

Chapitre : L'E MICROPROCESSEUR

OBJECTIFS :

- ✓ Connaître le rôle de microprocesseur
- ✓ Connaître les différents composants d'un microprocesseur et leurs rôles.
- ✓ Comprendre le fonctionnement du microprocesseur et l'interaction entre les différentes unités.
- ✓ Maîtriser les étapes nécessaires à l'exécution des programmes par le microprocesseur.

1. Définition

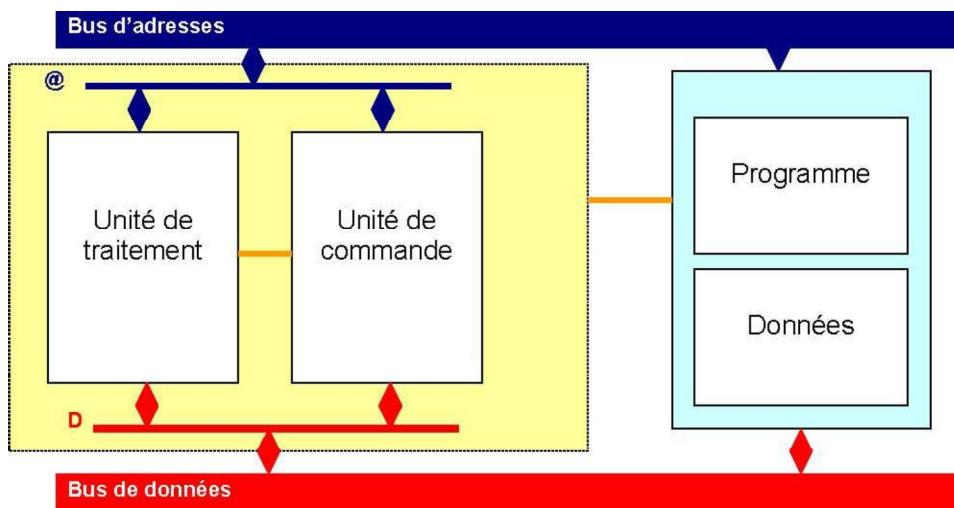
Un microprocesseur est un circuit intégré complexe caractérisé par une très grande intégration et doté des facultés d'interprétation et d'exécution des instructions d'un programme. Il est chargé d'organiser les tâches précisées par le programme et d'assurer leur exécution. Il doit aussi prendre en compte les informations extérieures au système et assurer leur traitement. C'est le cerveau du système. 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 (1).

2. Architecture de base d'un microprocesseur

Un microprocesseur est construit autour de deux éléments principaux :

- Une unité de commande : appelé aussi Unité de commande et de contrôle (UCC)
- Une unité de traitement

associés à des registres chargés de stocker les différentes informations à traiter. Ces trois éléments sont reliés entre eux par des bus interne permettant les échanges d'informations.



3. L'unité de commande

Elle permet de séquencer le déroulement des instructions. 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 puis effectue la préparation de l'instruction suivante. Pour cela, elle est composée par :

- **le compteur de programme** : (en anglais Program Counter PC) appelé aussi compteur ordinal (CO). Le CO est constitué par un registre dont le contenu représente l'adresse de la prochaine instruction à exécuter. Il est donc initialisé avec l'adresse de la première instruction du programme. Puis il sera incrémenté automatiquement pour pointer vers la prochaine instruction à exécuter.
- **le registre d'instruction** : Contient l'instruction en cours de traitement.
- **le décodeur d'instruction** :
- **Le séquenceur** : 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 par exemple. Il s'agit d'un automate réalisé soit de façon câblée (obsolète), soit de façon micro-programmée, on parle alors de micro-microprocesseur.

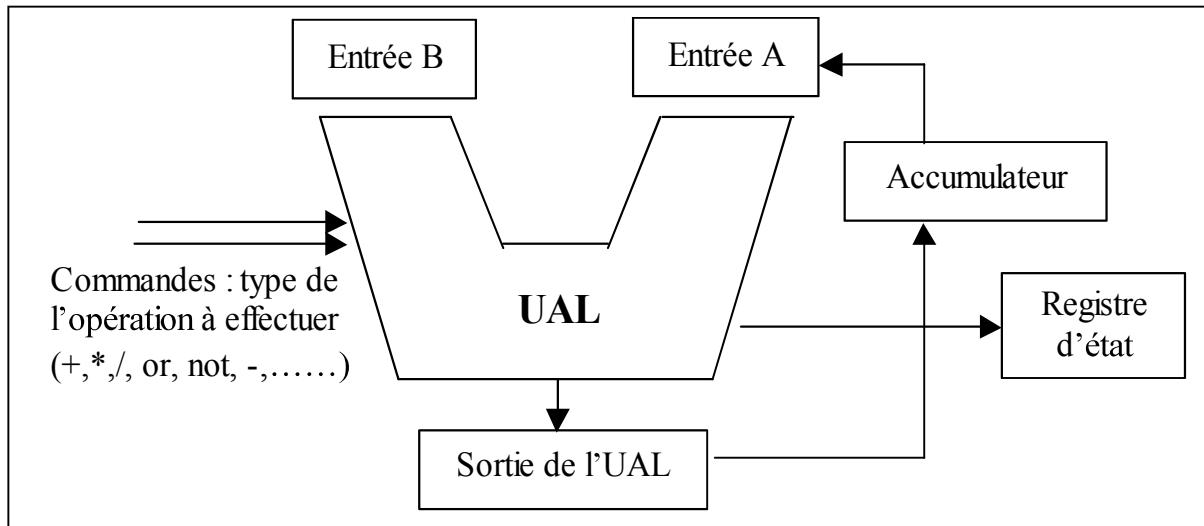
4. L'unité de traitement

C'est le cœur du microprocesseur. Elle regroupe les circuits qui assurent les traitements nécessaires à l'exécution des instructions. L'unité de traitement est Composée de trois principaux unités d'exécution, la première est l'unité arithmétique et logique (UAL) puis deux autres ont été ajoutés qui sont l'unité de calcul en virgule flottante et l'unité multimédia pour des raisons d'optimisation des performances des microprocesseurs.

4.1. Unité arithmétique et logique : (UAL)

Elle est composée de circuits logiques tels que les additionneurs, soustracteurs, comparateurs logiques...etc., afin d'effectuer les calculs et les opérations logiques des différents instructions à exécuter, les données à traiter se présentent aux entrées de l'UAL, sont traités, puis le résultat est fourni en sortie et généralement stocké dans un registre dit accumulateur. Les informations qui concernent l'opération sont envoyées vers le registre d'état.

Le schéma suivant montre l'UAL ainsi que ses entrées et ses sorties.



4.2. Unité de calcul en virgule flottante :

C'est une unité qui est capable de réaliser les opérations de calcul pour les réels ainsi que les calculs mathématiques et scientifiques complexes.

A l'origine la tâche de cette unité était réalisée par tout un processeur à part, en 1989 elle a été intégré dans les microprocesseurs afin d'optimiser les calculs.

4.3. Unité multimédia :

C'est une unité qui est chargée d'accélérer l'exécution des programmes multimédia comportant des vidéos, du son, graphisme en 3D etc....

Cette unité porte le nom de MMX pour les premiers pentium (MultiMedia eXtensions) intégrants des fonctions de gestion du multimédia, de même la technologie 3DNOW pour les AMD et SSE pour les pentiumIII.

Ces unités ont été créées vu la grande tendance vers la multimédia dans tous les types des programmes informatiques (jeux, logiciels sur Internet, encyclopédies...).

