Chapitre III - 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 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.

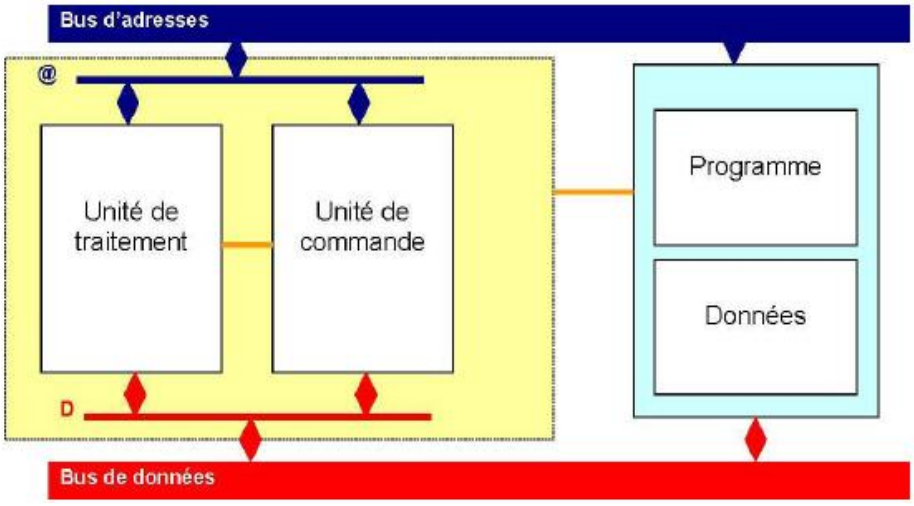


Figure 1. Schéma de l'Architecture de base

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 d'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** : Il traduit en commandes internes ce que l'instruction lui dit de faire.
- **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 principales 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.

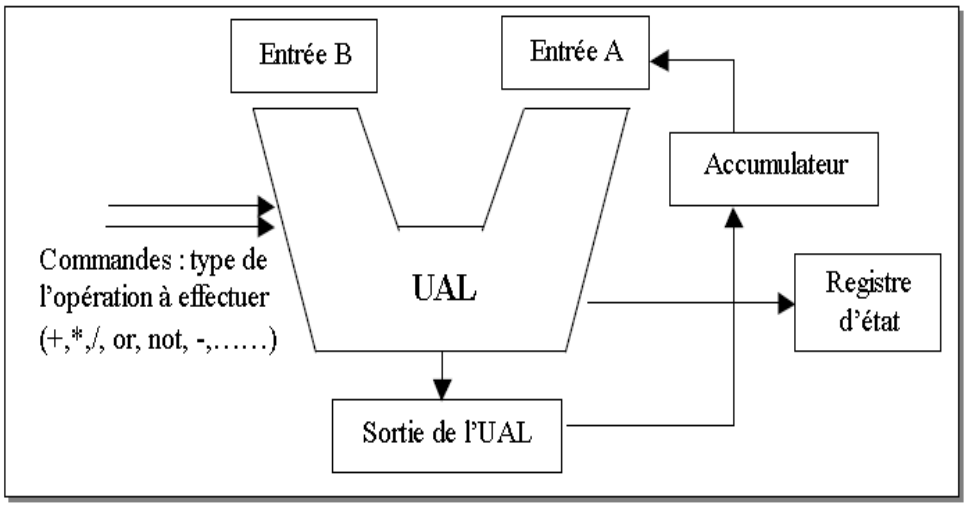


Figure 2. Schéma de l'unité arithmétique et logique

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 pentiums (MutiMedia eXtensions) intégrants des fonctions de gestion du multimédia.

