Un ordinateur est constitué de processeurs, de mémoires et des dispositifs d'entrée/sortie interconnectés.

1. Le processeur ou unité centrale

L'architecture d'un ordinateur très simple est représentée la figure (Fig.01.).

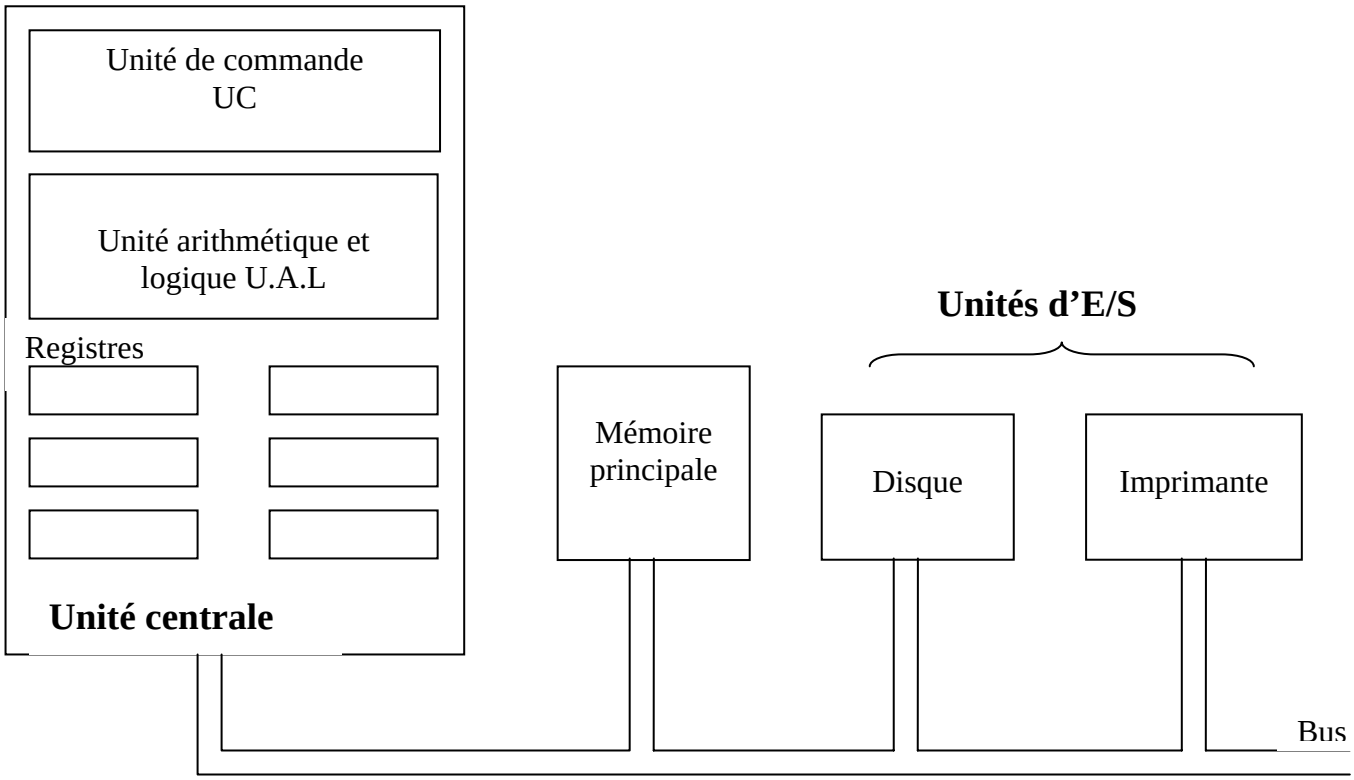

Fig.*01. Organisation d'un ordinateur simple comportant une UC et d'E/S.*

1.1. Processeur CPU : appelé aussi unité central UC, et le cerveau de l'ordinateur. Son rôle est d'exécuter les programmes stockés en mémoire principale en chargeant les instructions en les décodant et en les exécutions l'une après l'autre. Les composants d'un ordinateur sont relies entre eux par un bus qu'est constitué d'un ensemble de fils électriques assurant la transmission des signaux d'adresses, de données et de commande. On trouve en générale plusieurs bus dans un ordinateur : un bus externe reliant l'UC à la MC et aux dispositifs d'E/S, un bus interne interconnectant les éléments fonctionnels internes a l'UC.

L'UC comprend deux unités distinctes: une unité de commande dont le rôle est de charger les instruction situées en mémoire principale et de décoder et une unité arithmétique et logique UAL qui est responsable de l'exécution des opérations indiquées par les instructions.

L'UC dispose d'une mémoire de travail privée qui lui permet de stoker des résultats temporaires ou des information de commande. Cette mémoire comprend notamment un certain nombre de registres, chacun ayant une fonction particulière. Un registre est une mémoire à accès rapide, il peut lu ou écrit (Ex : CO, RI, registres généraux).

1.2. Organisation de l'unité centrale :

La figure (Fig.02.) montre l'organisation interne partielle d'une UC VON NEWMAN typique. On appelle cette partie le chemin des données (*data path*). Celui ci comprend un ensemble de registres généraux (1 à 32 registres) et une UAL. Plusieurs bus interconnectent ces divers éléments. Les registres généraux alimentent en données deux registres aux entrées de l'UAL, appelés A et B. Ces registres ont pour rôle de stocker temporairement les données sur les entrées de l'UAL pendant qu'elle effectue un traitement, le chemin des données est un élément important de l'architecture des ordinateurs.

L'UAL exécute les additions, les soustractions et d'autres opérations simples portant sur ses les données présentes sur ses entrées. Elle place le résultat du traitement dans un registre de sortie qui peut être recopié dans un registre général ou en mémoire principale. Tous les chemins de données des ordinateurs ne disposent pas de registres en entrée de l'UAL , ni même d'un registre de sortie.

