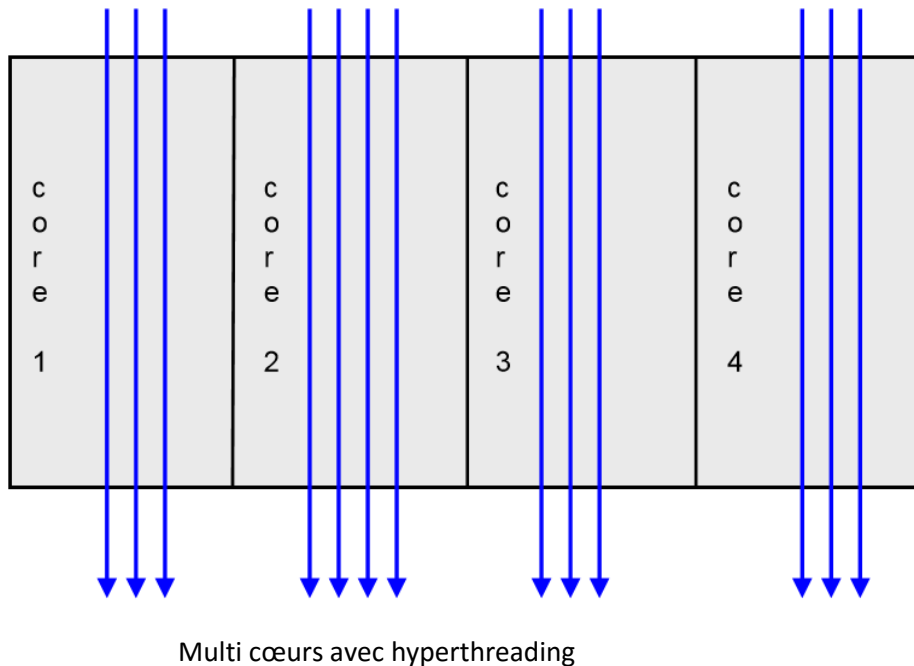


Figure 13 : Représentation du multi-cœur avec hyperthreading

9. Les processeurs vectoriels

Un **processeur vectoriel** est un [processeur](#) possédant diverses fonctionnalités architecturales lui permettant d'améliorer l'exécution de programmes utilisant massivement des tableaux, des matrices, et qui permet de profiter du parallélisme inhérent à l'usage de ces derniers.

Développé pour des applications scientifiques et exploité par les machines Cray et les supercalculateurs qui lui feront suite, ce type d'architecture a rapidement montré ses avantages pour des applications grand public (on peut citer la manipulation d'images). Elle est implémentée en partie dans les processeurs grand publics par des instructions SIMD, soit grâce à une unité de calcul vectoriel dédiée (Altivec), soit simulée par des instructions bas niveau de type vectoriel (MMX/SSE).

Les processeurs vectoriels peuvent être vus comme des processeurs normaux, auxquels on a ajouté un certain nombre d'instructions optimisées pour la gestion des tableaux. Ces instructions optimisées pour les tableaux peuvent être vues comme des variantes d'instructions normales, mais optimisées pour traiter de plus grandes données (pour les accès mémoires), ou capables d'effectuer des opérations en parallèle. Ces instructions sont appelées des instructions vectorielles. Il en existe plusieurs types, qu'on va présenter dans ce qui suit.

Les processeurs vectoriels utilisent une architecture qualifiée de SIMD (Single Instruction Multiple Data). Ils sont capables d'effectuer la même opération sur plusieurs données différentes en même temps.

Les processeurs actuels intègrent des unités de calcul vectoriel dédiées au multimédia :

- MMX (Multi-Media eXtension) intégrée au Pentium MMX (registres 64 bits, entiers)

- 3D Now ! version MMX/SSE des processeurs AMD apparue sur les K6-II (64 bits)
- SSE (Streaming Simd Extension) apparue sur les Pentium III (128 bits, entiers ou flottants 32 bits)
- SSE2 évolution du SSE sur le Pentium 4 (flottants 64 bits)

Des consoles de jeu sont équipées de processeurs vectoriels. Par exemple, la Playstation 2 dispose d'un processeur dédié au calcul vectoriel alors que la Playstation 3 est équipée du processeur vectoriel Cell.