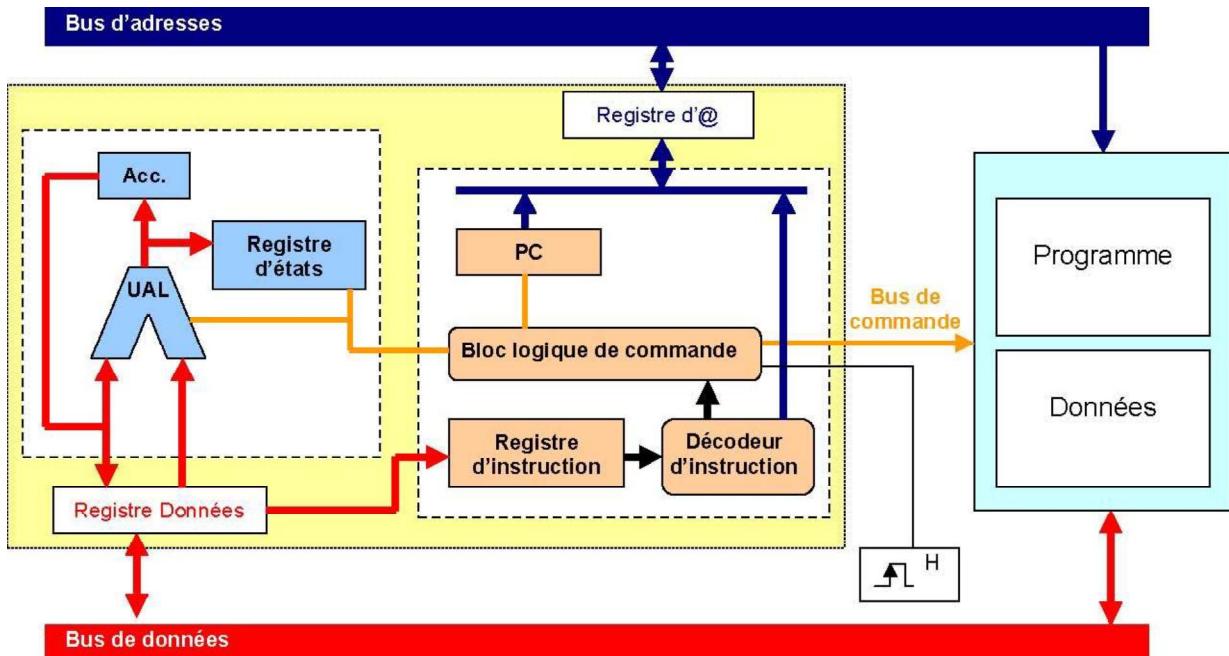


Figure 1.Schéma fonctionnel

5. Autres unités du microprocesseur

5.1. Unité de mémoire cache :

La mémoire cache a pour fonction principale d'optimiser les accès aux différentes instructions et données que le microprocesseur a besoin lors de l'exécution des programmes. La tâche de cette unité est de mettre dans la mémoire cache qui est beaucoup plus rapide que la mémoire centrale, les informations les plus utilisées et que le microprocesseur a besoin fréquemment. L'accès aux informations sera donc plus rapide et l'exécution des programmes est plus optimale.

Le principe de stockage des informations dans cette mémoire se fait par un contrôleur qui utilise des algorithmes spécifiques qui permettent de choisir quelles sont les données et les instructions dans la mémoire centrale à mettre dans cette mémoire et quand il faut les remplacer puisque la taille de la mémoire cache est très limitée par rapport à la taille de la mémoire centrale.

Les performances du microprocesseur sont très liés à la validité des prédictions faites par le contrôleur de mémoire cache. Il existe deux niveaux de mémoire cache. La cache de niveau 1 est toujours intégrée au microprocesseur, sa taille est de quelques kilooctets et peut atteindre 256 KO dans certains microprocesseurs. La cache de second niveau est généralement sur la carte mère, sa taille varie de 256 KO jusqu'à 1 MO.

5.2. Unité d'interface de bus :

Gère les échanges à travers le bus entre microprocesseur et les autres composantes, si le microprocesseur a besoin par exemple d'une donnée en mémoire vive, l'unité d'interface de bus se charge de la ramener en contrôlant l'accès mémoire et en mettant le résultat sur le bus de données qui va l'acheminer vers le registre de données.

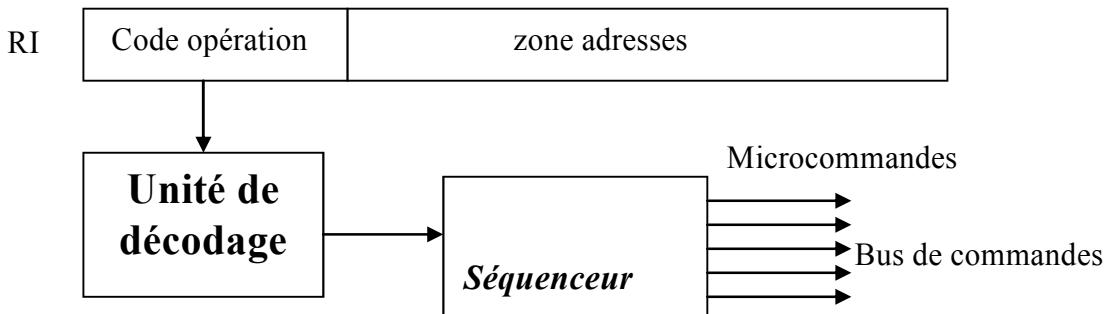
5.3. Unités de segmentation et de pagination :

Ces unités sont devenues nécessaires surtout pour les microprocesseurs de la famille INTEL X86 en raison de leurs façons particulières de gestion de la mémoire.

Ces unités permettent de traduire les adresses logiques manipulées dans les programmes en adresses physiques qui correspondent à des adresses réelles en mémoire. Ces unités varient d'un microprocesseur à l'autre selon la technologie de gestion de la mémoire utilisée.

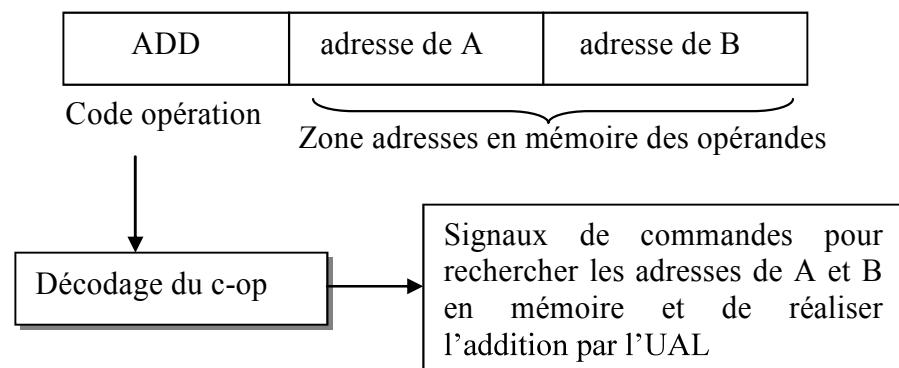
5.4. Unité de décodage :

Elle décompose et analyse l'instruction se trouvant dans le registre d'instructions, selon le code opération, elle envoie la nature des opérations à effectuer à l'unité de commande et précisément le séquenceur qui va ensuite générer les **microcommandes nécessaires aux différents composants** participant à l'exécution de l'instruction en cours.



Exemple :

Pour un microprocesseur ayant un format d'instructions à deux adresses, l'instruction d'addition de deux variables en mémoire A et B est représenté dans le registre d'instructions comme suit :



5.5. Unité d'anticipation et la queue (ou file d'attente) :

L'unité d'anticipation a pour rôle d'aller chercher en mémoire les instructions à exécuter. Ces instructions sont d'abord recherchées dans la mémoire cache interne du processeur. Si elles ne s'y trouvent pas, l'unité d'anticipation s'adresse à l'unité d'interface du bus afin qu'elles soient lues dans la mémoire centrale.

Lors de cette opération, le contenu des adresses suivantes de la mémoire est lu également et placé dans la mémoire cache du processeur. De cette façon, les prochaines instructions recherchées seront disponibles plus rapidement (à condition que le contenu ne soit pas modifié d'ici là).

L'unité d'anticipation place l'instruction dans une queue et s'occupe d'aller chercher la suivante. Grâce à ce dispositif, l'unité d'exécution n'a pratiquement jamais d'attendre que les instructions à exécuter lui soient amenées (cela peut cependant se produire si une série d'instructions très rapides à exécuter se présente). A l'inverse, si les instructions demandent un temps d'exécution important, la queue se remplit. Dans ce cas l'unité d'anticipation cesse de travailler jusqu'à ce que de l'espace se libère pour de nouvelles instructions.

6. Les registres du microprocesseur

Un registre est une zone mémoire à l'intérieur du microprocesseur de faible taille, qui permet de mémoriser des mots mémoires ou des adresses d'une façon temporaire lors de l'exécution des instructions.

Le nombre et le type de registres que possède le CPU sont une partie déterminante de son architecture et ont une influence importante sur la programmation. La structure des registres du CPU varie considérablement d'un constructeur à l'autre. Cependant les fonctions de base réalisées par les différents registres sont essentiellement les mêmes. Nous allons décrire les registres les plus importants, leur fonction et la façon dont ils peuvent être modifiés par programme.

Il existe deux types de registres : les registres généraux, et les registres d'adresses :

➤ *Les registres généraux :*

Ce sont des mémoires rapides, à l'intérieur du microprocesseur, qui permettent à l'unité d'exécution de manipuler des données à vitesse élevée. Ils sont connectés au bus données interne au microprocesseur. L'adresse d'un registre est associée à son nom (on donne généralement comme nom une lettre) A,B,C... Ces registres permettent de sauvegarder des informations utilisées dans les programmes ou des résultats intermédiaires, cela évite des accès à la mémoire, accélérant ainsi l'exécution des programmes.

Les registres généraux sont à la disposition du programmeur qui a normalement un choix d'instructions permettant de les manipuler comme

- Chargement d'un registre à partir de la mémoire ou d'un autre registre.
- Enregistrement dans la mémoire du contenu d'un registre.