On peut regrouper les instructions en 2 catégories :

- instruction registres mémoire: permet de charger des mots mémoires dans des registres généraux qui pourront être utilisés par d'autres instructions comme entrées de l'UAL. Il existe aussi des instructions registre mémoire qui permettent d'écrire en mémoire le contenu d'un registre.
- instruction registres - registres: charge deux opérandes pris dans les registres, les place dans les registres d'entrée de l'UAL, exécute sur eux une certaine opération et range le résultat dans un registre.

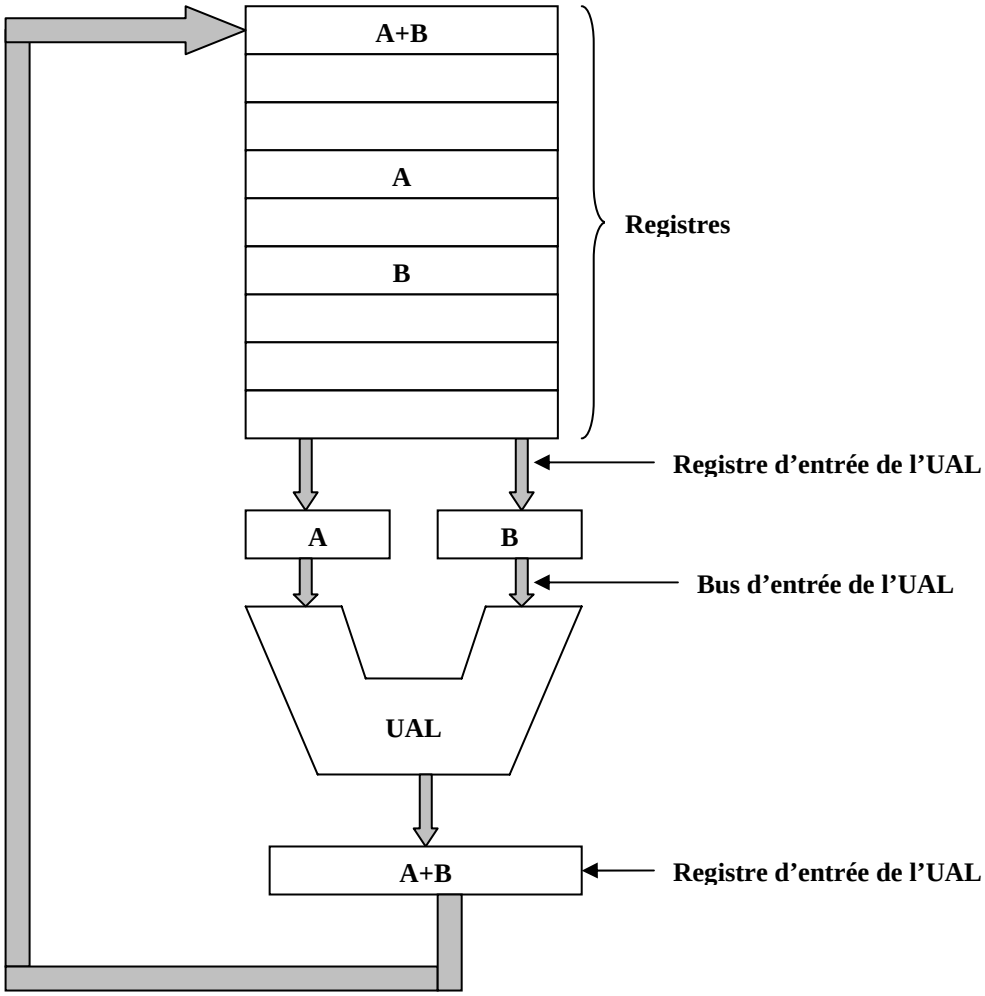

Fig .02. Chemin des données d'une machine VON NEWMAN typique.

Exécution d'une instruction

L'exécution d'une instruction par l'UC passe par les étapes suivants :

1. Charger la prochaine instruction à exécuter depuis la mémoire dans le registre instruction.
2. Modifier le compteur ordinal pour qu'il pointe sur l'instruction suivante.
3. Décoder (analyser) l'instruction que l'on vient de charger.
4. Localiser en mémoire d'éventuelles données nécessaires à l'instruction.
5. Charger, si nécessaire, les données dans des registres généraux de l'UC .
6. Exécuter l'instruction.
7. Revenir à l'étape 1 pour entreprendre l'exécution de l'instruction suivante.

Cette suite d'étapes est appelé "Cycle de chargement décodage exécution" (fetch decode execute cycle).

1.3. Architecture RISC et CISC

Le concept **RISC** (Reduced Instruction Set Computer)est né de la construction suivante: dans 80 % des cas, un processeur n'utilise que 20 % de son jeu d'instructions. Les instructions les plus utilisées sont celles de transfert entre l'unité centrale et la mémoire et les branchements aux sous programmes.

L'évolution des processeurs a donné naissance à des processeurs ayant des jeux d'instructions de plus en plus complexes. Ces instructions complexes sont des programmes micro codés dont le décodage est effectué à l'intérieur du processeur et dont l'exécution peut prendre plusieurs cycles d'horloge. Ces machines sont appelées machines à jeu d'instruction complexes **CISC** (Complex Instruction Set Computer). Le temps passé au décodage n'est pas pénalisant comparé au temps d'accès à la mémoire.

Les caractéristiques fondamentales d'un processeur **RISC** sont **:**

- Exécution des instructions en un seul cycle d'horloge.
- Simplification du format des instructions (généralement 32 bits).
- Réduction et simplification du jeu d'instruction (modes d'adressage limités).
- Utilisation intensive des registres.
- Séquenceur câblé.