XI. LES PROCESSEURS A ARCHITECTURE ARM

Les **architectures ARM** sont des architectures matérielles RISC 32 bits (ARMv1 à ARMv7) et 64 bits (ARMv8) développées par ARM Ltd depuis 1990 et introduites à partir de 1983 par Acorn Computers.

Dotés d'une architecture relativement plus simple que d'autres familles de processeurs, et bénéficiant d'une faible consommation électrique, les processeurs ARM sont devenus dominants dans le domaine de l'informatique embarquée, en particulier la téléphonie mobile et les tablettes.

Ces processeurs sont fabriqués sous licence par un grand nombre de constructeurs.

Aujourd'hui, ARM est surtout connu pour ses systèmes sur puce (SoC), intégrant sur une seule puce : microprocesseur, processeur graphique (GPU), DSP, FPU, SIMD, et contrôleur de périphériques. Ceux-ci sont présents dans la majorité des smartphones et tablettes. ARM propose des architectures, qui sont vendues sous licence de propriété intellectuelle aux concepteurs. Ils proposent différentes options dans lesquelles les constructeurs peuvent prendre ce qui les intéresse pour compléter avec leurs options propres ou de concepteurs tiers. ARM propose ainsi pour les SoC les plus récents, les microprocesseurs Cortex (Cortex-A pour les dispositifs portables (smartphones et tablettes), Cortex-M pour le couplage à un microcontrôleur, Cortex-R pour les microprocesseurs temps réel), des processeurs graphiques (Mali), des bus AMBA sous licence libre, ainsi que les divers autres composants nécessaires à la composition du SoC complet. Certains constructeurs comme Apple, modifient certains composants du microprocesseur en mélangeant plusieurs architectures processeur ARM (l'Apple A6 par exemple, mixe les technologies de microprocesseur Cortex-A9 et Cortex-A15). Le processeur le Snapdragon 835 de Qualcomm (Exynos 8895) équipe les Samsung Galaxy S8. NVIDIA Tegra pour tablette Windows RT de Microsoft.

De nombreux systèmes d'exploitation sont compatibles avec cette architecture :

- Symbian S60 avec les Nokia N97 ou Samsung Player HD ;
- iOS avec l'iPhone et l'iPad ;
- Linux, avec la plupart des distributions ou avec Android ;
- BlackBerry OS avec les BlackBerry
- Windows CE, Windows Phone 7 et Windows RT, une version de Windows 8.
- le système PlayStation Vita
- ReactOS

- Risc OS
- etc.

Depuis, l'architecture ARM a rencontré un succès qui ne se dément pas, notamment ces dernières années. Des processeurs ARM, on en trouve dans tous les smartphones récents : iPhone 4, Samsung Galaxy S, tout comme dans l'iPad, le Nokia N97 ou le Nexus One de Google. Mais la société ARM Ltd ne fabrique pas, à proprement parler, les processeurs. Elle conçoit les architectures des processeurs, leurs plans en quelque sorte. Ce sont les clients qui, après achat des licences, produisent les processeurs. Par exemple, Samsung fabrique un processeur ARM pour ses téléphones haut de gamme (Samsung Galaxy S et Wave), reposant sur l'architecture ARM Cortex-A8, cadencée à 1 GHz.



Figure 14 : Processeurs à architecture ARM

XII. LES PROCESSEURS SPECIAUX

1. Le microcontrôleur

Ce sont des systèmes minimum sur une seule puce. Ils contiennent un CPU, de la RAM, de la ROM et des ports d'Entrée/Sorties (parallèles, séries, I2C, etc..).

Un **microcontrôleur** (en notation abrégée **µc**) est un circuit intégré qui rassemble les éléments essentiels d'un ordinateur : processeur, mémoires (mémoire morte et mémoire vive), unités périphériques et interfaces d'entrées-sorties. Les microcontrôleurs se caractérisent par un plus haut degré d'intégration, une plus faible consommation électrique, une vitesse de fonctionnement plus faible (de quelques mégahertz jusqu'à plus d'un gigahertz) et un coût réduit par rapport aux microprocesseurs polyvalents utilisés dans les ordinateurs personnels.