- Transfert du contenu d'un registre dans l'accumulateur et vice versa.
- Incrémentation ou décrémentation d'un registre.
-

➤ ***Les registres d'adresses (pointeurs)***

Ce sont des registres connectés sur le bus adresses, leur contenu est une adresse en mémoire centrale.

Il existent plusieurs types On peut citer comme registres:

- Le compteur ordinal (pointeur de programme PC)
- Le pointeur de pile (Stack pointer SP)
- Les registres d'index (Index source SI et index destination DI)

1- Le compteur ordinal : (pointeur de programme PC)

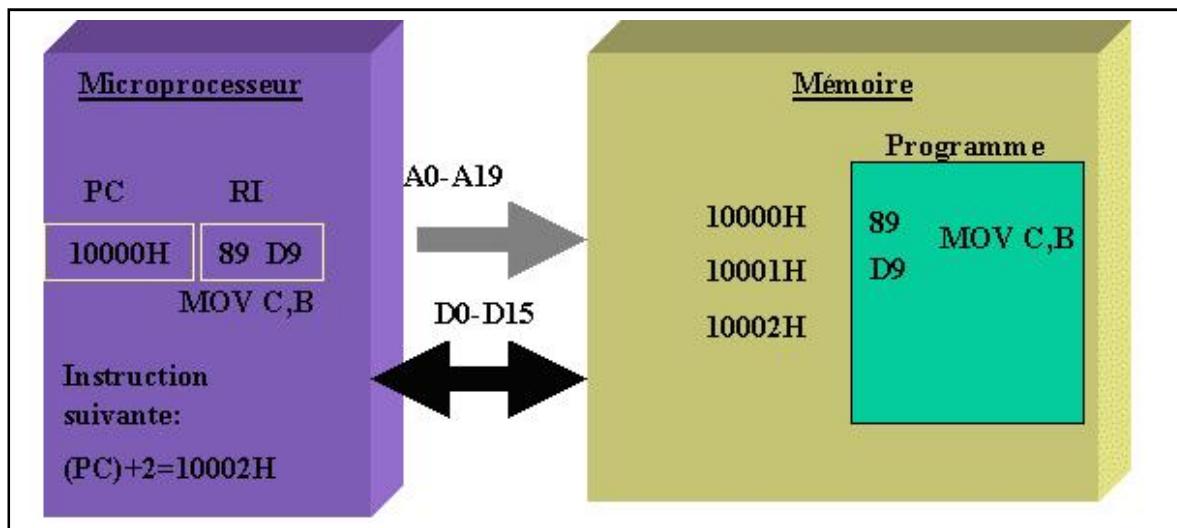


Figure 2. Schéma descriptif du fonctionnement du registre compteur ordinal

Il contient l'adresse de l'instruction à rechercher en mémoire. L'unité de commande incrémentera le compteur ordinal (PC) du nombre d'octets sur lequel l'instruction, en cours d'exécution, est codée. Le compteur ordinal contiendra alors l'adresse de l'instruction suivante.

Exemple : (PC)=10000H ; il pointe la mémoire qui contient l'instruction MOV C,B qui est codée sur deux octets (89 D9H) ; l'unité de commande incrémentera de deux le contenu du PC : (PC) = 10002H (la mémoire sera supposée être organisée en octets).

2- Le registre d'instruction :

contient l'instruction qui doit être traitée par le microprocesseur, cette instruction est recherchée en mémoire puis placée dans ce registre pour être décodée par le décodeur et préparée pour l'exécution.

Une instruction est une opération élémentaire d'un langage de programmation, c'est à dire le plus petit ordre que peut comprendre un ordinateur.

Exemple : ADD A,B cette instruction correspond à un ordre donné à l'ordinateur en langage assembleur qui permet de faire l'addition entre les données A et B.

Toute instruction présente en fait deux types d'informations :

- Ce qu'il faut faire comme action (addition, affichage, division ...)

- Avec quelles données réaliser cette action.

Zone opération	Zone adresses
----------------	---------------

Composition d'une instruction

- Zone opération : Cette zone permet de déterminer la nature de l'opération à effectuer par le microprocesseur.
- Zone adresse : contient les adresses en mémoire des opérandes qui participent dans une opération, dans certains cas elle contient l'opérande même. Il existe plusieurs modes d'adressages pour accéder aux données.

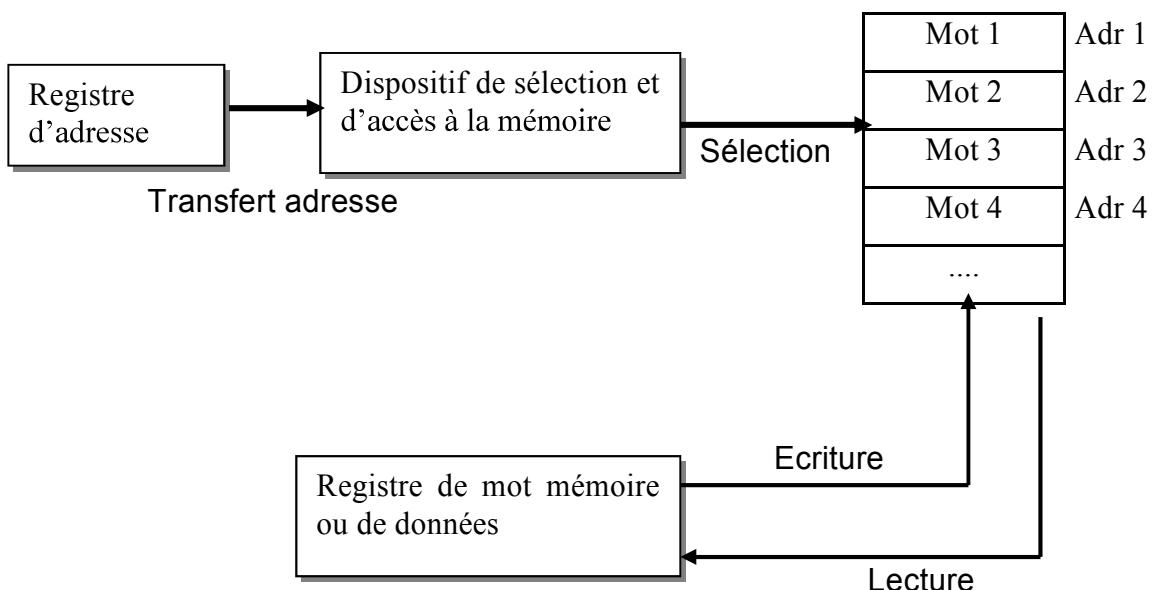
3- Le registre d'adresse :

C'est un registre qui contient l'adresse du mot à accéder en mémoire centrale. A chaque accès mémoire, l'adresse recherchée est stockée dans ce registre. Il a la taille d'une adresse qui est la même que celle du bus d'adresses ce qui permet de déterminer le nombre de mots mémoires adressables et l'espace mémoire adressable.

4- Le registre mot mémoire ou registre de données :

Contient le mot mémoire faisant objet d'une opération de lecture ou d'écriture dans la mémoire centrale. Ce registre a la taille d'un mot mémoire qui est la même que celle des registres de travail et l'accumulateur qui est égale à la taille du bus de données.

Le schéma ci dessous montre le fonctionnement des deux registres d'adresses et de mot mémoire.



La lecture et l'écriture dans la mémoire centrale se fait de la façon suivante :

- Lecture : l'adresse du mot à lire est transférée au registre d'adresse, le dispositif d'accès à la mémoire se charge de chercher le mot et de le mettre dans le registre mot mémoire.

- **Ecriture** : le registre d'adresse est chargé par l'adresse mémoire ou on va écrire, le mot à écrire est placé dans le registre mot mémoire. Lorsque l'ordre de l'écriture est donné, le contenu des cases mémoires sera écrasé et remplacé par la nouvelle valeur. Par contre, en cas de lecture, le contenu des cases mémoire n'est pas détruit.

5- Le registre accumulateur :

C'est un registre de travail très important de l'UAL, dans la plus part des opérations arithmétiques et logiques, l'accumulateur contient un des opérandes avant l'exécution et le résultat après. Il peut aussi servir de registre tampon dans les opérations d'entrée/sortie. Généralement, l'accumulateur a la même taille qu'un mot mémoire.

Naturellement, le programmeur a accès à ce registre qui est toujours très sollicité pendant le traitement des données. Certains processeurs ont plusieurs accumulateurs et dans ce cas, les codes opérations précisent l'accumulateur utilisé.

6- Le registre d'état: (**PSW program status word**)

C'est un registre qui contient les différents bits appelés drapeaux (Flags) indiquant l'état d'une condition particulière dans le CPU. Ces bits peuvent être testés par un programme et donc décider des suites d'actions à prendre.

Pour exécuter correctement son travail, le séquenceur doit en outre connaître l'état d'un certain nombre d'autres composants et disposer d'informations concernant la ou les opérations qui ont déjà été exécutées (par exemple, doit-on tenir compte dans l'addition en cours d'une éventuelle retenue préalable générée par une addition précédente). Le registre d'état fournit ces informations.

