Architecture des ordinateurs

J.L. Damoiseaux

1 octobre 1998

Avertissements

Ce document est le support des 5 heures de cours consacrées à l'enseignement de l'architecture des machines. Il décrit donc succinctement l'architecture d'un ordinateur actuel, et son seul objectif est de donner à l'étudiant une idée du fonctionnement général d'un ordinateur. De nombreux aspects et non des moindres ayant été volontairement oubliés, pour plus de précisions, vous pourrez consulter avec profit les nombreux ouvrages existant sur le domaine, et notamment ceux cités en référence :

- Architecture de l'ordinateur
  Andrew Tannenbaum
  3^{e}édition, InterEditions
- Introduction au microprocesseur
  Charles M-Gilmore
  MC Graw Hill
- Traité d'Electricité, Volume XIV, Calculatrices
  Jean-Daniel Nicoud
  Presses Polytechniques Romandes
- Architecture des systèmes d'exploitation
  Michael Griffiths, Michel Vayssade
  Hermes
- Architecture des ordinateurs
  John L. Hennessy, David A. Patterson
  Mc Graw-Hill

Table des matières

Chapitre 1

Présentation

1.1 Historique

L'histoire de l'informatique débute dans les années 1940, date à laquelle J. Presper Ecker et John Mauchly ont construit, pour le compte de l'armée américaine, le premier ordinateur électronique universel. Cette machine, appelée l'ENIAC (Electronic Numerical Integrator And Calculator), était utilisée pour calculer des tables de tir d'artillerie. D'une taille respectable (30 m de long, 2,80 m de haut et quelques mètres de large), il était constitué entre autre choses de 18000 tubes à vide. D'une programmation agréable puisqu'il fallait enficher des câbles, positionner des interrupteurs, saisir les données sur des cartes perforées, il permettait avec ces 20 registres de 10 chiffres de faire une addition en 200 microsecondes.

C'est en 1950 que le premier ordinateur fut commercialisé. L'UNIVAC coûtait alors 250000 dollars et connu un succès retentissant puisqu'il fut vendu à 48 exemplaires ! ! ! En 1950, une petite entreprise spécialisée dans les machines comptables et les cartes perforées, décida de construire des ordinateurs. Ainsi, en 1952, IBM commercialisa l'IBM 701 qui se vendit à 19 unités. Depuis, des milliers d'ordinateurs se sont succédés, et il est possible de les classer en 5 générations selon la technologie employée :

1. 1950–1959 ordinateurs électroniques, technologie des tubes à vide ;
2. 1960–1968 ordinateurs meilleur marché, technologie des transistors ;
3. 1969–1977 miniordinateurs, technologie des circuits intégrés ;
4. 1978–199 ? station de travail et PC, technologie LSi et VLSI ;
5. 199 ? ordinateurs massivement parallèle, technologie multiprocesseurs ;

1.2 L'ordinateur de M. et Mme Toutlemonde

L'ordinateur de M. et Mme Toutlemonde se décompose (Fig. 1.1) en trois entiés reliées entre elles par un bus d'interconnexion, et qui sont l'unité centrale, l'unité de gestion des périphériques, et la mémoire,

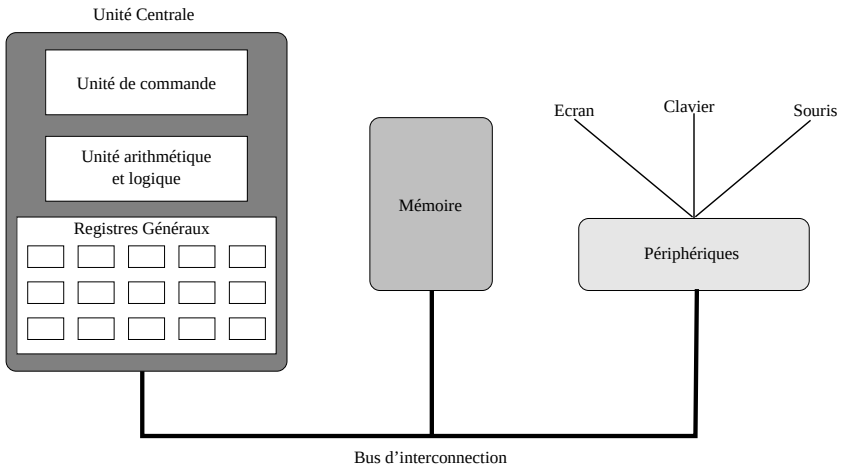

FIG. 1.1 – Structure d'un ordinateur

1.2.1 L'unité centrale

L'unité centrale est le "cerveau" de l'ordinateur. Tel un chef d'orchestre elle commande l'ensemble du système, tel un ouvrier elle exécute les programmes stockés en mémoire centrale. Cette unité centrale est composée d'une unité de commande qui charge et décode les instructions, d'une unité arithmétique et logique qui les exécute, et de registres qui servent en autres choses à stocker les résultats intermédiaires. L'une des caractéristiques essentielle de l'unité centrale est sa fréquence de cadencement (ou fréquence d'horloge) qui détermine sa rapidité d'exécution.

1.2.2 La mémoire centrale

La mémoire d'un ordinateur sert à stocker les programmes et les données qu'ils manipulent. Elle est composée d'une suite de cellules (appelées également mots) rangées consécutivement, chaque cellule étant référencée par une adresse et accessible indépendamment des autres. Une cellule est elle-même composée d'éléments appelés bit(`binary digit`), et mémorisant deux états (0 ou 1) ; un mot de 8 bits est appelé un octet.

On distingue toujours deux types de mémoire centrale :

– la mémoire vive ou `RAM` (`Random Access Memory`) qui mémorise des informations tant que l'ordinateur est sous tension ; cette mémoire contiendra les données et les programmes qui seront exécutés sur l'ordinateur ;
– la mémoire morte `ROM` (`Read Only Memory`) qui contient des informations non modifiables et perdurant même après l'arrêt de l'ordinateur ; cette mémoire contient des informations utilisées lors du démarage de l'ordinateur.

Les caractéristiques essentielles de la mémoire sont :
– la taille du mot exprimé en nombre de bits : celle-ci varie de 8 bits pour les vieux ordinateurs, jusqu'à 64 bits pour les ordinateurs à vocation scientifique ;
– sa capacité exprimée en nombre de mots : celle-ci variant entre plusieurs milliers et plusieurs milliards, la communauté informatique, par soucis de simplification, utilise comme unité de taille le "`K`" (`1 K` = `1024` octets, `1000 K` = `1 Mo`, `1000 Mo` = `1 Go`) ;
– le temps d'accès qui mesure le temps écoulé entre la soumission d'une requête de lecture à la mémoire, et le moment où le résultat est délivré.

1.2.3 L'unité de gestion des périphériques

L'unité de gestion des périphériques assure le lien entre l'ordinateur et le monde extérieur, en gérant les transferts d'informations entre les périphériques et la mémoire centrale. Ces périphériques, reliés à l'ordinateur par l'intermédiaire de prises appelées souvent port (port serie, port parallèle, ...), se répartissent en deux classes :
– ceux qui permettent à l'ordinateur d'échanger des informations avec l'extérieur (écran, clavier, imprimantes, capteurs de mesure, etc.) ;
– et ceux qui permettent de stocker de façon permanente des informations (disques, bandes, cassettes, etc.).

L'écran

Bien qu'il nous soit impossible de concevoir un ordinateur sans écran, il faut cependant savoir que celui-ci ne fait réellement partie intégrante d'un ordinateur que depuis les années 1965. Les principales caractéristiques d'un écran sont :
– la taille de sa diagonale (15", 17", 19", 21", ...) exprimée en pouce ;
– son pitch, à savoir la finesse des points (ou pixels) composant l'image.

Les récents progrès technologiques laissent entrevoir des changements considérables dans le domaine de l'affichage avec l'apparition d'écran à cristaux liquides, d'écran à plasma ou à affichage électroluminescent.

Associé à l'écran, la carte graphique est aussi très importante, car c'est elle qui convertit les signaux numériques issus de la mémoire, en signaux interprétables par l'électronique de contrôle de l'écran. A l'heure actuelle, certaines cartes graphiques intègrent des fonctionnalités très sophistiquées comme l'animation 3D des images, la manipulation des tables de couleurs, la compression vidéo, etc.

Le clavier

Un clavier d'ordinateur ressemble à celui d'une machine à écrire, mais au lieu d'imprimer les caractères sur une feuille de papier, les touches engendrent des signaux électriques correspondant au caractère choisi.