C'est à partir des années 80 que de nombreux développements liés au **RISC** ont eu lieu. Les instructions complexes des processeurs **CISC** sont remplacées par des séquences d'instructions simples. Un programme génère donc beaucoup plus d'instructions pour un processeur **RISC** que pour un **CISC**. Les temps d'accès à la mémoire jouent un rôle plus important d'où l'utilisation de mémoires caches et d'un grand nombre de registres.

Un des principaux problèmes rencontrés lors du développement des processeurs **RISC** est lié à la compilation des programmes et plus particulièrement à la génération du code (une grande différence entre le programme source et les instructions simples) à l'inverse de **CISC**. Le processeur **RISC** est plus simple que le **CISC** ce qui entrain un temps de conception plus court, des circuits plus petits laissant la place pour des registres, des co-processeurs permettant d'augmenter les performances.

1.3.1. Principes de conception des ordinateurs actuels

Les principes de conception associés aux machines **RISC** sont largement adoptés et reconnus par les constructeurs d'ordinateurs et des composants électroniques intégrés complexes (comme par exemple les microprocesseurs VLSI). Les principes de conception évoqués constituent ce qu'on appelle les **RISC** design principales.

a) Toute instruction est traitée directement par des composants matériels : Toute instruction est traitée par le matériel, sans aucune interprétation logicielle. C'est le principe de base des machines **RISC**. Cette élimination d'un niveau d'interprétation augmente considérablement la vitesse d'exécution de la plupart des instructions. Les ordinateurs qui implémentent des instructions complexes disposent de deux parties fonctionnelles:

> L'une de type **RISC**, l'autre de type **CISC** comprenant une mémoire de microprogrammation et des micro-instructions pour l'interprétation des instructions complexes. Cela entraîne une baisse des performances (acceptable).

b) maximiser la vitesse d'exécution des instructions : Les ordinateurs actuels (modernes) utilisent des techniques pour maximiser leurs performances. L'objectif principal consiste à exécuter un grand nombre d'instructions par seconde (un ordinateur peut 500 MIPS « million instruction par seconde »), pour atteindre de tel performances, il faut mettre en œuvre le parallélisme dans ces processeurs (traiter plusieurs instructions simultanément).

c) les instructions doivent être simples à décoder : Un des points critiques dans la vitesse d'exécution des instructions est relatif à leur décodage (déterminer le type d'instruction). Cela conduit à disposer d'instructions à structure régulière : format de longueur fixe avec un faible nombre de champs (pas différence entre les formats d'instructions).

d) seules les instructions de rangements et de élargement doivent faire référence à la mémoire principale : Pour rompre avec l'exécution séquentielle des instructions, il faut utiliser au maximum les registres internes du processeur et faire en sorte que tous les opérandes proviennent des registres du processeur et y soient stockés. Les opérations de déplacement d'opérandes entre les registres et la MC sont effectuées par des instructions spéciales de type LOAD et STORE. Seules ces instructions peuvent faire référence à la mémoire. Comme les instructions d'accès à la mémoire principale prennent un temps important comparativement à celles faisant référence à des registres, il est possible de les effectuer en parallèle avec d'autres. Cela réduit d'autant leur influence sur les performances.

1.4. Parallélisme

Les architectes d'ordinateurs ont commencé par améliorer les composants en les rendant de plus en plus rapides (vitesse d'horloge) afin d'améliorer les performances des machines. Toutefois, il y a une limite technologique à ne pas dépasser. Donc, il faut trouver d'autres moyens pour améliorer les performances. L'un d'entre eux consiste à paralléliser des actions. C'est la notion de parallélisme qui permet d'augmenter la vitesse d'exécution des programmes.

1.4.1. Parallélisme des instructions : Consiste à lancer plusieurs instructions simultanément, ce qui permet d'augmenter le nombre d'instructions exécutées en une seconde, donc les performances de l'ordinateur.

a) Technique du pipeline ou pipelining : Le temps de recherche des instructions en MC est le facteur le plus pénalisant dans l'exécution d'un programme. Pour minimiser l'effet de ce facteur, les concepteurs utilisent une technique déjà mise en œuvre sur *IBM Stretch* en 1959 qui consiste à chercher les instructions à traiter par le processeur avant que celui-ci n'en ait besoin. Ainsi, quand le processeur avait besoin d'une instruction, elle était disponible sans aucune perte de temps. Les instructions étaient enregistrées dans un ensemble de registres constituant une mémoire tampon (buffer) d'instructions en file d'attente appelée « Prefetch buffer ». De cette façon, dés qu'une instruction devait être traitée, il suffisait de la prendre dans le buffer plutôt que de lancer une opération d'accès à la mémoire.

Ce mécanisme est un mécanisme d'anticipation (Prefetching) qui convient au processus d'exécution des instructions qui comprend plus d'étapes fonctionnelles indépendantes et séquentielles. L'anticipation découpe ces séquences fonctionnelles en deux phases ; la recherche des instructions en mémoire (fetch cycle) et leur exécution (execute cycle). Le pipeline adopte cette stratégie fonctionnelle. Toutefois, plutôt que de diviser l'exécution des instructions en deux phases, il la découpe en plusieurs phases, chacune étant à la charge d'une unité fonctionnelle (matérielle) spécifique. Chaque unité fonctionnelle est indépendante, elle exécute sa tâche en simultanéité avec les autres unités fonctionnelles. La figure (Fig.03.) illustre le principe d'un pipeline de cinq unités fonctionnelles (étages).

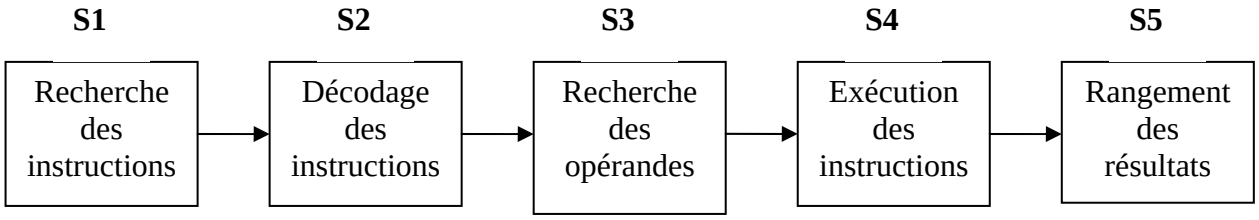

Fig.03. Pipeline à cinq étages (unités fonctionnelles)

L'étage 1: Va chercher les instructions en en MC et les range dans un buffer.

