Support de cours de la matière : Architecture des Ordinateurs (AO)

Chapitre III : Notions sur les instructions d'un ordinateur

III.1 Langage de haut niveau, assembleur, langage machine

Langage machine

Un ordinateur ne peut comprendre que directement son propre langage machine. les langages machine sont constitués d'une série de nombre de 0 et de 1, qui chargent les ordinateurs d'exécuter les opérations plus élémentaires une par une. Les langages machine sont dépendant de la machine. Autrement dit, un langage machine donné ne peut être employé que sur un seul type d'ordinateur. Les langages machines sont fastidieux pour les humains.

Langage d'assemblage

Plus les ordinateurs gagnaient en popularité, plus il devenait évident que la programmation en langage machine exigeait trop de temps et d'efforts. Au lieu d'utiliser les suites de nombres que les machines pouvaient comprendre directement, les programmeurs commencèrent donc à utiliser des mots tirées de l'anglais pour représenter les opérations élémentaires de l'ordinateur, établissant ainsi la base des langages d'assemblage. Ils développent aussi des programmes de traduction appelés *assembleurs* pour convertir en langage machine les programmes écrits en langage d'assemblage.

Langage de haut niveau

Les langages d'assemblage contribuèrent à augmenter la popularité des ordinateurs mais exigeaient encore beaucoup d'instructions pour accomplir mêmes tâches les plus simples. Afin d'accélérer le processus de la programmation, on développa des langages de haut niveau dans lesquels les instructions uniques exécutent des tâches considérables. Des programmes de traduction appelés *compilateurs* convertissent en langage machine des programmes écrits en langage de *haut niveau*. Les langages de haut niveau permettent d'écrire des instructions qui ressemblent à la langue courante et qui contiennent des énumérations mathématiques usuelles.

De toute évidence, les langages de haut niveau sont beaucoup plus attrayants du point de vue du programmeur, comparativement aux langages machines ou d'assemblage. Le C, le C++, le Python et le Java figurent parmi les langages de haut niveau les plus puissants.

III.2 Les instructions machines usuelles

III.2.1 Constitution

Une instruction est stockée dans une cellule de mémoire.

Elle est constituée de 2 champs :

- Un code opération noté OP qui indique quelle opération doit être effectuée.
- Une adresse notée AD désignant l'opérande de cette opération.

o On peut en distinguer plusieurs types :

- Les instructions élémentaires, comprenant elles-mêmes :
  1. Les instructions de transfert entre la mémoire et l'accumulateur de l'UAL : chargement et stockage ;
  2. Les instructions effectuant des calculs arithmétiques et logiques.
- Les instructions de contrôle de séquence, permettant de réaliser les instructions conditionnelles et répétitives.

On conçoit que, pour chacune de ces catégories, une instruction doit être d'une extrême simplicité et ne peut agir que sur un seul argument, appelé ici un *opérande*.

III.2.2 Exécution

L'instruction est contenue dans le registre d'instruction RI.

Instructions de transfert

Le champ AD du registre d'instruction RI désigne l'adresse de l'opérande.

Instruction de chargement

Le contenu de la cellule désignée par cette adresse est transféré dans l'accumulateur :

ACC ⟵ [AD]

Instruction de stockage

Le contenu de l'accumulateur est transféré dans la cellule désignée par cette adresse :

[AD] ⟵ ACC

Instructions arithmétiques et logiques

Le champ AD du registre d'instruction RI désigne l'adresse de l'opérande.

Le champ OP du registre d'instruction RI désigne l'opération à faire exécuter par l'UAL.

Le contenu de la cellule désignée par cette adresse est transféré dans le registre auxiliaire :

AUX ⟵ [AD] .

Le code opération active la fonction correspondante dans l'UAL.

Le résultat de l'opération est retourné dans l'accumulateur.

Instructions de contrôle de séquence

Le champ AD du registre d'instruction RI désigne la valeur de l'opérande.

Instruction de rupture de séquence inconditionnelle

Cette valeur est transférée dans le compteur ordinal : CO ⟵ AD

Instruction de rupture de séquence conditionnelle

Cette valeur est transférée dans le compteur ordinal si, et seulement si, le test invoqué par l'instruction est positif.

III.3 Principe de compilation et d'assemblage (notions de base)

La majorité des programmes actuels sont écrits dans des langages de haut niveau, comme le C, le Java, le C++, l'Erlang, le PHP, etc. Mais ces langages ne sont pas compréhensibles par le processeur, qui ne comprend que le langage machine. Les codes sources écrits dans des langages de haut niveau doivent donc être traduit en langage machine par la *chaine de compilation*. A l'heure actuelle, la majorité des compilateurs ne traduit pas directement un langage de haut niveau en langage machine, mais passe par un langage intermédiaire : l'assembleur. Cet assembleur est un langage de programmation, utilisé autrefois par les programmeurs, assez proche du langage machine. Il va de soit que cet assembleur doit être traduit en langage machine pour être exécuté par le processeur. Tout ce qui se passe entre la compilation et l'exécution du programme est pris en charge par trois programmes qui forment ce qu'on appelle la chaîne d'assemblage.