Les imprimantes

Il existe un grand choix d'imprimantes couvrant une gamme étendue de prix et de performances, et de nos jours on trouve des imprimantes électromécaniques, thermiques, électrostatiques, à jet d'encre, et laser. Outre la technique d'impression, de nombreux autres critères permettent de différencier les imprimantes. Par exemple, il est possible de considérer la façon d'imprimer (avec ou sans impacts), la capacité d'imprimer un caractère (imprimante série) ou plusieurs caractères à la fois (imprimante parallèle), le mode de génération des caractères (par points ou préformés), la rapidité d'impression, etc.

Le modem

La connection d'un ordinateur au réseau téléphonique passe par l'utilisation d'un modem (MOdulateur/DEModulateur) relié au port série de l'ordinateur. Un modem convertit les signaux numériques binaires de l'ordinateur en signaux analogiques à deux états pouvant être transmis sur les lignes téléphoniques. Les premiers modems avaient une vitesse de transmissions avoisinant les 300 bits/s, les modems actuels arrivant à des taux de transfert de l'ordre de 56000 bits/s.

Pour assurer que deux dispositifs séries puissent communiquer entre-eux, la norme RS232C a été élaborée en 1969 par l'EIA (Electronics Industries Association). Cette norme définit le circuit de communication série en précisant la fonction des signaux de l'interface série et la connexion physique de cette interface.

Les unités de sauvegarde

Les unités de sauvegarde, appelées mémoires auxiliaires, permettent de conserver de très grands volumes d'informations (jusqu'à plusieurs Téra oc-

tets) avec une bien meilleure fiabilité dans le temps que la mémoire principale. D'un coût très faible, ces mémoires sont toutefois en moyenne 1 million de fois moins rapide que la mémoire principale.

Les solutions technologiques les plus courantes sont :

– la disquette qui est constituée d'un disque tournant à grande vitesse (plusieurs milliers de tours par minute) et dont les deux faces sont recouvertes d'une couche magnétique composée de cellules magnétisables dans un sens ou dans un autre (ce qui correspond aux valeurs `0` et `1`). L'accès à l'information se fait par l'intermédiaire d'une tête de lecture/écriture. Les surfaces du disque sont structurées en pistes (plusieurs centaines par surface) et secteurs (de `10` à `100` par piste) (Fig. 1.2), chaque secteur étant divisé en bloc (nombre minimal d'octets lus ou écrit au cours d'une opération physique). Un disque neuf doit être formaté, c'est-à-dire structuré en pistes et en secteurs. La capacité courante des disquettes est `1,44 Mo` et l'on parle déjà de disquettes d'une capacité de `120 Mo` ;

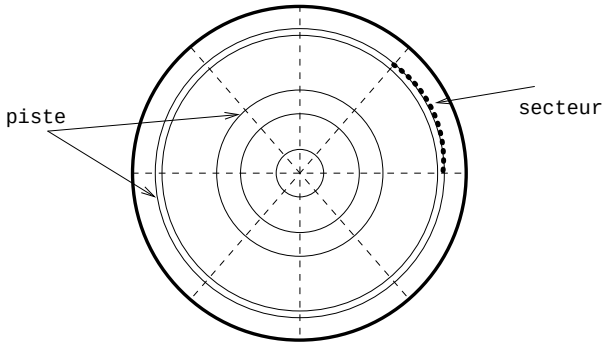

FIG. 1.2 – Structure d'un disque magnétique

– les disques durs sont comparables aux disquettes dans leur principe, mais sont constitués d'un empilement de plusieurs disques magnétiques, avec entre deux disques une tête de lecture/écriture. La capacité moyenne de stockage est de `3 Go` ;
– les bandes magnétiques sont constituées d'un ruban souple magnétisé et enroulé sur un support plastique. Elles sont économiques, possèdent une grosse capacité de stockage, leur durée de vie est longue, mais le temps d'accès aux données est très lent ;
– les cartouches magnétiques suivent le même principe que les bandes magnétiques, mais le format est réduit et la manipulation plus aisée ;
– les disques optiques, derniers nés de la technologie, offrent les meilleures capacités de stockage. Outre les classiques CD-ROM (Compact Disk Read Only Memory), on trouve maintenant sur le marché les

disques WORM (Write Once Read Many) et les disques effaçables.

1.2.4 Le bus d'interconnexion

Le bus d'interconnexion est un ensemble de "fils" reliant entre elles toutes les unités de l'ordinateur. Ces fils peuvent être séparés en trois groupes :

- le bus des adresses qui transmet des informations depuis l'unité centrale vers la mémoire ou les périphériques ;
- le bus des données qui transmet des données entre l'unité centrale, la mémoire et les périphériques ;
- le bus des commandes qui transmet divers signaux (sélection, validation, horloge, alimentation électrique, etc.) entre les divers éléments de l'ordinateur.

Chapitre 2

Mémoire, Représentation des données, et Arithmétique

2.1 La mémoire

De part la technologie employée pour le stockage de l'information dans un ordinateur, celle-ci est codée en base 2. Cependant, l'homme utilisant quotidiennement le système décimal, il est important de savoir convertir un nombre exprimé en base 2 en un nombre exprimé en base 10, et réciproquement.

2.1.1 Conversion binaire–décimal

La conversion binaire–décimal est très simple, comme toutes les conversions base–décimale : soit $d_n d_{n-1} ... d_0$ un nombre exprimé en base b alors $\sum_{i=n}^{0} d_i b^i$ est la valeur de ce nombre en base 10. Pratiquement, cela revient à associer à chaque colonne du nombre binaire un poids correspondant à une puissance de 2, puis à faire le calcul. Par exemple :

- le nombre 11000101001 exprimé en base 2 a pour valeur $1 * 2^{10} + 1 * 2^9 + 1 * 2^5 + 1 * 2^3 + 1 * 2^0$ soit 1577 en base 10
- le nombre 110,101 exprimé en base 2 a pour valeur $1 * 2^2 + 1 * 2^1 + 1 * 2^{-1} + 1 * 2^{-3}$ soit 6.625 en base 10

2.1.2 Conversion décimal–binaire

La conversion décimal–binaire est un peu plus compliquée, car elle consiste en une succession de division entière par 2. Le quotient de la division est écrit directement sous le nombre, tandis que le reste (0 ou 1) est écrit à coté du quotient. On répète ce processus sur le quotient jusqu'à l'obtention de la valeur 0. On obtient alors deux colonnes (les quotients et les restes), le nombre binaire est lu directement à partir du bas sur la colonne des restes. Par exemple, le nombre 1492 en base 10 s'écrit 10111010100 en base 2 :

Nombre/quotient	Reste
1492	0
746	0
373	1
186	0
93	1
46	0
23	1
11	1
5	1
2	0
1	1
0	

2.1.3 Les bases octale et hexadécimale

Il est également pratique d'utiliser en informatique les systèmes de numération octale (base 8) et hexadécimale (base 16). Les nombres exprimés en base 8 sont écrits avec les symboles `0 1 2 3 4 5 6 7`, tandis que les nombres exprimés en base 16 sont écrits avec les symboles `0 1 2 3 4 5 6 7 8 9 A B C D E F`. Le nombre `1F2` appartient sans l'ombre d'un doute au système hexadécimal, mais le nombre `111` peut appartenir aux quatre systèmes. Pour lever cette ambiguïté, on indicera dorénavant le nombre par sa base : 111_{10} 111_2 111_8 111_{16}.

Les conversions octal–binaire, binaire–octal sont très simples. Pour convertir en nombre exprimé en octal en un nombre exprimé en binaire, il suffit de séparer à partir de la droite ce nombre en groupes de 3 digits. Puis, chaque groupe de trois digits est directement exprimé par un chiffre de `0` à `7` ; il peut être nécessaire de compléter le début du nombre binaire par des zéros. La conversion octal–binaire consiste à remplacer chaque chiffre du nombre exprimé en octal par un groupe de trois bits. Par exemple, le nombre 1100101001000_2 s'écrit en octal 14510_8

001	100	101	001	000
1	4	5	1	0

Les conversions hexadécimal–binaire et binaire–hexadécimal suivent le même principe, à ceci près que l'on manipule des groupes de quatre digits. Par exemple, le nombre 1100101001000_2 s'écrit en hexadécimal 1948_{16}

0001	1001	0100	1000
1	9	4	8

2.2 Représentation des données

La représentation des données par un ordinateur consiste à associer à chaque type d'information un certain nombre d'octets. Pour un type d'information donné, le domaine des valeurs possibles dépendra bien entendu du nombre d'octets choisi pour sa représentation, mais également de l'utilisation faite de cette suite d'octets. Etudions donc les conséquences de ces choix sur la représentation des nombres et sur l'arithmétique utilisée par la machine.

