Suite du chapitre 2

**3. Cycle d'exécution d'une instruction:**

Le traitement de l'instruction peut etre décomposé on trois phase .

**Phase 1:** Reherche de l'instuction à traiter



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



**Phase 3:** Exécution de l'instruction



**4. Jeu d'intruction:**

**4.1 Définition:** La premiere étape de la conception d'un microprocesseur est la définition de son jeu d'instruction. Le jeu d'instruction décrit l'ensemble des opérations élémetaires que le microprocesseur pourra éxécuter.

**4.2 Type d'intruction:**

Les instructions peuvent etre classées en 4 groupes:

- transfert de donnée: pour charger ou sauver en mémoire, effectuer des transfert de registre à registre, etc...

- opérations arithmétiques: addition, soustraction, division, et mutiplication

- opérations logiques: ET, OU, NAND, comparaison, test, etc...

- controle de séquence: branchement, test , etc...

**4.3 Codage:**

Les instructions et leurs opérandes (paramétres) sont stockés en mémoire principale, chaque instruction est toujours codée sur un nombre entier d'octets afin de faciliter son décodage par le processeur.

Une instruction est composé de deux champs:

- le code d'instruction, qui indique an processeur quelle instruction réalise

- le champs opérande qui contient la donnée, ou la réference à une donnée en mémoire (son adresse)

**4.4 Mode d'adressage:**

Un mode d'adressage définit la maniére dont le microprocesseur va accéder à l'opérande

- L'adressage de registre ou l'on traite la donnée contenue dans un regitre

-L'adressage immédiat ou l'on défiit immédiatement la valeur de la donnée

- L'adressage direct ou l'on traite une donnée en mémoire

**4.5 Temps d'exécution:**

Chaque instruction nécessite un certain nombre de cycles d'horloge our s'effectuer. La durée d'un cycle dépend de la fréquence d'horloge du séquenceur.

**5. Langage de programmation:**

Le langage machine est le langage compris par le microporcesseur. ou chaque instruction est codée par une séquence propre de bits.

Le langage assembleur est le langage le plus proche du langage machine, il est composé par des instructions que l'on appelle des mnémoniques. Ce sont essentiellemet des opérations de transfert de données entre les registres et l'éxtérieur du microprocesseur (mémoire et périphérique), ou des opérations arithmétiques ou logiques.

La difficulté de mise en oeuvre de ce type de langage et leur forte dépendance avec la machine à nécessite la concepion de langage de haut niveau. Chaque instruction en langage haut niveau correspondra à une succession d'instruction en lngage assembleur, il faut le compiler pour le traduire en langage assembleur puis l'assembler pour le convertir en code machine comprehensible par le microprocesseur.





**6. Notions d'architecture RISC e CISC:**

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

- l'architecture CISC (complexe Instruction Set Computer)

- l'architecture RISC (Reduced Instruction Set Computer)

**6.1 Architecture CISC:**

C'est une architecture avec un grand nombre d'intructions ou le microprocesseur doit exécuter des taches complexes par instruction unique. Pour une tache donnée, une machine CISC exécute ainsi un petit nombre d'instructions mais chacune nécessite un grand nombre de cycles d'horloge. Le code machine de ces instructions varie d'une instruction à l'autre et nécessite un décodeur complexe (micro-code).

**6.2 Architecture RISC:**

C'est une architecture dans laquelle les instructions sont en nombre réduit (chargement, branchement, appel sous-programmme). Chacune de ces instructions s'exécutent ainsi en un cycle d'horloge. Bien souvent, ces instuctions ne disposent que d'un seul mode d'adressage. Les accés à la mémoire s'effectue seulement à partir de deux instuctions (Load, Store) .

**6.3 Comparaison:**

Si on diminue le nombre d'instructions, on crée des instructions plus complexe CISC qui nécessitent plus de cycle pour etre décodées, et si en diminue le nombre de cycle par instruction on crée des instructions simples RISC mais en augmente le nombre 'instructions nécessaires pour réaliser le meme traitement.

**\*Architecture RISC :**

- instructions simples ne prenant qu'un seule cycle

-instructions au format fixe

-décodeur simple (cablé)

- beaucoup de registre

- seule les instructions Load et Store on accés à la mémoire

- peu de mode d'adressage

- compilateur complexe

**\* Architecture CISC :**

- instructions complexes prenant plusieurs cycles

- instructions au format variable

- décodeur complexe (micro-code)

- peu de registre

- toutes les instructions sont succeptible d'accéder à la mémoire

- beaucoup de mode d'adressage

- compilateur simple