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

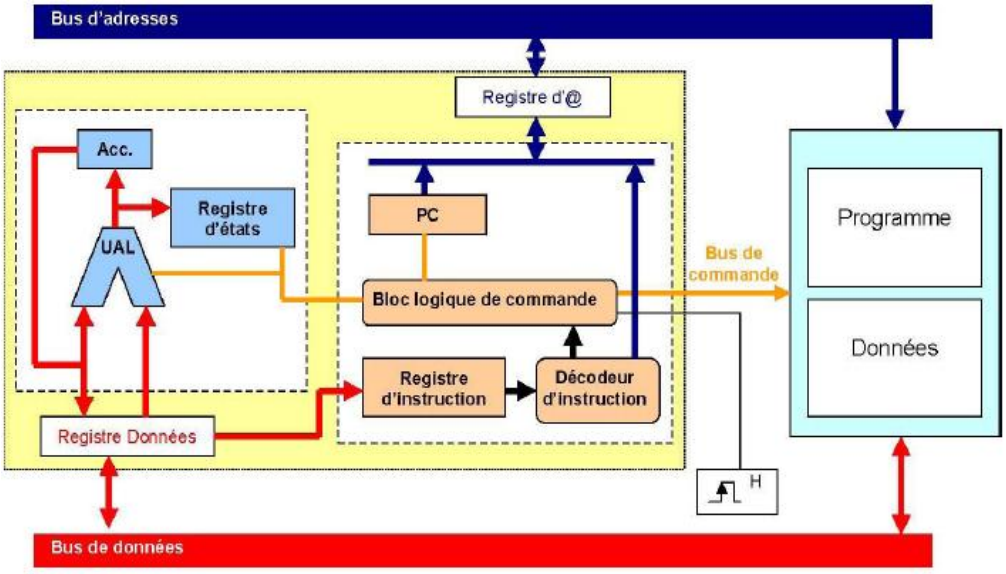


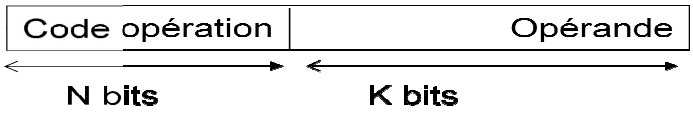
Figure 3. Schéma fonctionnel

5. Jeu d'instructions

- Chaque microprocesseur possède un certain nombre limité d'instructions qu'il peut exécuter. Ces instructions sont appelées jeu d'instructions.
- Le jeu d'instructions décrit l'ensemble des opérations élémentaires que le microprocesseur peut exécuter.
- Les instructions peuvent être classifiées en 4 catégories :
  - Instruction d'affectation : elle permet de faire le transfert des données entre les registres et la mémoire
    - Écriture : registre → mémoire
    - Lecture : mémoire → registre
  - Les instructions arithmétiques et logiques (ET, OU, ADD,….)
  - Instructions de branchement (conditionnel et inconditionnel)
  - Instructions d'entrées sorties.

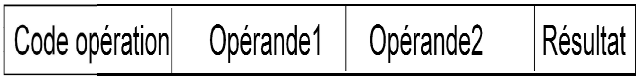
5.1 Codage d'une instruction

- Les instructions et leurs opérandes (données) sont stockés dans la mémoire.
- La taille d'une instruction (nombre de bits nécessaires pour la représenter en mémoire) dépend du type de l'instruction et du type de l'opérande.
- L'instruction est découpée en deux parties :
  - Code opération (code instruction) : un code sur N bits qui indique quelle instruction.
  - Le champ opérande : qui contient la donnée ou la référence (adresse) à la donnée.



Machine à 3 adresses

- Dans ce type de machine pour chaque instruction il faut préciser :
  - l'adresse du premier opérande
  - du deuxième opérande
  - et l'emplacement du résultat



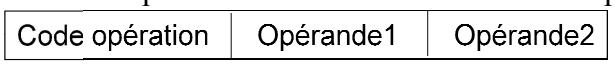
Exemple :

ADD A,B,C (C←B+A)

- Dans ce type de machine la taille de l'instruction est grande.
- Pratiquement il n'existe pas de machine de ce type.

Machine à 2 adresses

- Dans de type de machine pour chaque instruction il faut préciser :
  - l'adresse du premier opérande
  - du deuxième opérande,
- l'adresse de résultat est implicitement l'adresse du deuxième opérande.