2.2.1 Représentation des entiers

Problèmes

L'ensemble des entiers étant infini, il contient des nombres dont la représentation écrite comporte autant de chiffres que l'on veut (par exemple, en astronomie, le nombre d'électrons dans l'univers s'écrit `1` suivi de `78` zéros). Le problème posé ici est donc de savoir comment l'ordinateur s'y prendra pour coder de tels entiers ? La réponse est toute simple, il n'y arrivera pas !!! En effet, le nombre d'octets utilisés pour stocker un entier étant limité, l'ordinateur ne pourra le représenter que par une suite finie de chiffres ; il est d'usage de dire que l'ordinateur travaille en précision finie.

Etudions donc les problèmes d'un système de représentation des entiers, dans lequel le nombre de chiffres utilisés pour cette représentation est fini. Soit l'ensemble des entiers positifs représentables avec trois chiffres décimaux (la transposition au système binaire est immédiate) ; cet ensemble contient exactement les mille éléments `000 001 ...999`. Outre le fait que l'on ne peut pas représenter dans ce système les entiers supérieurs à `999` et les entiers négatifs, l'arithmétique d'un tel système est quelque peu particulière. Ainsi, à la différence de l'arithmétique entière usuelle, dans ce système, l'addition, la multiplication, et la soustraction de deux entiers ne donnent pas toujours un entier. Par exemple, les opérations suivantes :

- $600 + 800 = 1400$
- $050 - 070 = -20$
- $050 * 050 = 20500$

donnent comme résultat des entiers impossibles à représenter dans le système choisi. Dans le même ordre d'idée, dans l'arithmétique usuelle, une expression comme $\mathtt{a} + (\mathtt{b} - \mathtt{c})$ est équivalente à l'expression $(\mathtt{a} + \mathtt{b}) - \mathtt{c}$. Dans le système de représentation précédemment défini, pour des valeurs de `a` égale à `400`, `b` égale à `700` et `c` égale à `300`, la première expression est égale à `800`, tandis que la seconde ne peut être calculée puisque l'expression $\mathtt{a} + \mathtt{b}$ est égale à `1100`, valeur impossible représenter dans ce système.

Les entiers relatifs

Une convention simple pour la représentation des entiers relatifs consiste à donner un sens particulier à l'un des bits servant au codage du nombre. En règle générale, le bit le plus à gauche (bit de poids de fort ou bpf) est choisi pour mémoriser le signe de l'entier relatif. Si le bpf est égal à `0` l'entier est positif, s'il est égal à `1` l'entier est négatif. La perte d'un bit pour la représentation ne permet plus que de coder, en valeur absolue, la moitié de l'intervalle des valeurs précédemment codées. Par exemple, si un seul octet est utilisé pour notre représentation, il est possible de coder soit les nombres compris entre `0` et `255` si tous les bits jouent le même rôle, soit les nombres compris entre -127 et $+127$ (le bpf indique le signe, les sept autres la valeur absolue). Il est toutefois important de comprendre que le nombre et son signe sont deux choses fondamentalement différentes, qui nécessitent des traitements particuliers, ce qui nous amènera à voir d'autre représentation pour les entiers négatifs.

2.2.2 Représentation des réels

Problèmes

Comme pour les entiers, c'est la nature finie de l'ordinateur qui posera des problèmes importants dans l'utilisation des réels. Ainsi, imaginez un système de représentation où un réel se note $\texttt{mantisse} * 10^{\texttt{exposant}}$, la mantisse indiquant la précision (nombre de chiffres significatifs) et l'exposant la grandeur [1]. Dans notre système de représentation, la mantisse signée est exprimée avec trois chiffres décimaux ($0.1 \leq |\texttt{mantisse}| < 1$), et l'exposant signé est exprimé avec deux chiffres décimaux. Les nombres que l'on peut représenter vont de $-0,100e^{+99}$ à $+0,999e^{+99}$. Si l'on représente sur une droite (Fig. 2.1) notre espace des réels :

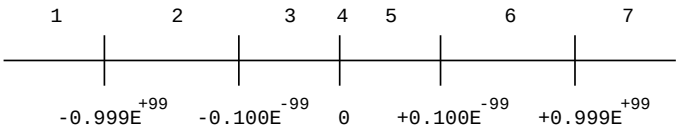

FIG. 2.1 – L'espace des réels dans un système de numération finie

on obtient les 7 régions suivantes :

- la région 1 comprenant les réels négatifs trop grands et donc non représentables ;
- la région 2 comprenant les réels négatifs représentables ;

[1] Cette représentation est appelée depuis fort longtemps par les mathématiciens "la notation scientifique"

– la région 3 comprenant les réels négatifs trop près de zéro et donc non représentables ;
– la région 4 pour le zéro ;
– la région 5 comprenant les réels positifs trop près de zéro et donc non représentables ;
– la région 6 comprenant les réels positifs représentables ;
– la région 7 comprenant les réels positifs trop grands et donc non représentables.

On constate deux différences essentielles entre les réels et les nombres en virgule flottante de notre représentation :

– les régions 1, 3, 5 et 7 n'existent pas dans la réalité ce qui se traduit sur l'ordinateur par un dépassement de capacité (régions 1 et 7), ou une mise à zéro (régions 3 et 5) ;
– notre espace de représentation n'est pas dense (entre deux réels il existe toujours un réel).

Les nombres flottants

Lorsque l'on calcule sur l'ensemble des réels, il est assez rare de devoir travailler avec une précision très importante. Aussi, la communauté informatique à repris à son compte la notation scientifique, et sa version informatique s'appelle la représentation en virgule flottante. Toutefois, la longueur des champs associés à la mantisse et à l'exposant, leurs positions relatives, la complexité des opérations résultantes, ont entraîné une grande variété des formats flottants. La figure suivante (Fig. 2.2) :

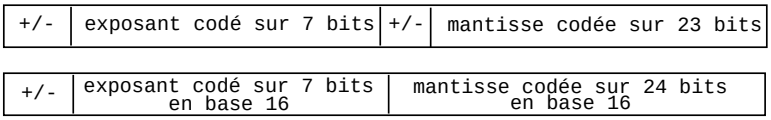

FIG. 2.2 – Deux représentations possibles d'un réel

illustre deux structures utilisées pour le codage des réels :

– la première code, en base 2, l'exposant sur 8 bits (1 bit de signe, 7 bits pour la valeur) et la mantisse sur 24 bits (1 bit de signe, 23 bits pour la valeur) ;
– la seconde, employée par I.B.M. sur ces gros sytèmes, code, en base 16, l'exposant sur 7 bits (représenté en excédent 26) et la mantisse sur 25 bits (1 bit de signe, 6 quartets pour la valeur normalisée)

Norme I.E.E.E. 754

La grande variété des formats de flottants a conduit l'Institut of Electrical and Electronic Engineers (I.E.E.E.) et la Commission Electrotechnique Internationale à proposer un format standard pour la représentation de ces nombres. Outre le fait de définir l'ensembles des nombres flottants, la norme proposée formalise la représentation des nombres particuliers $\pm\infty$, et permet la représentation d'objets qui ne sont pas des nombres `NaN` (ces deux derniers points permettant par exemple d'écrire des programmes traitant de la division par zéro, ou de racine carré de nombres négatifs).

Suivant cette norme, un nombre réel N s'écrit $(-1)^{\mathtt{s}}\ 2^{\alpha}\ (\mathtt{m})$. La mantisse étant normalisée, la partie entière sera toujours égale à 1 (donc implicite et non représentée), et seule la partie purement fractionnaire sera codée (m s'écrit donc 1, f). L'exposant sera codé en excédent : si α est l'exposant dont la valeur absolu s'écrit avec r chiffres ($-\mathtt{p}^{\mathtt{r}} \leq \alpha < \mathtt{p}^{\mathtt{r}}$), alors on pose $\alpha^{+} = \alpha + \mathtt{p}^{\mathtt{r}}$, et l'on obtient $\mathtt{0} \leq \alpha^{+} < \mathtt{2p}^{\mathtt{r}}$ ($\mathtt{p}^{\mathtt{r}}$ est appelé l'excédent). La figure 2.3 décrit cette norme pour le format dit en "simple précision" :