Par rapport à des systèmes électroniques à base de microprocesseurs et autres composants séparés, les microcontrôleurs permettent de diminuer la taille, la consommation électrique et le coût des produits. Ils ont ainsi permis de démocratiser l'utilisation de l'informatique dans un grand nombre de produits et de procédés.

Il est adapté pour répondre au mieux au besoin des applications embarquées (appareil électroménagers, chaîne d'acquisition, lecteur carte à puce, téléphone mobile, télécommande, jouet, etc...). Il est par contre généralement moins puissant en termes de rapidité, de taille de données traitables ou de taille de mémoire adressable qu'un microprocesseur.

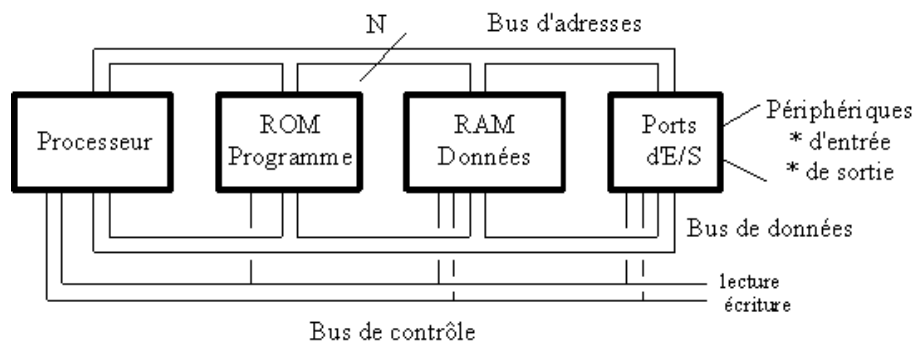


Figure 15 : Structure d'un système à microprocesseur

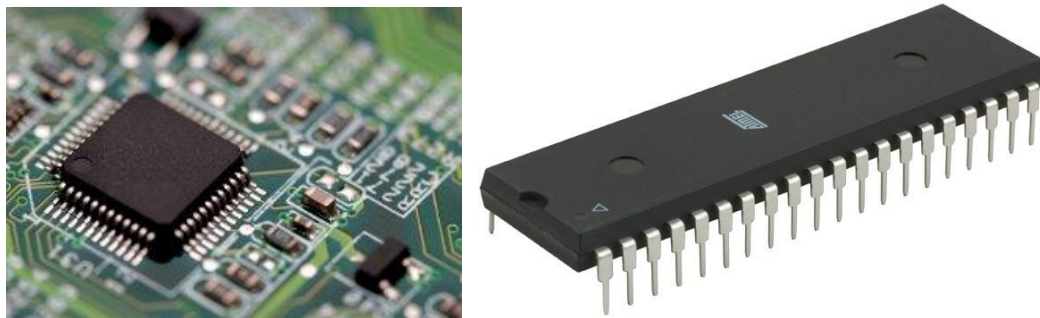


Figure 16 : Microcontrôleur

2. Le processeur de signal

Le processeur de signal est beaucoup plus spécialisé. Alors qu'un microprocesseur n'est pas conçu pour une application spécifique, le processeur DSP (Digital Signal Processor) est optimisé pour effectuer du traitement numérique du signal (calcul de FFT, convolution, filtrage numérique, etc...).

Les domaines d'application des D.S.P étaient à l'origine les télécommunications et le secteur militaire. Aujourd'hui, les applications se sont diversifiées vers le multimédia (lecteur CD, MP3, etc..) l'électronique grand public (télévision numérique, téléphone portable, etc...), l'automatique, l'instrumentation, l'électronique automobile, etc...

Date de chat en ligne: le 20 juillet 2017 à partir de 10h

XIII. EVALUATIONS

Date de l'évaluation en ligne : le 22 juillet 2017