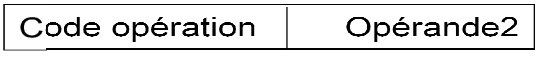


Exemple :

ADD A,B (B←A +B)

Machine à 1 adresse

- Dans de type de machine pour chaque instruction il faut préciser uniquement l'adresse du deuxième opérande.
- Le premier opérande existe dans le registre accumulateur.
- Le résultat est mis dans le registre accumulateur.



Exemple :

ADD A (ACC←(ACC) + A)

Ce type de machine est le plus utilisé.

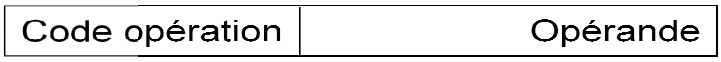
6. Mode d'adressage

- Le champ opérande contient la donnée ou la référence (adresse) à la donnée.
- Le mode d'adressage définit la manière dont le microprocesseur va accéder à l'opérande.
- Le code opération de l'instruction comporte un ensemble de bits pour indiquer le mode d'adressage.

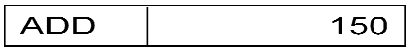
- Les modes d'adressage les plus utilités sont :
  - Immédiat
  - Direct
  - Indirect
  - Indexé
  - relatif

6.1 Adressage immédiat

- L'opérande existe dans le champ adresse de l'instruction
-



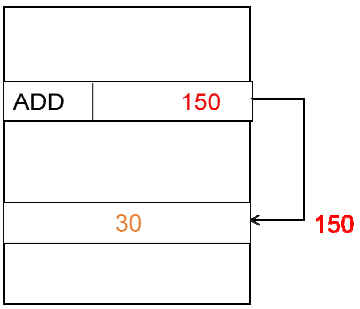
Exemple :
ADD 150



Cette commande va avoir l'effet suivant : ACC ← (ACC) + 150
Si le registre accumulateur contient la valeur 200 alors après l'exécution son contenu sera égal à 350

6.2 Adressage direct

- Le champ opérande de l'instruction contient l'adresse de l'opérande (emplacement en mémoire)
- Pour réaliser l'opération il faut le récupérer l'opérande à partir de la mémoire. ACC ← (ACC) + (ADR)
  Exemple : On suppose que l'accumulateur continent la valeur 20.



A la fin de l'exécution nous allons avoir la valeur 50 (20 + 30)

6.3 Adressage indirect

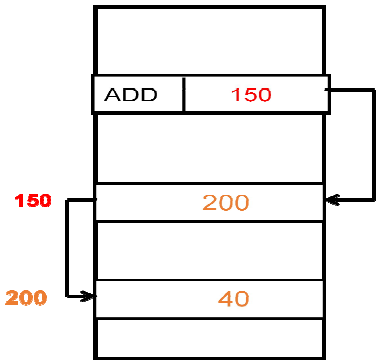
- Le champ adresse contient l'adresse de l'adresse de l'opérande.
- Pour réaliser l'opération il faut :
  - Récupérer l'adresse de l'opérande à partir de la mémoire.
  - Par à la suite il faut chercher l'opérande à partir de la mémoire.

ACC← (ACC) + ((ADR))

Exemple :

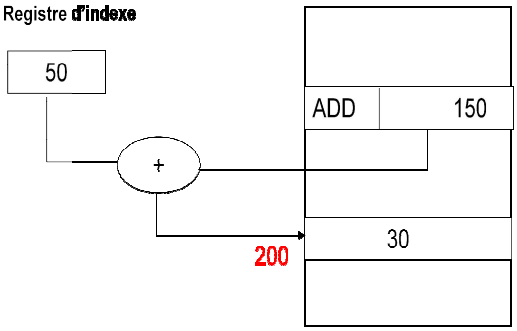
- Initialement l'accumulateur contient la valeur 20
- Il faut récupérer l'adresse de l'adresse (150).
- Récupérer l'adresse de l'opérande à partir de l'adresse 150 (la valeur 20)
- Récupérer la valeur de l'opérande à partir de l'adresse 200 (**la valeur 40**)