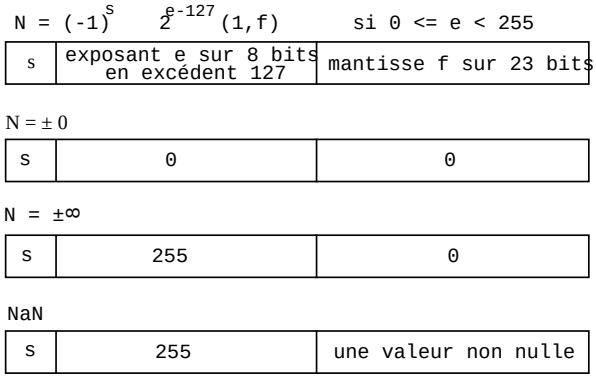

FIG. 2.3 – La norme I.E.E.E. 754

Il est donc possible de représenter les nombres selon l'étendue suivante : pour le plus petit (en valeur absolue) $\mathtt{2}^{\mathtt{0}-\mathtt{127}}.\mathtt{1} = \mathtt{2}^{-\mathtt{127}} = \mathtt{10}^{-\mathtt{38}}$ et pour le plus grand $\mathtt{2}^{\mathtt{254}-\mathtt{127}}.\mathtt{2} = \mathtt{2}^{\mathtt{128}} = \mathtt{10}^{\mathtt{38}}$. Si $\mathtt{e} = \mathtt{255}$ et $\mathtt{f} = \mathtt{0}$, on a la représentation du zéro qui peut être signée. Si $\mathtt{e} = \mathtt{0}$ et $\mathtt{f} = \mathtt{0}$, on a la représentation de l'infini avec son signe. Si $\mathtt{e} = \mathtt{255}$ et $\mathtt{f} \neq \mathtt{0}$, il s'agit d'un "`non nombre`" dont la signification dépendra de l'utilisateur.

2.2.3 Conclusion

De part la nature finie de la mémoire, il existe des différences importantes entre l'arithmétique entière et/ou réelle et l'arithmétique utilisée par les or-

dinateurs. Ces différences peuvent malheureusement entraîner des erreurs ou des imprécisions importantes, alors même que l'ordinateur fonctionne normalement. De là à en conclure que l'ordinateur n'est pas fait pour travailler avec des nombres, reste un peu hâtif. Il est simplement important de connaître ces limites, et d'en tirer les conséquences.

2.3 Numération

Jusqu'à présent, nous avons étudié l'influence de la taille de la représentation de l'information sur l'arithmétique d'un ordinateur. Nous allons maintenant voir, au travers de l'addition, quelques problèmes engendrés par la sémantique de cette représentation.

2.3.1 Addition binaire de nombres entiers

Principe

L'addition binaire est très similaire à l'addition décimale, et suit les principes suivants :

- lorsque nous ajoutons deux nombres, nous commençons par opérer par la colonne de droite,
- tout résultat qui occupe plus d'une colonne entraîne l'apparition d'une retenue qui est ajoutée à la colonne suivante.

Dans le système décimal, tout addition de colonne dont le résultat est supérieur à `9` génère une retenue, dans le système binaire celle-ci est générée pour tout résultat supérieur à 1. D'où la table de l'addition binaire :

+	0	1
0	0	1
1	1	0+r

Un petit exemple (la somme des nombres 99_{10} et 95_{10}) permettra de mieux comprendre cela :

11	1111111	retenue engendrée
99	1100011	A
95	1011111	B
194	11000010	A+B

Il est à noter que dans le cas d'une retenue sur le bit le plus à gauche, celle-ci est généralement perdue ce qui entraîne des résultats faux (débordement de capacité).

Autres représentations des entiers négatifs

Comme il a été précisé précédemment, le codage des entiers relatifs fait jouer au bpf le rôle d'un indicateur de signe. Naïvement, on pourrait considérer qu'un entier relatif est représenté sous la forme de sa valeur absolue et de son signe. Par exemple `01000110111` représente l'entier $+$`567` et `11000110111` représente l'entier $-$`567`. Apparemment naturel, la sémantique de cette représentation génère deux problèmes importants. Tout d'abord, il existe une double représentation du nombre `0`. Ensuite, les règles de l'arithmétique sont rendues très compliquées ; à titre d'exemple, l'addition simple de $+$`567` et de $-$`567` ne donne pas zéro.

D'autres sémantiques ont donc été proposées pour tenter de corriger ces problèmes :

- le complément à `1` ; pour représenter un nombre négatif on inverse chaque bit de sa représentation positive ; par exemple `01000110111` représente l'entier $+$`567` et `10111001000` représente l'entier $-$`567`. Malheureusement les mêmes remarques que pour la sémantique naïve peuvent être faites.
- le complément à `2` ; pour représenter un nombre négatif on le complémente à `1`, puis on y ajoute `1` ; par exemple `0000001000110111` représente l'entier $+$`567` et `1111110111001001` l'entier $-$`567`. Cette sémantique résout les problèmes précédents, simplifie les opérations de calcul, mais complexifie les tests de débordements de capacité.

Voici enfin pour terminer, un exemple d'addition de deux nombres dans les deux systèmes de complément :

décimal	complément à 1	complément à 2
10	00001010	00001010
+ (-3)	11111100	11111101
7	1 00000110	1 00000111
retenue rajoutée	+1	retenue ignorée
	00000111	

2.3.2 Addition binaire de nombres réels

Soit `A` et `B` deux nombres flottants, que l'on peut écrire : $\mathtt{A} = \mathtt{a}.\mathtt{p}^{\alpha}$ et $\mathtt{B} = \mathtt{b}.\mathtt{p}^{\beta}$. Les mantisses `a` et `b`, ainsi que les exposants α et β sont des nombres arithmétiques. La somme de `A` et `B` peut s'écrire :

$$\begin{aligned} \mathtt{A}+\mathtt{B} &= \mathtt{a}.\mathtt{p}^{\alpha} + \mathtt{b}.\mathtt{p}^{\beta} \\ &= \mathtt{a}.\mathtt{p}^{\alpha} + (\mathtt{b}.\mathtt{p}^{\beta-\alpha}).\mathtt{p}^{\alpha} \\ &= (\mathtt{a}+\mathtt{b}.\mathtt{p}^{\beta-\alpha}).\mathtt{p}^{\alpha} \end{aligned}$$

La mantisse $\mathtt{b} \,.\, \mathtt{p}^{\beta-\alpha}$ s'obtient par décalage de la mantisse $\mathtt{b}$:
- si $\beta > \alpha$; un décalage à gauche de $\beta - \alpha$ positions correspond à une multiplication par $\mathtt{p}^{\beta-\alpha}$, ce qui peut entraîner des dépassements de capacité ;
- si $\beta < \alpha$; un décalage à droite de $\alpha - \beta$ positions correspond à une multiplication par $\mathtt{p}^{\alpha-\beta}$, et il convient alors de conserver le signe de $\mathtt{b}$;
- si $\beta = \alpha$, il n'y a pas de décalage à réaliser.

Mathématiquement, cette somme peut également être effectuée de la manière suivante :

$$\begin{aligned} \mathtt{A} + \mathtt{B} &= \mathtt{a} \,.\, \mathtt{p}^{\alpha} + \mathtt{b} \,.\, \mathtt{p}^{\beta} \\ &= (\mathtt{a} \,.\, \mathtt{p}^{\alpha-\beta}) \,.\, \mathtt{p}^{\beta} + \mathtt{b} \,.\, \mathtt{p}^{\beta} \\ &= (\mathtt{a} \,.\, \mathtt{p}^{\alpha-\beta} + \mathtt{b}) \,.\, \mathtt{p}^{\beta} \end{aligned}$$

Malheureusement, l'erreur d'arrondi consécutive à ces deux opérations, peut engendrer des résultats très différents. Ainsi, dans le système décimal, la somme des nombres $\mathtt{A} = 0,0440 \,.\, 10^{+2}$ et $\mathtt{B} = 0,1234 \,.\, 10^{-1}$ est :
- soit égale à

$$\begin{aligned} \mathtt{A} + \mathtt{B} &= 0,0440 \,.\, 10^{+2} + 0,1234 \,.\, 10^{-3} \,.\, 10^{+2} \\ &= (0,0440 + 0,0001234) \,.\, 10^{+2} \\ &= 0,0441234 \,.\, 10^{+2} \end{aligned}$$

- soit égale à

$$\begin{aligned} \mathtt{A} + \mathtt{B} &= 0,0440 \,.\, 10^{+3} \,.\, 10^{-1} + 0,1234 \,.\, 10^{-1} \\ &= (44, +0,1234) \,.\, 10^{-1} \\ &= 44,1234 \,.\, 10^{-1} \end{aligned}$$

Si maintenant nous supposons que les mantisses sont des nombres fractionnaires de quatre chiffres, le premier résultat est donc égal à $0,0441 \,.\, 10^{+2}$, tandis que le second vaut $0,1234 \,.\, 10^{-1}$. Ainsi, lors de l'application du deuxième mode de calcul, le décalage à gauche du premier nombre à entraîné un dépassement de capacité, rendant le résultat complètement faux.