L'étage 2: Décode l'instruction qui va être exécutée, détermine son type et les opérandes dont elle a besoin.

L'étage 3: Localise les opérandes et va les chercher, soit en MC, soit dans un registre selon leur localisation.

L'étage 4: Réalise effectivement l'exécution de l'instruction, en faisant subir aux opérandes le traitement défini par l'instruction.

L'étage 5: Transfère le résultat du traitement dans le registre approprié ou en MC.

La figure (Fig.04.) montre le fonctionnement des pipelines.

S1	1	2	3	4	5	6	7	8	9
S2		1	2	3	4	5	6	7	8
S3			1	2	3	4	5	6	7
S4				1	2	3	4	5	6
S5					1	2	3	4	5

Fig .04. État de chaque étage en fonction de temps

Pendant le cycle d'horloge 1, l'étage S1 travaille sur l'instruction 1 qui consiste à aller la chercher en mémoire. Pendant le cycle d'horloge 2, l'étage S2 décode l'instruction 1 et l'étage S1 va chercher en mémoire l'instruction 2. Au cours de cycle d'horloge 3, l'étage S3 va chercher les opérandes nécessaires à l'instruction 1, l'étage S2 décode l'instruction 2 et l'étage S1 va chercher en mémoire l'instruction 3. Pendant le cycle d'horloge 4, l'étage S4 exécute l'instruction 1, l'étage S3 va chercher les opérandes de l'instruction 2, l'étage S2 décode l'instruction 3 et l'étage S1 va chercher en mémoire l'instruction 4. Au cours de cycle d'horloge 5, l'étage S5 écrit le résultat du traitement de l'instruction1 dans le registre approprié ou en mémoire. Les autres étages continuent leurs traitements intermédiaires sur cette d'exécution des instructions qu'est le pipeline.

Si on suppose que le cycle d'horloge de cette machine est de 2 ns cela signifie qu'il faut 10 ns pour qu'une instruction progresse et travers le pipeline de l'étage S1 à l'étage S5 où se termine son traitement. Le temps d'exécution d'une instruction est de 10 ns, donc cette machine fonctionne à 100 MIPS. En réalité ce n'est pas le cas, la performance est bien meilleure. En effet, à chaque nouveau cycle d'horloge (toute les 2ns), une nouvelle instruction est complètement terminée (exécutée), cela correspond à une vitesse de 500 MIPS.

Le mécanisme de pipeline fait un compromis la latence (le temps d'exécution globale d'une instruction) el la bande passante du processeur (la puissance du processeur en MIPS). Avec un temps de cycle d'horloge de *T* ns, un pipeline à n étages présente une latence de *nT* ns et une bande passante de *1000/T* MIPS. La latence présente le temps d'amorçage du pipeline. Ce n'est qu'a l'issue de ce temps de latence (*nT*) que le pipeline fonctionne à la vitesse nominale de *1000/T* MIPS.

b) Architecture superscalaire: L'utilisation de deux pipelines est plus efficace que celle à un seul pipeline. La figure (Fig.05.) illustre un exemple d'une telle architecture à double pipeline, chacun des deux pipelines étant semblable à celui de la figure (Fig.03.).

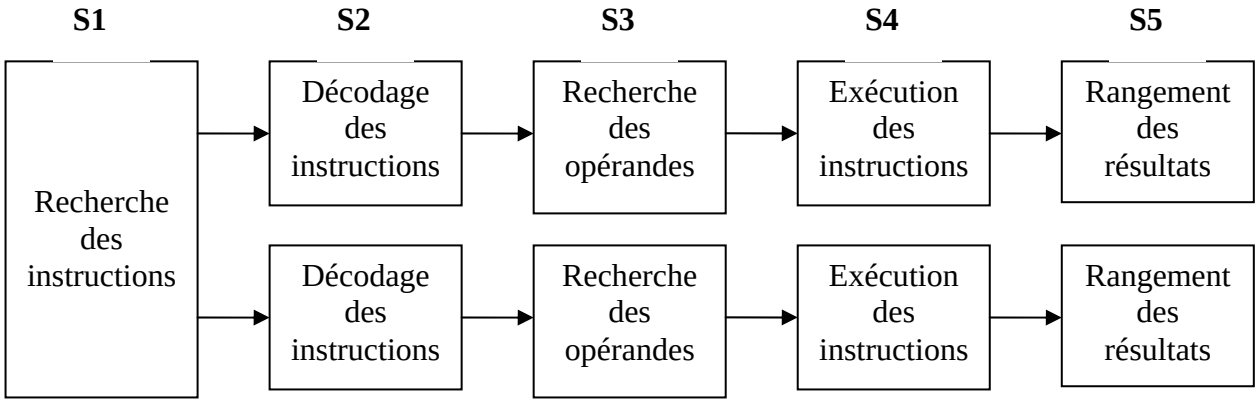

Fig .05. Exemple d'architecture à double pipeline à cinq étages
Disposant d'une seule unité de recherche d'instructions.