- Additionner la valeur 40 avec le contenu de l'accumulateur (20) et nous allons avoir la valeur **60**



6.4 Adressage indexé

- L'adresse effective de l'opérande est relative à une zone mémoire.
- L'adresse de cette zone se trouve dans un registre spécial (registre indexe).
- Adresse opérande = ADR + (X)

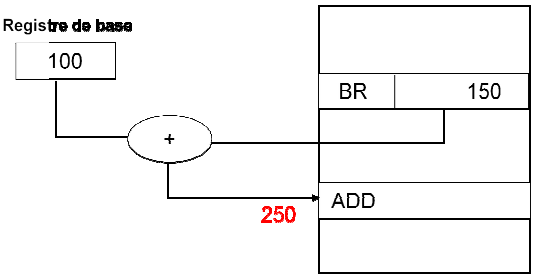


Remarque : si ADR ne contient pas une valeur immédiate alors l'**Adresse opérande** = (ADR)+ (X)

6.5 Adressage relatif

- L'adresse effective de l'opérande est relative à une zone mémoire.
- L'dresse de cette zone se trouve dans un registre spécial (registre de base).
- Ce mode d'adressage est utilisé pour les instructions de branchement.

Adresse = ADR + (base)



7. Autres unités du microprocesseur

7.1. 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 1MO.

7.2. 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.

7.3. 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.

7.4. 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.

7.5. 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 remplie. Dans ce cas l'unité d'anticipation cesse de travailler jusqu'à ce que de l'espace se libère pour de nouvelles instructions.

8. 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 de 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 existe 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)

Il contient l'adresse de l'instruction à rechercher en mémoire. L'unité de commande incrémente 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.

2- Le registre d'instruction :
Il 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.
NB : 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.

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.

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. 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 retenu 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 retenu*** : c'est le bit C ou carry qui est le bit n°0. Ce bit est à 1 s'il y a une retenu qui est générée lors d'une opération arithmétique.
- ***Bit de parité*** : le bit P n°2, ce bit est mis à 1 si le résultat d'une opération arithmétique est pair.
- ***Bit de retenu intermédiaire*** : le bit AC (auxiliary carry), c'est le bit n°4, il est mis à 1 si une retenu est générée entre groupes de 4 bits (passage de retenu).
- ***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.

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 pointeur de pile SP pointe le haut de la pile, il est décrémenté avant chaque empilement, et incrémenté après chaque dépilement.
- Il existe deux instructions pour empiler et dépiler : PUSH et POP.

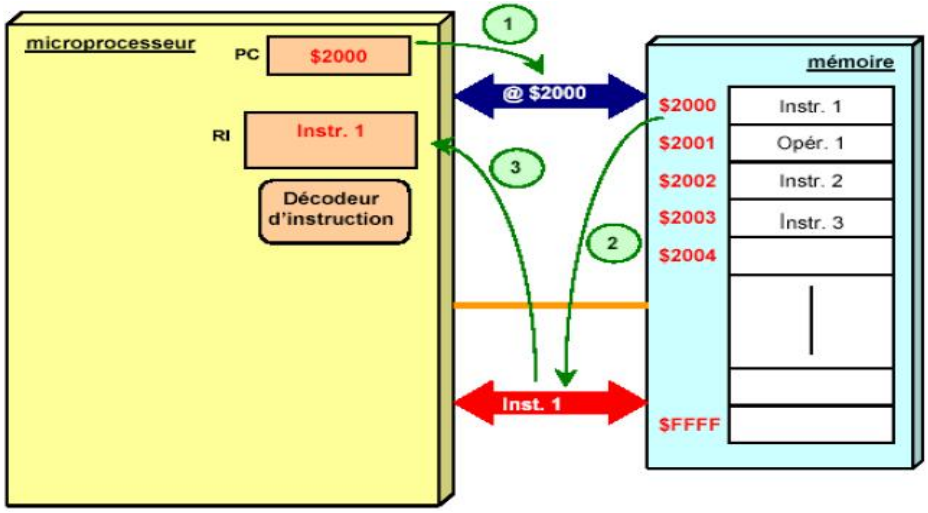
8- Les registres d'index (index source SI et index destination DI)
Les registres d'index peuvent être utilisés comme des registres généraux pour sauvegarder et pour compter. Mais en plus ils ont une fonction spéciale qui est de grande utilité dans la manipulation des tableaux de données. Ils peuvent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

9. 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.