Chapitre 3

L'Unité centrale

3.1 Présentation de l'unité centrale

Nous allons étudier le détail de l'architecture d'un microprocesseur n'existant pas dans le commerce, mais présentant les caractéristiques essentielles de toute unité centrale moderne (Fig. 3.1). Notre microprocesseur possède trois entités fonctionnelles principales qui sont l'Unité Arithmétique et Logique, les registres et la logique de contrôle.

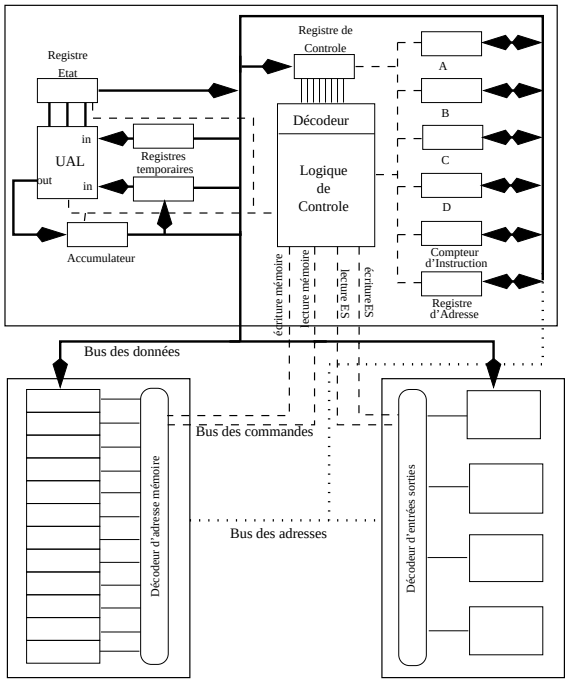

FIG. 3.1 – Architecture d'un microprocesseur

3.1.1 L'Unité Arithmétique et Logique

L'Unité Arithmétique et Logique est un ensemble de circuits combinatoires réalisant, sur les données à traiter, des opérations arithmétiques, logiques, ou de comparaisons. Toutes ces opérations varient bien sûr d'un processeur à l'autre, mais on retrouve au moins les suivantes : l'égalité ou la non égalité à `0`, les opérateurs booléens et ou ou-exclusif, la complémentation logique, les décalages, l'addition, la soustraction, l'incrémentation et la décrémentation.De part la nature des opérations précédemment citées, l'Unité Arithmétique et Logique possède deux entrées reliées au bus interne des données et à l'accumulateur par l'intermédiaire de registres tampons, et une sortie reliée à l'accumulateur.

3.1.2 Les registres

Les registres de l'unité centrale sont des mémoires extrêmement rapides (100 fois plus rapide que la mémoire centrale) servant à stocker temporairement des informations de nature diverse comme le résultat d'un calcul, des données issues de la mémoire centrale, des adresses, etc. De part cette diversité de l'information manipulée, les registres sont donc plus ou moins spécialisés. Là encore, bien que leur nombre varie d'une machine à l'autre, six d'entre eux sont fondamentaux et donc toujours présents. Il s'agit de :

- l'accumulateur ; registre le plus important pour la manipulation des données (les transferts de données entre les entrées-sorties et la mémoire passe par l'accumulateur), il est le complément indispensable de l'Unité Arithmétique et Logique puisque la plupart des opérations l'utilisent. Ce registre dispose en outre d'instructions propres comme la mise à 0 ou à 1 de ces bits, des décalages, etc.
- le compteur d'instruction contient à tous moments l'adresse de la prochaine instruction du programme à exécuter. En effet, un programme étant une suite d'instructions simples (stockées dans la mémoire) décrivant des opérations nécessaires à la résolution d'un problème, suite d'instructions devant se dérouler dans un ordre précis, il est important de connaître en permanence la prochaine instruction à exécuter.
- le registre d'adresse contient l'adresse de l'information (instructions, donnée) située en mémoire ou sur un périphérique, et dont le processeur a besoin. Cette adresse sera le moment venue mise sur le bus des adresses et communiquée à la mémoire ou aux module d'entrées sorties.
- le registre d'instruction contient l'instruction en cours d'exécution ; le décodage de cette instruction permettra à la logique de contrôle de piloter le fonctionnement de l'unité centrale.
- le registre d'état mémorise le résultat de certaines opérations de l'Unité

Arithmétique et Logique, ou des registres. A chaque bit du registre d'état est attribué une signification (le bit z pour savoir si le résultat de l'opération est nul, le bit s pour savoir si le résultat de l'opération est positif ou négatif, le bit r pour savoir si le résultat de l'opération a engendré une retenue). En testant l'état de ces bits, on introduit la notion d'alternative conditionnelle dans le déroulement du programme.
- les registres temporaires permettent à l'Unité Arithmétique et Logique de disposer de moyens de stockage. En effet, celle-ci étant un ensemble de circuits combinatoires, toute donnée placée sur l'une de ses entrées apparaît immédiatement sur sa sortie ; les registres temporaires jouent donc le rôle de verrou.

3.1.3 La logique de contrôle

La logique de contrôle assure le bon fonctionnement de tous les éléments de l'ordinateur, afin que ceux-ci exécutent l'instruction demandée. La logique de contrôle extrait l'instruction du registre d'instruction, détermine ce que l'on doit faire avec les données, et créé les commandes nécessaires à l'exécution de la tache considérée. Généralement, la logique de contrôle est microprogrammée : l'architecture de la logique de contrôle est un véritable petit microprocesseur (compteur d'instruction, registre d'instruction, etc.) sur lequel un microprogramme s'exécute. Ce microprogramme décode et interprète l'instruction puis génère les signaux de contrôle nécessaire à son exécution. A noter que la logique de contrôle assure également le démarrage de l'ordinateur lors de sa mise sous tension, et la gestion des interruptions.

3.2 Fonctionnement de l'unité centrale

3.2.1 Notion d'instruction

Un programme pour être exécuté par le microprocesseur doit être une suite d'instructions, chaque instruction demandant au microprocesseur de réaliser une tâche élémentaire. Concrètement, une instruction est une suite de `0` et de `1` dont la taille varie entre un et plusieurs octets. Cependant, afin de simplifier leur utilisation, elles sont généralement associées à des codes mnémoniques (pour additionner les nombres `5` et `2`, on écrira `ADD 5 2` plutôt que `10110101 00000101 00000010`). Enfin, une instruction doit d'une part indiquer au microprocesseur ce qu'il doit faire, mais d'autre part où se trouvent les données sur lesquelles elle doit travailler. Le format d'une instruction est donc composé d'une partie commande (remettre à zéro, additionner, transférer, etc.) et d'une partie adresse indiquant la localisation des opérandes. Sur l'exemple précédent, on peut imaginer que la partie commande est constituée des `4` premiers bits `1011` et la partie adresse des `4` derniers `0101`.

3.2.2 Exécution d'un programme

L'activité de l'unité centrale au cours de l'exécution d'un programme suit totalement le schéma d'extraction-exécution se résumant à la boucle suivante :

1. chargement dans le registre d'instruction de l'instruction dont l'adresse est contenue dans le compteur d'instruction ;
2. décodage de l'instruction contenue dans le registre d'instruction : incrémentation du compteur d'instruction pour qu'il pointe sur la prochaîne instruction à exécuter, obtention du code de l'opération et des adresses des données mises en jeu ;
3. si nécessaire, transfert des données requises depuis la mémoire vers les registres de l'unité centrale,
4. exécution de l'opération ;
5. si nécessaire, transfert du résultat vers la mémoire ;
6. retourner à l'étape numéro 1.

Ce cycle extraction-exécution est réalisé sous le commandement de la logique de contrôle. Toutefois, pour un déroulement correct de ce cycle, il est nécessaire de synchroniser tous les circuits réalisant ces opérations. C'est l'horloge interne du microprocesseur, qui en délivrant périodiquement des impulsions électriques, assure cette synchronisation. A chaque top de l'horloge une étape de ce cycle aura lieu (Fig. 3.2) ; on dira qu'une instruction prend `n` cycles pour s'exécuter.

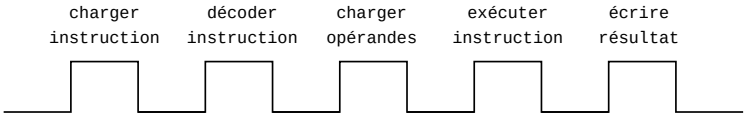

FIG. 3.2 – Le cycle extraction-exécution

Il est également nécessaire de décomposer un cycle d'horloge en sous-cycles ceci afin d'organiser et de synchroniser au mieux les actions du cycle principal. L'exemple suivant (Fig. 3.3) montre la décomposition en sous-cycles du chargement de l'instruction dans le registre d'instruction : le contenu du compteur d'instruction est déposé dans le registre des adresses, puis transmis via le bus des adresses au décodeur d'adresse mémoire, la logique de contrôle génère alors un signal de lecture mémoire, et le contenu de la case adressée est déposé sur le bus des données pour être transmis au registre d'instruction.

