Chapitre IV : Langage Machine

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

Le langage **assembleur** est le langage le plus « proche » du langage machine. Il est composé par des instructions en général 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.

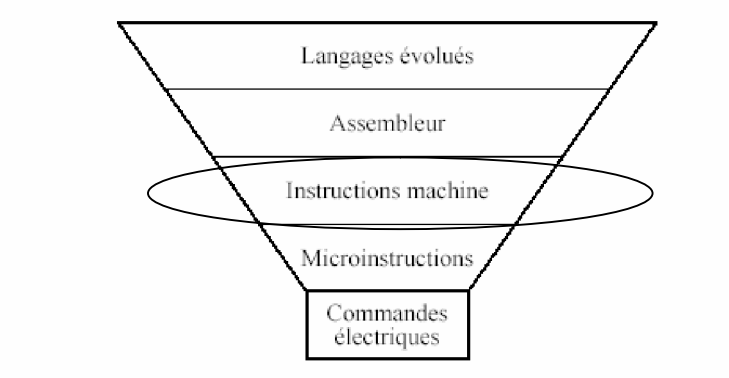

Figure 1

La figure 1 schématise les différents niveaux de programmation. Lorsque l'utilisateur peut accéder au niveau de la micro-programmation la machine est dite micro-programmable.

Pour programmer un ordinateur on utilise généralement des langages dits évolués ou de haut niveau : C, C++, Java, Basic, Fortran, Pascal, Ada, Assembleur, etc. Cependant l'unité centrale ne peut exploiter que les instructions machine : les codes binaires qui sont chargés dans le registre instruction.

Le terme de langage désigne un jeu d'instructions et de règles syntaxiques. A l'aide d'un langage évolué le programmeur écrit un code source. Celui-ci n'est pas directement exécutable par l'ordinateur. Il faut le traduire en code machine ou code objet. C'est le rôle des compilateurs ou assembleurs et des interpréteurs. Un interpréteur ne produit pas de code objet il traduit les instructions directement au fur et à mesure de l'exécution du programme.

I/- Registres de l'unité centrale

Le nombre et le type des registres implantés dans une unité centrale font partie de son architecture et ont une influence importante sur la programmation et les performances de la machine. Nous voudrions ici passer en revue les registres fondamentaux, que l'on retrouve sur toutes les machines ou presque.

Compteur ordinal (CO) : Ce registre (Program Counter : PC) contient l'adresse de la prochaine instruction à exécuter. Après chaque utilisation il est automatiquement incrémenté du nombre de mots correspondant à la longueur de l'instruction traitée : le programme est exécuté en séquence. En cas de rupture de séquence (branchement conditionnel ou non, appel à une routine, etc.) il est chargé avec la nouvelle adresse. Le compteur ordinal, dont la taille dépend de l'espace adressable, n'est généralement pas accessible directement au programmeur.

Registre instruction (RI) : C'est le registre de destination dans lequel le CPU transfert l'instruction suivante à partir de la mémoire. Sa taille dépend du format des instructions machines. Le décodeur utilise le registre instruction pour identifier l'action (ou le micro-programme) à entreprendre ainsi que les adresses des opérandes, de destination ou de saut. Le programmeur n'a pas accès au registre instruction.

Accumulateur (ACC) : L'accumulateur est un registre de l'unité arithmétique et logique. Il a de nombreuses fonctions. Il peut contenir un des deux opérandes avant l'exécution et recevoir le résultat après. Cela permet d'enchaîner des opérations. Il peut servir de registre tampon pour les opérations d'entrées/sorties : dans certaines machines c'est le seul registre par lequel on peut échanger des données directement avec la mémoire. Sa taille est égale à la longueur des mots en mémoire. Il possède souvent une extension (Q), pour les multiplications, décalages, divisions, etc. Le registre ACC est accessible au programmeur et très sollicité. Certaines machines possèdent plusieurs accumulateurs.

Registres généraux ou banalisés : Ils permettent de limiter les accès à la mémoire, ce qui accélère l'exécution d'un programme. Ils peuvent conserver des informations utilisées fréquemment, des résultats intermédiaires, etc. Ils sont accessibles au programmeur.

Registres d'indice ou d'index : (XR) Ils peuvent être utilisés comme des registres généraux mais ils ont une fonction spéciale utilisée pour l'adressage indexé. Dans ce cas l'adresse effective d'un opérande est obtenue en ajoutant le contenu du registre d'index à l'adresse contenue dans l'instruction. Ce type d'adressage et de registre est très utile pour manipuler des tableaux. Le programmeur dispose alors d'instructions permettant l'incrémentation ou la décrémentation du registre d'index. En particulier les registres d'index peuvent être incrémentés ou décrémentés automatiquement après chaque utilisation. Dans certaines machines ces instructions sont applicables à tous les registres généraux, il n'y a alors pas de registre d'index spécifique.

Registre de base : A de très rares exceptions à l'intérieur d'un programme on ne fait référence qu'à des adresses relatives ou virtuelles. Par contre l'unité centrale a besoin de connaître les adresses physiques où se situent réellement instructions et données. Celles-ci dépendent de l'endroit où a été chargé le programme en mémoire, l'espace physique occupé par un programme pouvant ne pas être contigu. Le rôle des registres de base est de permettre le calcul des adresses effectives.