Le nombre de bits de ce registre change d'un microprocesseur à un autre, par exemple pour le cas d'un 8088 ce registre est de 16 bits :

U	U	U	U	O	D	I	T	S	Z	U	AC	U	P	U	C
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

U : undefined désigne un bit non défini.

Les différents bits du registre d'état sont :

- ***Bit de retenue*** : c'est le bit C ou carry qui est le bit n°0. ce bit est à 1 s'il y a une retenue qui est générée lors d'une opération arithmétique.
- ***Bit de parité*** : le bit P n°2, ce bit est mis à 1 si le nombre de 1 dans l'accumulateur est pair.
- ***Bit de retenue intermédiaire*** : le bit AC (auxiliary carry), c'est le bit n°4, il est mis à 1 si une retenue est générée entre groupes de 4 bits (passage de retenue).
- ***Le bit zéro*** : c'est le bit n°6, il est à 1 si le résultat d'une opération arithmétique est nul.
- ***Le bit de signe*** : bit n°7, il est à 1 si le résultat d'une opération arithmétique est négatif.

- **Le bit de dépassement de capacité** : le bit n°11, O pour overflow, il est à 1 s'il y a dépassement de capacité dans les opérations arithmétiques. Par exemple sur 8 bits en complément à 2, les nombres binaires sont codés sur 8 bits. On peut coder les nombres variants de -128 (10000000) à +127(01111111). Si on fait 107+28 cela donne 135 qui ne peut pas être représenté, d'où génération de débordement.

Exemple 1: addition de nombres binaire sur 8 bits

$$\begin{array}{r}
 11111100 \\
 + 10000010 \\
 \hline
 \text{carry : } 1 = 01111110
 \end{array}
 \quad
 \begin{array}{r}
 \text{FCH} + 82H = 17EH \\
 (252)_{10} + (130)_{10} = (382)_{10}
 \end{array}$$

Exemple 2 : La **bascule C (carry)** sert aussi à capter le bit expulsé lors d'une opération de **décalage ou de rotation**

Décalage à gauche d'un bit de cet octet : **10010110**
la carry recueille le **1** du bit de poids fort carry : **1 00101100**

Exemple 3 : overflow

$$\begin{array}{rrrr}
 104 & 0110\ 1000 & - 18 & 1110\ 1110 \\
 + 26 & + \underline{0001\ 1010} & - \underline{118} & \underline{1000\ 1010} \\
 \hline
 = 130 & = 1000\ 0010 \quad (-126) & - 136 & 0111\ 1000 \quad (120) \text{ avec C=1}
 \end{array}$$

L'**indicateur débordement** est une fonction logique (**OU exclusif**) de la retenue (**C**) et du signe (**S**).

7- Le registre pointeur de pile : (Stack Pointer)

- Il contient l'adresse de la pile. Celle-ci est une partie de la mémoire, elle permet de stocker des informations (le contenu des registres) relatives au traitement des interruptions et des sous-programmes
- La pile est gérée en **LIFO** : (Last IN First Out) dernier entré premier sorti. Le fonctionnement est identique à une pile d'assiette
- Le pointeur de pile SP pointe le haut de la pile (31000H dans le schéma), il est décrémenté avant chaque empilement, et incrémenté après chaque dépilement.
- Il existe deux instructions pour empiler et dépiler : PUSH et POP.
exemple:

PUSH A empilera le registre A et POP A le dépilera.

Sur la figure 1, on va montrer le fonctionnement de la pile lors d'instructions comme PUSH et POP.

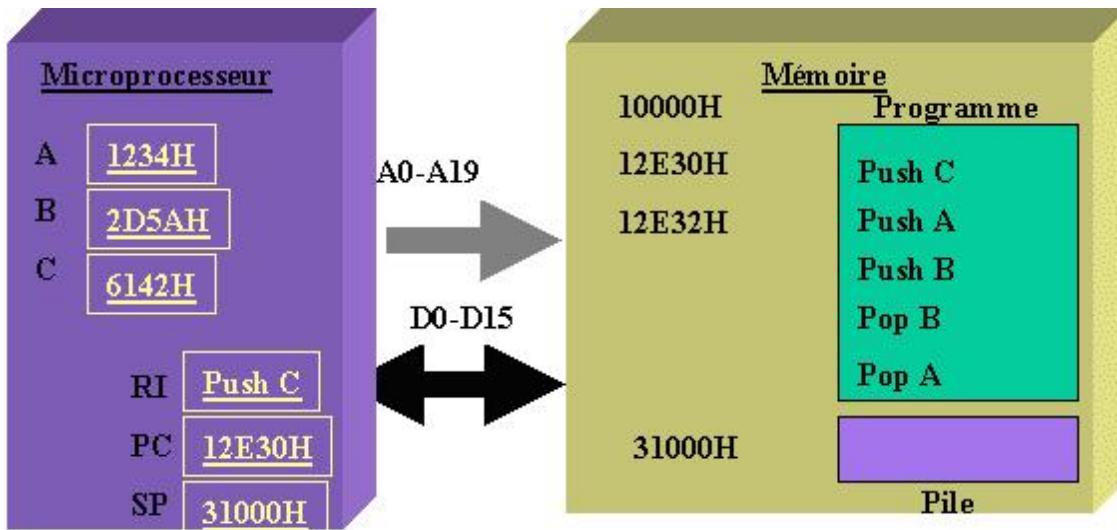
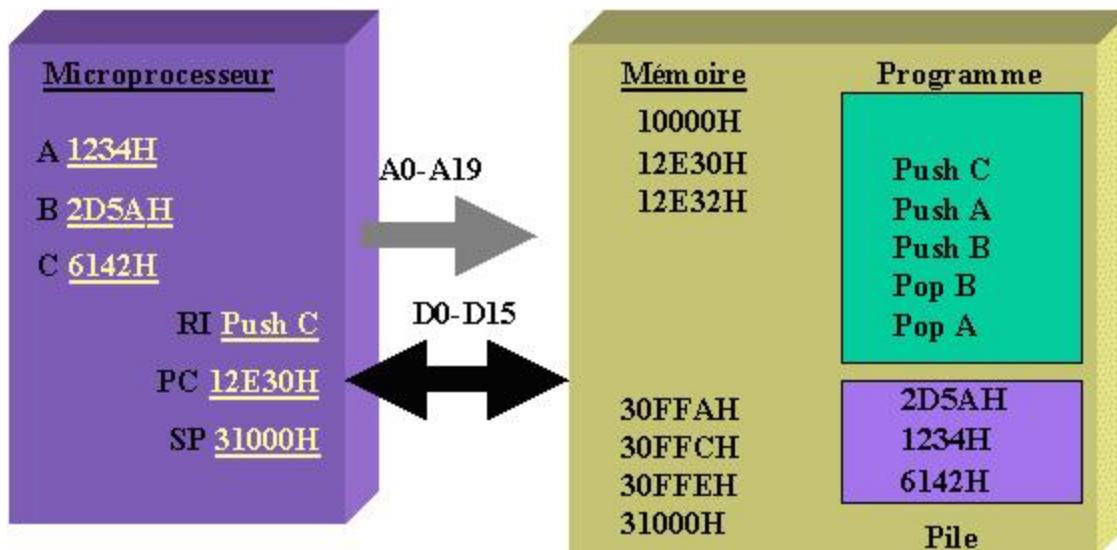


Figure 3. Schéma de Sauvegarde sur la pile

Question?

Que se passera-t-il durant l'exécution du programme commençant en 12E30H? Que vaudra SP et que contiendra la pile à cette adresse, à la fin du programme ?



Réponse.

Le programme commence par sauvegarder le contenu de C dans la pile (PUSH C). Pour cela (SP) est décrémenté de deux ($(SP)=31000H-2=30FFE$ H), puis on effectue l'écriture de (C) dans la mémoire à l'adresse (SP) : $(30FFE)=6142H$.

Pour PUSH A on obtient : $(30FFCH)=1234H$, et pour PUSH B : $(30FFAH)=2D5AH$.

Pour l'instruction POP B, ((SP)) est chargé dans le registre B ((SP)=30FFAH ; (B)=2D5AH) puis (SP) est incrémenté de deux ((SP)= 30FFAH+2=30FFCH).

Enfin, pour POP A on obtient : (A)=1234H et (SP)=30FFCH + 2 = 30FFE

8- Les registres d'index (index source SI et index destination DI)

Les registres d'index peuvent être utilisés comme des registres généraux pour sauvegarder et pour compter. Mais en plus ils ont une fonction spéciale qui est de grande utilité dans la manipulation des tableaux de données. Ils peuvent en effet être utilisés pour manipuler des adresses et permettent de mémoriser une adresse, suivant une forme particulière d'adressage, appelée adressage indexée.