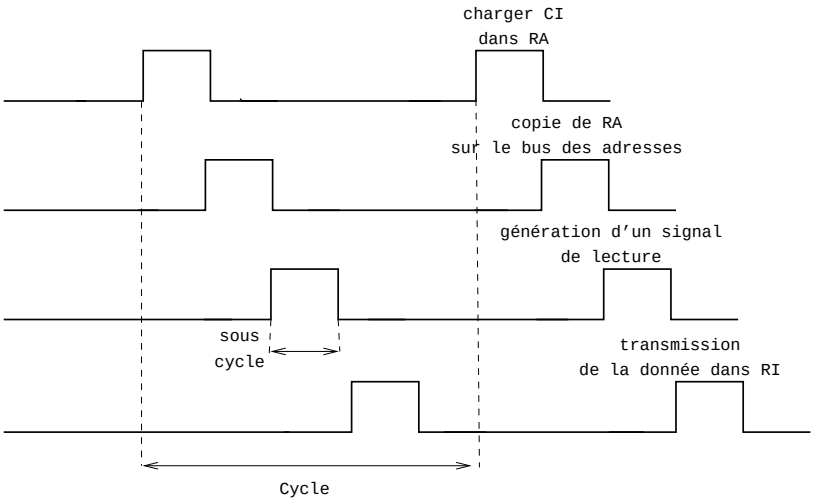

FIG. 3.3 – Décomposition d'un cycle d'horloge en sous cycle

3.2.3 Exemple

Pour illustrer le fonctionnement de notre machine, voici l'exécution d'un programme additionnant un nombre avec le contenu de l'accumulateur. L'unité centrale est à l'arrêt, le compteur d'instruction contient l'adresse de la prochaine instruction à exécuter (Fig. 3.4)

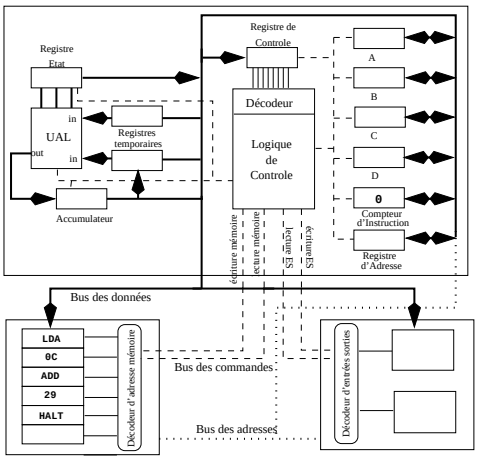

FIG. 3.4 – Exécution d'un programme

La première étape du cycle d'extraction-exécution consiste au chargement de l'instruction `LDA` dont l'adresse est contenue dans le compteur d'instruction (Fig. 3.5).

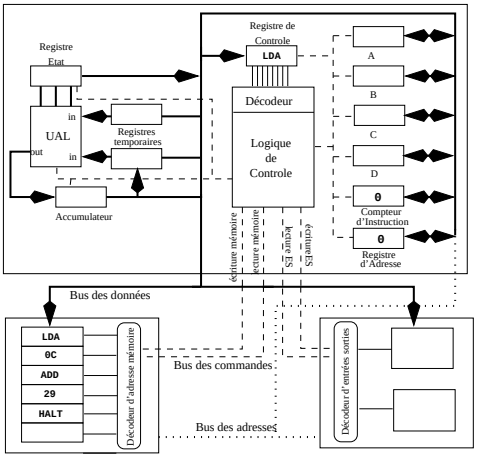

FIG. 3.5 – Chargement de l'instruction `LDA`

La suite du cycle d'extraction-exécution continue avec l'exécution de l'instruction `LDA` (Fig. 3.6) :

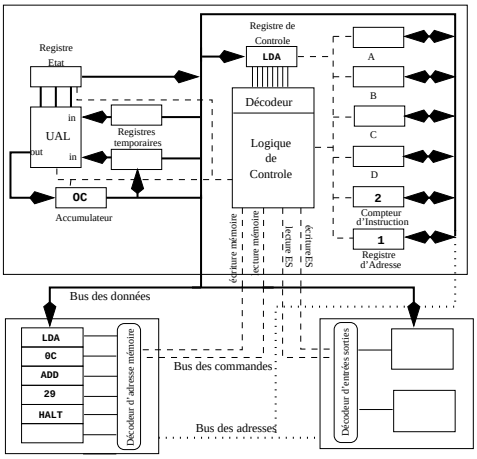

FIG. 3.6 – Chargement de l'instruction `LDA`

1. décodage de l'instruction `LDA` : le compteur d'instruction est incrémenté, l'adresse de la donnée à charger est copiée dans le registre des adresses ;
2. chargement de la donnée dont l'adresse est située dans le registre des adresses : dépôt du contenu du registre des adresses sur le bus des adresses, génération par la logique de contrôle d'un signal de lecture mémoire, transfert de la donnée dans l'accumulateur via le bus des données ;

Le programme se poursuit ensuite avec le chargement de l'instruction `ADD` (Fig. 3.7) dont l'adresse est contenue dans le compteur d'instruction.

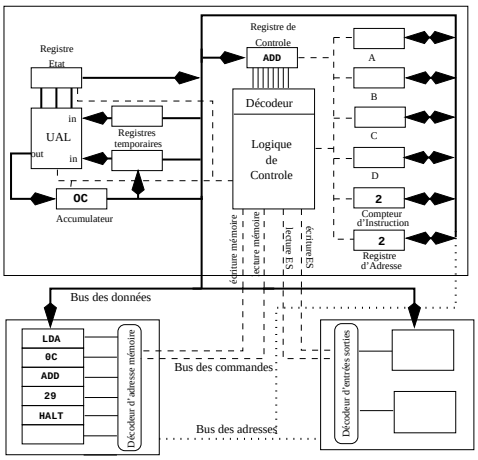

FIG. 3.7 – Chargement de l'instruction `ADD`

On exécute alors cette instruction (Fig. 3.8) :

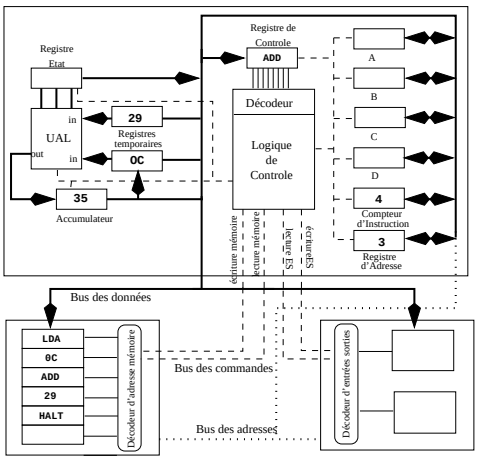

FIG. 3.8 – Exécution de l'instruction `ADD`

1. décodage de l'instruction (le compteur d'instruction est incrémenté, l'adresse de la donnée à charger est copiée dans le registre des adresses),
2. chargement de la donnée dont l'adresse est située dans le registre des adresses (dépôt du contenu du registre des adresses sur le bus des adresses, génération par la logique de contrôle d'un signal de lecture

mémoire, transfert de la donnée dans l'un des registres temporaires, copie du contenu de l'accumulateur dans l'autre registre temporaire),

3. exécution par l'unité arithmétique et logique de l'addition, le résultat est placé dans l'accumulateur

Pour finir, l'unité centrale chargera et exécutera l'instruction `HALT`, ce qui aura pour effet de terminer ce programme.

Chapitre 4

Interruption

4.1 Généralités

La notion d'interruption est fondamentale pour une meilleure compréhension du fonctionnement des ordinateurs modernes. Comme le montre la figure suivante (Fig. 4.1) :

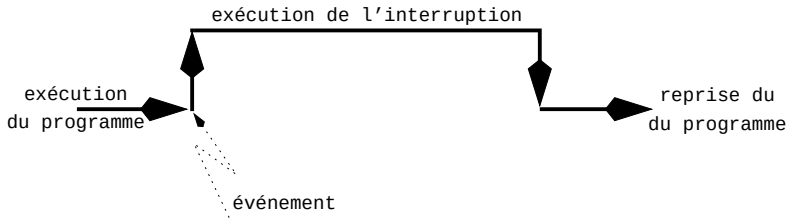

FIG. 4.1 – Notion d'interruption

une interruption est un "signal" associé à un événement. Ce signal provoque une rupture de séquence dans l'exécution d'un programme, et conduit à l'exécution d'une routine de traitement adaptée. Une fois cette routine exécutée, le programme interrompu reprendra son déroulement comme si rien ne s'était passé.