Cette chaine d'assemblage commence par le logiciel d'assemblage qui traduit le code assembleur en code machine. Ce code machine est alors mémorisé dans un fichier *objet*. Ensuite, *l'éditeur de liens*, ou *linker*, qui combine plusieurs fichiers objets en un fichier *exécutable*. Enfin, le chargeur de programme, aussi appelé *loader*, charge les programmes en mémoire. L'ensemble regroupant compilateur et chaine d'assemblage, avec éventuellement des interpréteurs, est appelé la chaine de compilation.

Traduire de l'assembleur en instructions-machine est la première tâche du logiciel d'assemblage, mais ce n'est pas la seule. Il s'occupe aussi de la résolution des symboles, à savoir qu'il transforme les labels (et autres symboles) en adresses mémoires (l'adresse de l'instruction cible d'un branchement, ou adresse d'une variable). Pour détailler plus, il faut faire la différence entre les symboles locaux, utilisés uniquement dans le module où ils sont définis (ils servent à fabriquer des fonctions, variables statiques et structures de contrôle), des symboles globaux, référencés dans d'autres fichiers (ils servent pour nommer les fonctions accessibles à l'extérieur du module, et les variables globales).

III.4 Unité de commande :

C’est la partie du processeur chargée d’ordonnancer les différentes étapes du traitement d’une instruction. Le fonctionnement peut être décrit de la façon suivante : l’unité de commande va chercher en mémoire centrale une instruction en envoyant une adresse et une commande à la mémoire. L’instruction, enregistrée sous forme binaire à l’adresse donnée, et transférée vers l’unité de commande, où son décodage permet de déterminer l’opération demandée. Cette information est utilisée pour générer les signaux nécessaires à l’UAL pour déclencher l’exécution de l’instruction. Les données à traiter sont aussi cherchées en mémoire par l’unité de contrôle et transférées directement à l’UAL.

Pour ce travail, le processeur requiert des petites mémoires propres (indépendamment de la mémoire principale). Cette mémoire à accès très rapide lui permet de stocker des résultats temporaires ou des informations de commande. Cette mémoire est formée de quelques registres. Chaque registre ayant une fonction particulière.

Les principaux dispositifs de l’unité de commande, qui entrent en jeu lors de la recherche en mémoire et du décodage d’une instruction (cycle de recherche), sont :

- Le compteur ordinal (CO), qui est un registre contenant l’adresse en mémoire où est stockée l’instruction à chercher ;
- Le registre instruction (RI), qui reçoit l’instruction qui doit être exécutée ;
- Le décodeur de code opération, qui détermine quelle opération doit être effectuée, parmi toutes les opérations possibles ;
- Le séquenceur, qui génère les signaux de commande ;
- L’horloge, qui émet des impulsions électroniques régulières, synchronisant ainsi toutes les actions de l’unité centrale.

III.5 Phases d’exécution d’une instruction

Le programme est représenté par une série d’instructions qui réalisent des opérations en liaison avec la mémoire vive de l’ordinateur. Il y a quatre étapes lors du traitement des instructions :

1. FETCH : Recherche de l’instruction.
2. DECODE : Décodage de l’instruction.
3. EXECUTE : Exécution des opérations.
4. WRITEBACK : Écriture du résultat.

FETCH

La première étape, FETCH (recherche), consiste à rechercher une instruction dans la mémoire vive de l'ordinateur. L'emplacement dans la mémoire est déterminé par le compteur de programme, qui stocke l'adresse de la prochaine instruction dans la mémoire de programme.

Après qu'une instruction a été recherchée, le compteur de programme est incrémenté par la longueur du mot d'instruction. L'instruction que le processeur recherche en mémoire est utilisée pour déterminer ce que le CPU doit faire.

DECODE

Dans l'étape DECODE (décodage), l'instruction est découpée en plusieurs parties telles qu'elles puissent être utilisées par d'autres parties du processeur. La façon dont la valeur de l'instruction est interprétée est définie par le jeu d'instructions du processeur. Souvent, une partie d'une instruction, appelée opcode (code d'opération), indique quelle opération est à faire, par exemple une addition. Les parties restantes de l'instruction comportent habituellement les autres informations nécessaires à l'exécution de l'instruction comme par exemple des valeurs pour l'addition.

EXECUTE

Après les étapes de recherche et de décodage arrive l'étape EXECUTE (exécution) de l'instruction. Au cours de cette étape, différentes parties du processeur sont mises en relation pour réaliser l'opération souhaitée. Par exemple, pour une addition, l'unité arithmétique et logique (UAL) sera connectée à des entrées et des sorties. Les entrées présentent les nombres à additionner et les sorties contiennent la somme finale. L'UAL contient le circuit électronique pour réaliser des opérations d'arithmétique et de logique simples sur les entrées (addition, opération sur les bits). Si le résultat d'une addition est trop grand pour être codé par le processeur, un signal de débordement est positionné dans un registre d'état.