Exemple :

MOV A,[SI+10000H] place le contenu de la mémoire d'adresse 10000H+le contenu de SI, dans le registre A.

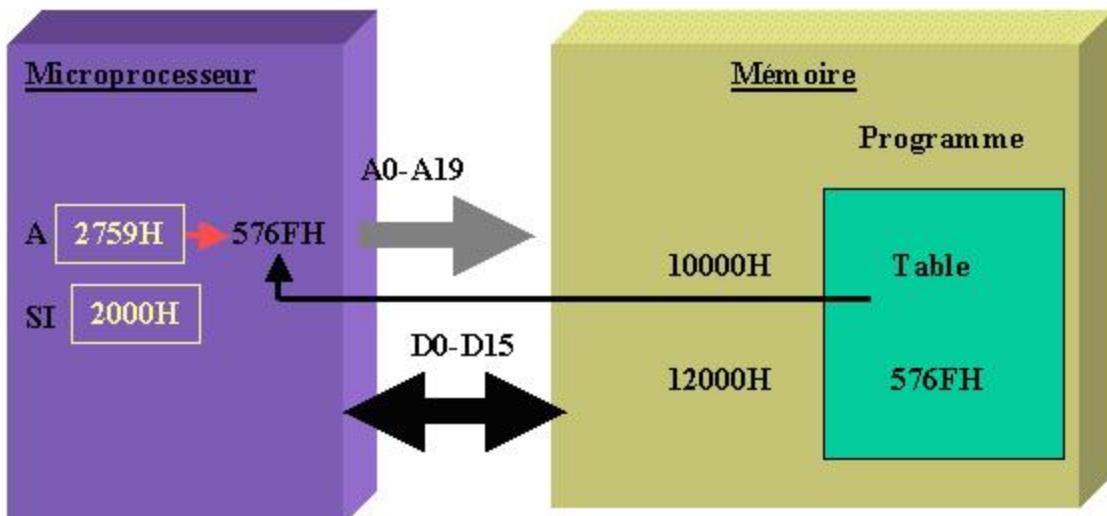


Figure 4. Exemple sur le registre d'index

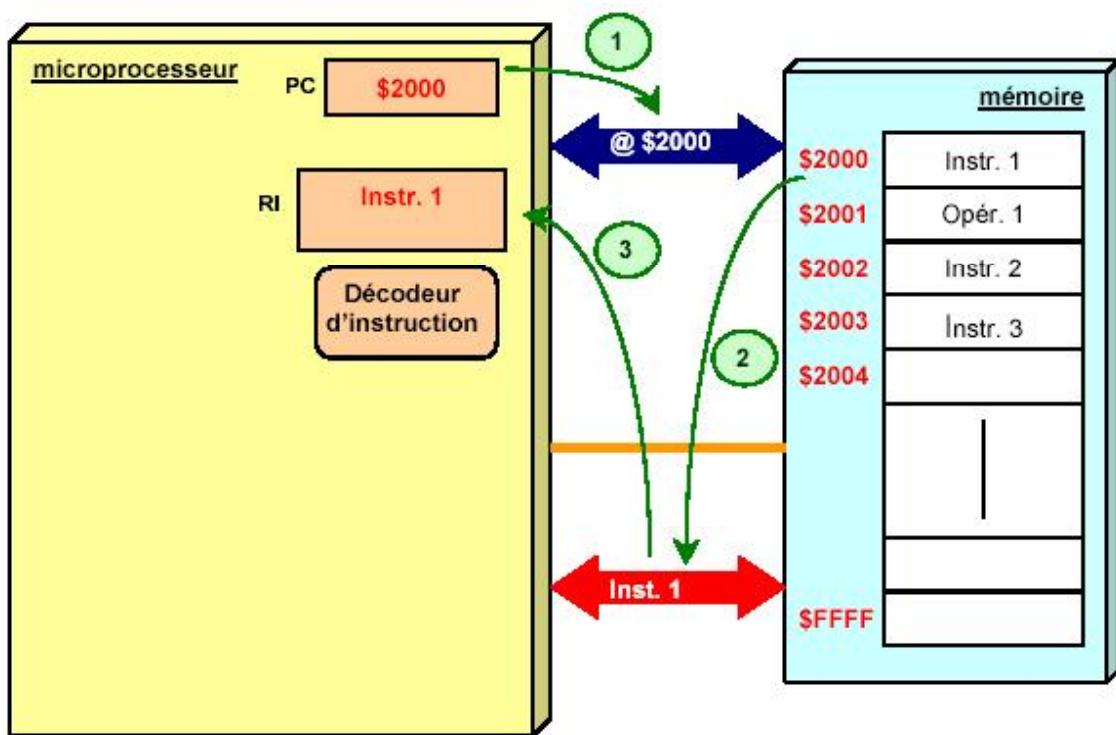
7. ÉTAPES D'EXÉCUTION D'UNE INSTRUCTION

(Cycle d'exécution d'une instruction)

Le microprocesseur ne comprend qu'un certain nombre d'instructions qui sont codées en binaire. Le traitement d'une instruction peut être décomposé en trois phases.

Phase 1: Recherche de l'instruction à traiter

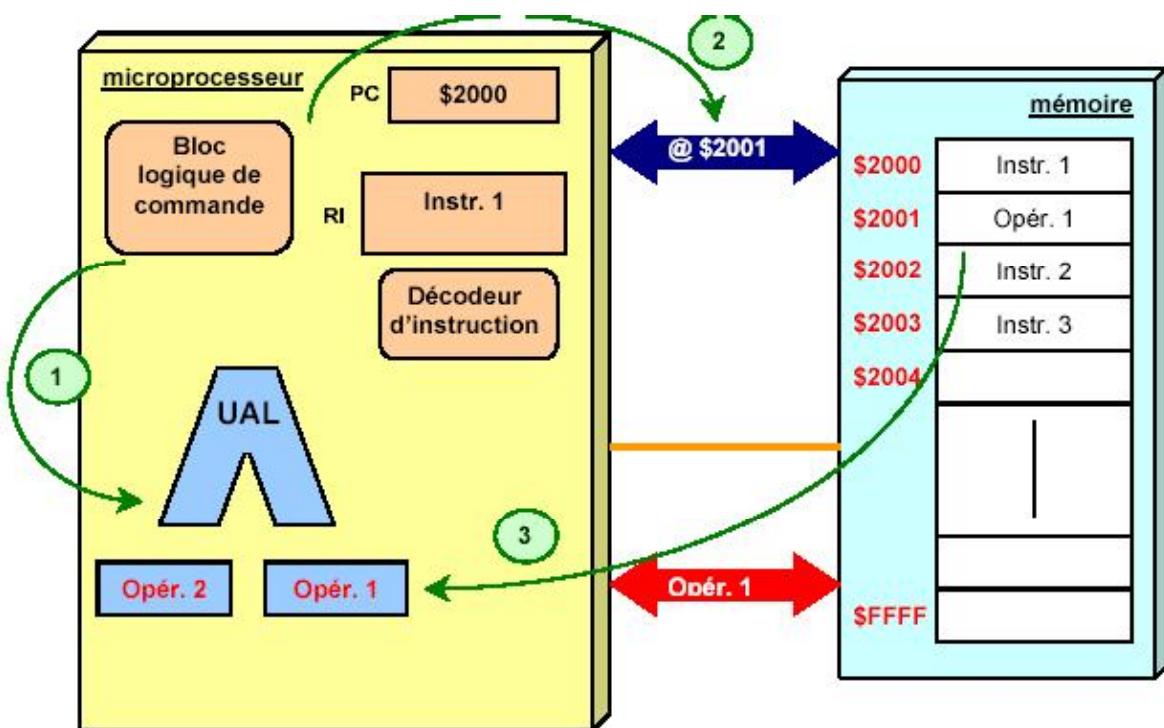
1. Le PC contient l'adresse de l'instruction suivante du programme. Cette valeur est placée sur le bus d'adresses par l'unité de commande qui émet un ordre de lecture.
2. Au bout d'un certain temps (temps d'accès à la mémoire), le contenu de la case mémoire Sélectionnée est disponible sur le bus des données.
3. L'instruction est stockée dans le registre instruction du processeur.