Dans cette architecture, une unité fonctionnelle unique est chargée d'aller chercher les instructions par paire puis d'ordonnancer leur exécution en alimentant en instructions les deux pipelines. Cette unité S1 place une instruction dans le premier pipeline, l'autre dans le second pipeline. Chaque pipeline est autonome et dispose de ses propres ressources de traitement. Pour que deux instructions puissent être exécutées en parallèle, il ne faut pas qu'elles aient d'interrelation ou de dépendance mutuelle. C'est-à-dire qu'elles ne doivent pas utiliser de ressources communes, ce qui pouvait générer un conflit d'accès, et qu'elles ne dépendent pas l'une ou l'autre d'un résultat de traitement de l'autre. Comme dans le cas d'un pipeline unique, c'est le compilateur qui doit garantir ces conditions d'indépendance fonctionnelle et qui doit organiser l'ordonnancement d'exécution des instructions. En outre, du matériel supplémentaire est nécessaire pour détecter d'éventuels conflits et les régler s'ils e produisent pendant l'exécution des instructions dans les deux pipelines.

Les pipelines sont presque toujours utilisés sans les, machines RISC. Intel 486 utilise un pipeline tandis que le Pentium comprend deux pipelines à cinq étages quelque peu similaires à ceux de la figure (Fig.04.). La division des étages 2 et 3 (appelés chez Intel décodage 1 et décodage 2) sont assez différentes de celle présentée sur la figure (Fig.04.). Le pipeline principal appelé (Pipeline u) peut exécuter n'importe laquelle des instructions du processeur Intel. L'autre pipeline appelé (Pipeline v) ne peut exécuter que les instructions simples portant sur des nombres entiers et quelques instructions à virgule flottante.

Le compilateur du système Pentium détermine si deux instructions sont compatibles (c'est-à-dire si elles peuvent s'exécuter en parallèle) en utilisant des règles complexes. Dans le cas où l'une des deux instructions d'une paire est incompatible, voire si elle est complexe, seule l'autre est exécutée dans le pipeline *u*. Quand à

celle qui reste, elle est retardée d'un ou plusieurs cycles d'horloge. Ainsi, l'exécution des instructions n'est plus ordonnée. Il s'agit du concept d'*ordonnancement dynamique* des instructions réalisé par le compilateur du système Pentium. Des études ont montré que le gain de performance est de l'ordre de deux, pour un même programme traitant des nombres entiers entre un Pentium et un 486 fonctionnant dans les mêmes conditions d'environnement matériel et avec une même fréquence d'horloge.

Des solutions à quatre pipelines ont été élaborées, mais la nécessité d'un nombre important de modules matériels (quatre fois) implique des coûts supplémentaires qui n'impliquent pas forcément un gain identique en performance.

D'autres approches existent, elles consistent á disposer les le pipeline (notamment á l'étage S4) de plusieurs unités de traitement spécialisées á haute performance (Fig.06.).

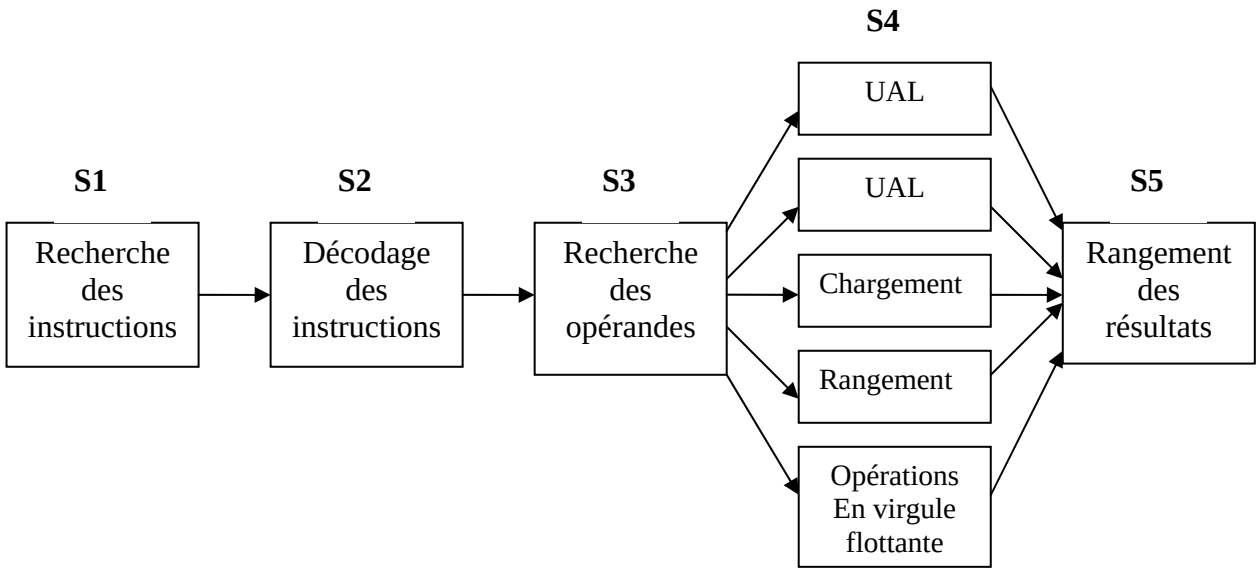

Fig.06. Exemple Architecture comprenant cinq unités de traitement spécialisées.

Le Pentium II d'Intel comporte une structure de pipeline similaire, cette organisation est appelée architecture superscalaire, elle permet de traiter plus instructions simultanément. Le problème qui se fait á jour avec l'architecture superscalaire, c'est ce que, selon la complexité des instructions en cours, l'étage S3 peut exécuter sa tâche plus rapidement que l'étage S4. Dans l'étage S4, certaines instructions peuvent demander plusieurs cycles d'horloge. Ainsi pour éviter que le système se bloque (dans une situation d'attente de fin d'exécution d'instruction), plus UAL sont mises en parallèle dans l'étage S4. Dés lors on met en œuvre, dans S4 un parallélisme du traitement de type parallélisme relatif au processeur.