Un exemple simple nous aidera à mieux comprendre l'importance de cette notion. Imaginez un processeur en charge de la gestion du clavier, et ne disposant pas du mécanisme d'interruption. Pour détecter une frappe au clavier, le processeur exécutera continuellement un petit programme qui scrute le circuit électronique associé au clavier. L'utilisateur étant parti boire un chocolat le processeur reste indéfiniment occupé pour rien. Imaginer maintenant le même processeur doté du mécanisme d'interruption. Il excécute tranquillement un programme de calcul, lorsqu'une touche du clavier et en-

foncée. Averti de cet événement par une interruption, il satisfait votre demande, avant de reprendre le travail délaissé. Ce mode de fonctionnement est quand même plus efficace, n'est-il pas ?

4.2 Types d'interruptions

Plusieurs types d'interruptions existent, et elles se répartissent schématiquement en deux familles : les interruptions internes et les interruptions externes.

4.2.1 Interruptions internes

Les interruptions internes sont essentiellement liées à des événements relatifs à l'exécution d'un programme. Ainsi, des erreurs de calcul entraînant un dépassement de capacité, un adressage incorrect de la mémoire au cours d'un débordement de tableau, des demandes d'accès à des fonctions du système, etc., engendrent des interruptions les caractérisant. Il est à noter que ces dernières ne sont pas toujours correctement traitées par le matériel et/ou le logiciel. Il existe une autre interruption interne essentielle au bon fonctionnement d'un ordinateur : l'interruption due à l'horloge. A intervalles de temps réguliers (la durée entre deux intervalles étant programmable), l'horloge génère une interruption afin de "donner la main" au système d'exploitation. Celui-ci choisira alors la prochaine tâche qui sera exécutée, ou assurera une gestion du temps.

4.2.2 Interruptions externes

Les interruptions externes sont toujours liées au fonctionnement des différents périphériques contrôlés par le microprocesseur. Une interruption externe signale un changement d'état du périphérique. Par exemple, le clavier avertira qu'il est prêt à fournir une information, le processeur graphique signalera qu'il a terminé l'affichage demandé, le joystick se réjouira de voir la souris rumuer sa queue [1].

4.3 Traitement d'une interruption

Le traitement d'une interruption est réalisé en partie par les circuits logiques (le matériel), et en partie par la routine d'interruption elle-même. Ainsi, lorsqu'une demande d'interruption survient, les circuits logiques reconnaissent et mémorisent l'interruption. Le programme en cours d'exécution devant-être interrompu, les circuits logiques sauvegardent alors le conte-

[1] Sauf coup du hasard, la lecture de cette annerie prouve au moins que vous avez lu le polycop jusqu'ici. Alors encore un peu de patience et vous pourrez vous interrompre

nu du compteur ordinal et du registre d'état (n'oubliez pas qu'une interruption doit être transparente pour le programme interrompu). Une fois cette sauvegarde faîte, les circuits logiques mettent en place la nouvelle valeur du compteur ordinal afin qu'ils pointe vers la routine d'interruption qui peut alors être exécutée par le processeur. La routine d'interruption commence par sauvegarder l'ensemble des registres (toujours la transparence), traite le problème pour lequel elle a été invoquée, et restaure l'ensemble des registres pour que le programme interrompu reprenne son travail (cette dernière phase se faisant souvent par une instruction particulière).

4.4 Gestion des interruptions

Lors du traitement d'une interruption, d'autres interruptions peuvent survenir, et le processeur doit disposer de moyens pour assurer une gestion de ces interruptions. Les deux mécanismes les plus courants sont le masquage et la hiérarchisation des interruptions.

4.4.1 Masquage et hiérarchisation des interruptions

Le masquage consiste à empêcher que de nouvelles interruptions soient prises en compte pendant le traitement d'une autre. Cette inhibition peut se faire soit par une instruction spécifique, soit par une règle générale spécifiant qu'en mode système un programme ne peut être interrompu. Dans tous les cas, cette inhibition doit être la plus courte possible car certains périphériques exigent des réponses très rapides. La hiérarchisation des interruptions consiste à définir des niveaux de priorités entre les différents périphériques. Ainsi, un périphérique prioritaire pourra interrompre le traitement d'une interruption issue d'un périphérique moins prioritaire, l'inverse n'étant pas vrai.

4.4.2 Identification de la source

La possibilité de devoir traiter plusieurs interruptions simultanément, induit la nécessité de découvrir la provenance de l'interruption. Deux techniques sont possibles : la scrutation et l'identification directe. L'identification de la source par scrutation consiste à consulter les unes après les autres toutes les sources possibles, pour savoir laquelle a généré l'interruption. Cette technique est très simple à mettre en oeuvre puisque toutes les lignes d'interruptions issues des différents périphériques sont réunies par un `OU` logique sur une bascule flip–flop(Fig. 4.2). Cette technique permet également de de définir une certaine priorité entre les interruptions puisque l'on peut choisit l'ordre dans lequel les périphériques seront interrogés.

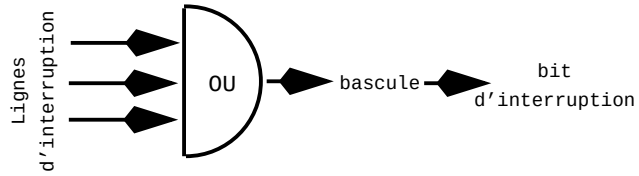

FIG. 4.2 – Identification de la source

Cette nécessité de scrutation peut-être supprimée si le processeur a le moyen de connaître la provenance de l'interruption. L'idéal est de disposer d'un vecteur de bits sur lequel chaque bit correspond à une cause possible d'interruption. Cette technique étant très coûteuse, elle est donc réservée aux ordinateurs hauts de gammes. Une technique, plus flexible pour le programmeur consiste à demander au périphérique de fournir sur le bus des données une partie de l'adresse de la routine d'interruption (Fig. 4.3) ; cette partie est appelée le vecteur d'interruption, et l'on parle d'interruptions vectorisées. L'adresse exacte de la routine sera alors obtenue par une combinaison du vecteur d'état et du vecteur d'interruption.

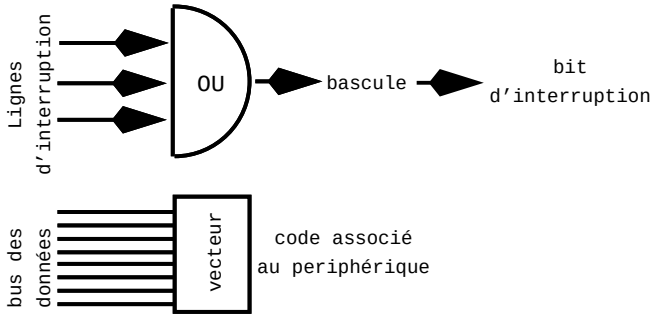

FIG. 4.3 – Vecteur d'interruption

Toutefois, en cas d'interruptions simultanées et à moins de disposer d'un matériel adéquat, la technique précédente peut conduire à ce que deux périphériques écrivent leur vecteurs d'interruption simultanément.

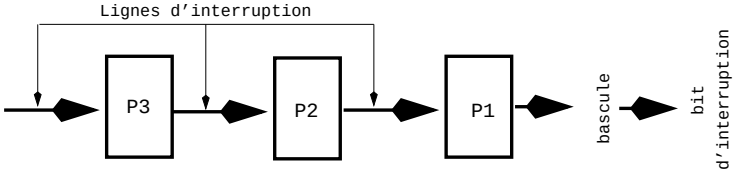

FIG. 4.4 – Daisy Chain

Une méthode simple pour éviter ces problèmes consiste à chaîner (daisy chain) les périphériques entre-eux (Fig. 4.4), sur une seule ligne d'interruption et par ordre de priorité. Ainsi, le périphérique `P3` ne peut faire passer d'interruption que si `P1` et `P2` n'en demandent à cet instant. De plus, lors du traitement d'une interruption issue de `P3`, si le processeur est interrompu par `P2` alors il traite complètement cette nouvelle demande.