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

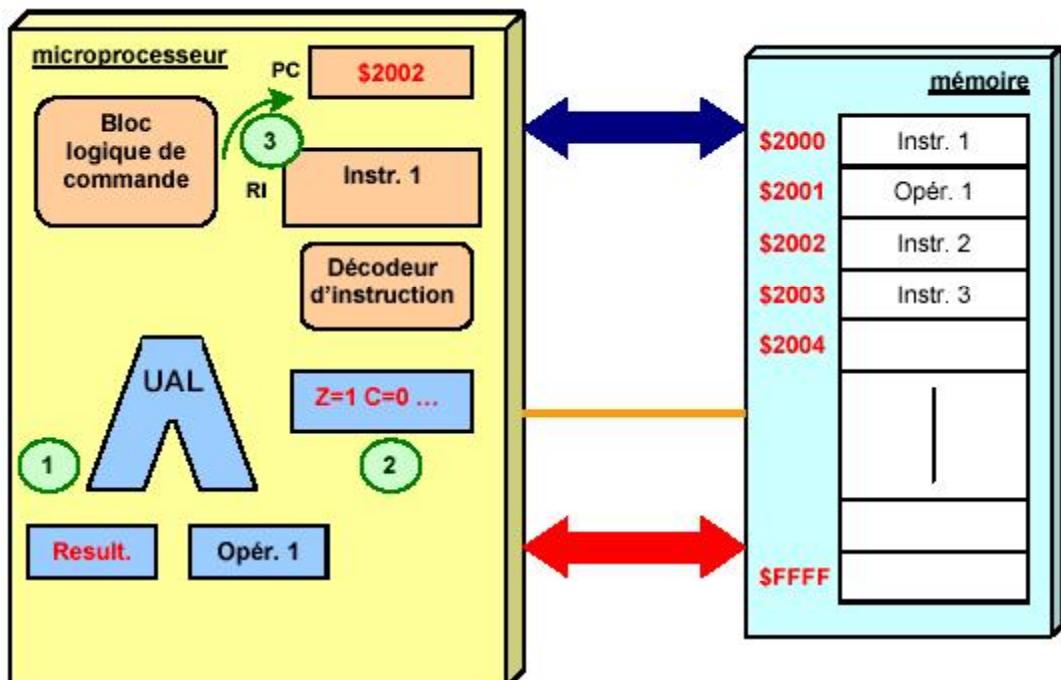
Le registre d'instruction contient maintenant le premier mot de l'instruction qui peut être codée sur plusieurs mots. Ce premier mot contient le code opératoire qui définit la nature de l'opération à effectuer (addition, rotation,...) et le nombre de mots de l'instruction.

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 stocké dans un registre.



Phase 3 : Exécution de l'instruction

1. Le micro-programme 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.



8. Jeu d'instructions

8.1. Définition

La première étape de la conception d'un microprocesseur est la définition de son jeu d'instructions. Le jeu d'instructions décrit l'ensemble des opérations élémentaires que le microprocesseur pourra exécuter. Il va donc en partie déterminer l'architecture du microprocesseur à réaliser et notamment celle du séquenceur. A un même jeu d'instructions peut correspondre un grand nombre d'implémentations différentes du microprocesseur.

8.2. Type d'instructions

Les instructions que l'on retrouve dans chaque microprocesseur peuvent être classées en 4 groupes :

- . Transfert de données pour charger ou sauver en mémoire, effectuer des transferts de registre à registre, etc...
- . Opérations arithmétiques : addition, soustraction, division, multiplication
- . Opérations logiques : ET, OU, NON, NAND, comparaison, test, etc...
- . Contrôle de séquence : branchement, test, etc...

8.3. Codage

Les instructions et leurs opérandes (paramètres) sont stockés en mémoire principale. La taille totale d'une instruction (nombre de bits nécessaires pour la représenter en mémoire) dépend du type d'instruction et aussi du type d'opérande. Chaque instruction est toujours codée sur un nombre entier d'octets afin de faciliter son décodage par le processeur. Une instruction est composée de deux champs :

- . le code instruction, qui indique au processeur quelle instruction réaliser
- . le champ opérande qui contient la donnée, ou la référence à une donnée en mémoire (son adresse).

Exemple :

Code instruction	Code opérande
1001 0011	0011 1110

Le nombre d'instructions du jeu d'instructions est directement lié au format du code instruction. Ainsi un octet permet de distinguer au maximum 256 instructions différentes.

8.4. Mode d'adressage

Un mode d'adressage définit la manière dont le microprocesseur va accéder à l'opérande. Les différents modes d'adressage dépendent des microprocesseurs mais on retrouve en général :

- . l'adressage de registre où l'on traite les données contenues dans un registre
- . l'adressage immédiat où l'on définit immédiatement la valeur de la donnée
- . l'adressage direct où l'on traite une donnée en mémoire

Selon le mode d'adressage de la donnée, une instruction sera codée par 1 ou plusieurs octets.

8.5. Temps d'exécution

Chaque instruction nécessite un certain nombre de cycles d'horloges pour s'effectuer. Le nombre de cycles dépend de la complexité de l'instruction et aussi du mode d'adressage. Il est plus long d'accéder à la mémoire principale qu'à un registre du processeur. La durée d'un cycle dépend de la fréquence d'horloge du séquenceur.

8.6. Langage de programmation

Le langage machine est le langage compris par le microprocesseur. Ce langage **Langage haut niveau** est difficile à maîtriser puisque chaque instruction est codée par une séquence propre de bits. Afin de faciliter la tâche du **compilation** programmeur, on a créé différents langages plus ou moins évolués.

Langage assembleur
([sta](#), [ld](#), [cmp](#), [mov](#), [bra](#), etc...)

Le langage assembleur est le langage le plus « proche » du langage machine. Il est composé par des instructions en général **assemblage** assez rudimentaires que l'on appelle des

Langage machine

([0001 1101](#), [1111 0110](#), etc...)

mnémoniques. Ce sont essentiellement des opérations de transfert de données entre les registres et l'extérieur du microprocesseur (mémoire ou périphérique), ou des opérations arithmétiques ou logiques. Chaque instruction représente un code machine différent. Chaque microprocesseur peut posséder un assembleur différent.

La difficulté de mise en œuvre de ce type de langage, et leur forte dépendance avec la machine a nécessité la conception de langages de haut niveau, plus adaptés à l'homme, et aux applications qu'il cherchait à développer. Faisant abstraction de toute architecture de machine, ces langages permettent l'expression d'algorithmes sous une forme plus facile à apprendre, et à dominer (C, Pascal, Java, etc...). Chaque instruction en langage de haut niveau correspondra à une succession d'instructions en langage assembleur. Une fois développé, le programme en langage de haut niveau n'est donc pas compréhensible par le microprocesseur. Il faut le compiler pour le traduire en assembleur puis l'assembler pour le convertir en code machine compréhensible par le microprocesseur. Ces opérations sont réalisées à partir de logiciels spécialisés appelés compilateur et assembleur.

Exemple de programme :

	Code machine (68HC11)	Assembleur (68HC11)	Langage C
@00	C6 64	LDAB #100	A=0 ;
@01	B6 00	LDAA #0	for (i=1 ; i<101 ; i++) A=A+i ;
@03	1B	ret	ABA
@04	5A		DECB
@05	26 03	BNE	ret

9. Performances d'un microprocesseur

On peut caractériser la puissance d'un microprocesseur par le nombre d'instructions qu'il est capable de traiter par seconde. Pour cela, on définit :

. le CPI (Cycle Par Instruction) qui représente le nombre moyen de cycles d'horloge nécessaire pour l'exécution d'une instruction pour un microprocesseur donné.

. le MIPS (Millions d'Instructions Par Seconde) qui représente la puissance de traitement du microprocesseur.

$$\frac{F}{H} \text{ avec } F_H \text{ en MHz}$$

$$\text{MIPS}=$$

$$\frac{CPI}{}$$

Pour augmenter les performances d'un microprocesseur, on peut donc soit augmenter la fréquence d'horloge (limitation matérielle), soit diminuer le CPI (choix d'un jeu d'instruction adapté).

10. Notion d'architecture RISC et CISC

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)

10.1. L'architecture CISC

Pourquoi

Par le passé la conception de machines CISC était la seule envisageable. En effet, vue 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.

Comment

C'est donc une architecture avec un grand nombre d'instructions 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 (micro-code)

10.2. L'architecture RISC

Pourquoi

Des études statistiques menées au cours des années 70 ont clairement montré que les programmes générés par les compilateurs se contentaient le plus souvent d'affectations, d'additions et de multiplications par des constantes. Ainsi, 80% des traitements des langages de haut niveau faisaient appel à seulement 20% des instructions du microprocesseur. D'où l'idée de réduire le jeu d'instructions à celles le plus couramment utilisées et d'en améliorer la vitesse de traitement.

Comment

C'est donc une architecture dans laquelle les instructions sont en nombre réduit (chargement, branchement, appel sous-programme). Les architectures RISC peuvent donc être réalisées à partir de séquenceur câblé. Leur réalisation libère de la surface permettant d'augmenter le nombres de registres ou d'unités de traitement par exemple. Chacune de ces instructions s'exécutent 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'effectue 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.

10.3. Comparaison

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
instructions simples ne prenant qu'un seul cycle	instructions complexes prenant plusieurs cycles
instructions au format fixe	instructions au format variable
décodeur simple (câblé)	décodeur complexe (microcode)
beaucoup de registres	peu de registres
seules les instructions LOAD et STORE ont accès à la mémoire	toutes les instructions sont susceptibles d'accéder à la mémoire
peu de modes d'adressage	beaucoup de modes d'adressage
compilateur complexe	compilateur simple

11. Améliorations de l'architecture de base

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.