1.4.2. Parallélisme du processeur: Le parallélisme des instructions permettait une augmentation des performances qui ne peut toutes fois que revenant supérieur a 10. Pour obtenir des gains de performances de l'ordre de 50 ou 100,vue plus, il faut utiliser d'autres techniques qui consistent a faire travailler plus processeurs (ou ordinateurs) en parallèle. Ce parallélisme consiste á mettre en place plusieurs processeurs et á les faire travailler simultanément sur un même programme, cela revient á réduire le temps de traitement de ce programme donc à améliorer les performances de l'ordinateur.

a) Processeur matriciel (array processor) et vectoriel (vector processor) : En science physique, de nombreux problèmes sont résolus en utilisant le calcul matriciel ou vectoriel, qui reposent sur des structures régulières d'opérations mathématiques. Le plus souvent les mêmes calculs sont réalisés au même instant sur des données différentes. La régularité et la structure des programmes qui réalisent ces traitements en font d'excellents candidats à la mise en œuvre du parallélisme pour accélérer leur vitesse de traitement. On distingue deux

méthodes de parallélisation permettant d'exécuter très rapidement de gros programmes scientifiques l'une peut être vue comme l'exécution d'un processeur conventionnel, l'autre comme un véritable ordinateur parallèle.

Un Processeur matriciel : Est constitué d'un nombre important de processeurs identiques qui réalisent une même séquence d'instructions sur des données différentes. L'Illiac IV a été le premier ordinateur matriciel (1972) figure (Fig.07.).

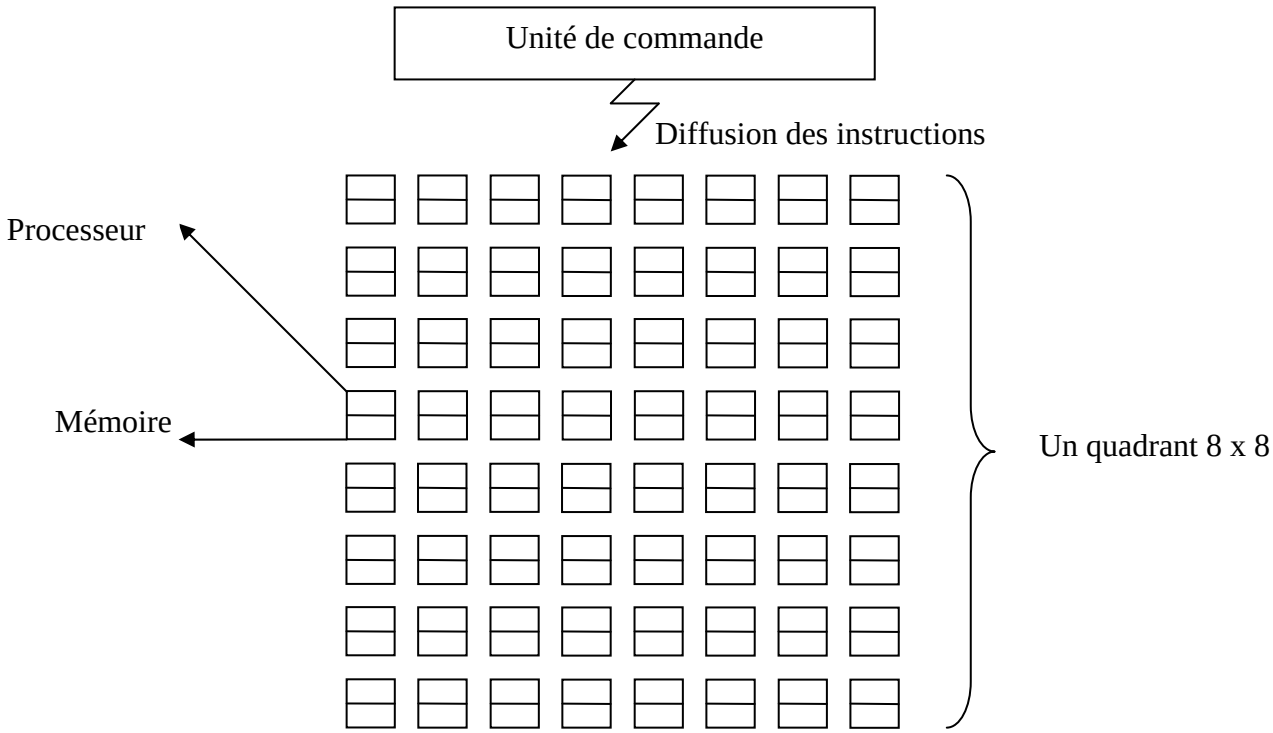

Fig.07. Le processeur matriciel type de l'Illiac IV .

A l'origine, l'ordinateur devait comprendre quatre quadrants, chaque quadrant étant composé d'une matrice 8x8 d'éléments processeurs/mémoires. Une unité de commande était associée à chaque quadrant. Elle avait pour rôle de diffuser les instructions dans le quadrant et d'organiser leur exécution en parfait synchronisme temporel. Chaque processeur utilisait les données issues de sa propre mémoire. Ces données étaient enregistrées dans les mémoires privées des processeurs lors d'une phase d'initialisation. Étant donné le coût d'une telle machine, un seul quadrant fut construit. Celui ci atteignait mémoires les performances remarquables de 50 MFLOPS (million d'opérations en VF par second). Il fut montré que la machine d'origine aurait atteint les performances de 1 GFLOPS (la machine la plus puissante du monde).

Un processeur vectoriel: Est similaire à un processeur matriciel. Il est très efficace pour exécuter une séquence d'opérations portant sur des paires de données. Mais, contrairement à ce qui se passe avec un processeur matriciel, sur un processeur vectoriel toutes les opérations d'addition, par exemple, sont exécutées par un unique additionneur pipeline à plusieurs étages (Cray-1 en 1974).