Chapitre 5

Autres architectures

5.1 Architecture de Harvard

L'une des causes possibles de la lenteur d'un ordinateur est le faible débit de circulation des informations entre le microprocesseur et la mémoire. Pour augmenter ce flux d'informations, une séparation entre la mémoire de données et la mémoire d'instructions a été introduite dans l'architecture de Harvard (Fig. 5.1). Ainsi, la mémoire dédiée aux données et la mémoire dédiée aux instructions étant accessibles par un bus différent, il est alors possible d'accéder en même temps aux instructions et aux données, ce qui permet de multiplier par deux la vitesse globale de la machine en multipliant simplement par deux la vitesse du processeur.

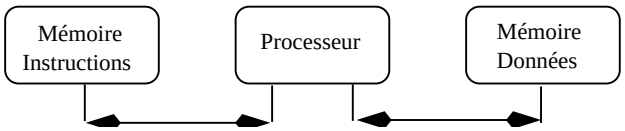

FIG. 5.1 – Architecture de Harvard

D'une complexité technologique importante, ce type d'architecture est très souvent utilisé dans les machines spécialisées dans le traitement des signaux numériques. Toutefois, ce concept de mémoire de données et de mémoire d'instructions a engendré la solution des mémoires caches, qui est employée sur toutes les machines actuelles. Le besoin d'un dispositif de mémoire cache apparaît lorsque l'on dispose de deux niveaux de mémoire :

- un niveau "éloigné", constitué par une mémoire importante, bon marché, et lente ;
- un niveau "proche", constitué par une mémoire réduite, chère, et rapide.

Le principe de la mémoire cache est de réserver une partie de la mémoire

rapprochée pour y conserver des parties de la mémoire éloignée. Deux cas de figures sont possibles :

- le niveau éloigné est la mémoire de masse (disque dur) et le niveau rapproché est la mémoire centrale. On réserve une partie de la mémoire centrale pour y conserver les derniers morceaux des fichiers utilisés. Ainsi, lors de la prochaine consultation d'un de ces morceaux, l'information sera directement trouvée en mémoire, évitant ainsi un accès au disque. Un tel choix se justifie par le fait que ce sont les mêmes fichiers ou parties de fichiers qui sont le plus souvent consultés.
- le niveau éloigné est la mémoire centrale et le niveau rapproché est la mémoire interne à l'unité centrale. Les unités centrales modernes disposant de mémoires ultra-rapides, lors d'un accès à la mémoire centrale, on transfère les 256, 512, ou 1024 cellules mémoire voisines. Ainsi, lors d'un accès ultérieur à l'une de ces cellules, celle-ci sera directement trouvée dans l'unité centrale, évitant ainsi d'avoir à activer le bus de données et la mémoire centrale. La justification de ce choix est qu'il est fréquent d'accéder à la cellule voisine de la cellule à laquelle on vient d'accéder (exécution d'un programme, parcours d'un tableau, etc.)

5.2 Architecture RISC

Une autre approche possible pour augmenter les performances d'une machine est celle adoptée par les machines RISC (Reduce Instruction Set Computer). Comme nous l'avons dit précédemment, les instructions exécutées par l'unité centrale sont en réalité interprétées par un microprogramme contenu dans la logique de contrôle. Jusqu'il y a peu de temps, le jeu d'instruction compris par les microprocesseur était très important (plusieurs centaines d'instructions). Suite à des études statistiques, on s'est aperçu que sur cet ensemble d'instructions, seulement 10% ou moins étaient utilisées par les compilateurs qui produisent 90% du code. Par conséquent, la logique de contrôle utilisait des microprogrammes très compliqués pour interpréter de nombreuses instructions jamais employées. Le concept de l'architecture RISC découle de cette remarque et s'exprime ainsi : la réduction du nombre des instructions disponibles entraîne une simplification du microprogramme et donc améliore considérablement les performances.

5.3 Architecture parallèle

Une dernière approche possible pour augmenter la vitesse d'un ordinateur consiste à développer des machines dont l'architecture est dite "parallèle". Dans ce domaine, de nombreuses solutions ont été trouvées :

– architecture pipeline (Fig. 5.2) : dans les processeurs classiques, une instruction est réalisée en plusieurs phases, d'où l'idée de mettre en série ce travail : dès qu'une phase d'une instruction est terminée, on la commence sur l'instruction suivante. Chaque étape de l'exécution d'une instruction est traitée par une unité spécialisée. Une machine pipeline est comme une chaîne de montage automobile "à un endroit de la chaîne, on est spécialisé dans l'exécution d'une seule tâche". Ce type d'organisation pose des problèmes lors de rupture de séquence dans le programme.

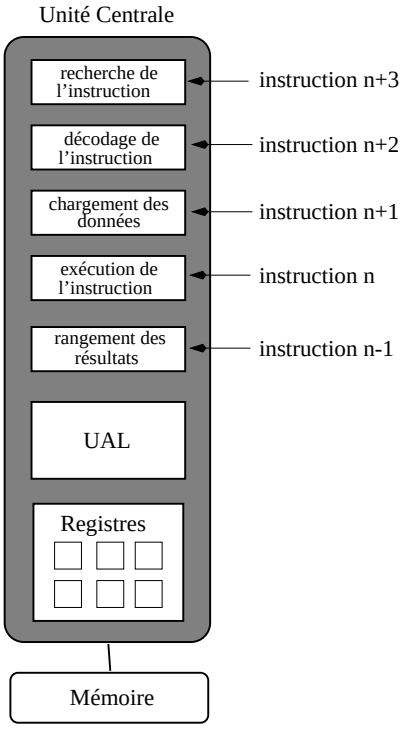

FIG. 5.2 – Machine Pipeline

– processeur vectoriel (Fig. 5.3) ou machine SIMD (Single Instruction Multiple Datas) ; sous le contrôle d'une unique unité de commande, un seul calcul est effectué en même temps sur des données différentes et sur plusieurs Unité Arithmétique et Logique. Ce type de machine est efficace quand le nombre de données est très important et que les instructions intrinsecquement parallèles (boucle dans les tableaux) ;
– architecture multiprocesseurs (Fig. 5.4)ou machine MIMD (Multiple Instructions Multiple Datas) ; plusieurs unités centrales autonomes partagent une même mémoire (ou plus généralement un même ensemble de ressources matérielles). Chaque unité centrale exécute son

propre programme indépendamment des autres. Les problèmes de ces architectures sont essentiellement dus à l'accès concurrent à une même ressource ;

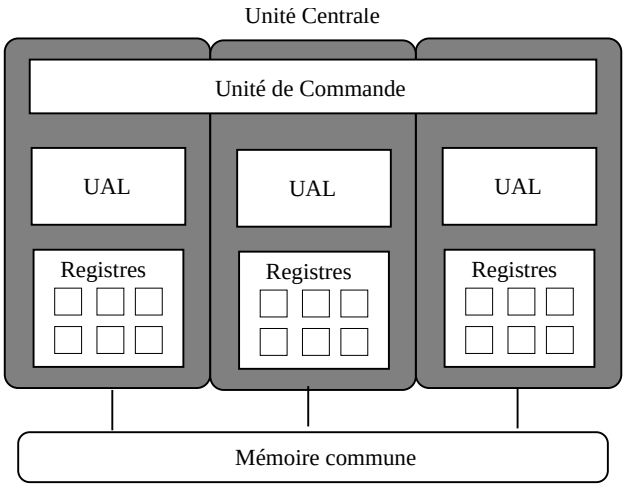

FIG. 5.3 – Machine Vectorielle

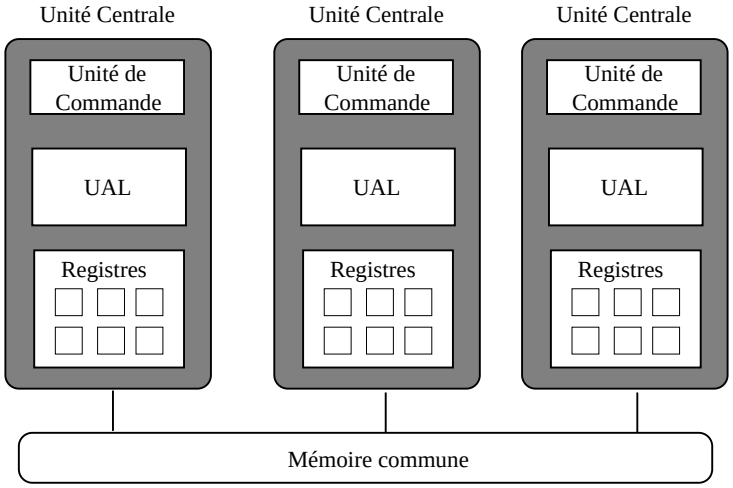

FIG. 5.4 – Machine multiprocesseurs