Un registre de base contient une adresse de référence, par exemple l'adresse physique correspondant à l'adresse virtuelle 0. L'adresse physique est obtenue en ajoutant au champ adresse de l'instruction le contenu du registre de base. Le registre de base est encore utilisé quand le nombre de bits du champ adresse ne permet pas d'accéder à toute la mémoire.

Registre d'état (RE ou PSW : Program Status Word) : Une partie des bits de ce registre, aussi appelé registre condition, constitue des drapeaux (flags) qui indiquent certains états particuliers.
Par exemple à la fin de chaque opération on peut y trouver le signe du résultat (Négatif, Zéro ou Positif), ainsi qu'une éventuelle retenue (Carry) ou un dépassement de capacité (Overflow). Ces bits indicateurs peuvent être testés pour déterminer la suite du déroulement du programme : branchements conditionnels. On trouve également le mode de fonctionnement de l'unité centrale.
Deux modes sont possibles le mode utilisateur et le mode système ou superviseur. Dans le mode utilisateur certaines instructions sont interdites : elles provoquent un déroutement vers le système d'exploitation. Un bit peut également indiquer un déroulement pas à pas : demande de trace (T).
Le registre peut aussi contenir le niveau de l'interruption en cours de traitement ou un masque des niveaux d'interruptions autorisés.

Registre pointeur de pile (PP) : Une pile est une zone mémoire dans laquelle les informations sont rangées de façon contiguë. L'usage d'une pile permet la récursivité des appels à des routines ou fonctions. Elle sert à sauvegarder l'adresse de retour, les registres qui sont utilisés par la fonction appelée. Elle peut également servir au passage direct ou indirect des arguments. Le pointeur de pile (Stack Pointer : SP) indique le sommet de la pile : la position de la dernière information enregistrée. Dans certaines machines le pointeur de pile indique la position où sera mémorisée la prochaine donnée. Le fonctionnement d'une pile est du type Dernier Entré Premier Sorti (LIFO : Last In First Out). Les deux principales opérations liées à la pile concernent l'ajout d'un élément dans la pile ou le retrait, souvent nommées respectivement PUSH et PULL. Lorsqu'une donnée est enregistrée dans la pile elle est placée à l'adresse qui suit celle du dernier mot stocké. Après l'opération le pointeur de pile est incrémenté. Lorsque un mot est retiré de la pile il correspond à la dernière information qui y a été entrée. Après l'opération le pointeur est décrémenté. Une pile est réservée à l'usage de l'unité centrale, en particulier pour sauvegarder les registres et l'adresse de retour en cas d'interruption ou lors de l'appel d'une procédure. Le pointeur de pile est accessible au programmeur, ce qui est souvent source d'erreur. Certaines machines sont dotées de plusieurs pointeurs de piles.

Pour améliorer les performances d'un processeur il faut disposer du plus grand nombre de registres possible. On réduit ainsi les accès à la mémoire. De plus, il est préférable d'éviter de les spécialiser. On évite ainsi des transferts entre registres, par exemple pour calculer la valeur d'un indice et utiliser ensuite cet indice pour modifier une case d'un tableau.

II/- Structures des instructions au niveau machine

II.1/- Format des instructions

Les ordinateurs sont capables d'effectuer un certain nombre d'opérations élémentaires.
Une instruction au niveau machine doit fournir à l'unité centrale toutes les informations nécessaires pour déclencher une telle opération élémentaire : type d'action, où trouver le ou les opérandes, où ranger le résultat, etc. C'est pourquoi une instruction comporte en général plusieurs champs ou groupes de bits. Le premier champ contient le code opération. Les autres champs peuvent comporter des données ou l'identification des opérandes. La figure (2) donne quelques exemples d'instructions à n adresses, pour n = 0, 1 et 2. Sur certaines machines les instructions sont toutes de même longueur, sur d'autres cette longueur peut varier avec le code opération ou le mode d'adressage.
On distingue six groupes d'instructions :

- transferts de données : de mémoire à registre, de registre à registre, de registre à mémoire.

- opérations arithmétiques : addition, soustraction, multiplication et division.

- opérations logiques : ET, OU inclusif, NON, OU exclusif, etc…

- contrôle de séquence : branchements conditionnels ou non, appel de procédure, etc..

- entrées/sorties.