Les processeurs matriciels et vectoriels travaillent tous deux sur des tableaux de données, exécutent des instructions simples, comme par exemple l'addition par paire d'éléments de 2 vecteurs ou 2 matrices. Alors que le processeur matriciel doit disposer d'autant d'additionneurs que de données pressentes dans le vecteur, le processeur vectoriel dispose de registres appelés registres vectoriels constitués chacun d'un ensemble de registres conventionnels pouvant être chargés avec des données extraites de la mémoire en une seule instruction. Une instruction d'addition vectorielle réalise l'addition de paire de données issues de deux registres vectorielles en les injectant dans un additionneur pipeline. Le résultat de l'addition est un nouveau vecteur qui peut être stocké dans un registre vectoriel pour servir d'opérande à une opération vectorielle, soit en mémoire principale.

Les processeurs matriciels sont plus performants que les processeurs vectoriels mais ils sont aussi beaucoup plus chers et beaucoup plus complexe à programmer. Un processeur vectoriel peut être associe a un processeur conventionnel. Dans ce cas, toutes les parties d'un programme qui peuvent être vectorisées sont sous traitées au processeur vectoriel, qui les exécute bien plus rapidement que le processeur conventionnel. Les autres parties du programme sont traitées par le processeur conventionnel.

b) Multiprocesseur: Dans un processeur matriciel ou vectoriel seules les éléments de traitements (les UAL) sont dupliqués. Ils se trouvent tous sous la tutelle fonctionnelle d'une unique unité de commande, qui pilote et organise la parallélisation d'exécution des instructions. Au contraire, dans système multiprocesseur tous les processeurs sont autonomes et indépendants. Ils disposent chacun de toutes les fonctions propose a un processeur : une UAL et une unité de commande, le seule lien qui les unit est une mémoire principale partagée.

Dans un multiprocesseur, chaque processeur peut a priori lire ou écrire en mémoire partagée, c'est au système d'exploitation de gérer (par logiciel) cette mémoire commune pour éviter qu'un processeur n'écrive de façon intempestive dans la zone attribuée a un autre processeur. Il existe diverses implémentations de systèmes multiprocesseurs, la plus simple consiste a disposer les processeurs et la mémoire partagée sur un bus unique (Fig.08.).

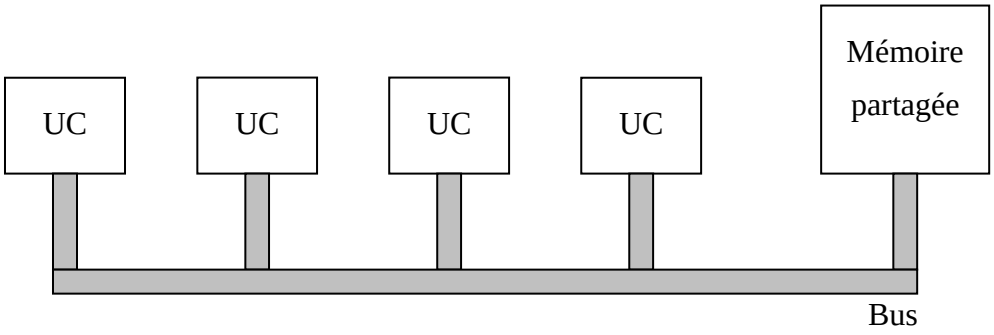

Fig.08. Un multiprocesseur à bus unique

Avec un nombre important de processeurs, ils passent leurs temps à tenter d'accéder à la mémoire partagée dans cette architecture. C'est une source de conflits permanents. Pour réduire ce problème de contention et augmenter les performances, les concepteurs ont proposé d'autres architectures Multiprocesseurs. L'une d'entre elle est représentée à la figure (Fig.09.)

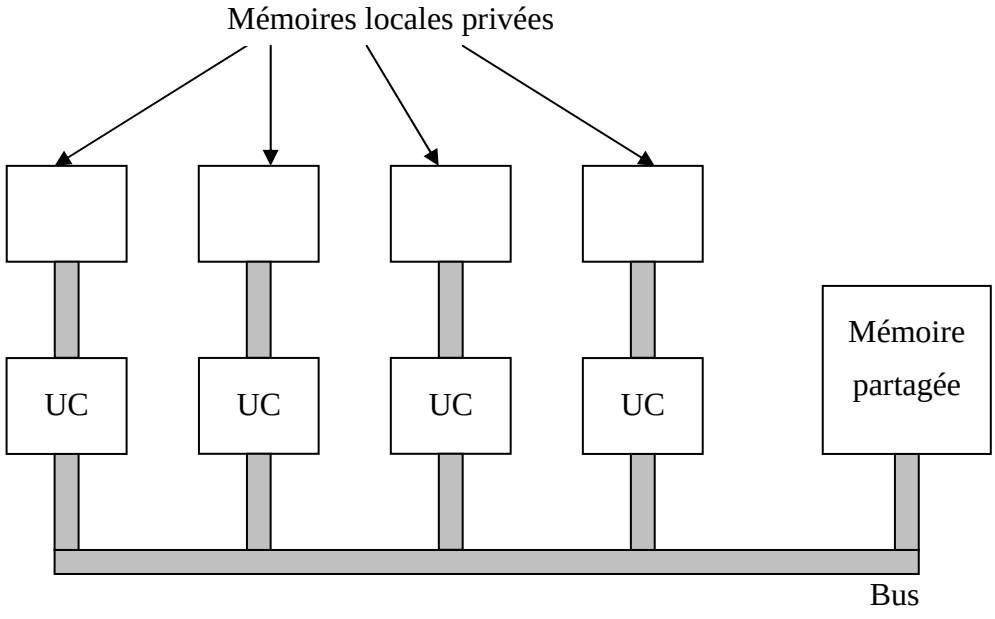

Fig.09. Un multiprocesseur mémoire privée

Dans cette organisation chaque processeur dispose d'une mémoire privée inaccessible aux autres processeurs. Cette mémoire peut contenir des instructions de programme et des données qui n'ont pas a être partagées. L'accès à cette mémoire se fait par un bus privé auquel ne peuvent pas accéder les autres processeurs. D'autres schémas d'architecture sont également utilisés. Ils sont basés notamment sur des systèmes de mémoire cache.