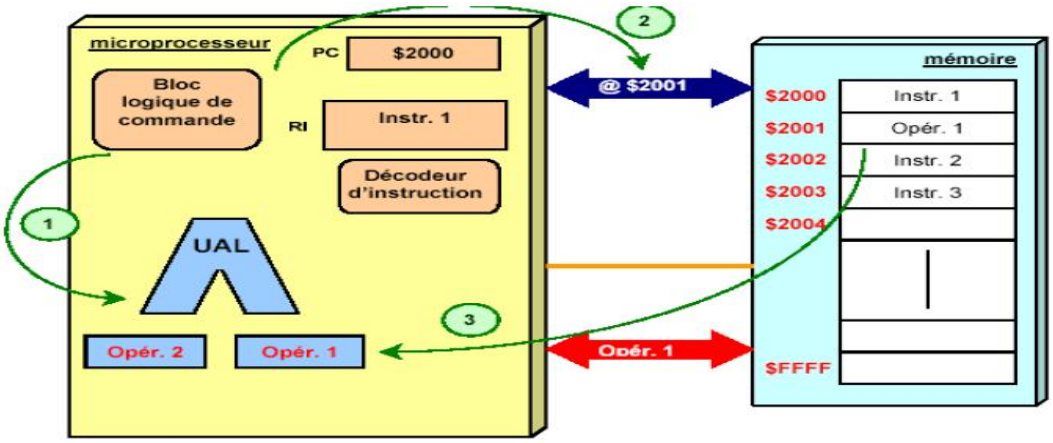
Phase 1: ***Recherche de l'instruction à traiter***
1. Le PC contient l'adresse de l'instruction suivante du programme. Cette valeur est placée sur le bus d'adresses par l'unité de commande qui émet un ordre de lecture.

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.

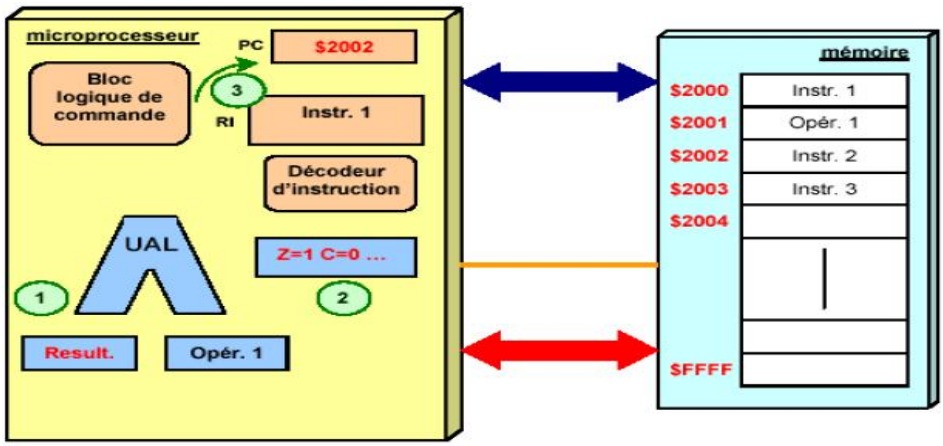


Phase 2 : ***Décodage de l'instruction et recherche de l'opérande***

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.



10. 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.

$$\text{MIPS} = \frac{\text{FH}}{\text{CPI}}$$

avec FH en MHz

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).

11. 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)

11.1. 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)

11.2. 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 nombre 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'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.

11.3. 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

12. 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.

12.1. *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 :

13. *Langage de programmation* **..**

Le langage machine est le langage compris par le microprocesseur. Ce langage Langage haut niveauest 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, lda, cmp, mov, bra, etc…) → Langage machine (0001 1101, 1111 0110, 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 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.