11.1. Architecture pipeline

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.

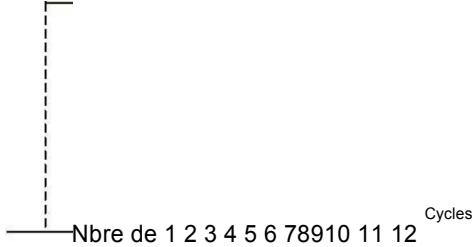
Exemple de l'exécution en 4 phases d'une instruction :



Modèle classique :

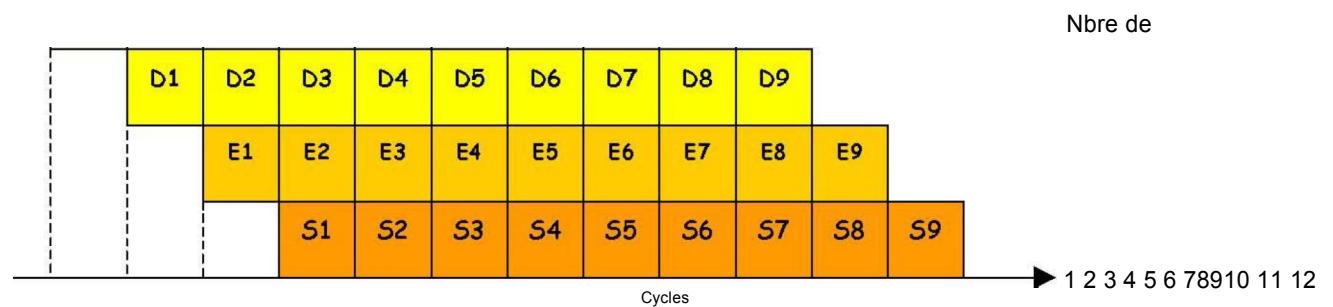
R1

R2 R3



Modèle pipeliné :

R1 R2 R3 R4 R5 R6 R7 R8 R9



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 :

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

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.

Remarques

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.

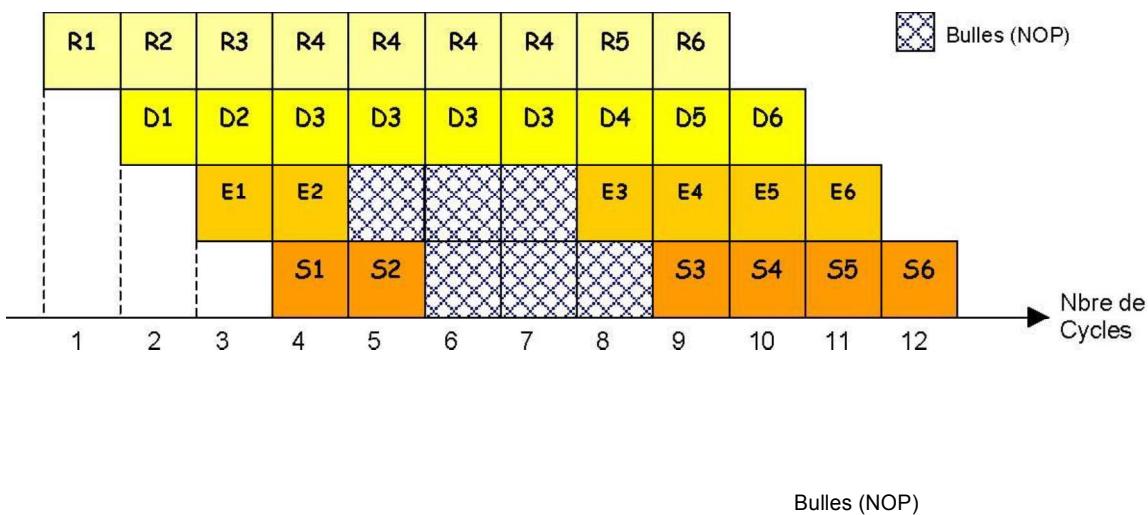
Problèmes

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 prédition de branchement qui a pour but de recenser lors de branchements le comportement le plus probable. Les mécanismes de prédition 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.

11.2. Notion de cache mémoire

Problème posé

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 données sont des étapes difficiles à accélérer. Ainsi, le temps de cycle processeur décroît plus vite que le temps d'accès mémoire entraînant un goulet 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.

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 défaut de cache. (b)

a)

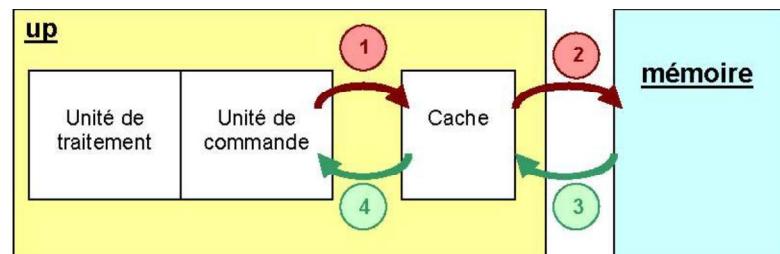
up

Unité de Unité de traitement commande Cache



mémoire

b)



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%.

Remarques :

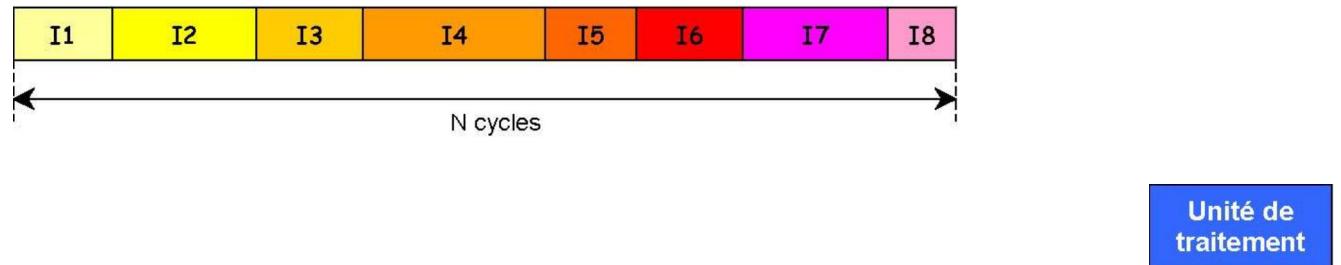
Un cache utilisera une carte pour savoir quels sont les mots de la mémoire principale dont il possède une copie. Cette carte devra avoir une structure simple.

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.

11.3. Architecture superscalaire

Une autre façon de gagner en performance est d'exécuter plusieurs instructions en même temps. L'approche superscalaire consiste à doter le microprocesseur de plusieurs unités de traitement travaillant en parallèle. 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.

Architecture scalaire :



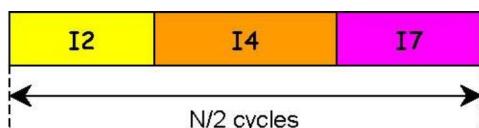
Architecture superscalaire :

I1



N/2 cycles

Remarque

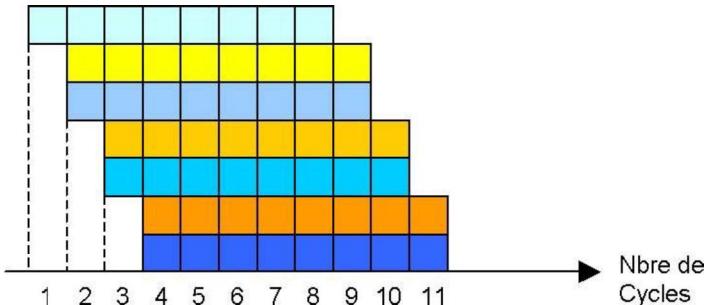


C'est le type d'architecture mise en oeuvre dans les premiers Pentium d'Intel apparus en 1993.

11.4. Architecture pipeline et superscalaire

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

UT1	Recherche	Décodage	Exécution	Sauv. résultat
UT2	Recherche	Décodage	Exécution	Sauv. résultat



12. Processeurs spéciaux

12.1. 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..). Ils comportent aussi des fonctions spécifiques comme des compteurs programmables pour effectuer des mesures de durées, des CAN voir des CNA pour s'insérer au sein de chaînes d'acquisition, des interfaces pour réseaux de terrain, etc ...