Les multiprocesseurs présentent des avantages par rapport aux ordinateurs parallèles. Ils utilisent un modèle de programmation basée sur un mémoire commune qui offre un nombre d'avantages et facilite l'écriture des programmes.

c) Système-Multi Ordinateur (Multi Computer) : Bien que les « petits » multiprocesseurs comportant un nombre de processeurs inférieur ou égale à 64 processeurs soient relativement souples à concevoir, les « gros » sont plus complexe et difficiles à construire. La difficulté majeure est dans l'interconnexion des processeurs avec une mémoire partagée. Il faut utiliser des structures d'interconnexion à hautes performances tout en réduisant au maximum la probabilité de conflit d'accès. Pour contourner ces problèmes, de nombreux concepteurs ont décidé d'abandonner totalement le concept de mémoire commune partagée et ont conçu les systèmes parallèles basés sur des ordinateurs autonomes interconnectés en réseau. On appelle ces systèmes, système distribués ou multi-ordinateurs chacun participe au sens du parallélisme a des activités concertées.

La communication entre les ordinateurs s'effectue par des échanges de messages qui peuvent prendre diverses formes : échange de fichiers, courrier électronique . . . etc. Les performances du système global sont en relation étroite avec les performances du réseau de communication qui interconnecte les divers ordinateurs. Plusieurs topologies sont utilisées tel que l'arborescence, l'anneau . . . .etc.

2. La mémoire principale

La mémoire principale est la partie de l'ordinateur dans laquelle sont rangés les programmes et les données. Le processeur lit et écrit de l'information dans la mémoire principale.

2.1. Les bits : L'unité d'information élémentaire représentée par un chiffre binaire appelé bit qui peut valoir 0 ou 1.

2.2. Les adresses mémoires

Une mémoire est formée d'un certain nombre de cellules (ou cases), chacune contenant une certaine information. Chaque cellule a un numéro appelé adresse, qui permet a un programme de la référencier. Si une mémoire a n cellules, les adresses iront de 0 à n-1 .Toutes les cases de la mémoire contiennent le même nombre de bits. Les ordinateurs qui utilisent la numérotation binaire expriment aussi les adresses en binaire. Si une adresse a m bits, le nombre maximum de cellules de la mémoire directement adressable est 2^m. Le nombre de bits d'une adresse ne dépend donc que du nombre de cellules mémoires directement adressables et non pas de leur taille.

La cellule mémoire est la plus petite quantité d'information adressable. La plupart des fabricants se sont mis d'accord sur une cellule de 8 bits qu'on appelle un octet. Les octets sont eux-mêmes regroupés en mots. Un ordinateur à mots de 32 bits a 4 octets par mot alors qu'un ordinateur à 64 bits a 8octet par mot.

Le mot est l'unité d'information sur laquelle opèrent la plupart des instructions. Une machine 32 bits aura des registres à 32 bits et les instructions qui manipulent des mots sont de 32 bits.

2.3. Organisation des octets

Les octets d'un mots peuvent être numérote de gauche à droit ou de droite à gauche. Le choix de numérisation est déterminant dans une mémoire. La figure (Fig.09.) montre la mémoire d'un ordinateur 32 bits dont les octets sont numérotés de gauche à droite ce qui est le cas des ordinateurs SPARC ou des gros mainframes de IBM. La figure (Fig.10.) montre une représentation analogue d'un ordinateur 32 bits où les octets sont numérotés de droite à gauche, choix qui a été adopté pour les microprocesseurs INTEL.

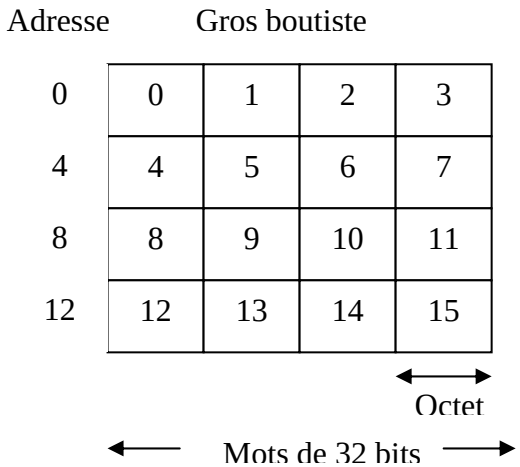

Fig.09. Mémoire «gros-boutiste»

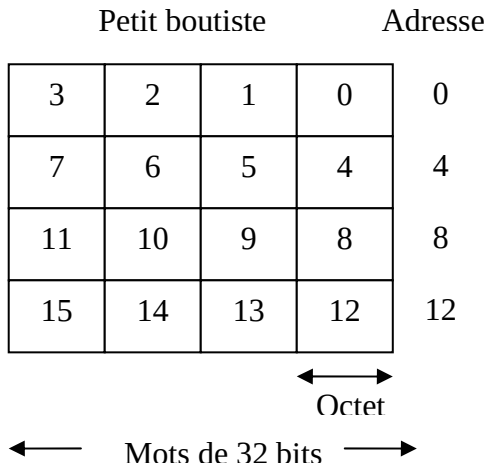

Fig.10. Mémoire «petit boutiste»

Historiquement un système dans lequel la numérotation commence par les « gros » octets (les octets de point fort), est Appelé ordinateur gros boutiste (big endian); par opposition, celui pour lequel le choix inverse a été fait est appelé petit boutiste (little endian). La représentation des nombres entiers dans les ordinateurs ne pose aucun problème dans les deux cas aussi que la représentation des informations constitués de chaîne de caractères et d'entières. Le problème se pose lorsque deux machines déférentes s'envoient des données à travers un réseau.