- manipulations diverses : décalage, conversion de format, permutation circulaire des bits, échange d'octets, incrémentation, etc…

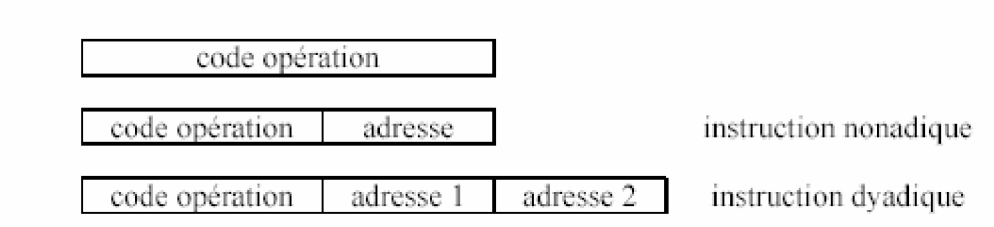

Figure 2

Le choix de la longueur et du format des instructions est une étape très importante dans la conception d'une nouvelle architecture. C'est encore une affaire de compromis. La longueur des instructions se répercute sur la dimension de la mémoire centrale donc sur le coût : il faut deux fois plus de bits pour stocker un programme de n instructions de 32 bits qu'un programme de n instructions de 16 bits. La longueur des instructions par rapport à celle du mot mémoire influence également le temps de traitement : il faut tenir compte du temps de transfert des mots qui constituent une instruction. Ce choix dépend des vitesses relatives d'accès mémoire et de traitement effectif par l'unité centrale. Le temps de recherche doit être minimisé pour les processeurs très rapides.

La largeur en bits de chacun des différents champs est également importante, en particulier pour le code opération. Le nombre de bits est déterminé par le nombre d'opérations distinctes envisagées : n bits autorisent 2^n instructions. Cependant toutes les opérations ne nécessitent pas forcément le même nombre de champs ou des champs de même longueur. Ainsi sur une machine même pour une longueur d'instruction donnée le format des instructions peut ne pas être fixe. Il peut dépendre du type d'opération. Pour illustrer le concept du code opération expansif imaginons des instructions de 16 bits découpées en quatre champs de 4 bits (fig. 3). Avec ce format nous pouvons définir 16 instructions à 3 adresses. C'est peu et toutes les instructions ne nécessitent pas trois adresses. Nous pouvons également définir 15 instructions à 3 adresses (code opération 0000 à 1110 dans le premier champ), 14 instructions à 2 adresses (code opération sur 8 bits 1111 0000 à 1111 1101 identifié par les quatre premiers bits à 1), 31 instructions à 1 adresse (code opération sur 12 bits 1111 1110 0000 à 1111 1111 1110 identifié par les sept premiers bits à 1) et 16 instructions sans champ adresse (identifiée par les douze premiers bits à 1). Soit un total de 76 instructions. Dans d'autres variantes la longueur et le format des instructions sont définis par les premiers bits du code. D'autre part comme la capacité mémoire ne cesse de croître les champs d'adresse demandent de plus en plus de bits. C'est pourquoi, pour éviter une inflation de certains registres comme le registre instruction, aujourd'hui on préfère les instructions à une adresse. On peut par exemple faire en sorte qu'un des opérandes soit toujours l'accumulateur et que ce même registre recueille le résultat.

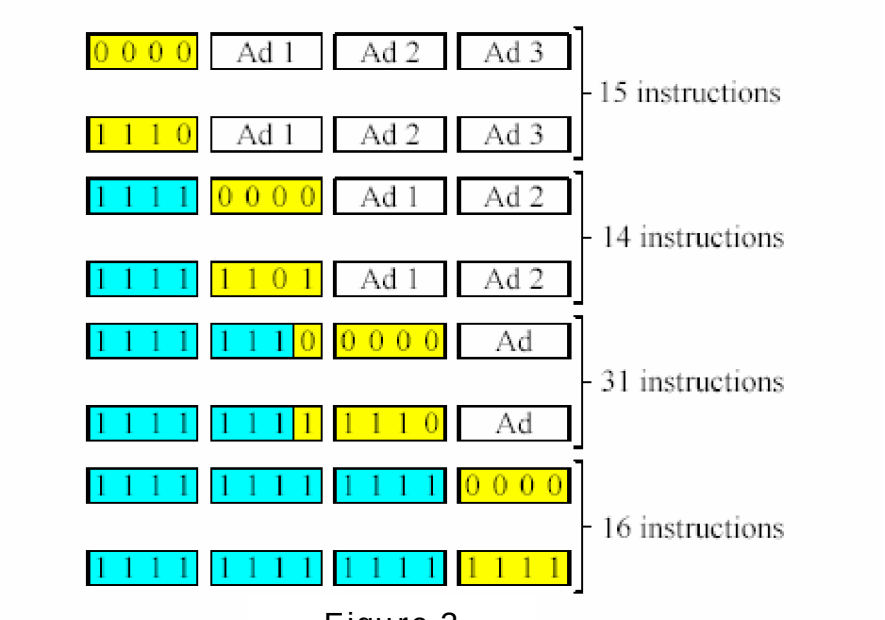
0 0 0 0
Ad 1
Ad 2
Ad 3
15 instructions
1 1 1 0
Ad 1
Ad 2
Ad 3
1 1 1 1
0 0 0 0
Ad 1
Ad 2
14 instructions
1 1 1 1
1 1 0 1
Ad 1
Ad 2
1 1 1 1
1 1 1 0
0 0 0 0
Ad
31 instructions
1 1 1 1
1 1 1 1
1 1 1 0
Ad
1 1 1 1
1 1 1 1
1 1 1 1
0 0 0 0
16 instructions
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1

Figure 3