WRITEBACK

La dernière étape WRITEBACK (écriture du résultat), écrit tout simplement les résultats de l'étape d'exécution en mémoire. Très souvent, les résultats sont écrits dans un registre interne au processeur pour bénéficier de temps d'accès très courts pour les instructions suivantes. Dans d'autres cas, les résultats sont écrits plus lentement dans des mémoires RAM, donc à moindre coût et acceptant des codages de nombres plus grands.

III.6 UCC pipeline

Un pipeline (ou chaîne de traitement), est l'élément d'un processeur dans lequel l'exécution des instructions est découpée en plusieurs étapes. Le premier ordinateur à utiliser cette technique est l'IBM Stretch, conçu en 1961. Avec un pipeline, le processeur peut commencer à exécuter une nouvelle instruction sans attendre que la précédente soit terminée. Chacune des étapes d'un pipeline est appelé *étage*. Le nombre d'étages d'un pipeline est appelé sa *profondeur*.

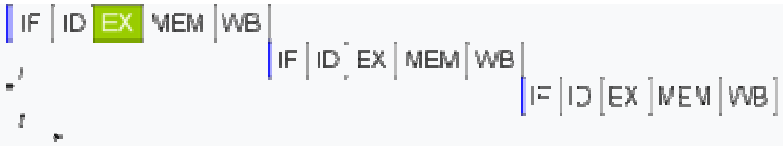
Pipeline classique

Pour donner un exemple de pipeline, nous allons aborder un pipeline parmi les plus connus. Celui-ci s'appelle le Classic RISC pipeline. Il s'agit du pipeline créé par David Patterson, inventeur des processeurs RISC et du concept de pipeline. Ce pipeline est utilisé dans de nombreux documents (cours de David Patterson, notamment) comme une introduction au concept de pipeline.

Avec ce pipeline, 5 étapes sont nécessaires pour traiter une instruction :

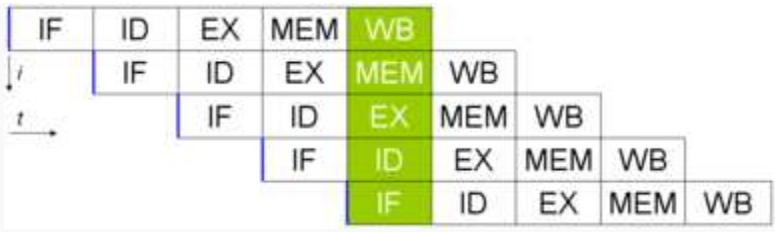
1. IF (Instruction Fetch) charge l'instruction à exécuter dans le pipeline.
2. ID (Instruction Decode) décode l'instruction et adresse les registres.
3. EX (Execute) exécute l'instruction (par la ou les unités arithmétiques et logiques).
4. MEM (Memory), dénote un transfert depuis un registre vers la mémoire dans le cas d'une instruction du type STORE (accès en écriture) et de la mémoire vers un registre dans le cas d'un LOAD (accès en lecture).
5. WB (Write Back) stocke le résultat dans un registre. La source de ce résultat peut être la mémoire (à la suite d'une instruction LOAD), l'unité de calcul (à la suite d'un calcul à l'étape EX) ou bien un registre (cas d'une instruction MOV).

En supposant que chaque étape met 1 cycle d'horloge pour s'exécuter, il faut normalement 5 cycles pour exécuter une instruction, 15 pour 3 instructions :



Séquençage des instructions dans un processeur sans pipeline. Il faut 15 cycles pour exécuter 3 instructions.

En utilisant la technique du pipeline, notre processeur peut alors contenir plusieurs instructions, chacune à une étape différente.



Séquençage des instructions dans un processeur doté d'un pipeline à 5 étages. Il faut 9 cycles pour exécuter 5 instructions. À t = 5, tous les étages du pipeline sont sollicités, et les 5 opérations ont lieu en même temps.

Les 5 instructions s'exécuteront en 9 cycles, et le processeur sera capable de terminer une instruction par cycle à partir de la cinquième, bien que chacune d'entre elles nécessite 5 cycles pour s'exécuter complètement. Au 5e cycle, tous les étages sont en cours d'exécution. Aujourd'hui tous les microprocesseurs sont pipelinés :

Processeur	Profondeur
Intel Pentium 4 Prescott	31
Intel Pentium 4	20
AMD K10	16
IBM POWER5	16
IBM PowerPC 970	16
Intel Core 2 Duo	14
Intel Pentium II	14
Sun UltraSPARC IV	14

Processeur	Profondeur
Sun UltraSPARC IIi	14
AMD Opteron 1xx	12
AMD Athlon	12
IBM POWER4	12
Intel Pentium III	10
Intel Itanium	10
MIPS R4400	8
Motorola PowerPC G4	7

Certaines architectures ont largement augmenté le nombre d'étages, celui-ci pouvant aller jusqu'à 31 pour la microarchitecture *Prescott* d'Intel. Une telle architecture sera appelée *superpipelinée*.