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

12.2. Le processeur de signal

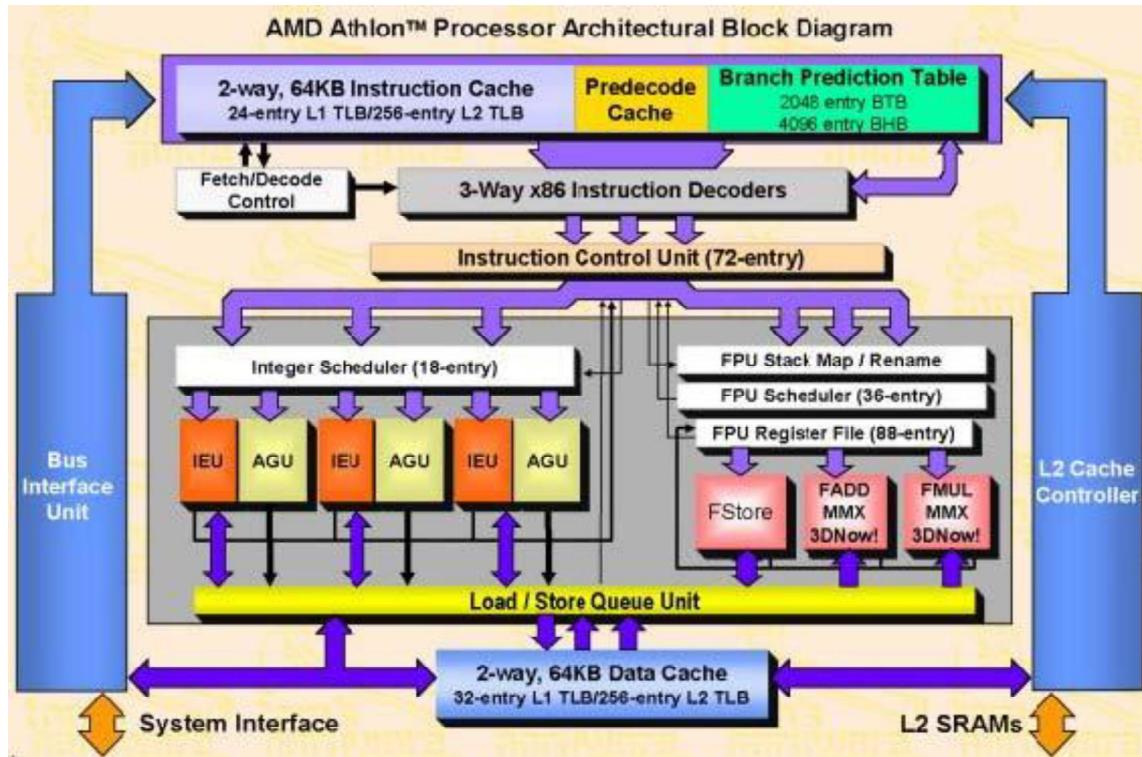
Le processeur de signal est beaucoup plus spécialisé. Alors qu'un microprocesseur n'est pas conçu pour une application spécifique, le processeur DSP (Digital Signal Processor) est optimisé pour effectuer du traitement numérique du signal (calcul de FFT, convolution, filtrage numérique, etc...).

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...

12.3. Exemples

Voici deux exemples d'architecture de deux processeurs qui tenaient le haut du pavé lors de leur sortie en 1999 : l'Athlon d'AMD et le Pentium III d'Intel. ($f \approx 500\text{MHz}$)

AMD Athlon :



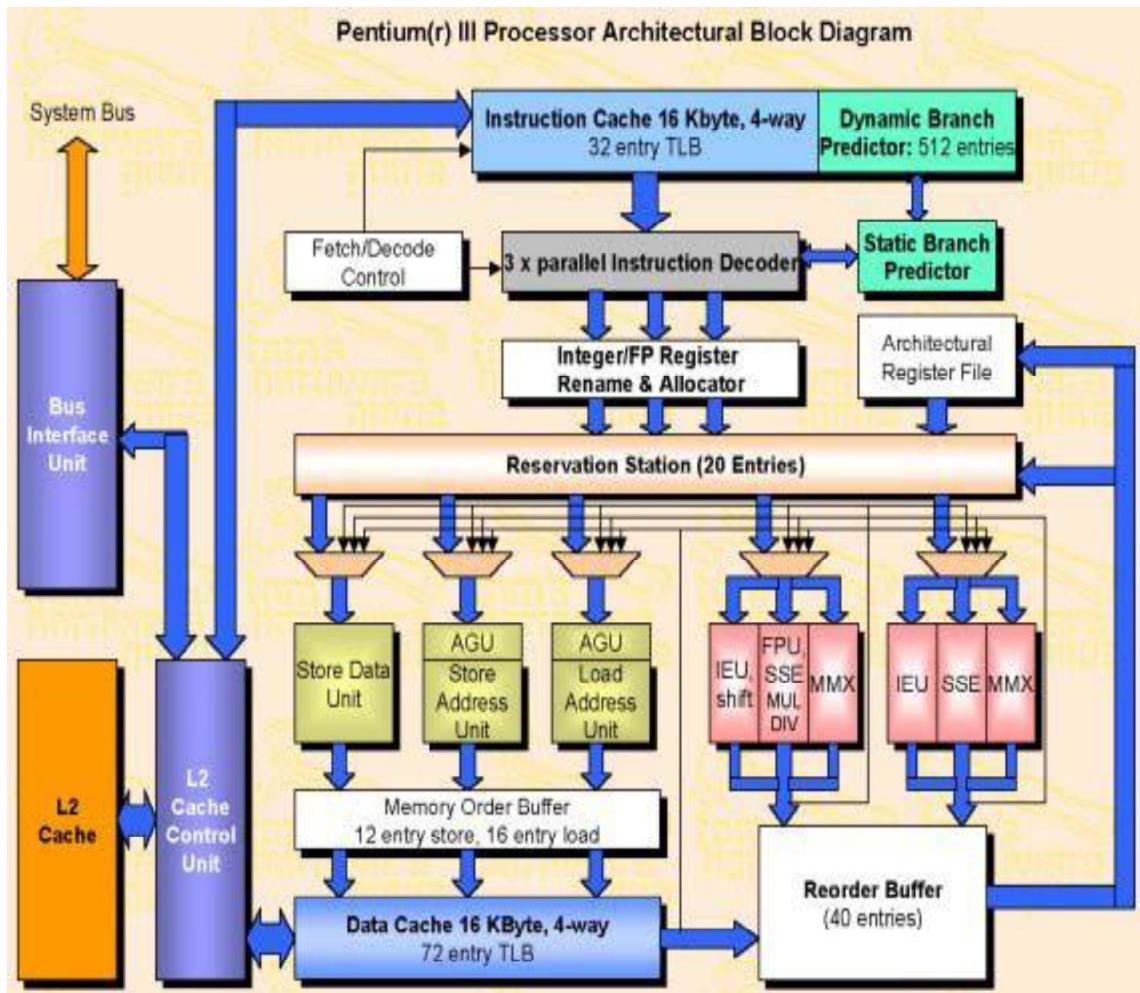
AGU : Adress Generation Unit BTB : Branch Target Buffer IEU : Integer Execution Unit BHB : Branch History Buffer

Caractéristiques:

- . 9 unité de traitement se composant de :
- o 1 ALU (traitement entier) comprenant 6 unités de traitement :
- . 3 unités pour le traitement des données (IEU)
- . 3 unités pour l'adressage des données (AGU)
- o 1 FPU (traitement réel) comprenant 3 unités :
- . 1 FPU store
- . 1 Fadd / MMX / 3Dnow !
- . 1 Fmul /MMX / 3Dnow !
- . Pipeline entier : 10 étages, pipeline flottant : 15 étages.
- . Prédiction dynamique et exécution du traitement "dans le désordre" (out-of-order)
- . 6 unités de décodage parallèles (3 micro-programmées, 3 câblées) mais seules 3 peuvent fonctionner en même temps.
- . Cache mémoire de niveau 1 (L1) : 128 Ko
- o 64 Ko pour les données
- o 64 Ko pour les instructions
- . Contrôleur de cache L2 supportant de 512Ko à 8Mo avec vitesse programmable (1/2 ou 1/3 de la vitesse CPU)



Intel Pentium III



Caractéristiques :

- Plusieurs unités de traitement mais au 5 instructions exécutées en même temps sur 5 ports :
- o Port 0 : ALU, FPU, AGU MMX et SSE
- o Port 1 : ALU, SSE, MMX
- o Port 2 : AGU (load)
- o Port 3 : AGU (store)
- o Port 4 : Store Data Unit
- Pipeline entier : 12 à 17 étages, pipeline flottant : environ 25 étages
- Prédiction dynamique et exécution du traitement "dans le désordre" (out-of-order)
- 3 unités de décodage parallèles : 1 micro-programmée, 2 câblées.
- 5 pipelines de 10 étages
- Cache mémoire de niveau 1 (L1) : 32 Ko
- o 16 Ko pour les données
- o 16 Ko pour les instructions
- Contrôleur de cache L2 supportant jusqu'à 512 Ko à ½ de la vitesse CPU
- 9.5 millions de transistors

Références

Cours malek zribi, ISET Sfax

Cours Dumartin (.pdf)

Livre : « Architectures des ordinateurs » M.C.BELAID »